Software Requirements Specification

Golf Course Mapper

Team Recursive Recursion

September 21, 2018

Contents

1 Overview

1.1 Purpose

This Software Requirements Specification (henceforth SRS) is intended to be a guide on the path of developing the Golf Course Mapper system. It will aid the developers by providing a layout of the project requirements.

1.2 Scope

Golf players require an application that allows them to view golf courses during play. Although there are existing platforms that aim to achieve this goal, many of them are simply mappers and do not allow the player to view their location on the map.

Furthermore, the proprietary and commercial nature of existing products make them difficult to be used widely and therefore contain only a handful of mapped courses. The Golf Course Mapper will attempt to achieve these goals by providing a rich set of features for both the course managers and players through an open source platform.

The Golf Course Mapper will allow golf course managers to draw the shape of their courses using a web application. The owners will map out the details of the golf course. This includes mapping the layout of the fairway, green, sand and water hazards and details such as the position of the hole and the teeing grounds. Additionally, owners will be allowed to view their courses and see the locations of active players currently using the mobile application.

Players currently on the course will be able to use the provided mobile application to view the map of the course and where they are on it currently. This has the benefit of allowing players to plan their shots more strategically. The app will show the distances between the player and various points of interest (such as the hole), further aiding the player during play.

The Golf Course Mapper is not a coaching tool and does not provide a channel of communication between the mapper and the player. The web application is not to be used by coaches, but solely by managers that wish to map out the details of their courses and view its current state. The Golf Course Mapper is also not a social media platform for golfers and will not allow communication among players.

1.3 Definitions, Acronyms & Abbreviations

- System: Henceforth used to refer to the Golf Course Mapper system as a whole.
- Manager: The manager or owner of a golf course, henceforth used to refer to the user of the website.
- Player: A golf player, henceforth used to refer to the users of the mobile application.

- Website: A web application that is used by golf course managers to draw the layout of their golf courses.
- Mobile App: A native Android mobile app that golf players can use to view the golf course.
- Location: A position on the map, modelled using latitude and longitude coordinates.
- Point: A point of interest, such as the hole or tee location. Points are represented as a single location on the golf course.
- Area: Specific areas on the golf course, such as the green or the fairway. Areas consist of an ordered set of locations, forming a polygon.
- DBMS: Database Management System.
- API: Application Programming Interface.
- Back End: The server side that consists of both the DBMS and the API.

1.4 Domain Model

Refer to figure 1, which represents the domain model of the system. At the core of the domain model is the *Course* which represents a single area where *Players* can go to play the game of golf. Each Player is associated with a single Course on which the Player is currently playing. Each Course has an assigned *Manager* that is solely responsible for mapping and managing the information of the Course. Note that a Manager can be assigned to more than one Course.

Each Course has one or more *Holes* associated with it. A Hole represents a single playable map with different map *Elements*, containing information. The Elements can be of two different types: *Point* Elements and *Area* Elements. A Point Element represents single-point information of a Hole such as the teeing grounds, location of the hole and other points of interest. Area Elements represent area information of a Hole, such as the area of the rough, fairway, green and water and sand hazards. Areas are represented as polygons.

A Course may also optionally have Elements that are not associated with a specific Hole. This allows the specification of Elements such as points of interest or hazardous areas such as bodies of water that may be important to more than one Hole. It therefore does not make sense to associate these Elements with a specific Hole, but rather with the Course as a whole.

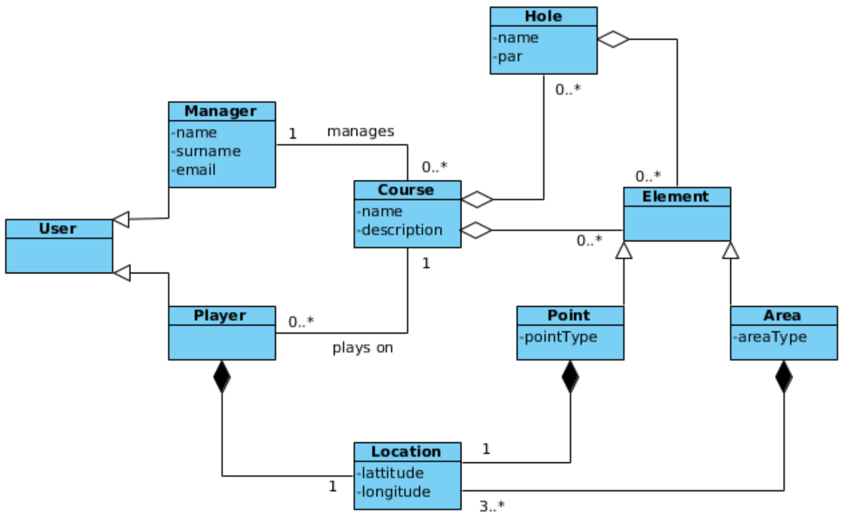

Figure 1: Domain model of the system.

2 Functional Requirements

2.1 Users

The system contains two different users, listed below. Further details about functions that each user will perform are stipulated in Section 2.3.

- **Manager:** These users are golf course managers or owners that use the web application. They primarily create or edit golf courses and view the positions of active players on the map.
- **Player:** These users are golf players that use the mobile application, primarily to view the course. These users are anonymous and do not require any authentication to access the application.

2.2 Subsystems

The system is composed of three subsystems, listed below. Further details about functions that each subsystem will perform are stipulated in Section 2.3. The component diagram in figure 2 illustrates the subsystems and their relationships.

- **Mobile Application:** This subsystem comprises of a native Android mobile application that is used to view mapped courses and other relative information. The Google Maps API will be used for displaying the map received through the Mapper API, described below.

- **Web Application:** This subsystem comprises of in interactive web application for creating, mapping and editing courses, as well as viewing the location of currently active players. Again the Google Maps API will be used for drawing the map elements which are loaded and saved using the Mapper API.
- **Mapper API:** This subsystem covers the backend of the system. This subsystem further contains the following two subsystems:

  DBMS: A PostgreSQL installation with the PostGIS extension, used to store all information relevant information.

  API: A REST API using the Microsoft .NET Core Entity Framework, which will grant simple and secure access to the information stored in the database.

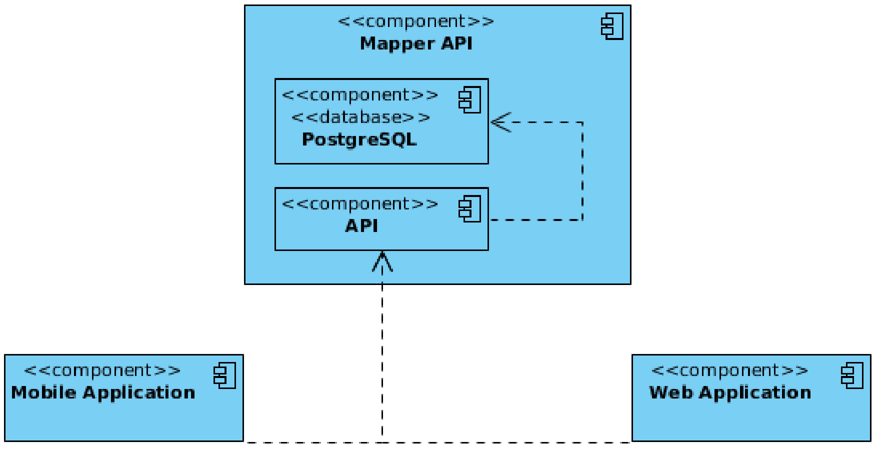

Figure 2: Component diagram illustrating subsystems.

2.3 Specific Requirements

The following functional requirements describes the functionality that the system will provide. It is drawn from user stories that are also illustrated by the use case diagram in figure 3.

- **R1** The manager will register to the website.
- **R2** The registered manager will log into the Website.
- **R3** The manager will manage courses.

  R3.1 The manager will create new courses, given a name and a description.

R3.2 The manager will remove existing courses.

R3.3 The manager will select an existing course to edit.

- **R4** The manager will control the holes on the course.

  R4.1 The manager will add new holes to the course.

  R4.2 The manager will remove existing holes from the course.

  R4.3 The manager will select an existing hole to edit.

- **R5** The manager will add areas to the active hole/course.

  R5.1 The manager will choose to add a new area.

  R5.2 The manager will draw a polygon on the map.

  R5.3 The manager will select the type of the area.

- **R6** The manager will add points to the active hole/course.

  R6.1 The manager will choose to add a new point.

  R6.2 The manager will place a point on the map.

  R6.3 The manager will select the type of the point and optionally enter additional info.

- **R7** The manager will edit existing elements (areas and points) on the active hole/course.

  R7.1 The manager will edit a selected element by dragging the component(s) of the element.

  R7.2 The manager will remove a selected element.

- **R8** The manager will enable the location view, which will display the locations of all current players (using the mobile app) on the course.

- **R9** The player will view a course on the mobile app.

  R9.1 The courses will be listed in order of ascending distance from the player.

  R9.2 The player will choose a course to view on the map.

  R9.3 The player will choose a specific hole to view on the selected course.

- **R10** The player will see his current position displayed on the map.

- **R11** The player will see the distance between their current location and a selected point on the course.

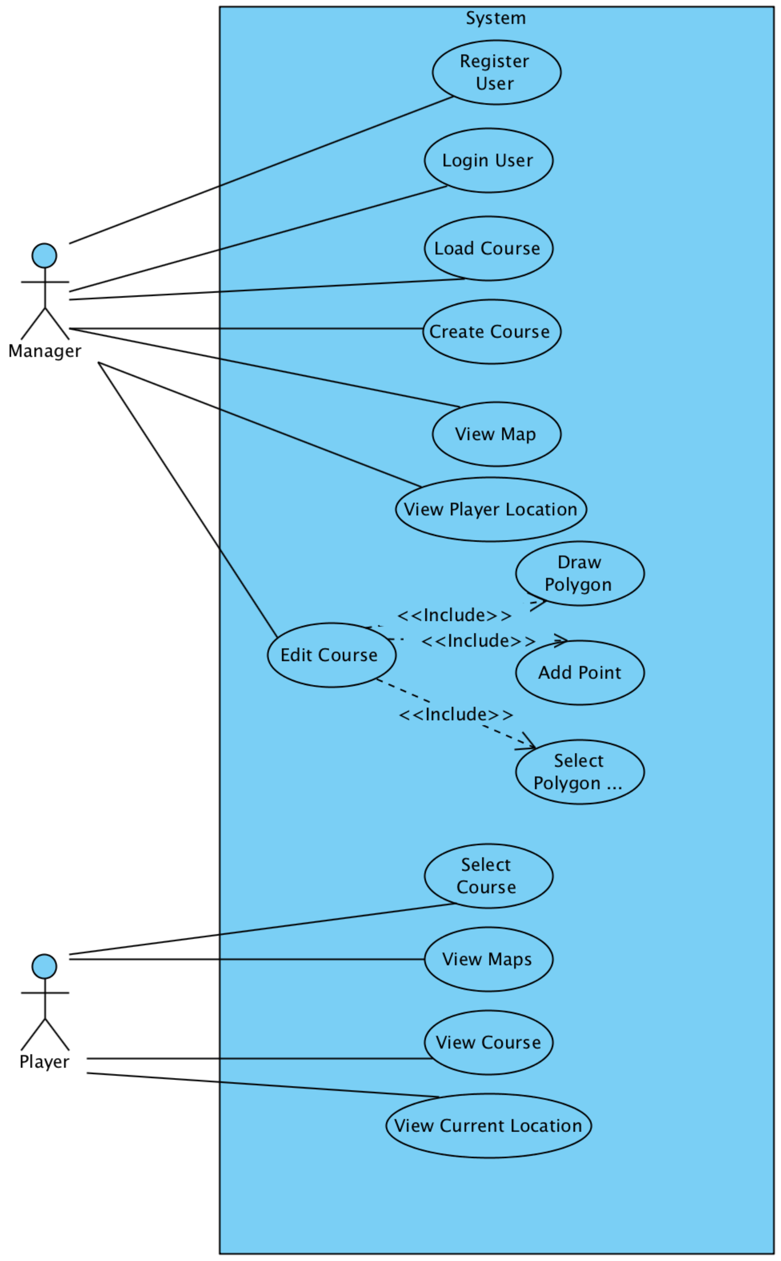

Figure 3: Use case diagram illustrating user stories.

3 Non-Functional Requirements

The following sections each describe different non-functional requirements and briefly explain how these requirements are tested.

3.1 Performance Requirements

The system must be optimized to make as few Google Maps API calls as possible. The Google Maps API allows 25000 calls per day, after which the calls are billed.

The performance is testing using stress testing, as layed out in the *Testing Policy*.

3.2 Usability Requirements

Both the mobile application and web application must be made as user friendly as possible. The users of the system will mostly have little to no technical background, therefore the system must be intuitive to use.

A tutorial video must also be provided to ensure that users can have a brief introduction to the system.

Usability is tested through usability tests, as explained in the *Testing Policy*. Briefly, this is done by presenting the web application or mobile application to different users, having them use the system and providing feedback at the end of the session.

4 Architecture

4.1 Interfaces

The next sections each describe the interfaces present in each specific subsystem.

4.1.1 Mobile Application

- *User:* The mobile application interfaces with the user through the mobile device's screen. The user's input is used to navigate the map and choose holes to view. The user is also presented with a map displaying the course and additional info.
- *Network:* The application interfaces with the networking devices of the mobile device. Network connection is required to use the both the Mapper and Google Maps APIs.
- *Mapper API:* The Mapper API is used to receive information on mapped courses, which is then displayed to the user.
- *Google Maps API:* In order to display the information obtained from the Mapper API, the Google Maps API is used. The API allows the map to be displayed and displays the polygons and points on the map. The user's location is also obtained through the same API.

4.1.2 Web Application

- *User:* The web application interfaces with the user through the browser's display. The user is presented with various controls and UI elements that are used to edit courses. The user is also presented with a satellite map for displaying and drawing courses, as well as active player's locations.
- *Mapper API:* The Mapper API is used to receive information on mapped courses, which is then displayed to the user. The mapper API is also used to create, modify or delete existing courses, holes or map elements.
- *Google Maps API:* In order to display the information obtained from the Mapper API, the Google Maps API is used. The API allows the satellite map to be displayed and displays the polygons and points on the map. The API is also used to draw the elements on the map.

4.1.3 Mapper API

- *API/DBMS:* The API and the DBMS interface with one another through the server's internal network. The API connects to the DBMS in order to execute queries to create, receive, update or delete data.
- *Network:* The API interfaces with the networking devices of the server. Network connection is required in order to communicate with the views, as explained in Section 4.2.1.

4.2 Styles

4.2.1 System Architecture

The system is designed according to the MVC (Model-View-Controller) architectural pattern. This pattern is a good choice for the system as there is a single database responsible for storing information regarding mapped courses and there are multiple views that users interface with.

The MVC pattern allows the abstraction of the model from the views, and also protects access to the model by the use of a controller. The abstraction created by the controller also increases the ease of maintaining the connection between the views and the model.

Figure **??** shows the deployment diagram of the system and its subsystems. The different components of the MVC pattern are then listed as the following:

- **Model** - The DBMS serves as the model component by storing the information required by the system. The DBMS is solely and exclusively responsible for the storage and retrieval of data, as well as enforcing integrity rules on the dataset.
- **Views** - The mobile app and website both serve as two different views. The website is a view that uses the controller to both create, update and destroy data stored within the model in order to manage the mapping of courses. The mobile app is a simpler view with the sole purpose of fetching and displaying data received from the model via the controller.
- **Controller** - The API serves as the controller. It acts as a level of abstraction between the model and the views by controlling the access to the model.

4.2.2 Mapper API Architecture

The Mapper API is designed as a persistence framework architecture. This architecture is a good choice for the API as it allows the hiding of the specific database implementation from the business objects.

This also allows the separation of the database access from the actual implementation of the database, which enables easy maintenance, replacement and even distribution of the database. The different compoments of the persistence framework are listed as the following:

- **Database** - The DBMS serves as the database which can store business objects (in this case, courses and their elements). The database is solely responsible for persisting objects.
- **Database Manager** - The API serves as the database manager and controller, which abstracts the implementation of the database by providing a fixed set of functions that can be performed with persisted objects.

4.3 Configuration

The Web Application and Mapper API will both be hosted on a Microsoft Azure Web Server. This includes the PostgreSQL DBMS, .NET Core Entity Framework and the Angular web service.

The mobile app will be available on the Google Play Store for download on any mobile device that hosts an Android system above version 4.4 (KitKat). Finally, the web application will be available through a public URL that can be accessed by any course managers that wish to use the web application.

The deployment diagram in figure 4 illustrates the configuration of the system.

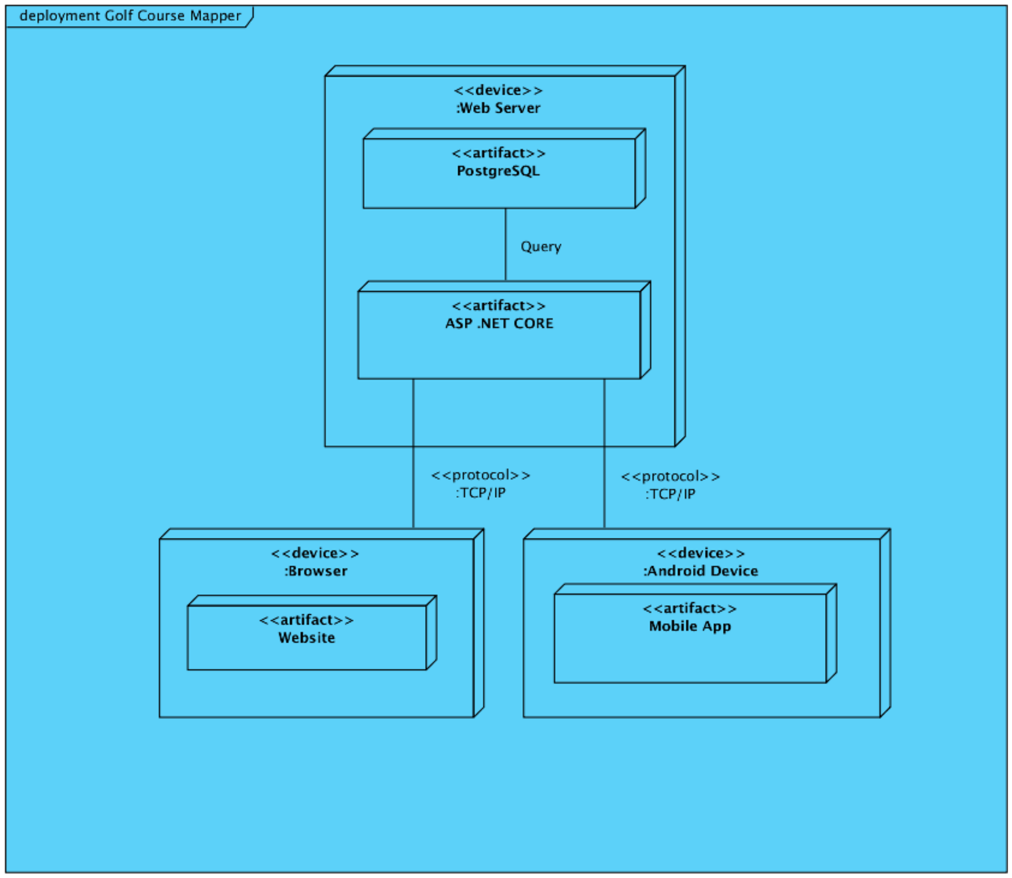

Figure 4: Deployment diagram illustrating system configuration.