CS4048 Project Proposal

HotWheels

Arnas Juravicius - 18257305

Group Members

Arnas Juravicius – 18257305
Cyiaph McCann - 17233453
Dylan Kearney - 18227023
Oisin McNamara - 18237398

Introduction

Main Idea

The main idea of our app is a car enthusiast social media platform where you can share pictures and videos of your car. We will allow users to take a picture using the in-app camera function or to use pictures from their personal gallery which will then be posted to their account for their followers to see. The user can add a description to the post they create, they can optionally input the specs of the car such as the make and model. We will also add the option for users to add a location to their post so whoever sees it can see the location it was taken. A trending page will be visible to anyone who opens the app and will show most popular posts by likes, however the user must register and log in to use the apps main functionality such as posting, commenting, reposting, and liking. Logged in users can also like, comment, and repost other user's posts. The user's profile page shows all the posts that they have uploaded and who they follow and their followers. When a user views another person's profile it will only display their posts. There will also be a search functionality where users can search for posts using a car brand or search for people to follow using their username.

Technology Used

API/framework	**Purpose**
Firebase Auth UI	Allow users to register accounts and log in.
Firebase FireStore	Store user profiles.
Firebase Storage	Store pictures and videos.
Google Maps Platform	Share location of posts.
Google Play Services	Push notifications to user.
Camera API	Used to take or import pictures from gallery.

Usability and Background Research

Background Research

For this app we used Instagram and VSCO as major influencers in our design and implementation. We implemented numerous items from these apps as well as ideas from other apps. Some of these were Home Feed, Trending Page, Post Likes, Followers, etc. We used these features with respect to our main idea, this is an app for car enthusiasts. Furthermore, we used the research to come up with user stories that may be useful for our app. Some of these were, what happens to popular posts? How will the home feed be ordered?

Usability

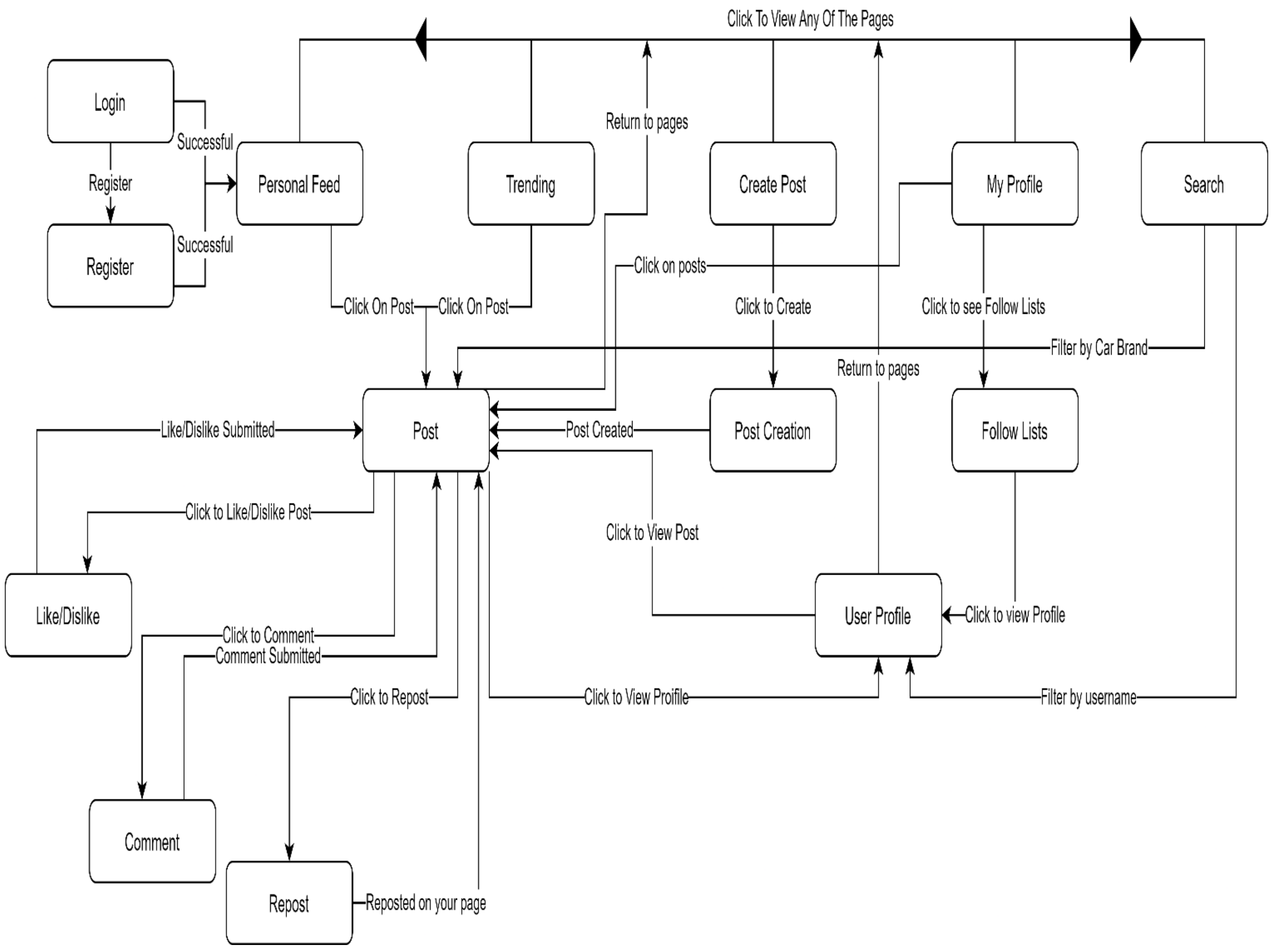
We used our background research to create several user stories. These user stories are real life examples that you would find in other apps as well as user stories tailored to our app. Many of these are user stories that you would find across most other apps and are realistic to what you would want to do on our app such as be able to like another person's post or add them to your followers.

1. As a user, I want to create a new account and login so that I can share posts of my car for my friends to see.
2. As a user, I want to post content such as pictures so that I can share my car with followers.
3. As a user, I want to be able to like and share other posts so that I can show appreciation to other people's cars.
4. As a user, I want to be able to add the location to my posts so that I can show where the picture was taken.
5. As a user, I want to be able to follow other accounts so that I can see what they post where and when.
6. As a user, I want to be able to view viral content that is trending so that I can see content posted by people that I may or may not follow.

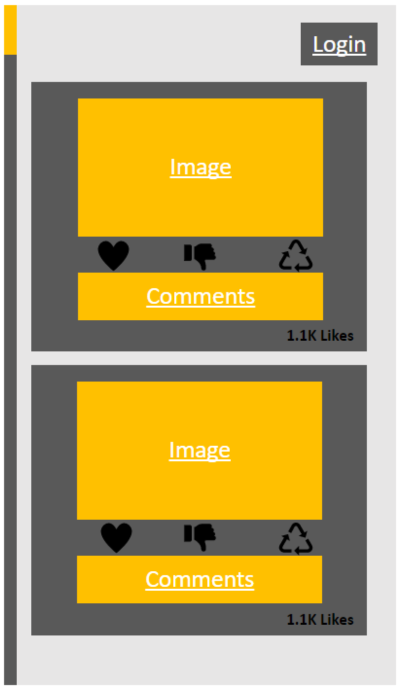
Design and Development Process

Design Process

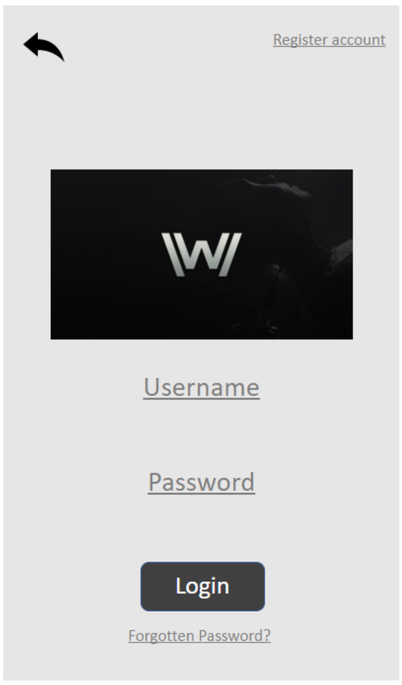
Firstly, we created a navigation structure using the user stories we came up with. This allowed us to find out how many and what pages exactly we need for the app. This gave us a form of basis to work from during the development process. We firstly created a UI mockup which we closely followed during the development stage. This made it much easier to structure our app and gave us something to follow such as a login template for the login screen. We also saved time here in designing the actual app and we could focus on spending time on the implementation of the functionality and user stories. Having a UI template made it very quick and easy to design the app itself.



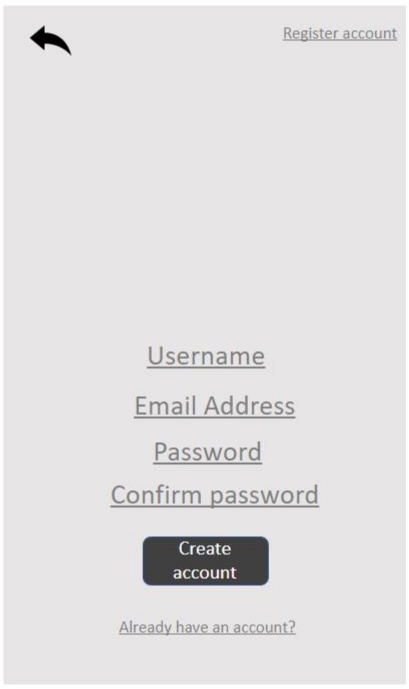
UI Mockup



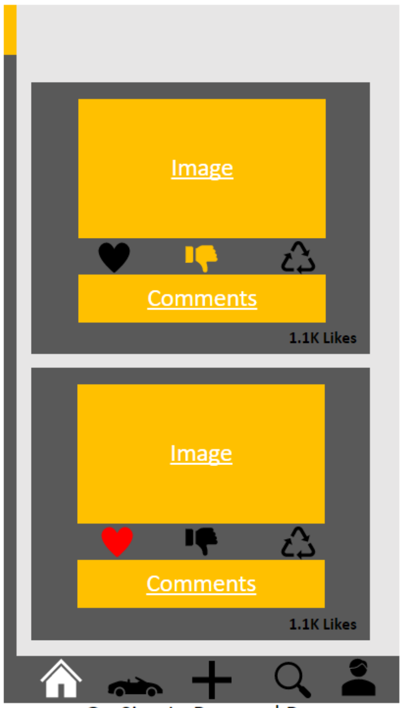
On-App Open Trending Page



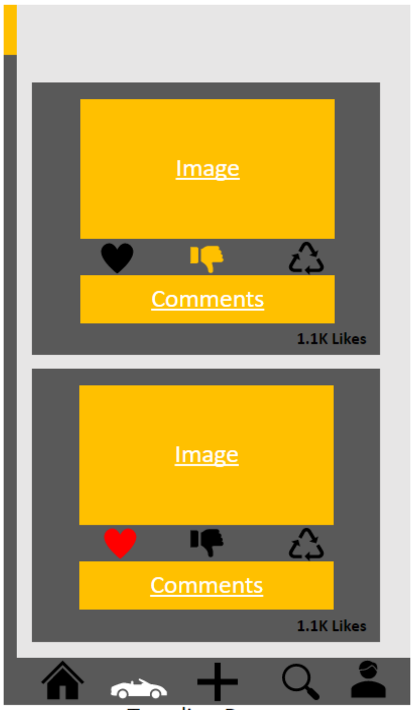
Login Screen



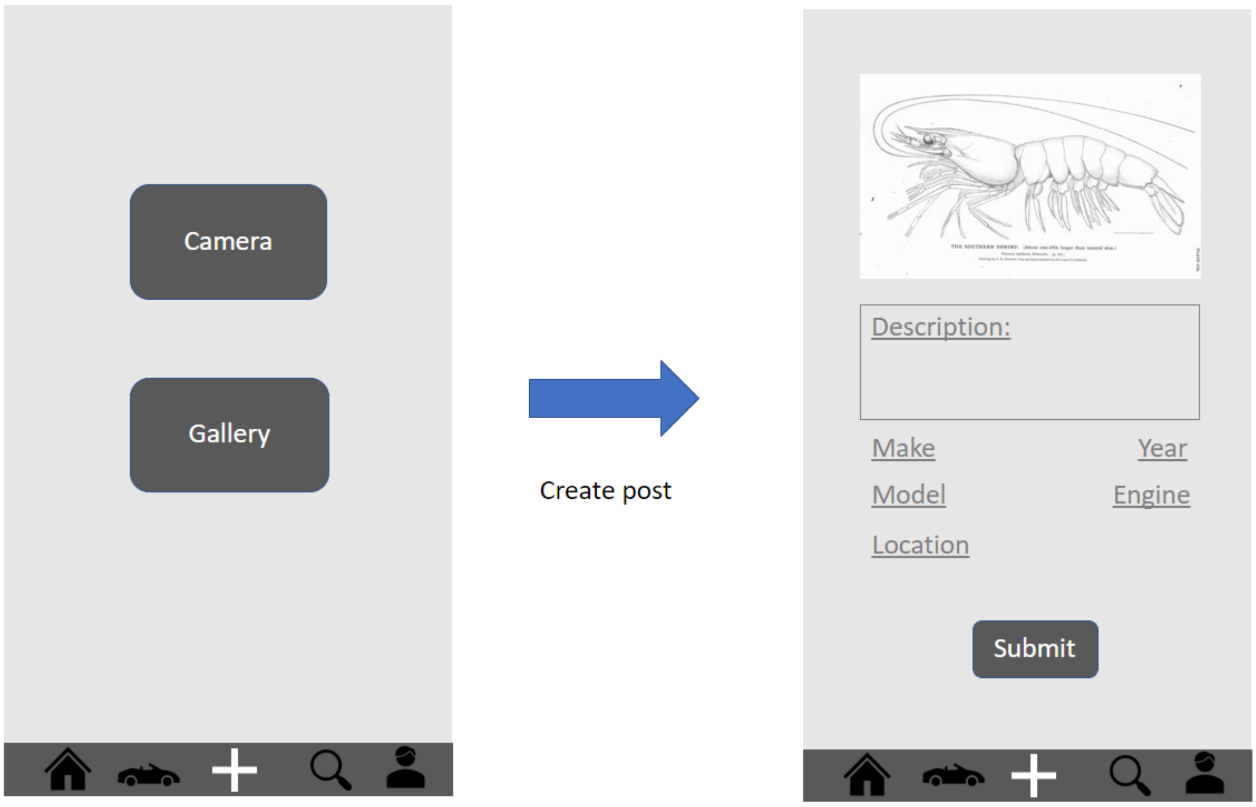
Register Screen

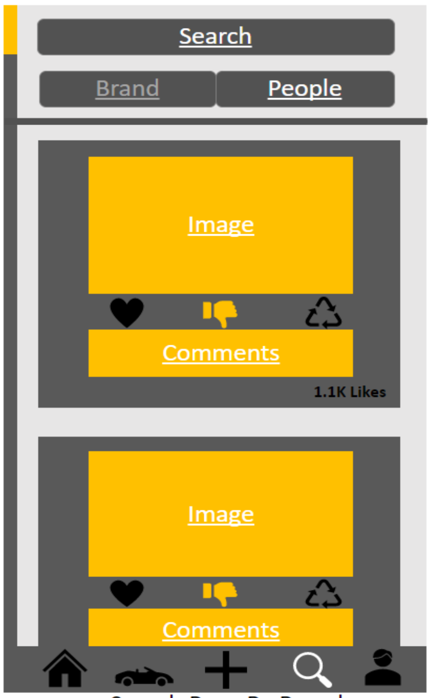


On Sign In Personal Page

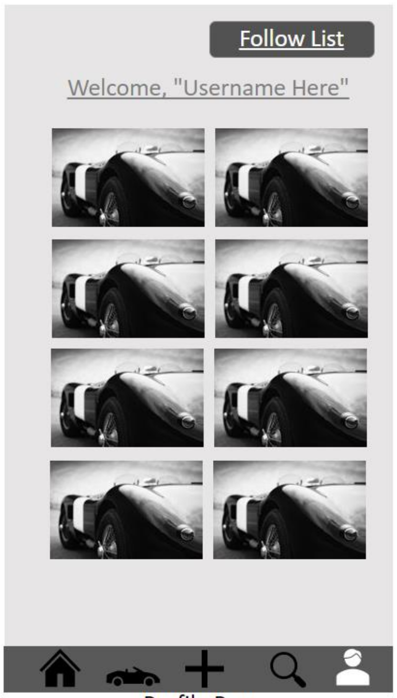


Trending Page





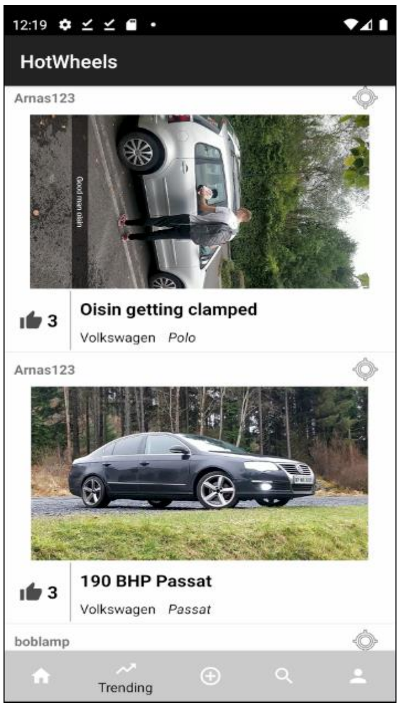
Search Page By Brand



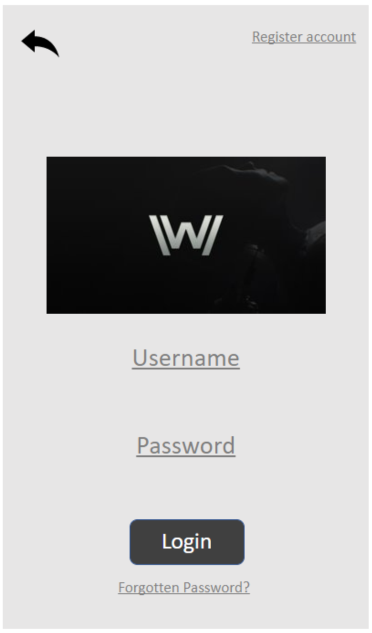
Profile Page

Development Process

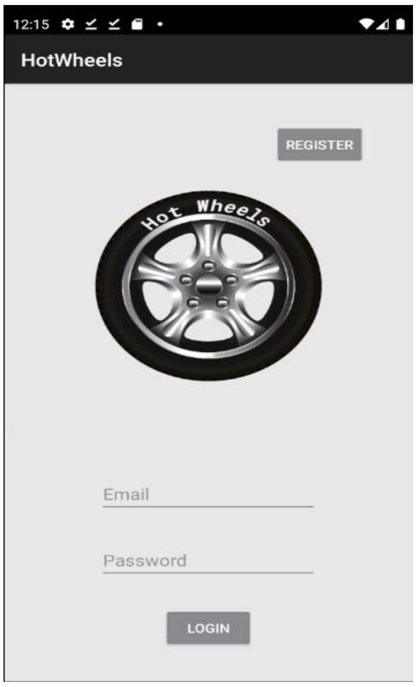
We took the app one page at a time. Starting with the register screen so we could create an actual user using their information that was entered on the register screen in the Firebase Authentication. This allowed to us to then create a login page where we could login as a user and a main home page where we would be brought to on successful login. We used the fragment manager as the basis for the app, having most of our pages be fragments. For this we used a navigation bar to cycle through all the fragments. Here the navigation bar is shown at the bottom of the app. It cycles through home page, trending page, add a post, search, and my profile. From there we all took individual tasks to do with the app and the app seemed to fall together slowly.



During the development process, we tried to follow our UI mockup as close as we could. Doing this we still improved the UI where we seemed fit to in the app itself. Below is an example of the left being the UI mockup and the right, the actual implementation in the app.

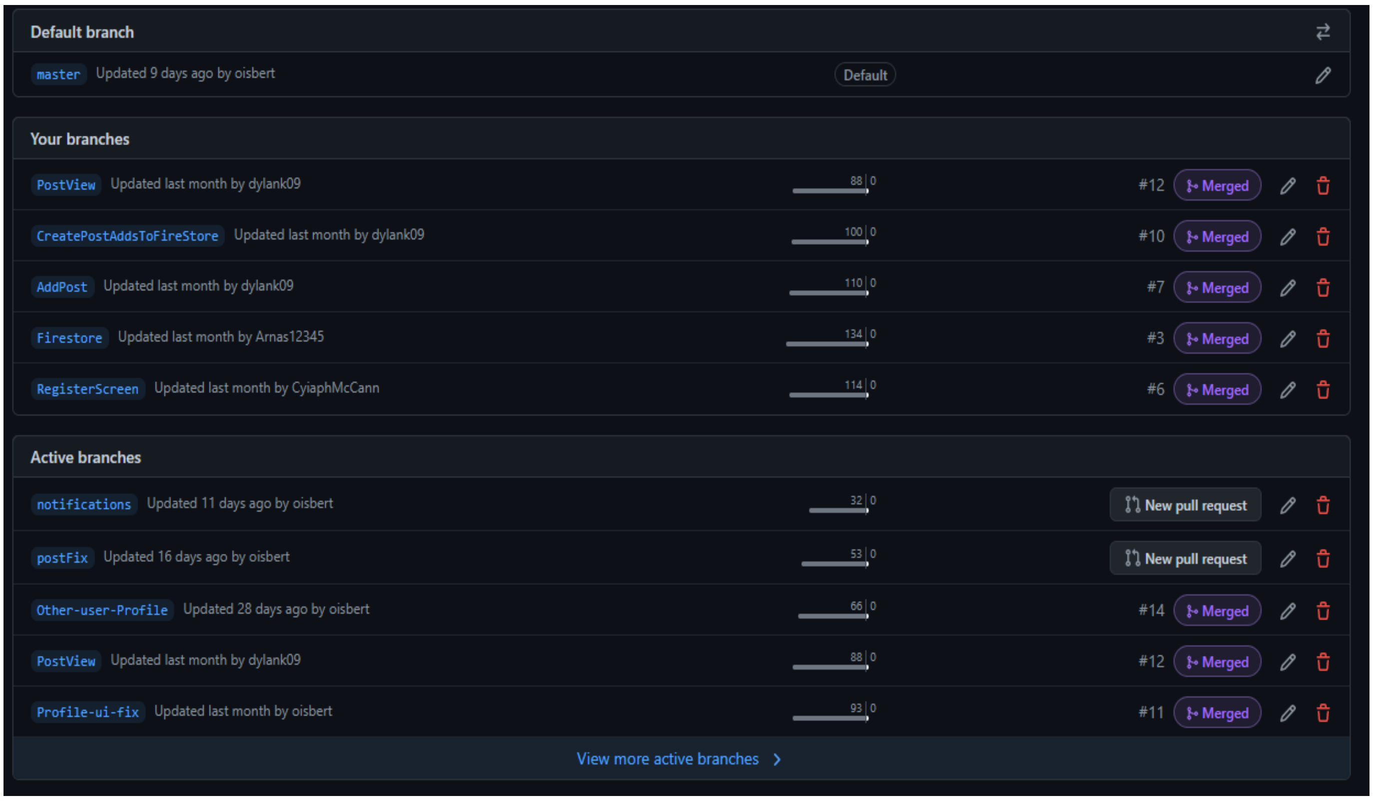


Login Screen



GitHub Version Control

During the development stage, we frequently used GitHub for Version Control. We done this creating numerous branches and using the master branch. Below are some of the many branches we done. Version control made it much easier to go back if something went wrong or did not work after implementation. Many times, when doing something and I or a member of the team broke the app, version control allowed us to go back to the previous patch where the app still worked. Although version control was great, we did come into trouble with it. Many times, when two or more people worked on the same file and we got a merge conflict, sometimes we did not merge correctly, and it would delete a user's work that he had done. For the app we had 16 different branches and over 170 commits



Default branch

master	Updated 9 days ago by oisbert			Default

Your branches

PostView	Updated last month by dylank09	88 | 0	#12	Merged
CreatePostAddsToFireStore	Updated last month by dylank09	100 | 0	#10	Merged
AddPost	Updated last month by dylank09	110 | 0	#7	Merged
Firestore	Updated last month by Arnas12345	134 | 0	#3	Merged
RegisterScreen	Updated last month by CyiaphMcCann	114 | 0	#6	Merged

Active branches

notifications	Updated 11 days ago by oisbert	32 | 0		New pull request
postFix	Updated 16 days ago by oisbert	53 | 0		New pull request
Other-user-Profile	Updated 28 days ago by oisbert	66 | 0	#14	Merged
PostView	Updated last month by dylank09	88 | 0	#12	Merged
Profile-ui-fix	Updated last month by oisbert	93 | 0	#11	Merged

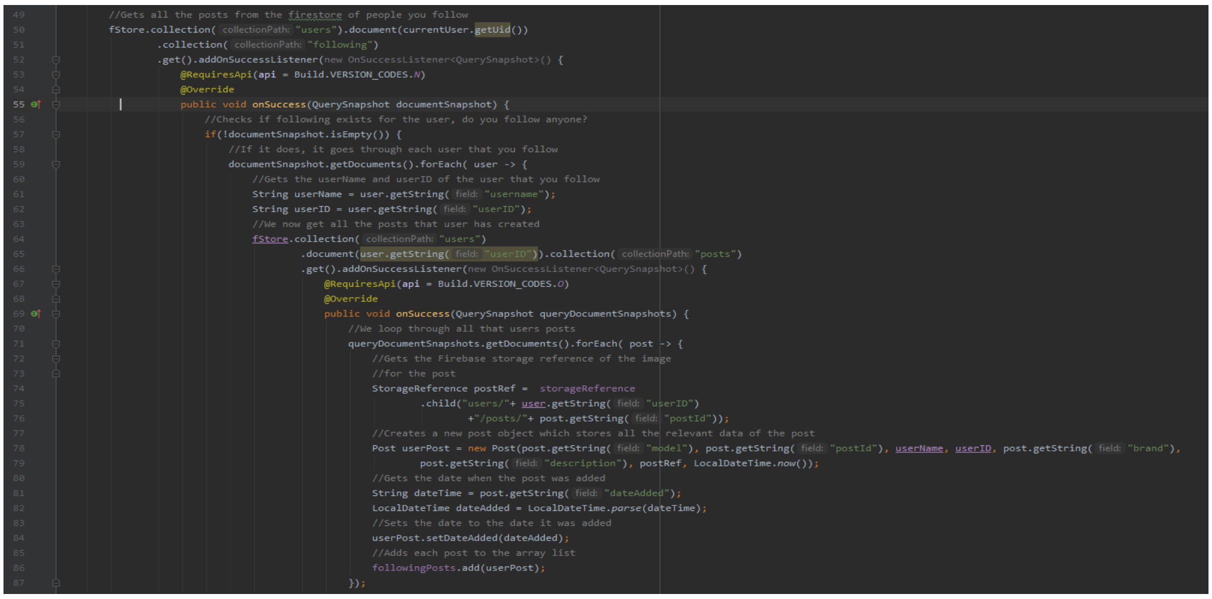
View more active branches

Conclusion

Obstacles

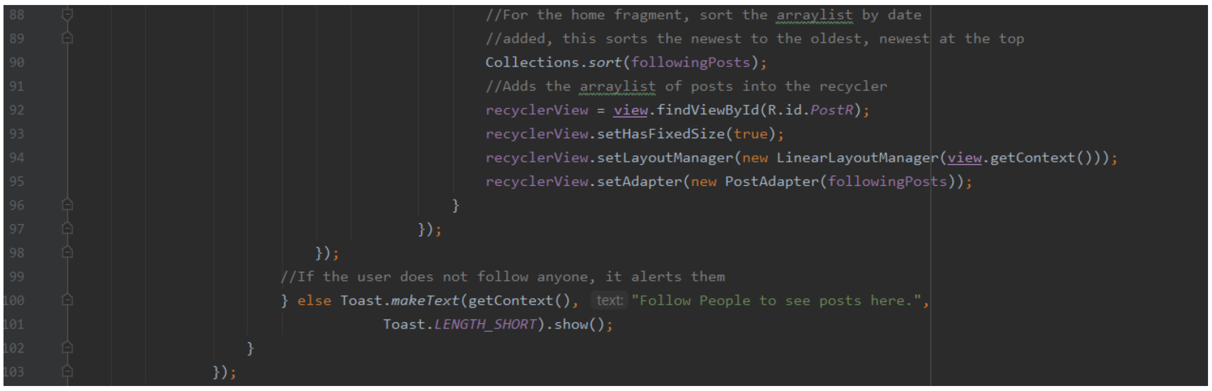
During the development stage we faced many obstacles within the app, with it breaking many times. One obstacle I will discuss in detail was getting posts data from the Firebase Store and Firebase Storage. This obstacle hindered our work for around a week in total and it was a team effort to get working.

For the home page, we wanted to get all the posts of the people the user follows. We done this by going through all the users in the users following on the firestore, these are the people the user follows. It then would go through all that users post and get all the details of the posts including the picture and add it to an arraylist. This arraylist would then be used to create a recycler view where each post would have the same format and be displayed on the home page. The posts were ordered in the arraylist by the date and time they were added showing newest posts at the top and older ones at the bottom.



```
49            //Gets all the posts from the firestore of people you follow
50            fStore.collection( collectionPath: "users").document(currentUser.getUid())
51                    .collection( collectionPath: "following")
52                    .get().addOnSuccessListener(new OnSuccessListener<QuerySnapshot>() {
53                @RequiresApi(api = Build.VERSION_CODES.N)
54                @Override
55                public void onSuccess(QuerySnapshot documentSnapshot) {
56                    //Checks if following exists for the user, do you follow anyone?
57                    if(!documentSnapshot.isEmpty()) {
58                        //If it does, it goes through each user that you follow
59                        documentSnapshot.getDocuments().forEach( user -> {
60                            //Gets the userName and userID of the user that you follow
61                            String userName = user.getString( field: "username");
62                            String userID = user.getString( field: "userID");
63                            //We now get all the posts that user has created
64                            fStore.collection( collectionPath: "users")
65                                    .document(user.getString( field: "userID")).collection( collectionPath: "posts")
66                                    .get().addOnSuccessListener(new OnSuccessListener<QuerySnapshot>() {
67                                @RequiresApi(api = Build.VERSION_CODES.O)
68                                @Override
69                                public void onSuccess(QuerySnapshot queryDocumentSnapshots) {
70                                    //We loop through all that users posts
71                                    queryDocumentSnapshots.getDocuments().forEach( post -> {
72                                        //Gets the Firebase storage reference of the image
73                                        //for the post
74                                        StorageReference postRef =  storageReference
75                                                .child("users/"+ user.getString( field: "userID")
76                                                        +"/posts/"+ post.getString( field: "postId"));
77                                        //Creates a new post object which stores all the relevant data of the post
78                                        Post userPost = new Post(post.getString( field: "model"), post.getString( field: "postId"), userName, userID, post.getString( field: "brand"),
79                                                post.getString( field: "description"), postRef, LocalDateTime.now());
80                                        //Gets the date when the post was added
81                                        String dateTime = post.getString( field: "dateAdded");
82                                        LocalDateTime dateAdded = LocalDateTime.parse(dateTime);
83                                        //Sets the date to the date it was added
84                                        userPost.setDateAdded(dateAdded);
85                                        //Adds each post to the array list
86                                        followingPosts.add(userPost);
87                                    });
```

We ran into a problem when pushing the arraylist into the recycler view. The method would get all the relevant data from the firestore, create a post, and add it to the arraylist. Yet, when we called on the arraylist in the recycler view, it was empty. We had the lines 90 – 95 where we add the arraylist to the recycler view outside of the on-Success listener for getting the users. Here, the arraylist would be empty. We found out that the problem was that the on-Success listener is actually an anonymous inner class. We were not able to call on the arraylist and add a post to it inside an anonymous inner class. We found this out using Stack Overflow on google.



```
88                                    //For the home fragment, sort the arraylist by date
89                                    //added, this sorts the newest to the oldest, newest at the top
90                                    Collections.sort(followingPosts);
91                                    //Adds the arraylist of posts into the recycler
92                                    recyclerView = view.findViewById(R.id.PostR);
93                                    recyclerView.setHasFixedSize(true);
94                                    recyclerView.setLayoutManager(new LinearLayoutManager(view.getContext()));
95                                    recyclerView.setAdapter(new PostAdapter(followingPosts));
96                                }
97                            });
98                        });
99                        //If the user does not follow anyone, it alerts them
100                   } else Toast.makeText(getContext(),  text: "Follow People to see posts here.",
101                           Toast.LENGTH_SHORT).show();
102               }
103           });
```

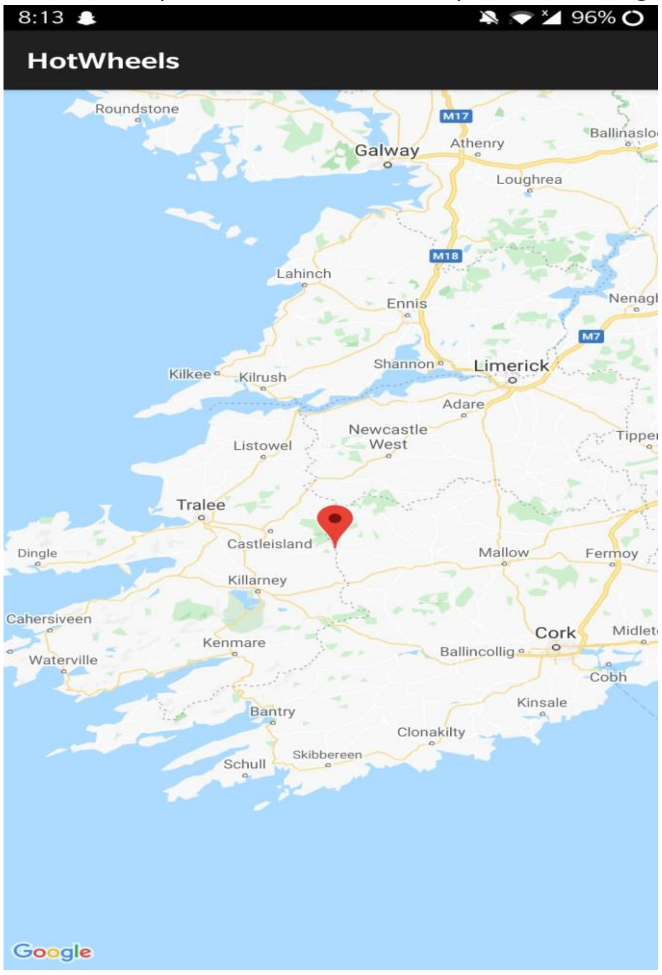
To fix this, we moved lines 90-95 inside of the on-Success listener. This then functioned as expected adding posts to an arraylist and then displaying them on the home page. Such a small error and such an easy fix, yet we spent around a week stuck on this problem.

Expanding the project

A major thought idea when designing the app was that we could create a car parts store where user could advertise and sell car parts. Building an e-commerce platform along in with our social media platform is something we would love to implement if we were given the opportunity. Furthermore, we would have loved to have implemented messaging and commenting on posts. This would add to the socialism of the social media platform. Lastly, we would have liked to include adding videos to the app as we currently can only add pictures. This was a result of the time constraint of the project and how long it took us to get the photos high resolution, and all formatted the same size.

GPS Example

In the app we also included the GPS location of where the photo was posted. To access it you would have to click on the location icon on a post. This functionality was broken in the emulator and only worked on a phone. Below is an example of it working on my current phone.



GitHub Access

https://github.com/Simon-Colreavy-CS4084/mobile-app-development-bois