# Software Design Description for Paternity testing using genetics

Youssif Assem, Mohamed Moataz, Kareem Ehab, Ahmed Gamal, Mohamed Akram
Supervised by: Dr.Ashraf Abdelraouf and Eng. Ahmed Hazem

March 9, 2022

Table 1: Document version history

| Version | Date | Reason for Change |
|---|---|---|
| 1.1 | 25-Jan-2022 | SDD first version's description are defined. |
| 1.2 | 2-Feb-2022 | Added Sequence Diagram. |
| 1.3 | 5-Feb-2022 | Requirement Matrix updated. |

**GitHub:** https://github.com/YoussifAssem/Genetics

# Contents

**Abstract**

The most important aspect in determining a family is a person's DNA. One of the most significant phenomena that have spread around the world and illustrate everything about a person. For street children, orphaned children, persons who are make criminals, or men who dispute paternity to a child, we want to develop a method that analyses the whole genome or alleles. The system will utilize numerous algorithms to compare it to all of the people in our data set, checking for resemblance with each one, and then telling them what family this child may be related to. In addition, the algorithm can determine whether or not the child is connected to the father. Furthermore, we intend to expand our system in the future to help the police with their criminal investigations and help the court with the proving fatherhood case.

# 1 Introduction

## 1.1 Purpose

The aim of this software design description (SDD) is to provide full description of the model system application (Paternity testing using genetics).The usage of genes analytics is required due to the nature of the subject, Therefore It will outline the system's objectives and characteristics, as well as the system's interfaces. It will also cover what the system will preform,the constraints under which It will operate, and, most importantly, how the system will react to external stimulation, where our application system module will target children or orphans that are lost in the streets,The software system will only be available to government entities.

## 1.2 Scope

Our goal is to develop an application that takes the whole genome or alleles of a child, father and illustrates if the child is related to this father or not. In addition, the system can take the whole genome or allele of a street child or orphanage child in the orphanage and compare it with our database and return the family this child may be related to.

## 1.3 Overview

We aim to create a system that allows some of the competent authorities to upload the data of whole-genome or allele for street children or children in the orphanage. Our system will check the similarity between the child it and each family in our database and the system will return to the family that this child may be related to. We aim to build our model depending on the Egyptian genome project because we aim to solve this problem in our countries. In addition, we aim to add another part in our system that proves the parentage of a child to a father depending on rs numbers or the whole genome.

## 1.4 Intended audience

**Governmental entity:** The module system will be used by government agents to determine which family that the child in the street or the child in the orphanage may be related to.

**Users:** The module system will be used by users in General via mobile application so that they could know the results of their case by entering the number of case and national ID.

**These audiences will use this documentation as a guideline in case a bug arose or there was a need for any modification on the system.**

## 1.5 Reference Material

v1.0 of "Software Requirement Specification Document for Paternity testing using genetics" v1.4 of Test Plan for Paternity testing using genetics

## 1.6 Definitions and Acronyms

| **Term** | **Definition** |
|---|---|
| **Allele** | A gene can have two or more variants. For each gene, an individual receives two alleles, one from each parent. |
| **Governmental entity** | signifies any government, including its agencies, bureaus, boards , commissions, courts, departments, officials, political subdivisions, tribunals, and other instrumentality, whether federal, state, or municipal, domestic or international. |
| **Whole genome sequencing** | Full genome sequencing, also known as complete genome sequencing or entire genome sequencing, is the technique of determining the entirety, or almost the entirety, of an organism's DNA sequence in a single moment. |
| **rs number** | is a unique label used by researchers and databases to identify a specific SNP ("rs" followed by a number) (Single Nucleotide Polymorphism). It is the naming convention for most SNPs and stands for Reference SNP cluster ID. |
| **SNP** | is a unique label used by researchers and databases to identify a specific SNP ("rs" followed by a number) (Single Nucleotide Polymorphism). It is the naming convention for most SNPs and stands for Reference SNP cluster ID. |
| **Short Tandem Repeat (STRs)** | are DNA segments that resemble accordions and contain core repeat units of two to seven nucleotides in length that are tandemly repeated from a half dozen to several dozen times. |

# 2 System Overview

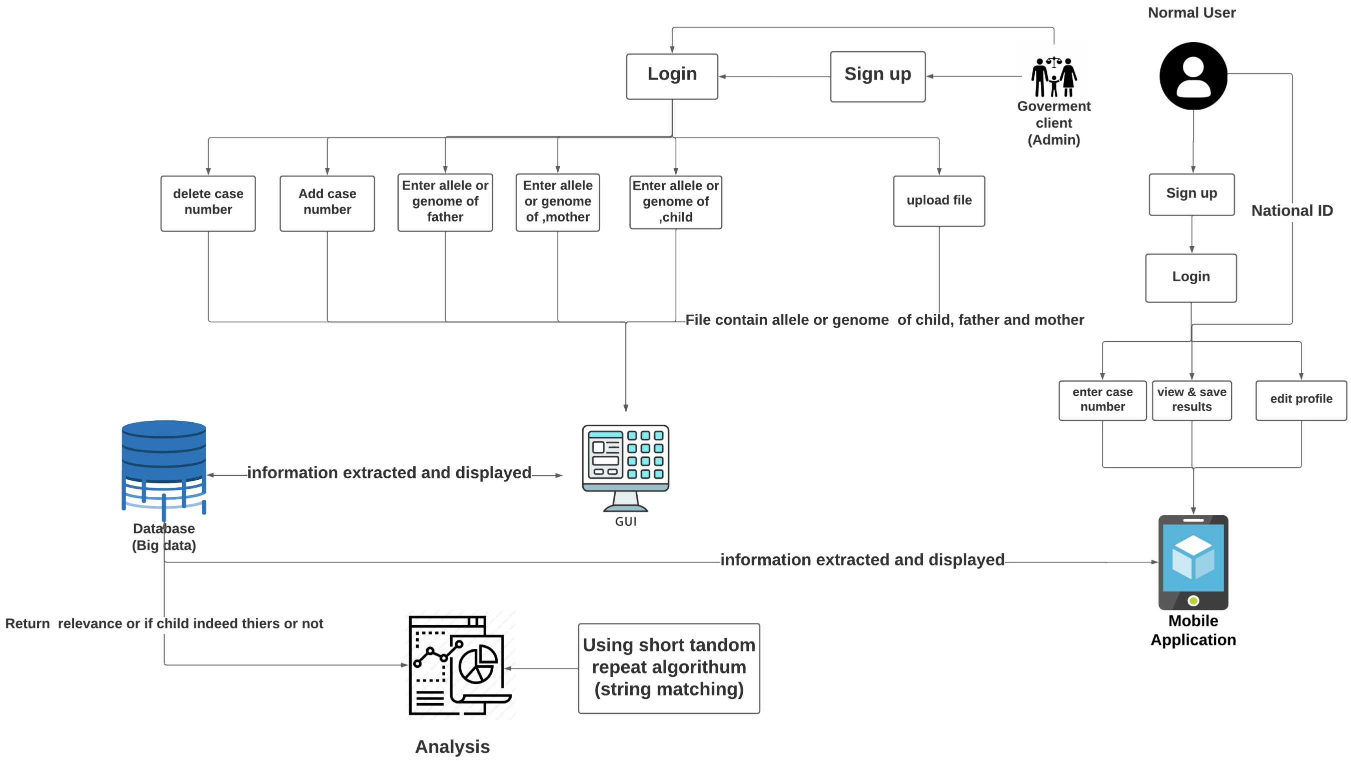

Normal User
Login
Sign up
Goverment client (Admin)
delete case number
Add case number
Enter allele or genome of father
Enter allele or genome of ,mother
Enter allele or genome of ,child
upload file
Sign up
National ID
Login
File contain allele or genome of child, father and mother
enter case number
view & save results
edit profile
Database (Big data)
information extracted and displayed
GUI
information extracted and displayed
Mobile Application
Return relevance or if child indeed thiers or not
Using short tandom repeat algorithum (string matching)
Analysis

Figure (1) : System overview diagram

In this project, it is possible to determine the paternity test results in different test cases of paternity. The system will be a mobile application that allows some of the competent authorities to upload the whole genome sequence. The system will check the similarity between it and each child in our database based on the case presented (different cases for paternity testing) and the system will return to the family the that this child may be related to the family and if there is some relevance using kinship analysis. The project will be divided to two parts. The first part will take the rs numbers as input from the family and analyze it to get the results of the test at the end. The second part the data will be represented as whole genome, then it will be analyzed for conducting the different test cases of paternity and kinship analysis.

## 2.1 System Scope

The system will be able to display paternity testing results by using data from both the father , mother and child's DNA analysis (input), where the model compares these rs numbers and determine whether child is linked to father/mother or not(output) in addition we will extend paternity test AKA proof of parentage through Whole genome sequencing(WGS) which is a next generation sequencing application that determines the entire DNA sequence all at once. Several test cases will be provided in the system such as the standard trio case (father, mother, child) and other paternity test cases.Our system shall:

- display the results of the paternity test based on the given test case
- State the different rs numbers that led to a false paternity
- Retrieve a report giving the final outcome for the intended audience

## 2.2 System objectives

The main goal in our project is to have a system that can output the results of a family's paternity test according to their rs numbers analysis or whole genome sequence, then output whether or not the alleged father is the actual parent or not and show the likelihood (for example if the father is the true parent then the likelihood should be ex: 99.999%). The second part would be kinship analysis and finding out if the child has a kinship to this family or not according to the number of rs numbers that can be found similar and finding their likelihood if the child is in fact related or not.

- Build a software system for paternity testing and kinship analysis
- A software program that reads a file containing a child's DNA (whole genome or alleles through specific rs numbers)
- The system examines the data and concludes which family is relevant to the child.
- Conduct multiple test cases for paternity and kinship and test it
- System should have high efficiency and performance
- System should output the rs numbers that contributed to false paternity
- System should output the rs numbers that contributed to true or false relevancy

## 2.3 System Timeline

Team members will be assigned numbers to be used in time plan table

1. Youssif Assem
2. Mohamed Moataz
3. Kareem Ehab
4. Ahmed Gamal
5. Mohamed Akram

Table 2: Project time Plan

| Id | Task | Start Date | Number of Days | Team member |
|---|---|---|---|---|
| 1 | Searching for datasets | 5/10/2021 | 32 | 1, 2, 3, 4 |
| 2 | Searching for Resources | 10/8/2021 | 145 | 1, 2, 3, 4 |
| 3 | Getting first dataset and analyzing it | 9/11/2021 | 5 | 1, 2, 3, 4 |
| 4 | Implementation of dataset and crossover section | 19/11/2021 | 3 | 1 |
| 5 | Begin Writing SRS | 16/12/2021 | 19 | 1,2,3,4 |
| 6 | Implementing a new dataset of family of five | 30/12/2021 | 5 | 1, 2 |
| 7 | Searching for a dataset for whole exome section | 2/1/2022 | 1 | 3, 4 |
| 8 | Implementation of paternity test section | 2/1/2022 | 2 | 1, 2 |
| 9 | Finding and interviewing the client | 8/1/2022 | 20 | 1, 2, 3, 4 |
| 10 | rewriting SRS document | 8/1/2022 | 2 | 1, 2, 3, 4 |
| 11 | Implementing login and register GUI code | 17/2/2022 | 3 | 3 |
| 12 | building encryption for system | 18/2/2022 | 2 | 2 |
| 13 | fixed code bugs and create gui | 25/1/2022 | 1 | 1 |
| 14 | SDD document implementation | 29/1/2022 | 30 | 2,3,1 |
| 15 | SDD code implementation | 29/1/2022 | 1 | 1 |
| 16 | writing test cases | 3/2/2022 | 3 | 3 |
| 17 | creating mobile app and Implementing user section | 25/2/2022 | 4 | 1 |
| 18 | Write code of FASTA files and short tandem repeats | 15/2/2022 | 12 | 5 |

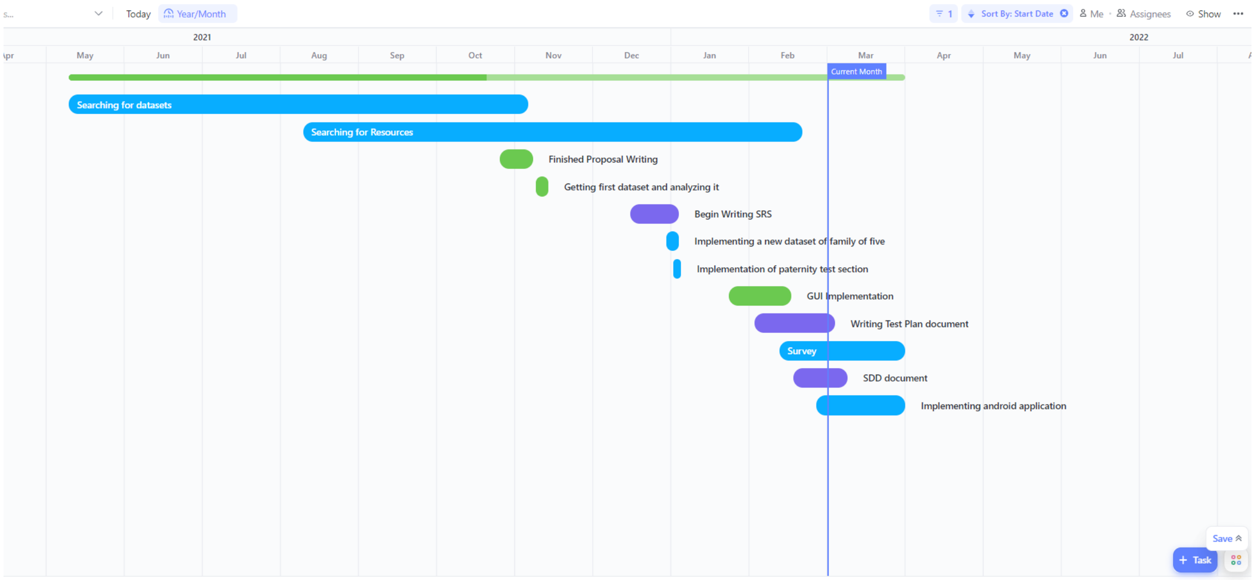


Figure (2) : GANTT chart

# 3 Design viewpoints

## 3.1 Context viewpoint

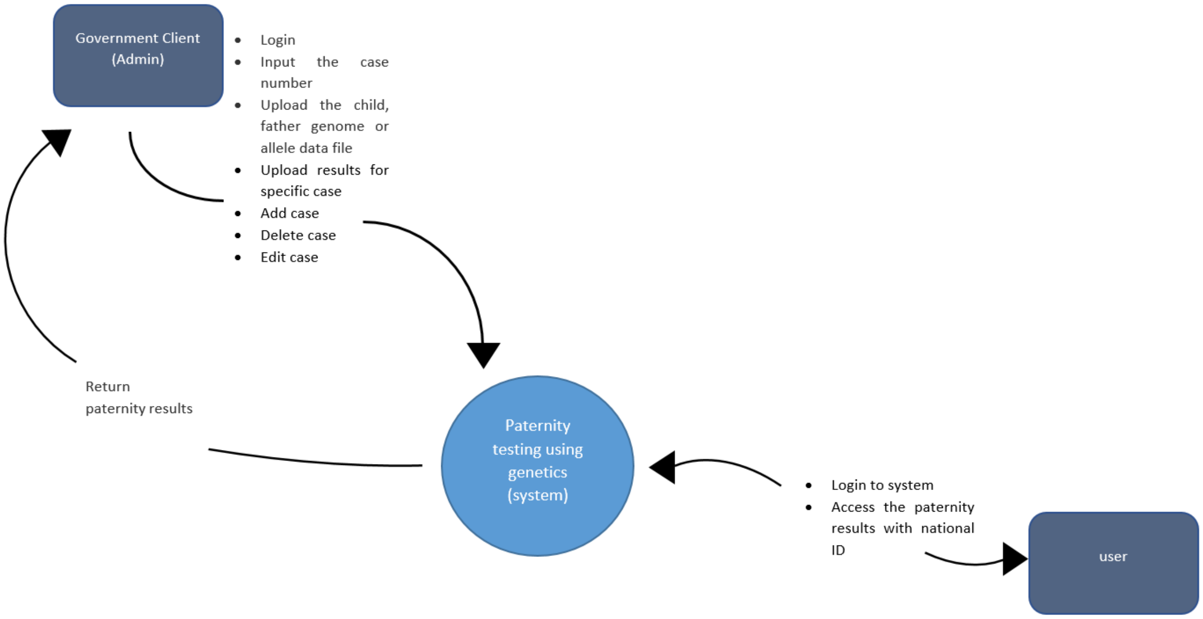

Government Client (Admin)
Login
Input the case number
Upload the child, father genome or allele data file
Upload results for specific case
Add case
Delete case
Edit case
Return paternity results
Paternity testing using genetics (system)
Login to system
Access the paternity results with national ID
user

Figure (3) : Context Diagram for the paternity testing

## 3.2 Use Cases

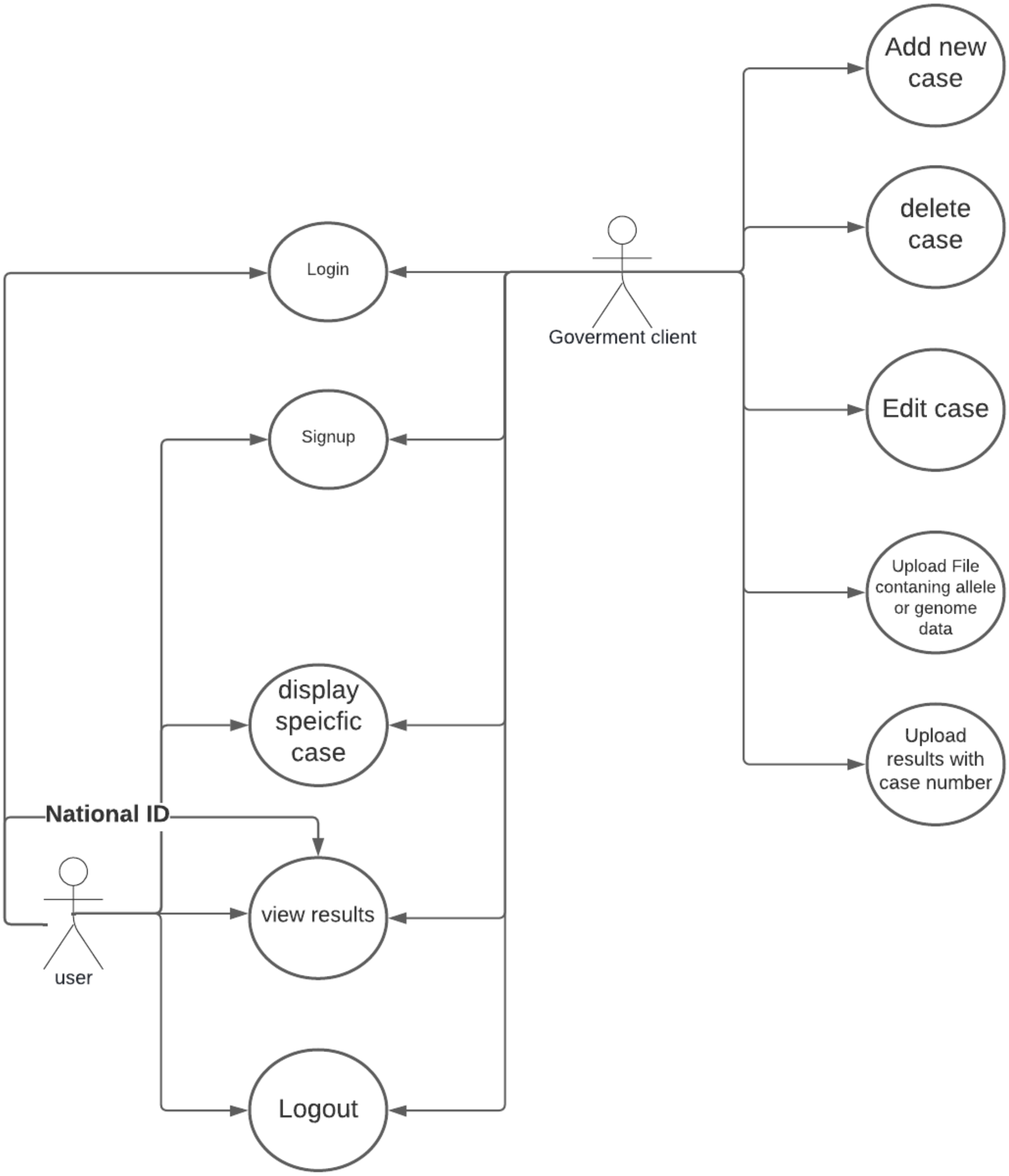


Figure (4) : use case diagram

## 3.3 Composition viewpoint

### 3.3.1 Design Rationale

Model-View-Controller (MVC) is a design pattern that divides an application into three basic logical components: Model, View, and Controller and we aim to use it in our system.

- a view is the component of the program that reflects how the results of the paternity test and report will be viewed.
  The data given to the system should be uploaded by the user for it to be processed and output to the user.

- The application's controller is the component that manages user interaction. The controller interprets the user's mouse and keyboard inputs, causing the model and display to adapt appropriately, much like when the user interacts with the testing process and uploads its data.

- Data and logic are contained in the model section. It represents data exchanged between controller components or any other business logic linked to it. Admin can obtain user information from the database using a Controller object, for example. Before sending data back to the database or displaying it, it manipulates it.

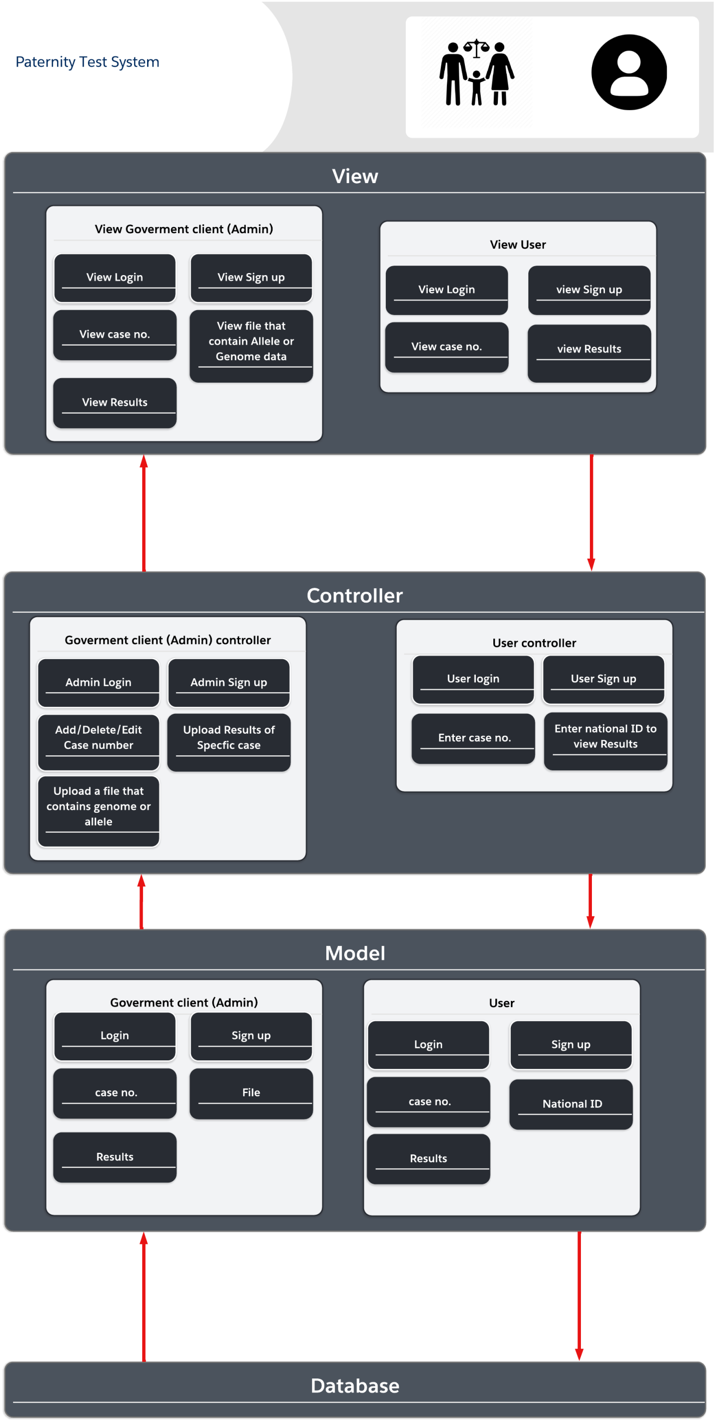


Figure (5) : architecture diagram

## 3.4 Logical viewpoint

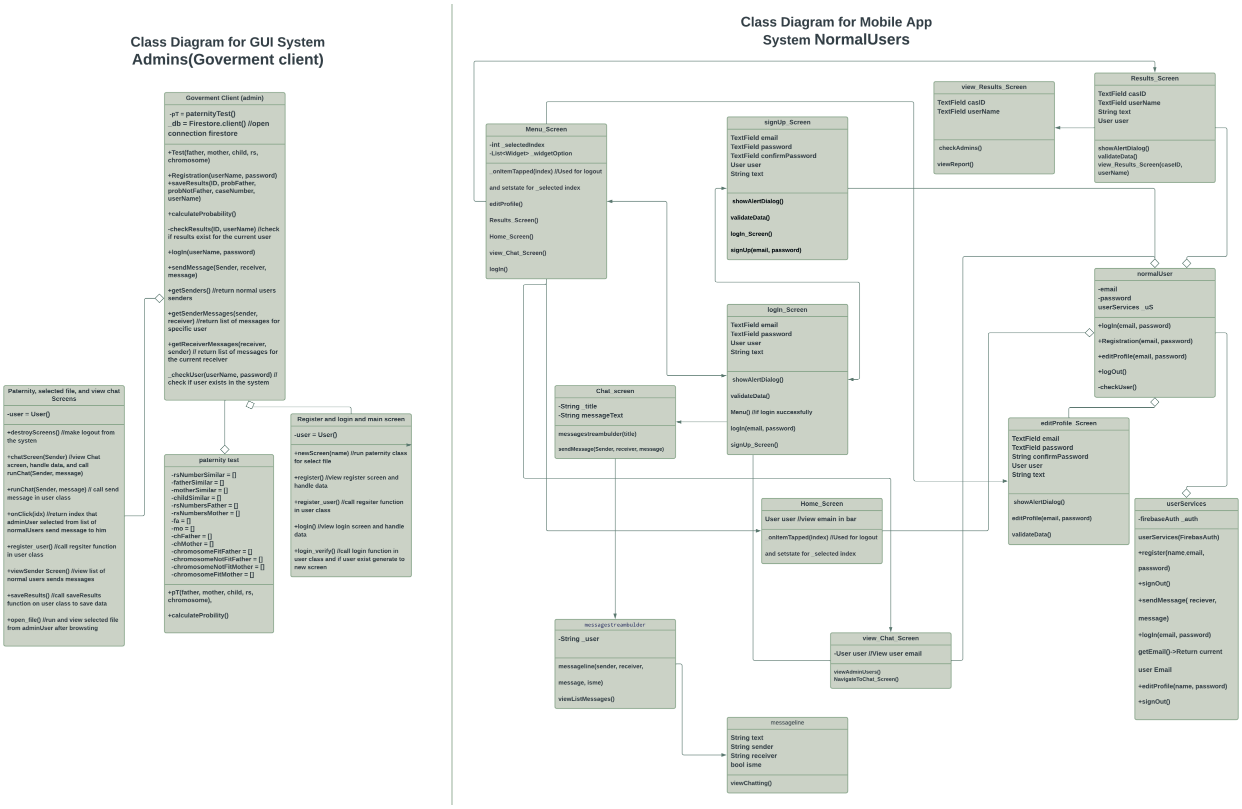

Class Diagram for GUI System
Admins(Goverment client)
Goverment Client (admin)
-pT = paternityTest()
_db = Firestore.client() //open connection firestore
+Test(father, mother, child, rs, chromosome)
+Registration(userName, password)
+saveResults(ID, probFather, probNotFather, caseNumber, userName)
+calculateProbability()
-checkResults(ID, userName) //check if results exist for the current user
+logIn(userName, password)
+sendMessage(Sender, receiver, message)
+getSenders() //return normal users senders
+getSenderMessages(sender, receiver) //return list of messages for specific user
+getReceiverMessages(receiver, sender) // return list of messages for the current receiver
_checkUser(userName, password) // check if user exists in the system
Paternity, selected file, and view chat Screens
-user = User()
+destroyScreens() //make logout from the systen
+chatScreen(Sender) //view Chat screen, handle data, and call runChat(Sender, message)
+runChat(Sender, message) // call send message in user class
+onClick(idx) //return index that adminUser selected from list of normalUsers send message to him
+register_user() //call regsiter function in user class
+viewSender Screen() //view list of normal users sends messages
+saveResults() //call saveResults function on user class to save data
+open_file() //run and view selected file from adminUser after browsting
paternity test
-rsNumberSimilar = []
-fatherSimilar = []
-motherSimilar = []
-childSimilar = []
-rsNumbersFather = []
-rsNumbersMother = []
-fa = []
-mo = []
-chFather = []
-chMother = []
-chromosomeFitFather = []
-chromosomeNotFitFather = []
-chromosomeNotFitMother = []
-chromosomeFitMother = []
+pT(father, mother, child, rs, chromosome),
+calculateProbility()
Register and login and main screen
-user = User()
+newScreen(name) //run paternity class for select file
+register() //view register screen and handle data
+register_user() //call regsiter function in user class
+login() //view login screen and handle data
+login_verify() //call login function in user class and if user exist generate to new screen
Class Diagram for Mobile App System NormalUsers
Menu_Screen
-int _selectedIndex
-List<Widget> _widgetOption
_onItemTapped(index) //Used for logout
and setstate for _selected index
editProfile()
Results_Screen()
Home_Screen()
view_Chat_Screen()
logIn()
signUp_Screen
TextField email
TextField password
TextField confirmPassword
User user
String text
showAlertDialog()
validateData()
logIn_Screen()
signUp(email, password)
view_Results_Screen
TextField casID
TextField userName
checkAdmins()
viewReport()
Results_Screen
TextField casID
TextField userName
String text
User user
showAlertDialog()
validateData()
view_Results_Screen(caseID, userName)
normalUser
-email
-password
userServices _uS
+logIn(email, password)
+Registration(email, password)
+editProfile(email, password)
+logOut()
-checkUser()
logIn_Screen
TextField email
TextField password
User user
String text
showAlertDialog()
validateData()
Menu() //if login successfully
logIn(email, password)
signUp_Screen()
Chat_screen
-String _title
-String messageText
messagestreambulder(title)
sendMessage(Sender, receiver, message)
editProfile_Screen
TextField email
TextField password
String confirmPassword
User user
String text
showAlertDialog()
editProfile(email, password)
validateData()
Home_Screen
User user //view emain in bar
_onItemTapped(index) //Used for logout
and setstate for _selected index
userServices
-firebaseAuth _auth
userServices(FirebasAuth)
+register(name,email, password)
+signOut()
+sendMessage( reciever, message)
+logIn(email, password)
getEmail()->Return current user Email
+editProfile(name, password)
+signOut()
messagestreambulder
-String _user
messageline(sender, receiver, message, isme)
viewListMessages()
view_Chat_Screen
-User user //View user email
viewAdminUsers()
NavigateToChat_Screen()
messageline
String text
String sender
String receiver
bool isme
viewChatting()

Figure (6) : class diagram

Table 3: adminUser

| | |
|---|---|
| **Abstract or Concrete:** | Absract |
| **Purpose** | Add a new user into the system and the firebase and let user add data of paternityTest. |
| **Collaborations** | the class will interact with government entity |
| **Attributes** | pT = object() |
| **Operations** | Test(father, mother, child, rs, chromosome)<br>Registration(userName, password)<br>saveResults(ID, probFather, probNotFather)<br>checkResults(ID)<br>logIn(userName, password) |

Table 4: normalUser

| **Abstract or Concrete:** | Absract |
|---|---|
| **Purpose** | Add a new user into the system and the firebase and let user extract result of the paternityTest. |
| **Collaborations** | the class will interact with normal users |
| **Attributes** | email<br>password |
| **Operations** | viewResults(nationalID)<br>Registration(email, password)<br>logIn(email, password)<br>editProfile(email, password)<br>sendMessage(adminUser) |

Table 5: paternity test

| **Abstract or Concrete:** | concrete |
|---|---|
| **Purpose** | check paternity testing between father,mother and child |
| **Collaborations** | the class will interact with government entity |
| **Attributes** | rsNumberSimilar = []<br>fatherSimilar = []<br>motherSimilar = []<br>childSimilar = []<br>rsNumbersFather = []<br>rsNumbersMother = []<br>fa = []<br>mo = []<br>chFather = []<br>chMother = []<br>chromosomeFitFather = []<br>chromosomeNotFitFather = []<br>chromosomeNotFitMother = []<br>chromosomeFitMother = [] |
| **Operations** | paternityTest(father, mother, child, rs, chromosome)<br>calculateProbility() |

Table 6: Chatting

| **Abstract or Concrete:** | Abstract |
|---|---|
| **Purpose** | Establish connection between admin and user |
| **Collaborations** | This class will be used to send and recieve messages between two classes (Admin and User) |
| **Attributes** | sender<br>recipient<br>messageContent |
| **Operations** | viewMessage(ID)<br>sendMessage(ID, message) |

Table 7: User Services

| **Abstract or Concrete:** | Abstract |
|---|---|
| **Purpose** | Establish connection system and database |
| **Collaborations** | This class will be used for operations related to the database for out system |
| **Attributes** | firebaseAuth |
| **Operations** | register(name, email, password)<br>signout()<br>sendMessage(receiver, message)<br>login(email, password)<br>getEmail()<br>editProfile(name, password) |

Table 8: View chat

| **Abstract or Concrete:** | Abstract |
|---|---|
| **Purpose** | Establish connection system and database |
| **Collaborations** | This class will be used to view chat and messages |
| **Attributes** | -User user //View user email |
| **Operations** | viewAdminUsers()<br>NavigateToChatScreen() |

Table 9: messageline

| | |
|---|---|
| **Abstract or Concrete:** | Concrete |
| **Purpose** | Establish connection system and database |
| **Collaborations** | This class will be used to view chat and messages |
| **Attributes** | String text<br>String sender<br>String receiver<br>bool isme |
| **Operations** | viewChatting() |

## 3.5 Patterns use viewpoint

The model view controller architecture was chosen for the system design because it has numerous advantages over alternative architectures, such as the ability to construct the application quickly and easily. It simplifies development for several developers because the system is divided into discrete pieces, allowing them to collaborate and work on the same project. It also makes it easier to update the application because only the areas that require upgrading are accessed.

## 3.6 Algorithm viewpoint

There are 8 cases according to dataset for alleles that father have 2 alleles or one allele and same for mother and child so, we have

```
’’’
father mother child alleles (two for each person inherited from both parents)
2 2 2 like -> A/A A/A A/A
2 2 1 like -> A/A A/G A
2 1 2 like -> A/A G A/G
2 1 1 like -> A/A G A
1 2 2 like -> A G/G A/G
1 2 1 like -> G A/T G
1 1 2 like -> T G T/G
1 1 1 like -> T G T ”’
define lists
rsNumberSimilar = []
fatherSimilar = []
motherSimilar = []
childSimilar = []
rsNumbersFather = []
rsNumbersMother = []
fa = []
mo = []
chFather = []
chMother = []
chromosomeFitFather = []
chromosomeNotFitFather = []
chromosomeNotFitMother = []
chromosomeFitMother = []
make paternity test function
def pT(father, mother, child, rs, chromosome)
define list for chromosomes
ch = [’1’,’2’,’3’,’4’,’5’,’6’,’7’,’8’,’9’,’10’,’11’,’12’,’13’,’14’,’15’,’16’,’17’,’18’,’19’,’ 20’,’21’,’22’,’X’,’Y’,’MT’]
loop for each element in the list
for f,m,c,r,chro in father,mother,child,rs,chromsome
case 2 2 2
if chro in ch:
```

```
check that case 2 occured
if len(c) == 2 and len(f) == 2 and len(m) == 2
check that child have an allele taked from father
if f[0] == c[0] or f[0] == c[1] or f[1] == c[0] or f[1] == c[1]
push data on lists
rsNumberSimilar.append(r)
fatherSimilar.append(f)
childSimilar.append(c)
chromosomeFitFather.append(chro)
check that child have an allele taked from mother
if m[0] == c[0] or m[0] == c[1] or m[1] == c[0] or m[1] == c[1]
motherSimilar.append(m)
chromosomeFitMother.append(chro)
check if the child have an allele not taken from father
if f[0] != c[0] and f[0] != c[1] and f[1] != c[0] and f[1] != c[1]
rsNumbersFather.append(r)
fa.append(f)
chFather.append(c)
chromosomeNotFitFather.append(chro)
check if the child have an allele not taken from mother
if m[0] != c[0] and m[0] != c[1] and m[1] != c[0] and m[1] != c[1]:
rsNumbersMother.append(r)
mo.append(m)
chMother.append(c)
chromosomeNotFitMother.append(chro)
```

we repeat the same things with each case and push the data on lists like the last example.
we have a function calculate probability that calculate if the child is related to this father or not.
Every get function in the calculate probability return the list.

```
def calculateProbability()
sumSimilar = len(getRsNumberSimilar())
total = (len(getRsNumberSimilar())+ len(getRsNumberFather()))
sumNotSimilar = len(getRsNumberFather())
rule = sumSimilar / total
ruleNotSimilar = sumNotSimilar / total
return rule * 100, ruleNotSimilar * 100
```

### 3.6.1 Short tandem repeat

Autosomal short tandem repeat (STR) markers are used in forensic DNA profiling to determine the identify of missing people[1] [2], validate familial ties, and link people of interest to crime scenes. The algorithm of STR is based on the of count repeats for some nucleotide on DNA ex: ATCATCATCATC should be repeated a lot of times on some specific locations on the DNA (locations are different for each country). The STR counts in the child should be the same as father and mother. The concept that genetic markers utilised in forensic applications are not predictive of phenotype is commonly held.[3]

### 3.6.2 Mendel’s Laws of Inheritance

The process by which a child inherits genetic information from a parent is known as inheritance. Every family with its rs numbers should have alleles. These alleles in child should be inherited from their father and mother ex: if the child has C/G in a specific rs number then the father should have a C or G in the same rs number and mother. This simply indicates that individuals of the same family share comparable qualities as a result of heredity [4].

## 3.7 Interaction viewpoint

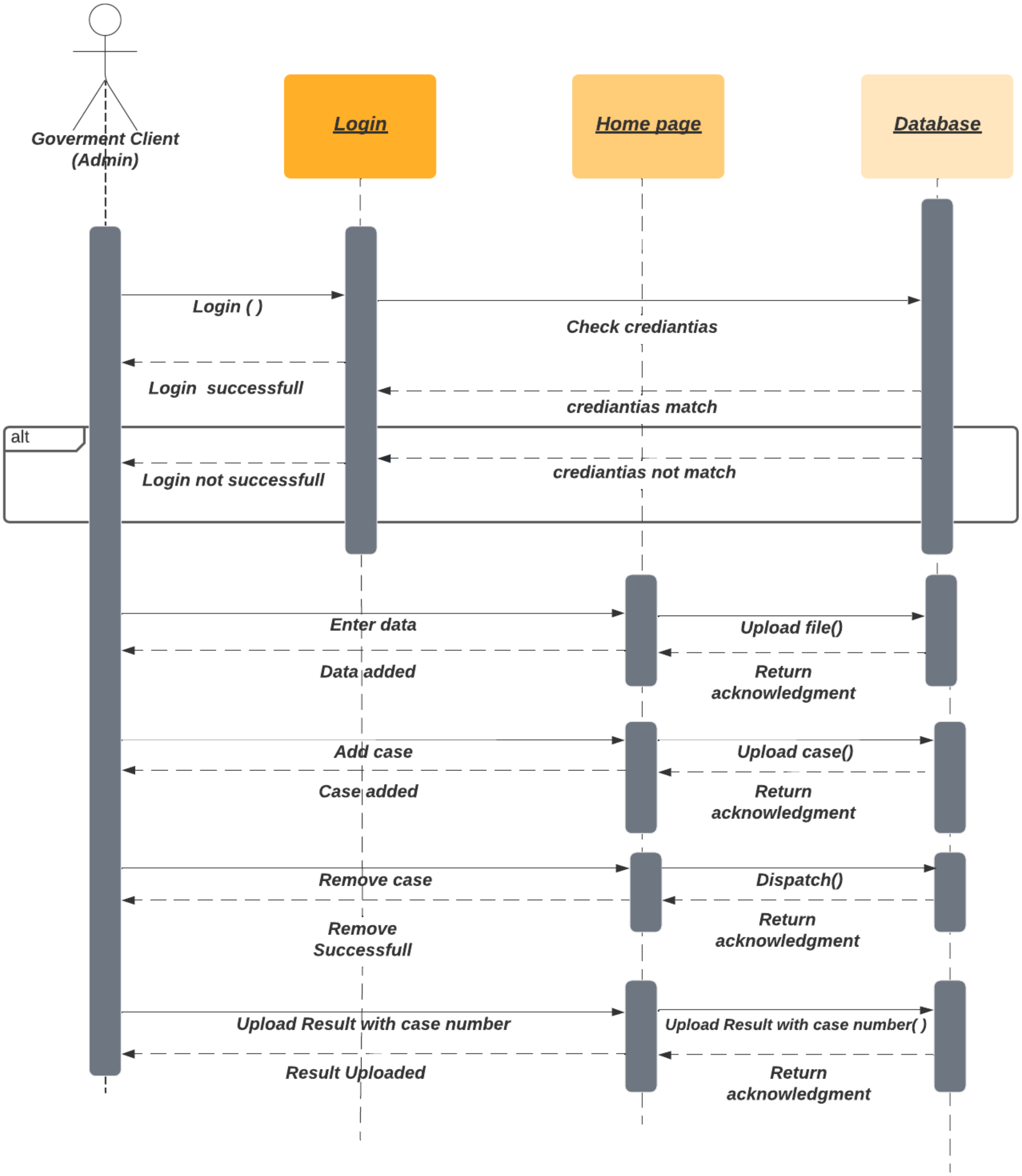


Figure (7) : Government entity sequence diagram

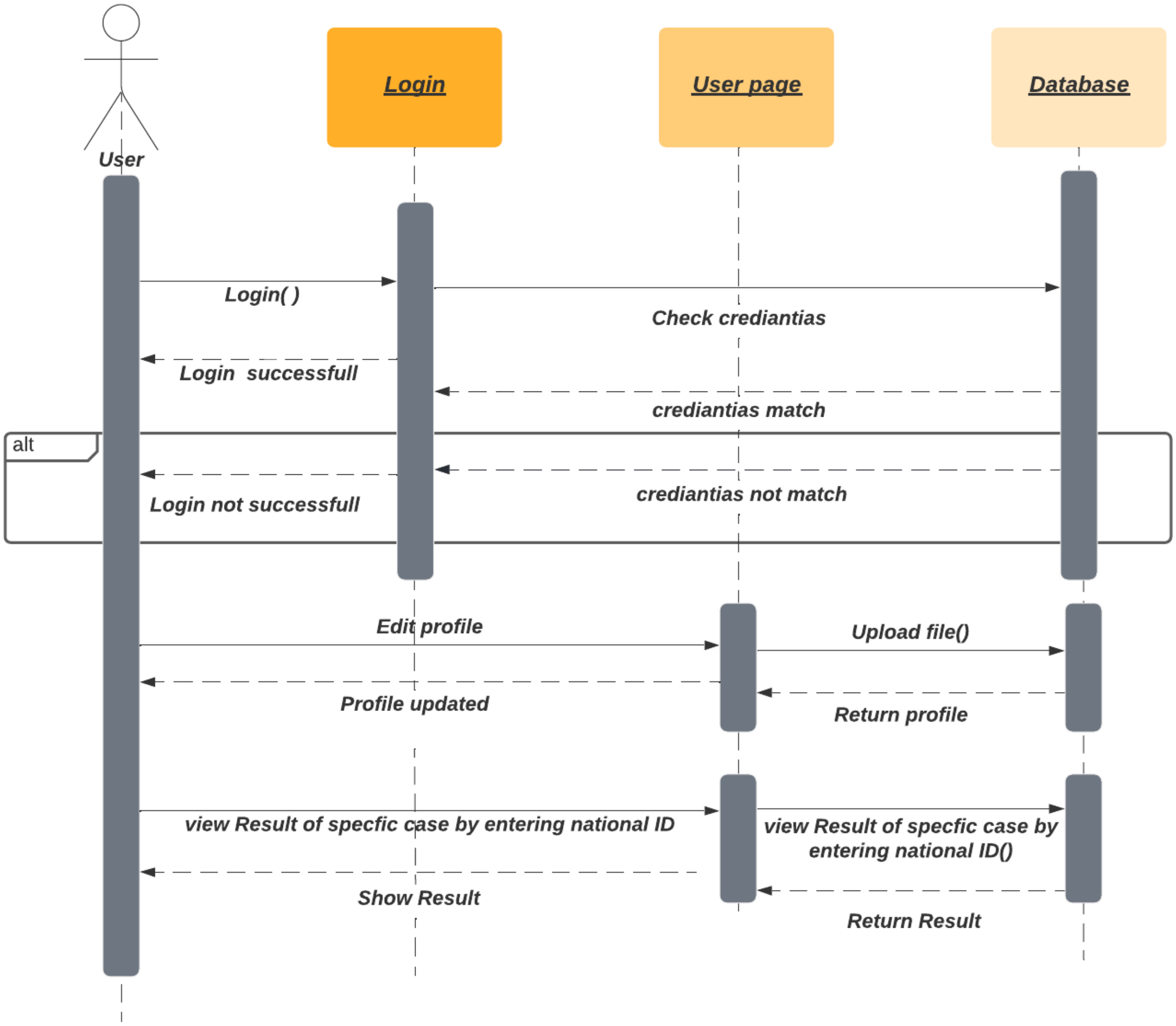


Figure (8) : User sequence diagram

## 3.8 Interface viewpoint

We build our User interface using python, to introduce mobile application and GUI system to be both easy and clear for the Government user to upload the data file that contains either rs numbers or whole genome sequence taken from the laboratories to show the results at the end based on the specific test case given (since there are multiple test cases for paternity and kinship testing).User interfaces are addressed separately in figure 7 and 8.

# 4 Data Design

## 4.1 Data Description

In this phase, our system is going to output the results of paternity and kinship testing based on multiple test cases such as standard trio case (father, mother, child), etc

- The original format of the data is excel files.
- The government user can upload the file containing rs numbers or whole genome sequence taken from the child or both parents
- for each case given there exists a row that contains the result of the system

## 4.2 Dataset Description

The dataset [5]contains a variety of rs numbers and genotypes. Every rs number addressee the position and the two alleles taken from the parents (1 from father, 1 from mother). In the paternity testing between father/mother and child, every rs number and genotypes in the dataset plays a role. Every rs number is made up of two alleles: one from the father, one from the mother, which are combined into one rs number under the label "combined." There are genotypes for each father, mother, and child one, child two, and child three in the data set as well. Another extension we aim to have another section where the data can be whole genome that will be uploaded to the system as well.

| | |
|---|---|
| Dataset Name | Child 1 genome |
| Link | https://www.kaggle.com/zusmani/mygenome?select=Child+1+Genome.csv |
| Size | 601803 samples for rs numbers and their genotypes |
| Number of classes | 4 |
| file size | 15 MB |
| Notes | It consists of 4 features describing each rs number which corresponds to a specific genotype and located at which chromosome |

| | |
|---|---|
| Dataset Name | Child 2 genome |
| Link | https://www.kaggle.com/zusmani/mygenome?select=Child+2+Genome.csv |
| Size | 631984 samples for rs numbers and their genotypes |
| Number of classes | 4 |
| file size | 16 MB |
| Notes | It consists of 4 features describing each rs number which corresponds to a specific genotype and located at which chromosome |

| | |
|---|---|
| Dataset Name | Child 3 genome |
| Link | https://www.kaggle.com/zusmani/mygenome?select=Child+3+Genome.csv |
| Size | 631984 samples for rs numbers and their genotypes |
| Number of classes | 4 |
| file size | 16 MB |
| Notes | It consists of 4 features describing each rs number which corresponds to a specific genotype and located at which chromosome |

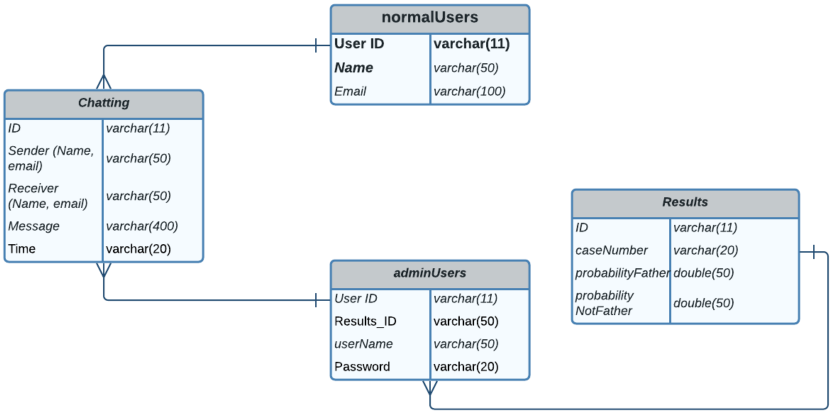
| | |
|---|---|
| Dataset Name | mother genome |
| Link | https://www.kaggle.com/zusmani/mygenome?select=Mother+Genome.csv |
| Size | 601803 samples for rs numbers and their genotypes |
| Number of classes | 4 |
| file size | 15 MB |
| Notes | It consists of 4 features describing each rs number which corresponds to a specific genotype and located at which chromosome |

| | |
|---|---|
| Dataset Name | father genome |
| Link | https://www.kaggle.com/zusmani/mygenome?select=Father+Genome.csv |
| Size | 601803 samples for rs numbers and their genotypes |
| Number of classes | 4 |
| file size | 15 MB |
| Notes | It consists of 4 features describing each rs number which corresponds to a specific genotype and located at which chromosome |

## 4.3 Database design description

There are 4 main tables in the database.

- normal users table: which will store normal users information (name, email) in strings
- admin users table: which will store the information of the admins (username, password) and also another column for results id for when to chose a result id to display
- results table: which will store the results that came from the test into a record having its case number, probability of true and false paternity
- Chatting table: which will be used to record the connection and messages for both normal users and admins containing the sender, receiver, content of message and the time it was sent



| normalUsers | |
|---|---|
| **User ID** | **varchar(11)** |
| ***Name*** | *varchar(50)* |
| *Email* | *varchar(100)* |

| ***Chatting*** | |
|---|---|
| *ID* | *varchar(11)* |
| *Sender (Name, email)* | *varchar(50)* |
| *Receiver (Name, email)* | *varchar(50)* |
| *Message* | *varchar(400)* |
| Time | varchar(20) |

| ***Results*** | |
|---|---|
| *ID* | *varchar(11)* |
| *caseNumber* | *varchar(20)* |
| *probabilityFather* | *double(50)* |
| *probability NotFather* | *double(50)* |

| ***adminUsers*** | |
|---|---|
| *User ID* | *varchar(11)* |
| Results_ID | varchar(50) |
| *userName* | *varchar(50)* |
| Password | varchar(20) |

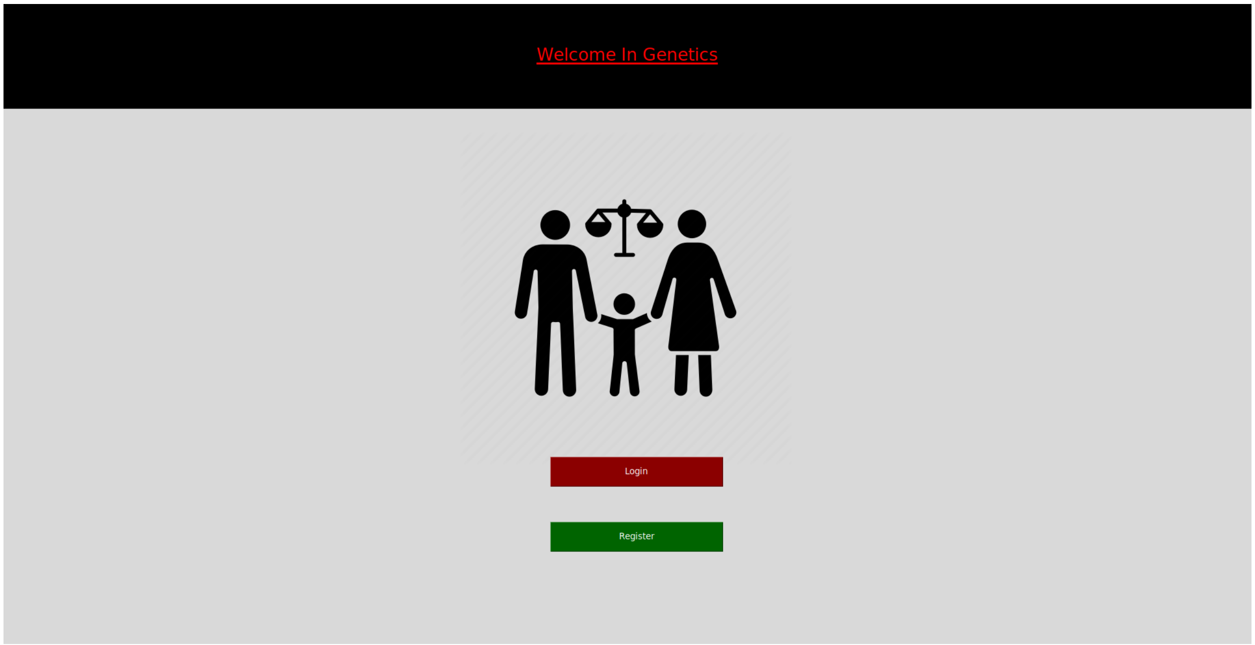
Figure (7) : Database

# 5 Human Interface Design

## 5.1 User Interface

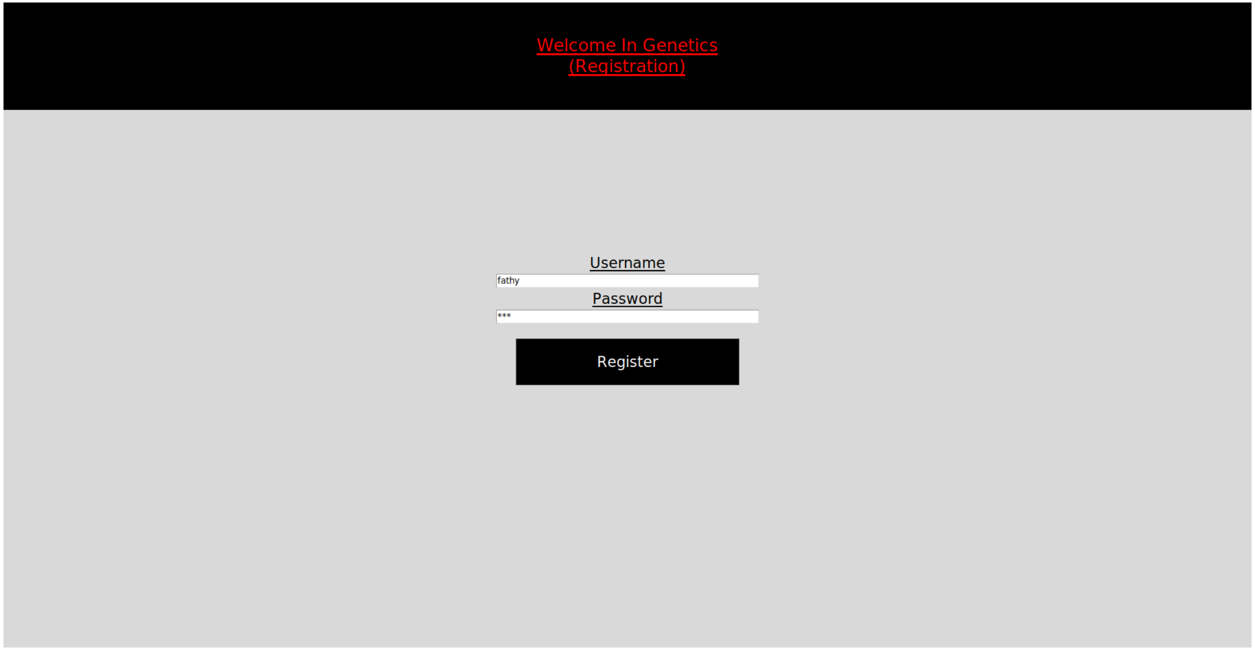
Our system is a mobile application and GUI The government user will be able to access our System through PC. First , The government user has to Sign in by entering the username and password to open account or to sign up if user doesn't have an account, Government user should enter his info as username and password. government user can access the "browse page" page where they will upload the sample files by entering child genome or allele data or both parents , where in the report page where shows Statistics and probability if this is the true father or not.

## 5.2 Screen Images



Welcome In Genetics

Figure (8) : logIn OR Registration

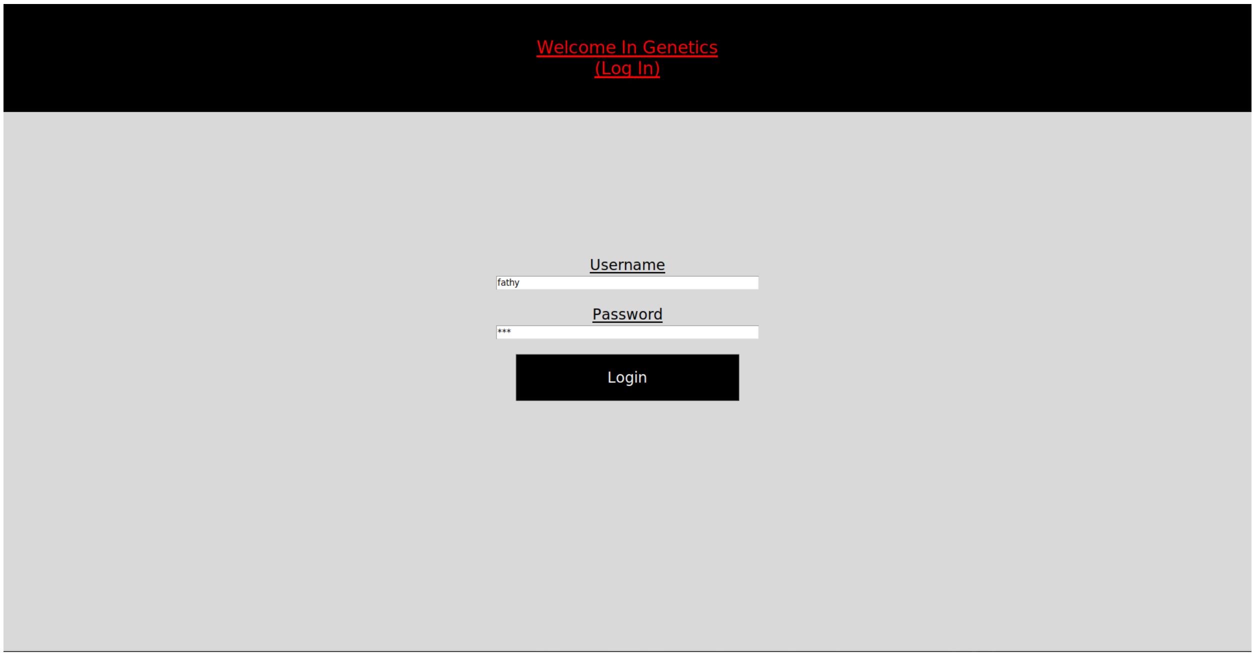


Welcome In Genetics
(Registration)

Username
fathy

Password
***

Figure (9) : Registration screen



Welcome In Genetics
(Log In)

Username
fathy

Password
***

Figure (10) : logIn screen

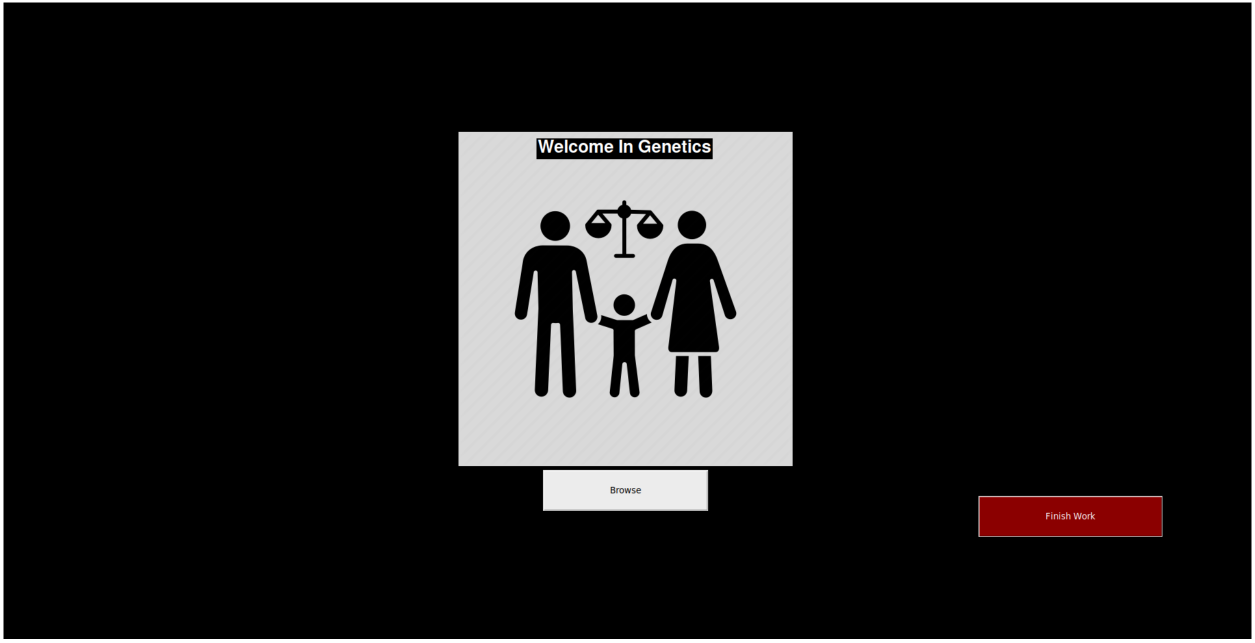
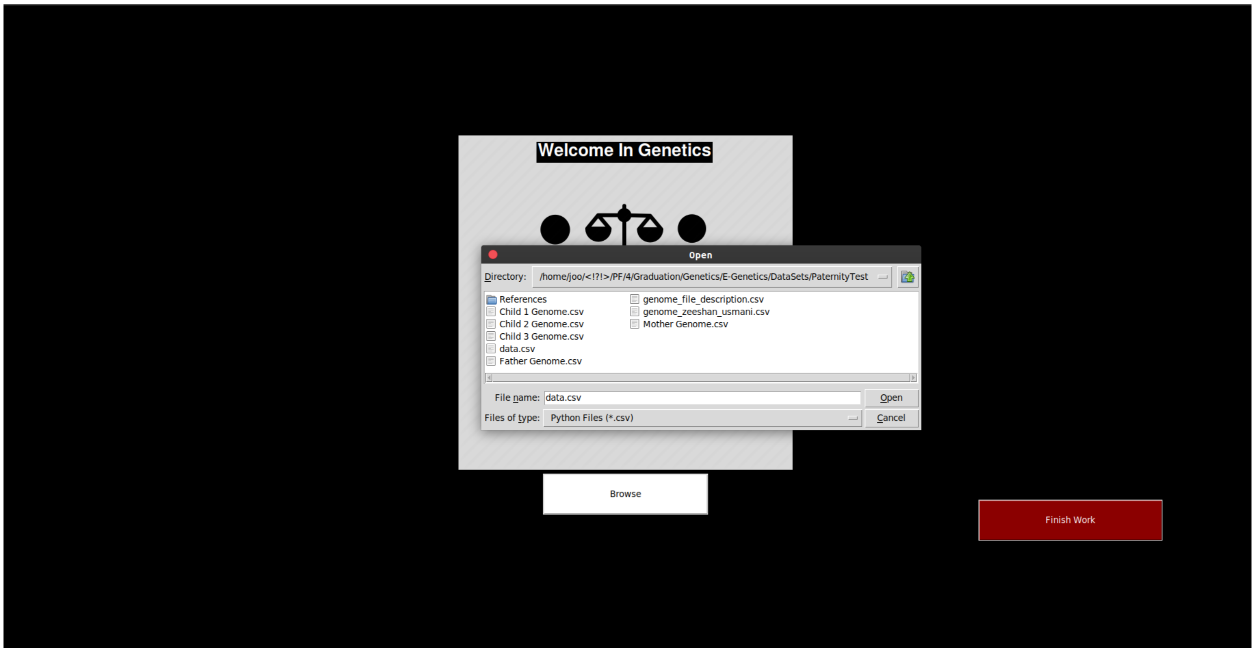


Figure (11) : Home screen

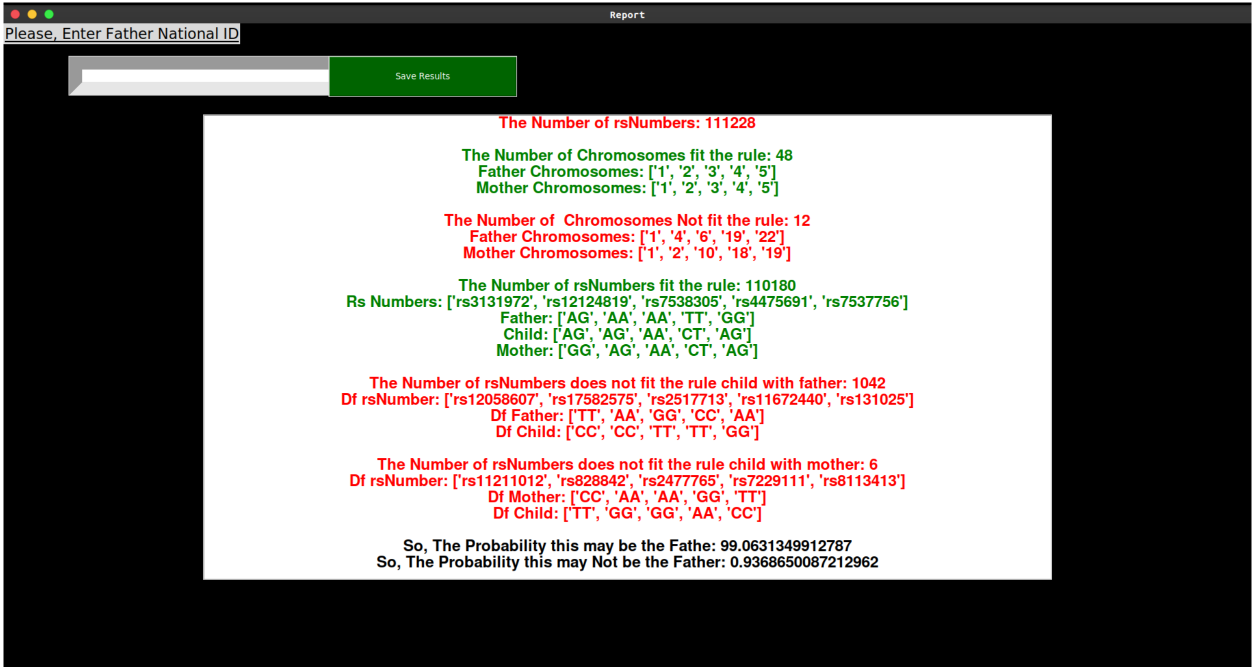


```
Directory: /home/joo/<!?!>/PF/4/Graduation/Genetics/E-Genetics/DataSets/PaternityTest

References                  genome_file_description.csv
Child 1 Genome.csv          genome_zeeshan_usmani.csv
Child 2 Genome.csv          Mother Genome.csv
Child 3 Genome.csv
data.csv
Father Genome.csv

File name: data.csv
Files of type: Python Files (*.csv)
```

Figure (12) : Browse screen



```
The Number of rsNumbers: 111228

The Number of Chromosomes fit the rule: 48
Father Chromosomes: ['1', '2', '3', '4', '5']
Mother Chromosomes: ['1', '2', '3', '4', '5']

The Number of  Chromosomes Not fit the rule: 12
Father Chromosomes: ['1', '4', '6', '19', '22']
Mother Chromosomes: ['1', '2', '10', '18', '19']

The Number of rsNumbers fit the rule: 110180
Rs Numbers: ['rs3131972', 'rs12124819', 'rs7538305', 'rs4475691', 'rs7537756']
Father: ['AG', 'AA', 'AA', 'TT', 'GG']
Child: ['AG', 'AG', 'AA', 'CT', 'AG']
Mother: ['GG', 'AG', 'AA', 'CT', 'AG']

The Number of rsNumbers does not fit the rule child with father: 1042
Df rsNumber: ['rs12058607', 'rs17582575', 'rs2517713', 'rs11672440', 'rs131025']
Df Father: ['TT', 'AA', 'GG', 'CC', 'AA']
Df Child: ['CC', 'CC', 'TT', 'TT', 'GG']

The Number of rsNumbers does not fit the rule child with mother: 6
Df rsNumber: ['rs11211012', 'rs828842', 'rs2477765', 'rs7229111', 'rs8113413']
Df Mother: ['CC', 'AA', 'AA', 'GG', 'TT']
Df Child: ['TT', 'GG', 'GG', 'AA', 'CC']

So, The Probability this may be the Fathe: 99.0631349912787
So, The Probability this may Not be the Father: 0.9368650087212962
```

Figure (13) : Report screen

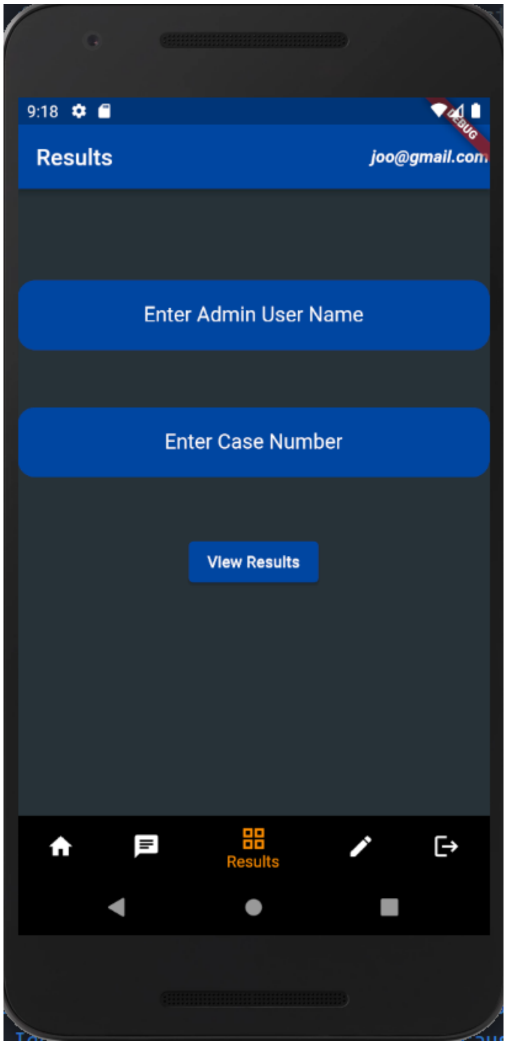
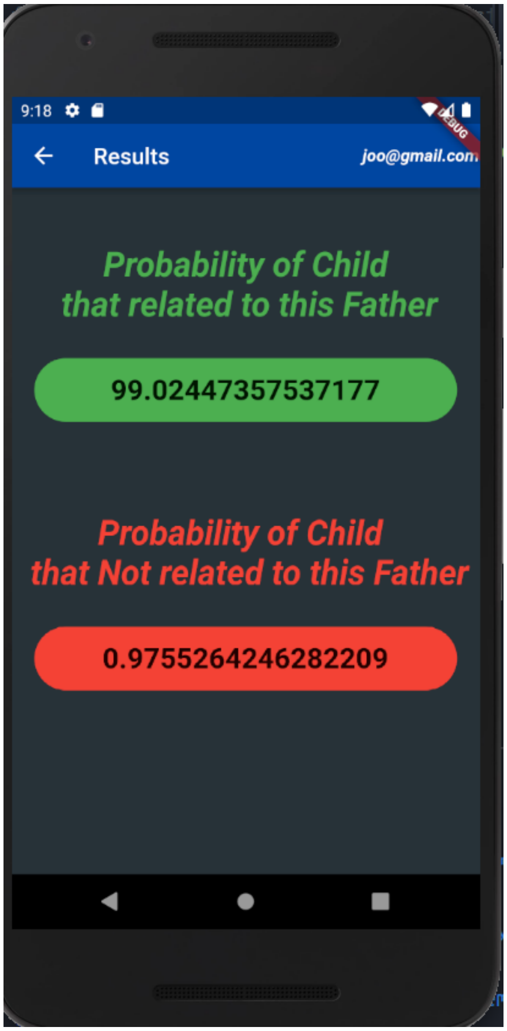


Figure (14) : Results Screen



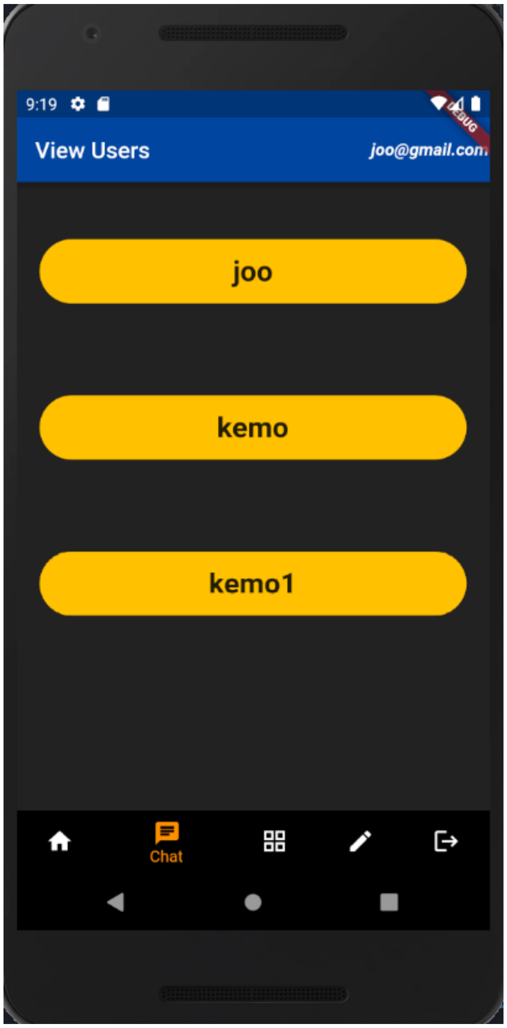
Probability of Child
that related to this Father

99.02447357537177

Probability of Child
that Not related to this Father

0.9755264246282209

Figure (15) : Results Screen

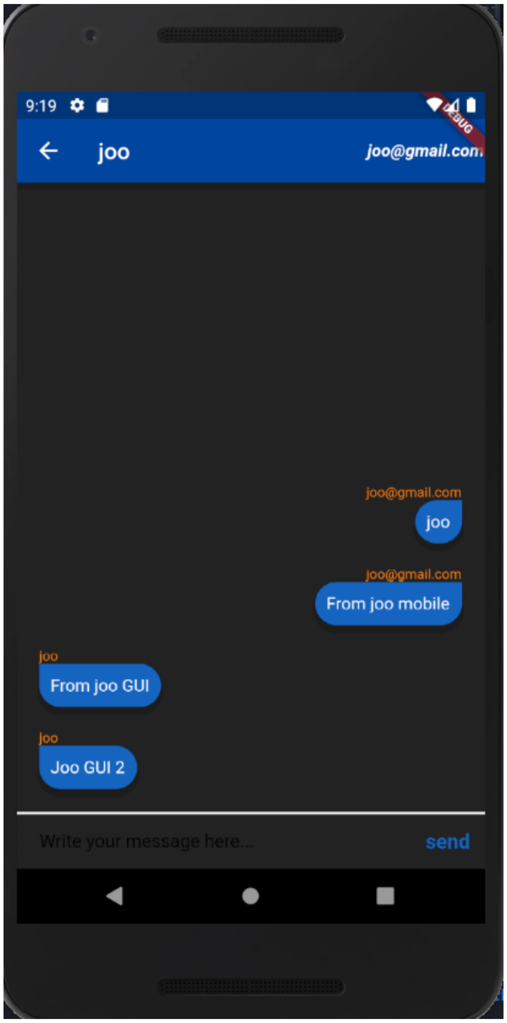


joo

kemo

kemo1

Figure (16) : View Chat Screen



joo@gmail.com

joo

joo@gmail.com

From joo mobile

joo

From joo GUI

joo

Joo GUI 2

Figure (17) : Chatting Screen

## 5.3 Screen Objects and Actions

- Registration: This button let user enter username and password and add it into the database.
- logIn: This button let user enter username and password and compare it with the database then make user join to the system if data exist.
- FinishWork: This Button close the app and make user log Out from the system.
- Browse: This button let user open browse and choose the file that contains (Rs numbers, father, mother, child alleles, and chromosome for each one of them) then generate report.
- Report Screen: This screen view Statistics and probability if this is the true father or not.

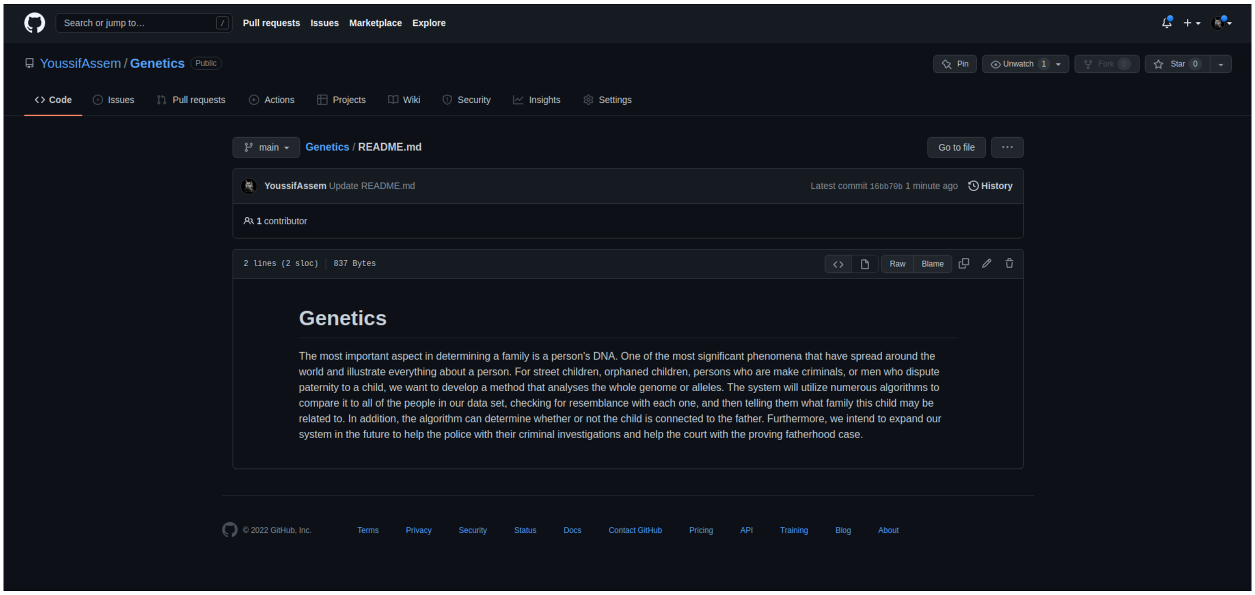
# 6 Requirements Matrix

Table 10: Requirements matrix

| Req. ID | Req Desc | Class | Test Cases ID | Status |
|---|---|---|---|---|
| FR01 | User Register by creating an account | Government user(admin) | TC:01,02,03,04 | Done |
| FR02 | User login to his profile page | Government user (admin),User | TC:06,07,08,19 | Done |
| FR03 | Government User(Admin) upload samples | Government user (admin) | TC22 | Done |
| FR04 | Users view Result as PDF file | Government user (admin) and User | TC26,TC18 | In Progress |
| FR05 | enter national Id to view results | User | TC09,TC10 | Done |
| FR06 | Create new case in a new record for a user | Admin | TC20 | In Progress |
| FR07 | Remove a case from a user's record | Admin | TC21 | In Progress |
| FR08 | Edit a case from a user's record | Admin | TC25 | In Progress |
| FR09 | Enter number of case to choose from | Admin, User | TC20 | Done |
| FR010 | System would show which rs numbers contributed to false paternity | System | TC15,TC14 | Done |
| FR011 | System would encrypt user's credentials with SHA-256 hashing algorithm | System | TC00 | Done |
| FR012 | User would view their profile | User,admin | TC23,TC12 | Done |
| FR013 | User would edit their profile | User | TC11 | Done |
| FR014 | verifying user's login info | User, Admin | TC:06,07,08 | Done |
| FR015 | Validation for all forms for user's info | User, Admin | TC:01,02,03,04 | Done |
| FR016 | User would logout from the system | User, Admin | TC13,TC24 | In Progress |

# 7 APPENDICES

## 7.1 Github



Genetics / README.md

YoussifAssem Update README.md

Latest commit 16bb70b 1 minute ago

1 contributor

2 lines (2 sloc) | 837 Bytes

Genetics

The most important aspect in determining a family is a person's DNA. One of the most significant phenomena that have spread around the world and illustrate everything about a person. For street children, orphaned children, persons who are make criminals, or men who dispute paternity to a child, we want to develop a method that analyses the whole genome or alleles. The system will utilize numerous algorithms to compare it to all of the people in our data set, checking for resemblance with each one, and then telling them what family this child may be related to. In addition, the algorithm can determine whether or not the child is connected to the father. Furthermore, we intend to expand our system in the future to help the police with their criminal investigations and help the court with the proving fatherhood case.

Figure (15) : Github

# References

[1] Sherif H El-Alfy and Ahmed F Abd El-Hafez. “Paternity testing and forensic DNA typing by multiplex STR analysis using ABI PRISM 310 Genetic Analyzer”. In: *Journal of Genetic Engineering and Biotechnology* 10.1 (2012), pp. 101–112.

[2] Ghada A Omran, Guy N Rutty, and Mark A Jobling. “Genetic variation of 15 autosomal STR loci in Upper (Southern) Egyptians”. In: *Forensic Science International: Genetics* 3.2 (2009), e39–e44.

[3] Nicole Wyner, Mark Barash, and Dennis McNevin. “Forensic Autosomal Short Tandem Repeats and Their Potential Association With Phenotype”. In: *Frontiers in Genetics* 11 (2020). ISSN: 1664-8021. DOI: 10.3389/fgene.2020.00884. URL: https://www.frontiersin.org/article/10.3389/fgene.2020.00884.

[4] Andrzej Doniec, Wojciech Łuczak, Maria Wróbel, et al. “Confirmation of Paternity despite Three Genetic Incompatibilities at Chromosome 2”. In: *Genes* 12.1 (2021), p. 62.

[5] zeeshan-ul-hassan usmani$_2$021. “My Complete Genome”. In: *kagle.* (2021).