Test Plan and Results

Overall Test Plan

We will approach testing our solution by first creating tests to check individual functionalities of the software. We can break up the tests into four categories: Front-End, Business Logic, Web API, and Database. By testing various functions within each of these categories individually, we can safely move forward to combining these levels of the software, forming the entire solution. This leads to a second level of testing, where we will test the entirety of the software. We will test key functionality of the software, which may take advantage of functionality within each of the four categories listed previously. For any of the tests, we will use fabricated data where necessary, and we will attempt to test any extreme cases that may occur. We will attempt to be thorough in our testing, so that if given positive test results in all tests, we can assume that the software will function properly.

Test Case Descriptions

CU1.1	**Create User Test - OK**
CU1.2	This test will make sure valid inputs can be accepted on user creation. Such as name, email, etc.
CU1.3	This test will go through the flow of the application using the first-time login. It will run against the system to ensure users can be created.
CU1.4	Inputs: all the inputs for creating a user need to be good, I.E appropriate username, no space in email address, etc.
CU1.5	Outputs: object representing a user profile in a valid state
CU1.6	Functional
CU1.7	Whitebox
CU1.8	Normal
CU1.9	Unit Test
CU1.10	Results: the flow works as expected, and the user is created.

CU2.1	**Create User Test - REJECTED**
CU2.2	This test will make sure invalid inputs are not accepted on user creation. Such as bad name, bad email, etc.
CU2.3	This test will go through the flow of the application using the first-time login. It will run against the system to ensure users can be created; however, it will purposefully fail to ensure the checks work as expected.
CU2.4	Inputs: all the inputs for creating a user need to be bad, I.E inappropriate username, space in email address, etc.
CU2.5	Outputs: object representing a user profile in an invalid state
CU2.6	Functional
CU2.7	Whitebox
CU2.8	Normal
CU2.9	Unit Test
CU2.10	Results: the flow works as expected, and the user inputs are rejected, and the user is not able to be created.

SP1.1 **Set Preferences**
SP1.2 This test will ensure that the preferences work as expected and the user is able to set preferences without any hiccups.
SP1.3 This test will follow the flow, running against the application. It will open the preference modal and the tester will select preferences, change preferences, remove preferences, etc.
SP1.4 Inputs: Text input of user preferences
SP1.5 Outputs: object representing a set of user preferences in a valid state
SP1.6 Functional
SP1.7 Blackbox
SP1.8 Normal
SP1.9 Unit Test
SP1.10 Results: the user will be able to manipulate preferences in any way they choose, and the modal will not throw any exceptions.

GA1.1 **Grab Activities Test**
GA1.2 This is a test to ensure the front end can correctly grab and display results from the API of things to do.
GA1.3 This test will go through the normal flow of the application, and see if the returned results are relevant to our preferences, and if items that shouldn't be returned aren't (bars for under 21)
GA1.4 Inputs: set of activities (fabricated data for testing purposes)
GA1.5 Outputs: set of activities (displayed)
GA1.6 Functional
GA1.7 Whitebox
GA1.8 Normal
GA1.9 Unit Test
GA1.10 Results: the flow will work as expected and relevant activities will be retrieved and displayed. Will ensure that activities like bars are not displayed if the user does not drink or is under and so on.

BL1.1 **Business Logic Test 1**
BL1.2 The test will test that our application will not suggest inappropriate activities to the user based on their preferences and information
BL1.3 In this test, the tester will enter an age under 18 into the application. The tester will enter any combination of preferences if these preferences would yield a result of a bar/club. The application still must not recommend going to a bar or club since the user indicated that they were a minor
BL1.4 Input: Age of the user (under 18)
BL1.5 Output: Recommendations do not include any bars or clubs
BL1.6 Normal
BL1.7 Blackbox
BL1.8 Functional
BL1.9 Unit Test
BL1.10 Results: the application will return activity results that are appropriate based on age

BL2.1 **Business Logic Test 2**

BL2.2 This test will test that our application will not suggest inappropriate activities to the user based on their preferences and information

BL2.3 In this test, the tester will enter an age over 21 into the application. The tester will enter any combination of preferences if these preferences would yield a result of a bar/club. The application should include bar options and club options since the user is of age.

BL2.4 Input: Age of the user (over 21)

BL2.5 Output: Recommendations do include bars and clubs.

BL2.6 Normal

BL2.7 Blackbox

BL2.8 Functional

BL2.9 Unit test

BL2.10 In this test, the tester will enter an age over 21 into the application. The tester will enter any combination of preferences if these preferences would yield a result of a bar/club. The application should include bar options and club options since the user is of age.

BL3.1 **Business Logic Test 3**

BL3.2 This test will test that our application will not suggest inappropriate activities to the user based on their preferences and information.

BL3.3 In this test, the tester will enter the preference that they do not want to travel more than 20 miles. After this, any combination of preferences can be inputted. The resulting recommendation list should not include any recommendations that are more than 20 miles away.

BL3.4 Input: Distance preference (20 miles or less)

BL3.5 Output: Recommendations that are within a 20-mile radius of the user

BL3.6 Normal

BL3.7 Blackbox

BL3.8 Functional

BL3.9 Unit test

BL3.10 This test will test that our application will not suggest inappropriate activities to the user based on their preferences and information.

WA1.1 **Web API Test 1**

WA1.2 This test will check whether a user's saved profile can be fetched from the database

WA1.3 A request will be sent for a user's profile, by user ID and the saved user profile will be returned

WA1.4 Input: User ID

WA1.5 Output: User's saved profile

WA1.6 Normal

WA1.7 Whitebox

WA1.8 Functional

WA1.9 Integration

WA1.10 Results: User's saved profile is returned

WA2.1 **Web API Test 2**
WA2.2 This test will check to see if activities can be found successfully
WA2.3 A query will be made for activities, using a set of preferences (could be empty), and a set of activities will be returned
WA2.4 Input: Set of user preferences (could be empty)
WA2.5 Output: List of activities
WA2.6 Normal
WA2.7 Blackbox
WA2.8 Functional
WA2.9 Unit
WA2.10 Results: Set of relevant activities are returned

WA3.1 **Web API Test 3**
WA3.2 This test will check whether past queries can be fetched from the database
WA3.3 A request will be sent to the database for queries made by a specific user, and the set of queries should be returned
WA3.4 Input: User ID
WA3.5 Output: Set of queries made in the past
WA3.6 Normal
WA3.7 Whitebox
WA3.8 Functional
WA3.9 Integration
WA3.10 Results: A set of queries (preferences set previously) should be returned

DB1.1 **Database Test 1**
DB1.2 This test will test whether a new row is added when a user finishes a query.
DB1.3 The user will enter desired inputs for the app and a new row will populate in the database.
DB1.4 Inputs: User's information in desired fields
DB1.5 Outputs: New row in database
DB1.6 Normal
DB1.7 Blackbox
DB1.8 Functional
DB1.9 Integration
DB1.10 Results: a new row is added to the database and the data row object is returned

DB2.1 **Database Test 2**
DB2.2 This test will test if the database pulls saved preferences
DB2.3 With previously saved data, the user will reopen the application and should have their preferences already populating the fields.
DB2.4 Inputs: Saved data in the database
DB2.5 Outputs: Populated fields
DB2.6 Normal
DB2.7 Blackbox
DB2.8 Functional
DB2.9 Integration
DB2.10 Results: database is queried, and specific data row object is returned

DB3.1 **Database Test 3**
DB3.2 This test will test if a user can save preferences with entering any information
DB3.3 The user will open a new query. Then enter some data into the fields, then leave all the fields blank. Then they will try to save and should not be able to.
DB3.4 Input: Blank page, save button pushed
DB3.5 Output: Error message
DB3.6 Abnormal
DB3.7 Blackbox
DB3.8 Functional
DB3.9 Unit Test
DB3.10 Results: Data will be attempted to be saved and should throw an exception

Test Case Matrix

	Normal/ Abnormal	**Blackbox/ Whitebox**	**Functional/ Performance**	**Unit/ Integration**
CU1	Normal	Whitebox	Functional	Unit
CU2	Normal	Whitebox	Functional	Unit
SP1	Normal	Blackbox	Functional	Unit
GA1	Normal	Whitebox	Functional	Unit
BL1	Normal	Blackbox	Functional	Unit
BL2	Normal	Blackbox	Functional	Unit
BL3	Normal	Blackbox	Functional	Unit
WA1	Normal	Whitebox	Functional	Integration
WA2	Normal	Blackbox	Functional	Unit
WA3	Normal	Whitebox	Functional	Integration
DB1	Normal	Blackbox	Functional	Integration
DB2	Normal	Blackbox	Functional	Integration
DB3	Abnormal	Blackbox	Functional	Unit