JagTrack
Test Plan

Version 1.0

Revision History

Date	Version	Description	Author
26/Apr/2012	1.0	Initial version	Xingyu Wang

Table of Contents

JagTrack Test Plan

Introduction

Purpose

The main purpose of this Test Plan is to ensure that the JagTrack application meets the necessary requirements defined by the Use Cases, Software Requirements Specifications, Supplementary Specifications, and/or other means.

This Test Plan document for the JagTrack Project supports the following objectives:

- Identify existing project information and the software components that should be tested.
- List the recommended Requirements for testing (high level).
- Recommend and describe the testing strategies to be employed.
- Identify the required resources and provide an estimate of the test efforts.
- List the deliverable elements of the test project.

Background

This Test Plan applies to the JagTrack application which is designed to provide various services for the passengers and administrators of the JagTran system at University of South Alabama. The major functions of the software are defined in the two fully dressed use cases: Get Arrival Time and Get Number of Passengers.

The software consists of three components: Web Application, Mobile Devices Application, and Database Server.

This Test Plan is based on the Test Plan template in the Rational Unified Process (RUP). The test process outlined in this document is based on the rational approach to software development and testing.

Scope

Test Stages outside of scope

- Acceptance Testing
- Installation Testing
- Unit Testing
- Integration Testing

Test Stages within scope

- Function Testing
- User Interface Testing
- Business Cycle Testing
- Database Integrity Testing
- Load, Stress, and Performance Testing
- Volume Testing
- Security and Access Control Testing

- Configuration Testing

1.1 Project Identification

The table below identifies the documentation and availability used for developing the *test plan*:

Document (and version / date)	Created or Available	Received or Reviewed	Author or Resource	Notes
Requirements Specification	Yes	Yes	Xingyu Wang	
Supplementary Specification	Yes	Yes	Jim Fletcher	
Functional Specification	No	No		
Use-Case Reports	Yes	Yes	Chase Bryant, Xingyu Wang	
Project Plan	Yes	Yes		
Design Specifications	No	No		
Prototype	Yes	Yes	Robert Fornof	
User's Manuals	No	No		
Business Model or Flow	No	No		
Data Model or Flow	Yes	Yes		
Business Functions and Rules	Yes	Yes	Leyue Wang	
Project or Business Risk Assessment	No	No		

2. Requirements for Test

The listing below identifies those items—use cases, functional requirements, and non-functional requirements—that have been identified as targets for testing. This list represents what will be tested.

- Functional requirements defined in the two fully dresses use cases: Get Arrival Time and Get Number of Passengers.
- Non-functional requirements specified in the Supplementary Specifications, including Usability, Reliability, Performance, and supportability.

Test Strategy

2.1 Testing Types

Data and Database Integrity Testing

Test Objective:	• Ensure database access methods and processes function properly and without data corruption.

Technique:	• Invoke each database access method and process, seeding each with valid and invalid data or requests for data. • Inspect the database to ensure the data has been populated as intended, all database events occurred properly, or review the returned data to ensure that the correct data was retrieved for the correct reasons.
Completion Criteria:	• All database access methods and processes function as designed and without any data corruption.
Special Considerations:	• Testing may require a DBMS development environment or drivers to enter or modify data directly in the databases. • Processes should be invoked manually. • Small or minimally sized databases (limited number of records) should be used to increase the visibility of any non-acceptable events. • Make sure that data from sensors could be written to the database properly.

Function Testing

Test Objective:	• Ensure proper application functionality, including navigation, data entry, processing, and retrieval.
Technique:	• Execute each use case, use-case flow, or function, using valid and invalid data, to verify the following: 1. The expected results occur when valid data is used in all test cases. 2. The appropriate error or warning messages are displayed when invalid data is used. 3. The correct arrival time of the next bus for a specific stop can be displayed to passengers when requested 4. The correct information of a bus can be displayed to administrators when requested.
Completion Criteria:	• All planned tests have been executed. • All identified defects have been addressed.
Special Considerations:	• Availability of test data.

Business Cycle Testing

This test will not be implemented or executed. This test is not appropriate currently.

Test Objective	N/A
Technique:	N/A
Completion Criteria:	N/A

3.1.4 User Interface Testing

Test Objective:	• Verify the following for both Android mobile devices UI and Web application UI: 1. Navigation through the target-of-test properly reflects business functions and requirements, including window-to-window, field-to-field, and use of access methods (tab keys, mouse movements, accelerator keys) 2. Window objects and characteristics, such as menus, size, position, state, and focus conform to standards.
Technique:	• Create or modify tests for each window to verify proper navigation and object states for each application window and objects.
Completion Criteria:	• Each window successfully verified to remain consistent with benchmark version or within acceptable standard.
Special Considerations:	None

3.1.5 Performance Profiling

Test Objective:	• Verify performance behaviors for designated transactions or business functions under the following conditions: 1. normal anticipated workload 2. anticipated worst case workload
Technique:	• Use Test Procedures developed for Function Testing. • Modify data files to increase the number of transactions or the scripts to increase the number of iterations each transaction occurs. • Scripts should be run on one machine (best case to benchmark single user, single transaction) and be repeated with multiple clients (virtual or actual, see Special Considerations below).
Completion Criteria:	• Single Transaction or single user: Successful completion of the test scripts without any failures and within the expected or required time allocation per transaction. For example, the arrival time of the next bus should be displayed to the passenger within the maximum response time specified in the Supplementary Specifications. • Multiple transactions or multiple users: Successful completion of the test scripts without any failures and within acceptable time allocation.

Special Considerations:	• Comprehensive performance testing includes having a background workload on the server. • There are several methods that can be used to perform this, including: "Drive transactions" directly to the server, usually in the form of Structured Query Language (SQL) calls. Create "virtual" user load to simulate many clients, usually several hundred. Remote Terminal Emulation tools are used to accomplish this load. This technique can also be used to load the network with "traffic". Use multiple physical clients, each running test scripts to place a load on the system. • Performance testing should be performed on a dedicated machine or at a dedicated time. This permits full control and accurate measurement. • The databases used for Performance Testing should be either actual size or scaled equally.

3.1.6 Load Testing

Test Objective:	• Verify performance behavior time for designated transactions or business cases under varying workload conditions.
Technique:	• Use tests developed for Function Testing. • Modify data files to increase the number of transactions or the tests to increase the number of times each transaction occurs.
Completion Criteria:	• Multiple transactions or multiple users: Successful completion of the tests without any failures and within acceptable time allocation.
Special Considerations:	• Load testing should be performed on a dedicated machine or at a dedicated time. This permits full control and accurate measurement. • The databases used for load testing should be either actual size or scaled equally.

Stress Testing

Test Objective:	• Verify that the application functions properly and without error under the following stress conditions: little or no memory available on the server (RAM and Direct Access Storage Device [DASD]); maximum actual or physically capable number of clients connected or simulated; multiple users performing the same transactions against the same data or accounts; worst case transaction volume or mix (see Performance Testing above). • Notes: The goal of Stress Testing might also be stated as identify and document the conditions under which the system FAILS to continue functioning properly. • Stress Testing of the client is described under section 3.1.11, Configuration Testing.
Technique:	• Use tests developed for Performance Profiling or Load Testing. • To test limited resources, tests should be run on a single machine, and RAM and DASD on server should be reduced or limited. • For remaining stress tests, multiple clients should be used, either running the same tests or complementary tests to produce the worst-case transaction volume or mix.
Completion Criteria:	• All planned tests are executed and specified system limits are reached or exceeded without the software failing or conditions under which system failure occurs is outside of the specified conditions.
Special Considerations:	• Stressing the network may require network tools to load the network with messages or packets. • The DASD used for the system should temporarily be reduced to restrict the available space for the database to grow. • Synchronization of the simultaneous clients accessing of the same records or data accounts.

Volume Testing

Test Objective:	• Verify that the target-of-test successfully functions under the following high volume scenarios: Maximum (actual or physically-capable) number of clients connected, or simulated, all performing the same, worst case (performance) business function for an extended period. Maximum database size has been reached (actual or scaled) and multiple queries or report transactions are executed simultaneously.
Technique:	• Use tests developed for Performance Profiling or Load Testing. • Multiple passengers and/or administrators should be used, either running the same tests or complementary tests to produce the worst-case transaction volume or mix (see Stress Testing above) for an extended period. • Maximum database size is created (actual, scaled, or filled with representative data) and multiple clients used to run queries and report transactions simultaneously for extended periods.
Completion Criteria:	• All planned tests have been executed and specified system limits are reached or exceeded without the software or software failing.
Special Considerations:	• What period of time would be considered an acceptable time for high volume conditions, as noted above?

3.1.9 Security and Access Control Testing

Test Objective:	• Application-level Security: Verify that an actor can access only those functions or data for which their user type is provided permissions. That is, the passenger cannot get access to those functions or data provided for administrators. The data comes from sensors and cannot be modified without permissions. • System-level Security: Verify that only those actors with access to the system and applications are permitted to access them.
Technique:	• Function/Data Security: Identify and list each user type and the functions or data each type has permissions for. • Create tests for each user type and verify each permission by creating transactions specific to each user type. • Modify user type and re-run tests for same users. In each case, verify those additional functions or data are correctly available or denied. • System-level Access: See Special Considerations below.

Completion Criteria:	• For each known actor type the appropriate function or data are available, and all transactions function as expected and run in prior Application Function tests.
Special Considerations:	• Access to the system must be reviewed or discussed with the appropriate network or systems administrator. This testing may not be required as it may be a function of network or systems administration.

3.1.10 Failover and Recovery Testing

This test will not be implemented or executed. This test is not appropriate currently.

Test Objective:	N/A
Technique:	N/A
Completion Criteria:	N/A
Special Considerations:	N/A

3.1.11 Configuration Testing

Test Objective:	• Verify that the application functions properly on both Android mobile devices (phones and tablets) and web browsers (IE, Chrome, Firefox, and Safari).
Technique:	• Execute all the test scripts on different browsers and different version of Android systems.
Completion Criteria:	• Same as for Function Testing.
Special Considerations:	• What non-target-of-test software is needed, is available, and is accessible on the desktop and Android mobile devices? • What applications are typically used? • What data are the applications running?

Tools

The following tools will be employed for this project:

	Tool	Vendor/In-house	Version
Test Management	JUnit	Open Source	
Defect Tracking	GitHub	GitHub	
ASQ Tool for functional testing	TBD		
ASQ Tool for performance testing	TBD		
Test Coverage Monitor or Profiler	TBD		

Project Management	GitHub, Email	Outside	
DBMS tools	PHPMyAdmin, MySQL	Oracle	

Resources

Roles

This table shows the staffing assumptions for the project.

Human Resources		
Worker	Minimum Resources Recommended (number of full-time roles allocated)	Specific Responsibilities or Comments
Test Manager, Test Project Manager		Provides management oversight. Responsibilities: • provide technical direction • acquire appropriate resources • provide management reporting
Test Designer		Identifies, prioritizes, and implements test cases. Responsibilities: • generate test plan • generate test model • evaluate effectiveness of test effort
Tester		Executes the tests. Responsibilities: • execute tests • log results • recover from errors • document change requests
Test System Administrator		Ensures test environment and assets are managed and maintained. Responsibilities: • administer test management system • install and manage access to test systems
Database Administrator, Database Manager		Ensures test data (database) environment and assets are managed and maintained. Responsibilities: • administer test data (database)

Designer		Identifies and defines the operations, attributes, and associations of the test classes. Responsibilities: • identifies and defines the test classes • identifies and defines the test packages
Implementer		Implements and unit tests the test classes and test packages. Responsibilities: • creates the test classes and packages implemented in the test model

System

The following table sets forth the system resources for the testing project.

System Resources	
Resource	Name / Type
Database Server	
Network or Subnet	dotCloud
Server Name	Apache
Database Name	MySQL
Client Test PC's	
Include special configuration requirements	None
Test Repository	
Network or Subnet	TBD
Server Name	TBD
Test Development PC's	Tester's Machine (Linux/Windows)

Project Milestones

Milestone Task	Effort	Start Date	End Date
Plan Test	See Appendix A	04/20/2012	04/25/2012
Design Test	See Appendix A	04/25/2012	04/26/2012
Implement Test	See Appendix A	04/26/2012	05/01/2012
Execute Test	See Appendix A	04/26/2012	05/01/2012
Evaluate Test	See Appendix A	04/26/2012	05/01/2012

Deliverables

Test Model

The artifact Test Evaluation Summary will be created from the test model. This should be done by the Test Team.

2.2 Test Logs

Recording and reporting on the test results and testing status are through the use of GitHub.

2.3 Defect Reports

GitHub is used to record, track, and report on test incidents and their status.

Appendix A Project Tasks

Below are the test-related tasks:

- Plan Test
  - identify requirements for test
  - assess risk
  - develop test strategy
  - identify test resources
  - create schedule
  - generate Test Plan
- Design Test
  - prepare workload analysis X
  - identify and describe test cases
  - identify and structure test procedures
  - review and assess test coverage
- Implement Test
  - record or program test scripts
  - identify test-specific functionality in the Design and Implementation Model
  - establish external data sets
- Execute Test
  - execute Test procedures
  - evaluate execution of Test
  - recover from halted Test
  - verify the results
  - investigate unexpected results
  - log defects
- Evaluate Test
  - evaluate Test-case coverage
  - evaluate code coverage
  - analyze defects
  - determine if Test Completion Criteria and Success Criteria have been achieved