Ahsanullah University of Science and Technology

Department of Computer Science and Engineering

Information System Design and Software Engineering Lab

CSE 3224

Try a Tutor

Submitted To

Mr. Md. Tawkat Islam Khondaker	**Mr. Nibir Chandra Mandal**
Lecturer	Lecturer
CSE, AUST	CSE, AUST

Submitted By

Tashfiq Ahmed	**170104038**
Hasanath Jamy	**170104039**
Dibya Nath	**170104049**
Sabiha Benta Sayed Badhon	**170104051**

September 21, 2020

Contents

1. Introduction:

Our project "Try a Tutor" is basically a hub between parents and tutors. It is basically an Android Application. As we all know, online education system is becoming quite popular now a days, so we tried to build up something through our project which will make the whole concept of : "finding a perfect tutor" and " getting a suitable tuition" more efficient and flexible.

2. Database Package:

This package holds the model classes along with some storage classes.

2.1 InternetConnect:

This class holds the methods required to check for the network availability and give proper prompt to the user.

2.1.1 showConnectivityError:

When called informs the user via dialogue that no internet connection is available.

2.1.2 showConnectivityErrorCloasApp:

When called informs the user that no internet connection is available and closes the app.

2.1.3 isNetworkAvailable:

Method used to check both Wi-Fi and mobile internet to find the availability of internet. Returns true when found and false when not.

2.2 AddressData Class:

A storage class that holds the data of a location. These data includes latitude, longitude and detailed address of a point.

2.3 User:

A class used to represent the common properties of the users, like parents and tutors. ParentDatahandler class and TutorDataHandler class are two of it's child classes.

2.4 ParentDataHandler:

A model class used to represent the PARENT table with all it's attributes and constructor. User class is a super class of this class.

2.5 TutorDataHandler:

A model class used to represent the TUTOR table with all it's attributes and constructor. User class is a super class of this class.

2.6 Userfactory:

A class used to implement the factory method design pattern to handle the multiple types of users in the app which are 'Parent' and 'Tutor'.

2.7 ConnectionDataHandler:

A model class used to represent the CONNECTION table with all it's attributes and constructor.

2.8 JobApplicationDataHandler:

A model class used to represent the JOB_APPLICATION table with all it's attributes and constructor.

2.9 JobBoardDataHandler:

A model class used to represent the JOB_BOARD table with all it's attributes and constructor.

2.10 ParentNotificatonDatahandler:

A model class used to represent the PARENT_NOTIFICATION table with all it's attributes and constructor.

2.11 TutorInvitationHandler:

A model class used to represent the TUTOR_INVITATION table with all it's attributes and constructor.

2.12 TutorNotificationDatahandler:

A model class used to represent the TUTOR_NOTIFICATION table with all it's attributes and constructor.

2.13 SharedPreferencesManager:

A class used to manage the shared preference associated with the app.

2.13.1 createParentLoginSharedPreferences:

Takes the information of a parent when he/she logs in or sign up and creates a shared preference to store the data.

2.13.2 createTutorLoginSharedPreferences:

Takes the information of a tutor when he/she logs in or sign up and creates a shared preference to store the data.

2.13.3 getParentDataSharedPreferences :

Returns the stored parent data in the shared preference.

2.13.4 getTutorDataSharedPreferences :

Returns the stored tutor data in the shared preference

2.13.5 checkLogin:

Returns true if a user data is stored in the shared preference meaning that a user is already logged in.

2.13.6 getLoginAccount:

Returns the account Type of the logged in account and returns a null string when no logged in account is found.

3. LoginAndRegistration Package:

This package holds the controller classes associated with the login and sign up

3.1 LoginActivity :

Controller class that handles the logics behind login.

3.1.1 validateData:

Used to validate the information inserted by the user such that it meets the correct format. Returns true when every element is in the right format

3.1.2 checkDataAndLogin:

Checks the inserted information n the database to confirm that the data are correct before logging in to the desired account. Fetched data from the database and creates a shared preference

3.2 SignUpActivity:

Controller class that handles the logic behind sign up.

3.2.1 isDataValid :

Used to validate the information inserted by the user such that it meets the correct format. Returns true when every element is in the right format.

3.2.2 checkPhoneAndSignUp :

Checks the inserted data such that it's meets the requirements for the correct format and also confirms that no user of the same type with the same phone number exists in the database. When all the conditions meet, the a new user is created with all the credentials and the user is logged in.

3.3 SignUpUserSelectionActivity:

A class that directs the user according to the account he/she wants to log in with.

4. Shared Package:

4.1 MainActivity :

Permissions for FINE LOCATION and COARSE LOCATION were asked to be granted in this class

4.2 SplashScreenActivity:

On the basis of data stored in the shared preferences it was determined if there was any account logged in and also the type of the account. If found then the next activity directly takes to that account or it takes to the login screen when not found.

4.3 GoogleMapActivity:

Google Map API was used to find location in the map when required.

4.3.1 sendData:

: It was determined which activity called the GoogleMapActivity by receiving a unique string from each of the calling classes as a signature and the required data was sent back those activities.

4.3.2 createMapWithLocation :

Asks permissions for the FINE LOCATION and COARSE LOCATION and if granted, it provides user's current location data.

4.3.3 onMapReady:

A callback method to execute the Google Map API and put the user's current location on the map with a marker. Upon interaction with the map this method detects it and puts a marker on the newly selected point on the map and stores the currently selected location with additional data.

4.3.4 getAddress:

Returns the additional data from the map from the location selected by the user.

5. Parent Package:

This package holds the classes that are associated with the 'Parent' account.

5.1 ParentHomeActivity:

This class holds methods that control the fragment container for all the fragment views associated with the parent home view as well as the toolbar, bottom navigation bar and left side sliding drawer.

5.1.1 parentBottomNavBar :

Takes user to the desired fragment on the basis of interaction with the bottom navigation bar. Initially the parent notification window is set to be shown.

5.1.2 parentDrawerListener :

Takes user to the desired Page on the basis of interaction with the left sliding drawer.

5.1.3 Logout:

Method used to log out of the current account. It clears the shared preference of the user's data and takes the user to the logic screen.

5.2 ParentNotificationFragment:

This class holds the methods that control the recycler view that shows the notification received by the current parent type user.

5.2.1 fetchData:

Based on the parent id stored on he shared preference, data is fetched from the PARENT_NOTIFICATION table in the database to show on the recycler view.

5.2.2 swipeToDelete:

: Upon user's left swipe interaction with the recycler view this method is called that determines if the user wants to delete row by invoking a snackbar showing a prompt to undo the operation. If left unattended this method then proceeds to call deleteData method.

5.2.3 deleteData:

When called, this method deletes the notification with the notification id passed in the parameters according to the user's requirement.

5.3 ParentJobBoardFragment:

This class holds the methods that control the recycler view that shows all the job posted by the parent type user.

5.3.1 fecthData:

Based on the parent id stored on the shared preference, data about the job posts are fetched from the JOB_BOARD table and are later shown on a recycler view.

5.3.2 swipeToDelete:

Upon user's swipe interaction with the recycler view this method is called and it determines if the user wants to delete that row by invoking a snackbar showing a prompt to undo the operation. If left unattended this method then proceeds to call deleteData method.

5.3.3 deleteData:

When called, this method deletes the job with the job id passed in the parameters according to the user's requirement.

5.4 ParnetFindTutorFragment:

This class holds the methods that allows the user to search for tutors according to their requirements.

5.4.1 validateData:

Used to validate the information inserted by the user such that it meets the correct format. Returns true when every element is in the right format.

5.4.2 fetchTurorData:

: Fetches all the tutor data from the database so that later the tutors that match the user's requirement can be separated.

5.4.3 getRequiredTutor:

Finds and separates data about all the tutor that meets the requirement from previously stored tutor data.

5.5 ParentJobApplicationFragment:

This class holds the methods that control the recycler view that shows the job applications received by the current parent type user.

5.5.1 fetchApplicationData:

Fetches all the application data from the JOB_APPLICATION table in the database associated with the current logged in parent type user.

5.5.2 swipeToDelete:

Upon user's swipe interaction with the recycler view this method is called and it determines if the user wants to delete that row by invoking a snackbar showing a prompt to undo the operation. If left unattended this method then proceeds to call deleteData method.

5.5.3 deleteData:

When called, this method deletes the job application with the job id passed in the parameters according to the user's requirement. Then a notification with the REJECTED status is added to the TUTOR_NOTIFICATION table for the tutor associated with a the application.

5.6 ParentEditProfileActivity:

This class holds the methods that are used to change and update the current user's data on the database. It receives the current user's data from the shared preference.

5.6.1 isChanged:

Checks and returns true if confirms that the user wants to change his/her current information and false otherwise.

5.6.2 validateData:

Used to validate the information inserted by the user such that it meets the correct format. Returns true when every element is in the right format

5.6.3 getNewParentData:

The new data entered by the user is taken and updated under the currently logged in user's id.

5.7 ParentPostedJobDetailsActivity:

This class holds the methods that are used to show the job details, that are posted by a parent type user to himself in details. The job id associated with the job is passed via intent.

5.7.1 loadData:

Fetches all the data about the job associated with the received job id from the JOB table about the currently logged in user.

5.8 PostNewJobActivity:

This class holds the methods that are used to allow the user to post new jobs and write data into the database about the new job. Upon interaction with a button in the view the data of the post is updated on to the JOB table in the database.

5.8.1 validateData:

Used to validate the information inserted by the user such that it meets the correct format. Returns true when every element is in the right format.

5.9 TutorDetailActivity:

This class holds the methods that are used see tutor details, control the contact information of the tutor that a parent can see, allow the parent to send invitation and also allow a parent to accept or reject a tutor's job application.

5.9.1 fetchTutorData:

Fetches all the data about the tutor from the TUTOR_TABLE from the database.

5.9.2 prevInvitation:

He associated tutor type user's id is checked with a parent type user's id to determine that if there is already a pending invitation in the database for the tutor sent by the parent. If such a case occurs the method returns true and false otherwise.

5.9.3 getTutorConnection:

The associated tutor type user's id is checked with a parent type user's id to determine that if the tutor is already connect to the parent. If such a case occurs the method sets the value of a global variable true and false otherwise.

5.9.4 sendInvitationDialogue:

This method invokes a custom dialogue with some edit text fields allowing the user to send a job invitation to the desired tutor. It checks and validates the data entered in the dialogue by the user.

5.9.5 sendInvitation:

Upon user's interaction with an application in the view a notification is inserted in the TUTOR_NOTIFICATION holding the type of interaction status passed on to this method as parameter.

5.9.6 createConnection:

Upon user's interaction with an application in the view, if the interaction type in ACCEPTED then a connection is added between the tutor and the parent n the CONNECTION table in the database.

6. Tutor Package:

This package holds the classes that are associated with the 'Tutor' account.

6.1 TutorHomeActivity:

This class holds methods that control the fragment container for all the fragment views associated with the tutor home view as well as the toolbar, bottom navigation bar and left side sliding drawer.

6.1.1 tutorBottomNavBar:

Takes user to the desired fragment on the basis of interaction with the bottom navigation bar. Initially the tutor notification window is set to be shown.

6.1.2 tutorDrawerListener:

Takes user to the desired Page on the basis of interaction with the left sliding drawer.

6.1.3 Logout:

Method used to log out of the current account. It clears the shared preference of the user's data and takes the user to the logic screen.

6.2 TutorNotificationFragment:

This class holds the methods that control the recycler view that shows the notification received by the current tutor type user.

6.2.1 fetchData:

Based on the tutor id stored on he shared preference, data is fetched from the TUTOR_NOTIFICATION table in the database to show on the recycler view.

6.2.2 swipeToDelete:

: Upon user's left swipe interaction with the recycler view this method is called that determines if the user wants to delete row by invoking a snackbar showing a prompt to undo the operation. If left unattended this method then proceeds to call deleteData method.

6.2.3 deleteData:

When called, this method deletes the notification with the notification id passed in the parameters according to the user's requirement.

6.3 TutorJobBoardFragment:

This class holds the methods that allows the user to search for jobs according to their requirements.

6.3.1 validateData:

Used to validate the information inserted by the user such that it meets the correct format. Returns true when every element is in the right format.

6.3.2 fetchJobData:

Fetched all the job data from the database so that later the job that match the user's requirement can be separated.

6.3.3 getRequiredJob:

Finds and separates data about all the jobs that meets the requirement from previously stored job data.

6.4 TutorJobInvitationFragment:

6.4.1 fetchData:

Based on the tutor id stored on he shared preference, data is fetched from the TUTOR_INVITATION table in the database to show on the recycler view.

6.5 TutorEditProfileActivity:

This class holds the methods that are used to change and update the current user's data on the database. It receives the user's current data from the shared preference.

6.5.1 isChanged:

Checks and returns true if confirms that the user wants to change his/her current information and false otherwise.

6.5.2 validateData:

Used to validate the information inserted by the user such that it meets the correct format. Returns true when every element is in the right format

6.5.3 getNewTutorData:

The new data entered by the user is taken and updated under the currently logged in user's id.

6.6 TutorInvitationDetailActivity:

This class holds the methods that are used to see invitation details and interact with an invitation

6.6.1 createConnection:

Upon user's interaction with an invitation in the view, if the interaction type is ACCEPTED then a connection is added between the tutor and the parent n the CONNECTION table in the database.

6.6.2 sendNotification:

Upon user's interaction with an invitation in the view a notification is inserted in the PARENT_NOTIFICATION holding the type of interaction status passed on to this method as parameter.

6.6.3 deleteInvitation:

Upon user's interaction with an invitation in the view the data about the application from the TUTOR_INVITATIOPN table is deleted using this method.

6.7 TutorJobDetailActivity:

This class holds the methods that allows a tutor to see a job post in detail and control his interaction based on his current connection with the job poster parent.

6.7.1 getTutorConnection:

Check if the tutor type user is already connected with the parent type user that posted the job by checking the database. Returns true if there is a connection and false otherwise.

6.7.2 fetchUserSentApplications:

Check if the tutor type user has already sent a job request to that post by fetching all the sent application by that tutor from the JOB_APPLICATION table. Returns true if a previous application for that same job under the same user was found pending.

6.7.3 sendJobApplication:

This method is used to insert a new application for the tutor in the JOB_APPLICATION table under for the required job post.

7. Conclusion:

In traditional teaching system, students and teachers join together in class rooms in schools, colleges for conducting classes. Sometimes students struggle with a particular subject and in that case, a tutor can be of great service to them. However, finding a perfect tutor according to preference can be difficult for a number of reasons. Similarly, from the tutor's perspective, sometimes finding a suitable tuition job gets so difficult for them. So we tried to make a suitable platform for both the parent and the tutor which will serve them to a great extent.