Submission 3

CSC490 Software Engineering

Nehmat Touma
nehmat.touma@lau.edu
201600300

Mahdi Hallal
mahdi.hallal@lau.edu
201906289

Pia Chouaifaty
pia.chouaifaty@lau.edu
201504706

Presented to Dr. Karim Youssef

LAU
الجامعة اللبنانية الأميركية
Lebanese American University

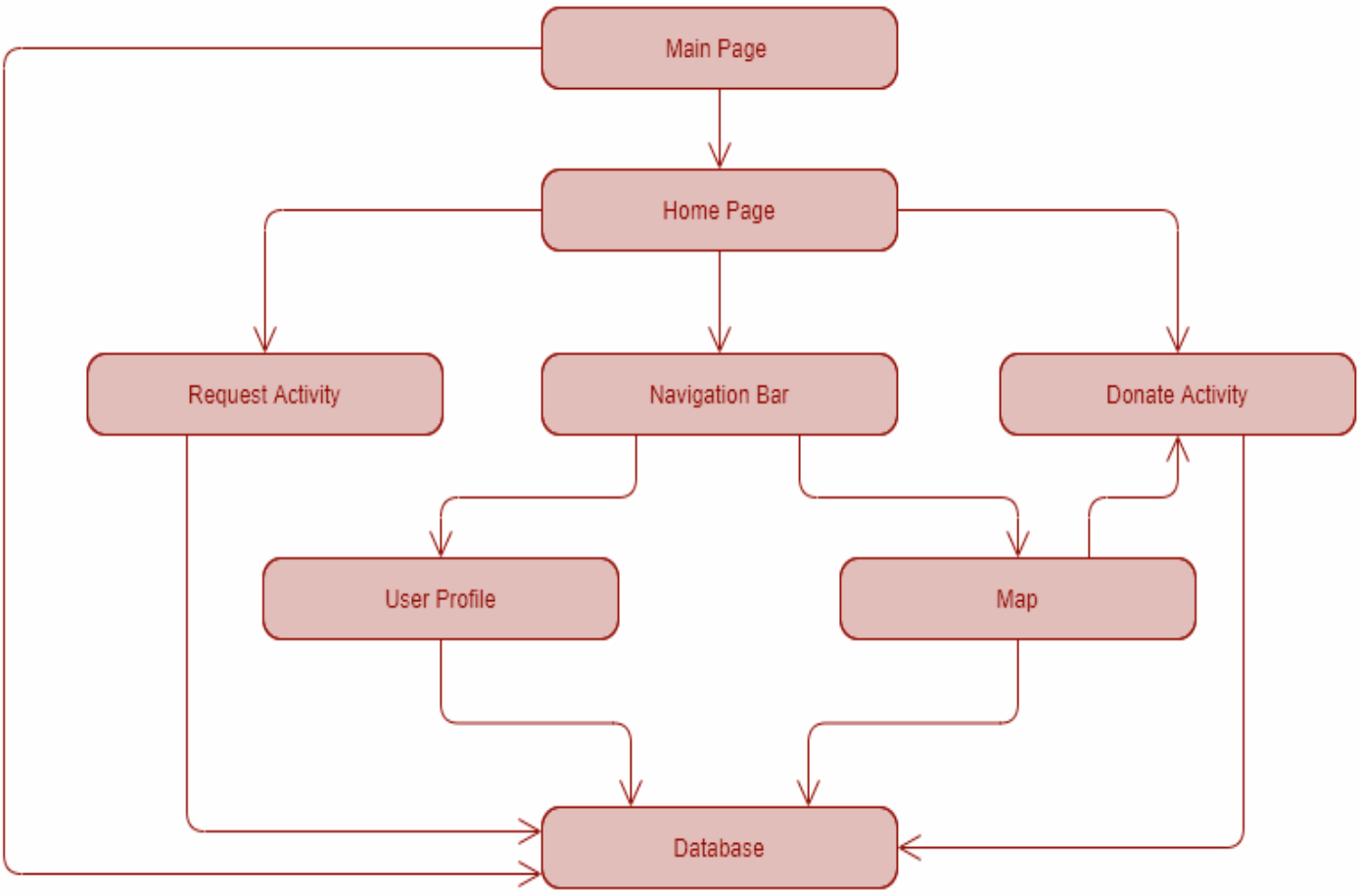
Table of Contents

Updates from Submission 2

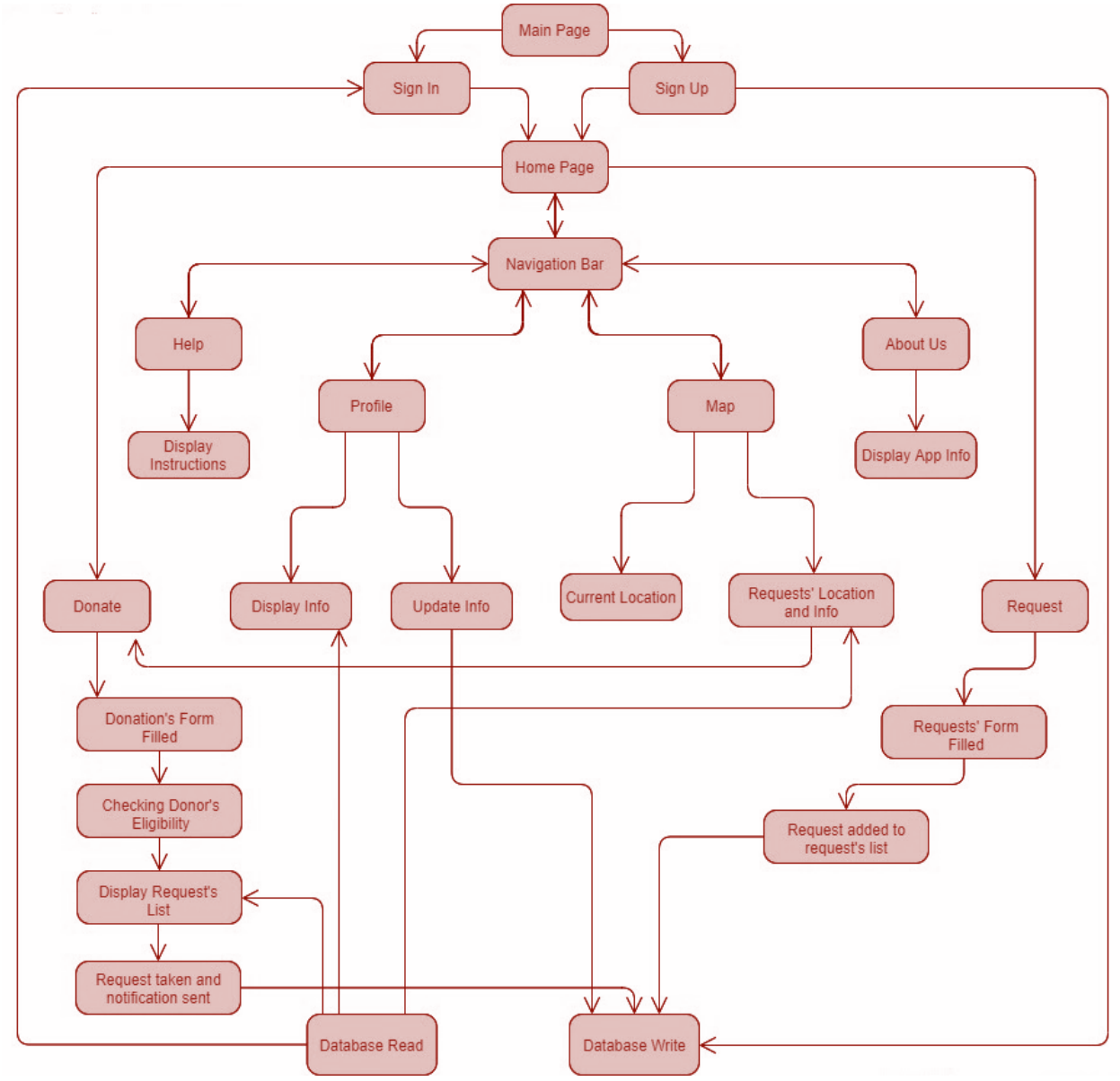
- The map option for users to locate hospitals where blood is needed is set for future implementation
- The sign-in and sign-up are to be developed to be more user-friendly
- The Donate process is to be developed further
- Given that no server is present in the current implementation, the notification options – whereby users receive a notification that their blood is needed, or that a patient will receive the needed blood – will be set for future implementation
- The app was implemented for Android devices, keeping the iOS devices for future developments

1. System Architecture

1.1. Subsystems Model



1.2. Module Architecture

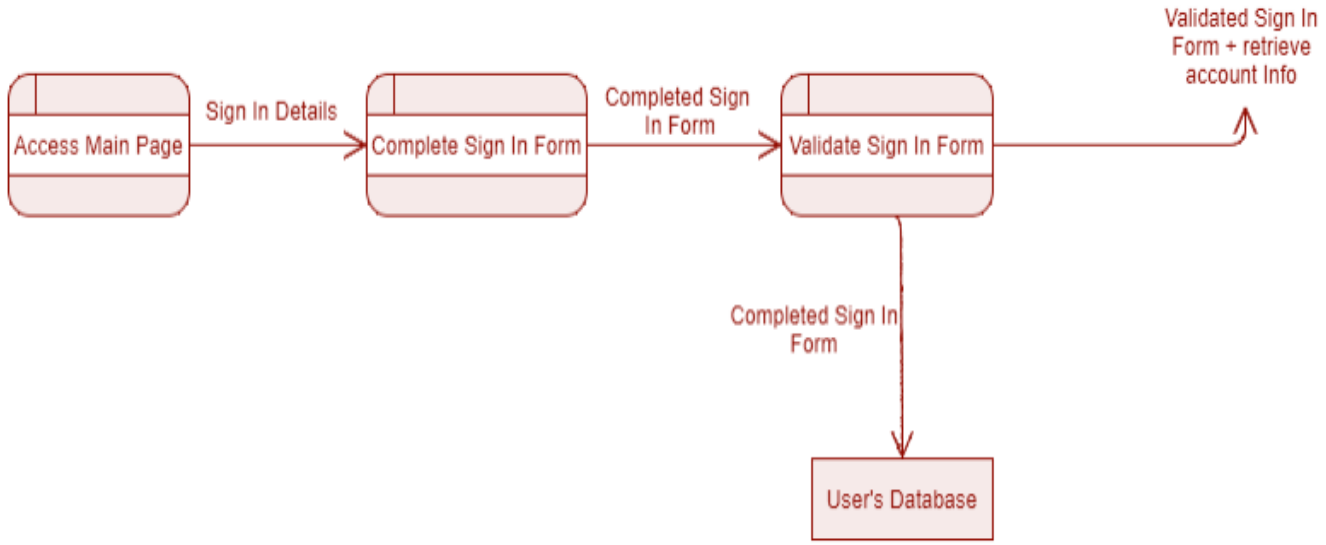


1.3. Subsystems Description

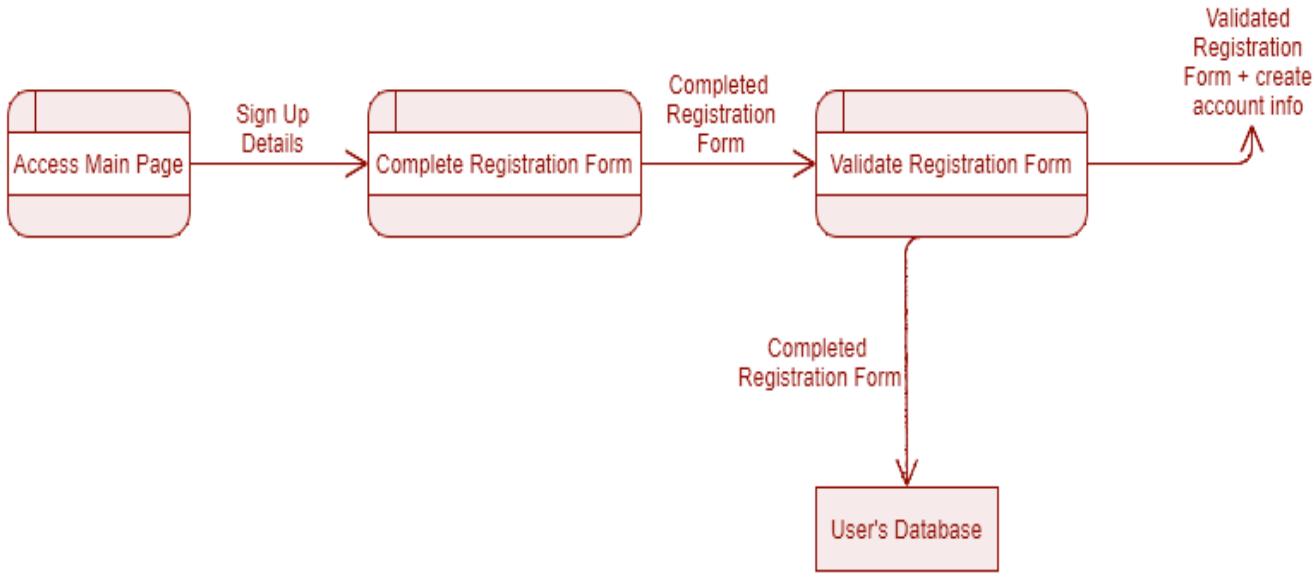
Sub-System	Description
Main Page	Contains a Sign Up functionality that first-time users can use to set up their accounts and access the application services. Has a Sign In functionality where users who already signed up can use their username and password to access the application services. Displays the logo of the application. Has access to the database of all users.
Home Page	Contains a navigation bar to help users traverse through all the services of the application. Shows a donate button that users can click on if they are willing to donate blood. Has a request button that users can click on if they want to issue a request for blood donation.
Navigation Bar	Contains a Home Icon that users can click on to go back to the home page whenever needed. Contains an About Us Icon that users can click on to view general information about the application with contact details. Contains a Help Icon that users can click on to see instructions about how to make a donation and how to request a blood donation if needed. Has a map Icon that users can click on to access the special Map functionality. Has a User Profile Icon that users can click on to access their profiles.
Donate Activity	Contains a donor registration form that users have to fill before making any donation. The form contains information that reflects if the donor is eligible to make a safe donation or not. Has a list of all blood donation requests that are of the same blood type as that entered in the blood registration form. Each blood donation request contains specific information about the request. Eligible users can view each request and take the requests that they desire. Once a donor takes a request then a notification containing donor information and confirming the donation will be sent to the user that issued the blood donation request. Has access to the database that contains information about all users and all requests.
Request Activity	Contains a request form that each user that wants to make a request has to fill to provide proper information about his/her blood donation request. After the user confirms his/her request, the request will be added to the list of blood donation requests. Has access to the database that contains all requests' information.
Map	Contains the current location of the user. Contains the location of all the requests that have the same blood type as that of the user. Users can click on a specific location to see more details and decide if they are willing to donate there. The app will take the user to the Donate Activity to complete the donation process. Has access to the database that contains information about all requests.
User Profile	Displays all the information that the user entered when first signing up to use the application services. Gives the users the option to update their information whenever needed. Has access to the database that contains information about all the users.
Database	Contains a database for users with all their basic information. Contains a database for requests with all the information needed for a complete request.

1.4 DataFlow Diagrams

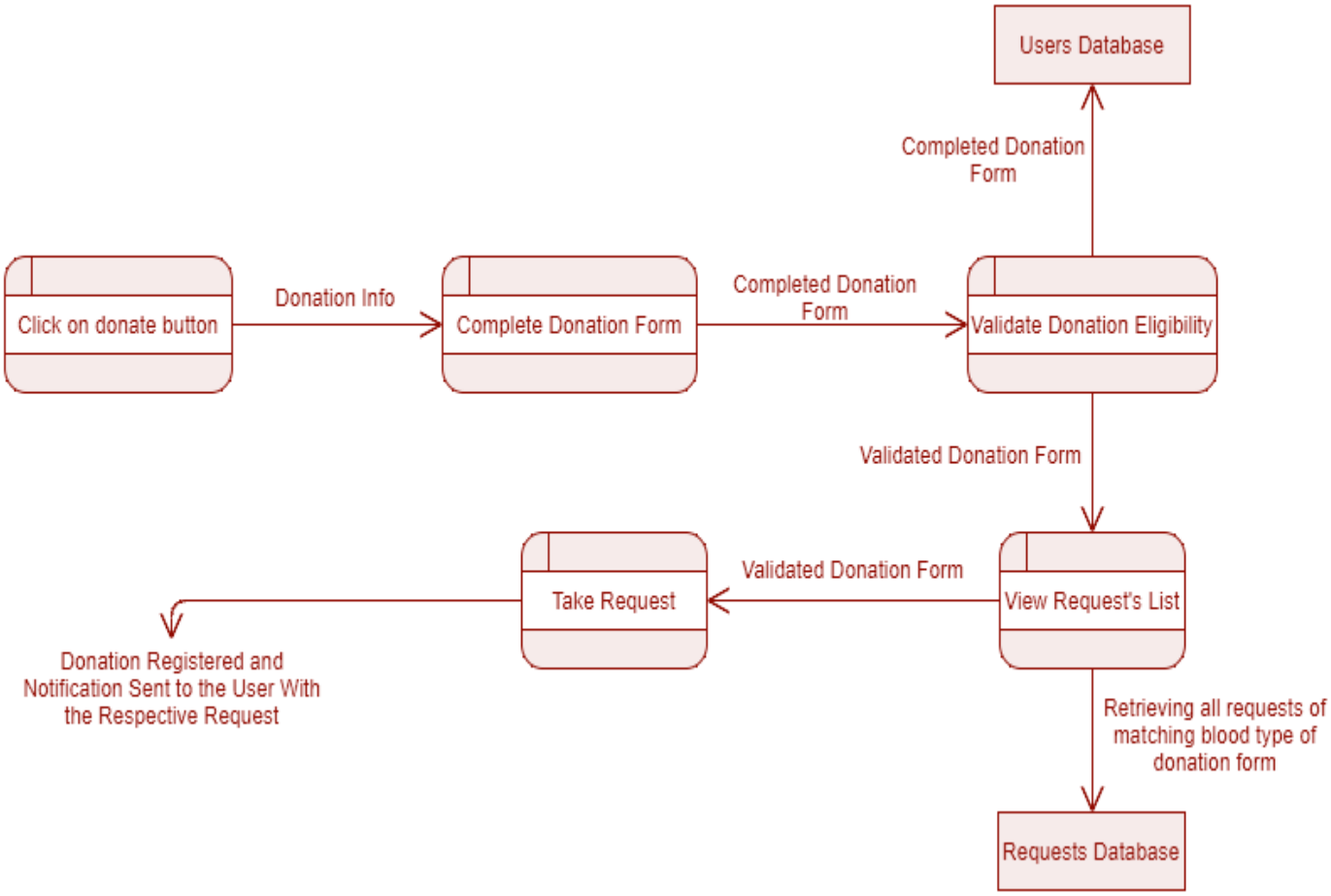
1.4.1. Sign In



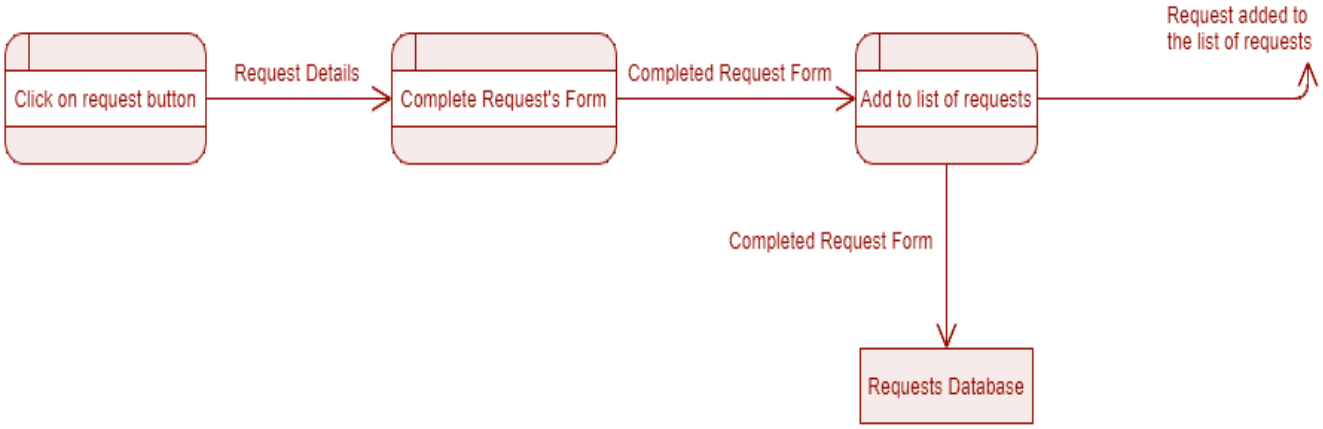
1.4.2. Sign Up



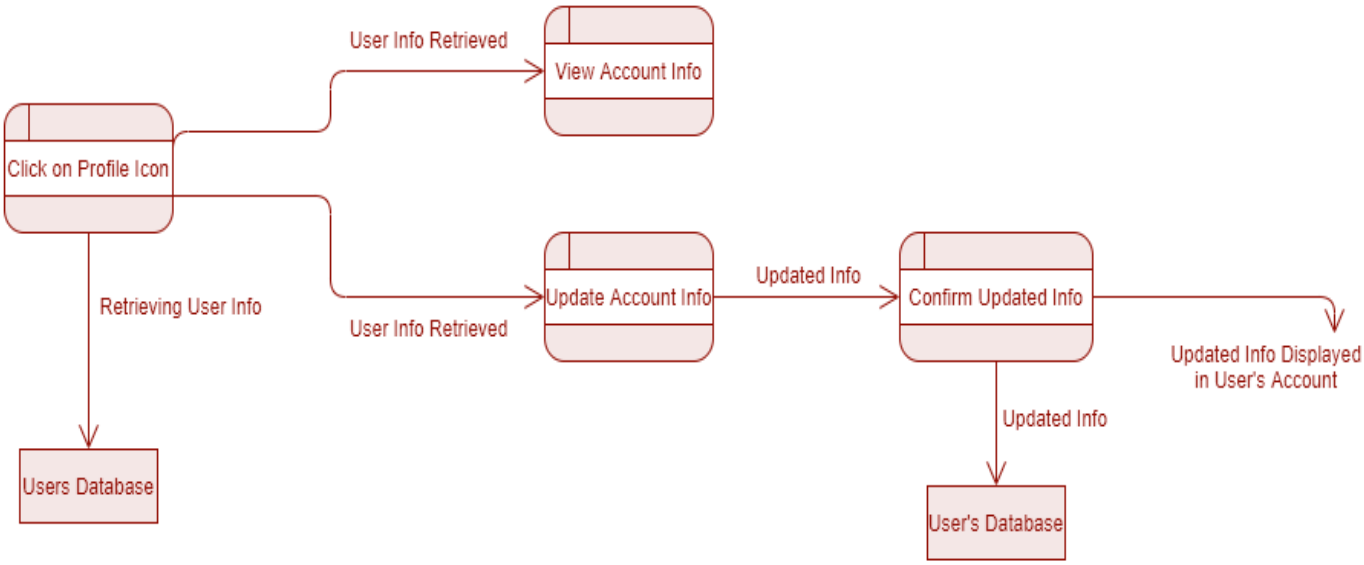
1.4.3. Donate Blood



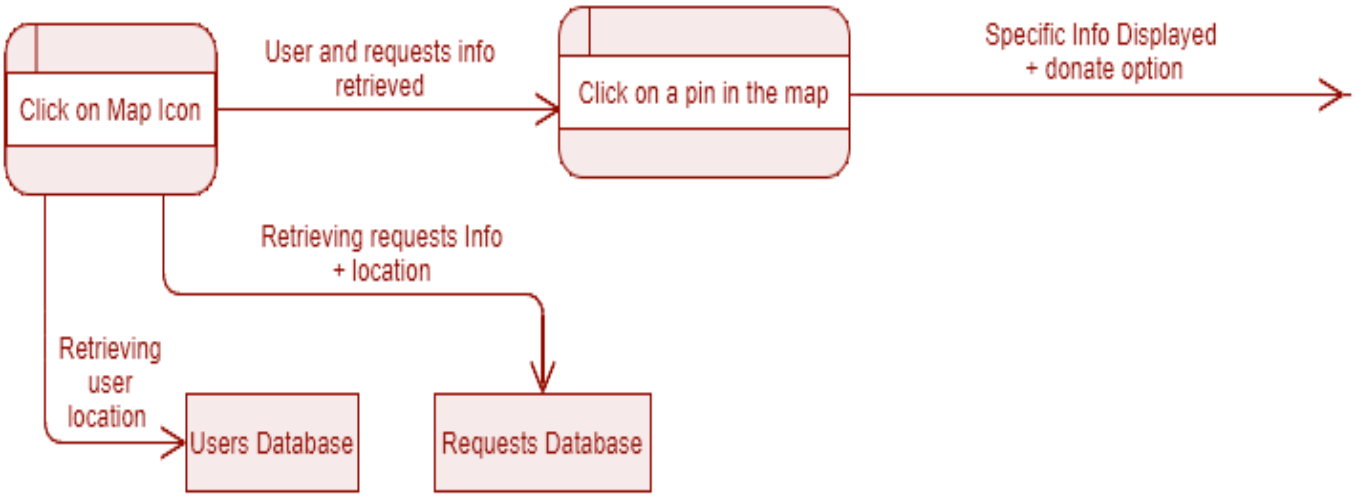
1.4.4. Request Blood Donation



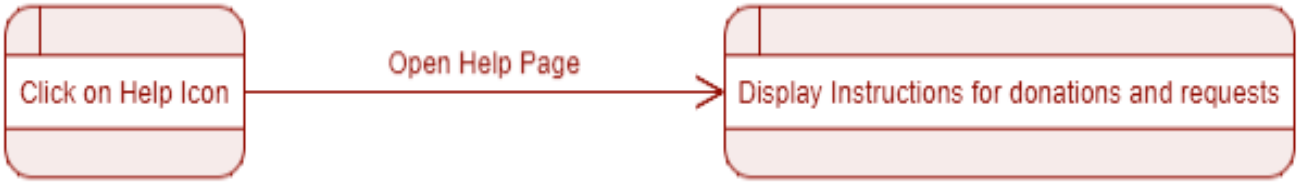
1.4.5. User Profile Activity



1.4.6. Map Activity

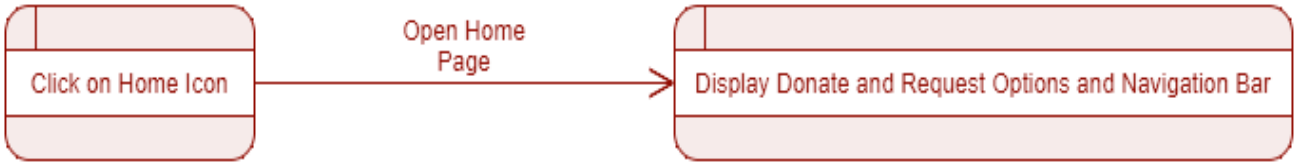


1.4.7. Help Activity



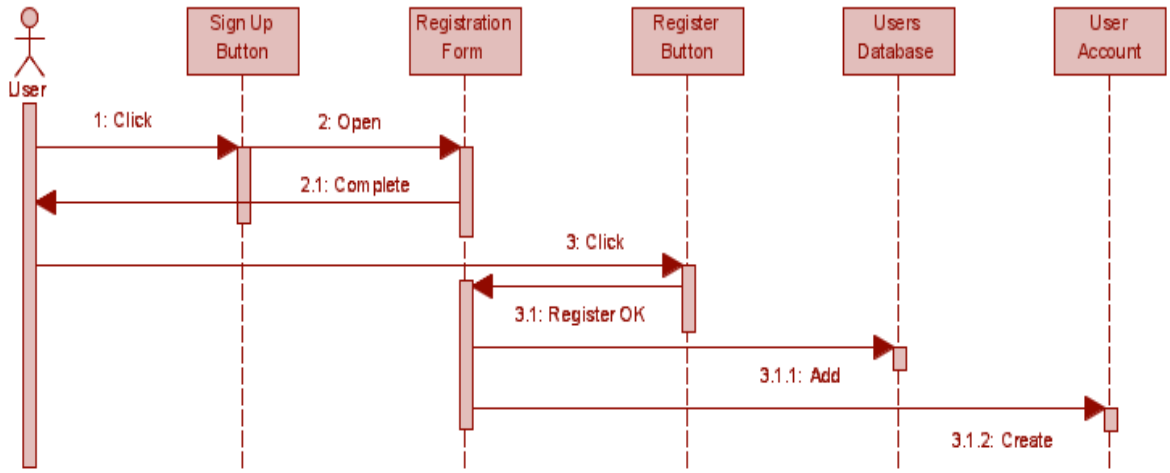
1.4.8. About Us Activity

1.4.9 Home Activity

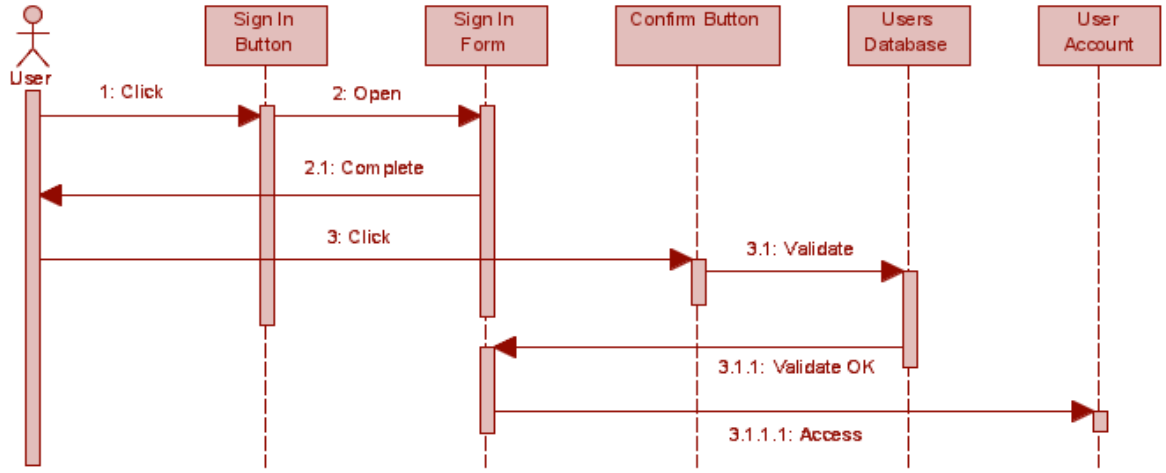


1.5 Sequence Diagrams

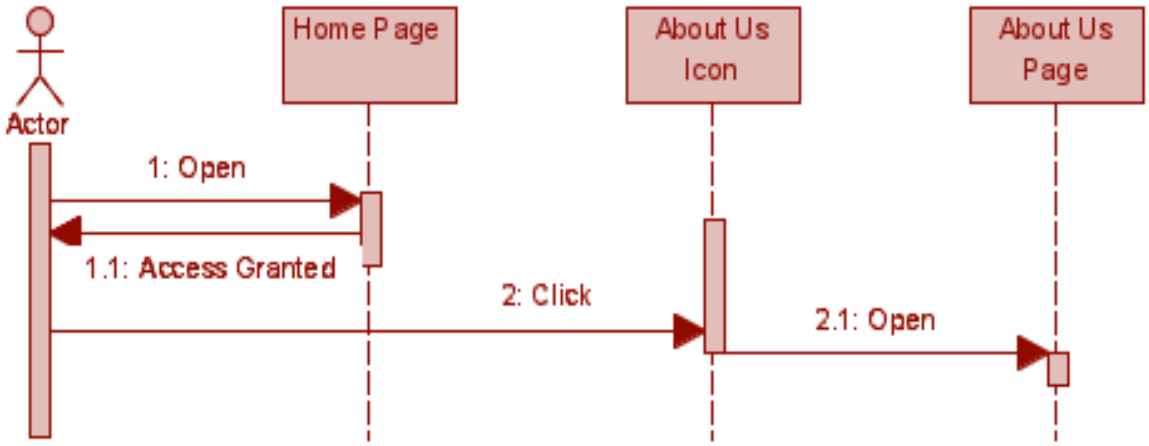
1.5.1. Sequence Diagram For Sign Up



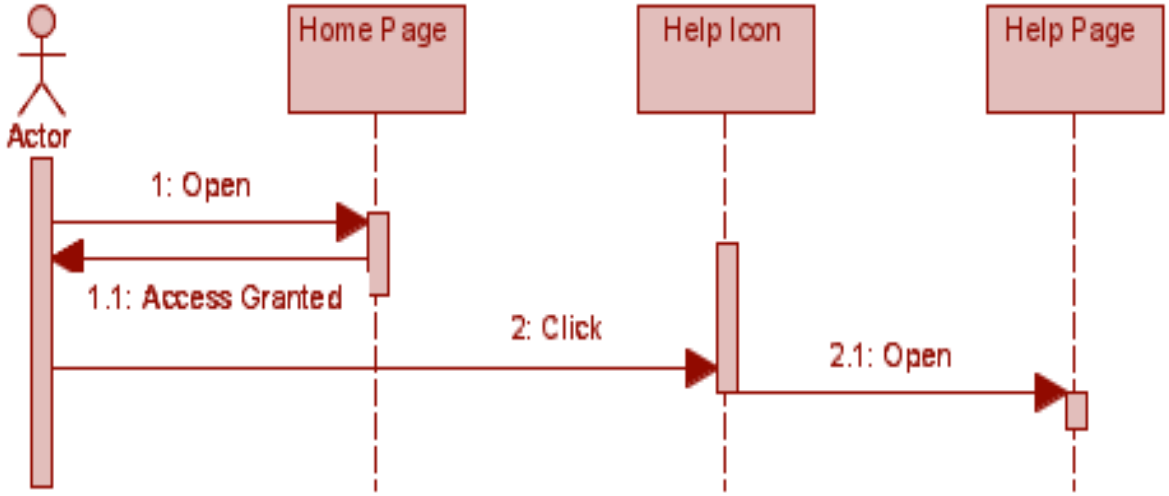
1.5.2 Sequence Diagram for Sign In



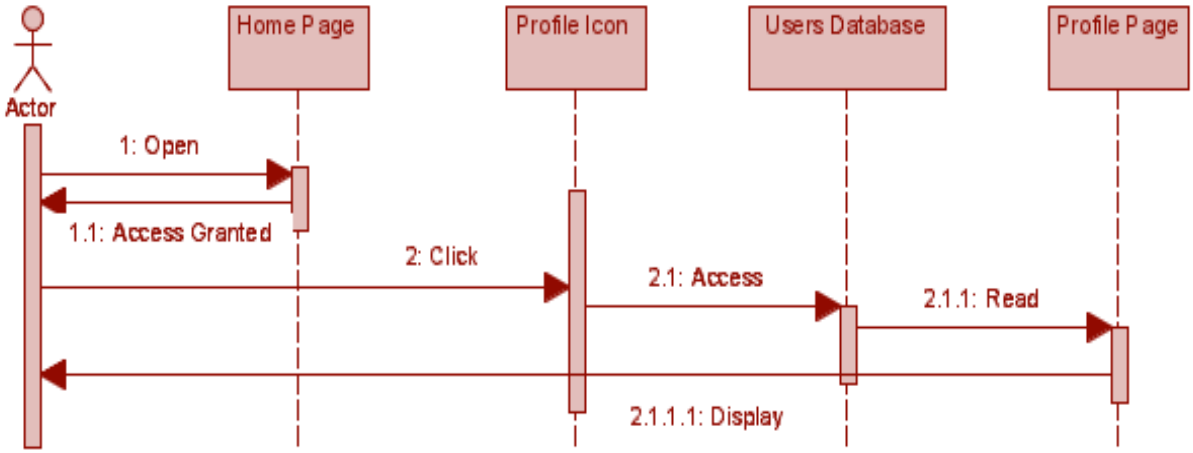
1.5.3 Sequence Diagram For About Us Page Access



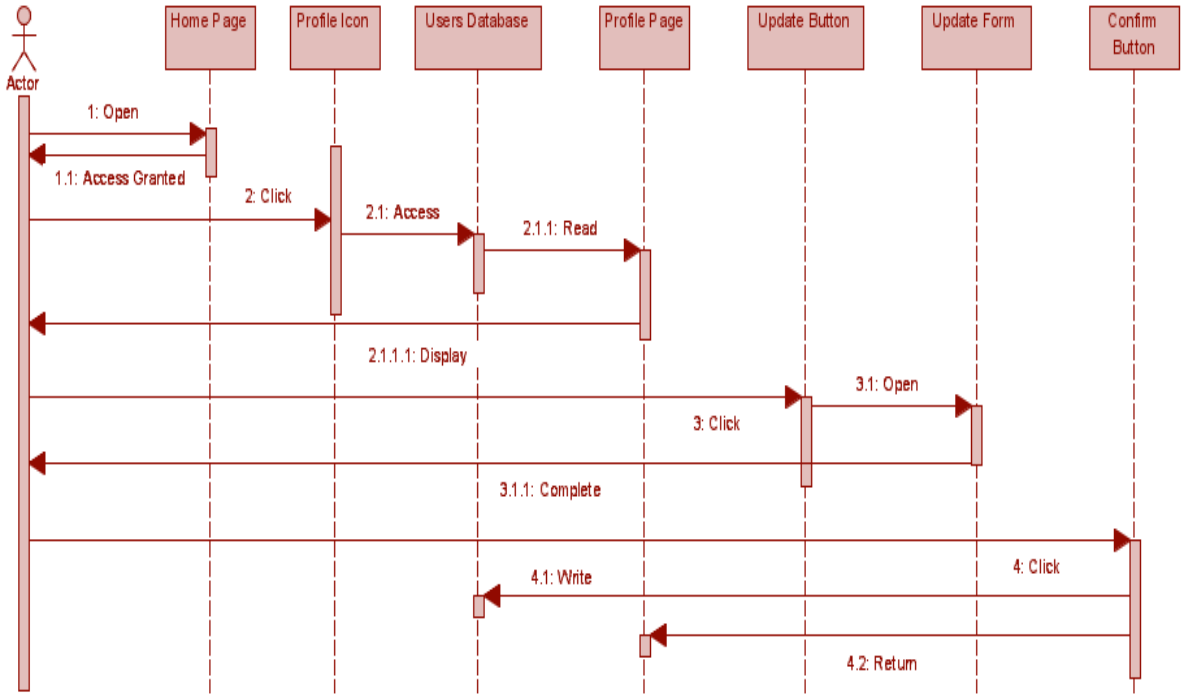
1.5.4 Sequence Diagram For Help Page Access



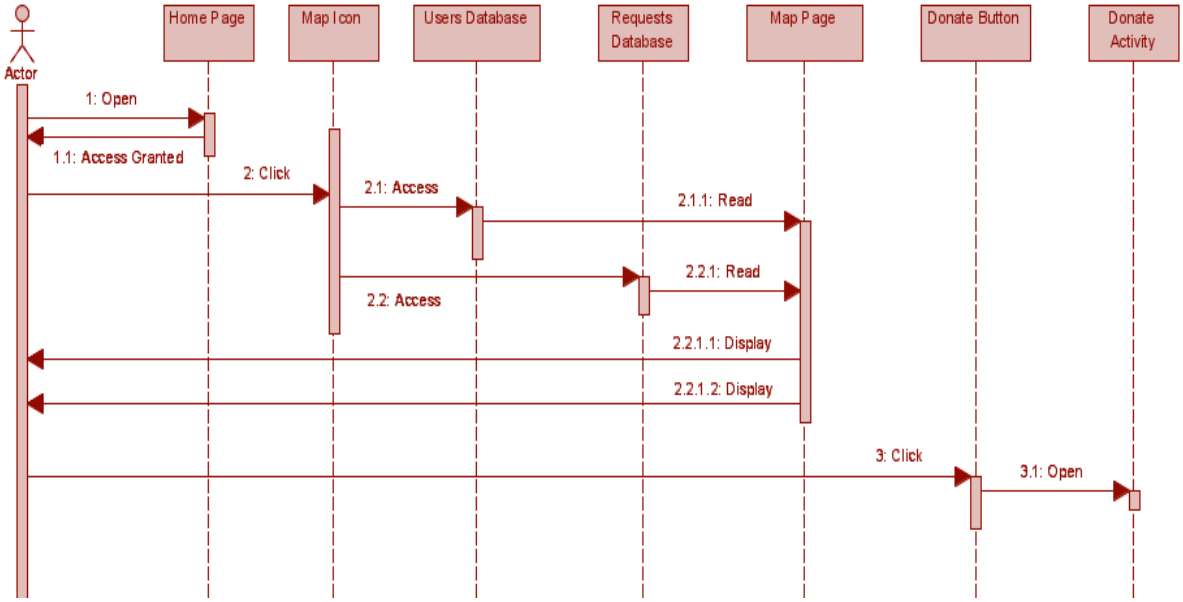
1.5.5 Sequence Diagram For Profile Page View:



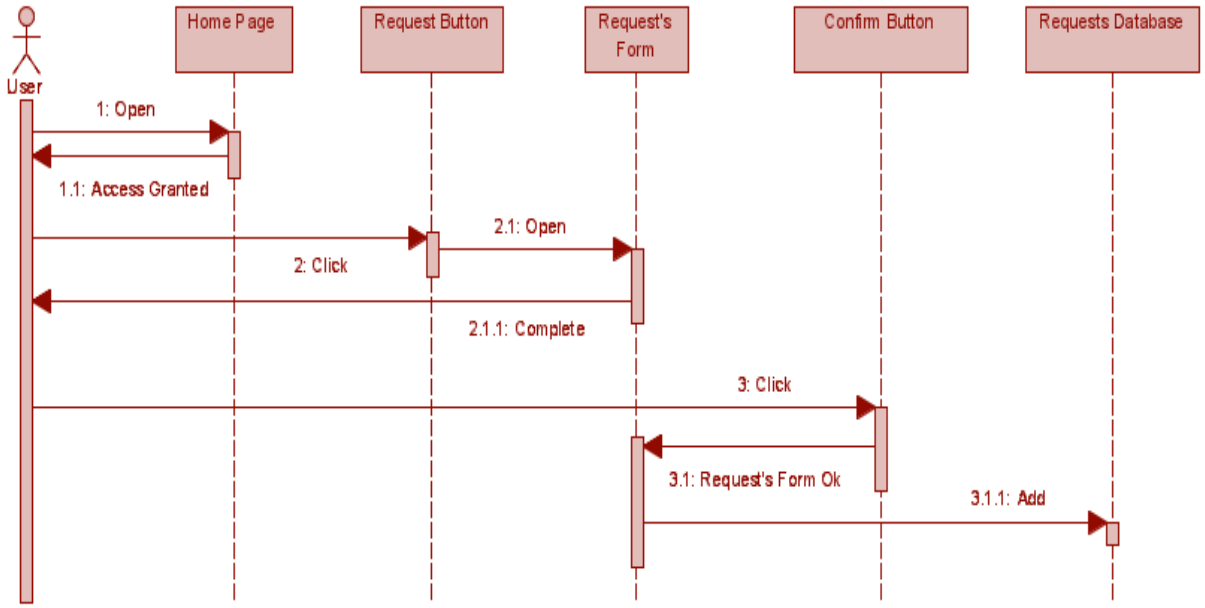
1.5.6 Sequence Diagram For Profile Page Update:



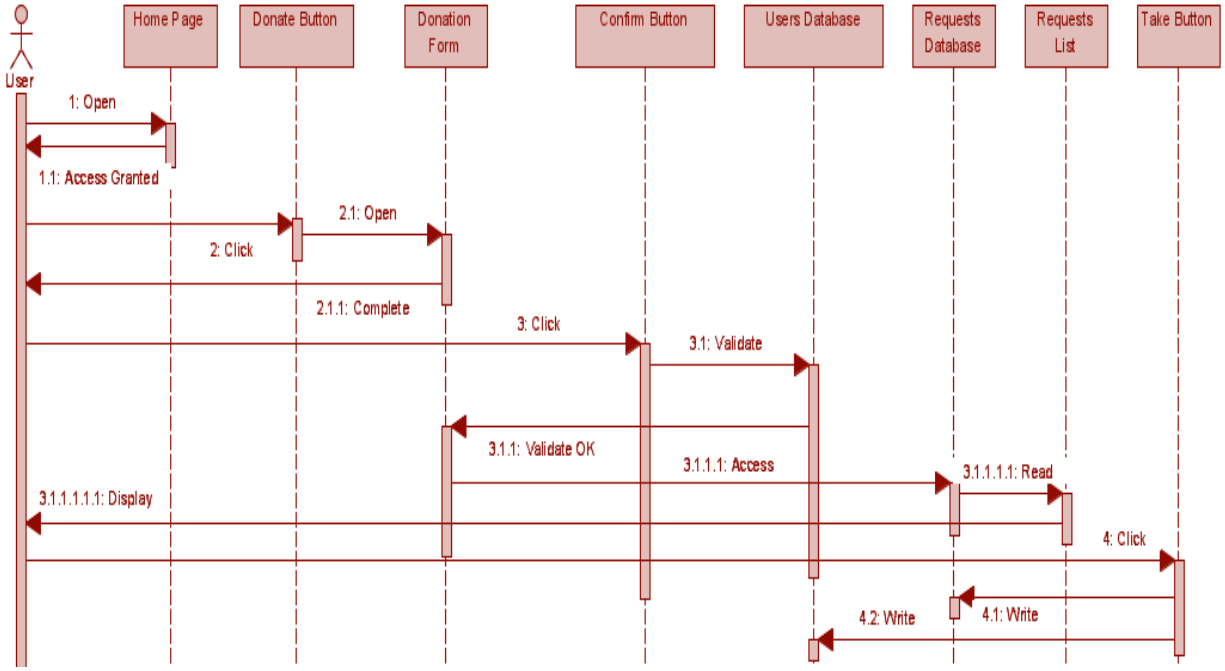
1.5.7 Sequence Diagram For Patient Locator Map View:



1.5.8 Sequence Diagram For Blood Donation Request:



1.5.9 Sequence Diagram For Blood Donation:



2. Function specifications: Back End Functions

`AboutUsActivity.java`

- **onCreate**

```
onCreate(Bundle savedInstanceState)
```

This function is called upon clicking on the About Us Icon. It sets the layout of the About Us page with access to the toolbar. Also, it is passed the credentials of the user (email or password) upon signing in or signing up.

Precondition: user clicks on the About Us icon.

Postcondition: The app control will be transferred to the About Us activity.

- **onCreateOptionsMenu**

```
onCreateOptionsMenu(Menu menu)
```

This function creates the layout for the toolbar (color, size, position, icons, etc). It will also be called upon clicking on the About Us Icon automatically after the "onCreate" function.

Precondition: user clicks on the About Us icon.

Postcondition: The toolbar will appear on top of the screen and the user will have access
all its services.

- **onOptionsItemSelected**

```
onOptionsItemSelected(@NonNull MenuItem item)
```

This function dictates what happens when the user clicks on any icon on the toolbar. Every icon has an Id and if the id of the icon clicked equals to one of the ids of the different icons present, the app control will be transferred to the activity corresponding to the icon clicked.

Precondition: user clicks on an icon in the toolbar.

Postcondition: the control of the application will be transferred to the page

corresponding to the icon clicked.

`CredentialsSQLiteOpenHelper.java`

- **CredentialsSQLiteOpenHelper**

```
CredentialsSQLiteOpenHelper(@Nullable Context context)
```

This function is a constructor for any object of CredentialsSQLiteOpenHelper class. It is used to instantiate and initialize objects of this class.

Precondition: The application is active on one of its activities.

Postcondition: The application now has access (read and write) to the credentials

database of the application.

- **onCreate**

```
onCreate(SQLiteDatabase db)
```

This function creates the credentials table for the credentials database. It creates a table that has Id, name, email, phone number, and password as its attributes. It is passed an object of SQLiteDatabase in its parameters that it uses to create the database. The function is called whenever using the SQLiteOpenHelper class.

Precondition: The user is using a certain service that requires access to his/her credentials.

Postcondition: The credentials table will be created and the database services can now be accessed.

- **onUpgrade**

```
onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)
```

This function upgrades the version of the database in case any major changes occur to the credentials database, then different versions of the database will be recorded. The function takes a SQLiteDatabase object that is a blueprint for the database class to be upgraded. Also, the function takes two numbers, one corresponding to the number of the old version and the other corresponding to the number of the new version.

Precondition: The database has to be changed in a way that it will change its design and the user should be using a service that has access to the credentials database.

Postcondition: A new version of the database will be created and now all accesses to the credentials database will be to the new version.

- **insertCredentials**

```
insertCredentials(SQLiteDatabase db, String name, String email,
String phone, String password)
```

This function is used to insert (write) instances to the credentials database. The arguments of the method have all attributes of the credentials table and an object

of the SQLiteDatabase that is going to be used to insert elements to the database class that it corresponds to.

Precondition: The user should be using a service that has access to the credentials

database.

Postcondition: A new elements (instance) is added to the credentials database.

`DonateActivity.java`

- **onCreate**

```
onCreate(Bundle savedInstanceState)
```

This function is called upon clicking on the “Donate Blood” button. It connects with the DonateFragment that will fill up the activity eventually with access to the toolbar. Also, it is passed the credentials of the user (email or password) upon signing in or signing up.

Precondition: user clicks on the “Donate Blood” button.

Postcondition: The app control will be transferred to the Donate Fragment.

- **onCreateOptionsMenu**

```
onCreateOptionsMenu(Menu menu)
```

This function creates the layout for the toolbar (color, size, position, icons, etc). It will also be called upon clicking on the “Donate Blood” button automatically after the “onCreate” function.

Precondition: user clicks on the “Donate Blood” button.

Postcondition: The toolbar will appear on top of the screen and the user will have access

all its services.

- **onOptionsItemSelected**

```
onOptionsItemSelected ( @NonNull MenuItem item)
```

This function dictates what happens when the user clicks on any icon on the toolbar. Every icon has an Id and if the id of the icon clicked equals to one of the ids of the different icons present, the app control will be transferred to the activity corresponding to the icon clicked.

Precondition: user clicks on an icon in the toolbar.

Postcondition: the control of the application will be transferred to the page

corresponding to the icon clicked.

`DonateFragment.java`

- **onCreate**

```
onCreate(Bundle savedInstanceState)
```

This function is called upon clicking on the “Donate Blood” button in the Home page. It creates a fragment that fills the DonateActivity layout. It has an object of type Bundle as some attributes that might be used in the fragment class from other classes can be passed through this bundle.

Precondition: The user accesses the DonateActivity.

Postcondition: A fragment for the DonateActivity class will be created.

- **onCreateView**

```
onCreateView(LayoutInflater inflater, ViewGroup container,
             Bundle savedInstanceState)
```

This function creates the layout for the fragment filling the DonateActivity class. It uses an object of class LayoutInflater to inflate the fragment and create the layout wanted. It uses an object of type Bundle also to pass value to variables of this class from different classes.

Precondition: The user should have access to the DonateActivity class.

Postcondition: The layout of the fragment will be displayed on the screen filling

(inflating) the DonateActivity class.

- **onStart**

```
onStart ()
```

This function is called automatically after the “onCreateView” function. This function is connected to the layout (donation form) and it accesses the credentials database (read only) in order to autofill the name and phone number of users in the donation form. The method will display an error message in case the database is not available. Also, the method has access to the confirm button of the layout. Moreover, the method uses another support method which is “onClick”.

Precondition: The user accessed the DonateActivity class.

Postcondition: The fragment will have direct access and control on the credentials

database and the fragment layout.

- **onClick**

```
onClick(View v)
```

This function determines what happens when the user clicks on the confirm button that is going to be displayed on the screen in the donate fragment filling the donate activity. The method provides access to the whole layout of the

fragments (including the donation form). The method ensures that the information entered in the form is valid and complete. If so, it transfers the control app to an activity containing all the requests of the same blood type inputted in the donation form.

Precondition: The user must have access to the DonateActivity class.

Postcondition: The application control is transferred to the RequestListActivity class that

will display matching donation requests in terms of blood type.

`HelpActivity.java`

- **onCreate**

```
onCreate(Bundle savedInstanceState)
```

This function is called upon clicking on the Help Icon. It sets the layout of the Help page with access to the toolbar. Also, it is passed the credentials of the user (email or password) upon signing in or signing up.

Precondition: user clicks on the Help icon.

Postcondition: The app control will be transferred to the Help activity.

- **onCreateOptionsMenu**

```
onCreateOptionsMenu(Menu menu)
```

This function creates the layout for the toolbar (color, size, position, icons, etc). It will also be called upon clicking on the Help Icon automatically after the "onCreate" function.

Precondition: user clicks on the Help icon.

Postcondition: The toolbar will appear on top of the screen and the user will have access

all its services.

- **onOptionsItemSelected**

```
onOptionsItemSelected (@NonNull MenuItem item)
```

This function dictates what happens when the user clicks on any icon on the toolbar. Every icon has an Id and if the id of the icon clicked equals to one of the ids of the different icons present, the app control will be transferred to the activity corresponding to the icon clicked.

Precondition: user clicks on an icon in the toolbar.

Postcondition: the control of the application will be transferred to the page

corresponding to the icon clicked.

`HomeActivity.java`

- **onCreate**

```
onCreate(Bundle savedInstanceState)
```

This function is called upon clicking on the Home Icon. It sets the layout of the Home page with access to the toolbar. Also, it is passed the credentials of the user (email or password) upon signing in or signing up.

Precondition: user clicks on the Home icon.

Postcondition: The app control will be transferred to the Help activity.

- **onCreateOptionsMenu**

```
onCreateOptionsMenu(Menu menu)
```

This function creates the layout for the toolbar (color, size, position, icons, etc). It will also be called upon clicking on the Home Icon automatically after the "onCreate" function.

Precondition: user clicks on the Home icon.

Postcondition: The toolbar will appear on top of the screen and the user will have access

all its services.

- **onOptionsItemSelected**

```
onOptionsItemSelected (@NonNull MenuItem item)
```

This function dictates what happens when the user clicks on any icon on the toolbar. Every icon has an Id and if the id of the icon clicked equals to one of the ids of the different icons present, the app control will be transferred to the activity corresponding to the icon clicked.

Precondition: user clicks on an icon in the toolbar.

Postcondition: the control of the application will be transferred to the page

corresponding to the icon clicked.

- **onClickRequest**

```
onClickRequest(View view)
```

This function determines what happens after clicking on the “Request Blood” button. It transfers the controls of the application to the RequestActivity class while also passing the necessary credentials (email or phone number) of the user to the RequestActivity class.

Precondition: user clicks on the “Request Blood” button in the HomeActivity.

Postcondition: application control transferred to RequestActivity class with all needed

values (user’s critical credentials).

- **onClickDonate**

```
onClickDonate(View view)
```

This function determines what happens after clicking on the “Donate Blood” button. It transfers the controls of the application to the DonateActivity class while

also passing the necessary credentials (email or phone number) of the user to the DonateActivity class.

Precondition: user clicks on the “Donate Blood” button in the HomeActivity.

Postcondition: application control transferred to DonateActivity class with all needed

values (user’s critical credentials).

`MainActivity.java`

- **onCreate**

```
onCreate(Bundle savedInstanceState)
```

This function is called as the first function in the entire application. It sets the layout of the Main page.

Precondition: User starts the application.

Postcondition: The user has access to the services (sign in/sign up) of the MainActivity

class.

- **onCreateOptionsMenu**

```
onCreateOptionsMenu(Menu menu)
```

This function creates the layout for the toolbar (color, size, position, etc). It will also be called automatically after the “onCreate” function.

Precondition: User opens the application.

Postcondition: The toolbar will appear on top of the screen.

- **onSignUpClick**

```
onSignUpClick(View view)
```

This function determines what happens after clicking on the “Sign Up” button. It transfers the controls of the application to the SignUpActivity class.

Precondition: user clicks on the “Sign Up” button in the SignUpActivity.

Postcondition: application control transferred to SignUpActivity class.

- **onSignInClick**

```
onSignInClick(View view)
```

This function determines what happens after clicking on the “Sign In” button. It transfers the controls of the application to the SignInActivity class.

Precondition: user clicks on the “Sign In” button in the SignInActivity.

Postcondition: application control transferred to SignInActivity class.

`ProfileActivity.java`

- **onCreate**

```
onCreate(Bundle savedInstanceState)
```

This function is called upon clicking on the Profile Icon. It sets the layout of the Profile page with access to the toolbar. Also, it is passed the credentials of the user (email or password) upon signing in or signing up.

Precondition: user clicks on the Profile icon.

Postcondition: The app control will be transferred to the Profile activity.

- **onCreateOptionsMenu**

```
onCreateOptionsMenu(Menu menu)
```

This function creates the layout for the toolbar (color, size, position, icons, etc). It will also be called upon clicking on the Profile Icon automatically after the “onCreate” function.

Precondition: user clicks on the Profile icon.

Postcondition: The toolbar will appear on top of the screen and the user will have access

all its services.

- **onOptionsItemSelected**

```
onOptionsItemSelected ( @NonNull MenuItem item)
```

This function dictates what happens when the user clicks on any icon on the toolbar. Every icon has an Id and if the id of the icon clicked equals to one of the ids of the different icons present, the app control will be transferred to the activity corresponding to the icon clicked.

Precondition: user clicks on an icon in the toolbar.

Postcondition: the control of the application will be transferred to the page

corresponding to the icon clicked.

- **onChangePasswordClick**

```
onChangePasswordClick ( View view)
```

This function determines what happens after clicking on the “Change Password” button. It transfers the controls of the application to the UpdatePasswordActivity class while also passing the necessary credentials (email or phone number) of the user to the UpdatePasswordActivity class.

Precondition: user clicks on the “Change Password” button in the ProfileActivity.

Postcondition: application control transferred to UpdatePasswordActivity class with all

needead values (user’s critical credentials).

`RequestActivity.java`

- **onCreate**

```
onCreate(Bundle savedInstanceState)
```

This function is called upon clicking on the “Request Blood” button. It connects with the RequestFragment that will fill up the activity eventually with access to the toolbar. Also, it is passed the credentials of the user (email or password) upon signing in or signing up.

Precondition: user clicks on the “Request Blood” button.

Postcondition: The app control will be transferred to the Request Fragment.

- **onCreateOptionsMenu**

```
onCreateOptionsMenu(Menu menu)
```

This function creates the layout for the toolbar (color, size, position, icons, etc). It will also be called upon clicking on the “Request Blood” button automatically after the “onCreate” function.

Precondition: user clicks on the “Request Blood” button.

Postcondition: The toolbar will appear on top of the screen and the user will have access to all its services.

- **onOptionsItemSelected**

```
onOptionsItemSelected ( @NonNull MenuItem item)
```

This function dictates what happens when the user clicks on any icon on the toolbar. Every icon has an Id and if the id of the icon clicked equals to one of the

ids of the different icons present, the app control will be transferred to the activity corresponding to the icon clicked.

Precondition: user clicks on an icon in the toolbar.

Postcondition: the control of the application will be transferred to the page

corresponding to the icon clicked.

`RequestListFragment.java`

- **onCreate**

```
onCreate(Bundle savedInstanceState)
```

This function is called upon clicking on the “Confirm” button in the Donate fragment filling the DonateActivity class and initiating a fragment transaction from the RequestListActivity class. It creates a fragment that fills the RequestListActivity layout. It has an object of type Bundle as some attributes that might be used in the fragment class from other classes can be passed through this bundle.

Precondition: The user accesses the RequestListActivity.

Postcondition: A fragment for the RequestListActivity class will be created.

- **onCreateView**

```
onCreateView(LayoutInflater inflater, ViewGroup container,
             Bundle savedInstanceState)
```

This function creates the layout for the fragment filling the RequestListActivity class. It uses an object of class LayoutInflater to inflate the fragment and create the layout wanted. It uses an object of type Bundle also to pass value to variables of this class from different classes. Before this, the function accesses the requests database (read only), reads all the requests that have the same

blood type that the user inputs in the blood donation form. All the requests will be added to a list that is going to be the layout of the fragment.

Precondition: The user should have access to the RequestListActivity class.

Postcondition: The layout of the fragment will be displayed on the screen filling

(inflating) the RequestActivity class.

`RequestsSQLiteOpenHelper.java`

- ## RequestsSQLiteOpenHelper

```
RequestsSQLiteOpenHelper(@Nullable Context context)
```

This function is a constructor for any object of RequestsSQLiteOpenHelper class. It is used to instantiate and initialize objects of this class.

Precondition: The application is active on one of its activities.

Postcondition: The application now has access (read and write) to the requests database

of the application.

- ## onCreate

```
onCreate(SQLiteDatabase db)
```

This function creates the requests table for the requests database. It creates a table that has Id, name, email, blood type, location, phone number, alternative name, output, and alternative phone as its attributes. It is passed an object of SQLiteDatabase in its parameters that it uses to create the database. The function is called whenever using the RequestsSQLiteOpenHelper class.

Precondition: The user hits on the “Confirm” button in the request fragment filing the

RequestActivity class.

Postcondition: The requests table will be created and the database services can now

be accessed.

- **onUpgrade**

```
onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)
```

This function upgrades the version of the database in case any major changes occur to the requests database, then different versions of the database will be recorded. The function takes a SQLiteDatabase object that is a blueprint for the database class to be upgraded. Also, the function takes two numbers, one corresponding to the number of the old version and the other corresponding to the number of the new version.

Precondition: The database has to be changed in a way that it will change its design and

the user should be using a service that has access to the credentials database.

Postcondition: A new version of the database will be created and now all accesses to the

requests database will be to the new version.

- **insertRequest**

```
insertRequest(SQLiteDatabase db, String name,String
bloodType, String location,String phone, String alternativeName,
String alternativePhone, String output)
```

This function is used to insert (write) instances to the requests database. The arguments of the method have all attributes of the requests table and an object of the SQLiteDatabase that is going to be used to insert elements to the database class that it corresponds to.

Precondition: The user should be using a service that has access to the requests database.

Postcondition: A new element (instance) is added to the requests database.

`SignInActivity.java`

- **onCreate**

```
onCreate(Bundle savedInstanceState)
```

This function is called upon clicking on the “Sign In” button in the MainActivity class. It connects with the SignInFragment that will fill up the activity eventually with access to an empty toolbar.

Precondition: User clicks on the “Sign In” button in the SignInActivity class.

Postcondition: The application control will be transferred to the SignIn activity.

- **onCreateOptionsMenu**

```
onCreateOptionsMenu(Menu menu)
```

This function creates the layout for the toolbar (color, size, position, etc). It will also be called automatically after the “onCreate” function.

Precondition: User clicks on the “Sign In” button in the MainActivity class.

Postcondition: The toolbar will appear on top of the screen.

`SignUpActivity.java`

- **onCreate**

```
onCreate(Bundle savedInstanceState)
```

This function is called upon clicking on the “Sign Up” button in the MainActivity class. It connects with the SignUpFragment that will fill up the activity eventually with access to an empty toolbar.

Precondition: User clicks on the “Sign Up” button in the MainActivity class.

Postcondition: The application control will be transferred to the SignIn activity.

♦ onCreateOptionsMenu

```
onCreateOptionsMenu(Menu menu)
```

This function creates the layout for the toolbar (color, size, position, etc). It will also be called automatically after the “onCreate” function.

Precondition: User clicks on the “Sign Up” button in the MainActivity class.

Postcondition: The toolbar will appear on top of the screen.

`SignInFragment.java`

♦ onCreate

```
onCreate(Bundle savedInstanceState)
```

This function is called upon clicking on the “Sign In” button in the Main page and initiating a fragment transaction in the SignInActivity class. It creates a fragment that fills the SignInActivity class layout. It has an object of type Bundle as some attributes that might be used in the fragment class from other classes can be passed through this bundle.

Precondition: The user accesses the SignInActivity.

Postcondition: A fragment for the SignInActivity class will be created.

♦ onCreateView

```
onCreateView(LayoutInflater inflater, ViewGroup container,
                Bundle savedInstanceState)
```

This function creates the layout for the fragment filling the SignInActivity class. It uses an object of class LayoutInflater to inflate the fragment and create the layout wanted. It uses an object of type Bundle also to pass value to variables of this class from different classes.

Precondition: The user should have access to the SignInActivity class.

Postcondition: The layout of the fragment will be displayed on the screen filling

(inflating) the SignInActivity class.

- **onStart**

```
onStart ()
```

This function is called automatically after the “onCreateView” function. This function is connected to the layout (Sign In form) and it accesses the credentials database (read and write) to approve that the information entered in the form is valid and correct. The method will display an error message in case the database is not available. Also, the method has access to the confirm button of the layout. Moreover, the method uses another support method which is “onClick”.

Precondition: The user accessed the SignInActivity class.

Postcondition: The fragment will have direct access and control on the requests

database and the fragment layout.

- **onClick**

```
onClick(View v)
```

This function determines what happens when the user clicks on the confirm button that is going to be displayed on the screen in the sign in fragment filling the sign in activity. The method provides access to the whole layout of the fragment (sign in form). The method ensures that the information entered in the form is valid and complete. If so, it transfers the control of the application to the HomeActivity class and passes the email or phone number inputted by the user to the HomeActivity class as well with a success message displayed on the screen. An error message will be displayed in case the user inputs an incorrect email/phone number, incorrect password, or submits an incomplete form. In case of incorrect email/phone number the application control will also be transferred back to the MainActivity class asking the user to Sign Up.

Precondition: The user must have access to the SignInActivity class.

Postcondition: The application control will be transferred to the HomeActivity class and passed the email or phone number inputted by the user to the HomeActivity class as well.

`SignUpFragment.java`

- **onCreate**

```
onCreate(Bundle savedInstanceState)
```

This function is called upon clicking on the “Sign Up” button in the Main page and initiating a fragment transaction in the SignUpActivity class. It creates a fragment that fills the SignUpActivity class layout. It has an object of type Bundle as some attributes that might be used in the fragment class from other classes can be passed through this bundle.

Precondition: The user accesses the SignUpActivity.

Postcondition: A fragment for the SignUpActivity class will be created.

- **onCreateView**

```
onCreateView(LayoutInflater inflater, ViewGroup container,
             Bundle savedInstanceState)
```

This function creates the layout for the fragment filling the SignUpActivity class. It uses an object of class LayoutInflater to inflate the fragment and create the layout wanted. It uses an object of type Bundle also to pass value to variables of this class from different classes.

Precondition: The user should have access to the SignUpActivity class.

Postcondition: The layout of the fragment will be displayed on the screen filling

(inflating) the SignUpActivity class.

- **onStart**

```
onStart ()
```

This function is called automatically after the “onCreateView” function. This function is connected to the layout (Sign Up form) and it accesses the credentials database (read and write) to insert the information added to the database after full and valid completion of the form. The method will display an error message in case the database is not available. Also, the method has access to the confirm button of the layout. Moreover, the method uses another support method which is “onClick”.

Precondition: The user accessed the SignUpActivity class.

Postcondition: The fragment will have direct access and control on the requests

database and the fragment layout.

- **onClick**

```
onClick(View v)
```

This function determines what happens when the user clicks on the confirm button that is going to be displayed on the screen in the sign up fragment filling the sign up activity. The method provides access to the whole layout of the fragment (sign up form). The method ensures that the information entered in the form is valid and complete. If so, it creates an instance that has all attributes of a credential in the credentials database and inserts the instance in the database. Moreover, it transfers the control of the application to the HomeActivity class and passes the email or phone number inputted by the user to the HomeActivity class as well with a success message displayed on the screen. An error message will be displayed in case the user inputs an invalid information in the sign up form or submits an incomplete form.

Precondition: The user must have access to the SignUpActivity class.

Postcondition: The application control will be transferred to the HomeActivity class and passed the email or phone number inputted by the user to the HomeActivity class as well. Also, an element will be added to the credentials database having the user inputted information as attributes.

`UpdatePasswordActivity.java`

- **onCreate**

```
onCreate(Bundle savedInstanceState)
```

This function is called upon clicking on the “Change Password” button in the ProfileActivity class. It sets the layout of the UpdatePasswordActivity class with access to the toolbar. Also, it is passed the credentials of the user (email or password) upon signing in or signing up.

Precondition: user clicks on the “Change Password” button in the ProfileActivity class.

Postcondition: The app control will be transferred to the update password activity.

- **onCreateOptionsMenu**

```
onCreateOptionsMenu(Menu menu)
```

This function creates the layout for the toolbar (color, size, position, icons, etc). It will also be called automatically after the “onCreate” function.

Precondition: User clicks on the “Change Password” button in the ProfileActivity class.

Postcondition: The toolbar will appear on top of the screen and the user will have access

all its services.

- **onOptionsItemSelected**

```
onOptionsItemSelected ( @NonNull MenuItem item)
```

This function dictates what happens when the user clicks on any icon on the toolbar. Every icon has an Id and if the id of the icon clicked equals to one of the ids of the different icons present, the app control will be transferred to the activity corresponding to the icon clicked.

Precondition: user clicks on an icon in the toolbar.

Postcondition: the control of the application will be transferred to the page

corresponding to the icon clicked.

- **onSignUpClick**

```
onConfirmClick(View view)
```

This function determines what happens after clicking on the “Confirm” button in the UpdatePasswordActivity layout. It has access to the credentials database

(read and write) and the layout of the class. It reads the input of the user and sets it as a new password in the credentials database for that user. It displays an error message if the database does not exist. In case of success, the application control will be transferred to the profile activity with a success message displayed on the screen and passing the email or phone number inputted by the user while signing up or signing in to the profile activity as well.

Precondition: user clicks on the “Confirm” button in the UpdatePasswordActivity.

Postcondition: application control transferred to ProfileActivity class with user’s email

or phone number.

This report has been signed by Nehmat Touma, Mahdi Hallal and Pia Chouaifaty on the 11th of May 2021.