CSC 191

Fall 2013

Team Sierra

Alex Chernyak

Joubin Jabbari

Kyle Matz

Mike McParland

Scott Livingston

Serge Lysak

System Test Specification Document

1. INTRODUCTION

This is the System Test Specification (STS) document for the Salon Scheduler project sponsored by Dragonfly Salon. This project is being undertaken by the Team Development development team. The team is comprised of undergraduate students majoring in Computer Science at California State University, Sacramento. The team members are enrolled in a two-semester senior project course required of all undergraduate majors. Successful delivery of the desired software product will fulfill the senior project requirement for the student team members.

PROJECT SPONSOR
Contact Person's Name: Lisa Sigurdson
Title: Owner and Hair Stylist
Organization Name: Dragonfly Salon and Boutique
Contact information: Alayna.Sigurdson@gmail.com
DEVELOPMENT TEAM:
"Team Sierra":

- Alex Chernyak
- Joubin Jabbari
- Kyle Matz
- Mike McParland
- Scott Livingston
- Serge Lysak

Team Contact Info: TeamSierra@googlegroups.com

1.1 Purpose

The purpose of the STS is 1) to describe the plan for testing the software, and 2) to specify the test cases and test procedures necessary to demonstrate that the software satisfies the requirements as specified in the project's System Requirements Specification document.

1.2 Scope

The plan contains a list and brief description of the use cases to be tested and the software components associated with each test case. The plan also provides a schedule for the testing and the assignment of team members to their respective testing tasks. The process for documenting resolving software errors and/or anomalies that are found during the testing is also specified. The test specification includes a list of the features to be tested for each of the use cases, the description each test case needed to fully test the use case, and the test procedures, or steps, necessary to execute each of the test cases.

1.3 Definitions, Acronyms, and Abbreviations

This subsection serves as a glossary for the document. All technical terms used as well as all acronyms and abbreviations used are arranged in alphabetical order. The purpose of this subsection is to provide the reader with quick access to the technical references used

throughout the document.
All references to the software technology used should be included.

NOTE. You will need to indicate with the appropriate symbol whether the products you reference are “trademarked” (using ® or ™) or “copyrighted” (using ©).

1.3.1. Definitions.

AJAX- Asynchronous JavaScript and XML. A group of interrelated web development techniques used on the client side to create asynchronous web applications.
AngularJS- Open source Javascript Framework the helps browser based application implement model-view-controller.
CSS- Cascade Style Sheets. A style sheet language that is used to create the look and formatting of documents created with a markup language such as HTML.
Fongo- Fake Mongo. In memory Mongo Database simulator used mainly for unit testing.
JAVA- Object oriented programming language to used create the backend of the application.
JUNIT- A Units Test Framework with Java that helps test the web application.
JSON- JavaScript Object Notation. A lightweight data-interchange format that is used to pass data between the server and client side.
Mockito- Testing framework for java that helps testing by allowing the creation of Mock Objects or Test Doubles.
HTML- Hypertext Markup Language. The primary markup language for creating content that can be displayed in a web browser.
SPRING MVC. Spring Model View Controller. Open source framework to help create the Java application.

1.3.2. Acronyms

none

1.3.3. Abbreviations

STS. System Test Specification
STR. System Test Report

1.4 References.

Dr. Buckleys STS template proved at CSU, Sacramento Spring 2013.

1.5 Overview of Contents of Document

Section 2: Test Plan Description
This section contains a summary of the Use Cases and the plan that will be used carry out the system test phase of the software development process. It will contain a description of the Use Cases that will be tested, which team members will be assign to test a particular Use Case, the schedule for testing, and the risk management plan.

Section 3: Test Design Specification
This section will contain the details of the testing approach, the features that will and will not be tested, the environmental needs, the suspension and resumption criteria, and the risk and contingencies.

Section 4: Test Specification
This section contains the test procedures, test procedures conventions, the test data needed for each use case, and a subsection for each of the use cases to be tested. Each of those subsections will include all the test cases for that use case along with the procedure of the test and the input and output of the test.

Section 5: Requirements Traceability
This section provides for a cross referencing of each test case to its software requirement specification (or specifications) and also to its design component (or components). The appropriate section and its title in each document are provided.

Section 6: Approvals
This section contains the list of the key signatories necessary to sign-off on the STS, thereby agreeing to the scope and content of the test plan and test cases specified within the document. Approval constitutes a guarantee that the development team has produced a test specification sufficient for validating the software to be delivered to the sponsor.

2. TEST PLAN DESCRIPTION

This section contains a summary of the Use Cases and the plan that will be used carry out the system test phase of the software development process. It will contain a description of the Use Cases that will be tested, which team members will be assign to test a particular Use Case, the schedule for testing, and the risk management plan.

The intent of system test plan is as follows:

- To specify the activities required to prepare for and conduct the system test.
- To clearly indicate the tasks that must be performed, the team members assigned to each of the tasks, and the schedule to be followed in performing the tasks.
- To identify the sources of information used to prepare the plan
- To identify the test tools and environmental needs for conducting the tests.
- Guide to Preparing the System Test Specification Document 5

2.1 Product Summary.

This product will allow customers to access a web application that they can requested appointments with a selected stylists. The Stylists can also access the application and create, approve, and deny appointments with a given client. The Stylists will also be able to set their availability and services offered. Both the client and the stylist will be able to edit a personal profile with basic information relating to hair and nails.

The following table contains a listing of the use cases, the system's features associated with each use case along with its files and database tables.

Use Case, Feature, and Component Table
(C denotes Client Use Cases, S denotes Stylists Use Cases and A denote admin Use Cases)

USE CASE	FEATURES	COMPONENTS
View Profile C	● View profile	● AuthenticationController.java ● UserController.java ● User.java ● UserRepository.java ● Tables: users
Update Profile C	● Edit profile	● AuthenticationController.java ● UserController.java ● User.java ● UserRepository.java ● Tables: users
View Stylist C	● Choose who to schedule an appointment with	● AuthenticationController.java ● UserController.java ● User.java ● UserRepository.java

		• Tables: users
Make Appointment C	• Schedule an appointment • Choose who to schedule an appointment with	• AppointmentController.java • AuthenticationController.java • UserController.java • Appointment.java • User.java • AppointmentRepository.java • UserRepository.java • Tables: users,appointments, stylistSchedule
Cancel Appointment C	• Cancel Appointments	• AppointmentController.java • AuthenticationController.java • UserController.java • Appointment.java • User.java • AppointmentRepository.java • UserRepository.java • Tables: users,appointments
View Past/Present Appointments C	• View previous appointments and appointment history	• AppointmentController.java • AuthenticationController.java • UserController.java • Appointment.java • User.java • AppointmentRepository.java • UserRepository.java • Tables: users,appointments
Login C	• login as a client	• AuthenticationController.java • UserController.java • Tables:appointments • login.html
Create Profile S	• Create Profile	• AuthenticationController.java • UserController.java • User.java • UserRepository.java • Tables: users
View Profile S	• View Profile	• AuthenticationController.java • UserController.java • User.java • UserRepository.java • Tables: users
View Client List S	• View all client	• AuthenticationController.java • UserController.java • User.java • UserRepository.java • Tables: users

View Client Profile S	• View client's profile	• AuthenticationController.java • UserController.java • User.java • UserRepository.java • Tables: users
Manage Appointment S	• Accept appointment • Deny appointment • Cancel appointment • Create appointment	• AppointmentController.java • AuthenticationController.java • UserController.java • Appointment.java • User.java • AppointmentRepository.java • UserRepository.java • Tables: users,appointments, stylistSchedule
Set Availability S	• Provide/Change availability	• AuthenticationController.java • UserController.java • User.java • UserRepository.java • Tables: users,appointments, stylistSchedule
Manage Services S	• Manage offered services	• AuthenticationController.java • UserController.java • User.java • UserRepository.java • Tables: users
Update Profile S	• Edit profile	• AuthenticationController.java • UserController.java • User.java • UserRepository.java • Tables: users
Create Appointment S	• Create appointment	• AppointmentController.java • AuthenticationController.java • UserController.java • Appointment.java • User.java • AppointmentRepository.java • UserRepository.java • Tables: users,appointments, stylistSchedule
Manage Pending Appointment S	• Accept appointment • Deny appointment	• AppointmentController.java • AuthenticationController.java • UserController.java • Appointment.java • User.java • AppointmentRepository.java • UserRepository.java • Tables: appointments

Cancel Appointment S	● Cancel appointment	● AppointmentController.java ● AuthenticationController.java ● UserController.java ● Appointment.java ● User.java ● AppointmentRepository.java ● UserRepository.java ● Tables: appointments
Login S/A	● Login as a stylist or admin	● AppointmentController.java ● AuthenticationController.java ● UserController.java ● Appointment.java ● AppointmentRepository.java ● staff-landing.html ● Tables:appointments
Create Stylist or Admin Account A	● Create users	● AuthenticationController.java ● UserController.java ● User.java ● UserRepository.java ● Tables: users
Reactivate/deactivate Accounts A	● Reactivate user ● Deactivate user	● AuthenticationController.java ● UserController.java ● User.java ● UserRepository.java ● Tables: users

2.2 Responsibilities.

The following table contains a listing each Use Case, the team member assigned to testing the Use Case, and whatever set-up is necessary for testing the Use Case.

Use Case Testing Table

USE CASE	TEAM MEMBER	SET-UP
View Profile C	● Serge	● Serge
Update Profile C	● Serge	● Serge
View Stylist C	● Serge	● Serge
Make Appointment C	● Serge	● Serge
Cancel Appointment C	● Serge	● Serge
View Past/Present Appointments C	● Serge	● Serge

Login C	● Serge	● Serge
Create Profile S	● Mike	● Mike
View Profile S	● Mike	● Mike
View Client List S	● Mike	● Mike
View Client Profile S	● Mike	● Mike
Manage Appointment S	● n/a	● n/a
Set Availability S	● Mike	● Mike
Manage Services S	● Mike	● Mike
Update Profile S	● Mike	● Mike
Create Appointment S	● Kyle	● Kyle
Manage Pending Appointment S	● Kyle	● Kyle
Cancel Appointment S	● Kyle	● Kyle
Login S/A	● Kyle	● Kyle
Create Stylist or Admin Account A	● Kyle	● Kyle
Reactivate/deactivate Accounts A	● Kyle	● Kyle

NOTE. Projects are expected to have at least as many FEATURES as there are team members. In assigning the development work, each team member is assigned to implement – either individually or with another team member. The testing assignment should be made in such a way that team members DO NOT test the use cases that they implemented.

2.3 Schedule.

This subsection contains the testing schedule for each FEATURE, specifying the date on which testing should begin and the date that testing should be expected to be completed. The schedule should include time for resolving all problems reported during the testing. In addition, if the software has been developed and tested in a development environment, a plan and
Guide to Preparing the System Test Specification Document 6
schedule should be included for testing in the sponsor's operational environment. At the end of the testing cycle the software should be ready for delivery.

Use Case Testing Schedule Table

USE CASE	START DATE	END DATE	RESOLVED DATE
View Profile C	11/27/2013	12/6/2013	12/13/2013
Update Profile C	11/27/2013	12/6/2013	12/13/2013
View Stylist C	11/27/2013	12/6/2013	12/13/2013
Make Appointment C	11/27/2013	12/6/2013	12/13/2013
Cancel Appointment C	11/27/2013	12/6/2013	12/13/2013
View Past/Present Appointments C	11/27/2013	12/6/2013	12/13/2013
Login C	11/27/2013	12/6/2013	12/13/2013
Create Profile S	11/27/2013	12/6/2013	12/13/2013
View Profile S	11/27/2013	12/6/2013	12/13/2013
View Client List S	11/27/2013	12/6/2013	12/13/2013
View Client Profile S	11/27/2013	12/6/2013	12/13/2013
Manage Appointment S	n/a	n/a	n/a
Set Availability S	11/27/2013	12/6/2013	12/13/2013
Manage Services S	11/27/2013	12/6/2013	12/13/2013
Update Profile S	11/27/2013	12/6/2013	12/13/2013
Create Appointment S	11/27/2013	12/6/2013	12/13/2013
Manage Pending Appointment S	11/27/2013	12/6/2013	12/13/2013
Cancel Appointment S	11/27/2013	12/6/2013	12/13/2013
Login S/A	11/27/2013	12/6/2013	12/13/2013
Create Stylist or Admin Account A	11/27/2013	12/6/2013	12/13/2013
Reactivate/deactivate Accounts A	11/27/2013	12/6/2013	12/13/2013

3. TEST DESIGN SPECIFICATION

This section will contain the details of the testing approach, the features that will and will not be tested, the environmental needs, the suspension and resumption criteria, and the risk and contingencies.

3.1 Testing Approach.

Our testing approach is to ensure that all the requirements and specification laid out in the System Requirements Specification and Software Design Document for our system have been fulfilled. These requirements and specification required are as follows:

Client can

- Edit profile (Personal Information)
- Login at a later date
- View Stylists
- Schedule an appointment
- Choose who to schedule an appointment with
- View previous appointments and appointment history
- Cancel Appointments

Stylist can...

- Provide/Change availability
- Accept/Deny/Cancel appointments
- Tender an appointment
- Schedule an appointment on behalf of a client
- View Client Profiles
- View previous appointments and appointment history of a given client (search)
- Cancel Appointments

Admin can...

- Including all Features above
- Create and Delete users
- Set Operational Hours

In order to ensure that our system fulfils these requirements we will using a battery of testing. The backend java files will be unit tested and the database will be checked to ensure it is correctly creating, retrieving, updating, and deleting information. Once we have ensured the backend and database are working correctly together we will be integrating it them with the front end (html, css and javascript).

After the systems front and back ends have been integrated they will be tested to ensure that all Use Cases can be executed in their entirety and without errors or exceptions. Each Use Case will be tested by entering in test data to a web page and observing the system outputs. The outputs would then be compared to outputs expected according the documentation. The system will be tested using the procedures laid out in section four.

The systems security will also be tested to ensure that no unauthorized login is approved and system data cannot be retrieve without proper credentials.

3.2 Feature or Combination of Features Not To Be Tested.

The following features of the system will not be tested. Most of these features were not specifically included in the Software Requirements Specifications or Software Design Document but are rather implied.

- Host Server's ability to deal with load and hardware issues. We will not be testing the systems ability to handle a large number of user or any hardware issues that may arise during the lifetime of the application.
- The system is intended to be able to email users automatically to notify them of certain events, however as this is a minor feature that the sponsor did not deem necessary it is going to be implemented after everything else is done and thus will be excluded from the testing specified in this document.

3.3 Environmental Needs.

The following are the requirements for testing the system:

- A host server capable of hosting Tomcat, MongoDB, Java, and Apache and a network connection.
- A client system with a web browser and connected to the same network as the host server.
- A mobile device capable of web browsing on the same network as the host server.

The following are the requirements for setting up the operation system.

- A host server capable of hosting Tomcat, MongoDB, Java, and Apache and a network connection capable supporting multiple users at once.

3.4 Suspension / Resumption Criteria.

Suspension: If during testing, once of the tests cannot be completely successfully completed the test must be suspended and the error(s) must be logged. Furthermore any other other components that are affected by the failed test must be retested also.

Resumption: Once a test has been suspended it can only be resumed after the logged errors have been investigated, identified, and fixed. At that point the test must be rerun in order to ensure it successfully completes in its entirety without any errors or exceptions. Any other components that were affected by the failed tests must also be retested in order to ensure the entire system is working correctly.

3.5 Risks and contingencies.

4. TEST SPECIFICATION

This section contains subsections for each of the FEATURES to be tested. Each subsection specifies the USE CASES to be tested, the procedures necessary to run the test cases, the items being tested.

4.1 Test Procedures.

The testing procedures (steps) will be listed for each test case. The operator will follow each step until complete. If no errors have occurred, then the test operator will need to validate that the outcome is correct by examining the database. If errors occur, a Software Problem Report (SPR) will be written (a sample SPR template is included in appendix A). Each SPR will then be analyzed and a determination made as to
Guide to Preparing the System Test Specification Document 8
whether a software change is warranted. If so, the test cases will be re-run after the changes have been implemented and the software appropriately updated.

4.2 Test Procedure Conventions.

Each test case will give specific instructions describing what steps need to be taken to complete the test case. All test cases will start at the home page. Also, provided will be the expected results. This subsection includes the software installation instructions.

4.3 Test Data.

This subsection should specify all the test data that is needed to run the test cases associated with each use case. This specification should include the specific data that must be set into the database along with instructions on to how this is to be done. For example, the data base would need to be appropriately set in order to test for user authentication for a login use case. In addition, permissions would need to appropriately set in order to test for various categories of restricted and unrestricted user access.
If appropriate, a cross reference listing should be provided of those data sets associated with specific use case tests.

4.4 FEATURE

1. List the software use cases to be tested and include a brief description for each feature. Indicate the path of links from the home page to the page (or pages) being tested. If access to the page to be tested requires login, provide the instructions for login.

Note: All the Test Cases will use the following Test Items (files and tables) and thus are excluded from the test case specific test items:

    angular.min.js
    bootstrap-select.min.js
    jquery-1.10.2.min.js
    less.min.js
    modernizr-2.6.2.min.js

ui-bootstrap-0.6.0.min.js
AuthInterceptor.java
UserRepository.java
User.java
controllers.js
gnmenu.js
angular-facebook.js
app.js
classie.js
directives.js
main.js
plugins.js
tables: users

4.4.1
Test Case 1: View Profile (Client)

Example: This test case will test the function required for the client to view a profile.

Steps:

1. Starting on the landing page of a logged in client. Click on the profile button (clients picture on menu bar or edit profile on the menu drop down list).
2. System will return current logged in users profile for viewing.

4.4.1.1 Test Items
- UserController.java
- edit-profile.html
- client-landing.html

4.4.1.2 Input Specifications
Steps:

1. Click on the profile picture on the menu bar or the edit profile button on the menu drop down list.
2. Verify all information displayed in correct.

4.4.1.3 Output Specifications
The system should return a profile page of the currently logged in user if everything is working correctly.

4.4.1.4 Intercase Dependencies
The client must already be logged in in order to view their own profile. The steps and procedure for this are listed in 4.4.7. Without being logged in the user will be unable to view a profile.

4.4.2
Test Case 2: Update Profile (Client)

Example: A logged in client can load their profile page and then select to update the profile information. They can then change most of their profile information.

Steps:

1. User has logged in and have selected to view their profile as described in Test Cases 4.4.7 and 4.4.1.
2. Once on the profile page the client will select to edit the profile information by click the fields they wish to edit.

4.4.2.1 Test Items

- UserController.java
- edit-profile.html
- client-landing.html

4.4.2.2 Input Specifications

Steps:

1. Enter the following information into the editable text boxes:
   Phone: 916 hello save
   Hair Color: Brown
   Hair Length: Medium

2. The save button should be disabled because the phone number is not a valid phone number.
3. Change the following text boxes:
   Phone: 916 555 5555
4. Now click on the save button to save the updated information.
5. Note that the clients names is pulled directly from the OAuth Api and cannot be edited.
6. Ensure that all the information is updated in the database and displayed correctly by clicking on the Salon Scheduler button on the menu bar.
7. Now navigate back to the view/edit profile page by clicking on the profile picture on the menu bar or the edit profile button on the menu drop down list.
8. Check over the information in the text boxes to ensure the data has been saved and updated.

4.4.2.3 Output Specifications

The system should return the profile page with the updated profile information if it working correctly.

4.4.2.4 Intercase Dependencies

The client must be already logged in and have navigated to view their own profile. The steps and procedures for these can be found in Test Cases 4.4.7 and 4.4.1. Without being logged in the user will not be able to navigate to their profile page and without being on the profile page the user will not be able to edit their profile.

4.4.3

Test Case 3: View Stylists (Client)

Example: A logged in client wants to view all the stylists that are available to make appointments with.

Steps:

1. The client is logged in as described in Test Case 4.4.7 and is on their landing page.
2. The client navigates to the view-stylist page by clicking on the view stylist link on the menu drop down.

4.4.3.1 Test Items

- UserController.java
- client-landing.html
- view-stylists.html

4.4.3.2 Input Specifications

Steps:

1. System will navigate to the view stylists page that has a table listing all the stylists available at the salon.
2. Check to ensure that the table lists all the active stylist that are currently in the database with the correct data in each table column.

4.4.3.3 Output Specifications

The system will return a table will all the active stylists in the database if everything is working correctly.

4.4.3.4 Intercase Dependencies

The client must already be logged in as describe in Test Cases 4.4.7 in order to view a list of stylists. This is to protect the stylists information so that only a client or future client can view the page.

4.4.4
Test Case 4: Make Appointment (Client)

Example: A logged on client can select to make an appointment. (request an appointment)

Steps:

1. Client is logged in and is currently at the landing page.
2. The client selects the make an appointment button on the right upper part of the screen or from the menu drop down list.
3. The system will load the calendar page that allows the client to request an appointment.

4.4.4.1 Test Items

- DateRange.java
- Availability.java
- StylistAvailability.java
- StylistAvailabilityRepository.java
- UserController.java

- Appointment.java
- AppointmentType.java
- AppointmentRepository.java
- AppointmentTypeRepository.java
- AppointmentController.java
- AppointmentTypeController.java
- client-landing.html
- calendar.html
- Tables:appointments, appointmentTypes, stylistAvailability

4.4.4.2 Input Specifications

Steps:

1. On the calendar page the client will select month and day to make an appointment.
2. Once a day has been select a module window will appear in which the client enters the following information:
   Stylist: Select any stylist from the drop down list.
   Type: Select any type of appointment from the drop down list.
   Start Time: set the start time to be at the beginning of the stylist's availability.
3. Select the button Make Appointment to submit the entered information.
4. The system will then take the user back the users landing page.
5. The new pending appointment will be listed on the users landing page.

4.4.4.3 Output Specifications

The system will display the landing page of the logged in user with the new requested appointment listed if everything has worked correctly.

4.4.4.4 Intercase Dependencies

The client must already be logged in in order to view their own profile. The steps and procedure for this are listed in 4.4.7. Without being logged in the user will be unable to view the landing page and therefore unable to request to make an appointment.

4.4.5

Test Case 5: Cancel Appointment (Client)

Example: A logged in client wish to cancel either a pending or approved appointment in the future that they have previously made.

Steps:

1. The client is logged in and is viewing the landing page.

4.4.5.1 Test Items

- AppointmentController.java
- AuthenticationController.java
- UserController.java
- Appointment.java

- AppointmentRepository.java
- client-landing.html
- Tables:appointments

4.4.5.2 Input Specifications

Steps:

1. At the landing page the client selects the cancel button for the appointment they wish to cancel whether pending or approved.
2. A confirm window will appear that asks the client to confirm the the cancellation of the appointment.
3. The client will select the cancel button on the confirm dialog.
4. The system will take the client back to the landing.
5. The landing page will still display all the appointments because the client did not select confirm when canceling the appointment.
6. The client will select cancel on the appointment they wish to cancel.
7. The system will show a confirm window.
8. The client will select the confirm button on the cancel appointment confirm window.
9. The system will take the client back the to the landing page.
10. The system should show all the appointments except the one the client cancelled.

4.4.5.3 Output Specifications

The system will show the client's landing page without the cancelled appointment listed if everything has worked correctly.

4.4.5.4 Intercase Dependencies

The client must have already logged into the system and created at least one appointment. These steps and procedures are listed in Test Cases 4.4.7 and 4.4.4. Without being logged in the client will not be able to view the landing page and without having previously created an appointment there will not be any appointment to be cancelled.

4.4.6

Test Case 6: View Past/Present Appointments (Client)

Example: A logged in client wishes to view all their past and present appointments.

Steps:

1. The client is logged in and is viewing their landing page.

4.4.6.1 Test Items

- AppointmentController.java
- AuthenticationController.java
- UserController.java
- Appointment.java
- AppointmentRepository.java
- client-landing.html
- Tables:appointments

4.4.6.2 Input Specifications

Steps:

1. The client must simply scroll through the list of appointments on the landing page that they are already at.

4.4.6.3 Output Specifications

The system will show all the past and present of the currently logged in user if everything is working correctly.

4.4.6.4 Intercase Dependencies

The client must have already logged into the system and created at least one appointment. These steps and procedures are listed in Test Cases 4.4.7 and 4.4.4. Without being logged in the client will not be able to view the landing page and without having previously created an appointment there will not be any appointment to be viewed.

4.4.7

Test Case 7: Login (Client)

Example: A client needs to log-in in order to use the system.

Steps:

1. Click on Facebook button.
2. Enter Facebook credentials.
3. Click Log In.

4.4.7.1 Test Items

- AuthenticationController.java
- UserController.java
- Tables:appointments
- login.html

4.4.7.2 Input Specifications

1. Enter the following into the email field: “fake@email.com”
2. Enter a password.
3. Click Log in.
4. Enter a real facebook account email.
5. Enter “fakep@$$word” for password (assuming this is not a valid password).
6. Click Log in.
7. Enter real password for the facebook account.
8. Click Log in.

4.4.7.3 Output Specifications

Verify that first and second log-in attempts failed.

Verify that the third attempt succeeded.

4.4.7.4 Intercase Dependencies

None.

4.4.8

Test Case 8: Create Profile (Stylist)

Example: This test case tests the creation of a stylist's profile. The profile is created when the user is created by an admin (see 4.4.19).

Steps:

1. Admin is logged in, see 4.4.19.
2. The admin creates a new stylist, see 4.4.20
3. Log in using the stylist's credentials created by the admin in step 2.

4.4.8.1 Test Items

- UsersController.java
- edit-profile.html

4.4.8.2 Input Specifications

Steps:

1. The stylist clicks on their picture in the upper left on the menu bar.

4.4.8.3 Output Specifications

The user should be redirected to the edit-profile.html page upon clicking their icon.

4.4.8.4 Intercase Dependencies

The stylist user must be logged in to the system in order to view their profile (4.4.19). The stylist would also have to have been created by an admin in order to log in (4.4.20).

4.4.9

Test Case 9: View Profile (Stylist)

Example: This test case tests the stylist's ability to view their profile.

Steps:

1. Log in using valid stylist credentials, see 4.4.19.

4.4.9.1 Test Items

- UsersController.java
- edit-profile.html

4.4.9.2 Input Specifications

Steps:

1. The stylist clicks on their picture in the upper left on the menu bar.

4.4.9.3 Output Specifications

The user should be redirected to the edit-profile.html page upon clicking their icon.

4.4.9.4 Intercase Dependencies

The stylist user must be logged in to the system in order to view their profile (4.4.19). The stylist would also have to have been created by an admin in order to log in (4.4.20).

4.4.10

Test Case 10: View Client List (Stylist)

Example: A logged in stylists wants to view a list of all the clients that have ever logged into the system. This will include clients that have not ever scheduled an appointment.

Steps:

1. The stylists has logged in as described in Test Case 4.4.19 and any of the site's pages.
2. The stylists clicks on the menu drop down list item called View Clients.

4.4.10.1 Test Items

- UsersController.java
- staff-landing.html
- view-clients.html

4.4.10.2 Input Specifications

Steps:

1. The system will take the stylist to the view clients page.
2. The view clients page will display a table with all the clients listed in it.
3. Verify that the table contains all the users that are in the database (that have ever logged into the system).

4.4.10.3 Output Specifications

The system will display a table with all the users and appropriate information for each user if everything is working correctly.

4.4.10.4 Intercase Dependencies

The stylist must be logged into their account (4.4.19) in order to view all the clients. This protects the client's information so that only authenticated stylist can view the page.

4.4.11

Test Case 11: View Client Profile (Stylist)

Example: When viewing the list of all the clients a logged in stylists wished to view the client's personal information.

Steps:

1. The stylist has logged in as described in Test Case 4.4.19 and is on any of the staff pages.
2. The stylist will select to view clients from the menu drop down list as described in Test Case 4.4.9.

4.4.11.1 Test Items

- UsersController.java
- staff-landing.html
- view-clients.html
- client-profile.html

4.4.11.2 Input Specifications

Steps:

1. From the view clients page the stylist can select any of the clients listed in the table.
2. Once a client has been selected the system will display that clients profile (personal information) for the stylist.
3. The stylists can then close the window and select a new client to view.
4. The newly selected clients will be displayed.

4.4.11.3 Output Specifications

For each client that the stylist selects the system will display that clients information to the stylist is everything is working correctly.

4.4.11.4 Intercase Dependencies

The stylist must be logged in already (Test Case 4.4.19) and must also have selected to view client page (Test Case 4.4.10). This protects the clients private information from being accessed by someone without the proper authentication. The stylists must also already be viewing the list of clients because each client name will be a link that displays a window with all the client's information.

4.4.12

Test Case 12: Manage Appointments (Stylist)

Example: This was originally intended to test a stylists ability to create appointments, cancel future appointments, and accept or decline pending appointments. All of these functions have been separated out into other test cases.

Steps:

1. none

4.4.12.1 Test Items

- Not Applicable.

4.4.12.2 Input Specifications

Steps:

1. none

4.4.12.3 Output Specifications

Not applicable.

4.4.12.4 Intercase Dependencies

The pending appointment management has been delegated out to 4.4.17, the create a new appointment is 4.4.16, and cancel appointment is 4.4.18.

4.4.13

Test Case 13: Set Availability (Stylist)

Example: A logged in stylists wishes to set for the first time or update their availability. This availability controls when appointments can and cannot be scheduled by clients.

Steps:

1. The stylist must be logged in as described in Test Case 4.4.19 and may be at any of the staff pages.
2. From any of the pages the stylist must select change availability from the menu drop down list.

4.4.13.1 Test Items

- UsersController.java
- staff-landing.html
- change-Availability.html
- DateRange.java
- Availability.java
- StylistAvailability.java
- StylistAvailabilityRepository.java
- Tables: stylistAvailability

4.4.13.2 Input Specifications

Steps:

1. Once on the change availability page the stylists can enter the availability for a week. This week will be used as the stylist availability until they choose to change it.
2. Enter the following start and end time for the lists days:
   Monday- Start: 8:00AM End: 5:00PM
   Tuesday- Start: 8:00PM End: 5:00PM
   Wednesday- Start: 9:00AM End: 3:00PM
   Thursday- Start: 8:00AM End: 5:00PM
   Friday- Start: 8:00AM End: 5:00PM
   Saturday- Start: 8:00AM End: 8:00AM
   Sunday- Start: 8:00AM End: 8:00AM
3. Click on the save availability button below the entered times.
4. The page should reload and display a warning message that the times set for Tuesday are not valid. This is because the start time is after the end time.
5. Change the fields for Tuesday to the following:
   Tuesday- Start: 8:00AM End: 5:00PM
6. Click the save availability again.
7. The stylist will be taken back to staff-landing page and the new availability will be updated in the database.
8. To ensure that the time has properly been saved view the calendar and ensure the currently logged in stylists is available on the times listed above.

4.4.13.3 Output Specifications

If the system is working correctly the stylists availability will be updated in the database which will also cause the calendar to display to new availability to any clients viewing the page.

4.4.13.4 Intercase Dependencies

The stylist must already be logged in (Test Case 4.4.19) in order to change the availability. This is so that the system knows which stylist availability to change as well as protects unauthorized access to the change availability page.

4.4.14

Test Case: Manage Services (Stylist)

Example: A logged in stylists want to change the services that they offer to the clients.

Steps:

1. The stylist must be logged in as described in Test Case 4.4.19 and may be at any of the staff pages.
2. The stylists selects manage services from the menu drop down list.

4.4.14.1 Test Items

- AppointmentType.java
- AppointmentTypeRepository.java
- AppointmentTypeController.java
- Tables:appointmentTypes

4.4.14.2 Input Specifications

Steps:

1. Once on the manage services page the stylist can either remove or add a service.
2. Select any of services already available for the stylist and delete that service.
3. The page will then reload and no longer display the selected service.
4. To add a new service in select the Add button.
5. A window will appear asking for the information for a new service.
6. Enter the following information:

   Name: Coloring
   Duration in Minutes: 120
   Base Price: 200

4.4.14.3 Output Specifications

If everything is working correctly the logged in stylist's services will be updated in the database. This will then change the available services that the clients can see on the stylists page.

4.4.14.4 Intercase Dependencies

The stylist must already be logged in (Test Case 4.4.19) in order to change the services they offer. This is so they system knows which stylist is changing the services they offer and provides security from and non-authorized user from changing a stylist availability.

4.4.15

Test Case 15: Update Profile (Stylist)

Example: This test case is to test a stylist's ability to change their profile information.

Steps:

1. Log in as a stylist with valid credentials, see 4.4.19.
2. Click on avatar picture in the upper right hand side of the menu bar to be taken to the profile page.

4.4.15.1 Test Items

- UserController.java
- staff-landing.html
- edit-profile.html

4.4.15.2 Input Specifications

Steps:

1. Enter in invalid information for each field.
2. verify that the update button is disabled while invalid information is entered.
3. Enter information that is desired to be updated.
4. Click on the update button.

4.4.15.3 Output Specifications

The page should indicate that the information has been updated. Revisit the stylist's profile page to verify the information is changed.

4.4.15.4 Intercase Dependencies

The stylist will need to be logged into the system in order to change their profile information (4.4.19).

4.4.16

Test Case 16: Create Appointment (Stylist)

Example: This test case is designed to test a stylist's ability to create an appointment on behalf of a client.

Steps:

1. A stylist logs in with valid credentials, see 4.4.20.
2. From the stylist's landing page, click on the calendar to be taken to the calendar.html page.

4.4.16.1 Test Items

- AppointmentController.java
- AuthenticationController.java
- UserController.java
- Appointment.java
- AppointmentRepository.java
- calendar.html
- Tables:appointments

4.4.16.2 Input Specifications

Steps:

1. Click on the desired day for the appointment.
2. Fill out the form that comes up to create a new appointment.

4.4.16.3 Output Specifications

The stylist will be redirected to the landing page where the newly created appointment should show up on the appointment list.

4.4.16.4 Intercase Dependencies

The stylist will need to be logged in in order to create a new appointment (4.4.19).

4.4.17

Test Case 17: Manage Pending Appointments (Stylist)

Example: A pending appointment can either be accepted or declined.

Steps:

1. Log in with valid stylist credentials, see 4.4.19.
2. create 2 appointments for the stylist, see 4.4.16.

4.4.17.1 Test Items

- DateRange.java
- Availability.java
- StylistAvailability.java
- StylistAvailabilityRepository.java
- Appointment.java
- AppointmentType.java
- AppointmentRepository.java
- AppointmentTypeRepository.java
- AppointmentController.java
- AppointmentTypeController.java
- staff-landing.html
- Tables:appointments, appointmentTypes, stylistAvailabilityAuthenticationController.java

4.4.17.2 Input Specifications

1. Enter message: “See you soon Sandy.”
2. Click the Accept button.
3. Validate the appointment block turned green, the buttons and text field are disabled.
4. Click on Deny on another pending appointment.
5. Validate the appointment block turned red, the buttons and text field are disabled.

4.4.17.3 Output Specifications

The system should now have the previously pending appointment blocks be two different blocks. One accepted with a message, the other a denied appointment block.

4.4.17.4 Intercase Dependencies

The client must be already logged in and have navigated to view their appointments. The steps and procedures for these can be found in Test Cases 4.4.19. Without being logged in the user will not be able to navigate to their landing page and without being on the landing page the user will not be able to manage their appointments.

4.4.18

Test Case 18: Cancel Appointment (Stylist)

Example: This test case will test a stylists' ability to cancel an appointment.

Steps:

2. Log in with valid stylist credentials, see 4.4.19.
3. create an appointments for the stylist, see 4.4.16.

4.4.18.1 Test Items

- DateRange.java
- Availability.java
- StylistAvailability.java
- StylistAvailabilityRepository.java
- Appointment.java
- AppointmentType.java
- AppointmentRepository.java
- AppointmentTypeRepository.java
- AppointmentController.java
- AppointmentTypeController.java
- staff-landing.html
- Tables:appointments, appointmentTypes, stylistAvailability

4.4.18.2 Input Specifications

Steps:

2. After creating the appointment the stylist should be on the landing page with the new appointments displayed as pending.
3. Accept one of the appointments.
4. The appointment should change status to accepted and turn green.
5. Cancel the appointment via the cancel button on the previously accepted appointment.

4.4.18.3 Output Specifications

The appointment should turn red and update it's status on the landing page to show that it is cancelled.

4.4.18.4 Intercase Dependencies

The stylist must be logged in in order to reach their landing page which has the appointments on it (4.4.19).

4.4.19

Test Case 19: Login (Stylist/Administrator)

Example: This test case will test the stylist's and admin's ability to log in to the system.

Steps:

1. On the entry point for the system.

4.4.19.1 Test Items

- AppointmentController.java
- AuthenticationController.java
- UserController.java
- Appointment.java
- AppointmentRepository.java
- staff-landing.html
- Tables:appointments

4.4.19.2 Input Specifications

Steps:

1. On the login page, which is the entry point to the system, select staff.
2. Enter valid email, with invalid password.
3. Click sign in.
4. The page will indicate that the credentials are invalid.
5. Enter valid credentials. (This will require previously making a stylist or admin user, see 4.4.19).
6. Click sign in.

4.4.19.3 Output Specifications

The user is redirected to the staff landing page.

4.4.19.4 Intercase Dependencies

The stylist is already created via 4.4.20. This is required so that the user has valid credentials to log in to the system with.

4.4.20

Test Case 20: Create Stylist or Admin Account (Administrator)

Example: This test case will test the function required for creating a new stylist or admin. This process is almost identical with the only exception being a single field which determines the account type.

Steps:

1. Admin is logged in, see 4.4.19.
2. Mouse over the menu drop down list and select the Admin button.
3. Click on the create user button in the upper left.

4.4.20.1 Test Items

- UserController.java
- StylistAvailability.java
- Availability.java
- DateRange.java
- create-user.html

4.4.20.2 Input Specifications

Steps:

1. Enter the following information:
   a. First name: Mike
   b. Last name: McParland
   c. Email: thisWillFailValidation.com
   d. Password: password
   e. User type: Stylist
2. The create button should be disabled due to the invalid email address.
3. Change the email:
   a. Email: validEmail@test.com
4. The create button should now be enabled.
5. Click the create button.
6. Back on the admin page, ensure that the new user is there.

4.4.20.3 Output Specifications

The system should return to the Admin page after clicking on create and the new user should be there.

4.4.20.4 Intercase Dependencies

The admin must already logged in in order to view the admin.html page, the login procedure is specified in 4.4.19.

4.4.21

Test Case 21: Reactivate/Deactivate Accounts (Administrator)

Example: This test case will test the administrator's ability to activate and deactivate any user.

Steps:

1. Admin is logged in, see 4.4.19.
2. Mouse over the menu drop down list and select the Admin button.

4.4.21.1 Test Items

- UserController.java
- admin.html
- Tables: users

4.4.21.2 Input Specifications

Steps:

1. The admin page has loaded and shows all of the users.
2. The admin finds the users they wish to modify.
3. Admin selects the activate/deactivate button.
4. The button should toggle to the opposite state.
5. Admin selects the activate/deactivate.

4.4.21.3 Output Specifications

The page should be as it originally was when the admin first viewed the page.

4.4.21.4 Intercase Dependencies

The admin must already logged in in order to view the admin.html page, the login procedure is specified in 4.4.19.

5. SYSTEM TEST / REQUIREMENTS TRACEABILITY

This section provides for a cross referencing of each test case to its software requirement specification (or specifications) and also to its design component (or components). The appropriate section and its title in each document are provided. Cells filled in with “n/a” represent areas that were not done in the specified document.

5.1 System Test / Requirements Specification / Design Component Traceability Matrix

System Test	Requirement Specification	Design Component
4.4.1 View Profile C	3.1.2 Use Case 2	5.1.1.3 Manage Profile - View Profile
4.4.2 Update Profile C	3.1.8 Use Case 8	5.1.1.3 Manage Profile - Update Profile
4.4.3 View Stylist C	3.1.6 Use Case 6	n/a
4.4.4 Make Appointment C	3.1.5 Use Case 5	5.1.1.1 Create/Manage Appointment - Create New Appointment
4.4.5 Cancel Appointment C	3.1.9 Use Case 9	5.1.1.1 Create/Manage Appointment - Cancel Appointment
4.4.6 View Past/Present Appointment C	3.1.10 Use Case 10	5.1.1.1 Create/Manage Appointment - View Past and Present Appointments
4.4.7 Login C	n/a	5.1.1.3 Manage Profile - Login
4.4.8 Create Profile S	3.1.11 Use Case 11	n/a
4.4.9 View Profile S	3.1.12 Use Case 12	5.1.2.1 Manage Profile - View Profile
4.4.10 View Client List S	3.1.13 Use Case 13	n/a
4.4.11 View Client Profile S	3.1.14 Use Case 14	n/a
4.4.12 Manage Appointment S	3.1.15 Use Case 15	n/a
4.4.13 Set Availability S	3.1.16 Use Case 16	n/a

4.4.14 Manage Services S	3.1.17 Use Case 17	n/a
4.4.15 Update Profile S	3.1.19 Use Case 19	5.1.2.1 Manage Profile - Edit Profile
4.4.16 Create Appointment S	3.1.20 Use Case 20	5.1.2.3 Manage Appointments - Create Appointment
4.4.17 Manage Pending Appointment S	3.1.21 Use Case 21	5.1.2.2 Manage Appointments - Approve/Deny Appointment
4.4.18 Cancel Appointment S	3.1.22 Use Case 22	5.1.2.2 Manage Appointments - Cancel Appointment
4.4.19 Login S/A	n/a	5.1.2.1 Manage Profile - Login
4.4.20 Create Stylist/Admin Account A	3.1.23 Use Case 23	5.1.3.1 Manage Users and Create Account - Create Account
4.4.21 Reactivate/deactivate accounts A	3.1.24 Use Case 24	5.1.3.1 Manage Users and Create Account - Manage Users

6. APPROVALS.

This section contains the list of the key signatories necessary to sign-off on the STS, thereby agreeing to the scope and content of the test plan and test cases specified within the document. Approval constitutes a guarantee that the development team has produced a test specification sufficient for validating the software to be delivered to the sponsor.
Each signatory should be identified by name, title and affiliation. A signature line should be provided along with a date line.

Advisor

By signing this document you are approving the agreements and design specifications defined within this document.

Ahmed M Salem ______________________________

Team Sierra

By signing this document you are agreeing to the design specifications defined within this document and complete the project as it is has been specified.

Alex Chernyak ______________________________

Joubin Jabbari ______________________________

Kyle Matz ______________________________

Mike McParland ______________________________

Scott Livingston ______________________________

Serge Lysak ______________________________

Appendix A - Software Problem Report Template

This section provides a sample template for reporting software problems that are discovered during the course of performing the test cases detailed in this document.

SOFTWARE PROBLEM REPORT   Problem Report ID ________

PROGRAM ________________________ RELEASE ________   VERSION _________

REPORT TYPE		SEVERITY	ATTACHMENTS: ☐ Yes ☐ No
☐ Coding Error	☐ Documentation	☐ Fatal	If yes, list attachments
☐ Design Error	☐ Hardware	☐ Serious	____________________________________
☐ Suggestion	☐ Query	☐ Minor	____________________________________

PROBLEM SUMMARY ____________________________________

CAN YOU REPRODUCE THE PROBLEM? (Y/N) ___

PROBLEM AND HOW TO REPRODUCE IT ____
____
____
____
____

SUGGESTED FIX (optional) ____
____
____
____
____

REPORTED BY _________________________   DATE __/__/__

Items Below Are For Use Only By the Development Team

FUNCTIONAL AREA ___________________________ ASSIGNED TO ________________________________

COMMENTS ____
____
____
____

STATUS:		PRIORITY		
☐ Open	☐ Closed	☐ High	☐ Medium	☐ Low

RESOLUTION: RESOLUTION VERSION NO: ___________

☐ Pending	☐ Deferred	☐ Withdrawn by reporter
☐ Fixed	☐ As designed	☐ Need more info
☐ Irreproducible	☐ Can't be fixed	☐ Disagree with suggestion

RESOLVED BY _____________   DATE __/__/__

RESOLUTION TESTED BY _________________________________   DATE __/__/__

TREAT AS DEFERRED: ☐ Yes ☐ No