|  |  |  |  |
| --- | --- | --- | --- |
| **CS102** | **Spring 2015/2016** | Project Group | g3A |
| Instructor: | **Hakan Ferhatosmanoğlu** |
| Assistant: | Hüseyin Celal Öner |

|  |  |  |
| --- | --- | --- |
| **Criteria** | **TA/Grader** | **Instructor** |
| Presentation |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| Overall |  |  |

~ UGotTime? ~

102ERS

Berfu Anıl - Berk Mandıracıoğlu - Emre Gürçay - Kaan Sancak Mehmet Ali Altuntaş - Özkan Tuğberk Kartal

|  |
| --- |
| Detailed Design Report  ( Revised )  1 May 2016 |

# Introduction

We realized that arranging your time at Bilkent is very difficult because of the homeworks, exams, projects so we have decided to create an android application that helps Bilkent students to use their time more efficiently. With our application, UGotTime, students can see common free time with their friends and arrange their schedules individually. Also they can search for other students and they can see others’ detailed information department name and etc.

# System Overwiew

Our application will be an android social media application. In order to build UGotTime, we will use Android Studio to design our user interface and GUI. The collected data such as different users, events and schedules will be stored in a database which we will implement by using PhP. Then we will connect the database to our application through Android Studio. Considering MCV, our database will behave like a Model. Any changes that has done will be updated in database so users can keep their choices (such as password, schedule etc.).

# Core Design Details

## Description of Classes and Their Methods

### Help Page

Help page enables user to have a general understanding of ow the app is working. By creating a Tab view we were able to show different aspects of the app in a single activity.

We believe that users need to see these little informations in a user friendly way. First tab is AboutUs which gives brief information about our app. Second tab is Terms and conditions which states the conditions that users need to be aware of. Third tab is describes the sign up conditions to application.

**Methods**

**1)addTab()**

Enables us to add a new tab to our tab host

**2)newTabSpec()**

Enables to set new Specification to the tab, it takes a String as parameter

**3)setIndicator()**

Specifies a label as the tab indicator name. It takes a String as parameter.

**4)setContent()**

Enables us to specify the id of the view that should be used as the content of the tab.

**5)onCreate()**

This method is called when the activity is initially started, in it you can do the necessary operations for your activity

### Welcome Page

User is greeted in welcome page where they can view our beautiful logo and go on deeper in our application. We wanted this page to be user friendly so we kept the the design simple but elegant. In this page there are three buttons first button (goLog) intents Welcome Page with LoginActivity when user clicks on it. The other button(goSign) intents Welcome Page with SignIn when user clicks on it. Last button(goHelp) intents Welcome Page with HelpPage when user clicks on it. Of course we have set onClick listener for each button in order to make the intents when clicked.

**Methods**

**1)setContentView(int)**

Enables to set the view of the activity by id. it takes int as parameter

**2)setImageResource(int)**

Enables to set the image to a picture file that has id in our app.It takes int as parameter

**3)startActivity(int)**

Runs the activity when called, it takes intent as parameter

**4)setOnClickListener ()**

Sets OnClick method in our app to a specific widget

**5)onCreate()**

This method is called when the activity is initially started, in it you can do the necessary ope rations for your activity

**6)onClick()**

This is called for the operations when a widget is clicked

### SignIn

SignIn activity enables user to insert their data in our database in order to make them a member for our application. User enters their account information and if the infos that are entered by the user are valid, user gets placed in our database. In this class we have bunch of properties. First of all we require to create json objects in order to execute our php script from our application.Therefore we import the json classes to our app. Secondly, we need va- lues to insert into databases we simply ask for values from user using EditText.When we re- ceive the values first we check if they filled all the files later we check wether they are valid. If they are valid, we create a SignUpRequest object that we coded in order execute the query then we put it on RequestQueue in order to waiter the response. If response is successful we intent the page to LoginActivity. Else we give the option to enter the values again.

**Methods**

**1)setOnClickListener()**

Sets OnClick method in our app to a specific widget

**2)startActivity (int)**

Runs the activity when called, it takes intent as parameter

**3)setMessage()**

Sets the message that will be show when sign in fails it takes String as Parameter

**4)create()**

Creates the error message

**5)show ()**

Shows the error message on screen

**6)add ()**

Adds a request to queue in order to wait for its response

**7)onCreate()**

This method is called when the activity is initially started, in it you can do the necessary operations for your activit

### SearchAdapter

SearchAdapter enables us to create new row for each user that is found in the database to list in our app. We use android’s ArrayAdapter class in order to set our search adapter. Such an adapter enables our search to be dynamically viewed by the user to select if the find their search users. GetView method enables us to return and use the changed view with the listed users on it.

**Methods**

**1)setText(String)**

Sets the text of the TextField

**2)setImageResource()**

Enables to set the image to a picture file that has id in our app.It takes int as parameter

**3)getItem() : User**

Returns the User in the specific location in list view

### MainUserPage

This activity, class, actually is the class that has nearly most of the functions, if the user logs in successfully the user comes to the MainUserPage. Basically, MUP(MainUserPage) has three tabs which holds different activities. In the first tab which we named as “ Profile” the user can see its basic information such as its name, surname, department and also there ex ists a image view which holds the user’s picture. However it a default image for now, we cannot upload unique images for each user. In the profile tab we also have two buttons. First one is “More Information” button which basically navigates to the more information page and also “Logout” button which logs out from the profile. Moreover, second tab which we called “Schedule” tab holds the schedules of users , it contains 45 checkboxes( each 9 be longs to one day(except weekend) ) We also have a “arrange schedule” button in this tab which enables or disables checkboxes so we can arrange our schedule, ıf user clicks on the button the text changes and becomes “complete arrangement” so user can update its schedule. Finally on the last tab which we called “Search” tab users can see other users and they can go to their profile.

**Methods**

**1)onCreate(Bundle)**

As it does in the other activities it creates the activity, sets the layout and finds the variables in the layout through (findViewById) method.

**2)onStart()**

When the activity starts, this methods checks whether a user logged in or not through another method(authenticate())

**3)displayUserPage()**

Basically, it sets texts of the TextView that we find in the onCreate() method. It does it by getting the user from userLocalStore.

**4)setTimeTable(ArrayList<String>)**

It gets the users current schedule and add them into the checkboxes array of the class.

**5)authenticate() : Boolean**

This methods checks if the users loggedin and returns a Boolean value

**6)onClick Method(View)**

Basically, this method navigates the user to the selected activity.

**7)setUsersListView(ArrayList<User>)**

This method takes a user ArrayList and displays them in a listview.

**8)findSearchedUsers()**

This method searches the user via search view’s input text.

**9)updateSchedule()**

Every time user arranges its schedule, this method updates the schedule information of the student on the database

**10)setSchedule()**

This method sets the schedule according to the data from database.

**11)checkSchedule()**

This methods checks the current condition of the checkboxes and return an ArrayList which holds the context of the checkboxes

**12)findCheckBoxes()**

This method finds the checkboxes from the layout

**13)setCheckBoxes()**

This method adds checkboxes in an arraylist of checkboxes.

### MoreInformationPage(User)

In this activity we are viewing the detailed information of a user. In this page we are using the LocalStore of our application where we keep the data of the user and display the infor- mations that we didn’t display on our Main User Page. This way we are able to view the users more detailed. This page is intended from the main user page.

**Methods**

**1)setText(String)**

Initialises the text with given string parameter

**2)getUserRank()**

Returns the user rank

**3)getName()**

Returns the name of user

**4)getSurname()**

Returns the surname of user

**5)getEmail()**

Returns the email of the user

**6)onCreate()**

This method is called when the activity is initially started, in it you can do the necessary operations for your activit

### SearchedUserMoreInformationPage

This activity is used for showing the details of the user that is searched from the database. It is intended from the profile page of the searched. We transfer the data of the data from the previous activity and we are viewing the details that we didn’t view on the profile page.

**Methods**

**1)getIntent()**

Returns the intent of the activity

**2)getStringExtra()**

Returns the value of the intended variable

**3) setText(String)**

Sets the text of the TextField

**4)onCreate()**

This method is called when the activity is initially started, in it you can do the necessary operations for your activit

### SearchedUserLocalData

In this class we are keeping the datas of the searched User in SharedPreferences of our app for later use in variable activities such as searched user main profile page. We import and roid content and context classes in order to keep the values in our app data. Moreover we use getters and setters to receive and alter the data. We have clearUserData() method to erase the data from the memory.

**Methods**

**1)getSharedPreferences()**

Returns the preferences that is settled locally on the device

**2)edit()**

Enables to edit the shared preferences

**3)putString(String)**

Takes String as parameter and sets the value of given variable

**4)commit()**

Saves the alterations that is made on SharedPreferences

**5)clear()**

Resets the data on the shared preferences

**6)setUserLoggedIn(boolean)**

takes boolean parameter handsets the loggedIn to the boolean value of the parameter

**7)getUserLoggedIn()**

Returns the user that is logged in

**8)storeUserData()**

Takes user object as parameter and stores it

### UserLocalStore

In this class we are keeping the datas of the user in SharedPreferences of our app for later use in variable activities. We import android content and context classes in order to keep the values in our app data. Moreover we use getters and setters to receive and alter the data. We have clearUserData() method to erase the data from the memory.

**Methods**

**1)getSharedPreferences()**

Returns the preferences that is settled locally on the device

**2)edit()**

Enables to edit the shared preferences

**3)putString(String)**

Takes String as parameter and sets the value of given variable

**4)commit()**

Saves the alterations that is made on SharedPreferences

**5)clear()**

Resets the data on the shared preferences

**6)getString()**

Returns the string value of a SharedPreferences object

**7)setUserLoggedIn(boolean)**

Takes boolean parameter handsets the loggedIn to the boolean value of the parameter

**8)getUserLoggedIn()**

Returns the user that is logged in

**9)storeUserData(User)**

Takes user object as parameter and stores it

### LoginActivity

Here we have two EditText fields to get value from the user. We are getting these values and trying to get the data from our databases to intent to their mainPage. We require to create json objects in order to execute our php script from our application.Therefore we import the json classes to our app. First we create a login request in order to receive the result of our php script.Later we put this request to request queue and wait for the message from our ser ver. Then if the login is successful then we intent this page to main user page. Else we we create an error message using AlertDialog.Builer and then user can enter their informations again.

**Methods**

**1)onResponse (String)**

Takes String as parameter and if response is successful intents to main page

**2) setOnClickListener()**

Sets OnClick method in our app to a specific widget

**3)getString()**

Returns the string value of the response

**4) finish()**

Finishes the activity

**5)setMessage(String)**

Sets the message that will be show when sign in fails it takes String as Para meter

**6)create()**

Creates the error message

**7)show()**

Shows the error message on screen

**8)onCreate()**

This method is called when the activity is initially started, in it you can do the necessary operations for your activity

**9)onClick()**

This is called for the operations when a widget is clicked

### SignupRequest

SignupRequest extends StringRequest of volley classes to make a request. We are putting the values which were entered by user in the Signin activity, as parameter into our construc tor in order to insert values into our database and to see if the request was successful. We use this class when we are accepting new users to our app.

**Methods**

**1)put(String)**

Takes String parameter and adds the string to map

**2) getParams()**

Returns the map that was created in the constructor

### UpdateUserScheduleRequest

We use this class in order to execute our php script which is for updating the schedule of a user in the database. We are putting the time\_id as parameter into our constructor to up date the time slot. Applying a loop enables us to update multiple time slots that is altered by the user. We use this class when we are viewing the schedule on our app.

**Methods**

**1)put(String)**

Takes String parameter and adds the string to map

**2) getParams()**

Returns the map that was created in the constructor

### ScheduleRequest

In this class we are using volley classes to make a request. We are putting the time\_id as pa rameter into our constructor to get the the specific time in the schedule and to see if the request was successful. Later, we use this class when we are viewing the schedule on our app. By creating a loop to get all the time slots and view it in our app.

**Methods**

**1)put(String)**

Takes String parameter and adds the string to map

**2) getParams()**

Returns the map that was created in the constructor

### NewScheduleRequest

Here we are using volley classes of android in order to use its StringRequest class. First we are connecting the web service where our php script is, with the android app. first we put user id into it as a parameter then later we use it for creating a new row in Schedule table of our database of schedule for the new user.

**Methods**

**1)put(String)**

Takes String parameter and adds the string to map

**2) getParams()**

Returns the map that was created in the constructor

### LoginRequest

In this class we are connecting to our web services which has PHP script required for SELECTING user from the database. We are using taking email, pw and more importantly the url of the php script. We are using the volley class in order to assign request to the web server. We use this class in our LoginActivity to get the values of the user and to know wet her the Login is successful or not.

**Methods**

**1)put(String)**

Takes String parameter and adds the string to map

**2) getParams()**

Returns the map that was created in the constructor

# Task Assignment

## Berfu Anıl

Coding

Debugging

## Berk Mandıracıoğlu

UI Design (Some Parts)

Coding (Some parts)

## Emre Gürçay

UI Design (Some Parts)

Coding (Some Parts)

## Kaan Sancak

Construction of database

Coding (Some Parts)

## Mehmet Ali Altuntaş

Reports

Coding (Some parts)

## Özkan Tuğberk Kartal

Reports

Coding (Some Parts