Test Plan Document

For

FLORA

1 Introduction 3

2 Business Background 3

3 Test Objectives 4

4 Scope 4

5 Test types Identified 5

6 Problems Perceived 5

7 Architecture 5

8 Environment 6

9 Assumptions 6

10 Functionality 6

11 Security 7

12 Performance 9

13 Usability 9

14 Compatibility 11

15 Test Team Organization 11

16 Schedule 12

17 Defects Classification Mechanism 12

18 Configuration Management 14

19 Release Criteria 14

Test Plan

# Introduction

The Flora application is a mobile application which resides on users’ physical mobile devices and allows users to access information stored on a variety of interconnected databases—both local and remote. The application software is compiled to Android (and potentially iOS) native code, will eventually be distributed via each operating system’s primary application store, and uses third-party APIs to interact with a variety of data sources.

The application aims to bring the field of plant identification into the current technological era. The goal is to make centuries of research and identification more easily accessible for professionals and amateurs in the field. An ideal outcome will allow researchers and amateurs to access hundreds of volumes’ worth of data about local plant species on-site, directly from their mobile devices.

# Business Background

Plant enthusiasts often find themselves standing in natural environments, wishing that they had brought resources to help identify local plants. Currently, books and awkward websites exist to aid in identification. Our goal is to make spontaneous plant identification a little easier by providing users with a mobile application where they can easily access publicly maintained data about local plant species. Based on a user’s current location and user-identified characteristics of the plant, our application produces a list of local plant species, along with data about each plant, simplifying the process of plant identification in the field.

Indiana University maintains herbaria in New Albany and Bloomington. These herbaria are linked to a larger net-work of herbaria through the Consortium of Midwest Herbaria, which freely shares data and plant samples to assist with identification of local and international plant species. Sites like the Consortium’s are just beginning to offer some of the features that our application aims to provide, but unfortunately the existing interfaces are relatively slow and difficult to use on mobile devices.

# Test Objectives

Our tests should verify that the following aspects of our program are fully functional:

• **User Interface** – The application should load via a loading screen, and by tapping the screen, a user should be able to move to the first series of menu options. Each button within the application should be “clickable” via a standard touch gesture on a mobile device, and each page should flow seamlessly into the next.

• **User Interface to Data Access Layer** – The buttons on the user interface should be able to reach the data access layer, and the data access layer should produce valid data.

• **Data Access Layer to Herbarium Database**—the data access layer component of the code should be able to access the database that is hosted by IUS.

• **Remote Data Source**—each remote data source should be tested apart from the data access layer—that is, directly by using a web browser or a tool like SOAP UI or Postman

• **Data Access Layer to Remote Data Source**—the data access layer component of the code should be able to access various APIs or remote web pages, depending upon the function and data desired. Each of these should be tested individually and should return predictable, consistent results.

# Scope

***Inclusions***

• **User Interface** – Every screen and button should be tested for clickability, and each page should reliably flow to the next.

• **User Interface to Data Access Layer** – each button should be wired correctly to reach the data access layer, even if the data access layer returns “mock” data.

• **Data Access Layer to Database**—every desired feature should be fully tested.

• **Remote Data Source**—The Herbarium API should be tested at this time, as well as manual navigation to various image sources.

• **Data Access Layer to Remote Data Source**—This should be fully tested for each function.

***Exclusions***

• **User Interface** – Some screens—to be identified at each stage of development—will contain either incomplete or mock data. These unimplemented features should be excluded at each phase of testing.

• **User Interface to Data Access Layer** – No exclusions.

• **Data Access Layer to Database**—No exclusions.

• **Remote Data Source**—All data sources with the exception of the Herbarium should be excluded from these tests until additional data sources are identified.

• **Data Access Layer to Remote Data Source**— All data sources with the exception of the Herbarium should be excluded from these tests until additional data sources are identified.

# Test types Identified

• **User Interface** – Manual testing of each feature will be performed. Eventually, a tool like Esprsso or UITest could be utilized to automate testing.

• **User Interface to Data Access Layer** – Initially, manual testing will be performed. Eventually, automated unit tests will be implemented.

• **Data Access Layer to Database** — initially, manual testing will be performed. Eventually, automated unit tests will be implemented.

• **Remote Data Source**— initially, manual testing will be performed. Eventually, test suites will be written in either SOAP UI or Postman, as well as automated browser queries. These will test for known data.

• **Data Access Layer to Remote Data Source**— initially, manual testing will be performed. Eventually, automated unit tests will be implemented.. Finally, actual calls to the remote data sources will be made via automated tests.

# Problems Perceived

Currently, all implemented features have been tested manually and have not produced errors. As additional features are implemented, additional testing will be performed, and it is anticipated that as our test suites continue to develop, we will identify more complex problems which must be addressed.

# Architecture

The application utilizes a simple touch-screen user interface. This interface interacts with various third-party application programming interfaces by way of an abstracted data access layer, which is programmed in the same framework as the user interface. See full Software Architecture Specification document for additional details.

# Environment

This application currently runs on Android devices and emulators. Additional development will be required to run the application on iOS.

# Assumptions

This application assumes that a system exists where:

• Plants are professionally identified

• Photographs of these plants exist

• Plants are consistently labelled according to physical characteristics

• The data is loaded into a database

• The collection is accessible over the public internet

Our application makes use of such data, and without this system, our application will not function.

# Functionality

***Constraints and Resolutions***

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Customer Constraints** | **Infosys Limitations** |
| Constraint 1 | Mobile device required | Not responsibility of development team, to be solved by user |
| Constraint 2 | Mobile data access required | Not responsibility of development team, to be solved by user |
| Constraint 3 | Access to application store required for download | Not responsibility of development team, to be solved by user |
| Constraint 4 | Access from application to each API required | The development team has full responsibility for maintaining access to third party |

***Risk Identified & Mitigation Planned***

The majority of our application’s functionality will be tested prior to deployment, and our code should not change over time. The most likely potential problem will be if a third party API ceases to function—either due to maintenance or a change in the API’s endpoint or rules. The development team should institute a check to verify that each API is accessible and should notify users when these are not accessible.

As API endpoints are updated and rules are changed, the development team will need to do additional development to resolve them. A website or portal of some kind will need to be created so that bugs can be reported and resolved.

***Test Strategy***

At this time, we are only utilizing manual testing. Eventually, these tests will be automated.

***Automation Plans***

We plan to utilize tools like Espresso or UITest (for UI testing), a tool like Selenium for automated browser testing of third party endpoints, and SOAP UI or Postman test suites for third-party API testing. No specific automated test plans have been implemented at this time.

***Deliverables***

A compiled version of the application will be deployed to the Google Play application store or to a website from which it can be downloaded. This application will contain the primary features of location-based and key-based species searches, as well as a direct species lookup tool. These features will all eventually result in the user viewing a series of plant profile pages which will provide detailed information about each species.

# Security

***Constraints and Resolutions***

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Customer Constraints** | **Infosys Limitations** |
| Constraint 1 | Alternatively to constraint 1, access to third party application stores is required | If our application is hosted on a third party site, it is only the development team’s responsibility to upload our application. The third party store (Google) will have responsibility for verifying that the download is accurate and secure at that point |
| Constraint 2 | Alternatively to constraint 1, access to a website where we host the application is required | If this route is taken, the development team will need to maintain a secure website that cannot have traffic re-routed to malicious downloads |
| Constraint 3 | Users will trust that access to third party APIs is safe | As long as the development team directs traffic to the appropriate endpoints, no malicious data should be returned, and access from the third party to the user will not be possible. By monitoring the third party endpoints, the development team can reduce risk in this area |

***Risk Identified & Mitigation Planned***

The only potential risk will be if the application is hosted on a website that is maintained by the development team (rather than hosted by a third party application store, like Google Play). In this case, the development team should verify that none of the OWASP Top 10 vulnerabilities exist, and a third party vulnerability scanning tool should be utilized as soon as possible. Security professionals should be consulted as well. If a user clicks on our application to download it and is redirected to a malicious site, damage could occur to that user and to the reputation of the product and development team.

***Test Strategy***

At this time, we are only utilizing manual testing. As no security vulnerabilities are currently identified, there is nothing to test in this area. At some point in the development process, we should consult with a security professional to determine which vulnerabilities may exist.

***Automation Plans***

No automated tests currently exist, but we should utilize an automated vulnerability scanner (like Nessus) if we decide to host the application directly on a website.

***Deliverables***

The current plan is to deploy the application to a third party application store, and we do not have any security-related deliverables planned at this time

# Performance

***Constraints and Resolutions***

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Customer Constraints** | **Infosys Limitations** |
| Constraint 1 | App should run without crashing in a reasonable amount of time | It is the responsibility of the development team to test the app for potential crashes before release |

***Risk Identified & Mitigation Planned***

Currently, the images take an unreasonable amount of time to load. A possible solution to rectifying this issue is to break the API search into multiple sections so each section can load directly to the user without causing them to wait for later objects that aren’t currently able to be accessed.

***Test Strategy***

At this time, we are only utilizing manual testing. Eventually, these tests will be automated.

***Automation Plans***

No specific automated test plans have been implemented at this time.

***Deliverables***

The current plan is to deploy the application to a third party application store, and we do not have any performance-related deliverables planned at this time

# Usability

***Constraints and Resolutions***

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Customer Constraints** | **Infosys Limitations** |
| Constraint 1 | Query results should return within a reasonable amount of time (approximately 20 seconds for large data sets and <1 second for small data sets) | The development team will need to identify which data sets are “large” and “small” then improve technology so that these speeds are achieved |
| Constraint 2 | The application should open, load, and close successfully | To be tested by the development team prior to release |
| Constraint 3 | Each button within the application should move to the correct screen | To be tested by the development team prior to release |
| Constraint 4 | Meaningful (and reportable) error messages should be displayed to the user | To be implemented by the development team prior to release |

***Risk Identified & Mitigation Planned***

Currently, constraints 2 and 3 do not appear to be presenting issues. Still, these will be tested prior to release. Error messages are being written (and will continue to be written) to address constraint 4. Currently, there is no system for submitting error messages to the development team, and such a system will need to be created.

The most likely problem in this area will be with the speed of the third party APIs. The speed of these systems (and the size of the requests that we send) is not within our control. Currently, the images take an unreasonable amount of time to load. A possible solution to rectifying this issue is to break the API search into multiple sections so each section can load directly to the user without causing them to wait for later objects that aren’t currently able to be accessed.

***Test Strategy***

At this time, we are only utilizing manual testing. Eventually, these tests will be automated.

***Automation Plans***

We plan to utilize tools like Espresso or UITest (for UI testing), xUnit or Moq (for unit testing and data mocking), a suite of SQL queries with known results for database testing, a tool like Selenium for automated browser testing of third party endpoints, and SOAP UI or Postman test suites for third-party API testing.

No specific automated test plans have been implemented at this time.

***Deliverables***

For each of the tests that we develop, we will establish an acceptable timeframe during which that action should complete (in most cases, by fully displaying a page or returning results).

***14 Compatibility***

***Constraints and Resolutions***

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Customer Constraints** | **Infosys Limitations** |
| Constraint 1 | The application should be compatible and fully functional for all Android device | The development team will fully test the application for a multitude of Android devices |
| Constraint 2 | The application should be compatible and fully functional for all iOS devices | After completion of Android, the development team will convert the application for iOS devices |

***Risk Identified & Mitigation Planned***

iOS development will take place after completion of Android development, if time allows. If not, app will be limited to Android devices.

***Test Strategy***

The application will be tested on multiple different Android devices and emulators.

***Automation Plans***

No specific automated test plans have been implemented at this time.

***Deliverables***

The current plan is to deploy the application to a third party application store, and we do not have any security-related deliverables planned at this time

# Test Team Organization

Each developer is expected to complete testing on each feature that he or she develops, with no code being submitted to source control until all are successful, or circumstances contributing to the failed test are documented and the failure is added to the backlog as a bug or feature improvement.

The team lead will periodically review these test results for accuracy and usefulness.

# Schedule

* Demo ready by Feb 5, 2020
* Multiple meetings with our sponsor throughout the semester
* We will be consulting with our sponsor for a final deadline for our project, which will need to be sometime before Apr 28, 2020

# Defects Classification Mechanism

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Type of Defects | Functionality | Performance | Security | Usability | Compatibility |
| Critical | For more than 10 users, one or more primary features of application is/are non-functional more than 5% of the time | For more than 10 users, results return 100-200% slower than highest acceptable timeframe more than 5% of the time | A single user reports that malware was installed on his or her PC while attempting to install our application | For more than10 users, one or more primary features of application is/are unusual more than 5% of the time | For more than 50 users, the application is unable to be installed or severely malfunctions on specific decives more than 25% of the time |
| Major | For 2-to-10 users, one or more primary features of application is/are non-functional more than 25% of the time  For more than 10 users, one or more secondary features of application is/are non-functional more than 25% of the time | For more than 10 users, results return 10-100% slower than highest acceptable timeframe more than 25% of the time | N/A- All security issues will be regarded as critical defects | For 2-to-10 users, one or more primary features of application is/are unusable more than 25% of the time  For more than 10 users, one or more secondary features of application is/are unusable more than 25% of the time | For more than 10 users, the application is unable to be installed or severely malfunctions on specific devices more than 25% of the time. |
| Minor | For a single user, one or more primary features of application is/are non-functional more than 25% of the time  For fewer than 10 users, one or more secondary features of application is/are non-functional more than 25% of the time | For fewer than 10 users, results return 10-50% slower than highest acceptable timeframe more than 25% of the time | N/A- All security issues will be regarded as critical defects | For a single user, one or more primary features of application is/are unusable more than 25% of the time | For more than 2 users, the application is unable to be installed or severely malfunctions on specific devices more than 25% of the time |
| Cosmetics | All major and minor features are fully functional, but some are slightly difficult to use due to display-related issues or performance issues | For any number of users, results return within 10% of highest acceptable timeframe more than 25% of the time | N/A- All security issues will be regarded as critical defects | All major and minor features are fully functional, but some are slightly difficult to use due to display-related issues or performance issues | For a single user, the application is unable to be installed or severely malfunctions on specific devices more than 25% of the time |

***Defects Logging and Status Changing Mechanism***

Defects will be logged through GitHub as issues are reported. The development team will be alerted for Critical and Major issues, while Minor and Cosmetic issues will not be reported and will be retrieved by development team members as they have availability.

***Turn Around Time for defect fixes***

Currently, no service level agreements are being proposed, as no full-time staff is employed for this project.

# Configuration Management

Code will be developed and tested through Visual Studio initially, then through additional tools as automation is implemented. GitHub will be utilized for source control and bug monitoring

# Release Criteria

The minimum viable product will be released on Android by the specified release date provided by our sponsor. APIs are fully functioning with possible API backups. There will be forms for potential bugs discovered by the client.