Bilkent University

Department of Computer Engineering

Senior Design Project

LIBRA: Genetic Filtering and Diagnosis Matching System

Analysis Report

Mahmud Sami Aydın, Berke Egeli, Naisila Puka, Halil Şahiner, Abdullah Talayhan

Supervisor: Can Alkan
Jury Members: Abdullah Ercüment Çiçek and Hamdi Dibeklioğlu

Analysis Report
Nov 11, 2019

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

Table of Contents

1 Introduction

Many hospitals, laboratories and medical research centers want to collect and store genomic data, as well as explore and interpret that data based on specific needs. There are open-source programs developed and utilized for such purposes that have been accepted by the authorities [1]. Yet, they are not suitable for direct use and the installation requires specific expertise.

The collective data produced by these institutes possess valuable information related to genetic profiles. These genetic profiles can be useful for comparing and diagnosing rare diseases. For example, a child in San Francisco Bay Area had a rare disease caused by not being able to produce tears. The doctors were suspicious about a gene called NGLY1 after the results of genetic profiling. Yet, they were not sure about the cause because of the lack of genetic profiles of other patients for comparison. The issue was resolved after finding a patient with similar phenotypes studied by Duke University and NGLY1 was indeed the gene causing the disease [2]. Examples like this have created a high demand for genomic discovery through the comparison of genotypic/phenotypic profiles.

The aim of LIBRA is to provide a user-friendly genetic filtering and annotation system equipped with a genetic profile matching platform, that can be quickly integrated and easily used by medical institutions in order to explore genetic variation, detect and diagnose rare diseases, as well as safely collect and store their data.

2 Current System

Currently, there exists two systems that are similar to major components of our project. The first one is the *GEMINI* framework and the second one is the *Matchmaker Exchange* project. GEMINI is a framework for exploring genetic variation, in which genetic variants are loaded using VCF files. It automatically annotates the variants by using existing databases. Since GEMINI is a command line tool, it does not have a user friendly interface [3]. Matchmaker Exchange project is a platform that aims to facilitate matching of patients with similar phenotypic and genotypic profiles in order to find the underlying genetic causes of rare diseases. The project incorporates many different platforms through its API [4].

3 Proposed System

3.1 Overview

LIBRA is going to be a web application composed of two main modules: *Genetic Variation Query Interface* and *Patient Matching Platform*. The first module provides an interface for storing and annotating genomic data in order to query variants and explore the data, whereas the second one acts as a patient social network for doctors who seek similar genetic profiles related to a specific disease in order to understand and diagnose the disease further. These modules will be integrated into the same user interface. The potential users of this project are medical doctors in medium-sized hospitals, laboratories and research centers in Turkey.

3.1.1 Genetic Variation Query Interface

This module stems from an existing framework called GEMINI [3]. GEMINI is a framework for exploring genetic variation, in which genetic variants are loaded using VCF files. It automatically annotates the variants by using existing databases. This module of LIBRA aims to improve this framework. Doctors will be able to perform variation queries from a convenient user interface supplied by the web application. LIBRA will support annotation by comparing with all open genetic databases such as *1000 genomes*, *dbSNP*, *dbVar*, *ClinVar*, *OMIM*, *COSMIC*, *ENCODE* etc. in order to become a common access point for making genetic variation queries.

GEMINI uses a legacy portable database. This database is not compatible with our security and scalability requirements, as will be explained in more detail in the following sections. Therefore, LIBRA will reconstruct this infrastructure using a modern database structure, later extended to a distributed one. LIBRA will be an *SaaS* (Software as a Service) product, which implies that the users will not be responsible for setting up the databases and configuration files.

LIBRA will also have an integrated query editor that will enable users to request real time queries. Running basic SQL queries will be supported. The additional benefit of LIBRA's query editor will be the support of *Genotype Query Tools* [5] which provides faster queries computed over genomes represented in a specific format (compressed bitmap).

3.1.2 Patient Matching Platform

In this module, doctors can create accounts for patients as in social networks. The main purpose of the module is to help doctors understand rare diseases by comparing different patients with similar phenotypes. This module will make use of MatchMaker API [4] protocols in order to make queries for genotype/phenotype comparison and matching. Additional data for phenotypes will be supplied by using Human Phenotype Ontology (HPO), which provides a standardized vocabulary of phenotypic abnormalities encountered in human disease [6].

In this module, the privacy of the patients is a main concern. Further elaboration will be done regarding privacy and security in the following sections.

3.2 Functional Requirements

3.2.1 User Accounts

- Users will be able to create an account in LIBRA given that they work in a hospital that LIBRA accepts (a list of hospitals will be prepared).
- After the hospital is accepted by LIBRA, they will provide personal and work information.
- A verification email or SMS will be sent to the hospital the user claims to be working at, based on the hospital's contact data found in LIBRA.

3.2.2 Genetic Variant Upload and Automatic Annotation

- Users will be able to load genetic variants of their patients of interest to LIBRA using a specific format (VCF).
- After loading, they will be given the list of the databases supported for annotation and the users will choose the ones they prefer.
- The genetic variants will then be automatically annotated by comparing them to several online genome annotation sources such as dbSNP, KEGG, etc.

3.2.3 Annotation Based Variant Querying

- Users can query variants based on specific requirements related to variants' attributes since the underlying system will organize the genotypes and annotations.

- Users can choose to write their own query or use LIBRA's built-in query editor. This editor provides most common genome queries (e.g. filtering on genotypes, finding which samples have a specific variant, variants with specified allele frequency percentage, etc.) Users will be able to modify these query templates through the editor.
- Users can also customize their own queries and add it as a template for future queries.
- Users will be able to combine results from different queries through the editor.
- Users can save previous query results and use them on another query.

3.2.4 Custom Annotation and Variant Analysis

- Users will be able to annotate the variants with their own specific annotated file, which might describe genome regions particularly relevant to user's purpose.
- Users will be able to share this annotated file with another hospital based on their desire. In this way, the users belonging to the other hospital will also be able to annotate their variants with that custom file.
- Users can run analytics tools in the system since the underlying system will support analytical queries. These analytics tools include identifying potential variants related to some specific disorder (e.g. compound heterozygotes cause many autosomal recessive disorders [7]).
- Users can choose to locally save a descriptive file of the run analytics.

3.2.5 Patient Profiles

- Users will be able to create accounts for their patients in LIBRA with the national ID of the patient if there is no account for that patient in the system.
- After creating the account, users will enter the information of the patient based on user's stored genomic data in LIBRA. While doing that, they can decide which information will be shared with matchmaker system.
- If patient account is already available, users can edit the current information of the patient, e.g. add new diseases for the patient.
- Users can alter/delete their patient accounts.

3.2.6 Patient Matching

- Users will be able to search for similar patients based on customized attributes of the genomic profile of their patient of interest.
- Users can also run customized filters on the matching results in terms of information points shared through the databases to focus on different groups of patients.

3.3 Non-functional Requirements

3.3.1 Scalability

The Gemini tool uses SQLite database. In order to increase the scalability of our system we plan to upgrade the infrastructure by switching to PostgreSQL database, later extended to the distributed version.

3.3.2 Backup and Recovery

The hospitals should be able to do their backup on their local database. Also the server needs to do regular backups for the accounts of doctors and their patients in the genetic matching platform.

3.3.3 Availability

The application should be accessible to users at all times, with only the possible exceptions of server maintenance.

3.3.4 Accessibility

The application should provide language support for Turkish language since the application's focus group is medical doctors in Turkey.

3.3.5 Reliability

The system will ensure that the comparison of the patients' information to find matches for diseases must give reliable results and handle unexpected failures. That is, if the system fails while doing comparison (server/client side errors), the error will be reported and the comparison procedure will handle the process appropriately, making sure only reliable matches/mismatches will be displayed.

3.3.6 Portability

The system should be compatible with different browsers. This means that LIBRA will be developed as a platform independent web application.

3.3.7 Sustainability

In order to increase the capacity of our system to endure through time, we are mainly focused in the underlying database. The more scalable the database is, the more sustainable it will be when compared with the exponential data growth (in particular genomic data) within the years. Also, updates according newly released versions will contribute in sustainability of the system.

3.4 Psuedo Requirements

3.4.1 Implementation Requirements

LIBRA will be implemented using Python Django as backend supplied with React JS as frontend. The underlying database will be PostgreSQL, later extended to a distributed version.

3.4.2 Economic Requirements

The primary economic constraints on this project will be imposed by the cost associated with server hosting and maintenance of the web application. The servers will be hosted on Google Cloud Platform [8].

3.4.3 Privacy and Legal Requirements

The application should comply with the instated laws and regulations of the host country/international community, such as KVKK (Kişisel Verilerin Korunması Kanunu) in Turkey [9].

3.5 System Models

3.5.1 Scenarios

Scenario 1

Use Case Name:

- Create Account

Actors:

- Doctor

Entry Conditions:

- Doctor is in the home page.

Exit Conditions:

- Doctor created an account and is in the account page.
- Doctor failed to create an account and is in the home page.

Main Flow of Events:

1. Doctor clicks on the "Sign Up" button in the home page.
2. LIBRA navigates to the "Sign Up" page.
3. Doctor selects a hospital that is registered in the LIBRA database.
4. Doctor provides personal and work information.
5. Doctor selects username.
6. Doctor selects password.
7. Doctor clicks on the sign up button.
8. LIBRA verifies the doctor works at the selected hospital.
9. Doctor account is created.
10. Doctor is redirected to the account page.

Alternative Flow of Events:

1. Doctor clicks on the "Sign Up" button in the home page.
2. LIBRA navigates to the "Sign Up" page.
3. Doctor selects a hospital that is registered in the LIBRA database.
4. Doctor provides personal and work information.
5. Doctor selects username.
6. Doctor selects password.
7. Doctor clicks on the sign up button.
8. LIBRA verifies the doctor works at the selected hospital.
9. Verification fails.
10. Doctor is redirected to the home page.

Scenario 2

Use Case Name:

- Verify Account

Actors:

- Hospital/Admin

Entry Conditions:

- Admin receives an email to verify a doctor account.

Exit Conditions:

- Admin verifies doctor.
- Admin rejects the verification request by the doctor.

Main Flow of Events:

1. Admin receives an email from the LIBRA system to verify a doctor.
2. Admin is redirected to the verification page in his admin account.
3. Admin checks the personal and work information of the doctor that requested the verification and verifies the doctor.

Alternative Flow of Events:

1. Admin receives an email from the LIBRA system to verify a doctor.
2. Admin is redirected to the verification page in his admin account.
3. Admin checks the personal and work information of the doctor that requested the verification and rejects the verification request by the doctor.

Scenario 3

Use Case Name:

- Genetic Variant Upload and Automatic Annotation

Actors:

- Doctor

Entry Conditions:

- Doctor is in the account page.

Exit Conditions:

- Patient data is uploaded to LIBRA.
- Patient data is annotated.

Main Flow of Events:

- Doctor clicks on the "Load Genetic Variant" button.
- Doctor loads the genetic variant of the patient of interest using VCF format.
- Doctor selects the preferred database among the databases supported by LIBRA.
- LIBRA compares the uploaded data to online annotation sources.
- LIBRA automatically annotates the genetic variants.

Scenario 4

Use Case Name:

- Query Creation Through Filtering

Actors:

- Doctor

Entry Conditions:

- Doctor is in the query editor.

Exit Conditions:

- Doctor receives the query results.

Main Flow of Events:

- Doctor selects the database they want to query.
- Doctor sets the filtering constraints on the columns of the database that will be queried.
- Doctor hits the "Return Query Results" button.
- LIBRA returns the query results.

Scenario 5

Use Case Name:

- Manual Query Creation

Actors:

- Doctor

Entry Conditions:

- Doctor is in the query editor.

Exit Conditions:

- Doctor receives the query results.

Main Flow of Events:

1. Doctor selects the database they want to query.
2. Doctor modifies the query templates provided by the query editor.
3. Doctor hits the "Return Query Results" button.
4. LIBRA returns the query results.

Scenario 6

Use Case Name:

- Query Creation Using Previous Results

Actors:

- Doctor

Entry Conditions:

- Doctor is in the query editor.

Exit Conditions:

- Doctor receives the query results.

Main Flow of Events:

1. Doctor selects the database they want to query.
2. Doctor manually enters the query to the query editor.
3. Doctor uses a previously saved query result in the new query.
4. Doctor hits the "Return Query Results" button.
5. LIBRA returns the query results.

Scenario 7

Use Case Name:

- Query Customization

Actors:

- Doctor

Entry Conditions:

- Doctor is in the query editor.

Exit Conditions:

- Customized query is saved as a template.

Main Flow of Events:

1. Doctor enters the customized query to the query editor.
2. They hit "Save" button to save the customized query as a template for later use.

Scenario 8

Use Case Name:

- Combine

Actors:

- Doctor

Entry Conditions:

- Doctor is in the query editor
- Doctor ran a query.
- Doctor saved the query results.

Exit Conditions:

- Two or more query results are combined.

Main Flow of Events:

1. Doctor clicks on the "Combine Results" button on the query editor.
2. Doctor is prompted to select two or more saved query results.
3. LIBRA combines the saved query results and displays/saves the combined query result.

Scenario 9

Use Case Name:

- Save

Actors:

- Doctor

Entry Conditions:

- Doctor is in the query editor
- Doctor ran a query.

Exit Conditions:

- Query result is saved.

Main Flow of Events:

1. Doctor clicks on the "Save Results" button on the query editor.
2. LIBRA stores the query results on a database/local file.

Scenario 10

Use Case Name:

- Create Patient

Actors:

- Doctor

Entry Conditions:

- Doctor is in Homepage

Exit Conditions:

- Doctor creates a Patient.

Main Flow of Events:

1. Doctor clicks create patient button.
2. Doctor fills disease profile form.
3. Doctor clicks "Attach VCF file" button.
4. Doctor selects VCF file from computer.
5. Doctor clicks "Attach Disease Photograph" button.
6. Doctor selects photograph from computer.
7. Doctor clicks submit button.

Scenario 11

Use Case Name:

- Add Disease to Patient Profile

Actors:

- Doctor

Entry Conditions:

- Doctor is in Patient Management Page.

Exit Conditions:

- Disease is added into Patient Profile.

Main Flow of Events:

1. Doctor clicks edit button for according patient.
2. Doctor check the "Diagnosis Code" box.
3. Doctor writes disease code in to the text field.
4. Doctor clicks submit button.

Scenario 12

Use Case Name:

- Choose Patient Privacy Level

Actors:

- Doctor

Entry Conditions:

- Doctor is in Patient Management Page.

Exit Conditions:

- Patient privacy level is selected.

Main Flow of Events:

1. Doctor clicks open sharing manager button.
2. Doctor selects patient whose privacy level will be changed.
3. Doctor make private the privacy selection level.
4. Doctor clicks submit button.

Alternative Flow of Events:

1. Doctor clicks open sharing manager button.
2. Doctor selects patient whose privacy level will be changed.
3. Doctor make public the privacy selection level.
4. Doctor choose institutions which can be use the patient in matcher.
5. Doctor clicks submit button.

Scenario 13

Use Case Name:

- Match Patient

Actors:

- Doctor

Entry Conditions:

- Doctor is in Patient Management Page.

Exit Conditions:

- Doctor saves similar patients list to the given patient.

Main Flow of Events:

1. Doctor clicks "Matchmaker" button of the patient.
2. Doctor selects matching algorithm from the given list.
3. Doctor selects Institute where search will be done.
4. Doctor clicks "Match" button.
5. Doctor see results in the table.
6. Doctor clicks "Save Results" button.

Alternative Flow of Events:

1. Doctor clicks "Matchmaker" button of the patient.
2. Doctor selects customized algorithm option.
3. Doctor selects algorithm which will be run.
4. Doctor selects Institute where search will be done.
5. Doctor clicks "Match" button.
6. Doctor see results in the table.
7. Doctor clicks "Save Results" button.

Scenario 14

Use Case Name:

- Suspend A User

Actors:

- Admin

Entry Conditions:

- Admin is in User Management Page.

Exit Conditions:

- User is suspended.
- A message about suspension is sent to the user.

Main Flow of Events:

1. Admin clicks to suspend button.
2. Admin writes the reason of suspension.
3. Admin clicks "OK" button.

Alternative Flow of Events:

1. Admin clicks "Suspend" button.
2. Admin clicks "Cancel" button.

3.5.2 Use-Case Model

Following is the use case diagram of LIBRA:

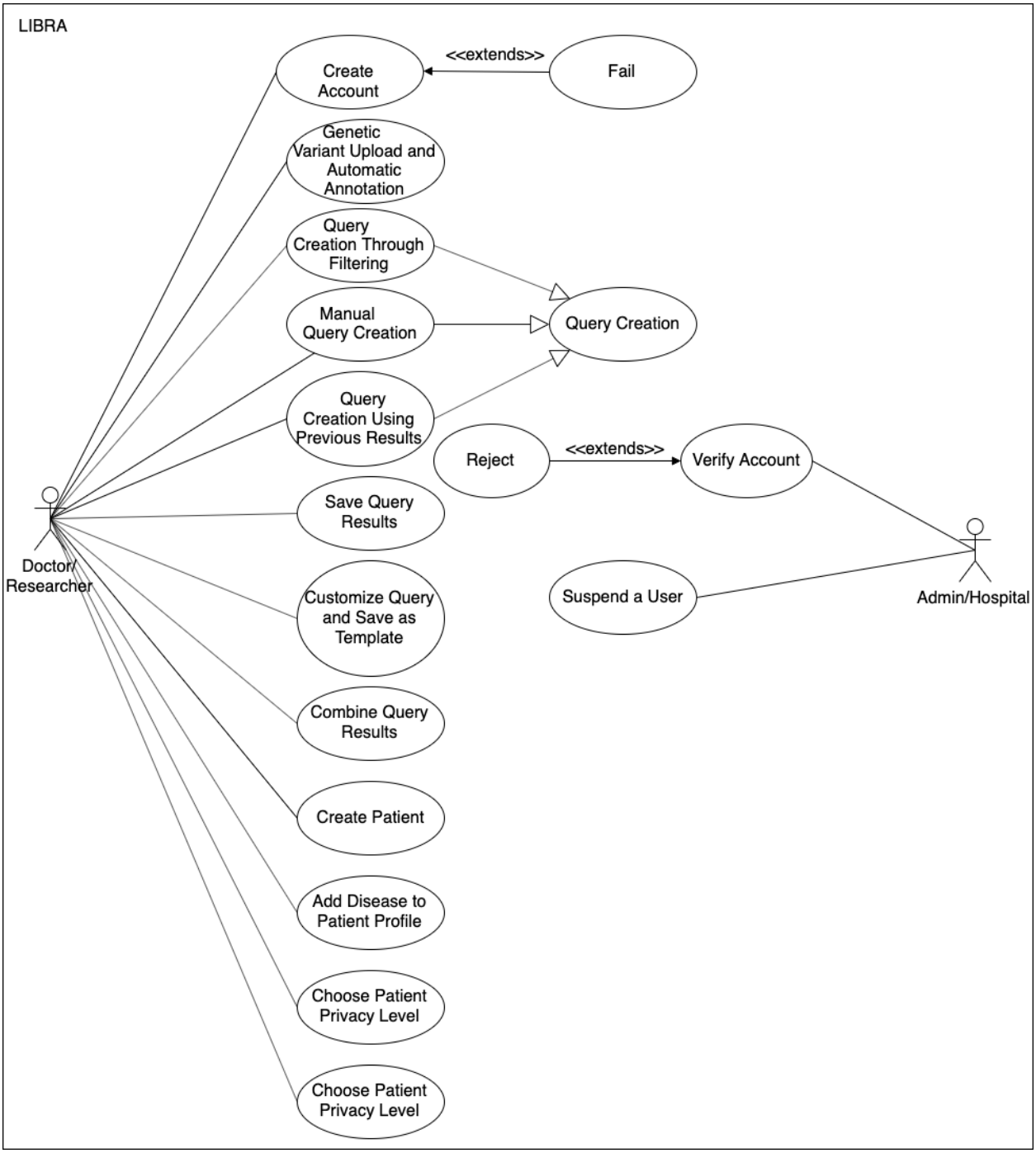


Figure 1: Use Case Diagram of LIBRA

3.5.3 Object and Class Model

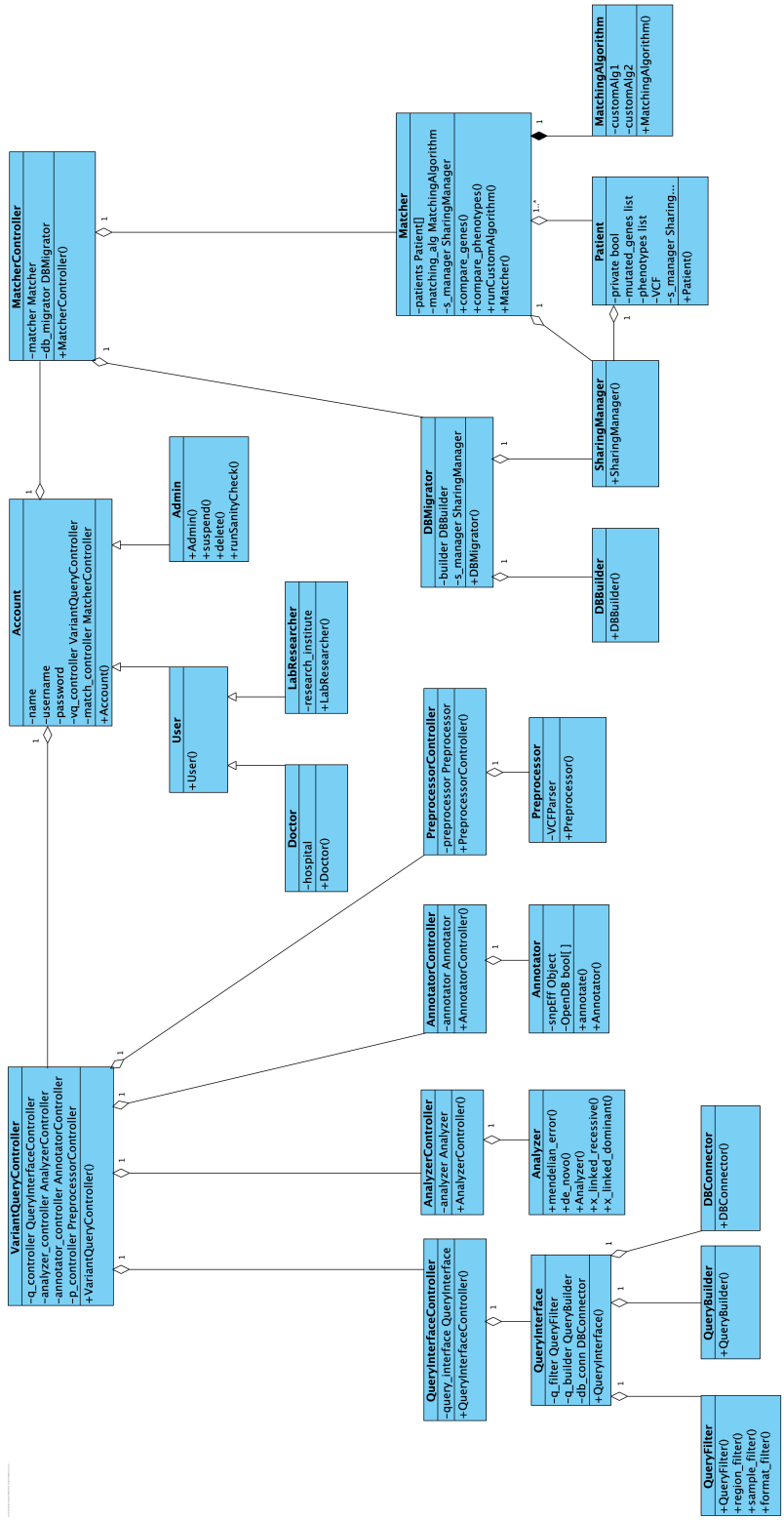


Figure 2: Class Diagram for LIBRA

Accounts

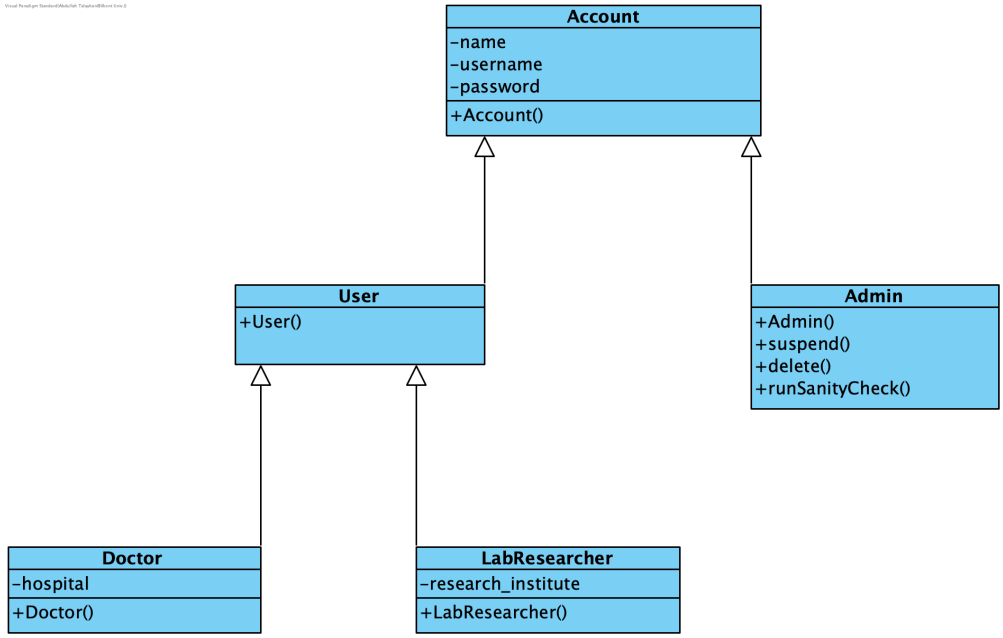


Figure 3: Class Diagram for Accounts

- **Account:** This class is the parent class for different account types that the system will have.
- **User:** This class is the parent user class responsible for the general user tasks such as storing user information.
- **Admin:** This class has the ability to manage user accounts such as suspension or deletion.
- **Doctor:** This class is a user account that has the ability to set diagnosis.
- **Lab Researcher:** This class is a user account similar to Doctor but does not have the ability to set diagnosis.

Genetic Variation Query Interface

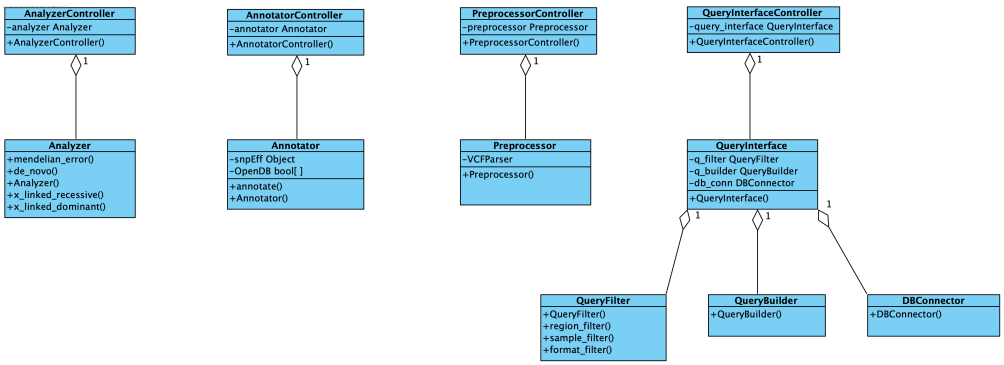


Figure 4: Class Diagram for Genetic Variation Query Interface

- **Preprocessor/Controller:** This class is responsible for parsing of the VCF files before annotation.
- **Annotator/Controller:** This class is responsible for automatic annotation of uploaded VCF files.
- **Analyzer/Controller:** This class is responsible for serving as an API for several analysis method such as mendelian error and de novo mutations.
- **QueryInterface/Controller:** This class is responsible for building the queries, sending queries to the databases and fetching results.
- **QueryBuilder:** This class creates a database query based on the user input.
- **QueryFilter:** This class sets the filters for a specific query.
- **DBConnector:** Auxilary class for sending queries to databases and fetching results.

Matchmaking System

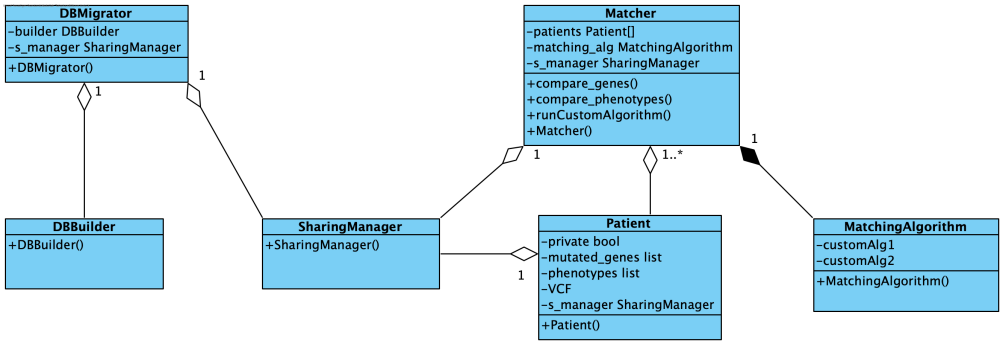


Figure 5: Class Diagram for Matchmaking System

- **Matcher:** This class contains instances of many objects related to patient matching and orchestrate them.
- **Patient:** Class for storing patient information.
- **MatchingAlgorithm:** This class is used for designating a customized matching algorithm.
- **SharingManager:** This class is used for managing the shared information related to patients.
- **DBMigrator:** This class is used for enrolling new databases to the matching process.
- **DBBuilder:** This class is used for setting the database up and running on LIBRA servers.

3.5.4 Dynamic Models

LIBRA as a web application has two main activities to offer: genetic variant exploration and patient matching procedures. Therefore, an activity diagram is the most appropriate diagram showing dynamic models of our system, since it provides depiction of the workflow in the system. Following are two activity diagrams, each describing one of LIBRA's main activities.

3.5.4.1 Genetic Variant Exploration - Activity Diagram

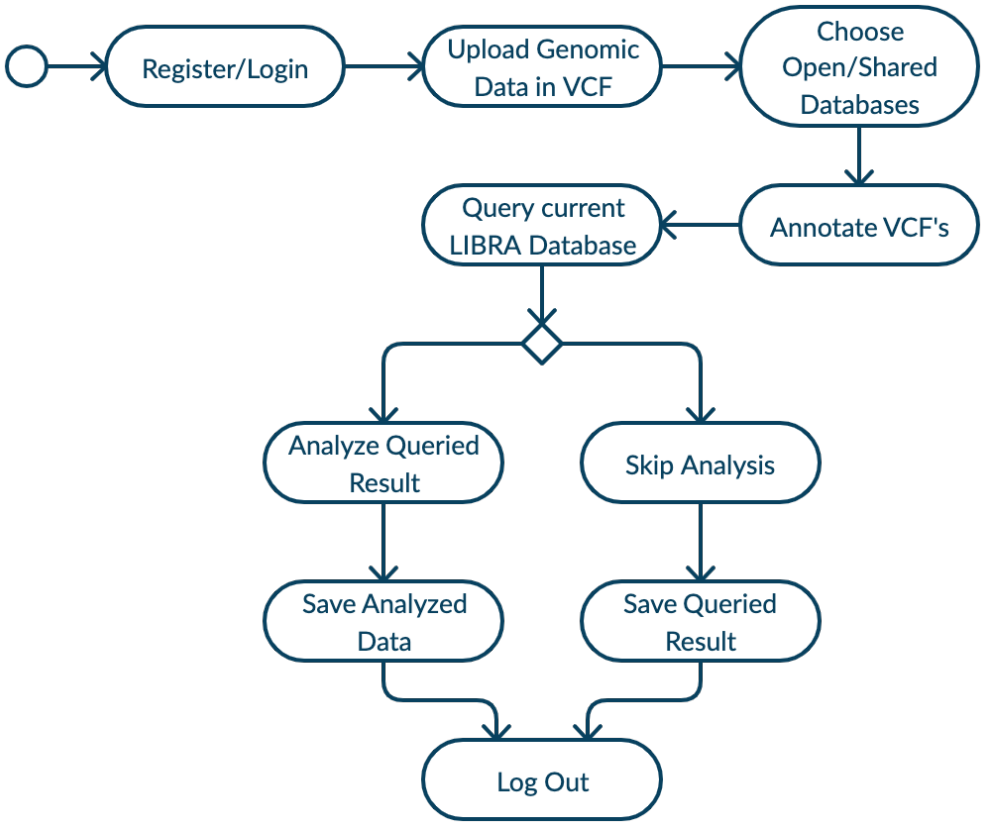


Figure 6: Genetic Variant Exploration Activity Diagram

Since there is only one swimlane for this activity diagram, and the object implementing the activity is the *Account* object of the users of our system, this part is ommited from both the diagrams. After registering/logging in to LIBRA, the user can upload a VCF file containing genomic data that needs annotation. Then, the user chooses the databases for annotating the variants (each variant is annotated by comparing it to several genome annotations from these database sources). After annotation, the variants are saved in the LIBRA database. The user can query the current database. At this point, a decision is modeled, where the user can choose to analyze the queried results before saving them or skip analyzing. Then, the activity is finished and the user logs out of the system.

3.5.4.2 Patient Matching - Activity Diagram

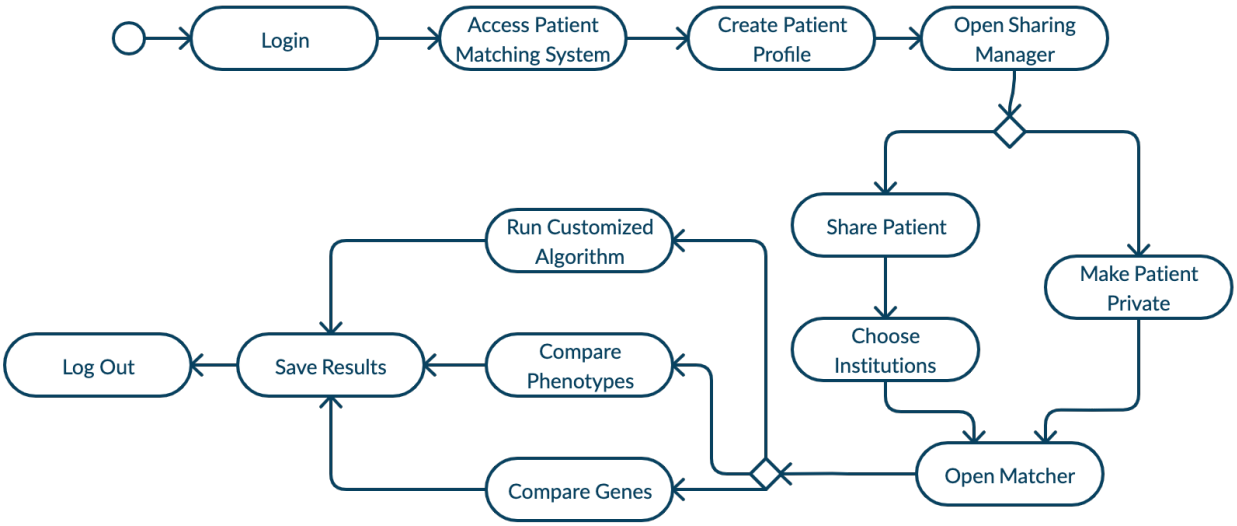


Figure 7: Patient Matching Activity Diagram

After logging in to LIBRA, the user accesses the *Patient Matching System*. Then, the user creates a new patient profile by following the appropriate steps (check mock-ups for further details). The user accesses the sharing manager of that new patient, and at this point, a decision is modeled on the privacy of the patient. The user can choose to keep the patient private (only that account can access the patient information), or share the patient with institutions that the user can choose. Then, the user opens the *Matcher* in order to run a patient matching process. There is another decision modeled at this point where the user chooses the algorithm to run for matching. This algorithm can be: comparing genes, comparing phenotypes or running a customized comparison algorithm (for example, run gene and phenotype comparison concurrently but give a larger weight coefficient to genes than phenotype). Matching results will appear and the user can save them. The activity is finished and the user logs out of the system.

3.5.5 User Interface

3.5.5.1 Login

Users of LIBRA (medical doctors, etc) will login to the system from here by providing their *Username* and *Password*. They will be directed to a password changing page if they forgot their password.

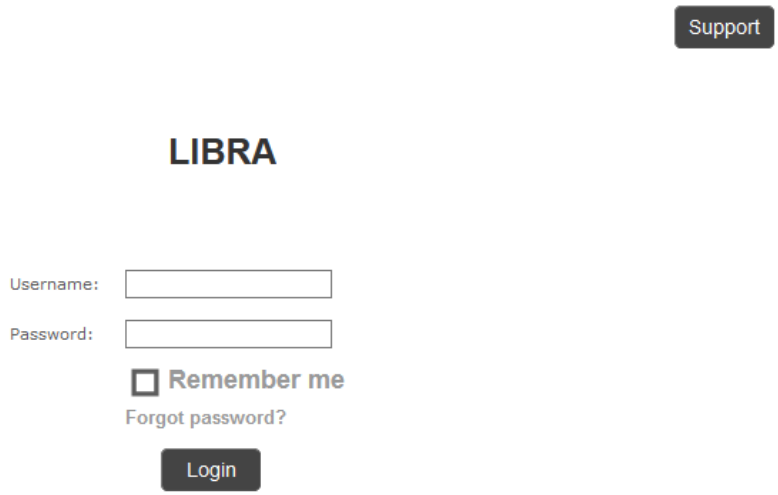


Figure 8: Mock-up for Login Page

3.5.5.2 Creating a Patient

Medical doctors will create a profile for a new patient to enroll the patient to the *Matcher* system. They can select which information will be shared publicly and attach a VCF file for the patient while creating the profile.

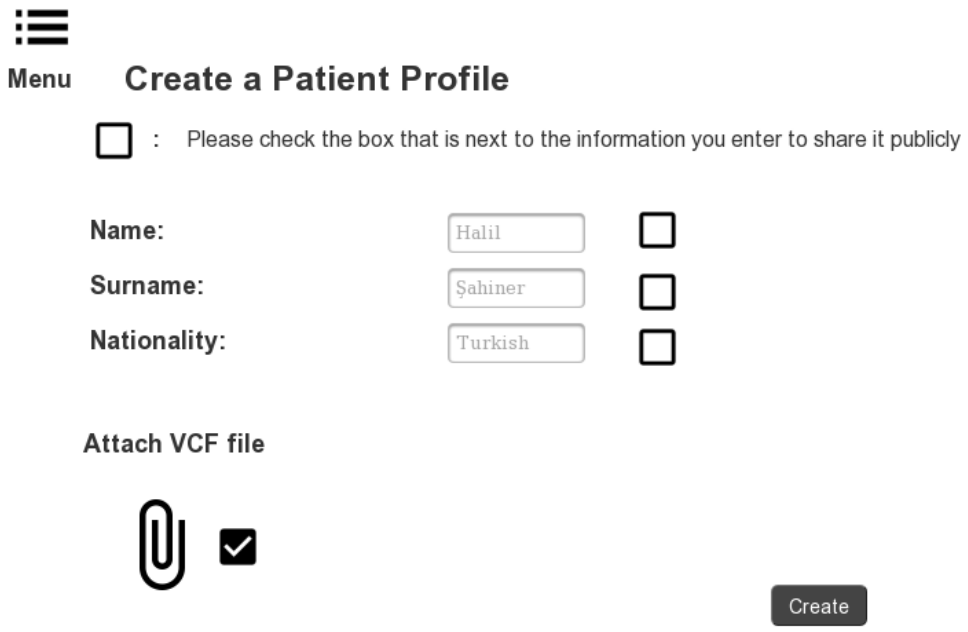


Figure 9: Mock-up for Patient Profile Creation Page

3.5.5.3 Management of Patients

Medical doctors can see the profiles they created for their patients from this page. They can edit patient information, share patients' profile with all users and initilialize *Matcher* program for a patient.

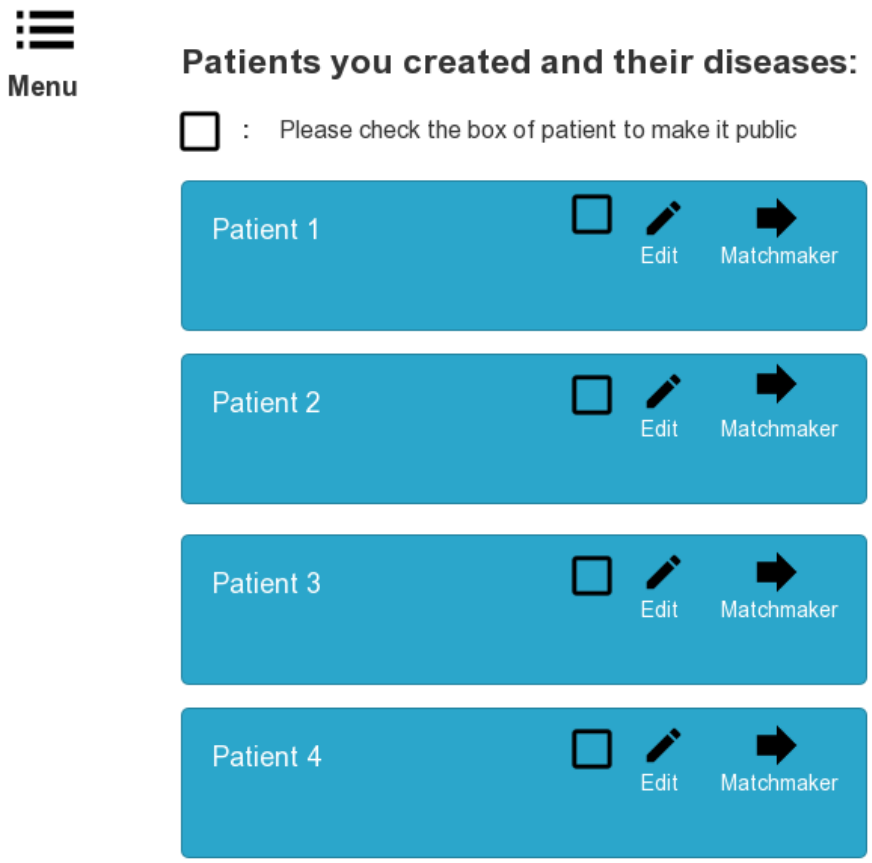


Figure 10: Mock-up for Management of Patients Page

3.5.5.4 Editing a Patient

Medical doctors can edit all the information for the patient they entered to system and their state of privacy. Also they can add disease for the patient and give information about the disease by filling pre-defined fields.

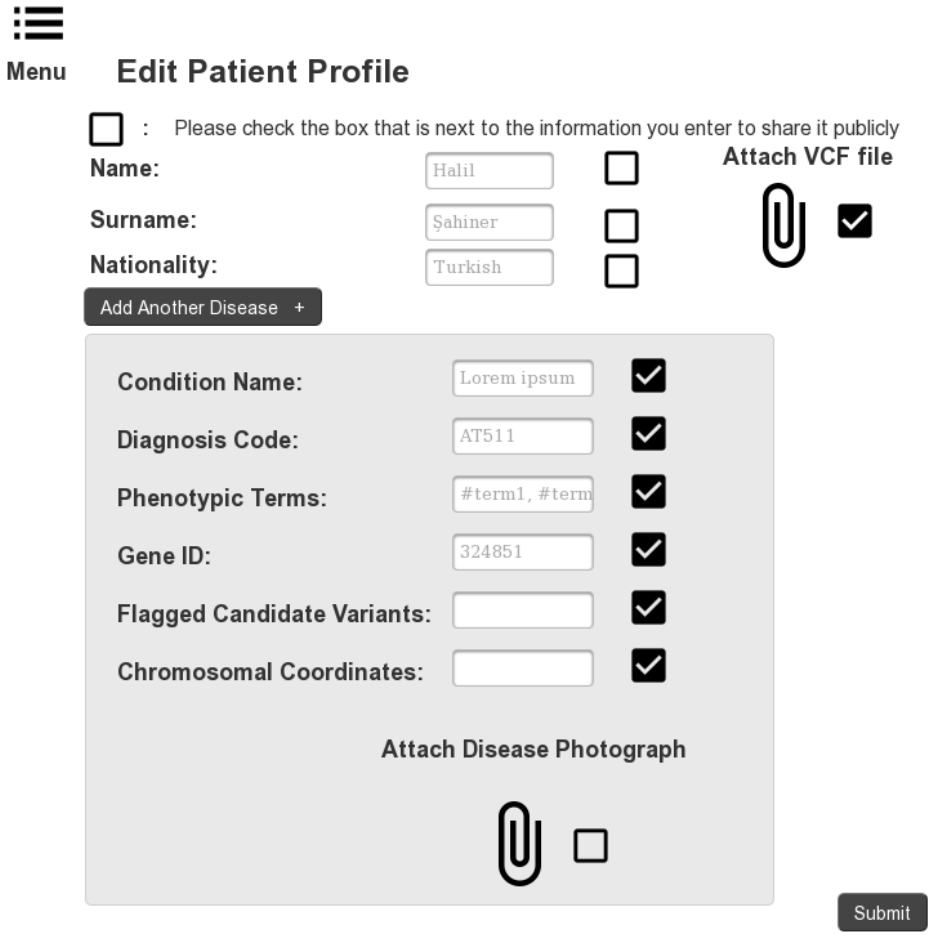


Figure 11: Mock-up for Editing of a Patient Profile Page

3.5.5.5 Matching Patients

After matchmaker button is selected for a patient in the management of patients page, this page will appear. To run matching algorithm, one will choose the disease of the patient, the databases to search for other patients and the algorithm type then click the "Start Matching" button to start it. A table of patients with their match percentage and the information of the patient according to algorithm chosen will appear. When they click on one of the rows, a pop-up will appear, asking whether the user wants to send an email to the owner of that patient's profile to communicate over the match.

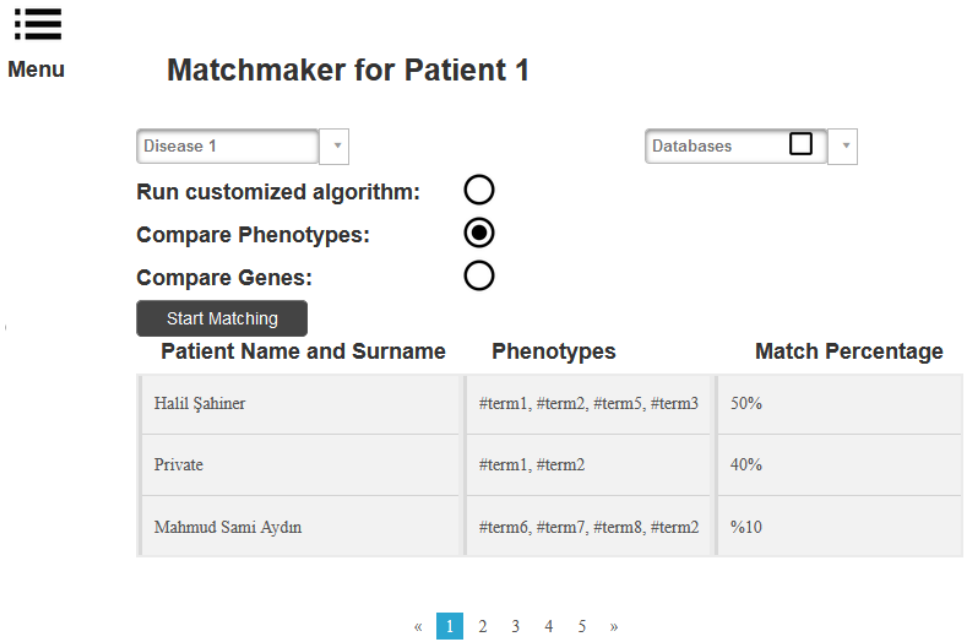
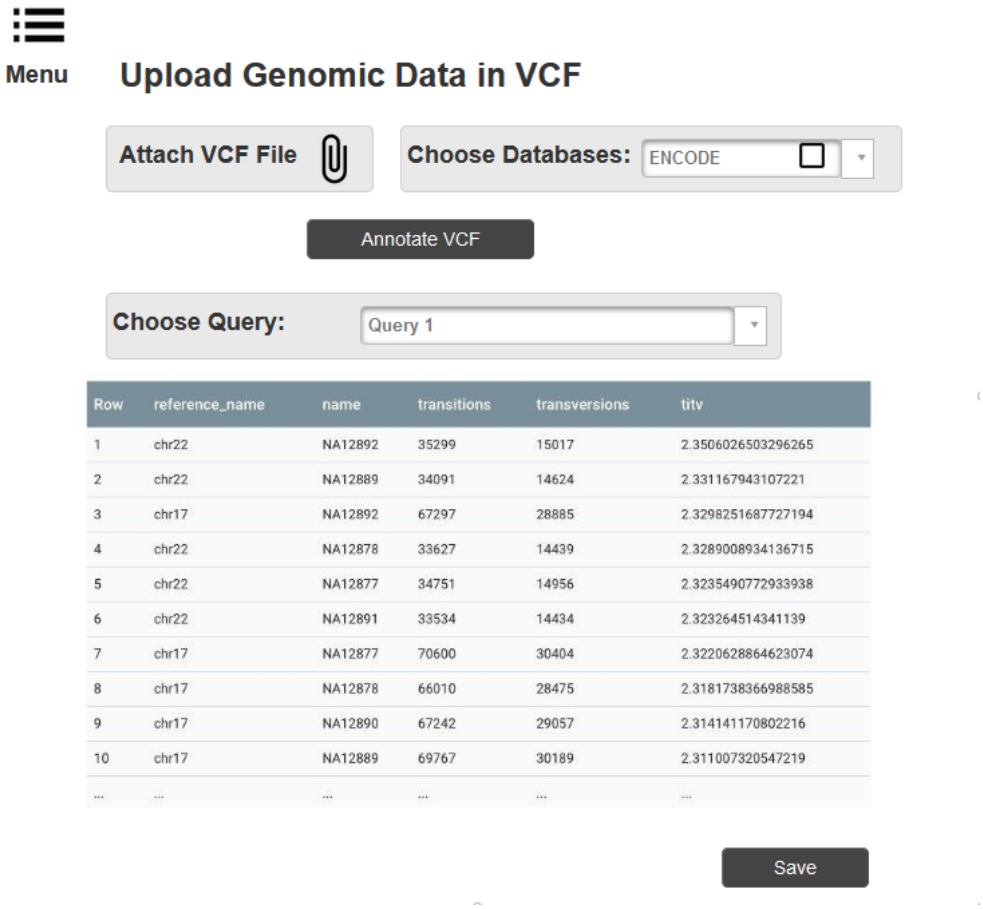


Figure 12: Mock-up for Running Matcher Algorithms

3.5.5.6 Uploading, Annotating and Running Query on a VCF

Medical doctors will upload VCF files to do querying and analysis on it from this page. They will attach a VCF file, choose a database to run query on it, annotate the VCF file to prepare for querying and choose a query that is currently on LIBRA database. Then, a table will appear with query results according to the query they chose. If they want, they can save this table in their intended format.



Row	reference_name	name	transitions	transversions	titv
1	chr22	NA12892	35299	15017	2.3506026503296265
2	chr22	NA12889	34091	14624	2.331167943107221
3	chr17	NA12892	67297	28885	2.3298251687727194
4	chr22	NA12878	33627	14439	2.3289008934136715
5	chr22	NA12877	34751	14956	2.3235490772933938
6	chr22	NA12891	33534	14434	2.323264514341139
7	chr17	NA12877	70600	30404	2.3220628864623074
8	chr17	NA12878	66010	28475	2.3181738366988585
9	chr17	NA12890	67242	29057	2.314141170802216
10	chr17	NA12889	69767	30189	2.311007320547219
...	...	...	...	...	...

Figure 13: Mock-up for Uploading, Annotating and Querying Variants

4 Other Analysis Elements

4.1 Consideration of Various Factors

In this section, various factors that may effect the analysis and design of the project will be discussed.

4.1.1 Public Health

The system is designed in a way that it handles any unexpected hardware/network failure while annotating genetic variants. No patient will be diagnosed with the wrong disease since that may impose a health risk.

4.1.2 Public Safety

The system will make sure that no wrong/incomplete results are displayed to the doctors, since that may impose a safety risk to the patients of those doctors.

4.1.3 Global Factors

The nationalities of patients should be considered because it is important for interpretation of genetic mutations. Also, it would be beneficial to make patient queries between different nations for increasing the chance of matching. Yet, it is hard to design such a system because of the strict regulations between different nations.

4.1.4 Social Factors

The privacy of the patients should be protected. The system should ensure that the sensitive data of the patients cannot be shared without the patients' consent. In the case where the patient data is shared, the system should ensure that the data cannot be traced back to the patient. The system should be designed in a way that protects patients against discrimination in their professional or social life.

Factors	Effect Level	Effect
Public Health	10	The system handles any unexpected hardware/network failure while annotating genetic variants. No patient will be diagnosed with the wrong disease.
Public Safety	9	The system will make sure that no wrong/incomplete results are displayed to the doctors.
Public Welfare	0	
Global Factors	7	The nationalities of patients should be considered because it is important for interpretation of genetic mutations. Also, It would be beneficial to make patient queries between different nations for increasing the chance of matching.
Cultural Factors	0	
Social Factors	8	The privacy of the patients should be protected. The system should ensure that the sensitive data of the patients cannot be shared without the patients' consent. In the case where the patient data is shared, the system should ensure that the data cannot be traced back to the patient. The system should be designed in a way that protects patients against discrimination in their professional or social life.

Table 1: Summary of Consideration of Various Factors

4.2 Risks and Alternatives

4.2.1 VCF Preprocessing overhead

Most of the VCF files contain huge amounts of information and preprocessing before genetic annotation can be done. This overhead may cause a huge load on the main application server if clients wants to upload custom VCFs. This load will affect the user experience of the genetic filtering and analysis tool negatively. If this scenario occurs, our plan is to obtain a dedicated server for handling this load.

The likelihood of this risk is high.

4.2.2 Low Incentive for Participating Hospitals

The hospitals may not want to participate in the matchmaking system based on the current privacy and security constraints. This will decrease the effectiveness of the matchmaking system because of the decrease in the number of patients in the system.

The likelihood of this risk is moderate. Several additional security constraints can be added in order to increase hospital participation.

4.2.3 Duplicate Patients

Usually, each patient will register to a hospital using their national ID or social security number. Yet, it is not possible to contain this information in the matchmaking system because of the privacy requirements. Hence, the same patient can be registered to the system in two hospitals which will result in an automatic match. This will naturally increase the false positive rate of the matchmaking system.

The likelihood of this risk is low. One obvious solution is to use unique identification for patients but since it is going to decrease patient anonymity this risk will go through further elaboration.

Risk	**Likelihood**	**Effect**	**B Plan**
VCF Preprocessing Overhead	High	Decrease responsiveness	Use a dedicated server for preprocessing
Low Incentive for Participating Hostpitals	Moderate	Decrease overall effectiveness of Matchmaking System	Enforce more privacy constraints
Duplicate Patients	Low	Increase False Positives in Matchmaking Systems	Decrease Anonymity

Table 2: Summary of Risks and Alternatives

4.3 Project Plan

4.3.1 Work Packages

WP	Work Package Title	Leader	Members Involved
WP1	Genetic Filtering and Analysis Back-End	Halil Şahiner	Berke Egeli Abdullah Talayhan
WP2	Genetic Filtering and Analysis Front End	Naisila Puka	Mahmud Sami Aydın Halil Şahiner
WP3	Query Builder	Abdullah Talayhan	Halil Şahiner Berke Egeli
WP4	Matchmaker Front End	Berke Egeli	Abdullah Talayhan Mahmud Sami Aydın
WP5	Database Sharing	Halil Şahiner	Mahmud Sami Aydın Abdullah Talayhan
WP6	Creation and Management of Profiles	Mahmud Sami Aydın	Naisila Puka Berke Egeli
WP7	Matching Patients	Naisila Puka	Halil Şahiner Abdullah Talayhan

Table 3: List of Work Packages

<table>
<tr><td colspan="4">WP 1: Genetic Filtering and Analysis Back-End</td></tr>
<tr><td colspan="4">Start date: 11/11/2019 End date: 12/01/2020</td></tr>
<tr><td>Leader:</td><td>Halil Şahiner</td><td>Members involved:</td><td>Berke Egeli
Abdullah Talayhan</td></tr>
<tr><td colspan="4">Objectives: Implementation of several functionalities such as genetic annotations, filtering, querying. This package will include the core algorithms for genetic analysis.</td></tr>
<tr><td colspan="4">Tasks:
Task 1.1 Genetic Annotation: Annotation of VCF Files. This includes preprocessing and loading
Task 1.2 Query Filtering: Obtaining results from databases based on the query and several predefined filters.
Task 1.3 Genetic Analysis Tools: Several analysis algorithms. For example, identifying mendelian error or de_novo mutations.</td></tr>
<tr><td colspan="4">Deliverables
D1.1: LIBRA Genetic analysis and filtering API
D1.2: User's Manual
D1.3: Documentation</td></tr>
</table>

WP 2: *Genetic Filtering and Analysis Front End*			
Start date: *11/12/2019* **End date:** *17/01/2020*			
Leader:	*Naisila Puka*	**Members involved:**	*Mahmud Sami Aydın* *Halil Şahiner*
Objectives: *Implementation of the UI for the Genetic Filtering and Analysis package except the Query Builder component. The UI will include the registration system for the doctors, database selection UI component, storing and annotating genomic data in order to query variants and explore the data.*			
Tasks: ***Task 2.1 Implement the skeleton of the UI*:** *This task consists of implementing the UI to accommodate for registration, storage and annotation of data and genetic database selection UI.* ***Task 2.2 Connect the UI with the API*:** *This task consists of connecting the storage and annotation of data UI component with the LIBRA genetic analysis and filtering API.*			
Deliverables ***D2.1:*** *Functional UI of the Genetic Filtering and Analysis package of the project LIBRA except the Query Builder* ***D2.2:*** *User's Manual* ***D2.3:*** *Documentation*			

WP 3: *Query Builder*			
Start date: *25/12/2019* **End date:** *20/02/2020*			
Leader:	*Abdullah Talayhan*	**Members involved:**	*Halil Şahiner* *Berke Egeli*
Objectives: *Implementation of the Query Builder UI and functionality.*			
Tasks: ***Task 3.1 Implement the skeleton of the UI*:** *This task consists of implementing the Query Builder UI where doctors will be able to specify queries based on self-determined constraints through using a UI. The UI does not require the user to know any database specific syntax.* ***Task 3.2 Connect the UI with the API*:** *This task consists of connecting the UI to the LIBRA Genetic analysis and filtering API to add querying functionality to it.*			
Deliverables ***D3.1:*** *A functional UI for the Query Builder* ***D3.2:*** *User's Manual* ***D3.3:*** *Documentation*			

WP 4: *Matchmaker Front End*			
Start date: *25/02/2020* **End date:** *26/04/2019*			
Leader:	*Berke Egeli*	**Members involved:**	*Abdullah Talayhan* *Mahmud Sami Aydın*
Objectives: *Implementation of the front end for the Patient Matching Platform. The UI will include the patient profile creation site, components for deciding which information of the patient will be shared with the platform, patient profile management components and a professional search component equipped with advanced search depictions.*			
Tasks: ***Task 4.1 Implement the skeleton of the UI*:** *This task consists in fulfilling all of the objectives explained. The different components of the user interface (patient profile creation component, profile management component etc) will be implemented separately and will be joined at the end.* ***Task 4.2 Connect the UI with the API*:** *This task consists of connecting the UI to the patient matching functionality.*			
Deliverables ***D4.1:*** *Patient profile creation UI component* ***D4.2:*** *Information sharing decision UI component* ***D4.3:*** *Patient profile management UI component* ***D4.4:*** *Patient matching UI component* ***D4.5:*** *User's Manual* ***D4.6:*** *Documentation*			

WP 5: *Database Sharing*			
Start date: *10/02/2020* **End date:** *15/04/2020*			
Leader:	*Halil Şahiner*	**Members involved:**	*Mahmud Sami Aydın* *Abdullah Talayhan*
Objectives: Providing support for custom databases.			
Tasks: ***Task 5.1 Importing Databases*:** *Importing custom databases to the system and their integration.* ***Task 5.2 Database Security Filters*:** *Setting the constraints for the public and private fields for a database.*			
Deliverables ***D5.1:*** *Database Migration Tool.* ***D5.2:*** *User's Manual* ***D5.3:*** *Documentation*			

WP 6: *Creation and Management of Profiles*			
Start date: *15/02/2020* **End date:** *15/03/2020*			
Leader:	*Mahmud Sami Aydın*	**Members involved:**	*Naisila Puka* *Berke Egeli*
Objectives: *Implementation of doctor and patient profiles and necessary functionalities for their maintenance.*			
Tasks: ***Task 6.1 Creation of profiles*:** *Parsing initial information related to the profile.* ***Task 6.2 Integration to matchmaking system*:** *This will be necessary for selecting what kind of information to share while registering a patient to the matchmaking system.* ***Task 6.3 Maintenance of profiles*:** *Update and delete functionalities for profiles.*			
Deliverables ***D6.1:*** *Patient Account Management System* ***D6.2:*** *Matchmaker account integration system.* ***D6.3:*** *User's Manual* ***D6.4:*** *Documentation*			

WP 7: *Matching Patients*			
Start date: *16/03/2020* **End date:** *10/05/2019*			
Leader:	*Naisila Puka*	**Members involved:**	*Abdullah Talayhan* *Halil Şahiner*
Objectives: *Obtain the functionalities of a patient social network, controlled and used by doctors who seek similar genetic profiles related to a specific disease in order to understand and diagnose the disease further.*			
Tasks: ***Task 7.1 Basic Search Feature*:** *Implement the basic search feature which will have the function of searching for similar patients based on customized attributes of the genomic profile of the patient of interest.* ***Task 7.2 Advanced Search Feature*:** *Since the system is not restricted on the ways of querying patients, the functionality of a more advanced search feature will be implemented. In this type of search, users can also run customized filters on the matching results in terms of information points shared through the databases to focus on different groups of patients.*			
Deliverables ***D7.1:*** Basic Search API ***D7.2:*** *Advanced Search API* ***D7.3:*** *User's Manual* ***D7.4:*** *Documentation*			

4.3.2 Gantt Chart

Name	Duration	Start	Finish
Genetic Filtering and Analysis Back-End	46 days	11/11/2019	01/13/2020
Genetic Filtering and Analysis Front End	28 days	12/11/2019	01/17/2020
Query Builder	42 days	12/25/2019	02/20/2020
Matchmaker Front End	44 days	03/03/2020	05/01/2020
Database Sharing	48 days	02/10/2020	04/15/2020
Creation and Management of Profiles	20 days	02/17/2020	03/13/2020
Matching Patients	41 days	03/06/2020	05/01/2020

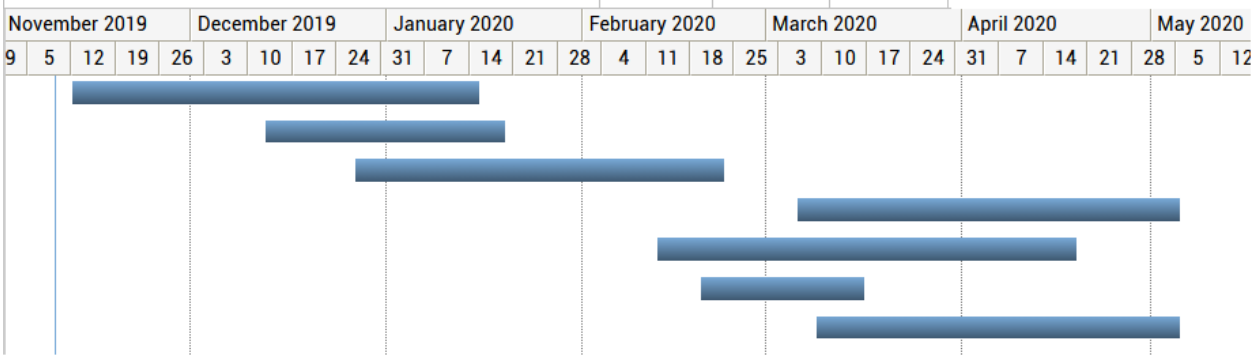


Figure 14: Gantt Chart (Date format: M/D/Y)

4.4 Ensuring Proper Team-Work

The team work will be established using *Github*. The procedure for collaboratively completing tasks is as follows:

1. Github Issues will be assigned to each member for a certain task.
2. After the completion of an issue, the code will be reviewed by at least one of the other members that are currently enrolled in the work package that covers the scope of the task.
3. Based on the feedback of Step 2, new issues will be assigned.

The distribution of issues for each member may differ based on the magnitude of the task. This load balancing will be observed by the leader of the work package that the issue belongs.

In addition, we use a *Telegram* group for exchanging general information and setting up meetings. Meetings include both voice calls and physical gatherings.

4.5 Ethics and Professional Responsibilities

There are not many commercial products like this project. Similar products exist to meet research demands. The existing products are either provided as additional services or are sold at high prices. Our project aims to provide a ubiquitous and uniform service to hospitals around Turkey to allow diagnosis, research and treatment of diseases, especially rare ones. Privacy of the patients is important. Certain genetic variations and diseases can lead to the discrimination against the patients in their social and professional lives. Therefore, it is imperative that the privacy of the patients is protected. Data will not be utilized other than the intended purposes of LIBRA. The doctors will filter what they want to share regarding their patients. The information shared between doctors in the platform will be restricted to ensure only the permitted data will be displayed.

4.6 New Knowledge and Learning Strategies

We are planning to make use of online resources as needed. One specific example is the documentation for the current system *Gemini* that we are planning to make a better version of.

We use online tutorials and documentations (Flask, Django, React JS etc.) for having an equal know-how related to web development over all members.

We have already attended a seminar at *Intergen* related to genetic diagnosis procedures. Further questions related to overall user experience and functionalities can be discussed with industry experts.

5 Glossary

- **Variant Call Format(VCF):** The Variant Call Format specifies the format of a text file used in bioinformatics for storing gene sequence variations.
- **Genetic Annotation:** Genetic Annotation is the process of identifying the locations of genes so that it serves as an explanation about the functionality of genes.
- **Genotype:** The genetic constitution of an individual organism.
- **Phenotype:** The set of observable characteristics of an individual resulting from the interaction of its genotype with the environment.
- **Mendelian Error:** A Mendelian error in the genetic analysis of a species, describes an allele in an individual which could not have been received from either of its biological parents by Mendelian inheritance.
- **De Novo Mutation:** A genetic alteration that is present for the first time in one family member as a result of a variant (or mutation) in a germ cell (egg or sperm) of one of the parent.

References

[1] J. E. Stajich and H. Lapp, "Open source tools and toolkits for bioinformatics: significance, and where are we?" *Briefings in Bioinformatics*, vol. 7, no. 3, pp. 287–296, Sep. 2006. [Online]. Available: https://doi.org/10.1093/bib/bbl026

[2] "Crying without tears unlocks the mystery of a new genetic disease - scope," Mar. 2014. [Online]. Available: https://scopeblog.stanford.edu/2014/03/20/crying-without-tears/

[3] U. Paila, B. A. Chapman, R. Kirchner, and A. R. Quinlan, "GEMINI: Integrative exploration of genetic variation and genome annotations," *PLoS Computational Biology*, vol. 9, no. 7, p. e1003153, Jul. 2013. [Online]. Available: https://doi.org/10.1371/journal.pcbi.1003153

[4] O. J. Buske, F. Schiettecatte, B. Hutton, S. Dumitriu, A. Misyura, L. Huang, T. Hartley, M. Girdea, N. Sobreira, C. Mungall, and M. Brudno, "The matchmaker exchange API: Automating patient matching through the exchange of structured phenotypic and genotypic profiles," *Human Mutation*, vol. 36, no. 10, pp. 922–927, Sep. 2015. [Online]. Available: https://doi.org/10.1002/humu.22850

[5] R. M. Layer, N. Kindlon, K. J. Karczewski, and A. R. Quinlan, "Efficient genotype compression and analysis of large genetic-variation data sets," *Nature Methods*, vol. 13, no. 1, pp. 63–65, Nov. 2015. [Online]. Available: https://doi.org/10.1038/nmeth.3654

[6] S. Köhler, L. Carmody, and N. V. et al., "Expansion of the human phenotype ontology (hpo) knowledge base and resources," *Nucleic Acids Research*, vol. 47, no. D1, pp. D1018–D1027, Nov. 2018. [Online]. Available: https://doi.org/10.1093/nar/gky1105

[7] "comp_hets: Identifying potential compound heterozygotes." [Online]. Available: https://gemini.readthedocs.io/en/latest/content/tools.html#comp-hets-identifying-potential-compound-heterozygotes

[8] "Cloud computing services — google cloud." [Online]. Available: https://cloud.google.com

[9] "Kişisel verilerin korunması kanunu," Apr. 2016. [Online]. Available: https://www.mevzuat.gov.tr/MevzuatMetin/1.5.6698.pdf