TRANSIT DROID

Test Plan

Transit Droid Team

Austin Takam
Daniel Magni
Paul Smelser
Razvan-Lada Moldovan
Yasser Al-Hasan

TABLE OF CONTENTS

REVISION HISTORY

Date	Version	Description
02/21/2013	1.0	Initial Test Plan
03/07/2013	1.1	Added extra testing levels(System and User Acceptance)
03/13/2013	1.2	Added test item exclusion rationale

1 INTRODUCTION

1.1 PURPOSE

The purpose of this Test Plan is to gather all of the information necessary to plan and control the test effort throughout the duration of the Transit Droid project. It describes the approach to testing the software, and is the top-level plan generated and used by the Transit Droid testing team to direct the test effort.

This Master Test Plan for the Transit Droid project supports the following objectives:

- Identifies the items that should be targeted by the tests.
- Identifies the motivation for and ideas behind the test areas to be covered.
- Outlines the testing approach that will be used.
- Identifies the required resources and provides an estimate of the test efforts.
- Lists the deliverable elements of the test project.

1.2 SCOPE

For the purposing of testing the Transit Droid system, the testing team has defined four distinct levels of testing, namely Unit Testing, Integration Testing, System Testing and User Acceptance Testing. More specifically, the unit testing effort will employ a white box testing technique to test each class and its methods in isolation. The integration testing will take the aforementioned unit tested items and will integrate them individually to ensure valid data flow amongst integrated modules. The system testing effort will be broken into three subcomponents: Graphical User Interface/Usability Testing, Stress Testing, and Load Testing. Finally, the User Acceptance Testing effort will mainly consist of asserting that the requirements of the project are met by testing them against the software under test.

The testing effort will exclude the rigorous testing of most domain layer objects. The reason for this is that the majority of the domain layer objects are POJOs (Plain Old Java Objects) that are basic bean classes consisting of simple, automatically generated mutator and accessor methods. Therefore, if certain domain

objects go beyond the scope of a simple bean definition, then these specific items will be targeted for further unit testing of their methods.

The entire system under test is the interplay and interaction between three main facets: The web application component, the Android component, and the web service infrastructure that allows both components to interact with each other and the datastore. For the sake of brevity and with respect with the time constraints of the project, stress and load testing will only be simulated using respective testing tools using a single server.

The rationale for this is that in production, the use of web services can be used in conjunction with a distributed system of servers that will service requests much more easily under a heavy throughput than under the development environment, using a single server.

1.3 Intended Audience

The intended audience for this Test Plan is split into three groups. The first group, the Transit Droid development team, will use this Test Plan as a guide for testing the development efforts.

The second group, the grading team, namely the professor and teaching assistants will be able to use this Test Plan as an evaluation guide to the process of testing that the Transit Droid team has taken in order to deliver a verifiable and valid end system.

Lastly, the final group is the Societe Des Transports De Montreal (STM), the industrial stakeholder of the Transit Droid project. The intended purpose of this Test Plan for the STM is to ensure that a solid testing process has been implemented for their product, and that through this testing process will minimize the probability of bugs when in production.

1.4 Document Terminology and Acronyms

There is no specific terminology used explicitly in the Test Plan that is not referenced in the project Glossary, which will be referenced in the next subsection.

1.5 References

Transit Droid Project Glossary

Transit Droid Software Requirements and Specifications Document

1.6 Document Structure

The following sections of the Test Plan consist of several important aspects of the testing process. The document will introduce the motivations behind the testing effort, and will then go into specific details of the different test cases and process for each level of testing that the Transit Droid team will target. Additionally, the Test Plan will also give an approximate timetable for testing workflow as well as a section describing the bug resolution process with respect to the testing effort.

2 EVALUATION MISSION AND TEST MOTIVATION

2.1 BACKGROUND

The Transit Droid project is an undertaking to provide a mobile alternative to the current STM Opus Card ticketing system. The mobile system will consist of two fronts: The Android platform that will interface with current STM terminal readers via the Near Field Communication technology, in order for an individual to manage and use their Opus account through a mobile means. Secondly, the Transit Droid system also consists of a web application end, which will serve similar purposes of Opus account management as the Android application, however excluding the use of tickets at terminal readers located in the public transit network.

These two systems will interface with a web service framework that will provide the end user with a consistent and cohesive way of managing their Opus account, something that STM clients were not able to do before this.

As this system will be potentially used by a majority of the city's public transit network, it is imperative to implement a structured test process to ensure a secure and bug-free usage, in which this Test Plan aims to address.

2.2 EVALUATION MISSION

The evaluation mission of this Test Plan is to verify that the Transit Droid project has maintained a structured, organized process for testing of the application, in such a way as to find and resolve as many bugs as possible in a timely and appropriate manner, as well as to validate and verify user requirements and specifications against the functioning of the system. This will ensure a quality product that can provide utility to the stakeholders of the project.

2.3 TEST MOTIVATORS

The key motivators of the testing effort is to ensure that the portion of the Montreal population that use the Montreal public transit system will be able to use the Transit Droid system in an accurate manner and that the system will be secure enough to shield the public from proprietary encryption methods utilized by the STM. This motivation ultimately will stem from the proper design and implementation of technical and

non-technical requirements, which will be rigorously tested using the process detailed in this Test Plan. The Transit Droid team is confident that by dividing the testing effort into distinct levels, each on that builds on top of its lower level, will provide with development team with the best way to minimize bugs while not compromising product quality and reliability.

3 TARGET TEST ITEMS

The following items have been selected as target test items for the Transit Droid system. They represent the software, hardware and supporting product elements that have been identified as targets for testing. The test items will be broken down into the two subcomponents of the Transit Droid project, and will describe target test items specific to each subsystem

3.1 TRANSIT DROID WEB APPLICATION

For the Transit Droid Web Application, the following is a **high-level** list of target test items, each sectioned into their appropriate testing level (Unit, Integration, User Acceptance, etc.). Furthermore, the list items will be given a category of assigned importance from the set: {**L=LOW, M=MEDIUM, H=HIGH**}.

Unit Testing Level

For unit testing, we shall cover the following list of high level test items as well as hardware and other aspects that should be considered. Each of the below modules represents the general area that will be tested and that will be described in more detail in the next section of the Test Plan:

Data Modules

- Card (**H)**
- Pass (**H**)
- Phone (**M**)
- User (**H**)

Domain Modules

- Card (**H**)
- Checkpoint (**L**)
- Contract (**H**)
- Pass (**L**)
- Phone (**M**)
- User (**H**)

Service Modules

- AccountManager (**H**)
- CreateAccountRequest (**M**)
- CreateAccountResponse (**M**)
- Transaction Manager (**H**)
- TransactionRequest (**M**)
- TranscationResponse (**M**)

Technical Modules

- EncryptionService (**H**)
- IdFactory (**L**)
- Registry (**M**)

In addition to the aforementioned modules, the unit testing process will also target:

- Java JDK/JRE 6/7
- Apache Tomcat 7
- Windows 7 Premium
- Windows 8 Pro
- Mac OS X Mountain Lion
- Google Chrome(v.25), Mozilla Firefox(v.19), Internet Explorer 10, Safari 6.0
- HP G60-551CA Notebook
- HP Pavilion AMD Phenom 6 Core 1065T Server

Integration Testing Level

The targeted items for the next level of integration for the Transit Droid Web Application, the integration level, will use the same modules that were tested in isolation in the unit testing phase, therefore they shall not be described again in this section. However, the integration testing level is interested in the data interaction between the previously unit tested items. Most integration tests will take the form of an element in a certain layer of the tiered architecture interacting with an element in a layer higher or lower than it (This is not to say that integration testing will ignore interaction between two given elements in the same architectural layer, however, the main goal is that technical services, domain logic and presentation layers can communicate correctly with one another. Some example integration tests would include:

- Interaction of User modules with User Data Gateway modules (**H**)
- Interaction of UserAction modules with User Domain Object modules (**H**)
- Interaction of Encryption Service modules with Key modules (**H**)

System Testing Level

The system level testing for the Transit Droid Web Application will be threefold – we will be performing GUI testing, stress testing and load testing on the system. The following is a high-level list of items that will be targeted with respect to this level of testing:

GUI Testing

- HTML presentation markup (**M**)
- JavaScript functionality (**H**)
- Manage Opus account (**M**)
- Login/logout (**H**)
- Purchase Opus fare (**H**)

Stress Testing

- Testing web application response in use with a specified upper bound of concurrent traffic.(**H**)

Load Testing

- Testing web application response in use with the expected average number of concurrent traffic. (**M**)

User Acceptance Testing Level

The user acceptance testing level will be targeting the requirements of the project that can be found in the Software Requirements and Specification. This level of testing is to verify that the requirements were carried out correctly and is to the satisfaction of the stakeholder. The primary targets for the web application will be:

- Login/Logout (**H**)
- Purchase Tickets (**H**)

- Add Device to Account (**M**)
- View Schedule (**L**)
- Account CRUD (Create, Read, Update, Delete) (**H**)

3.2 TRANSIT DROID ANDROID APPLICATION

For the Transit Droid Android Application, the following is a **high-level** list of target test items, each sectioned into their appropriate testing level (Unit, Integration, User Acceptance, etc.). Furthermore, the list items will be given a category of assigned importance from the set: {**L=LOW, M=MEDIUM, H=HIGH**}.

Unit Testing Level

For unit testing the Android application, the Transit Droid team will focus mainly on the main activity module of the application. As the Android architecture usually contains few Activity classes to implement an application, we will be adding testing effort to the main activity class:

- OpusActivity (**H**)

Integration Testing Level

As the Android application will have very little interaction with any other Android modules, we shall defer integration testing of the Android subsystem until a significant number of modules are implemented in the future, as the application expands. The primary focus for the Android subsystem will be more on GUI testing and acceptance tests.

System Testing Level

The system testing for the Android subsystem will primarily take the form of GUI testing, as stress and load testing will not be an issue for the single user Android application. The high level GUI testing items that look to be targeted are:

- Proper configuration of XML GUI layout files (**H**)

- Logging in/out (**H**)
- Manage Opus account (**M**)
- Using Opus fare (**H**)
- Purchase Opus fare (**H**)

User Acceptance Testing Level

The user acceptance testing for the Android application will be very similar to the Web application, as they are practically the same use cases (with the addition of the utilization of Opus fare for the Android application). Therefore the high level list of target test items for acceptance testing is:

- Login/Logout (**H**)
- Purchase Tickets (**H**)
- Add Device to Account (**M**)
- View Schedule (**L**)
- Account CRUD (Create, Read, Update, Delete) (**H**)
- Enter Subway/Bus (**H**)

4 OUTLINE OF PLANNED TESTS

This section will outline in detail the items that will be included and excluded from testing by the testing team.

4.1 OUTLINE OF TEST INCLUSIONS

The following section will outline the items that will be tested from the unit testing phase to the user acceptance phase. Each phase must be completed before moving to the next phase in the testing hierarchy.

Unit Testing

The following classes and methods will be included for testing on the unit test level. Formal test cases will not be shown, nor is this list exhaustive for the sake of brevity. The Test Log document will have a more in depth description of the actual unit tests However, the following items will be thoroughly tested:

Web Application Unit Tests

AccountManager

- UTC-01: addContract(String name)

TransactionManager

- UTC-02: process(TransactionRequest request)

UnitOfWork

- UTC-03: commit()
- UTC-04: registerDirty(O object)
- UTC-05: registerNew(O object)
- UTC-06: registerRemoved(O Object)

MapperRepository

- UTC-07: getInputMapper(Class<?> objectClass)
- UTC-08: getOutputMapper(Class<?> objectClass)

CardTIG

- UTC-09: findAll()

- UTC-10: find(UUID id)

CardTOG

- UTC-11: insert(UUID uuid, int version, UUID uuid3, UUID uuid4, UUID uuid2, UUID uuid5,UUID uuid6, UUID uuid7)
- UTC-12: findAll()
- UTC-13: find(long id)

DailyPassTIG

- UTC-14: find(UUID id)

DailyPassTOG

- UTC-15 insert(UUID uuid, int version, Date date)
- UTC-16: findAll()
- UTC-17: find(long id)

MonthlyPassTIG

- UTC-18: findAll()
- UTC-19: find(UUID id)

MonthlyPassTOG

- UTC-20: insert(UUID uuid, int version)
- UTC-21: findAll()
- UTC-22: find(long id)

PhoneTIG

- UTC-23: findAll()
- UTC-24: find(UUID id)

PhoneTOG

- UTC-25: insert(UUID uuid, int version, String phoneMAC)
- UTC-26: findAll()
- UTC-27: find(long id)

Card

- UTC-28: Card(UUID id, int version, MonthlyPass monthlyPass, NightlyPass nightlyPass, DailyPass, dailyPass, SinglePass singlePass, ThreeDayPass threeDayPass, YearlyPass yearlyPass)

CardFactory

- UTC-29: createNew(MonthlyPass monthlyPass, NightlyPass nightlyPass, DailyPass dailyPass, SinglePass singlePass, ThreeDayPass threeDayPass, YearlyPass yearlyPass)
- UTC-30: createClean(UUID id, int version, MonthlyPass monthlyPass, NightlyPass nightlyPass, DailyPass,dailyPass, SinglePass, singlePass, ThreeDayPass threeDayPass, YearlyPass yearlyPass)

ContractFactory

- UTC-31: createNew(UUID id, List<byte[]> keys, IPhone phoneId, Card card)
- UTC-32: createNew(List<byte> keys, IPhone phoneId, Card card)

PhoneIdentityMap

- UTC-33: getUniqueInstance()
- UTC-34: get(UUID id)
- UTC-35: put(UUID uuid, Phone phone)

CardIdentityMap

- UTC-36: getUniqueInstance()
- UTC-37: get(UUID id)
- UTC-38: put(UUID id, Card card)

UserProxy

- UTC-39: getFromMapper(UUID id)

BaseProxy

- UTC-40: getFromMapper(UUID id)
- UTC-41: getInnerObject()

DailyPassProxy

- UTC-42: hashCode()
- UTC-43: getFromMapper(UUID id)

YearlyPassProxy

- UTC-44: hashCode()
- UTC-45: getFromMapper(UUID id)

Android Application Unit Tests

MainActivity

- UTC-46: onCreateOptionsMenu(Menu menu)
- UTC-47: setContentView()
- UTC-48: onSlideMenuItemClick(int itemId)
- UTC-49: opusCard()
- UTC-50: handleTag(Intent intent)
- UTC-51: profile()
- UTC-52: account()
- UTC-53: settings()
- UTC-54: getPath()
- UTC-55: saveSettings(View v)

System Testing

The following section will describe the several test cases that will be targeted at the System Testing level.

Test Case Field	**Details**
Test Case ID: Test Case Name	STC-01: Successful login-Web
Purpose	To verify if a user is able to login successfully with the proper login credentials
Pre-Conditions	User has already created a Transit Droid account
Execution Steps	1. Browse to Transit Droid website on internet browser 2. Enter correct user name 3. Enter correct password 4. Click Login button
Expected Results	User is successfully logged in and has access to the site
Environmental Needs	Windows 7/8, Google Chrome Browser v.25

Tested By	Daniel Magni

Test Case Field	**Details**
Test Case ID: Test Case Name	STC-02: Failed Login-Web
Purpose	To verify that a user with invalid credentials does not have access to the Transit Droid site
Pre-Conditions	User is on Transit Droid website
Execution Steps	1. Enter either incorrect username and/or password 2. Click Login button
Expected Results	Red error message telling user that login credentials are invalid, please try again. Also provide optional link to create account and password recovery.
Environmental Needs	Windows 7/8, Google Chrome Browser v.25
Tested By	Daniel Magni

Test Case Field	**Details**
Test Case ID: Test Case Name	STC-03: Forgot Password-Web
Purpose	To verify that a user can recover their login information by email recovery
Pre-Conditions	User has a valid Transit Droid account
Execution Steps	1. Click the 'Forgot Password?' link 2. Enter username in input box 3. Click 'Recover my password' button
Expected Results	Message is displayed to check email for password reset instructions
Environmental Needs	Windows 7/8, Google Chrome Browser v.25
Tested By	Daniel Magni

Test Case Field	**Details**
Test Case ID: Test Case Name	STC-04: Purchase Daily Pass-Web
Purpose	To verify that a user is able to purchase a daily Transit Droid ticket from the Transit Droid website
Pre-Conditions	User has an account with Transit Droid. Logged in.
Execution Steps	1. Click on the 'Purchase Tickets' link 2. Select the Daily option 3. Select Quantity

	4. Click Continue button
Expected Results	User is forwarded to the payment confirmation screen
Environmental Needs	Windows 7/8, Google Chrome Browser v.25
Tested By	Daniel Magni

Test Case Field	**Details**
Test Case ID: Test Case Name	STC-05: Purchase Monthly Pass-Web
Purpose	To verify that a user is able to purchase a monthly pass from the Transit Droid website
Pre-Conditions	User has an account with Transit Droid. Logged in.
Execution Steps	1. Click on the 'Purchase Tickets' link 2. Select the Monthly option 3. Click Continue button
Expected Results	User is forwarded to the payment confirmation screen. Purchase will be for the current month.
Environmental Needs	Windows 7/8, Google Chrome Browser v.25
Tested By	Daniel Magni

Test Case Field	**Details**
Test Case ID: Test Case Name	STC-06: Purchase Three Day Pass-Web
Purpose	To verify that a user is able to purchase a three day pass from the Transit Droid website
Pre-Conditions	User has an account with Transit Droid. Logged in.
Execution Steps	1. Click on the 'Purchase Tickets' link 2. Select the Three Day Pass option 3. Select start date from calendar 4. Click Continue button
Expected Results	User is forwarded to the payment confirmation screen.
Environmental Needs	Windows 7/8, Google Chrome Browser v.25
Tested By	Daniel Magni

Test Case Field	**Details**
Test Case ID: Test Case Name	STC-07: Purchase Nightly Pass-Web
Purpose	To verify that a user is able to purchase a three day pass from the Transit Droid website
Pre-Conditions	User has an account with Transit Droid. Logged in.
Execution Steps	1. Click on the 'Purchase Tickets' link 2. Select the Nightly Pass option 3. Click Continue button

Expected Results	User is forwarded to the payment confirmation screen. Nightly pass is only valid for the night of purchase
Environmental Needs	Windows 7/8, Google Chrome Browser v.25
Tested By	Daniel Magni

Test Case Field	**Details**
Test Case ID: Test Case Name	STC-08: Purchase Yearly Pass-Web
Purpose	To verify that a user is able to purchase a yearly pass from the Transit Droid website
Pre-Conditions	User has an account with Transit Droid. Logged in
Execution Steps	1. Click on the 'Purchase Tickets' link 2. Select the Yearly Pass option 3. Click Continue button
Expected Results	User is forwarded to the payment confirmation screen. Ticket will be valid until end of current year only
Environmental Needs	Windows 7/8, Google Chrome Browser v.25
Tested By	Daniel Magni

Test Case Field	**Details**
Test Case ID: Test Case Name	STC-09: Create Account-Web
Purpose	To verify if a user is able to successfully register for a new account on the Transit Droid website
Pre-Conditions	User has a capable web browser
Execution Steps	1. Browse to Transit Droid website 2. Click the 'Sign Up' link 3. Enter a unique user ID for the user name 4. Enter a valid email address 5. Enter a first and last name 6. Enter a phone number 7. Click Submit button
Expected Results	A page that confirms that the account has been created successfully. The account is automatically
Environmental Needs	Windows 7/8, Google Chrome Browser v.25
Tested By	Daniel Magni

Test Case Field	**Details**
Test Case ID: Test Case Name	STC-10: View Transit Schedule-Web
Purpose	To verify that a user is able to view the schedule of bus/metro times

Pre-Conditions	User has a valid account and is logged in
Execution Steps	1. Browse to 'View Schedule' link and click it. 2. Select desired bus route from dropdown list 3. Select desired direction from dropdown list 4. Click 'Submit' button
Expected Results	User is able to view a schedule of upcoming buses/metro
Environmental Needs	Windows 7/8, Google Chrome Browser v.25
Tested By	Daniel Magni

Test Case Field	**Details**
Test Case ID: Test Case Name	STC-11: Add A Device To Account-Web
Purpose	To verify that a user is able to add a mobile device to their account
Pre-Conditions	User has an existing account and is logged in
Execution Steps	1. Click on the 'Add Device' link 2. Enter the IMEI of the device 3. Click the 'Submit' button
Expected Results	Device is added to account
Environmental Needs	Windows 7/8, Google Chrome Browser v.25
Tested By	Daniel Magni

Test Case Field	**Details**
Test Case ID: Test Case Name	STC-12: Successful Login-Android
Purpose	To verify that a user is able to successfully log in their Transit Droid account via their mobile device
Pre-Conditions	The user already is registered with a valid account.
Execution Steps	1. Open the Transit Droid application 2. On the main login screen, type username into the username field 3. Type the password into the password field 4. Tap on the 'Login' button
Expected Results	The user is able to fully access the Transit Droid application
Environmental Needs	Samsung Galaxy SIII, Jelly Bean (4.1) Android OS
Tested By	Daniel Magni

Test Case Field	Details
Test Case ID: Test Case Name	STC-13: Failed Login-Android
Purpose	To verify that a user with invalid login credentials
Pre-Conditions	User is on Transit Droid website
Execution Steps	1. Enter invalid username and/or password 2. Click Login button
Expected Results	Red error message telling user that login credentials are invalid, please try again. Also provide optional link to create account and password recovery.
Environmental Needs	Samsung Galaxy SIII, Jelly Bean (4.1) Android OS
Tested By	Daniel Magni

Test Case Field	Details
Test Case ID: Test Case Name	STC-14: Purchase Daily Pass-Android
Purpose	To verify that a user is able to purchase a three day pass from the Transit Droid Android application
Pre-Conditions	User has a valid account and is logged in
Execution Steps	1. Tap on the 'Purchase Tickets' link 2. Select the Daily Pass option 3. Tap Continue button
Expected Results	User is forwarded to the payment confirmation screen
Environmental Needs	Samsung Galaxy SIII, Jelly Bean (4.1) Android OS
Tested By	Daniel Magni

Test Case Field	Details
Test Case ID: Test Case Name	STC-15: Purchase Three Day Pass-Android
Purpose	To verify that a user is able to purchase a three day pass from the Transit Droid Android application
Pre-Conditions	User has a valid account and is logged in
Execution Steps	1. Tap on the 'Purchase Tickets' link 2. Select the Three Day Pass option 3. Select the start date by tapping on the calendar widget 4. Tap Continue button
Expected Results	User is forwarded to the payment confirmation screen
Environmental Needs	Samsung Galaxy SIII, Jelly Bean (4.1) Android OS
Tested By	Daniel Magni

Test Case Field	Details
Test Case ID: Test Case Name	STC-16: Purchase Nightly Pass-Android
Purpose	To verify that a user is able to purchase a nightly pass from the Transit Droid Android application
Pre-Conditions	User has a valid account and is logged in
Execution Steps	1. Tap on the 'Purchase Tickets' link 2. Select the Nightly option 3. Tap Continue button
Expected Results	User is forwarded to the payment confirmation screen. Nightly pass is only valid for the night of purchase
Environmental Needs	Samsung Galaxy SIII, Jelly Bean (4.1) Android OS
Tested By	Daniel Magni

Test Case Field	Details
Test Case ID: Test Case Name	STC-17: Purchase Monthly Pass-Android
Purpose	To verify that a user is able to purchase a monthly pass from the Transit Droid Android application
Pre-Conditions	User has a valid account and is logged in.
Execution Steps	4. Tap on the 'Purchase Tickets' link 5. Select the Monthly Pass option 6. Tap Continue button
Expected Results	User is forwarded to the payment confirmation screen. Monthly pass is valid only for the current month.
Environmental Needs	Samsung Galaxy SIII, Jelly Bean (4.1) Android OS
Tested By	Daniel Magni

Test Case Field	Details
Test Case ID: Test Case Name	STC-18: Purchase Yearly Pass-Android
Purpose	To verify that a user is able to purchase a yearly pass from the Transit Droid Android application
Pre-Conditions	User has a valid account and is logged in.
Execution Steps	1. Tap on the 'Purchase Tickets' link 2. Select the Yearly Pass option 3. Tap Continue button
Expected Results	User is forwarded to the payment confirmation screen. Yearly pass is valid only for the current year.
Environmental Needs	Samsung Galaxy SIII, Jelly Bean (4.1) Android OS
Tested By	Daniel Magni

Test Case Field	Details
Test Case ID: Test Case Name	STC-19: Forgot Password-Android
Purpose	To verify that a user can recover their login information by email recovery
Pre-Conditions	User is on main login screen of the Android application
Execution Steps	1. Click the 'Forgot Password?' link 2. Enter username in input box 3. Click 'Recover my password' button
Expected Results	Message is displayed to check email for password reset instructions
Environmental Needs	Samsung Galaxy SIII, Jelly Bean (4.1) Android OS
Tested By	Daniel Magni

Test Case Field	Details
Test Case ID: Test Case Name	STC-20: Scan Device Through Reader-Single Pass Use-Android
Purpose	To verify that a user can successfully enter the subway or bus by swiping the Transit Droid application
Pre-Conditions	User is logged in and has at least one daily pass on account
Execution Steps	1. Scan phone near STM Terminal Reader 2. Wait for a beep to confirm communication between the phone and the reader
Expected Results	The device decrements the number of single tickets used by one.
Environmental Needs	Samsung Galaxy SIII, Jelly Bean (4.1) Android OS, STM Terminal Reader
Tested By	Daniel Magni

Test Case Field	Details
Test Case ID: Test Case Name	STC-21: View Schedule-Android
Purpose	To verify that a user is able to view the schedule of bus/metro times
Pre-Conditions	User is logged into the Android Application
Execution Steps	1. Tap the 'View Schedule' link 2. Select bus route from dropdown list 3. Select direction from dropdown list

	4. Tap the ‘View’ button
Expected Results	User is presented with the desired bus route and direction schedule
Environmental Needs	Samsung Galaxy SIII, Jelly Bean (4.1) Android OS
Tested By	Daniel Magni

Test Case Field	**Details**
Test Case ID: Test Case Name	STC-22: Create Account-Android
Purpose	To verify if a user is able to successfully register for a new account on the Transit Droid Android application
Pre-Conditions	The user does not already have an account. Is on the Transit Droid login screen
Execution Steps	1. Click the ‘Create Account’ link 2. Enter the correct username
Expected Results	User is shown a welcome/confirmation of account creation and is logged into the
Environmental Needs	Samsung Galaxy SIII, Jelly Bean (4.1) Android OS
Tested By	Daniel Magni

Test Case Field	**Details**
Test Case ID: Test Case Name	STC-23: Purchase Weekly Pass-Android
Purpose	To verify that a user is able to purchase a weekly Transit Droid ticket from the Transit Droid Android application
Pre-Conditions	User has a valid account and is logged in.
Execution Steps	1. Click on the ‘Purchase Tickets’ link 2. Select the Weekly option 3. Select the start date of the weekly pass from the calendar widget 4. Click the ‘Continue’ button
Expected Results	User is forwarded to the payment confirmation screen
Environmental Needs	Samsung Galaxy SIII, Jelly Bean (4.1) Android OS
Tested By	Daniel Magni

Test Case Field	**Details**
Test Case ID: Test Case Name	STC-24: Purchase Weekly Pass-Web
Purpose	To verify that a user is able to purchase a weekly Transit Droid ticket from the Transit Droid website

Pre-Conditions	User has a valid account and is logged in.
Execution Steps	1. Click on the 'Purchase Tickets' link 2. Select the Weekly option 3. Select the start date of the weekly pass from the calendar 4. Click the 'Continue' button
Expected Results	User is forwarded to the payment confirmation screen
Environmental Needs	Windows 7/8, Google Chrome Browser v.25
Tested By	Daniel Magni

Stress Testing

[Will be included in the next deliverable milestone]

Load Testing

[Will be included in the next deliverable milestone]

User Acceptance Testing

The following section describes the acceptance tests that were developed for the stakeholder to perform in order to validate our system's requirements:

Test Case Field	**Details**
Test Case ID: Test Case Name	ATC-01: Create An Account-Web
Purpose	To validate that a user is able to create a Transit Droid account
Description	User creates a new account through the registration process

Test Case Field	**Details**
Test Case ID: Test Case Name	ATC-02: Login-Web
Purpose	To validate that a user can log into the Transit Droid website

Description	User enters a valid username/password and log into the Transit Droid website

Test Case Field	**Details**
Test Case ID: Test Case Name	ATC-03: Change Language-Web
Purpose	To validate that a user can change their language at any point during their interaction with the Transit Droid website
Description	User toggles between English and French at any point during the interaction with the Transit Droid website

Test Case Field	**Details**
Test Case ID: Test Case Name	ATC-04: Purchase a Daily Pass-Web
Purpose	To validate that a user can successfully purchase a daily pass
Description	User adds a daily pass to their Transit Droid account

Test Case Field	**Details**
Test Case ID: Test Case Name	ATC-05: Purchase a Three Day Pass-Web
Purpose	To validate that a user can successfully purchase a three day pass
Description	User adds a three day pass to their Transit Droid account

Test Case Field	**Details**
Test Case ID: Test Case Name	ATC-06: Purchase a Nightly Pass-Web
Purpose	To validate that a user can successfully purchase a nightly pass
Description	User adds a nightly pass to their Transit Droid account

Test Case Field	**Details**
Test Case ID: Test Case Name	ATC-07: Purchase a Weekly Pass-Web

Purpose	To validate that a user can successfully purchase a weekly pass
Description	User adds a weekly pass to their Transit Droid account

Test Case Field	**Details**
Test Case ID: Test Case Name	ATC-08: Purchase a Monthly Pass-Web
Purpose	To validate that a user can successfully purchase a monthly pass
Description	User adds a monthly pass to their Transit Droid account

Test Case Field	**Details**
Test Case ID: Test Case Name	ATC-09: Purchase a Yearly Pass-Web
Purpose	To validate that a user can successfully purchase a yearly pass
Description	User adds a yearly pass to their Transit Droid account

Test Case Field	**Details**
Test Case ID: Test Case Name	ATC-10: View Schedule-Web
Purpose	To validate that a user can successfully view a schedule of upcoming bus times for any desired bus route
Description	User browses to the schedules section of website, enters a desired bus route/direction and is able to see the next available times for their choice

Test Case Field	**Details**
Test Case ID: Test Case Name	ATC-11: Add a device to the account-Web
Purpose	To validate that a user can successfully add a mobile device to their active account
Description	User enters the device information in the ‘Add Device’ section of the website and can thus add fare to this device

Test Case Field	Details
Test Case ID: Test Case Name	ATC-12: Create an Account-Android
Purpose	To validate that a user can successfully create a new account in the Android application
Description	User enters personal information in the registration section of the website

Test Case Field	Details
Test Case ID: Test Case Name	ATC-14: Purchase a Daily Pass-Android
Purpose	To validate that a user can successfully purchase a daily pass using their Android device
Description	User adds a daily pass to their account

Test Case Field	Details
Test Case ID: Test Case Name	ATC-15: Purchase a Three Day Pass-Android
Purpose	To validate that a user can successfully purchase a three day pass using their Android device
Description	User adds a three day pass to their account

Test Case Field	Details
Test Case ID: Test Case Name	ATC-16: Purchase a Nightly Pass-Android
Purpose	To validate that a user can successfully purchase a nightly pass using their Android device
Description	User adds a nightly pass to their account

Test Case Field	Details
Test Case ID: Test Case Name	ATC-17: Purchase a Weekly Pass-Android
Purpose	To validate that a user can successfully purchase a weekly pass using their Android device
Description	User adds a weekly pass to their account

Test Case Field	Details
Test Case ID: Test Case Name	ATC-18: Purchase a Monthly Pass-Android
Purpose	To validate that a user can successfully purchase a monthly pass using their Android device
Description	User adds a monthly pass to their account

Test Case Field	Details
Test Case ID: Test Case Name	ATC-19: View Schedule-Android
Purpose	To validate that a user can successfully purchase a yearly pass using their Android device
Description	User adds a yearly pass to their account

Test Case Field	Details
Test Case ID: Test Case Name	ATC-21: Remote Wipe-Web
Purpose	To validate that a user can successfully wipe their Android device remotely to secure all Transit Droid account data
Description	User goes to the website and performs a remote wipe on a selected device in their account

Test Case Field	Details
Test Case ID: Test Case Name	ATC-22: Accessing the STM network-Android
Purpose	To validate that a user can use their Android device to enter the metro
Description	User scans the Android device on an STM reader terminal and depending on their available fares added to the account, will decrement the fare type from the account.

4.2 Outline of Other Candidates for Potential Inclusion

[To be included in next milestone deliverable]

4.3 Outline of Test Exclusions

For the purposes of testing, there will be certain items that will be excluded for testing. The following high level test items will be excluded from testing:

1. The presentation layer code will NOT be unit tested for the reason that it is too tedious to create a white box unit test for such modules. Instead, this layer of the architecture will be rigorously system tested, via GUI testing
2. The low level, technical services layer will not be explicitly unit tested for the very reason that this layer contains many static methods that if tested, may alter the state of the database, which may overwrite or corrupt important pre-existent fields. We will instead be able to assess if the technical services layer and its modules are working correctly by system testing, in which we can use a black box approach with the user interface to verify that certain reads and writes ultimately completed by the technical services layer are consistent, correct and valid.
3. The Web Services will not be tested for the reason that the testing would take the team out of the time scope of the project. Web services testing will be deferred to a later milestone.

5 TEST APPROACH

The following section will describe the test approach that the testing team will undertake to test the system. The test approach every testing level take, as well as the techniques, tools required and success criteria for each level tested.

5.1 TESTING TECHNIQUES AND TYPES

5.1.1 UNIT TESTING

The unit testing phase is the first phase of testing that the Transit Droid team will target for testing. The approach taken will be that of a white box technique, targeting statement, branch and basis path coverage. However, the true focus will be on basis path testing, as this is the strongest type of testing, compared to the other two types. This takes into account the testing of every linearly independent path in a unit tested module.

Technique Objective:	To attain a basis path coverage of at least 80% for the entire project.
Technique:	1) Use CoView to view independent basis paths required for testing for a given unit test 2) Create the manual unit test using JUnit. Repeat 1 and 2 until all units are tested. 3) Create a JUnit test suite that includes all of the tested units. This will now serve as an automation tool for the unit testing. New unit tests will be added to the test suite so that they can all be run at the same time 4) View statement and branch coverage using eCobertura 5) View basis path coverage using CoView 6) If a test fails because of a bug, report the bug using Jira 7) Document test results in Test Report document
Required Tools:	• JUnit 4 • Eclipse IDE • Coview Eclipse plugin • eCobertura Eclipse plugin

Success Criteria:	The stopping criteria will be when path coverage reaches 80%. The testing team has decided this as a suitable threshold level. However, testing should continue until the discovery of bugs is minimal and the system is thus deemed reliable.
Special Considerations:	The same general procedure will be followed when testing either web application or Android application, as they both support the JUnit testing framework.

5.1.2 Graphical User Interface Testing

The user interface testing level of the testing effort will take a black box function testing technique that will provide the team with a way to test the graphical user interface. The testing effort will fortunately be aided with the automation provided by the Selenium IDE plugin for Mozilla Firefox.

Technique Objective:	To attain a 100% test coverage for all user interface test cases.
Technique:	1) Start recording with Selenium 2) Take a user interface test case from the system test case backlog and follow the execution steps in order to perform the test 3) Document the results of the test in the Test Report document 4) Perform steps 1-3 for every new system test that has not been tested 5) With all system tests recorded with Selenium, this can provide us with automation of our system tests, in which we can run all the system tests together 6) If a bug occurs at any time during testing, log an issue in Jira for resolution
Required Tools:	• Selenium IDE • Mozilla Firefox 19
Success Criteria:	When all tests pass. If there is a failed test, log a ticket to Jira for resolution
Special Considerations:	The Selenium plugin is only available for the Firefox browser

5.1.3 LOAD TESTING <DEFERRED UNTIL NEXT MILESTONE DELIVERABLE>

[Load testing is a performance test that subjects the target-of-test to varying workloads to measure and evaluate the performance behaviors and abilities of the target-of-test to continue to function properly under these different workloads. The goal of load testing is to determine and ensure that the system functions properly beyond the expected maximum workload. Additionally, load testing evaluates the performance characteristics, such as response times, transaction rates, and other time-sensitive issues).]

[Note: Transactions in the following table refer to "logical business transactions". These transactions are defined as specific functions that an end user of the system is expected to perform using the application, such as add or modify a given contract.]

Technique Objective:	[Exercise designated transactions under varying workload conditions to observe and log target behavior and system performance data.]
Technique:	• [Use Transaction Test Scripts developed for Function as a basis, but remember to remove unnecessary interactions and delays. • Modify data files to increase the number of transactions or the tests to increase the number of times each transaction occurs. • Workloads should include (for example, Daily, Weekly, Monthly and so forth) Peak loads. • Workloads should represent both Average as well as Peak loads. • Workloads should represent both Instantaneous and Sustained Peaks. • The Workloads should be executed under different Test Environment Configurations.]
Oracles:	[Outline one or more strategies that can be used by the technique to accurately observe the outcomes of the test. The oracle combines elements of both the method by which the observation can be made and the characteristics of specific outcome that indicate probable success or failure. Ideally, oracles will be self-verifying, allowing automated tests to make an initial assessment of test pass or failure, however, be careful to mitigate the risks inherent in automated results determination.]
Required Tools:	[The technique requires the following tools: Test Script Automation Tool Transaction Load Scheduling and control tool installation-monitoring tools (registry, hard disk, CPU, memory, and so on) resource-constraining tools (for example, Canned Heat) Data-generation tools]
Success Criteria:	[The technique supports the testing of Workload Emulation, which is the successful emulation of the workload without any failures due to test implementation problems.]

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.]

5.1.4 STRESS TESTING <DEFERRED UNTIL NEXT MILESTONE DELIVERABLE>

[Stress testing is a type of performance test implemented and executed to understand how a system fails due to conditions at the boundary, or outside of, the expected tolerances. This typically involves low resources or competition for resources. Low resource conditions reveal how the target-of-test fails that is not apparent under normal conditions. Other defects might result from competition for shared resources, like database locks or network bandwidth, although some of these tests are usually addressed under functional and load testing.]

[Note: References to transactions in the following table refer to logical business transactions.]

Technique Objective:	[Exercise the target-of-test functions under the following stress conditions to observe and log target behavior that identifies and documents the conditions under which the system fails to continue functioning properly • little or no memory available on the server (RAM and persistent storage space) • maximum actual or physically capable number of clients connected or simulated • multiple users performing the same transactions against the same data or accounts • "overload" transaction volume or mix (see Performance Profiling above)]
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 persistent storage space on the 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.
Oracles:	[Outline one or more strategies that can be used by the technique to accurately observe the outcomes of the test. The oracle combines elements of both the method by which the observation can be made and the characteristics of specific outcome that indicate probable success or failure. Ideally, oracles will be self-verifying, allowing automated tests to make an initial assessment of test pass or failure, however, be careful to mitigate the risks inherent in automated results determination.]

Required Tools:	[The technique requires the following tools: Test Script Automation Tool Transaction Load Scheduling and control tool installation-monitoring tools (registry, hard disk, CPU, memory, and so on) resource-constraining tools (for example, Canned Heat) Data-generation tools]
Success Criteria:	The technique supports the testing of Stress Emulation. The system can be emulated successfully in one or more conditions defined as stress conditions and an observation of the resulting system state during and after the condition has been emulated can be captured.]
Special Considerations:	• [Stressing the network may require network tools to load the network with messages or packets. • The persistent storage used for the system should temporarily be reduced to restrict the available space for the database to grow. • Synchronize the simultaneous clients accessing of the same records or data accounts.]

5.1.5 User Acceptance Testing

User acceptance testing is the final level of the testing process. This type of testing with validate our system with respect to the stakeholder requirements. If user acceptance tests pass, this can ensure that the requirements are purely met, since it is the actual stakeholders that will be doing this type of testing. This will be a manual test, as automation would be difficult when involving stakeholders.

Technique Objective:	To attain a 100% coverage for the all acceptance tests by the user.
Technique:	1) Read a test case to perform to a stakeholder 2) Allow stakeholder to perform given test case. This will correspond to meeting the requirements of the system 3) If stakeholder approves, document this in the Test Report document 4) If there is a test case that the stakeholder does not approve, note it down, log an issue on Jira as an “acceptance level” bug for resolution 5) Repeat this process for each end of milestone, as this is a way for us to see if the stakeholder accepts the iterative build of the system.
Required Tools:	• None

Success Criteria:	If the user accepts the software after doing the tests, then testing is successful. If more tests are added in future milestones, acceptance testing will be repeated with the new functionality.
Special Considerations:	None

6 ENTRY AND EXIT CRITERIA <DEFERRED UNTIL NEXT MILESTONE DELIVERABLE>

7 DELIVERABLES

This section will describe the various artifacts that will be created by the test effort that are useful deliverables to the various stakeholders of the test effort.

7.1 TEST EVALUATION SUMMARIES

[TO BE DEFERRED UNTIL NEXT MILESTONE DELIVERABLE]

7.2 REPORTING ON TEST COVERAGE

A Test Report will be included as an artifact of the testing effort, whose goal is to demonstrate the results and coverage. The Test Report will contain all results of all different levels of testing along with the dates of test execution.

7.3 PERCEIVED QUALITY REPORTS

A Software Quality report will also be delivered that will display the several metrics that we use to measure the quality of the code base. Metrics used range from Weighted Methods Per Class to Lack of Cohesion Methods.

7.4 INCIDENT LOGS AND CHANGE REQUESTS

7.5 TRACEABILITY MATRIX

A Traceability Matrix will be provided on the Transit Droid ReqWiki page (TransitWiki). It will link the following elements together in order to allow easy tracking. This will ensure that all requirements have been met by the end of the project, and that each respective requirement is fully tested.

1. **Features**
2. **Use Cases**
3. **Test Cases**

8 TESTING WORKFLOW <DEFERRED UNTIL NEXT MILESTONE DELIVERABLE>

9 ENVIRONMENTAL NEEDS <DEFERRED UNTIL NEXT MILESTONE DELIVERABLE>

10 RESPONSIBILITIES, STAFFING, AND TRAINING NEEDS

This section presents the required resources to address the test effort outlined in the Test Plan—the main responsibilities, and the knowledge or skill sets required of those resources.

10.1 PEOPLE AND ROLES

This table shows the staffing assumptions for the test effort.

Human Resources		
Role	Minimum Resources Recommended (number of full-time roles allocated)	Specific Responsibilities or Comments
Test Manager	1 full-time role	Provides management oversight. Responsibilities include: • planning and logistics • agree mission • identify motivators • acquire appropriate resources • present management reporting • advocate the interests of test • evaluate effectiveness of test effort
Test Designer	1 full-time role	Defines the technical approach to the implementation of the test effort. Responsibilities include: • define test approach • define test automation architecture • verify test techniques • define testability elements • structure test implementation

Human Resources		
Role	Minimum Resources Recommended (number of full-time roles allocated)	Specific Responsibilities or Comments
Tester	2 full-time roles	Implements and executes the tests. Responsibilities include: • implement tests and test suites • execute test suites • log results • analyze and recover from test failures • document incidents
Implementer	1 full-time role	Implements and unit tests the test classes and test packages. Responsibilities include: • creates the test components required to support testability requirements as defined by the designer

11 Risks, Dependencies, Assumptions, and Constraints

This section will be deferred until the next milestone.

11.1 Management Process and Procedures –[Deferred until next milestone]

This section represents an outline of what processes and procedures are to be used when issues arise with the Test Plan and its enactment.

11.2 Measuring and Assessing the Extent of Testing

[Outline the measurement and assessment process to be used to track the extent of testing.]

11.3 Assessing the Deliverables of this Test Plan

[Outline the assessment process for reviewing and accepting the deliverables of this Test Plan]

11.4 Problem Reporting, Escalation, and Issue Resolution

[Define how process problems will be reported and escalated, and the process to be followed to achieve resolution.]

11.5 Managing Test Cycles

[Outline the management control process for a test cycle.]

11.6 Traceability Strategies

[Consider appropriate traceability strategies for:

Coverage of Testing against Specifications — enables measurement the extent of testing

Motivations for Testing — enables assessment of relevance of tests to help determine whether to maintain or retire tests

Software Design Elements — enables tracking of subsequent design changes that would necessitate rerunning tests or retiring them

Resulting Change Requests — enables the tests that discovered the need for the change to be identified and re-run to verify the change request has been completed successfully]

11.7 Approval and Signoff

[Outline the approval process and list the job titles (and names of current incumbents) that initially must approve the plan, and sign off on the plans satisfactory execution.]