COP290 - Moodle Android App: Design Document

Aditi(2014CS10205), Ayush Bhardwaj(2014CS10091), Nikhil Gupta(2014CS5140462)

February 24, 2016

1 Overall Design

Our overall design of the Moodle app would consist of three main activities. The first one would be the splash screen, which would be displayed for a fixed amount of time. The next activity is the main login page, where there are separate fields to login as a student or as a teacher. After the user logs in successfully, the main dashboard appears. From there, the user can navigate to the activity for each course. Also, there are separate activities for displaying details of the assignments and threads.
As per the requirements of the assignment,

- Different layouts [3] have been made for tablet and mobile phone view for all the three activities.
- Different classes have been defined in separate files to maintain modularity in code.
- **Doxygen** has been implemented to create an HTML documentation of the application structure.
- Various resources, such as strings, images and styles, have been included in separate files.
- Volley has been used to send requests and receive responses.

2 User Interface

2.1 Splash Screen

The files activity_moodle_plus.xml and content_moodle_plus.xml define the layout of the splash screen. It consists of a big image that contains the logo and name of the application. An animation [1] has been applied on the app logo to make it grow and shrink. Below is the screen shot of the splash screen in tablet view.

2.2 Design Of Login Activity

This page appears just after the splash screen. It contains fields for the user to enter his login details, alongwith a LogIn button.

- The layout of main login page has been defined by activity_login_page.xml and content_login_page.xml files in the layout folder.

The following are the screen shots of the main page on different orientations as well as screen sizes [2]:

2.3 DashBoard (MainActivity)

It appears when the user has been logged in successfully, as a student. It contains all the main information about the student and his courses. It is divided into multiple tabs, as explained below:

- **Overview** : It contains the details about the student's semester, year, alongwith his user name.

- **Grades** : It displays the list of the student's grades, of all courses. Each item in the ListView expands on clicking, to display the details of the particular exam.
- **Courses** : It contains the list of the courses the student is enrolled in.The elements in the list are clickable, and a new page(CourseTab) containing the details of the particular course opens up on clicking on a ListItem.
- **Notifications** : It displays the list of the recent notifications for the user. A CustomListView has been used to improve the interface.

The tabs are horizontally scrollable, as they cannot fit simultaneously on all screen widths.
To make the navigation among various sections of the app easier, a **Navigation drawer** has also been included in the MainActivity activity. It appears on clicking on the icon on the top left, or also by swiping to the right on the screen. Below are the screenshots of the various tabs of the dashboard, with and without the navigation drawer.

2.4 Course Page (CourseTab)

This page appears on clicking on any course in the list on the dashboard. The layout of theis page is defined by the files, activity_course_tab.xml and content_course_tab.xml. It contains a TabLayout, comprising of the five fragments, listed as follows:

- **Overview** : It contains the brief information about the activities in the course in the past weeks.
- **Assignments** : This tab displays a list of the assignments corresponding to the particular course. Each item in the ListView expands on clicking, to show the title and deadline for the assignment. On clicking on the expanded view of the item,another activity, that displays the details of the assignment, opens up.
- **Grades** : It contains the list of grades of the student in the selected course. It is again an expandable list view, which on clicking, display the details of the particular examination of the course.
- **Threads** : This fragment contains the option of posting a new thread. Also, it displays a list(Custom ListView) of the previous threads posted regarding the course. Each item in the custom list view is clickable, and opens another activity, to display the description, and comments made on the selected thread.
- **Resources** : It simply displays the faculty name and a button to upload a private resource.

The following are the screen shots for the CourseTab page:

2.5 Assign_details Page

This page appears on clicking on the expanded list item of a particular assignment. It contains all the details, including description, deadline, late days, and past submissions of the assignment. It also contains a button to submit another file.

2.6 thread_details Page

This page appears on clicking on a particular thread in the list of threads of a course in the CourseTab page. It contains the user name of the person who posted the thread, along with its description, time of creation and updation. Also, it displays the list of the comments made on the thread previously. The user also has an option of podting a new comment on the thread.

3 Implementation Details

The application structure is divided into two parts, one set of classes to handle the view of various fragments, and the other set forms the back-end and application flow.

3.1 Layout Handling

These classes are used to populate data in the fragments created in the CourseTab and the MainActivity page.

3.1.1 CustomListAdapter

This class has been made to implement a custom List View. It extends the BaseAdapter class, which basically handles the population of data into the items of the ListView. The description of its fields and methods(other than the auto generated methods) is as follows:

Listing 1: Fields & Methods of CustomListAdapter

```
class CustomListAdapter
{
        List<String> Titles;      // list of strings to populate title textbox
        List<String> Times;           // times of creation/latest update
        List<Integer> Serial;     // serial numbers
        Context context;              // main application context
        LayoutInflater inflater; // to inflate the listview item

        public:
        View getView(int, View, ViewGroup);
        // populates comment_item layout, inflates view

};
```

3.1.2 CustomAdapter_Thread

This class is similar to CustomListAdapter, but populates a different ListView item layout. It has three lists to populate different text views in the layout defined in the file customlistitem.xml.

3.1.3 Course_Overview

It does not contain any method/field other than the auto generated ones. This fragment simply displays the course name.

3.1.4 Course_Assignments

This is a fragment in the CourseTab page. It displays the list of assignments in the form of an Expandable List view. The adapter for the list view is set to an instance of the class, ExpandableListAdapter.
It stores a list for the headings and a hashmap for the two children(Title, time of creation of assignment int this case)of each list item. It has the following methods to populate the list view:

- *prepareListData* : Takes as input two arrays, containing data corresponding to two children of each list item. Sets the list and hash map fields.
- *onCreateView* : Receives the **Bundle** passed on from the CourseTab page and extracts the lists of data. Also, creates an instance of ExpandableListView class and sets OnGroupClickListener to expand/collapse the children.
- *setOnChildClickListener* : Starts an **Intent** to navigate to **Assign_detail page** of the clicked assignment ID.

3.1.5 Course_Threads

It is another fragment on the CourseTab page. It has the following methods:

- *onCreateView* : Receives the **Bundle** sent by the CourseTab activity, and creates an instance of the CustomAdapter_Thread class using the lists in the bundle. Implements the setOnItemClickListener() method of ListView to use an **Intent** to open the activity **thread_detail** for the corresponding thread ID.
- *Post_thread* : Handles the onClick event of the Post thread button. Uses **volley** to send a JsonObjectRequest to post the thread.

3.1.6 Course_Grades

This is another fragment in the CourseTab page. It displays the grades using an Expandable ListView. The adapter for the list view is set to an instance of the class, ExpandableListAdapter.
It stores a list for the headings and a hashmap for the children(Score, Weightage, Total marks, Absolute marks in this case)of each list item. It has the following methods to populate the list view:

- *prepareListData* : Takes as input four arrays, containing data corresponding to four children of each list item. Sets the list and hash map fields of the fragment.
- *onCreateView* : Receives the **Bundle** passed on from the CourseTab page and extracts the lists of data. Also, creates an instance of ExpandableListView class and sets OnGroupClickListener to expand/collapse the children.

3.1.7 Course_Resources

This class also does not have any method/field other than the auto generated ones.

3.1.8 OneFragment

This class extends the Fragment class. It is the main fragment of the MainActivity page that uses ListView to display list of courses. It uses an instance of the MyApp_cookie class to obtain the code of a course corresponding to its id.
The method *onCreateView* extracts the list of courses from the **Bundle** received from the MainActivity, and populates the ListView using an ArrayAdapter.
It also uses *setOnItemClickListener* method to start an **Intent** to open a particular Course Page on clicking any list item, along with a Toast.

3.1.9 TwoFragment

This class also extends the Fragment class. It uses an Expandable ListView to display the list of grades on the MainActivity page.
It stores an array of headers(listDataHeader) of all list elements and a hash map(listDataChild) storing the data corresponding to the children of each item().The following methods are used to populate the list view:

- *prepareListData* : Sends a **JsonObjectRequest** to the server to receive the list of grades. The JsonArray is then parsed so as to set the listDataHeader and listDataChild fields.
- *onCreateView* : Creates an instance of ExpandableListView class and sets OnGroupClickListener to expand/collapse the children to display the details of the grade clicked.

3.1.10 ThreeFragment

This class also extends the Fragment class. It uses a Custom ListView to display the list of notifications on the MainActivity page.
The method *onCreateView* uses the **Bundle** received from MainPage activity to form the arrays containing the text and the time of each notification. An instance of the class **CustomAdapter_Thread** is then created and set as the list view's adapter.

3.2 BackEnd Classes

3.2.1 MoodlePlus Class

This class contains the various auto-generated methods to initialize an activity.
A handler has been added to execute the method Run() of a runnable after the fixed time specified by a private variable, splash_time.
The Run() method creates a new Intent to start the main activity.

3.2.2 MyApp_cookie

This class has been made to handle cookies. It also maintains a global list of course codes and id's.
It also holds the global JsonObjectRequest queue for volley.

3.2.3 Login Class

The Login class handles the dynamics and events of the main login page. The following methods have been included other than the standard methods:

- **login** : Sends a JsonObjectRequest to the server with the user name and password entered by the user. If the login is successful, an **Intent** to navigate to the MainActivity page is started and otherwise, an error message is displayed accordingly.

3.2.4 MainActivity

This class handles requests and data handling for the main page that displays tabs for courses, grades and notifications.

Listing 2: Fields & Methods of MainActivity

```
class MainActivity
{
        private:
                long mRequestStartTime; //request start time
                Toolbar toolbar;                // action bar
                TabLayout tabLayout;    // UI element
                ViewPager viewPager;    // instance of ViewPager class
                String Username;
                String Password;
                MyApp_cookie app_list;  // provides list of course codes &ids
                OneFragment One;                // populated with courses list
                TwoFragment two;                // populated with grades list
                ThreeFragment three;    // populated with notifications list

                void setupViewPager();
                // adds fragments to a ViewPagerAdapter object
        protected:
                void onCreate();
                // calls UpdateCourses & UpdateNotif, sets up ViewPager.
```

```
        public:
                void UpdateCourses();
                // Adds a JsonObjectRequest to fetch list of courses, adds data to a bund
                void UpdateNotif();
                // Adds a JsonObjectRequest to fetch list of notifications, adds data to

                class ViewPagerAdapter
                {
                        private:
                                List<Fragment> mFragmentList;     // list of fragments
                                list<String> mFragmentTitleList; // list of tab titles
                        public:
                                ViewPagerAdapter;
                                Fragment getItem(int);
                                int getCount();
                                void addFragment(Fragment,String);
                                CharSequence getPageTitle(int);
                }
};
```

3.2.5 CourseTab

This class handles requests and data for tabs on the page for a particular course. It stores the course code as a private String. It has new instances of the classes Course_Assignments, Course_Threads and so on, that handle data population for the tabs in this page. The following methods are used as described:

- *onCreate()* : Sets course code according to data from **Bundle** and calls the update functions described below, after which it calls setupViewPager method to populate the UI.
- *UpdateAssgt()* : Adds a JsonObjectRequest for the list of assignments of the course. Creates a **Bundle** and sets it as the argument for the Assignments fragment.
- *UpdateGrades()* : Adds a JsonObjectRequest for the list of grades of the course. Creates a **Bundle** and sets it as the argument for the Grades fragment.
- *UpdateThreads()* : Adds a JsonObjectRequest for the list of threads of the course. Creates a **Bundle** and sets it as the argument for the Threads fragment.

The method *setupViewPager* and the class ViewPagerAdapter are same as described in the class MainActivity.

3.2.6 Assign_details

This class handles the data population on the page that displays the details for a particular assignment. The method *onCreate()* adds a JsonObjectRequest for all the data regarding the assignment. Sets the text fields of the page accordingly. It also creates an instance of the **CustomAdapter_Thread** class with the list of previous submissions to populate the ListView.

3.2.7 thread_Details

This class handles the data population on the page that displays the details for a particular thread.

- *onCreate()* : Adds a JsonObjectRequest for all the data regarding the thread, using the thread ID passed with the **Intent**. Sets the text fields of the page accordingly. It also creates an instance of the **CustomListAdapter** class with the list of comments to populate the ListView.

- *PostComment()* : Handles the onClick event of Post Comment button. If the comment is not empty, then sends a JsonObjectRequest to post the comment on the server. Otherwise, displays a toast stating the same. Also calls the UpdateComments() function on successful posting.
- *UpdateComments()* : Adds a JsonObjectRequest to fetch the list of comments from the server. It also creates a new instance of the **CustomListAdapter** class with the list of comments to refresh the ListView.

3.2.8 ExpandableListAdapter

This class extends the BaseExpandableListAdapter class. It has been made to use a user defined layout for each item of the expandable listView. It's fields and methods are listed as follows:

Listing 3: Fields & Methods of CustomListAdapter

```
class ExpandableListAdapter
{
        private:
                List<String> _listDataHeader;      // header titles
                Context context;                   // main application context
                HashMap<String, List<String>> _listDataChild;

        public:
                ExpandableListAdapter(Context, List<String>,HashMap<String, List<String>>)
                // constructor
                Object getChild(int, int);
                // returns data in the specified child of the specified group
                long getChildId(int, int);
                // returns child position
                View getChildView(int, int, boolean, View, ViewGroup);
                // sets data in layout(list_item.xml) for one child of specified group
                int getChildrenCount(int);
                // returns number of children of specified group
                Object getGroup(int);
                // returns title of the specified group
                int getGroupCount();
                // returns number of groups
                long getGroupId(int);
                // returns group position
                View getGroupView(int, boolean, View, ViewGroup);
                // sets data in layout(list_group.xml) for the group title
                boolean hasStableIds();
                boolean isChildSelectable(int, int);
};
```

3.2.9 OtherWorks

It contains the following methods:

- *time_left(String)* : Takes as input the string representing the deadline, and returns a string containing the time remaining from the system current time.

4 Error scenarios

- **Login** : If the user name and password are not correct, then a suitable error message is displayed.
-

5 Future endeavours

- Keep local cache of changes done, at mobile level and sync them with the global server as soon as internet connectivity is supplied.
- Decrease the time taken to receive data from the server and populate the view.

6 Source Code

The source code of the project is maintained in the following repository:
https://github.com/aditi741997/Moodle-ANA.git

References

[1] Adding animations. http://www.tutorialspoint.com/android/android_animations.htm.

[2] Adding figures in latex. https://www.latex-tutorial.com/tutorials/beginners/latex-figures//.

[3] Supporting tab and phone view. http://developer.android.com/training/multiscreen/screensizes.html.