DTutor Design Document

Arvind Kumaran, Travis Peters, Haider Syed

Overview

Application Description

DTutor is a specialty smartphone app designed **by** students **for** students. The primary goal of this application is to help students of all disciplines find tutors to help them with their courses. We provide an interface that allows tutors and tutees to connect and schedule tutoring sessions.

User Interface

Main Activity

When the user opens the application, they are presented with the Dartmouth authentication page where they have to log in using their student ID and password before they can start using the App. Once they log in, they will be presented with the application, which consists of the home, search and profile tabs, respectively.

As soon the user logs into the application, the local database is synced with the server database. This ensures that the app always has the latest version of the records.

Once the user logs in, they are taken to the Home fragment.

Home Fragment

Once the user enters this fragment, they can sign up as a Tutor, a Tutee or both. If the user has already signed up, this tab will present a toggle switch between Tutor and Tutee. Only users that have signed up for both roles can use the toggle switch to change whether they navigate the app as a tutor or a tutee. Depending on the role that is selected, the search tab will present the user with different search options in the search tab (tutees can only search for tutors; tutors can only search for tutees). If the user is only signed up for one role, they will not be able to use the toggle switch.

The home fragment also feature an "I'm Studying for Course X, Come Join Me" option where users select the course they are currently studying for and opt to share their location with all app users that are enrolled in that course. If the user selects this option, their location coordinates will be stored in the server database and broadcast to all users that are enrolled in Course X.

The home tab will display a list of the courses that the user is enrolled in; clicking a course will bring up a Google Maps with markers identifying the location of all students in that course that are currently studying and have opted to share their location.

This tab will also show requests from other tutors/tutees that are requesting a meeting with the user. For example, tutors will see requests from tutees, and tutees will see requests from tutors. The users will be given the option of accepting the request, rescheduling, or declining the request. They will also be able to review the responses to previous requests they have sent out to other users (i.e. whether they were accepted, declined, or have asked for rescheduling). Before accepting requests, the user will be shown a calendar with the dates requested for the session.

Search Fragment

User is a Tutee:

The user can search by department and course using two spinners that will be on the top of the screen. Based on the values, the local database is queried and a list view is populated with data on the tutors (such as picture, name and a short description). Upon choosing a tutor, they can either contact the tutor through email (or a phone call if the tutor has provided a phone number) or they can schedule a tutoring session directly. If the user opts to schedule a session, they will be shown a calendar populated with the tutor's availability. The requesting user's availability will also be overlaid on the calendar. The user can then choose any date where they tutor is available and submit the tutoring request. In the situation that the tutee finds none of the provided dates suitable, they will be given the option of requesting other dates.

User is a Tutor:

When a tutor opens the Search fragment, they will be able to view the list of all tutor requests made by tutees for a particular subject. Once again they will be able to filter the search by department and course number. Once they click on a particular tutee then they will be able to take a look at the dates requested by that tutee and then submit a request to be their tutor. As with the tutee options, the tutor will also be given a chance to contact the tutee before they finalize the tutoring session.

Profile Fragment

User is Tutee:

When a tutee opens the profile fragment, they will be able to view their personal information as well as a list of their preferred tutors, and an availability calendar. The user will be allowed to edit this profile whenever required and this change will be reflected in the database. If the tutee has any specific details they would like to add about themselves, there will be a comment box that can be used.

User is Tutor:

When a tutor opens the profile fragment, they will be able to view their personal information (name, courses they can tutor, comments, etc.) as well as a list of the all of the courses they can tutor for. The tutors will also have an availability calendar, which they can set.

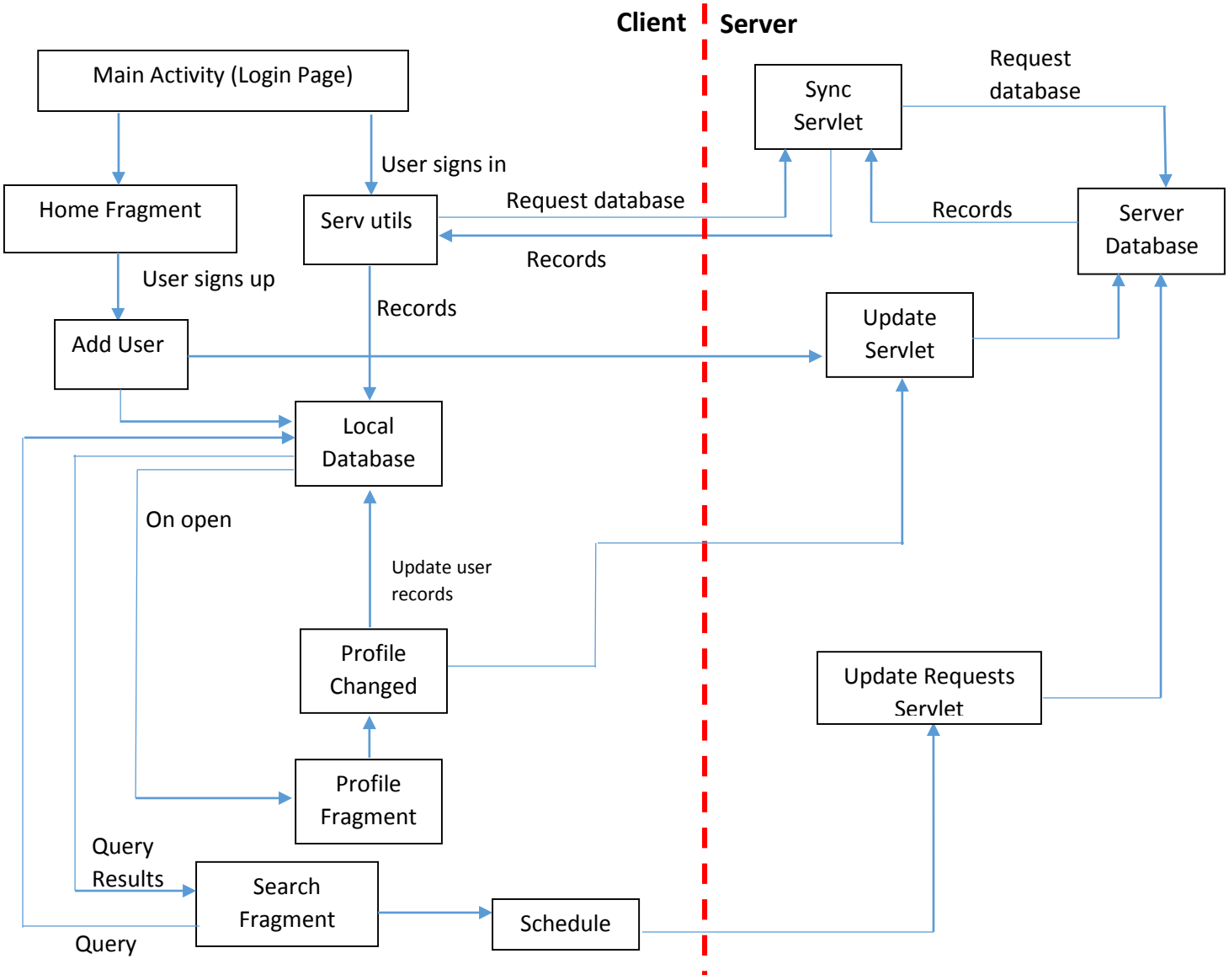
Any changes made by the user in the profile fragment will cause a servlet to be triggered, which will update the remote server database; and the local database will also be updated

Project Scope

We will implement the following components/features in our application:

- Tutor/Tutee sign-up page
    - Students will fill out their basic information, specify courses they want to be tutored for or/and courses they want to tutor, and their fill out their availability.
- Tutor/Tutee search
    - Search for a tutor using the course number.
    - Connect with a tutor from the list of tutors through email (or by phone) for an initial conversation
    - Schedule a session with a tutor

System Design Diagram



Packages & Classes Overview

Data Package

Member

The member class is a data carrier object that holds personal information about the user of the DTutor app.

Application

Super class that provides generic/universal components for Dartmouth Tutor Clearinghouse applications.

TutorApplication extends Application

Subclass of Application that provides specific components for Dartmouth Tutor Clearinghouse Tutor application.

TuteeApplication extends Application

Subclass of Application that provides specific components for Dartmouth Tutor Clearinghouse Tutee application.

StudyGroupLeaderApplication extends Application

Subclass of Application that provides specific components for Dartmouth Tutor Clearinghouse Study Group Leader application.

Utils Package

Update Servlet

Servlet will be used to update the remote server every time there is a change in any of the user data.

Sync Servlet

The servlet will be used to sync all the data from the server database to the local database

UpdateRequestServlet

The servlet will inform the remote server whenever a new tutor request has been made or when a tutor has picked a tutee from the search fragment.

Database Implementation

Data Structures

A primary component of our application is the data that we are handling. In order to build an app that would be consistent with the infrastructure that the Tutor Clearinghouse already has in place to manage their Tutoring system, we need various data carrying classes to collect/store data that they require. The following are our primary data carrier classes:

```
public class Member {

    // Required fields
    private long mId;
    private String mStudentId;
    private String mFirstName;
    private String mLastName;
    private List<String> mRoles;
    private int mClassYear;
    private String mEmail;

    // Optional fields
    private List<Course> mCourses;
    private String mAddress;
    private String mPhoneNumber;
    private String mGender;
    private String mEthnicity;
    private Comments mComments;
    private Lists<tRequests> mRequests;

    private LatLng mLocation;
}
public class Application {

    private long mId;
    private String mAppType;
    private List<Course> mCourses;
    private int mMaxStudents;
    private String mAttrTutorType;
    private String mAttrAthlete;
    private String mGreekAffiliation;
    private String mVisaStatus;
    private Calendar mVisaStartDate;
    private Calendar mVisaEndDate;
    private Boolean mEmploymentFormsSubmitted;
    private String mHeardAboutUs;
    private String mSpecialComments;
}
```

Note: *The majority of the class implementation has been excluded for the sake of brevity, but we have already defined getters/setters (where necessary), various helper methods for working with the objects, and a series of validation methods to ensure data is accurate/complete*

Database Table Schema

Arguably the most important data that is stored in this app pertains to the office`al applications that are filled out by any student that wishes to be tutored, tutor, lead a study group, etc. Below is the database schema for recoding any/all tutoring applications that members submit. Local to the device, a user may have one or even a few applications, however, the power of our system will come into storing all of these records remotely, at which point we will be able to automate the process of tutor/tutee pairing.

```
private static final String CREATE_TABLE_MEMBERS =
                    "CREATE TABLE IF NOT EXISTS " + TABLE_MEMBERS
                        + " ("
                        + COLUMN_ID
                        + " INTEGER PRIMARY KEY AUTOINCREMENT, "
                        + COLUMN_STUDENT_ID
                        + " TEXT NOT NULL, "
                        + COLUMN_FIRSTNAME
                        + " TEXT NOT NULL, "
                        + COLUMN_LASTNAME
                        + " TEXT NOT NULL, "
                        + COLUMN_ROLES
                        + " TEXT NOT NULL, "
                        + COLUMN_CLASS_YEAR
                        + " INTEGER NOT NULL, "
                        + COLUMN_EMAIL
                        + " TEXT NOT NULL, "
                        + COLUMN_MEMBER_COURSES
                        + " TEXT NOT NULL, "
                        + COLUMN_PHONE
                        + " TEXT NOT NULL, "
                        + COLUMN_ADDRESS
                        + " TEXT NOT NULL, "
                        + COLUMN_GENDER
                        + " TEXT NOT NULL, "
                        + COLUMN_ETHNICITY
                        + " TEXT NOT NULL "
                        + COLUMN_COMMENTS
                        + " TEXT NOT NULL "
                        + ");";
private static final String CREATE_TABLE_APPLICATIONS =
                    "CREATE TABLE IF NOT EXISTS " + TABLE_APPLICATIONS
                        + " ("
                        + COLUMN_ID
                        + " INTEGER PRIMARY KEY AUTOINCREMENT, "
                        + COLUMN_APP_TYPE
                        + " INTEGER NOT NULL, "
                        + COLUMN_APP_COURSES
                        + " TEXT NOT NULL, "
                        + COLUMN_MAX_STUDENTS
```

```
                        + " INTEGER NOT NULL, "
                        + COLUMN_TUTOR_TYPE
                        + " TEXT NOT NULL, "
                        + COLUMN_ATHLETE
                        + " TEXT NOT NULL, "
                        + COLUMN_GREEK
                        + " TEXT NOT NULL, "
                        + COLUMN_VISA_STATUS
                        + " TEXT NOT NULL, "
                        + COLUMN_VISA_START
                        + " DATETIME NOT NULL, "
                        + COLUMN_VISA_END
                        + " DATETIME NOT NULL, "
                        + COLUMN_EMPLOYMENT_FORM_SUBMITTED
                        + " INTEGER NOT NULL, "
                        + COLUMN_HEARD_ABOUT_US
                        + " TEXT, "
                        + COLUMN_COMMENTS
                        + " TEXT "
                        + COLUMN_LOCATION
                        + " BLOB "
                        + ");";
private static final String CREATE_TABLE_MEMBERS_APPLICATIONS =
                    "CREATE TABLE IF NOT EXISTS " + TABLE_MEMBERS_APPLICATIONS
                        + " ("
                        + COLUMN_ID
                        + " INTEGER PRIMARY KEY AUTOINCREMENT, "
                        + COLUMN_MEMBER_ID
                        + " INTEGER NOT NULL, "
                        + COLUMN_APPLICATION_ID
                        + " INTEGER NOT NULL "
                        + ");";
```

The autoincremented *_id* is used as the primary key, though the *student_id* is also a unique identifier and may be used if something warrants it (such as remote storage in something like a Google App Engine Datastore).

Implement Database Operations

```
// Database Methods: Setup/Configuring & Handling

public synchronized static DatabaseHelper getInstance(Context context) {
    if(singleton == null) {
        singleton = new DatabaseHelper(context.getApplicationContext());
    }
    return(singleton);
}

private DatabaseHelper(Context context) {
    super(context, DATABASE_NAME, null, DATABASE_VERSION);
}

@Override
public void onCreate(SQLiteDatabase database) {
    database.execSQL(CREATE_TABLE_MEMBERS);
```

```
    database.execSQL(CREATE_TABLE_APPLICATIONS);
    database.execSQL(CREATE_TABLE_MEMBERS_APPLICATIONS);
}

@Override
public void onUpgrade(SQLiteDatabase database, int oldVersion, int newVersion)
{}

// Database Methods: Members

public long insertMember(Member member) {}

public void updateMember(Member member) {}

public void deleteMember(long rowId) {}

public Member fetchMember(long rowId) {}

public ArrayList<Member> fetchAllMembers() {}

public ArrayList<Application> fetchMemberApplications(long rowId) {}

public void deleteMemberApplication(Member member, long rowId) {}

private static Member cursorToMember(Cursor cursor) {}

// Database Methods: Applications

public long insertApplication(Application app) {}

public long updateApplication(Application app) {}

public void deleteApplication(long rowId) {}

public Application fetchApplication(long rowId) {}

public ArrayList<Application> fetchAllApplications() {}

private static Application cursorToApplication(Cursor cursor) {}

// Database Methods: Members-Applications

private long createMemberApplicationBind(long memberId, long appId) {}

private long deleteMemberApplicationBind(long memberId, long appId) {}
```

Service Implementation

BroadcastReciever Service

Whenever the user logs in the server's database is loaded into the application. Given that we do not expect to sign up very often it may not be essential to update the local database with new changes while the user is using the application. However we may set up a service that will implement this real time functionality, which would involve setting up a broadcast receiver as a service that would listen for update notifications from the server. On receiving these notifications we will update our local database with the modified records.

Noteable Technical Details/Algorithms

- One of the features we hope to provide in this app is the ability to automate the process of matching tutors/tutees and study group members with study group leaders. While a lot of the procedure can be scripted, there are often "personal touches" that go into finalizing any matches. For this reason we propose to implement a sort of approval system or recommendation system by which authorized staff can review the system's suggested matchings and have the final say in whether or not a particular matching will be finalized.