

Mobile App Development
In-Class Assessment 5 (2 Hours)

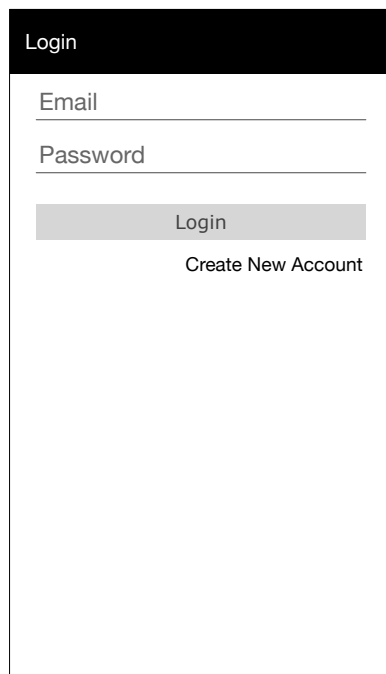
Basic Instructions:

1. This is an In Class Assessment, which counts for 10% of the total course grade.
2. This assessment is an individual effort. Each student is responsible for her/his own assessment and its submission.
3. Once you have picked up the assessment, you may not discuss it in any way with anyone until the assessment period is over.
4. During the assessment, you are allowed to use the course videos, slides, and your code from previous home works and in class assignments. You can use the internet to search for answers. You are NOT allowed to use code provided by other students or solicit help from other online persons.
5. Answer all the assessment parts, all the parts are required.
6. During the assessment the teaching assistants and Instructors will pass by each student and ask them to demonstrate their application. Your interaction with the teaching assistants and instructors will be taken into consideration when grading your assessment submission.
7. Please download the support files provided with the assessment and use them when implementing your project.
8. Your assignment will be graded for functional requirements and efficiency of your submitted solution. You will loose points if your code is not efficient, does unnecessary processing or blocks the UI thread.
9. Create a zip file which includes all the project folder, any required libraries, and your presentation material. Submit the exported file using the provided canvas submission link.
- 10. Do not try to use any Social Messenger apps, Emails, Or Cloud File Storage services in this exam.**
- 11. Failure to follow the above instructions will result in point deductions.**
- 12. Any violation of the rules regarding consultation with others will not be tolerated and will result disciplinary action and failing the course.**

In-Class Assessment 5 (100 Points)

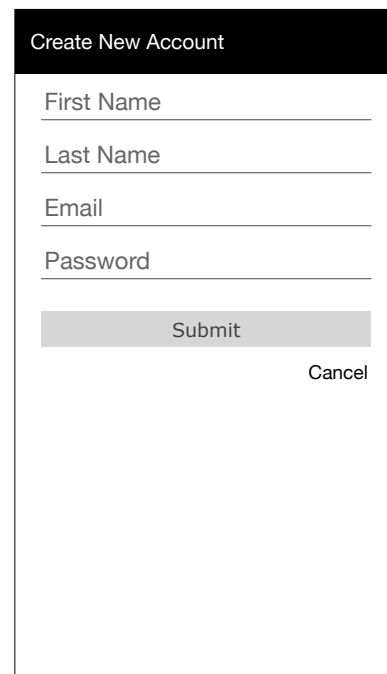
In this assignment you will develop a forums application that allows users to create forums and post messages in these forums. You are provided with a Postman file that contains all the APIs for this app.

1. Use the OkHttp library in this app in order to make all the http connections and API calls. All the data returned by the APIs is in JSON format.
2. All the network calls should be done in a background thread.
3. All UI changes, updates and edits should be performed on the main thread.
4. This app will have one Activity and multiple fragments, all communication between fragments should be managed by the activity.



The Login Fragment wireframe features a black header bar with the text "Login" in white. Below the header, there are two text input fields: "Email" and "Password". A grey rectangular button labeled "Login" is positioned below the password field. At the bottom of the form, the text "Create New Account" is displayed as a link.

(a) Login Fragment



The Register Fragment wireframe features a black header bar with the text "Create New Account" in white. Below the header, there are four text input fields: "First Name", "Last Name", "Email", and "Password". A grey rectangular button labeled "Submit" is positioned below the password field. At the bottom right of the form, the text "Cancel" is displayed as a link.

(b) Register Fragment

Figure 1, App Wireframes

Part 1: Checking User Authentication (10 Points)

You should use the shared preferences to store and retrieve the authentication token and any required user information. For information on shared preferences check the documentation <https://developer.android.com/training/data-storage/shared-preferences>. The requirements are as follows:

1. If a user has successfully logged in or registered, then the shared preferences should be used to store the retrieved information.
2. When the Main Activity starts, you should check the shared preferences for the presence of the token and user information if present the the user is authenticated the app should display the Forums Fragment. If the token and user information are not present in the shared preferences then the Login Fragment should be displayed.
3. Upon user logout the token and user information should be deleted from the shared preferences.

Part 2: Login Fragment (10 Points)

The interface should be created to match Figure 1(a). The requirements are as follows:

1. Upon entering the email and password:
 - a. Clicking “Login” button, if all the inputs are not empty, you should attempt to login the user by using the **/api/login** API.
 - b. If login is successful, then communicate the returned and parsed authentication token and user information (See Figure 2) to the activity, which should store this information in the shared preferences and **replace** the current fragment with the Forums Fragment.
 - c. If login is not successful, show an alert dialog showing the error message returned by the api.
 - d. If there is missing input, show an alert dialog indicating missing input.
2. Clicking the “Create New Account” should **replace** this fragment with the Register Fragment.

```
{
  "status": "ok",
  "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpYXOTI5MDUsImV4cC..",
  "user_id": 1,
  "user_email": "b@b.com",
  "user_fname": "Bob",
  "user_lname": "Smith",
  "user_role": "USER"
}
```

Figure 2, Highlighting the Auth Token returned by login and signup

Part 3: Register Fragment (10 Points)

This fragment allows a user to create a new account. The interface should be created to match Figure 1(b). The requirements are as follows:

1. Upon entering the full name, email and password, clicking the Submit button should:
 - a. If all the inputs are not empty, you should attempt to signup the user by using the **/api/signup** API.
 - b. If the registration is successful, then parse the returned response, and send the authentication token and user information to the activity which should store this information in the shared preferences and **replace** the current fragment with the Forums Fragment.
 - c. If the registration is not successful, show an alert dialog showing the error message returned by the api.
 - d. If there is missing input, show an alert dialog indicating missing input.
2. Clicking “Cancel” should **replace** this fragment with the Login Fragment.

```
//token received /posts/login or /posts/signup
String token = "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpYXOTI5MDUsImV4cC..";
Request request = new Request.Builder()
    .url(postsUrl)
    .addHeader("Authorization", "BEARER " + token)
    .build();
```

Figure 3, Code snippet showing how to add Authorization Header to Request

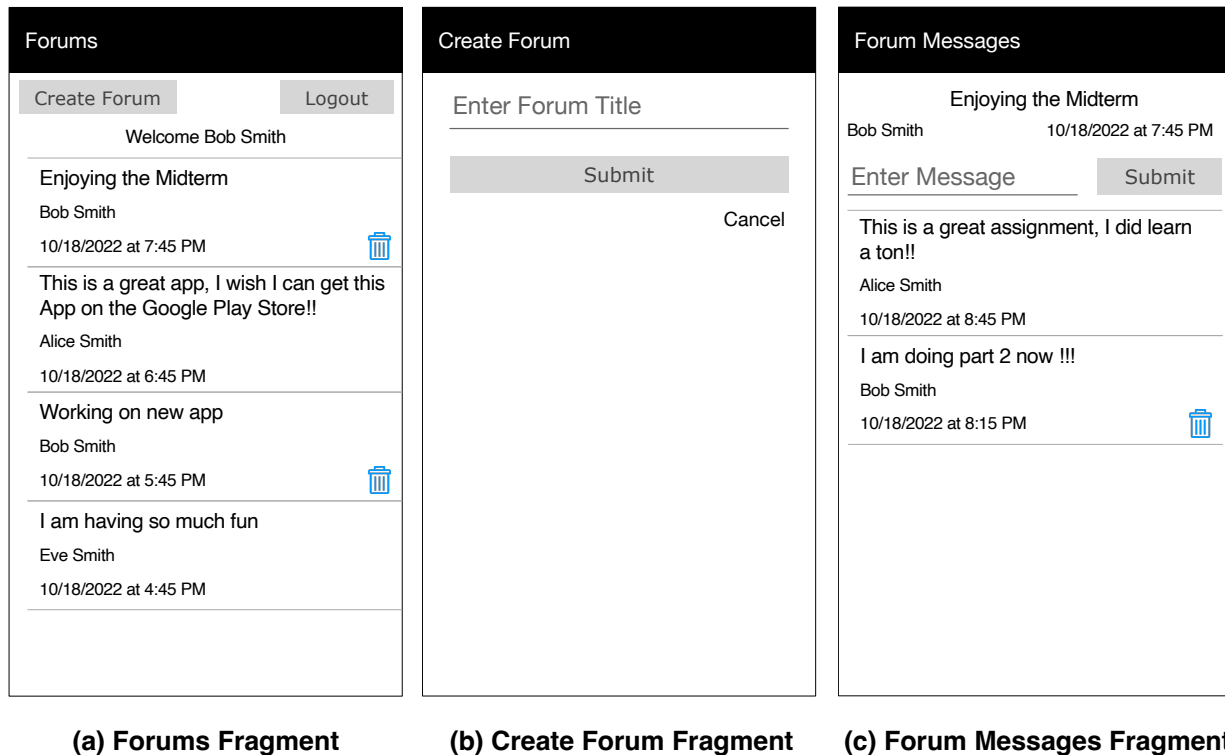


Figure 4, App Wireframes

Part 4 : Forums Fragment (30 Points)

This screen enables the user to view the forums list. As shown in Figure 4(a), The requirements are as follows:

1. Clicking the “Logout” button should communicate with the activity to:
 - a. Delete the authentication token and user information from the activity.
 - b. Delete the authentication token and user information from the shared preferences
 - c. Replace the current fragment with the Login Fragment.
2. Clicking the “Create Forum” button should **replace** the current fragment with the Create Forum Fragment, and put the current fragment **on the back stack**.
3. The greeting TextView should show “Welcome XX” where XX is the name of the logged in user. Note, this information should have been captured from the response of either the **/api/login** or **/api/signup** apis.
4. The list of forums should be retrieved by calling the **/api/threads** API. **Note that this API requires the Authorization header to include the token, please create the OkHttp request and include the header as shown in Figure 3.**
 - a. The **/api/threads** api will return an array of threads, where each thread is forum and will include the thread id, title, created at, information about the user who created the thread.
 - b. Create a Forum class, and parse the returned list of threads into an ArrayList containing the parsed Forum objects. Use the parsed list of Forum objects to display the posts list in RecyclerView as shown in Figure 4(a).
5. For the Forums list, each forum row item should display the thread title, creator’s

name, creation date as shown in Figure 4(a). The trash icon should only be visible for forum items that were created by the currently logged in user, you should use the user id to perform this comparison. For example, in Figure 4(a) the currently logged in user is “Bob Smith” who has created the first and third forums displayed.

- a. Clicking on the trash icon, the app should present an alert dialog asking the user if the selected forum should be deleted. If the user picks “OK” the selected forum should be deleted by using the `/api/thread/delete/{thread_id}` api, note that this api requires the authorization header, see Figure (3) for reference. Upon successfully deleting a forum item, the `/api/threads` API should be called to retrieve the latest list of forums and the forums list should be refreshed with the retrieved forums.
6. Clicking on a forum row item should send the activity the selected forum and perform the following tasks:
 - a. **Replace** the current fragment with the Forum Messages Fragment, and put the current fragment **on the back stack**.
 - b. Send the selected forum object to the Forum Message Fragment.

Part 5 : Create Forum Fragment (10 Points)

This screen enables the user to create a new forum. The requirements are as follows:

1. Clicking the “Cancel” button should **pop the back stack** which should display the Forums Fragment.
2. Upon entering the forum title, clicking the Submit button should:
 - a. If all the inputs are not empty, a new forum should be created by calling the `/api/thread/add` api. Note that this api requires the authorization header, see Figure (3) for reference.
 - b. If the api is successful, then **pop the back stack** which should display the Forums Fragment and should refresh the forums list to show the latest forums retrieved using the `/api/threads` api.

Part 6: Forum Messages Fragment (30 Points)

This screen enables the user to view the messages posted in a given forum. As shown in Figure 4(c), The requirements are as follows:

1. At the top show the Forum title, the name of the forum creator, and the date/time the forum was created.
2. The list of messages should be retrieved by calling the `/api/messages/{thread_id}` api. **Note that this API requires the Authorization header to include the token, please create the OkHttpClient request and include the header as shown in Figure 3.**
 - a. The `/api/messages/{thread_id}` api will return an array of messages, where each message will include the message id, message text, created at, information about the user who created the message.
 - b. Create a Message class, and parse the returned list of messages into an ArrayList containing the parsed Message objects. Use the parsed list of Message objects to display the messages list in RecyclerView as shown in Figure 4(c).
3. For the Messages list, each message row item should display the message text, creators name, creation date as shown in Figure 4(c). The trash icon should only be visible for message items that were created by the currently logged in user, you

should use the user id to perform this comparison. For example, in Figure 4(c) the currently logged in user is “Bob Smith” who has created the last message displayed.

- a. Clicking on the trash icon, the app should present an alert dialog asking the user if the selected post should be deleted. If the user picks “OK” the selected message should be deleted by using the **/api/message/delete/{message_id}** api, note that this api requires the authorization header, see Figure (3) for reference. Upon successfully deleting a message item, the **/api/messages/{thread_id}** API should be called to retrieve the latest list of messages posted under this thread and the messages list should be refreshed with the retrieved messages.
4. Upon entering the message text, clicking the Submit button should:
- a. If all the inputs are not empty, a new message should be created under the current thread by calling the **/api/message/add** api. Note that this api requires the authorization header, see Figure (3) for reference.
 - b. If the api is successful, the **/api/messages/{thread_id}** API should be called to retrieve the latest list of messages posted under this forum and the messages list should be refreshed with the retrieved forum messages.