Test Plan Document

UPlan

Richard Hartnett, Joe Menzie, Zach Matthews, Steve Lucey

1. Introduction

The software product we are developing is a self-planning planner called UPlan. How it works is first the user will create an account. When creating this account they will fill in certain preferences such as sleep schedule to help the application schedule efficiently based on when the user goes to sleep. The next thing the user will have to do is add events. These events could be classes or other events going on in a specific week. Next the user will enter their assignments as they come in. When entering these assignments they will be asked to enter how long they believe it is going to take, when the due date is, and whether it is a high priority assignment or not. UPlan's algorithm will take these assignments, compare them to the classes or other events the user has for the day, and plan a study schedule based on the free time available. This study schedule will include which assignments to work on that day and for how long. The idea is, the user will use this generated schedule and roughly follow it to keep them on track for their day to day and long term assignments.

The user will be able to edit these events and assignments to alter the time they occur or other settings related to them. One very important aspect the user will need to know for the application to work correctly is at the end of each day or couple days, they will need to go back and update how much time they have left for each assignment. This needs to be done because a lot of time when someone starts an assignment they predict it will take a certain amount of hours but in reality to will take maybe half as long or twice as long. To factor in for this inaccuracy, the user will need to continuously update how far they have come and how much more they have to do.

2. Quality Control

2.1 Test Plan Quality

The test process we are going to take is testing a the methods made within the client side of the software. This will ensure that anything added to the database through the server is happening and has no errors. The selection of hour test cases came from having errors adding information to the client during the building of the program. Now being able to test what it added to the client and ensure that it made it to the database allows us to remove a lot of errors. This test plan is concrete because if these tests fail then our program itself will no longer be running. These test are testing the base of our project which ensures that the test plan is solid.

2.2 Adequacy Criterion

Testing is done when the inputs into the client make it into the database. The test come back as positive if and only if the output matches the solution to the given inputs of the test. The testscript.sh file that is held within our application code. This file is what allows Travis CI, our testing machine, to come back with a zero or a one as the output. Zero means the test ran smoothly and one means the test failed.

3. Test Strategy

The testing tool Travis CI, which links right to you GitHub, needs many files to ensure that it can run a test. The first file that needs to be created is a .travis.yml file. This file tells travis what kind of test it is going to be running and which directory it is going to be testing. The next file that needs to be made it a makefile. This makefile is part

of what complies the test cases that are being run. Next is the test script file. This file is the bases of the output of the test that are being run. This will return a zero if output matches solution file and one is output does not match. The way the program works is the the driver takes in the input files and then makes an output file which then compares the results to the solution file.

4. Test Cases

Here we specify all the test cases to be used for system testing. These test cases form a part of the system testing plan. For test case specifications we specify the different conditions that need to be tested, along with the test cases used for testing those conditions and the expected outputs. The test cases have been selected using the functional approach. Test cases have been selected for valid inputs.

I. Login and Account

Test Cases	**Purpose**	**Steps**	**Expected results**	**Actual Results**
Login	Ensures login works	**-** Creates Account - Prints Username and Password	- Username - Password	**-** Username - Password
Account	Ensures account can be made	**-** Creates Account - Returns Account info	**-** Username - Password - Email - Fullname - Bedtime - Waketime	**-** Username - Password - Email - Fullname - Bedtime - Waketime

II. Creating Events and Assignment

Test Cases	**Purpose**	**Steps**	**Expected results**	**Actual Results**
Create Assignment	Ensures an assignment can be made	**-** Creates Account - Logs into program - Creates assignment - Returns 1 if made 0 if failed	- 1 - 1 - 1	- 1 - 1 - 1
Create Event	Ensures an event can be made	**-** Creates Account - Logs into program - Creates event - Returns 1 if made 0 if failed	- 1 - 1 - 1	- 1 - 1 - 1

III. Updating Information

Test Cases	Purpose	Steps	Expected results	Actual Results
Update Event	Ensures that Events can be updated	- Creates Account - Logs into program - Creates an update to made to event - Returns 1 if made 0 if failed	- 1 - 1 - 1	- 1 - 1 - 1
Update Assignment	Ensures that Assignments can be updated	- Creates Account - Logs into program - Creates an update to make to an assignment - Returns 1 if made 0 if failed	- 1 - 1 - 1	- 1 - 1 - 1
Update Profile	Ensures that Profile can be updated	- Creates Account - Logs into program - Creates an update to make to the profile - Returns 1 if made 0 if failed	- 1 - 1 - 1	- 1 - 1 - 1

IV. Deleting Information

Test Cases	Purpose	Steps	Expected results	Actual Results
Delete Event	Ensures that Events can be deleted	- Creates Account - Logs into program - Deletes an event - Returns 1 if made 0 if failed	- 1 - 1 - 1	- 1 - 1 - 1
Delete Assignment	Ensures that Assignments can be deleted	- Creates Account - Logs into program - Deletes an assignment - Returns 1 if made 0 if failed	- 1 - 1 - 1	- 1 - 1 - 1
Delete Profile	Ensures that Profile can be deleted	- Creates Account - Logs into program - Deletes a profile - Returns 1 if made 0 if failed	- 1 - 1 - 1	- 1 - 1 - 1