WARE

Web Application for Robotics Education

University of Nevada, Reno

Department of Computer Science and Engineering

Revised Specification and Design

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

February 26, 2021

Table of Contents

1. Abstract

WARE is a website dedicated to providing users with an interactive way to use the OpenAI Gym robotics framework. The OpenAI Gym framework is an open-source toolkit used for developing reinforcement learning algorithms. WARE combines the framework's robotics environments with a web interface, allowing users to experiment with the framework while avoiding the time and complexity of installing and running it on a local machine.

WARE provides eight different robotics environments for users to interact with. Users develop their code within the environment and the back-end server processes it and outputs the results. WARE also offers user account management, allowing students and instructors to create accounts that integrate together to form classrooms.

2. Recent Project Changes

The only change that has occurred in the project since the last project assignment is the addition of requiring users to enter a first and last name in the account signup process. These two fields will be stored as part of each user in the user database. These fields were added to increase personalization of the website for the user and allow instructors to easily identify students in a WARE classroom

3. Updated Specification

3.1 Summary of Changes in Project Specification

Since the submission of the original project specification document, our project's requirements have remained largely unchanged. However, as development of each of the project components has progressed, the difficulty of implementing various features has been reassessed and adjustments to the specification have been made.

Although functional requirements for the WARE project have remained largely unchanged, four functional requirements listed in Table 3.2.1.1 were added to the project: FR21, FR22, FR23, and FR30. Functional requirements 22 and 23 were added to ensure class management functionality, while functional requirements 21 and 30 were added as a result of our project demonstration to the instructor and our project sponsor last semester. Additionally, an original functional requirement, FR19 (WARE shall check user input for syntax errors), was removed as this requirement was already being met with FR8.

Another change made to the WARE functional requirements was the reordering of requirements between priority levels. Functional requirements 16, 17, and 18 were moved from priority level 2 to priority level 1, functional requirement 24 was moved from priority level 1 to priority level 2, and functional requirements 26 and 27 were moved from priority level 2 to priority level 3.

Additionally, as our team continues to refine our understanding of the project's requirements, we have introduced two new use cases: UC05 and UC06, which are described in Table 3.3.2.1. These use cases account for the implementation of classes within the WARE project.

Finally, our team has revised the requirement traceability matrix designed in the original specification document to account for the aforementioned changes to functional requirements as well as the addition of use cases 5 and 6. The updated requirement traceability matrix can be seen in Figure 3.3.3.1.

3.2 Updated Technical Requirements Specification

The technical requirements shown below in Tables 3.2.1.1 and 3.2.2.1 describe the updated functional and non-functional requirements of the WARE project. Each requirement provided below is tagged with a priority level as follows:

- [1] - Planned for implementation by the end of this semester (Spring 2021).
- [2] - Planned for possible implementation by the end of this semester.
- [3] - Implementation is unlikely to occur this semester.

3.2.1 Functional Requirements

FR1.	[1]	WARE shall allow the user to create an account.
FR2.	[1]	WARE shall allow the user to log into their account.
FR3.	[1]	WARE shall allow the user to log out of their account.
FR4.	[1]	WARE shall allow the user to submit their code to the web server for processing.
FR5.	[1]	WARE shall process code submitted by a user to produce an output.
FR6.	[1]	WARE shall display to the user textual results of code submitted to the web server.
FR7.	[1]	WARE shall display to the user visual results of code submitted to the web server.
FR8.	[1]	WARE shall display to the user errors that result from code submitted to the web server.
FR9.	[1]	WARE shall track user progress in each robotics environment.

FR10.	[1]	WARE shall allow the user to view their progress in each robotics environment.
FR11.	[1]	WARE shall allow the user to select a robotics environment.
FR12.	[1]	WARE shall display a brief description of each robotics environment to the user.
FR13.	[1]	WARE shall display examples of working input for each robotics environment to the user.
FR14.	[1]	WARE shall display to the user a visual preview for each robotics environment.
FR15.	[1]	WARE shall allow the user to input their code through the use of a text box.
FR16.	[1]	WARE shall log user sessions and associated interactions over time.
FR17.	[1]	WARE shall use python syntax highlighting for code written within the textbox on the website.
FR18.	[1]	WARE shall save a user's most recently submitted code for future use.
FR19.	[1]	WARE shall allow the user to view environments which they have made progress in from their homepage.
FR20.	[1]	WARE shall allow the user to input their code through the use of a file upload.
FR21.	[1]	WARE shall display to the user a loading icon for the time between when a user submits their solution and when that user receives the results of their code.
FR22.	[2]	WARE shall allow instructors to create a class for their students to join.
FR23.	[2]	WARE shall allow students to join a class created by an instructor.
FR24.	[2]	WARE shall allow instructors to choose a list of environments that their students should have access to.
FR25.	[2]	WARE shall allow instructors to view and run their student's last code submission in an environment.
FR26.	[3]	WARE shall check user input for formatting mistakes.
FR27.	[3]	WARE shall allow users to make feature requests and report errors.
FR28.	[3]	WARE shall implement a representational state transfer application programming interface for advanced user interaction.
FR29.	[3]	WARE shall allow users to communicate with other students and their instructors within each robotics environment.
FR30.	[3]	WARE shall provide the user with hints on how to progress with their solution to a robotics environment following the submission of a failed solution.

Table 3.2.1.1: Updated Functional requirements for the Web Application for Robotics Education (WARE) project.

3.2.2 Non-Functional Requirements

NFR1.	[1]	WARE will utilize a database to store user information and robotics environments.
NFR2.	[1]	WARE will minimize CPU usage and perform intensive computations remotely.
NFR3.	[1]	WARE will provide an intuitive and elegant front-end website.
NFR4.	[1]	WARE will be compatible with Chromium-based web browsers.
NFR5.	[1]	WARE will utilize the Bootstrap, JavaScript, and Flask platforms.
NFR6.	[1]	WARE will be fully responsive to accommodate a wide range of devices.
NFR7.	[2]	WARE will utilize fully encrypted communications for each user session.
NFR8.	[2]	WARE will be compatible with Chromium-based and Firefox web browsers.
NFR9.	[3]	WARE will implement accessibility standards to accommodate screen readers and users with sight impairments.
NFR10.	[3]	WARE will be compatible with Chromium-based, Firefox, and Internet Explorer web browsers.

Table 3.2.2.1: Updated Non-Functional requirements for the Web Application for Robotics Education (WARE) project.

3.2.3 Requirements Pertaining to Hardware

Our project does not currently require the use of hardware for deployment, and as such, all functional and non-functional requirements for WARE pertain to its software implementation.

3.3 Updated Use Case Modeling

3.3.1 Use Case Diagram

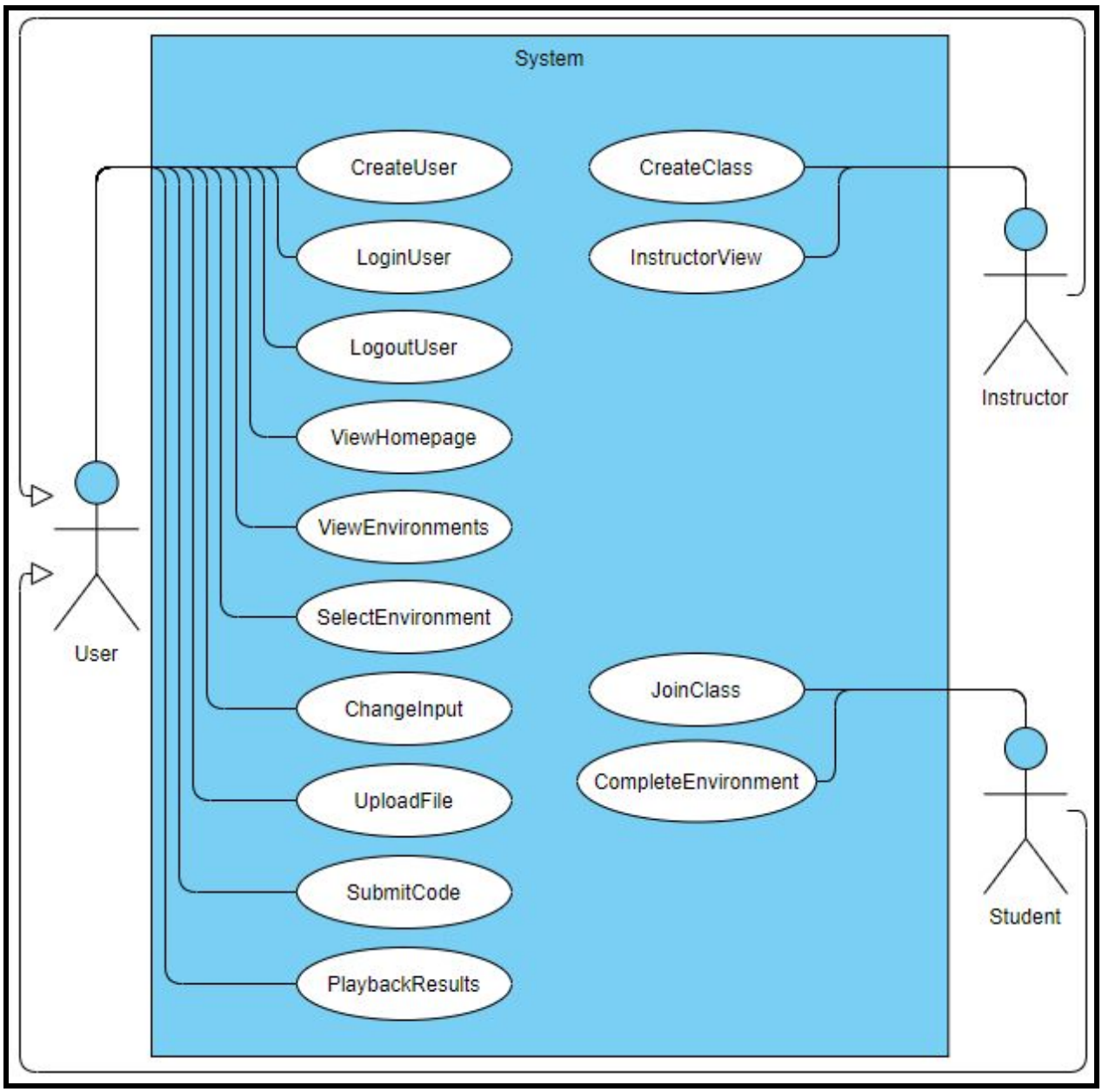

Figure 3.3.1.1. Updated Use Case Diagram showing the generalized relationship between the Instructor, Student, and User actors as well as their relationships with the various use cases.

3.3.2 Detailed Use Case Descriptions

ID	Use Case	Description
UC01	CreateUser	Upon accessing the website homepage, the user has the option to create a new account in order to utilize the website's robotics environment labs.
UC02	LoginUser	Upon accessing the website homepage, the user has the option to log in to an existing user account. This login queries the webserver for the validity of user credentials and either grants the user access or denies login.
UC03	LogoutUser	The user may log out of their account and return to the website's homepage at any time, using a "Logout" button.
UC04	ViewHomepage	Students will be able to navigate to their own homepage, which will include information on environments they have started experimenting with.
UC05	CreateClass	Upon accessing their homepage, instructors will have the option to create a class and choose a subset of available environments for their students to have access to and experiment with.
UC06	JoinClass	Students will be able to join a class created by an instructor and engage in the robotics environments assigned to them.
UC07	ViewEnvironments	The user is able to view a list of all available robotics environments and their progress in each of them. This list includes brief descriptions of each environment.
UC08	SelectEnvironment	Upon accessing the list of available environments, the user may select a specific environment, and the user is then redirected to the page for that environment.
UC09	CompleteEnvironment	Students will be able to mark a robotics environment as completed once experimentation is completed. This option will be available from both the user's homepage and the environment list for the user.
UC10	ChangeInput	Upon accessing a specific robotics environment, the user may change the provided sample code to fit their experimentation needs and develop a unique solution to the environment with which they are engaged.
UC11	UploadFile	Upon accessing a specific robotics environment, the user may utilize a Python file upload to overwrite the provided sample code with code contained in the uploaded file.

UC12	SubmitCode	Upon accessing a specific robotics environment, the user may submit code contained in the code input area to the server for processing.
UC13	InstructorView	Instructors are able to access any of their students' code and progress for environments which have been assigned, accessible from the instructor's homepage.
UC14	PlaybackResults	Upon receipt of results for code submitted by a user, the user is able to view the state of the robotics environment throughout the execution of their code.

Table 3.3.2.1: Detailed use case descriptions.

3.3.3 Updated Requirement Traceability Matrix

The Updated Requirement Traceability Matrix for the WARE project can be found in Figure 3.3.3.1. Each column of the Requirement Traceability Matrix represents a use case of the WARE project, for which details can be found in Table 3.3.2.1. Each row of the Requirement Traceability Matrix represents a functional requirement of the WARE project, as described in Table 3.2.1.1.

	UC01	UC02	UC03	UC04	UC05	UC06	UC07	UC08	UC09	UC10	UC11	UC12	UC13	UC14
FR1	■													
FR2		■												
FR3			■											
FR4												■	■	
FR5												■	■	
FR6														■
FR7														■
FR8														■
FR9									■			■		
FR10				■			■		■				■	
FR11				■				■						
FR12				■	■		■							
FR13								■		■				
FR14							■	■						
FR15										■				
FR16		■	■						■			■		
FR17										■	■		■	
FR18						■						■	■	
FR19				■									■	
FR20											■			
FR21												■		
FR22					■									
FR23	■					■								
FR24					■									
FR25												■	■	
FR26										■	■			
FR27							■						■	
FR28														■
FR29								■					■	
FR30												■		

Figure 3.3.3.1: Updated Requirement Traceability Matrix displaying the relationships between use cases and functional requirements.

4. Updated Design

4.1 Summary of Changes in Project Design

The project design has not been significantly altered since the original specification document. Many of the program units and diagrams are still relevant to the current build of the project. Since these elements are integral to the design of the project, they have stayed consistent.

As for the frontend, the style and view was updated. The website now looks more modern with a white and blue color palette for simplicity. The environment page was also updated to the standard of more modern online code editors, and with a calmer color palette. These changes have helped make the website look more professional and fresh.

4.2 Updated High-level and Medium-level Design

Our prior system-level diagram has managed to withstand the test of time and stay up to date. This is because small changes haven't impacted the fundamental sections of our website. As a web application consisting of a front end, back end, and a database, it is best to view WARE as a layered architecture. WARE's core development and planning has been held relatively steady without major changes since the last semester. Any items that were scrapped or picked up have been minor over the last few months. Hence, most of the program units are the same from our prior design document. The one that has been updated is the 'UpdateProgress' program unit. Additionally, a program unit that was added was the 'UploadFile' program unit. The current up to date system level diagram and program units are shown below in their respective sections.

4.2.1 System-level Diagram

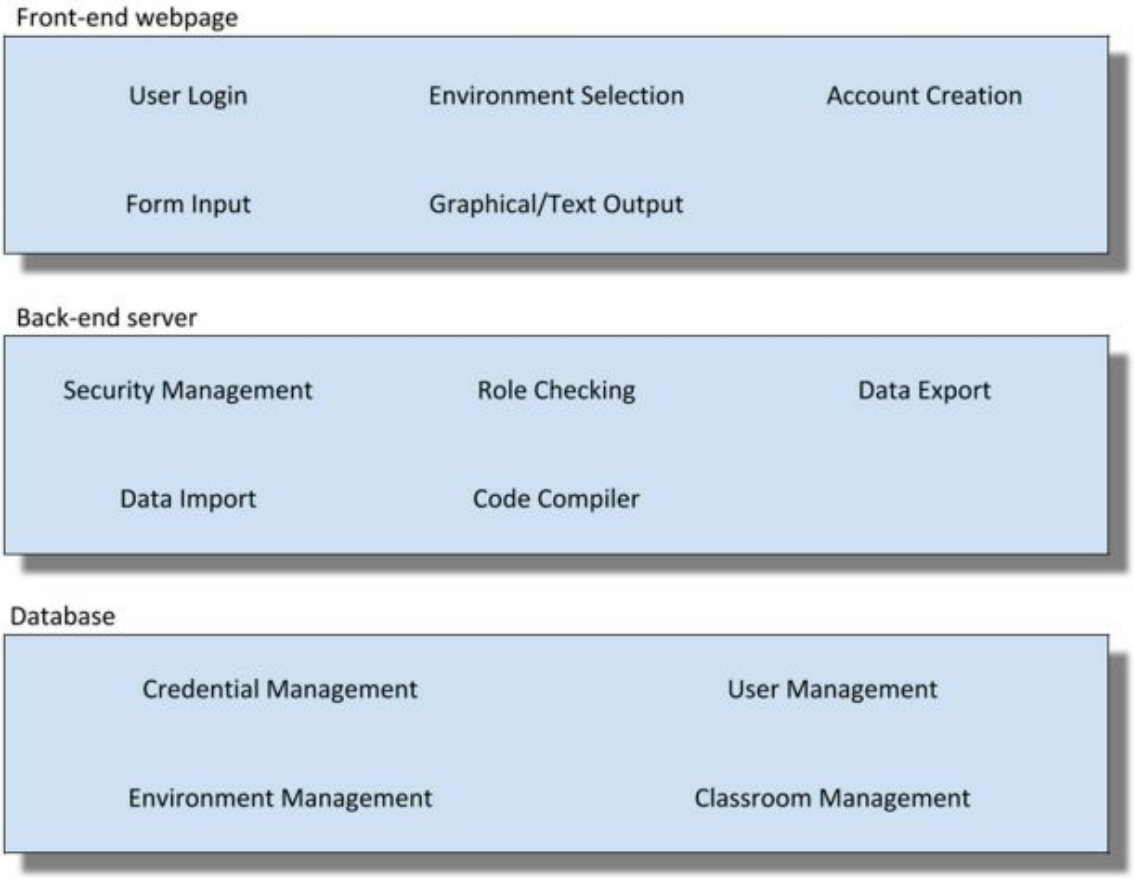

Figure 4.2.1.1: The layered architecture is broken up into three separate layers.

4.2.2 Program Units

	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	s 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	One of three states - Not started, in progress, and completed.
Unit Calls	None
Exceptions	Failure to update progress will halt the unit and notify the user.
Comments	Updates the progress status 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.

	Unit Name: UploadFile
Description	This function will take in a Python File and will output the contents of the file and place it into the text editor.
Subsystem	Back end server
Input	HTTP POST Request
Output	HTTP response containing code in the text editor.
Unit Calls	None
Exceptions	Failure to read file data of any non .py file.
Comments	This can only read in file data for python files (.py).

4.2.3 Data Structures

User Account data will be tracked using a database table with the schema shown in Table 4.2.3.1. New entries are added to this table whenever a user registers for an account, and these details will be used to keep track of the user and their progress in the various environments offered by the WARE service. Each user will have an identifier generated for their account to ensure uniquity and allow for continuous data tracking.

This identifier will be used as the primary key in the user database table. In addition to this primary key, each user will have an associated class_id which references the class to which that user is engaged in, and acts as a foreign key.

User Database Table Schema							
Value	id	class_id	first_name	last_name	password	user_type	email
Type	<int>	<int>	<string>	<string>	<string>	<bool>	<string>

Table 4.2.3.1: Database table template including attributes and their respective types for user information.

Information pertaining to classes will be tracked using a database table with the schema shown in Table 4.2.3.2. New entries will be added to this table whenever an instructor creates a new class from their homepage, from which point students may join the class to engage in the environments selected by the instructor for practice. Each class will have a unique identifier generated for it that will act as a primary key and allow students to join the class.

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

Table 4.2.3.2: Database table template including attributes and their respective types for class information.

Information specific to each environment will be tracked using a database table with the schema shown in Table 4.2.3.3. This database table will contain the name of the environment, its description, and the filename of the video used as a preview for the environment. This database table is planned to be static in nature, such that only WARE developers may add new entries, and each environment is assigned a unique identifier upon creation that will act as the primary key.

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

Table 4.2.3.3: Database table template including attributes and their respective types for environment information.

User progress data will be tracked using a database table with the schema shown in Table 4.2.3.4. This database table is updated whenever the user submits code to an environment which they have been assigned, with progress being tracked in three stages: not started, started, and complete. Additionally, the progress entry database table will contain the filepath of the associated user's most recent submission for an environment, such that it can be retrieved and reviewed by the user or their instructor later on. Within this table, each entry is associated with a unique identifier, which acts as a primary key. There are two foreign keys, user_id and environment_id, that reference the user and the environment associated with this progress entry.

Progress Entry Database Table Schema						
Value	id	user_id	envrironment_id	progress	submission_path	submission_date
Type	<int>	<int>	<int>	<float>	<filepath>	<datetime>

Table 4.2.3.4: Database table template including attributes and their respective types for user progress data.

4.3 Updated Hardware Design

Our project is fully software based and does not require any hardware.

4.4 Updated User Interface Design

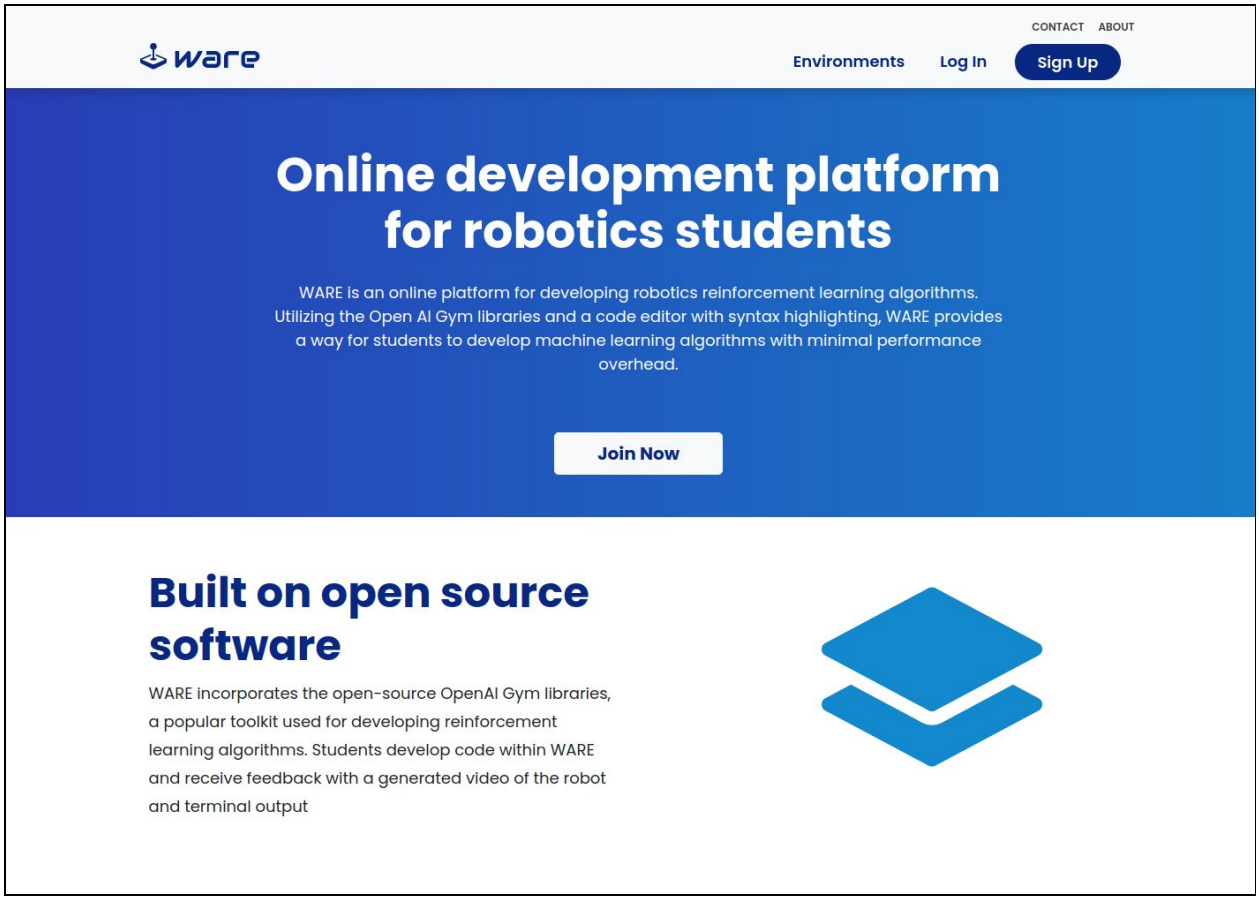

Figure 4.4.1: The WARE homepage as viewed from the Google Chrome web browser. The home page gives general information about WARE. The masthead displays a call-to-action button to create a user account and begin using WARE.

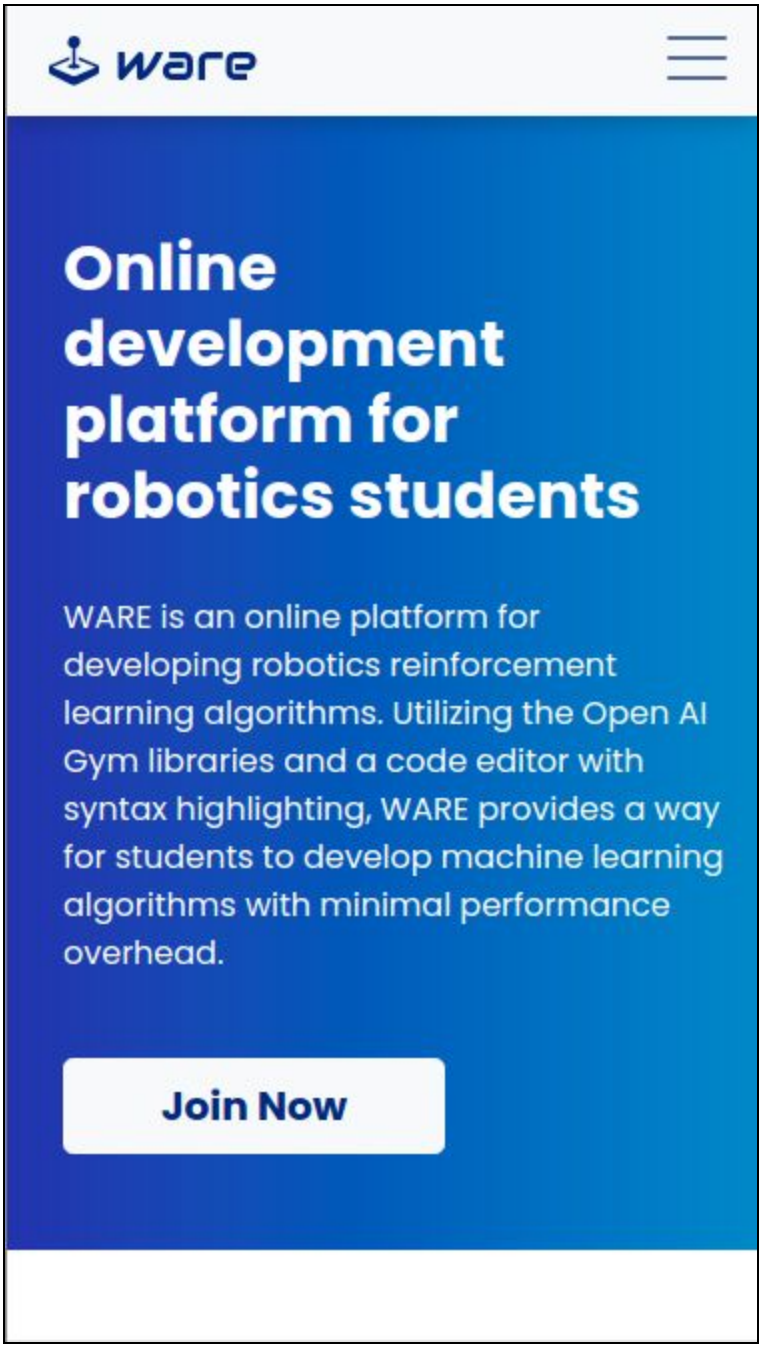

Figure 4.4.2: The homepage as viewed from an iPhone 6 smartphone. The navigation bar on the top features a hamburger icon that lists all of the navigation links. Fonts for the masthead and call to action elements below are optimized for smaller screens.

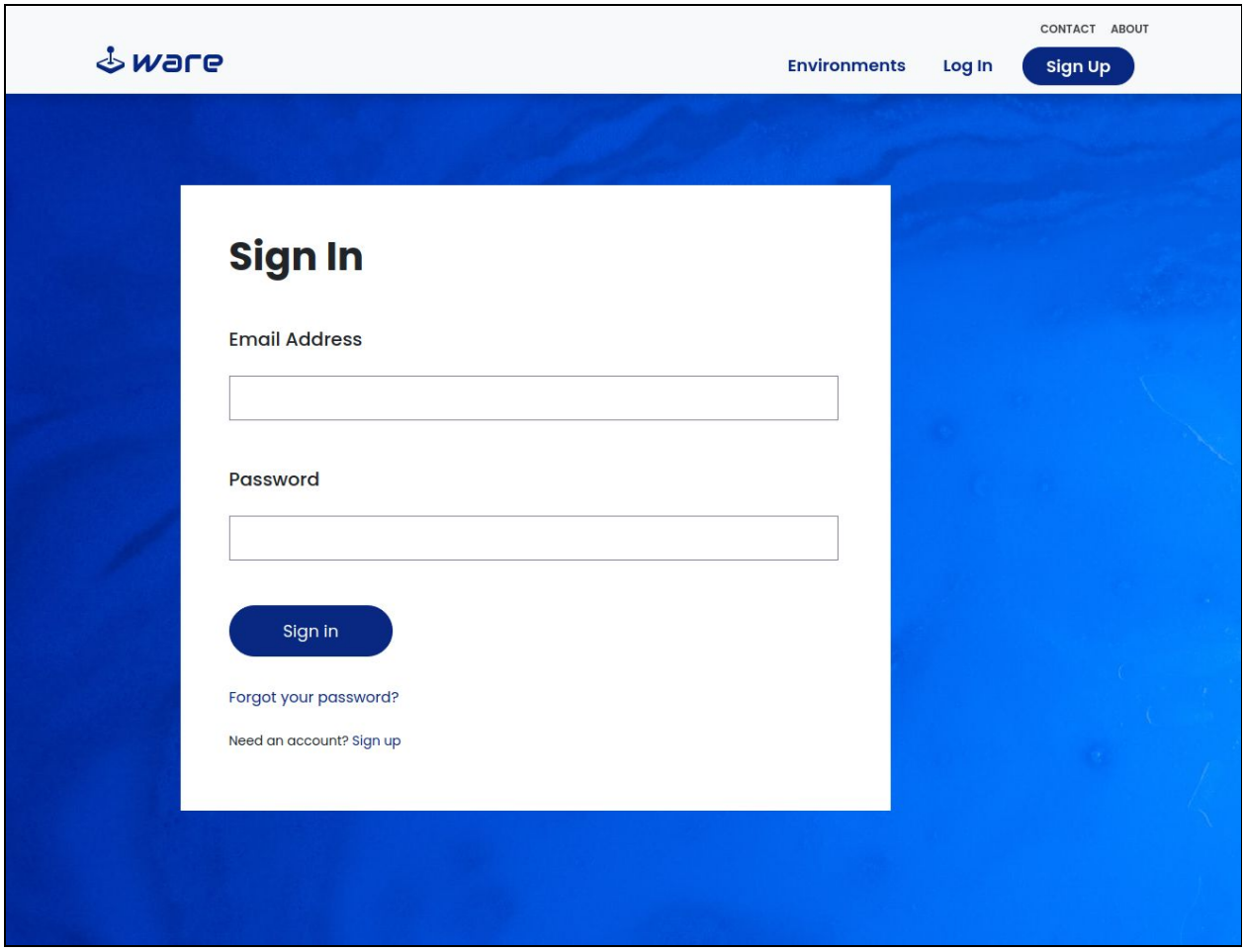

Figure 4.4.3: The WARE sign in page. Existing users will use this page to log in to their account using their email address and password. Additional links are provided below the form: one link navigates to a page to assist the user in resetting their password and the second link navigates to the sign up page

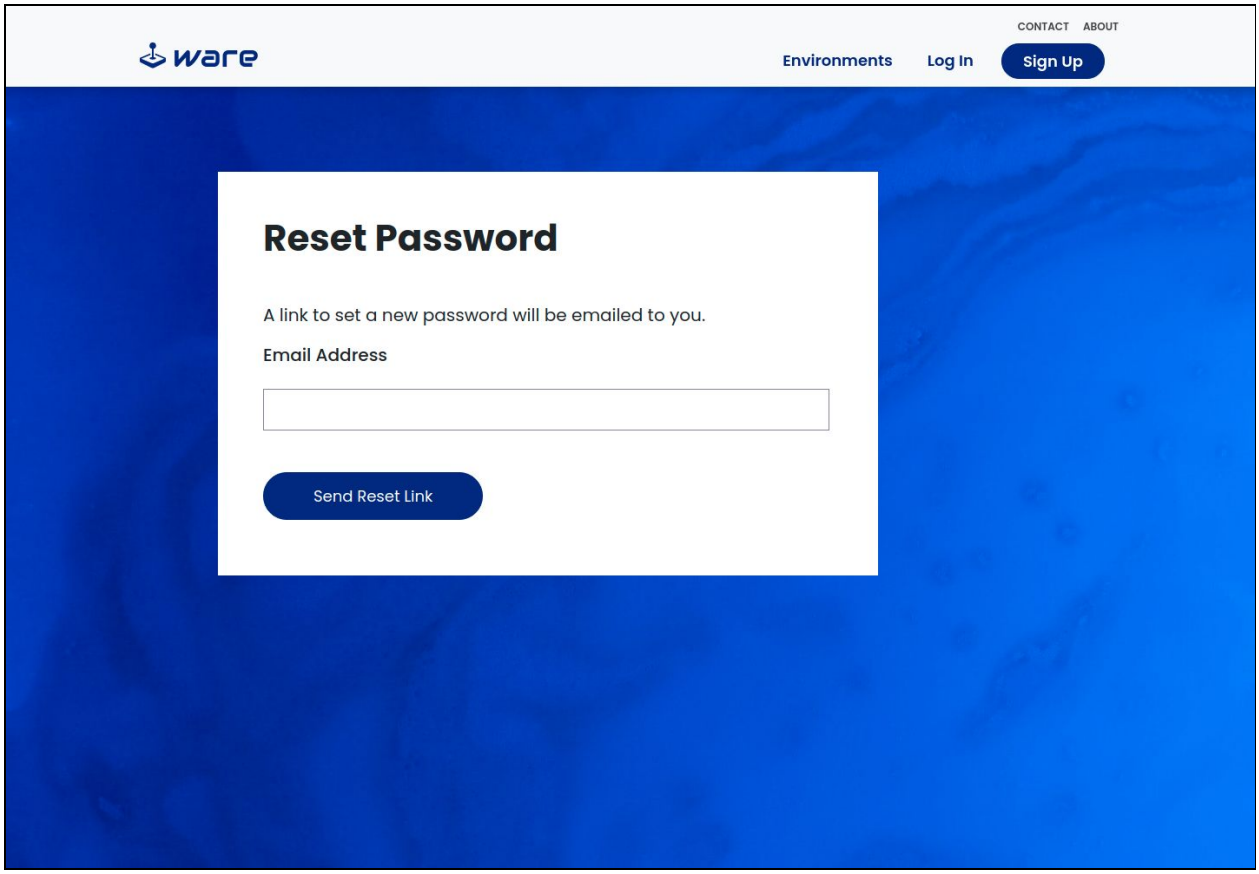

Figure 4.4.4: The reset password page. This form is displayed to the user after clicking the reset link provided in the login page. The form has one field for an email address. After the user clicks the button to submit the reset request, an email will be sent to the provided email address containing a link to reset their password.

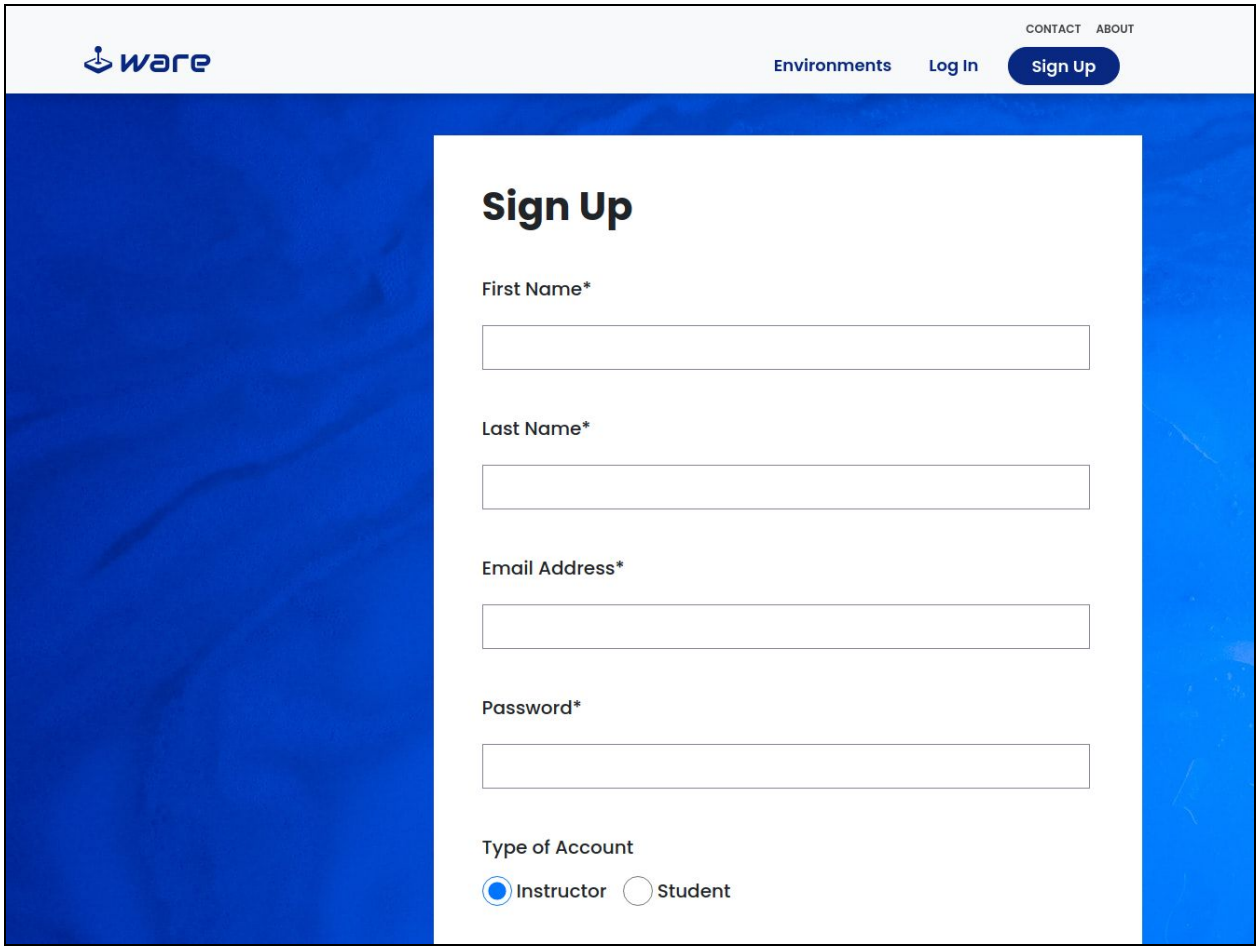

Figure 4.4.5: The WARE sign up page. Users are required to enter their first and last name, email address, and password to create an account. Radio buttons are given to give the user a choice between an instructor or a student account. The submit button is below the form and out of view in the screenshot.

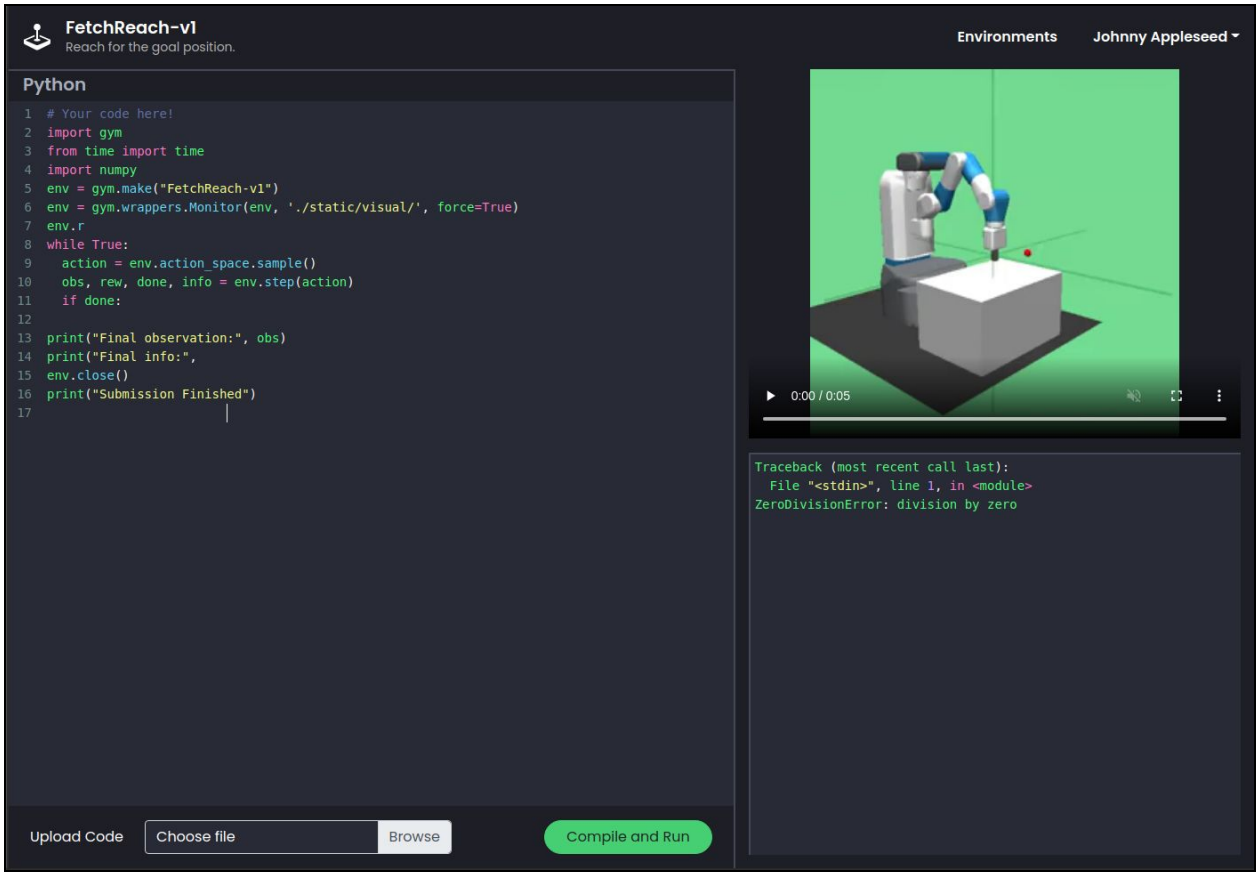

Figure 4.4.6: An environment page with code editor and outputs. The header displays the name of the current environment and its goal. On the right, a link is given for the environments landing page and a dropdown with the user's name is given for user options. In the main section, a code editor is presented that utilizes the CodeMirror Javascript plugin for syntax highlighting. Next to the code editor is a generated video of the robot in use and a terminal output.

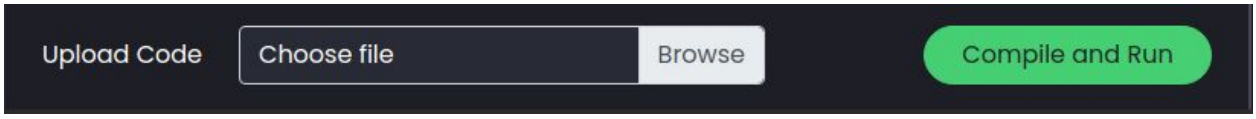

Figure 4.4.7: Options for uploading and compiling code. Users are allowed to upload their own code from a file and have it display in the code editor for further modification and submission. The name of the file uploaded is displayed within the input box next to "Upload Code." On the right, the submission button submits the form to the back-end server for processing.

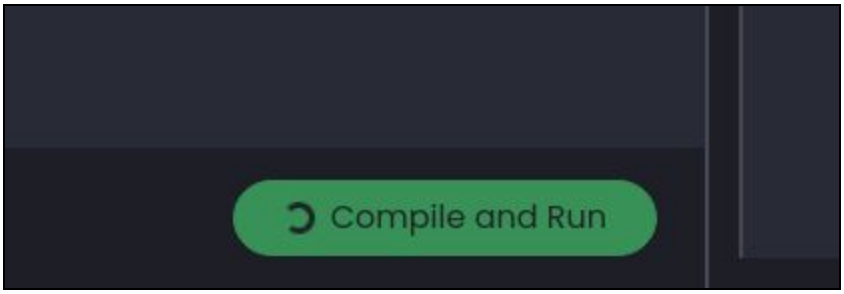

Figure 4.4.8: The compile button after it is clicked. Once the code is submitted, the button is disabled from being clicked again and a progress spinner is displayed to indicate that the code is being processed.

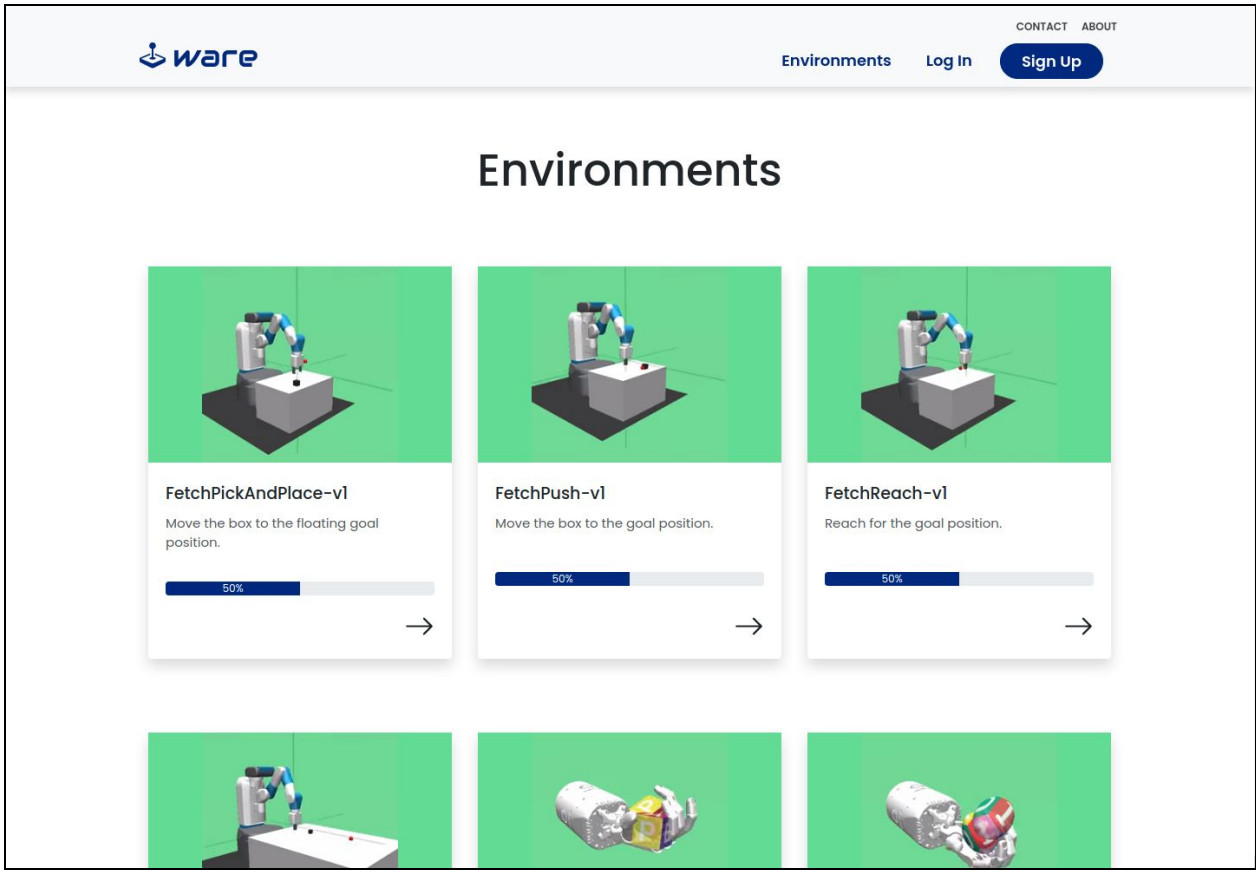

Figure 4.4.9: The environments landing page. This page lists each environment that is available to the user.

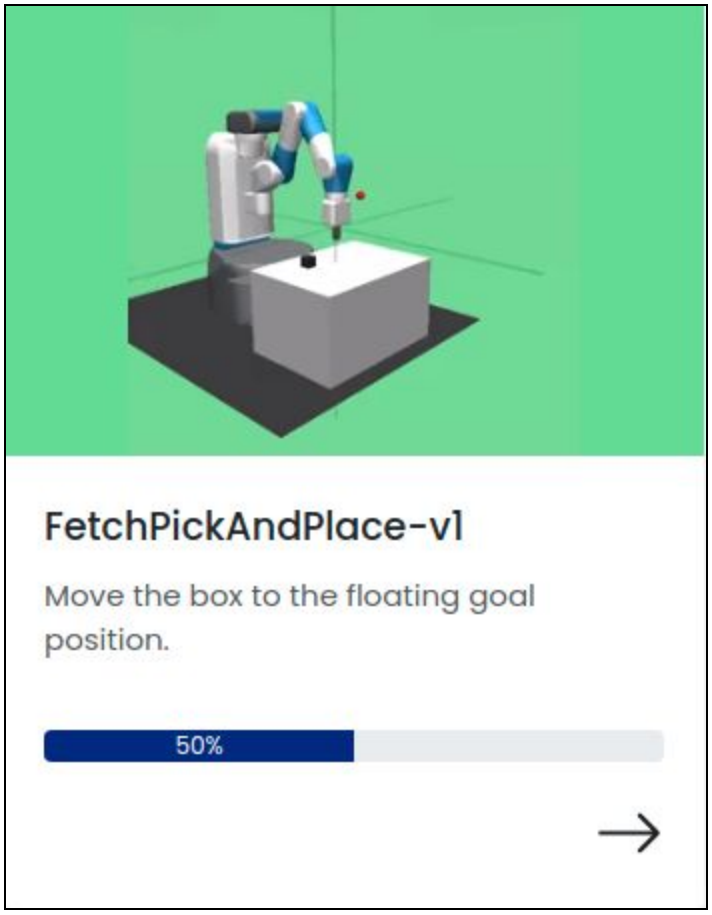

Figure 4.4.10: An example of an environment card that is displayed on the environments landing page. Each card lists the title of the environment, a short description, a progress bar indicating the amount of progress made on the environment by the user, and a decorative arrow.

5. Updated Glossary of Terms

1. Bootstrap - A CSS and Javascript toolkit for implementing pre-developed website components with basic styling into a web application.

2. Chromium-based - An application that utilizes the Chromium Blink engine to render web content

3. Code editor - The input box displayed in an environment that allows a user to input code.
4. Compile - The term used to describe the process of generating an output using code written in a programming language

5. CSS (Cascading Style Sheets) - A programming language that is used by a web browser to style HTML components and interpret how a web page is displayed to users.

6. Database - An organized collection of information stored electronically that can be accessed, modified, and controlled.

7. Environment - Refers to a specific robotics learning exercise in which the user can input code and see the results from compilation of their code.

8. Firefox - An open-source internet browser developed by the Mozilla Corporation.

9. Flask - A basic back-end framework that utilizes Python to run a lightweight web server.

10. HTML (HyperText Markup Language) - The widely adopted programming language used to create and display web pages

11. Internet browser - An application that transmits and receives HTTP web page requests and renders the content received by such requests. Some examples of browsers are Mozilla Firefox, Google Chrome, and Apple Safari

12. Instructor - A user that is given the option within WARE to create a classroom, invite student users to join the classroom, and view code submissions made by student users.

13. JavaScript - An object-oriented programming language used for developing interactive elements and design effects that are displayed in websites.

14. Login - The process that a user undergoes to access their WARE account. Users are required to provide an email address and a password to login.

15. Mobile - Term used to describe cordless computing devices that are smaller than

general computers, more power-efficient, and utilize a touch screen for user input. Examples include smartphones and tablets.

16. MySQL - a popular relational database management system used for implementing an SQL database

17. OpenAI Gym - An open source library that consists of environments for creating reinforcement learning algorithms.

18. Python - A popular, interpretive programming language that is often used for its increased readability and minimal learning curve

19. Student - A user that is given the option within WARE to add themselves to an instructor's classroom and access environments that are allowed by an instructor.

20. Terminal output - The output in the form of text that is displayed to the user within an environment after compiling their code

21. WARE (Web Application for Robotics Education) - The project's name. A web-based learning platform that offers Open AI Gym robotics environments for users to interact with.

6. Engineering Standards and Technologies

Term: Web Content Accessibility Guidelines (WCAG)	
Standard or Technology:	Standard
Description:	WCAG is a set of standards that focus on web accessibility, in a variety of ways, from business policy to individuals with disabilities.
Use or future use in project:	We will take note of various standards and guidelines and what makes them important. We will design our web application with them in mind; an example being keeping our design colorblind friendly.
Reference (if applicable):	https://www.w3.org/WAI/standards-guidelines/wcag/

Term: SHA-2	
Standard or Technology:	Standard
Description:	Hashing is used to secure information from unauthorized personnel. In a web application it is most commonly used to prevent user's sensitive data from being exposed.
Use or future use in project:	We will be using SHA-2 to better secure our user's passwords. We will not be storing the user's password in the database, but rather a hash of it which cannot be reversed if it were to somehow be exposed.
Reference (if applicable):	https://csrc.nist.gov/csrc/media/publications/fips/180/2/archive/2002-08-01/documents/fips180-2.pdf

Term: HyperText Markup Language (HTML)	
Standard or Technology:	Standard
Description:	HTML is the main structure of any website. It's how the web browser is able to format web pages to be displayed to the user.
Use or future use in project:	Our website's front end / user interface is primarily done by using HTML and styled with CSS.
Reference (if applicable):	Not applicable

Term: Python	
Standard or Technology:	Technology
Description:	Python is a multipurpose programming language that is highly flexible, simple to use, and versatile.
Use or future use in project:	WARE has a backend and database fully written in Python. We utilize many of its libraries for versatile uses.
Reference (if applicable):	Not applicable

Term: Flask	
Standard or Technology:	Technology
Description:	Flask is a web framework that is used for creating web applications.
Use or future use in project:	Since the beginning of our project first being created we have been using Flask.
Reference (if applicable):	Not applicable

Term: Visual Studio Code	
Standard or Technology:	Technology
Description:	Visual Studio Code is a professional code editor and can be used for a variety of programming languages.
Use or future use in project:	We have been using Visual Studio Code since the beginning. It has many features and plug-ins that improve quality of life, as well as keeps files and folders organized nicely.
Reference (if applicable):	Not applicable

Term: Bootstrap	
Standard or Technology:	Technology
Description:	Bootstrap is a framework that focuses on CSS, which inturn makes web applications look better and professional.
Use or future use in project:	We began using bootstrap close to the beginning of creating the project. We need to utilize CSS to make our website's user interface friendly and easy to navigate.
Reference (if applicable):	Not applicable

Term: JavaScript	
Standard or Technology:	Technology
Description:	JavaScript is a programming language that adds functionality to a webpage.
Use or future use in project:	We have used JavaScript to aid with submission processing on our environment page. We will be implementing a text editor called ‘CodeMirror’ soon which is also made with JavaScript.
Reference (if applicable):	Not applicable

Term: OpenAI Gym	
Standard or Technology:	Technology
Description:	The OpenAI Gym toolkit is for developing and comparing reinforcement learning algorithms
Use or future use in project:	The OpenAI Gym is the core of our project. Users will be programming in Python to complete the objectives of each environment.
Reference (if applicable):	Not applicable

7. Project Impact and Context Considerations

Our project holds great importance due to the fact that in order for the field of robotics to grow and evolve, we must also make it accessible and allow new students to learn and practice with it. Robotics is becoming more and more important within our daily lives, from roombas to self driving vehicles. The field is constantly expanding to improve the lives of everyone. By making robotics training more accessible to students, we can find new bright minds to create innovative technologies to push the field even further. The robotics innovations can be applied to things such as life saving machines in the medical field, or improved road safety, the possibilities are endless. WARE gives students an opportunity to discover and learn about this world and how they can contribute to it.

The economic aspect of WARE is very positive because having an online resource means less money and resources need to be spent on creating real life robots to practice on. Robotics can be quite difficult for a lot of people to get into because they don't have access to those resources to test, but being online allows students to practice code without having to order expensive robot parts or assembling the robot. This will help globally as well because more places around the world that may not have been able to previously practice coding with robots can now more easily practice with them through the internet.

8. Updated List of References

8.1 Problem Domain Book

Khine, M. S. (2017). Robotics in STEM education redesigning the learning experience. Cham: Springer. doi:10.1007/978-3-319-57786-9

The whole purpose of our project is to bring more accessibility to the underexplored field of robotics and help students learn about robotics concepts and principles. Our goal is to help draw interest and attention to this field despite its generally higher barrier of accessibility compared to most other subjects. Khine’s book helps bring light to the importance of robotics and brings light to better methodologies to best teach others.

8.2 Reference Articles

Wang, Qianxiang, et al. “Educational programming systems for learning at scale.” 2014, pp. 177-178. Doi = 10.1145/2556325.2567868.

This article is about teaching students in an ideal way to best understand programming. It mentions having a strong system that is close to a level of mentorship. This can be of use to us because we need to create a help / tip section for if a student is not submitting code that completes objectives multiple times.

Chalmers, Christina. "Preparing Teachers to Teach STEM through Robotics." *International Journal of Innovation in Science and Mathematics Education*, vol. 25, no. 4, 2017.

Chalmers’ article focuses mainly on teachers and the various data from being surveyed after certain activities regarding STEM / robotics activities. While it’s main purpose is to better educate teachers with regards to teaching activities involving STEM, it gives an important perspective to anyone who needs to teach students through any means which is needed as we’re essentially setting up a teaching activity.

Berndt, Sara, et al. “The SDM Finger: Teaching engineering design through soft robotics.” Science scope (Washington, D.C.), vol. 43, no. 4, 2019, pp. 14–21.

This research article helps prove the point on the importance of giving students early exposure to a field or subject to draw more interest in it early on. With the field of robotics having a higher barrier of entry than most other fields, it’s difficult to get those who are younger interested in it. Lowering that barrier is partially what WARE is trying to accomplish.

8.3 Websites

https://flask.palletsprojects.com/en/1.1.x/

Flask is the primary framework we are utilizing in order to create the WARE website. Flask provides many of the tools we need to develop our web application while remaining simple and easy to use. It is important that we have the documentation to follow for reference whenever we get stuck.

https://gym.openai.com/docs/

OpenAI's Gym is what our project is centered around. All of the user environments utilize Gym for their various goals and exercises that the students will be working on and completing. With all members of our team being newly introduced to this we will be reliant on documentation.

https://getbootstrap.com/docs/5.0/getting-started/introduction/

Bootstrap is another important framework that is being used to create a simple, user friendly, clean interface through CSS. With Bootstrap we are able to create a professional looking website with plenty of design and colors while still maintaining a simple style which looks good to the user but also helps them navigate the website.

9. Contributions of Team Members

Sean Griffith

Time spent: 7 hours
Sean wrote the Summary of Changes in Project Specification section, updated and reorganized the Updated Technical Requirements Specification section, updated and reorganized the Updated Use Case Modeling section, contributed to the Program Units section, and revised and updated the Data Structures section.

Ryan Lunt

Time spent: 6.5 hours
Ryan created and filled the tables in the Engineering Standards and Technologies section, found and added references to the Reference section, and contributed to the Program Units section by updating and adding program units.

Herman Hira

Time spent: 3 hours
Herman wrote the Summary of Changes in Project Design section, helped revise the Program Units section, and wrote the Project Impact and Context Considerations section.

Zachery Wiles

Time spent: 10 hours
Zach wrote the Abstract, drafted the Recent Project Changes section, contributed to the system level diagram in the Updated Design section, created the user interface components for the screenshots provided in the Update Design section, and updated the glossary terms.