402 Software Test Plan/Test Procedure Document - USchedule

Owners: Annie Flora, Amelia Jay, Liam Namba, Cristalina Nguyen, Sophia Prochnow, Christian Santander

8.0 Software Test Plan

8.1 Introduction

This test plan document covers all the tests we will use to do testing on our project, and covers all levels of the testing processes we will perform. We will provide details of the unit test, integration test, and acceptance test. The document will serve to describe what is tested/how and to describe the actual test steps we will use to prove the code and application works the way it is intended to. Both positive and negative tests will be performed.

8.2 Unit Test Plan

Our algorithm will be tested to make sure all constraints are satisfied and edge cases accounted for. We will hard code tests for “mock” residents and their information. We will use this mock hard coded data to make sure that our block, ward, and shift schedule algorithm are working properly.

We will also be testing our front end to make sure everything is working as intended. We will perform manual tests on all buttons, dropdowns, checklists, etc.

8.2.1 Unit Tests Planned

For both our block, ward, and shift scheduling algorithms, we are testing these using mock data. We are creating residents and assigning them to a name and year.

With the block algorithm, this is taking in the residents and assigning them to a block in the schedule. There are 3 blocks in each schedule and in each block, every ward (4 wards total) needs to have at least 2 residents working (one first year, one second year). Thus, this algorithm should assign roughly all 24 residents to a specific block labeled 1-3. We are testing to make sure that when running this algorithm, no resident is

assigned more than 1 block, and each block should have roughly an even number of first year and second year residents.

With the ward schedule, we are taking these 8 mock residents who have been assigned to the same block, and are placing them in wards either B,C, D, or F. When testing this algorithm, we are making sure there are at least 2 residents assigned to each ward, one being a first year and one being a second year. And only a second year is placed in ward F. When placed in a ward, this should edit the resident's resident.ward field.

Once the ward schedule is run, the shift algorithm will assign these 8 mock residents to 8 shifts (one for each week in the block they are assigned). The shift options are: off, day float (DF), short call (SC), long call (LC), and night float (NF). One of these weeks will be off which will also be mock data we assign for testing. This week off will be a resident field numbered 1-8. When testing our shift algorithm, we make sure that no 2 residents in the same ward are working the same shift in the same week. We also have to make sure certain constraints are satisfied such as LC always precedes NF. We are assigning the shifts in either of 2 orders depending on which week the resident will have off. OFF, DF, SC, LC, NF, DF, SC, LC Or SC, LC, NF, DF, OFF, SC, LC, DF. This algorithm should then change the resident.shift field.

8.2.2 Unit Test Procedures

For the algorithms we will test each on the terminal to ensure that they return the intended information.

For manual testing, we will follow the steps below:

1. Start the application
    a. Should take user to the home page
2. Navigate to the resident page by clicking on the residents page button
    a. Ensure that you can see a list of residents and resident data
3. Click on the button to add a new resident
    a. Should see fields to input this data
4. Fill in proper information and click the button "add resident"
    a. Should see fields resident name, year
5. Ensure that this new resident is added to the list and has the proper information inputted
6. Enter the search bar on the top of the resident page
    a. User should be able to input text and search through the residents
    b. Once something is entered into the search bar, the relevant residents should be displayed below

7. Click either name or block
    a. Should sort the residents based on which was selected
8. Click the USchedule logo to ensure it brings the user back to the home screen
9. Visit the Create Schedule page by clicking on the create schedule button
10. Click the button create schedule on this page
    a. Should display a schedule either block or full using all of the residents in the resident management page
11. Click the button to visit schedule page
12. On this page the user should see the most recent full schedule

8.3 Integration Test Plan

8.3.1 Integration Tests Planned

The tests will be performed on the different parts of the algorithm separately, and together. The first test will be run on the block component of the algorithm. The code will be run on the terminal and ensure that the output includes all residents, assigning them to a block 1-3 and satisfies all constraints.

We will then run the ward section of the algorithm on the terminal with the output from the block algorithm. We will take one of the blocks with 8 residents and use this algorithm to assign them each to a ward B, C, D, or F. We will make sure that each resident is assigned to a ward and that at least one first year and one second year is in each of these.

Lastly, we will run the ward algorithm with the shift algorithm. We will take these residents in each ward and assign them to 8 shifts, which gives them one per week. When we run this algorithm on the terminal, we ensure that the returned results satisfy all constraints.

We will also ensure that we run each of these algorithms separately, with hard coded resident data.

8.3.2 Integration Test Procedures

To test each of our algorithms, we will open up the terminal and run each of the algorithms separately. Once run, we will evaluate the results, to make sure that what is returned is what is intended and all constraints are satisfied

To test the front end, we will assign multiple team members to navigate around the application to make sure each page is linked, all buttons work, and all intended output is visible.

8.4 Acceptance Test Plan

8.4.1 Acceptance Tests Planned

To ensure that the entire application works as planned, we will test that the database is linked with our back end and that our algorithm is able to run with our intended resident data. We will add residents and resident information to the database, and refresh our application.

8.4.2 Final Acceptance Test Procedures

Once the application is refreshed, the inputted residents should show up on our resident page in the application with all the correct information about each.

We will then run our algorithm with the resident data from the database. We will create a block schedule which should include residents in our database. We will look at the resident information such as year to confirm that constraints are satisfied.

When running the shift algorithm with the database, we will assign each resident a week off number 1-8.Then, when we run the algorithm in the application, we should see that each resident has been assigned the week off that we had given to them in the database.

8.5 Test Configuration Control

Our tests will fit into our Configuration Management plan as they are being used to document, control, implement, account for, and mark changes that are being made to various components of our project. These tests will identify changes that need to be made with any bugs or mistakes in our code. Continuously testing our algorithms separately, as well as together, will help to manage change and help us to meet our deliverables on time and without errors.

We are not using any separate software for our testing. We are simply creating mock data, running this through our algorithms, and evaluating it to ensure we are outputting the correct results. All our front end testing will be manual.

8.6 Items Not Tested

We are not testing our database. However, when we run our algorithm with the database we will be making sure that this information is accurate. We will also call queries to the database to ensure we are receiving the correct resulting information.

We will also not be writing specific tests for the front end. We will be manually testing this.

8.7 Test Verification Matrix

Requirement	Specifics	Test
5.2.1 Graphical User Interface (GUI)	5.2.1.1 The GUI shall display a Login Page 5.2.1.2 The GUI shall display a Home Page 5.2.1.3 The GUI shall display a Scheduling Page 5.2.1.4 The GUI shall display a Resident Management Page 5.2.1.5 The GUI shall display an Output Page	Manual tests navigating through the front end
5.2.2 Server CSC	5.2.2.1 The server shall get data from database 'GET' 5.2.2.2 The server shall put data into database 'POST' 5.2.2.3 The server shall run the algorithm to determine optimized schedule 5.2.2.4 The server shall validation check for schedule	Manual tests navigating through the front end and adding residents on the to ensure they are then stored in the back end
5.2.3 Database CSC	5.2.3.1 The database shall display a schedule schema 5.2.3.1.1 The schedule schema shall display an Employee table 5.2.3.1.1.1 The Employee table shall display an ID, First Name,	Manually looking through the database No set tests for this

	Last Name, year, office ID, shift ID 5.2.3.1.2 The schedule schema shall display an Organization table 5.2.3.1.3 The schedule schema shall display a Shift table	
5.3 Functional Requirements by CSC	5.3.1 The GUI shall display a login page 5.3.2 The GUI shall display a home page 5.3.3 The GUI shall display a scheduling page 5.3.4 The GUI shall display a resident management page 5.3.5 The GUI shall display an output page 5.3.6 The server shall get data from and push data to the database using API calls 5.3.7 The server shall support our algorithm for creating schedules to run on it 5.3.8 The server shall support unit testing 5.3.9 The database shall have a schema named ‘Schedule’	Manual tests navigating through the front end Algorithm testing
5.4 Performance Requirements by CSC	5.4.1 The user shall be logged in within 10 seconds 5.4.2 The user shall be directed from the home page to their desired page within 5 seconds 5.4.3 The user shall be able to navigate between pages in 4 seconds	Manually testing the front end Algorithm tests - block, ward, and shift

	5.4.4 The system will read information from the database within 30 seconds 5.4.5 The database shall make changes submitted by the user within 10 seconds 5.4.6 The schedule shall be generated within 1 minute 5.4.7 The schedule shall be displayed for viewing within 15 seconds 5.4.8 The schedule will be re-generated for validation check within 1 minute	