CS 307

DESIGN DOCUMENT

GetGuru

Team 12

Sharoon Srivastava

Ankush Jain

Pranav Punjabi

Murtaza Kainan

Vijay Srinivas

TABLE OF CONTENTS

PURPOSE

Tutors, though often available, are hard to find. Students often lack the resources to find and judge the quality of academic help available locally. With a wide range of subjects to study with varying difficulty levels, it becomes hard to choose the right tutor from a large pool of contenders. Already existing apps do not provide a medium to rate the tutors in order to judge their quality. This could be a huge drawback for students trying to make a crucial choice about their academics. The purpose of our application is to allow students to browse through a pool of tutors and enable them to filter tutors based on location, subjects and ratings. It would also provide them with a chat feature to communicate with tutors.

Requirements:

Functional:

- As a user, I'd like to create an account
- As a user, I'd like to register either as a student or as a student and tutor
- As a user, I'd like to receive notifications when contacted via chat
- As a user, I'd like to view my chat history
- As a tutor, I'd like to create a profile (Including a profile picture, relevant contact information, teaching preferences, and schedule)
- As a tutor, I'd like to list my subjects of expertise
- As a tutor, I'd like to edit my style of teaching (group, one-on-one, etc)
- As a tutor, I'd like to view my own ratings & reviews
- As a student, I'd like to search for tutors based on location
- As a student, I'd like to search for tutors based on ratings and reviews
- As a student, I'd like to search for tutors based on subject
- As a student, I'd like to shortlist tutors for future reference
- As a student, I'd like to view tutor profiles
- As a student, I'd like to instantly connect to the tutors through chat
- As a student, I'd like to rate & review the tutors

- As a student, I'd like to schedule a meeting with the tutor using an in-app scheduling assistant (if time allows)
- As a user, I'd like to set my chat status (if time allows)
- As a user, I'd like to view other's chat status' (if time allows)

Non-Functional:

- As a user, I'd like to have fast response times
- As a developer, I'd like the least possible amount of data storage on the client side
- As a developer, I'd like my service to be scalable by using mySQL to manage my database
- As a developer, I'd like my database to be secure by preventing SQL injections using string validations.
- As a developer, I'd like a RESTful API built to handle requests from a variety of clients apart from the android app
- As a developer, I'd like the RESTful API to retrieve information from the database whenever there's a request from the client side
- As a developer, I'd like to validate all user accounts to prevent redundancy
- As a developer, I'd like to be able to test on a local development server easily.
- As a developer, I'd like to have a development API server and a production API server
- As a developer, I'd like to be able to switch the server (development server or productions server) the android app is sending requests to
- As a developer, I'd like the amount of allowable downtime per month to be one hour

DESIGN OUTLINE

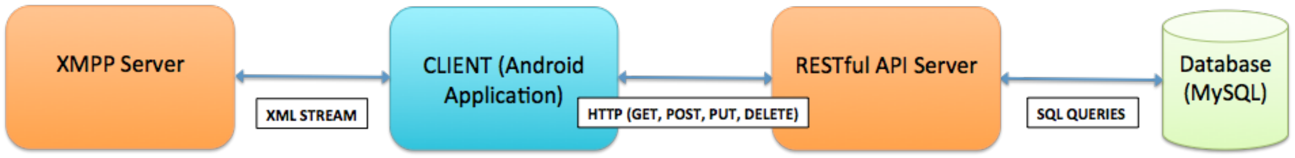
Overview and Components

The software product will consist of:

1. An Android app as the client.
2. A back-end server exposing a RESTful API.
3. A database being accessed by the server.

Architecture

Abstractly, the product will make use of the client-server paradigm. Users will use the Android application which will interact with the server. The client and server will be implemented separately and interact using HTTP requests. All the information will be stored in the database which would be accessed by the server whenever there is a HTTP request from the client.



RESTful API Server

The RESTful API server will interact with the client and the database, maintain abstractions, and send data to the client. The server is independent from the client since unlike normal web applications, web pages do not need to be served. JSON data format will be used to send and receive data on the client and the server.

XMPP Server

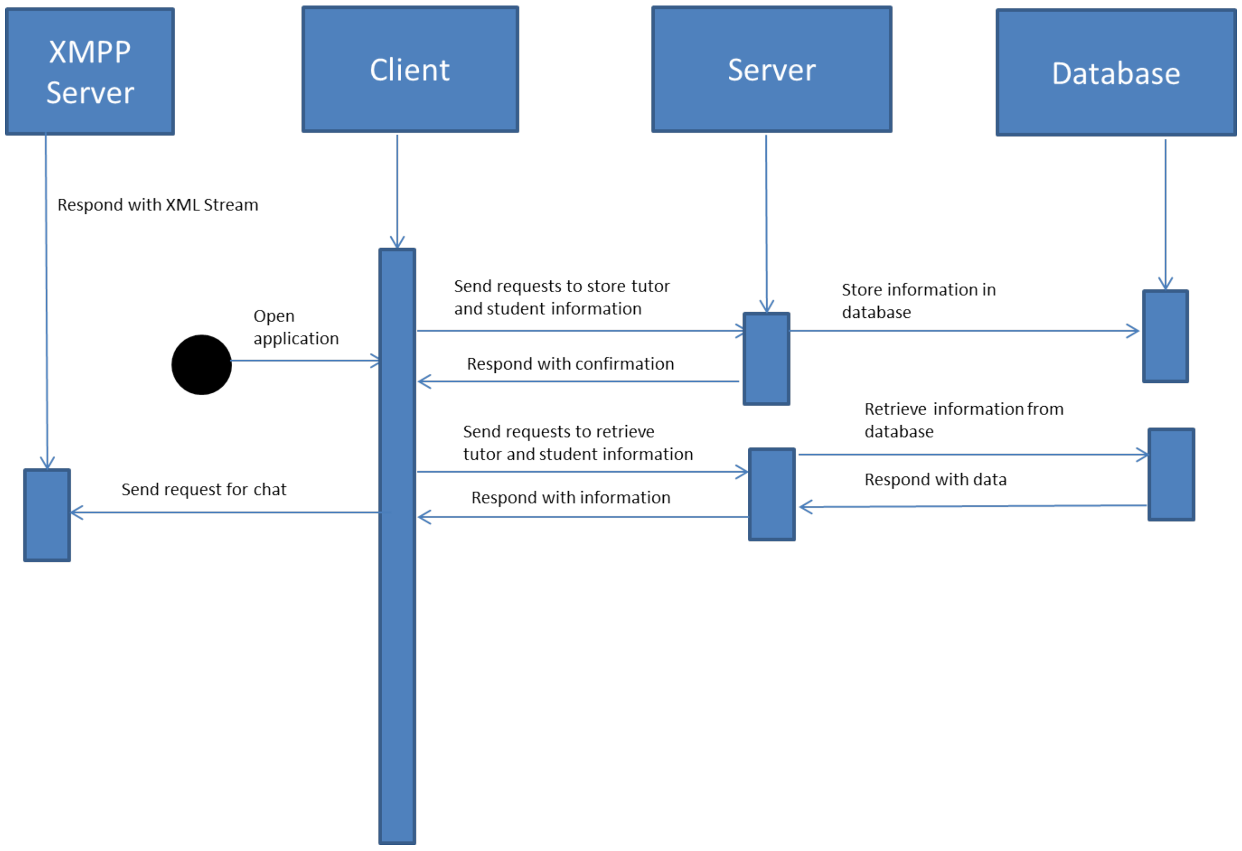
The XMPP server will facilitate chat between two clients by parsing XML streams.

Client

The client will be an Android application that will communicate with the server via HTTP requests. The app will be written natively using Java and XML in the Android Studio Integrated Development Environment. The app needs to be well designed and easy to use; As such the UI and UX elements for the app will need to be taken into consideration.

Database

The database being used is MySQL that would be accessed via the SQLAlchemy module. The database will include information (name, location, subjects, etc.) about the tutors and the students.


XMPP Server
Client
Server
Database
Respond with XML Stream
Open application
Send requests to store tutor and student information
Store information in database
Respond with confirmation
Send requests to retrieve tutor and student information
Retrieve information from database
Respond with data
Send request for chat
Respond with information

DESIGN ISSUES

Non Functional

ISSUE 1: Software Framework to be used

Option 1: Flask
Option 2: Jersey
Option 3: Django

Decision: We're planning to use the Flask framework since:

- It's easier to define dependencies in Flask as compared to Jersey framework
- Django is structured in a way which could take up a significant amount of time to learn
- Flask is a Python framework and Python is the easiest programming language to learn because of its simplicity yet great capabilities

ISSUE 2: Database Architecture

Option1: MongoDB
Option 2: MariaDB
Option 3: SQLite
Option 4: MySQL

Decision: We're planning to use MySQL since:

- MongoDB is a non relational database and hence would not provide efficient searching and storing for the tutors and students and their linked data

- MySQL and MariaDB are both relational databases. However, MySQL provides better performance than MariaDB when the number of threads is small.
- SQLite is not as scalable as MySQL

ISSUE 3: Reducing redundancy to avoid multiple profiles from a single user

Option 1: Sign in using a social media platform
Option 2: Enter a valid email address and confirm it
Option 3: Register via a phone number and confirm with a verification code sent as a text message

Decision: We're planning to use social media sign in since:

- Phone number validation requires a monetary overhead
- Verification through emails requires email validation and sending of temporary verification links to a user's inbox. This entails a development and maintenance overhead that could be avoided
- Facebook or social media validation process is easier to implement and is free of charge. We could use the existing Graph API to save time and implement reusability

<u>Functional</u>

ISSUE 4: Classification of users (students or tutors) and login, profiles, UI elements for each

Option 1: The default view is set for whatever the user registers as (i.e.: student or tutor). Accounts will be created differently for students and tutors and will later be modified in app if a student wishes to use the service as a tutor as well or vice versa

Option 2: Every user is a student by default and can sign up as a tutor in-app. Users simply toggle in and out of student and tutor views

Decision: We're planning to choose the option 2 since option 1 overly complicates the UI design and also causes problems by forcing system administrators to maintain a larger number of tables; whereas option 2 allows for a more consistent UI with a unified account.

ISSUE 5: Enabling chat between every user or specific user types.

Option 1: Enable chat between all types of users. In this scenario a student can search for and chat with other students

Option 2: Limit chat between students and tutors. Only students can search for and initiate a chat with tutors

Decision: Option 2 was chosen. It does not make sense as of now to allow students to search for and chat with each other. Since students will search for tutors, it follows that they should be the only ones allowed to initiate a chat with a tutor. Moreover, enabling student-student chat would put unnecessary load on the server.

DESIGN DETAILS

System Components

1. **Client:** The client will be an Android application built natively using Java and XML. It will make use of material design to drive a tabbed user interface that allows the user to quickly switch between views to accomplish different tasks. Every view will be associated with an activity

that will drive its behavior as well as a layout that will arrange components. The client will make HTTP requests to the API server and will also make use of the XMPP protocol for the purposes of chatting. The client will be built to parse JSON data format as it will make calls to the API server as well as receive login information from the Facebook API.

2. **API Server:** The API Server will be built using the Python programming language based Flask framework. It will respond to HTTP Get requests from client by responding with JSON. It will make use of SQL Queries to communicate with the database.

3. **XMPP Server:** The XMPP server will be used to connect client in order to facilitate chat. The Server will send and receive XML streams from clients.

4. **Database:** MySQL will be used for the database. It will hold a various related tables encompassing user information and ratings.

Search functionality

Users making use of the app as students will be able to search based on various filters such as subjects, locality (radius), and style of teaching. Additionally, the option of sorting the search results based on ratings will be available. Search based on locality will make use of location services on the available on the client in order to return search results that are relevant.

Chat functionality

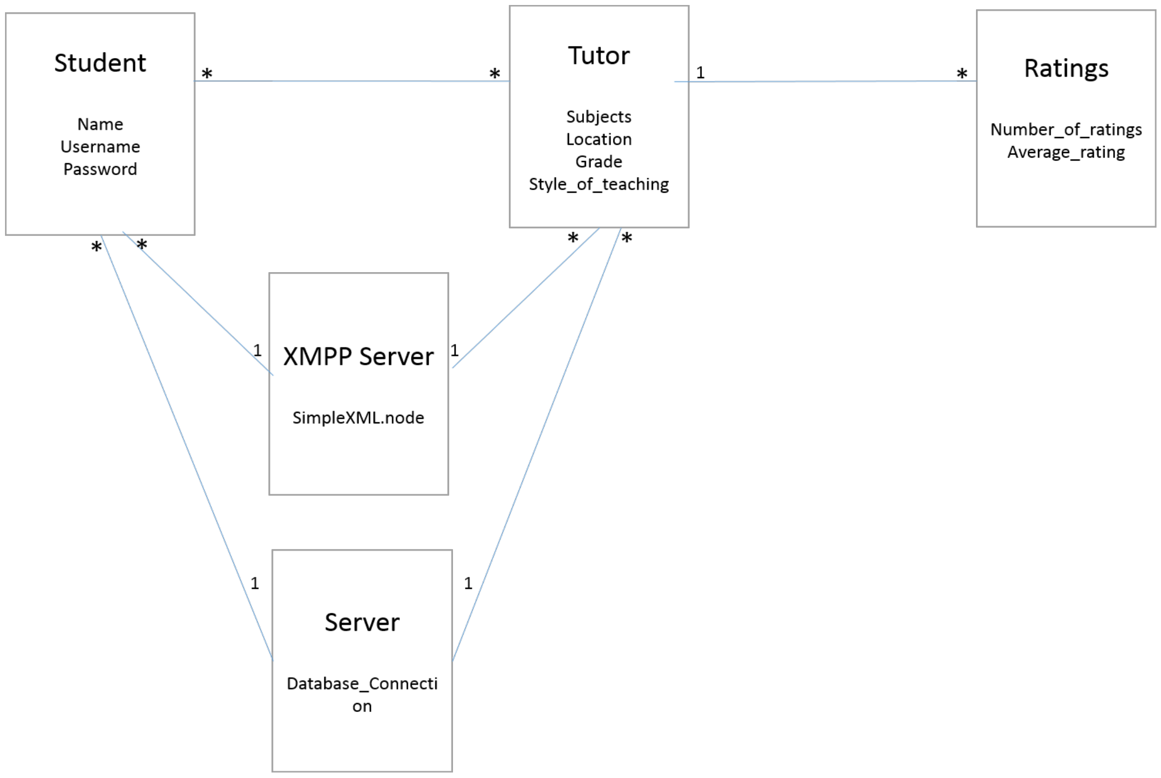
Students and tutors will have the option of sending each other messages. The option to start a chat will be available only to students who are willing to get in touch with a tutor. Students will also not be able to start a chat with another student. These constraints are design to provide a fluid experience that ensures only meaningful communication between students and tutors takes place. As mentioned earlier, the XMPP protocol will be used to implement chat.

Unified accounts and User type toggling

The app will allow toggling so that only a single account is needed in order to use the app. Every user by default is a student. An option to toggle will be available on the app so that a user can use the app as a student or a tutor. As a tutor, a user will be able to edit what subjects tutoring is offered for, where the tutoring is offered, and other relevant information that applies to a tutor.

Facebook login

The app will make use of the Facebook graph API for registration and login. The API will generate a token for every user that will be stored in the database and referred to for login. The API will also provide general information about the user thus limiting the need for a registration or login form on the client and ensuring faster, user friendly logins.



Class Descriptions

Student: The default user class associated with every user once an account is created. The class store basic information such as name, username, and password. This class has a many to many relationship because a tutor can have multiple students and vice versa. It has a many to one relationship with the XMPP server as multiple students can use the server for the chat feature. It also has a many to one relationship with the back end server as a single server can handle multiple students.

Tutor: The class used for every user who enables tutor status. This class inherits from the aforementioned Student class and has additional fields that include location, subjects, grade level taught, and teaching preferences. It has a one to many relationship with the ratings class since one tutor can have multiple ratings. It has a many to many relationship with the Student class because one tutor can have multiple students and vice versa. It also has a many to one relationship with the XMPP server as multiple tutors can use the server for the chat feature. It also has a many to one relationship with the back end server as a single server can handle multiple tutors.

Ratings: The Ratings class stores the ratings and reviews for a particular tutor. It stores a value between one and five for a rating and a textual review. It has a many to one relationship with the Tutor class because each rating class is associated with one tutor and the tutors are associated with multiple ratings.

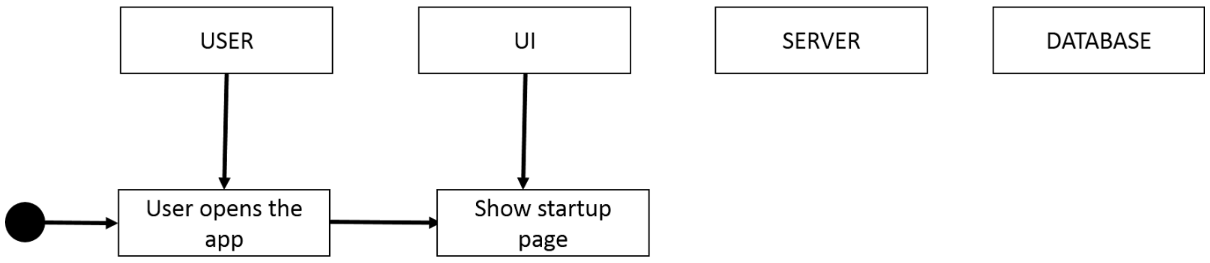
API Server: The class that will represent the API server and its functionality. The server will have connection strings to the database and clients as members and will have a host of functions that will query the database and communicate with clients. As explained above, this class has a one to many relationship with the Tutor and Student classes.

XMPP Server: The class that will represent the XMPP server and facilitate chat between two clients. It has a one to many relationship with the Tutor and Student classes. This is because many tutors and students will communicate with the XMPP server.

Sequence Diagrams

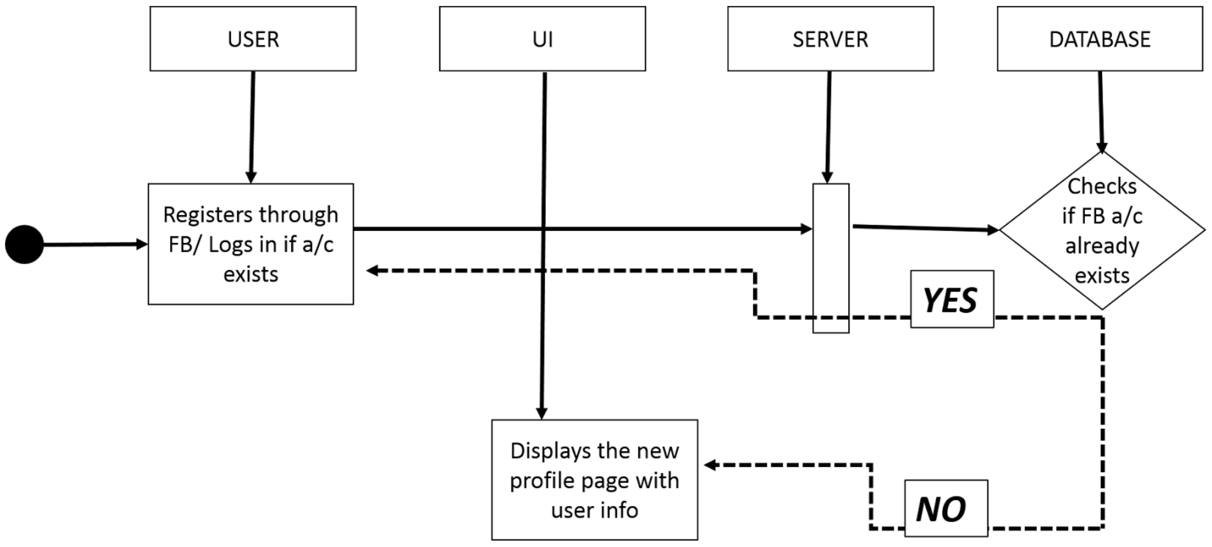
1. **Figure 4.1**

Sequence of events when the user first opens the app



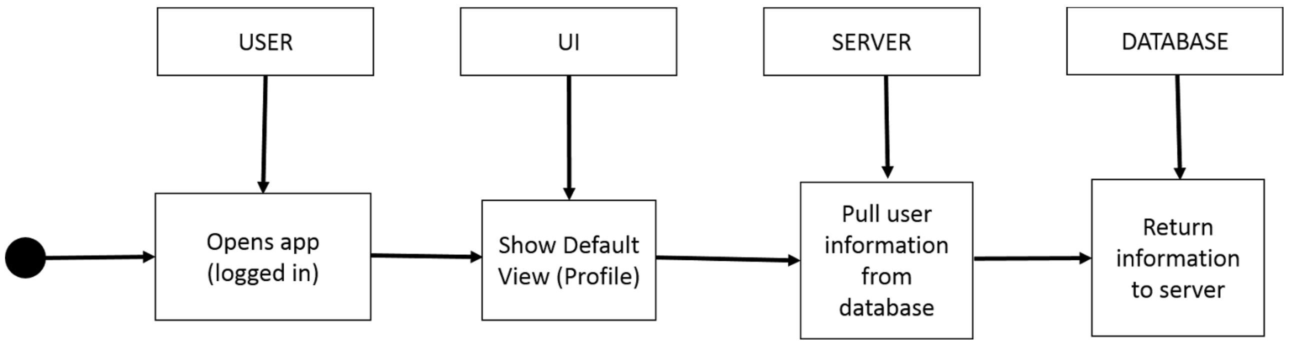
2. **Figure 4.2**

Sequence of events when the user registers for the app


USER
UI
SERVER
DATABASE
Registers through FB/ Logs in if a/c exists
Checks if FB a/c already exists
YES
NO
Displays the new profile page with user info

3. **Figure 4.3**

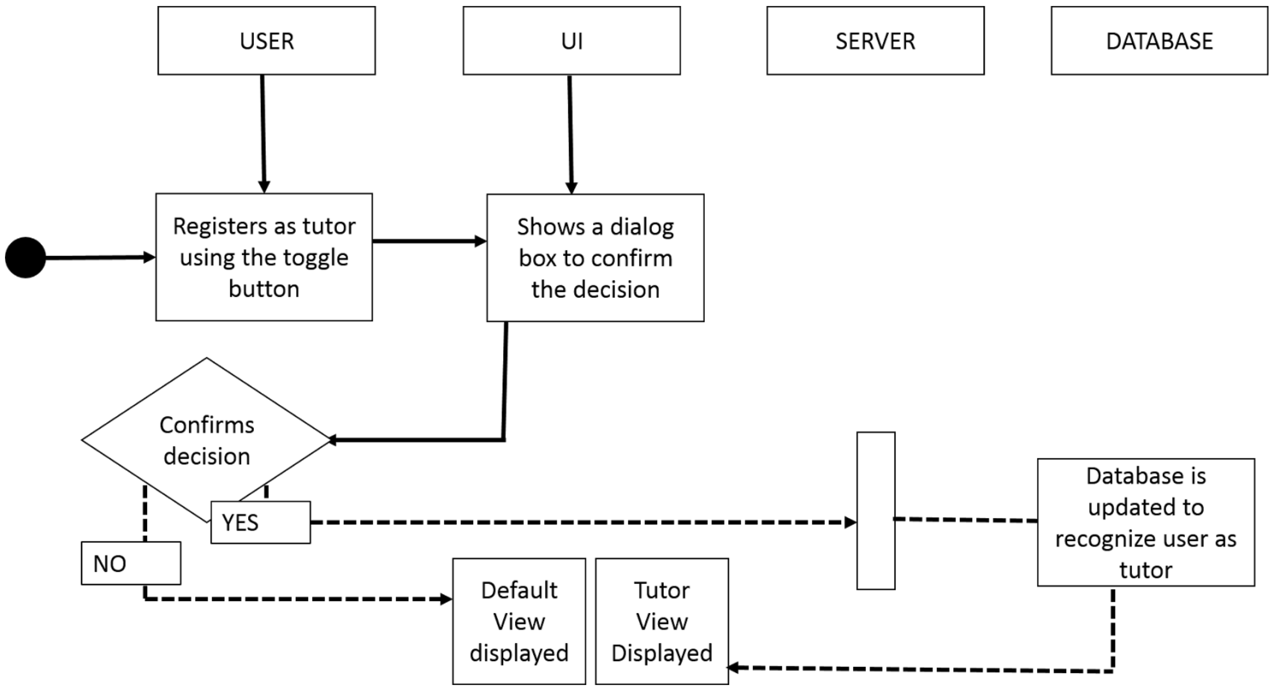
Sequence of events when the user opens the app when he's already registered (or signed in)



USER
UI
SERVER
DATABASE
Opens app (logged in)
Show Default View (Profile)
Pull user information from database
Return information to server

4. **Figure 4.4**

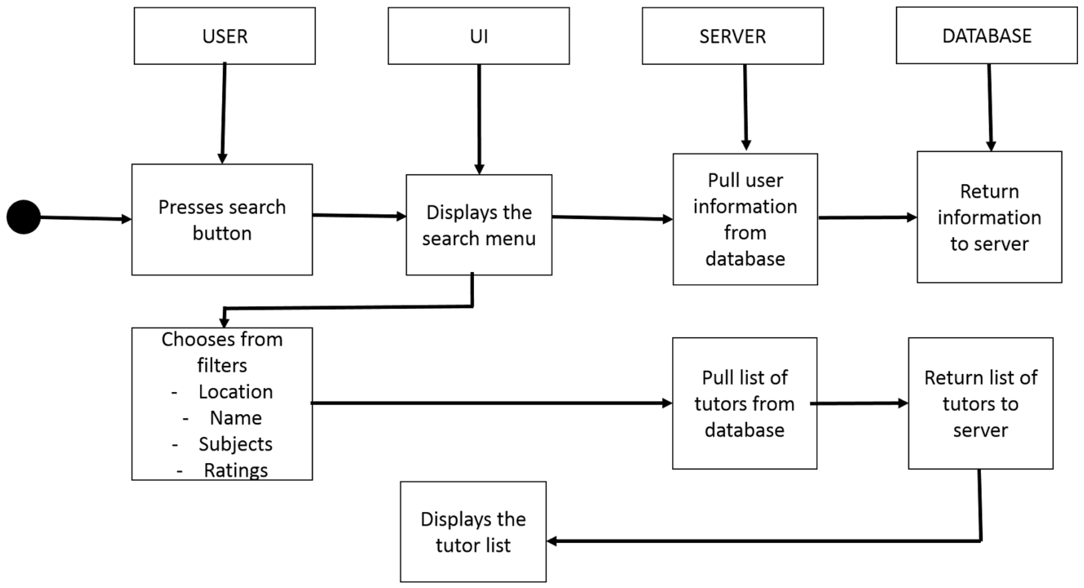
Sequence of events when the user registers as a tutor



USER
UI
SERVER
DATABASE
Registers as tutor using the toggle button
Shows a dialog box to confirm the decision
Confirms decision
YES
NO
Database is updated to recognize user as tutor
Default View displayed
Tutor View Displayed

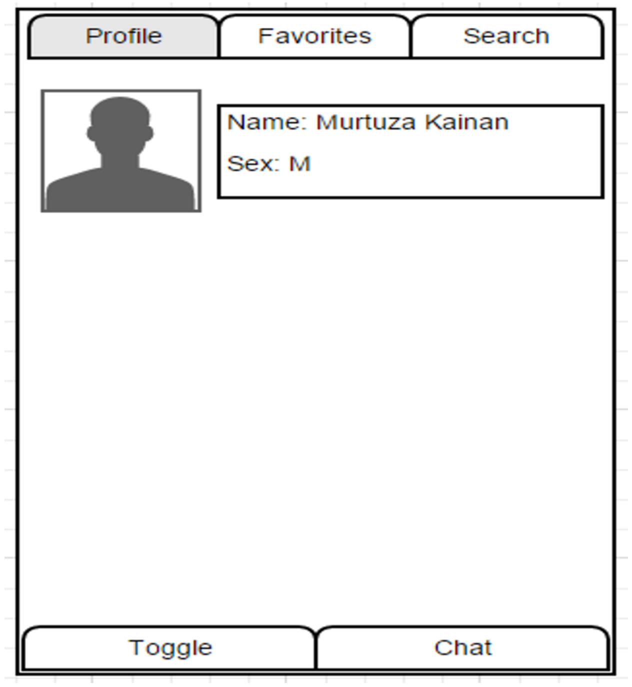
5. **Figure 4.5**

Sequence of events when the user tries to search for tutors

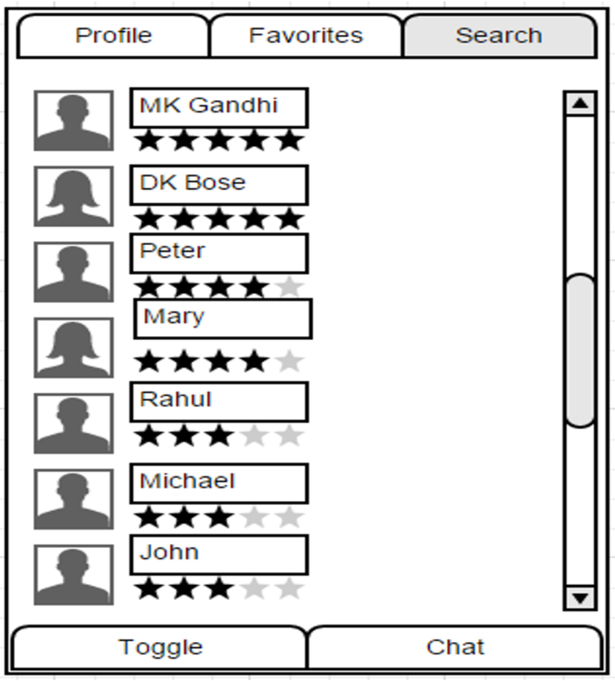


UI Mockup

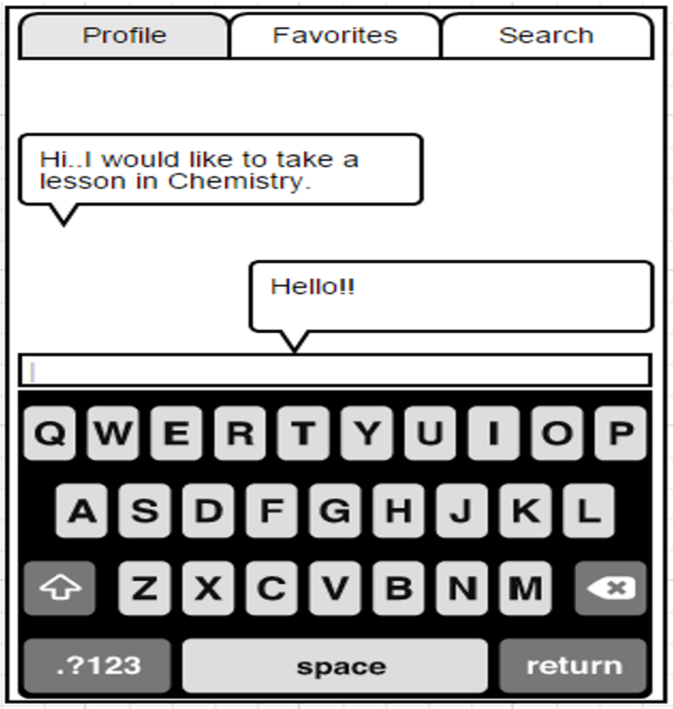
- Profile View



- Search Results



- Chat


Profile
Favorites
Search
Hi..I would like to take a lesson in Chemistry.
Hello!!
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M
.?123
space
return