**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Weather Web Application: Test Plan**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**By Charbel Marche, Dylon Garrett, and Rachel Mooney**

**March 29, 2018**

**Table of Contents:**

Table of Contents……………………………………………………………………………….....1

1. Introduction……..…………………………………………………………………………........2

1.1 Purpose...…………………………………………………………………………...….2

1.2 Scope……………………………………………………………………………..…....2

1.3 References………………………………………………………………………..…....2

1.4 Overview…………………………………………………………………………..…..2

2. Project Description………….……………………………………………………………..…....3

2.1 System Overview...…………………………………………………………….......….3

2.2 Client Characteristics………...……………………………………………………......3

2.3 User Characteristics……………….………………………………………………......4

2.4 Functional Requirements….………....……………………………………………......4

2.5 General Constraints…………………………………………………………………...5

3. Test Plan………………...……………………………………………………………………...5

3.1 Testing Strategy……………………………………………………………….……...5

3.2 Testing Resources and Staffing………….…………………………………...…........6

3.3 Test Work Products………….……………..………………………………...……....6

3.4 Test Record Keeping……..………………………………………………….….........7

3.5 Test Schedule……..…………………………………………………………..….......7

4. Test Procedure…………………………………...……………………………………..….......8

4.1 Unit Tests……………………………………………………………….…………....8

4.2 Integration Tests………….………………………………...…………………….......9

4.3 Validation Tests………….………….……..………………………………….……...11

4.4 System Tests……..………………………………………………….……..................14

5. Appendices……………………………………………………………………………..….......16

5.1 Glossary……………………………………………………………………….….….16

5.2 Author Information………….……………………………………..………….….….17

5.3 Additional documents………………………………………..…………...…...……..17

**1. Introduction**

**1.1 Purpose**

The purpose of this document is to provide pertinent information about the Weather Web Application and to specify the different tests that need to be done in order to verify that the application is working properly and that it fulfills all the requirements specified by the client. It contains information such as the details about the application, its functionality and who will use it. It will also list all of the requirements necessary for the application and how these requirements and the system will be tested. The testing information includes the testing strategy, schedule, test procedures and a thorough break down of each test. The document is intended for the implementation team, the testing team, the client, and any developers that will maintain the project in the future.

**1.2 Scope**

The Weather Web Application is an application that displays a current and a historical weather forecast for a specific date and location. The client, Dr. Zeitz, wants to be able to see the average temperature, precipitation and other information for a specific date over a ten year span. The application returns information from a historical and current weather database when the user types in a location and date in the search bar. When the user types in a future date that is greater than ten days away, the application will return a prediction based off of what the weather has been like on that day in the past. The goal for the Weather Web Application is to provide Dr. Zeitz with a reliable weather source so that she and other users are able to track future weather forecasts more accurately.

**1.3 References**

There are no references. Please refer to section **5.3 Additional Documents** to access the link provided by the client.

**1.4 Overview of the Remainder of the Document**

The following four sections in this document provide more information about the product itself, the test plan, procedures that will be used in testing and ending with additional resource information. Section two describes the product and how it can be used in greater detail. It introduces the client and their intentions for the product, as well as who will use the application. This section will also go into detail about the functional requirements and the constraints present within the application. Section three provides details about the test plan for the product. It will describe the overall testing strategy, the resources required, how test results will be recorded and the test schedule for the application. The test schedule is broken down so that it is easily understood which tests will be completed by either the implementation team or the testing team. Section four delves into the test procedure that will be used, containing information about test tactics as well as test cases for the application. Section five concludes the document with the appendix, author information, and any additional documents provided by the client in regard to the product.

**2. Project Description**

**2.1 System Overview**

The Weather Web Application requested by Dr. Zeitz is an online application that will provide the user with an easy-to-use interface by which to ascertain the weather for a specific date. It will not only be able to display the current weather forecast, but also make predictions about any date up to a year into the future. The option to become a registered user on the application will be present as well, providing frequent users will some added conveniences.

The interface of the application will be simplistic and straightforward, maximizing user friendliness. In order to successfully conduct a search, a user need only know a date and location, designated by state and city or zip code. Once the user has identified a date and location by which to search and submitted the information, the results will be displayed in clearly marked sections and include helpful visual aids, such as graphs, to depict the information. The two different types of graphs that will be used to depict different types of information are a bar graph and a pie graph. The bar graph will show the average temperature for each half decade that the Dark Ski API has information about at the given location, while the pie graph will display the historical odds of different types of weather (sunshine, clouds, snow, rain) at the given location.

To aid in the planning of future events, the Weather Web Application will provide the user with an approximation of the weather based upon a user specified date and a location. This will be accomplished by analyzing the historical weather data for that location and computing an average of the results in order to provide a future forecast prediction on the chosen date. Depending upon the chosen location, the historical data used in the analysis will range from the past five to fifty years. The future forecast prediction will include information about the highest and lowest temperature on record for that day, average temperature, and precipitation. As mentioned above, visual aids will be utilized to show past weather trends and provide the user with more information about their selected search. For searches up to ten days into the future, the current weather forecast for that location will be displayed as well.

Any user may become a registered user on the Weather Web Application. In doing so, a user gains the additional functionality of being able to save and export the information from a search. The ability to save a particular search allows a registered user to forgo repeating a particular search numerous times. The registered user need only to log in to access a search they have saved and thus easily receive check updates about the weather forecast for that particular search. The ability to export the data return from a search is also a feature only available to registered users. Once a search has been made, the registered user will have the ability to choose to export the data results. When exporting, the data will retain an organized format and the same ease of readability found when content from any search is returned.

**2.2 Client Characteristics**

The client for the Weather Web Application is Dr. Zeitz, an Assistant Professor of Computer Science at the University of Mary Washington. Dr. Zeitz requested this application because she previously had tried to find a service to predict the weather on her sister’s planned wedding day. She was not able to find an application that would allow her to conveniently access this information. She resorted to having to use an almanac to make predictions by hand. Her predictions ended up not accurately predicting the weather for her sister’s wedding day. As a result, the user believes that an application that would do all that work for the user by presenting the historical weather information in a convenient matter would be greatly useful. She also believes the additional feature of displaying the current forecast would be extremely helpful when the date being planned for is within a few days time.

**2.3 User Characteristics**

The intended users of the Weather Web Application are not limited to Dr. Zeitz exclusively, but will be available to anyone who is looking for a weather forecast. However, it is primarily focused towards users who desire to know a weather prediction for a date that exceeds past the range of the 10 day current weather forecast. Allowing for the wide range of potential users, the application will be designed in a way that provides easy use and readability for anyone, regardless of if a user is very familiar with computers or not. While Dr. Zeitz is very familiar and comfortable with software and coding, users are not required to have any understanding of computer science to utilize the application. The application will have online capabilities so that users can utilize the application from a desktop, laptop, or through mobile devices.

**2.4 Functional Requirements**

The functional requirements are features of the application that the system should provide to the user throughout its use.

2.4.1 The system must allow the user to search for the average weather data for a given location on a specific day, going back at least 2-5 years, but preferably up to 50 years back.

2.4.2 The system should have a drop down calendar to allow the user to choose the date, with the current date being highlighted in one color, and the chosen date being highlighted in another color.

2.4.3 The system should accept a location in the form of a zip code, city or state.

2.4.4 The location and date fields should have default text prompting the user to enter the appropriate information.

2.4.5 The system should provide instructional prompts for the user on how to enter information.

2.4.6 The system must have a submit button that should only show results if there are valid inputs for date and/or location.

2.4.7 If there are not valid inputs for date and/or location upon submitting the search, the system must tell the user what (date and/or location) they entered that was not valid.

2.4.8 The system must show the percent of days it has precipitated on the chosen day, no matter how close or far, as well as the average temperature for that day.

2.4.9 If the date selected is with in 10 days of the current date, the forecast for the weather at that location must also be shown.

2.4.10 The system should have a bar graph showing the average temperature for each half decade that the system has information about at the given location.

2.4.11 The system should have a pie graph displaying the historical odds of different types of weather (sunshine, clouds, snow, rain) at the given location.

2.4.12 The system should allow the user to export data to any file format of the implementation team’s choice, and this feature should only be available to registered users.

2.4.13 The system should allow users to register, log in, and log out.

2.4.14 The system should remember past searches for registered users.

**User Stories:**

1. As a newly engaged couple, we would like to see the historical weather information for our planned wedding date which is more than 10 days in the future so that we can know if it is reasonable to have the wedding outdoors or not.
2. As an engaged couple with a wedding within the next 10 days, we would like to see the forecast for our wedding date is so that we can decide whether to keep the wedding outdoor or rearrange our plans.
3. As a user who has trouble visualizing data, I want to see return weather data in the form of graphs and pie charts so that I can better understand the predicted forecast.
4. As a user I want to become a registered user so that I can save my search results online and access them quickly.
5. As a registered user with a planned event in the far future, I would like to export graphical information about the planned events historical data and forecasts (if in the next 10 days) so that I can share the results with my friends and have a copy of the information while offline.

**2.5 General Constraints**

2.5.1 The time it takes for the system to return weather information from the API based off of what the user searches should be no more than 5 seconds.

2.5.2 The website needs to be intuitive. The user should be able to navigate through website without outside instruction

2.5.3 The hosting needs to be free for the application.

**3. Test Plan**

**3.1 Testing Strategy**  
 The Weather Web Application will be thoroughly tested by both the implementation team, and a testing team who is not familiar with the application’s code before hand. The implementation team will handle the majority of the unit testing and integration testing, while the testing team will handle the validation testing and overall system testing. Below are the different types of testing that will be done on the Weather Web Application. The tests to be performed for each category of test type will be enumerated in detail in a later section.

The unit testing will test the different functionalities of all the components of the weather web application using the white box method. This will test several functionalities like handling user inputs, displaying information on user inputs, adding data to the different tables in the database, and receiving proper data from the Dark Sky API. If possible, the different components will be isolated to minimize dependencies on other components, and the tester will programmatically test each unit by simulating different inputs, outputs, actions, and situations and ensuring that the application reacts properly. The unit testing has and will be continue to be done by the implementation team as they add new components to the Weather Web Application.

The integration testing will test the interactions between components being integrated in order to expose any defects in the interaction. This is aimed to test the integration of the smaller components first before moving on to larger and larger component integration until system testing is performed. Generally it is used to test components made by different developers as those components are integrated into a consolidated whole. Testing in that regard will not be applicable because the implementation team utilizes pair programming while working on just one environment. The application is always one piece that is expanding by combined effort and not by any integration of different pieces. Since this is the case, integration testing will be used to verify that: the application runs on the host site, the graphical user interface interacts with the weather and user databases and returns the correct data, and that a search returns the correct data from the Dark Sky API. Integration testing will be done by the implementation team as it becomes applicable with the addition of the aforementioned components.

The validation testing will test that each of the requirements, described in sections **2.4 Functional Requirements** and **2.5 General Constraints**, have been fulfilled. The tester will use the black box method of testing to do so.

The system testing will test the overall functionality of the Weather Web Application by using a black box approach as well. In essence, the tester will pretend to be a regular user, and attempt to walk through a series of different user stories in an attempt to attain current or historical weather information. The tester will try both roles of user types, registered and non-registered, and they will use the application for a whole interaction as that particular user with a specific goal and set of actions in mind.

**3.2 Testing Resources and Staffing**

Since the testing team is only carrying out black box testing, specifically for all system testing and validation testing, the testing team will merely be provided with a link to the Weather Web Application website. They will not need to set up any environment to do any testing. However, it is recommended to test on Google Chrome since that is the browser targeted by the implementation team. The implementation team is responsible for all other testing, such as unit testing and integration testing.

**3.3 Test Work Products**

The products produced as a consequence of each testing strategy will be four different spreadsheets, one for each individual test type. Making a seperate spreadsheet for each type of test type will help to keep the tester from accidentally jumping from one kind of testing into another. Besides containing information about the various tests to be performed, the spreadsheet will also include a couple additional columns for recording results. One column will be used to record if the test was successful or not, and the other column will be used for recording any bugs found when testing. These column will be filled out by the tester and include the date on which the test was performed. This will help to keep testing and result information together and will help to keep test results in a format that is easily referenced.

**3.4 Test Record Keeping**

Due to the lack of a central work location for this project and the input from the members of the implementation team at different times and locations, all test recording will be done by utilizing Google Sheets online. This will provide easy accessibility for the implementation team at all times. When recording testing results from either a member of the implementation team or the testing team into Google Sheets, the test results will be evaluated. Any test that was marked as not having provided the expected test results will be recreated and evaluated by the bug or error information supplied by the tester. Once all the problems have been identified, all features that produced a bug will be put through extensive testing of various types in order to identify and solve the problem. The links below are the spreadsheets that will be used during testing by both the implementation team and testing team.

[Unit Tests Spreadsheet](https://docs.google.com/spreadsheets/d/1jBLW6GO9I9F1o0yEXtcGnKHDOxRuCYtx0bgOqvKdatM/edit?usp=sharing)

[Integration Tests Spreadsheet](https://docs.google.com/spreadsheets/d/1f-SdwuhGnsT7WIflExkuZTugvO3djTKy2Mlj9YcXm9A/edit?usp=sharing)

[Validation Tests Spreadsheet](https://docs.google.com/spreadsheets/d/1kRP8JeCrnirK2hOBr-zS_7XKlbAbaiFZEhtLBcHMglA/edit?usp=sharing)

[System Tests Spreadsheet](https://docs.google.com/spreadsheets/d/1axHYVTB3bpv4VggsaTxFKE5uAXbY1KLB_RqEUOWSQfw/edit?usp=sharing)

**3.5 Test Schedule**

The unit tests and integration tests will be completed by the implementation team. The validation tests will be completed by the testing team. In order to minimize any bugs in the application compounding on one another, the implementation team will be carrying out unit tests and integration tests continuously as code as produced. Since this is the procedure being used to test components as they are developed, the schedule for the unit and integration tests will coincide with the dates over which tasks for the project will be completed. The tests for each task will be completed by the specific task end date and any issues will be corrected before that task is marked complete. In doing this, the testing that the implementation team carries out will be performed in such a way that each test must be completed before moving on to the next task.

**Unit Tests**

**3.5.1** Test number 4.1.1 date to be completed is 4/11/18.

**3.5.2** Test number 4.1.2 date to be completed is 4/13/18.

**3.5.3** Test number 4.1.3 date to be completed is 4/13/18.

**3.5.4** Test number 4.1.4 date was completed on 3/15/18.

**3.5.5** Test number 4.1.5 date was completed on 3/15/18.

**3.5.6** Test number 4.1.6 date to be completed is 4/12/18.

**3.5.7** Test number 4.1.7 date to be completed is 4/12/18.

**3.5.8** Test number 4.1.8 date was completed on 3/15/18.

**3.5.9** Test number 4.1.9 date to be completed is 4/18/18.

**3.5.10** Test number 4.1.10 date to be completed is 4/13/18.

**Integration Tests**

**3.5.11** Test number 4.2.1 date was completed on 3/1/18.

**3.5.12** Test number 4.2.2 date to be completed is 4/5/18.

**3.5.13** Test number 4.2.3 date was completed on 3/14/18.

**3.5.14** Test number 4.2.4 date was completed on 3/15/18.

**Validation Tests**

**3.5.15** The validation tests will be completed by the testing team on 4/10/18 and 4/12/18.

**4. Test Procedure**

This section contains a subsection for each of the different types of tests (unit, integration, validation, system), the methods used for these types of tests, and who will be carrying out these tests. Each subsection will contain a detailed breakdown of the different tests of that type.

**4.1 Unit Tests:**

The procedure used for unit testing will be white box testing. This testing will be done by the implementation team.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Number** | **Related to Requirement** | **Subsystem** | **Purpose** | **Test Case Data/Instructions** | **Expected Results** |
| 4.1.1 | 2.4.13 | Register users. | Test user registration functionality. | Create username:  user  Create password:  password | Successful registration and new entry in “users” table. |
| 4.1.2 | 2.4.13 | Login. | Test the password. | Username:  user  Password:  password | Successful login. |
| 4.1.3 | 2.4.13 | Logout. | Test logout functionality. | Register as a user and login. Click logout button at the top of the screen. | Successful logout. |
| 4.1.4 | 2.4.1  2.4.3  2.4.4 | Location field. | Test location field . | Valid 5 digit zip code or City and State:  22401  or  Fredericksburg, VA  or  Fredericksburg,  Virginia | System accepts  the location entered by the user. |
| 4.1.5 | 2.4.1  2.4.2  2.4.4 | Date field  including drop down calendar. | Test date field. | Valid date:  12/20/19  or  Choose a date using calendar tool. | System accepts the date entered by the user. |
| 4.1.6 | 2.4.10 | Bar graph. | Test to see accurate bar graph is produced. | Based from the users searched location and date results. | System produces a bar graph displaying the average temperature for each half decade for given location. |
| 4.1.7 | 2.4.11 | Pie graph. | Test to see accurate pie graph is produced. | Based from the users searched location and date results. | System produces pie graph displaying the historical odds of different types of weather for given location. |
| 4.1.8 | 2.4.6 | Search button. | Test location and date inputs. | Valid location:  22401  or  Fredericksburg, VA  Valid date:  12/20/19 | Location and date is accepted and search is successful. |
| 4.1.9 | 2.4.12 | Export data. | Test export data feature. | Register as a user and provide valid location and date.  Select the export button. | User data is exported to specific file format. |
| 4.1.10 | 2.4.14 | Saved searches. | Test searches. | Register as a user or sign into an existing account. Enter a valid location and date to search. Click saved searches button at the top of the screen. | All user searches saved successfully. |

**4.2 Integration Testing:**

The procedures used for integration testing will be a combination of both black box and white box testing. This testing will be done by the implementation team.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Number** | **Purpose** | **Test Case Data** | **Expected Results** |
| 4.2.1 | Test if application is running on the host site. | Navigate to the domain that the application is being hosted on. | The application should be accessible at the domain. |
| 4.2.2 | Test input made on application interface will return desired information from **user** database. | In the homepage of the website add a line attempting to create a user in the users table of the users database.  These lines of code should be added in the mainIndex() function:  connection = connectToDB()  query = "INSERT into users (username, password) VALUES (intTestUser, pwd);"  curr.execute(query)  connection.commit() | The user ‘intTestUser’ should be in the users table when tester runs this query on database:  “select \* from users where username=’intTestUser’; |
| 4.2.3 | Test input made on application interface will return desired information from **world** database (contains cities, states, zips, lat, and long). | In the homepage of the website add a line attempting to select latitude and longitude from the cities table of the world database.  These lines of code should be added in the mainIndex() function:  connection = connectToDB()  query = "SELECT lat, lng FROM cities where city=’King George’ and (state\_id=’VA’);"  curr.execute(query)  result = curr.fetchone()  print(“The result is ” + result) | The system should print the following data:  Lat = 38.26818370  Lng = -77.18442220 |
| 4.2.4 | Does searching using the Dark Sky API return the correct data user specified date and location? | Enter valid location and date as below:   * Location: King George, VA * Date: August 1, 2018 | System accepts the valid input and returns the weather results matching the specified location and date. This data should match the data at this link <https://darksky.net/details/38.2786,-77.1754/2018-8-1/us12/en>. |

**4.3 Validation Testing:**

The procedure used for validation testing will be black box testing. This testing will be done by the testing team.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Number** | **Related to Requirement** | **Test Case Data/Instructions** | **Expected Results** |
| 4.3.1 | 2.4.1 | Enter location and date as:   * Location: King George, VA * Date: August 1, 2018   Then hit the search button. | The historical weather data that is available at this link <https://darksky.net/details/38.2786,-77.1754/2018-8-1/us12/en> should be displayed (in a different format but the same information). |
| 4.3.2 | 2.4.2 | None, select the date field. | There should be a drop down calendar allowing the user to only select future dates. The current date should be highlighted in blue and the chosen date should be highlighted in red. |
| 4.3.3 | 2.4.3 | Enter location in 3 formats:   * King George, VA * King George, Virginia * 22485   Along with the date as:   * August 1, 2018   Then hit the search button after each of the three different inputs. | Each location input should be valid and return the same data as in this link  <https://darksky.net/details/38.2786,-77.1754/2018-8-1/us12/en>. |
| 4.3.4 | 2.4.4 | None, navigate to site and observe the location input field and date input field. | The location field should say “Enter a city and state or a zip code” and the date field should have a mm/dd/yyyy prompt. |
| 4.3.5 | 2.4.5 | None, navigate to site and observe the area around the location input field and date input field. | The system should provide detailed instructions on the format to enter for the location and date fields, and how to search for weather information. |
| 4.3.6 | 2.4.6 | Enter 3 sets of invalid inputs and hit the search button after each step:   * No input in location and 08/01/2018 in date. * Enter “King George, VA” in location and no date input in the date. * Enter “Kin Gorge” in location and no input in the date field. | The system should not provide any weather information for any of these searches. |
| 4.3.7 | 2.4.7 | Enter 3 sets of invalid inputs and hit the search button after each step:   * No input in location and 08/01/2018 in date. * Enter “King George, VA” in location and no date input in the date. * Enter “Kin Gorge” in location and no input in the date field. | In the case of the first input, the system should display an error message that says to enter a date in “city, state” format or enter a 5 letter zip code.  For the second input, the system should display an error message that states that the user needs to select a valid future date.  For the third input, the system should display an error message that says to enter a date in “city, state” format or enter a 5 letter zip code and another error message that states that the user needs to select a valid future date. |
| 4.3.8 | 2.4.8 | Enter location and date as:   * Location: King George, VA * Date: August 1, 2018   Then hit the search button. | The expected outcome should be a display showing the percent chance of precipitation and average temperature highs and lows for that day. This is the same data that is found at this link <https://darksky.net/details/38.2786,-77.1754/2018-8-1/us12/en>. |
| 4.3.9 | 2.4.9 | Enter location and date as:   * Location: King George, VA * Date: Tomorrow’s date   Then hit the search button. | The weather data returned should include forecasts provided by searching at this link <https://darksky.net/forecast/38.2786,-77.1754/us12/en> for the same location and date. On this link scroll down below the map and select tomorrow’s date. |
| 4.3.10 | 2.4.10 | Enter location and date as:   * Location: King George, VA * Date: August 1, 2018   Then hit the search button. | There should be a bar graph displayed on the results page showing the average temperature for each half decade of data provided by the Dark Sky API. |
| 4.3.11 | 2.4.11 | Enter location and date as:   * Location: King George, VA * Date: August 1, 2018   Then hit the search button. | There should be a pie chart displayed on the results page showing the historical odds of different types of weather at that location on that day. |
| 4.3.12 | 2.4.12 | Log in with these credentials:   * Username: user * Password: password   Enter location and date as:   * Location: King George, VA * Date: August 1, 2018   Then hit the search button.  After the search returns results, hit the export button at the bottom of the page. | The results should be downloaded to the “Downloads” folder on the computer, with all the same data present in the file that is present on the website. |
| 4.3.13 | 2.4.13 | Select the “Sign Up” button from the menu and enter these credentials:   * Username: newtestuser1 * Password: Password1   Select the “Create an Account!” button.  Select the “Log In” button from the menu and enter the username and password from above.  After logging in, select the “Log Out” button on the menu. | After selecting “Sign Up” and entering the new user account credentials into the field the account should be created. This can be tested by seeing if one can log in with these credentials. Once the tester has logged in as the user with the given credentials, the only options on the menu should be “Log out”, “View Past Searches”, and “About”, showing that the user is logged in. After selecting the “Log Out” button, the user should be logged out and the menu should have “Log In”, “Sign Up”, and “About” buttons once more. |
| 4.3.14 | 2.4.14 | Log in with these credentials:   * Username: user * Password: password   Enter location and date as:   * Location: King George, VA * Date: August 1, 2018   Then hit the search button.  After search is complete, select the “View Past Searches” button.  Select the website logo to return to the home page.  Enter location and date as:   * Location: Fredericksburg, VA * Date: July 1, 2018   Then hit the search button.  After search is complete, select the “View Past Searches” button. | Upon selecting the “View Past Searches” button the first time, the latest search of King George, VA and August 1, 2018 should be present.  After searching for Fredericksburg, VA and selecting the “View Past Searches” button the second time, the second search of Fredericksburg, VA on July 1, 2018 along with the first search of King George, VA on August 1, 2018 should be present. |
| 4.3.15 | 2.5.1 | Enter location and date as:   * Location: King George, VA * Date: August 1, 2018   Then hit the search button.  Count on a timer the seconds it takes for the results to display. | The application should return weather data in less than 5 seconds. |

**4.4 System Testing**

System testing will be done using the black box method, and will be carried out by the testing team. The system tests are meant to be done in order.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Number** | **Purpose** | **Test Case Data/Instructions** | **Expected Results** |
| 4.4.1 | To test end-to-end usage for a user that wants to create an account and search weather data for a location and a date not within the next 10 days and download a file containing the resulting data. | Navigate to the website and select the “Sign Up” button and enter these credentials:   * Username: newtestuser2 * Password: Password2   Select the “Create an Account!” button.  Enter location and date as:   * Location: King George, VA * Date: August 1, 2018   Then hit the search button.  After the search returns results, hit the export button at the bottom of the page.  Select the “Logout” button from the menu. | The username “newtestuser2” should be registered and after registration the user should be logged in. After searching for the weather information the resulting weather data should be displayed, matching the weather data at this link <https://darksky.net/details/38.2786,-77.1754/2018-8-1/us12/en>.  After exporting the same weather data should be downloaded to the user’s computer.  After logging out the user should no longer be logged in. |
| 4.4.2 | To test end-to-end user for a registered user that wants to look at their past searches, and take the location for the most recently searched location and search for the weather data for a date within the next 10 days, and then download a file containing the weather data. | Navigate to the site.  Log in with these credentials:   * Username: user * Password: password   Enter location and date as:   * Location: King George, VA * Date: Tomorrow   Then hit the search button.  Select the “Logout” button from the menu.  Log in with the same credentials.  After search is complete, select the “View Past Searches” button.  Enter the same location and date as above.  After the search returns results, hit the export button at the bottom of the page.  Select the “Logout” button from the menu. | The user should be logged in as “user”, then view results of their most recent search. Logging out should log the user out, and when the user logs back in, the most recent search to King George, VA for tomorrow should be in the user’s past searches. After searching again, the results should be displayed, matching the data at <https://darksky.net/forecast/38.2786,-77.1754/us12/en>  when searching for the same date.  After exporting the same weather data should be downloaded to the user’s computer.  After logging out the user should no longer be logged in. |
| 4.4.3 | To test end-to-end usage for a user that is not interested in becoming a registered use, but wants to see weather data for a location at two dates, one within the next 10 days and one beyond the the next 10 days. | Enter location and date as:   * Location: King George, VA * Date: Tomorrow   Then hit the search button.  Enter location and date as:   * Location: King George, VA * Date: August 1, 2018   Then hit the search button. | The first search should return the forecasted weather for tomorrow in King George, VA which should be the same as the data at <https://darksky.net/forecast/38.2786,-77.1754/us12/en>  when searching for the same date.  The second search should result in data matching the weather data at this link <https://darksky.net/details/38.2786,-77.1754/2018-8-1/us12/en>. |

**5. Appendices**

**5.1 Glossary**

The terms used in this document are explicitly documented below:

**API:** Application Programing Interface, used in the document to refer to potential API’s from outside applications that provide free historical and forecast weather information.

**Application:** The Weather Web Application that is used to search for a weather forecast.

**Black Box Testing:** A testing approach that does not allow the tester to look at the code underneath, forcing the tester to focus on the user experience.

**Bootstrap Template:** A pre-designed framework that is used as a tool to assist in the development web applications.

**Bug:** A bug in software is an error or fault in a computer program or system that causes incorrect or unexpected results.

**NOAA:** National Oceanic and Atmospheric Administration. It is a government agency whose mission is: “To understand and predict changes in climate, weather, oceans, and coasts, to share that knowledge and information with others, and to conserve and manage coastal and marine ecosystems and resources.” <http://www.noaa.gov/>

**Non-registered User:** A user who has not made an account and become a registered user. The only actions available to a non-registered user are to make a search and become a registered user.

**Registered User​:** ​A user who has made an account. A registered user enjoys additional functionality such as saving and exporting searches.

**System**: The device that acts as the platform for this application, the one the user is interacting with.

**White Box Testing:** A testing approach in which the tester has access to the code and can design the test with the code in mind.

**5.2 Author Information**

The co-authors of this document are Charbel Marche, Dylon Garrett and Rachel Mooney.

Charbel authored sections: 1.1, 2.2, 2.4, 3.1, 3.2, 4.3, and 4.4.

Dylon authored sections: 1.3, 1.2, 2.5, 3.5, 4.1 and 5.3

Rachel authored sections: 1.4, 2.1, 2.3, 3.1, 3.3, 3.4, 4.2, 5.1, and 5.2.

**5.3 Additional Documents**

Dark Sky (<https://darksky.net/dev>) is a weather API recommended to us by the client. The API allows for 1,000 free calls per day. After 1,000 calls it will cost $0.0001 per call.

