

National College of Ireland

BSHC

Cyber Security

Academic Year 2020/2021

Mark Noble

X17402254

x17402254@student.ncirl.ie

Hidden Meaning

Technical Report

Contents

Executive Summary	2
1.0 Introduction	2
1.1. Background	2
1.2. Aims.....	3
1.3. Technology	3 - 4
1.4. Structure	3-4
2.0 System.....	4 - 17
2.1. Requirements.....	4 - 14
2.1.1. Functional Requirements.....	4 - 12
2.1.1.1. Requirement 1 User Registration / Login.....	4
2.1.1.1.2 Description & Priority.....	4
2.1.1.4.2 Use Case	5
2.1.1.2. Requirement 2 Add Image / Add Text File	7
2.1.1.2.2 Description & Priority.....	7
2.1.1.4.2 Use Case	7
2.1.1.3. Requirement 3 Image Encoding.....	8
2.1.1.3.2 Description & Priority.....	8
2.1.1.4.2 Use Case	9
2.1.1.4. Requirement 4 Download Encoded File.....	10
2.1.1.3.2 Description & Priority.....	10
2.1.1.4.2 Use Case	11
2.1.2. Data Requirements	12 - 13
2.1.3. User Requirements	13
2.1.4. Environmental Requirements	13
2.1.5. Usability Requirements.....	14
2.2. Design & Architecture	14
2.3. Implementation	14
2.4. Graphical User Interface (GUI)	14 - 17
2.5. Testing.....	17
2.6. Evaluation	17
3.0 Conclusions	17
4.0 Further Development or Research	17
5.0 References	17
6.0 Appendices.....	17 - 22
6.1. Project Plan	18 - 21

6.1.	Ethics Approval Application (only if required)	22
6.2.	Reflective Journals	22
6.3.	Other materials used	22

Executive Summary

This technical report will outline the Background, Aims, System Requirements, Data Requirements, User Requirements, Environmental Requirements, Usability Requirements, etc for an application that is designed to allow a user to register or login an account that is authenticated on a user database from here upload an image file or take a picture with included text and have an encoded image file produced that looks no different to the naked eye as the original image but will include an encoded text file. This project addresses a complex and real problem in Cyber Security highlighting the topic of message delivery and password storage both of which is addressed in the Aims and Implementation of this project. The purpose of this project is to give the average user access to an image steganography service implementing a image steganographic algorithm with all original text and image files being used as well as the newly produced image files being produced which will be sent to a centralized repository where they can be stored and distributed. The application features a User Authentication service where the username and password are held to current standards when registering and are compared to the user database to ensure that it does not match a current user account. From here a storage section is provided to allow for the storage of any encoded files if they need to be archived for the future use. This project will demonstrate and evaluate the aims outlined below showcasing a react-native front end allowing the user to use this application on web, android, and ios with ease allowing for greater accessibility and usability, all user authentication, image storage, and entry creation is handled using Firebase as the backend.

1.0 Introduction

1.1. Background

My main objective for undertaking this project was to gain a deeper understanding of steganography as well as steganographic principles and techniques. I am interested in investigating the various types of steganography whether that be text, image, audio, video, etc and how you would implement them into a technology in a practical and useful manor. Through my research I was unable to find a easy to use intuitive web based application on the market that allows your everyday user to avail of image steganography services as a way of protecting user information such as confidential details, passwords, confirmation numbers, order numbers, etc and as image steganography is a method unknown to a large amount of the population I thought it would be an interesting and secure concept to develop an application that would allow your average non-technical user to have access to these kinds of services. I plan to develop several different strategies on how to approach

this problem of implementation showcasing the multitude of different technologies and uses that can be used to complete the objective of this project and finally settle on one method I feel best suits all my objectives.

1.2. Aims

- I aim to have a greater understanding of steganography and steganographic techniques with knowledge on how to effectively implement it in a practical application. I will achieve this by investigating the different types of steganography as well as attempting to implement a finally selected type into several methods and see which one would be best suited for use.
- I aim to develop an application that sufficiently secures user information as well as provides a proper method for encoding user information. I will achieve user security using Cloud Based User Authentication using a username and password as well as a randomly generated userID provided on account creation. Following this I will achieve encoding through implementing steganographic techniques and algorithms.
- I aim to have a fully working product that is up to date with programming standards with an intuitive graphic user interface. I will achieve this with the Gantt chart that was provided in my project proposal, this will be used to develop soft deadlines and give an overall timeline in the production time of the product. Following this I will research all current up to date programming standards to be implemented while fine tuning a GUI through various iterations of the product.
- I aim to create a service that any average user could make use on any android, ios or computer giving them access to a image steganography tool that will allow them to hide messages within any image they chose allowing for the safe communication between parties online as well as allowing for the safe storage of passwords.

1.3. Technology

The technologies that will be used in this project to allow it to attain full functionality as well as be scalable in the case that the user base was for some reason to expand at a rapid rate are as follows:

- Microsoft VS Code – This will be used to develop and test the application backend and front end through development.
- Github – This will be used to store the iterations of the application serving as a backup.
- Firebase – This is a cloud based Storage that will be used in order to Authenticate and Register new users as well as store all files images, text files, etc associated with user accounts.
- React Native – The framework for the project allows for rapid development on web, android, and ios.
- Image Stenography –Encoder, Text Decoder, ImageLoader, ImageSaver,TextToByte Converter, ByteToText Converter.

- Firebase(JSON)(FireAuth,FireStorage,FireStore) – To be used as a cloud hosted database for user authentication, image storage, and entry creation.
- JavaScript / jQuery – Updating a static page with information.
- CSS3 – Cascading Style Sheet.

1.4. Structure

This report is broken down further into a number of different sections that detail the requirements for the Hidden Meaning application it will include:

Section 2 -

- Requirements – This includes Functional Requirements in a ranked order detailing what the system is required to achieve in its processes, as well as Data requirements, User Requirements Environmental Requirements and Usability Requirements.
- Design and Architecture – This will include how the full system architecture is designed showcasing what components, algorithms etc that are required in order for the application to function.
- Implementation – This section shows the different components, algorithms, etc shown in the project similar to the design and architecture section but in greater details breaking down each section of the code with snippets to accompany them explaining each section in detail.
- Graphic User Interface (GUI) – This section will be used in order to showcase various section of the user interface through screenshots with different explanations on the various functions displayed on screen with an accompanying section explaining design details and features.
- Testing – In this section testing tools, test plans and test specifications are detailed and broken down showing how proper and rigorous Unit Testing as well as other types of testing has taken place and their results.
- Evaluation – In this section the system performance, usability, scalability, etc are measured and charted in order to give a thorough breakdown of the system in order to best showcase its capabilities with clear and concise explanation.

Section 3 – This section includes the Conclusion this is used in order to describe the advantages/disadvantages, strengths and limitations of the project.

Section 4 – Further Development and Research is included in this section, this section of the project is used in order to describe a hypothetical scenario whereas if you had more time and resources in what direction would you take the project or what changes would you make to the project.

2.0 System

2.1. Requirements

2.1.1. Functional Requirements

2.1.1.1 Requirement 1: User Registration / Login

2.1.1.1.1 Description & Priority

This requirement is necessary for a new user to gain access to the application services. The reason for implementing a user registration process coupled with user login authentication is to track what users are using the system process for what purposes as well as to track what users are accessing the services. As the application is not accessible without a user account this is of medium importance though it is used to register a user it is not integral to the steganographic system.

2.1.1.1.3 Use Case

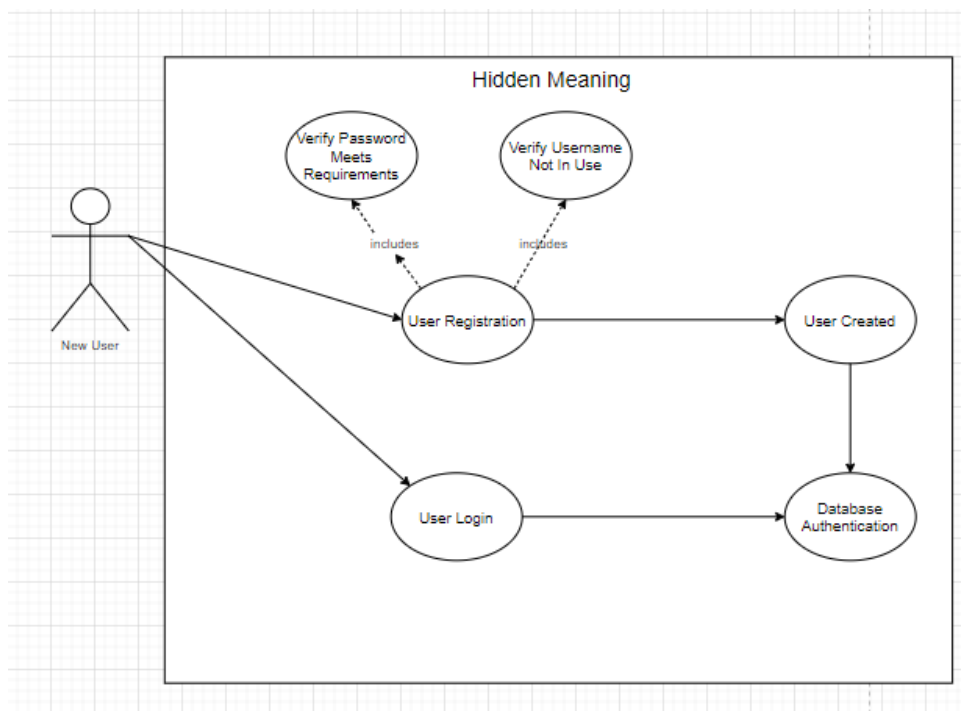
Scope

The scope of this use case is to implement a user registration system adhering to NIST standards where all fields must be verified before a successful account registration is possible. The system will be able to read input fields and display a message for incorrect entry detailing the error, if all fields are correctly filled a new user will be created in the user database. User will be able to login to application through database authentication

Description

This use case describes the user registration/login feature.

Use Case Diagram



Flow Description

Precondition

- The user that is being registered must not already exist in the system.
- The user that is being registered must enter a password complying with NIST standards.
- The user that is being logged in must already exist in the system.

Activation

This use case starts when a User attempts to register / login.

Main flow

1. The User selects register on the home screen. (See A1)
2. The User registers a username. (See A2)
3. The User registers a password complying with NIST standards.(See A3)
4. The system identifies the username is not in use and password comply with NIST standards.
5. The system sends the user information to create a database entry.
6. The User is brought to the application home screen.

Alternate flow

A1 : <User selects login screen>

1. The system displays the login screen.
2. The User enters the username and password.
3. The system identifies the user account and compares the login information (See A4).
4. The use case continues at position 6 of the main flow

A2 : <Username is already in use>

1. The system compares the username entered with those on the database.
2. The system identifies the username is already in use.
3. The use case continues at position 2 of the main flow.

A3 : <Password doesn't meet NIST standards>

1. The system compares the password entered with NIST Standards.
2. The system identifies the password does not meet these standards.
3. The use case continues at position 3 of the main flow.

A4 : <Login Credentials are incorrect>

1. The system compares the username and password entered against the database.
2. The system identifies the username or password does not match.
3. The use case continues at position 2 of A1.

Termination

If the registration flow is successful the user is brought to the login screen. If the login flow is successful the login / registration process is completed.

Post condition

- The user will have a created account.
- The user will successfully login.

2.1.1.2 Requirement 2: Add Image / Add Text File

2.1.1.2.1 Description & Priority

This requirement is an integral function of the application it will allow a logged in user to upload a new image and new text file to the application so that the new image uploaded will be encoded with the accompanying text file. This is a high priority function as it directly relates to the functionality of the application.

2.1.1.2.2 Use Case

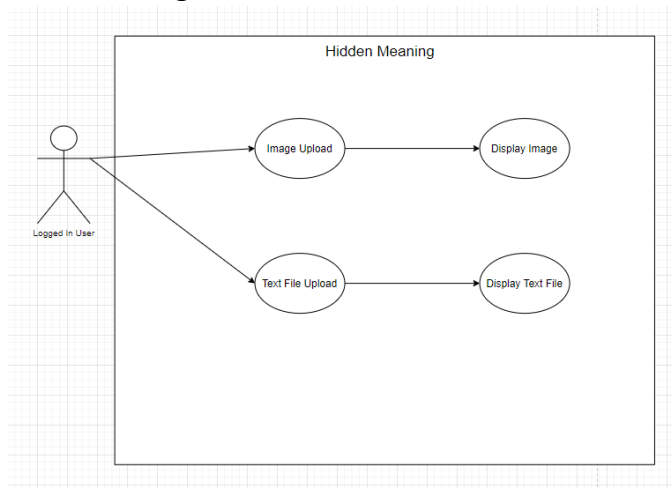
Scope

The scope of this use case is to show how a logged in user will upload images and text files to the application to be used.

Description

This use case describes the logged in user uploading an image and text file.

Use Case Diagram



Flow Description

Precondition

- The user must have an image to upload.
- The user must have a text file to upload.
- The user must be logged in.

Activation

This use case starts when a logged in user clicks onto the upload section.

Main flow

1. The system identifies the upload button has been pressed.
2. The system will prompt the user to upload an image with certain file types followed by a text file in the .txt file type.
3. The user will upload a image file and .txt file. (See A1)
4. The system will display the image and text file to be uploaded.
5. The system will prompt the user to confirm selection.
6. The user will confirm the selection.
7. The system will upload the two files. (See E1)

Alternate flow

A1 : <Incorrect File type uploaded>

1. The system compares the image and text file types to useable types.
2. The system finds incompatible file types.
3. The system prompts the user to change the file type.
4. The use case continues at position 2 of the main flow.

Exceptional flow

E1 : <Upload Failure>

1. The system is unable to upload the files.
2. The system prompts the user to reupload the files.
3. The use case continues at position 2 of the main flow

Termination

If the image and text file are uploaded successfully the Upload flow is completed.

Post condition

- The user will have an image file available for use in the system.
- The user will have a text file available for use in the system.

2.1.1.3 Requirement 3: Image Encoding

2.1.1.3.1 Description & Priority

This requirement is an necessary function of the applications performance and function as a whole. The image encoding module is necessary in order to make the application

function as it has been outlined in the initial project proposal as this is the case a high priority has been placed on this process.

2.1.1.3.2 Use Case

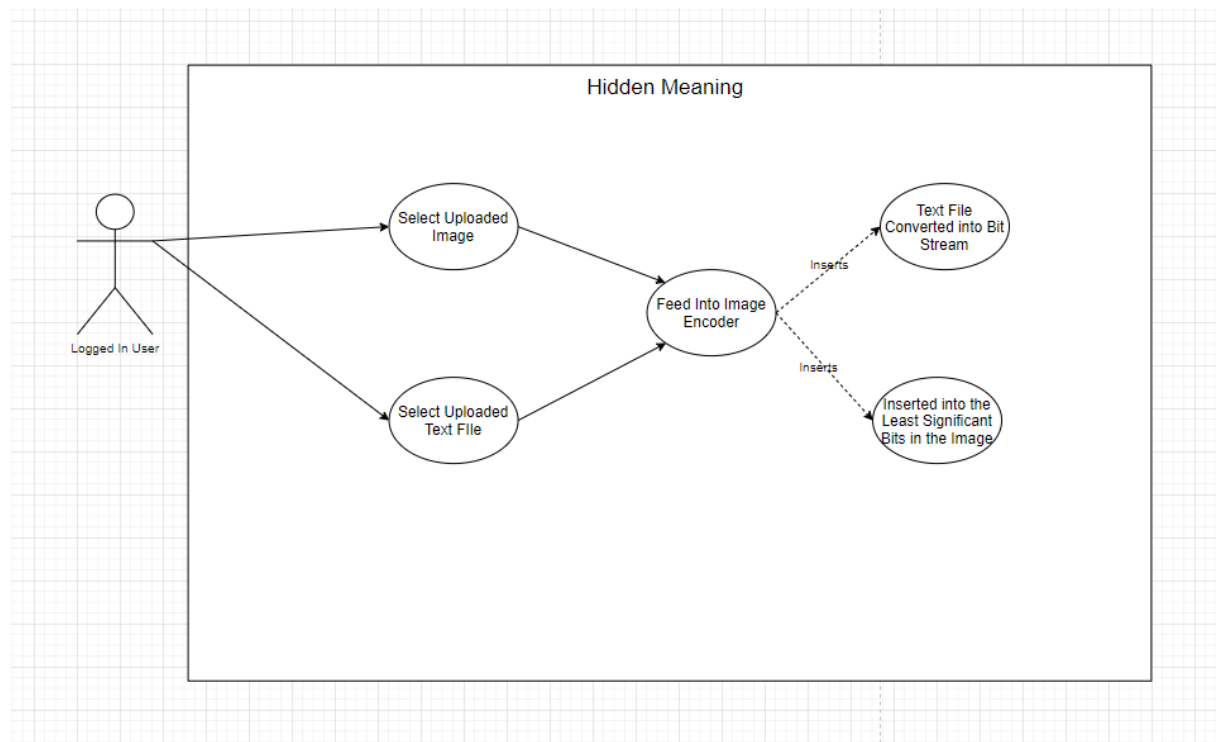
Scope

The scope of this use case is to show how the image encoding process works when a text file and image and been passed through their use case ready to be encoded.

Description

This use case describes the logged in user encodes a text file into an image.

Use Case Diagram



Flow Description

Precondition

- The user must have an image uploaded already.
- The user must have a text file uploaded already.
- The user must be logged in.

Activation

This use case starts when a logged in user clicks onto image encoding section.

Main flow

1. The system identifies the image encoding button has been pressed.
2. The system will prompt the user to select an already uploaded image.
3. The system will prompt the user to select an already uploaded text file.
4. The user will select an image file and .txt file.
5. The system will display the image and text file to be uploaded.
6. The system will prompt the user to confirm selection.
7. The user will confirm the selection.
8. The system will upload the two files to the image encoder. (See E1)
9. The system will convert the text file into a bit stream.
10. The system will analyse the image to determine the least significant bytes / unused bytes and insert the bit stream that has been generated into these insignificant bytes.
11. The system will prompt the user with a “process successful” message

Exceptional flow

E1 : <Upload Failure>

4. The system is unable to upload the files.
5. The system prompts the user to reupload the files.
6. The use case continues at position 2 of the main flow

Termination

If the image encoding is success the Encoding flow is completed.

Post condition

- The user will have a newly produced encoded image to download.

2.1.1.4 Requirement 4: Download Encoded File

2.1.1.4.1 Description & Priority

This requirement is a part of the core system but is not integral to the core function of the system as the image that is being downloaded is generated from Requirement 3 Image Encoding and is uploaded onto a cloud-based server. This is ranked as a medium priority requirement.

2.1.1.4.2 Use Case

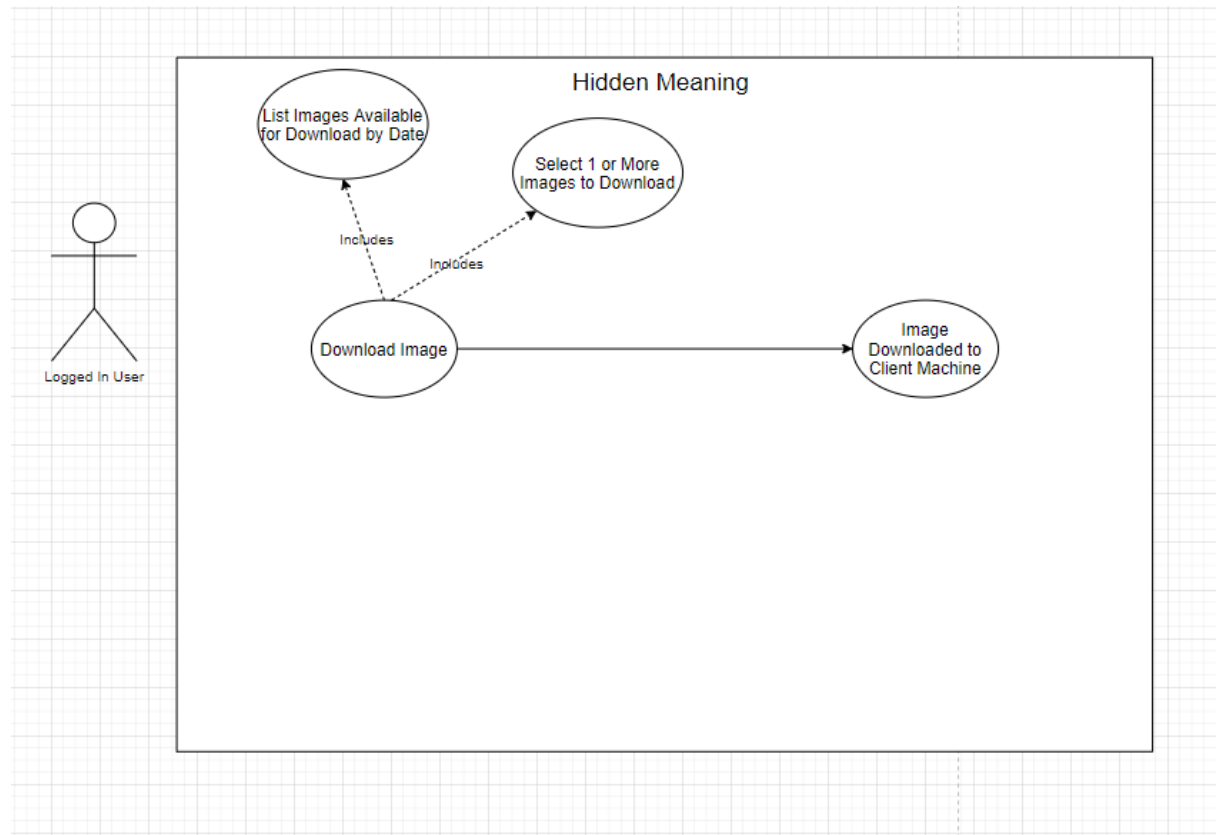
Scope

The scope of this use case is to show how a logged in user can download a newly generated encoded image.

Description

This use case describes the logged in user downloading a image that has been encoded and generated as mentioned in the image encoding requirement.

Use Case Diagram



Flow Description

Precondition

- The one or more image(s) must already exist on the database.
- The user must be logged in.

Activation

This use case starts when a logged in user clicks onto the download section.

Main flow

1. The system identifies the download button has been pressed.
2. The system will prompt the user to select one or more files to download.
3. The system will prompt the user to confirm selection.
4. The user will confirm the selection. (A1)
5. The user will download 1 or more image files.
6. The system will prompt the user that one or more images have been downloaded. (See E1)
7. The user will be brought to the home screen.

Alternate flow

A1 : <No Files Selected>

5. The system compares the selected image against the database.
6. The system finds that no images have been selected.
7. The use case continues at position 2 of the main flow.

Exceptional flow

E1 : <Download Failure>

7. The system is unable to confirm the files for download.
8. The system prompts the user to reselect the files.
9. The use case continues at position 2 of the main flow

Termination

If the image(s) are successfully downloaded the Download Encoded File requirement is completed.

Post condition

- The user will have an image file that has been encoded downloaded.
- The user will be brought to the home screen.

List further functional requirements here, using the same structure as for Requirement1.

2.1.2 Data Requirements

In the Data Requirements section, I will review and explain the various data requirements that will be included in this project that are integral for the functionality of this application. The data that is collected and processed by this application will not be held on the application for any log period to improve system performance and allow for use on a wider variety of systems.

Firebase Authentication

What is Firebase Authentication?

Firebase Authentication is essentially a backend cloud-based database service that can be integrated with several different code bases and applications implementing several easy-to-use SDK's as well as a UI library that allows for incoming users to be authenticated securely and effectively. It supports a variety of different authentication methods phone, email, etc but for the purposes of this application only username, password and for some user's pin will be implemented for extra security.

The purpose of using Firebase Authentication in this application is to allow for a cloud-based service to be used providing a higher level of security tracking user actions and usage as well as only allowing registered users to read and right to the database. The username and NIST password conditions are set on the application when the inputs are fed into the registration function if both the username and

password pass through this they will be passed to the Firebase Authentication database where they can then be used for any future login or in order to assign any information such as images, text files or encoded text files to a specific user.


Identifier	Providers	Created	Signed In	User UID ↑
mark@mail.com		Nov 30, 2019	Dec 12, 2019	0TIP9vxNkehQ6W56k2epnROTzuo1

Figure 1 shows an example of a user entry in the Firebase Auth Database.

Firestore Storage

Firestore Storage is a cloud-based storage solution that can be used with the previously mentioned Firebase Authentication database. When used in tandem with Firebase Auth a user can link any content that is stored in the Firestore Storage whether that be a stored image file, text file or encoded image file they can be stored here and assigned to a user account at the same time. Previously to store an image file on Firestore you would have had to store then as a base64 encoded file and store them within a JSON database with this new method the image quality can be preserved without the need for any conversion.

Firestore Firestore

Image Files

The user of this application will need to be able to upload images though an ImagePicker alternatively they should be able to use their computer, phone, or tablets integrated camera to provide image data to be used in the encoder. This image data will be necessary to fulfil one of the use cases of image steganography.

Text Files

The user of this application will need to be able to provide text (String) input through the use of a <TextInput> this is necessary in order to fulfil the other part of the use case for image steganography where a text input and image data is needed in order for encoding to take place.

2.1.3 User Requirements

User Requirements are defined as what the User needs from the system with this being the case there are a few user requirements for this application and they are as follows:

- The User must be able to register a user account.
- The User must be able to login to their user account.
- The User must be able to upload images.
- The User must be able to upload text.
- The User must be able to encode a text file into an image file.
- The User must be able to download the encoded image file.
- The User must be able to view all stored images files that are associated with their account.
- The User must be able to easily navigate from the home screen to all essential functions of the application and back to the home screen.

2.1.4 Environmental Requirements

To use the application, you will need to have a pc, laptop, or mobile device to access the web-based application. Along with this you will need a stable internet connection coupled with your device to gain access to the web-based application and the cloud-based database containing the user authentication database and associated user files. The user will also need to have the free space on their machine to download the newly encoded image file.

2.1.5 Usability Requirements













Hidden meaning will have an intuitive user interface allowing for seamless navigation from the home screen to any of the core function such as registration, login, upload, download, view database functions as well as navigating back to the home screen. To complete any function of the application as well as the fact that it is a web-based application a stable and constant connection to the internet through hard connection, Wi-Fi or cellular 2G,3G,4G,5G etc is necessary as there is constant reading and writing to the Firebase Auth, Firebase Storage and the hosted location of the web application.

The visuals of the project will include an off-white background as well as a minimalist layout to ensure there is little clutter on screen and any colour that is used to style the page will use soft colours so as not to offend the eyes.

2.2 Design & Architecture

In this section I will be outlining the structure of my project essentially highlighting how the react native front end communicates with the firebase backend and how this system of communication functions furthermore I will detail the steganographic techniques used and how it functions in this project. To start off I will begin to describe the steganographic technique I used in this project, there are a number of functions included in the steganography javascript file in this project (steggo.js) the process is as follows. For the encoding of an image you will need two inputs to the function

imageData and text this will be provided by the. The backend of this application is handled by Firebase, firebase is a Google run and developed platform for project backend it handles a variety of services and functions that can be integrated with web based frameworks such as react native as well as with Android development and much more. It provides SDKs that can be used on the client side this is typically used to avoid having to repetitively rehash the same apis and functions that could be considered as boilerplate when you could be focusing on the unique development of the application instead. Firebase is a cloud based service using JSON formatting for its entries and as mentioned before it provides the developer with Client side SDKs that can be used to directly communicate with the Firebase services essentially cutting out the need for Middleware altogether where some database services and languages may require you to log your queries on the Server side using firebase this can be done on the Client side. In this application I will be using the Firebase User Authentication, FireStorage, and Firestore. For the User Authentication portion of the project I wanted to keep the sign in and register process as seamless as possible to achieve this I needed to assess the various options provided to me by Firebase and determine what would be the best to fit my use case. Firebase Authentication allows for the use of a variety of mediums for register and sign in such as Phone, Google Sign In, Facebook, Twitter, Github, as well as Email and Password which was the service that I determined would be the best to fit my use case.

Sign-in providers	
Provider	Status
 Email/Password	Enabled
 Phone	Disabled
 Google	Disabled
 Play Games	Disabled
 Game Center	Disabled
 Facebook	Disabled
 Twitter	Disabled
 GitHub	Disabled
 Yahoo	Disabled
 Microsoft	Disabled
 Apple	Disabled
 Anonymous	Disabled

Using the email and password option I was able to ensure accessibility to a larger user base was achieved in the case that I selected Google, Apple, Yahoo, or one of the other options I might have limited the scope of who could avail of my application contradicting one of the core aims of this project. The User Authentication takes place on as part of

the StackNavigator of the Landing page this allows for the navigation between the Landing, Login, and Register pages in the application this is one of the key aims and usability requirements of the application. The firebase API Key, authDomain, projectId, storageBucket, messagingSenderId, appId, measurementID are located in the App.js this is done in order to ensure that the firebase credentials can be located in all pages making use of App.js further then this there is a initialization checker and auth state change listener in place to ensure that there are no current firebase instances initialized already and to note when the auth change takes place.

```
//This checks to ensure that there are no current firebase instances initialized.  
if(firebase.apps.length === 0){  
  firebase.initializeApp(firebaseConfig)  
}
```

```
//This is a listener for auth state change  
  
componentDidMount(){  
  firebase.auth().onAuthStateChanged((user) => {  
    if(!user){  
      this.setState({  
        loggedIn: false,  
        loaded: true,  
      })  
    }else{  
      this.setState({  
        loggedIn: true,  
        loaded: true,  
      })  
    }  
  })  
}
```

The navigation in this app is done through a NavigationContainer and StackNavigator initially when you are signed out you will land on the Landing page where you can select to login or register this is the initial route set for non authenticated users.

```

if(!loggedIn){
  return (
    <NavigationContainer>
      <Stack.Navigator initialRouteName = "Landing">
        <Stack.Screen name = "Landing" component = {LandingPage} options = {{ headerShown: false}}/>
        <Stack.Screen name = "Register" component = {RegisterPage}/>
        <Stack.Screen name = "Login" component = {LoginPage}/>
      </Stack.Navigator>
    </NavigationContainer>
  );
}
return(
  //Provider must be the parent of MainPage to ensure that redux will work inside MainPage
  //NavigationContainer must wrap the Stack Navigator in order for the BottomTab React Navigation to work.
  //The Stack in the Navigator is responsible for moving from page to page allowing for this to be done as seen in the Confirm Button
  //on the Upload.js section.
  <Provider store = {store}>
    <NavigationContainer>
      <Stack.Navigator initialRouteName = "Main">
        <Stack.Screen name = "Main" component = {MainPage} options = {{ headerShown: false}}/>
        <Stack.Screen name = "Upload" component = {UploadPage} navigation={this.props.navigation}/>
        <Stack.Screen name = "Entry" component = {GeneratePage} navigation={this.props.navigation}/>
      </Stack.Navigator>
    </NavigationContainer>
  </Provider>
)

```

Once you are successfully signed in you will be brought to the Home page where you are greeted with a Bottom Navigator to change between the various page views this is accomplished through a react native package BottomTabNavigator in tandem with MaterialCommunityIcons to give a UI to navigate with.

```

return (
  <Tab.Navigator initialRouteName="Home">
    <Tab.Screen name="Home" component={HomePage}
      options={{
        tabBarIcon: ({ color, size }) => (
          <MaterialCommunityIcons name = "home" color={color} size ={26}/>
        ),
      }}
    />
  </Tab.Navigator>
)

```

Another aim of the project was for the storage of user images to a cloud-based database this was accomplished through firebase storage and firebase firestore. The firebase storage occurs on the Upload and Entry page where the image is taken through the integrated camera or is selected through the image picker after you confirm the image selection you will be routed to the Entry page where the image you have selected will be brought in the form of a prop assigned to a variable called uri and then passed into the firebase query to save it to the firebase storage appending the currentUser uid to it to ensure it is unique to the user only.

```
let newerURI
```

Further down the entry page you will also see the function put in place to create a new entry this passes the downloadURL that is used to identify the URI of the image this is then passed to the entries folder in the firebase firestore and then a .doc with the currentUser uid is assigned as the name of the folder which will add the downloadURL, and creation time within uniqueEntries.

2.3 Implementation

As outlined in the Design and Architecture portion of the project the Firebase Authentication is an integral and key portion of the project it is stated in the project App.js file there is the API Key, authDomain, projectId, storageBucket, messagingSenderId, appId, measurementID as stated above this is used in order for a link to be established with the Firebase services through an API allowing the other sections of the project to communicate with the various firebase services. As well as that there is a firebase initialization checker to ensure that there are no current firebase instances established and a onComponentMount to listens for authentication state changes this is shown in the code segments below.

```
//This checks to ensure that there are no current firebase instances initialized.  
if(firebase.apps.length === 0){  
  firebase.initializeApp(firebaseConfig)  
}
```

```
//This is a listener for auth state change  
  
componentDidMount(){  
  firebase.auth().onAuthStateChanged((user) => {  
    if(!user){  
      this.setState({  
        loggedIn: false,  
        loaded: true,  
      })  
    }else{  
      this.setState({  
        loggedIn: true,  
        loaded: true,  
      })  
    }  
  })  
}
```

When authentication is taking place it will utilise all of this before the fire query is even made to ensure a successful connection and secure experience with the auth listener. The authentication takes place on the Register page for creating a user account and on the Login page to validate the credentials in the text inputs. The register page uses a series of text input to gather the username, email, and password and after these fields are successfully filled out the onSignUp() function will be called.

```

render() {
  return (
    <View>
      <TextInput
        style={styles.textInput}
        placeholder="Username"
        onChangeText={(name) => this.setState({ name })}
        maxLength={8}
        autoCapitalize='none'
      />
      <TextInput
        style={styles.textInput}
        placeholder="Password"
        secureTextEntry={true}
        onChangeText={(password) => this.setState({ password })}
        secureTextEntry={true}
        maxLength={16}
        autoCapitalize='none'
      />
      <TextInput
        style={styles.textInput}
        placeholder="Email"
        onChangeText={(email) => this.setState({ email })}
        autoCapitalize='none'
      />
      <Button
        onPress={() => this.onSignUp()}
        title="Sign Up"
      />
    </View>
  )
}
}

```

The onSignUp() function passes the email, password, and name fields from the text input and equal them to this.state after this the email and password fields are passed to a firebase query with the .auth identifier calling on the createUserWithEmailAndPassword function passing the previously mentioned email and password through a .then(result) we get the final result of running this command this is then passed through the firebase.firestore.collection directly into the "users" folder in the firebase firestore essentially creating a new user entry with a unique user ID which is gotten from the .doc(firebase.auth().currentUser.uid) using this the email and password is passed through to firebase creating an auth entry and generating a user entry in the firestore with a unique uid.

```

onSignUp(){
  const { email , password, name } = this.state;
  firebase.auth().createUserWithEmailAndPassword(email, password)
    .then((result) => {
      firebase.firestore().collection("users")
        .doc(firebase.auth().currentUser.uid)
        .set({
          name,
          email
        })
      console.log(result)
    })
    .catch((error) => {
      console.log(error)
    })
}

```

The Login page functions in a similar manor to this there is an onSignIn() method that creates a const using the email and password and assigning that to the this.state using a firebase.auth().signInWithEmailAndPassword passing the email and password previously mentioned we can get the result as positive or negative after a successful result you are routed to the Home page when authenticated.

```

onSignIn(){
  const { email , password } = this.state;
  firebase.auth().signInWithEmailAndPassword(email, password)
    .then((result) => {
      console.log(result)
    })
    .catch((error) => {
      console.log(error)
    })
}

```

The core of the project is based around image steganography the image steganography Java script file contains a total of eight functions all of which I will outline here: The first function is loadImage() the load Image function will take the imageData that has been passed through from the Upload page in the form of a uri converted from the camera/ image picker input. That image data is then loaded into an image canvas that is declared in the return section of the Upload Page. When the image is loaded into the canvas the RGB values can be attained.

```
const loadImage = (uri) => {
  return new Promise(((resolve) => {
    const ctx = document.getElementById('canvas').getContext('2d')
    const image = new Image()
    image.onload = function () {
      ctx.canvas.hidden = true
      ctx.canvas.width = image.width
      ctx.canvas.height = image.height
      ctx.drawImage(image, 0, 0)
      image.style.display = 'none'
      resolve(ctx.getImageData(0, 0, ctx.canvas.width, ctx.canvas.height))
    }
    //This is to circumvent Cors rules
    image.crossOrigin = "anonymous"
    image.src = uri
  })))
}
```

The next function in the steganography section is the saveImage function in this function the imageData that has been ran through the encoding function will be saved here and converted to a blob this is in order to circumvent an uploading issue pertaining to firebase firestore.

```
const saveImage = (imageData, callback) => {
  const ctx = document.getElementById('canvas').getContext('2d')
  ctx.putImageData(imageData, 0, 0)
  return ctx.canvas.toBlob(callback)
}
```

The textToBytes function is responsible for as the name would suggest converting the inputted text being passed from the entry page into an array of bits each character in the string is akin to 1 bits in the array.

```
const textToBytes = (text) => {
  let bytes = []
  for (const char of text) {
    let code = char.charCodeAt(0).toString(2)
    if (code.length > 8) {
      throw Error(`Invalid character for 8-bit binary encoding: ${char}`)
    }
    bytes.push("0".repeat(8 - code.length) + code)
  }
  return bytes
}
```

The fourth function and the complete opposite to the textToBytes function is the bytesToText function where an array of bits is taken in and then parsed converting the array of bits to a string.

```
const bytesToText = (bytes) => {
  let text = ""
  for (const byte of bytes) {
    text += String.fromCharCode(parseInt(byte, 2))
  }
  return text
}
```

The fifth and one of the most important functions is the encodeMessage function this is responsible for taking in the imageData and text it will iterate over every byte in the string for each bit in the string it will set the next RGB value to be even if it is 0 or odd if it is 1. In this function the Alpha in RGBA is skipped as it is unnecessary to include it as it only relates to opacity. At the end of each byte the RGB will be set in order to determine if the encoder needs to continue iterating through the bytes or not if the byte ends on an odd then there is more bytes to encode if it falls on an even it can stop.

```
const encodeMessage = (imageData, text) => {
  let bytes = textToBytes(text)
  let data = imageData.data
  let i = 0
  let byte
  for (let j = 0; j < bytes.length; j++) {
    // Encode jth byte.
    byte = bytes[j]
    for (const bit of byte) {
      if (bit === "0" && data[i] % 2 !== 0) {
        data[i] -= 1 // Make data entry even if bit is zero.
      } else if (bit === "1" && data[i] % 2 === 0) {
        data[i] += 1 // Make data entry odd if bit is one.
      }
      i += i % 4 === 2 ? 2 : 1 // Skip alpha value.
    }

    // Signify whether message is complete.
    if (j < bytes.length - 1 && data[i] % 2 === 0) {
      data[i] += 1 // Still more bytes to encode, make data entry odd.
    } else if (j === bytes.length - 1 && data[i] % 2 !== 0) {
      data[i] -= 1 // No more bytes to encode, make data entry even.
    }
    i += 2 // Skip alpha value.
  }
  return imageData
}
```

Just as the `encodeMessage` function is integral to the functioning of the steganography portion of the project the decoding function is just as important. The decoding function works essentially in the opposite way of the encoding function the decoder will iterate over the RGB values if it detects an odd value it assigns it to 0 and if its even it will be assigned a 1 after 8 bits are read there is then a byte saved. This works in the opposite way to the encoding function in that it will choose the odd and even RGB values and reconstruct the image using a canvas in the Entry page to reconstruct the image onto the canvas.

```
const decodeMessage = (imageData) => {
  const bytes = []
  let data = imageData.data
  let bits = ""
  for (let i = 0; i < data.length; i++) {
    if (i % 12 === 10) { // Check if signal bit
      bytes.push(bits)
      if (data[i] % 2 === 0) {
        return bytesToText(bytes)
      } else {
        bits = ""
        i++
      }
    } else if (i % 4 !== 3) { // Skip alpha value.
      bits += data[i] % 2 === 0 ? "0" : "1"
    }
  }
}
```

Finally there are two final functions in the steganography portion of the project those are the `encode` and `decode` functions, the `encode` function takes the input of `uri`, `text`, and `callback` (for the blob) and passes the `uri` into the image loader to be put on the canvas and assigned to the value `imageData` this is then passed along with the `text` variable to the `encodeImageData` function this is then passed to the `saveImage` variable and is passed as a blob to the upload page.

```
export function encode(uri, text, callback) {
  return loadImage(uri).then(imageData => {
    const encodedImageData = encodeMessage(imageData, text)
    return saveImage(encodedImageData, callback)
  })
}
```



```
encode(uri,embText,blob => {
```

The decode function is used in order to decode the secret image within the modifiedImage a modifiedImage uri is passed to the decode function here the image is then passed to imageLoader in order for it to be put on a canvas and the information from this is then converted to imageData it is then passed through the decodeMessage function and returned to the user on the DBView page.

```
export function decode(uri) {  
  // const uri = URL.createObjectURL(blob)  
  return loadImage(uri).then(imageData => {  
    return decodeMessage(imageData)  
  })  
}
```

On the side of decoding and displaying the image data this will be possible on the DBView page a const is declared of the name secretText this is used in order to print the revealed message contained within the image. A entry const is also declared = props this is done in tandem with the redux state passing the firestore storage as an array object containing the downloadURL of the image stored on firebase. When the decode function is called it pass the downloadURL that is declared under the return statement it is found in the Flatlist that is used for displaying the images the items are rendered and in the <Image> there is a source assigned for the downloadURL. A const blob is declared to assign the incoming raw png data to and then passes that information into a file reader passing the result into the let dataUrl this is then passed into the decode function and the resulting text is assigned to the secretMessage state. The secretMessage is displayed with value.

```
const [secretText, setText] = useState('');  
const {entry} = props;  
console.log({entry})  
  
const onDecode = async (downloadURL) => {  
  const blob = await fetch(downloadURL).then(r => r.blob())  
  let dataUrl = await new Promise(resolve => {  
    let reader = new FileReader();  
    reader.onload = () => resolve(reader.result);  
    reader.readAsDataURL(blob);  
  });  
  decode(dataUrl).then(text => {  
    setText(text)  
    // console.log(secretText)  
  })  
}
```

```

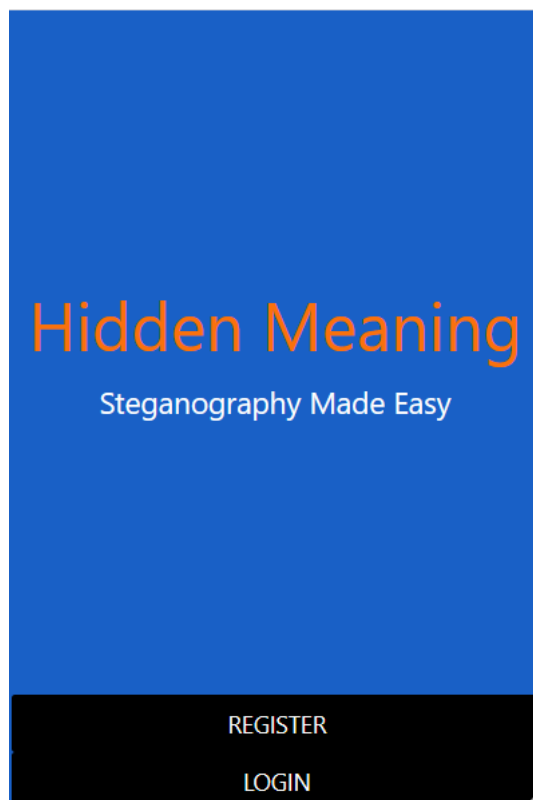
<FlatList
  numColumns={3}
  horizontal={false}
  data={entry}

  renderItem={({item}) => (
    <View style={styles.imageWrapper}>
      <TouchableOpacity onPress={() => onDecode(item.downloadURL)}>
        <Image
          style={styleImage.image}
          source={{uri:item.downloadURL}}
        />
      </TouchableOpacity>
    </View>
  )}
/>

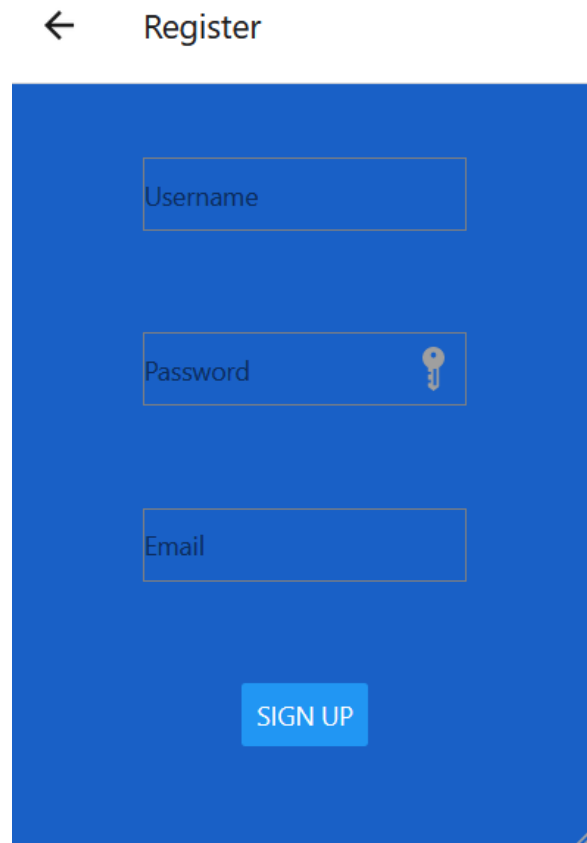
```

2.4 Graphical User Interface (GUI)

The screenshots provided below are all wireframe mockups of what the website could potentially look like when it has been fully developed as I do not have a working demonstration to showcase at present I have opted to showcase wireframes in their place.

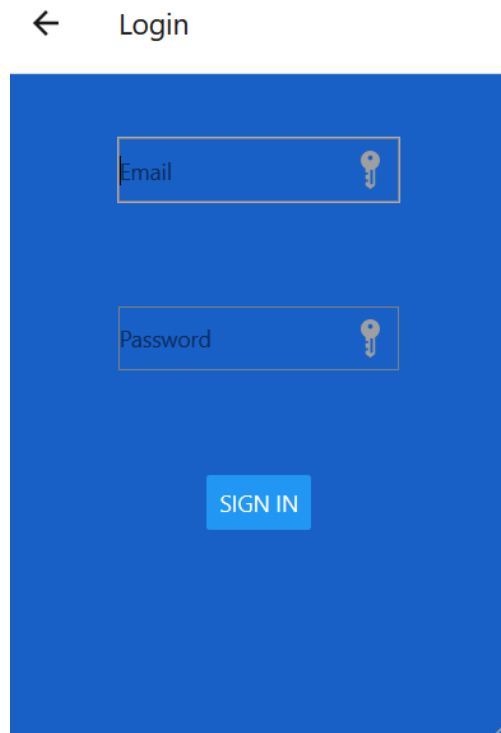


The image shown above is the initial landing page of the application if is intuitively laid out allowing you to select between the login and register pages. It states the name of the project and brief description of what can be achieved.

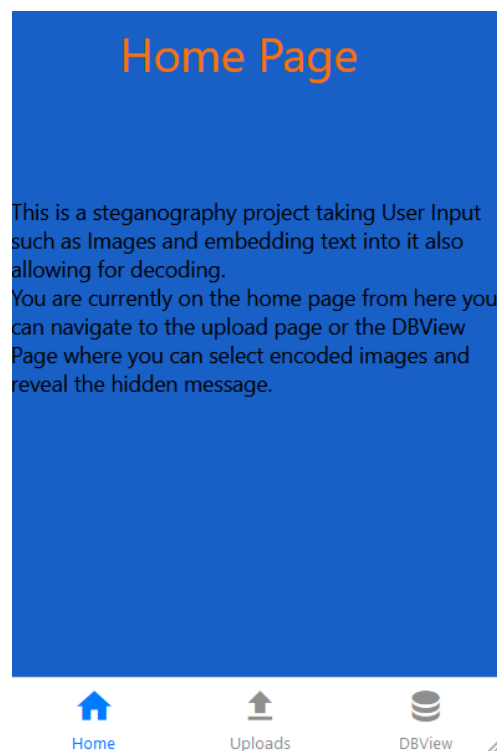


A screenshot of a mobile application's 'Register' screen. At the top, there is a back arrow icon and the title 'Register'. Below this, there is a blue rectangular area containing three input fields: 'Username', 'Password' (with a key icon on the right), and 'Email'. At the bottom of this blue area is a light blue button with the text 'SIGN UP' in white capital letters.

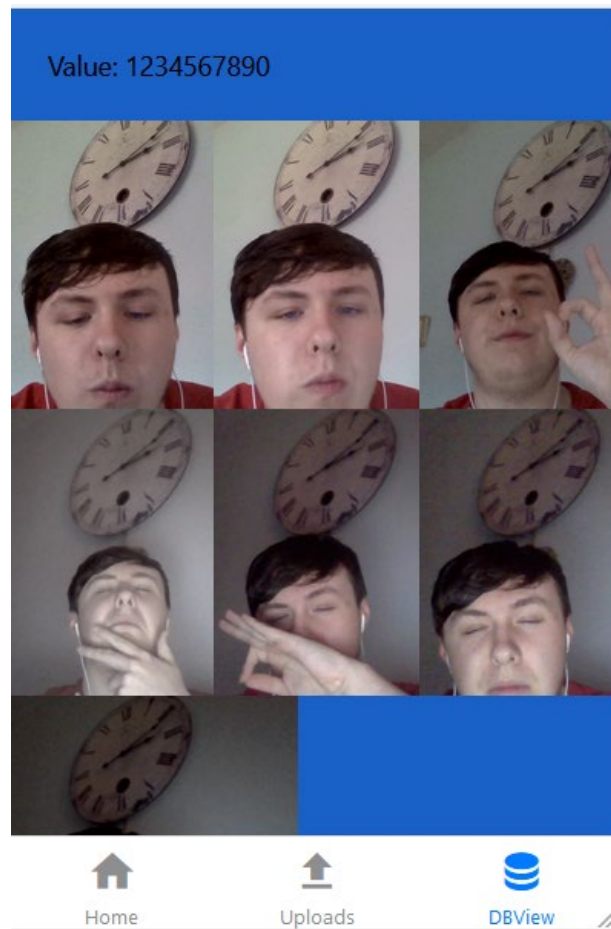
Shown here is the registration page you can get here from the landing page you will be able to enter a username, password and email that is authenticated through the firebase backend upon signup you will be signed into the application.



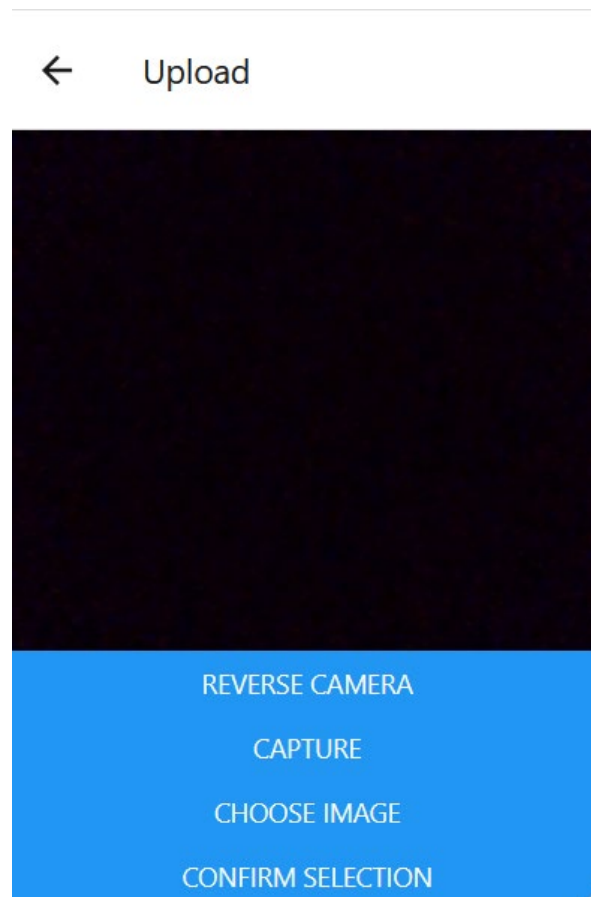
Shown here is the login page you can get here from the landing page you will be able to enter a password and email that is authenticated through the firebase backend upon signin you will be able to access the application.



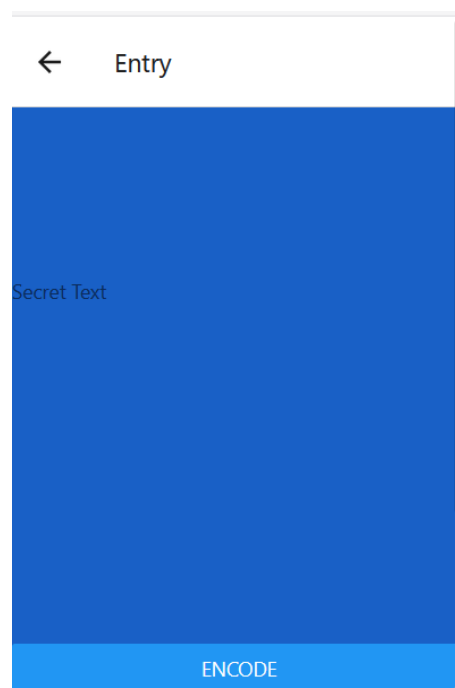
Shown here is the home page it is the initial landing page after signing in it states the pages name and a brief description of the project and what it can do.



Shown here is the DBView page it is the page where you can view all saved items on your account and upon clicking you will be able to see the concealed message in the image.



This is the upload page it is responsible for the capturing or selection of images through the capture or choose image button that will open your camera or image library. After you hit choose selection you will be brought to the entry page.



This is the final page of the application it is the entry page the URI of the image on the previous page is brought here and after you enter your message into the secret text field when you hit encode the message will be sent to the steganography section you will then be able to view it on the DBView page.

2.5 Evaluation

When evaluating this project I think it is extremely important to look back on the initial aims of the project and see have I completed those aims and if I feel satisfied with the overall result of the aims I set out to accomplish. The first aim I in the project was do I feel like I have a greater understanding of steganography and steganographic techniques after the completion of this project and was I happy with the implementation. I feel through the process of this project I have gained a great amount of knowledge in the area of cryptography and image steganography through the creation of this project and the integration of new steganographic techniques I feel positive that I have achieved the overall aim I set out to when I began the project. Through the use of firebase authentication I am happy that the application is safe for users to use knowing that only user specific data is shown to the user who logged in originally it is inaccessible to anyone else using the application ensuring security of user data. Regarding the use of an intuitive user interface and fully working application I am positive that I have fulfilled this use case of this project tying in with the intuitive design fully working application and the secure aspects of this project I feel that the final aim has also been achieved in making a application fully accessible to the average user.

3 Conclusions

Through this project I have outlined what is steganography the various technologies used in this project whether that be Firebase Auth, Firestore, Storage which is responsible for handling the back end of the application to the core of the project which is the steganography itself I feel this project fully fulfils the use cases and initial aims set out and outlined in the beginning of the project it perfectly fulfils the core aspect of steganography changing image data to hide text information and then extracting that information when needed as well as that it featured a intuitive user interface that allows even your average tech user to make full use of all application services with ease. Some of the areas I feel that could be improved upon are in the GUI specifically the design the implementation of animations and gestures might contribute even more to a better user experience in a functional aspect I feel that there is no further improvement to be made in the areas of steganography in the area of user input for the Login and Register pages I feel that greater use of React Native packages could be implemented to allow for better user field validation.

4 Further Development or Research

The only further development I would make to the project is as outlined in the conclusion in relation to the GUI I would implement new packages and techniques to improve user experience.

5 References

Please include references throughout your document where appropriate. See [here](#) for a guide on referencing from the NCI library.

6 Appendices

This section should contain information that is supplementary to the main body of the report.

6.1 Project Plan

1.0 Objectives

For this project I am attempting to implement the technology Steganography to allow me to embed information into the low-end pixels of an image so it will not be distinguishable to the human eye. Using this technology, I intend to create a secure way for users to store sensitive information within an image. I have added proposed using this technology and the additional layer of security so that the information can be embedded within an image that cannot be determined to be an image embedded with any additional image from simply looking at it. I will try to implement that technology using a web app, so it is accessible on both PC and mobile so as not to limit the accessibility of the application.

I would like to continue my research into the topic of steganographic principles and methods of implementation into the end of the first semester first of all, to allow me to gain a deeper understanding of the topic to allow me to apply it to me project better. Starting in the second semester I would like to have a solid base to allow me to begin creating the user interface and algorithm that will allow my newly developed knowledge base to be utilised. At the end of this project I would like to have a web application with a intuitive user interface that allows for the embedding of user inputted data into images using steganographic techniques.

2.0 Background

Image steganography is the embedding of messages or data into images using an algorithm which replaces the low level pixels in an image that are not being used with other pieces of information that does not appear to change the image visually to the human eye. The way that image steganography operates is as follows: both the original image as well as the hidden image that you wish to embed are fed into the encoder that you have generated. There are a variety of different types of steganography:

1. Text Steganography.
2. Image Steganography
3. Video Steganography
4. Audio Steganography
5. Network Steganography

For my project I have decided to use image steganography, I feel as though with an image I will be able to store more information in the unused pixels of the image without making it obvious that there is information embedded within the image. With image steganography there are a variety of ways to store these hidden files within an image they are as follows:

1. Insertion of information into the unused or insignificant bits.
2. Masking and Filtration.
3. Redundant Pattern Encoding.
4. Encrypt and Scatter.
5. Coding and Cosine Transformation.

With my use of image steganography there will be a need to feed both the original image as well as the hidden message into an encoder as I have mentioned previously, This being the case I will need an algorithm to help me to successfully encode this information into the image. For this project I will be using the LSB steganography method which stands for Least Significant Bit Steganography allowing me to replace the unused or insignificant bits of the image with different data.

3.0 Technical Approach

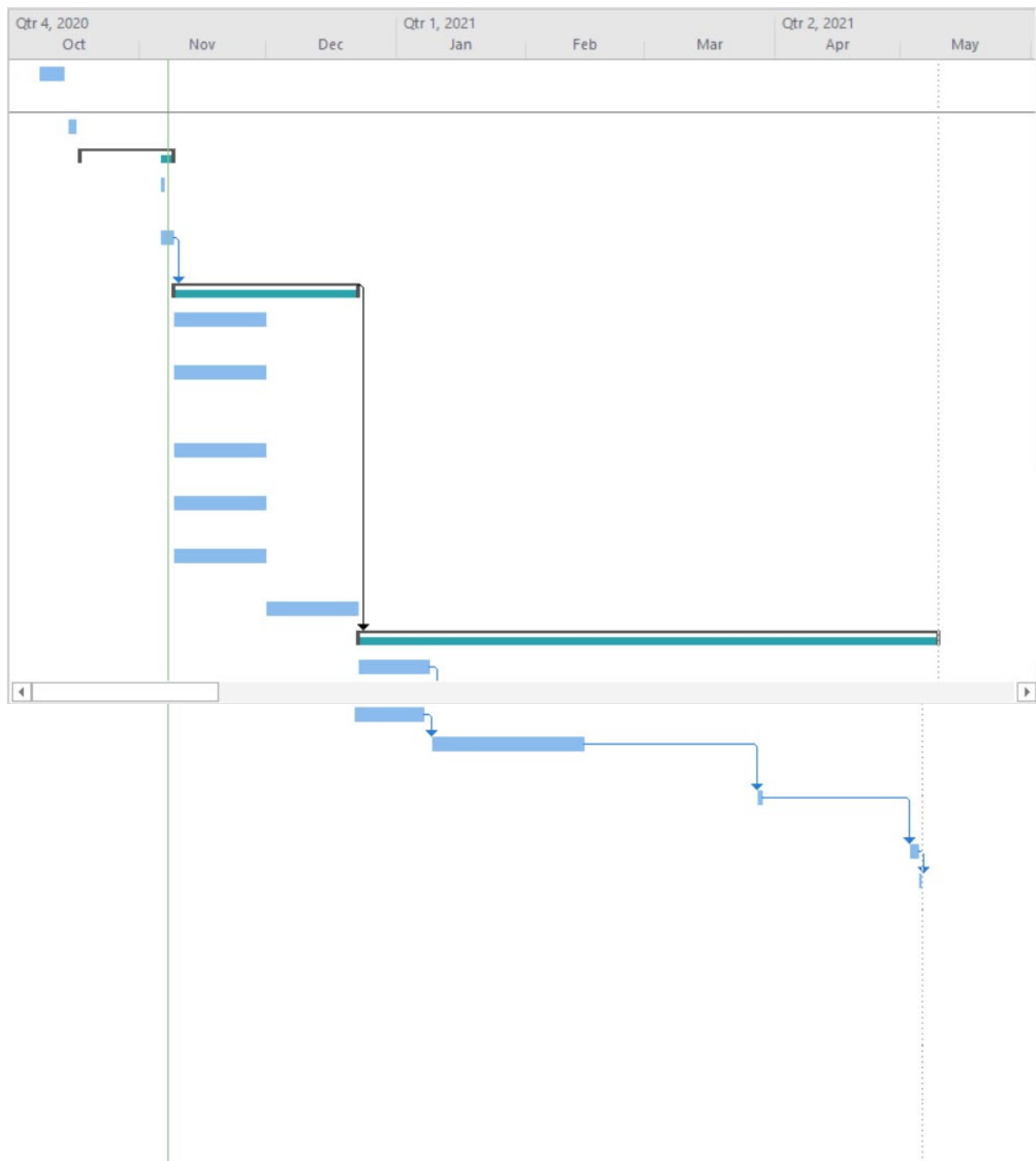
As for how to approach this project I will need to do significant research on the different steganographic techniques that can be used in relation to Image Steganography as well as potentially using a secret key to further encrypt the data so as if the image is ran through the correct decoder it will still require a key to be entered to decrypt the secret information within the image. I have decided on used the LSB method of steganography at present, but I will look further in detail into the other methods as listed above. As well as this I will further investigate the algorithms, I will used to implement these methods that will be used within the encoder.

For the implementation of this image embedder and encoder I have decided to use a web based application utilising Angular8 as the base platform from which I will build my application to allow for a seamless user interface with intuitive design. I have plans to use an external cloud based database in the project which may allow for the hosting of user data or other application functionality, Firebase which utilises JSON and XML may be a good platform to use to fulfil this requirement but I will further investigate appropriate database languages and services to use.

4.0 Special Resources Required

N/A

5.0 Project Plan



		Task Mode ▾	Task Name ▾	Duration ▾	Start ▾	Finish ▾	Predecessor	
GANTT CHART	1			Investigate possible project ideas relating to Cyber Security	5 days	08/10	13/10	
	2			Create a Project Pitch	2 days	15/10	16/10	
	3			⚡ Await Project Approval	17 days?	18/10	08/11	
	4			Speak to Acedemic Supervisor Regarding your Pitch	1 day?	06/11	06/11	
	5			Create Project Proposal and Fill Out Ethics Form	1 day?	06/11	08/11	
	6			⚡ Create a project prototype	32 days?	09/11	22/12	5
	7			Investigate Steganographic Methods and Techniques	16 days?	09/11	30/11	
	8			Investigate How to implement Steganography through Angular 8	16 days?	09/11	30/11	
	9			Investigate possible Cloud Based Databases to Use	16 days?	09/11	30/11	
	10			Investigate Algorithms and how to create an encoder	16 days?	09/11	30/11	
	11			Investigate how to further encrypt embedded messages	16 days?	09/11	30/11	
	12			Create Encoder	16 days	01/12	22/12	
GANTT	13			⚡ Create Final Project	99 days?	23/12	09/05	6
	14			Create a Web App (UI)	13 days?	23/12	08/01	
	15			Integrate Steganographic Encoder into Web App	27 days?	11/01	16/02	14
	16			Integrate Cloud Based Data Base for User Auth, ETC	1 day?	31/03	31/03	15
	17			Test Project	1 day?	07/05	08/05	16
	18			Present Final Project	1 day?	09/05	09/05	17

6.0 Technical Details

I will be using Angular8 and React as the platform from which I will build this project following from that I will implement a cloud-based database for user auth and other features. I will implement this feature using Firebase or another applicable cloud-based database. I will use an encoder that features an algorithm that has to be determined through investigation and research following this I will attempt to add an additional level of security through an encryption key.

7.0 Evaluation

I will evaluate the effectiveness of this application through visual inspection and open source applications that are designed to try and identify images that have been encoded with additional data.

2.1. Ethics Approval Application (only if required)

7.1 Reflective Journals

7.2 Other materials used

Any other reference material used in the project for example evaluation surveys etc.