Test Plan

Advisely, An Education Support Application

Big Muscle Boys

Sam Elliott, Jens Hansen, Connor Langlois, Eli Legere, Eric Schessler, Spencer Ward

COS 420 UMaine Spring 2019

Professor: Sepideh Ghanavati

Unit Testing:

Component: Fireauth : Authenticate

Setup: Let test@test.com and “testPass” be an email/password pair that do not correspond to an Advisely account.

Call: token : Authenticate(“test@test.com”, “testPass”)

Assertion: assertNull(token)

Teardown: N/A

Component: Fireauth : Authenticate

Setup: Let test@test.com and “testPass” be an email/password pair that correspond to an Advisely account.

Call: token : Authenticate(“test@test.com”, “testPass”)

Assertion: assertNotNull(token)

Teardown: Remove the above account from the database.

Component: Fireauth : CreateAccount

Setup: Let test@test.com and “testPass” be an email/password pair that correspond to an Advisely account.

Call: token : CreateAccount(“test@test.com”, “testPass”)

Assertion: assertNull(token)

Teardown: Remove the above account from the database.

Component: Fireauth : CreateAccount

Setup: Let test@test.com and “testPass” be an email/password pair that do not correspond to an Advisely account.

Call: token : CreateAccount(“test@test.com”, “testPass”)

Assertion: assertNotNull(token)

Teardown: Remove the above account from the database.

FOR ALL REMAINING UNIT TEST CASES, THE USER MUST BE LOGGED IN.

Component: Controller : BuildSchedule

Setup: Let the database contain all courses designated in the course list passed to the method.

As well, let there be no conflicts among the courses.

Call: schedule : BuildSchedule(courses)

Assertion: assertNotNull(schedule)

Teardown: Remove the constructed schedule from the database.

Component: Controller : BuildSchedule

Setup: Let the course list passed in be empty.

As well, let there be no conflicts among the courses.

Call: schedule : BuildSchedule(courses)

Assertion: assertNotNull(schedule)

Teardown: Remove the constructed schedule from the database.

Component: Controller : BuildSchedule

Setup: Let the course list contain conflicting courses.

Call: schedule : BuildSchedule(courses)

Assertion: assertNotNull(schedule)

Teardown: Remove the constructed schedule from the database.

Component: Controller : UpdatePref

Setup: The list of preferences is empty

Call: newPrefs : UpdatePref(preferences)

Assertion: assertTrue(newPrefs)

Teardown: Revert preferences to initial state.

Component: Controller : NewClass

Setup: The user enters a list of courses they have previously taken.

Call: newClass : NewClass(courses)

Assertion: assertNull(newClass)

Teardown: Remove the courses from the account history

Component: Controller : GeneratePDF

Setup: The user has an empty schedule.

Call: pdf : GeneratePDF(schedule)

Assertion: assertNotNull(pdf)

Teardown: N/A

Component: Controller : GeneratePDF

Setup: The user has an arbitrary non-empty schedule.

Call: pdf : GeneratePDF(schedule)

Assertion: assertNotNull(pdf)

Teardown: N/A

Component: Controller : GenerateTXT

Setup: The user has an empty schedule.

Call: txt : GenerateTXT(schedule)

Assertion: assertNotNull(txt)

Teardown: N/A

Component: Controller : GenerateTXT

Setup: The user has an arbitrary non-empty schedule.

Call: txt : GenerateTXT(schedule)

Assertion: assertNotNull(txt)

Teardown: N/A

Component: Controller : RecommendMajors

Setup: The user has taken zero courses.

Call: majors : RecommendMajors(courses, currMajors)

Assertion: assertNull(majors)

Teardown: N/A

Component: Controller : RecommendMajors

Setup: The user has taken some arbitrary number of courses that fit several graduation requirements.

Call: majors : RecommendMajors(courses, currMajors)

Assertion: assertNotNull(majors)

Teardown: N/A

Component: Controller : RecommendCourses

Setup: The user has taken zero courses

Call: courses : RecommendCourses(courses, currMajors)

Assertion: assertNotNull(courses)

Teardown: N/A

Component: Controller : RecommendCourses

Setup: The user has taken some arbitrary number of courses.

Call: courses : RecommendCourses(courses, currMajors)

Assertion: assertNotNull(courses)

Teardown: N/A

Component: Controller : ImportSchedule

Setup: N/A

Call: schedule : ImportSchedule(scheduleTXT)

Assertion: assertNotNull(schedule)

Teardown: Remove the newly entered schedule from the users page.

Component: Schedule : IsValid

Setup: The schedule contains zero courses.

Call: b : schedule.IsValid()

Assertion: assertTrue(b)

Teardown: N/A

Component: Schedule : IsValid

Setup: The schedule contains some number of non-conflicting courses.

Call: b : schedule.IsValid()

Assertion: assertTrue(b)

Teardown: N/A

Component: Schedule : IsValid

Setup: The schedule contains at least 2 courses which conflict.

Call: b : IsValid()

Assertion: assertNotTrue(b)

Teardown: N/A

Component: DatabaseManager : SaveSchedule

Setup: N/A

Call: b : SaveSchedule(schedule)

Assertion: assertTrue(b)

Teardown: Remove the schedule from the database.

Component: DatabaseManager : SavePreferences

Setup: N/A

Call: b : SavePreferences(preferences)

Assertion: assertTrue(b)

Teardown: Remove the new preferences from the database.

Component: DatabaseManager : SaveCourse

Setup: N/A

Call: b : SaveCourse(course)

Assertion: assertTrue(b)

Teardown: Remove the course from the database.

Component: DatabaseManager : GetClasses

Setup: N/A

Call: courses : GetClasses(criteria)

Assertion: assertNotNull(courses)

Teardown: N/A

Component: DatabaseManager : GetSchedule

Setup: N/A

Call: schedule : GetSchedule()

Assertion: assertNotNull(schedule)

Teardown: N/A

Component: DatabaseManager : GetClassesTaken

Setup: N/A

Call: courses : GetClassesTaken()

Assertion: assertNotNull(courses)

Teardown: N/A

Component: DatabaseManager : GetMajors

Setup: N/A

Call: majors : GetMajors()

Assertion: assertNotNull(majors)

Teardown: N/A

Use Case Testing:

Use Case: Login

User	System
1) User enters username and password	
1a) The system fails to read the input from the user	
1a1) The system prompts the user to try again	
2) The user presses the “Login” button	
2a) The login button fails to register	
2a1) The system prompts the user to reload the app	3) System authenticates the username and password
	3a) The system fails to authenticate
	3a1) The system prompts the user to try again
	3b) The user entered an invalid password
	3b1) The system prompts the user to try again
	3c) The user entered an invalid email
	3c1) The system prompts the user to try again
	3d) The entered password and email don’t match
4) The user is now logged in	3d1) The system prompts the user to try again

Use case: Create Account

User	System
1) User enters username and password	
1a) The system fails to read the input from the user	
1a1) The system prompts the user to try again	
2) The user presses the “Create Account” button	
2a) The Create Account button fails to register	
2a1) The system prompts the user to reload the app	3) System authenticates the username and password
	3a) The system fails to authenticate
	3a1) The system prompts the user to try again
	3b) The user entered an invalid password
	3b1) The system prompts the user to try again
	3c) The user entered an invalid email
	3c1) The system prompts the user to try again
	3d) The entered password already exists
4) The user is logged in on new account	3d1) The system prompts the user to try again
4a) The system fails to log the user in	
4a1) The system prompts the user to re-enter login credentials	

*Invalid passwords are defined as empty password fields.

Use case: Logout

User	System
1) The user presses the logout button	
1a) The button fails to register	
1a1) The system prompts the user to try again	2) The system displays a confirmation message
	2a) The system fails to display the confirmation message
3) The user is logged out	2a1) The system proceeds to 3)
3a) The system fails to log the user out	
3a1) The system prompts the user to close the app	

Use case: View Profile

User	System
1) The user presses the view profile button	
1a) The button fails to register	
1a1) The system prompts the user to try again	2) The system displays the view profile page
	2a) The system fails to display the page
	2a1) The system prompts the user to try pressing the button again or restart app

Use case: Share Schedule

User	System
1) The user presses the share button	
1a) The button fails to register	
1a1) The system prompts the user to try again	2) The system displays a social media selection pop up
	2a) The system fails to display the pop up
3) The user selects a social media platform	2a1) The system prompts the user to try again or restart the app
3a) The selection fails to register	
3a1) The system prompts the user to select another platform	4) The system displays the platform-specific sharing interface
	4a) The system displays to fail the interface
5) The user selects the people they wish to share with	4a1) The system prompts the user to try a different platform
5a) Selection fails	
5a1) System prompts user to select new platform due to interface failure	6) The System shares the schedule as a png
	6a) The System fails to share
	6a1) System prompts user to select new platform due to interface failure

Use case: Customize Appearance

User	System
1) The user presses the customize appearance button	

1a) The button fails to register	
1a1) The system prompts the user to try again	2) The system displays the customization interface
	2a) The system fails to display the interface
3) The user selects customization options	2a1) The system prompts the user to try again or restart the app
3a) Customization option selections fail to register	
3a1) The system prompts the user to restart the customization interface	
4) The user selects the “Okay” button to confirm	
4a) The button fails to register	
4a1) The system prompts the user to try again	5) The system updates the user’s preferences and the the schedule appearance
	5a) The system fails to update the appearance
	5a1) The system prompts the user to try again or restart the app

Use case: Submit Evaluations

User	System
1) The user selects the class in their schedule	
1a) The class no longer exists or fails selection	
1a1) The system prompts the user to try again or select a different class	2) The system displays the page for that class

	2a) The system fails to display the page
3) The user presses the review button	2a1) The system prompts the user to try again or restart the app
3a) The button fails to register	
3a1) The system prompts the user to try again	4) The system displays review input page
	4a) The system displays to fail the page
5) The user enters feedback into the review field	4a1) The system prompts the user to try again or restart the app
5a) Review field is empty	
5a1) System prompts the user to enter valid input into the field	
6) The user hits the “Okay” button	
6a) The button fails to register	
6a1) The system prompts the user to try again	7) The system stores the review

Use case: Select Desired Majors

User	System
1) The user presses “edit majors” button	
1a) The button press is not registered	2) The system displays the edit majors form
	2a) The system doesn’t display the form
3) The user selects which majors they’d like	2a1) The system prompts the user to try again
3a) the user does not select any majors	
3a1) The system prompts the user to select some modification	4) The system updates the users majors in the database

	4a) The system fails to enter the majors in the database
	4a1) The system attempts to enter the courses again.

Use case: Import Previous Courses

User	System
1) The user clicks “Import previous courses” button	
1a) The system doesn’t register the button press	
1a1) The user tries again	2) The system displays the “Import previous courses form”
	2a) The form fails to display
3) The user enters the courses they’ve already completed	2a1) The system attempts to display the form again
3a) The user enters no courses	
3a1) The system displays an invalid input message and requests new input.	
3b) The user enters partial course information	
3b1) The system requests the remaining information about the course	4) The system adds the course to the users history
	4a) The system fails the enter the course information
	4a1) The system attempts to add the course again.

Use case: Suggest Schedule Options

User	System
1) User requests course suggestions	2) The system displays a criteria form
	2a) The system fails to display the above form
3) The user selects criterias	2a1) The system attempts to display the form again
3a) The user enters no search criteria	
3a1) The system displays random courses	
3b) The user enters conflicting criteria	
3b1) The system displays “Invalid criteria” and requests a change	4) The system searches and displays valid courses
	4a) The system fails to display courses
	4a1) The system attempts to display the courses again

Use case: Suggest Additional Degrees

User	System
1) The user requests additional degree options	
1a) The system fails to recognize the button press	
1a1) The user presses the button again	2) The system reviews courses the user has completed to generate suggestions.
	2a) The system fails to find any recommendations

	2a1) The user displays a “no results found” message.
	3) The system displays all valid results to the user

Use case: Build Schedule

User	System
1) The user clicks the “add to schedule” button	
1a) The button press is not registered by the system	
1a1) the user attempts to click the button again.	2) The system displays the “add class” form
	2a) The form fails to display
3) The user selects the semester to add this course to	2a1) The system attempts to display the form again
3a) The user selects no semester	
3a1) The system notifies the user it requires a semester.	4) The system adds the selected course to the selected semester
	4a) The system fails to add the course
	4a1) The system attempts to add the course again.

Use case: Search For Classes

User	System

1) User clicks search	
1a) The system doesn’t register the press	
1a1) The user attempts to click search again	2) The system displays the search form.
	2a) the system fails to display the search form
3) The user enters their search criteria	2a1) The system attempts to display the search form again.
3a) The user enters no criteria	
3a1) The system displays 10 random classes	
3b) The user enters criteria which conflict	
3b1) The system notifies the user and requests new input	
3c) The criteria are too restrictive and yield no courses	
3c1) the system informs the user and displays a new search form.	4) The system displays the courses that are found
	4a) The system fails to display the courses
	4a1) The system attempts to display the courses again

Use case: Specify Class Constraints

User	System
1) The user clicks the “Specify class constraints” button	
1a) The system doesn’t register the	

button press	
1a1) The user clicks the button again	2) the system displays the “specify constraints form”
	2a) The form fails to display
3) The user enters their desired criteria	2a1) The system attempts to display the form again
3a) The user does not enter any criteria	
3a1) The system removes all criteria from the user’s account	
3b) the user enters invalid constraints	
3b1) The system requests valid constraints and leaves the form as it.	4) The system stores the new constraints within the database
	4a) The system fails to store the constraints
	4a1) The system attempts to store the constraints again

Acceptance Testing: Acceptance testing shall be done through periodic usability studies. These studies will be as described in the “Usability Study” document. They will have users perform tasks in order to determine if the application is functional, intuitive, and overall useful/pleasant to use. Because this form of testing cannot be completed by the development team internally, screenshots are omitted. Instead, Q&A results will be submitted along with general comments from users. This information will be analyzed and changes will be made based on the results.