*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*

Treasure Hunt

Table of Contents

[1. Introduction 3](#_Toc39149906)

[2. Requirements 4](#_Toc39149907)

[2.1 Use Case Overview 5](#_Toc39149908)

[2.1.1 Create account and Encrypt user info 5](#_Toc39149909)

[2.1.2 Log in 5](#_Toc39149910)

[2.1.3 Create hunt and post photos 5](#_Toc39149911)

[2.1.4 Edit settings 5](#_Toc39149912)

[2.1.5 Curate content 5](#_Toc39149913)

[2.1.6 Create account and Encrypt user info 6](#_Toc39149914)

[2.1.7 Log in 6](#_Toc39149915)

[2.1.8 Browse hunts 6](#_Toc39149916)

[2.1.9 Start hunt and post photos 6](#_Toc39149917)

[2.1.10 Edit settings 7](#_Toc39149918)

[2.1.11 Report content 7](#_Toc39149919)

[3. Design and Implementation 7](#_Toc39149920)

[3.1 Database 7](#_Toc39149921)

# Introduction

Treasure Hunt 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 website to create their own custom travel routes or treasure hunts, and use the Android application to capture photos or mark GPS coordinates. Hunts made by creators can be played by other users.

The hunts work like the following: creators/users take photos at specific locations using their phone that can then be used in hunts. These photos are stored, as well as the GPS location of where the photos was taken. Creators can then use the website to see any photos of locations in the area they selected. They can choose a selection of these to set up a hunt. These hunts may span larger distances, or perhaps be confined to smaller locations: i.e. a park.

Users can select a hunt on the app. The app will then give some options on how to do the hunt. *It might be possible to select whether locations should be shown in sequence or all at once, as well as possible multiplayer options.* Then the hunt starts. The user gets to see the photo linked to a GPS location and has to try to find that place. Then the user needs to go there and take a selfie with the location clearly visible. *This photo can then be used for a possible social part of the app, proving that they have completed the hunt for that location.* A hunt can consist of one or multiple locations. Hunts may be performed by walking, cycling, or driving, as long as done safely.

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. At this stage the database will be constructed in Google’s Cloud Platform using Cloud Firestore to store String data such as user details and hunt information. In this  Estimated time: 20 hours |
| 3 | **Website:** Building the frontend of the website for users to log in and create their interactive maps. At this stage the web app will be built using Visual Studio for the Google Cloud Platform using the Cloud Tools for Visual Studio extension.  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

Patrons will be able to create an account using the web app, which will have a web role and worker role which will connect to the database, 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. There is no setup or configuration required and no need to modify the way you access the service. 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 worker role using information from the database as per Requirement 5.

### Create hunt and post photos

Registered users can create Treasure Hunts using the Google Maps API to select particular locations and post pictures of the corresponding locations for hunters to find as per Requirement 3. The hunt and the photos are then stored

### Edit settings

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

### 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.

### 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.

### 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 data once a hunt is selected. Once the user finds the selected location, which will or won’t be verified by the app depending on their location, they can post their own photo of the location to prove they have been there. 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

A close up of a map

Description automatically generated

**Figure 3: The entity-relational diagram of the database to be implemented in BigTable.**

## Classes

<Description of classes. List one class per sub chapter and add some class diagrams to illustrate relations (inheritance and/or associations) between the main classes. The UML does not need to be extremely detailed, but the most important attributes and methods shall be shown.>

3.1.1 <Name of Class>

< Description of this class, including UML. >

3.1.2 <Name of Class>

…

3.2 Class Interactions and Use Case mappings

<Instructions: the sub chapters here shall correspond to the use cases in chapter 2, and each use case shall contain a UML sequence diagram of the classes that are involved in that use case, and how they interact to implement the use case, including method calls. >

3.2.1 <Name of Use Case>

<A sequence diagram of the classes involved in this use case, and how they interact. You may write some explaining text here, and/or you may use notes in the diagram itself.>

3.3 Database

<Show your database design with ER diagram(s). >

# Test Results

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

<Instructions: This table shall map 1-1 to the table in Chapter 2. The test result for each requirement shall be one of the following: NOT IMPLEMENTED, PASSED or FAILED.>

Table 2 - Test Results

Req.

No Req. Name Test Result

1 <Requirement 1 name> <NOT IMPLEMENTED/PASSED/FAILED>

2 <Requirement 2 name> …

.. .. ..

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 Weekly Progress

Below is a short summary of what was done each week.

5.1.1 Week 1

<Instructions: Describe what you did this week. You can see it as a developer’s weekly diary. Try to answer the following questions: What did you do this week? Did you meet any challenges? What was difficult? Did you get stuck with something? What went well and what went bad? What have you learned during this week?>

5.1.2 Week 2

TODO

5.1.3 Week 3

TODO.

5.1.4 Week 4

TODO.

5.1.5 Week 5

TODO.

5.1.6 Week 6

TODO.

5.1.7 Week 7

TODO.

5.2 Difficulties and challenges

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

5.2.1 <Name of Challenge/Difficulty 1>

<Instructions: List the most difficult tasks in this project and describe why they were difficult. Did you learn something, e.g. how to handle very difficult programming problems?>

5.3 Correctness of time estimates

<Instructions: Look back on your time estimates and discuss your results. How accurate were they? What have you learned about time estimates and how can you get better in next project?>

5.4 Priority decisions

<Instructions: Look back on your feature priority settings. Did you prioritize the right features? Did you succeed to deliver the highest prioritized features? Have you learned anything about setting priorities>

5.5 Conclusion

<Instructions: Look back on the whole project. Here you can write a bit more freely about your thoughts on this project. What was your overall experience? How was the teamwork? What did you learn? Can you list some points that you will do better in next project? Other thoughts. >