ENSE 477
Capstone Final Project
Experience Report
April 10th, 2021
Joseph Bello, Samuel Iregbu, Onisokien Ayonoadu, Jaskirat Josan
Team BEEJ

Table of Contents

Table of Figures

Introduction

For the ENSE 400/477 Capstone project, Team BEEJ which consists of Joseph Bello, Jaskirat Josan, Onisokien Ayonoadu, and Samuel Iregbu, chose to build a Progressive Web App in an effort to bridge the gap of communication between the citizens and the City of Regina. We decided to call our project *The Link* to symbolize the link of communication we hope to create between the citizens and the City of Regina. This project was designed for ENSE 400/477 Capstone classes with Team BEEJ planning, designing, and implementing this application for a due date of April 10th, 2021. With the help of Timothy Maciag, the professor of ENSE 400/477, and our project mentor, Craig Gelowitz, this project was conceptualized, and a minimum viable product (MVP) was completed for April 10th, 2021.

Background Information

While brainstorming ideas for our project we realized that there were currently three known methods to report a problem in your city. The first is to simply call into the City of Regina contact center, the second is by filling out a form on the City of Regina website, and the third method is to contact the City of Regina through their social media accounts. Here are the problems that we identified with the current methods; the call center can get very busy at times and lead to long wait times. The form on the City of Regina requires some searching to be found and can be tedious to fill out. Not everyone has social media or would think of it as an appropriate way to report a problem. Also, the city's activity on their social media accounts can vary significantly and can lead to

reports being unread and unrecorded. Furthermore, there is no way for other citizens to keep track of the problems that are reported and receive updates. Based on this information, we identified the following problems. Currently, there is not one streamlined way for the civilians to get hold of the ministries to report a problem. So, we want to design a progressive web app that will be the one-stop-shop for civilians to report problems and for the city to view and act upon them.

Technologies

Github was utilized immensely throughout the year as a way to accommodate all of our project requirements such as documents, diagrams, vlogs, and code. Github allowed members of the team to make changes on their respective local devices before pushing changes to the project. This allowed for a seamless collaborative experience in all aspects of the project.

The project was initially meant to be an app and we initially designed our lofi and hifi diagrams for a mobile application. Initially, we were looking into Ionic to create a cross-platform mobile app. However, as the project progressed and constructive input from our project supervisor, Timothy Maciag, and our project mentor, Craig Gelowtiz, we realized that our project would best be presented as a progressive web application. This was because our two main user groups on our app, the citizens and the City of Regina employees, would most likely use the app on two different devices. The City of Regina employees would most likely be using the app on their desktops and the citizens would prefer to report problems on their phone. Thus, it would only make sense to create an

experience that would work seamlessly on all types of devices. That is why a Progressive Web Application (PWA) was chosen.

We used Adobe XD to design our wireframes. Adobe XD allowed us all to work together on the wireframe in real-time and this proved to be a beneficial collaborative experience. The learning curve on Adobe XD was also relatively small and we were all able to quickly pick up how to create, edit and link pages. We were able to split the work and also see what everyone else was designing. This was beneficial when we linked our pages and buttons together on Adobe XD.

Our PWA was initially hosted on Amazon Amplify, but we realized that its features were too broad for our app, and the learning curve was extremely steep. So based on further research, we eventually decided to host on Google Firebase as it proved to be better for the scale of our app. Most of our backend work is handled by firebase. It provides us with pre-built user authentication, Firestore real-time database, and Storage.

The front-end framework we used for our PWA is React JS - a JavaScript library for building user interfaces or UI components. This framework allows us to create web apps for both mobile and web platforms seamlessly. The react framework does not handle backend logic or databases; it simply creates a frontend build pipeline, which means we could use it with any backend we wanted. React also allows us to effortly use third party resources like icons from Material ui or Geocode from Google.

Design

Since our project is meant to be used by a wide demographic which includes individuals of all ages and technological capabilities, we had to make our design as user-friendly and easy to follow as possible. We chose a fairly minimalistic design and chose to loosely base our design off of the popular social media site, Instagram.

Initially, we all decided to brainstorm on individual low fidelity prototypes, where each member of the group designed and presented for a final decision. We analyzed each member's design and we were able to pick up unique features from each design, resulting in our final low-fidelity prototype.

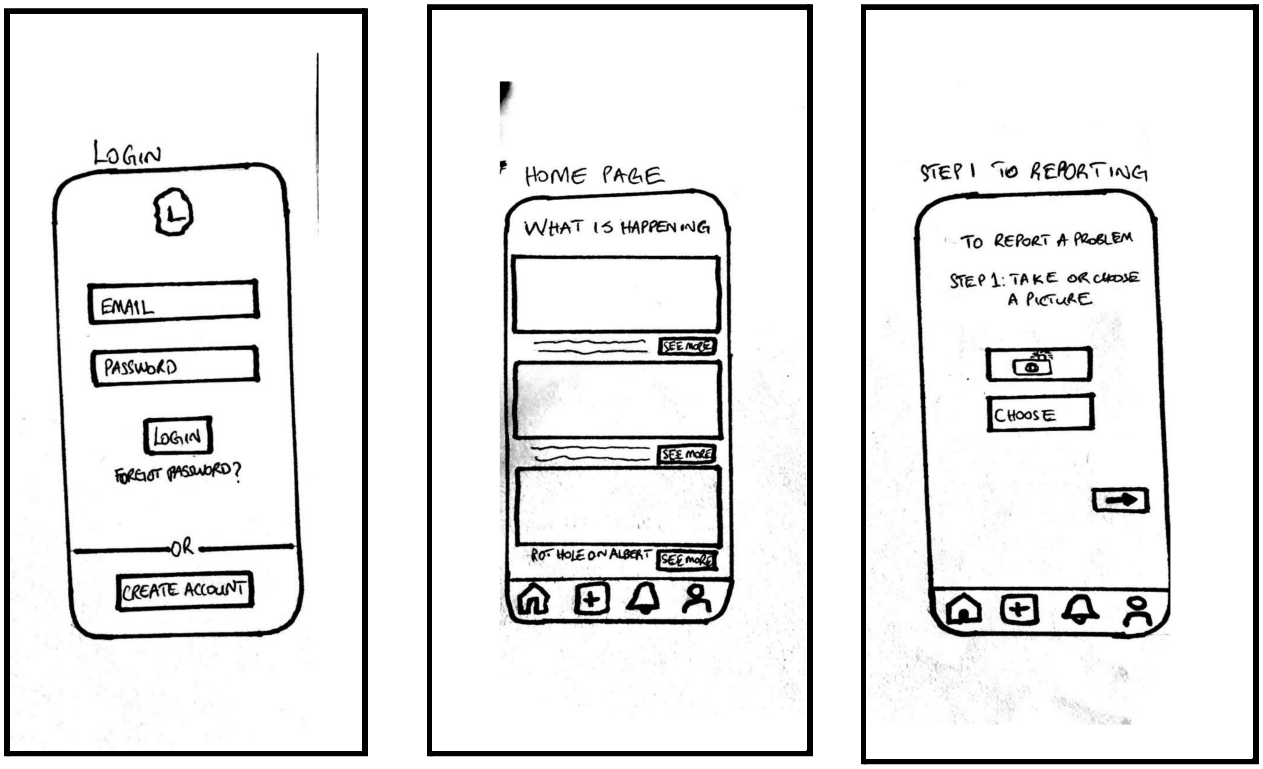

Figure 1, 2, 3 Low Fidelity Prototypes (October 2020)

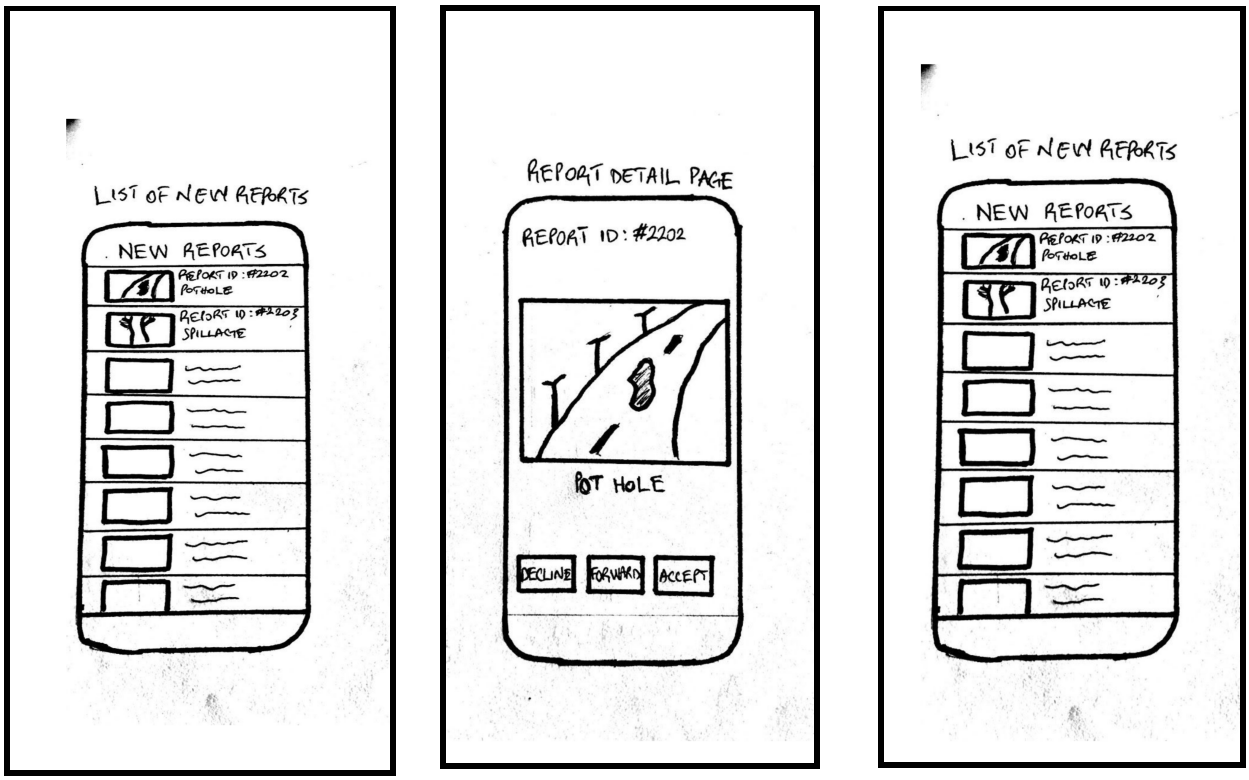

Figure 4, 5, 6: Low Fidelity Prototypes (October 2020)

We progressed to designing a high-fidelity prototype in October, using Adobe XD, which was recommended by our project supervisor, Timothy Maciag. Please note that these prototypes were constructed when the team was still planning to build a mobile application instead of a PWA.

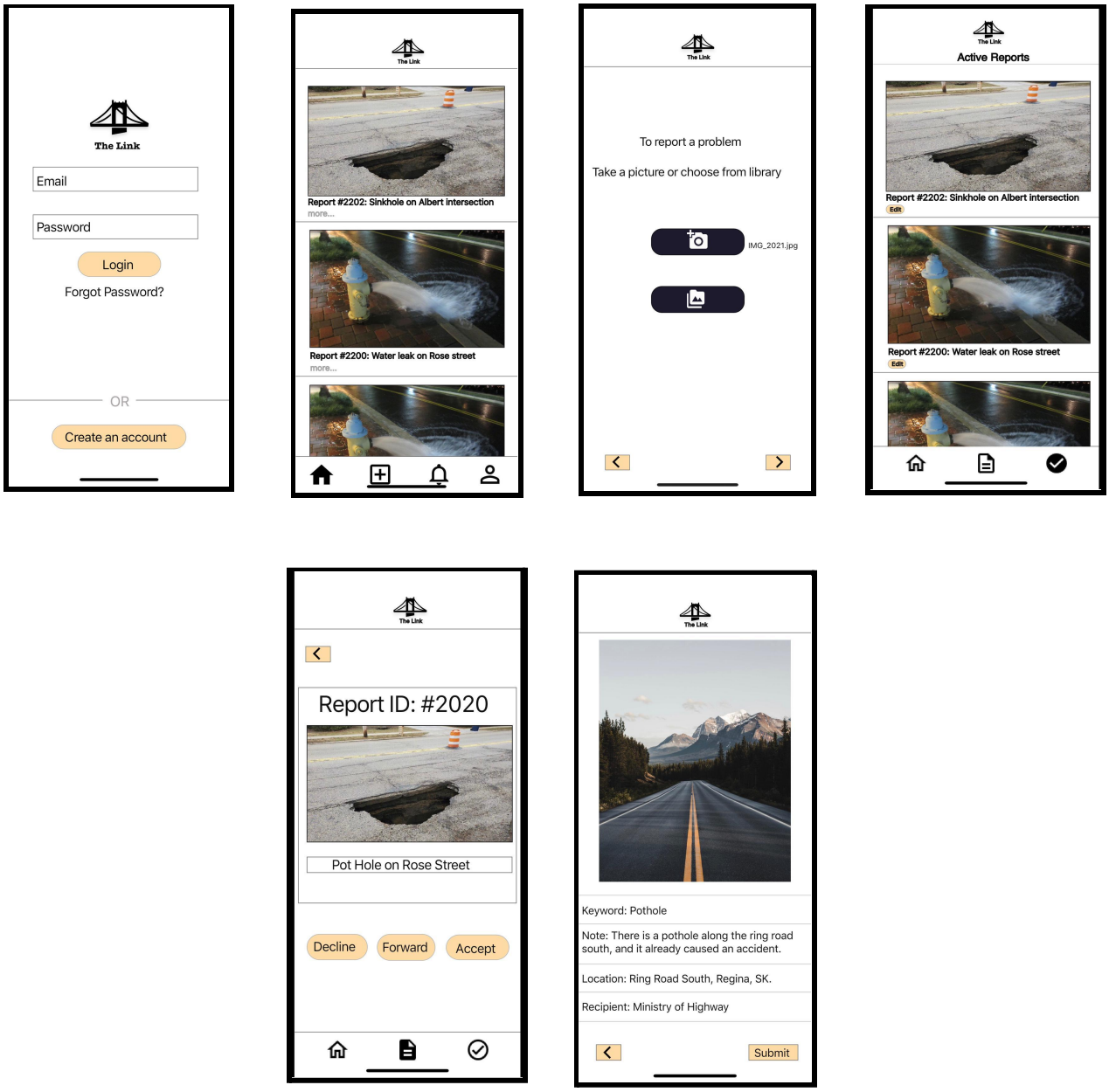

Figure 7, 8, 9, 10, 11, 12: High Fidelity Prototype (October 2020)

Once we were done with our hi-fi, we proceeded to begin the development of our application. During the early stages of development, we encountered some problems that were not ideal for what we desired as an outcome for the project. This led to restrategizing and pivoting to developing a PWA instead, using Firebase and React JS.

As we started development, the original hi-fi design was altered, leading to what we currently have as our final result.

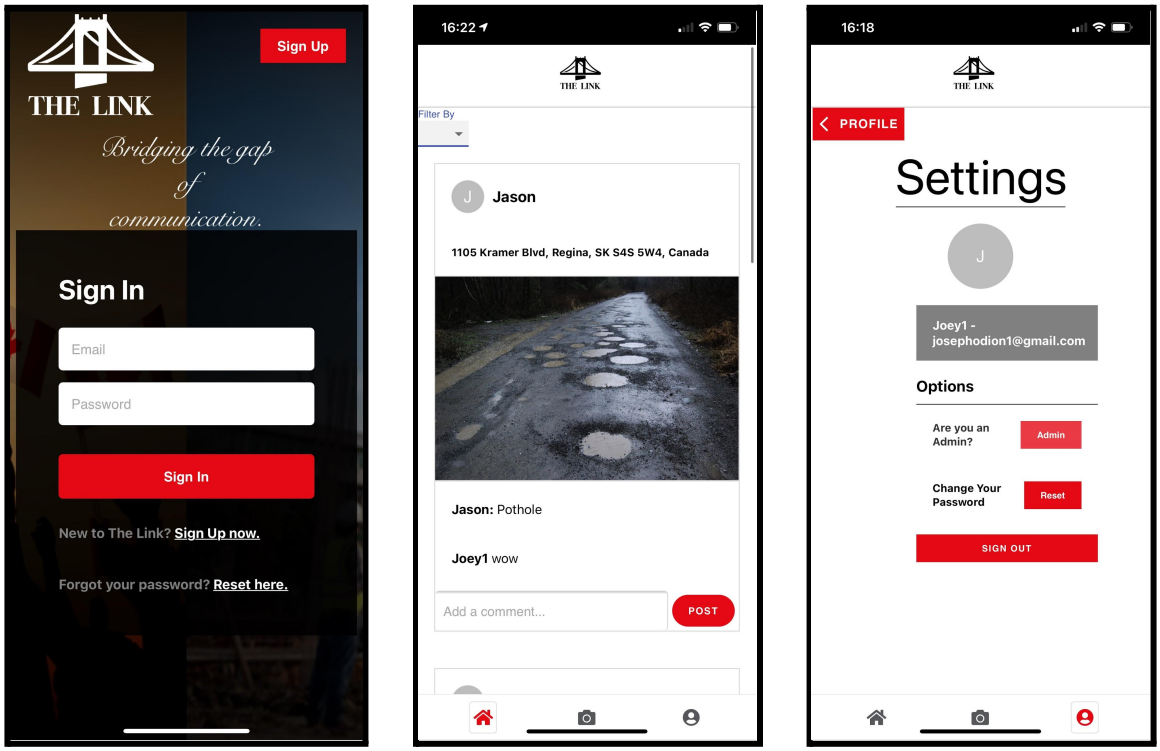

Figure 13, 14: Mobile View of Final UI Design (March 2021)

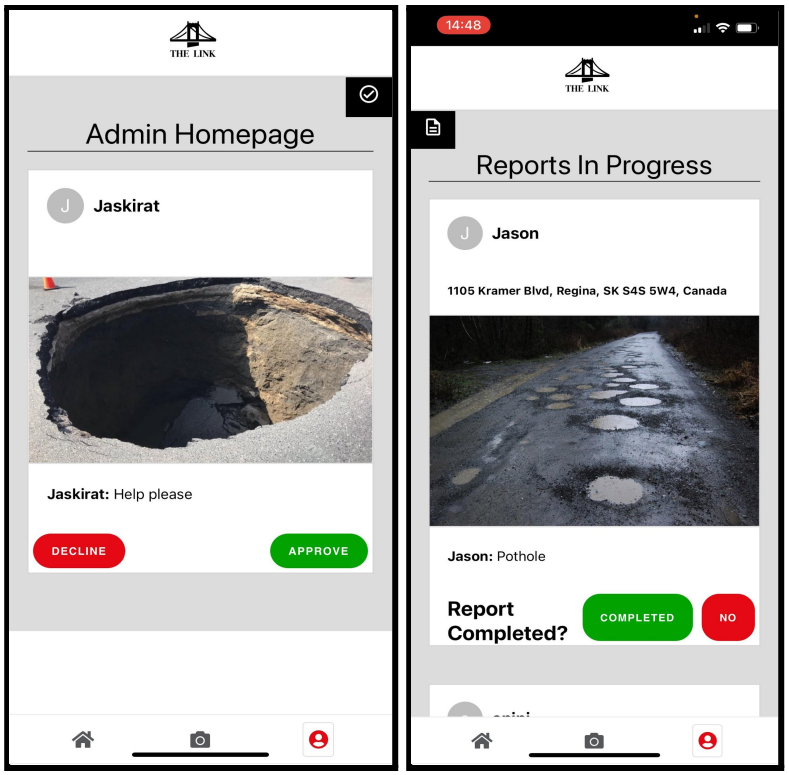

Figure 15, 16, 17: Mobile View of Final UI Design (March 2021)

Our final design still maintains a minimalistic design. Minimalistic designs are what keep users at ease with a product and help keep them easily engaged, with a staying interest. All the user needs is all they see in the application and nothing more or less.

We utilized the principle of learnability when designing the application with respect to its navigation. We went for an intuitive navigation design. Making the app similar to what the users are already used to, helps them better learn the navigation even after one use. We made sure that buttons are clear in colors, the texts on the

buttons are also clear, and we made sure that the users are able to identify when a button is clickable, by making them dynamic when you hover over them. All these help to make the navigation very intuitive. Legibility for our texts and icons were important in our design as it offers clear communication to the users as they navigate through the app. Visibility of elements was quite very important for us as most users will be using this application outdoors. There is a clear contrast between our contents and their backgrounds, which enables the user's identity elements when they are outside.

Considering the fact that our app is made for users who encounter problems outdoors, who will be using their phones, and Admins(City of Regina) who handle the reports in a work environment, we made our application have a seamless experience, as it can be used both on mobile devices and desktop. Finally, we made sure to create a clear distinction between their citizens interface and the admin interface. This can be seen in the color of the background and also in the colour of some of the navigation buttons.

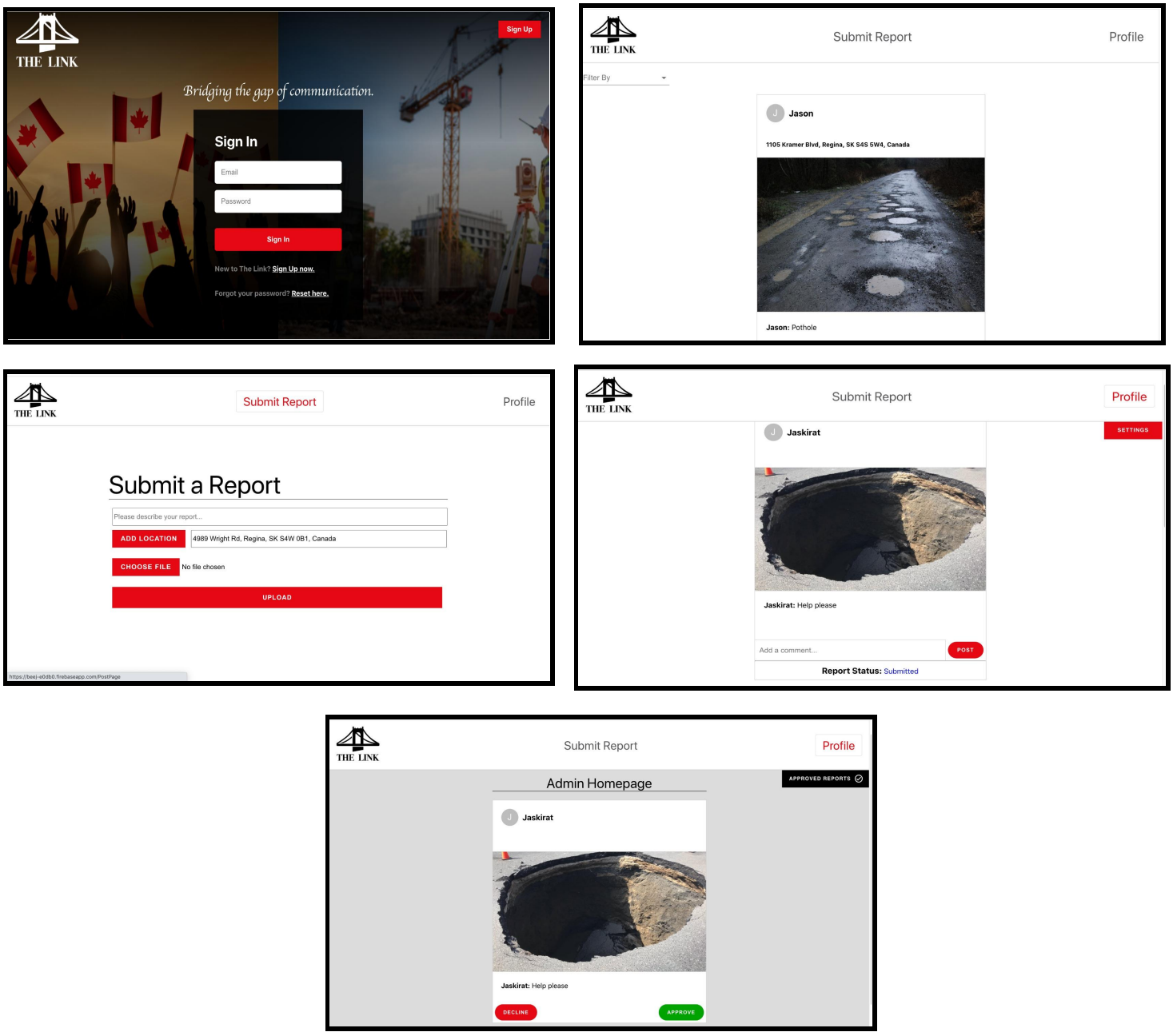

Figure 18,19, 20, 21, 22: Desktop View of Final UI Design(March 2021)

Implementation

Implementation began in late November with the team building the various HTML and CSS pages that would constitute our PWA. We have two views in our PWA:

Ministry and Citizen. Many of our pages for both our views were similar with slight tweaks so we were able to easily make pages. However, it is to be noted that our backend was still being explored at that time and we were solely designing the front end of our PWA.

After a few initial pages were complete, we began working on the backend. In the beginning, we were experimenting with Amazon Amplify to help set up our backend. Amazon Amplify had features like; the AWS Lambda Service - a compute service that allows us to create serverless backend functions that will be triggered based on a specific event you will define in the code; the Amazon API Gateway - a service that allows us to create a RESTful API (defined by HTTP methods) that will allow us to make calls to our Lambda function from a web browser; Amazon DynamoDB - a key-value database service(i.e. we would not need to create a schema for our data) that has consistent performance at any scale and there are no servers to manage when using it. Amazon Amplify had excellent features that would be useful to any web app but they were too broad and we could not have learned everything within the timeframe we had. Hence, we pivoted to Google Firebase.

In early January, the team came to a collective decision to host our PWA on Google Firebase and use React JS as our front-end framework. The Google firebase console was much easier to use and understand. It provided us with prebuilt user authentication functions (including sign in, sign up, password reset, and email verification). It also provided us with a real time database that would allow us to store and sync our web app data in real time. We thought this was important for our app

because it meant that the users would not have to refresh their page to see newly submitted reports on the app. Firebase provided us with storage, which we believed was important for us as we needed somewhere to store images that would be uploaded by the users when they submit a report.

There was a major learning curve while using React JS but the learning experience was worth it. It allowed us to create a web app that can change data, without reloading the page. With react, our app is fast, scalable, and simple. A web manifest file (manifest.json) was created by default which contains customizable configurations that controls the browser's behavior when installed on a user's desktop or mobile device. Our app also came with a service worker - a JavaScript file that runs separately from the main browser thread, intercepting network requests, and caching resources from the cache. It helps ensure that a user will have a seamless offline experience. All static assets are fetched from the service worker cache by default and if that fails the network request is made. However, posts submitted on our app are dynamically generated and can only be cached by adding more configuration to the service worker. This means that when a user loses connection, every other thing on the app loads up except the posts. We are currently thinking about adding that configuration but it is not a priority for our MVP.

Reflection

With our project, we had a hurdle trying to choose an authentication service to handle the authentication of our PWA. We initially had chosen Amplify but after

spending a great deal of time on it we realized it was too complex for our needs and it unnecessarily complicated our project and also took too much time. Upon further research, we decided to switch to Google Firebase which proved to be a much better option for us. However, the time we spent between these two applications could have been avoided if we further researched our options earlier on. However, the learning experience was invaluable in regards to how we can approach future projects and how we can go about researching all options available to us before making a sound decision.

The group broke down the app into a list of features and ordered it by order of importance. This was done to keep everyone on the same page on what was the most important task at hand currently. Due to the switch from AWS Amplify to Google Firebase in early January, the team had to go into overdrive to compensate for lost time spent on AWS Amplify. Due to this, we simply had a google doc with the tasks ordered from most important to least important. It would have been a better idea if the team utilized a Kanban Board to organize tasks and features.

Looking back on the entire experience, there were multiple things we could have improved or changed but there were also multiple things that we did well. Regardless, this project was a great learning experience for all of us, and the lessons we all learned as a group and individually are incomparable.