System Design

Class name: activity_main
Parent class (if any): MainActivity
Classname Subclasses (if any): -
Responsibilities:

- Navigate to other pages
- Search area

Collaborators:

- activity_medical_page

Class name: MainActivity
Parent class (if any): -
Classname Subclasses (if any): activity_main
Responsibilities:

- View UI
- Navigation buttons

Collaborators:

- MedicalPage

Class name: activity_medical_page
Parent class (if any): MedicalPage
Classname Subclasses (if any): -
Responsibilities:

- Navigate to other pages
- Search area

Collaborators:

- activity_main

Class name: MedicalPage
Parent class (if any): activity_medical_page
Classname Subclasses (if any): -
Responsibilities:

- View UI
- Navigation buttons

Collaborators:

- MainActivity

Class name: menu_main
Parent class (if any): -
Classname Subclasses (if any): -
Responsibilities:

-

Collaborators:

- MainActivity
- MedicalPage

Class name: User
Parent class (if any): -
Classname Subclasses (if any): -
Responsibilities:

- Create a user based on user input
- Update user information (passwords etc)

Collaborators:

- ERADBHandler

Class name: EmergencyIssue
Parent class (if any): -
Classname Subclasses (if any): -
Responsibilities:

- Create user provided medical issues
- Add appropriate solutions to user provide issues
- Update solutions to medical issues
- Validate medical issues to prevent spam

Collaborators:

- ERADBHandler

Class name: ERADBHandler
Parent class (if any): SQLiteOpenHelper
Classname Subclasses (if any): -
Responsibilities:

- Create a connection to SQLite Database
- Create tables (user, and emergency_issue)
- Provide CRUD support for the tables created

Collaborators:

- User
- EmergencyIssue

Class name: Bookmark
Parent class (if any):
Classname Subclasses (if any): -
Responsibilities:

- Create issues to be bookmarked

Collaborators:

- ERADBHandler

Class name: DiagnosisClient
Parent class (if any):
Classname Subclasses (if any): -
Responsibilities:

- Create Strings for GET request to APIMEDIC
- Store output from GET request into a String
- Parse JSON Output

Collaborators:

- User

Class name: AccessToken
Parent class (if any):
Classname Subclasses (if any): -
Responsibilities:

- Generate Access Token for API

Collaborators:

Class name: DiagnosedIssue
Parent class (if any): HealthItem
Classname Subclasses (if any): -
Responsibilities:

- Factory Design for Health Item related to the issue diagnosis

Collaborators:

- SampleDiagnosis

Class name: Gender
Parent class (if any):
Classname Subclasses (if any): -
Responsibilities:

- Selecting user gender

Collaborators:

- SampleDiagnosis

Class name: HealthDiagnosis
Parent class (if any):
Classname Subclasses (if any): -
Responsibilities:

- Creating a list of doctors with specialization appropriate to the diagnosis

Collaborators:

- SampleDiagnosis

Class name: HealthIssueInfo
Parent class (if any):
Classname Subclasses (if any): -
Responsibilities:
- Collect and store information on a potential health issue

Collaborators:
- SampleDiagnosis

Class name: HealthItem
Parent class (if any):
Classname Subclasses (if any): -
Responsibilities:
- Collect the corresponding ID for a health issue
- Store the name of a selected health issue

Collaborators:
- SampleDiagnosis

Class name: HealthSymptomSelector
Parent class (if any): HealthItem
Classname Subclasses (if any): -
Responsibilities:
- Store symptoms with ids
- Store body locations associated with symptoms

Collaborators:
- SampleDiagnosis
- DiagnosisClient
- DiagnosedIssue

Class name: MatchedSpecialization
Parent class (if any):
Classname Subclasses (if any): -
Responsibilities:
- Store id of api generated specialization

Collaborators:
- SampleDiagnosis

Class name: SampleDiagnosis
Parent class (if any):
Classname Subclasses (if any): -
Responsibilities:
- Make GET request to APIMEDIC to get a sample diagnosis
- Parse and return output from the GET request

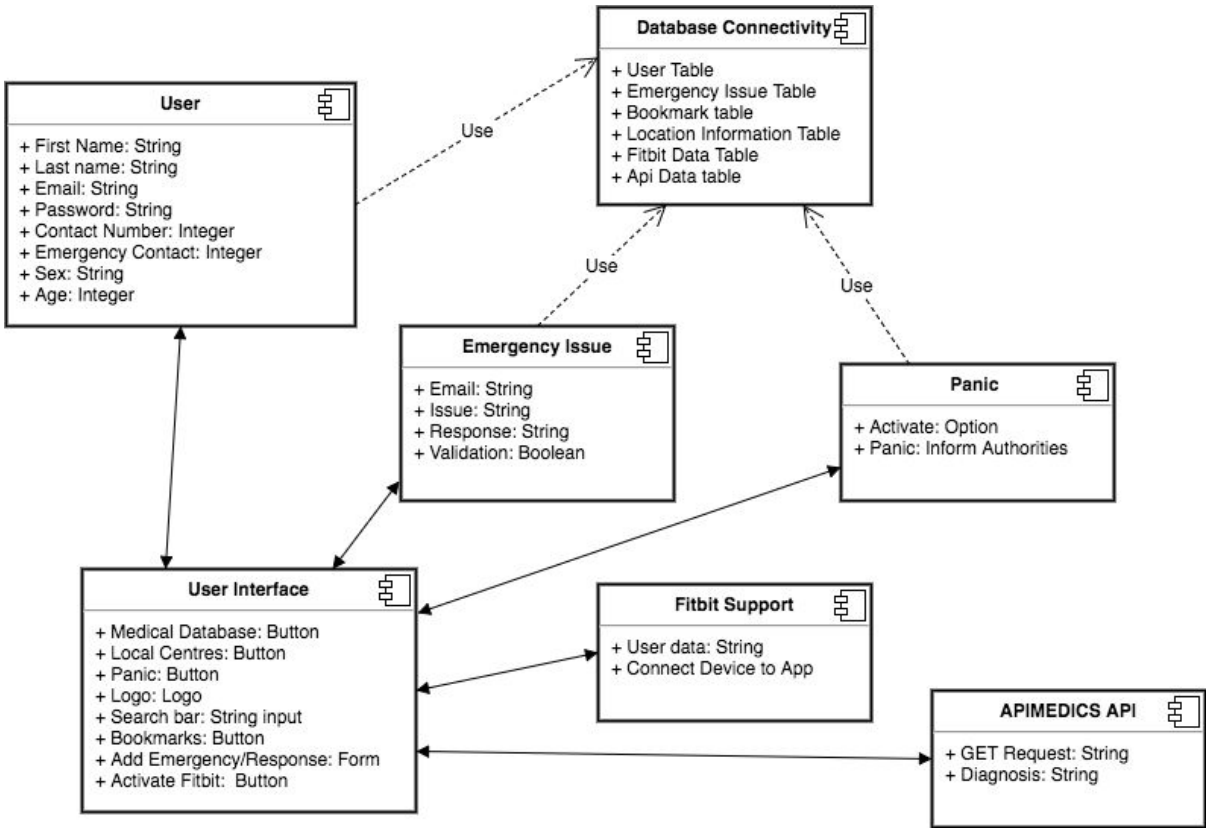
Collaborators:
- DiagnosedIssue
- DiagnosisClient

Design Architecture

The design components are split into two main categories:

- User Interface Components
- Back End Components

The Component UML diagram to illustrate an abstract design architecture of the application is following:



The Individual components are listed below with details of how they interact with one another

Front End:

- Search Bar
- Login Page
- Sign Up Page
- Panic Button
- Form for user provided medical issues and solutions
- Connectivity for Fitbit

Back End:

- Database table for keeping track of user information (username, password, etc)

- Database table for keeping track of login user (authentication purposes).
- API(s) to provide information of medical issues matching user input (i.e apimedic)
- Database table for keeping track of validated user input
- Database table for keeping track of user location with timestamp for panic button
- Function for informing authorities if panic button is activated
- Connecting with Fitbit API

Search Bar: The search bar component of the user interface will allow user to input the either the symptoms or the medical issues of concern. The component will connect to the backend component database table for user provided medical issues to find the issue matching the user input as well as connect to the *apimedics* API to search for issue matching user input. The user input will also be parsed before the search to ensure valid input and to prevent sql injections.

User Login Page / Sign Up page: Login page and signup page will provide user the opportunity to either login or register so that they can use the panic button functionality, and input medical issues and solutions for others to utilize. In the backend The database will keep track of valid user, login info and authenticating process.

Panic Button: For registered user, it will allow authorities to respond in case of an emergency. In the backend if panic button feature is activated, users geo location will be stored in the database in case panic button is pressed. Geolocation is update in the database every 7 minutes.

Form for user provided medical issues and associated solutions: Allows registered users to provide medical issues and solutions for others to utilize. The form connect with the database table in the backend that keeps track of all validated medical issues and associated solutions.

Fitbit Connectivity: Allowing user the option to connect external device (Fitbit) to the application. In the backend the Fitbit Api will keep track of user vitals and allow for appropriate response (panic functionality) in case of an emergency.