Testing Plan

Team 7

Project Name:	Meadle
Members:	David Tschida, Eric Templin, Tylor Garrett, Michael Hockerman, Kyle Potts, Jeremy Meyer
Repository:	https://github.com/vidia/meadle https://github.com/vidia/meadle-android https://github.com/mhoc/meadle-backend

Severity 1

Test Case 001 APP Launch App

System - Android App

Severity - 1

Type - Functionality

Instructions:

1. Find the application in the Android phone's list of applications.
2. Tap on the app icon to launch the application.

Expected Result:

1. App opens to a screen with the option to join or create a new Meadle.

Test Case 002 APP Create A Meadle (From Requirement 1)

System - Android App

Severity - 1

Type - Functionality

Instructions:

1. Open the application
2. Select "Create Meadle"

Expected Result:

1. A new Meadle ID will be displayed on the screen, along with a button to share the ID with a friend.

Test Case 003 APP Join a Meadle (From Requirement 2)

System- Android App

Severity- 1

Type- Functionality

Instructions:

1. On one phone, create a meadle
2. On another phone
   a. Open the application
   b. Select "Join Meadle"
   c. Insert your ID

        d. Press submit

Expected Result:

1. The first phone should get a push notification that you have joined.

Test Case 004 APP Receive a list of possible meeting locations (From Requirement 3)

System- Android App

Severity- 1

Type- Functionality

Instructions:

1. Join/Create A Meadle
    a. If Creating a Meadle, wait for another user to join
    b. If Joining a Meadle, insert your code/id and press submit.
2. Make sure all users have joined

Expected Result:

1. A loading animation will run until a list of possible meeting locations is returned.

Test Case 005 API Users2 join Meeting:

System - Server

Severity - 1

Type - Functionality

Instructions:

1. Send a PUT Request to /meeting/{id}/join
2. In the Body of the request, send the JSON Data:

```
{
        "userId": "unique user id of joiner, static throughout each meeting",
        "lat": 12.34,
        "lng": 56.78
}
```

Expected Results:

1. If the id of the meeting id valid, then the server will send a 200 response
2. If the id of the meeting does not exist, then the server will send the JSON Response:

```
{
        "error": 404,
        "message": "The specified meadle does not exist."
}
```

Test Case 006 API Get Meeting:

System - Server

Severity - 1

Type - Functionality

Instructions:

1. Send a Get request to /meeting/{id}/
2. In the HTTP headers, include the userId to authenticate that only one of two users are accessing the meeting information
3. Where id is the id of a specific meeting, assigned when a meeting is created

Expected Result:

1. If the meetingId is valid, and the userId matches one of the two users involved in the meeting then the response is a JSON formatted response

   HTTP Response 200

```
{
  "id": "meeting id",
  "datetime": "time of meeting",
  // value is null until all users have joined
  "topLocations": {
    "yelpid": 5,
    "yelpId2": 4, ...
  // value is null unless voting is complete
  "location": "yelp id"
}
```

2. If the meetingId is not valid, then it will send

```
{
  "error": 404,
  "message": "The specified id was not found"
}
```

3. If the UserId is not valid,

```
    {
  "error": 401,
  "message": “You are not authorized to view this meeting"
}
```

Test Case 007 API Create Meeting - POST /meeting/{id}

System - Server

Severity- 1

Type - Functionality

Instruction:

1. Send a POST request to /meeting/{id}
2. In the POST Body, JSON data needs to be included in the format

```
{
  "userId": "unique user id of joiner, static throughout each meeting",
  "lat": 12.34,
  "lng": 56.78,
  "datetime": "date and time of meeting"
```

}

Expected Result:

1. If the userId is a correct Id then the server sends a JSON formatted response
```
{
        "id": "meadle id generated by server"
}
```

2. Id the userId is not valid then the server responds with
```
{
  "error": 404,
  "message": "The specified id was not found"
}
```

Severity 2

Test Case 008 SERVER Determine final meeting location (From Requirement 5)

System- Server

Severity- 2

Type- Functionality

Instructions:

1. Users 1 will send in a request to the server his or her top 5 ranked locations from the Android App
2. Users do will also send the same information to the server
3. The server will perform an selection algorithm the and decide which location is best for both and send it to the users

Expected Result:

1. The android application will go to a new screen containing information about the meeting location.

Test Case 009 APP Receive a notification when other people Join my Meadle (From Requirement 7)

System- Android App

Severity- 2

Type- Functionality

Instructions:

1. Create a meadle
2. Have a second user join the meadle

Expected Result:

1. Notification should appear on the first device to notify that the second user has joined.

Test Case 010 APP Vote on A List of Possible Meeting Locations

System- Android App

Severity- 2

Type- Functionality

Instructions:

1. Join/Create A Meadle
2. Once involved in a Meadle, a list of possible meeting locations will appear.
3. Arrange the five locations in order from most desired to least desired
4. Press enter to submit your results.
5. Wait for the other party to submit their results.

Expected Result:

1. You receive a notification that the votes were sent

Server

Test Case 011 API Verify meeting data is expired

System - Server

Severity - 2

Type - Functionality

Instructions:

1. Create a collection(mongodb) with a specified expiration (12-24 hours?)
2. Create a meeting (look at test cases above)
3. After the specified number of hours, either look up the meeting in the database (with the meeting id) or send a Get request to /meeting/id

Expected Result

1. If the meeting is looked up in the data, then no document should be found
2. If a GET request is sent, then the response will be

```
{
 "error": 404,
"message": "The specified meadle does not exist."
}
```

Test Case 012 API Only participants in a meadle should be able to access that meadle.

System - Server

Severity - 2

Type - Security/Functionality

Instructions -

1. Make a request to a valid meadle id with an invalid user id after a meadle has been created.

Expected Result:

1. Server should return an Unauthorized access.
```
{
    "error": 401,
    "message": “You are not authorized to view this meeting"
}
```

Test Case 013 Locations should be close to the midpoint.

System - Server

Severity - 2

Type - Functionality

Instructions:

1. Create a meadle.
2. Have someone else join this meadle at a known location.

Expected Result:

1. Possible meeting locations returned to the app should be roughly (+/-10%) equidistant from each participant in the meeting

Severity 3

Test Case 014 APP Display link to google maps for easy navigation (From Requirement 6)

System - APP

Severity - 3

Type - Functionality

Instructions:

1. Follow above instructions to start a meadle
2. Follow above instructions for users to join meadle
3. Follow above instructions for having users vote on a location
4. The app will display the location on the screen with information from Yelp about the destination.

Expected Result:

1. Decided location should display on the screen with a button or link that sends that location to the default maps app for navigation.

Test Case 015 Server No Sharing of Personal Information

System - Server

Severity - 3

Type - Functionality

Instructions:

1. Create a meeting and have another user join this meeting

Expected Result:

1. Make sure no user identifiable data is available on the server
    a. No phone number, address, etc
2. Make sure that locations are deleted after a certain period of time
    (See above for expiration of meeting data)

Test Case 016 Server Make sure the app does not support more than two users

System- Server

Severity- 3

Type- Boundary Value

Instructions:

1. Follow above instructions to add two users to the meadle.
2. Attempt to add a third user to the meadle (using the /meadle/{id}/join endpoint)

Expected Result:

1. Request should return with a failure and the user should not be added to the meadle.

Test Case 017 Android Battery Use

System - App

Severity - 3

Type - Functionality

Instruction:

1. Keep the application installed for 72 hours, keeping it in the foreground for no less than one hour during this time.
2. Open battery usage reports (Instructions vary by phone)

Expected output:

1. Meadle should be within the bottom 30% of all applications installed when ranked by battery usage.

Test Case 018 GitHub Code Storage

System - All

Severity - 3

Type - Nonfunctional
Instruction:
1. Open a terminal on a Linux computer.
2. Run “git clone https://github.com/vidia/meadle.git”
3. Run “git clone https://github.com/vidia/meadle-android.git”
4. Run “git clone https://github.com/mhoc/meadle-backend”

Expected output:
1. In the current directory, the meadle directory should contain all of our documentation and references to the meadle-android and meadle-backend repositories.
2. In the current directory, the meadle-android directory should contain the most recent version of our Android code.
3. In the current directory, the meadle-backend directory should containt the most recent version of our backend code.

Test Case 019 GitHub Main Repository Update
System - All
Severity - 3
Type - Nonfunctional
Instruction:
1. Open a terminal on a Linux computer.
2. Run “git clone --recursive https://github.com/vidia/meadle.git”
3. Run “cd meadle”

Expected output:
1. In the current directory, meadle/meadle-android and meadle/meadle-backend should contain the most recent version of those repositories.