**50.001 Team 1-4 1D**

UskApp

**Members**

Lim Zhen Dong, Arnold (1004117)

Chen Ken (1004497)

Sun Neng (1004246)

Justin Peng (1004234)

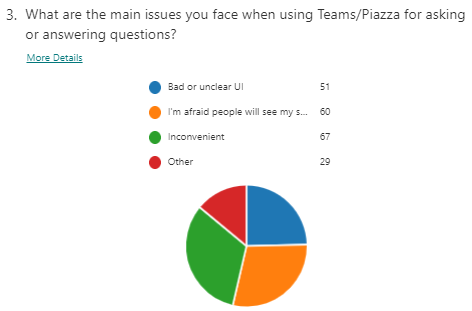
Leon Tjandra (1004353)

**1. Background**

There exists a vast amount of information and educational content on the web, which is convenient to access and refer to. Online learning forums, which foster an open community for people to freely ask any questions related to a topic and others can share their knowledge, are a common source of information. One example is StackOverflow, a question and answer site which features a wide range of topics in computer programming, which has proven successful and a commonly referred-to resource for both new and experienced developers. In our 1D project, we plan to enhance the online learning experience for students by creating a learning forum specifically targeted for school content through UskApp.

**2. Problem Statement and Proposed Solutions**

The current available online learning forum such as Piazza and Microsoft Teams offers a medium for students and instructors to collaborate and share their knowledge and answer questions. However, we found that a majority of students pay minimal attention towards these platforms. We conducted an online survey among the student population to analyse the possible reasons why such platforms were not taken advantage of; the results of the survey can be seen in Figure 1.

**Figure 1**: Results of online survey sent out to the SUTD student population. The full survey results can be found [here](https://forms.office.com/Pages/AnalysisPage.aspx?id=drd2NJDpck-5UGJImDFiPSspaXZ5-mlHtbPALBQvB3JUOEg1TTBFSzVCVDFaQ0RYR0I0Q0E1WFBRMCQlQCN0PWcu&AnalyzerToken=1CEJG3Hs9n0Z2xrhh62Wgf4nqrPYWZqx).

Through the survey, we identified 3 main issues to solve:

1. **Bad or unclear user interface**: We aim for UskApp to provide a simple and user-friendly interface with an intuitive menu and options for ease of use.
2. **Lack of Anonymity**: In Microsoft Teams, the student’s name is revealed when they make a query in the chat. Uskapp provides users with the choice of anonymity to secure their privacy and boost their confidence when posting questions, motivating users to speak up.
3. **Inconvenient**: Searching is simplified not only with an implementation of a search bar but also post tags which foster categorization of questions, and the inclusion of a post following feature allows users to have a point of reference they can refer to and keep track of updates.

Another additional implementation UskApp has is the “Ups” system, which adds a layer of gamification to the forum. Users can accumulate experience points through the amount of upvotes their posts have, and a ranking system provides users with a sense of progress and milestone achievements to complete. This upvoting system has proven successful in many online forums such as StackOverflow and Reddit, and we believe the implementation of this system in UskApp would also incentivise students to be more active in posting queries or giving substantial answers to other questions.

**3. System Design and Implementation**

**Firebase**

Our team decided to use Firebase Realtime Database as well as Firebase Storage for our database stores. We also used Firebase built in authentication methods by email so that we do not need to build authentication and security from scratch.

The data is split into Users, AnswerPost, QuestionPost, Subject and image data is stored in Firebase storage where it is categorized into ProfilePictures, QuestionPictures, AnswerPictures. We had to store the images in the Firebase storage as we found out that image data could not be stored in the realtime database. Therefore we worked around this limitation by storing the image id of a post or profile picture of the user under the realtime database thus when we pull a user’s data or a post’s data we would then get the picture ID which we then use to pull from the respective branch in Firebase storage. Using this same idea, we store the IDs of AnswerPosts and user IDs of users who have upvoted inside each QuestionPost, this allows us to fetch the data of all the replies to a question post which is then later utilised in another activity where the data of each reply is displayed. By storing the user IDs of users who have upvoted a post we can ensure that no user is able to vote more than once on a single post. When a user favourites a post, the ID of that post is stored in an array which contains all the user’s favorited posts. This data is then displayed in a recyclerview in the profile activity which shows all the favourite posts of a user

**Login, Sign up and Forget Password**

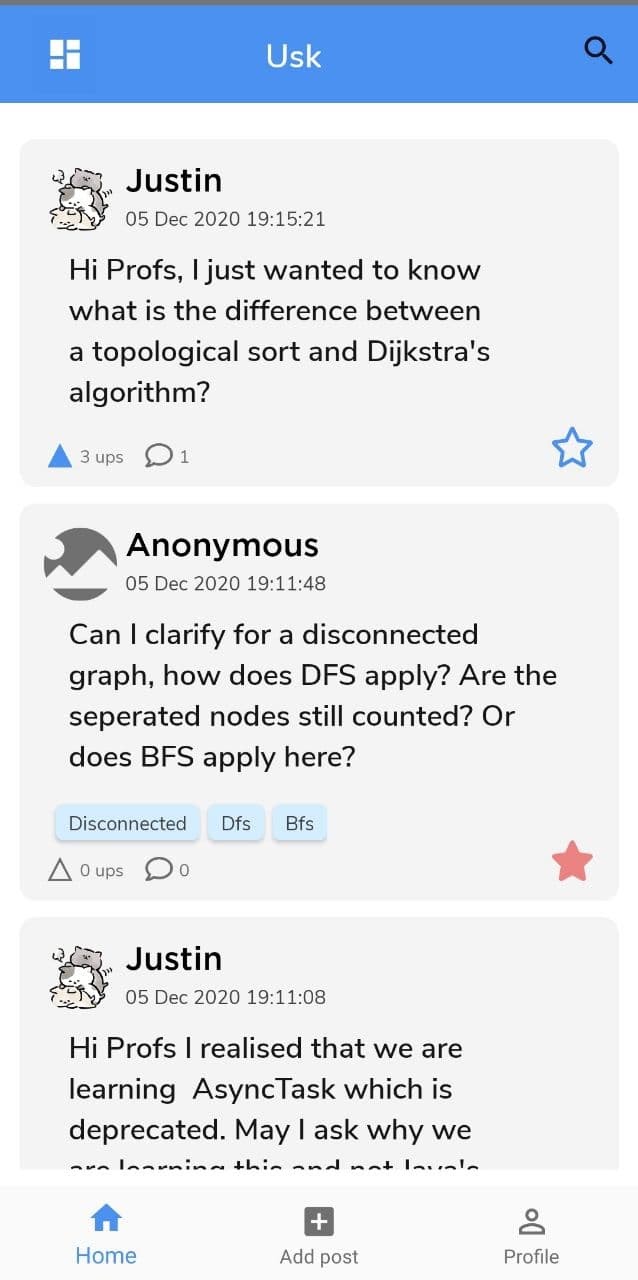
User authentication is usually taken for granted but it is core to a social media app like ours. We need to discern different users so they can post with their own unique account.

Sign up:

* Link UI up with SignupActivity.java
* Check each entry has been filled, else raise an error by using setError on EditText fields.  We also used requestFocus on the EditText so that it will be prominent to the user.
* Check validity of password and email format locally. This is just String checking.
* Push all these data to firebase using mAuth.createUserWithEmailAndPassword function, Firebase will automatically serialize the data and hash the password, thus abstracting these difficult tasks away from us.

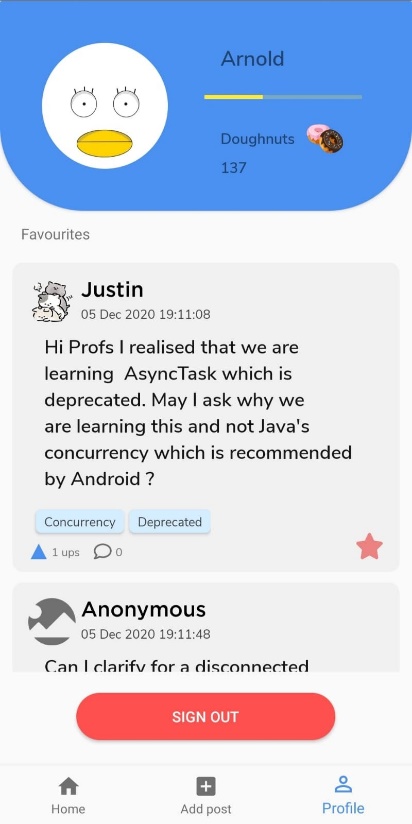
Login: Similar to Sign up, but now calling the mAuth.signInWithEmailAndPassword.

Forget Password: Similar to the other 2 activities, but it uses the mAuth.sendPasswordResetEmail function to send a reset password link to the user’s desired inbox.

**Home Screen**

The home activity screen is the heart of the system. The majority of the screen is occupied by the RecyclerView, which holds all the question posts retrieved from Firebase sorted by timestamp. There are a few options that can be selected in the home page:

* The search button in the top toolbar allows the user to look for posts containing the text the user searches for. This is done through the Filter class which returns a list of posts filtered by the input sequence of characters.
* The subject selection button on the top left of the screen switches to another activity consisting of a RecyclerView containing the subjects the user is enrolled in.
* Clicking on a tag (indicated by the blue rectangle) brings up a list of posts with that particular tag.
* The star button and arrow button indicate whether the user has favorited or upvoted the post respectively. Clicking on any one of the buttons toggles the upvote or favourite.
* Clicking on the background of a post brings up another activity showing the post and its image, while listing the answers below the post. Users can upvote an answer or add their own answer in this activity.

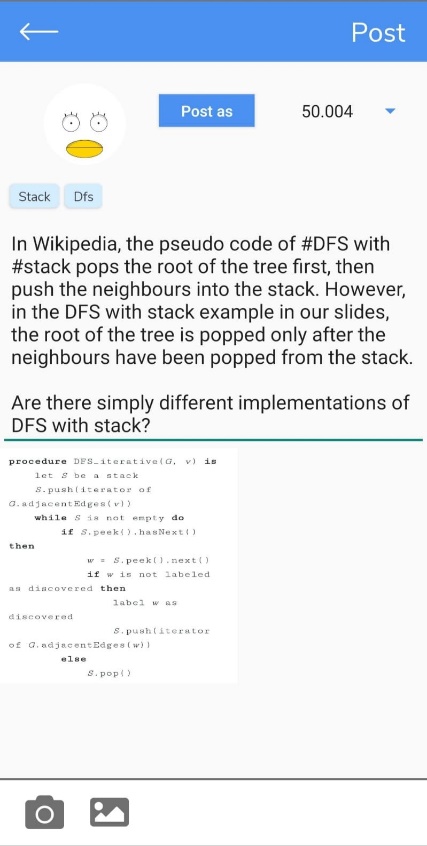
**User Profile**

The user profile details the user information fetched from Firebase.

Users are also able to change their current profile picture by clicking their profile picture circle, which fires an implicit intent that brings up the gallery. The progress bar tells the user how close they are to achieving the next ranking. The list of rankings the user can achieve is listed below:

1. Ups ≤ 10: Pudding
2. Ups ≤ 20: Pancake
3. Ups ≤ 50: Waffle
4. Ups ≤ 100: Crepe
5. Ups ≤ 150: Doughnuts
6. Ups ≤ 200: Macaroon
7. Ups > 200: Sundae

The favourites list is a RecyclerView consisting of the posts the user has favourited from the home page. The adapter class used is the same as that of the home screen, which means the different post functions still apply (upvoting, clicking of tags etc.)

**Posting Questions**

To post questions, the user clicks the middle option in the navigation bar, which brings up the NewPostActivity screen. Clicking the back button would bring the user back to the home screen.

Selecting the ‘Post As’ button next to the user profile picture would bring up a dialog which pops up at the bottom of the screen. This dialog presents options to the user - either the user can post with their profile, or select the anonymous option.

Users can write their question out in the edit text provided. To add a tag to the post, simply add a hashtag (‘#’) in front of the word to be used. Spaces can also be added to a tag using the hyphen (‘-’). For example, writing out ‘#breadth-first-Search’ in the post will create the tag ‘Breadth first search’, which is shown on top of the post.

The user can also attach an image to the post, by selecting either the camera or gallery option at the bottom of the screen. The attached photo is placed below the post.

Lastly, users can indicate which subject to post their question to, by clicking the drop down menu (Spinner widget component)  located next to the ‘Post as’ button.

**4. Design Principles**

The code for UskApp is designed with the single responsibility principle in mind, where each class holds responsibility over a portion of the app’s functionality. The activity classes are each responsible for fetching data from Firebase to fill the UI components. A Utility class was created (Utils.class) which holds some functions that are commonly used throughout the app, such as the creation of a bottom dialog. There exists an abstract class for Posts, which is inherited by QuestionPost and AnswerPost respectively, each detailing the data required for the respective type of posts. We also implement an abstract class BaseNavigationActivity, which handles the switching of activities using the navigation bar. When an item on the navigation bar is switched, the app starts the corresponding activity with zero transitions, to create the appearance of the user being able to switch screens based on selection. The HomeActivity and ProfileActivity inherits from this abstract class.

The adapter design pattern is applied here, as the RecyclerView provides the core functionality for the app, listing out the post data and subjects. For each RecyclerView (the home recycler view showing all the questions, the answer recycler view which shows all the answers to a question in the PostFocusActivity, the subject recycler view showing selections for each enrolled subject, and lastly the recyclerview showing the tags in a post) in the application, there is an adapter class inheriting from RecyclerView.Adapter which translates data to a Viewholder, which holds a layout resource required.

Lastly, fetching from Firebase is a process that occurs asynchronously with the loading of user interface, to ensure that the app runs smoothly the fetching of data is done on the worker thread which then works to fetch the relevant data from Firebase. Once this is complete, we call the adapter of the recyclerview to refresh itself and display the newly fetched data from Firebase onto the recyclerview. As we utilise the a ValueEventListener which uses the event call back onDataChange, everytime the data that the ValueEventListener is looking at changes, the onDataChange method is called which then causes the data to be fetched again and thus our recyclerview adapter is then notified of these changes and refreshes itself again. This allows us to instantly reflect changes in our application as users will get updated whenever the data they are looking at is modified and thus the users get a real time update when using the app. <more stuff about how Firebase works separately>

There exist many other different design principles in this UskApp, as the app consists of many different components, some for quality of life improvements such as the creation of tags or the viewing of pictures full screen. However, due to space constraints, we listed down the most defining design patterns in our application.

**5. Future Work**

Improving on our karma and experience features, we are planning to include special badges/awards to add gamification elements to our app that is proven to be a stimulus towards users for using the app more frequently. The badges will be displayed on each user’s profile as his or her trophy of achievement in a recycler view layout, and each one requiring different conditions to obtain, for example, helping to answer three questions from one particular category or answering three questions from three different categories, prompting users to engage in “missions” to help enrich knowledge of one another.

We also think of expanding the current functionalities of our search bar system. Currently we are only able to search based on the text content of a post that checks whether the string in the search view bar matches any sequence of text within a post, however in the future we are planning to allow searching by post author, date and time. This feature can be carried out using a menu which allows users to specify their search by query type, focusing on different aspects of a post. Our current mode of sorting is based on the date or time the post is created, future improvement will include a menu that allows sorting based on different parameters such as alphabetical order, number of upvotes, number of replies, which fits different user needs.

Currently, our Uskapp system allows users to attach images towards questions and posts to help visualize the problem or provide a more intuitive explanation, future possibilities may include attachment of video and sound based material in their respective formats to accommodate a wider and broader realm of questions, ranging from critical movie clip interpretation to musical learning and education. The concept used here will be related to implicit intents which access the gallery to search for items of the corresponding type, furthermore we can replace the picture/photos button to a menu which allows users to choose their preferred type of attachment (image, video, audio).

**6. Summary**

In conclusion our team managed to complete a working application that achieves the goals we had when planning for this application and solves the problems that we have identified in our problem statement. Throughout this project we learnt that writing code for an online learning forum was not as easy as expected, since there are lots of unexpected behaviors due to interactions with the real time database. Furthermore, we realized that writing clean and quality code while collaborating is very hard. There are tendencies to leave the code at “good enough that it works” level when we are rushing features to push to github so our team mates can work on something else that’s intertwined to our work. All in all, this was a tremendous learning experience in Java, the Android framework and Github collaboration.

**7. Appendix**

**Task Division**

|  |  |
| --- | --- |
| Arnold | * Tag System * Navigation bar switching * Question and answer card layouts * Question and answer recyclerview adapters * Favourite posts system * Exhibition Poster |
| Sun Neng | * UI design with Adobe XD * Exhibition Video * Login, sign up and forget password activities |
| Justin | * Firebase data implementation * Profile activity |
| Leon Tjandra | * Search function * New post activity |
| Chen Ken | * Exhibition Poster * Choosing subject feed * Ranking system |

**External Libraries Used**

* Google Flexbox Layout: <https://github.com/google/flexbox-layout>
  + Lays out views similar to CSS Flexible Layout module, where views are stacked next to each other irrespective of view length
  + Used for layout manager for the tag recycler view in the question post
* CircleImageView by Henning Dodenhof: <https://github.com/hdodenhof/CircleImageView>
  + Used for profile pictures in ProfileActivity and MainRecyclerViewAdapter
* TouchImageView: <https://github.com/MikeOrtiz/TouchImageView>
  + Used for ViewImageActivity, where user can pinch or double tap to zoom in

**Github Repository**

UskApp for 50.001 1D project: <https://github.com/arnoldd98/uskapp>