WARE

Web Application for Robotics Education

University of Nevada, Reno

Department of Computer Science and Engineering

Design Document

Team 17

Zachery Wiles, Ryan Lunt, Sean Griffith, Herman Hira

Instructors: Sergiu Dascalu and Devrin Lee

External Advisor: Ben Gallagher

November 17, 2020

Table of Contents

1. Abstract

WARE is a robotics learning web application developed for students taking robotics classes. The goal of WARE is to give students a live coding experience to learn and interact with, while minimizing performance costs associated with compilation activities. WARE will comprise multiple learning environments in which students can learn and test their code. In a classroom format, instructors can evaluate student performance to gauge how well the students are learning the concepts. In addition, WARE will be optimized to run on low power devices such as smartphones and tablets to give students more opportunities to interact with the platform. The application will be built with HTML, Bootstrap libraries, JavaScript, Flask, and an SQL variant.

2. Introduction

This project focuses on the creation of a web application to be utilized by students to be able to better understand robotics and robotics-based principles by interacting and experimenting with several robotic-based environments. Since the last report, the specifications for this project have been made abundantly clear and requirements have been further solidified. Upon turning in our specification, we sent the sponsor of this project, Dr. Rui Wu, a copy so we could hold a meeting about the report. During the meeting, we discussed the strengths and weaknesses of the report, the areas in which we could improve, and what components we should focus on to begin the project. Additionally, we have revised the user interfaces that were originally created. We also went over requirements regarding us needing to host a webserver which we believe we have covered. While having a better understanding of our project as a whole is no doubt a step in the right direction, it comes with the many daunting realizations of the obstacles and challenges that lay ahead of us in creating this sort of application.

Overall, the main goal for this project is to be able to set up a web platform that communicates with a back-end web server to compile and execute user uploaded code and output the results to the user. The results are not just textual - but also a visual representation of what the code does. Another important goal we have is creating an accessible web application to allow students with any type of device to be able to access and use this resource in a simple and easy to use manner. For the time being we plan on following Dr. Wu's biggest advice to us - which is to try to implement the frontend-backend communication with regards to user submitted Python code as soon as we can as that is the core of this project.

3. High-level and Medium-level Design

The following diagrams and explanations give an overview of the architecture, subsystems, and program units that comprise the WARE application.

3.1 System Level Diagram

WARE is best characterized as having a layered architecture. The user accesses a front-end webpage. A back-end server processes information sent from the front-end webpage. For permanent storage, a database is utilized to store user and environment information for future access.

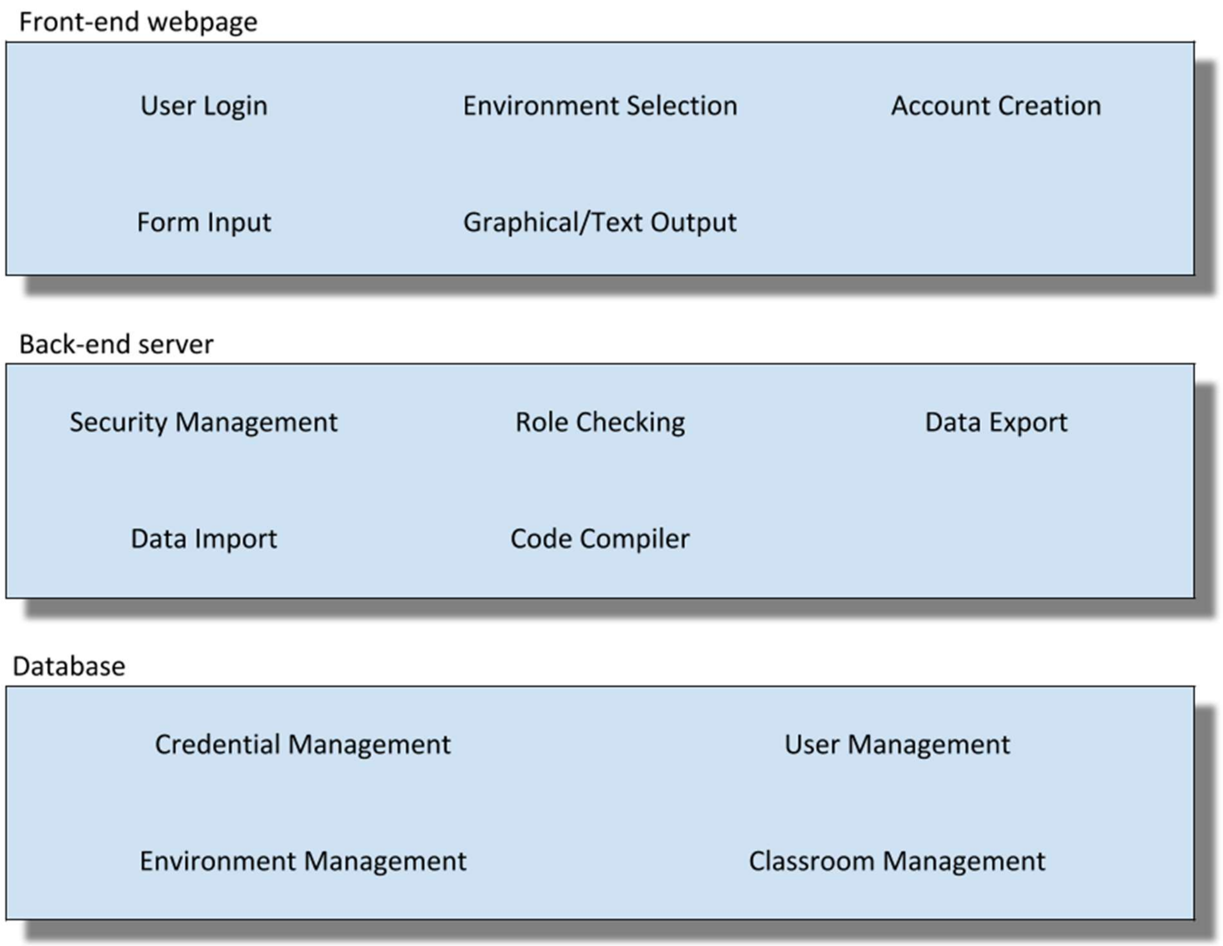


Figure 3.1: A layered architecture is presented in which there are three layers.

3.2 Program Unit Structure

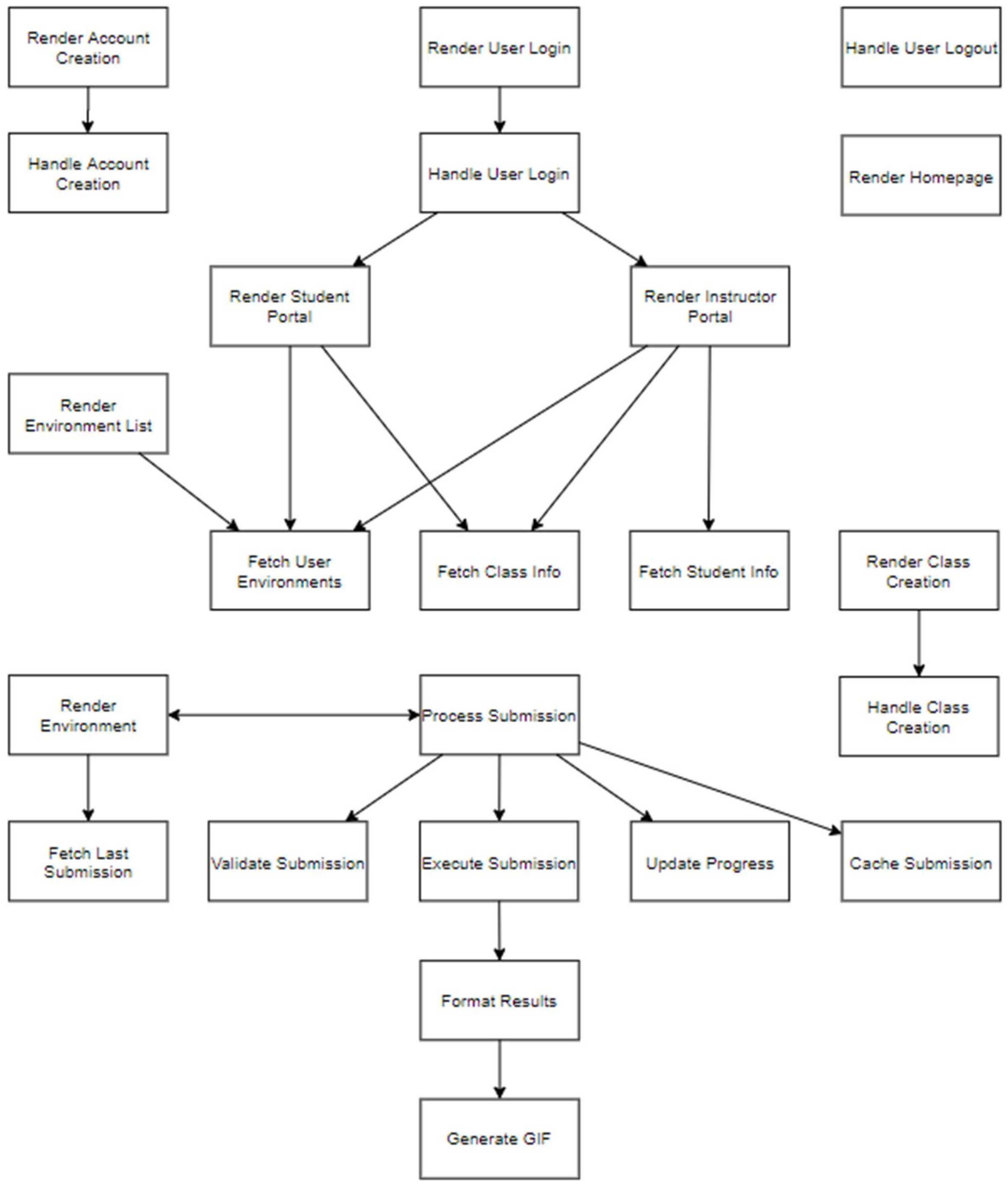


Figure 3.2.1: Call diagram for WARE program units. This diagram depicts the organization of program units and their associated unit calls within the WARE system.

Unit Name: RenderHomePage	
Description	This function generates the WARE home page using a stylized HTML template that includes a brief website description and links to the login and account creation pages.
Subsystem	Back-end server
Input	HTTP GET request
Output	HTTP response containing WARE home page data
Unit Calls	None
Exceptions	Failure to access the home page template will result in HTTP 500 internal server error.
Comments	Once the home page has been rendered, a user will be presented with a brief website description as well buttons that redirect the user to either the account creation page or the login page.

Unit Name: RenderAccountCreation	
Description	This function generates the WARE account creation page using a stylized HTML template that includes the account creation form.
Subsystem	Back-end server
Input	HTTP GET request
Output	HTTP response containing WARE account creation page data
Unit Calls	HandleAccountCreation
Exceptions	Failure to access the account creation page template will result in HTTP 500 internal server error.
Comments	Once the account creation page has been rendered, the user will have access to a registration form containing fields for their account information (first name, last name, email, class ID, password, confirmation of password, and account type). Once form data is filled out, the user may submit the form and attempt registration.

Unit Name: RenderUserLogin	
Description	This function generates the WARE login page using a stylized HTML template that includes a login form.
Subsystem	Back-end server
Input	HTTP GET request
Output	HTTP response containing WARE login page data
Unit Calls	HandleUserLogin
Exceptions	Failure to access the login page template will result in HTTP 500 internal server error.
Comments	Once the login page has been rendered, the user will have access to a login form containing fields for their account information (email and password). Once form data is filled out, the user may submit the form and attempt login.

Unit Name: RenderInstructorPortal	
Description	This function generates an instructor portal from an HTML template customized with a description of the instructor's class, and a list of all environments associated with that class.
Subsystem	Back-end server
Input	HTTP GET request
Output	HTTP response containing the instructor's user portal data
Unit Calls	FetchClassInfo, FetchUserEnvironments, FetchStudentInfo
Exceptions	Failure to access the instructor portal template will result in HTTP 500 internal server error.
Comments	Once the instructor portal has been rendered, the instructor will be able to select an environment, view a specific student's progress in an environment, or create a class.

Unit Name: RenderStudentPortal	
Description	This function generates a student portal from an HTML template customized with the student's class description, active environments and associated progress in those environments.
Subsystem	Back-end
Input	HTTP GET request
Output	HTTP response containing the student's user portal data
Unit Calls	FetchClassInfo, FetchUserEnvironments
Exceptions	Failure to find class info or environments will return HTTP 404 not found. Failure to access the student portal template will result in HTTP 500 internal server error.
Comments	Once the student portal has been rendered, the student will be able to select an active environment for experimentation or view the full environment list.

Unit Name: RenderEnvironmentList	
Description	This function generates the WARE environment list using a stylized HTML template that includes a listing of all environments associated with a user account.
Subsystem	Back-end
Input	HTTP GET request
Output	HTTP response containing a list of available environments
Unit Calls	FetchUserEnvironments
Exceptions	Failure to find environments associated with the user will return HTTP 404 not found. Failure to access the environment list template will result in HTTP 500 internal server error.
Comments	Upon the environment list being rendered the user will be able to view a selection of environments with their respective titles, thumbnails, and previews.

Unit Name: RenderEnvironment	
Description	This function generates the environment selected by the user.
Subsystem	Back-end
Input	HTTP GET request
Output	HTTP response containing specific information, data, and details regarding a single environment
Unit Calls	FetchLastSubmission
Exceptions	Failure to find environment data for the selected environment or an attempt to access an unassigned environment will return HTTP 401 unauthorized. Failure to access the environment template will result in HTTP 500 internal server error.
Comments	Upon the environment being rendered the user will be able to enter python code as well as being able to view the textual and visual result panels of the robotics environment.

Unit Name: FetchUserEnvironments	
Description	This function queries the database for a list of environments associated with a specified user.
Subsystem	Database
Input	User ID (UID)
Output	List of environments associated with the user
Unit Calls	None
Exceptions	Failure to find environments associated with the user will return HTTP 404 not found.
Comments	The list of user environments returned will depend on the class in which the user is participating in.

Unit Name: HandleAccountCreation	
Description	This function ensures valid account form data as well as that each account created is unique. If the form is valid, a user id is generated for the account and it is submitted to the database. If the form is invalid, the request for registration is denied and the user notified.
Subsystem	Back-end server
Input	HTTP POST request containing account creation form data
Output	HTTP response confirming outcome of registration request
Unit Calls	None
Exceptions	Failure to validate account creation form data will return HTTP 400 bad request.
Comments	Once a registration attempt is validated, the password is salted and hashed before storage in the database to ensure security of user information.

Unit Name: HandleUserLogin	
Description	This function compares account credentials submitted through login form to the account credentials stored within the user database table (Table 3.3.1) to validate or invalidate the login attempt.
Subsystem	Back-end server
Input	HTTP POST request containing login form data
Output	HTTP response confirming outcome of login request
Unit Calls	RenderStudentPortal, RenderInstructorPortal
Exceptions	Excessive login attempts will return HTTP 429 too many requests.
Comments	Upon a successful login, the user will be redirected to their user portal. Upon an unsuccessful login, the user will be notified of the failed login attempt and asked to retry.

Unit Name: HandleUserLogout	
Description	This function enables the user to logout of their account and redirects them to the WARE homepage
Subsystem	Back-end server
Input	HTTP POST request
Output	HTTP Response confirming successful logout request
Unit Calls	RenderHomepage
Exceptions	Failure to detect an active user session will immediately exit this unit.
Comments	Upon a successful logout the user will be signed out of his current session and redirected to the WARE homepage.

Unit Name: ProcessSubmission	
Description	This function processes a user's Python code submission and returns results of submission.
Subsystem	Back-end server
Input	HTTP POST request containing user submitted code
Output	HTTP response containing the textual and visual results from executing a user's code submission.
Unit Calls	ValidateSubmission, ExecuteSubmission, UpdateProgress, CacheSubmission, RenderEnvironment
Exceptions	Failure to validate the request will return HTTP 400 bad request.
Comments	This function will update the user's progress in the environment after submission, and save the most recent submission for querying by the instructor student who made the submission.

Unit Name: ValidateSubmission	
Description	This function validates code submitted to the web server by a user to ensure file type, valid file data, and the inclusion of environment-related library calls.
Subsystem	Back-end server
Input	User submitted code
Output	Validation or invalidation flag based on validity of user code
Unit Calls	None
Exceptions	Failure to recognize file type or data will raise an exception and exit the unit without validating the submission.
Comments	Upon successful validation, the user's code submission continues processing. Upon unsuccessful validation, the user is notified.

Unit Name: ExecuteSubmission	
Description	This function encapsulates a user's code submission in a separate process, executes the submission to capture results, and returns formatted results for user viewing.
Subsystem	Back-end server
Input	Validated user code submission
Output	Formatted results from executing the user's submission
Unit Calls	FormatResults
Exceptions	If an exception occurs during submission execution, it is captured for display to the user and execution is aborted.
Comments	Once user code has been executed and results are formatted, the results will be returned to the process submission unit for display to the user.

Unit Name: FormatResults	
Description	This function evaluates results generated during the execution of the submitted code and separates them into textual and visual results.
Subsystem	Back-end server
Input	Unformatted results of executing the user submission
Output	Formatted textual and visual results of executing the user submission
Unit Calls	GenerateGIF
Exceptions	Unexpected result values will halt execution of the unit and invalidate the submission.
Comments	Once the unformatted results are received during the execution of submitted code, the formatted results are returned for display to the user.

Unit Name: GenerateGIF	
Description	This function compiles visual results generated throughout the execution of user submitted code into a GIF file for visualizing the environment throughout execution to the user.
Subsystem	Back-end server
Input	Visual results of executing the user submission
Output	GIF data to visualize the environment throughout execution
Unit Calls	None
Exceptions	Failure to recognize visual results interrupts this unit and returns a preview image for the associated environment.
Comments	If no visual data is received, this unit is aborted and the preview image for the associated environment is returned.

Unit Name: CacheSubmission	
Description	This function will save the user's validated code submission to the database for later use by the student, or for analysis by the instructor.
Subsystem	Database
Input	Validated user code submission, Environment ID (EID), User ID (UID)
Output	Result of caching attempt (success or failure)
Unit Calls	None
Exceptions	Failure to store a submission due to file size will halt the unit and notify the user.
Comments	Only one submission per user may be saved to the database, if more than one submission is made, the latest submission replaces the prior submission.

Unit Name: UpdateProgress	
Description	This function will update the current progress percentage and progress bar for an environment.
Subsystem	Database
Input	Validated user code submission, Environment ID (EID), User ID (UID)
Output	Percentage value
Unit Calls	None
Exceptions	Failure to update progress will halt the unit and notify the user.
Comments	Updates the percent complete value for a student in an environment.

Unit Name: FetchLastSubmission	
Description	This function will retrieve a specific student’s last code submission in a certain environment.
Subsystem	Database
Input	User ID (UID), Environment ID (EID)
Output	Textual data for the student’s last submission
Unit Calls	None
Exceptions	Failure to retrieve last submission returns HTTP 404 not found
Comments	Data for the student’s last submission remains in the database.

Unit Name: FetchStudentInfo	
Description	This function will retrieve metadata relating to a specific student as well as that student’s progress in each environment associated with their class.
Subsystem	Database
Input	User ID (UID)
Output	Student metadata, progress in assigned environments
Unit Calls	None
Exceptions	Failure to retrieve student info returns HTTP 404 not found.
Comments	This function can only be called by a user with the instructor role.

Unit Name: RenderClassCreation	
Description	This function generates the class creation page using a stylized HTML template that includes the class creation form.
Subsystem	Back-end server
Input	HTTP GET request
Output	HTTP response containing class creation page data
Unit Calls	HandleClassCreation
Exceptions	Failure to access the class creation page template will result in HTTP 500 internal server error.
Comments	Upon the class creation page being rendered the user will have a form with various fields to fill out - including class name, and environments chosen.

Unit Name: HandleClassCreation	
Description	This function ensures valid class form data as well as that each class created has a unique name. If the form is valid, a class id is generated for the class and it is submitted to the database. If the form is invalid, the request for registration is denied and the instructor notified.
Subsystem	Back-end server
Input	HTTP POST request containing class creation form data
Output	HTTP response confirming outcome of registration request
Unit Calls	None
Exceptions	Failure to validate class creation form data will return HTTP 400 bad request.
Comments	This function can only be called by a user with the instructor role.

Unit Name: FetchClassInfo	
Description	This function will retrieve metadata related to the specified class ID, as well as metadata for any environment associated with that class.
Subsystem	Database
Input	Class ID (CID)
Output	Class metadata including name, associated environments
Unit Calls	None
Exceptions	Failure to retrieve class info returns HTTP 404 not found.
Comments	Class info cannot be changed by users after the class has been instantiated, however, it is available to all users (student or instructor) participating in the class.

3.3 Data Structures

User account information will be tracked using a database table utilizing the fields shown in table 3.3.1. New entries will be added to this database table as new users register to the system. Each user will have an associated user id (uid) generated for their account to accommodate a change in the user's email address used to access the website and will act as a primary key.

User Database Table Schema							
Value	**uid**	first_name	last_name	cid	password	user_type	email
Type	**<int>**	<string>	<string>	<int>	<string>	<string>	<string>

Table 3.3.1: Database table template including attributes and their respective types for user information (uid: User ID, cid: Class ID).

Class information will be tracked using a database table utilizing the fields shown in table 3.3.2. New entries will be added to this database table as users with the instructor role create them from their user portal. Each class will have an associated class id (cid) generated for it that instructors can distribute to their students to allow them to register for the class when creating their account, and will act as a primary key.

Class Database Table Schema				
Value	**cid**	class_name	instructor_name	environment_list
Type	**<int>**	<string>	<string>	<list<int>>

Table 3.3.2: Database table template including attributes and their respective types for class information (cid: Class ID).

Environment specific information will be tracked using a database table utilizing the fields shown in table 3.3.3. This database table will contain information critical to the display of each environment including the name, descriptions, and the filename of the preview image for that environment's display. Each environment will have an associated environment ID (eid) that will act as a primary key.

Environment Database Table Schema					
Value	**eid**	name	brief_description	description	preview_filename
Type	**<int>**	<string>	<string>	<string>	<filename>

Table 3.3.3: Database table template including attributes and their respective types for environment information (eid: Environment ID).

Student progress will be tracked using a database table utilizing the fields shown in table 3.3.4. This database table will be updated when a user submits their code within an environment, and is primarily used to track user progress in each environment. Additionally, the most recent code submission made by a user will be stored to allow a user to save their code for future use as well as for an instructor to review. Within this database table, the user id (uid) field will act as a primary key, with an associated environment id (eid) acting as a foreign key that can be used to reference the environment database table (Table 3.3.3).

Student Progress Database Table Schema					
Value	**uid**	*eid*	progress	last_submission_path	submission_date
Type	**<int>**	*<int>*	<float>	<filepath>	<datetime>

Table 3.3.4: Database table template including attributes and their respective types for student progress data (uid: User ID, eid: Environment ID).

4. Detailed Design

The following diagrams illustrate the user login, student information, code submission, and environment selection processes that are fundamental to the operation of WARE.

4.1 User Login

The sequence diagram shown in Figure 4.1 depicts the user login process, which involves the user inputting and submitting login credentials. The back-end server authenticates the user with the credentials and either allows or denies the user access.

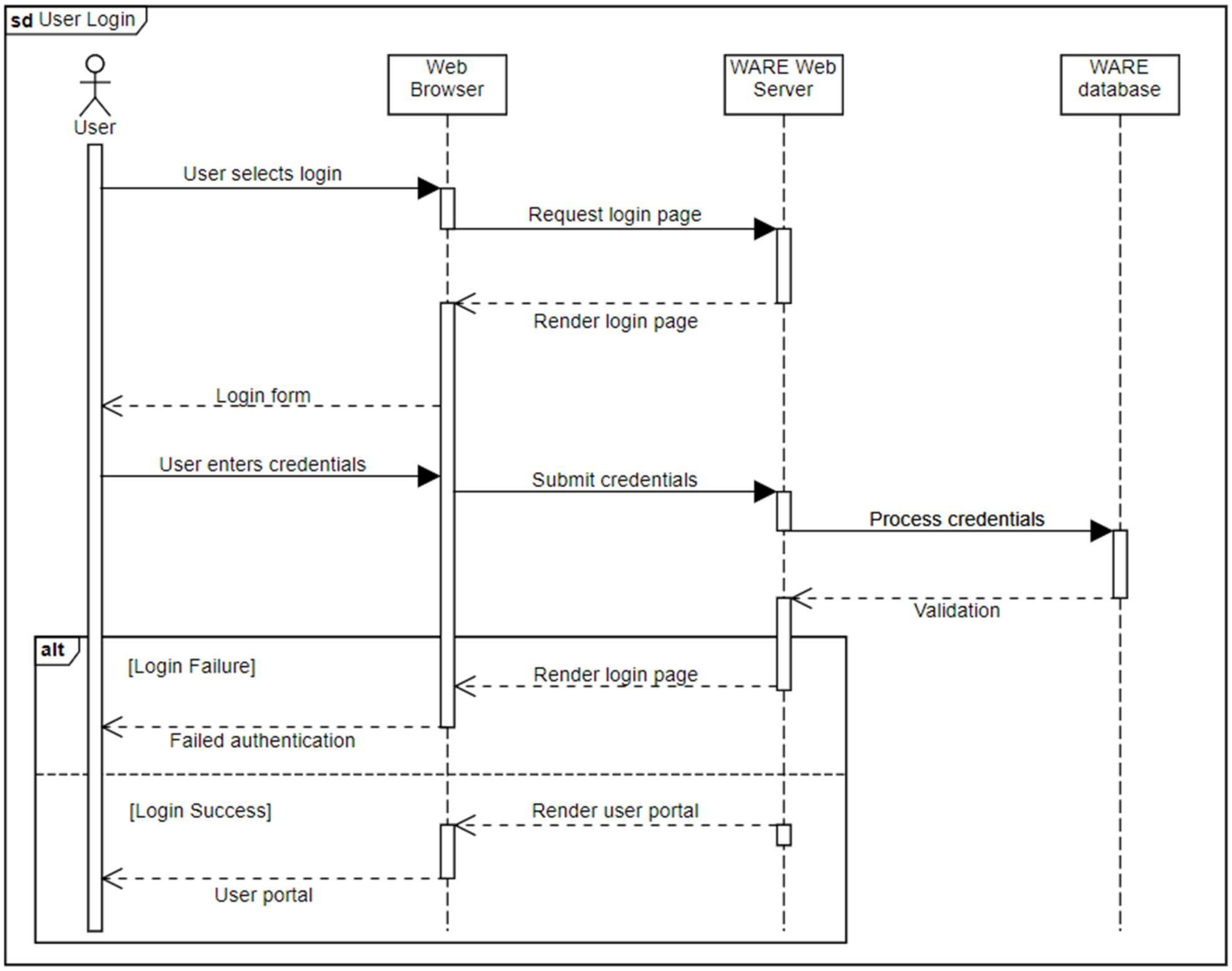


Figure 4.1: Sequence diagram for the user login process.

4.2 Student Information

The sequence diagram shown in Figure 4.2 shows the student information process, which allows an instructor to view information pertaining to a specific student including their progress in an environment and their last submission.

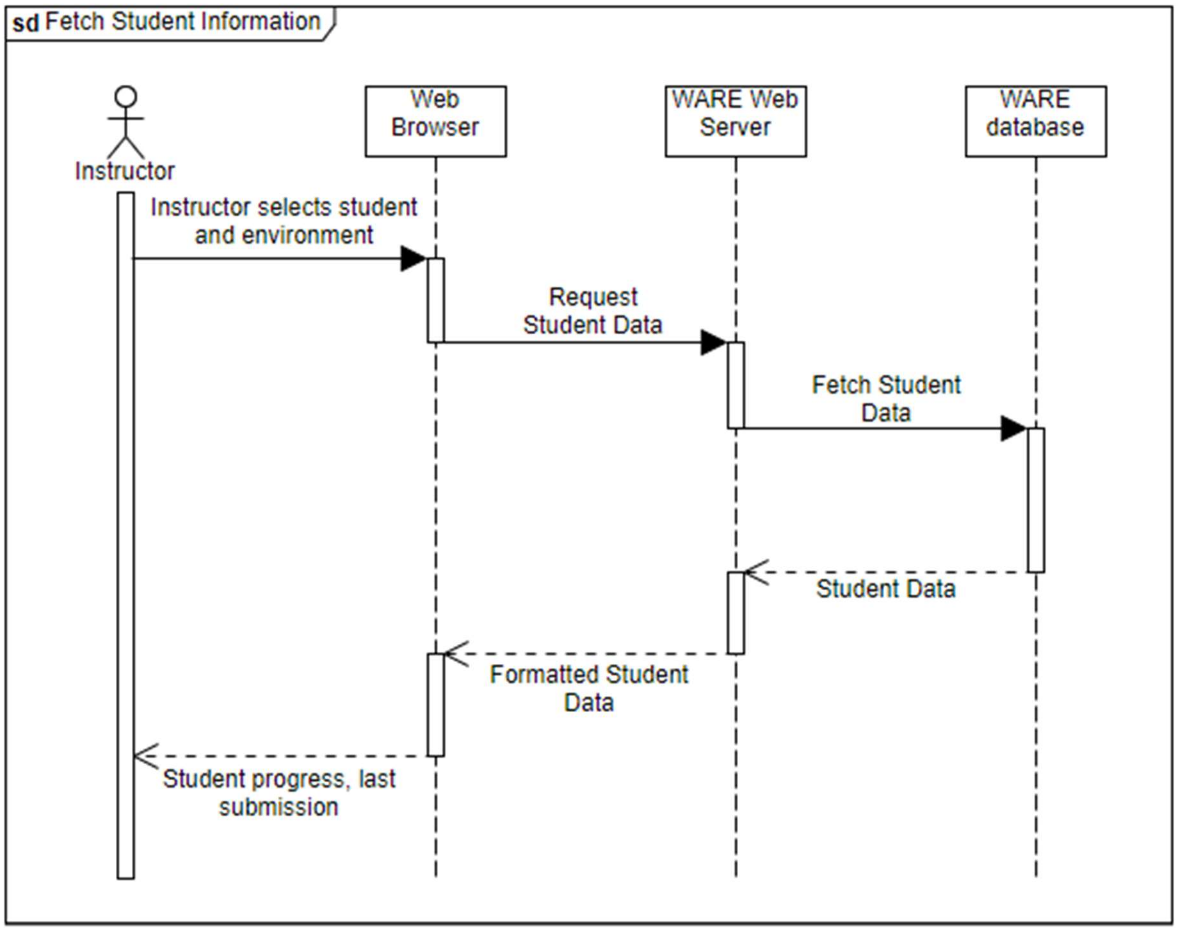


Figure 4.2: Sequence Diagram for the process of viewing student information.

4.3 Code Submission

The activity diagram shown in Figure 4.3 depicts the process for handling user submitted code. This process includes the receipt of user code, the validation process, execution of user code, the generation of visual and textual results for display to the user, updating user progress, and caching the submission.

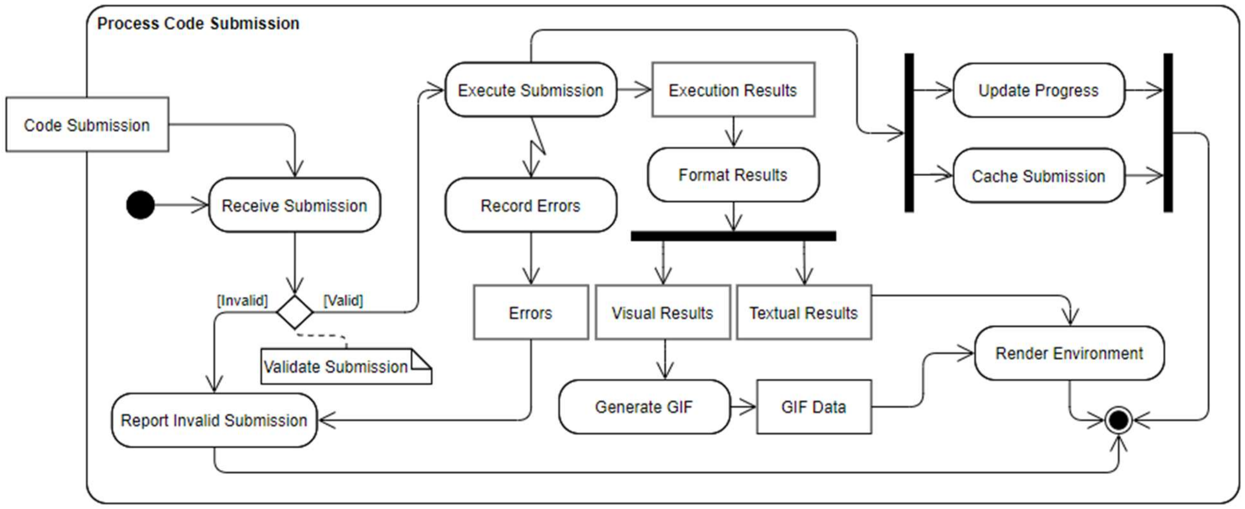


Figure 4.3: Activity Diagram for the code submission process.

4.4 Environment Selection

The environment selection process allows a user to select a robotics environment. Furthermore, it allows a user to return to a previously visited environment to restore their progress.

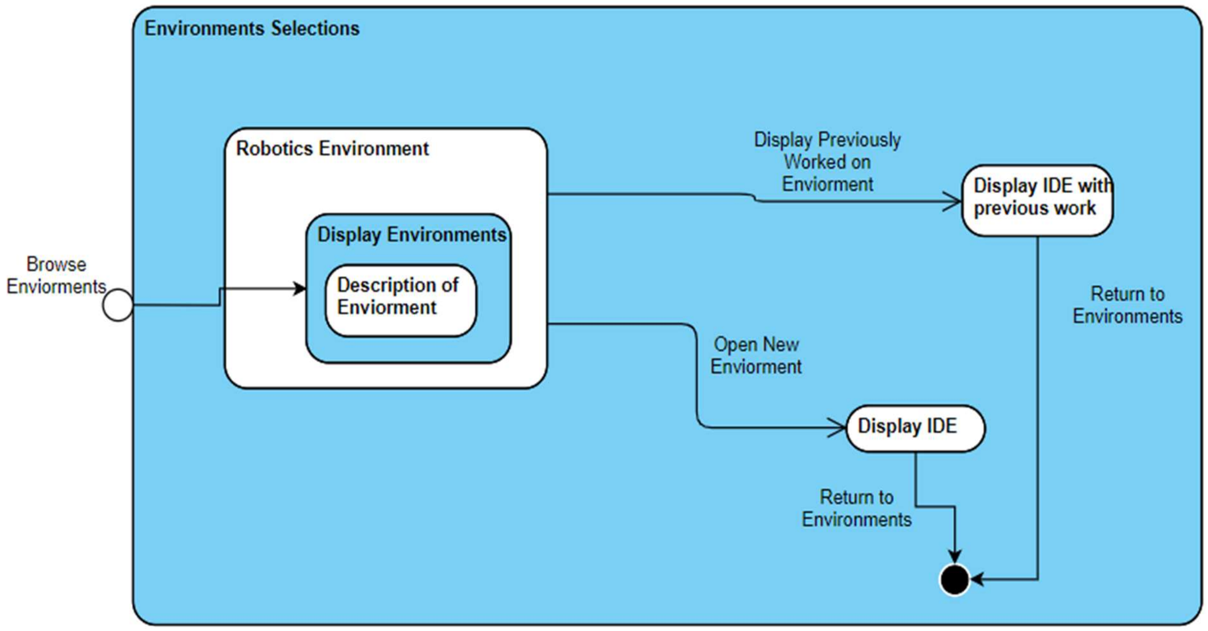
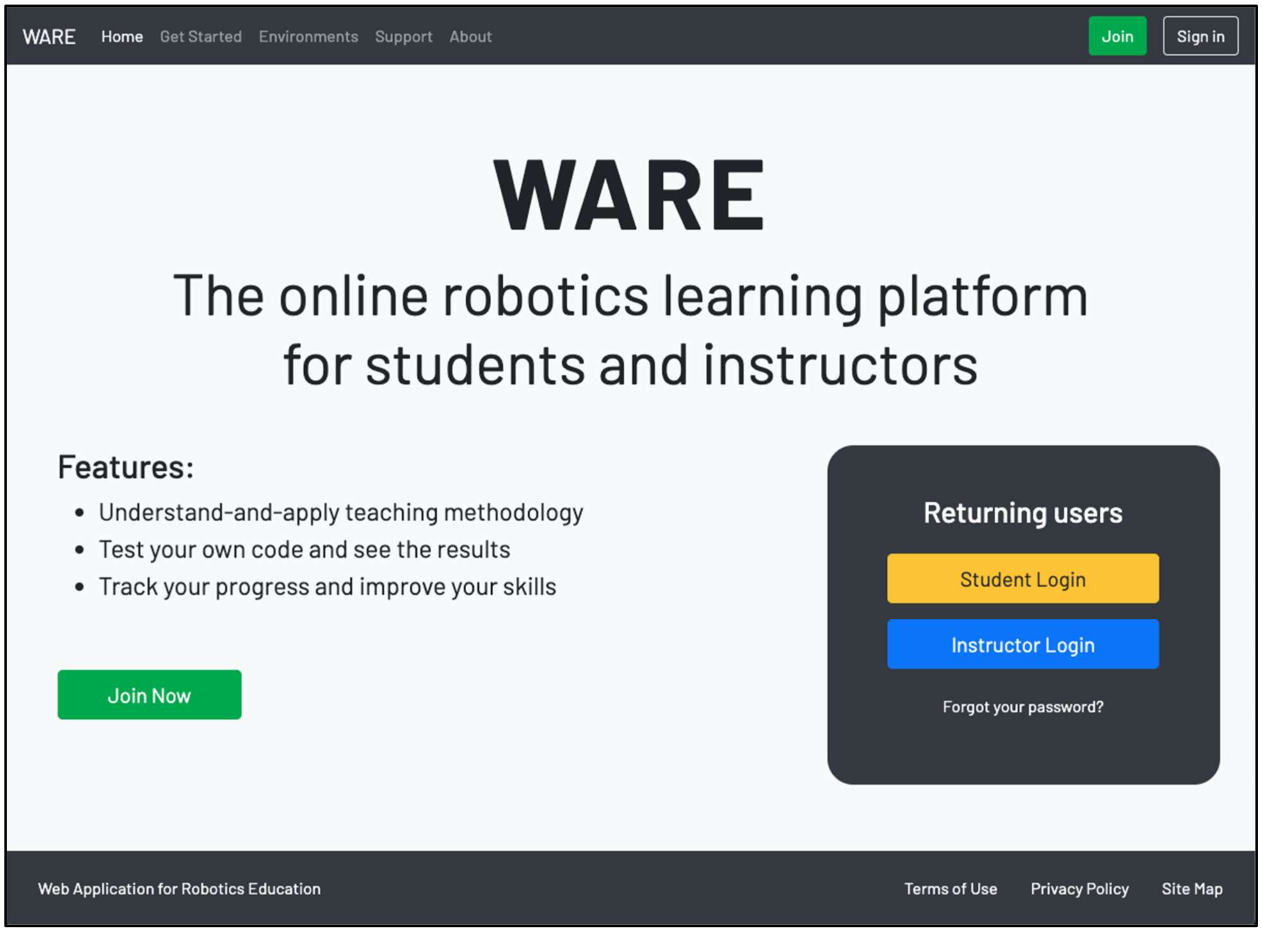


Figure 4.4: State Diagram for the environment selection process

5. User Interface Design

5.1 Homepage

The desktop homepage is presented to a user who is currently not signed in. The homepage tells the user about WARE and gives a call-to-action button to become a member. Existing users have the option to login by clicking the respective button on the right hand side. The mobile view of the homepage features a collapsed navigation bar that is accessible by the menu button on the right hand side. The part of the navigation bar that is displayed is fixed and stays in place on scroll.



WARE

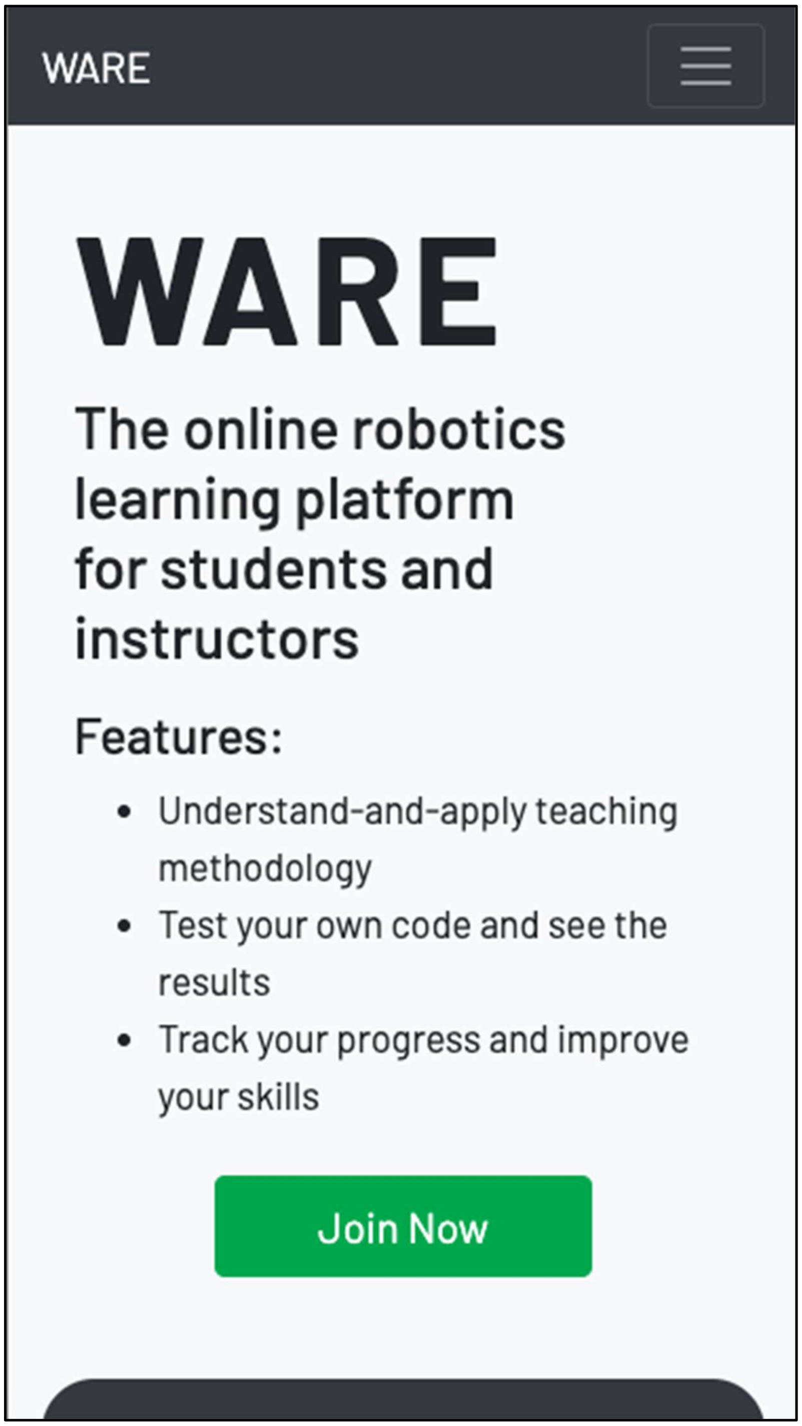
The online robotics learning platform for students and instructors

Features:

- Understand-and-apply teaching methodology
- Test your own code and see the results
- Track your progress and improve your skills

Web Application for Robotics Education

Figure 5.1.1: The desktop view of the WARE homepage.



WARE

The online robotics learning platform for students and instructors

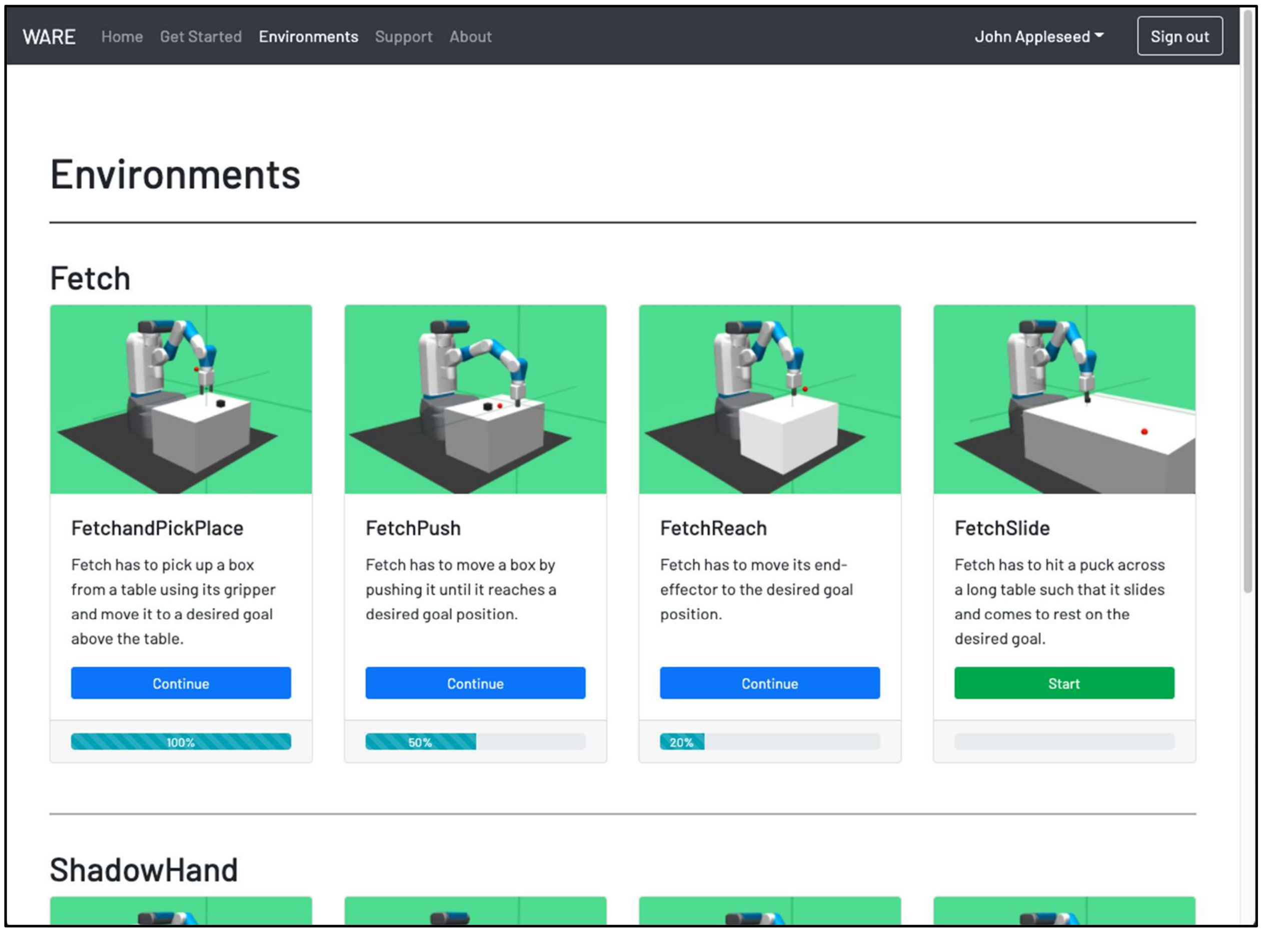
Features:

- Understand-and-apply teaching methodology
- Test your own code and see the results
- Track your progress and improve your skills

Figure 5.1.2: The mobile view of the homepage.

5.2 Environments Page

The environments page is accessible only to registered users. The user's name is displayed on the right hand side of the navbar. Each environment available indicates to the user the amount of progress made by the user. The mobile version presents the environment cards vertically aligned and stacked on top each other. The remaining cards on the mobile view are accessed by scrolling down on the page.



Environments

Fetch

FetchandPickPlace

Fetch has to pick up a box from a table using its gripper and move it to a desired goal above the table.

Continue

100%

FetchPush

Fetch has to move a box by pushing it until it reaches a desired goal position.

Continue

50%

FetchReach

Fetch has to move its end-effector to the desired goal position.

Continue

20%

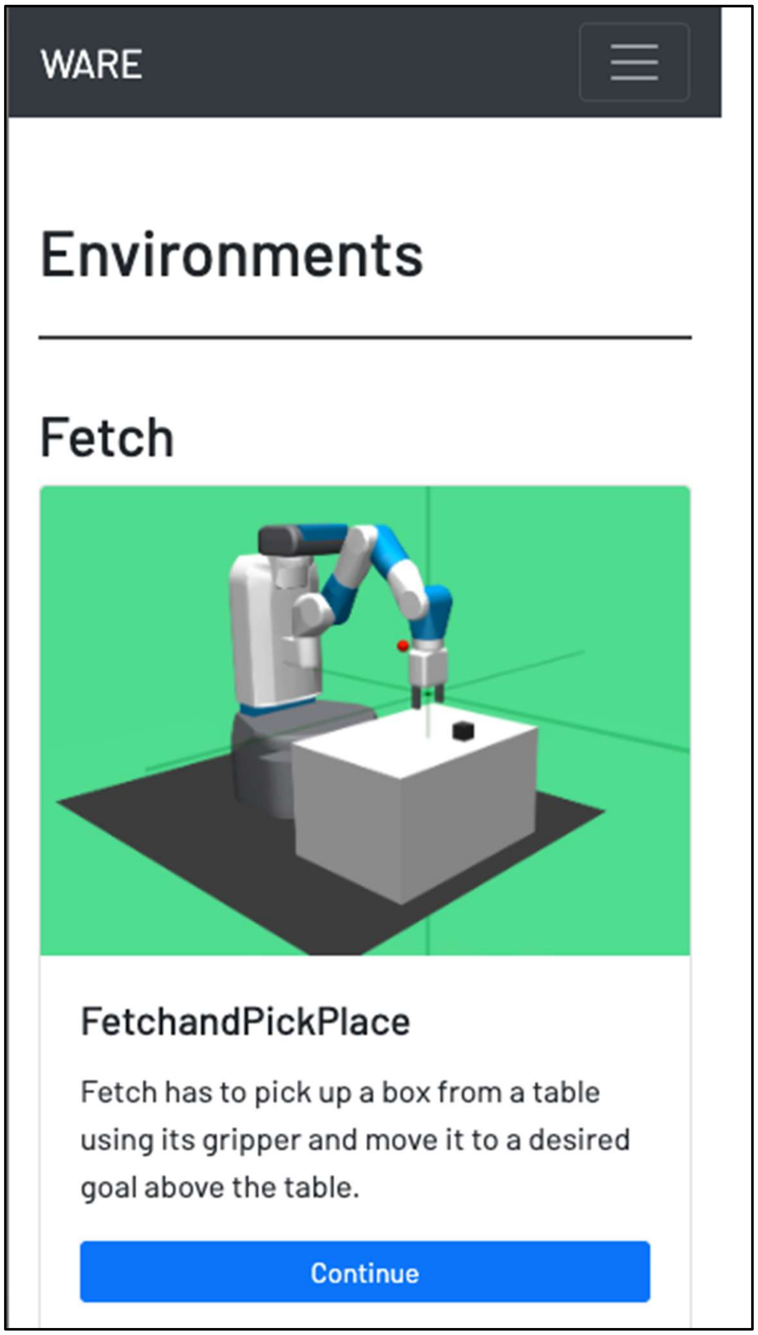
FetchSlide

Fetch has to hit a puck across a long table such that it slides and comes to rest on the desired goal.

Start

ShadowHand

Figure 5.2.1: The desktop version of the environments page.



Environments

Fetch

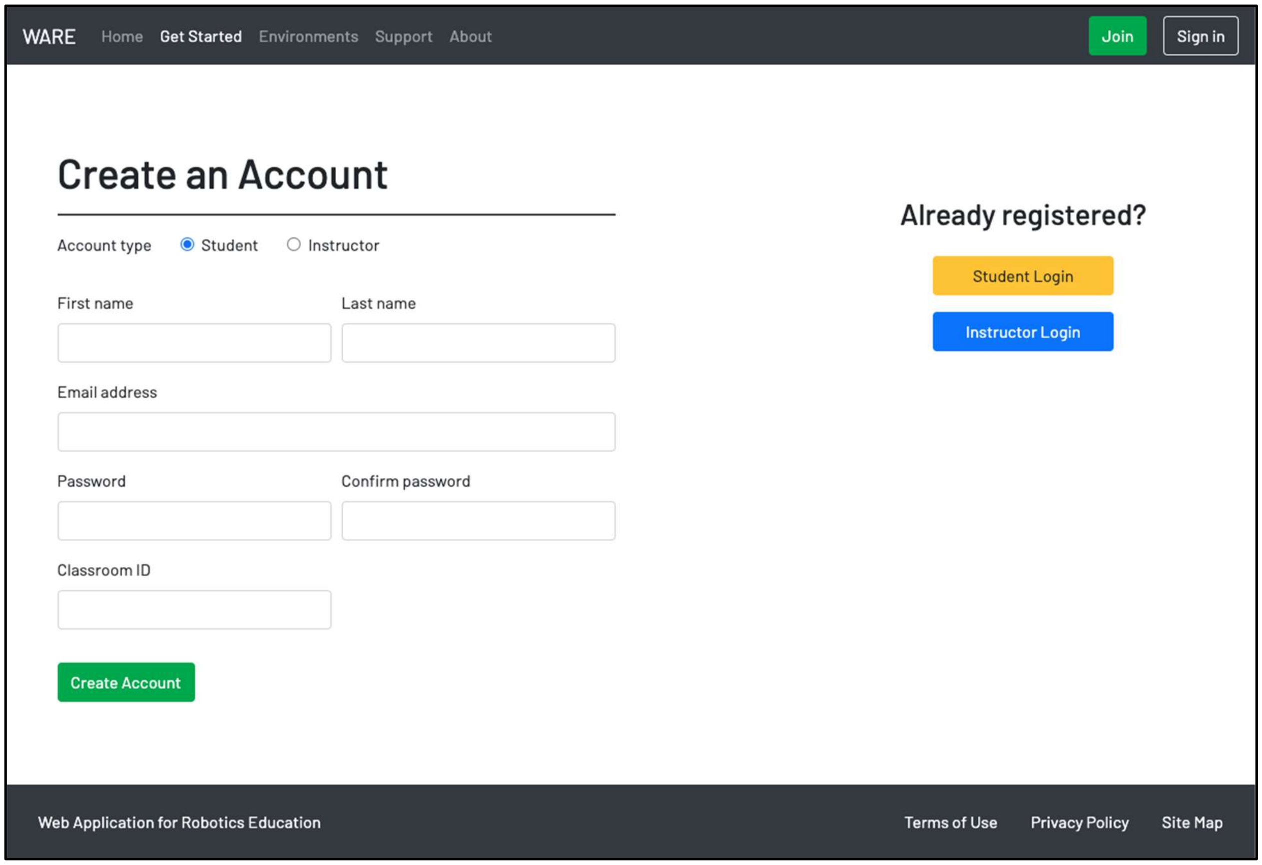
FetchandPickPlace

Fetch has to pick up a box from a table using its gripper and move it to a desired goal above the table.

Figure 5.2.2: The mobile view of the environments page.

5.3 Account Creation Page

The account creation page presents a form to the user to fill out to create an account. Options are given as to the type of account that will be created. If a Student type is chosen, then a Classroom ID field will be presented. The form will be validated before the contents are sent to the back-end server. If there are errors in the form data, appropriate feedback will be given to the user to assist in fixing these errors before retrying the submission.



WARE Home Get Started Environments Support About Join Sign in

Create an Account

Account type [✓ Student] Instructor

First name

Last name

Email address

Password

Confirm password

Classroom ID

Create Account

Already registered?

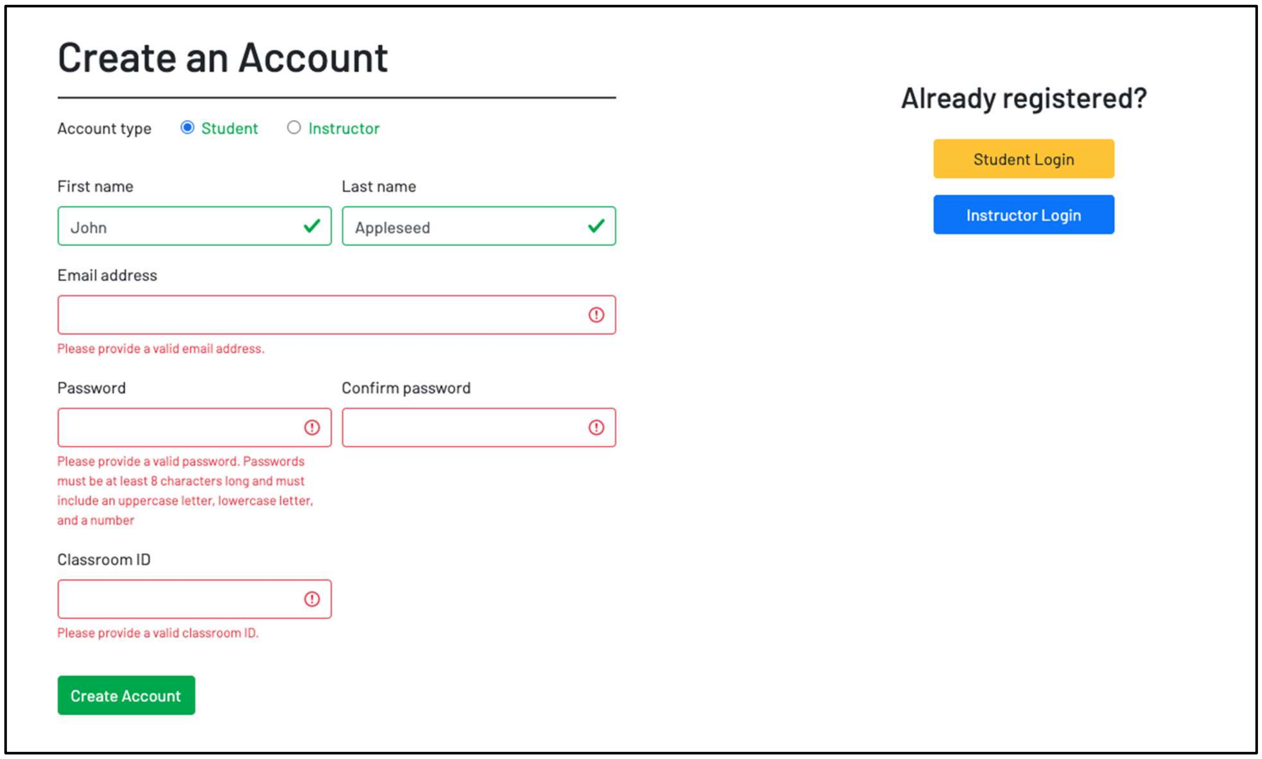
Student Login

Instructor Login

Web Application for Robotics Education

Terms of Use Privacy Policy Site Map

Figure 5.3.1: The desktop view of the account setup page.


Create an Account

Account type [✓ Student] Instructor

First name: John ✓

Last name: Appleseed ✓

Email address

Please provide a valid email address.

Password

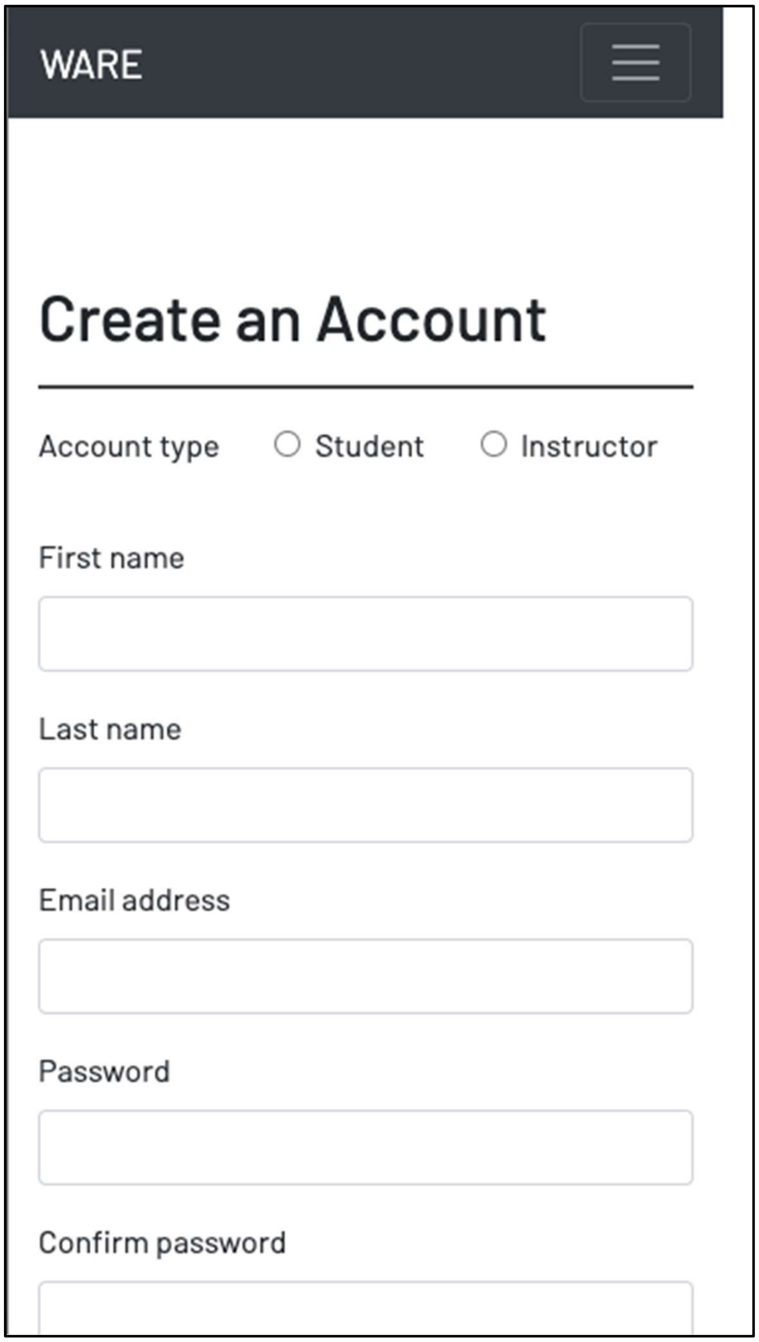
Confirm password

Please provide a valid password. Passwords must be at least 8 characters long and must include an uppercase letter, lowercase letter, and a number

Classroom ID

Please provide a valid classroom ID.

Figure 5.3.2: Form validation for creating an account.


WARE

Create an Account

Account type ○ Student ○ Instructor

First name

Last name

Email address

Password

Confirm password

Figure 5.3.3: The mobile view of the account creation page.

5.4 User Login Pages

The user login page presents fields for the email address and password to sign in. The student login page and the instructor login page are similar in design. For both of these pages, an additional button is given below the login form to navigate to the other login page if the user accidentally navigated to the wrong one.

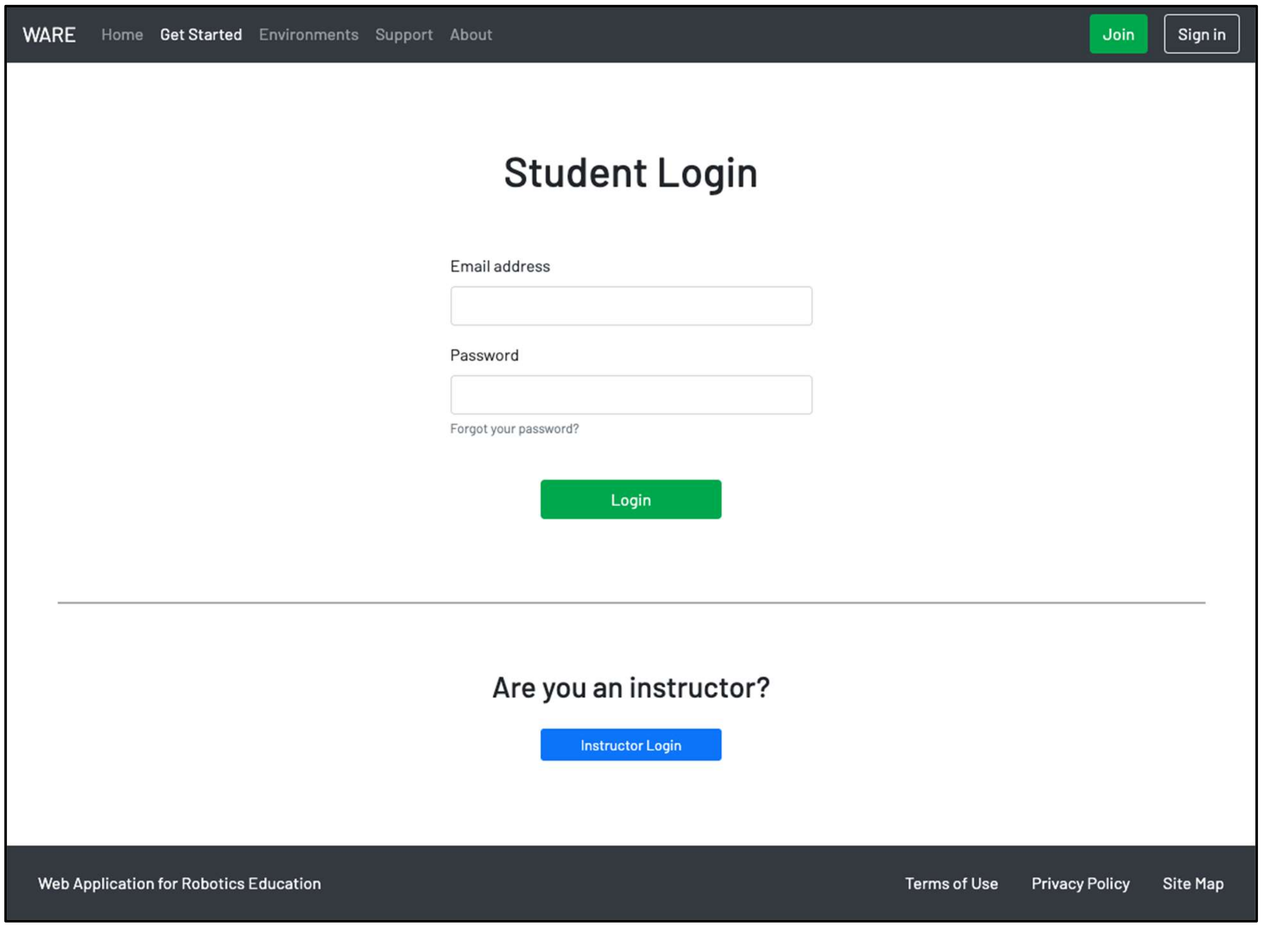


Figure 5.4.1: The desktop view of the student login page.

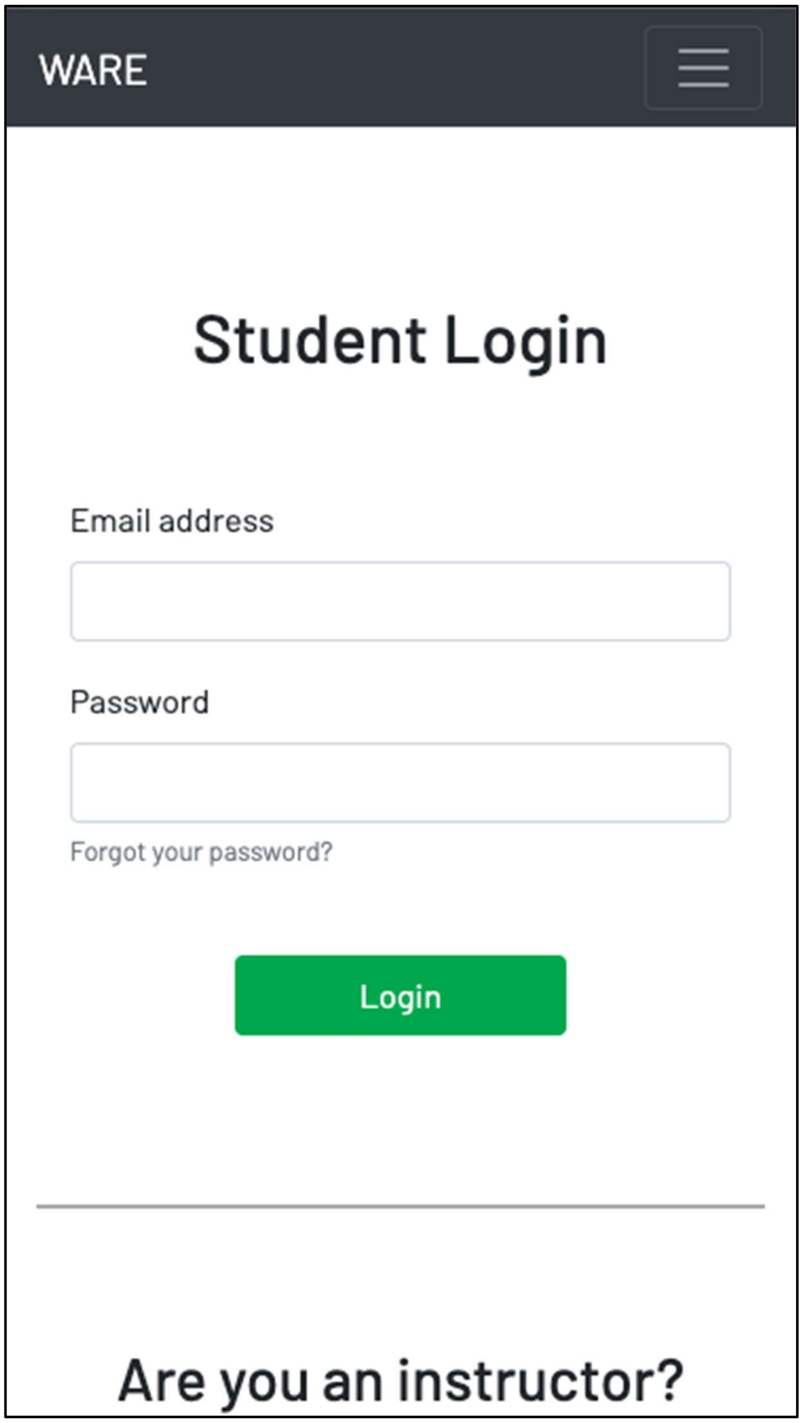
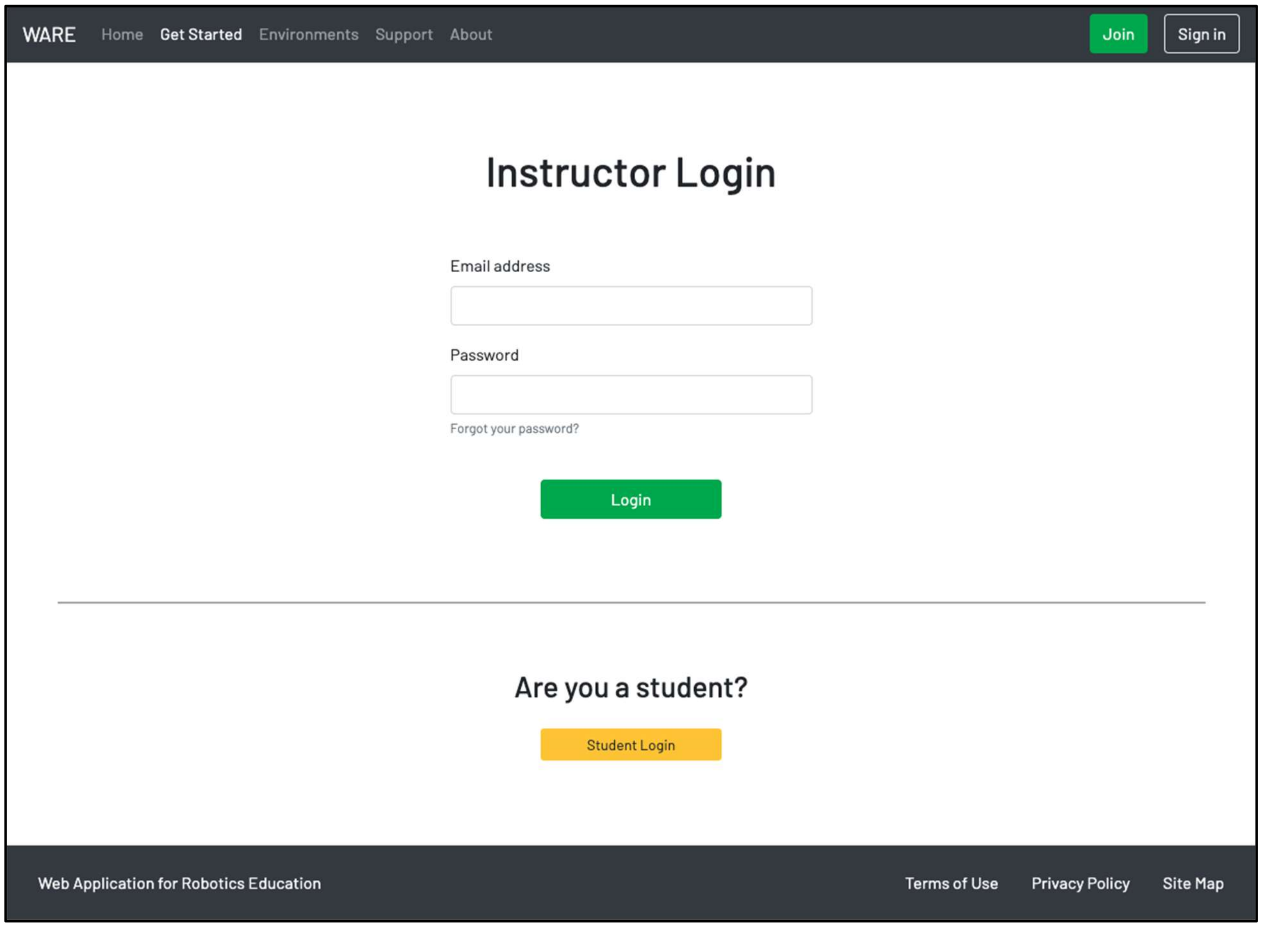


Figure 5.4.2: The mobile view of the student login page.



Instructor Login

Email address

Password

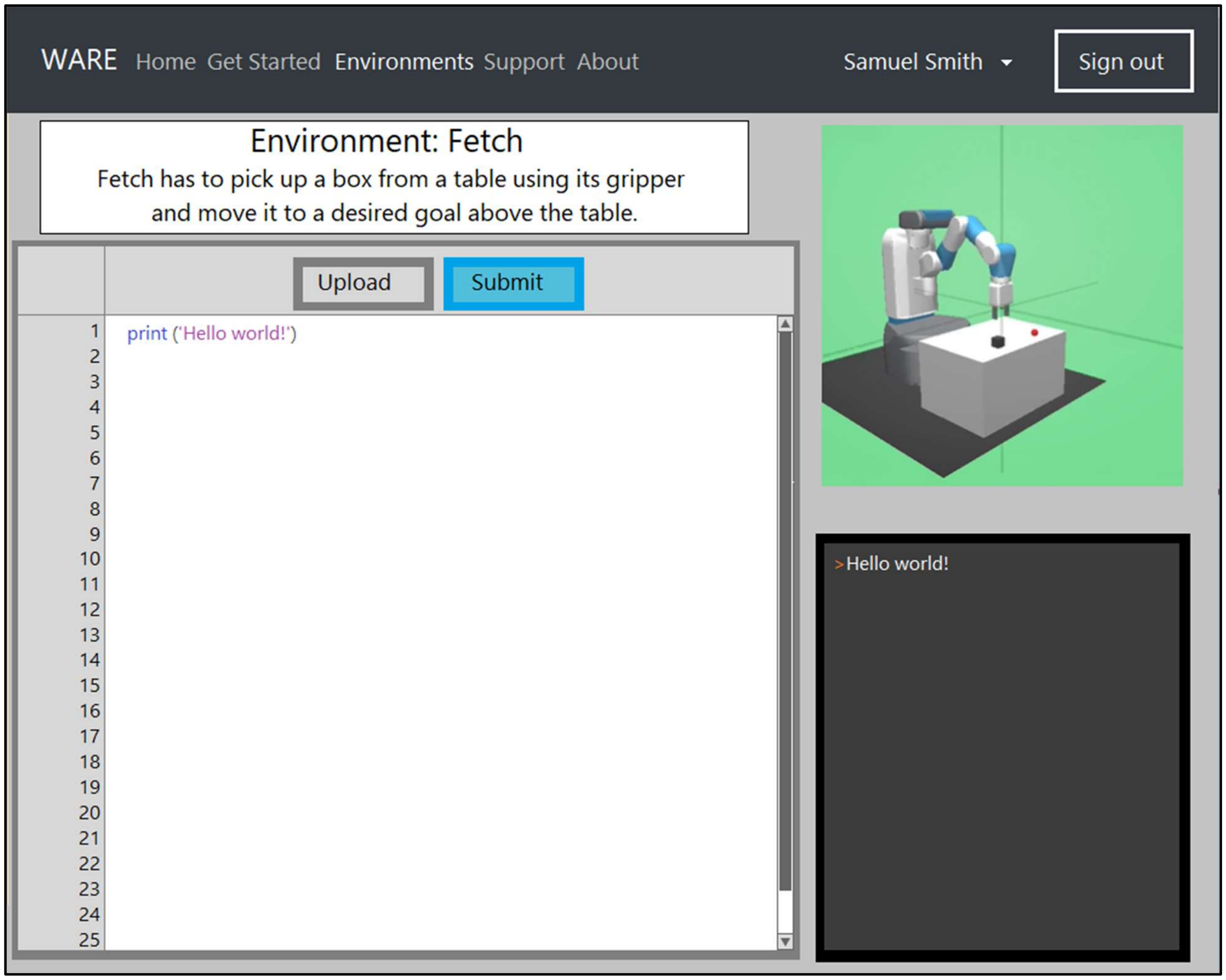
Forgot your password?

Are you a student?

Figure 5.4.3: The desktop view of the instructor login page.

5.5 Single Environment Page

The single environment page is where the user will spend the majority of their time on this website. This is where programming and learning takes place. The left side is where the user will code in Python with buttons to either upload the code to the web server for testing and results or to submit it to potentially complete the lab. The upper right is the visual output with regards to the robot. Lastly, the bottom right is for the textual output.



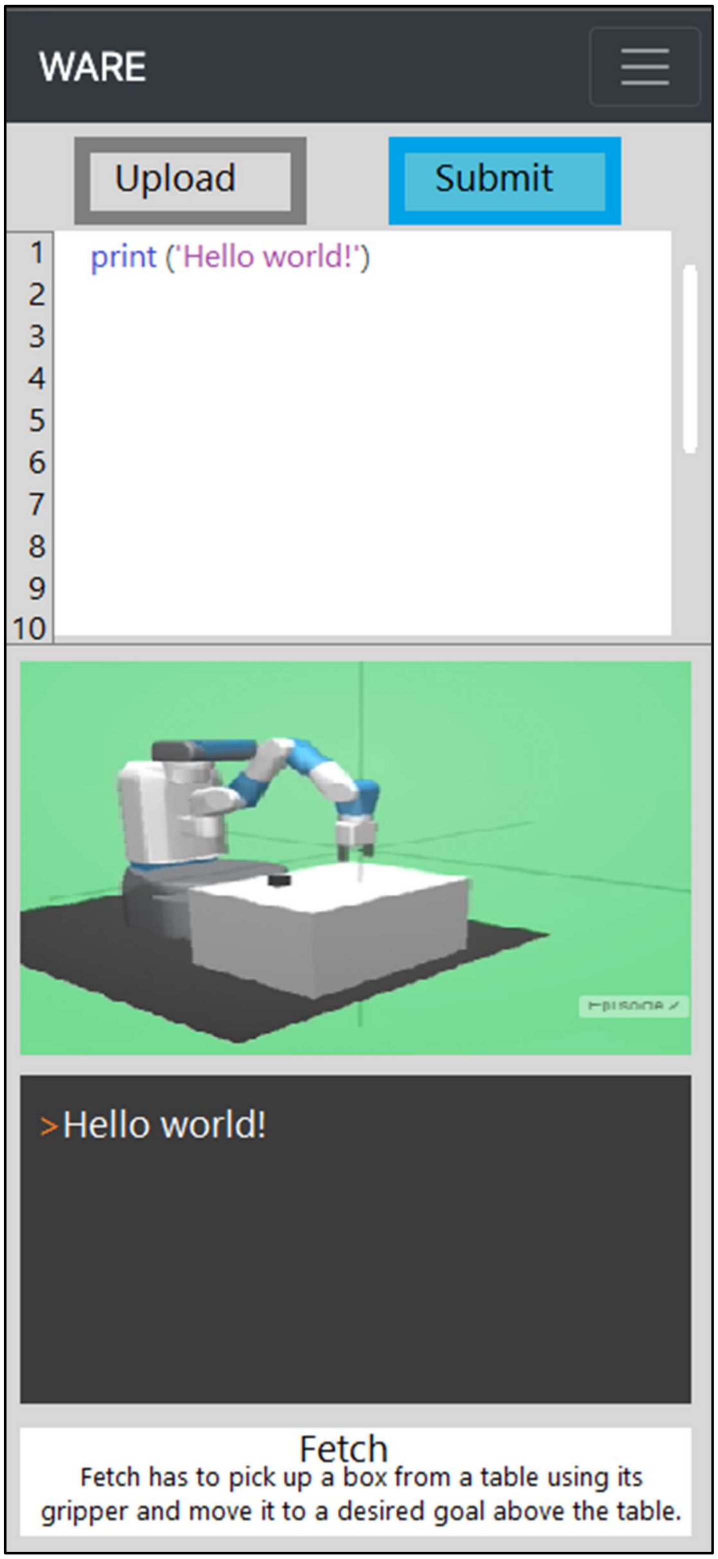
Environment: Fetch

Fetch has to pick up a box from a table using its gripper and move it to a desired goal above the table.

```
print ('Hello world!')
```

```
>Hello world!
```

Figure 5.5.1: The desktop view of the single environment page.



```
print ('Hello world!')
```

```
>Hello world!
```

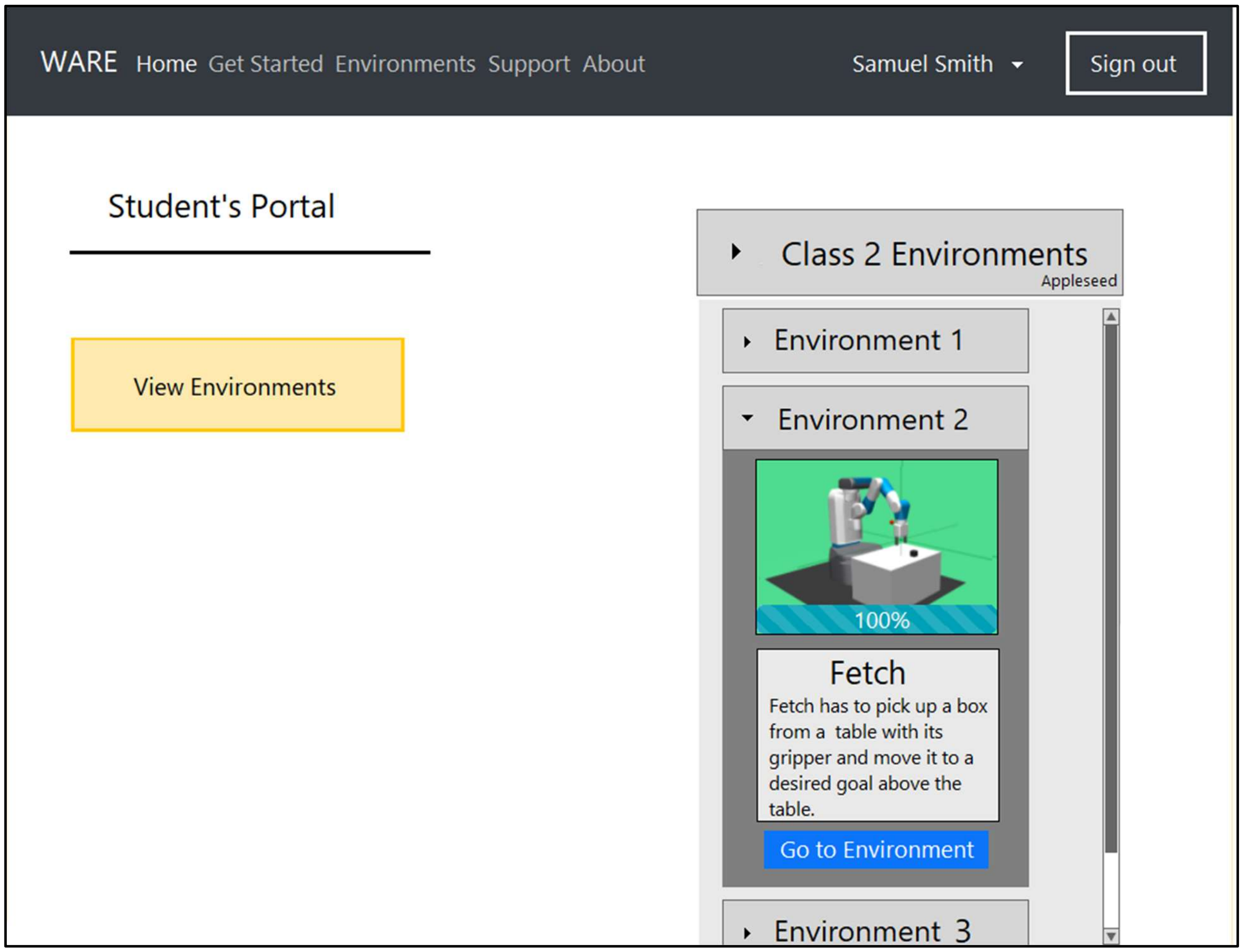
Fetch

Fetch has to pick up a box from a table using its gripper and move it to a desired goal above the table.

Figure 5.5.2: The mobile view of the single environment page. While a bit compact due to having to cram everything from the prior picture onto a small screen, the user will be able to receive the same experience and knowledge as if they were on a desktop.

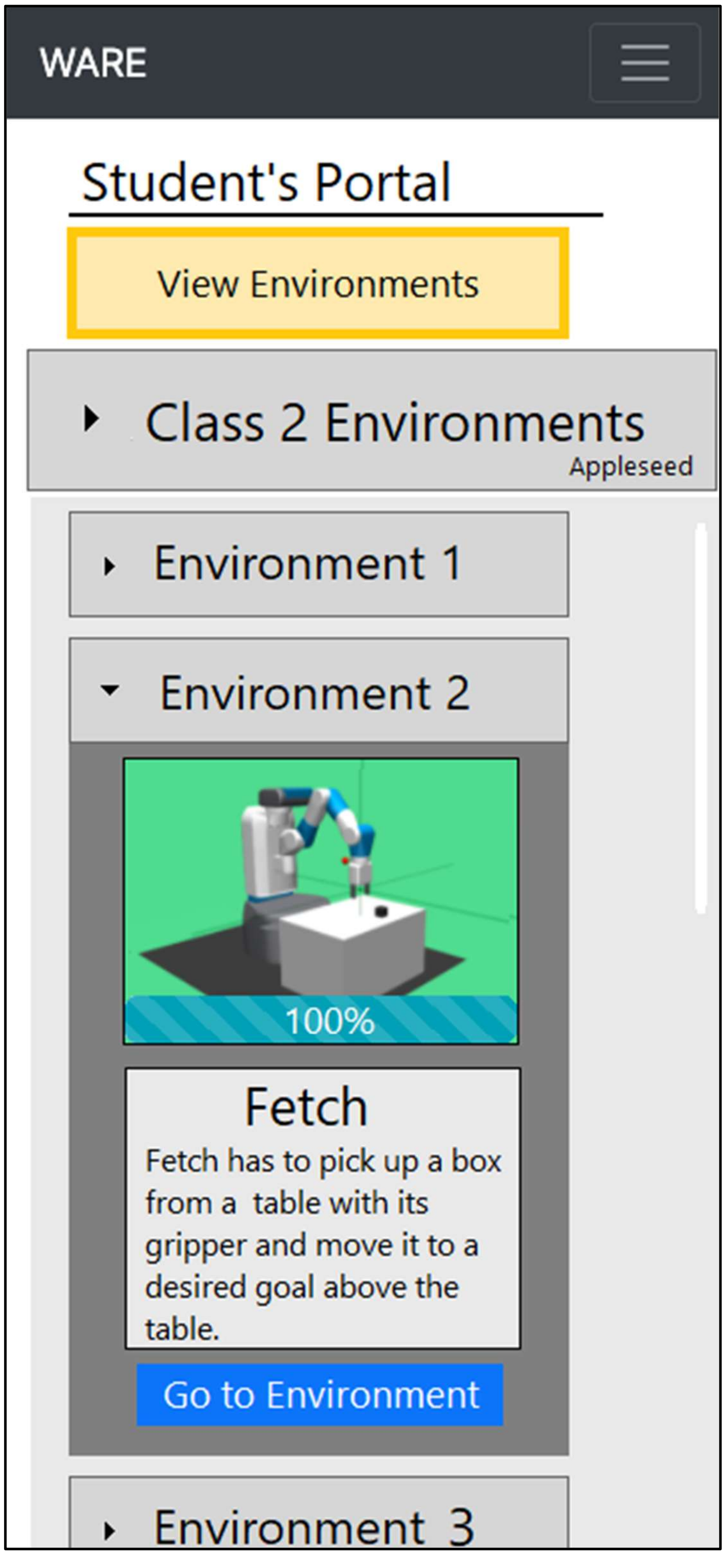
5.6 Student’s Portal Page

The student’s portal page is effectively the homepage of student-based accounts. Here students will have access to their class’s respective environments - as well as having the ability to view all environments we have to offer.



Fetch has to pick up a box from a table with its gripper and move it to a desired goal above the table.

Figure 5.6.1: The desktop view of the student’s portal page - displaying the class the user is registered to with its respective environments.



WARE

Student's Portal

View Environments

Class 2 Environments

Appleseed

Environment 1

Environment 2

100%

Fetch

Fetch has to pick up a box from a table with its gripper and move it to a desired goal above the table.

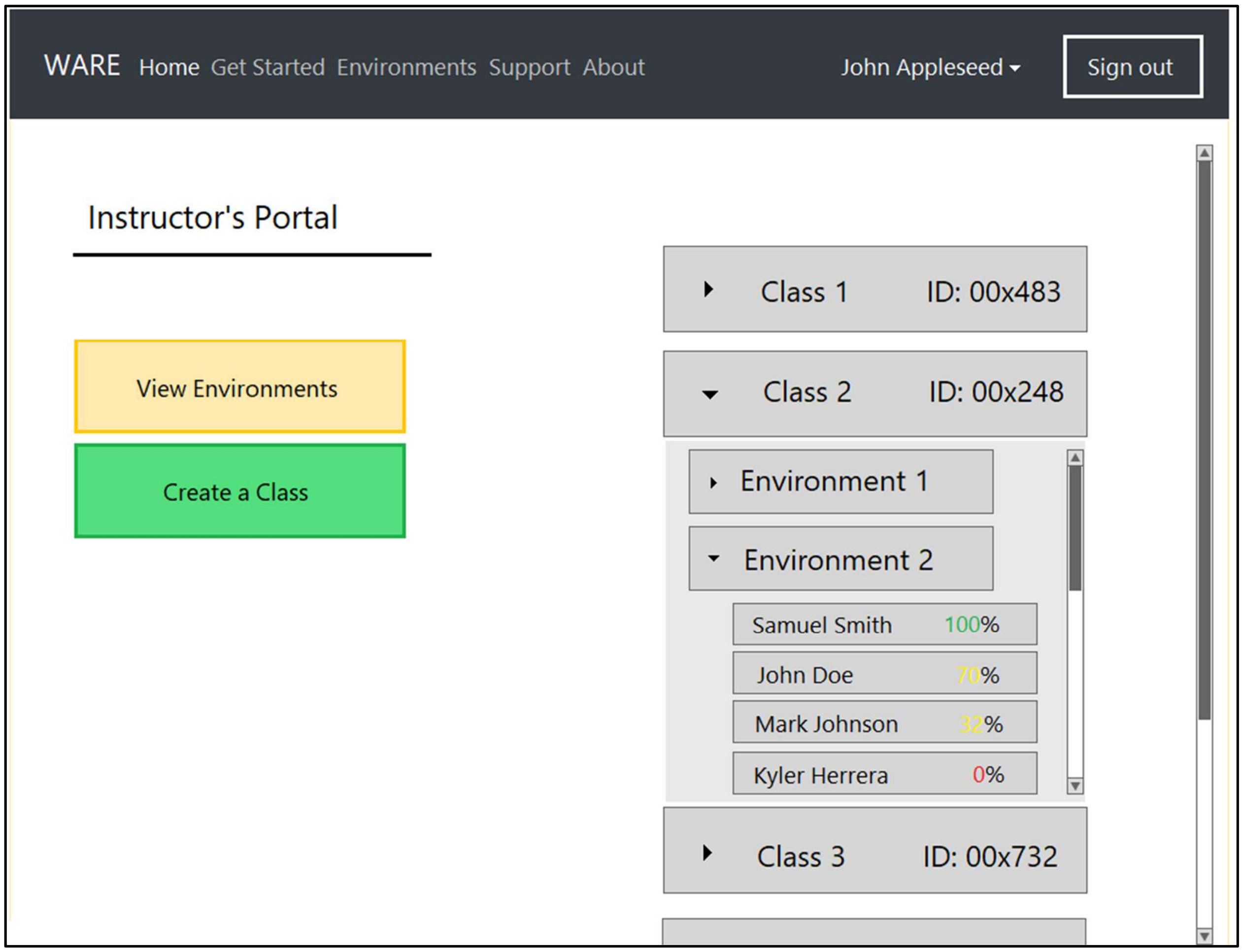
Go to Environment

Environment 3

Figure 5.6.2: The mobile view of the student’s portal page.

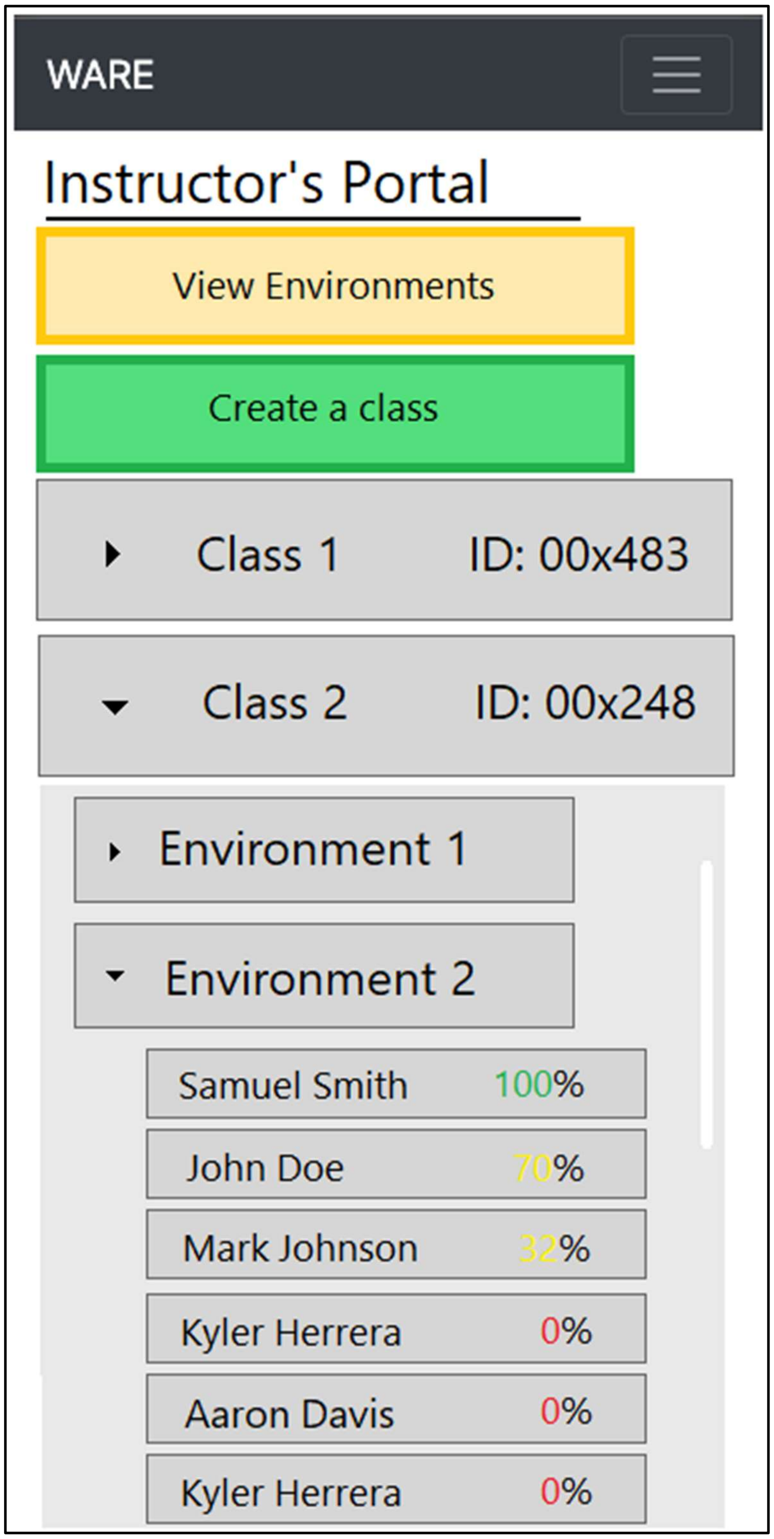
5.7 Instructor’s Portal Page

The instructor’s portal page is effectively the home page of the instructor. Here they will be able to create classes for students to join, select which environments students will utilize, and also be able to see student’s progress throughout each environment.



Samuel Smith	100%
John Doe	70%
Mark Johnson	32%
Kyler Herrera	0%

Figure 5.7.1: The desktop view of the instructor’s portal. Displays all classes created with their respective environments as well as letting the instructor view progress of his students for each environment.



WARE

Instructor's Portal

View Environments

Create a class

Class 1 ID: 00x483

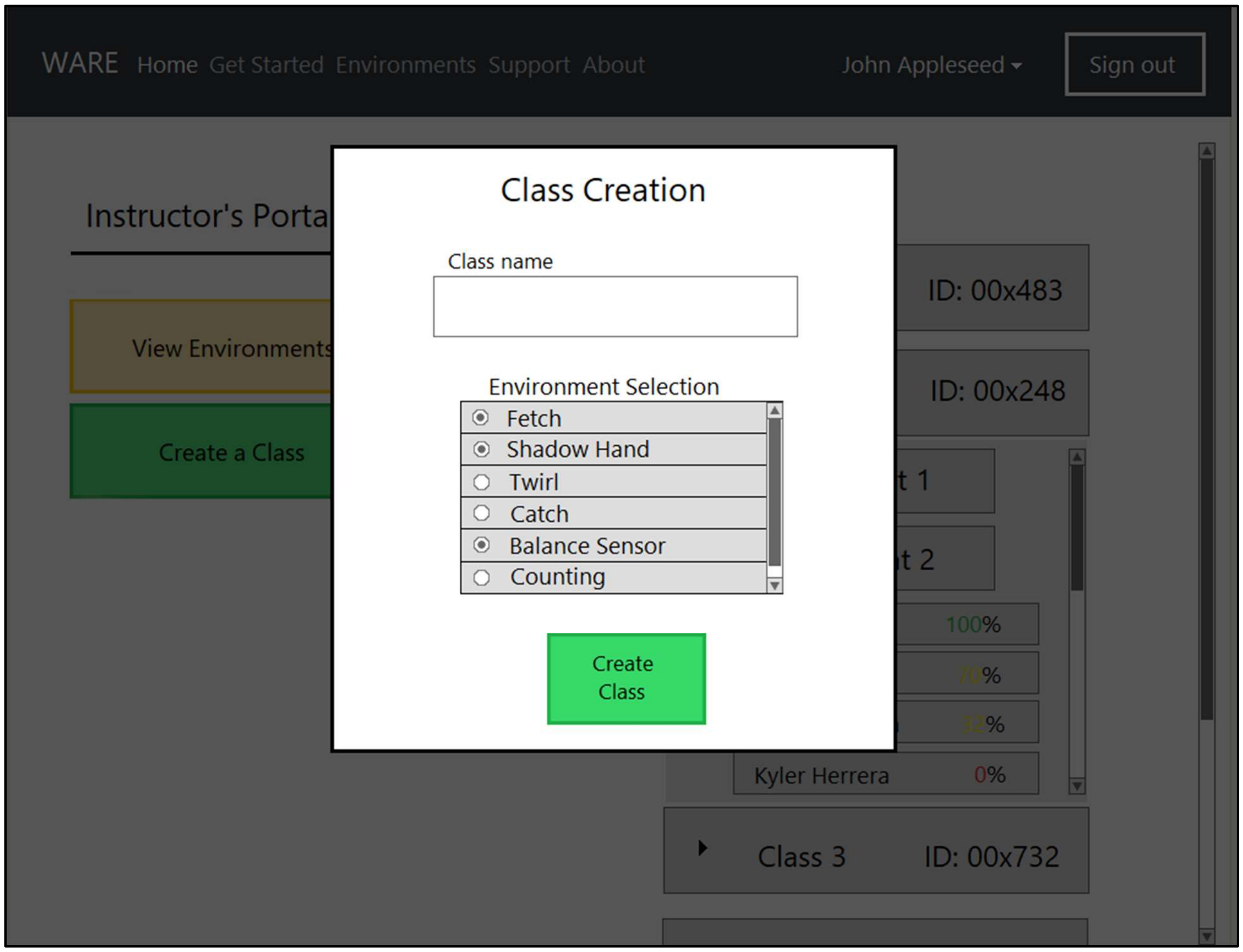
Class 2 ID: 00x248

Environment 1

Environment 2

Samuel Smith	100%
John Doe	70%
Mark Johnson	32%
Kyler Herrera	0%
Aaron Davis	0%
Kyler Herrera	0%

Figure 5.7.2: The mobile view of the instructor's portal.



Class Creation

Class name

Environment Selection
- [✓ Fetch]
- [✓ Shadow Hand]
- Twirl
- Catch
- [✓ Balance Sensor]
- Counting

Create Class

Figure 5.7.3: The desktop view of the class creation form. The instructor can select a class name and choose which environments will be related to the class.

6. Contributions of Team Members

The team's contributions are summarized below. The name of each team member, the time each member allocated for this assignment, and the tasks that were completed are given.

Sean Griffith
Total time commitment: 14 hours
Created the table of contents, designed the program unit structure diagram, created most of the unit tables, generated the data structures and their respective table templates, and designed the diagrams for the user login and code submission processes.

Zachery Wiles
Total time commitment: 10.5 hours
Drafted the abstract section, designed the system-level diagram; created the homepage, environments, account creation, and user login webpage design mockups. Contributed to the whole design of the document.

Ryan Lunt
Total time commitment: 9.5 hours
Helped create some of the unit tables, created the single environment page, instructor portal, student portal, and class creation webpage design mockups.

Herman Hira
Total time commitment: 4 hours
Drafted the introduction section, designed the instructor view and environment selection diagrams