Test Plan and Cases (TPC)

Field Progress Webapp

Team #4

Akanksha Diwedy	Operational Concept Engineer, Developer, Tester
Kevin Grimes	Quality Focal Point, IIV&V, Website Maintainer, Developer, Tester
Aishwarya Joisa	Feasibility Analyst, Developer, Tester
Mayank Kulkarni	Requirements Engineer, Developer, Tester
Madhavi Shantharam	Life Cycle Planner, Developer, Tester
Uche Uba	Project Manager, Developer, Tester
Sahithi Velma	Software Architect, Developer, Tester

December 9, 2019

Version History

Date	Author	Version	Changes made	Rationale
12/09/19	KG	1.0	• Initial release	• Deliverable for as-built package

Table of Contents

Table of Tables

Table of Figures

1. Introduction

The purpose of the Field Progress Webapp (FPWA) Test Plan and Cases (TPC) document is to describe which procedures need to be undertaken to ensure that the delivered application meets all of the requirements of the success-critical stakeholders while producing reliable, accurate results. Testing throughout the project's lifecycle has primarily been at a manual level; however, as the project has evolved, more and more of the testing procedures have been automated using software testing tools.

There are three primary components of the FPWA application that need to be tested:

- The backend module,
- The frontend module, and
- Integration of the backend and frontend modules

In some cases, the tests described in the following document may overlap other test cases in terms of what they are testing.

The backend module, a Django server written in Python, is tested using Django's built-in testing framework, which itself is built upon Python's internal unittest framework. The frontend module is written as a React application, and is tested using the Selenium web browser automation tool and a few other technologies. The integration of the frontend and backend modules, in contrast to the first two modules, is performed manually by a combination of interacting with the web app and running HTTP requests using the API testing tool Postman.

Testing each module according to this document's specification will ensure that the application behaves as expected, while implementing all requirements outlined by the success-critical stakeholders.

2. Test Strategy and Preparation

Test cases are developed to verify that each feature that exists within the application at any given point in time works according to the requirements provided by success-critical stakeholders. This approach helps to ensure that implementation of new features does not have any unintended side effects on existing features.

However, because of the intense focus on ensuring that high-level features of our application work according to the stakeholders' specification, some lower-level features are not tested as intensely as the higher-level features. For example, while we test to ensure that valid slices of turf are returned to the user given valid inputs, we do not test lower-level Python functionality, such as list concatenation. We focus our efforts on testing the high-level features which we control, and assume that the lower-level features behave as expected. Should the team find itself without additional tasks to complete, tests which verify the functionality of such lower-level features may be implemented.

In general, the following procedure is followed when creating a new test case:

1. Enumerate the various possible valid inputs and outputs that may be passed into and retrieved from the feature respectively,
2. Enumerate the various possible invalid inputs,
3. Determine which function(s) of which module(s) need to be tested to wholly test the feature,
4. Write a test case for each valid input, and verify that it produces a valid output,
5. Write a test case for each invalid input, and verify that it produces a sensible error, and
6. Document how to run the test case

In order to ease the burden on the individuals executing the test cases, efforts have been made to automate as much as possible. That being said, there are still some steps which must be taken to prepare one's workstation to execute the tests. These are described in detail in the following sections.

2.1 Hardware preparation

The system has been successfully tested on the following hardware system:

- MacBook Pro (15-inch, 2016)
- Processor: 2.9 GHz Intel Core i7
- Memory: 16 GB 2133 MHz LPDDR3

- WiFi and/or ethernet connection

However, systems with lower hardware specifications are expected to perform sufficiently as well. Additionally, the system is expected to work on MacOS, Linux, and Windows operating systems.

2.2 Software preparation

2.2.1 Backend module

The backend component of the application is implemented as a Django web server. The following software packages need to be downloaded:

- Python 3.7, and
- Pip

Once both packages have been downloaded, a new Python virtual environment needs to be created:

```
$ pip3 install virtualenv
$ python3 -m venv venv
$ source venv/bin/activate
```

Next, install all of the Python requirements from the `requirements.txt` file:

```
$ pip3 install -r requirements.txt
```

Your environment is now ready to begin testing the backend module. Note that the Django server itself does not need to be running in order to perform these backend tests. Additionally, ensure that you are within the virtual environment you created when you run the following tests: failing to do so may result in some unexpected "module not found" errors.

2.2.2 Frontend module

The frontend component of the application is implemented as a React web application. The following software packages need to be downloaded:

- Google Chrome version 78.*,
- node version 13.2.0, and
- npm version 6.13.1

Once these packages have been downloaded and installed, navigate to the `frontend/FP_Frontend` subdirectory of the project, and run the following to install all additional software packages:

```
$ npm install
```

Your environment is now ready to begin testing the frontend module of the application.

2.2.3 Integration of frontend and backend modules

Assuming that both the backend and the frontend are running on a web server, the integration between the two can be achieved. The following packages are required:

- Google Chrome version 78.*, and
- Postman version 7.13.0

Once downloaded and installed, the environment is ready to begin testing the integration between the frontend and backend modules of the application.

2.3 Other pre-test preparations

Procedures for starting and linking the frontend and backend modules of the application are described in detail in the User Guide document for the application and is not covered here. Test data for testing the backend component is included with the application, and plugging in new data sets is also covered in the User Guide.

Finally, also included with the code is a Postman Collection, which may be imported into your local copy of the Postman API testing application.

To import the Postman collection, first open the Postman application and bring it to the forefront. Next, navigate to the "File" drop down menu, and click "Import…".

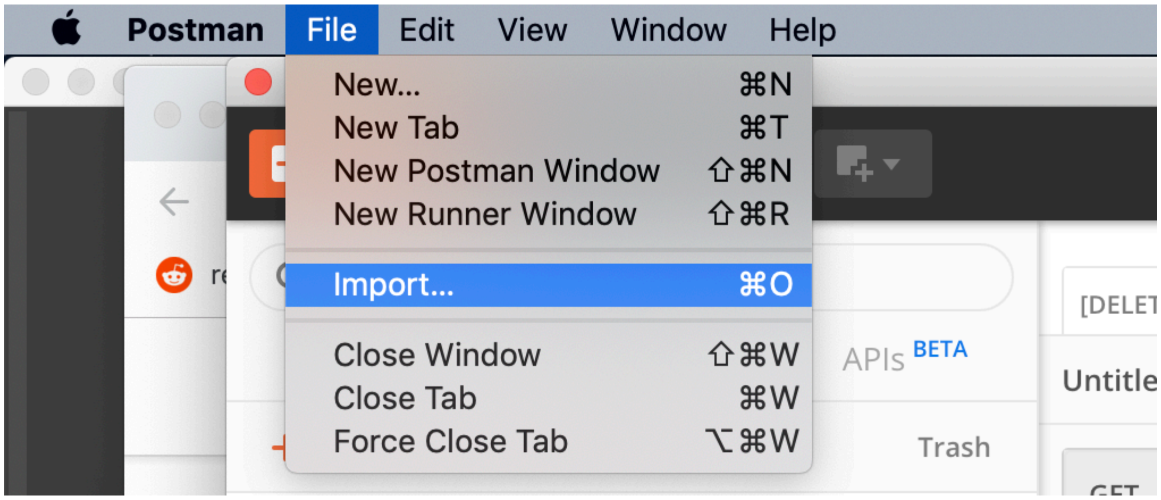

Figure 1: Inserting a new Collection into Postman.

Next, select the collection `Field Progress.postman_collection.json` from wherever you downloaded the application code, and open it.

You should see a new Collection appear in the left of the Postman interface called "Field Progress."

2.4 Requirements Traceability

Table 1: Requirements Verification Matrix

Requirement ID	Verification Type	Test Case ID
WC_5507 Load a voter list and produce a clustered voter list	Testing	TC-01-01, TC-01-02, TC-01-03, TC-01-04, TC-01-05
WC_5508 Number of volunteers, time per volunteer, time per conversation, and likelihood per conversation as parameters	Testing	TC-02-01, TC-02-02, TC-02-03
WC_5488 Accept as input a variable number of volunteers	Testing/Demonstration	TC-02-01, TC-02-02, TC-02-03, TC-04-01, TC-04-01, TC-04-02, TC-05-01, TC-05-02, TC-05-03, TC-05-04, TC-06-01, TC-06-02

WC_5487 Render a map with voters placed as points, and line(s) indicating the best route(s) for the volunteers to take. Groups of voters shall be represented with colors	Demonstration	TC-03-01, TC-03-02, TC-03-03
WC_5696 Clustered voter data should be efficiently visualized on an interactive map	Demonstration	TC-03-01, TC-03-02, TC-03-03

3. Test Identification

The following sections outline more information about each test case described in the requirements verification matrix.

3.1 TC-01 Loading voter files into backend

3.1.1 Test Level

Software item level

3.1.2 Test Class

- Functionality test, and
- Erroneous test

3.1.3 Test Completion Criteria

The following breaks down the criteria that must be met for each test case to be completed:

- TC-01-01 Fail to load voter file when it does not exist
- TC-01-02 Fail to load voter file when it is corrupt
- TC-01-03 Fail to load voter file if reference to it is null
- TC-01-04 Successfully load voter file with two volunteers
- TC-01-05 Successfully load voter file with five volunteers

3.1.4 Test Cases

Table 2: TC-01-01 Fail to load voter file when it does not exist

Test Case Number	TC-01-01 Fail to load voter file when it does not exist
Test Item	If the individual setting up the application for a given campaign fails to provide a file containing the voters to consider canvassing, then the application shall render an error, but continue to function.
Test Priority	Should have

Pre-conditions	• Python virtual environment has been enabled, and • Required Python modules have been downloaded via pip
Post-conditions	An error message shall be returned to the user indicating that an error has occurred
Input Specifications	Run the following command from within the base application directory: `$ python3 manage.py test -k test_voter_csv_does_not_exist`
Expected Output Specifications	"OK" message printed in console.
Pass/Fail Criteria	The unit test completes without any failed assertions.
Assumptions and Constraints	None.
Dependencies	None.
Traceability	WC_5507.

Table 3: TC-01-02 Fail to load voter file when it is corrupt

Test Case Number	TC-01-02 Fail to load voter file when it is corrupt
Test Item	If the voter list file used by the application is corrupted for any reason, then the application shall render an error indicating as much, but shall continue to otherwise function.
Test Priority	Should have
Pre-conditions	• Python virtual environment has been enabled, and • Required Python modules have been downloaded via pip
Post-conditions	An error message shall be returned to the user indicating

	that an error has occurred
Input Specifications	Run the following command from within the base application directory: `$ python3 manage.py test -k test_voter_csv_corrupt`
Expected Output Specifications	“OK” message printed in console.
Pass/Fail Criteria	The unit test completes without any failed assertions.
Assumptions and Constraints	None.
Dependencies	None.
Traceability	WC_5507.

Table 4: TC-01-03 Fail to load voter file when references to it are null

Test Case Number	TC-01-03 Fail to load voter file when references to it are null
Test Item	If the pointer referencing the voter list becomes null due to an error internal to Python, an error shall be rendered, but otherwise the application shall continue to function.
Test Priority	Should have
Pre-conditions	• Python virtual environment has been enabled, and • Required Python modules have been downloaded via pip
Post-conditions	An error message shall be returned to the user indicating that an error has occurred
Input Specifications	Run the following command from within the base application directory: `$ python3 manage.py test -k`

	`test_voter_csv_none`
Expected Output Specifications	“OK” message printed in console.
Pass/Fail Criteria	The unit test completes without any failed assertions.
Assumptions and Constraints	None.
Dependencies	None.
Traceability	WC_5507.

Table 5: TC-01-04 Successfully load voter file with two volunteers

Test Case Number	TC-01-04 Successfully load voter file with two volunteers
Test Item	Tests that the backend can successfully load a voter file if it is present, not corrupt, and two volunteers are available to canvass.
Test Priority	Must have
Pre-conditions	• The “Field Progress” Collection has been loaded into Postman, and • The user’s workstation has network access to a running instance of the application
Post-conditions	A JSON payload shall be returned containing grouped voter points.
Input Specifications	From the Postman application, choose the “Cut turf w/ two volunteers” request. Then, select the big blue “Send” button toward the upper-right corner of the interface.
Expected Output Specifications	After several seconds, the Postman interface shall indicate that a response was returned from the server with a 200 OK status code, and a dump of JSON.

Pass/Fail Criteria	• Each point returned in the GeoJSON FeatureCollection shall belong to at most one of two groups • Each group shall have at least one point associated with it • Each group ID shall correspond to a single volunteer
Assumptions and Constraints	None.
Dependencies	None.
Traceability	WC_5507.

Table 6: TC-01-05 Successfully load voter file with five volunteers

Test Case Number	TC-01-05 Successfully load voter file with five volunteers
Test Item	Tests that the backend can successfully load a voter file if it is present, not corrupt, and five volunteers are available to canvass.
Test Priority	Must have
Pre-conditions	• The “Field Progress” Collection has been loaded into Postman, and • The user’s workstation has network access to a running instance of the application
Post-conditions	A JSON payload shall be returned containing grouped voter points.
Input Specifications	From the Postman application, choose the “Cut turf w/ two volunteers” request. Then, select the big blue “Send” button toward the upper-right corner of the interface.
Expected Output Specifications	After several seconds, the Postman interface shall indicate that a response was returned from the server with a 200 OK status code, and a dump of JSON.

Pass/Fail Criteria	• Each point returned in the GeoJSON FeatureCollection shall belong to at most one of five groups • Each group shall have at least one point associated with it • Each group ID shall correspond to a single volunteer
Assumptions and Constraints	None.
Dependencies	None.
Traceability	WC_5507.

3.2 TC-02 Changing number of volunteers

3.2.1 Test Level

Software item level

3.2.2 Test Class

• Functionality test, and
• Erroneous test

3.2.3 Test Completion Criteria

The following breaks down the criteria that must be met for each test case to be completed:

•. TC-02-01 Cut turf when given sufficient voters and volunteers
• TC-02-02 Fail to cut turf when no volunteers are provided and no voters are provided
• TC-02-03 Fail to cut turf when no volunteers are provided and some voters are provided

3.2.4 Test Cases

Table 7: TC-02-01 Cut turf when given sufficient voters and volunteers

Test Case Number	TC-02-01 Cut turf when given sufficient voters and volunteers
Test Item	Tests that, if given a positive number of voters and volunteers, the turf encompassing the voters shall be cut into sections. There shall be the same number of sections as there are volunteers.
Test Priority	Must have
Pre-conditions	• Python virtual environment has been enabled, and • Required Python modules have been downloaded via pip
Post-conditions	No error message shall be returned to the user, and the turf shall be cut into a number of sections equal to the number of volunteers available to canvass.
Input Specifications	Run the following command from within the base application directory: `$ python3 manage.py test -k test_cut_sufficient_volunteers_and_voters`
Expected Output Specifications	"OK" message printed in console.
Pass/Fail Criteria	The unit test completes without any failed assertions.
Assumptions and Constraints	None.
Dependencies	None.
Traceability	WC_5508 and WC_5488.

Table 8: TC-02-02 Fail to cut turf when no volunteers are provided and no voters are provided

Test Case Number	TC-02-02 Fail to cut turf when no volunteers are provided and no voters are provided
Test Item	Tests that, if not given any volunteers or voters, the turf encompassing the voters shall not be cut into sections.
Test Priority	Should have
Pre-conditions	• Python virtual environment has been enabled, and • Required Python modules have been downloaded via pip
Post-conditions	An error message shall be returned to the user, and the turf shall not be cut into sections.
Input Specifications	Run the following command from within the base application directory: `$ python3 manage.py test -k test_cut_no_volunteers_no_voters`
Expected Output Specifications	"OK" message printed in console.
Pass/Fail Criteria	The unit test completes without any failed assertions.
Assumptions and Constraints	None.
Dependencies	None.
Traceability	WC_5508 and WC_5488.

Table 9: TC-02-03 Fail to cut turf when no volunteers are provided and some voters are provided

Test Case Number	TC-02-03 Fail to cut turf when no volunteers are provided

	and some voters are provided
Test Item	Tests that, if given a sufficient number of voters but no volunteers, the turf encompassing the voters shall not be cut into sections.
Test Priority	Should have
Pre-conditions	• Python virtual environment has been enabled, and • Required Python modules have been downloaded via pip
Post-conditions	An error message shall be returned to the user, and the turf shall not be cut into sections.
Input Specifications	Run the following command from within the base application directory: `$ python3 manage.py test –k test_cut_sufficient_volunteers_no_voters`
Expected Output Specifications	“OK” message printed in console.
Pass/Fail Criteria	The unit test completes without any failed assertions.
Assumptions and Constraints	None.
Dependencies	None.
Traceability	WC_5508 and WC_5488.

3.3 TC-03 Map rendering

3.3.1 Test Level

Software item level

3.3.2 Test Class

- Functionality test, and

- Erroneous test

3.3.3 Test Completion Criteria

The following breaks down the criteria that must be met for each test case to be completed:

- TC-03-01 Providing two volunteers and cutting turf
- TC-03-02 Providing five volunteers and cutting turf
- TC-03-03 Providing no volunteers and failing to cut turf

3.3.4 Test Cases

Table 10: TC-03-01 Providing two volunteers and cutting turf

Test Case Number	TC-03-01 Providing two volunteers and cutting turf
Test Item	The user shall be able to provide two distinct volunteers with their own availabilities, choose a precinct, and cut the turf.
Test Priority	Must have
Pre-conditions	The user's workstation has network access to a running instance of the application
Post-conditions	The user should then have a map rendered for them which displays each voter as a colored point, along with route information.
Input Specifications	Volunteer #1: • Name: Kevin • Availability: 0.2 Volunteer #2: • Name: Uche • Availability: 0.5 Precinct: 130960
Expected Output	After a few seconds, the voters of the precinct shall be

Specifications	grouped into two different groups. Some voters may not have been assigned to a group due to the limited availability of the volunteers.
Pass/Fail Criteria	The test shall pass if and only if: • Voters are clustered into either one of the two groups or none, • There are exactly two groups, • Each group has a distinct color
Assumptions and Constraints	None.
Dependencies	None.
Traceability	WC_5487 and WC_5696.

Table 11: TC-03-02 Providing five volunteers and cutting turf

Test Case Number	TC-03-02 Providing five volunteers and cutting turf
Test Item	The user shall be able to provide five distinct volunteers with their own availabilities, choose a precinct, and cut the turf.
Test Priority	Must have
Pre-conditions	The user's workstation has network access to a running instance of the application
Post-conditions	The user should then have a map rendered for them which displays each voter as a colored point, along with route information.
Input Specifications	Volunteer #1: • Name: Kevin • Availability: 0.2

	Volunteer #2: • Name: Uche • Availability: 0.5 Volunteer #3: • Name: Ronald • Availability: 0.3 Volunteer #4: • Name: Arnold • Availability: 0.4 Volunteer #5: • Name: Lisa • Availability: 0.2 Precinct: 130960
Expected Output Specifications	After a few seconds, the voters of the precinct shall be grouped into two different groups. Some voters may not have been assigned to a group due to the limited availability of the volunteers.
Pass/Fail Criteria	The test shall pass if and only if: • Voters are clustered into either one of the five groups or none, • There are exactly five groups, and • Each group has a distinct color
Assumptions and Constraints	None.
Dependencies	None.
Traceability	WC_5487 and WC_5696.

Table 12: TC-03-03 Providing no volunteers and failing to cut turf

Test Case Number	TC-03-03 Providing no volunteers and failing to cut turf
Test Item	Should the user not provide any volunteers, choose a precinct, and attempt to cut turf, they shall be provided with an error message.
Test Priority	Must have
Pre-conditions	• The user's workstation has network access to a running instance of the application
Post-conditions	The user should be provided with an error message and an opportunity to correct their mistake.
Input Specifications	Precinct: 130960
Expected Output Specifications	A dialog should open in the user's web browser alerting them that their input is invalid.
Pass/Fail Criteria	The test shall pass if and only if: • The user is provided with an error message, and • The user is unable to cut turf until they correct the error
Assumptions and Constraints	None.
Dependencies	None.
Traceability	WC_5487 and WC_5696.

3.4 TC-04 Adding volunteer cards

3.4.1 Test Level

Software item level

3.4.2 Test Class

• Functionality test, and

- Erroneous test

3.4.3 Test Completion Criteria

The following breaks down the criteria that must be met for each test case to be completed:

- TC-04-01 Adding a single volunteer card, and
- TC-04-02 Adding three volunteer cards

3.4.4 Test Cases

Table 13: TC-04-01 Adding a single volunteer card

Test Case Number	TC-04-01 Adding a single volunteer card
Test Item	The user shall be able to add a new volunteer.
Test Priority	Must have
Pre-conditions	• The user's workstation has network access to a running instance of the application • All node dependencies have been installed via npm install
Post-conditions	A new voter card shall be created.
Input Specifications	From within the `frontend/FP_Frontend` directory, run: `$ node_modules/.bin/cucumber-js features/add_volunteer.feature`
Expected Output Specifications	Google Chrome will open automatically and run the test. The results of the test shall be printed to the console from which the test was executed.
Pass/Fail Criteria	The test shall pass if and only if: • No scenarios fail, and • No steps fail
Assumptions and	None.

Constraints	
Dependencies	None.
Traceability	WC_5488.

Table 14: TC-04-02 Adding three volunteer cards

Test Case Number	TC-04-02 Adding three volunteer cards
Test Item	The user shall be able to add three new volunteers.
Test Priority	Should have
Pre-conditions	• The user's workstation has network access to a running instance of the application • All node dependencies have been installed via npm install
Post-conditions	Three new voter cards shall be created.
Input Specifications	From within the `frontend/FP_Frontend` directory, run: `$ node_modules/.bin/cucumber-js features/add_three_volunteers.feature`
Expected Output Specifications	Google Chrome will open automatically and run the test. The results of the test shall be printed to the console from which the test was executed.
Pass/Fail Criteria	The test shall pass if and only if: • No scenarios fail, and • No steps fail
Assumptions and Constraints	None.
Dependencies	None.

Traceability	WC_5488.

3.5 TC-05 User volunteer input

3.5.1 Test Level

Software item level

3.5.2 Test Class

- Functionality test, and
- Erroneous test

3.5.3 Test Completion Criteria

The following breaks down the criteria that must be met for each test case to be completed:

- TC-05-01 Overwriting name placeholder text,
- TC-05-02 Overwriting availability placeholder text,
- TC-05-03 Only allowing alphabetic characters for name, and
- TC-05-04 Only allowing numbers for availability

3.5.4 Test Cases

Table 15: TC-05-01 Overwriting name placeholder text

Test Case Number	TC-05-01 Overwriting name placeholder text
Test Item	The user shall be able to overwrite the placeholder text for volunteer name in each card.
Test Priority	Should have
Pre-conditions	• The user's workstation has network access to a running instance of the application • All node dependencies have been installed via `npm`

	`install`
Post-conditions	The volunteer's name shall overwrite the placeholder text.
Input Specifications	From within the `frontend/FP_Frontend` directory, run: `$ node_modules/.bin/cucumber-js features/placeholder_allows_input.feature`
Expected Output Specifications	Google Chrome will open automatically and run the test. The results of the test shall be printed to the console from which the test was executed.
Pass/Fail Criteria	The test shall pass if and only if: • No scenarios fail, and • No steps fail
Assumptions and Constraints	None.
Dependencies	None.
Traceability	WC_5488.

Table 16: TC-05-02 Overwriting availability placeholder text

Test Case Number	TC-05-02 Overwriting availability placeholder text
Test Item	The user shall be able to overwrite the placeholder text for volunteer availability in each card.
Test Priority	Should have
Pre-conditions	• The user's workstation has network access to a running instance of the application • All node dependencies have been installed via `npm install`
Post-conditions	The volunteer's availability shall overwrite the placeholder

	text.
Input Specifications	From within the `frontend/FP_Frontend` directory, run: `$ node_modules/.bin/cucumber-js features/placeholder_allows_input.feature`
Expected Output Specifications	Google Chrome will open automatically and run the test. The results of the test shall be printed to the console from which the test was executed.
Pass/Fail Criteria	The test shall pass if and only if: • No scenarios fail, and • No steps fail
Assumptions and Constraints	None.
Dependencies	None.
Traceability	WC_5488.

Table 17: TC-05-03 Only allowing alphabetic characters for name

Test Case Number	TC-05-03 Only allowing alphabetic characters for name
Test Item	The user shall only be able to provide a sequence of alphabetic characters for a volunteer name.
Test Priority	Should have
Pre-conditions	• The user’s workstation has network access to a running instance of the application • All node dependencies have been installed via `npm install`
Post-conditions	Any non-letter (and space) characters shall be rejected by the input box for volunteer name.

Input Specifications	From within the `frontend/FP_Frontend` directory, run: `$ node_modules/.bin/cucumber-js features/only_letter_names.feature`
Expected Output Specifications	Google Chrome will open automatically and run the test. The results of the test shall be printed to the console from which the test was executed.
Pass/Fail Criteria	The test shall pass if and only if: • No scenarios fail, and • No steps fail
Assumptions and Constraints	None.
Dependencies	None.
Traceability	WC_5488.

Table 18: TC-05-04 Only allowing numbers for availability

Test Case Number	TC-05-04 Only allowing numbers for availability
Test Item	The user shall only be able to provide a number for a volunteer's availability.
Test Priority	Should have
Pre-conditions	• The user's workstation has network access to a running instance of the application • All node dependencies have been installed via `npm install`
Post-conditions	Any non-number (and period) characters shall be rejected by the input box for volunteer availability.
Input Specifications	From within the `frontend/FP_Frontend` directory, run: `$ node_modules/.bin/cucumber-js`

	`features/only_time_avails.feature`
Expected Output Specifications	Google Chrome will open automatically and run the test. The results of the test shall be printed to the console from which the test was executed.
Pass/Fail Criteria	The test shall pass if and only if: • No scenarios fail, and • No steps fail
Assumptions and Constraints	None.
Dependencies	None.
Traceability	WC_5488.

3.6 TC-06 Toggling input panel

3.6.1 Test Level

Software item level

3.6.2 Test Class

- Functionality test, and
- Erroneous test

3.6.3 Test Completion Criteria

The following breaks down the criteria that must be met for each test case to be completed:

- TC-06-01 Toggle input panel off, and
- TC-06-02 Toggle input panel off and on

3.6.4 Test Cases

Table 19: TC-06-01 Toggle input panel off

Test Case Number	TC-06-01 Toggle input panel off
Test Item	The user shall only be able to toggle the input panel off, hiding it from view.
Test Priority	Should have
Pre-conditions	• The user's workstation has network access to a running instance of the application • All node dependencies have been installed via `npm install`
Post-conditions	The input panel shall disappear.
Input Specifications	From within the `frontend/FP_Frontend` directory, run: `$ node_modules/.bin/cucumber-js features/toggle_input_off.feature`
Expected Output Specifications	Google Chrome will open automatically and run the test. The results of the test shall be printed to the console from which the test was executed.
Pass/Fail Criteria	The test shall pass if and only if: • No scenarios fail, and • No steps fail
Assumptions and Constraints	None.
Dependencies	None.
Traceability	WC_5488.

Table 20: TC-06-02 Toggle input panel off and on

Test Case Number	TC-06-02 Toggle input panel off and on
Test Item	The user shall only be able to toggle the input panel off, hiding it from view, and then back on.
Test Priority	Should have
Pre-conditions	• The user's workstation has network access to a running instance of the application • All node dependencies have been installed via `npm install`
Post-conditions	The input panel shall disappear, and then reappear.
Input Specifications	From within the `frontend/FP_Frontend` directory, run: `$ node_modules/.bin/cucumber-js features/toggle_input_on.feature`
Expected Output Specifications	Google Chrome will open automatically and run the test. The results of the test shall be printed to the console from which the test was executed.
Pass/Fail Criteria	The test shall pass if and only if: • No scenarios fail, and • No steps fail
Assumptions and Constraints	None.
Dependencies	None.
Traceability	WC_5488.

4. Resources and Schedule

4.1 Resources

The hardware and software required to run these test cases is listed in sections 2.1 and 2.2, respectively. Limited monetary resources are required for procuring the software, as each component is available for download free of charge from their maintainers. Procuring the necessary hardware may prove to be more costly, however.

4.2 Staffing and Training Needs

Management, designing, and preparing of test items is the responsibility of IIV&V. Additional stakeholders may be involved in executing, witnessing, inspecting, and resolving the test items, however.

Developing additional tests requires deep knowledge of the application, as well as the technologies that enable it, namely Python and JavaScript. Many free resources are available online for learning how to develop with them.

Running the unit tests themselves requires minimal technical knowledge, other than familiarity with the command line.

Verifying that test items satisfy the stakeholders' requirements requires that an individual communicate directly with the stakeholders, while maintaining an open dialog with the development team to ensure that all requirements are being tested and verified.

4.3 Schedule

Table 21: Testing Schedule

Date	Test Identifier	Responsible person	Resources	Training needs
11/25/19	TC-01-01 to TC-01-05	Kevin Grimes	Backend code base, sample voters	Django, Python
11/29/19	TC-02-01 to TC-02-03	Kevin Grimes	Backend code base, sample voters	Django, Python
12/01/19	TC-03-01 to TC-03-03	Kevin Grimes	Frontend code base	React.js, JavaScript
12/05/19	TC-04-01 to TC-04-02, TC-05-01 to TC-05-04, TC-06-01 to TC-06-02	Kevin Grimes	Frontend code base	React.js, JavaScript

12/06/19	Run all tests	Aishwarya Joisa	Entire test suite	N/A
12/06/19	Run frontend tests	Uche Uba	Frontend test suite	N/A

5. Test Results

Table 22: Test Results

ID	Category	Description	Pass/Fail?	Date
TC-01-01	Backend I/O	If the individual setting up the application for a given campaign fails to provide a file containing the voters to consider canvassing, then the application shall render an error, but continue to function.	Pass	12/06/2019
TC-01-02	Backend I/O	If the voter list file used by the application is corrupted for any reason, then the application shall render an error indicating as much, but shall continue to otherwise function.	Pass	12/06/2019
TC-01-03	Backend I/O	If the pointer referencing the voter list becomes null due to an error internal to Python, an error shall be rendered, but otherwise the application shall continue to function.	Pass	12/06/2019
TC-01-04	Postman	Tests that the backend can successfully load a voter file if it is present, not corrupt, and two volunteers are available to canvass.	Pass	12/06/2019
TC-01-05	Postman	Tests that the backend can successfully load a voter file if it is present, not corrupt, and five volunteers are available to canvass.	Pass	12/06/2019
TC-02-01	Algorithm	Tests that, if given a positive number of voters and volunteers, the turf encompassing the voters shall be cut into sections. There shall be the same	Pass	12/06/2019

		number of sections as there are volunteers.		
TC-02-02	Algorithm	Tests that, if not given any volunteers or voters, the turf encompassing the voters shall not be cut into sections.	Pass	12/06/2019
TC-02-03	Algorithm	Tests that, if given a sufficient number of voters but no volunteers, the turf encompassing the voters shall not be cut into sections.	Pass	12/06/2019
TC-03-01	Web testing	The user shall be able to provide two distinct volunteers with their own availabilities, choose a precinct, and cut the turf.	Pass	12/06/2019
TC-03-02	Web testing	The user shall be able to provide five distinct volunteers with their own availabilities, choose a precinct, and cut the turf.	Pass	12/06/2019
TC-03-03	Web testing	Should the user not provide any volunteers, choose a precinct, and attempt to cut turf, they shall be provided with an error message.	Pass	12/06/2019
TC-04-01	Selenium	The user shall be able to add a new volunteer.	Pass	12/06/2019
TC-04-02	Selenium	The user shall be able to add three new volunteers.	Pass	12/06/2019
TC-05-01	Selenium	The user shall be able to overwrite the placeholder text for volunteer name in each card.	Pass	12/06/2019
TC-05-02	Selenium	The user shall be able to overwrite the placeholder text for volunteer availability in each card.	Pass	12/06/2019

TC-05-03	Selenium	The user shall only be able to provide a sequence of alphabetic characters for a volunteer name.	Pass	12/06/2019
TC-05-04	Selenium	The user shall only be able to provide a number for a volunteer's availability.	Pass	12/06/2019
TC-06-01	Selenium	The user shall only be able to toggle the input panel off, hiding it from view.	Pass	12/06/2019
TC-06-02	Selenium	The user shall only be able to toggle the input panel off, hiding it from view, and then back on.	Pass	12/06/2019