

## School of Computing Science and Digital Media

### Coursework Assignment

Surname	Quinn
First name	Alistair
Matriculation Number	1701183
Contact phone number	07547233274
Course + Year	Computing: Application Software Development Year 4
Module Co-ordinator	Kit Ying Hui
Module Number + Name	CM4108/CMM529 Cloud Computing
Coursework Title	Find-My-Friend Web App
Coursework Part	Part 1 of 1
Due Date	30 November 2018 (16:00)
Feedback Due Date	31 December 2018

The University operates a Fit to Sit Policy which means that if you undertake an assessment then you are declaring yourself well enough to do so. Further details are available at:  
[www.rgu.ac.uk/academicregulationsstudentforms](http://www.rgu.ac.uk/academicregulationsstudentforms)

**Declaration** \*\* This **must** be affirmed by adding your name below with the date of submission

**I acknowledge that by submitting the work, accompanied by this front cover, I take responsibility for the ownership of the submitted work.**

**I confirm:**

- that the work undertaken for this assignment is entirely my own and that I have not made use of any unauthorised assistance
- that the sources of all reference material has been properly acknowledged.

Student Signature	Alistair Quinn
Date Submitted	2/11/2018

#### Marker's Comments

<b>Marker</b>	<b>Grade</b>

**\*\*see over for regulations on plagiarism**

**\*\*** An extract from the University Regulations

**6. Academic Misconduct**

Refer also to Schedule 3.3 of this Regulation for guidance on this procedure.

**6.1 Academic Misconduct** is defined as any attempt by students to gain an unfair advantage in assessments and examinations. Examples of academic misconduct include plagiarism, cheating, falsifying data, collusion, bribery or attempted bribery, personation or any other activity intended to provide an unfair advantage.

- (i) **Plagiarism** is the practice of presenting the thoughts or writings of another or others as original, without acknowledgement of their source(s). All material used to support a piece of work should be carefully referenced and should not normally be copied directly unless as an acknowledged quote. Text translated into the words of the individual student should in all cases acknowledge the source.
- (ii) **Cheating** includes:
  - the taking of any unauthorised material into an examination;
  - obtaining copy of “unseen” papers in advance of an examination;
  - communicating or attempting to communicate in any way with another student during an examination;
  - copying or attempting to copy from another student during an examination or in the production of coursework;
  - wilful deception in any element of an examination or assessment.
- (iii) **Falsification of data** consists of the misrepresentation of the results of experimental work or the presentation of results from fictitious work.
- (iv) **Collusion** is the representation of unauthorised group work as that of an individual student.
- (v) **Bribery** is the paying, offering or attempted exchange of an inducement for information or material intended to advantage the recipient in an examination or assessment.
- (vi) **Personation** consists of a substitute taking the place of a student in an examination.

**A student who aids and abets a fellow student to commit academic misconduct shall be deemed to have committed academic misconduct and will be dealt with accordingly.**

**CM4108 Cloud Computing Coursework**

**Find-My-Friends Web App**

**Design Report**

**Alistair Quinn**

Moodle Submission

## Intro

This report details the design of my Find-My-Friends Web App created for CM4108 Cloud Computing Course. It contains a statement of compliance stating how all of the requirements were met, designs of both the backend RESTful API and of the front-end HTML/CSS/JS. The report finishes with screenshots and an evaluation of the design.

## Scenario

The coursework task was to create a Find-My-Friends web app, the app would let you see your friends last checked in locations on a map. The app had to let you check in your own location on a map using a drag and drop maker interface. The app had to let you make subscription requests to friends which they could then accept or deny. The app would then show last checked in locations of subscribed to friends using the map interface.

The app backend had to be designed using the JAX-RS RESTful Java API specification, this was done using an implementation of JAX-RS called Jersey. The app had to store data using AWS DynamoDB and a web interface had to be designed for users to interact with.

For the front-end web interface Leaflet.js Map API was used to provide the check in and friend location functionality. Bootstrap CSS framework was used to provide pop-up dialogs and to create a flexible desktop/mobile friendly layout. jQuery was used to provide interactivity in the web interface.

## Contents

Statement of Compliance .....	6
Functional Requirements .....	6
Non-Functional Requirements .....	6
Design .....	7
API .....	7
Specifications .....	7
Models .....	9
Resources .....	12
Front End .....	14
HTML .....	14
CSS .....	18
JS .....	19
Screenshots .....	26
Login .....	26
Profile .....	26
Subscription Management .....	26
Location Management .....	27
Evaluation .....	28
Functional Testing .....	28
Discussion .....	31
References .....	32

## Statement of Compliance

For this coursework certain functional and non-functional requirements had to be met. Below are two tables specifying these requirements and how they were met in the design.

### Functional Requirements

Requirement	How Requirement Was Met
User must enter his/her user ID to use the web app	When the web app loads a bootstrap modal pops up with an input and login button. This modal can't be closed and will only close once the user has entered their ID and clicked login
Send a subscription request to another user identified by user ID	In the subscription management section there is a Send Friend Request box that takes in an ID. This creates a request by making a call to the request service.
See a list of incoming subscription requests from other users	In the subscription management section of the web app there is a Friend Requests section that contains a list of pending friend requests
Approve or deny incoming subscription requests	In the subscription management Friend Requests section each pending request has the friends ID and a approve and deny button. The approve and deny buttons allow approval and denying of incoming requests.
Check-in the current users location using a marker on the map	In the location management section, there is a map. Blue markers on the map represent friends locations and the green marker represents the current user. Users can drag the green marker to a new location and click the Check in Location button to check in their current location
See a list of friends (IDs) who have approved the requests	In the location management section is the friends list where friends who have approved requests will be shown
See a map with markers showing the last check-in locations of friends	In the location management section, the map shows the last check in locations of friends via blue markers on the map
Click on a friend's name to center the map on that friend	In the location management section, the user can click on a friends ID in the Friends list and that friends location will be centered on the map.

### Non-Functional Requirements

Requirement	How Requirement Was Met
You must use AWS DynamoDB for data storage. If you do not have AWS access, you can use a local DynamoDB server	Local DymanoDB server was used
You must use a System-oriented Architecture. There must be web services that expose functionalities of your backend server	Implemented using Jersey for JAVA. Three web services were created, Request, Subscription and Location. These provided functionalities in a RESTful way (API Specification is defined later in this report)

## Design

### API

#### Specifications

##### Request Service

Operation	HTTP Method	URL Pattern	Request Parameters(s)	Status Codes and Replies	Reply
Add a new Request	POST	CM4108Coursework/api/request	requesterID recipientID	201: Request created 400:missing parameters(s) 500:server-side error	Status code
Retrieve all requests for one user	GET	CM4108Coursework/api/request/{recipientID}		200: OK 404: not found 500:server-side error	JSON { [ { recipientID:recipientID, requesterID:requesterID } ] }
Delete a request (given its ID?)	DELETE	CM4108Coursework/api/request/{recipientID}	requesterID	200:OK 404:not found 500:server error	Status code

## Subscription Service

Operation	HTTP Method	URL Pattern	Request Parameters(s)	Status Codes and Replies	Reply
Add a new Subscription	POST	CM4108Coursework/ api/subscription	SubscriptionID SubscriberID	201: subscription created 400:missing parameters(s) 500:server-side error	Status Code
Retrieve all subscriptions for one user	GET	CM4108Coursework/ api/subscription/{subscriberID}		200: OK 404: not found 500:server-side error	JSON { [ { subscriberID:subscriberID, subscriptionID:subscriptionID } ... ]} }
Delete a subscription (given its ID?)	DELETE	CM4108Coursework/ api/subscription/{subscriberID}	SubscriptionID	200:OK 404:not found 500:server error	Status Code

## Location Service

Operation	HTTP Method	URL Pattern	Request Parameters(s)	Status Codes and Replies	Reply
Add a new/update Location	POST	CM4108Coursework/ api/location	id Latitude Longitude	201: subscription created 400:missing parameters(s) 500:server-side error	Status code
Retrieve location of single id	GET	CM4108Coursework/ api/location/{id}		200: OK 400:missing paramter(s) 500: Server side error	JSON { "latitude":lat, "longitude":long }



## Models

### Request

Model used to represent a subscription request. Takes in the ID of the person making the request(requesterID) and the person receiving the request (recipientID). Uses Java annotations for DynamoDB. RecipientID is the hash key or main key as this table is most likely to be searched by users who are receiving requests. Range key is the requesterID, this allows identification of specific requests.

```
@DynamoDBTable(tableName=RequestConfig.DYNAMODB_TABLE_NAME)
public class Request {
    private String recipientID;
    private String requesterID;

    public Request(){

    }

    public Request(String requesterID, String recipientID){
        this.setRequesterID(requesterID);
        this.setRecipientID(recipientID);
    }

    @DynamoDBHashKey(attributeName="recipientID")
    public String getRecipientID() {
        return recipientID;
    }

    public void setRecipientID(String recipientID) {
        this.recipientID = recipientID;
    }

    @DynamoDBRangeKey(attributeName="requesterID")
    public String getRequesterID() {
        return requesterID;
    }

    public void setRequesterID(String requesterID) {
        this.requesterID = requesterID;
    }
}
```

### DynamoDB Mapping

A local DynamoDB was used, table for Request class mapping was created with the following attributes:

Table Name: cm4108-coursework-request

Hash Key: recipientID

Range Key: requesterID

## Subscription

Model used to represent a subscription. Takes in the ID of the person subscribing to a person's location (subscriberID) and the ID of the person being subscribed to (subscriptionID). Uses Java annotations for DynamoDB. SubscriberID is the hash key or main key as this table is most likely to be searched by users who require all subscriptions they have subscribed to. Range key is the subscriptionID this allows identification of specific subscriptions.

```
@DynamoDBTable(tableName=SubscriptionConfig.DYNAMODB_TABLE_NAME)
public class Subscription {
    private String subscriberID;
    private String subscriptionID;

    public Subscription() {

    }

    public Subscription(String subscriberID,String subscriptionID) {
        this.setSubscriberID(subscriberID);
        this.setSubscriptionID(subscriptionID);
    }

    @DynamoDBHashKey(attributeName="subscriberID")
    public String getSubscriberID() {
        return subscriberID;
    }

    public void setSubscriberID(String subscriberID) {
        this.subscriberID = subscriberID;
    }

    @DynamoDBRangeKey(attributeName="subscriptionID")
    public String getSubscriptionID() {
        return subscriptionID;
    }

    public void setSubscriptionID(String subscriptionID) {
        this.subscriptionID = subscriptionID;
    }
}
```

### DynamoDB Mapping

A local DynamoDB was used, table for Subscription class mapping was created with the following attributes:

Table Name: cm4108-coursework-subscription

Hash Key: subscriberID

Range Key: subscriptionID

## Location

Model used to represent a location. Takes in the ID of the user, and the users latitude and longitude. Uses Java annotations for DynamoDB. ID is the hash key, as for each user only one location will be stored.

```
@DynamoDBTable(tableName = LocationConfig.DYNAMODB_TABLE_NAME)
public class Location {
    private String id;
    private double longitude,latitude;

    public Location() {

    }

    public Location (String id,double longitude,double latitude) {
        this.setId(id);
        this.setLongitude(longitude);
        this.setLatitude(latitude);
    }

    @DynamoDBHashKey(attributeName="id")
    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    @DynamoDBAttribute(attributeName="longitude")
    public double getLongitude() {
        return longitude;
    }

    public void setLongitude(double longitude) {
        this.longitude = longitude;
    }

    @DynamoDBAttribute(attributeName="latitude")
    public double getLatitude() {
        return latitude;
    }

    public void setLatitude(double latitude) {
        this.latitude = latitude;
    }
}
```

### DynamoDB Mapping

A local DynamoDB was used, table for Location class mapping was created with the following attributes:

Table Name: cm4108-coursework-location

Hash Key: id

## Resources

### Request Service

```
@Path("/request")
public class RequestResource {
    @POST
    @Produces(MediaType.TEXT_PLAIN)
    public Response addRequest(@FormParam("requesterID") String requesterID, @FormParam("recipientID") String recipientID) {
        try {
            //Create new request from params
            Request request = new Request(requesterID, recipientID);
            //Try and save in DB
            DynamoDBMapper mapper = DynamoDBUtil.getDBMapper(RequestConfig.REGION, RequestConfig.LOCAL_ENDPOINT);
            mapper.save(request);
            return Response.status(201).entity("Request Saved").build();
        } catch (Exception e) {
            //Error
            return Response.status(400).entity("Error in saving Request").build();
        }
    }

    @Path("/{recipientID}")
    @GET
    @Produces(MediaType.APPLICATION_JSON)
    public Collection<Request> getRequests(@PathParam("recipientID") String recipientID){
        //Query table for all requests with this recipientID
        DynamoDBMapper mapper = DynamoDBUtil.getDBMapper(RequestConfig.REGION, RequestConfig.LOCAL_ENDPOINT);
        Map<String, AttributeValue> eav = new HashMap<String, AttributeValue>();
        eav.put(":val1", new AttributeValue().withS(recipientID));
        DynamoDBQueryExpression<Request> queryExpression = new DynamoDBQueryExpression<Request>().withKeyConditionExpression("recipientID = :val1").
            withExpressionAttributeValues(eav);
        Collection<Request> requests = mapper.query(Request.class, queryExpression);
        //If nothing returned throw not found error
        if (requests==null)
            throw new WebApplicationException(404);
        return requests;
    }

    @Path("/{recipientID}")
    @DELETE
    public Response deleteOneRequest(@PathParam("recipientID") String recipientID, @QueryParam("requesterID") String requesterID) {
        //Load request from db
        DynamoDBMapper mapper = DynamoDBUtil.getDBMapper(RequestConfig.REGION, RequestConfig.LOCAL_ENDPOINT);
        Request response = mapper.load(Request.class, recipientID, requesterID);
        //If null throw not found error
        if (response==null)
            throw new WebApplicationException(404);
        //Delete
        mapper.delete(response);
        return Response.status(200).entity("Request Deleted").build();
    }
}
```

### Subscription Service

```
@Path("/subscription")
public class SubscriptionResource {
    @POST
    @Produces(MediaType.TEXT_PLAIN)
    public Response addSubscription(@FormParam("subscriberID") String subscriberID, @FormParam("subscriptionID") String subscriptionID) {
        try {
            //Create new subscription from params
            Subscription subscription = new Subscription(subscriberID, subscriptionID);
            //Save subscription
            DynamoDBMapper mapper = DynamoDBUtil.getDBMapper(SubscriptionConfig.REGION, SubscriptionConfig.LOCAL_ENDPOINT);
            mapper.save(subscription);
            return Response.status(201).entity("Subscription Saved").build();
        } catch (Exception e) {
            //Error
            return Response.status(400).entity("Error in saving Subscription").build();
        }
    }

    @Path("/{subscriberID}")
    @GET
    @Produces(MediaType.APPLICATION_JSON)
    public Collection<Subscription> getSubscriptions(@PathParam("subscriberID") String subscriberID){
        //Query db for any subscriptions with subscriberID
        DynamoDBMapper mapper = DynamoDBUtil.getDBMapper(SubscriptionConfig.REGION, SubscriptionConfig.LOCAL_ENDPOINT);
        Map<String, AttributeValue> eav = new HashMap<String, AttributeValue>();
        eav.put(":val1", new AttributeValue().withS(subscriberID));
        DynamoDBQueryExpression<Subscription> queryExpression = new DynamoDBQueryExpression<Subscription>().withKeyConditionExpression("subscriberID = :val1").
            withExpressionAttributeValues(eav);
        Collection<Subscription> subscriptions = mapper.query(Subscription.class, queryExpression);
        //If null return throw not found
        if (subscriptions==null)
            throw new WebApplicationException(404);
        return subscriptions;
    }

    @Path("/{subscriberID}")
    @DELETE
    public Response deleteOneSubscription(@PathParam("subscriberID") String subscriberID, @QueryParam("subscriptionID") String subscriptionID) {
        //Load subscription from db
        DynamoDBMapper mapper = DynamoDBUtil.getDBMapper(SubscriptionConfig.REGION, SubscriptionConfig.LOCAL_ENDPOINT);
        Subscription subscription = mapper.load(Subscription.class, subscriberID, subscriptionID);
        //If null throw not found
        if (subscription==null)
            throw new WebApplicationException(404);
        //Delete
        mapper.delete(subscription);
        return Response.status(200).entity("Subscription Deleted").build();
    }
}
```

## Location Service

```
@Path("/location")
public class LocationResource {
    @POST
    @Produces(MediaType.TEXT_PLAIN)
    public Response addLocation(@FormParam("id") String id,@FormParam("longitude") double longitude,@FormParam("latitude") double latitude) {
        try {
            //Create new location from params
            Location location = new Location(id,longitude,latitude);
            //Save new location
            DynamoDBMapper mapper = DynamoDBUtil.getDBMapper(LocationConfig.REGION, LocationConfig.LOCAL_ENDPOINT);
            mapper.save(location);
            return Response.status(201).entity("Location Saved").build();
        } catch (Exception e) {
            //Error
            return Response.status(400).entity("Error in saving location").build();
        }
    }

    @Path("/{id}")
    @GET
    @Produces(MediaType.APPLICATION_JSON)
    public Location getOneLocation(@PathParam("id") String id) {
        //Load location
        DynamoDBMapper mapper = DynamoDBUtil.getDBMapper(LocationConfig.REGION, LocationConfig.LOCAL_ENDPOINT);
        Location location = mapper.load(Location.class, id);
        //If null throw not found error
        if (location == null) {
            throw new WebApplicationException(404);
        }
        return location;
    }
}
```

## Front End

### HTML

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="ISO-8859-1">
    <title>Find My Friends</title>

    <!-- Load Bootstrap and Required JS (Includes jQuery) -->
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css"
integrity="sha384-MCW98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO"
crossorigin="anonymous">
    <script src="https://code.jquery.com/jquery-3.3.1.min.js" integrity="sha256-
FgpCb/KJQLLNF0u91ta32o/NMZxltwRo8QtmkMRdAu8=" crossorigin="anonymous"></script>
    <script
src="https://cdn.jsdelivr.net/npm/popper.js/1.14.3/umd/popper.min.js"
integrity="sha384-ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjAK/L8WvCWPm49"
crossorigin="anonymous"></script>
    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js"
integrity="sha384-ChfqqxuZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ60W/JmZQ5stwEULTy"
crossorigin="anonymous"></script>

    <!-- Load the LeafletJS API. -->
    <link rel="stylesheet"
href="https://unpkg.com/leaflet@1.3.4/dist/leaflet.css" integrity="sha512-
puBpdR07980zVTTbP4A8Ix/L+A4dHDD0DGqYW6RQ+9jxkRfCLaxxQb/SJAWZfwAkuyeQUyt07+7N4QKrDh+drA=="
crossorigin=""/>
    <script src="https://unpkg.com/leaflet@1.3.4/dist/leaflet.js"
integrity="sha512-
nMMmRyTVoLYqjP9hrbed9S+FzjZHW5gY1TWCHA5ckwXZBadntCNS8kEqAWdrb907rxbCaA4LKTiWjDXZxfLOcA=="
crossorigin=""></script>

    <!-- Load local CSS and JavaScript. -->
    <link rel="stylesheet" href="css/index.css">
    <script src="script/index.js"></script>
  </head>
  <body>
    <!-- NavBar -->
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
      <a class="navbar-brand" href="#">Find My Friends</a>
    </nav>
    <!-- Main Content -->
    <div class="container-fluid" id="mainContent">
      <!-- Profile -->
      <div id="profile" class="card">
        <div class="card-body">
          <h1 class="card-title">Profile</h1>
          <div class="row">
            <div class="col-sm-3">
              <h5 id="username"></h5>
              <button id="logoutButton" type="submit"
class="btn btn-primary">Logout</button>
            </div>
          </div>
        </div>
      </div>
      <!-- Subscription -->
      <div id="subscription" class="card">
        <div class="card-body">
          <h1 class="card-title">Subscription Management</h1>

```

```

<div class="row">
  <!-- Send Request -->
  <div class="col-lg-3">
    <h5>Send Friend Request</h5>
    <form>
      <div class="form-row">
        <div class="col-sm-9">
          <input type="text"
class="form-control" id="friendRequestID" placeholder="Friend ID">
        </div>
        <div class="col-sm-3">
          <button
id="sendRequestButton" type="submit" class="btn btn-primary btn-block">Submit</button>
        </div>
      </div>
    </form>
  </div>
  <!-- Handle Incoming Requests -->
  <div class="col-lg-9">
    <h5>Friend Requests</h5>
    <div>
      <ul class="list-group" id="requests">
      </ul>
    </div>
  </div>
</div>
</div>
<!-- Location -->
<div id="Location" class="card">
  <div class="card-body">
    <h1 class="card-title">Location Management</h1>
    <div class="row">
      <!-- Friends List -->
      <div class="col-sm-3">
        <h5>Friends List</h5>
        <p>Click a friend to center their location
on the map</p>
        <ul class="list-group" id="friendsList">
        </ul>
      </div>
      <!-- Map -->
      <div class="col-sm-9">
        <h5>Map</h5>
        <p>To check in your current location drag
your own <span class="text-success">green</span> marker to the location and click 'Check
In Location'</p>
        <div id="map">
        </div>
        <button class="btn btn-primary btn-block"
id="checkInButton">Check In Location</button>
        <h6>Last Check In</h6>
        <p id="LastCheckIn"></p>
      </div>
    </div>
  </div>
</div>
<!-- Modal for Entering ID -->
<div class="modal fade" id="idModal" data-backdrop="static" tabindex="-1"
role="dialog" aria-labelledby="idModalCenterTitle" aria-hidden="true">

```

```

<div class="modal-dialog modal-dialog-centered" role="document">
  <div class="modal-content">
    <div class="modal-header">
      <h5 class="modal-title" id="idModalLongTitle">Login</h5>
    </div>
    <div class="modal-body">
      <form>
        <div class="form-row">
          <div class="col-sm-9">
            <input type="text" class="form-control"
id="userID" placeholder="ID">
          </div>
          <div class="col-sm-3">
            <button id="loginButton" type="submit"
class="btn btn-primary btn-block">Login</button>
          </div>
        </div>
      </form>
    </div>
    <div class="modal-footer">
      <p id="idModalError"></p>
    </div>
  </div>
</div>
<!-- Loading Modal -->
<div class="modal fade" id="LoadingModal" data-backdrop="static" tabindex="-
1" role="dialog" aria-labelledby="LoadingModalCenterTitle" aria-hidden="true">
  <div class="modal-dialog modal-dialog-centered" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="LoadingModalLongTitle">Loading</h5>
      </div>
      <div class="modal-body">
        <div class="loader"></div>
      </div>
      <div class="modal-footer">
      </div>
    </div>
  </div>
</div>
<!-- Notification Modal -->
<div class="modal fade" id="notificationModal" tabindex="-1" role="dialog"
aria-labelledby="notificationModalCenterTitle" aria-hidden="true">
  <div class="modal-dialog modal-dialog-centered" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="notificationModalTitle"></h5>
        <button type="button" class="close" data-dismiss="modal" aria-
label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
      </div>
      <div class="modal-body">
        <p id="notificationModalBody"> </p>
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-primary" data-
dismiss="modal">OK</button>
      </div>
    </div>
  </div>
</div>
</div>

```



```
</body>  
</html>
```

Moodle Submission

## CSS

```
/*Content Styles*/
#mainContent{
    margin-bottom:1vh;
}

/* Bootstrap Styles */
.card {
    margin-top:1vh;
}

/* Map Styles */
#map {
    width:100%;
    height: 45vh;
}

/* Loading Spinner https://www.w3schools.com/howto/tryit.asp?filename=tryhow_css_loader*/
.loader {
    border: 16px solid #f3f3f3;
    border-radius: 50%;
    border-top: 16px solid #3498db;
    width: 120px;
    margin:auto;
    height: 120px;
    -webkit-animation: spin 2s linear infinite; /* Safari */
    animation: spin 2s linear infinite;
}

/* Safari */
@-webkit-keyframes spin {
    0% { -webkit-transform: rotate(0deg); }
    100% { -webkit-transform: rotate(360deg); }
}

@keyframes spin {
    0% { transform: rotate(0deg); }
    100% { transform: rotate(360deg); }
}
```

## JS

//API Keys

**var**

mapBoxApiKey="pk.eyJ1IjoiYWxkb2FiZG4iLCJhIjoiY2pubHV5am93MDJnNzNxcXh3ZTVyOTY4dSJ9.JRxf6HQz9nb6XzJPhsX5zQ";

**var**

openWeatherAPIKey="pk.eyJ1IjoiYWxkb2FiZG4iLCJhIjoiY2pubHV5am93MDJnNzNxcXh3ZTVyOTY4dSJ9.JRxf6HQz9nb6XzJPhsX5zQ";

**var** baseUrl="api";

**var** id = null;

**var** user = {};

**var** map = null;

//Document Ready

**try** {

\$(**function**()) {init()}}

} **catch** (e){

alert("jQuery not loaded");

}

//Initialise

**function** init() {

//ID Modal

\$("#idModal").modal('show');

//Login Click

\$("#loginButton").click(**function**(e){

e.preventDefault();

//Check ID Value

id = \$('#userID').val();

//If valid hide model else show error

**if** (id != ""){

//Map Setup

map = makeMap("map",1,0.0,0,0);

//Clear Login

\$('#userID').val("");

\$('#idModalError').text("");

//Show Loading Modal

\$("#idModal").modal('hide');

\$("#loadingModal").modal('show');

setUser();

//Delay as there seems to be a bug where modal('hide') doesn't work

setTimeout(**function**(){\$('#loadingModal').modal('hide');},1000);

} **else** {

\$('#idModalError').text("Invalid ID");

}

});

//Logout Click

\$("#logoutButton").click(**function**()) {

//Reset

id = "";

map.remove();

//Call to set user with blank user to clear all the user related data

//Creates request errors and ID is blank

setUser();

//Show Login Modal

\$("#idModal").modal('show');

});

//Send Friend Request Click

\$("#sendRequestButton").click(**function**(e){

e.preventDefault();

**var** friendID = \$('#friendRequestID').val();

**if** (friendID==" " || friendID==user.id)

```

        return;
        //Start Loading
        $('#loadingModal').modal('show');
        var url = baseUrl + "/request";
        data = {"requesterID": user.id, "recipientID": friendID};
        $.post(url, data)
            .done(function(){
                //Reset Input Text
                $('#friendRequestID').val("");
                //Show Modals
                setTimeout(function(){
                    //Finish Loading
                    $('#loadingModal').modal('hide');
                    //Show Notification
                    setNotification("Friend Request", "Friend Request Sent");
                    $('#notificationModal').modal('show');
                }, 1000);
            })
            .fail(function(){
                setTimeout(function(){
                    //Finish Loading
                    $('#loadingModal').modal('hide');
                    //Show Notification
                    setNotification("Error", "An error occurred please try again
later");
                    $('#notificationModal').modal('show');
                }, 1000);
            })
    });

    //Approve Click
    $(document).on("click", '.approveRequestButton', function(){
        //Start Loading
        $('#loadingModal').modal('show');
        //Delete Request
        listItem = $(this).parent();
        span = listItem.find("span");
        requesterID = span.text();
        data = {"requesterID": requesterID};
        var url = baseUrl + "/request/" + user.id + "?requesterID=" + requesterID;
        $.ajax({
            url: url,
            type: 'DELETE',
            success: function(results){
                //Create Subscription
                url = baseUrl + "/subscription";
                subscriberID = requesterID;
                //Subscribe both parties to each other
                data = {"subscriberID": subscriberID, "subscriptionID": user.id};
                $.post(url, data, function(){
                    setTimeout(function(){
                        //Finish Loading
                        $('#loadingModal').modal('hide');
                        //Show Notification
                        setNotification("Friend Request", "Friend Request
Approved");
                        $('#notificationModal').modal('show');
                    }, 1000);
                }).fail(function(){
                    setTimeout(function(){
                        //Finish Loading
                        $('#loadingModal').modal('hide');
                        //Show Notification

```

```

        setNotification("Error", "An error occurred please try
again later");

        $('#notificationModal').modal('show');
    },1000);
});
//Subscribe user to requester
dataReturn =
{"subscriberID":user.id,"subscriptionID":subscriberID};
$.post(url,dataReturn,function(){
    setSubscription();
    setLocation();
}).fail(function(){
    setTimeout(function(){
        //Finish Loading
        $('#loadingModal').modal('hide');
        //Show Notification
        setNotification("Error", "An error occurred please try
again later");

        $('#notificationModal').modal('show');
    },1000);
});
},
error: function(results){
    setTimeout(function(){
        //Finish Loading
        $('#loadingModal').modal('hide');
        //Show Notification
        setNotification("Error", "An error occurred please try again
later");

        $('#notificationModal').modal('show');
    },1000);
}
});
});

//Deny Click
$(document).on('click','deniedRequestButton',function(){
    //Start Loading
    $('#loadingModal').modal('show');
    //Delete Request
    listItem = $(this).parent();
    span = listItem.find("span");
    requesterID = span.text();
    var url = baseUrl + "/request/" + user.id + "?requesterID=" + requesterID;
    $.ajax({
        url:url,
        type:'DELETE',
        success: function(results){
            setSubscription();
            setTimeout(function(){
                //Finish Loading
                $('#loadingModal').modal('hide');
                //Show Notification
                setNotification("Friend Request", "Friend Request Denied");
                $('#notificationModal').modal('show');
            },1000);
        },
        error: function(results){
            setTimeout(function(){
                //Finish Loading
                $('#loadingModal').modal('hide');
                //Show Notification
                setNotification("Error", "An error occurred please try again
later");

```

```

        $('#notificationModal').modal('show');
    },1000);
    }
    });

//Friend Click
$(document).on('click','.friend',function(){
    //Focus map on friends marker

    map.setView(L.latLng($(this).data('lat'),$(this).data('long')),map.getZoom());
});

//Check In Click
$('#checkInButton').click(function(){
    //Start Loading
    $('#loadingModal').modal('show');
    var url = baseUrl + "/location";
    var lat = user.marker.getLatLng().lat
    var long = user.marker.getLatLng().lng
    data = {
        "id":user.id,
        "longitude": long,
        "latitude": lat
    }
    $.post(url,data)
        .done(function(){
            $('#lastCheckIn').text("Lat: " + lat + " Long: " + long);
            setTimeout(function(){
                //Finish Loading
                $('#loadingModal').modal('hide');
                //Show Notification
                setNotification("Checked In", "You checked in at Lat: " + lat +
" Long: " + long);

                $('#notificationModal').modal('show');
            },1000);
        })
        .fail(function(xhr,status,error){
            setTimeout(function(){
                //Finish Loading
                $('#loadingModal').modal('hide');
                //Show Notification
                setNotification("Error", "An error ocured please try again
later");

                $('#notificationModal').modal('show');
            },1000);
        });
    });
}

//Set User
function setUser(){
    //Set Attributes
    user.id = id;
    //Set Map Marker
    url = baseUrl + "/location/"+user.id;
    $.getJSON(url,function(jsonData){
        user.marker =
makeMarker(map,jsonData['longitude'],jsonData['latitude'],user.id,true,colouredIcon('#32C
D32'));
        $('#lastCheckIn').text("Lat: " + jsonData['latitude'] + " Long: " +
jsonData['longitude']);
    }).fail(function(){
        user.marker = makeMarker(map,0,0,user.id,true,colouredIcon('#32CD32'));
    });
}

```

```

        $('#lastCheckIn').text("Lat: 0 Long: 0");
    });
    //Set Cards
    setProfile();
    setSubscription();
    setLocation();
}

//Set Profile Card
function setProfile(){
    //Set Username
    $('#username').text(user.id);
}

//Set Subscription Management Card
function setSubscription(){
    var friendRequests = [];
    var url = baseUrl+"/request/"+user.id;
    //Clear List
    $('#requests').empty();
    //Get Friend Requests
    $.getJSON(url,function(jsonData){
        friendRequests = jsonData;
        //Set Friend Request List
        for(var request of friendRequests){
            $('#requests').append("<li class='list-group-item'><span>" +
request.requesterID + "</span><button class='approveRequestButton btn-success btn float-
right'>Approve</button><button class='denyRequestButton btn-danger btn float-
right'>Deny</button></li>");
        }
    });
}

//Set Location Management Card
function setLocation(){
    var friendsList = [];
    var url = baseUrl + "/subscription/" + user.id;
    //Clear List
    $('#friendsList').empty();
    //Set Friends List
    $.getJSON(url,function(jsonData){
        friendsList = []
        //Convert Subscriptions to Friends
        for(var subscription of jsonData){
            let friend = {id:subscription.subscriptionID};
            friendsList.push(friend);
        }
        //Get locations of friends and create list
        $.each(friendsList,function(i,friend){
            url = baseUrl + "/location/"+friend.id;
            $.getJSON(url,function(jsonData){
                friend.marker =
makeMarker(map,jsonData['longitude'],jsonData['latitude'],friend.id,false,colouredIcon('#
0000FF'));
                $('#friendsList').append("<button class='list-group-item list-
group-item-action friend' data-lat='" + jsonData['latitude'] + "' data-long='" +
jsonData['longitude'] + "'>" + friend.id + "</button>");
            }).fail(function(){
                friend.marker =
makeMarker(map,0,0,friend.id,false,colouredIcon('#0000FF'));
                $('#friendsList').append("<button class='list-group-item list-
group-item-action friend' data-lat='" + 0 + "' data-long='" + 0 + "'>" + friend.id +
"</button>");
            });
        });
    });
}

```

```

    });
  });
}

//Set Notification Modal
function setNotification(title,body){
  $('#notificationModalTitle').text(title);
  $('#notificationModalBody').text(body);
}

//Map Functions
function makeMap(divId,zoomLevel,longitude,latitude) {
  var location=L.latLng(longitude,latitude); //create location
  var map=L.map(divId).setView(location,zoomLevel); //put map into division
  L.tileLayer('https://api.tiles.mapbox.com/v4/{id}/{z}/{x}/{y}.png?access_token='+mapBoxApiKey,
    {attribution: 'Map data &copy; <a
href="https://www.openstreetmap.org/">OpenStreetMap</a> contributors, <a
href="https://creativecommons.org/licenses/by-sa/2.0/">CC-BY-SA</a>, Imagery &copy; <a
href="https://www.mapbox.com/">Mapbox</a>',
    maxZoom: 18,
    id: 'mapbox.streets',
    accessToken: mapBoxApiKey}
  ).addTo(map);
  return map; //return map object
} //end function

//
//create a marker on a map
//the marker is returned as we need to get its position later
//
function makeMarker(map,longitude,latitude,title="",draggable=true,icon=L.Icon.Default){
  var location=L.latLng({lon:longitude,lat:latitude}); //create marker at given
position
  var marker=L.marker(location,{title:title,draggable:draggable,icon:icon});
  //make a draggable marker
  marker.addTo(map);
  //add marker to map
  return marker; //return marker object
} //end function

//Custom Leaflet Marker
//https://stackoverflow.com/questions/23567203/leaflet-changing-marker-color
function colouredIcon(colour){
  //Set Icon Style
  var markerHtmlStyles = `
background-color: ${colour};
width: 1.5rem;
height: 1.5rem;
display: block;
left: -1.5rem;
top: -1.5rem;
position: relative;
border-radius: 3rem 3rem 0;
transform: rotate(45deg);
border: 1px solid #FFFFFF`

  //Create Icon
  var icon = L.divIcon({
    className: "my-custom-pin",
    iconAnchor: [0, 24],
    labelAnchor: [-6, 0],
    popupAnchor: [0, -36],
    html: `

```

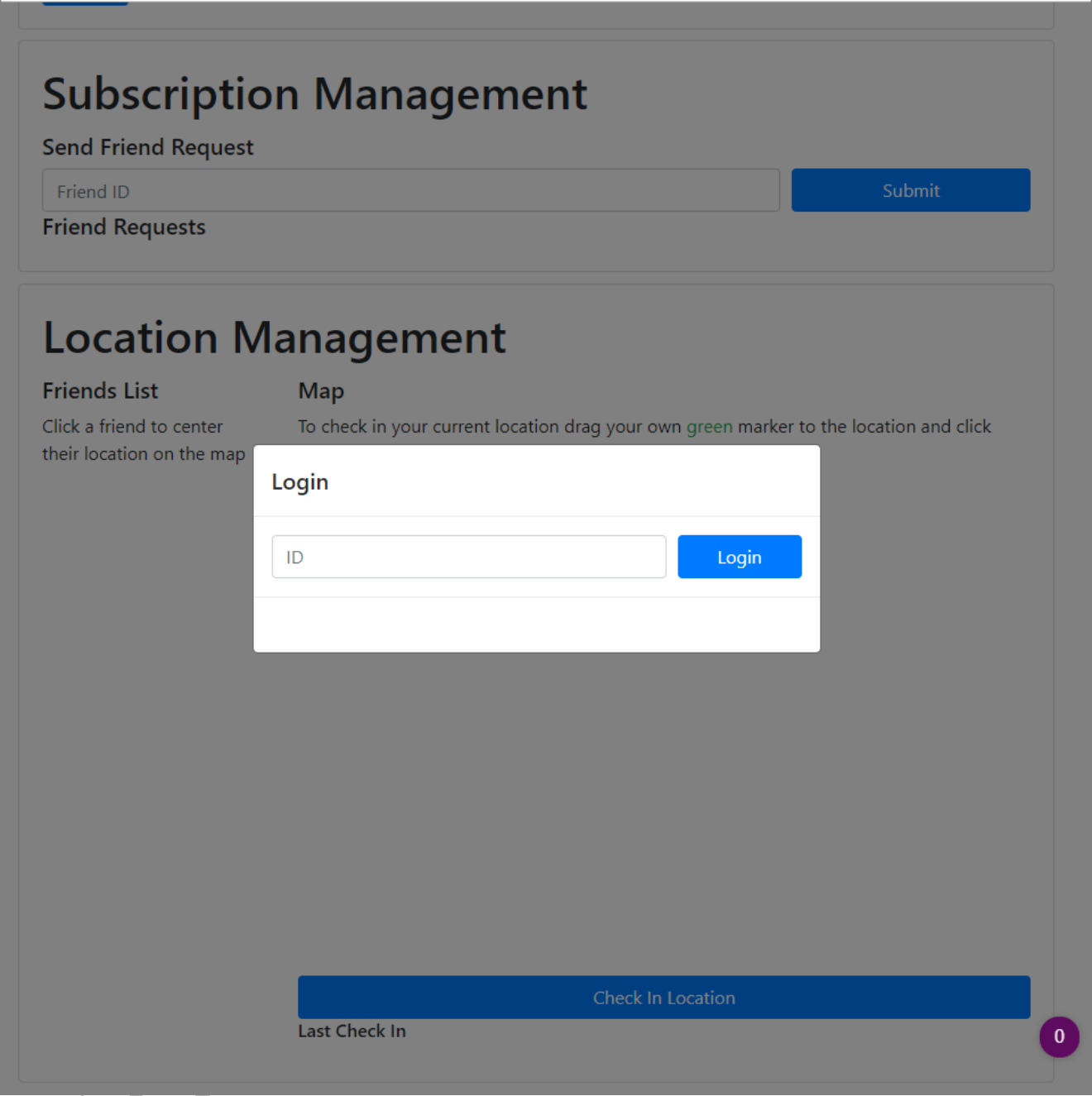


```
    })  
    return icon  
}
```

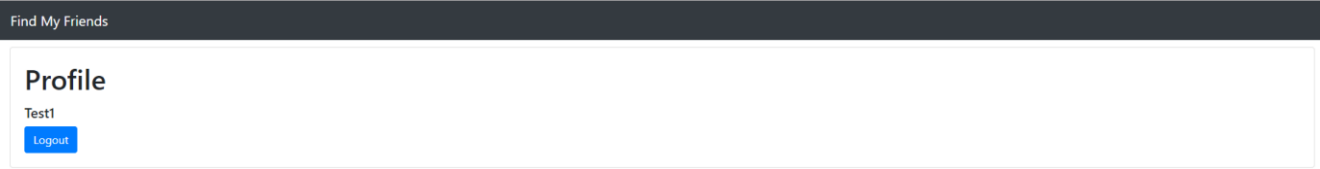
Moodle Submission

Screenshots

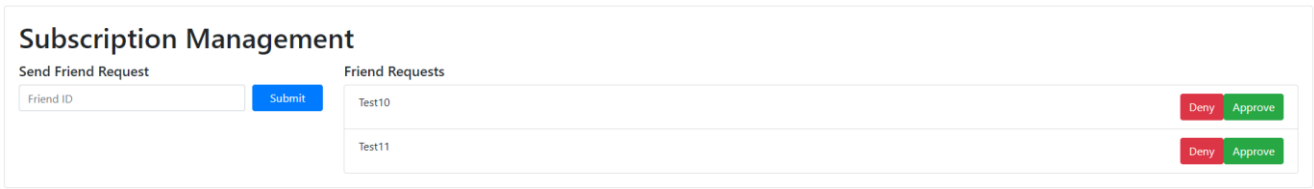
Login



Profile



Subscription Management



Location Management

### Location Management

Friends List

Click a friend to center their location on the map

Test2

Test3

Test4


Test6

Test7

Test8

Map

To check in your current location drag your own green marker to the location and click 'Check In Location'



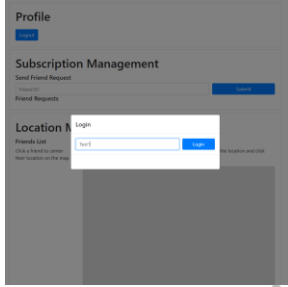

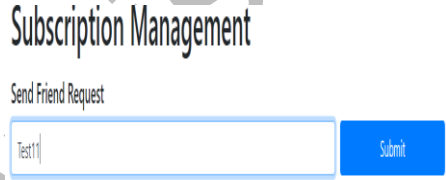
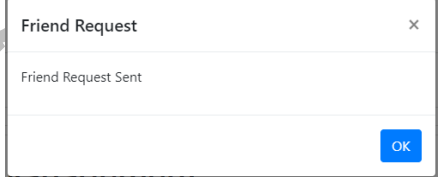
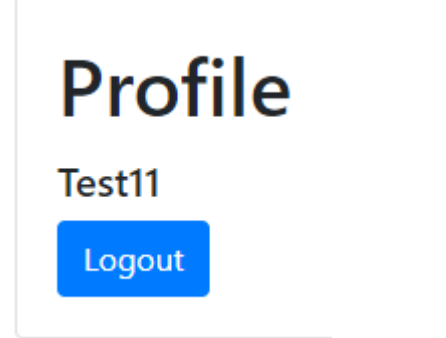
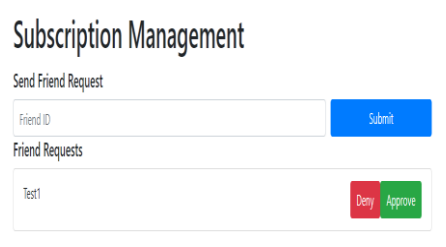
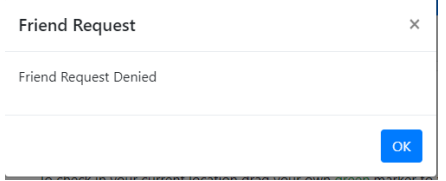
Last Check In


Lat: 43.10555191416177 Long: 7.031250000000001



27

## Evaluation

### Functional Testing

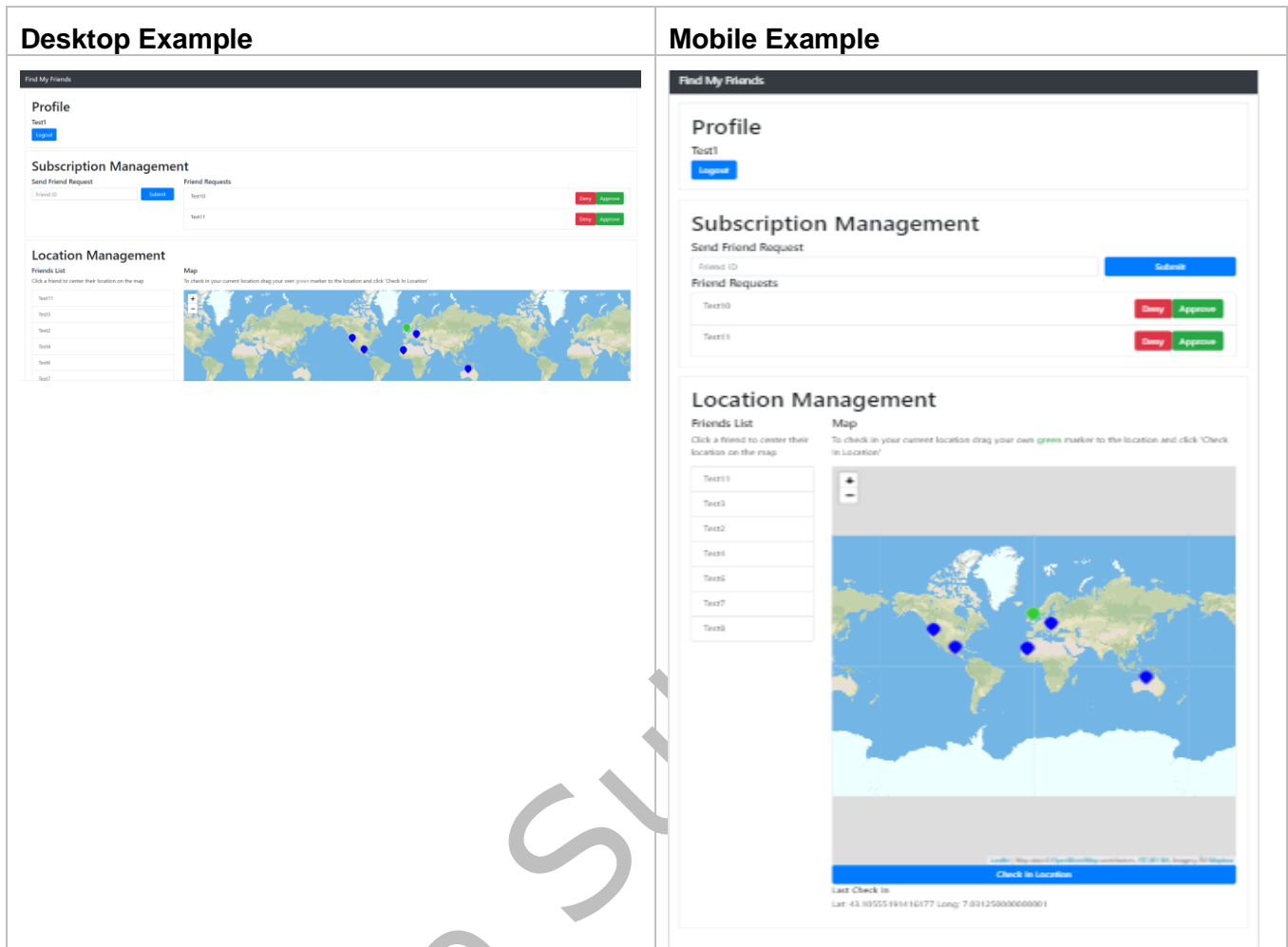
Task	Input	Expected Output	Actual Output	Screenshots	Note
User enter ID to login/use the web app	Test1	Login as Test1	Logged in as Test 1	 	Worked as intended
Send a subscription request to another user	Test11	Request sent to Test11	Request sent to Test11	 	Worked as intended
See a list of subscription requests from other users	N/A	Request list populated	Request list populated	 	Worked as intended
Deny subscription request	Click deny button	Subscription request denied	Subscription request denied		Worked as intended

				<div><div>Friends List</div><div>Click a friend to center their location on the map</div></div>	
Approve subscription request	Click approve button	Subscription request approved	Subscription request approved	<div><div>Friend Request</div><div>Friend Request Approved</div><div>OK</div></div> <div><div>Friends List</div><div>Click a friend to center their location on the map</div><div>Test1</div></div>	Worked as intended
Check in users current location using a marker on a map	Move marker click check in location button	Location checked in	Location checked in	<div><div><div><div><div>+</div><div>-</div></div><div></div><div>Leaflet   Map data © OpenStreetMap contributors, CC-BY-SA, Imagery © Mapbox</div></div><div>Check In Location</div><div>Last Check In</div><div>Lat: 0 Long: 0</div></div><div><div>Checked In</div><div>You checked in at Lat: 33.75565014700787 Long: -109.64493597206052</div><div>OK</div></div></div>	Worked as intended

See a list of friends IDs who have approved requests	N/A	Friends list displayed	Friends list displayed	<p><b>Friends List</b></p> <p>Click a friend to center their location on the map</p> <div> <div>Test2</div> <div>Test4</div> <div>Test3</div> <div>Test7</div> <div>Test6</div> <div>Test8</div> </div>	Worked as intended
See a map with markers showing the last check-in locations of friends	N/A	Map of friends last check-in-locations displayed	Map of friends last check-in-locations displayed		Worked as intended
Click on a friend's name to centre the map on that friend	Test clicked	Map centred on clicked friends location	Map centred on clicked friends location	<div> <div> <div>Test2</div> <div>Test4</div> <div>Test3</div> <div>Test7</div> <div>Test6</div> <div>Test8</div> </div>  </div>	Worked as intended

## Discussion

The application meets the functional and non-functional requirements set out in the coursework specification. The use of bootstrap means that the web app adapts to different screen sizes easily.



One major disadvantage of using bootstrap is that bootstrap uses CSS3 and a newer version of JS. This means that it isn't compatible with older browsers. The bootstrap modals used in the web app do not work with older browsers including the inbuilt browser of eclipse.

The application uses a simple contracting color scheme, text size has been considered to make sure the web app is readable by the majority of users.

Another limitation of the web app is due to the RESTful nature of the backend. The web app is set up so that when someone accepts a request it creates a subscription for both users. So the user who sent the request is subscribed to the user who accepted the request, and the user who accepted the request is then also subscribed to the person who made the request. This is so the requests behave more like friend requests in conventional apps. When a user accepts a request due to the stateless nature of the RESTful backend the application cannot notify the user who sent the request or cause a refresh ect. So a user will only be able to tell that a request was accepted if they refresh the web app and a user will never be able to tell if their request was denied.

The user's login data is not saved, so after a refresh of the app users need to login again. This could be fixed by using a cookie or other client-side persistence method such as a JS session.

## References

- Getbootstrap.com. (2018). *Bootstrap Documentation*. [online] Available at: <https://getbootstrap.com/docs/4.1/getting-started/introduction/> [Accessed 1 Nov. 2018].
- jQuery. (2018). *jQuery API*. [ONLINE] Available at: <https://api.jquery.com/>. [Accessed 1 November 2018].
- Leaflet. (2018). *Leaflet API Reference*. [online] Available at: <https://leafletjs.com/reference-1.3.4.html> [Accessed 1 Nov.2018].

Moodle Submission