Log of Features

1. **Hosting**
    a. AWS Amplify

    What Worked
    - By following tutorials on the AWS website, we successfully hosted web pages on the domain.
    - We were able to also Sign in and sign up with Amazon cognito Userpool.

    Issues
    - The features that work with Amplify are too broad for the scope of our app.
    - Amplify has different services including the lambda service, API gateway, and dynamoDB which have a very high learning curve.

    What we ended up using

    b. Google Firebase
    - Firebase console was easier to use and understand
    - Based on research, it proved to be better for the scale of our app.
    - Based on further research, we realized that most of our backend work could be done by firebase. It allows us to use pre-built user-authentication
    - Firebase provides a real time database that would allow us to store and sync our web app data in realtime. We thought this was important because this meant the users would not have to refresh their page when someone submits a new report.
    - It also provides storage, which we believed was important for our app as we needed somewhere to store the pictures that would be uploaded by the users when submitting a report. Hence, we made the decision to switch from Amplify to Firebase.

2. **Front-end Framework**
    a. React JS
    - We chose React JS because based on research, we could use it with firebase by simply creating a JS file that contains the configuration of our existing project in the firebase console. React Apps also come with Web App Manifest and service worker.
    - Major learning curve to become accustomed to React JS. Trial and error as we continue to code our app.

        - Keep learning from various websites and YouTube as problems arise. Lord of trial and error.

3. **<u>Authentication</u>**
    a. Sign up
        - Enable firebase console to handle user sign up with email and password
        - Using modals from Material UI library to implement sign up modal.
        - Use code from firebase documentation to implement the firebase pre-built user-authentication in our app
    b. Sign In
        - Using modals from Material UI library to implement sign in modal.
    c. Email verification
        - Edit configuration file on firestore to customize template email body and subject.
        - Stop sign in if email is not verified.
    d. Sign out
        - Implement logout feature using authentication component provided by firestore library.
4. **<u>Homepage Features</u>**
    a. Submitting a report that includes the ability to:
        - Upload a picture - update code in Home.js to handle this
        - Give a description of the report - update code in Home.js to handle this
        - Choose keyword - update code in Home.js to handle
        - Add comments - update code in Post.js to handle this
    b. Navigation bar (Homepage and profile page)
    c. Filter Homepage

5. **<u>Put choose keyword to choose recipient</u>**
    a. Used drop down feature from material-ui
    b. Scope of our project changed
    c. Had to scrap this feature because of the scope.

6. **<u>User Profile (Citizen)</u>**
    a. Filter posts to show only the user's post on their profile page.
        - Achieved this by adding a filter to the post collection in firebase
    b. Display the status (Approved or declined) of each report on the profile page. This should depend on what decision (approve or decline) was made on the Ministry side
7. **<u>Admin Sign In</u>**

a. Create authorized access for the Admins to the Admin side of the app.
b. Have an authorization code that will be available to the Admins

8. **<u>Approve/Decline on Ministry Side</u>**
    a. Admins should be able to approve and decline a report from the citizens, and the citizens should be able to see this on their profile page.
        - When a report is submitted, an attribute for the status is created in the firebase that will show if a report is approved or declined, after the Admin approves or declines the report.
        - Once a report is approved it moves to the homepage of the citizens side and to the active reports of the admin side.
    b. Reason for Declination
        - Allow Admins give feedback on why the report was declined. We achieved this by providing a textbox that allows the Admins to state their reason for declining a report, and when the reason for declining is submitted, it shows in the users profile page on the post that was declined.
    c. Filter Admin Page
9. **<u>GPS</u>**
    a. Have geolocation services that show a more precise location of a reported problem.
    b. Used Google API to implement location services on our web app.