**Lab III – Prototype Test Plan/Procedure**

Prototype Test Plan/Procedure

For

Current ITS

CS 411W

Prepared by: Nathan Lutz

Date: 11-20-2012

Version 2.0

Table of Contents

[List of Tables 2](#_Toc341133848)

[List of Figures 2](#_Toc341133849)

[1. Objectives 3](#_Toc341133850)

[2. References 3](#_Toc341133851)

[3. Test Plan 3](#_Toc341133852)

[3.1 Testing Approach 4](#_Toc341133853)

[3.2 Identification of Tests 5](#_Toc341133854)

[3.3 Test Schedule 12](#_Toc341133855)

[3.4 Fault Reporting and Data Recording 13](#_Toc341133856)

[3.5 Resource Requirements 14](#_Toc341133857)

[3.6 Test Environment 15](#_Toc341133858)

[4. Test Responsibilities 16](#_Toc341133859)

[5. Test Procedures 16](#_Toc341133860)

[6. Traceability Requirements 16](#_Toc341133861)

# List of Tables

[Table 1. Test Category Identification 12](#_Toc341122576)

[Table 2. Test Schedule 13](#_Toc341122577)

[Table 3. Fault Reporting and Data Recording 13](#_Toc341122578)

# List of Figures

[Figure 1. Prototype Major Functional Component Diagram 4](file:///Y:\cs411\lab3v2.docx#_Toc341176306)

[Figure 2. Presentation Layout 15](#_Toc341176307)

# Objectives

The purpose of this document is to provide a framework for evaluating the Current ITS prototype. The purpose of these tests is to verify the requirements set forth in Lab 2 are being met and to demonstrate the full functionality of the prototype. The Tests will be performed on individual software units of Current ITS, utilizing a Test Harness GUI. The inter-component communication will be tested as well, to demonstrate that interfaces and dependencies have been completed by using the Web App Engine to view their GUI screens. The system as a whole will also be tested, by demonstrating the capabilities outlined in the description and specification documents. By utilizing the testing framework laid out in the following pages, Current ITS will be validated as a prototype.

# 2. References

Messina, Debbie (2011, July 31). *Control room at NSU servers as brains for light rail*.

From the Virginian Pilot: http://hamptonroads.com/2011/07/control-room-nsu-serves-brains-light-rail

Southeastern Institute of Research, Inc. (2011). *Hampton Roads Transit: Light Rail Marketing Research Study.* Norfolk: Southeastern Institute of Research, Inc.

Lutz, Nathan. Lab I - Current ITS Product Description. Old Dominion University CS411 Professional Workforce Development II. Norfolk, VA. 2012.

Lutz, Nathan. Lab II – Current ITS Product Specification. Old Dominion University CS411 Professional Workforce Development II. Norfolk, VA. 2012.

# 3. Test Plan

This section provides a thorough guide to the Current ITS testing process. It contains a summary of the types of testing that will be performed, the testing schedule, recording procedures, required resources to perform the tests and the test environment itself. Individual team member responsibilities are also defined.

# 3.1 Testing Approach

Current ITS will consist of three main types of tests. These test types are: unit tests, integration tests, and system tests, each with a specific evaluation of Current ITS. The unit tests will test individual functions or software modules within Current ITS. Integration tests will test the performance of interrelated software components as they work and communicate with each other. Finally, system tests will demonstrate the prototype in full, showcasing the functionality desired within it. These tests will have to cover each aspect of Current ITS, down to the smallest function, in order to prove the completeness and functionality of its design.

Figure 1. Prototype Major Functional Component Diagram

CS Dept Virtual Machine



Web App

Engine

Decision Engine



**DB**

Test Harness

Simulated GPS Data

Simulated APC Data

The major functional components of Current ITS can be seen in Figure 1. Each component will be tested, including the Database, Decision Engine, Web App Engine and the Test Harness itself. Each component will be tested separately beginning with individual unit tests, then as the software components that make them up with integration tests, and finally as a whole with system tests. The Database will be tested using PHPmyAdmin, verifying the contents within using SQL queries. The Decision Engine will have to be tested using manual inputs to individual functions and examining the outputs from system logs. The Web App Engine and all GUI screens therein will be tested for functionality and correct display. The Test Harness will be tested by verification of the control GUI and its associated functions in controlling the prototype.

# 3.2 Identification of Tests

Test cases for the Current ITS prototype are identified in Table 1. These test cases have been divided into unit, integration and system tests, as well as sub categories which are named for the individual software component that they are testing. Test cases should assert one or more functional requirements as specified for the Current ITS prototype as defined in Lab 2. A more complete and detailed listing of each test case and the testing procedures within is found in Section 5 of this document.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Category** | **Description** | | **Test Case** | **Description** | **Objective** | |
| Unit | Database | 1.1 | | Formats | | Visually examine the entries for date (YYYYMMDD) and time (HHMMSS) formats. |
| 1.2 | | Read/Write Procedures | | Test create, read, update, and delete stored procedures on the database. |
| 1.3 | | Refresh Times | | Test updateable views for functionality and response/refresh time. |
| 1.4 | | Read/Write Triggers | | Test triggers for proper functionality on various create, read, and delete commands. |
| 1.5 | | Schema Efficiency | | Test tables for proper schema, functionality, and efficiency and ease of use, and for proper data values. |
| Integration | Decision Engine –  DB Interface | | 2.1 | Database Connectivity | Test Current ITS database connection | |
| 2.2 | Select Ability | Test ability to query the database and necessary tables | |
| Unit | Decision Engine –  Ridership Trend Analysis | | 2.3 | Input Validation | Validate Input of date, time range, stop id | |
| 2.4 | Interval Validation | Test future/past date determination | |
| 2.5 | Average Function Test | Test average of embark/disembark for past 15 days | |
| 2.6 | Past Event Test | Test past Event Detection | |
| 2.7 | Future Event Test | Test future Event Detection | |
| 2.8 | Ridership Variance Function Test | Test accuracy of variance between established disembark/embark averages and past event values | |
| Integration | 2.9 | Ridership  Output Validation | Test output of embark/disembark to Ridership Trend Report function | |
| Unit | Decision Engine –  Delay Impact Calculator | | 2.10 | Input Validation | Validate input of GPS coordinates and dates | |
| 2.11 | Train Activity Test | Test ability to determine if a train is active or not | |
| 2.12 | Delay Average Calculation Test | Test accuracy of average variance from the schedule | |
| 2.13 | Alert Detection Test | Test ability to correctly identify active alerts | |
| 2.14 | Alert Delay Interval Application Test | Test accuracy of alert severity level delay interval on variance average | |
| 2.15 | Total Calculated Delay Test | Test accuracy of comparison between calculated expected time-of-arrival and schedule | |
| Integration | 2.16 | Delay Estimate Output Validation | Test output of delay time to Train Data Report Module | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Unit | Decision Engine – Ontime Performance Reporting | 2.17 | Input Validation | Validate input of Date range and Stop ID |
| 2.18 | Ontime Accuracy Test | Test accuracy of variance between past arrival times and schedule times |
| Integration | 2.19 | Ontime Performance Output Validation | Test output of variance from the schedule to the Train Data Report Module |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Harness | GPS Data tester | 3.1 | GPS stop tester | Verify each virtual stop has an associated GPS coordinate. |
|  |  | 3.2 | GPS format test | Verify GPS coordinate is in the correct format |
| 3.3 | GPS route test | Verify the GPS route list contains a proper GPS coordinates |
| 3.4 | GPS train test | Verify a train’s coordinate is updated and valid. |
| Ridership Data Control tester | 3.5 | Ridership Data generation test | Test virtual rider generation for each stop |
| Ridership Data Control tester | 3.6 | Ridership Data test | Ensure realistic proportions of riders are generated conforming to variable thresholds which can be changed by the user |
| Train control tester | 3.7 | Train GPS test | Verify each active train return assigned GPS coordinate |
| 3.8 | Train sensor failure test | Verify each train has the ability to simulate sensor failure |
| 3.9 | Train outage failure | Verify the ability for a train to no return a GPS coordinate. |
| 3.10 | Ridership test | Verify each train has the ability to return the current amount of riders on board |
| Business ad control tester | 3.11 | End date test | Verify access to each advertisement’s end date |
| 3.12 | Start date test | Verify access to each advertisement’s start date |
|  | 3.13 | Advertisement stop test | Verify Ability to assign and view advertisements assigned at each stop |
|  | 3.14 | Advertisement start time | Verify access to each advertisement’s start time |
| 3.15 | Advertisement end time | Verify access to each advertisement’s end time |
| Test harness Interface | 3.16 | Train property GUI | Verify interface has the ability to display each trains properties |
| 3.17 | Train settings test | Verify interface has the ability to edit different train settings |
| 3.18 | Stop property test | Verify interface has the ability to display ridership at each stop |
| 3.19 | Stop Property edit test | Verify interface has the ability to edit different ridership numbers at each stop. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| System | Alert Module | 4.1 | Manage Alerts | Verify an alert can be created/modified/closed within the HRT GUI |
| 4.2 | View Alerts | Demonstrate alerts are viewable in module |
| Unit | Feedback Module | 4.3 | Submit Feedback | Verify a rider or business user can submit feedback via web form |
| System | System Overview Module | 4.4 | Retrieve Stop Information | Verify DB interface provides results for stops & vehicles in operation |
| 4.5 | Map Overlay | Demonstrate stop & vehicle information is displayed on a dynamic map |
| Unit | Google Maps Web Form | 4.6 | Google Maps Web Redirection | Verify Google Maps search is performed for direction request |
| System | Calendar Event Module | 4.7 | View Events | Demonstrate Events are viewable within the module |
| 4.8 | Manage Events | Verify that Events can be added, edited or removed within the Business & HRT GUI |
| Unit | User Management | 5.1 | Webpage Layout | Verify user a able to insert name, desired user name, email address and password. |
| 5.2 | Username Validation | Ensure username is validated and return error if username exists. |
| 5.3 | Username Retrieve | Prove username is provided after validation method. |
| 5.4 | User Self Password Reset | Confirm user is able to reset password. |
| 5.5 | Administration Application | Validate administrator can change user information, groups, and passwords |
| 5.6 | User Information Update | Verify user can edit personal information. |
| 5.7 | Group Administration | Demonstrate the administrators ability to create and manage groups |
| Unit | Authentication Module | 5.8 | Application Access | Confirm user is able to access application through one-factor authentication. |
| 5.9 | Token Generation | Validate proper identification token generation. |
| 6.0 | Access Control | Prove access is granted only after token validation. |
| 6.1 | Security Time-out | Confirm user is unable to access the system after a predetermined time of non-use. |
| 6.2 | Account Locking | Validate user account is locked after multiple login failure attempts. |
| 6.3 | Logging | Verify recording of a users login time, location, authentication success or failure and the page requested. |
| 6.4 | Group Administration | Demonstrate the administrators ability to create and manage groups |
| Integration | Authentication Module | 6.5 | Database Connection | Confirm application opens access to the database for data transfer |
| 6.6 | Query Transfer | Validate query and result transfer between the database and the Web Application Engine modules. |
| 6.7 | Security | Demonstrate capability to prevent a SQL injection attack. |
| 6.8 | Decision Engine Connection | Verify data is sent between the Decision Engine and the Web Application Engine. |
| 6.9 | Test Harness Connection | Validate data is sent between the Test Harness and the Web Application Engine. |
| Unit | Business Ad Campaign Module | 7.1 | List Advertisements | Verify listing of advertisements. |
| 7.2 | Create Advertisement | Verify submission of advertisement input fields. |
| 7.3 | Edit Advertisement | Verify modification of advertisement input fields. |
| Integration | 7.4 | Database Interface | Verify interface with database for data input/output. |
| Unit | Ridership Trend Report | 8.1 | Display Default Report | Verify display of default ridership trend report. |
| 8.2 | Display Detailed Report | Verify display of custom ridership trend report. |
| 8.3 | Request Custom Report | Verify input fields for custom report query. |
| Integration | 8.4 | Decision Engine Interface | Verify interface with Decision Engine for data retrieval. |
| Unit | Train Data Report | 9.1 | Display Default Report | Verify display of default train data report. |
| 9.2 | Display Detailed Report | Verify display of custom |
| 9.3 | Request Custom Report | Verify input fields for custom report query. |
| Integration | 9.4 | Decision Engine Interface | Verify interface with Decision Engine for data retrieval. |
| Unit | Graphical User Interface Framework | 10.1 | Display Rider Modules | Verify display of modules on Rider GUI. |
| 10.2 | Display Business Modules | Verify display of modules on Business GUI. |
| 10.3 | Display HRT Modules | Verify display of modules on HRT GUI. |
| 10.4 | Module Interface | Verify interface with all Web Application Engine modules. |

Table 1. Test Category Identification

# 3.3 Test Schedule

The Current ITS Team will have a total of forty-five minutes to demonstrate the prototype of Current ITS. The first ten minutes of the presentation will be allotted to setup time and presentation of the scope of the prototype. Table 2 defines the testing schedule and the time allotted to each section of testing that will take place. After the conclusion of the prototype demonstration, the final fifteen minutes will be dedicated for the Current ITS team to take questions from the review board.

|  |  |  |  |
| --- | --- | --- | --- |
| Start Time (minutes) | Duration (minutes) | Description | Test Cases Covered |
| 0:00 | 10 | Feasibility |  |
| 0:10 | 5 | Database Demo | 1.0 |
| 0:15 | 10 | Algorithm Unit Tests | 2.0,3.0 |
| 0:25 | 10 | Integration Tests  -HRT  -Business  -Rider | 4.1 – 4.7 |
| 0:35 | 10 | System Tests  -HRT  -Business  -Rider | 5.0-10 |
| 0:45 | 15 | Q&A |  |

Table 2. Test Schedule

# 3.4 Fault Reporting and Data Recording

The Current ITS team will be recording the results, success or failure, of the tests during the prototype demonstration. The test components are defined as Database, Decision Engine Algorithms, Test Harness and Web App Engine GUI screens. Table 3 describes these testing components and the actions that will be taken to record potential failures of the components.

|  |  |
| --- | --- |
| **Component** | **Recording Process** |
| Database | * Report failures through visual inspection of database records using SQL queries * Document using paper forms |
| Test Harness GUI | * Report failures through visual inspection of output * Document using paper forms |
| Decision Engine Algorithms | * Report failures through visual inspection of output in log * Document using paper forms |
| Web App Engine GUI | * Report failures using visual inspection of GUI screens in Web App Engine and in Test Harness * Document using paper forms |

Table 3. Fault Reporting and Data Recording

# 3.5 Resource Requirements

To fully demonstrate the Current ITS prototype, the following hardware and software resources will be required. Hardware resources will consist of a desktop workstation with network connectivity to access the Current ITS server and the Test Harness. Software resources that make up Current ITS consist of CentOS 6 operating system software to run the Current ITS server, MySQL database software for the Current ITS database, PHPMyAdmin software to view the database, Apache webserver to host the Current ITS application.

(This Space Intentionally Left Blank)

# 3.6 Test Environment

The demonstration presentation for the Current ITS prototype will take place at Old Dominion University in Norfolk, VA, in the Gornto Center building second floor classroom. Figure 2 is the layout of the classroom being used for the presentation. The Current ITS team will use the front of the room, utilizing the workstation available to them to demonstrate the prototype. The CS 411 review board will be present as the audience for the demonstration.

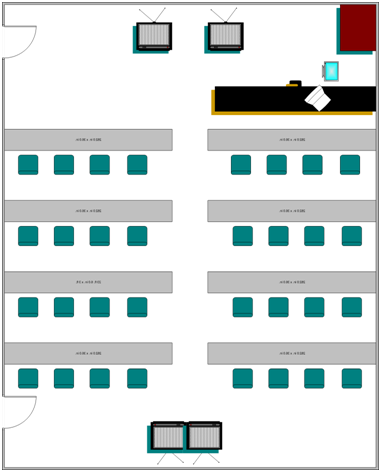


Figure 2. Presentation Layout

(This Space Intentionally Left Blank)

# 4. Test Responsibilities

Please see collaborative document Lab III Sections 4-6.

# 5. Test Procedures

Please see collaborative document Lab III Sections 4-6.

# 6. Traceability Requirements

Please see collaborative document Lab III Sections 4-6.

(This Space Intentionally Left Blank)