Privacy-preserving Genomic Disease Susceptibility Testing using Secure Multiparty Computation

A special problem in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science

Submitted by:

Joseph Niel I. Tuazon

May 2016

Permission is given for the following people to have access to this SP:

<table>
<thead>
<tr>
<th>Available to the general public</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Available only after consultation with author/SP adviser</td>
<td>No</td>
</tr>
<tr>
<td>Available only to those bound by confidentiality agreement</td>
<td>No</td>
</tr>
</tbody>
</table>
ACCEPTANCE SHEET

The Special Problem entitled “Privacy-preserving Genomic Disease Susceptibility Testing using Secure Multiparty Computation” prepared and submitted by Joseph Niel I. Tuazon in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science has been examined and is recommended for acceptance.

Richard Bryann L. Chua, M.Sc.
Adviser

EXAMINERS:

<table>
<thead>
<tr>
<th></th>
<th>Approved</th>
<th>Disapproved</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Gregorio B. Baes, Ph.D. <em>(candidate)</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. Avegail D. Carpio, M.Sc.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. Perlita E. Gasmen, M.Sc. <em>(candidate)</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. Marvin John C. Ignacio, M.Sc. <em>(cand.)</em></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Accepted and approved as partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science.

Ma. Sheila A. Magboo, M.Sc.
Unit Head
Mathematical and Computing Sciences Unit
Department of Physical Sciences and Mathematics

Marcelina B. Lirazan, Ph.D.
Chair
Department of Physical Sciences and Mathematics

Leonardo R. Estacio Jr., Ph.D.
Dean
College of Arts and Sciences
Abstract

The paradigm shift invoked by next-generation sequencing (NGS) technologies gives rise to the applications of *in silico* human genomic data in the clinical setting – one application being genomic disease susceptibility testing (DST). DST utilizing sensitive genomic data implicates the need for keeping the test private and secure: where both medical unit of a patient and research group with the disease markers want their genomic data private from each other and to any other adversary that could stigmatize and/or discriminate the medical unit’s patients and that could compromise the profitable disease markers of the research group. To contribute to this need for privacy in DST’s genomic data, this project aims to deliver a privacy-preserving genomic disease susceptibility testing over a secure multiparty computation environment using Sharemind.

*Keywords:* genomic data privacy, privacy-preserving disease susceptibility testing, secure multiparty computation, Sharemind
## Contents

### Acceptance Sheet

#### Abstract

#### List of Figures

#### List of Tables

#### I. Introduction

- A. Background of the Study
- B. Statement of the Problem
- C. Objectives of the Study
- D. Significance of the Project
- E. Scope and Limitations
- F. Assumptions

#### II. Review of Related Literature

#### III. Theoretical Framework

- A. Disease Susceptibility Testing
- B. Variant Call Format
- C. Secure Multiparty Computation
  - 1. Definition
  - 2. Classification
  - 3. Peer Model
- D. Sharemind
  - 1. Sharemind Framework
  - 2. SecreC and Controller Library

#### IV. Design and Implementation

- A. Use Cases
| 3. | PHP source codes | 94 |
| 4. | Other codes | 122 |

XI. Acknowledgement | 127
List of Figures

1. Sample VCF file [55] .................................................. 18
2. Generic vs. Representative-based SMC .......................... 20
3. Sharemind’s representative-based architecture with three private peers (miners) [60] ................................................. 22
4. Node.js servers connecting the client and Sharemind miners .. 24
5. Use Case – System Access ............................................. 26
6. Use Case – System Computation ..................................... 26
7. Use Case – Manage Users ............................................ 27
8. Use Case – Manage Diseases ......................................... 27
9. Use Case – Manage Disease Markers ............................. 27
10. Use Case – Start Computation ....................................... 28
11. Use Case – Sharemind Processes .................................. 28
12. Entity Relations Diagram for research group database .......... 32
13. Genomic DST System Stack ......................................... 33
14. Sharemind Framework Stack ....................................... 34
15. Genomic DST system – homepage .................................. 42
16. Login page – invalid email alert .................................... 42
17. Login page – invalid password alert ............................... 43
18. Login page – for-approval alert ..................................... 43
19. Genomic DST system – about page ............................... 44
20. Genomic DST system – sign-up page .............................. 45
21. Sign-up page – invalid email alert .................................. 45
22. Sign-up page – mismatched passwords alert ...................... 46
23. Admin pages – homepage ........................................... 46
25. Disease catalog page – add disease name already listed ....... 48
26. Disease catalog page – invalid disease name alert .............. 48
27. Disease catalog page – disease name successfully added ....... 48
28 Disease catalog page – edit disease name

29 Disease catalog page – disease name edited alert

30 Disease catalog page – add disease marker modal

31 Disease catalog page – add disease marker with invalid Risk SNP

32 Disease catalog page – invalid input alert

33 Disease catalog page – disease marker added alert

34 Disease catalog page – batch upload disease marker

35 Disease catalog page – batch upload successful alert

36 Disease catalog page – edit disease marker

37 Disease catalog page – disease marker edited alert

38 Disease catalog page – delete disease marker

39 Disease catalog page – disease marker deleted alert

40 Disease catalog page – delete disease

41 Disease catalog page – disease deletion from Sharemind

42 Disease catalog page – disease deleted alert

43 Genomic DST system – user catalog page

44 User catalog page – edit user information

45 User catalog page – confirm edit alert

46 User catalog page – user information edited alert

47 User catalog page – invalid user email alert

48 User catalog page – delete user

49 User catalog page – user deleted alert

50 Genomic DST system – user homepage

51 Computation pages – select disease

52 Computation pages – disease selected

53 Computation pages – upload page

54 Computation pages – upload VCF file

55 Computation pages – uploading to Sharemind miners

56 Computation pages – upload completed
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Variant Calling Software</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>Examples of $z_{ij}$ values</td>
<td>17</td>
</tr>
<tr>
<td>3</td>
<td>Sharemind’s secret-shared table for patient variant</td>
<td>29</td>
</tr>
<tr>
<td>4</td>
<td>Sharemind’s secret-shared table for disease markers</td>
<td>29</td>
</tr>
<tr>
<td>5</td>
<td>Research group’s marker table</td>
<td>30</td>
</tr>
<tr>
<td>6</td>
<td>Research group’s disease table</td>
<td>30</td>
</tr>
<tr>
<td>7</td>
<td>Research group’s user information table</td>
<td>30</td>
</tr>
<tr>
<td>8</td>
<td>Research group’s user log table</td>
<td>31</td>
</tr>
<tr>
<td>9</td>
<td>Research group’s user table status</td>
<td>31</td>
</tr>
<tr>
<td>10</td>
<td>Calculation time per marker count (in seconds)</td>
<td>67</td>
</tr>
</tbody>
</table>
I. Introduction

A. Background of the Study

The Human Genome Project—an international scientific research collaboration aimed to determine the sequence of the entire human genome [1]—served as the key contributor to the rise of advanced technologies that process in silico human genomic data since the project’s completion in 2003 [2, 3]. Having sequenced the entire human genome, sequencing technologies started to evolve: from Sanger sequencing to whole exome sequencing (WES); and ultimately to whole genome sequencing (WGS) [4, 5]. These advanced sequencing technologies that produce high-throughput human genomic data—collectively known as next-generation sequencing (NGS) technologies—fueled the drive among researchers to unravel the underlying information construed in an individual’s DNA. Eventually, these research undertakings facilitated a continuing understanding of the role of DNA at its molecular level in different fields such as preventive medicine and pharmacogenomics [6, 7, 8].

Disease susceptibility testing (DST)—a test performed on a patient to infer probability of incurring certain diseases [9]—is one process affected by the increase in understanding of a DNA’s role in preventive medicine. Initially, DST is performed only by medical units over a patient’s environmental and/or clinical data [10, 11]. The availability of NGS technologies allowed direct-to-consumer (DTC) companies to offer genome-based DST in the market. [12, 13]. However, due to the absence of essential factors such as clinical data, as well as the lack of a medical professional to oversee and interpret the test, DST for these DTC companies have been dismissed [14, 15]. Nevertheless, the utilization of genomic data alongside clinical and/or environmental data in performing DST remains in the clinical setting.

In April 2015, Mayo Clinic—a non-profit medical practice and medical re-
search group [16]—included the use of genomic data alongside clinical data in its DST service to devise a more efficient risk test for breast cancer. [17, 18]. In Mayo Clinic’s intensive research of readily available variants, they ascertained seventy-seven single nucleotide polymorphisms (SNPs) from the \( BRCA1 \), \( BRCA2 \) and other related genes. The combined effects of these ascertained variants increased the degree of risk discrimination for breast cancer prevention and early detection [19].

With Mayo Clinic pioneering the clinical actionability of genomic variants (i.e. SNPs) in DST, the forthcoming prevalence of utility for these variants in the general clinical setting would inevitably approach viability [20, 21, 22]. This premise of general clinical utility for genomic variants, while being able to improve the risk discrimination for DST, raises privacy concerns for the genomic data of the DST patients.

B. Statement of the Problem

NGS technologies that enabled a wide coverage of the entire human genome in WES and WGS brought about privacy concerns in genome sequence processing and storing due to the nature of genomic data which includes (but are not limited to) the following properties: identification of genomic data of an individual [23], revelation of genomic data of family members [24], and association of genomic data of traits and diseases [4, 5, 25].

In DST, when the patient’s genomic data is compromised, the adversary can infer various diseases which the patient might exhibit. Moreover, with genomic data revealing family members, not only can the adversary infer the patient’s diseases but also can it infer those of the patient’s relatives through the concept of disease heritability (i.e. Mendelian inheritance) [9, 26, 27]. This information indicating the presence of a possible disease can be exploited by the adversary in stigmatizing and/or discriminating the related individuals in different fields such as health insurance and employment [28].
With the consequences brought by the compromise of a patient’s genomic data, many patients developed distrust in providing their genomic data even to the medical unit they want to have themselves tested. Patients may opt to only reveal selected regions (loci) of their DNA that are associated with the disease they want to have analysed. This setting imposes that medical units keep the privacy of their patients’ genomic data. On the other hand, the research group with the disease markers cannot give details about the loci of their markers due to the possibility of the medical unit inferring these disease markers – compromising the research group’s profitable undertaking [9, 26, 28].

One possible way to resolve the medical unit and research group’s situation in performing DST is by utilizing secure multiparty computation (SMC). With SMC, the medical unit and the research group can input their genomic data in the computation without revealing anything about their respective inputs – keeping both their genomic data private while being able to produce a relevant output.

C. Objectives of the Study

This project aims to create a web-based privacy-preserving genomic DST using Sharemind with the following functions:

1. The medical unit with the patient’s DNA sequence can:

   (a) sign up to the system.

   (b) log in to the system.

   (c) select disease for testing.

   (d) upload a patient’s variant-called sequence in the system.

   (e) retrieve genomic DST computation result (i.e. regression coefficient).

2. The research group with the necessary disease markers can:

   (a) log in to the system.

   (b) add, edit, or delete diseases.
(c) add, edit, or delete genomic disease markers.

(d) approve or disapprove sign up request.

(e) edit, or delete system users (i.e. medical unit).

3. The medical unit, research group, and a third party can:

(a) secretly store the medical unit patient’s variants and the research group’s disease markers.

(b) securely perform genomic DST amongst themselves.

D. Significance of the Project

Genome-inclusive DST, being an emerging process in the medical field for clinical applications, implicates the importance of securing the computation and keeping private both the inputs of the genomic DST stakeholders: the medical unit with their patient’s genomic data and the research group with their disease markers. With a secure and private process, medical units need not worry about revealing their patients’ genomic data to the research group which can cause serious privacy problems for the patient; and research groups need not worry about revealing their disease markers to the medical units which could compromise their for-profit businesses.

By creating a privacy-preserving DST using SMC, the application of genome-inclusive DST in the clinical setting would be more viable – removing the hindrance of privacy concerns for both the medical unit’s and the research group’s genomic data. With SMC, particularly in Sharemind, the genomic DST computation will be handled by three machines (miners) which facilitates privacy-preservation of the inputs. In this setting, no single miner performing the computation can infer any private input unless this miner colludes with another miner. With only three miners, it is improbable that any two miner can collude given that the miner setup could be one for the medical unit, one for the research group, and one for any trusted institution (e.g. DOH).
In addition, a privacy-preserving genomic DST provides a secure way for the medical units in performing further DST computation using a patient’s clinical and/or environmental data. With an obtained genomic DST result (i.e. regression coefficient), medical units can further compute for the patient’s overall DST by aggregating this obtained result with the result from similar computation using the same patient’s clinical/environmental data [29, 30].

Lastly, it is significant for a medical unit to outsource the DST using the research group’s disease marker because reliable disease markers only come from intensive research of hundreds of thousands of samples in genome-wide association studies (GWAS) [31] in which no average medical unit can produce [32].

E. Scope and Limitations

1. This system will only handle genomic DST and will not include the use of clinical/environmental data.

2. This system will only handle processes from variant annotation to variant filtering (i.e. computation of DST using the patient’s variant-called genomic data and the research group’s disease markers).

3. This system will not handle DNA sampling, DNA sequencing, variant calling, and clinical interpretation in the genome analysis pipeline.

4. This system will use Sharemind version 3 as the SMC framework.

5. This DST computation will only handle SNP variants.

F. Assumptions

1. The medical unit sequence their patient and stores these sequences in its variant-called form in Variant Call Format (VCF).

2. The medical unit maintains a set of security measures in storing their patient’s genomic variants.
3. There are three parties (miners) computing for the genomic DST.

4. The miners will be the medical unit, the research group, and a third party trusted by both the medical unit and the research group.

5. These miners are all online during the computation.

6. These miners are not colluding with one another.
II. Review of Related Literature

The completion of the Human Genome Project in 2003 enabled the advancements in sequencing technologies—generally known as next-generation sequencing (NGS) technologies—producing high-throughput human genomic data in the form of whole exome (WES) and whole genome sequences (WGS) [4, 5]. The availability of WES and WGS fueled a paradigm shift in the clinical setting – now including applications utilizing high-throughput genomic data [25, 33] such as carrier testing, diagnostic testing, and predictive testing.

Carrier testing is a test performed usually between couples (i.e. compatibility testing) to check for possibility of their offspring inheriting an autosomal recessive disease by means of Mendelian inheritance. Moreover, carrier testing is performed over NGS genomic data by identifying the presence of a recessive allele in a heterozygous genotype for a particular disease in both individuals [9, 34].

Diagnostic testing is a type of genetic testing performed before birth or at any time during a person’s life to confirm a previous diagnosis or to serve as a differential diagnostic testing for a symptomatic individual. This test is performed using NGS genomic data by checking for the presence of genomic variants related to a specific disease in a symptomatic individual’s genotype [22, 34]. Predictive testing is a type of genetic testing performed to an individual to detect gene mutations associated with disorders that appear after birth or often later in life. Using NGS genomic data in predictive testing is to check if a causal/associated genomic variant for a particular disease/disorder is present in the genotype of an individual [22, 34].

Disease susceptibility testing (DST), a type of predictive testing, is a promising clinical procedure where medical units check for the probability of a patient incurring a disease [9]. Through the use of the acquired knowledge of disease-gene relationships, it is possible to determine the probability of incurring a disease based solely from an individual’s genome through the use of statistical models (e.g. logistic regression) and metrics (e.g. Odds Ratio) [30, 29, 35].
<table>
<thead>
<tr>
<th>Software</th>
<th>Caller type</th>
<th>Identifies</th>
<th>Free</th>
</tr>
</thead>
<tbody>
<tr>
<td>GATK</td>
<td>germline</td>
<td>SNP, InDel</td>
<td>No</td>
</tr>
<tr>
<td>SAMtools</td>
<td>germline</td>
<td>SNP, InDel</td>
<td>Yes</td>
</tr>
<tr>
<td>Varscan 2</td>
<td>germline</td>
<td>SNP, InDel</td>
<td>Yes</td>
</tr>
<tr>
<td>GATK</td>
<td>somatic</td>
<td>InDel</td>
<td>No</td>
</tr>
<tr>
<td>SAMtools</td>
<td>somatic</td>
<td>SNP, InDel</td>
<td>Yes</td>
</tr>
<tr>
<td>Varscan 2</td>
<td>somatic</td>
<td>SNP, InDel, CNV</td>
<td>Yes</td>
</tr>
<tr>
<td>CNVnator</td>
<td>CNV</td>
<td>CNV</td>
<td>No</td>
</tr>
<tr>
<td>CONTRA</td>
<td>CNV</td>
<td>CNV</td>
<td>Yes</td>
</tr>
<tr>
<td>RDXplorer</td>
<td>CNV</td>
<td>CNV</td>
<td>Yes</td>
</tr>
<tr>
<td>breakdancer</td>
<td>SV</td>
<td>InDel, Inv, CNV</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 1: Variant Calling Software

DST, being an emerging process in the clinical setting, utilizes a number of disparate software from the start of its *in silico* pipeline that could call nucleotide bases, call variants from the base-called sequence, and annotate these called variants [4, 5, 36]. In **base calling**, a number of dedicated software are available such as Bowtie/Bowtie2, BWA, Novoalign, SOAP, and SSAHA2. **Variant calling** also has a number of dedicated software performing different variant calls. Table 1 shows some of the most commonly used software dedicated to variant calling. **Variant annotating**, similar to base- and variant-calling, also has a number of dedicated software performing annotation such as ANNOVAR, AnnTools, and NGS-SNP, among others. Software such as CLCBIO Genomic Workbench, Geno-matrix, and others provide most of these processes as a package, but are only commercially available.

Ideally, all these software should be present in only one stakeholder of the genome-inclusive DST – the medical unit performing DST. Yet, in the perceived general clinical setting, medical units that sequence their patients genomic data do/could not provide annotation of their patients variants due to the lack of disease markers for annotation use. This setting assumes the medical unit either outsource the disease markers from a research group or provide their patient’s
variant-called genomic data to the research group. Given both the privacy concern of the medical unit and the research group, genome-inclusive DST would not be viable without creating protocols that would keep private not only the medical unit patient’s genomic data but also the research group’s disease markers [5, 26, 31]. To contribute to the clinical viability of DST, different security protocols have been implemented in preserving the privacy of either only the patient’s genomic data [37] or both the patient’s genomic data and the research group’s disease markers (under certain assumptions) [26].

Zhao et al. [37] performed genomic variant matching securely over a public cloud using “site-wise” encryption. The idea behind site-wise encryption is that encryption of genomic data is done per variant site (i.e. by using the CHROM and POS as site indicators of rows in a Variant Call Format (VCF) file). The encryption scheme used for encrypting site-based genomic data is Advanced Encryption Standard (AES). This scheme was chosen because of its symmetric property where a ciphertext can be decrypted using the same key used in its encryption. Decryption of ciphertext is important in the whole process of variant matching because through decryption can the user know what genomic variants have matched – as the program that performs the variant matching outputs a list of matched variants. In matching genomic variants, a Java-based program—SWECloud—was installed in a public cloud to perform exact string matching over the ciphertexts generated by AES. Exact string matching was performed on the ciphertexts from the patient’s genomic data and the medical unit’s disease markers. In this setting, the medical unit holds both their patient’s genomic data and the disease markers used for genomic variant matching. With access to both genomic data, the medical unit was able to produce similar ciphertexts from their local machine by using the same encryption key for both their patient’s genomic data and their disease markers. The encryption of both patient’s genomic data and medical unit’s disease markers were done by utilizing the ALT field from their respective VCF files. In addition to using the ALT field, a patient’s genotype was considered by utilizing the GT
attribute from the `FORMAT` field. After locally encrypting both genomic data, the patient’s encrypted genomic data were uploaded and stored in the public cloud. Whenever variant matching is done, the encrypted disease markers are uploaded to perform exact string matching to produce a list of ciphertexts that have matched. This method keeps the genomic data of the patient private while having obtained results of possible variant matches. With the encrypted match results, the user can decrypt the matched ciphertexts to identify matching variants. However, this method assumes that the process of DST is done only in the medical unit – where the medical unit curates their own set of disease markers.

Ayday et al. [26] securely computed DST while keeping private the genomic data of both the patient and the medical center (MC) using homomorphic encryption. In their work, the genomic data of the patient is encrypted and stored in a storing-processing unit (SPU). The SPU and the MC holds a public key provided by the patient—who holds the secret key—for performing homomorphic encryption. In performing DST, the MC requests for particular regions of the patient’s genome from the SPU. In an attempt to keep private the disease marker of the MC, the MC only provided the loci of their disease markers to the patient (who will encrypt these loci to be sent to the SPU). This method only partially secures the disease markers of the MC – given that the patient might infer these disease markers based on the provided loci. Thus, the work assumes that the MC with profitable disease markers provide additional “dummy” SNP loci in order for the patient to not directly infer the disease marker. After the patient encrypt and send the disease marker loci to the SPU, homomorphic operations were performed by the SPU over the encrypted loci using the provided public key to retrieve the SNPs requested by the MC. Having received the requested SNPs in encrypted form, the MC can either: perform homomorphic encryption to the encrypted SNPs for the patient to decrypt the final result of the DST; or send the encrypted SNPs back to the patient for the patient to decrypt and send back to the MC in performing DST computation.
Aside from these approaches, secure multiparty computation (SMC) can be used to keep private the genomic data of both the medical unit patient and the research group [3, 38]. By using SMC, the medical unit of the patient and the research group with the disease markers can perform DST by secret sharing their confidential genomic data to the SMC computation that produces an output for the medical unit. With a private computing environment, the research group’s disease markers would also be considered in the DST computation; and adding “dummy” SNPs that increases the computation input would not be anymore necessary.

Zhang et al. [39] and Kamm et al. [32] implemented SMC over genomic data in a genome-wide association studies (GWAS). The work of Zhang et al. is an entry to the iDASH competition\(^2\) where the objective is to compute for the minor allele frequency and the chi-square statistics of a given aggregated genomic dataset for two groups (i.e. case group and control group) without having to reveal the datasets of both groups. Kamm et al., on the other hand, implemented GWAS in an independent research to compute for statistical values in chi-square tests, Cochran-Armitage test for trend, and transmission disequilibrium test (TDT) using the Sharemind framework. Zhang et al. [39], on the same competition, had also implemented SMC over genomic data in computing for edit distances. Edit distances in genomics are metrics used for scoring the similarity of two sequences when aligned with each other. Edit distances represent the insertions, deletions, and mutations necessary to transform one sequence into the other.

Secure multiparty computation (SMC) has only been a theoretical concept in cryptography since 1982 when it was first introduced by Yao in the millionaire problem. In this problem, two millionaires want to know who is richer between them without having to reveal their respective wealth [40]. Subsequently, Yao introduced multiple protocols which has then become bases for optimizations and extensions of many authors in their theoretical study of the field – focusing only on feasibility results [41].

\(^2\)http://www.humangenomeprivacy.org/2015/competition-tasks.html
As early as 2008, practical applications \cite{42, 43} of the then-theoretical concept of SMC have surfaced. In 2008, Bogetoft et al. \cite{42} reported of the first large-scale practical application of SMC at a bidding between sugar beet farmers and Danisco, the only sugar beets processor in Denmark. By having the quantity of goods farmers are willing to sell and the quantity of goods the company is willing to buy kept secret, the market clearing price—price per unit of the traded commodity—computed by an auctioneer is insusceptible of being manipulated by either groups. In 2010, Bogdanov et al. \cite{43} described the first recorded use of SMC over the Internet for a system collecting and analyzing financial data for a consortium of ITL (Information Technology and Telecommunications) companies. The goal of the system is to provide member companies of summary data from their own input of confidential data so that they, the members themselves, can adapt to changes in the economic sector without the consortium compromising their confidential data.

The rise of the practical applications of SMC paved the way for the development of different framework implementations such as Fairplay \cite{44} and its multiparty variant FairplayMP \cite{45}, VIFF \cite{46}, SEPIA \cite{47}, and Sharemind \cite{43, 48, 49, 50}.

**Fairplay**\(^3\) was released in 2004 and was considered the first implementation of an SMC for only two parties. The framework was designed to run on a Boolean circuit, implementing Yao’s garbled circuit \cite{40}. FairplayMP, or Fairplay Multi-Party, was released in 2006 as a modification to the original Fairplay to allow more than two parties in the computation. The Fairplay framework uses a generic SMC architecture that could be implemented using a high-level programming language called Secure Function Definition Language (SFDL) which can be compiled into a low-level representation for a Boolean circuit. Due to the nature of Boolean circuits and the availability of arithmetic circuits, the original Fairplay framework has no known large-scale practical application.

\(^3\)http://www.cs.huji.ac.il/project/Fairplay/
FairplayMP, however, has been modified since and has been implemented in Java. The only published application of the framework is from a small bidding at a second-price auction where everyone learns the second-highest bid, but only the seller learns the identity of the winner; and where the winner pays the second highest bid, not his winning bid [51].

VIFF (Virtual Ideal Functionality Framework) was released in 2008 and was first used to run the first large-scale SMC project in 2008 to determine the market clearing price for sugar beets in Denmark [42]. The framework also uses a generic SMC architecture. The downside of using VIFF is the minimal support for the framework (latest version was released in 2009) 4.

SEPIA (Security through Private Information Aggregation) was also released in 2008. SEPIA is one of the first frameworks to design a different approach to multiparty computation where players can act solely as an input peer, or as both input and private peer. An input peer is an entity sharing its secret while a private peer is a secure entity computing a shared secret. The private peer can be hosted by an external party or can be hosted among the input peers. Given these settings, SEPIA can be described as either a representative-based SMC or as a generic SMC.

SEPIA was built using Java to provide platform independence. The framework was designed in such a way that developers can include only the necessary protocols in their project and discard unnecessary protocols. Given the modular nature of SEPIA, the framework was also designed to be robust with its ability to cater custom protocols.

In 2010, SEPIA outperformed VIFF and FairplayMP both in multiplication and comparison operations. However, using SEPIA is not as preferable as it seems given the lack of a detailed documentation and the lack of continuous support for the framework (latest version was released March 2012) 5.

4http://viff.dk/
5http://sepia.ee.ethz.ch/
Sharemind⁶, an SMC similar to SEPIA, was also released in 2008. The framework was designed to have three different miners—the private peers computing the shared secret—that could be independent of the input peers. This design lead to the inclusion of exactly three parties independently computing their portion of the shared secret from any number of input peers. The number of miners was limited to three because adding more parties to the computation would lead to a quadratic increase in computational complexity. Designed similarly to SEPIA, Sharemind’s miners can also be the input peers themselves, if the design for the system is not reliant on external trusted parties.

The difference between Sharemind and SEPIA is the handling of technical procedures (e.g. network setup and exact details of message delivery). In SEPIA, only the protocols are available; the implementation for other technical procedures have been left for the developers.

Sharemind was initially implemented using the C++ language and has three available versions – SDK Emulator, Academic Server, and Application Server. The SDK Emulator is the only Open Source software among the three. Later versions of Sharemind are implemented using Sharemind assembly as the low-level language with SecreC as its high-level language representation. SecreC is a privacy-preserving programming language specifically created for Sharemind applications.

Performance-wise, Sharemind version 1 also outperformed FairplayMP in multiplication and comparison operations. The performance of Sharemind version 2, when compared to SEPIA, was similar. The current version for Sharemind is version 3.

⁶https://sharemind.cyber.ee/
III. Theoretical Framework

A. Disease Susceptibility Testing

Disease susceptibility testing (DST)—a test that checks for the probability of a patient incurring a certain disease [9]—has been performed in the clinical setting for a relatively long time – with many researches focusing on cardiovascular diseases (CVDs) [10, 11, 52]. Before, DST was only performed using the clinical and/or environmental data of a patient [10, 11]. Because of the rise of NGS technologies that produce high-throughput human genomic data, these genomic data are now also being utilized in the clinic – where a patient’s genomic data is used in performing DST alongside clinical and/or environmental data to yield a better risk discrimination [19].

Most of the computations for genome-inclusive DST uses an additive model (i.e. regression model) that computes for every factor (e.g. clinical, environmental, and genomic) and performs a summation of the resulting risk coefficient for each factor to come up with a combined risk coefficient that serves as the overall risk coefficient for a disease [30, 29]. A general formula for the overall risk coefficient is denoted by:

\[
\beta_{fi} = \mu_f + \beta_{gi} + \beta_{ci/\epsilon_i}
\]

\(\beta_{fi}\) denotes the regression coefficient for the overall disease risk of individual \(i\); \(\mu_f\) denotes the new intercept for the combined regression coefficients model; \(\beta_{gi}\) denotes the regression coefficient for the genomic risk; and \(\beta_{ci/\epsilon_i}\) denotes the coefficient for the clinical and/or environmental risk. The genomic risk coefficient \(\beta_{gi}\) is computed as follows:

\[
\beta_{gi} = \mu + \sum_{j \in C} z_{ij}u_j
\]

\(\mu\) denotes the regression intercept of the current model, \(C\) denotes the list of
causal SNPs; $u_j$ denotes the effect/weight of the $j$th causal SNP to the overall genetic risk; and $z_{ij}$ denotes the genotypic value of the $i$th individual’s genotype compared to the $j$th causal SNP.

- $u_j$ can be obtained by using the Odds Ratio—a metric obtained from a GWAS study that determines how likely (or unlikely) an SNP contributes to the disease risk [35]—of the $j$th causal SNP in the following equation [30]:

$$u_j = \ln(OR_j)$$

- $z_{ij}$ can be obtained by counting the number of occuring causal SNP per genotype of the same chromosome and position. Given the diploid DNA of individuals, a causal SNP can occur in an individual’s genotype once or twice, or none at all – leading to $z_{ij} \in \{0, 1, 2\}$. $z_{ij} = 0$ represents no occuring $j$th causal SNP for the $i$th individual’s genotype, $z_{ij} = 1$ represents a single occuring $j$th causal SNP, and $z_{ij} = 2$ represents a $j$th causal SNP occuring twice [53].

In VCF files, a genotype and a disease marker obtained from the same reference genome assembly having the same CHROM and POS field values can be compared for causal SNP occurrence. Let $x_1x_2$ be the genotype obtained from GT in a specific genomic location of an $i$th individual’s VCF file – where $x_i \in \{A, C, T, G\} \forall i \in \{1, 2\}$; and $y$ be the $j$th causal SNP from the ALT field of the same genomic location as $x$ from the disease marker VCF – where $y \in \{A, C, T, G\}$. $z_{ij} = 0$ if $(x_1 \neq y) \land (x_2 \neq y)$; $z_{ij} = 1$ if $(x_1 \oplus x_2) = y$; and $z_{ij} = 2$ if $(x_1 = y) \land (x_2 = y)$.

For example, a disease marker causal SNP $j$ with CHROM = 20, POS = 14370, and ALT = $C$; and three possible values for an $i$th individual’s genotype (e.g. GG, CT, and CC) with the same genomic location will yield the following values for $z_{ij}$:
<table>
<thead>
<tr>
<th>Location</th>
<th>jth ALT value</th>
<th>ith GT value</th>
<th>( z_{ij} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>20:14370C</td>
<td>GG</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>20:14370C</td>
<td>CT</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20:14370C</td>
<td>CC</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Examples of \( z_{ij} \) values

The first row of Table 2 indicates \( z_{ij} = 0 \) because the ALT nucleotide \( \text{C} \) of the \( j \)th causal SNP is not present in the genotype \( \text{GG} \) of the \( i \)th individual; the second row indicates \( z_{ij} = 1 \) because a single ALT nucleotide \( \text{C} \) is present in the genotype \( \text{CT} \); and the last row indicates \( z_{ij} = 2 \) because the nucleotide \( \text{C} \) is present twice in the genotype \( \text{CC} \).

After obtaining the overall regression coefficient \( \beta_{fi} \), computation of the risk probability \( P \) can be done as follows:

\[
P = \frac{e^{\beta_{fi}}}{1 + e^{\beta_{fi}}}
\]

Computation for the clinical/environmental coefficient factor can be done similarly as the genomic coefficient – where both the causal SNP and SNP weight represent the clinical/environmental factor and the factor’s weight respectively.

**B. Variant Call Format**

The Variant Call Format (VCF) was developed for the 1000 Genome Project to standardize the formatting of called variants present in their database [54]. As of today, this format has also been the standard format used for variant calls for many applications [36].

A VCF file contains three (3) sections, namely: header, column names, and body. The header section contains metadata starting with `##` which are succeeded by a `key=value` pair. These metadata are all optional but is recommended to describe the file and its contents.
Figure 1: Sample VCF file [55]

The column names section is a single row containing the identifiers for the contents of the body. The column name row starts with a # and is succeeded by the following columns in the same order [54, 55]:

- **CHROM** identifies the chromosome of a particular variant.
- **POS** identifies the position in the chromosome of a variant based on a specific reference genome.
- **ID** uniquely identifies a variant on a particular database (e.g. dbSNP). Missing value for this column and for the other columns listed is defined by a single dot (.)
- **REF** identifies the original nucleotide sequence for the specified POS and CHROM derived from a particular reference genome.
- **ALT** is the variant identified. Change in the REF nucleotide indicates an SNP. Additional letters on the ALT indicate insertions, and missing letters on the ALT indicate deletions. This field can have multiple values separated by a comma and can only be empty whenever the current variant is a deletion.
- **QUAL** is a *Phred quality score* based metric that determines how accurate the base calling (DNA sequencing) is. A score of < 10 determines an accuracy rate for base calling of < 90%.
• **FILTER** determines which of the following filters in the header metadata `FILTER=<some value>` the variant did not pass.

• **INFO** contains additional information about a specific variant that can be traced in the metadata `INFO=<some value>`.

• **FORMAT** contains the genotype and other information for the samples succeeding this column. The contents of this format can be specified in the metadata `FORMAT=<some value>`.

• `<+SAMPLES>` contain the values for a particular sample based on the **FORMAT** field of the particular row.

A human genotype can be represented in the VCF file using the `<+SAMPLES>` columns after the **FORMAT** column – where a single column represents a single sample (i.e. individual human sample). By utilizing the genotype (GT) attribute of the **FORMAT** field, a diploid human (having two copies of every chromosome) can be represented a GT value of \( n_1|n_2 \) where \( n_i \geq 0 \forall i \in \{1, 2\} \). \( n_i = 0 \) represents the nucleotide in the **REF** field, and \( n_i \geq 1 \) represents the \( n_i \)th nucleotide listed in the **ALT** field.

For instance, Figure 3 contains two samples: **NA00001** and **NA00002**. The genotype in the first row with **REF** = **G**, and **ALT** = **A** for sample **NA00001** is **GG** because the GT value for that particular variant row in **NA00001** is 0|0; while for **NA00002**, **AG** is the genotype because the GT value in **NA00002** is 1|0. For the third variant row entry with **REF** = **A**, and **ALT** = **G,T**, the genotype of sample **NA00001** is **GT** because 1|2 represents the 1st and the 2nd nucleotides in **ALT** which are **G** and **T** respectively; while the genotype of sample **NA00002**, given a GT value of 2|1, is **TG**.

Aside from using a bar (|) to differentiate nucleotides in a genotype, a slash (/) can also be used. A bar represents a phased genotype and a slash otherwise. A phased genotype is a genotype with identifiable chromosome copy location.
C. Secure Multiparty Computation

1. Definition

Secure Multiparty Computation (SMC) is the process of securely computing for a function \( f(x_1, x_2, \ldots, x_m) \) by \( P_i \) entities \( (i \in 1, 2, \ldots, m) \), where \( x_i \) is known to only \( P_i \), to obtain an output \( y_i \) [40]. In simple terms, an SMC protocol is a collaborative computation among different entities using a secret input to obtain a necessary output.

2. Classification

SMC protocols can be classified based on the protocol’s architecture: generic SMC and representative-based SMC. The generic SMC architecture requires all input parties be online [44, 45, 46, 56] – having the input peer and the private peer identical. The representative-based SMC, on the other hand, does not require all input parties to be online during the computation. This is accomplished by assigning external parties as the private peers [47, 48].

Figure 2: Generic vs. Representative-based SMC

SMC protocols can also be classified according to the computation medium – a boolean circuit or an arithmetic circuit. In a boolean circuit, Yao’s garbled circuit [40] is the scheme implemented by many boolean circuit-based multiparty protocols [44, 57]. In an arithmetic circuit, Shamir’s Secret Sharing scheme [58] is the reference scheme used by many multiparty protocols [46, 47, 48].
3. Peer Model

The types of peers in an SMC are as follows: honest peers, semi-honest adversaries, and malicious adversaries [46].

1. **Honest peers** are peers that follow strictly the given protocol and does not collude with other peers.

2. **Semi-honest adversaries** (honest-but-curious, passive) are peers that follow the given protocol but tries to determine a peer’s secret by colluding with other peers.

3. **Malicious adversaries** (active) are peers not following the given protocol with the purpose of aborting the protocol, producing incorrect outputs, or obtaining other peers’ secrets.

In an SMC, secure computation can be accomplished whenever the majority of the participating peers are honest [59].

D. Sharemind

1. Sharemind Framework

Sharemind is an SMC framework that utilizes the representative-based architecture approach in implementing its multiparty computation. Sharemind has three private peers called miners. The number of miners is limited to three as a compromise for both efficiency and security of the framework – as adding new miners increases the complexity of the computation quadratically [48].

The Sharemind framework utilizes the non-standard *additive sharing scheme* over the ring $\mathbb{Z}_{2^{32}}$ because Shamir’s secret sharing scheme cannot be applied over the ring $\mathbb{Z}_{2^{32}}$ [48]. Yet, improvements for Sharemind which includes a protocol suite that utilizes the standard Shamir’s secret sharing scheme—which provides more flexibility in the number of participants and access of results—have only been proposed and has not been implemented into production [61].
The only developed protocol suite of Sharemind, \textit{shared3p}, uses the additive sharing scheme with passive security (i.e. the protocol is safe against a semi-honest peer). Other unreleased versions of Sharemind include: \textit{shared2p} – a two-party version of \textit{shared3p}; \textit{sharednp} – a multiparty version of \textit{shared3p}; and \textit{shared2a} – a two-party protocol suite with active security (i.e. the protocol is secure against a malicious adversary) [62].

Sharemind was created with the purpose of bringing secure multiparty computation to a playing field where developers can easily develop privacy-preserving applications [48]. In Sharemind, primitive operations used for multiparty computation protocol implementations are available and accessible with the use of a high-level programming language called SecreC.

2. SecreC and Controller Library

Computation over Sharemind is programmable using a high-level language called SecreC [50]. SecreC is a privacy-preserving programming language built to create a high-level program interface specifically for Sharemind. This programming language is inspired by the C language - having its structure similar to that of C [49]. Below is an example code written in the original SecreC [63]:
void main ()
{
    private uint a, b, c; // private data
    a = b + c; // private computation
    public uint d; // public data
    d = declassify (a); // private -> public
    publish (d); // send to client
}

Sharemind version 3 utilizes SecreC 2—a rewritten and improved version of the original SecreC. Improvements on SecreC 2 includes: protection domain polymorphism, modularity, and more primitive types, among others [63]. An example code written in SecreC 2:

    import stdlib; // import stdlib
    import additive3pp; // import PDK
    domain pd_a3p additive3pp; // declare PD

    void main ()
    {
        pd_a3p uint a, b, c; // private data
        a = b + c; // private computation
        public uint d; // public data
        d = declassify (a); // private -> public
        publish (d); // send to client
    }

In developing web-based applications, a Javascript controller library has been designed for client-side communications with the Sharemind miners. With this client-side Javascript controller, browsers are able secret-share the client’s input directly to the three Sharemind miners instead of passing on the input to an intermediary server that partitions the shares. This controller also enables the browser to retrieve the resulting multiparty computation output from the three miners.

For the browser to communicate with the Sharemind miners, a Node.js server—a Javascript-based server—is utilized alongside each Sharemind miner. The use of Node.js servers alongside Sharemind miners contributed to the workaround implemented by the Javascript controller library that bypasses the same-origin policy implemented in many modern browsers. Same-origin policy is a policy
restricting a browser to send requests to other servers different from the source server. Through the inclusion of Socket.io—a library for Node.js applications—in the Javascript controller library of the browser, the controller library was able to implement Cross Domain Resource Sharing (CORS) – a workaround for bypassing the same-origin policy. CORS works by modifying HTTP headers in your requests to access resources on a different domain [60, 64, 65].

An example Javascript code implemented over JQuery, a Javascript framework, connecting to the Node.js servers with ports 8081, 8082, and 8083 using the Javascript controller with namespace \texttt{sm}: \[
\text{\texttt{(function(app, sm, $){}}}
\]

\[
\begin{array}{c}
\text{var hosts = [}
\text{"http://192.167.19.1:8081",}
\text{"http://175.33.11.129:8082",}
\text{"http://168.53.1.32:8083"}
\text{]};
\text{var servers = null;}
\text{function serversConnect(callback) {}
\text{if(servers == null) {}
\text{app.debug("Sharemind not yet connected.");}
\text{try {}
\text{app.log("Connecting to Sharemind.");}
\text{servers = sm.ctrl.connect( // controller connect fnctn}
\text{hosts, // array of Node.js server addresses}
\text{function() {}
\text{ // success connection callback}
\text{}}}}
\text{);}}
\end{array}
\]

Figure 4: Node.js servers connecting the client and Sharemind miners
function(error) {
    // error callback
}

.catch(err) {
    app.debug("Failed to connect to Sharemind: " + err);

    serversDisconnect(); // controller disconnect function
    return;
}

})(this.app = this.app === undefined ? {} : this.app, sm, jQuery);
IV. Design and Implementation

A. Use Cases

The medical unit can sign up to the system for the system administrator to approve/disapprove the user profile. The medical unit, after having been approved, can log-in to the system to start the disease risk computation. The system administrator (i.e. research group) can also log in to the system to perform certain administrative functions such as managing the users, disease catalog, and disease marker catalog.

Figure 5: Use Case – System Access

Figure 6: Use Case – System Computation
Under manage users, the system administrator can approve/disapprove the sign-up request of a user, edit system user information, and delete system users.

For manage diseases, the system administrator can add disease, edit disease name, and delete disease.

Last for system administrator functions, the system administrator can add disease marker per disease in the system, edit disease marker per disease, and delete disease markers.
For the medical unit, the functions available in starting the computation are as follows: selecting a disease of interest, uploading a patient’s genomic variants in a VCF file, calculating the risk coefficient, and generating a PDF report of the result.

The system processes done in the Sharemind miners (represented by three entities; e.g. medical unit, research group, and third party) are the upload of the medical unit of patient variants, the upload of medical unit of disease markers, and the retrieval of Sharemind computation result. The Sharemind servers can be hosted at any entity agreed upon by the research group and the medical unit/s using the system.
B. Database Design

1. Data Dictionary

The Genomic DST system will have two separate databases: one shared by the Sharemind miners, and one for the research group. The database of the Sharemind miners will contain two tables, particularly: a table for the medical unit patient variant and a table for the research group’s disease markers.

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>chromosome</td>
<td>uint8</td>
<td>CHROM value in VCF file</td>
</tr>
<tr>
<td>position</td>
<td>uint32</td>
<td>POS value of current genotype row in VCF file</td>
</tr>
<tr>
<td>nucleotide_a</td>
<td>uint8</td>
<td>Count of nucleotide A present in the current genotype</td>
</tr>
<tr>
<td>nucleotide_c</td>
<td>uint8</td>
<td>Count of nucleotide C present in the current genotype</td>
</tr>
<tr>
<td>nucleotide_t</td>
<td>uint8</td>
<td>Count of nucleotide T present in the current genotype</td>
</tr>
<tr>
<td>nucleotide_g</td>
<td>uint8</td>
<td>Count of nucleotide G present in the current genotype</td>
</tr>
</tbody>
</table>

Table 3: Sharemind’s secret-shared table for patient variant

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>chromosome</td>
<td>uint8</td>
<td>Autosomal chromosome location of the marker.</td>
</tr>
<tr>
<td>position</td>
<td>uint32</td>
<td>chromosome position of current marker</td>
</tr>
<tr>
<td>nucleotide_a</td>
<td>uint8</td>
<td>Count of nucleotide A present in the current marker</td>
</tr>
<tr>
<td>nucleotide_c</td>
<td>uint8</td>
<td>Count of nucleotide C present in the current marker</td>
</tr>
<tr>
<td>nucleotide_t</td>
<td>uint8</td>
<td>Count of nucleotide T present in the current marker</td>
</tr>
<tr>
<td>nucleotide_g</td>
<td>uint8</td>
<td>Count of nucleotide G present in the current marker</td>
</tr>
<tr>
<td>odds_ratio</td>
<td>float64</td>
<td>Odds ratio for the current marker</td>
</tr>
</tbody>
</table>

Table 4: Sharemind’s secret-shared table for disease markers
The research group’s database will contain five tables: disease information table, variant information table, user information table, user log table, and user table status table.

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>marker_id</td>
<td>int</td>
<td>Unique identifier of the variant.</td>
</tr>
<tr>
<td>disease_id</td>
<td>int</td>
<td>Foreign key for the disease table.</td>
</tr>
<tr>
<td>chromosome</td>
<td>int</td>
<td>Autosomal chromosome location of the marker.</td>
</tr>
<tr>
<td>position</td>
<td>int</td>
<td>Position of the variant in the chromosome.</td>
</tr>
<tr>
<td>risk_snps</td>
<td>char[]</td>
<td>Original nucleotide sequence.</td>
</tr>
<tr>
<td>odds_ratio</td>
<td>float</td>
<td>The Odds Ratio of this variant obtained from GWAS.</td>
</tr>
</tbody>
</table>

Table 5: Research group’s marker table

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>disease_id</td>
<td>int</td>
<td>Unique identifier of the disease.</td>
</tr>
<tr>
<td>disease_name</td>
<td>char[]</td>
<td>Common name of the disease.</td>
</tr>
</tbody>
</table>

Table 6: Research group’s disease table

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unit_id</td>
<td>int</td>
<td>Unique identifier of the system user (i.e. medical unit). This value is kept even for the system’s Sharemind table naming convention.</td>
</tr>
<tr>
<td>unit_name</td>
<td>char[]</td>
<td>Name of the medical unit</td>
</tr>
<tr>
<td>email</td>
<td>char[]</td>
<td>Email address of the medical unit; serves as the username of the user.</td>
</tr>
<tr>
<td>password</td>
<td>char[]</td>
<td>Password of the user.</td>
</tr>
<tr>
<td>status</td>
<td>boolean</td>
<td>Status of the user; 0 if disapproved, 1 if approved.</td>
</tr>
</tbody>
</table>

Table 7: Research group’s user information table
<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unit_id</td>
<td>int</td>
<td>Foreign key for the system user (i.e. medical unit).</td>
</tr>
<tr>
<td>content</td>
<td>char[]</td>
<td>Base64-encoded JSON string containing log timestamp and log message.</td>
</tr>
</tbody>
</table>

Table 8: Research group’s user log table

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unit_id</td>
<td>int</td>
<td>Unique identifier of the system user (i.e. medical unit). This entity rep-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>resents the unit table name in the Sharemind database.</td>
</tr>
<tr>
<td>table_id</td>
<td>int</td>
<td>This table id represents the marker table name present in the Share-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>mind database.</td>
</tr>
<tr>
<td>disease_name</td>
<td>char[]</td>
<td>The name of the selected disease; displayed in the retrieve results page.</td>
</tr>
<tr>
<td>timestamp</td>
<td>char[]</td>
<td>The time the upload of the patient variant is completed; also displayed in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the retrieve results page.</td>
</tr>
<tr>
<td>isset</td>
<td>boolean</td>
<td>Indicates whether the user have selected and uploaded a VCF in the Share-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>mind miners or not.</td>
</tr>
</tbody>
</table>

Table 9: Research group’s user table status
2. Entity Relations Diagram

![Entity Relations Diagram](image.png)

Figure 12: Entity Relations Diagram for research group database

C. System Architecture

The genomic DST system is developed over the conventional LAMP (Linux, Apache, MySQL, PHP) stack for server processing and is hosted in the research group’s server. CodeIgniter was the PHP framework used. The client-side application is composed mainly of AngularJS and JQuery as JavaScript frameworks, and Foundation as CSS framework. The system views are developed with HTML5 compliance.

The utilization of two Javascript frameworks (i.e. Angular and JQuery) for the system is necessary because the PHP-based part of the system utilizes the Angular framework for its Javascript functions client-side; while the part integrated with the Sharemind framework utilizes the JQuery framework for its web-based controller library.
Moreover, the choice of using two Javascript frameworks instead of just using the JQuery framework as the sole Javascript framework is driven by the need for robustness of the system: as AngularJS serves better and more optimal rendering of necessary functions for the overall system when compared to the JQuery framework.

Additional client-side application components used for the genomic DST system are as follows: Angular directives for Foundation for Sites 6, MathJax and PDFMake. The Angular directives are used as alternative to the JQuery-based Javascript modules of the Foundation framework. MathJax, on the other hand, is used to render the equations in the about page. PDFMake was used to create a PDF on the browser.
The Sharemind framework utilizes a scripting language called SecreC. This scripting language can be compiled and run in the virtual machine provided by Cybernetica. This virtual machine also comes with pre-compiled miner binary files and a NodeJS controller module. These miner binary files are used to run the Sharemind miners that runs the compiled scripts from SecreC (i.e. SecreC bytecodes); while the NodeJS module, called JSWCP, is used as the main module that connects the Sharemind miners to the PHP-based system and vise versa.

The NodeJS module is used in all three NodeJS servers paired with each miner. These NodeJS servers act as intermediary between the Javascript-based controller library of the genomic DST system and the three Sharemind miners, as illustrated earlier in Figure 4.
Lastly, aside from the use of JQuery, additional Javascript libraries have been used by the controller library of the genomic DST system which are as follows: JSAES, JSBN and JSBN2, TypedArray, and Socket.io. JSAES is used by the controller library in generating pseudo-random numbers which are used in initializing the connection to the NodeJS sever in all three miners; JSBN and JSBN2 are used as libraries that enable the use of Big Integers for type-casting the unsigned integers from the browser to the Sharemind miners; TypedArray is used for type-casting signed integers and floats; and Socket.io is used as the library that enables the connection between the browser and the NodeJS servers.

D. System Development

1. Developing overall system functionalities

The overall system functionalities (i.e. log in, add, edit, and delete functions of the admin; and log in function of the system user) rely heavily on both AngularJS and CodeIgniter. AngularJS was used in the system mainly to retrieve data asynchronously from the PHP server, process these data in the browser, and post the processed data asynchronously to the PHP server. AngularJS, along with Foundation and its corresponding Angular Directives, have been used to generate the alert function, sorting function, pagination function, and modal function. All browser interactions are handled by AngularJS.

On the other hand, CodeIgniter was used for implementing an MVC (Model-View-Controller) architectural pattern for the PHP-based system. CodeIgniter handles the system routing, page generation, database connection, and other utility classes used by the system (e.g. cookie class used for keeping states in page interactions, encryption class used for cookie encryption and decryption, and output class used for the AngularJS asynchronous call response). The system routing function, aside from allowing custom URLs, was mainly utilized in handling user access restrictions (i.e. log in function) in coordination with the user information stored in the browser cookies.
A distinct feature of the administrator function for deleting a disease and user from the system that demands the need for an elaboration is that, aside from performing normal deletion from the MySQL database, a SecreC bytecode is run through the use of the Javascript controller library to delete tables in the Sharemind miners’ database.

Furthermore, most of the functions for the system user (i.e. medical unit) do not rely on CodeIgniter (i.e. PHP server). Instead, a mix of plain Javascript, AngularJS, the Sharemind controller library, and other Javascript libraries were used to perform most of the user functions (i.e. functions processing sensitive data) in the browser.

2. Uploading and processing patient variants

The upload function for the patient variant in VCF file of the system user is processed in the browser by having the browser read the file. This is accomplished by using the Javascript FileReader() class.

```javascript
$scope.processVCF = function( event ) {
    var input = event.target;
    var reader = new FileReader();

    reader.onload = function() {
        app.storeGenotype( reader.result, ... );
    }
    reader.readAsText( input.files[0] );
}
```

The reader.result stores the file contents in String format. The function app.storeGenotype( input string, ... ), on the other hand, parses the reader.result based on VCF specifications and then eventually stores the parsed file content in the Sharemind miners by calling a function from the controller library.
3. Storing disease markers and patient variants

In developing a Javascript procedure to store both the patient variants and research group’s disease markers in the Sharemind miners, a function that stores only a single variant entry is utilized. Processing of the whole variant set is done by recursively calling the function. Thus for every run of the servers.emit() function from the controller library, the callback of the function recursively executes the same function but with updated parameters.

```javascript
app.storeMarkers = function( markers, currentIndex, callback ) {

    serversConnect( function( err_msg ) {

        /** PROCESSING OF arguments USING currentIndex */

        servers.emit( "run_code", arguments, function( data ) {

            // SM RESPONSE
            var row_count = data["row_count"][0];

            if( row_count == markers.length && callback ) {
                callback( function() {
                    $( window ).trigger( "allDone" );
                });
            } else {
                app.storeMarkers( markers, currentIndex + 1, callback );
            }
        });
    });
}
```

The functions serversConnect( callback )—which enables a connection between the controller library and the NodeJS server per miner—and servers.emit( command, arguments, callback )—which enables the browser to send (and receive) data to the NodeJS servers using the established connection—are part of the Sharemind controller library. The command parameter "run_code" in servers.emit() prompts each miner (i.e. through the NodeJS servers) to run the SecreC bytecode indicated in the arguments parameter. The arguments parameter also contains the necessary inputs used by the SecreC bytecode.
arguments = {
    "proxyParams": {
        "codefile": "app_store_markers.sb"
    },
    "smParams": {
        "table_name": table_name,
        "input_chromosome": input_chromosome,
        "input_position": input_position,
        "input_a": input_a,
        "input_c": input_c,
        "input_t": input_t,
        "input_g": input_g,
        "input_odds_ratio": input_odds_ratio,
        "marker_count": marker_count,
    }
};

The inputs in smParams can be classified as a public input or a private input. Classification of these inputs in Javascript is done by utilizing the sm.types.base module of the controller library for the public classification of the input and sm.types.shared3p for private classification.

```javascript
var public_value = sm.types.base.UInt8Array( size );

public_value.set(
    index,
    new BigInteger( value, pseudo-random number )
);

var private_value = sm.types.shared3p.UInt8Array( public_value );
```

4. Calculating risk coefficient

Calculation of the risk coefficient also utilizes the functions serversConnect() and server.emit() in initializing the SecreC bytecode that calculates the risk coefficient of the patient. The bytecode uses Sharemind’s simpleLinearRegression() function in obtaining the y-intercept of the regression line generated by the weight of the marker and its genotypic value in the patient’s variants.

```javascript
D float64 [[1]] simpleLinearRegression(
    D int32/64 [[1]] explanatory variable sample,
    D int32/64 [[1]] dependent variable sample,
    D bool [[1]] available sample filter
);
```
In identifying matches, a nested for-loop is utilized where the outer loop is for the markers while the inner loop is for the patient variants. Inside this for-loop is a conditional statement that declassifies the chromosome and position values for both the marker and the patient variant to check for a match.

```plaintext
for( uint i = 0; i < marker_row_count; i++ ) {
  for( uint j = 0; j < variant_row_count; j++ ) {
    if( declassify( marker_chromosome_column[i] ) ==
        declassify( variant_chromosome_column[j] ) &&
        declassify( marker_position_column[i] ) ==
        declassify( variant_position_column[j] ) ) {
    }
  }
}
```

Calculation of the genotypic value per disease marker match is done by multiplying a vector with elements nucleotide_a, nucleotide_c, nucleotide_t, and nucleotide_g of the marker table to their corresponding vector equivalent in the patient variant table; and finally adding up the elements of the resulting vector to come up with the genotypic value. Markers with no match are automatically assigned a genotypic value of 0.

\[
\begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix}_\text{marker} \times \begin{bmatrix} 0 & 1 & 1 & 0 \end{bmatrix}_\text{patient} = \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix}
\]

\[\Rightarrow 0 + 1 + 0 + 0 = \text{genotypic value} = 1\]

The weight, on the other hand, is obtained by simply calling the \texttt{ln()} function in Sharemind for the odds ratio of the disease marker. Each weight and genotypic value is initially stored in separate vectors. These vectors are then used by the \texttt{simpleLinearRegression()} function. Afterwards, these vectors are multiplied with each other to generate a single vector corresponding the phenotypic values of the patient variants for each disease marker weight. Finally, the resulting vector elements are added; and the resulting sum is added to the obtained \(y\)-intercept of the regression line to obtain the genomic risk coefficient.
5. Generating PDF report

After obtaining the genomic risk coefficient, a simple browser-based PDF report generator is presented. PDFMake was used to generate the PDF report in the browser, without the intervention of a server. AngularJS form field models were used to obtain the form field values using Javascript; PDFMake’s `createPDF(object)` function renders the PDF; and PDFMake’s `getDataUrl(function(uri){})` function returns the PDF data URI.

```javascript
pdfMake.createPDF({
  content: [
    { text: "Patient Information", style: "header" },
    { table: 
      { headerRows: 1, 
        widths: [ '30%', '70%' ],
        body: [ 
          { text: "Patient Name", style: "tableAttr" },
          { text: $scope.patient.name } 
        ],
        body: [ 
          { text: "Patient Age", style: "tableAttr" },
          { text: $scope.patient.age } 
        ]
      }
    },
  ],
  styles: [
    header: { 
      bold: true,
      fontSize: 18,
      margin: [ 0, 20, 0, 10 ]
    },
    tableAttr: { 
      bold: true
    }
  ]
}).getDataUrl( function( uri ) {
  $scope.iframeSrc = uri;
});
```
E. Technical Architecture

The requirements for the overall system are as follows:

- Operating system: Any Linux-based operating system
- For the research group’s server:
  - **Apache**: Any Apache version ≥ 2.4.10
  - **MySQL**: Any MySQL version ≥ 14.4
  - **PHP**: Any PHP version ≥ 5.6.20
- For the Sharemind miners:
  - **Sharemind**: version 3
  - **SecreC**: version 2
  - **NodeJS**: Any NodeJS version ≥ 0.10.29

The Javascript controller library used for connecting to the Sharemind miners, along with its external library dependencies, has been provided for by Cybernetica. The rest that are listed here are those that can be replaced with a different yet complying setting. Also, the system relies on Apache’s `mod_rewrite` for its routing function (i.e. to remove the `index.php` generated by CodeIgniter from the browser URL). Make sure to enable the function in the Apache configuration file of the research group’s server.
V. Results

The genomic DST system is divided into three page types, namely: the general access pages, the admin pages, and the user pages. The general access pages are the pages anyone, including those who are not registered in the system, can access. This includes the log in page, the about page, and the sign-up page.

![Figure 15: Genomic DST system – homepage](image1)

Figure 15: Genomic DST system – homepage

The homepage is also the log in page where approved registered users can calculate for the disease risk of a patient given his/her SNPs in VCF file and generate a PDF report of the calculation result.

![Figure 16: Login page – invalid email alert](image2)

Figure 16: Login page – invalid email alert
Figures 16-18 shows an alert for the different types of errors the user might encounter when logging in to the system – entering an invalid username, entering an invalid password, and accessing the page while still having an unapproved account. The approval of user account can be done by the system administrator. Creation of accounts, however, is performed by the users themselves through the sign-up page.
Figure 19: Genomic DST system – about page

Next in the general access pages is the about page showing an overview of the theoretical concepts for both the computation and the privacy-preservation of patient data used in the system.
Lastly, the sign-up page only contains form fields necessary in identifying one user from the other (i.e. medical unit name, medical unit email, and the account password). Validation and alert for the sign-up page is similar to the log in page and to the rest of the pages in the system.

Figure 21: Sign-up page – invalid email alert
The admin pages are the pages accessible to only the system administrator (i.e. research group) where they can: add, edit, and delete both disease and disease markers; and approve, edit, and delete system users. The admin pages includes the admin homepage, disease catalog page, and user catalog page.

The admin homepage contains information on how to use the system and a simple logging system to track actions done solely by the admin.
The disease catalog page contains all actions relating to both the disease name and disease marker management.

The part of the page enclosed in a yellow box is where the admin can add, edit and delete a disease. In here can the admin also navigate through different disease available by using the pagination provided. By navigating though the diseases using the pagination, the user also updates the visible disease markers (i.e. enclosed in a green box) which corresponds to the markers of the currently navigated disease.

In the disease marker part of the page, the admin can add, edit, and delete disease markers. The admin can also update the markers present in the Sharemind database by pressing the “Update Sharemind Database” button at the bottom part of the page.
Figure 25: Disease catalog page – add disease name already listed

Figure 26: Disease catalog page – invalid disease name alert

Figure 27: Disease catalog page – disease name successfully added
Different from the log in page and the sign up page, the rest of the form submissions in the system for both the admin pages and the user pages is done asynchronously – where there would be no need in changing pages for processing form submissions. This design technique is driven by the purpose of making the user feel at ease with the workarounds in the system. Other features this page and other pages have include the use of pop-up modals and table sorting through clicking the table header.
Figure 30: Disease catalog page – add disease marker modal

Figure 31: Disease catalog page – add disease marker with invalid Risk SNP

Figure 32: Disease catalog page – invalid input alert
Adding disease marker for a specific disease can be done by first navigating through the disease (as discussed earlier), and then clicking the “Add Marker/s” button at the bottom part of the page.

There are two ways to add markers: by single addition or by batch upload. The addition of a single disease marker can be accomplished by simply filling up the form fields in the modal; while the batch upload can be accomplished by uploading a VCF file of the markers to the system.
Editing of disease markers can only be done one by one. This is accomplished by clicking on the marker row displayed in the page. Upon clicking, a modal with form fields already filled up with the selected disease marker information appears.
Figure 38: Disease catalog page – delete disease marker

Figure 39: Disease catalog page – disease marker deleted alert

Figure 40: Disease catalog page – delete disease
Similar to editing disease markers, deleting them can only be done one by one, where the button “Delete Marker” is clicked from the inside of the same modal the edit form appears.

Deleting diseases, however, exhibits a different behavior. Having deleted a disease causes all the markers along it to be deleted as well; and having deleted this disease causes a Sharemind function call that deletes the markers present in each miner’s database. The deletion process is illustrated as a loading screen in the system, as shown in figure 41.
Last for the admin page type is the user catalog page. In this page, the system admin can approve a user sign-up access by ticking on the approve checkbox in the edit modal, edit user information (i.e. medical unit name and medical unit email), and delete user.
Figure 46: User catalog page – user information edited alert

Figure 47: User catalog page – invalid user email alert

Figure 48: User catalog page – delete user
Last of the page types is the user page type. The pages in the user page types are accessible to system users where they can: select a disease for testing, upload a patient’s variant in VCF file, compute for the risk coefficient of the patient for a selected disease, and generate report for the result of the computation.

The homepage for the user is similar to that of the admin homepage: where the user sees a panel containing simple user guidelines for using the system, and a “Recent Activities” panel where users can track their activities in the system. Aside from these, a panel containing a button to start the computation process is accessible in the homepage.
Figure 51: Computation pages – select disease

Figure 52: Computation pages – disease selected

Figure 53: Computation pages – upload page
Figure 54: Computation pages – upload VCF file

Figure 55: Computation pages – uploading to Sharemind miners

Figure 56: Computation pages – upload completed
Figure 57: Computation pages – calculation page

Figure 58: Computation pages – calculating over the Sharemind miners

Figure 59: Computation pages – calculation finished
The pages from selecting a disease to calculating the risk coefficient of a patient given a disease is straightforward: the user selects a disease, uploads a patient variant to the Sharemind miners, and then calculates the risk coefficient of the patient among the Sharemind miners. After the Sharemind calculations comes the generation of report for the result.

Figure 60: Computation pages – generate report page

Figure 61: Computation pages – patient information form fields
The generation of PDF report for this page is all done client-side: where the Javascript library PDFMake was used. The generated PDF report contains essential information like patient name, patient age, patient sex, disease tested, time calculation was done, and the risk coefficient result.
### Patient Information

<table>
<thead>
<tr>
<th>Patient Name</th>
<th>Joseph Niel Tuazon</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patient Age</td>
<td>20</td>
</tr>
<tr>
<td>Patient Sex</td>
<td>Male</td>
</tr>
<tr>
<td>Remarks</td>
<td>none</td>
</tr>
</tbody>
</table>

### Genomic Risk Information

<table>
<thead>
<tr>
<th>Disease Name</th>
<th>Coronary Artery Disease</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generated last</td>
<td>5/22/2016, 5:44:58 PM</td>
</tr>
<tr>
<td>Risk Coefficient</td>
<td>5.535425999316294</td>
</tr>
</tbody>
</table>

Figure 64: Computation pages – sample report PDF
VI. Discussions

The medical unit’s patient variants and research group’s disease markers, along with the computation of the risk coefficient using these data, has been kept private through the use of secure multiparty computation. With the use of different yet complementary application stacks in developing the whole system (i.e. LAMP stack for the web-based system and Sharemind Stack for private storage and computation), privacy-preservation of the genomic disease risk test has been successfully performed. The Sharemind stack consisting of three miners was able to securely store and compute data.

The idea behind connecting the browser directly to the three Sharemind miners was elucidated by the need for easy access of the computation – as web-based systems provide the best mode of access to the computation. To be able to perform a web-based computation while still having preserved a patient’s genomic variants, functions that process sensitive information (i.e. uploading and parsing a patient’s genomic variants, and generating a PDF report with sensitive user information) are run only in the medical unit’s browser using Javascript. This is to remove the conventional process of having the server perform these functions.

A. Storing patient variants and disease markers

The procedure behind storing patient variants is significantly different from storing disease markers. As introduced in the system development section, patient variants are processed by uploading the corresponding VCF file to the browser, having the browser parse the file contents, and sending the parsed contents to the Sharemind database; while for the disease markers, the data stored in the system’s MySQL database are stored to Sharemind by clicking the “Update Sharemind database” button.

For the patient variants, the upload procedure has been successful to an extent. The Javascript file reader has successfully kept the file processing in the browser;
and the use of Javascript has successfully been implemented in parsing the contents of the read file. However, because all the processing is done in the browser, only a certain threshold (i.e. $\lesssim 30,000$) in the number of variants can be tolerated by the current design of the system. This threshold was tested on a Mozilla Firefox 46.0.1 browser.

For both the patient variants and the disease markers, storage of data to the Sharemind database has been successfully implemented by having the controller library recursively send individual row data to the miners instead of sending the whole set of data once. The design technique of sending individual row data instead of the whole data is driven by Sharemind’s lack of methods for indexing multiple arguments (that represents an array) passed onto its SecreC code. Moreover, in sending individual row data, recursion using callbacks was used instead of iteration because of the single-threaded nature of Javascript. Generally, callbacks are used so that the Javascript procedure can wait until a function has finished execution (e.g. an asynchronous reply from the NodeJS servers is received). With iteration, there is no way of waiting for a function running on a different domain (i.e. Sharemind) to finish. Instead, the iteration design will unknowingly call the SecreC bytecode continuously until the iteration has finished – not waiting for the first function call to be finished, thus causing the miners to emit an error back to the controller library.

![Iteration vs. Recursion](image)

Figure 65: Iteration vs. recursion
Another issue with storing data to the Sharemind database is the lack of feature to fully update a table. The only possible way of editing a table row content in the Sharemind database is to delete the existing table and resend the whole dataset. It is possible to add rows, but not to delete or edit particular rows. Thus, for every update in marker entry by the administrator, the whole marker table is deleted and is replaced with a table containing the new set of markers. Storage in patient variant, on the other hand, has no issue regarding updating its table contents because the variants are only temporarily stored (i.e. immediately deleted the first time the calculation bytecode has finished execution). A drawback of temporarily storing patient variants, however, is that the medical unit needs to re-upload the patient’s variants every time a disease risk test is performed.

B. Calculating risk coefficient

In calculating the disease risk of the patient using his/her genomic variants and the research group’s disease markers stored in the Sharemind database, a multiplier of base 10 to represent the odds ratio of the markers as an integer was used. This is done since SecreC’s `simpleLinearRegression()` function only accepts integer arguments. Weights for each marker (i.e. dependent variable sample), as derived by the natural logarithm of the odds ratio with bounds (0, 2), is generally a decimal value between $(-\infty, 0.7)$. The nature of the value of the weights necessitates having a multiplier to represent these weights as integers for regression analysis in SecreC. On the other hand, the explanatory variable sample need not have any multiplier, as the set contains values $\in \{0, 1, 2\}$ (i.e. genotypic value corresponding to the number of causal SNP present in a particular genotype of the patient).

```c
D float64 [[1]] simpleLinearRegression(    D int32/64 [[1]] explanatory variable sample,    D int32/64 [[1]] dependent variable sample,    D bool [[1]] available sample filter);
```

66
Initially, the calculation bytecode only accepts weights ranging from $(0, 0.7)$ which represents a probability range of $(50\%, 100\%)$. This was due to the pre-processing done to the odds ratio before storage where, instead of storing the odds ratio, the unsigned integer representation of the weights are stored. This was done to make use of the `simpleLinearRegression()` function which only accepts integer values.

However, a different approach was implemented to yield a probability range of $(0\%, 100\%)$. By having the odds ratio in its raw form be stored instead of the unsigned integer representation of the weights, the calculation bytecode was able to derive the real value of the weights using the odds ratio, thus yielding a decimal value between $(-\infty, 0.7)$. As a consequence, the probability range is now $(0\%, 100\%)$.

Given this change in approach, the `simpleLinearRegression()` function cannot be used given that the obtained weights are now in decimal value. As a solution, the weight multiplier is now passed onto the bytecode during calculation. This is to represent the weights as unsigned integers while still utilizing the weights’ original decimal values. However, since the weight parameter passed onto the function is in the magnitude similar to the weight multiplier, the resulting $y$-intercept reflects the same magnitude. To obtain the real value of the $y$-intercept, division should be accomplished in the bytecode. Yet, only addition and multiplication can be done in the current version of Sharemind. As a consequence, the intercept is passed along with the obtained base coefficient to a Javascript procedure that performs the division process.

<table>
<thead>
<tr>
<th>number of markers</th>
<th>25</th>
<th>50</th>
<th>75</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>100 patient variants</td>
<td>22 sec</td>
<td>40 sec</td>
<td>61 sec</td>
<td>80 sec</td>
</tr>
<tr>
<td>300 patient variants</td>
<td>33 sec</td>
<td>62 sec</td>
<td>98 sec</td>
<td>126 sec</td>
</tr>
</tbody>
</table>

Table 10: Calculation time per marker count (in seconds)
Performance-wise, calculation of risk coefficient varies directly with the number of markers being processed. Table 10 shows the execution time for the bytecode calculating the risk coefficient with 100% marker coverage in its worst case (i.e. markers are present in the end of the list of patient variants).

![Graph of calculation time per marker count](image)

Figure 66: Graph of calculation time per marker count

The difference in slope of the red line (i.e. 300 variants) and the blue line (i.e. 100 variants) in the above graph indicates that declassification of SecreC variables in the conditional statement of the nested for-loop contributes significantly to the overall execution time of the bytecode. Tests based on best case (i.e. markers are present in the beginning of the list of patient variants), on the other hand, does not exhibit significant change in execution time given varying patient variant counts – therefore consolidating the idea that the positioning of a variant that represents a marker affects greatly the execution time of the bytecode.

C. Sharemind miners

The Sharemind miner that performs both storage and calculation has two components: the Sharemind server and the NodeJS server. As discussed previously, the NodeJS server acts as intermediary between the Javascript controller and the Sharemind server.
For the NodeJS server to be able to securely connect to the Sharemind server, connection using the TLS protocol was used. TLS protocol is a cryptographic protocol that provides Internet security over a computer network [66]. Cybernetica also provided a guide to generating the self-signed certificates used by the TLS connection.

```
openssl req -x509 -days 3650 -nodes -newkey rsa:4096 -keyout host_name-private-key -out host_name-public-key -outform der

openssl rsa -in host_name-private-key -out host_name-private-key -outform der
```

The above command-line code is used in creating a public key and a private key for the host identified as `host_name`. The public key is the key provided to connecting parties while the private key is the key used in the TLS handshake by `host_name`.

The configuration where the certificate of the NodeJS is declared, along with the certificate of the Sharemind server the NodeJS will connect to, can be found in the `serverX.cfg` file of the NodeJS server directory. The identification for both the public and private certificates of the NodeJS server can be set in the `PublicKeyFile` and `PrivateKeyFile` keys respectively; while the identification of the Sharemind server public key can be set in the `PublicIdentity` key of the same configuration file.

```
[Network]
; Identity information
PublicKeyFile=client-public-key
PrivateKeyFile=client-private-key

ConnectTimeout=30000

[MinerNode1]
Address=192.168.17.1
Port=30001
PublicIdentity=server1-public-key
```
Aside from the NodeJS servers connecting to Sharemind and vice versa, the Sharemind servers also use TLS to securely connect to one another. The same key-generation method is used to generate the self-signed certificates used by each server.

The configuration where the certificate of the Sharemind server is identified can be found in the `server.cfg` file in the server root directory. The identification of the server’s public key in `server.cfg` can be set in the `PublicIdentity` key; while the configuration for the identification of the public key for the NodeJS server can be set in the `serverX-whitelist` file.

```ini
[Server 1]
Address = 192.168.17.1
Port = 30001
PublicIdentity = server1-public-key
;OutgoingTlsPriorities = NONE:+CTYPE-X509:+VERS-TLS1.2
:+SIGN-RSA-SHA512:+CURVE-SECP521R1
;MaxPayloadSize = 131059

[Server 2]
Address = 192.168.17.2
Port = 30002
PublicIdentity = server2-public-key
;OutgoingTlsPriorities = NONE:+CTYPE-X509:+VERS-TLS1.2
:+SIGN-RSA-SHA512:+CURVE-SECP521R1
;MaxPayloadSize = 131059
```

The `OutgoingTlsPriorities` key in both configuration files of the NodeJS and the Sharemind servers (i.e. client host) broadcasts to the specified server (i.e. server host) the priority of the TLS versions the client host can use. The configuration that sets the TLS versions supported by the server is identified by the `IncomingTlsPriorities` key found in the server host’s configuration file. Both the `OutgoingTlsPriorities` and the `IncomingTlsPriorities` are used in determining the best version of TLS the connecting parties can use [67].

```ini
; The default TLS priorities for incoming connections
;IncomingTlsPriorities = NONE:+CTYPE-X509:+VERS-TLS1.2
:+SIGN-RSA-SHA512:+CURVE-SECP521R1
```
In relation to network connections, the current implementation for the Share-mind servers can only accept a single connection at a given time; thus, different process executions by multiple connections at a particular time will only return an error to the incoming connections once the first connection is established. Moreover, the first connection will still execute its current process.
VII. Conclusions

The genomic risk coefficient calculator was able to preserve the privacy of both the medical unit’s patient variants and the research group’s disease markers by using Sharemind in implementing secure multiparty computation for disease susceptibility testing.

The system was able to store the patient variants and the disease markers without having to depend on any entity aside from the browser – where they access the system, send their data directly to the NodeJS servers of the Sharemind miners, retrieve the risk result directly, and generate a browser-based PDF report for the result.

The system was also able to calculate the risk coefficient of the patient among the three Sharemind miners, including the calculation of the y-intercept of the regression line used in the additive model that calculates the overall genomic risk coefficient.

Overall, the system was able to fulfill its principal objective: to create a privacy-preserving, web-based genomic disease susceptibility testing which protects the privacy of the medical unit’s patient variants and the research group’s disease markers.
VIII. Recommendations

The genomic disease risk calculator presented in this paper fulfills the fundamental requirements of preserving the privacy of the input data and the computation. However, a lot is yet to be improved in order to prepare the system for real life usage.

A. Patient variant browser processing

The current patient variant upload mechanism of the system can only smoothly handle variants with magnitude below 30,000. Variants higher than 30,000 causes the browser to alert of a continuously-running script, which indicates the ongoing parsing of the uploaded VCF file in the browser. It is therefore recommended that a more optimal way of uploading and parsing the VCF file of the patient in the browser is created so that the browser can still perform other Javascript functions while still running the file parser (i.e. delve into the possible use of multi-threaded Javascript).

B. Calculation bytecode

The current implementation of the calculation for the risk coefficient is designed with basic nested iterations and conditional statements.

```javascript
for( uint i = 0; i < marker_row_count; i++ ) {
    /* process current marker */
    for( uint j = 0; j < variant_row_count; j++ ) {
        /* process current variant */
        if( declassify( marker_chromosome_column[i] ) ==
            declassify( variant_chromosome_column[j] ) &&
            declassify( marker_position_column[i] ) ==
            declassify( variant_position_column[j] ) ) {
            
        }
    }
}
```
The above code indicates that for every marker, the whole variant list (i.e. still including the processed variant matches) is parsed. This imposes unnecessary performance overhead, most especially when many variants are matched and the number of markers is relatively large. Therefore, it is recommended to optimize the code to handle the unnecessary overhead; or rather find a better way to compare for matches between two vectors.

Another improvement for the calculation bytecode is the nature of the input values processed during bytecode execution. The current design of the calculation bytecode relies plainly on raw odds ratio of the disease markers. Thus, calculation of the weight (i.e. natural logarithm of the odds ratio) is performed in the bytecode which adds to its overall execution time. Therefore, it is recommended to find a way to move parts of the calculation from the bytecode and place it into the Javascript controller used by the system in the browser (e.g. perform logarithm calculations in Javascript instead).

Lastly, the possible removal of Javascript calculation used to obtain the real value of the risk coefficient can also be accomplished by finding a way to revert the weights back into its real value. It is recommended to look into multiplying the decimal value of the reciprocal of the weight multiplier so that a division can technically be performed in SecreC.

C. Overall patient disease risk

The system currently calculates only the genomic risk of a patient. As discussed in the Theoretical framework, the overall probability of the patient incurring a disease is based not only from genomic risk but also from clinical and environmental risk factors. It is therefore recommended that clinical and environmental data be integrated into the system. With an additive model for the risk calculation, it is possible for clinical and environmental data to be integrated into the system given that these data can be represented in integer and float values for Sharemind computation.
IX. Bibliography


**X. Appendix**

**A. Source Codes**

1. SecreC source codes

```c
import stdlib;
import table_database;
import shared3p;
import shared3p_table_database;
import shared3p_statistics_regression;

domain pd_shared3p shared3p;

void main() {
    // INITIALIZE DATABASE NAME
    public string dataStore = "DS1";
    // RETRIEVE MARKER TABLE NAME
    uint8 [1] marker_table_name = argument( "marker_table_name" );
    public string marker_table = _vectorToString( marker_table_name );
    // RETRIEVE UNIT TABLE NAME
    uint8 [1] unit_table_name = argument( "unit_table_name" );
    public string unit_table = _vectorToString( unit_table_name );
    // OPEN DATABASE CONNECTION
    tdbOpenConnection( dataStore );
    // GET MARKER TABLE ROW COUNT
    uint marker_row_count = tdbGetRowCount( dataStore, marker_table );
    // GET UNIT TABLE ROW COUNT
    uint variant_row_count = tdbGetRowCount( dataStore, unit_table );
    // GET MARKER TABLE COLUMNS
    pd_shared3p uint8 [1] marker_chromosome_column = tdbReadColumn( dataStore, marker_table, "chromosome" );
    pd_shared3p uint32 [1] marker_position_column = tdbReadColumn( dataStore, marker_table, "position" );
    pd_shared3p uint8 [1] marker_nuc_a_column = tdbReadColumn( dataStore, marker_table, "nucleotide_a" );
    pd_shared3p uint8 [1] marker_nuc_c_column = tdbReadColumn( dataStore, marker_table, "nucleotide_c" );
    pd_shared3p uint8 [1] marker_nuc_t_column = tdbReadColumn( dataStore, marker_table, "nucleotide_t" );
    pd_shared3p uint8 [1] marker_nuc_g_column = tdbReadColumn( dataStore, marker_table, "nucleotide_g" );
    pd_shared3p float64 [1] marker_odds_ratio_column = tdbReadColumn( dataStore, marker_table, "odds_ratio" );
    // GET UNIT TABLE COLUMNS
    pd_shared3p uint8 [1] variant_chromosome_column = tdbReadColumn( dataStore, unit_table, "chromosome" );
    pd_shared3p uint32 [1] variant_position_column = tdbReadColumn( dataStore, unit_table, "position" );
    pd_shared3p uint8 [1] variant_nuc_a_column = tdbReadColumn( dataStore, unit_table, "nucleotide_a" );
    pd_shared3p uint8 [1] variant_nuc_c_column = tdbReadColumn( dataStore, unit_table, "nucleotide_c" );
    pd_shared3p uint8 [1] variant_nuc_t_column = tdbReadColumn( dataStore, unit_table, "nucleotide_t" );
    pd_shared3p uint8 [1] variant_nuc_g_column = tdbReadColumn( dataStore, unit_table, "nucleotide_g" );
    // PREPARE VARIABLES FOR REGRESSION COMputation
    pd_shared3p int [1] weight_vector (marker_row_count) = 0;
    pd_shared3p int [1] genotypic_vector (marker_row_count) = 0;
    pd_shared3p bool [1] bool_array (marker_row_count) = true;
    // GET WEIGHT MULTIPLIER
    pd_shared3p float64 weight_multiplier = argument( "weight_multiplier" );
    // LOOP THROUGH EACH DISEASE MARKER
    for( uint i = 0; i < marker_row_count; i++ ) {
        // COMPUTE CURRENT WEIGHT
        pd_shared3p float64 current_weight = ln( marker_odds_ratio_column[i] );
        // ADD CURRENT WEIGHT TO VECTOR
        weight_vector[i] = (int)(current_weight*weight_multiplier);
        // LOOP THROUGH EACH PATIENT VARIANT
        for( uint j = 0; j < variant_row_count; j++ ) {
            // CHECK IF CHROMOSOME AND POSITION ARE THE SAME
            if( declassify( marker_chromosome_column[i] ) == declassify( variant_chromosome_column[j] ) &&
                declassify( marker_position_column[i] ) == declassify( variant_position_column[j] ) ) {
                // GET MARKER NUCLEOTIDE
                pd_shared3p uint8 marker_a = marker_nuc_a_column[i];
                pd_shared3p uint8 marker_c = marker_nuc_c_column[i];
                pd_shared3p uint8 marker_t = marker_nuc_t_column[i];
                pd_shared3p uint8 marker_g = marker_nuc_g_column[i];
                // GET PATIENT VARIANT GENOTYPE
                pd_shared3p uint8 variant_a = variant_nuc_a_column[j];
                pd_shared3p uint8 variant_c = variant_nuc_c_column[j];
                pd_shared3p uint8 variant_t = variant_nuc_t_column[j];
                pd_shared3p uint8 variant_g = variant_nuc_g_column[j];
                // COMPUTE FOR GENOTYPIC VALUE
                genotypic_vector[i] = ((int)(marker_a*variant_a)) +
                                    ((int)(marker_c*variant_c)) +
                                    ((int)(marker_t*variant_t)) +
                                    ((int)(marker_g*variant_g));
            } // ESCAPE INNER LOOP
        } // LOOP THROUGH EACH PATIENT VARIANT
    } // LOOP THROUGH EACH DISEASE MARKER
    // PERFORM SIMPLE LINEAR REGRESSION
    pd_shared3p float64 [1] result_vector = simpleLinearRegression(
        genotypic_vector, weight_vector, bool_array
    );
    // GET THE INTERCEPT OF THE CURRENT MODEL
    pd_shared3p float64 intercept = result_vector[0];
    // GET THE GENOMIC REGRESSION COEFFICIENT
    pd_shared3p int [1] vector_product = weight_vector*genotypic_vector;
    pd_shared3p int scalar_sum = sum( vector_product );
    // PUBLISH BOTH THE REGRESSION COEFFICIENT
    // AND THE INTERCEPT OF THE REGRESSION MODEL
    publish( "coefficient", scalar_sum );
}
```

---

84
publish( "intercept", intercept );

// DELETE TABLE OF UNIT AFTER PUBLISHING RESULTS
tdbTableDelete( dataStore, unit_table );

// CLOSE DATABASE CONNECTION
tdbCloseConnection( dataStore );
}

app_store_genotypic_value.sc
import stdlib;
import table_database;
import shared3p;
import shared3p_table_database;
domain pd_shared3p shared3p;

void main() {

  // INITIALIZE DATABASE NAME
  public string dataStore = "DS1";

  // RETRIEVE UNIT TABLE NAME
  uint8 [[1]] table_name = argument( "table_name" );
  public string table = _vectorToString( table_name );

  // OPEN DATABASE CONNECTION
tdbOpenConnection( dataStore );

  // GET CURRENT GENOTYPE ROW COUNT
  // NOTE - RUN_CODE IS EXECUTED PER GENOTYPE ROW
  pd_shared3p uint8 marker_count = argument( "marker_count" );
  uint8 marker_start_count = 0;

  // CHECK IF TABLE EXISTS FOR THE FIRST RUN
  if( tdbTableExists( dataStore, table ) &&
      declassify( marker_count ) == marker_start_count ) {
    // DELETE TABLE IF PRESENT FOR THE FIRST RUN
    tdbTableDelete( dataStore, table );
  }

  // CHECK IF TABLE DOES NOT EXIST
  if ( !tdbTableExists( dataStore, table ) ) {
    // PREPARE TABLE COLUMNS BY CREATING A VECTOR MAP
    uint parameters = tdbVmapNew();
    // CHROMOSOME COLUMN
    pd_shared3p uint8 vtype_chromosome;
    tdbVmapAddType( parameters, "types", vtype_chromosome );
    tdbVmapAddString( parameters, "names", "chromosome" );
    // POSITION COLUMN
    pd_shared3p uint32 vtype_position;
    tdbVmapAddType( parameters, "types", vtype_position );
    tdbVmapAddString( parameters, "names", "position" );
    // NUCLEOTIDE A COUNT COLUMN
    pd_shared3p uint8 vtype_a;
    tdbVmapAddType( parameters, "types", vtype_a );
    tdbVmapAddString( parameters, "names", "nucleotide_a" );
    // NUCLEOTIDE C COUNT COLUMN
    pd_shared3p uint8 vtype_c;
    tdbVmapAddType( parameters, "types", vtype_c );
    tdbVmapAddString( parameters, "names", "nucleotide_c" );
    // NUCLEOTIDE T COUNT COLUMN
    pd_shared3p uint8 vtype_t;
    tdbVmapAddType( parameters, "types", vtype_t );
    tdbVmapAddString( parameters, "names", "nucleotide_t" );
    // NUCLEOTIDE G COUNT COLUMN
    pd_shared3p uint8 vtype_g;
    tdbVmapAddType( parameters, "types", vtype_g );
    tdbVmapAddString( parameters, "names", "nucleotide_g" );
    // CREATE NEW TABLE
tdbTableCreate( dataStore, table, parameters );

    // CLEAR VECTOR MAP VALUE
    tdbVmapClear( parameters );

    // DELETE VECTOR MAP
    tdbVmapDelete( parameters );

    // PREPARE TABLE ROW TO INSERT BY CREATING A VECTOR MAP
    uint parameters = tdbVmapNew();
    // CHROMOSOME INPUT VALUE
    pd_shared3p uint8 input_chromosome = argument( "input_chromosome" );
    tdbVmapAddValue( parameters, "values", input_chromosome );

    // POSITION INPUT VALUE
    pd_shared3p uint32 input_position = argument( "input_position" );
    tdbVmapAddValue( parameters, "values", input_position );

    // NUCLEOTIDE A COUNT INPUT VALUE
    pd_shared3p uint8 input_a = argument( "input_a" );
    tdbVmapAddValue( parameters, "values", input_a );

    // NUCLEOTIDE C COUNT INPUT VALUE
    pd_shared3p uint8 input_c = argument( "input_c" );
    tdbVmapAddValue( parameters, "values", input_c );

    // NUCLEOTIDE T COUNT INPUT VALUE
    pd_shared3p uint8 input_t = argument( "input_t" );
    tdbVmapAddValue( parameters, "values", input_t );

    // NUCLEOTIDE G COUNT INPUT VALUE
    pd_shared3p uint8 input_g = argument( "input_g" );
    tdbVmapAddValue( parameters, "values", input_g );

    // INSERT MAPPED VALUES TO DATABASE TABLE
    tdbInsertRow( dataStore, table, parameters );

    // CLEAR VECTOR MAP VALUE
    tdbVmapClear( parameters );

    // DELETE VECTOR MAP
    tdbVmapDelete( parameters );

    // PUBLISH CURRENT ROW COUNT
    public uint row_count = tdbGetRowCount( dataStore, table );
publish( "row_count", row_count );

    // CLOSE DATABASE CONNECTION
tdbCloseConnection( dataStore );
  }
}

app_store_markers.sc
import stdlib;
import table_database;
import shared3p;
import shared3p_table_database;
domain pd_shared3p shared3p;

void main() {

  // INITIALIZE DATABASE NAME
  public string dataStore = "DS1";

  // RETRIEVE TABLE NAME
  uint8 [[1]] table_name = argument( "table_name" );
  public string table = _vectorToString( table_name );

  // OPEN DATABASE CONNECTION
tdbOpenConnection( dataStore );

  // GET CURRENT GENOTYPE ROW COUNT
  // NOTE - RUN_CODE IS EXECUTED PER GENOTYPE ROW
  pd_shared3p uint8 marker_count = argument( "marker_count" );
  uint8 marker_start_count = 0;

  // CHECK IF TABLE EXISTS FOR THE FIRST RUN
  if( tdbTableExists( dataStore, table ) &&
      declassify( marker_count ) == marker_start_count ) {
    // DELETE TABLE IF PRESENT FOR THE FIRST RUN
    tdbTableDelete( dataStore, table );
  }

  // CHECK IF TABLE DOES NOT EXIST
  if ( !tdbTableExists( dataStore, table ) ) {
    // PREPARE TABLE COLUMNS BY CREATING A VECTOR MAP
    uint parameters = tdbVmapNew();
    // CHROMOSOME COLUMN
    pd_shared3p uint8 vtype_chromosome;
    tdbVmapAddType( parameters, "types", vtype_chromosome );
    tdbVmapAddString( parameters, "names", "chromosome" );
    // POSITION COLUMN
    pd_shared3p uint32 vtype_position;
    tdbVmapAddType( parameters, "types", vtype_position );
    tdbVmapAddString( parameters, "names", "position" );
    // NUCLEOTIDE A COUNT COLUMN
    pd_shared3p uint8 vtype_a;
    tdbVmapAddType( parameters, "types", vtype_a );
    tdbVmapAddString( parameters, "names", "nucleotide_a" );
    // NUCLEOTIDE C COUNT COLUMN
    pd_shared3p uint8 vtype_c;
    tdbVmapAddType( parameters, "types", vtype_c );
    tdbVmapAddString( parameters, "names", "nucleotide_c" );
    // NUCLEOTIDE T COUNT COLUMN
    pd_shared3p uint8 vtype_t;
    tdbVmapAddType( parameters, "types", vtype_t );
    tdbVmapAddString( parameters, "names", "nucleotide_t" );
    // NUCLEOTIDE G COUNT COLUMN
    pd_shared3p uint8 vtype_g;
    tdbVmapAddType( parameters, "types", vtype_g );
    tdbVmapAddString( parameters, "names", "nucleotide_g" );
    // CREATE NEW TABLE
tdbTableCreate( dataStore, table, parameters );

    // CLEAR VECTOR MAP VALUE
    tdbVmapClear( parameters );

    // DELETE VECTOR MAP
    tdbVmapDelete( parameters );

    // PREPARE TABLE ROW TO INSERT BY CREATING A VECTOR MAP
    uint parameters = tdbVmapNew();
    // CHROMOSOME INPUT VALUE
    pd_shared3p uint8 input_chromosome = argument( "input_chromosome" );
    tdbVmapAddValue( parameters, "values", input_chromosome );

    // POSITION INPUT VALUE
    pd_shared3p uint32 input_position = argument( "input_position" );
    tdbVmapAddValue( parameters, "values", input_position );

    // NUCLEOTIDE A COUNT INPUT VALUE
    pd_shared3p uint8 input_a = argument( "input_a" );
    tdbVmapAddValue( parameters, "values", input_a );

    // NUCLEOTIDE C COUNT INPUT VALUE
    pd_shared3p uint8 input_c = argument( "input_c" );
    tdbVmapAddValue( parameters, "values", input_c );

    // NUCLEOTIDE T COUNT INPUT VALUE
    pd_shared3p uint8 input_t = argument( "input_t" );
    tdbVmapAddValue( parameters, "values", input_t );

    // NUCLEOTIDE G COUNT INPUT VALUE
    pd_shared3p uint8 input_g = argument( "input_g" );
    tdbVmapAddValue( parameters, "values", input_g );

    // INSERT MAPPED VALUES TO DATABASE TABLE
    tdbInsertRow( dataStore, table, parameters );

    // CLEAR VECTOR MAP VALUE
    tdbVmapClear( parameters );

    // DELETE VECTOR MAP
    tdbVmapDelete( parameters );

    // PUBLISH CURRENT ROW COUNT
    public uint row_count = tdbGetRowCount( dataStore, table );
publish( "row_count", row_count );

    // CLOSE DATABASE CONNECTION
tdbCloseConnection( dataStore );
  }
}
2. Javascript source codes

```javascript
import stdlib;
import table_database;
import shared3p;
import shared3p_table_database;

domain pd_shared3p shared3p;

void main() {
    // INITIALIZE DATABASE NAME
    public string dataStore = "DS1";

    // RETRIEVE TABLE NAME
    uint8 table_name = argument("table_name");
    public string table = _vectorToString( {table_name} );

    // OPEN DATABASE CONNECTION
    tdbOpenConnection( dataStore );

    // CHECK IF TABLE EXISTS
    if( tdbTableExists( dataStore, table ) ) {
        // DELETE TABLE
        tdbTableDelete( dataStore, table );
    }
    // PUBLISH NAME OF TABLE DELETED
    publish("table_name", table_name);

    // CLOSE DATABASE CONNECTION
    tdbCloseConnection( dataStore );
}
```

---

```
// INITIALIZE DATABASE NAME
public string dataStore = "DS1";

// RETRIEVE TABLE NAME
uint8 table_name = argument( "table_name" );
public string table = _vectorToString( {table_name} );

// OPEN DATABASE CONNECTION
tdbOpenConnection( dataStore );

// CHECK IF TABLE EXISTS
if( tdbTableExists( dataStore, table ) ) {
    // DELETE TABLE
    tdbTableDelete( dataStore, table );
}
// PUBLISH NAME OF TABLE DELETED
publish( "table_name", table_name );
// CLOSE DATABASE CONNECTION
tdbCloseConnection( dataStore );
```
app.js

(function() {

  "use strict";

  angular

    .module("tuazon.genomicdst", [
    "mm.foundation", "ngCookies"])

    .factory("alertFactory", function() {
        return {
            alerts: [],
            addAlert: function(type, msg) {
                this.alerts.push({
                    type: type,
                    msg: msg
                });
            },
            closeAlert: function(index) {
                this.alerts.splice(index, 1);
            }
        }
    })

    .controller("alertCtrl", ['$scope', '$timeout', 'alertFactory', function($scope, $timeout, alertFactory) {
        $scope.alerts = alertFactory.alerts;
        $scope.credentials = {};
        $scope.initializeAlert = function(settings) {
            if (settings.alert != undefined) {
                var alertIndex = alertFactory.addAlert(settings.alert.type, settings.alert.msg);
                if (settings.type == "login") {
                    $scope.credentials = { username: settings.email };
                } else if (settings.type == "signup") {
                    $scope.credentials = {
                        name: settings.name,
                        email: settings.email
                    };
                }
                $timeout(function() {
                    alertFactory.closeAlert(alertIndex);
                }, 5000);
            }
        }
    }])

    .controller("adminCtrl", ['$http', '$scope', 'alertFactory', function($http, $scope, alertFactory) {
        $scope.alerts = alertFactory.alerts;
        $scope.initializeAlert = function(settings) {
            if (settings.alert != undefined) {
                var alertIndex = alertFactory.addAlert(settings.alert.type, settings.alert.msg);
                if (settings.type == "login") {
                    $scope.credentials = { username: settings.email };
                } else if (settings.type == "signup") {
                    $scope.credentials = {
                        name: settings.name,
                        email: settings.email
                    };
                }
                $timeout(function() {
                    alertFactory.closeAlert(alertIndex);
                }, 5000);
            }
        }
    }])

    .controller("adminDiseaseCtrl", ['$http', '$scope', '$timeout', '$filter', 'alertFactory', '$modal', '$log', function($http, $scope, $timeout, $filter, alertFactory, $modal, $log) {
        $scope.alerts = alertFactory.alerts;
        $scope.diseaseCount = 0;
        $http.get('./get/diseases').success(function(data) {
            $scope.diseases = data.diseases;
            $scope.currentDisease = $scope.diseases[0];
            $scope.diseaseCount = $scope.diseases.length;
            $scope.noOfDisease = $scope.diseases.length + " disease" + (1 > $scope.diseases.length ? "s" : "");
        });
        $scope.markerSearches = [{
            id: 0,
            predicate: "Chromosome",
            key: "chromosome"
        }, {
            id: 1,
            predicate: "Position",
            key: "position"
        }, {
            id: 2,
            predicate: "Risk SNP",
            key: "risk_snp"
        }, {
            id: 3,
            predicate: "Odds Ratio",
            key: "odds_ratio"
        }];
        $scope.markerSearchOption = $scope.markerSearches[0];
        $scope.searchForMarker = function() {
            var e = {};
            e[$scope.markerSearchOption.key] = $scope.markerSearchTerm;
            var currentMarkers = $filter('filter')($scope.currentMarkers, e);
            $scope.tempCurrentMarkers = currentMarkers;
            $scope.tempMarkerCount = currentMarkers.length;
            $scope.tempNoOfMarker = (1 > $scope.tempMarkerCount ? "s" : "");
        };
        $scope.order = function(e) {
            var e = {},
                a = (e.orderBy) ? e.orderBy : "chromosome",
                e = (a == "chromosome") ? a : "";
            $scope.currentMarkers = $scope.currentMarkers.sort($filter('orderBy')(currentMarkers, e, $scope.reverse), $scope.reverse ? "asc" : "desc");
            $scope.currentMarkers = e == "chromosome" ? ($scope.currentMarkers.filter($scope.currentMarkers[0].disease_id == $scope.currentMarkers[1].disease_id)) : ($scope.currentMarkers.filter($scope.currentMarkers[0].disease_id == $scope.currentMarkers[1].disease_id));
        };
        $scope.count = function() {
            $scope.currentMarkers = ($scope.currentMarkers.count) ? $scope.currentMarkers.count : 0;
        };
    })

admin.app.js

(function() {

  "use strict";

  angular

    .module("tuazon.genomicdst.admin", [
    "mm.foundation", "tuazon.genomicdst"])

    .controller("adminCtrl", ['$http', '$scope', function($http, $scope) {
        $scope.get({
            start: 0,
            end: 10
        }).success(function(data) {
            $scope.user_name = data.user_name;
            $scope.logs = data.log;
            $scope.max_log = data.max_log;
            $scope.start = 0;
            $scope.end = 10;
        });
    }])

78
a.diseasePaginationMaxSize = 1,
a.diseasePaginationCurrentPage = 1,
a.diseasePaginationDisplayNumber = 5,
a.diseasePaginationMinIndex = function () {
    return a.diseasePaginationMaxSize * a.diseasePaginationCurrentPage -
a.diseasePaginationMaxSize,
a.diseasePaginationMaxIndex = function () {
    return a.diseasePaginationMaxSize * a.diseasePaginationCurrentPage -
a.diseasePaginationMaxSize,
a.diseasePaginationMaxIndex = function () {
    return a.diseasePaginationMaxSize * a.diseasePaginationCurrentPage -
a.diseasePaginationMaxSize,
a.diseasePaginationShoutPage = function (e) {
    return a.diseasePaginationMinIndex() <= e &&
a.diseasePaginationMaxIndex() > e
}, a.diseasePaginationShoutPage = function (e) {
    return a.diseasePaginationMinIndex() <= e &&
a.diseasePaginationMaxIndex() > e
}, a.diseasePaginationShowPage = function (e) {
    return a.diseasePaginationMinIndex() <= e &&
a.diseasePaginationMaxIndex() > e
}, a.paginationMaxSize = 10, a.paginationCurrentPage = 1,
a.paginationDisplayNumber = 2, a.paginationMinIndex = function () {
    return a.paginationMaxSize * a.paginationCurrentPage -
a.paginationMaxSize
}, a.paginationMaxIndex = function () {
    return a.paginationMaxSize * a.paginationCurrentPage
}, a.paginationShowPage = function (e) {
    return a.paginationMinIndex() <= e &&
a.paginationMaxIndex() > e
}, a.addDiseaseName = function () {
    e.post('./post/add_disease', {
        disease_name: a.disease_name
    }).success(function (e) {
        angular.copy(e.diseases, a.diseases), t.addAlert(e.alert.type,
        e.alert.msg), s(function () {
            t.closeAlert()
        }, 5e3), a.diseaseCount = a.diseases.length,
a.noOfDisease = a.diseaseCount +
        " disease" + (a.diseaseCount > 1 ? "s" : ""), a.noOfMarker = a.markerCount +
        " marker" + (a.markerCount > 1 ? "s" : ""), a.diseasePaginationCurrentPage = 1,
a.updateMarkers(a.diseasePaginationCurrentPage, !1)
}), a.editDiseaseName = function (r) {
    a.diseases[r].disabled || e.post('./post/edit_disease', {
        disease: a.diseases[r]
    }).success(function (e) {
        angular.copy(e.diseases, a.diseases), t.addAlert(e.alert.type,
        e.alert.msg), s(function () {
            t.closeAlert()
        }, 5e3), a.updateMarkers(a.diseasePaginationCurrentPage, !1)
}), a.deleteDiseaseName = function (e) {
    var m = a.diseases[r].disease_name;
    if (confirm("Are you sure you want to delete " + m + 
        " from the database?")
    ) {
        var i = function () {
            e.post('./post/delete_disease', {
                disease: m
            }).success(function (e) {
                a.diseases.splice(r, 1), a.diseases.length <= r ? a.updateMarkers(
                    r, 1) : a.updateMarkers(r + 1, 11),
                t.addAlert(e.alert.type, e.alert.msg),
                s(function () {
                    t.closeAlert()
                }, 5e3), a.diseaseCount = a.diseases.length,
                a.noOfDisease = a.diseaseCount +
                " disease" + (a.diseaseCount > 1 ? "s" : ""),
                a.noOfMarker = a.markerCount +
                " marker" + (a.markerCount > 1 ? "s" : "")
            }, 5e3)
        }, function (e) {
            if ("Modal dismissed at: " + new Date) {
                a.deleteTable(a.diseases[r].disease_id, i)
            }
        }, function (e) {
            i.info("Modal dismissed at: " + new Date)
        })
    })
}, a.updateToSharemind = function () {
    var e = function () {
        t.addAlert("success", "Sharemind database updated!")
    },
    function () {
        t.closeAlert()
    }, 5e3,
    e && e()
}, app.init(10), app.storeMarkers(a.currentMarkers, 0, a)
), a.openAddMarkerModal = function () {
    n.open({
        templateUrl: "adminAddMarkerModal.html",
        controller: "adminAddMarkerModal",
        size: "small",
        resolve: {
            data: function () {
                return {
                    disease_id: a.currentDisease.disease_id,
                    disease_name: a.currentDiseaseName
                }
            }
        }
    })
}, a.openModal = function (e, r) {
    n.open({
        templateUrl: "adminEditMarkerModal.html",
        controller: "adminEditMarkerModal",
        size: "small",
        resolve: {
            data: function () {
                return e
            },
            index: function () {
                return r
            }
        }
    })
}.

.controller("adminAddMarkerModal", ['$scope', $modalInstance, $http, function (e, a, s, r) {
    e.marker = {}, e.currentDiseaseName = r.disease_name,
    e.batchUpload = function (e) {
        app.init(10), app.deleteTable(a.diseases[r]
var t = e.target, n = new FileReader;
n.onload = function () {
  var e = {
    file: n.result
  };
  angular.extend(e, r.disease_id), $($(".loader"))
  .fadeIn(), s.post("./post/add_batch_marker", e)
  .success(function (e) {
    $(".loader")
    .fadeOut(), a.close(e)
  });
}, n.readAsText(t.files[0])
}, e.addMarker = function () {
  angular.extend(e.marker, r.disease_id), s.post("./post/add_marker", e.marker)
  .success(function (e) {
    a.close(e)
  })
}, e.discardChanges = function () {
  a.dismiss(null)
}, e.deleteMarker = function () {
  confirm("Proceed to deleting marker?"),
  var t = {
    marker: {
      old: r,
      new: e.marker
    }
  }, s.post("./post/delete_marker", t)
  .success(function (e) {
    a.close({
      edit: e
    })
  })
}, e.saveChanges = function () {
  a.dismiss(null)
}, e.deleteUser = function () {
  confirm("Proceed to deleting user: ", r.unit_name + "? This cannot be undone.")
  s.post("./post/delete_user", {
    user: r
  })
  .success(function (e) {
    a.close({
      edit: e
    })
  })
}, e.isApprovedToggled = function () {
  return 0 == e.user.status.value ? !1 : !0
}, e.saveChanges = function () {
  if (confirm('Proceed to editing user: "' + r.unit_name + '"?')) {
    var t = {
      user: {
        old: r,
        new: e.user
      }
    }, s.post("./post/edit_user", t)
    .success(function (e) {
      a.close({
        edit: e
      })
    })
  }
}, e.discardChanges = function () {
  a.dismiss(null)
}, e.deleteUser = function () {
  confirm('Proceed to deleting user: "' + r.unit_name + '" This cannot be undone.')
  s.post("./post/delete_user", {
    user: r
  })
  .success(function (e) {
    a.close({
      edit: e
    })
  })
}, a.openModal = function (e, r) {
  n.open({
    templateUrl: "adminUserModal.html",
    controller: "AdminUsersModal",
    size: "small",
    resolve: {
      data: function () {
        return e
      },
      index: function () {
        return r
      }
    }
  })
}.
user.js

```javascript
function () {
    "use strict";
    angular.module("tuazon.genomicst.user", ["mm.foundation", "tuazon.genomicst"])
        .controller("userCtrl", ["$http", "$scope", function (A, m) {
            A.get('./user/get/home').success(function (A) {
                m.user_name = A.user_name, m.logs = A.log, m.max_log = A.max_log
            }), m.start = 0, m.end = 10, m.logs = [], m.hasActivityLog = function () {
                return m.logs.length > 0
            }, m.isPreviousDisabled = function () {
                var A = m.start + 10 <= m.max_log
            }, m.isNextDisabled = function () {
                return 0 == m.start
            }, m.label = function () {
                var A = m.isPreviousDisabled() ? m.max_log : m.end
                return "Activities " + (m.start + 1) + " - " + A
            }, m.showMore = function (z) {
                var M, E, a = !1;
                !m.isPreviousDisabled() && z ? (M = m.start + 10, E = m.end + 10, a = !0) : m.isNextDisabled() || z || (M = m.start - 10, E = m.end - 10, a = !0), a && A.post('./user/post/update_log', {start: M, end: E}).success(function (A) {
                    m.logs = A.log, m.start = M, m.end = E
                })
            }
        }])
        .controller("userDiseaseSelectionCtrl", ["$http", "$scope", function (A, m) {
            A.get('./get/diseases').success(function (A) {
                m.diseases = A.diseases, m.numOfDiseases = m.diseases.length
            }), m.isDiseaseSelected = function (A) {
                return m.diseaseName == m.diseases[A].disease_name
            }, m.diseaseName = "", m.radioToggled = function (A) {
                m.diseaseName = m.diseases[A].disease_name
            }
        }])
        .controller("userPatientUploadCtrl", ["$http", "$scope", "$timeout", "alertFactory", function (A, m, z, M) {
            A.get('./get/upload').success(function (A) {
                m.unit_table_name = A.unit_table_name
            }), m.isUploaded = !1, m.processVCF = function (E) {
                var a = E.target, I = new FileReader, i = function (E) {
                    A.post('./post/upload_time', {upload_time: (new Date).toLocaleString()}).success(function (A) {
                        M.addAlert(A.alert.type, A.alert.msg), z(function () {
                            M.closeAlert()
                        }, 5e3), m.isUploaded = !0, E && E()
                    });
                    I.onload = function () {
                        app.init(!0, !0), app.storeGenotype(I.result, m.unit_table_name, i)
                    }
                    I.readAsText(a.files[0])
                }
            }
        }])
        .controller("userPatientRetrieveCtrl", ["$http", "$scope", "$timeout", "$cookies", "alertFactory", function (A, m, z, M, E) {
            A.get('./get/retrieve').success(function (A) {
                m.unit_table_name = A.unit_table_name, m.marker_table_name = A.marker_table_name, m.disease_name = A.disease_name, m.upload_time = A.upload_time, m.weight_multiplier = A.weight_multiplier
            }), m.isFinished = !1, m.isReady = function () {
                return 0 != m.marker_table_name
            }, m.retrieveResult = function () {
                var z = function (z, E) {
                    A.get('./get/retrieve_multiplier').success(function (A) {
                        m.isFinished = !0, m.coefficient = parseFloat(parseFloat(E) / parseInt(A.weight_multiplier)) + parseFloat(parseInt(z) / parseInt(A.weight_multiplier)) + parseFloat(parseInt(z) / parseInt(A.weight_multiplier))
                    });
                }
                app.init(!0), app.retrieveComputationResult(m.unit_table_name, m.marker_table_name, m.weight_multiplier, z)
            }
        }])
        .controller("userPatientGenerateCtrl", ['$scope', '$cookies', function (A, m) {
            A.generate_data = m.getObject("generate_data"), A.isFinished = !1, A.generateReport = function () {
                $scope.$apply(function () {
                    fadeIn(function () {
                        (new Date).toLocaleString().replace(/\d{2}/g).replace(/\d{2}/g).replace(/\d{2}/g) + ".pdf";
                        pdfMake.createPdf({
                            content: [
                                "columns: 1,
                                image: "upmLogo",
                                width: 70,
                                height: 70,
                                text: "Privacy-preserving Genomic ...",
                                style: "title"
                            ],
                            content: [
                                "columns: 1,
                                image: "upmLogo",
                                width: 85,
                                height: 70,
                                text: "Patient Information",
                                style: "header"
                            ],
                            table: {
                                headerRows: 1,
                                width: ["300", "700"],
                                body: [
                                    [
                                        "Patient Information",
                                        "Cornerstone",
                                        "Patient Identification Number",
                                        "Gender",
                                        "Date of Birth",
                                        "Weight"
                                    ],
                                    [
                                        "0001",
                                        "Male",
                                        "1984-05-12",
                                        "80"
                                    ]
                                ]
                            }
                        });
                    });
                });
            }
        }])
```
| Patient Name | A.patient.name |
| Patient Age | A.patient.age.toString() |
| Patient Sex | A.patient.sex |
| Remarks | A.patient.remarks |

### Genomic Risk Information

| Disease Name | A.generate_data.disease_name |
| Generated last | A.generate_data.time_calculated |
| Risk Coefficient | A.generate_data.coefficient.toString() |

admin.sm.app.js

```javascript
function (e, n, r) {
    if (null === i) {
        e.debug("Sharemind not yet connected.");
        try {
            e.log("Connecting to Sharemind.");
            var l = function () {
                o.log("Connected to Sharemind.");
                var r = new n.types.base.Uint32Array();
                r.set(0, 32);
                o.proxyParams = {
                    codefile: "get_random.sb",
                    generate: true
                },
                l.bytes = r, o.smParams = l, i.emit("run_code",
                    function (r) {
                        e.log("Got randomness for PRNG."),
                        n.types.prng = new PRNG(r.random),
                        n.types.prng ? (e.log("PRNG created."), void (t)())
                    })
            },
            u = function (n) {
                e.error(n), l(), r(window).trigger("allDone")
            },
            t = function () {
                e.debug("Sharemind not yet connected.");
                try {
                    e.log("Connecting to Sharemind.");
                    var l = function () {
                        o.log("Connected to Sharemind.");
                        var r = new n.types.base.Uint32Array();
                        r.set(0, 32);
                        var o = {};
                        o.proxyParams = {
                            codefile: "get_random.sb"
                        },
                        l.bytes = r, o.smParams = l, i.emit("run_code",
                            o,
                            function (r) {
                                return e.log("Got randomness for PRNG."),
                                    n.types.prng = new PRNG(r.random),
                                    n.types.prng ? (e.log("PRNG created.",
                                    void (t)()) : (e.log("Failed to create PRNG!",
                                    void u()))
                            }),
                            u = function (n) {
                                e.error(n), t(n), r(window)
                                    trigger("allDone")
                            };
                    }
                } catch (n) {
                    e.error(n), t(n), r(window).trigger("allDone")
                }
            };
            u = function (n) {
                e.error(n), t(n), r(window).trigger("allDone")
            };
        } catch (n) {
            e.error(n), t(n), r(window).trigger("allDone")
        }
    })
}
```
i = n.ctrl.connect(o, l, u)
} catch (s) {
    return e.debug("Failed to connect to Sharemind: " + s), t("Failed to connect to Sharemind: " + s), void a()}
} else t();

function a() {
    null != i && (e.log("Disconnecting from Sharemind."). disconnect(), delete i, i = null, e.log("Disconnected from Sharemind."))
}

var o = 
    i = null,
    l = n.types.base,
    u = n.types.shared3p;

    e.init = function (n) {
        n && r(".loader").fadeIn(), r(window).on("allDone", function () {
            n && r(".loader").fadeOut(), e.debug("All done!"). a()})
    },
    e.debug = function (e) {
        console.debug("[DEBUG]| " + e)
    },
    e.log = function (e) {
        console.log("[INFO] | " + e)
    },
    e.error = function (e) {
        console.error("[ERROR]| " + e)
    },
    e.deleteTable = function (n, a) {
        e.log("[deleteDiseaseTable()] FUNCTION INITIALIZED."). t(function (t) {
            if (t) return void e.error(t);
            e.log("[deleteDiseaseTable()] DELETING TABLE "+ n + ");
            var o = {};
            o.proxyParams = {
                codefile: "app_delete_table.sb"};
            var u = new l.Uint8Array(1);
            u.set(0, new BigInteger(n + ", 10"));
            o.smParams = {
                table_name: u,
                is: function () {
                    e.log("[storeMarkers()] SUCCESSFULLY DELETED TABLE "+ n + ");
                    a && a(function () {
                        r(window).trigger("allDone")})}}
        });
    },
    e.storeMarkers = function (n, a, o) {
        e.log("[storeMarkers()] FUNCTION INITIALIZED."). t(function (t) {
            if (t) return void e.error(t);
            e.log("[storeMarkers()] PROCESSING MARKERS ");
            var s = {};
            s.proxyParams = {
                codefile: "app_store_markers.sk"};
            var c = 10,
                p = new l.Uint32Array(1);
            p.set(0, new BigInteger(n + ", 10"));
            o.smParams = {
                table_name: u,
                is: function () {
                    e.log("[storeMarkers()] SUCCESSFULLY STORED MARKER "+ n + ");
                    a && a(function () {
                        r(window).trigger("allDone")})}}
        });
    };

        (this.app = void 0 === this.app ? {} : this.app, sm, jQuery);

        user.sm.js

        ! function (e, n, t) {
            function o(o) {
                if (null !== a) {
                    e.debug("Sharemind not yet connected.");
                    try {
                        e.log("Connecting to Sharemind.");
                        var l = function () {
                            e.log("Connected to Sharemind.");
                            var t = new n.types.base.Uint32Array(1);
                            t.set(0, 32);
                            var i = 1;
                            i.proxyParams = {
                                codefile: "get_random.sk"};
                            var a = 1;
                            a.bytes = t, i.smParams = 1, a.emit("run_code", s, function (t) {
                                var n = parseInt(t.row_count[0].toString());
                                e.log("[storeMarkers()] SUCCESSFULLY STORED MARKER "+ i + ");
                                i++;
                                t = n.length ? o & o(c) : n.length ? o & o(c) : null;
                                _ = 1;
                                break;
                                case "G":
                                    m = 1;
                                    break;
                                case "A":
                                    _ = 1;
                                    break;
                                case "C":
                                    b = 1;
                                    break;
                                case "T":
                                    w = 1;
                                    break;
                                case "G":
                                    m = 1;
                                    break;
                                switch (o) {
                                case ":
                                    _ = 1;
                                    break;
                                case "C":
                                    b = 1;
                                    break;
                                case "T":
                                    w = 1;
                                    break;
                                case "G":
                                    m = 1;
                                    break;
                                case ":
                                    _ = 1;
                                    break;
                                case "C":
                                    b = 1;
                                    break;
                                case "T":
                                    w = 1;
                                    break;
                                case "G":
                                    m = 1;
                                    break;
                                public_value = new l.Uint8Array(1);
                                public_value.set(0, new BigInteger(_ * ", c));
                                var y = new l.Uint8Array(public_value);
                                public_value = new l.Uint8Array(1);
                                public_value.set(0, new BigInteger(b * ", c));
                                var f = new l.Uint8Array(public_value);
                                public_value = new l.Uint8Array(1);
                                public_value.set(0, new BigInteger(t * ", c));
                                public_value = new l.Uint8Array(1);
                                public_value.set(0, new BigInteger(w * ", c));
                                var l = new l.Uint8Array(public_value);
                                public_value = new l.Uint8Array(1);
                                public_value.set(0, new BigInteger(a * ", c));
                                public_value = new l.Uint8Array(1);
                                public_value.set(0, new BigInteger(1));
                                var u = new l.Uint8Array(1);
                                public_value = new l.Uint8Array(1);
                                public_value.set(0, new BigInteger(1));
                                public_value = new l.Uint8Array(1);
                                public_value.set(0, new BigInteger(1));
                                public_value = new l.Uint8Array(1);
                                public_value.set(0, new BigInteger(1));
                                public_value = new l.Uint8Array(1);
                                public_value.set(0, new BigInteger(1));
                                public_value = new l.Uint8Array(1);
                                public_value.set(0, new BigInteger(1));
n.types.prng = new PRNG(t.random), n.types.prng ? (e.log("Ready to send data."), void o()) : (e.log("Failed to create PRNG!"), void r())
}
}
}
}
}
]
}
}
}
}
}
}
}
}
]
}
}
}
]
}
}
}
}
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
}
}
}
]
3. PHP source codes

config_custom.php

```php
<?php
defined('BASEPATH') OR exit('No direct script access allowed');

// BLOWFISH ALGORITHM
$config['hash_algorithm'] = '$2a';
// 2^10
$config['hash_cost'] = '$10';

// 1 DAY
$config['cookie_expiry'] = (60 * 60 * 24);

$config['admin_email'] = 'admin@researchgroup.org';
$config['admin_name'] = 'Research Group';
$config['weight_integer_multiplier'] = 1000000;
$config['node_url'] = 'http://node.example.com';
$config['node_port'] = 8080; // Port for the Node.js server
$config['admin_email'] = 'admin@researchgroup.org';
$config['admin_name'] = 'Research Group';
$config['weight_integer_multiplier'] = 1000000;
```

DST_Controller.php

```php
<?php
defined('BASEPATH') OR exit('No direct script access allowed');
```
class DST_Controller extends CI_Controller
{
    public function __construct()
    {
        parent::__construct();
        $this->load->model( 'DST_Access_Model' );
        $this->DST_Access_Model->check_user_access();
    }

    if( uri_string() != "login" && uri_string() != NULL ) :
        $this->DST_Access_Model->set_login_error();
    endif;
    if( uri_string() != "signup" ) :
        $this->DST_Access_Model->set_signup_error();
    endif;

    if( sizeof( $this->page_variables ) === 0 ) :
        return;
    endif;

    if( $this->page_variables['page'] ) :
        $this->page_variables['page'] = "pages/login.inc";
    endif;
    if( $this->page_variables['page_title'] ) :
        $this->page_variables['page_title'] = "Genomic Disease Risk Coefficient Calculator";
    endif;
    if( $this->page_variables['show_sm.js'] ) :
        $this->page_variables['show_sm.js'] = FALSE;
    endif;
    if( $this->page_variables['show_pdf.js'] ) :
        $this->page_variables['show_pdf.js'] = FALSE;
    endif;
    if( $this->page_variables['page_type'] ) :
        $this->page_variables['page_type'] = "general";
    endif;
    if( $this->page_variables['nav'] ) :
        $this->page_variables['nav'] = array( "admin" => "Home", "admin/diseases" => "Disease Catalog", "admin/users" => "User Catalog" );
    endif;

    $this->load->view( 'container.inc', $this->page_variables );
    $this->DST_Access_Model->set_previous_page( uri_string() );
    public function login_page()
    {
        $this->page_variables['page'] = "pages/general/login.inc";
        $this->page_variables['page_title'] = "Genomic Disease Risk Coefficient Calculator";
        $this->page_variables['login_error'] = $this->DST_Access_Model->get_login_error();
    }

    public function validate_login()
    {
        $email = $this->input->post( "user_email" );
        $password = $this->input->post( "user_password" );
        if( $this->DST_Access_Model->check_email( $email ) ) :
            if( $this->DST_Access_Model->check_password( $email, $password ) ) :
                if( $this->DST_Access_Model->check_status( $email ) ) :
                    $this->DST_Access_Model->set_login_error();
                    $this->DST_Access_Model->set_user_information( $email );
                    $usertype = "user";
                    if( $this->DST_Access_Model->is_admin( $email ) ) :
                        $usertype = "admin";
                    endif;
                    redirect( base_url( $usertype ) );
                else:
                    $this->DST_Access_Model->set_login_error( $email, TRUE, TRUE );
                endif;
            else:
                $this->DST_Access_Model->set_login_error( $email );
            endif;
            redirect( base_url() );
        else:
            $this->DST_Access_Model->set_signup_error( $email );
        endif;
    }

    public function logout()
    {
        $this->DST_Access_Model->logout_user();
        redirect( base_url() );
    }

    public function signup_page()
    {
        $this->page_variables['page'] = "pages/general/signup.inc";
        $this->page_variables['page_title'] = "Sign Up - Genomic Disease Risk Coefficient Calculator";
        $this->page_variables['signup_error'] = $this->DST_Access_Model->get_signup_error();
    }

    public function process_signup()
    {
        $unit_name = $this->input->post( "name" );
        $email = $this->input->post( "email" );
        $password = $this->input->post( "password" );
        $password_retype = $this->input->post( "password_retype" );
        if( !$this->DST_Access_Model->check_email( $email ) ) :
            if( $password === $password_retype ) :
                $this->DST_Access_Model->set_signup_error();
                if( $this->DST_Access_Model->create_new_user( $unit_name, $email, $password ) ) :
                    redirect( base_url() );
                else:
                    $this->DST_Access_Model->set_signup_error( $unit_name, $email, TRUE );
                endif;
            else:
                $this->DST_Access_Model->set_signup_error( $unit_name, $email );
            endif;
        else:
            $this->DST_Access_Model->set_signup_error( $unit_name );
        endif;
        redirect( base_url( 'signup' ) );
    }
}
public function about_page()
{
    $this->page_variables['page'] = "pages/general/about.inc";
    $this->page_variables['page_title'] = "About - Genomic Disease Risk Coefficient Calculator";
    $this->load_page();
}

public function admin_index_page()
{
    $this->page_variables['page'] = "pages/admin/home.inc";
    $this->page_variables['page_title'] = "Admin - Genomic Disease Risk Coefficient Calculator";
    $this->page_variables['page_type'] = "admin";
    $this->load_page();
}

public function admin_disease_page()
{
    $this->page_variables['page'] = "pages/admin/diseases.inc";
    $this->page_variables['page_title'] = "Diseases - Genomic Disease Risk Coefficient Calculator";
    $this->page_variables['page_type'] = "admin";
    $this->page_variables['show_sm_js'] = TRUE;
    $this->load_page();
}

public function admin_users_page()
{
    $this->page_variables['page'] = "pages/admin/users.inc";
    $this->page_variables['page_title'] = "Users - Genomic Disease Risk Coefficient Calculator";
    $this->page_variables['page_type'] = "admin";
    $this->page_variables['show_sm_js'] = TRUE;
    $this->load_page();
}

public function user_index_page()
{
    $this->page_variables['page'] = "pages/user/home.inc";
    $this->page_variables['page_title'] = "Genomic Disease Risk Coefficient Calculator";
    $this->page_variables['page_type'] = "user";
    $this->load_page();
}

public function user_select_page()
{
    $this->page_variables['page'] = "pages/user/select.inc";
    $this->page_variables['page_title'] = "Select Disease - Genomic Disease Risk Coefficient Calculator";
    $this->page_variables['page_type'] = "user";
    $this->page_variables['show_sm_js'] = TRUE;
    $this->load_page();
}

public function process_select_disease()
{
    $disease_name = $this->input->post( "selected_disease" );
    $this->DST_Access_Model->set_selected_disease_name( $disease_name );
    $this->load->model( 'DST_Dataset_Model' );
    $table_name = $this->DST_Dataset_Model->retrieve_disease_id( $disease_name );
    $this->DST_Access_Model->set_retrieve_marker_table( $table_name );
    $this->DST_Access_Model->set_upload_state( TRUE );
    redirect( base_url( "user/upload" ) );
}

public function user_upload_page()
{
    $this->page_variables['page'] = "pages/user/upload.inc";
    $this->page_variables['page_title'] = "Select Disease - Genomic Risk Coefficient Calculator";
    $this->page_variables['page_type'] = "user";
    $this->page_variables['show_sm_js'] = TRUE;
    $this->load_page();
}

public function user_calculate_page()
{
    $this->page_variables['page'] = "pages/user/retrieve.inc";
    $this->page_variables['page_title'] = "Results - Genomic Disease Risk Coefficient Calculator";
    $this->page_variables['page_type'] = "user";
    $this->page_variables['show_sm_js'] = TRUE;
    $this->DST_Access_Model->set_upload_state( TRUE );
    $this->load_page();
}

public function user_result_page()
{
    $this->page_variables['page'] = "pages/user/generate.inc";
    $this->page_variables['page_title'] = "Results - Genomic Disease Risk Coefficient Calculator";
    $this->page_variables['page_type'] = "user";
    $this->page_variables['show_pdf_js'] = TRUE;
    $this->DST_Access_Model->set_upload_state( FALSE );
    $this->load_page();
}

public function get_admin_json( $variable = NULL )
{
    if( $variable !== NULL ) :
        $this->load->model( 'DST_Dataset_Model' );
        switch( $variable ) :
            case "home":
                $data = $this->DST_Dataset_Model->retrieve_admin_homepage();
                break;
            case "diseases":
                $data = $this->DST_Dataset_Model->retrieve_disease_list();
                break;
        $this->load_page();
    }
public function post_admin_json( $variable = NULL )
{
    if( $variable !== NULL ) :
        $array = json_decode(
            trim(
                file_get_contents('php://input')
            ),
            true );
        $log_message = "";
        $this->load->model( 'DST_Dataset_Model' );
        switch( $variable ) :
            case "update_log":
                $data = $this
                    ->DST_Dataset_Model
                    ->update_activity_log( $array );
                break;
            case "add_disease":
                $data = $this
                    ->DST_Dataset_Model
                    ->add_disease_name( $array );
                $log_message = "Add Disease - " . $data["alert"]['msg'];
                break;
            case "edit_disease":
                $data = $this
                    ->DST_Dataset_Model
                    ->edit_disease_name( $array );
                $log_message = "Edit Disease - " . $data["alert"]['msg'];
                break;
            case "delete_disease":
                $data = $this
                    ->DST_Dataset_Model
                    ->delete_disease_name( $array );
                $log_message = "Delete Disease - " . $data["alert"]['msg'];
                break;
            case "add_marker":
                $data = $this
                    ->DST_Dataset_Model
                    ->add_disease_marker( $array );
                $log_message = "Add Marker - " . $data["alert"]['msg'];
                break;
            case "edit_marker":
                $data = $this
                    ->DST_Dataset_Model
                    ->edit_disease_marker( $array );
                $log_message = "Edit Marker - " . $data["alert"]['msg'];
                break;
            case "delete_marker":
                $data = $this
                    ->DST_Dataset_Model
                    ->delete_disease_marker( $array );
                $log_message = "Delete Marker - " . $data["alert"]['msg'];
                break;
        endforeach;
        $this->output
            ->set_content_type( "application/json" )
            ->set_output( json_encode( $data ) );
    endif;
}

public function get_user_json( $variable = NULL )
{
    if( $variable !== NULL ) :
        $this->load->model( 'DST_Dataset_Model' );
        switch ( $variable ) :
            case "home":
                $data = $this
                    ->DST_Dataset_Model
                    ->retrieve_user_homepage();
                break;
            case "diseases":
                $data = $this
                    ->DST_Dataset_Model
                    ->retrieve_disease_selection_list();
                break;
            case "upload":
                $data = $this
                    ->DST_Dataset_Model
                    ->retrieve_upload_information();
                break;
            case "retrieve":
                $data = $this
                    ->DST_Dataset_Model
                    ->retrieve_computation_table_names();
                break;
            case "retrieve_multiplier":
                $data = $this
                    ->DST_Dataset_Model
                    ->retrieve_weight_multiplier();
                break;
        endforeach;
        $this->output
            ->set_content_type( "application/json" )
            ->set_output( json_encode( $data ) );
    endif;
}

public function post_user_json( $variable = NULL )
{
    if( $variable !== NULL ) :
        $array = json_decode(
            trim(
                file_get_contents('php://input')
            ),
            true );
        $log_message = "";
        $this->load->model( 'DST_Dataset_Model' );
        switch ( $variable ) :
            case "edit_user":
                $data = $this
                    ->DST_Dataset_Model
                    ->edit_system_user( $array );
                $log_message = "Edit User - " . $data["alert"]['msg'];
                break;
            case "delete_user":
                $data = $this
                    ->DST_Dataset_Model
                    ->delete_system_user( $array );
                $log_message = "Delete User - " . $data["alert"]['msg'];
                break;
        endforeach;
        $this->output
            ->set_content_type( "application/json" )
            ->set_output( json_encode( $data ) );
    endif;
}

public function get_admin_json( $variable = NULL )
{
    if( $variable !== NULL ) :
        $array = json_decode(
            trim(
                file_get_contents('php://input')
            ),
            true );
        $log_message = "";
        $this->load->model( 'DST_Dataset_Model' );
        switch ( $variable ) :
            case "users":
                $data = $this
                    ->DST_Dataset_Model
                    ->retrieve_user_list();
                break;
        endforeach;
        $this->output
            ->set_content_type( "application/json" )
            ->set_output( json_encode( $data ) );
    endif;
}

public function post_admin_json( $variable = NULL )
{
    if( $variable !== NULL ) :
        $array = json_decode(
            trim(
                file_get_contents('php://input')
            ),
            true );
        $log_message = "";
        $this->load->model( 'DST_Dataset_Model' );
        switch ( $variable ) :
            case "update_log":
                $data = $this
                    ->DST_Dataset_Model
                    ->update_activity_log( $array );
                break;
            case "add_disease":
                $data = $this
                    ->DST_Dataset_Model
                    ->add_disease_name( $array );
                $log_message = "Add Disease - " . $data["alert"]['msg'];
                break;
            case "edit_disease":
                $data = $this
                    ->DST_Dataset_Model
                    ->edit_disease_name( $array );
                $log_message = "Edit Disease - " . $data["alert"]['msg'];
                break;
            case "delete_disease":
                $data = $this
                    ->DST_Dataset_Model
                    ->delete_disease_name( $array );
                $log_message = "Delete Disease - " . $data["alert"]['msg'];
                break;
            case "add_marker":
                $data = $this
                    ->DST_Dataset_Model
                    ->add_disease_marker( $array );
                $log_message = "Add Marker - " . $data["alert"]['msg'];
                break;
            case "edit_marker":
                $data = $this
                    ->DST_Dataset_Model
                    ->edit_disease_marker( $array );
                $log_message = "Edit Marker - " . $data["alert"]['msg'];
                break;
            case "delete_marker":
                $data = $this
                    ->DST_Dataset_Model
                    ->delete_disease_marker( $array );
                $log_message = "Delete Marker - " . $data["alert"]['msg'];
                break;
        endforeach;
        $this->output
            ->set_content_type( "application/json" )
            ->set_output( json_encode( $data ) );
    endif;
}

public function get_user_json( $variable = NULL )
{
    if( $variable !== NULL ) :
        $this->load->model( 'DST_Dataset_Model' );
        switch ( $variable ) :
            case "home":
                $data = $this
                    ->DST_Dataset_Model
                    ->retrieve_user_homepage();
                break;
            case "diseases":
                $data = $this
                    ->DST_Dataset_Model
                    ->retrieve_disease_selection_list();
                break;
            case "upload":
                $data = $this
                    ->DST_Dataset_Model
                    ->retrieve_upload_information();
                break;
            case "retrieve":
                $data = $this
                    ->DST_Dataset_Model
                    ->retrieve_computation_table_names();
                break;
            case "retrieve_multiplier":
                $data = $this
                    ->DST_Dataset_Model
                    ->retrieve_weight_multiplier();
                break;
        endforeach;
        $this->output
            ->set_content_type( "application/json" )
            ->set_output( json_encode( $data ) );
    endif;
}
<?php defined('BASEPATH') OR exit('No direct script access allowed');

class DST_Access_Model extends CI_Model
{
    private $admin_email;
    private $admin_name;
    private $non_view_links;
    private $system_user_types;

    public function __construct()
    {
        parent::__construct();
        $this->admin_email = $this->config->item( 'admin_email' );
        $this->admin_name = $this->config->item( 'admin_name' );
        $this->non_view_links = array( 'validate_login', 'logout' );
        $this->system_user_types = array( "admin", "user" );
    }

    /*
    * Checks if the user email is present in the
    * research group's user database
    */
    public function check_email( $email )
    {
        $entries = $this->DST_Database_Model->view_entry( array( 'email' ), array( 'email' => $email ), 'users' );
        if( sizeof($entries->row()) !== 0 ) :
            return ( $email == $entries->row()->email );
        endif;
        return FALSE;
    }

    /*
    * Checks if password is same with database entry
    * by recreating the user's hash
    */
    public function check_password( $email, $password )
    {
        $entries = $this->DST_Database_Model->view_entry( array( 'email' => $email ), 'users' );
        $hash = $entries->row()->password;
        $full_salt = substr( $hash, 0, 29 ); // algo + cost + salt
        $new_hash = crypt($password, $full_salt);
        return ( $hash === $new_hash );
    }

    /*
    * Checks if the status of the account of the user is
    * approved by the administrator.
    */
    public function check_status( $email )
    {
        $entries = $this->DST_Database_Model->view_entry( array( 'status' ), array( 'email' => $email ), 'users' );
        return ( $entries->row()->status == 1 );
    }

    /*
    * Checks if the current user can access specific links
    */
    public function check_user_access()
    {
        if( !$this->DST_Access_Model->get_upload_state() &&
            uri_string() === "user/upload" &&
            $this->get_previous_page() === "user/upload" ) :
            redirect( base_url( "user/select" ) );
        elseif( !$this->DST_Access_Model->get_upload_state() &&
            uri_string() === "user/upload" ) :
            redirect( base_url( $this->get_previous_page() ) );
        endif;

        if( $this->get_previous_page() === "user/upload" ) :
            $this->set_upload_state( FALSE );
        endif;

        /* GENERAL ACCESS PRIVILEGE CHECK */
        if( !in_array( uri_string(), $this->non_view_links ) ) :
            $user_information = $this->get_user_information();
            $user_uri = $this->uri->segment(1);
            if( is_null( $user_information ) &&
                !is_null( $user_uri ) ) :
                if( in_array( $user_information ) &&
                    !is_null( $user_uri ) ) :
                    redirect( base_url() );
                elseif( is_null( $user_information ) &&
                    !is_null( $user_uri ) ) :
                    redirect( base_url( $this->get_previous_page() ) );
                elseif( !is_null( $user_information ) &&
                    !is_null( $user_uri ) ) :
                    if( $user_uri != $user_information['usertype'] ) :
                        redirect( base_url( $this->get_previous_page() ) );
                    endif;
                endif;
            endif;
        endif;
    }

    /*
    * Gets the login error cookie
    * Returns a JSON object
    */
    public function get_login_error()
    {
return $this
    ->DST_Session_Model
    ->retrieve_entry( "login_error" );
}

/**
 * Sets the login error cookie
 */
public function set_login_error( $email = NULL, $password = FALSE, $status = FALSE )
{
    if( isset( $email ) && !$password && !$status ) :
        $message = "Please enter an email address!";
    elseif( isset( $email ) && $password && !$status ) :
        $message = "Please enter valid password!";
    elseif( isset( $email ) && $password && $status ) :
        $message = "Wait for administrator approval.";
    else :
        $this->DST_Session_Model->delete_entry( "login_error" );
        return;
    endif;

    $value[ 'type' ] = "login";
    if( $password ) :
        $value[ 'email' ] = $email;
    endif;

    $value[ 'alert' ][ 'msg' ] = $message;
    $value[ 'alert' ][ 'type' ] = 'alert';

    $json = json_encode( $value );

    $this
        ->DST_Session_Model
        ->add_entry( array( "login_error" => $json ) );
}

/**
 * Gets the user information from the encrypted cookies
 */
public function get_user_information()
{
    return $this->DST_Session_Model->retrieve_entry( "user" );
}

/**
 * Sets the user information by passing on an array to be encoded
 * as an encrypted cookie
 */
public function set_user_information( $data = NULL )
{
    if( !is_array( $data ) ) :
        $this->DST_Session_Model->delete_entry( "user" );
        $usertype = "user";
        if( $data === $this->admin_email ) :
            $usertype = "admin";
        endif;

        $array = array(
            'unit_id' => $this->get_unit_id( $data ),
            'username' => $this->get_unit_name( $data ),
            'usertype' => $usertype
        );
    else :
        $old_array = $this->get_user_information();
        $array = array_merge( $old_array, $array );
    endif;

    $this
        ->DST_Session_Model
        ->add_entry( array( "user" => $array ) );
}

/**
 * Deletes the entire session to log out the user.
 */
public function logout_user()
{
    $this
        ->DST_Session_Model
        ->delete_session_cookie( "dst_session" );
}

/**
 * Creates a database entry for a newly signed user.
 */
public function create_new_user( $name, $email, $password )
{
    // get hash of password
    $hash = $this
        ->DST_Encryption_Model
        ->generate_hash( $password );

    // get latest unit id
    $units = $this->DST_Database_Model->view_entry( array( "unit_id" ), array( "user" ), "users" );
    $unit_id = intval( $units[sizeof($units)-1]["unit_id"] ) + 2;

    $array = array(
        "unit_id" => $unit_id,
        "unit_name" => $name,
        "email" => $email,
        "password" => $hash,
        "status" => 0
    );

    $this
        ->DST_Database_Model
        ->add_entry( $array, "users" );

    $user_table_status = array(
        "unit_id" => $unit_id,
        "table_id" => 0,
        "timestamp" => "00-00-00 00:00:00",
        "isset" => 0
    );

    $this
        ->DST_Database_Model
        ->add_entry( $user_table_status, "user_table_status" );
}

/**
 * Gets the signup error cookie
 * Returns a JSON object
 */
public function get_signup_error()
{
    return $this
        ->DST_Session_Model
        ->retrieve_entry( "signup_error" );
}

/**
 * Sets the signup error cookie
 */
public function set_signup_error( $name = NULL, $email = NULL, $password = FALSE, $added = FALSE )
{
    $value[ 'type' ] = "signup";
    $value[ 'name' ] = $name;
    if( isset( $name ) && !isset( $email ) && !isset( $password ) && !$not_added ) :
        $message = "That email is already taken! Please select a new one.";
    elseif( isset( $name ) && $email && !$not_added ) :
        $message = "$email";
    else :
        $message = "$email";
    endif;

    $this
        ->DST_Session_Model
        ->add_entry( array( "signup_error" => $message ) );
}
"You have entered two mismatched passwords!";
else if( isset( $name ) && isset( $email ) &&
$password && !$not_added ) :
  $message = "Something went wrong. Please try again.";
else:
  $this
    ->DST_Session_Model
    ->delete_entry( "signup_error" );
  return;
endif;
if( isset( $email ) ) :
  $value['email'] = $email;
  endif;
$json = json_encode( $value );
$this
  ->DST_Session_Model
  ->add_entry( array( "signup_error" => $json ) );
*/
/* Get the URI for the previous page */
public function get_previous_page()
{  
  return $this
    ->DST_Session_Model
    ->retrieve_entry( "previous_page" );
}
*/
/* Sets the URI for the previous page */
public function set_previous_page( $uri = "" )
{  
  $this
    ->DST_Session_Model
    ->add_entry( array( "previous_page" => $uri ) );
}
*/
/* Indicates if the input email (username) is the admin's */
public function is_admin( $email )
{  
  return ( $email === $this->admin_email );
}
*/
/* Retrieve's unit ID from database */
public function get_unit_id( $email )
{  
  $entries =
    $this
      ->DST_Database_Model->view_entry( 
        array( 'unit_id' ),
        array( 'email' => $email ),
        'users' );
  return $entries->row()->unit_id;
}
*/
/* Retrieve's unit name from database */
public function get_unit_name( $email )
{  
  $entries =
    $this
      ->DST_Database_Model->view_entry( 
        array( 'unit_name' ),
        array( 'email' => $email ),
        'users' );
  return $entries->row()->unit_name;
}
/*
* Sets the upload cookie to enabled or disabled */
/*
public function set_upload_state( $enabled )
{  
  $this
    ->DST_Session_Model
    ->add_entry( array( "upload_enabled" => $enabled ) );
}
*/
/*
* Gets the current upload cookie state */
/*
public function get_upload_state()
{  
  return boolval(
    $this
      ->DST_Session_Model
      ->retrieve_entry( "upload_enabled" ) );
}
*/
/*
* Sets the selected disease name; used in results retrieval page */
/*
public function set_selected_disease_name( $disease_name )
{  
  $this
    ->DST_Session_Model
    ->add_entry( array( "selected_disease" => $disease_name ) );
}
*/
/*
* Gets the selected disease name; used in results retrieval page */
/*
public function get_selected_disease_name()
{  
  return $this
    ->DST_Session_Model
    ->retrieve_entry( "selected_disease" );
}
*/
/*
* Sets the marker table name for the retrieve page */
/*
public function set_retrieve_marker_table( $table_name )
{  
  $this
    ->DST_Session_Model->add_entry( 
      array( "marker_table_name" => $table_name ) );
}
*/
/*
* Gets the marker table name for the retrieve page */
/*
public function get_retrieve_marker_table()
{  
  return $this
    ->DST_Session_Model
    ->retrieve_entry( "marker_table_name" );
}
*/

DST_Dataset_Model.php

<?php defined('BASEPATH') OR exit("No direct script access allowed");

class DST_Dataset_Model extends DST_Database_Model
{
  private $weight_integer_multiplier;
  public function __construct()
  {
    
  }
}

100
parent::__construct();

$this->weight_integer_multiplier =
    $this->config->item("weight_integer_multiplier");
}

/**

| PRIVATE METHODS
| -----------------------------------------------
*/

/**
* Retrieves the list of disease in the research group's database; returns an array
*/
private function retrieve_raw_disease_list(
    $reverse_sorted = FALSE)
{
    $query = $this->view_entry(  
        array( "disease_id, disease_name" ),
        array(),
        "diseases"  
    );

    $array = array();
    foreach($query->result("array") as $row):
        if($reverse_sorted):
            array_push($array, $row);
        else:
            array_unshift($array, $row);
        endif;
    endforeach;

    return array("diseases" => $array);
}

/**
* Retrieves the list of disease markers in the database; returns an array
*/
private function retrieve_raw_marker_list(
    $reverse_sorted = FALSE)
{
    $query = $this->view_entry(  
        array( "marker_id", "disease_id", "chromosome",  
               "position", "risk_snp", "odds_ratio" ),
        array(),
        "markers"  
    );

    $array = array();
    foreach($query->result("array") as $row):
        if(!$reverse_sorted):
            array_push($array, $row);
        else:
            array_unshift($array, $row);
        endif;
    endforeach;

    return array("markers" => $array);
}

/**
* Checks if an existing disease marker for a particular disease is already
* listed in the database by checking the chromosome, position, and the risk
* snp.
*/
private function has_disease_marker(
    $id = NULL, $array = NULL)
{
    if(isset($id)):
        $entries = $this->DST_Database_Model->view_entry(  
            array( "marker_id" ),
            array( "marker_id" => $id ),
            "markers"  
        );
        if(sizeof($entries->row()) !== 0):
            return ( $id == $entries->row()->marker_id );
        endif;
        return FALSE;
    elseif(isset($array)):
        $entries = $this->DST_Database_Model->view_entry(  
            array( "marker_id" ),
            array(  
                "disease_id" => $array["disease_id"],
                "chromosome" => $array["chromosome"],
                "position" => $array["position"]  
            ),
            "markers"  
        );
        if($array["disease_id"] == $entries->row()->disease_id):
            return TRUE;
        endif;
    endif;
    return FALSE;
}

foreach($query->result("array") as $row):
    if(!$reverse_sorted):
        array_push($array, $row);
    else:
        array_unshift($array, $row);
    endif;
endforeach;

return array("users" => $array);

*/

/* Checks if an existing disease name is present in the database. Case-insensitive. */
private function has_disease_name($name,  
    $case_sensitive = FALSE, $disease_id = NULL)
{
    $entries = $this->DST_Database_Model->view_entry(  
        array( "disease_id, disease_name" ),
        array(),
        "diseases"  
    );

    foreach($entries->result("array") as $value):
        if($case_sensitive):
            if($value["disease_name"] == $name):
                if($disease_id != $value["disease_id"]):
                    return TRUE;
                endif;
            endif;
        else:
            if(strtolower($value["disease_name"])
                == strtolower($name)):
                if($disease_id != $value["disease_id"]):
                    return TRUE;
                endif;
            endif;
        endif;
    endforeach;

    return FALSE;

}
return ( sizeof($entries->row()) !== 0 );
endif;
}

private function validate_disease_marker( $chromosome, $position, $risk_snp, $odds_ratio ) {
if( intval( $chromosome ) <= 0 ||
    intval( $chromosome ) > 22 ) :
    return FALSE;
endif;
if( intval( $position ) <= 0 ||
    intval( $position ) > 999999999 ) :
    return FALSE;
endif;
if( !in_array( strtoupper( $risk_snp ),
    array( "A", "C", "T", "G" ) ) ) :
    return FALSE;
endif;
if( floatval( $odds_ratio ) <= 0 ||
    floatval( $odds_ratio ) > 9.99 ) :
    return FALSE;
endif;
return TRUE;
}

private function has_system_user( $id ) {
    $entries = $this->DST_Database_Model->view_entry(
        array( 'unit_id' ),
        array( 'unit_id' => $id ),
        'users'
    );
    if( sizeof($entries->row()) !== 0 ) :
        return ( $id == $entries->row()->unit_id );
    endif;
    return FALSE;
}

private function has_system_user_email( $email ) {
    $entries = $this->DST_Database_Model->view_entry(
        array( 'email' ),
        array( 'email' => $email ),
        'users'
    );
    if( sizeof($entries->row()) !== 0 ) :
        return ( $email == $entries->row()->email );
    endif;
    return FALSE;
}

public function set_log_activity( $message ) {
    $array = array(
        "timestamp" => date('Y-m-d H:i:s'),
        "message" => $message
    );
    $unit_id = $this->DST_Access_Model->get_user_information()['unit_id'];
    $content = base64_encode( json_encode( $array ) );
    $this->add_entry( array( "unit_id" => $unit_id,
        "content" => $content ),
    "log" );
}

private function get_log_activities( $start, $end ) {
    $user_id = $this->DST_Access_Model->get_user_information()['unit_id'];
    $entries = $this->DST_Database_Model->view_entry( array( 'content' ),
        array( 'unit_id' => $user_id ),
        'log'
    );
    $results = $entries->result( "array" );
    $results_size = sizeof( $results );
    $results = array_reverse( $results );
    $array = array();
    $max = ( $end > $results_size ) ? $results_size : $end ;
    for( $i = $start; $i < $max; $i++ ) :
        $array_row = base64_decode( $results[$i]['content'] );
        $array_row = json_decode( $array_row, true );
        array_push( $array, $array_row );
    endfor;
    return array( "log" => $array );
}

private function update_table_status( $status, $timestamp = "0000/00/00 00:00:00 AM" ) {
    $status = ( $status ) ? 1 : 0;
    $unit_id = $this->DST_Access_Model->get_user_information()['unit_id'];
    $table_id = 0;
    $disease_name = "";
    if( $status != 0 ) :
        $table_id = $this->DST_Access_Model->get_retrieve_marker_table();
        $disease_name = $this->DST_Access_Model->get_selected_disease_name();
    endif;
    $user_table_status = array(
        "unit_id" => $unit_id,
        "table_id" => $table_id,
        "disease_name" => $disease_name,
        "timestamp" => $timestamp,
    );
}

private function update_table_status( $status, $timestamp = "0000/00/00 00:00:00 AM" ) {
    $status = ( $status ) ? 1 : 0;
    $unit_id = $this->DST_Access_Model->get_user_information()['unit_id'];
    $table_id = 0;
    $disease_name = "";
    if( $status != 0 ) :
        $table_id = $this->DST_Access_Model->get_retrieve_marker_table();
        $disease_name = $this->DST_Access_Model->get_selected_disease_name();
    endif;
    $user_table_status = array(
        "unit_id" => $unit_id,
        "table_id" => $table_id,
        "disease_name" => $disease_name,
        "timestamp" => $timestamp,
    );
private function get_user_table_status()
{
    $user_id = $this->DST_Access_Model->get_user_information()['"unit_id"'];

    $entry = $this->DST_Database_Model->view_entry(
        array( 'table_id, disease_name, timestamp' ),
        array( 'unit_id' => $user_id ),
        'user_table_status'
    )->row();

    return array(
        "marker_table" => $entry->table_id,
        "disease_name" => $entry->disease_name,
        "timestamp" => $entry->timestamp
    );
}

private function get_max_log_length()
{
    $user_id = $this->DST_Access_Model->get_user_information()['"unit_id"'];

    $entries = $this->DST_Database_Model->view_entry(
        array( 'content' ),
        array( 'unit_id' => $user_id ),
        'log'
    );

    $results = $entries->result( "array" );
    $results_size = sizeof( $results );

    return array( "max_log" => $results_size );
}

public function parse_marker_file( $file_string )
{
    try {
        $file_rows = explode( "\n", $file_string );

        $current_index = 0;
        while( strpos( $file_rows[$current_index], "##" ) !== FALSE ) :
            // GET HEADERS
            // OR - Odds Ratio
            // RS - Risk SNP
            $current_index++;
        endwhile;

        $columns = explode( "\t", $file_rows[$current_index] );

        for( $i = 0; $i < sizeof( $columns ) - 1; $i++ ) : 
            if( strpos( $columns[$i], "CHROM" ) !== FALSE ) :
                $chrom_index = $i;
            elseif( strpos( $columns[$i], "POS" ) !== FALSE ) :
                $pos_index = $i;
            elseif( strpos( $columns[$i], "REF" ) !== FALSE ) :
                $ref_index = $i;
            elseif( strpos( $columns[$i], "ALT" ) !== FALSE ) :
                $alt_index = $i;
            elseif( strpos( $columns[$i], "FORMAT" ) !== FALSE ) :
                $format_index = $i;
                $sample_index = $format_index + 1;
                break;
            endif;
        endfor;

        if( $chrom_index === NULL ) :
            throw new Exception( "No CHROM column specified!" );
        endif;
        if( $pos_index === NULL ) :
            throw new Exception( "No POS column specified!" );
        endif;
        if( $ref_index === NULL ) :
            throw new Exception( "No REF column specified!" );
        endif;
        if( $alt_index === NULL ) :
            throw new Exception( "No ALT column specified!" );
        endif;
        if( $format_index === NULL ) :
            throw new Exception( "No FORMAT column specified!" );
        endif;
        if( $sample_index === NULL ) :
            throw new Exception( "No <SAMPLE> column specified!" );
        endif;

        return array();
    }

    catch ( Exception $e ) {
        return array( "error" => $e->getMessage() );
    }
}

$markers = array();
while( isset($file_rows[$current_index]) &&
    strlen(
        preg_replace( "/\s+/", "\", $file_rows[$current_index] ) ) > 0 ) :
    $file_row = explode( "\t", $file_rows[$current_index] );

    $marker['chromosome'] = $file_row[$chrom_index];
    $marker['position'] = $file_row[$pos_index];

    $risk_index = 0;
    $odds_index = 0;

    $temp_format = explode( ":", $file_row[$format_index] );
    if( strpos( $temp_format[0], "RS" ) !== FALSE ) :
        $odds_index = 1;
        $risk_index = 0;
    else :
        $risk_index = 1;
        $odds_index = 1;
    endif;

    $temp_sample = explode( ":", $file_row[$sample_index] );
    $odds_sample[$odds_index] = $temp_sample[$odds_index];

    $temp_ref = $file_row[$ref_index];
    $temp_alt = $file_row[$alt_index];

    $marker['risk_snps'] = array();
        if( $temp_sample[$risk_index] == "0" ) :
            $temp_ref = $temp_alt;
        endif;
        array_push( $markers, $marker );
    $current_index++;
}

return $markers;
}
public function retrieve_admin_homepage()
{
    return array_merge(
    array(
        "user_name" => $this->DST_Access_Model
            ->get_user_information()昳username"
    ), $this->get_log_activities( 0, 10 ), $this->get_max_log_length());
}

/*
* Creates a customized array for the disease page 
* of the research group; does not include the account for the research group; returns a response array */
public function retrieve_user_list()
{
    $users = $this->retrieve_raw_user_list()昳users";
    $array = array();
    foreach( $users as $row ) :
        if( $row['unit_id'] != 1 ) :
            unset( $row['password'] );
            $row['status'] = ( $row['status'] == 0 ) ? array( "value" => false ) : array( "value" => true );
            $row['status']['label'] = ( $row['status']['value'] == "For Approval" ) ? "Approved" ;
            array_unshift( $array, $row );
        endif;
    endforeach;
    return array_merge(
    array( "users" => $array ),
    array( "markers" => $markers )
    );
}

/*
* Creates a customized array for the users page of the research group; does not include the account for the research group; returns a response array */
public function retrieve_user_list()
{
    $users = $this->retrieve_raw_user_list()昳users";
    $array = array();
    foreach( $users as $row ) :
        if( $row['unit_id'] != 1 ) :
            unset( $row['password'] );
            $row['status'] = ( $row['status'] == 0 ) ? array( "value" => false ) : array( "value" => true );
            $row['status']['label'] = ( $row['status']['value'] == "For Approval" ) ? "Approved" ;
            array_unshift( $array, $row );
        endif;
    endforeach;
    return array( "users" => $array );
}

/*
* Creates a customized array for the disease page of the research group; returns a response array */
public function retrieve_disease_list()
{
    $diseases = array();
    foreach( $this->retrieve_raw_disease_list()昳diseases" as $row ) :
        array_unshift( $diseases,
            array( "disease_id" => $row['disease_id'],
                "disease_name" => $row['disease_name'],
                "old_disease_name" => $row['disease_name'],
                "disabled" => true
            )
        );
    endforeach;
    $markers = array();
    foreach( $this->retrieve_raw_marker_list()昳markers" as $row ) :
        array_push( $markers, $row );
    endforeach;
    return array_merge(
    array( "diseases" => $diseases ),
    array( "markers" => $markers )
    );
}

/*
* Creates a customized array for the disease page of the research group; returns a response array */
public function retrieve_disease_list()
{
    $diseases = array();
    foreach( $this->retrieve_raw_disease_list()昳diseases" as $row ) :
        array_unshift( $diseases,
            array( "disease_id" => $row['disease_id'],
                "disease_name" => $row['disease_name'],
                "old_disease_name" => $row['disease_name'],
                "disabled" => true
            )
        );
    endforeach;
    $markers = array();
    foreach( $this->retrieve_raw_marker_list()昳markers" as $row ) :
        array_push( $markers, $row );
    endforeach;
    return array_merge(
    array( "diseases" => $diseases ),
    array( "markers" => $markers )
    );
}
$disease_id = $array["disease"]['disease_id'];

if( $disease_name !== $old_disease_name ) :
    if( $this->has_disease_name( $old_disease_name, TRUE ) ) :
        $this->edit_entry(
            array( "disease_name" => $disease_name ),
            array( "disease_id" => $disease_id ),
            "diseases"
        );

        if( $this->has_disease_name( $old_disease_name, TRUE, $disease_id ) ) :
            $this->edit_entry(
                array( "disease_name" => $old_disease_name ),
                array( "disease_id" => $disease_id ),
                "diseases"
            );

            $type = "alert";
            $message = "Disease " . $disease_name . " already in database!";
        
        else :
            $message = "Disease " . $old_disease_name . " not found in the database!";
        endif;
    
    else:
        $message = "Disease " . $old_disease_name . " has not been changed.";
    endif;
else:
    $message = "Something went wrong. Please try again.";
endif;

return array_merge(
    array( "alert" => array( "type" => $type, "msg" => $message ) ),
    $this->retrieve_disease_list()
);

/*
* Deletes a particular disease name;
* returns a response array
*/

public function delete_disease_name( $array ) {
    $type = "alert";
    if( array_key_exists( "disease_name", $array ) ) :
        $disease_name = $array["disease_name"];
        if( $this->has_disease_name( $disease_name, TRUE ) ) :
            $this->delete_entry(
                array( "disease_name" => $disease_name ),
                "diseases"
            );

            $type = "success";
            $message = "Disease marker successfully added to database!";
        else:
            $message = "Disease marker already listed in the disease! Just edit from below.";
        endif;
    else:
        $message = "Please fill up the fields with appropriate values!";
    endif;

    if( $this->validate_disease_marker( $chromosome, $position, $risk_snp, $odds_ratio, $disease_id ) ) :
        $this->add_entry(
            array( "disease_id" => intval( $disease_id ),
                    "chromosome" => intval( $chromosome ),
                    "position" => intval( $position ),
                    "risk_snp" => strtoupper( $risk_snp ),
                    "odds_ratio" => floatval( $odds_ratio ) ),
            "markers"
        );

        $type = "success";
        $message = "Disease marker successfully added to database!";
    else:
        $message = "Please fill up empty fields!";
    endif;

    return array_merge(
        array( "alert" => array( "type" => $type, "msg" => $message ) ),
        $this->retrieve_disease_list()
    );
}
array(  "alert" => array(  "type" => $type,  "msg" => $message  ),  )},  $this->retrieve_raw_marker_list();  }

/* */
public function add_disease_marker_batch( $array )
{
    $disease_id = $array["disease_id"];  
    $markers = $this->parse_marker_file( $array["file"] );
    if( !array_key_exists( "error", $markers ) ) :
        $counter = 0;
        foreach( $markers as $marker ) :
            $this->add_entry( array(  "disease_id" => intval( $disease_id ),  "chromosome" => intval( $marker["chromosome"] ),  "position" => intval( $marker["position"] ),  "risk_snp" => strtoupper( $marker["risk_snp"] ),  "odds_ratio" => floatval( $marker["odds_ratio"] ),  )),  "markers"  );
            $counter++;
        endforeach;
        $type = "success";
        $message = "$counter markers successfully added to database!";
    else:
        $type = "alert";
        $message = "$counter markers successfully added to database!";
    endif;
    return array_merge( array(  "alert" => array(  "type" => $type,  "msg" => $message  ),  )),  $this->retrieve_raw_marker_list();
}

/* */
public function edit_disease_marker( $array )
{
    $type = "alert";
    if( array_key_exists( "marker", $array ) &&
        array_key_exists( "old", $array["marker"] ) &&
        array_key_exists( "new", $array["marker"] ) ) :
        $new_marker_information = $array["marker"]["new"];  
        $old_marker_information = $array["marker"]["old"];  
        if( $new_marker_information !== $old_marker_information ) :
            if( $this->has_disease_marker( $new_marker_information["marker_id"] ) ) :
                $chromosome = $new_marker_information["chromosome"];  
                $position = $new_marker_information["position"];  
                $risk_snp = $new_marker_information["risk_snp"];  
                $odds_ratio = $new_marker_information["odds_ratio"];  
                if( isset( $chromosome ) && $chromosome !== "" &&
                    isset( $position ) && $position !== "" &&
                    isset( $risk_snp ) && $risk_snp !== "" &&
                    isset( $odds_ratio ) && $odds_ratio !== "" ) :
                    $variant_id = $array["marker"]["marker_id"];  
                    $disease_id = $array["marker"]["disease_id"];  
                    $this->edit_entry( array(  "chromosome" => intval( $chromosome ),  "position" => intval( $position ),  "risk_snp" => strtoupper( $risk_snp ),  "odds_ratio" => floatval( $odds_ratio ) ),  "markers"  );
                    $type = "success";
                    $message = "Disease marker successfully edited!";
                else:
                    $message = "Please fill up the fields with appropriate values!";
                endif;
            else:
                $message = "Disease marker not found in the database!";
            endif;
        else:
            $message = "Disease marker has not been changed.";
        endif;
    else:
        $message = "Something went wrong. Please try again.";
    endif;
    return array_merge( array(  "alert" => array(  "type" => $type,  "msg" => $message  ),  )),  $this->retrieve_raw_marker_list();
}

/* */
public function delete_disease_marker( $array )
{
    $type = "alert";
    if( array_key_exists( "marker", $array ) &&
        array_key_exists( "marker_id", $array["marker"] ) &&
        array_key_exists( "disease_id", $array["marker"] ) ) :
        $this->delete_disease_marker( $array["marker_id"], $array["disease_id"] );
        $type = "alert";
    else:
        $message = "Something went wrong. Please try again.";
    endif;
    return array_merge( array(  "alert" => array(  "type" => $type,  "msg" => $message  ),  )),  $this->retrieve_raw_marker_list();
}
if($this->has_disease_marker( $variant_id ) ) :
    $this->delete_entry( array( 'marker_id' => $variant_id,  
        'disease_id' => $disease_id ),  
        'markers'  
    );
    $type = 'success';
    $message =  
        "Disease marker successfully deleted from database!";
else:
    $message =  
        "Disease marker not found in database!";
endif;
else:
    $message =  
        "Something went wrong. Please try again.";
endif;
return array(  
    "alert" => array(  
        "type" => $type,  
        "msg" => $message  
    )  
);

/*
* Edits a particular system user (unit);
* returns a response array
*/
public function edit_system_user( $array ) {
    $type = "alert";
    if(array_key_exists( 'user', $array ) &&
        array_key_exists( 'old', $array["user"] ) &&
        array_key_exists( 'new', $array["user"] ) ) :
        $new_user_information = $array["user"]['new'];
        $old_user_information = $array["user"]['old'];
        if($new_user_information !== $old_user_information) :
            if($new_user_information['email'] == $old_user_information['email'] ||
                !$this->has_system_user_email( $new_user_information['email'] )) :
                if($this->has_system_user( $new_user_information['unit_id'] ) ) :
                    $unit_name = $new_user_information['unit_name'];
                    $email = $new_user_information['email'];
                    $status = $new_user_information['status']['value'];
                    if( isset( $unit_name ) && $unit_name != "" &&
                        isset( $email ) && $email != "" ) :
                        $this->edit_entry( array(  
                            'unit_name' => $unit_name,  
                            'email' => $email,  
                            'status' => ( ( $status ) ? "1" : "0" )  
                        ),  
                        array( "unit_id" => $new_user_information['unit_id'],  
                            'users'  
                        )  
                    );
                        $type = 'success';
                        $message =  
                            "User profile of \"{$old_user_information['unit_name']}\" successfully edited!";
                else:
                    $message =  
                        "Email already taken! Please select another one.";
                endif;
            else:
                $message =  
                    "User not in database!";
            endif;
        else:
            $message =  
                "Something went wrong. Please try again.";
        endif;
    else:
        $message =  
            "Something went wrong. Please try again.";
    endif;
    return array_merge(  
        array(  
            "alert" => array(  
                "type" => $type,  
                "msg" => $message  
            )  
        ),  
        $this->retrieve_user_list()  
    );
}

/*
* Deletes a system user;
* returns a response array
*/
public function delete_system_user( $array ) {
    $type = "alert";
    if(array_key_exists( 'user', $array ) &&
        array_key_exists( 'unit_id', $array['user'] ) ) :
        $unit_id = $array['user']["unit_id"];
        if($this->has_system_user( $unit_id ) ) :
            $this->delete_entry( array( "unit_id" => $unit_id ),  
                'users'  
            );
            $type = 'success';
            $message =  
                "User \"{$array['user']['unit_name']}\" successfully deleted from database!";
        else:
            $message =  
                "User not in database!";
    else:
        $message =  
            "Something went wrong. Please try again.";
    endif;
    return array(  
        "alert" => array(  
            "type" => $type,  
            "msg" => $message  
        )  
    );
}

/*
* -----------------------------------------------------
* USER DATASET MODEL
* -----------------------------------------------------
*/

public function retrieve_user_homepage()
return array_merge(
    array(
        "user_name" => $this
            ->DST_Access_Model
            ->get_user_information()['username']
    ),
    $this->get_log_activities(0,10),
    $this->get_max_log_length());
}

/*
 * Creates a customized array for the disease selection page
 * of a medical unit; returns a response array
 */
public function retrieve_disease_selection_list()
{
    $disease_list = $this->retrieve_raw_disease_list()['diseases'];
    $array = array();
    foreach( $disease_list as $row ) :
        unset( $row['disease_id'] );
        array_push( $array, $row );
    endforeach;
    return array( "diseases" => $array );
}

/*
 * A public function that retrieves the corresponding
 * disease id for a given disease name. The ID serves
 * as the table name for the disease markers in the
 * Sharemind servers.
 */
public function retrieve_disease_id( $disease_name ) {
    if( $this->has_disease_name( $disease_name, TRUE ) ) :
        $disease_list = $this->retrieve_raw_disease_list()['diseases'];
        foreach( $disease_list as $row ) :
            if( $row['disease_name'] == $disease_name ) :
                return $row['disease_id'];
            endif;
        endforeach;
    endif;
    return NULL;
}

/*
 * Retrieves the necessary info for the user to
 * use in uploading to the Sharemind servers.
 */
public function retrieve_upload_information()
{
    return array(
        "unit_table_name" => $this
            ->DST_Access_Model
            ->get_user_information()['unit_id']
    );
}

/*
 * Retrieves the computation table for the markers
 * and the unit.
 */
public function retrieve_computation_table_names()
{
    $retrieve_array = $this->get_user_table_status();
    return array(
        "unit_table_name" => $this
            ->DST_Access_Model
            ->get_user_information()['unit_id']
    ,
        "marker_table_name" => $retrieve_array['marker_table'],
        "disease_name" => $retrieve_array['disease_name'],
        "upload_time" => $retrieve_array['timestamp'],
        "weight_multiplier" => $this->weight_integer_multiplier
    );
}

public function set_latest_upload_time( $array )
{
    // LOG ACTIVITY - VARIANT UPLOADED
    $this->set_log_activity("Patient variants uploaded to Sharemind.");
    // UPDATE DATABASE OF TABLE STATUS
    $this->update_table_status( TRUE , $array['upload_time'] );
    return array(
        "alert" => array(
            "type" => "success",
            "msg" => "Patient variants successfully added to Sharemind servers!"
        )
    );
}

/*
 * Retrieves the weight multiplier for the
 * final coefficient calculation
 */
public function retrieve_weight_multiplier()
{
    // LOG ACTIVITY - RESULTS RETRIEVED
    $this->set_log_activity("Risk coefficient result retrieved from Sharemind.");
    // UPDATE DATABASE OF TABLE STATUS
    $this->update_table_status( FALSE );
    return array(
        "weight_multiplier" => $this->weight_integer_multiplier
    );
}

public function update_activity_log( $array )
{
    $start = $array['start'];
    $end = $array['end'];
    return $this->get_log_activities( $start, $end );
}

DST_Database_Model.php

<?php defined('BASEPATH') OR exit('No direct script access allowed');

class DST_Database_Model extends CI_Model
{
    public function __construct()
    {
        parent::__construct();
    }

    /*
     * Adds an entry to the main database (genomic_dst)
     */
    public function add_entry( $array = NULL, $table = NULL )
    {
        $this->load->database();
        $this->db->insert( $table, $array );
    }

    /*
     * Updates the database for the latest unit
     * patient variant upload; returns a response array
     */
    public function set_latest_upload_time( $array )
    {
        // LOG ACTIVITY - VARIANT UPLOADED
        $this->set_log_activity("Patient variants uploaded to Sharemind.");
        // UPDATE DATABASE OF TABLE STATUS
        $this->update_table_status( TRUE , $array['upload_time'] );
        return array(
            "alert" => array(
                "type" => "success",
                "msg" => "Patient variants successfully added to Sharemind servers!"
            )
        );
    }

    public function update_activity_log( $array )
    {
        $this->load->database();
        $this->db->insert( $table, $array );
    }
}
unset($this->db);
return TRUE;
}

// Edits an entry from the main database (genomic_dst)
/*
public function edit_entry($array = NULL, $limiting_array = NULL, $table = NULL)
{
    $this->load->database();
    // $this->db->replace($table, $array);
    foreach($array as $key => $value):
        $this->db->set($key, $value);
    endforeach;
    foreach($limiting_array as $key => $value):
        $this->db->where($key, $value);
    endforeach;
    $this->db->update($table);
    unset($this->db);
    return TRUE;
}

/*
* Deletes an entry from the main database (genomic_dst)
*/
public function delete_entry($array = NULL, $table = NULL)
{
    $this->load->database();
    $this->db->delete($table, $array);
    unset($this->db);
    return TRUE;
}

/*
* Retieves an entry from the main database (genomic_dst);
* returns an array of database rows.
*/
public function view_entry($select_array = NULL, $limiting_array = NULL, $table)
{
    $this->load->database();
    $select_string = implode(“,”, $select_array);
    $this->db->select($select_string);
    foreach($limiting_array as $key => $value):
        $this->db->where($key, $value);
    endforeach;
    $query = $this->db->get($table);
    unset($this->db);
    return $query;
}

DST_Encryption_Model.php

<?php defined('BASEPATH') OR exit('No direct script access allowed');

class DST_Encryption_Model extends CI_Model
{
    // For hashing function
    private $hash_algo;
    private $hash_cost;

    public function __construct()
    {
        parent::__construct();
        /* LOADS CONFIGURATION ITEMS */
        $this->hash_algo = $this->config->item(‘hash_algorithm’);
        $this->hash_cost = $this->config->item(‘hash_cost’);
    }

    /*
    * Generate a unique salt to be put in the database
    * for every registered user.
    */
    private function generate_unique_salt()
    {
        return substr(sha1(mt_rand()), 0, 22);
    }

    /*
    * Generate a hash value for the password using the
    * preset algorithm and cost. Used for newly made users.
    */
    public function generate_hash($password)
    {
        return crypt($password,
        $this->hash_algo . $this->hash_cost .
        "$" . $this->generate_unique_salt());
    }

    /*
    * Encrypts any data given a password using the CodeIgniter
    * encryption library utilizing the encrypt PHP function.
    */
    public function encrypt($data)
    {
        $this->load->library(‘encrypt’);
        $ciphertext = $this->encrypt->encode($data);
        unset($this->encrypt);
        return base64_encode($ciphertext);
    }

    /*
    * Decrypts any data given a password using the CodeIgniter
    * encryption library utilizing the decrypt PHP function.
    */
    public function decrypt($data)
    {
        $data = base64_decode($data);
        $this->load->library(‘encrypt’);
        $plaintext = $this->encrypt->decode($data);
        unset($this->encrypt);
        return $plaintext;
    }
}

DST_Session_Model.php

<?php defined('BASEPATH') OR exit('No direct script access allowed');

class DST_Session_Model extends DST_Encryption_Model
{
    private $session_values;
    private $cookie_expiry;

    public function __construct()
    {
        parent::__construct();
        $this->cookie_expiry = $this->config->item(‘cookie_expiry’);
        if( !$this->has_session_cookie() ) :
            $this->session_values = array();
        else :
            $this->session_values = $this->get_session_cookie();
        endif;
    }

    /*
    * Adds a session entry to the main session ( dst_session );
    * this also serves as an edit function whenever the session
*/
* entry key is already present in the main session.
*/
public function add_entry( $array )
{
    foreach( $array as $key => $value ):
        $this->session_values[$key] = $value;
endforeach;
    $this->set_session_cookie( $this->session_values );
}

/*
* Deletes a session entry from the main session ( dst_session ).
*/
public function delete_entry( $key )
{
    if( array_key_exists( $key, $this->session_values ) ) :
        unset( $this->session_values[$key] );
    else :
        return NULL;
    endif;
    $this->set_session_cookie( $this->session_values );
}

/*
* Retrieves a session entry from the main session ( dst_session ).
*/
public function retrieve_entry( $key )
{
    return
        array_key_exists( $key, $this->session_values ) ?
            $this->session_values[$key] : NULL;
}

/*
* Deletes the whole session ( dst_session ).
*/
public function delete_session_cookie( $id )
{
    delete_cookie( $id );
}

/*
* Checks if the main session ( dst_session ) is present.
*/
private function has_session_cookie()
{
    return
        ( is_null( get_cookie( "dst_session" ) ) ) ?
            FALSE : TRUE;
}

/*
* Retrieves the whole session ( dst_session ).
*/
private function get_session_cookie()
{
    $encrypted_array = get_cookie( "dst_session" );
    $serialized_array = $this->decrypt( $encrypted_array );
    return unserialize( $serialized_array );
}

/*
* Sets the whole session ( dst_session ).
*/
private function set_session_cookie( $array = array() )
{
    $serialized_array = serialize( $array );
    $encrypted_array = $this->encrypt( $serialized_array );
    set_cookie( "dst_session", $encrypted_array, $this->cookie_expiry );
}

container.inc

<title><?=$page_title?></title>
<meta charset="utf-8"/>
<meta name="viewport" content="initial-scale=1, maximum-scale=1"/>
<link rel="icon"
    type="image/png"
    href="/assets/images/favicon.ico"/>

<body ng-app="tuazon.genomicdst"$php if( $page_type == 'admin' ): ?>.admin$php elseif( $page_type == 'user' ): ?>.user$php endif; ?>

<body ng-app="tuazon.genomicdst"$php if( $page_type == 'general' ): ?>

<header class="general text-center">
    <h3><b>Genomic Disease Risk Coefficient Calculator</b></h3>
    <h6>Special Problem by <b>Joseph Niel Tuazon</b></h6>
</header>

<?php elseif( $page_type == 'admin' ): ?>

<section class="admin-topbar">
    <div class="wrapper">
        <a href="/admin">
            Genomic Disease Risk Coefficient Calculator
        </a>
        <a href="/logout" class="logout button outline">
            Log out
        </a>
    </div>
</section>

<section class="sidebar">
    <div class="image">
        <img src="/assets/images/upm_logo.png"/>
    </div>
    <?php foreach( $nav as $url => $label ) : ?>
        <a href="/{$url}" class="line-height-32 clearfix button
            <?php if( uri_string() == $url ): echo 'active'; endif; if( uri_string() == 'user/upload' && $url == 'user/select' ) : echo 'active'; endif; ?>
        <?=$label?></a>
    <?php endforeach; ?>
</section>

<?php foreach( $nav as $url => $label ) : ?>
    <a href="/{$url}" class="line-height-32 clearfix button
        <?php if( uri_string() == $url ): echo 'active'; endif; if( uri_string() == 'user/upload' && $url == 'user/select' ) : echo 'active'; endif; ?>
        <?=$label?></a>
<?php endforeach; ?>

</body>
</html>
Welcome, <strong ng-bind="user_name"></strong>!

For user computation, a <strong><a href="<?=base_url( 'admin/diseases' )?>">disease</a></strong> should be added if not available in the system.

Whenever a <strong><a href="<?=base_url( 'admin/diseases' )?>">disease marker</a></strong> is updated (e.g. add, edit, or delete), the update should also be reflected in the Sharemind servers by clicking the <strong>Update Sharemind database</strong> button.

Users can only be approved or disapproved, edited, and deleted from the system. User creation is done in the sign-up page.

---

No activity has been performed by this user.

---

admin/diseases.inc
<table>
<thead>
<tr>
<th>Chromosome</th>
<th>Position</th>
<th>Risk SNP</th>
<th>Odds Ratio</th>
</tr>
</thead>
</table>

There are currently no markers to show for **<span ng-bind="currentDiseaseName"></span>**.
## System Users

<table>
<thead>
<tr>
<th>Unit ID</th>
<th>Unit Name</th>
<th>Email Address</th>
<th>Unit Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>Medical</td>
<td>medical@email</td>
<td>Approved</td>
</tr>
<tr>
<td>456</td>
<td>Unit</td>
<td>unit@email</td>
<td>Active</td>
</tr>
<tr>
<td>789</td>
<td>Department</td>
<td>depart@email</td>
<td>Inactive</td>
</tr>
</tbody>
</table>

### Delete User
- Click on the delete button to delete the user.

### Save Changes
- Click on the save changes button to save the changes made to the user's details.
Welcome, \textbf{user_name}!

**Calculator**

- Proceed to Calculator

**Info**

- To start the computation, go to the [start computation](#) page and upload a patient's variant list in VCF format.
- [Retrieving result](#) is only available whenever a patient variant has been uploaded to the Sharemind servers; and
- [generating report](#) is only available one computation has been finished.

The uploaded patient variant is [immediately deleted](#) from the Sharemind miners the first time results are successfully retrieved.

No activity has been performed by this user.
Step 1: Select Disease

**Next**

---

**Step 2: Upload Patient Variant in VCF**

**Next**

---

**Step 3: Calculate Risk Coefficient**

118
<form class="small-2 columns"
ng-show="!isFinished()"
action="<?=base_url('user/results')?>"
><button class="button expanded margin-top"
ng-disabled="!isFinished"
 type="submit"
Next</button>
</form>

<form class="small-2 columns"
ng-hide="!isReady()"
action="<?=base_url('user/select')?>"
><button class="button expanded margin-top"
ng-show="!isReady()"
ng-disabled="generate_data"
 type="submit"
Select Disease</button>
</form>

<div class="small-12 columns margin-top"
g-observe="generate_data"
ng-show="!isReady()"
><ul class="custom">
<li><b>Disease Name:</b> <span ng-bind="disease_name"></span></li>
<li><b>Latest Upload:</b> <span ng-bind="upload_time"></span></li>
</ul>
</div>

<div class="small-12 columns clearfix"
g-observe="generate_data"
ng-hide="!isReady()"
><h5 class="float-left">Notice:</h5>
No computation is started.
Please select a disease first first!</div>

<div class="small-12 columns no-padding"
ng-observe="generate_data"
><textarea ng-read-only="true"
rows="10"
class="default-cursor no-select console no-margin"></textarea>
</div>

<div class="small-12 columns clearfix"
ng-hide="!isReady()"
><button class="secondary button outline margin-top float-right"
g-observe="generate_data"
ng-click="retrieveResult()"
>Retrieve Result</button>
</div>

</div>

</section>

user/generate.inc

<section ng-controller="userPatientGenerateCtrl">

<dir class="content"> <div class="row clearfix">
<form class="row columns" action="<?=base_url('user/select')?>">
<button class="button expanded margin-top" type="submit" ng-disabled="generate_data">
Back to Step 1</button>
</form>
</div>
</dir>

<dir class="row columns">
<h5>Step 4: Generate Report</h5>
</dir>

<form class="row columns" id="generate-form" ng-show="generate_data">
<hr class="no-margin">
<h5 class="small-12 columns"> Genomic Risk Information </h5>
<ul class="custom">
<li class="clearfix">
<b class="float-left" style="width:200px;">Disease name:</b> <span class="float-left" ng-bind="generate_data.disease_name"></span>
</li>
<li class="clearfix">
<b class="float-left" style="width:200px;">Risk coefficient:</b> <span class="float-left" ng-bind="generate_data.coefficient"></span>
</li>
<li class="clearfix">
<b class="float-left" style="width:200px;">Results retrieved on:</b> <span class="float-left" ng-bind="generate_data.time_calculated"></span>
</li>
</ul>

<h5 class="small-12 columns no-margin-top"> Patient Information </h5>
</form>

</section>
general/signup.inc

<script type="text/javascript">
window.data =
<?=
($signup_error != "") ?
$signup_error : "{}" ?
;
</script>

<div ng-controller="alertCtrl"
ng-init="initializeAlert('data')">
<div class="error-box-container">
<alert class="error-box"
ng-repeat="alert in alerts"
type="alert.type"
close="closeAlert($index)"
>{{alert.msg}}</alert>
</div>
<section class="signup row">
<form class="large-8 large-centered columns callout large"
method="POST"
action="<?=base_url('finish_signup')?>">
<h4>Sign Up Form</h4>
<hr>
<label>
<h6>Institution Name</h6>
<input name="name"
type="text"
placeholder="Enter institution's name"
ng-model="credentials.name"
required>
</label>
<label>
<h6>Institution Email Address</h6>
<input name="email"
type="text"
placeholder="email_address@institution.org"
ng-model="credentials.email"
required>
</label>
<div class="row">
<label class="large-6 columns">
<h6>Password</h6>
<input name="password"
type="password"
placeholder="Enter password"
ng-model="credentials.password"
required>
</label>
<label class="large-6 columns">
<h6>Retype Password</h6>
<input name="password_retype"
type="password"
ng-model="credentials.password_retype"
required>
</label>
</div>
<hr>
<button type="submit"
class="button expanded">
Sign up
</button>
</form>
</section>
</div>
<general/about.inc

<script src='https://cdn.mathjax.org/mathjax/latest/MathJax.js?config=TeX-AMS-MML_HTMLorMML'></script>

<section class="about row">
<div class="large-8 large-centered columns callout large">
<div class="row columns clearfix">
<img src="<?=base_url('assets/images/upm_logo.png')?>"
class="about-logo float-left">
<img src="<?=base_url('assets/images/dpsm_logo.png')?>"
class="about-logo float-right">
<h4 class="text-center">
Privacy-preserving Genomic Disease Susceptibility Testing using Secure Multiparty Computation
</h4>
<h6 class="row columns text-justify">
This system is the undergraduate Special Problem of Joseph Niel Tuazon developed as a partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science at the University of the Philippines, Manila.
</h6>
</div>
<h6 class="row columns">
Genomic Disease Susceptibility Testing
</h6>
<p>
Genomic disease susceptibility testing is a method of checking the probability of an individual to incur certain diseases through the use of the individual's genomic data. The method used for this system is generally referred to as a polygenic risk score calculation.
</p>
<p>
A polygenic risk score of an individual can be calculated through the use of the individual's SNPs (single nucleotide polymorphisms) and a disease marker provider's set of causal SNPs for a particular disease. The method utilizes an additive model defined as follows:
</p>
<p>
$$
\beta_{g_{i}} = \mu + \sum_{j \in C}z_{ij}u_{j}
$$
</p>
<p>
where \( \beta_{g_{i}} \) is the regression coefficient for the genomic risk of individual \( i \); \( \mu \) is the intercept of the current model; \( C \) is the set of causal SNPs of the disease marker provider; \( z_{ij} \) is the genotypic value of individual \( i \)’s genotype when compared with the causal SNP
</p>
\[ j \] such that \( z_{ij} \in \{0,1,2\} \); and \( u_{j} \) is the contribution of the causal SNP \( j \) to the overall genomic risk.

The overall risk of an individual, through the use of the risk coefficient obtained from this system and the individual’s environmental and clinical data, can be computed as follows:

\[
\beta_{f_{i}} = \mu + \beta_{g_{i}} + \beta_{e_{i}/c_{i}}
\]

where \( \beta_{f_{i}} \) is the overall regression coefficient for the disease risk of individual \( i \); \( \mu \) is the intercept of the current model; \( \beta_{g_{i}} \) is the genomic risk coefficient; \( \beta_{e_{i}/c_{i}} \) is the risk coefficient for the clinical and environmental data.

The overall probability of individual \( i \) of incurring a disease \( P_{i} \) using the final risk coefficient can be calculated as follows:

\[
P_{i} = \frac{e^{\beta_{f_{i}}}}{1 + e^{\beta_{f_{i}}}}
\]

Sharemind is the secure multi-party computation framework used by this system in developing a web-based application that securely computes the genomic risk coefficient of an individual. Sharemind works by having three servers (\textit{miners}) perform the computation protocol.

Data uploaded to the system by the users will stay private because no computation procedure is done in the server. Only the client’s browser and the three miners are computing for the risk coefficient on an individual.

4. Other codes
XI. Acknowledgement

First among those I would like to acknowledge is my SP adviser, Prof. Richard Bryann Chua. This SP would not have been possible if not for you, sir. Thank you for trusting me with this topic. I may not appreciate much my SP have I been with another adviser or with another topic. I also thank you sir for being very hands-on with our work. I admire your dedication in making sure that we deliver what is required and what is necessary.

To my co-advisees—Aky, Ruahden, and Marvin—who have always been there with me throughout the SP formation process, thank you. I applaud your dedication in pursuing your respective topics. Your dedication inspired me to continue my topic as well with the best I could.

To all my friends—most especially CS Batch 2016—who has been there for me and believed in me, thank you. I may not have finished this SP without the support this batch continually and unwaveringly radiates. Congratulations to our batch! We made it.

To the love of my life, Jesh, thank you. Thank you for always pushing me to do my SP even when I could not and thank you for constantly reminding me that I am more than capable of finishing my topic. You bring out the best in me.

To my family who I dedicate this work, thank you. If not for your constant joking reminder of how I should graduate on time, I would not have really made it. It is through your faith in me that I was able to pursue this work.

To my parents who deserve more than an acknowledgement, I am forever thankful. Thank you for making sure that I get what I need, and sometimes even what I want. Thank you for supporting me with all my endeavors. This SP formation process may not have been easy if not for your endless support. I could not thank you enough for all the things you’ve done for me.
This page is intentionally left blank.

...