WARE

Web Application for Robotics Education

University of Nevada, Reno

Department of Computer Science and Engineering

Acceptance Criteria and Testing Strategy and Plan

Team 17

Zachery Wiles, Ryan Lunt, Sean Griffith, Herman Hira

Instructors: David Feil-Seifer and Devrin Lee

Project Sponsor
Dr. Rui Wu
Associate Professor
East Carolina University

External Advisor
Ben Gallagher
Security Analyst
University of Nevada, Reno

March 12, 2021

1. Abstract

The project team is developing a web application named WARE, a full stack web application designed specifically for higher education robotics classes. WARE utilizes the OpenAI Gym framework, an existing open-source project, and provides a user interface to the framework's robotics environments.

The project is important as it allows users to develop and compile robotics code in an online environment. The client-server architecture gives users freedom from the time consuming and process intensive task of installing and running the framework on a local machine.

WARE incorporates the following major features: front-end code editor with syntax highlighting, back-end code compilation, the ability to create user accounts for students and instructors, and the ability to access the site on the public Internet.

2. Project Updates and Changes

The WARE project development is progressing with the implementation of additional features that are integral to the application. One of these features is user management, which gives the application the ability to create user accounts for students and instructors. In addition, parallel submission processing has been implemented. This will allow multiple users to simultaneously submit code to the application without error. Progress tracking has also been implemented within WARE. This feature will display to the user the progress made on an environment and whether or not the environment is complete on the environments page. Submission caching was also implemented, allowing the application to save the code entered by the user in an environment so that they can come back to it at a later time without starting from scratch. Finally, submission validation has been added into the overall code submission process within an environment. This added process ensures that the user has entered code that invokes the correct environment.

Security within the application will be a major focus in future development of WARE. Because the application stores email addresses and passwords, it is important that the application provides adequate safeguards to protect this information from attackers. The project team will be using a Flask extension called Flask-Security which gives access to basic security and account management controls that are commonly seen in web applications today. Furthermore, implementing error messages for key user interaction sequences will be a priority in order to improve the overall user experience when an unexpected input is received.

3. User Stories and Acceptance Criteria

3.1 Front-End Subsystem

User Story: **Class Creation**
As an instructor that has signed in, I want to be able to create a new class so that I can assign my students a list of environments to experiment with that supplement their learning outside of lectures.
Acceptance Criteria
● **AC1**: The class creation page is only accessible to users with an account type of instructor. ● **AC2**: Upon login to their account, and after navigating to their homepage, the instructor is able to select a “Create Class” button. ● **AC3**: Upon selecting the “Create Class” button, the user is redirected to the class creation page. ● **AC4**: Upon accessing the class creation page, the user is presented with a form that can be filled out with class information. ● **AC5**: Instructors will be able to select as many environments as they wish from the list of available environments for their students to be able to experiment with. ● **AC6**: The class creation page form may not be submitted by the instructor until all fields are filled with relevant information. ● **AC7**: Upon successful class creation, a randomly generated class ID is displayed beneath the newly created class’s name on the instructor’s homepage. ● **AC8**: Upon successful class creation, the instructor is returned to their homepage where they are able to see a dropdown menu for the class they have created, containing the list of environments they have selected as well as any students that have registered for their class. ● **AC9**: Upon unsuccessful class creation, the instructor is notified of the problem and the class creation page’s form fields are cleared.

User Story: **Student Progress Display**
As an instructor of a class, I want to be able to see the progress each of my students have made in every assigned environment so that I can provide feedback on their work.
Acceptance Criteria
• **AC1**: When the instructor opens their homepage, the list of classes they are instructing is displayed. • **AC2**: For each class created by an instructor, a dropdown menu may be opened to display each of the environments selected for that class. • **AC3**: For each environment displayed in the class environment list dropdown menu, an additional dropdown menu may be selected to display the progress of each student in the selected environment.

3.2 Environment Page Subsystem

User Story: **Retention of Submitted Code**
As a user experimenting with an environment, I want to be able to retain code submitted to the server so that I can make changes to it instead of having to start from scratch.
Acceptance Criteria
• **AC1**: The user can input their own custom solution to the environment code box. • **AC2**: The user can submit their code to the web server using the “Compile and Run” button. • **AC3**: After the submission is processed by the web server, the environment page is refreshed with the results of the submission. • **AC4**: After the submission is processed by the web server, the code used for the submission is loaded in the environment code box for the user to continue their work. • **AC5**: If the web server was able to execute the user’s submission, that user’s submission is saved in a unique file on the server so that the user may return at another time to continue their work.

User Story: **Submission Processing Indication**
As a user experimenting with an environment, I want to be shown an animation after submitting code so that I can be sure my submission is being processed.
Acceptance Criteria
● **AC1**: The user can submit their code to the web server using the “Compile and Run” button. ● **AC2**: Upon submitting their code using the “Compile and Run” button, the button is darkened, disabled, and an animated loading circle is prepended to the button’s text. ● **AC3**: Upon receiving the results for their submission, the “Compile and Run” button is returned to its original state.

3.3 Account Management Subsystem

User Story: **User Registration**
As a student who has not registered for an account, I want to be able to register for an account so that I can experiment in the robotics environments assigned to me by my instructor.
Acceptance Criteria
● **AC1**: WARE Homepage has a link to the sign-up page that can be utilized by a user who is not logged in. ● **AC2**: Upon accessing the sign-up page, the user is presented with a form that can be filled out with account information. ● **AC3**: If the user selects the “Student” option, they are presented with an additional field to input the code for the class they wish to join. ● **AC4**: The sign-up page form may not be submitted by the user until all fields are filled with relevant information. ● **AC5**: Upon successful account creation, the user is redirected to the sign-in page. ● **AC6**: Upon unsuccessful account creation, the user is notified of the problem and the sign-up page’s form fields are cleared.

User Story: **User Login**
As a User who has already registered for an account, I want to be able to login to my account so that I can begin experimenting in the robotics environments available to my account.
Acceptance Criteria
● **AC1**: WARE Homepage has a link to the sign-in page that can be utilized by a user who is not logged in. ● **AC2**: Upon accessing the sign-in page, the user is presented with a form that can be filled out with their account's information. ● **AC3**: The sign-in page form may not be submitted by the user until all fields are filled with relevant information. ● **AC4**: Upon successful login, the user is redirected to their homepage and can engage in available environments. ● **AC5**: Upon unsuccessful login, the user is notified of the problem and the sign-in page's form fields are cleared.

3.4 Submission Processing Subsystem

User Story: **Submission Result Display**
As a student submitting code for evaluation, I want to receive results related to my submission so that I can improve my work and understand what my submission accomplished.
Acceptance Criteria
● **AC1**: The user can submit their code to the web server using the “Compile and Run” button. ● **AC2**: The web server validates the submitted code to ensure the user is using the correct environment for their submission. ● **AC3**: The web server records all console output from the submitted code. ● **AC4**: The web server records all error output from the submitted code. ● **AC5**: The web server records the state of the environment as the submitted code solution is running and saves that data in a video. ● **AC6**: Upon receiving the results of their submission, the user who made the submission can see text output from their submission, including any errors that occurred. ● **AC7**: Upon receiving the results of their submission, the user who made the submission can see a video playback depicting the state of the environment as their code was executed.

User Story: **Tracking Student Progress**
As an instructor of a classroom, I want my students' progress in each assigned environment to be tracked so that I can ensure that my students are progressing as expected.
Acceptance Criteria
● **AC1**: When the web server receives a valid submission from a user, the submission is executed, and submission results are logged. ● **AC2**: Submission results are checked to determine if the user's submission managed to complete the objective of the environment. ● **AC3**: If the user's submission managed to complete the objective, the progress of the user in that environment is changed to "completed". ● **AC4**: If the user's submission successfully executed, the progress of the user in that environment is changed to "started". ● **AC5**: If the user has already submitted a solution for the environment that completed the objective, the user's progress does not regress from "completed" to "started" if a new submission fails to complete the objective.

3.5 Database Subsystem

User Story: **Sensitive Data Security**
As a user of the WARE website, I want sensitive information related to my account to be secured so that malicious actors cannot gain access to it.
Acceptance Criteria
● **AC1**: User account information is stored in the database. ● **AC2**: The password for each user account is stored as a hash value so that it is not retrievable if a malicious actor gains access to the database.

User Story: **Saving the Best Submission**
As a student in a class, I want my best submission in each assigned environment to be saved so that I will never lose progress when discovering a solution to an assigned environment.
Acceptance Criteria
• **AC1**: Each user is uniquely identifiable. • **AC2**: Each environment is uniquely identifiable. • **AC3**: For every environment a user has made progress in, the code submission which led to the accomplished progress is saved. • **AC4**: If a user submits a faulty solution to an environment, that solution will not overwrite a solution which made more progress. • **AC5**: If a user submits a solution to an environment that made less progress than a previous solution, that solution will not overwrite the previous best solution.

4. Testing Workflow

4.1 Happy Path Workflows

4.1.1 Happy Path Workflow 1: Registering an Instructor Account

This testing workflow is designed to test the process of registering an instructor account within WARE. Creating an instructor account is the first step to using WARE, as it is required that an instructor create an account before students are allowed to create their own accounts and start using WARE. In this workflow, the tester visits the account sign up page and provides a first name, last name, email address, and password. After the form is submitted, the workflow provides steps to validate that the account was created by having the tester provide the same email address and password at the login screen and check that the site accepts the login credentials.

Table 4.1: Testing workflow for registering an instructor account

Step #	Step Description	Expected Result
1	Navigate to the WARE homepage.	The web browser successfully loads and displays the WARE homepage.
2	Click the “Sign Up” button located	The Sign Up page is displayed to the

	on the right side of the navigation bar	user.
3	Enter a name into the First Name field	The First Name field displays the name entered.
4	Enter a name into the Last Name field	The Last Name field displays the name entered.
5	Enter a valid email address into the Email Address field	The Email Address field displays the email address entered.
6	Enter a password into the Password field. Ensure that the password meets the complexity requirements detailed underneath the Password field	The Password field displays the password entered using dots (·) for increased privacy.
7	Choose “Instructor” for Type of Account	The radio button under Type of Account highlights the “Instructor” radio button.
8	Click the “Sign Up” button at the bottom of the form	If successful, a page stating that the account has been created will be displayed.
9	Click the “Log In” button located in the navigation bar	The Login page is displayed to the user.
10	Enter the same email address used in Step 5 into the Email Address field	The Email Address field displays the email address entered.
11	Enter the same password used in Step 6 into the Password field.	The Password field displays the password entered using dots (·)
12	Click the “Sign In” button at the bottom of the form	If the account was successfully created, the instructor portal will be displayed. The user’s name will be displayed within the portal with the designation “Instructor.” Additionally, the name of the user will be displayed in the navigation bar.

4.1.2 Happy Path Workflow 2: Creating a Class

This testing workflow is designed to test the ability to create a class to be used by an instructor. Instructors will create a class after creating an account in order to create a unique identifier that their students can use when creating their own accounts. Additionally, instructors will be able to use classes to limit which environments are accessed by their students. This workflow guides the tester in creating a class in the instructor portal. The tester validates that a class has been created by viewing the instructor portal and verifying that the class is listed.

Table 4.2: Testing workflow for creating a class

Step	Step Description	Expected Result
1	Navigate to the WARE homepage.	The web browser successfully loads and displays the WARE homepage.
2	Click the “Log In” button located in the navigation bar	The Login page is displayed.
3	Enter an email address that is registered to an Instructor account into the Email Address field.	The Email Address field displays the email address entered.
4	Enter the password to the Instructor account that is linked to the email address used in Step 3 into the Password field.	The Password field displays the password entered using dots (·)
5	Click the “Sign In” button	The instructor portal is displayed for the specified account.
6	In the portal, click the “Create Class” button	A page will be displayed with a form.
7	Enter a name for the class in the “Class Name” field	The “Class Name” field displays the name of the class entered.
8	Under “Enabled Environments,” click the checkboxes next to the environments that should be available to students in the class.	The checkboxes selected under “Enabled Environments” are filled with a checkmark.
9	Click the “Create” button	If successful, the instructor portal will be displayed with the newly created class name listed under Classes. Additionally, a randomly generated

		class ID is listed under the Class Name that can be distributed to students.

4.1.3 Happy Path Workflow 3: Registering a Student Account

This testing workflow is similar to Happy Path Workflow 1, except that a Student account is created instead. This workflow guides the tester in creating a Student account with a valid classroom identifier. Afterwards, the workflow documents how to validate that an account with the Student type is created successfully, that the student account is correctly linked to the classroom that was used to sign up, and that the student account has access to the environments enabled by the classroom's instructor.

Table 4.3: Testing workflow for registering a student account

Step	Step Description	Expected Result
1	Navigate to the WARE homepage.	The web browser successfully loads and displays the WARE homepage.
2	Click the "Sign Up" button located on the right side of the navigation bar	The Sign Up page is displayed to the user.
3	Enter a name into the First Name field	The First Name field displays the name entered.
4	Enter a name into the Last Name field	The Last Name field displays the name entered.
5	Enter a valid email address into the Email Address field	The Email Address field displays the email address entered.
6	Enter a password into the Password field. Ensure that the password meets the complexity requirements detailed underneath the Password field	The Password field displays the password using dots (·) for increased privacy.
7	Choose "Student" for Type of Account	The "Student" radio button under "Type of Account" is highlighted.
8	Enter a valid class ID into the "Class ID" field	The "Class ID" field displays the ID entered.

9	Click the “Sign Up” button at the bottom of the form	If successful, a page stating that the account has been created will be displayed.
10	Click the “Log In” button located in the navigation bar	The Login page is displayed to the user.
11	Enter the same email address used in Step 5 into the Email Address field	The Email Address field displays the email address entered.
12	Enter the same password used in Step 6 into the Password field.	The Password field displays the password entered using dots (·)
13	Click the “Sign In” button at the bottom of the form	If the account was successfully created, the student portal will be displayed. Additionally, the name of the user will be displayed in the navigation bar.
14	Verify that the Class ID entered in step 8 is listed next to “Class ID” below the user’s name in the student portal.	If verification is successful, the student was added to the correct class.
15	Verify that the environments listed in the student portal are the same environments limited by the class that the student was registered in.	If verification is successful, the student account has access to only those environments enabled by the class.

4.1.4 Happy Path Workflow 4: Opening Robotics Environment

This test workflow provides a process for conducting a test procedure on opening a robotics environment within WARE. Being able to navigate to a robotics environment is of prime importance, as an environment is where a student writes, compiles, and runs code. The workflow first has the tester log in as a student. Afterwards, the tester chooses an environment from the list of available choices within the user portal. The workflow is validated when the tester can verify that the environment loads successfully within the tester’s browser.

Table 4.4: Testing workflow for opening a robotics environment

Step	Step Description	Expected Result
1	Navigate to the WARE homepage.	The web browser successfully loads

		and displays the WARE homepage.
2	Click the “Log In” button located in the navigation bar	The Login page is displayed.
3	Enter an email address that is registered to an Student account into the Email Address field.	The Email Address field displays the email address entered.
4	Enter the password to the Student account that is linked to the email address used in Step 3 into the Password field.	The Password field displays the password entered using dots (·)
5	Click the “Sign In” button	The student portal is displayed for the specified account.
6	Click an environment in the list of available environments presented in the portal	If successful, the environment loads in the web browser. A code editor is displayed on the left. A video of a robot and a terminal output is presented on the right. Beneath the code editor, two buttons are displayed: “Upload File” and “Compile and Run”

4.2 Unhappy Path Workflows

4.2.1 Unhappy Path Workflow 1: Logging in with a user account

This test workflow details the steps required to test the login functionality using incomplete or incorrect login information. The login mechanism is designed to be user friendly, indicating whether the username or password fields are empty and whether the username and password submitted together are incorrect. The mechanism is also designed to be secure, preventing unauthorized individuals from accessing a user’s account without the correct combination of username and password. This workflow validates that the login mechanism does not allow invalid username/password combinations to access the system. It also validates that the login mechanism displays helpful information to users in order to resolve legitimate login problems.

Table 4.5: Testing workflow for logging in with a user account

Step	Step Description	Expected Result

1	Navigate to the WARE homepage.	The web browser successfully loads and displays the WARE homepage.
2	Click the “Log In” button located in the navigation bar	The Login page is displayed.
3	Click the “Sign In” button at the bottom of the form.	A tooltip is displayed under the Email Address field that says, “Please fill out this field.”
4	Enter some text that is not in the form of an email address into the “Email Address” field	The “Email Address” field displays the text entered.
5	Click the “Sign In” button at the bottom of the form.	A tooltip is displayed under the Email Address field that says, “Please enter an email address.”
6	Enter an email address into the “Email Address” field that is not currently in the login database.	The “Email Address” field displays the email address entered.
7	Click the “Sign In” button at the bottom of the form.	A tooltip is displayed under the Password field that says, “Please fill out this field.”
8	Enter a password of the tester’s choice into the Password field	The Password field displays the password entered using dots (·)
9	Click the “Sign In” button at the bottom of the form.	A message appears above the “Email Address” field that states “Incorrect email address or password.”
10	Enter an email address that is registered to a Student account	The “Email Address” field displays the email address entered.
11	Enter an incorrect password for the email address entered in Step 7 into the Password field	The Password field displays the password entered using dots (·)
12	Click the “Sign In” button at the bottom of the form.	A message appears above the “Email Address” field that states “Incorrect email address or password.”
13	Enter an email address that is registered to an Instructor account	The “Email Address” field displays the email address entered.
14	Enter an incorrect password for the	The Password field displays the

	email address entered in Step 7 into the Password field	password entered using dots (·)
15	Click the “Sign In” button at the bottom of the form.	A message appears above the “Email Address” field that states “Incorrect email address or password.”
16	Enter an email address that is registered to a Student account	The “Email Address” field displays the email address entered.
17	Enter the password linked to the email address entered in Step 7 into the Password field	The Password field displays the password entered using dots (·)
18	Click the “Sign In” button at the bottom of the form.	If the email address and password match an account in the login database, the student portal is loaded. This validates that only the correct username and password combination, when matched against the database, will allow the user into the site.

Unhappy Path Workflow 2: Signing up for an account

This test workflow documents the procedure for verifying that the required restraints are met for a user when they are signing up for an account. In the user account database, an email address can only be linked to one user at a time. Additionally, the password must meet the minimum complexity requirements to establish a baseline form of security. Lastly, if a student is registering an account, it is important to ensure that the student uses a valid class ID. This workflow validates that all of these database requirements are being fulfilled.

Table 4.6: Testing workflow for signing up for an account

Step #	Step Description	Expected Result
1	Navigate to the WARE homepage.	The web browser successfully loads and displays the WARE homepage.
2	Click the “Sign Up” button located on the right side of the navigation bar	The Sign Up page is displayed to the user.
3	Enter a name into the First Name field	The First Name field displays the name entered.

4	Enter a name into the Last Name field	The Last Name field displays the name entered.
5	Enter an email address that is already linked to a user account.	The Email Address field displays the email address entered.
6	Enter a password into the Password field. Ensure that the password meets the complexity requirements detailed underneath the Password field	The Password field displays the password entered using dots (·) for increased privacy.
7	Choose “Instructor” for Type of Account	The radio button under Type of Account highlights the “Instructor” radio button.
8	Click the “Sign Up” button at the bottom of the form	An error message appears below the “Email Address” field that states “This email address is currently registered to an account.”
9	Enter an email address that is not linked to a user account.	The Email Address field displays the email address entered.
10	Enter a password into the Password field that does not meet the complexity requirements stated underneath the Password field	The Password field displays the password entered using dots (·) for increased privacy.
11	Click the “Sign Up” button at the bottom of the form	An error message appears below the Password field that states “The password does not meet the minimum complexity requirements”
12	Enter a password into the Password field. Ensure that the password meets the complexity requirements detailed underneath the Password field	The Password field displays the password entered using dots (·) for increased privacy.
13	Choose “Student” for Type of Account	The radio button under Type of Account highlights the “Student” radio button.
14	Enter some text into the “Class ID” field that does not match an existing Class ID in the database.	The “Class ID” field displays the ID entered.

15	Click the “Sign Up” button at the bottom of the form	An error message displays below the “Class ID” field that states “The Class ID entered does not match an existing class.
16	Enter a valid Class ID into the “Class ID” field	The “Class ID” field displays the ID entered.
17	Click the “Sign Up” button at the bottom of the form	If the email address does not currently exist in the database, the password meets the complexity requirements, and (if registering a Student account) a valid Class ID is given, then the user account will be created. A page stating that the account has been created will be displayed.

Unhappy Path Workflow 3: Submitting Code in an Environment

This test workflow documents the procedure for verifying that valid Python code is entered in the code editor upon submission. The Python code submitted by a user is considered valid if they are submitting code with the proper functions that call the environment that is currently in use. This workflow guides the tester in entering invalid code and checking if an alert is shown. The workflow validates that the submission validation mechanism is working as designed.

Table 4.7: Testing workflow for submitting code in an environment

Step	Step Description	Expected Result
1	Navigate to the WARE homepage.	The web browser successfully loads and displays the WARE homepage.
2	Click the “Log in” button located in the navigation bar	The Login page is displayed.
3	Choose a Student account that has access to the FetchPickAndPlace-v1 environment. Enter the email address of this account into the Email Address field.	The Email Address field displays the email address entered.
4	Enter the password to the Student account that is linked to the email address used in Step 3 into the	The Password field displays the password entered using dots (·)

	Password field.	
5	Click the “Sign In” button	The student portal is displayed for the specified account.
6	Choose the FetchPickAndPlace-v1 environment from the list of available environments.	If successful, the environment loads in the web browser.
7	In the code editor, change the environment name inside the gym.make() function from FetchPickAndPlace-v1 to FetchPush-v1.	The code editor displays the code entered.
8	Click the “Compile and Run” button	The terminal output window will display an error saying that the submission is invalid for the current environment.
9	In the code editor, delete the line that contains the gym.make() function.	The code editor deletes the code removed.
10	Click the “Compile and Run” button	The terminal output window will display an error saying that the submission is invalid for the current environment.
11	In the code editor, enter the following line into the code: env = gym.make("FetchPickAndPlace-v1")	The code editor displays the code entered.
12	Click the “Compile and Run” button	If successful, the environment will load the video and terminal output of the code entered in the code editor. This validates that the code submission component is checking for valid code that pertains to the current environment.

Unhappy Path Workflow 4: Browsing to Invalid Pages

This test workflow documents the procedure to check whether the dedicated 404 error page is displayed when a user attempts to navigate to a page that does not have a valid URL in WARE. If a user tries to navigate to a webpage that is not valid in WARE, a

dedicated error page will be displayed stating that the page does not exist. This workflow validates that the dedicated error page is being properly displayed and that no other internal server error pages are shown to the user.

Table 4.8: Testing workflow for browsing to invalid pages

Step	Step Description	Expected Result
1	Navigate to the WARE homepage.	The web browser successfully loads and displays the WARE homepage.
2	In the address bar of the web browser, add “/testing” to the domain.	Nothing happens within WARE.
3	Press the “Enter” key on your keyboard	A page with the WARE navigation bar appears. Below the bar, a message stating that the page could not be found is displayed.
4	Navigate to the WARE homepage.	The web browser successfully loads and displays the WARE homepage.
5	In the address bar of the web browser, add “/envs/20” to the domain.	A page with the WARE navigation bar appears. Below the bar, a message stating that the page could not be found is displayed.
6	Navigate to the WARE homepage.	The web browser successfully loads and displays the WARE homepage.
7	In the address bar of the web browser, add “/envs/2” to the domain.	If successful, an environment is loaded in the browser. This validates that all valid URLs are properly loaded, as opposed to invalid URLs which result in a dedicated error page.

5. Testing Strategy

For our web application, we will be using a variety of testing methodologies to make sure our application reaches high standards and performs as we intend it to. These testing methods include acceptance tests, component tests, and lastly, integration tests. Primarily, acceptance tests will overlap with component and integration tests. For the first phase of testing, we will be focusing solely on component tests. As a web application containing many different systems and subsystems, component tests will allow us to confirm that all the small parts of our website work independently. However, they also need to work together in systems and with the website as a whole. Once all of our component tests are successful, it isn't to say that everything will work when pieced together. We will need to follow up and make use of integration tests afterwards. This testing method will allow us to see how subsystems work together in a system or in conjunction with other systems that aren't entirely related to one another. We will then need to succeed with our integration tests to have a working website.

As far as testing strategy goes, we will be ensuring that simple rules will be met to maintain validity of such tests. Firstly, the tester(s) of a certain system or subsystem will not have been one of the members who developed that specific component. This is meant to bring a mindset of someone who is less familiar with the code or component (similar to the mindset of an actual user) to push boundaries and ‘not play by the rules’ or do what’s intended by the developer so to speak when using the component. The developer of the component wouldn’t be the best fit to test their own work for those two reasons despite knowing the ins and outs of their code. We have delegated testing responsibilities for each test in the chart below in the ‘Test Data or Situation’ column based on who didn’t work on a specific component. For example, Sean primarily focused on the backend so he will be testing the front end. Inversely, Zach worked primarily on the frontend side of things so we will be helping test the backend. Secondly, bugs or problems will be documented and reported to the team, along with an integer value associated with how severe the problem may be which will lead to prioritization. The developer associated with the component will then attempt to solve the issue and once again allow for a tester to test the component. This will be repeated until successful results occur.

We will declare our web application complete once we successfully implement our functional requirements and all testing has been conducted appropriately and all systems and subsystems operate in conjunction as intended. This will mean each test must be done at least once to ensure each component works. This will more than likely require multiple examinations due to finding potential defects for certain subsystems or components to make sure they are in working order. We plan on using compatibility testing on all subsystems to make sure everything works on its own, and then we will

move on to integration testing to test how all the small parts work with each other. Lastly, while all of these overlap with acceptance tests, we will consider our main acceptance test to be a full examination / integrated test of the website seeing if everything works together perfectly. The below chart shows important tests that need to be done based on the User Stories as well as from the Workflows.

Spreadsheet 5.4: Test Plan of important functionality that needs to be thoroughly tested.

Test #	Test Type	Target File or Screen	Test Name	Purpose of Test	Test Data or Situation	Expected Result	Actual Result	Outcome and Actions Required
1	Front-end Subsystem / Component Testing	ClassPage .html	Class Page Accessibility	To see if only the instructor accounts have the permissions to view the class creation page.	Tester: Sean 1. No account 2. Student account 3. Instructor account	1. Student accounts met with a '403 Forbidden' status. 2. Not being logged in will be met with a '403 Forbidden' status. 3. Instructor accounts can access the class creation page.	1. TBA 2. TBA 3. TBA	Wait for implementation.
2	Front-end Subsystem / Component Testing	ClassPage .html	Render Class Page	To see if the instructor is able to see the webpage that allows for a way to set up a class and input information.	Tester: Sean 1. Instructor account 2. Non-instructor based account.	1. Instructor accounts can access the various forms / other html widgets associated with creating a class. 2. The user will be met with a '403 Forbidden' status.	1. TBA 2. TBA	Wait for implementation.
3	Front-end Subsystem / Component Testing	ClassPage .html	Environment Selection	To see if the instructor is able to select which environments their class should have access too.	Tester: Sean 1. Instructor account 2. Non-instruc	1. The Instructor will be able to successfully choose environments that the class will have access to.	1. TBA 2. TBA	Wait for implementation.

					tor based account.	2. The user will be met with a '403 Forbidden' status.		
4	Front-end Subsystem / Component Testing	Class view	Class Dropdown	To see if the class creation was successful and that the user was able to view it.	Tester: Sean 1. Instructor account 2. Non-instructor based account.	1. The Instructor will be able to see a drop down menu from the navigation bar containing information of the class they created. 2. The user will be met with a '403 Forbidden' status.	1. TBA 2. TBA	Wait for implementation.
5	Front-end Subsystem / Component Testing	InstructorPage.html	Class Display	To see if the instructor is able to view their classes on their homepage.	Tester: Sean 1. Instructor account. 2. Non-instructor based account.	1. The instructor will be able to see all classes that they are instructing. 2. The user will be met with a '403 Forbidden' status.	1. TBA 2. TBA	Wait for implementation.
6	Front-end Subsystem / Component Testing	InstructorPage.html	Class Environment Display	To see if the instructor is able to successfully see all the environments they selected for a particular class.	Tester: Sean 1. Instructor account. 2. Non-instructor based account.	1. The instructor will be able to view all the environments they selected for any of their classes. 2. The user will be met with a '403 Forbidden' status.	1. TBA 2. TBA	Wait for implementation.
7	Front-end Subsystem / Component Testing	InstructorPage.html	Class Progress Display	To see if the instructor is successfully able to view all of the student's progress in a class for a given	Tester: Sean 1. Instructor account. 2. Non-instructor based	1. The instructor will be able to view all the status or progress of each student for any given environment they have	1. TBA 2. TBA	Wait for implementation.

				environment.	account	access to in any class. 2. The user will be met with a '403 Forbidden' status.		
8	Environment Page Subsystem / Component Testing	environment.html	Render Environment Page	To see if the environment page renders and formats correctly.	1. Render page is rendered	1. Environment page is formatted correctly and is user friendly	1.As expected	Standby and await as more developments or changes are made.
9	Environment Page Subsystem / Component Testing	environment.html	Submit Code	To see if the user is able to successfully submit and compile their code.	Tester: Herman 1. Student account 2. A user who isn't logged in	1. The user will be able to submit and compile their Python code. 2. The user will be redirected to the login page.	1.As expected 2. As expected	Standby and await as more developments or changes are made.
10	Environment Page Subsystem / Component Testing	environment.html	Code Results	To see if code is successfully processed and displays results to the user.	Tester: Herman 1. Student account	1. The user sees the page refresh with appropriate results in the visual and textual outputs.	1. As expected	Standby and await as more developments or changes are made.
11	Environment Page Subsystem / Component Testing	environment.html	Present Code	To see if the latest code the user submitted is saved and loaded into the environment to continue working on it.	Tester: Herman 1. Student account	1. The last code that the user submitted is present in the environment code box.	1. As expected	Standby and await as more developments or changes are made.
12	Environment Page Subsystem / Component Testing	environment.html	Save Code	To see if the current most successful submission of code the user submitted is stored in a	Tester: Herman 1. Student account	1. The user's latest code is successfully stored and saved in a file, ready to be continued on at any time the user chooses.	1. As expected	Standby and await as more developments or changes are made.

				file for later use.				
13	Environment Page Subsystem / Component Testing	environment.html	Disable Submission	To see if after a user submitted their code an animated loading circle is present along with a temporary disabling of the “Compile and Run” button.	Tester: Herman 1. After user pushes button	1. The user is presented with a loading animation while they wait momentarily for code to be processed, as well as no longer being able to press the “Compile and Run” button to prevent possible errors from occuring.	1. As expected	Standby and await as more developments or changes are made.
14	Environment Page Subsystem / Component Testing	environment.html	Enable Submission	To see if the “Compile and Run” button is enabled again after the page is refreshed and results are presented to the user.	Tester: Herman 1. After refresh	1. The user is able to press the “Compile and Run” button again to submit future changes to their work.	1. As expected	Standby and await as more developments or changes are made.
15	Environment Page Subsystem / Component Testing	environment.html	Upload File	To see if a user is able to upload a Python file (.py) and have code uploaded in the code editor.	Tester: Herman 1. File is a .py file. 2. File is not a .py file.	1. Code is placed in the code editor. 2. File will be rejected.	1. As expected 2. As expected	Standby and await as more developments or changes are made.
16	Account Management Subsystem / Component Testing	signup.html	Sign Up Page render	To see if the user is presented with the form to input information to create their own account.	1. A non logged in user 2. A logged in user	1. The not logged in user will see the form and can fill out information needed to create an account. 2. A logged in user clicking a link to this template will be	1. As expected 2. As expected	Standby and await as more developments or changes are made.

						redirected to the homepage.		
17	Account Management Subsystem / Component Testing	signup.html	Student Account Creation	To see if a user creating an account selected the 'Student' option an additional field will be presented that a user can input a class code into.	1. A non logged in user creating a student account. 2. A non logged in user creating an instructor account.	1. The user creating an account who selected the 'Student' option will be able to see a new field to input a code given by their instructor. 2. A user has the 'Instructor' button selected they will not be presented with the additional field.	1. TBA 2. As expected	Wait for implementation, as well as standby and await as more developments or changes are made.
18	Account Management Subsystem / Component Testing	signup.html	Sign Up Field Validity	To see if a user creating an account has to place relevant information into the fields in order to create the account.	1. User inputting valid information. 2. User inputting invalid information.	1. Users will be able to press the sign up button. 2. The user will be told of error(s) in the field(s) and will have to change to a valid input before being able to press the sign up button.	1. As expected 2. As expected	Standby and await as more developments or changes are made.
19	Account Management Subsystem / Component Testing	login.html	Render Login Page	To see if the fields are visible to the user in a clean format on the login page.	1. A non logged in user. 2. A logged in user	1. The user will be presented with fields to input their credentials as well as a few other account related buttons. 2. A logged in user will be redirected to the home page.	1. As expected 2. As expected	Standby and await as more developments or changes are made.
20	Account Management Subsystem /	reset-password.html	Recover Account	To see if a user is able to recover their account	Tester: Zach 1. A user requests an	1. The user will be sent a link to click on which allows them to change their	1. TBA	Wait for implementation.

	Component Testing			credentials through email.	email to be sent to change their password to recover their account.	password.		
21	Account Management Subsystem / Component Testing	login.html	User Login Field Validity	To see if the user is able to successfully log in to their account.	Tester: Zach 1. User inputs invalid information. . 2. User inputs valid field information.	1. The user will be displayed an error message telling them that the fields contained invalid information and will alert the user to input valid information. 2. The user will either be logged in or met with an error depending on if the credentials inputted matched the database.	1. As expected 2. As expected	Standby and await as more developments or changes are made.
22	Account Management Subsystem / Component Testing	login.html	User Login	To see if the user is able to successfully log in to their account	Tester: Zach 1. User inputs invalid credentials. 2. User inputs valid credentials	1. User is displayed with an error stating incorrect email or password. 2. User is logged in and redirected to the homepage.	1. As expected 2. As expected	Standby and await as more developments or changes are made.
23	Account Management Subsystem / Component Testing	login.html	User Session	To see if a user has an active session.	Tester: Zach 1. User is logged in	1. The user is logged in and has access to the environments made available to them.	1. As expected	Standby and await as more developments or changes are made.
24	Submission Processing Subsystem / Component	Submissio n handling	Submission Validation	To ensure the user is using the correct environment	Tester: Zach 1. The user is using the	1. Submit correctly 2. Error in terminal	1. As expected 2. As expected	Standby and await as more developments or changes are made.

	Testing			for their submission.	correct environment. 2. The user is not using the correct environment.			
25	Submission Processing Subsystem / Component Testing	Submission handling	Save Video Data	To see if video data is saved to be able to display it to the user.	Tester: Zach 1. The video data is saved 2. The video data is not saved	1. Video presented to the user on refresh. 2. Error in terminal	1. As expected 2. As expected	Standby and await as more developments or changes are made.
26	Submission Processing Subsystem / Component Testing	Submission handling	Submission Textual Output	To see if the web server will display textual output to the user, including error messages.	Tester: Zach 1. There is an error in the user's code 2. There is no error in the user's code	1. Error messages will be displayed helping the user find where the mistake was. 2. Any output produced by the code will be outputted as normal.	1. As expected 2. As expected	Standby and await as more developments or changes are made.
27	Submission Processing Subsystem / Component Testing	Submission handling	Submission Visual Output	To see if the web server will display the video / visual results to the user.	Tester: Zach 1. The user has not submitted any code. 2. The user has submitted code.	1. By default the visual will be there when they open the environment. 2. An error in the code will still display a visual to the user, although it won't be of the user completing the objective.	1. As expected 2. As expected	Standby and await as more developments or changes are made.
28	Submission Processing Subsystem / Component	Submission handling	Log Results	To see if the submission results are logged.	Tester: Zach 1. Data is logged and	1. Logs are successfully created.	1. As expected	Standby and await as more developments or changes are made.

	Testing				recorded on the backend.			
29	Submission Processing Subsystem / Component Testing	Submission handling	Check Objective	To determine if the user has successfully completed the environment's objective.	Tester: Zach 1. The user completed the environment's objective. 2. The user did not complete the environment's objective.	1. Upon successfully achieving the objective for the environment the status will be set to 'Completed'. 2. Upon any submission that does not complete the objective the user's status will be set to 'Started''.	1. As expected 2. As expected	Standby and await as more developments or changes are made.
30	Submission Processing Subsystem / Component Testing	Submission handling	Update Progress	To see if the progress of the user is successfully stored as well as updated for the user and instructor to see.	Tester: Zach 1. User has not started the environment. 2. User has started the environment but has not successfully completed it. 3. The user has completed the environment.	1. The user who has not started or submitted anything in an environment will have the status of 'Not Started' which appears as a blank loading bar. 2. Upon any submission that does not complete the objective the user's status will be set to 'Started''. 3. Upon successfully achieving the objective for the environment the status will be set to 'Completed'	1. As expected 2. As expected 3. As expected	Standby and await as more developments or changes are made.
31	Submission Processing Subsystem / Component	Submission handling	Lock Progress	To see if a user who has 'Completed' an	Tester: Zach 1. User alters code	1. User's 'Completed' status for the environment will stay the same	1. As expected 2. As expected	Standby and await as more developments or changes are made.

	Testing			environment is able to play around in the environment but not regress the status to an uncomplete d one.	in an environmen t and does not meet the objective. 2. User alters code in the environmen t and completes it.	even if their latest submission failed to meet the objective. 2. User's status will remain 'Completed' if they manage to complete it a second time despite using different code.		
32	Database Subsystem / Component Testing	Database System	Store User	To see if a user's data from the signup page are stored in the database.	Tester: Zach 1. User's data is stored successfull y and is able to be queried from the database. 2. User Data is not able to be stored or queried from the database	1. User data is stored in the database and is able to login as well as submit code in environments. 2. Error in console.	1. As expected 2. As expected	Standby and await as more developments or changes are made.
33	Database Subsystem / Component Testing	Database System	Hash Password	To see if a password from the signup page is successfully converted and stored as a hash.	Tester: Zach 1. User's password is successfull y hashed and stored in the database.	1. Password is no longer readable and will be compared with the hash of whatever the user inputs for the password field on login.	1. As expected	Standby and await as more developments or changes are made.
34	Database Subsystem / Component Testing	Database System	Check User	To check if a user is distinguisha ble from other users.	Tester: Ryan 1. A specific user is distinguisha ble in the	1. The user is distinguishable and will be able to submit code in environments.	1. As expected	Standby and await as more developments or changes are made.

					database.			
35	Database Subsystem / Component Testing	Database System	Check Environment	To check if an environment is distinguishable from other environments.	Tester: Ryan 1. A specific environment is distinguishable in the database	1. The environment is distinguishable and will be able to potentially accomplish objectives..	1. As expected	Standby and await as more developments or changes are made.
36	Database Subsystem / Component Testing	Database System	Prevent Bad Overwrite	To see if a submission with less progress is not saved over a submission with more progress towards the objective.	Tester: Ryan 1. A user submits code that makes more progress. 2. A user submits code that makes less progress.	1. Database saves the result. 2. Database does not save the result.	1. As expected 2. As expected	Standby and await as more developments or changes are made.
37	Browser / Integration Testing	Browser Compatibility	Chrome Compatibility	To see if WARE is compatible with Chrome web browser.	Tester: Ryan 1. WARE is accessed from Chrome	1. Browser displays everything correctly and in a user friendly way.	1. As expected	Standby and await as more developments or changes are made.
38	Browser / Integration Testing	Browser Compatibility	Microsoft Edge Compatibility	To see if WARE is compatible with Microsoft Edge web browser.	Tester: Ryan 1. WARE is accessed from Microsoft Edge	1. Browser displays everything correctly and in a user friendly way.	1. TBA	Wait for implementation.
39	Browser / Integration Testing	Browser Compatibility	FireFox Compatibility	To see if WARE is compatible with Firefox web browser.	Tester: Ryan 1. WARE is accessed from Firefox	1. Browser displays everything correctly and in a user friendly way.	1. As expected	Standby and await as more developments or changes are made.
40	Database &	Intersyste	Database	To see if the	Tester:	1. The user will	1. TBA	Wait for further

	Process handling / Integration Testing	m compatibility testing.	and Process Handling compatibility	database system and the Process handling system work successfully together.	Ryan 1. Database is able to successfully store and manage user submissions 2. Database is not able to successfully store and manage user submissions	be able to have code saved as well as being able to see results from submitted code. 2. Severe error - the user will not be able to do what the website expects of the user.	2. TBA	implementation and component tests.
41	Database & Account management / Integration Testing	Intersystem compatibility testing.	Database and Account management compatibility	To see if the database system and the account management system work successfully together.	Tester: Zach 1. Database is able to successfully store and manage user accounts and information relating to it. 2. Database is not able to successfully store and manage user accounts and information relating to it.	1. The user will be able to create an account, recover an account, and access environments. 2. Severe error - the user will not be able to access the majority of the website including environments.	1. TBA 2. TBA	Wait for further implementation and component tests.
42	Database & Account management & User	Intersystem compatibility testing.	Website Functionality	To see if all the main systems of the website	Tester: Ryan 1. All	1. The website is fully operational and just needs minor tweaking	1. TBA 2. TBA	Wait for further implementation and component tests.

	management / Acceptance Testing			work successfully together and create a working product.	systems and subsystems work together perfectly. 2. All systems and subsystems do not work together perfectly.	or changes. 2. Severe error - the website is not fully operational and will not be able to perform functions that must be completed from the student perspective as well as from the instructor perspective.		

6. Contributions of team members

- Outline **the** contribution of each team member on this document (specific activities or parts and amount of time spent, in hours).

Sean

Time spent: 5 hours

Sean worked on the User Stories and Acceptance Criteria section.

Ryan

Time spent: 5 hours

Ryan worked on the testing strategy section and created the testing plan.

Herman

Time spent: 1 hour

Herman helped write the Project and Updates section.

Zach

Time spent: 5 hours

Zach drafted the abstract, helped write the Project Updates and Changes section, and wrote the Testing Workflows.