*Faculty of Natural Science*

*Department of Computer Science*

*Kristianstad University*

*Course: DA376C Software Development for the Cloud*

*Program: TBSEH Bachelor Programme in Software Development*

*Authors: Jonathon Bisiach,* *Kristian Kokoneshi and Martijn de Redelijkheid*

TreasureHunt

Table of Contents

[1. Introduction 4](#_Toc42697231)

[2. Requirements 5](#_Toc42697232)

[2.1 Use Case Overview 6](#_Toc42697233)

[2.1.1 Create account and Encrypt user info 6](#_Toc42697234)

[2.1.2 Log in 6](#_Toc42697235)

[2.1.3 Create hunt and post photos 6](#_Toc42697236)

[2.1.4 Edit settings 7](#_Toc42697237)

[2.1.5 Curate content 7](#_Toc42697238)

[2.1.6 Create account and Encrypt user info 7](#_Toc42697239)

[2.1.7 Log in 7](#_Toc42697240)

[2.1.8 Browse hunts 8](#_Toc42697241)

[2.1.9 Start hunt and post photos 8](#_Toc42697242)

[2.1.10 Edit settings 8](#_Toc42697243)

[2.1.11 Report content 8](#_Toc42697244)

[3. Design and Implementation 8](#_Toc42697245)

[3.1 Database 8](#_Toc42697246)

[3.2 Classes and dependencies 11](#_Toc42697247)

[3.2.1 res/layout 12](#_Toc42697248)

[3.2.2 java/utils 12](#_Toc42697249)

[3.3 Class Interactions and Use Case mappings 13](#_Toc42697250)

[3.3.1 puplic/js 14](#_Toc42697251)

[4. Test Results 14](#_Toc42697252)

[5. Summary and Conclusion 15](#_Toc42697253)

[5.1 Difficulties and challenges 15](#_Toc42697254)

[5.1.1 Overwhelming choice 15](#_Toc42697255)

[5.1.2 Platform inexperience 15](#_Toc42697256)

[5.1.3 Time and conflicting responsibilities 15](#_Toc42697257)

[5.2 Correctness of time estimates 15](#_Toc42697258)

[5.3 Conclusion 15](#_Toc42697259)

# Introduction

TreasureHunt is an interactive application that can be used to create and play interactive tours, competitive games and treasure hunts in any town or city.

Users can use the TreasureHunt website to create their own custom tour routes or treasure hunts, and use the Android application to capture photos or mark GPS coordinates. Hunts created by users can be undertaken by other users.

The hunts work as follows: creators/users use the Android application or the Treasure Hunt location to points on the map that form the route of a hunt. Each location can be tagged with title and some short text, such as a description, and will tagged with a photo of that location.

Users can browse and select created hunts using the Android application. Once a hunt has been selected, users will be shown a photograph of their first hunt location and is it up to them to find it. If needed users can request a hint, upon which the hunt location will be displayed on the map.

Once a location has been correctly identified and travelled to, once the hunter gets within a close proximity of a hunt location, an alert will be triggered by the phone’s GPS and the user will be given the next hunt location, or if there are no more locations, they will be notified that the end of the hunt has been reached.

Hunts may be performed by walking, cycling or driving (as long as done safely). The goal of of TreasureHunt is to promote physical activity and teamwork through the act of participating in hunts, and potentially helping to promote local attractions and/or tourism.

A screenshot of a cell phone

Description automatically generated

# Requirements

Table 1 below lists some of the requirements for this project. The requirements are currently provisional and might be expanded/changed in the future.

**Table 1 - Requirements**

|  |  |
| --- | --- |
| **Req.**  **No** | **Req. Name: Description** |
| 1 | **Mobile application:** Building the mobile application using Android Studio that contains the user interface and connects to the cloud. Estimated time: 80 hours |
| 2 | **Database:** Constructing the database containing user information, set locations and user photos. The database will be constructed in Google’s Cloud Platform using Cloud Firestore to store String data such as user details and hunt information. Within this will be links to files (user photos) stored in Google’s Cloud Platform Storage.  Estimated time: 20 hours |
| 3 | **Website:** Building the frontend of the website for users to log in and create their interactive maps. The web app will be built from JavaScript in Visual Studio Code and hosted from Google Firebase.  Estimated time: 60 hours |
| 4 | **Cloud setup:** Using the selected cloud toolset to process and store information sent from the mobile the application, and provide the backend to the website. To align with the use of the various Google APIs such as Maps, the Google Cloud platform will be used.  Estimated time: 30 hours |
| 5 | **Account creation and login:** Implementing account creation and login in the interface and database. Users will be able to create accounts and login from both the mobile application and web app. User access will be available in two tiers, regular patron and administrator, with the later being able to curate inappropriate content.  Estimated time: 10 hours |
| 6 | **GPS utilization:** using the GPS in combination with a map provider for the app and website. This will be implemented in both the mobile application and web app.  Estimated time: 20 hours |
| 7 | **Security:** the app should protect the information of the user, including their current GPS location. When an account is created, the user’s information will be encrypted in the cloud and we will research the best way to sanitise user inputs.  Estimated time: 10 hours |

## Use Case Overview

A close up of a map

Description automatically generated

**Figure 1: Use case diagram of the Treasure Hunt web app**

### Create account and Encrypt user info

Users will be able to create an account using the web app, as per Requirement 1. Once a user creates an account, the information will be stored and encrypted in the database as per Requirement 7. Cloud Firestore automatically encrypts all data before it is written to disk. The data is automatically and transparently decrypted when read by an authorised user.

### Log in

Registered users will be able to login to their accounts, with login details being verified using information from the database as per Requirement 5. Session management is handled automatically by Firebase.

### Create hunt and post photos

Registered users can create hunts in the web app using the Google Maps API to select particular locations, name them and write brief descriptions, which are then stored on the Firestore database. This database will contain a reference to the photo that will be attached to their corresponding location for hunters to find (as per Requirement 3), which will be stored in Google’s Cloud Platform Storage.

### Edit settings

Registered users can edit and update the settings entered during their account creations. These are then sent to the database.

### Curate content

Administrators have the ability to remove any inappropriate content posted by users using a search and delete table operations contained in the worker role.

A picture containing text, map

Description automatically generated

**Figure 1: Use case diagram of the Treasure Hunt Android application**

### Create account and Encrypt user info

Patrons will be able to create an account using the mobile app, as per Requirement 1. Once a user creates an account, the information will be encrypted in the database as per Requirement 7. Cloud Firestore automatically encrypts all data before it is written to disk. The data is automatically and transparently decrypted when read by an authorized user.

### Log in

Registered users will be able to login to their accounts, with login details being verified in the app with information from the database as per Requirement 5.

### Browse hunts

Once logged in, users can browse hunts created by users as per Use Case 2.1.3. The app will select the nearest posted hunts in the database using the user’s geographical position as per Requirement 6. The selected hunts will display the hunt title, a short description of the hunt and the number of locations in the hunt.

### Start hunt and post photos

Once a Treasure Hunt is selected, the user can participate in the Hunt by looking at posted photos and trying to find their locations in the environment. The app will retrieve hunt details from the database once a hunt is selected. When a user on a hunt comes within certain proximity of a location on a hunt, this will be recognised by the phone’s GPS and an alert will be sent by the app, letting the user know they are close to their location and giving them the option to upload a photo, which will then be sent to the database.

### Edit settings

Registered users can edit and update the settings entered during their account creations. These are then sent to the database by the application.

### Report content

Users have the ability to report any inappropriate content posted by other users. This is done by submitting a form on the app, when is sent to the database, which can then be investigated by an administrator.

# Design and Implementation

## Database

Initially our plan was for the implementation of a cloud database using Google BigTable due to its flexibility. However, as latency is not of paramount importance in the application and as there is not a large amount of data to be processed, using BigTable seemed to be overkill in this instance.

Instead, our cloud database is hosted on Google Firestore in combination with Cloud Platform Storage. This decision was made taking into account the following considerations:

* Familiarity with the Firestore platform from previously using Firebase
* Easy scalability of the platform
* In-built security with automatic data encryption and decryption
* Support for indexed queries
* Reduced cost when compared with BigTable (and free trial use)

A close up of a map

Description automatically generated

**Figure 3: Conceptual entity-relational diagram of the database implemented in SQL.**

The Firestore database contains a hierarchical folder structure that contains user information, login details and the details of their hunts.

A screenshot of a social media post

Description automatically generated

**Figure 4: Firestore database showing user details.**

**A screenshot of a computer screen

Description automatically generated**

**Figure 5: FireStore database showing details of stored locations.**

Each hunt has at least one location. For each hunt location, several details are stored including a link to a photo stored in the Google Cloud Store.

A screenshot of a computer

Description automatically generated

**Figure 5: FireStore database showing details of stored locations.**

When a user takes a photo using the mobile app, it is sent to the Google cloud store with a reference to it being stored as part of the location in the database.

A screenshot of a computer screen

Description automatically generated

**Figure 6: Location photos stored in Google Cloud Storage.**

## Classes and dependencies

The TreasureHunt application in Android has a vast number of classes and dependencies required to make the application function. In the interests of levity, only some will be mentioned in this report.

A picture containing drawing

Description automatically generated

**Figure 7: A class diagram showing all dependencies in the TreasureHunt Android application.**

### 3.2.1 res/layout

The user interface of the Android application is driven by fragments, with each fragment representing a different component within the interface

A screenshot of a cell phone

Description automatically generated

**Figure 8: Layout fragments in the TreasureHunt Android application.**

### 3.2.2 java/utils

A database handler utility class contains a Singleton class that is used to initiate all database call. Each fragment that makes database calls does so separately as needed to reduce latency application latency.

A screenshot of a social media post

Description automatically generated

**Figure 9: The database handler class in the TreasureHunt Android app.**

**3.3 Class Interactions and Use Case mappings**

The TreasureHunt web application is hosted using Google Firebase and was constructed in Atom and Visual Studio Code using basic HTML, CSS and JavaScript.

A close up of a map

Description automatically generated

**Figure 10: A class diagram showing dependencies in the TreasureHunt web application.**

### 3.3.1 puplic/js

All calls to the database are made using a single class in JavaScript. Once configured, database initialisation and login are all handled by Firebase. Methods for registering a new user, storing a new hunt and editing a hunt is all contained in this class.

# Test Results

Table 2 below contains the current status of implemented and tested requirements.

Table 2 - Test Results

|  |  |  |
| --- | --- | --- |
| **Req.**  **No** | **Req. Name** | **Test Result** |
| 1 | Mobile application | PASSED |
| 2 | Database | PASSED |
| 3 | Website | PASSED |
| 4 | Cloud setup | PASSED |
| 5 | Account creation and setup | PASSED |
| 6 | GPS utilization | PASSED |
| 7 | Security | PASSED |

# 5. Summary and Conclusion

This chapter contains a summary and conclusion of the work that was carried out in this project as well as reflections and thoughts about working methods and challenges.

## 5.1 Difficulties and challenges

Below is a list of notable challenges that came up during this project and that took a long time to solve.

### 5.1.1 Overwhelming choice

As we were relatively inexperienced with the cloud platform, the overwhelming number of options in the Google Cloud platform let alone options outside this one vendor was quite daunting.

**5.1.2 Platform inexperience**

Additionally, due to our inexperience with the cloud platform, initial decisions regarding what technologies to use for the database and web hosting changed part-way into development. Specifically, BigTable was substituted with the combination of Firestore and Cloud Storage, and GitHub Web was substituted with Firestore Host.

**5.1.3 Time and conflicting responsibilities**

As this project ran parallel to other responsibilities, namely the degree thesis project, the amount of time we could reasonably devote to the project was more limited than was ideal.

## 5.2 Correctness of time estimates

The estimates of time taken on all aspects of the project were reasonably accurate, if not somewhat overestimated:

* The web application was the most complicated aspect of the project and therefore took the most time.
* The database and the cloud setup, once the decision on which technology to use, was relatively smooth. The hierarchical folder structure of Firestore was easy to understand and he had used Firebase in a previous project.
* Once the website hosting decision had been settled on, the development of the website was also relatively smooth. The Firestore Hosting service proved easy to use with our existing knowledge and toolset.

## 5.3 Conclusion

This project proved to be an extremely interesting experience. On its surface, development for the cloud seems to be rather intimidating, with so many platforms, options, programming languages and storage to consider. However, once the appropriate development decisions were made regarding platforms and such, we found cloud development to be extremely adaptable to our existing programming skills. As such, we came the conclusion that cloud develop is as difficult as one wants to make it.