WikiWalks – Master Test Plan

# Introduction

## Project & Project Objective

WikiWalks is a project that aims to create an Android application to crowdsource and display bush paths in the user’s area. Both a front and back-end will need to be created, and it’s important that features can work bug-free for the best user experience.

## Objective of the Master Test Plan

The objective of the master test plan is to provide information about testing methodology and activities that can help ensure that all features implemented into the WikiWalks app are functional, bug-free, and intuitive to use.

# Documentation

## Basis for the Master Test Plan

The following documents are used as basis for this master test plan:

|  |  |  |  |
| --- | --- | --- | --- |
| Document name | Version | Date | Author |
| Vision | 1.1 | 2020-05-31 | Tyler |
| Architecture notebook | 1.1 | 2020-05-31 | Joey |
| Project plan | 1.1 | 2020-05-31 | Tyler |

## Test Basis

|  |  |  |  |
| --- | --- | --- | --- |
| Document name | Version | Date | Author |
| Use case model | 1.0 | 2020-04-07 | Tyler |
| Domain model | 1.0 | 2020-04-11 | Tyler |
| NFR analysis | 1.0 | 2020-04-11 | Tyler |

# Test Strategy

## Risk Analyses

### Product Risk Analysis

|  |  |  |  |
| --- | --- | --- | --- |
| Product Risk | Characteristic | Description | Risk Class |
| 1 | Performance | Software is inefficient and slow | A |
| 2 | Data integrity | Database corruption causes loss of paths | A |
| 3 | Usability | UI is too complex or too over-simplified to use efficiently | B |
| 4 | Reliability | Software crashes or is buggy | A |
| 5 | Scalability | Software can’t handle many requests at once | B |

### Technical Risk Analysis

|  |  |  |  |
| --- | --- | --- | --- |
| Technical Risk | Characteristic | Description | Risk Class |
| 1 | Getting paths | Getting all paths within coordinates efficiently | B |
| 2 | Adding paths | Failing to add all path details or allowing incomplete / invalid paths | A |
| 3 | Getting additional path data | Getting additional path data for paths (such as points of interest, reviews, pictures, group walks, etc.) | B |
| 4 | Adding paths | Failing to add additional path data for paths (such as points of interest, reviews, pictures, group walks, etc.) or accepting bad data | A |
| 5 | Working offline | Not being able to efficiently store the correct data in case of connection loss during walk | B |
| 6 | Editing data | Unauthorised users being able to edit data | A |
| 7 | Recording paths | Failing to record a detailed, accurate path | A |

## Test Strategy

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Risk | Description | Risk Class | Test Level | | | | | |
| SR | Unit | Int | FAT | UAT | ST |
| Performance | Software is inefficient or slow | A |  |  |  | \*\* | \*\*\* | \*\* |
| Data integrity | Database corruption causes loss of paths | A |  |  |  |  |  | \*\*\* |
| Usability | UI is too complex or too over-simplified to use efficiently | B |  |  |  | \*\* | \*\* |  |
| Reliability | Software crashes or is buggy | A |  | \*\* | \*\* |  |  | \*\*\* |
| Scalability | Software can’t handle many requests at once | B |  |  |  | \*\* | \*\*\* | \*\*\* |
| Getting paths | Getting all paths within coordinates efficiently | B |  | \*\* | \*\* | \*\* | \*\*\* |  |
| Adding paths | Failing to add all path details or allowing incomplete / invalid paths | A |  | \*\* | \*\* | \*\*\* | \*\*\* |  |
| Getting additional path data | Getting additional path data for paths (such as points of interest, reviews, pictures, group walks, etc.) | B |  | \*\* | \*\* | \*\* | \*\*\* |  |
| Adding additional path data | Failing to add additional path data for paths (such as points of interest, reviews, pictures, group walks, etc.) or accepting bad data | B |  | \*\* | \*\* | \*\*\* | \*\*\* |  |
| Working offline | Not being able to efficiently store the correct data in case of connection loss during walk | B | \* |  |  |  | \*\* | \*\*\* |
| Editing data | Unauthorised users being able to edit data | A | \* |  | \*\* |  | \*\* | \*\*\* |
| Recording paths | Failing to record a detailed, accurate path | A | \* |  |  | \*\* | \*\*\* | \*\* |

# Test Levels

For this MTP, the following test levels are acknowledged:

|  |  |
| --- | --- |
| Test Level | Goal |
| Unit testing | The aim is to test individual components of the software to check that it is doing all that it needs to correctly. |
| Integration testing | The aim is to test components working together to ensure they can successfully integrate and pass the required information between each other. |
| System testing | The aim is to test the completed, fully integrated system to ensure that everything is functioning correctly and it meets non-functional requirements. |
| Acceptance testing | The aim is to test in real-world situations to ensure the requirements are met and achieves the goal it was trying to achieve. |

## The Unit Test Level

### Entrance & Exit Criteria

This test level will be entered each time a major individual component is completed. It will exit once all problems detected with it have been resolved.

### Test Environment

The testing will be done in the development environments. PyCharm for the server, and Android Studio for the app. Test driver scripts will be developed for Python functions, and JUnit and Mockito will be used for testing the Java functions.

### Test Objectives

|  |  |  |  |
| --- | --- | --- | --- |
| Risk | Test Goals | Risk Verification | Schedule |
| Software crashes or is buggy | Ensure all major methods run individually as expected | Methods run as intended and have required error handling | During implementation of each major component |
| Getting all paths within coordinates efficiently | Ensure the component can return correct coordinates given boundaries | Method returns the correct list of paths | After implementation of CCRD (final iteration of LCAM) |
| Failing to add all path details or allowing incomplete / invalid paths | Ensure the component can add valid input paths to the database | Validation method correctly rejects invalid methods, and add to database method does so successfully | After implementation of CCRD (end of LCAM) |
| Failing to add additional path data | Ensure the component can accept correctly add valid additional path data to the database | Validation method correctly rejects invalid additional path data, and method to add to database does so successfully | Upon implementation of functionality for all types of additional path data |
| Getting additional path data for each path | Ensure the component can return additional path data for each path | Paths return additional path data details with themselves | Upon implementation of functionality for all types of additional path data |

## Integration Testing

### Entrance & Exit Criteria

This test level will be entered once all major components have been completed. It will exit once it is verified that they work together as they should.

### Test Environment

The testing will be done using Python test scripts to request and print the list of paths, and to send requests of correctly and incorrectly formatted JSONs to be added and rejected respectively.

### Test Objectives

|  |  |  |  |
| --- | --- | --- | --- |
| Risk | Test Goals | Risk Verification | Schedule |
| Getting all paths within coordinates efficiently | Ensure that the server can correctly generate and return a JSON | Server responds with correct JSON with all paths within boundary | After implementation of CCRD (final iteration of LCAM) |
| Failing to add all path details or allowing incomplete / invalid paths | Ensure the server can accept correctly formatted JSON requests to add new entries to the database | Validation method correctly rejects invalid paths, and method to add to database does so successfully | After implementation of CCRD (final iteration of LCAM) |
| Failing to add additional path data | Ensure the server can accept correctly add valid additional path data to the database | Validation method correctly rejects invalid additional path data, and method to add to database does so successfully | Upon implementation of functionality for all types of additional path data |
| Getting additional path data for each path | Ensure the server can return additional path data for each path | Paths return additional path data details with themselves | Upon implementation of functionality for all types of additional path data |

## System Testing

### Entrance & Exit Criteria

This test level will be entered once the server and app are at their final stages. It will exit when any identified issues have been resolved.

### Test Environment

The testing will be done by running builds of both the server and the app and using the functionalities that interact with each other.

### Test Objectives

|  |  |  |  |
| --- | --- | --- | --- |
| Risk | Test Goals | Risk Verification | Schedule |
| Software is inefficient or slow | Ensure the server can quickly respond to requests  Ensure the app runs smoothly without noticeable lag | Server responds with correct path information in <2 seconds  App runs fluidly and doesn’t feel slow | Upon completion of all components |
| Database corruption causes loss of paths | Ensure the server backs up and can quickly restore the database in the event of corruption | Server restores from backup upon trying to read a corrupted or invalid database | Upon completion of all components |
| Software crashes or is buggy | Ensure both the server and the app don’t crash completely with normal use | Neither the server nor app crash while using common functionality, and have error handlers to keep running smoothly in the case that they do | Upon completion of all components |
| Software can’t handle many requests at once | Ensure server can handle a large number of requests at once | Server successfully responds to all requests with the correct information | Upon completion of all components |
| Not being able to efficiently store the correct data in case of connection loss during walk | Ensure the app can go offline in the middle of a walk and continue functioning as normal for the remainder of the walk | App continues to show the right information when internet connection is lost | Upon completion of all components |
| Unauthorised users being able to edit data | Ensure that random users cannot edit paths submitted by others | Server rejects edit request when incorrect device ID is sent | Upon completion of all components |
| Failing to record a detailed, accurate path | Ensure that paths recorded are detailed and accurate | Path has many points and is in the correct location | Upon completion of all components |

## Acceptance Testing

### Entrance & Exit Criteria

This test level will be entered when all other testing is completed, and will be exited once any identified issues are resolved.

### Test Environment

This test level will be using the app in the real world like an average user would be. No test scripts are required.

### Test Objectives

|  |  |  |  |
| --- | --- | --- | --- |
| Risk | Test Goals | Risk Verification | Schedule |
| Software is inefficient or slow | Ensure test users are happy with the speed of the application | There are no complaints from users about slowness | Upon completion of all components and all other testing |
| UI is too complex or too over-simplified to use efficiently | Ensure test users can find all features easily without being overwhelmed | Users report an easy experience using the app’s functionality | Upon completion of all components and all other testing |
| Software can’t handle many requests at once | Ensure the server can handle real life users in real life situations | Server monitoring does not report maxing out resources  Test users do not complain of slowdowns | Upon completion of all components and all other testing |
| Getting all paths within coordinates efficiently | Ensure the app quickly pulls all paths in the user’s location | Test users do not report any missing paths in their area | After implementation of CCRD (final iteration of LCAM)  Again upon completion of all components and all other testing |
| Failing to add all path details or allowing incomplete / invalid paths | Ensure the app can successfully send new paths to the database | Test users report success submitting new paths | After implementation of CCRD (final iteration of LCAM)  Again upon completion of all components and all other testing |
| Not being able to efficiently store the correct data in case of connection loss during walk | Ensure the app can go offline in the middle of a walk and continue functioning as normal for the remainder of the walk | Test users do not report problems when connection drops | Upon completion of all components and all other testing |
| Unauthorised users being able to edit data | Ensure that random users cannot edit paths submitted by others | Test users do not report being able to edit other people’s paths, or having their paths edited by others | Upon completion of all components and all other testing |
| Failing to record a detailed, accurate path | Ensure that paths recorded are detailed and accurate | Test users report no problems recording detailed, accurate paths | After implementation of CCRD (final iteration of LCAM)  Again upon completion of all components and all other testing |
| Failing to add additional path data | Ensure the server can accept correctly add valid additional path data to the database | Test users report no problem adding additional path data to the server | Upon completion of all components and all other testing |
| Getting additional path data for each path | Ensure the server can return additional path data for each path | Test users report being able to see accurate additional path data for each path | Upon completion of all components and all other testing |