Report for a software engineering project to develop a web-based Task Management System software.

Advanced Software Engineering Report

Task Management System

Authour: S275931

Advanced Software Engineering Report

Table of Contents

Introduction / Background

This is an Advanced Software Engineering report about the creation of Task Management System web application that is accessible in a web browser and manages a task management database.

Item	Link
Task Management System GitHub Repository	https://github.com/AlexanderJohnRobertson/task-management-system.git
Task Management System Clean Install GitHub Repository:	https://github.com/AlexanderJohnRobertson/task-management-system-clean-install
Task Management System Documentation GitHub Wiki Home:	https://github.com/AlexanderJohnRobertson/task-management-system/wiki
Task Management System Documentation GitHub Wiki Documentation	https://github.com/AlexanderJohnRobertson/task-management-system/wiki/Task-Management-System-Documentation
Task Management System Documentation GitHub Wiki Documentation License	https://github.com/AlexanderJohnRobertson/task-management-system/wiki/Task-Management-System-GNU-AGPLv3-License-Agreement
Task Management System Documentation Clean Install GitHub Wiki Home	https://github.com/AlexanderJohnRobertson/task-management-system-clean-install/wiki

Task Management System Documentation Clean Install GitHub Wiki Documentation	https://github.com/AlexanderJohnRobertson/task-management-system-clean-install/wiki/Task-Management-System-(Clean-Install)-Documentation
Task Management System Documentation Clean Install GitHub Wiki Documentation License	https://github.com/AlexanderJohnRobertson/task-management-system-clean-install/wiki/Task-Management-System-GNU-AGPLv3-License-Agreement
Render.com Deployment:	https://task-management-system-hnmb.onrender.com/
Render.com Deployment (Clean Install):	https://task-management-system-clean-install.onrender.com
DockerHub Repository:	https://hub.docker.com/r/alexjrobertson/taskmanagementsystem
Docker Pull command:	`docker pull alexjrobertson/taskmanagementsystem`
DockerHub Repository (Clean Install):	https://hub.docker.com/r/alexjrobertson/taskmanagementsystemcleaninstall
DockerHub Pull command (Clean Install):	`docker pull alexjrobertson/taskmanagementsystemcleaninstall`

Login Credentials:

User Role	Username	Password
Root User	Root	TaskManagement123#
Administrator	barneyPurpleDinosaur	BabyBop123#
Standard	Birmingham888	SouthSide123@

Definition of Software Engineering

Software Engineering is the process of planning, analysis, design, development, coding, implementation, testing, integration, deployment and maintenance and retirement of computer software. These phases form the Software Development Life Cycle (SDLC). There are three main types of software engineering:

1. Operational Software Engineering – How the software performs in a computer system, planning, design, coding, development, project and team management, building, testing.
2. Transitional Software Engineering – Software's scalability and adaptability outside its original environment such as testing, deployment.
3. Software Engineering Maintenance – Maintaining the software such as updates, upgrades, bug fixes, compatibility with newer technologies, security patches, technical support, new features, retirement. (Yasar, 2024)

Purpose of the Task Management System

The Task Management System (TMS) is a simple lightweight web application that makes it easier to manage projects and tasks associated with these projects. This/It also requires users, user authentication and user roles (standard, administrator, and root) for security to protect data and prevent unauthorized use or misuse. The need for project stakeholders, project clients, project managers, organizations, companies, corporations, governments, educational institutions, and individuals to have a straightforward way to keep track of the progress of projects as a list table and the progress of associated tasks in another list table. The TMS provides simple to use forms to make changes to projects and tasks by creating, viewing, updating, and deleting tasks and projects. The TMS is Cross-Platform and can be deployed locally using Docker or on the cloud with a webhosting

service such as render.com Users can also manage their own accounts and administrators can manage other user accounts.

What are you trying to solve with the app?

The Task Management System solves the problem of trying to make an easy to use, secure, reliable, lightweight, and cross-platform Task Management System web application that can be deployed locally using a containerisation platform such as Docker or on a cloud web hosting service such as reder.com. The TMS helps keep track of projects, tasks, and has user authentication for security.

How are you solving it?

My solution is to create a project and task management web application that I currently call the Task Management System. The backend of the Task Management System is written in Python, using the Flask API framework for web development using Python. The backend Python code also manages the Task Management System's SQLite database by executing SQLite code to perform Create, Read, Update, Delete (CRUD) operations.

The front-end code for the Task Management system is written as webpages in HTML, CSS, and JavaScript.

The Task Management System application can be deployed locally as an image/container on a containerisation platform such as Docker or on a webhosting service such as render.com. This means that the Task Management System is cross-platform as Docker is compatible with Windows 10 and 11, macOS and Linux. The online version on render.com is accessible on any device with an internet connection and a compatible web browser.

Design Choices and Implementation

Requirements Re-engineering

When existing requirements of a software project change, after analysing and defining them. E.g. change in business requirements, technology advancements, rectify mistakes made during design, improvements, feedback from users, performance issues, integrate legacy systems

I have re-engineered the Task Management System because the requirements changed due to design smells .The first design flaw was not having an interface for the administrators to manage other users such as changing their account type, deleting, blocking, and unblocking other users, resetting their passwords, viewing a list of all users and factory resetting the Task Management System. I also re-engineered it to accommodate a Root user for first time setup. I also re-engineered the authentication to use a randomly generated session ID for each login (stored in the database and in a cookie and must match) as a user testing the software was able to circumvent the authentication by copying the username from the cookie while logged in and pasting it into the empty cookie when logged out to gain unauthorized access.

Design Diagrams

UML Use Case Diagram

I have created a Use Case Diagram to summarise how users can interact with the Task Management System web application, their roles, and what operations they can perform. The actors outside the system boundary represent the users and automated processes by the Task Management System.

The ovals containing text are the use cases that define features and functionality where the users interact with the Task Management System to perform operations with the application. The rectangle is the system boundary. The lines (called associations) connecting the actors to the use cases define what roles different actors can do. Use Case Diagrams form a part of the Unified Modelling Language (UML), used to design software. (Lucidchart, 2024)

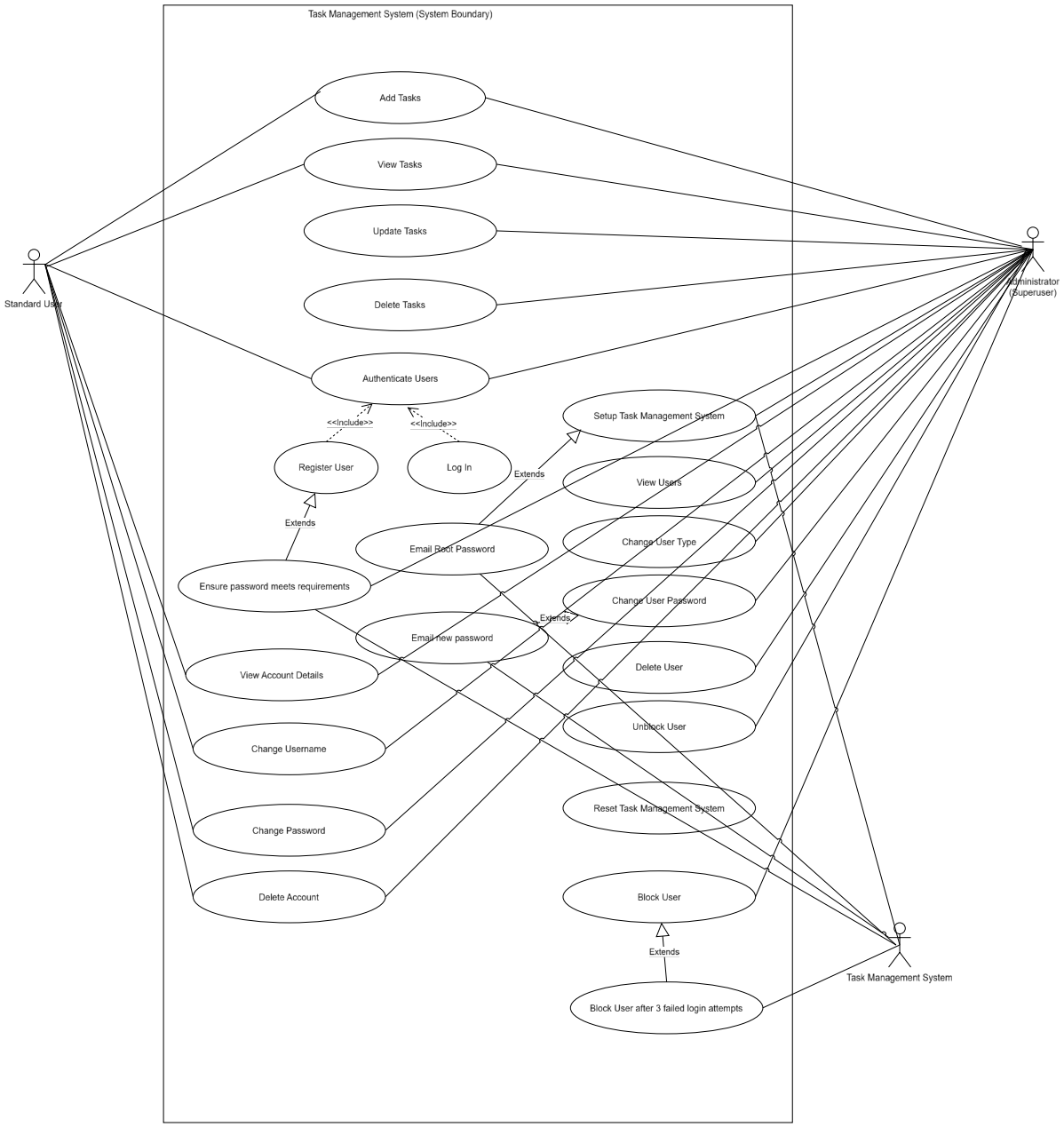


Figure 1 - Task Management System UML Use Case Diagram

UML Class Diagram

I have created a UML Class Diagram to define the Task Management System's program structure by defining its classes, objects, methods and relationship between the objects and classes and used to define the data and data types for the database. Each class consists of a name (top section), the class attributes (middle section) and the class methods (bottom section). The attributes define the data associated with each object and the type of data. The methods define the operations that each class can do. There are five classes (Task Management System, Tasks, Projects, User, Administrator). Attributes and methods with the + symbol are public attributes (such as username) while the attributes and methods with the - symbol are private attributes (such as password). The methods are the functions that each class has access to. There are two association types in my class diagram – composition where one class is part of another class and is can only exist if the other class exists, and inheritance where a subclass is part of superclass sand inherits attributes and methods from the superclass. (Visual Paradigm, 2024)

Task Management System Class Diagram

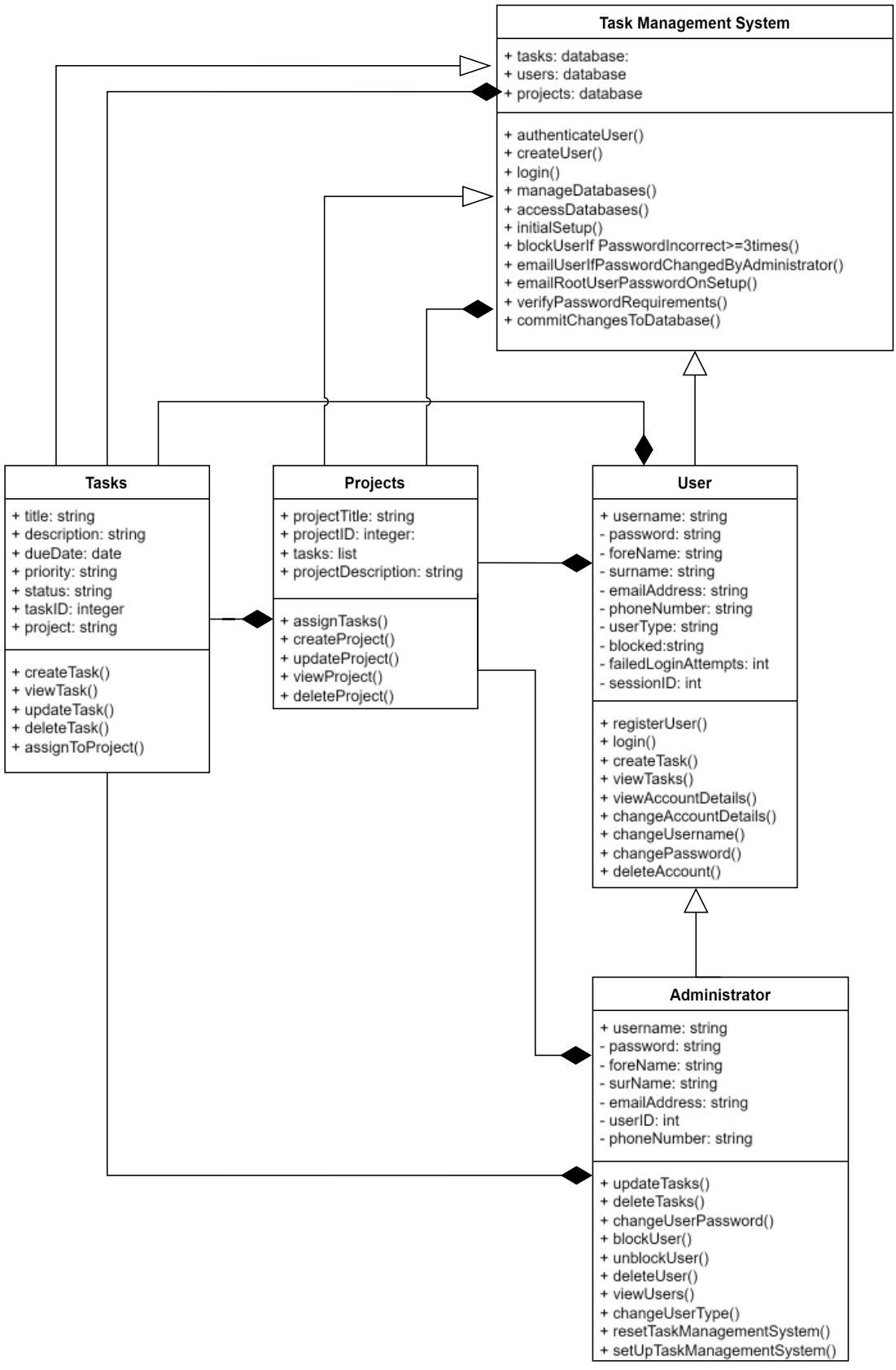


Figure 2 - Task Management System Class Diagram

Wireframe Diagram

I have created a wireframe of the Task Management System to plan and design its user interface (UI) and user experience (UX) before creating a mock-up in full colour. The wireframe helped me design the forms tor task, project and user management, the tables, navbar, and overall webpage layout. The wireframe design is also a sitemap that shows how the webpages are linked to demonstrate navigation.

Wireframe diagram available here: https://managing-projects-and-teams-diagrams.onrender.com/wireframe

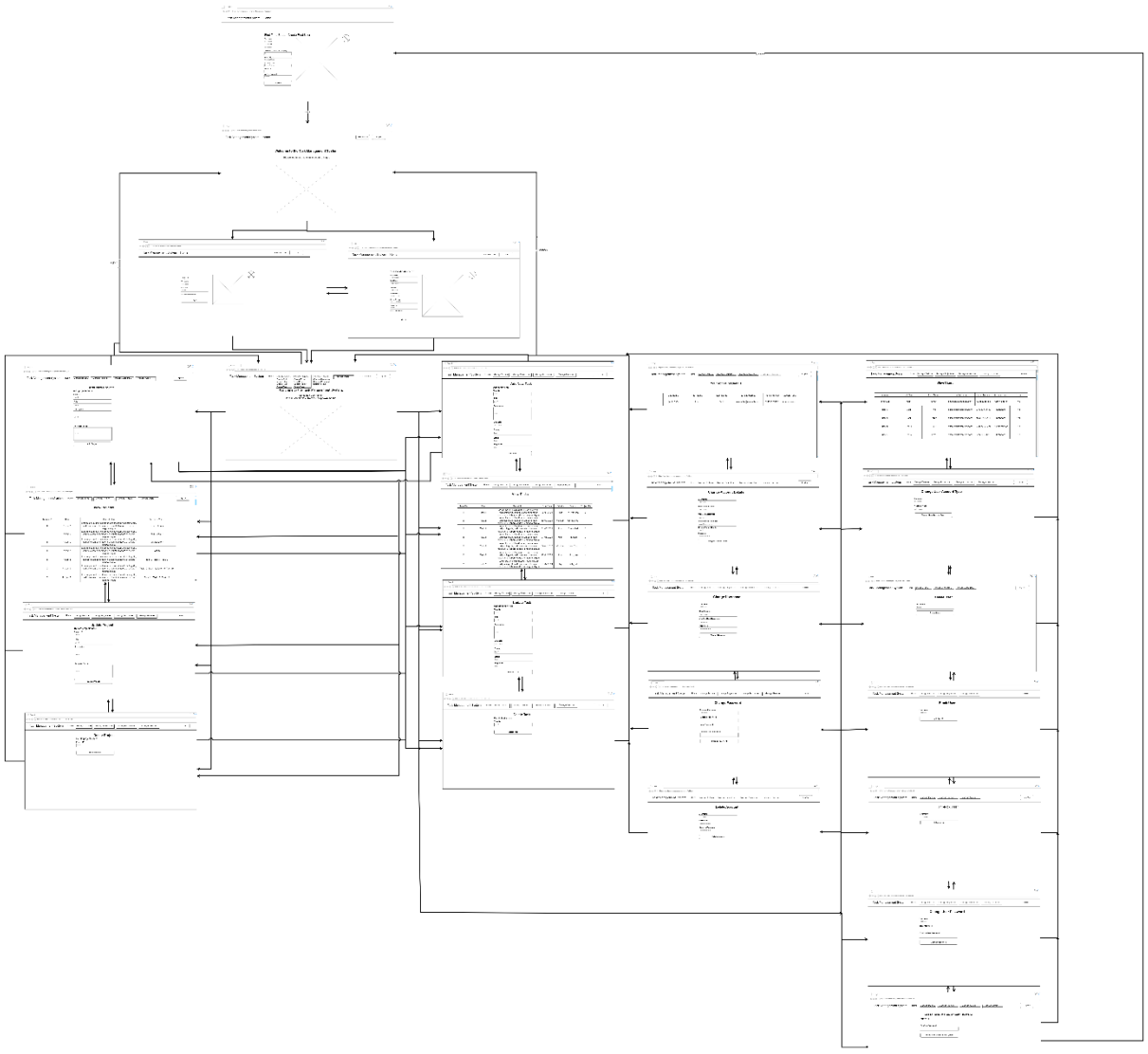


Figure 3 - Overall Task Management System Wireframe Diagram

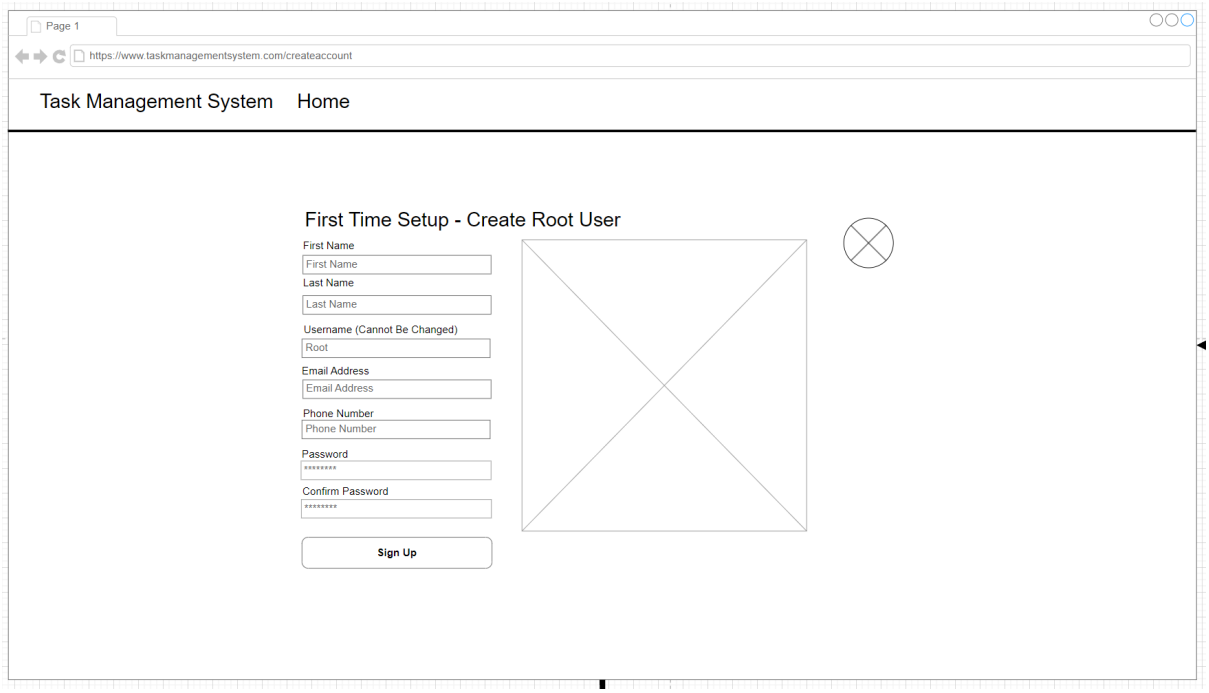


Figure 4 - Setup Wireframe

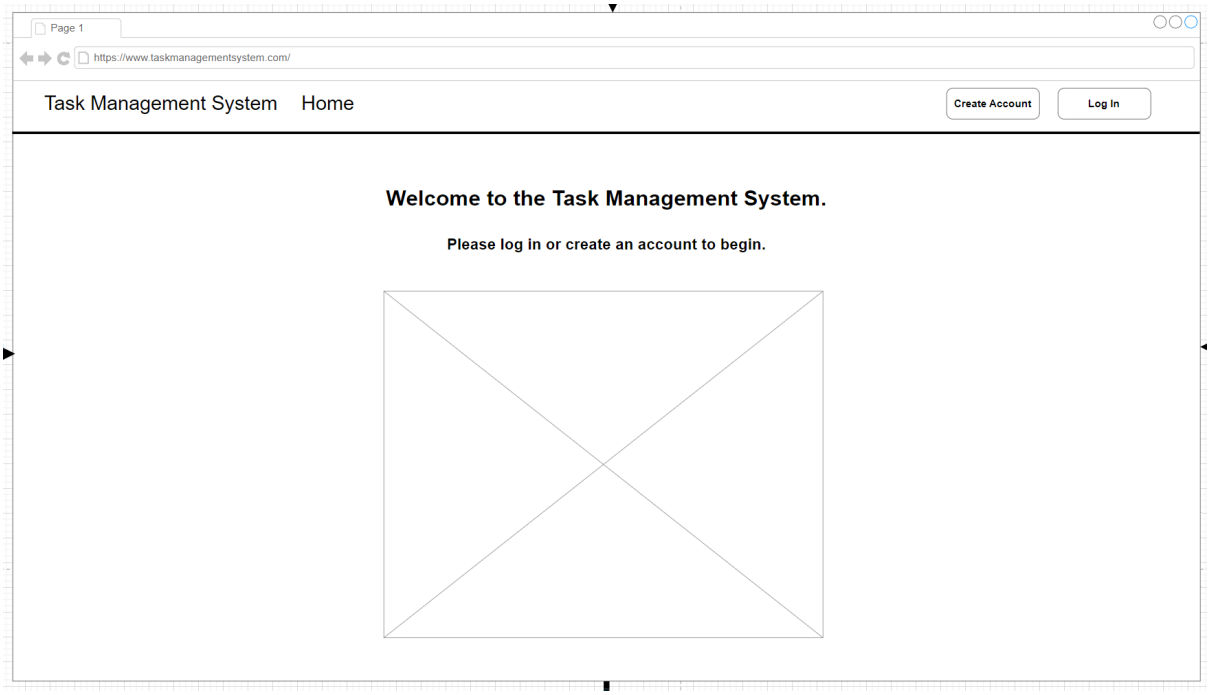


Figure 5 - Landing Page Wireframe

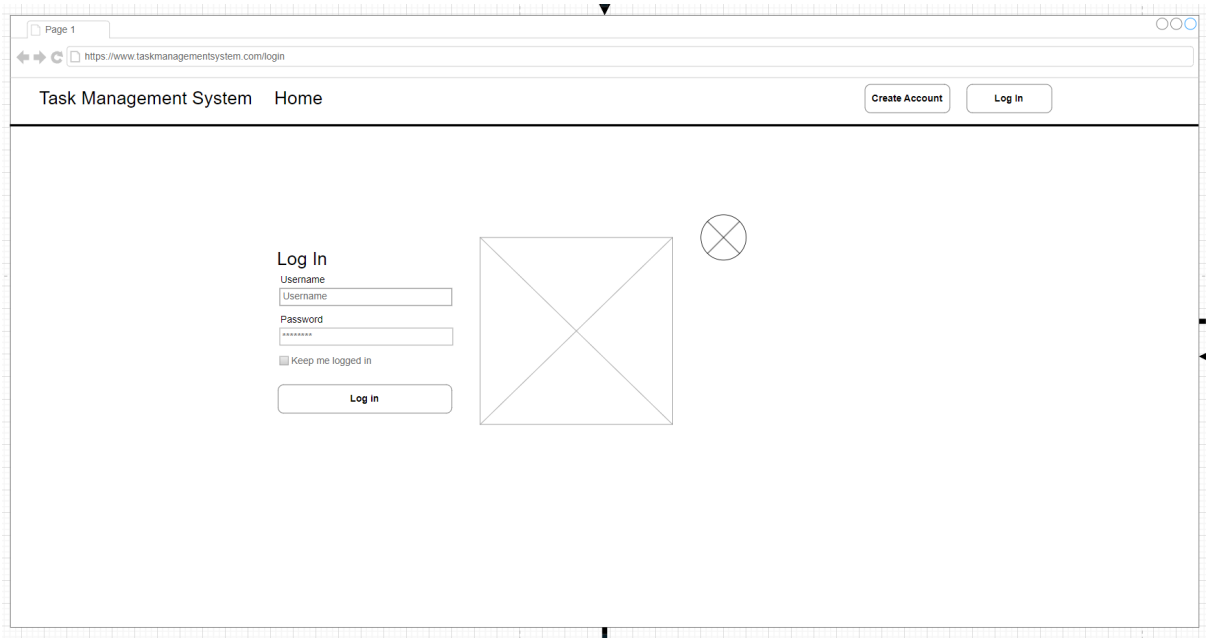


Figure 6 - Login Wireframe

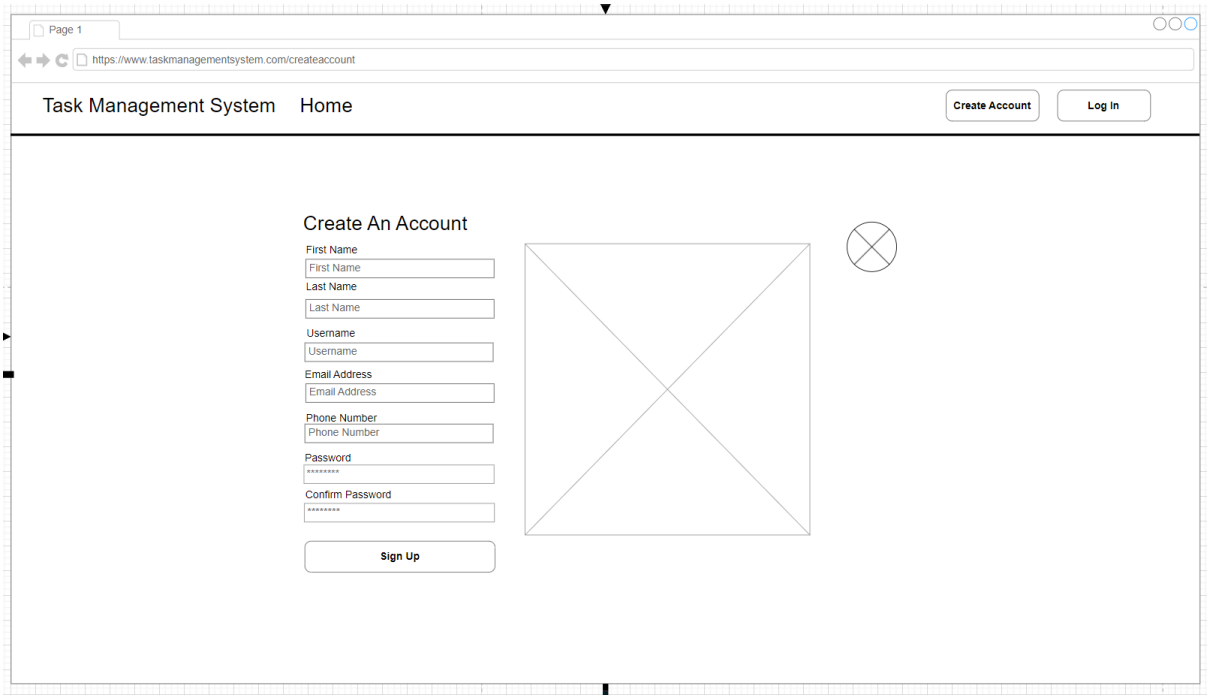


Figure 7 - Create Account Wireframe

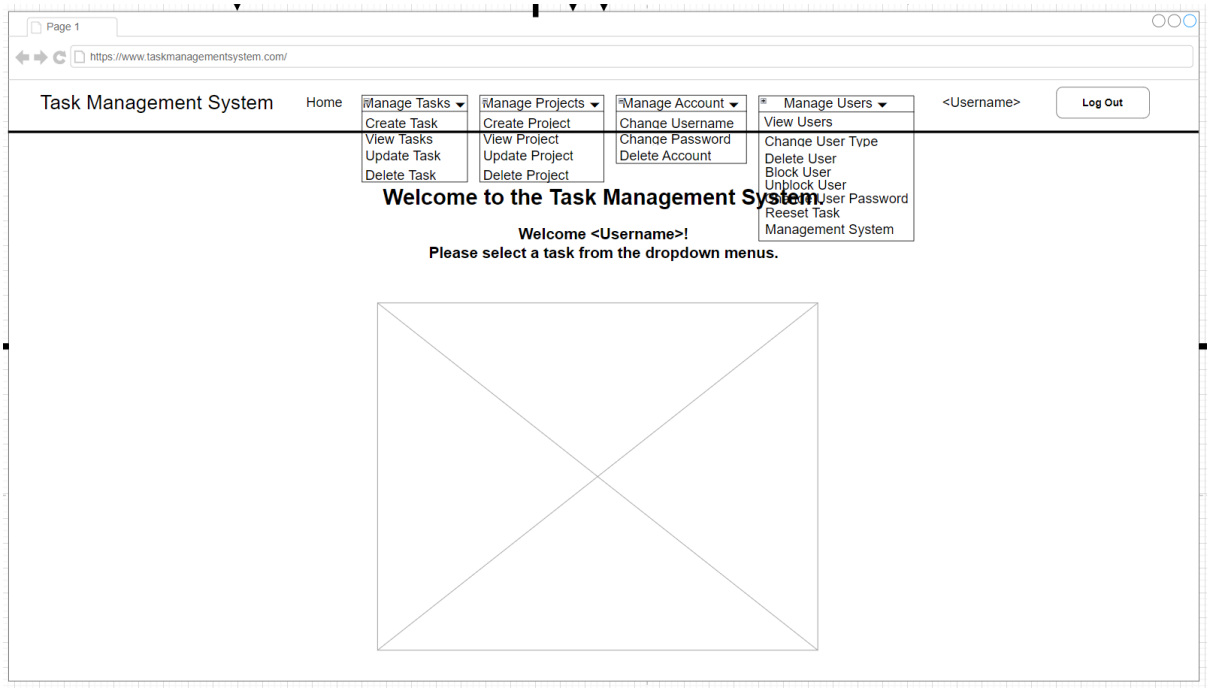


Figure 8 - User Home Wireframe

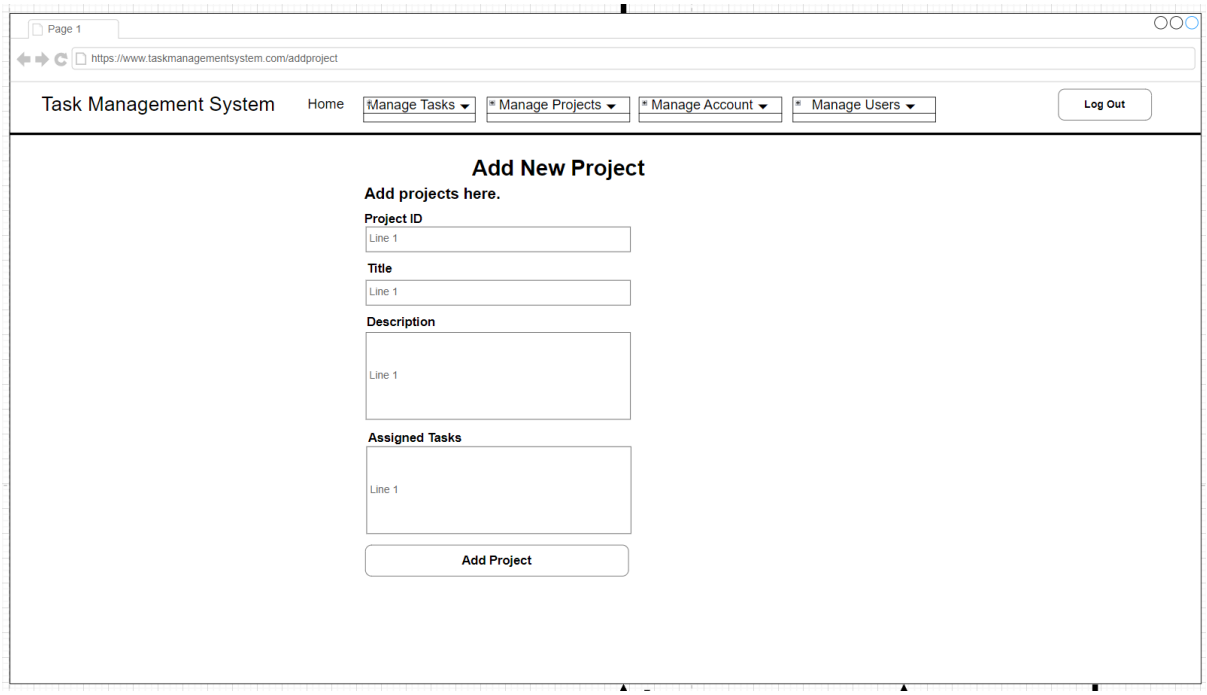


Figure 9 - Add Project Wireframe

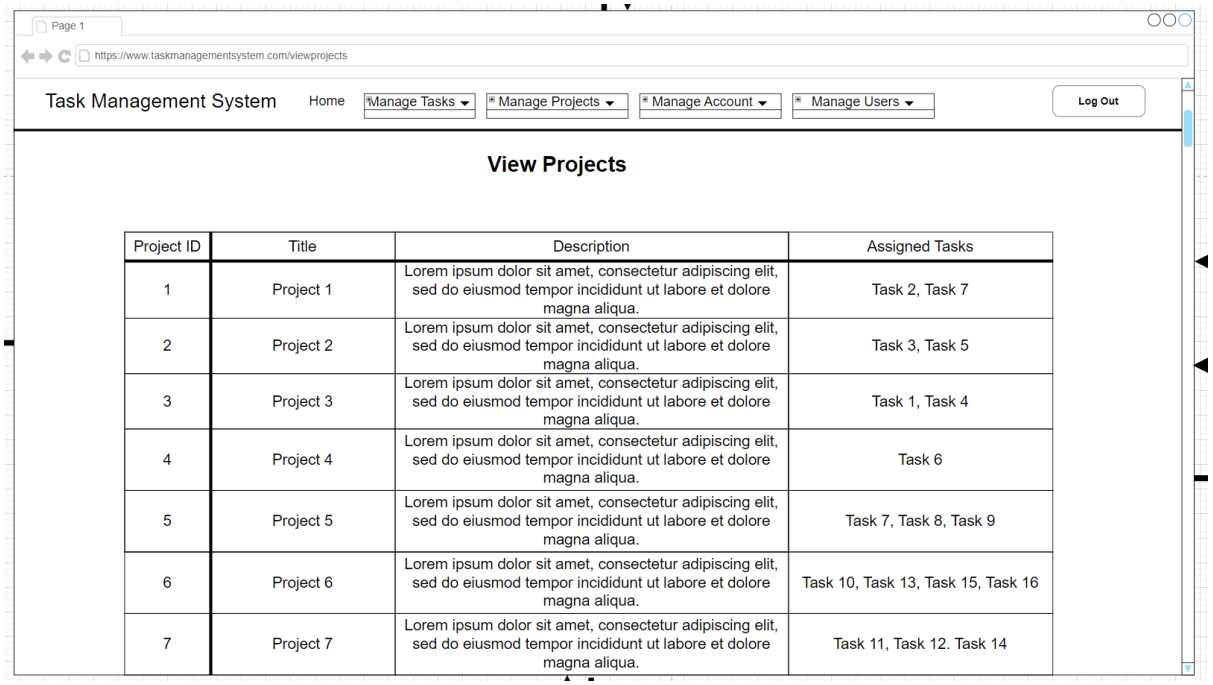


Figure 10 - View Projects Wireframe

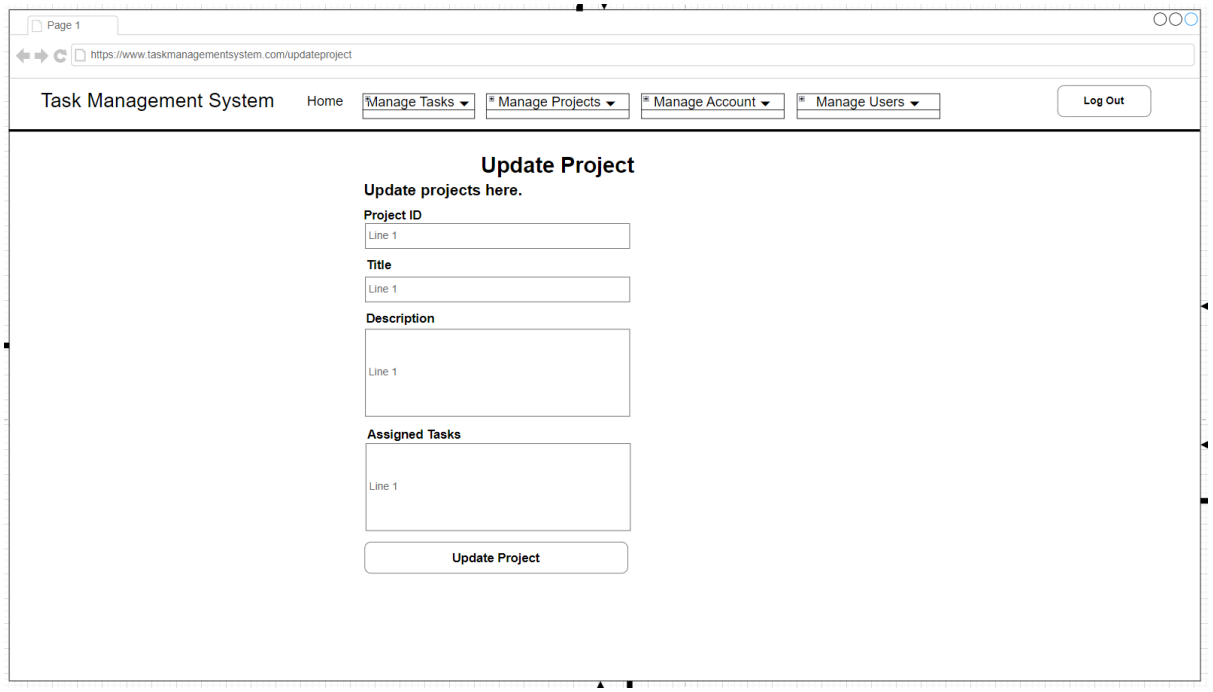


Figure 11 - Update Project Wireframe

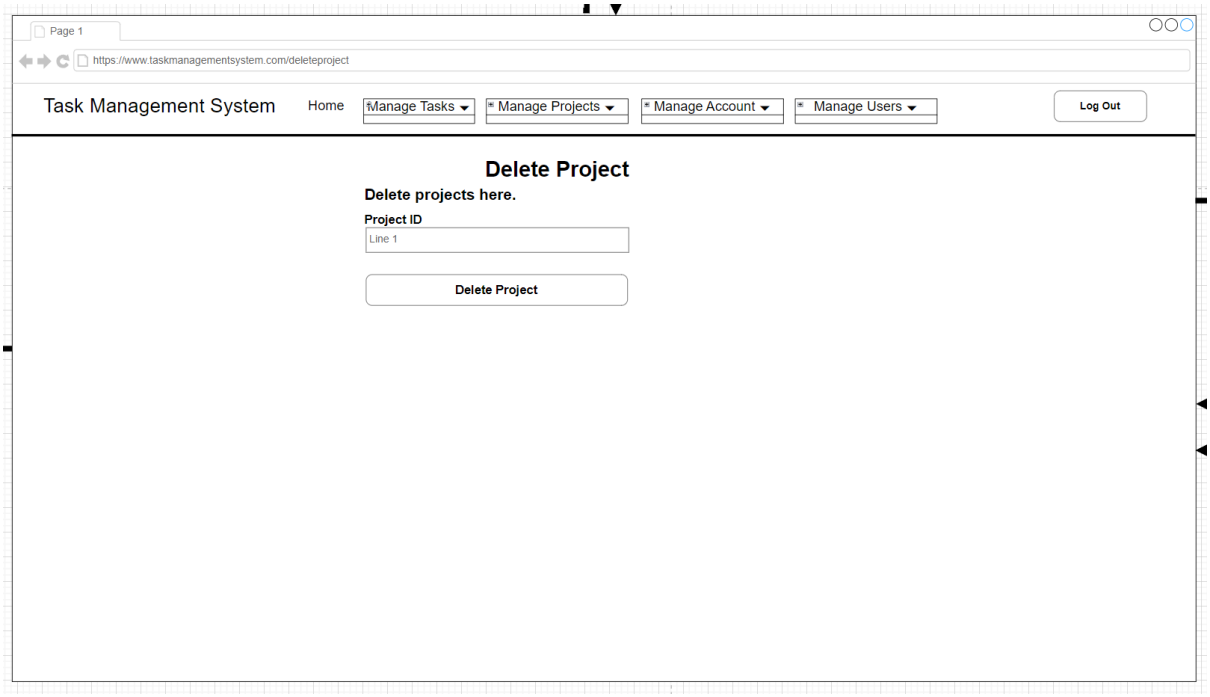


Figure 12 - Delete Project Wireframe

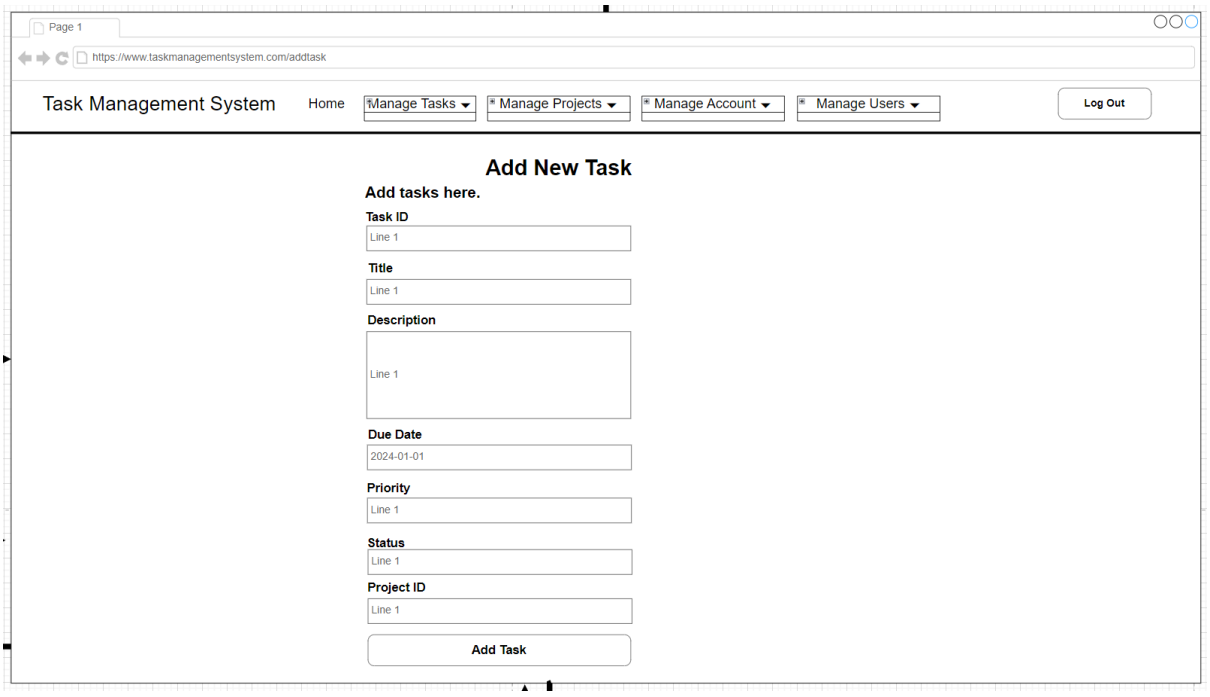


Figure 13 - Add Task Wireframe

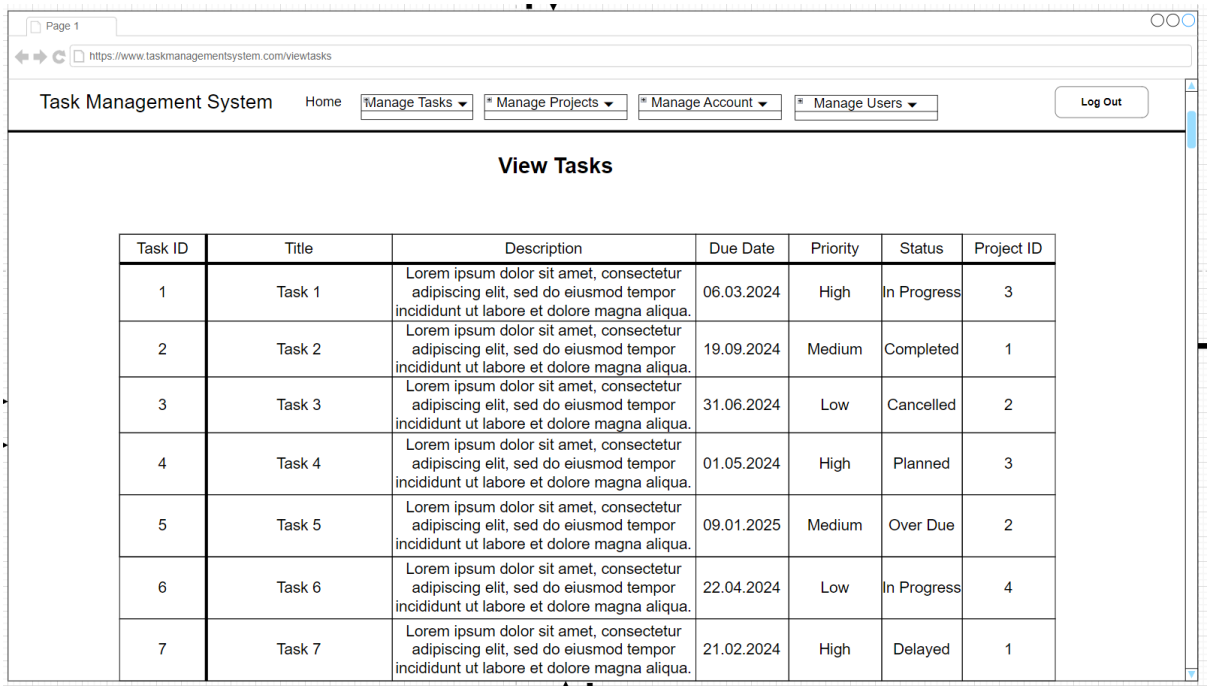


Figure 14 - View Tasks Wireframe

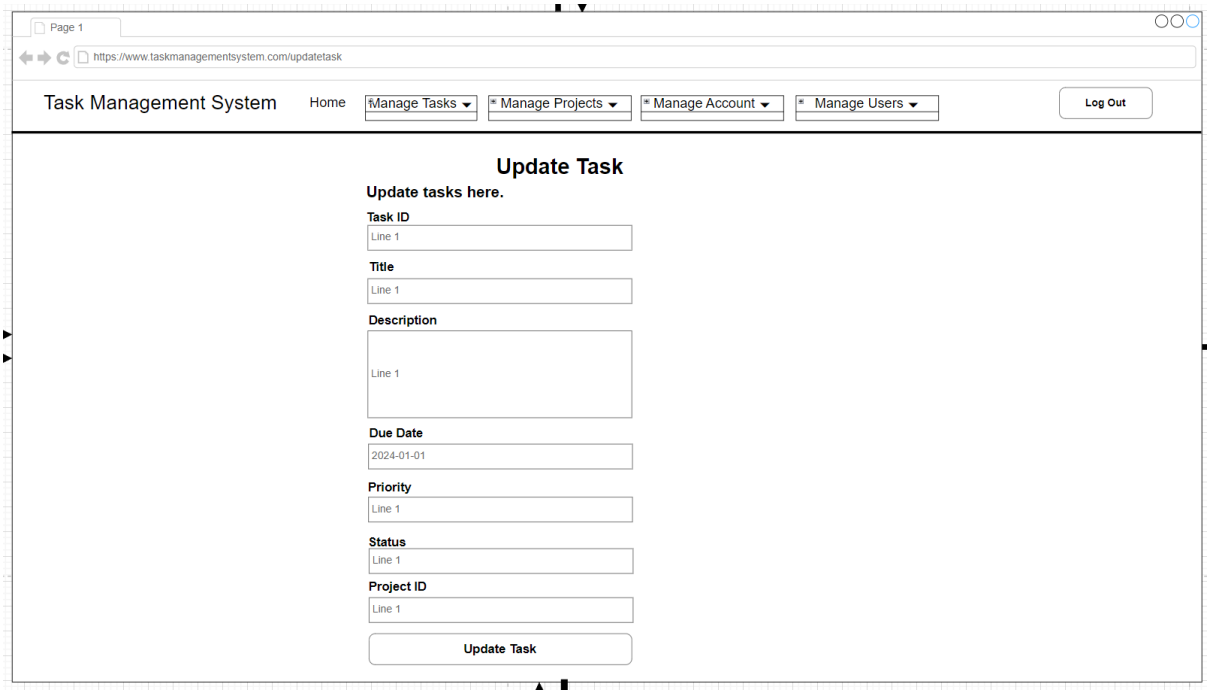


Figure 15 - Update Task Wireframe

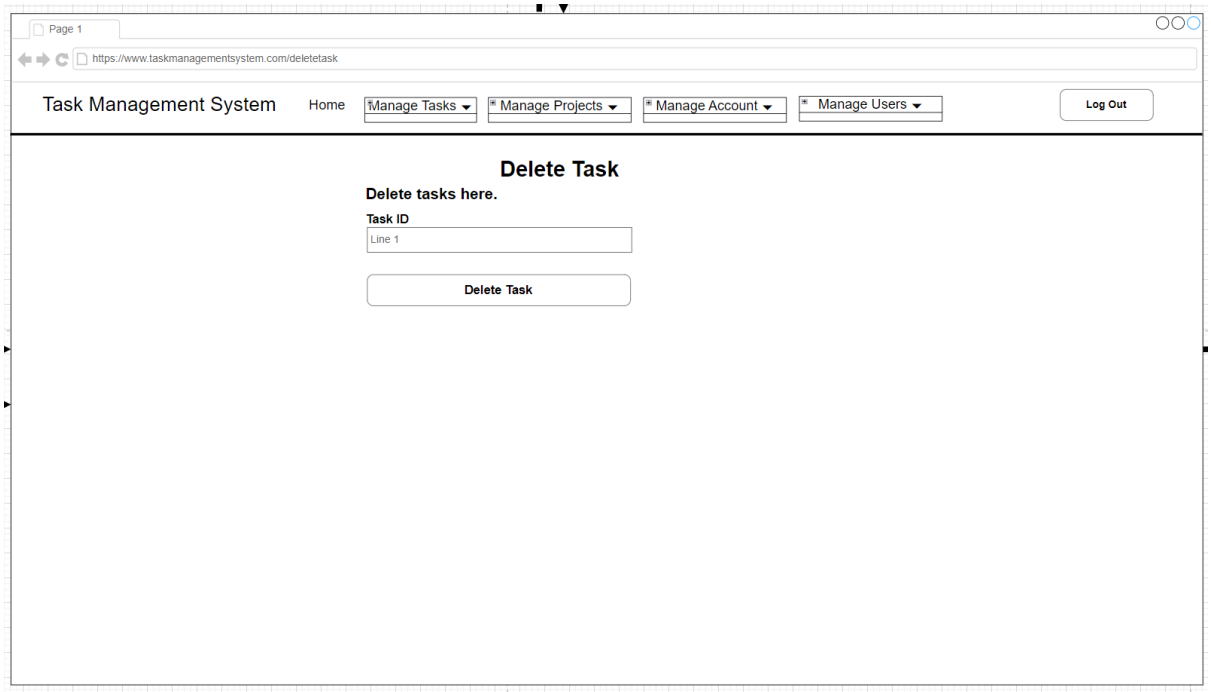


Figure 16 - Delete Task Wireframe

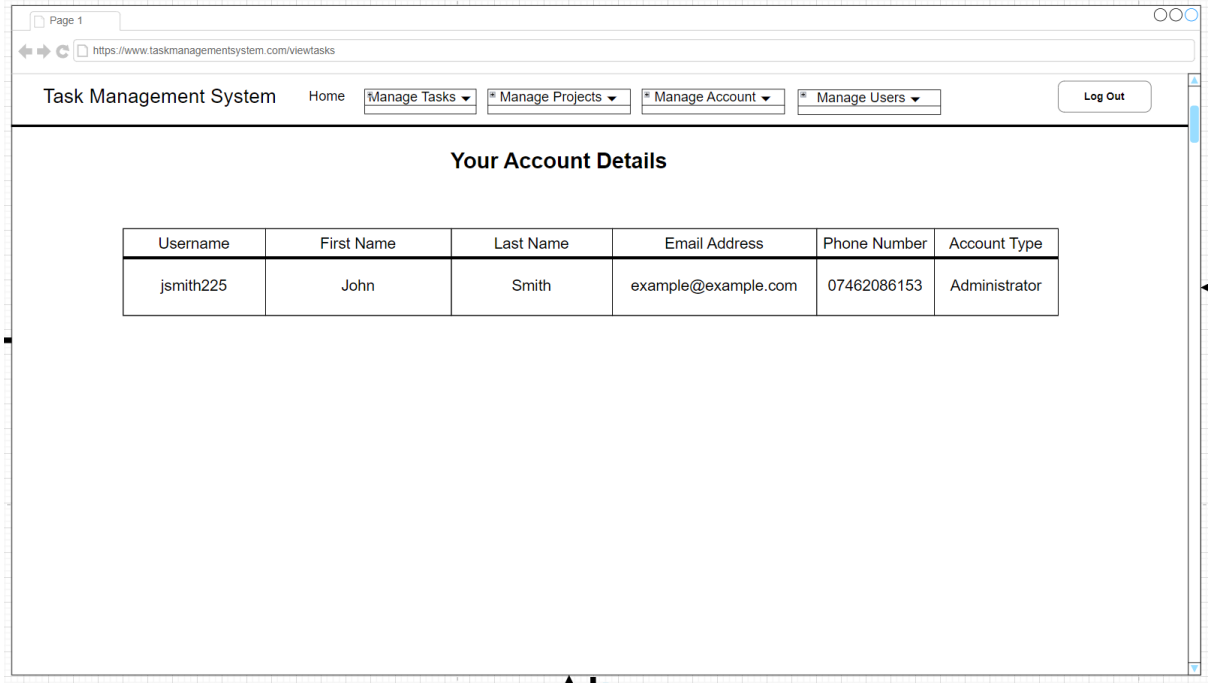


Figure 17 - Account Details Wireframe

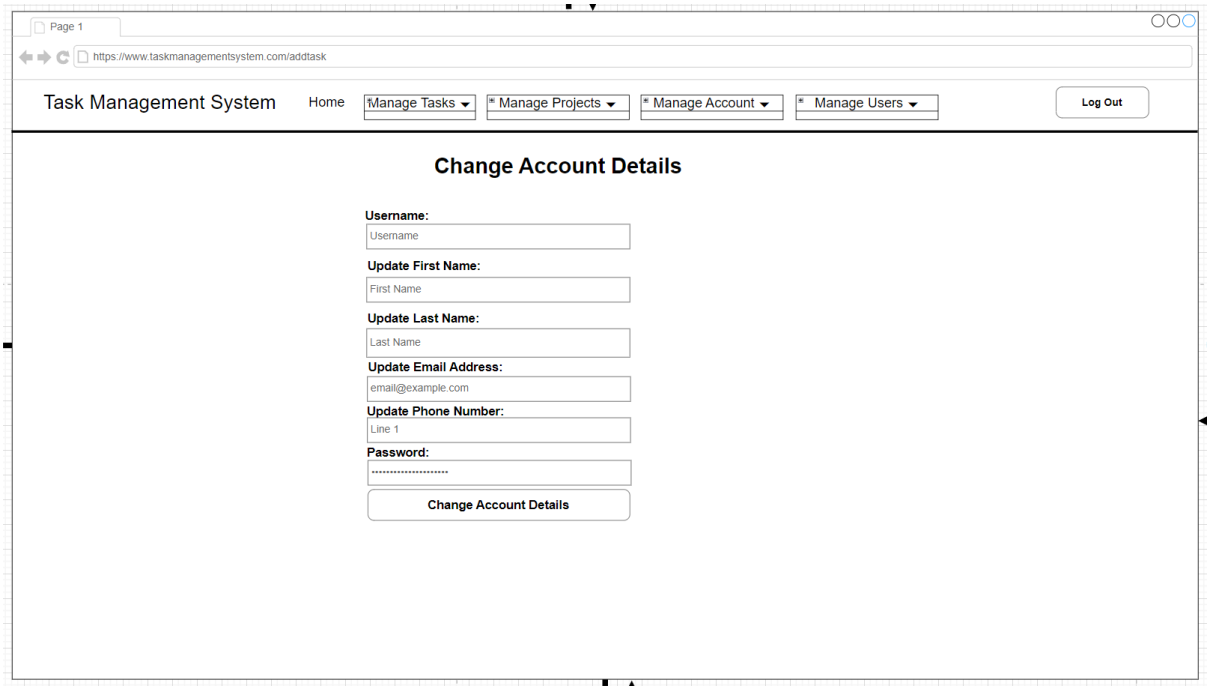


Figure 18 - Change Account details Wireframe

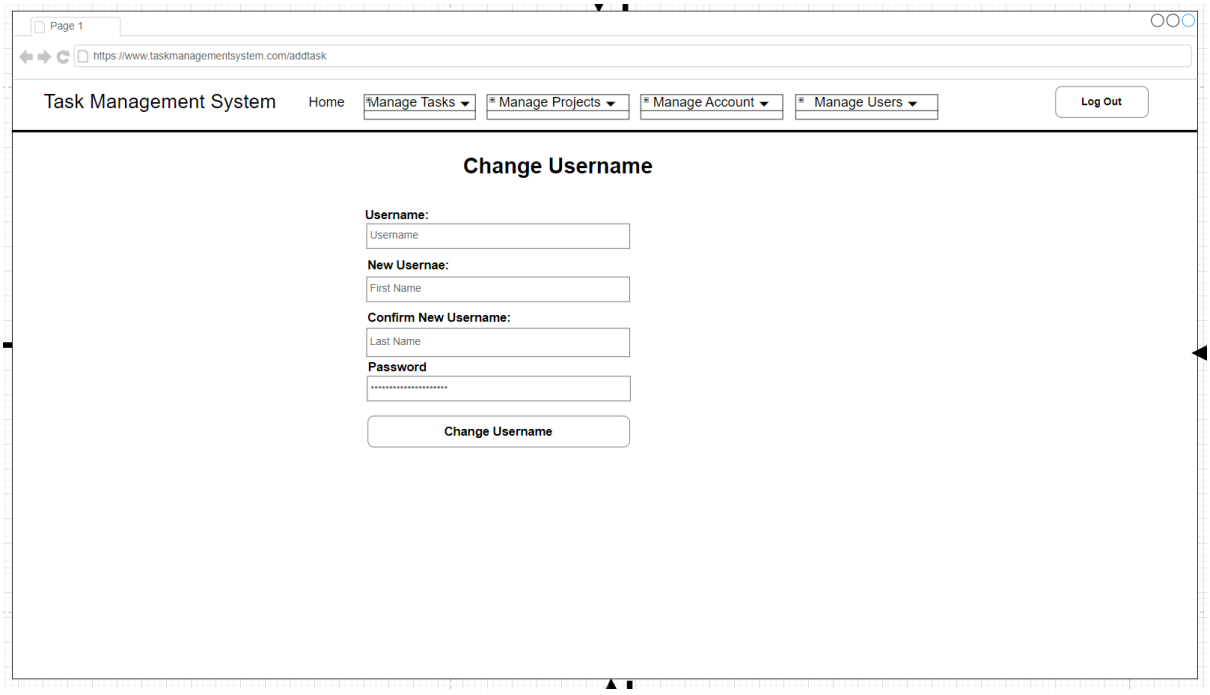


Figure 19 - Change Username Wireframe

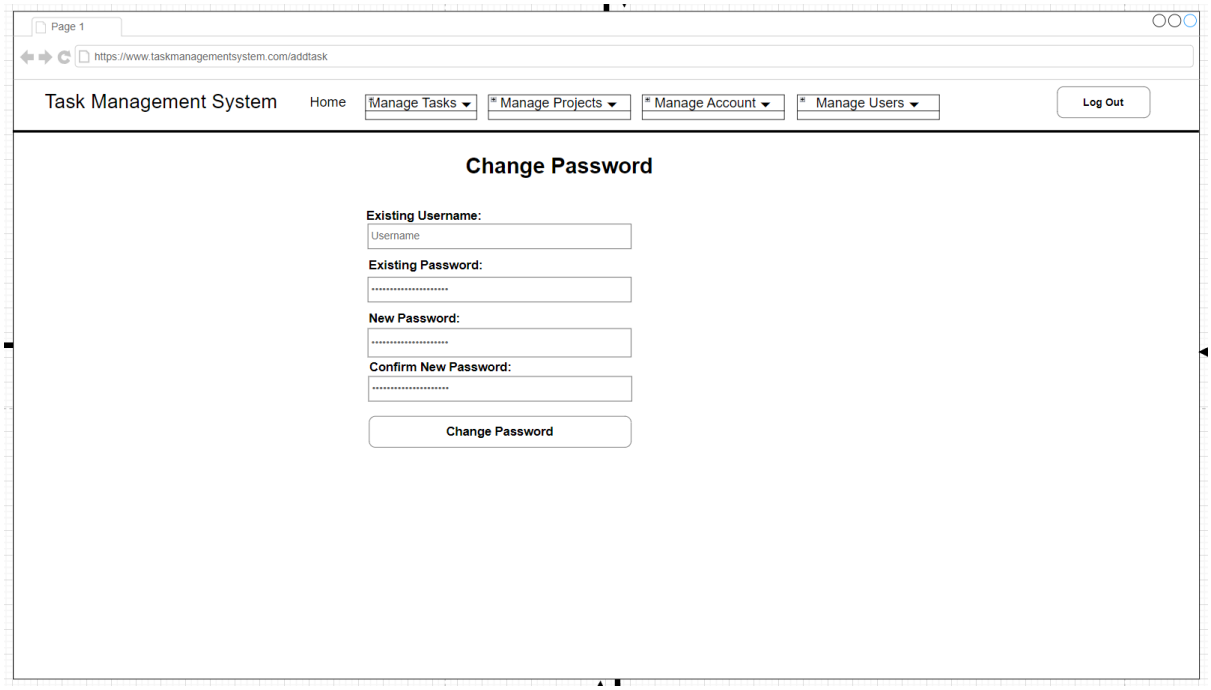


Figure 20 - Change Password Wireframe

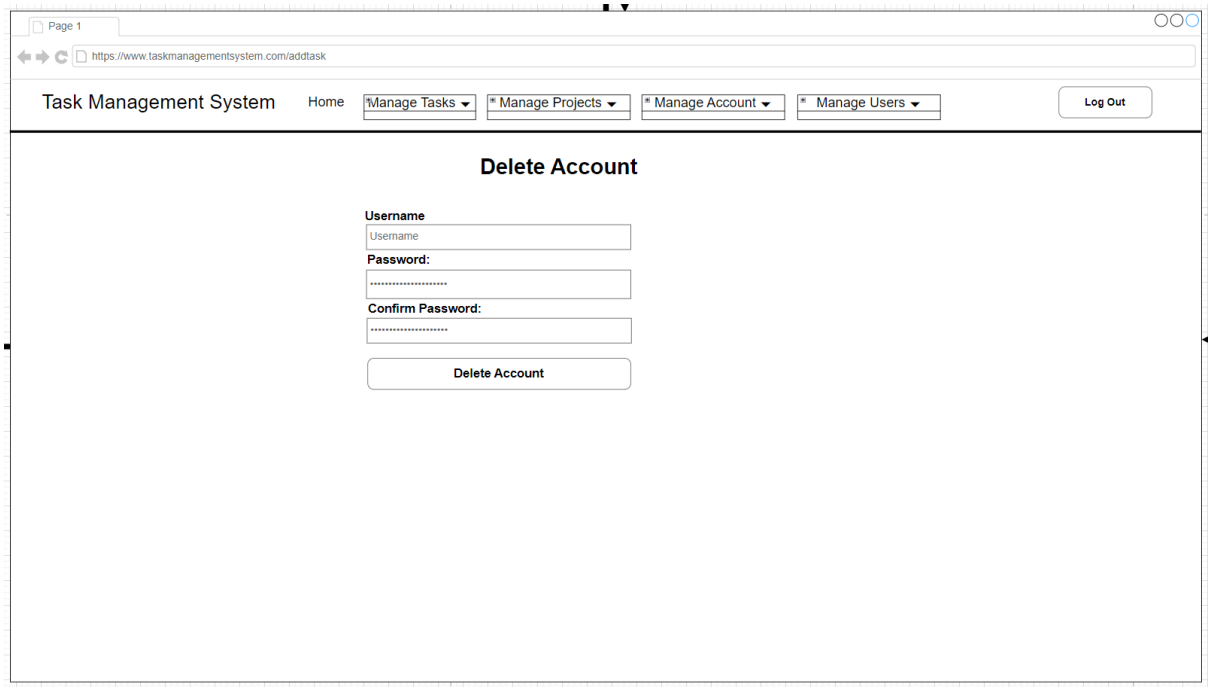


Figure 21 - Delete Account Wireframe

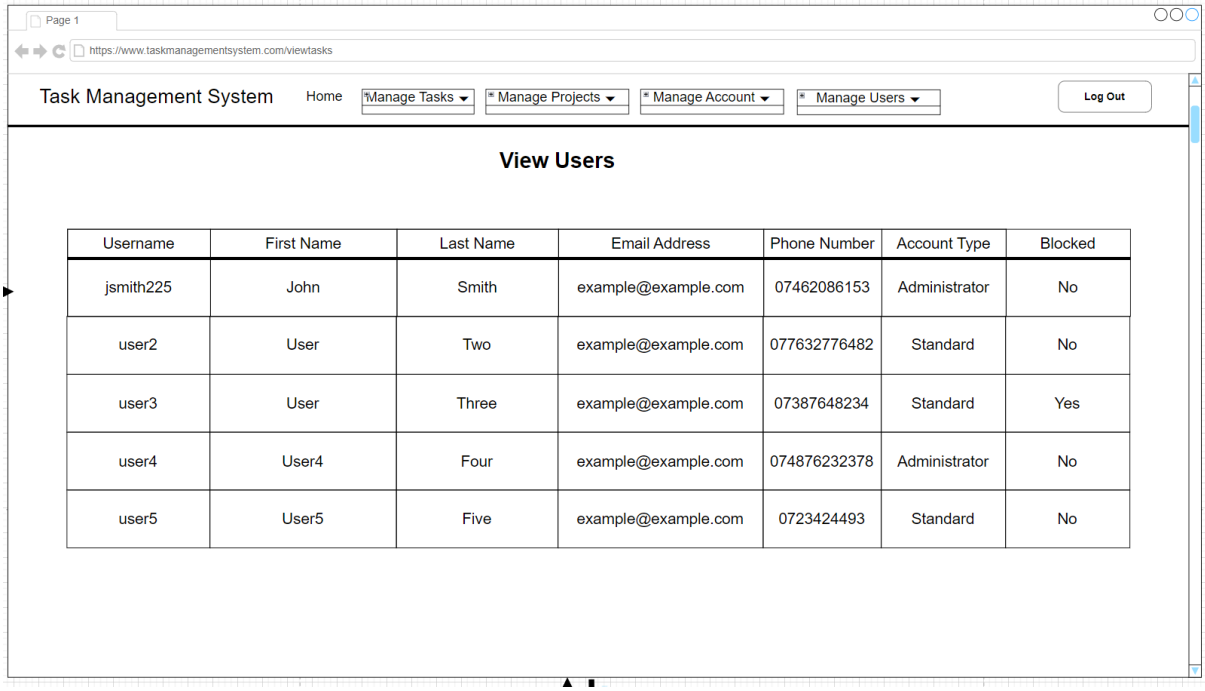


Figure 22 - View Users Wireframe

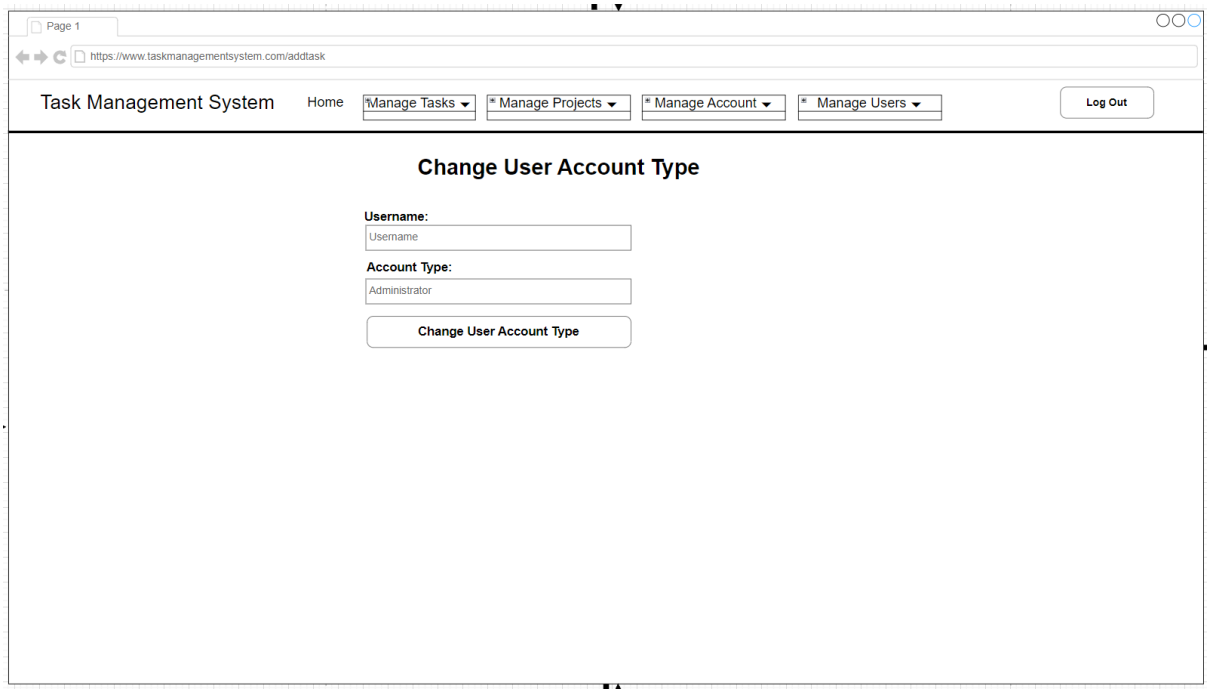


Figure 23 - Change User Account Type Wireframe

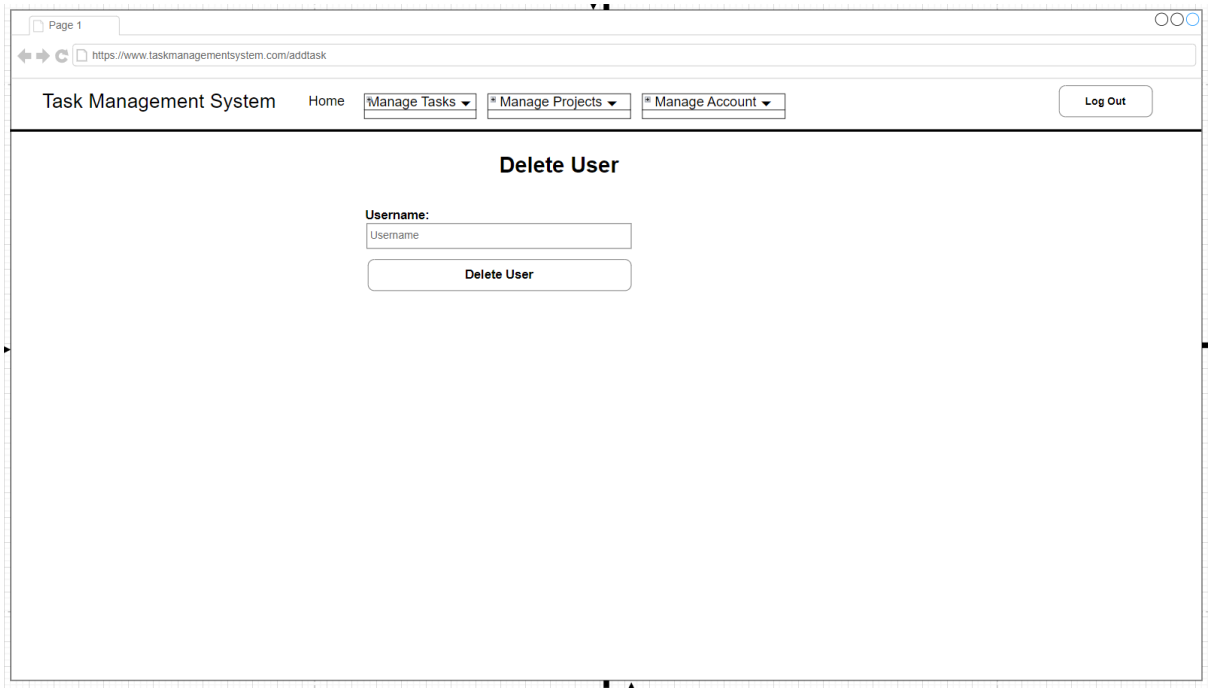


Figure 24 - Delete User Wireframe

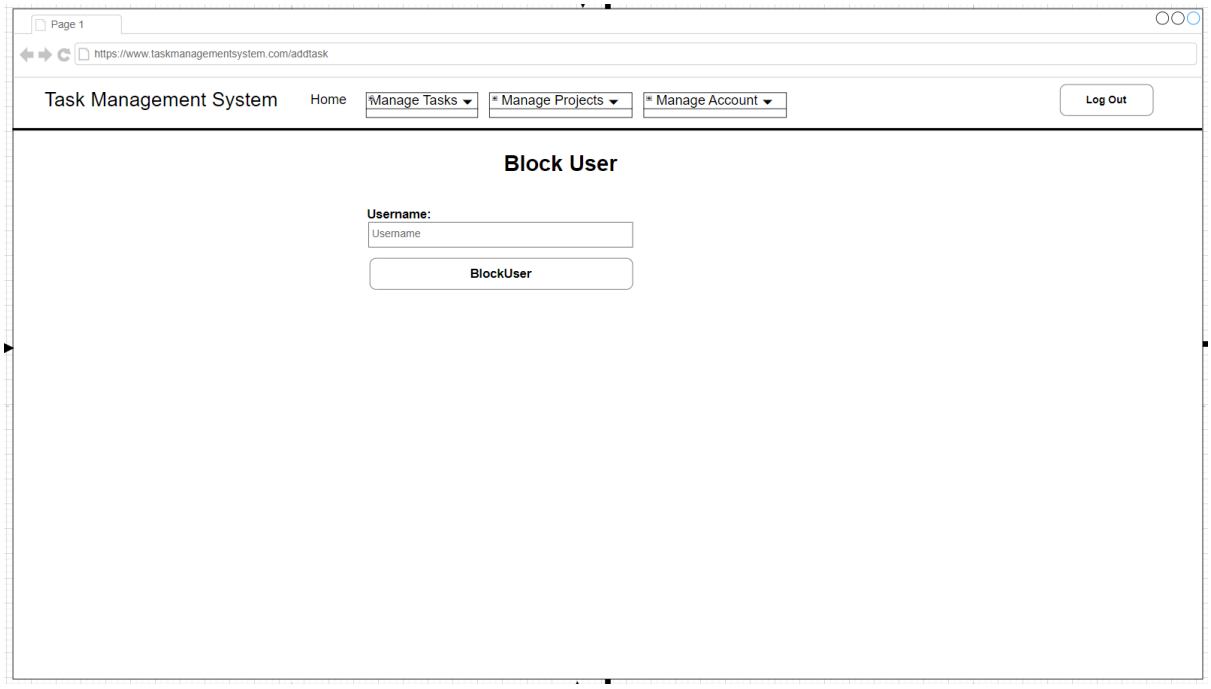


Figure 25 - Block User Wireframe

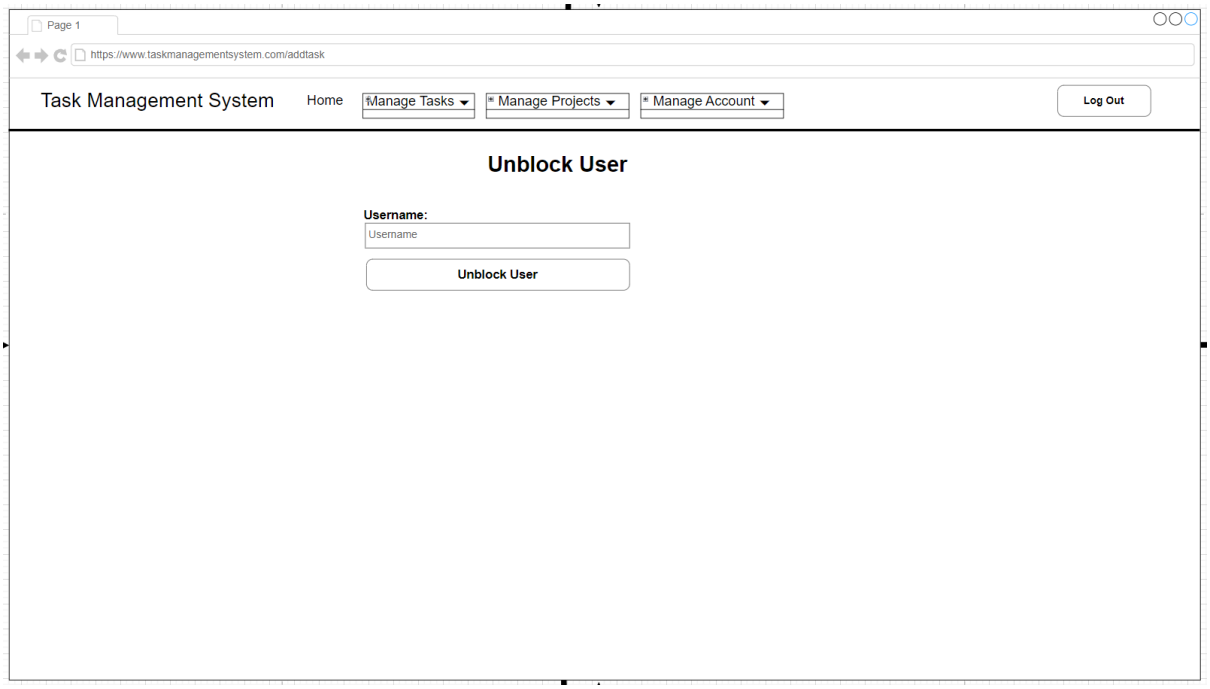


Figure 26 - Unblock User Wireframe

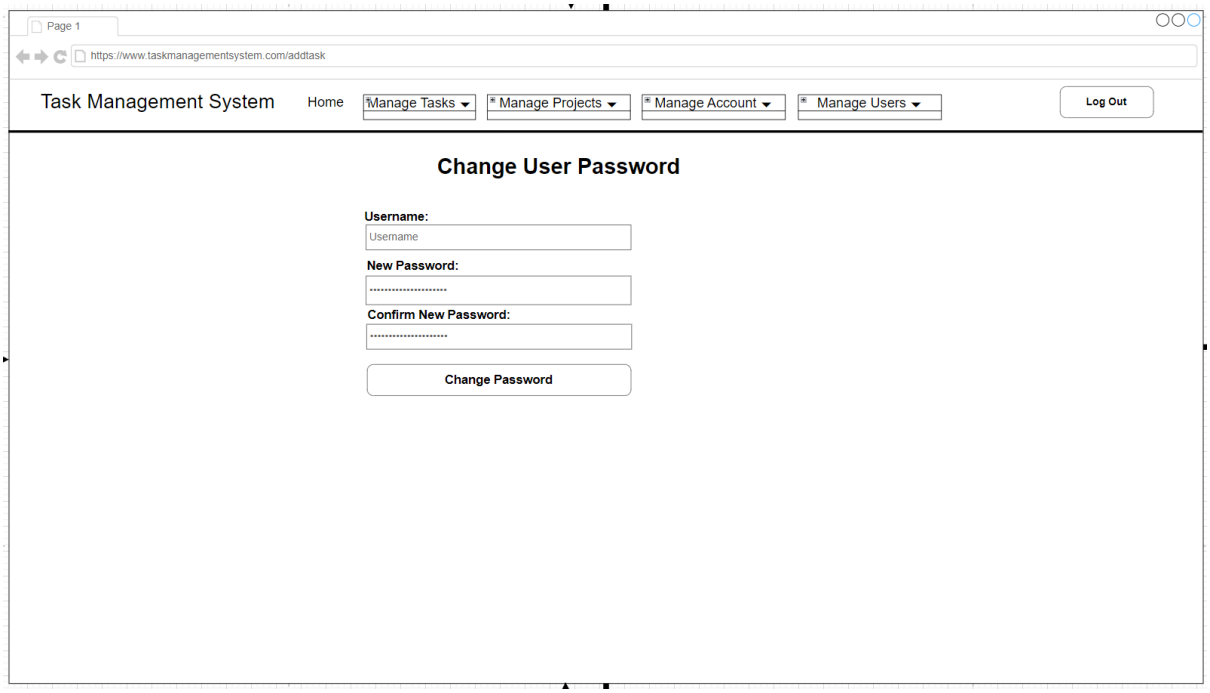


Figure 27 - Change User Password Wireframe

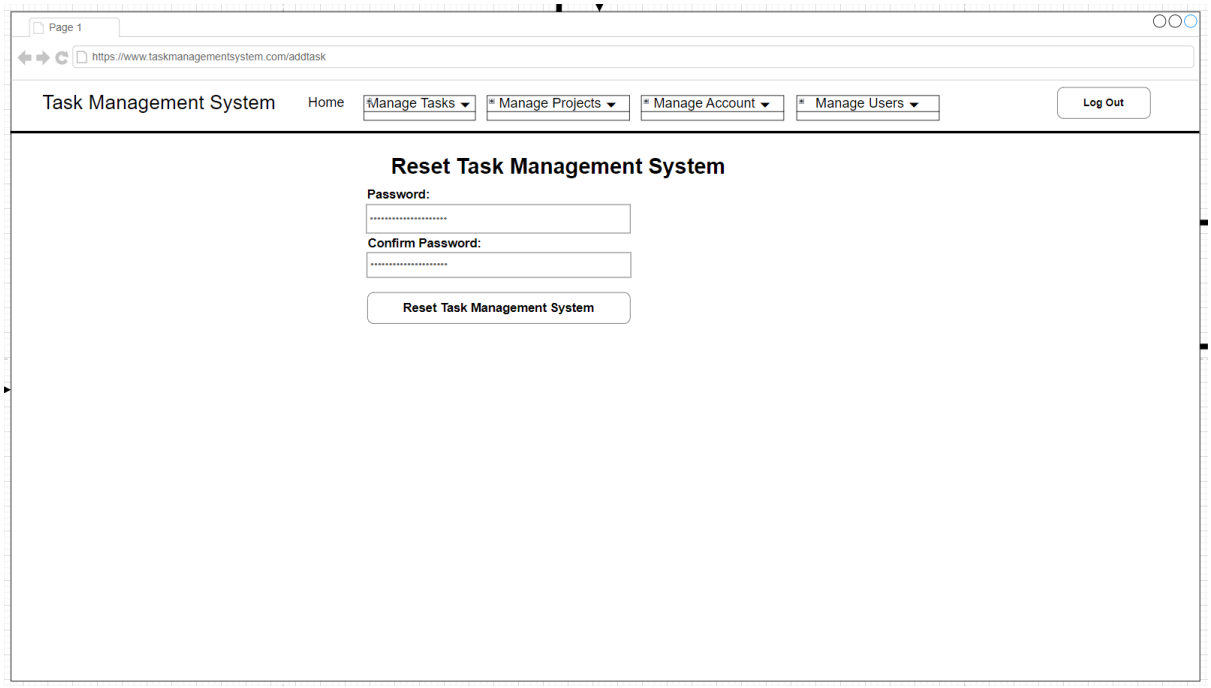


Figure 28 - Factory Reset Wireframe

Mock-up of Application

The mock-ups were created using Adobe Experience Design (Adobe XD), an industry standard software for creating software UI design mock-ups. The mock-up also demonstrates how the individual web pages are linked together for navigation.

Below are screenshots of the Task Management System mock-up:

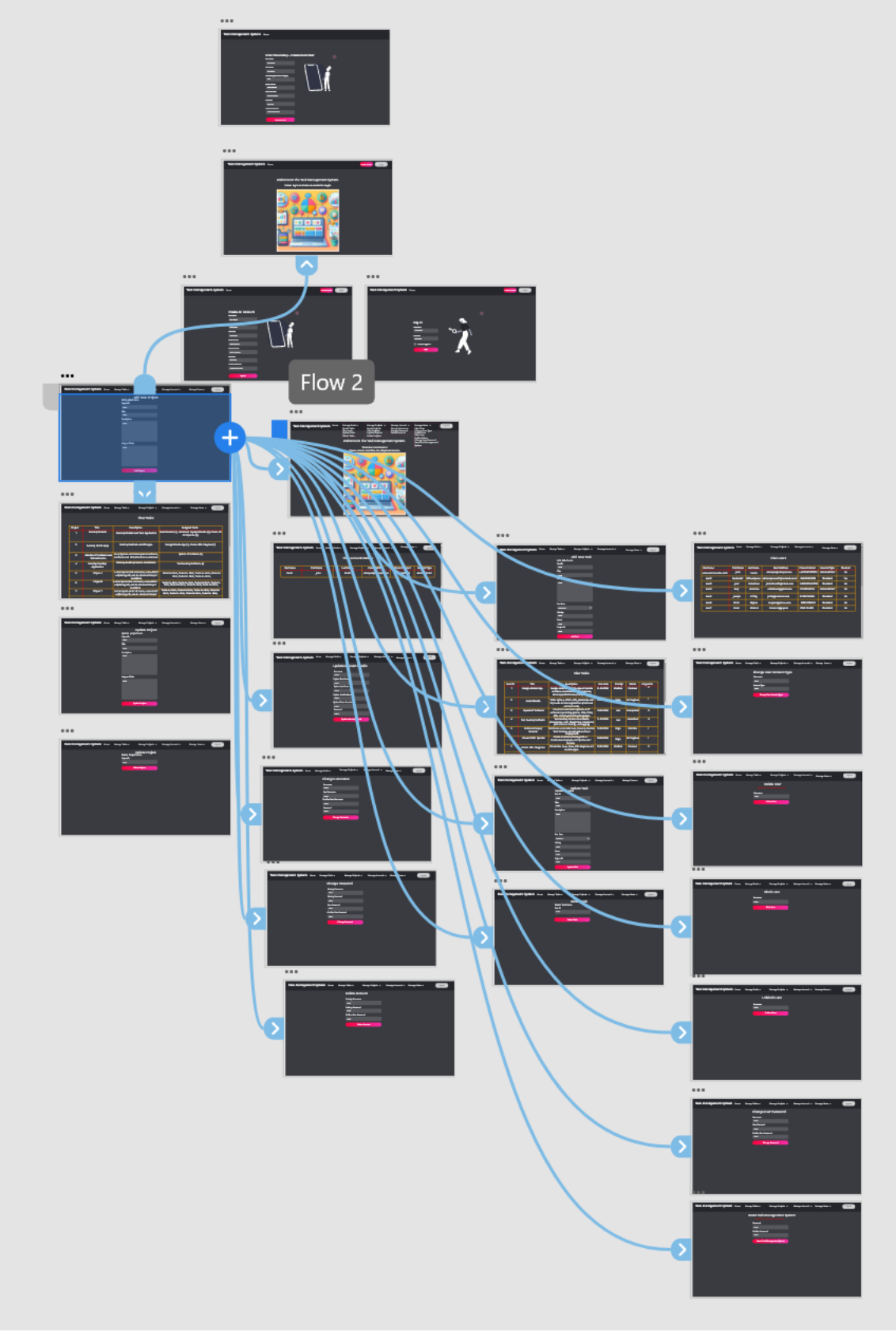


Figure 29 - Overall Mock-up of Task Management System showing how a webpage is linked to other webpages.

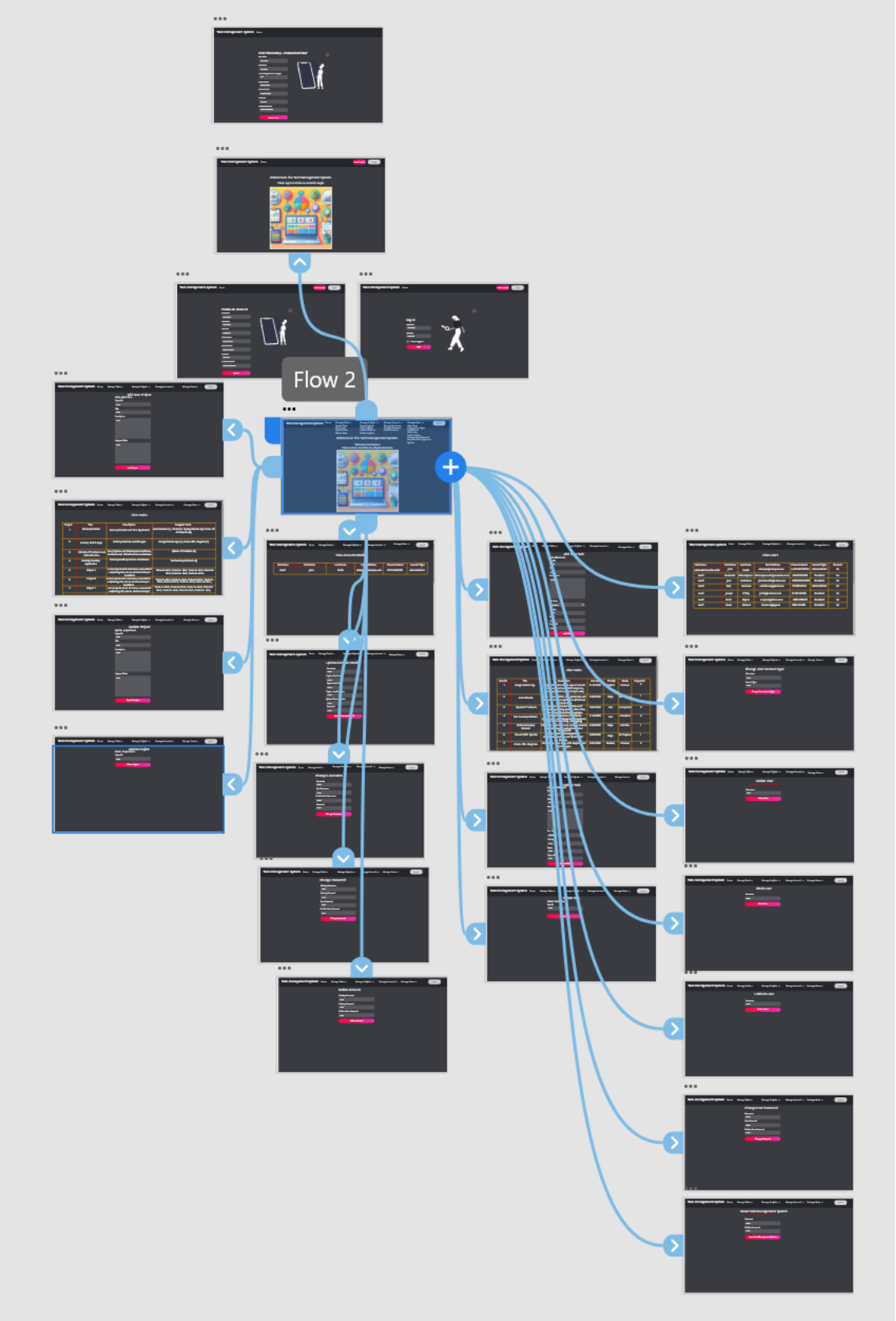


Figure 30 - Overall Mock-up of Task Management System showing how another webpage is linked to other webpages.

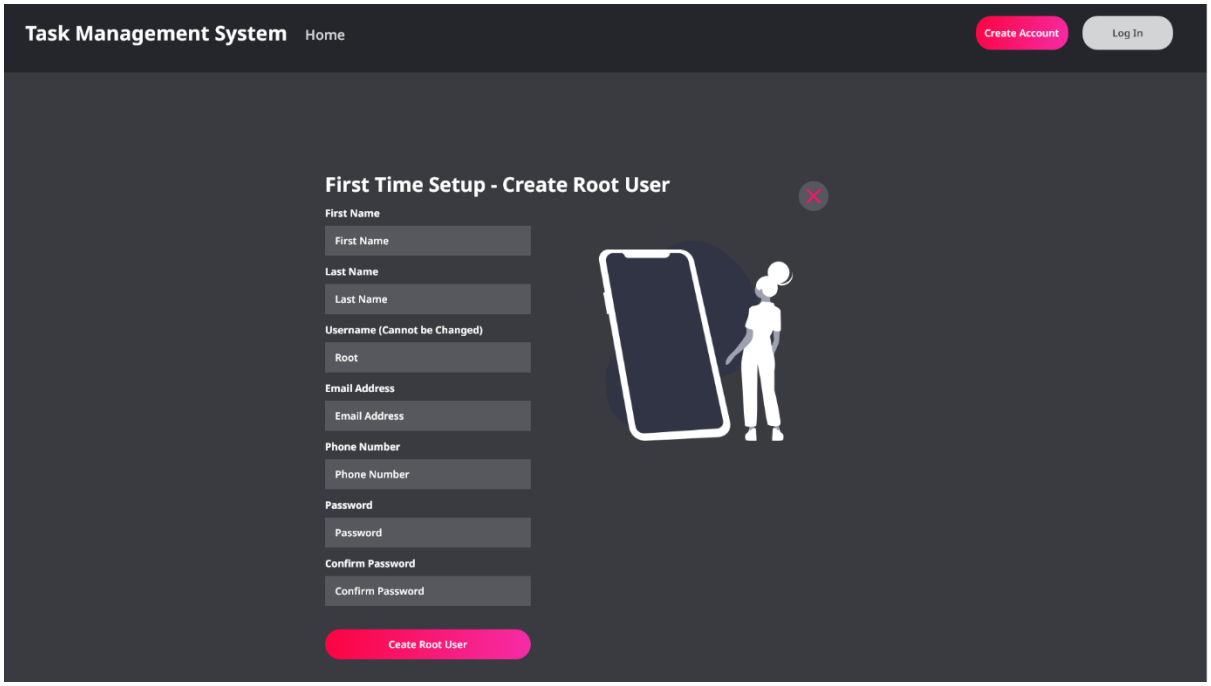


Figure 31 - Setup Page Mock-up.

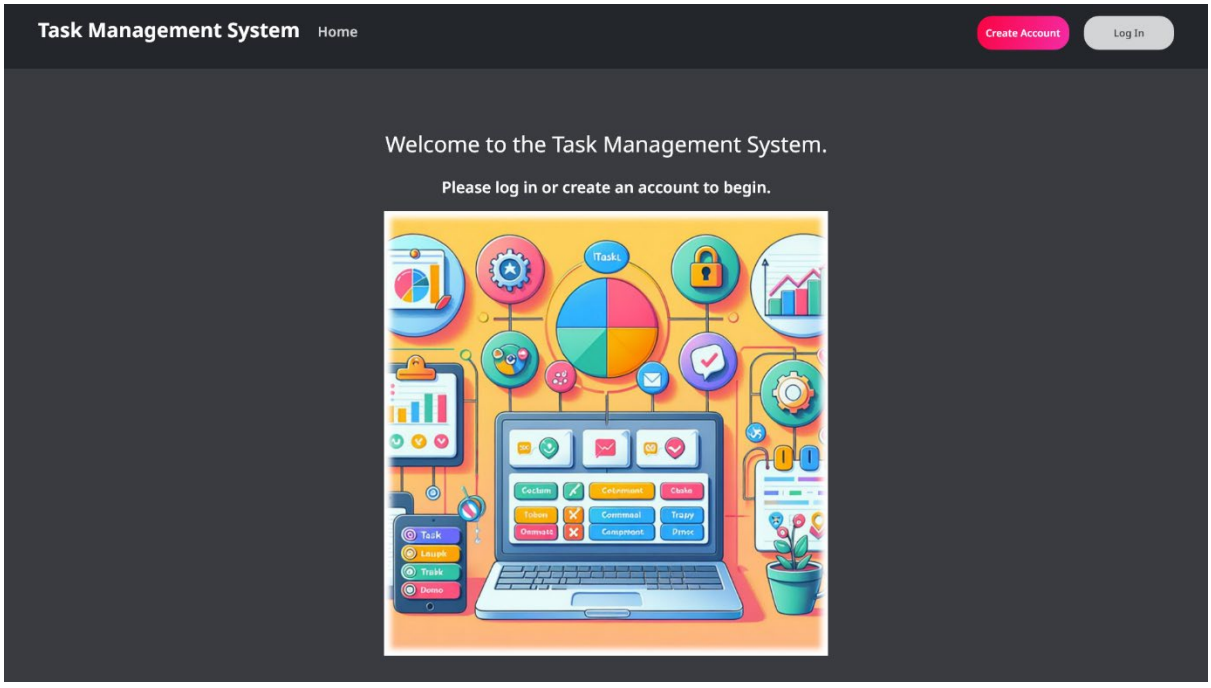


Figure 32 - Landing Page Mock-up.

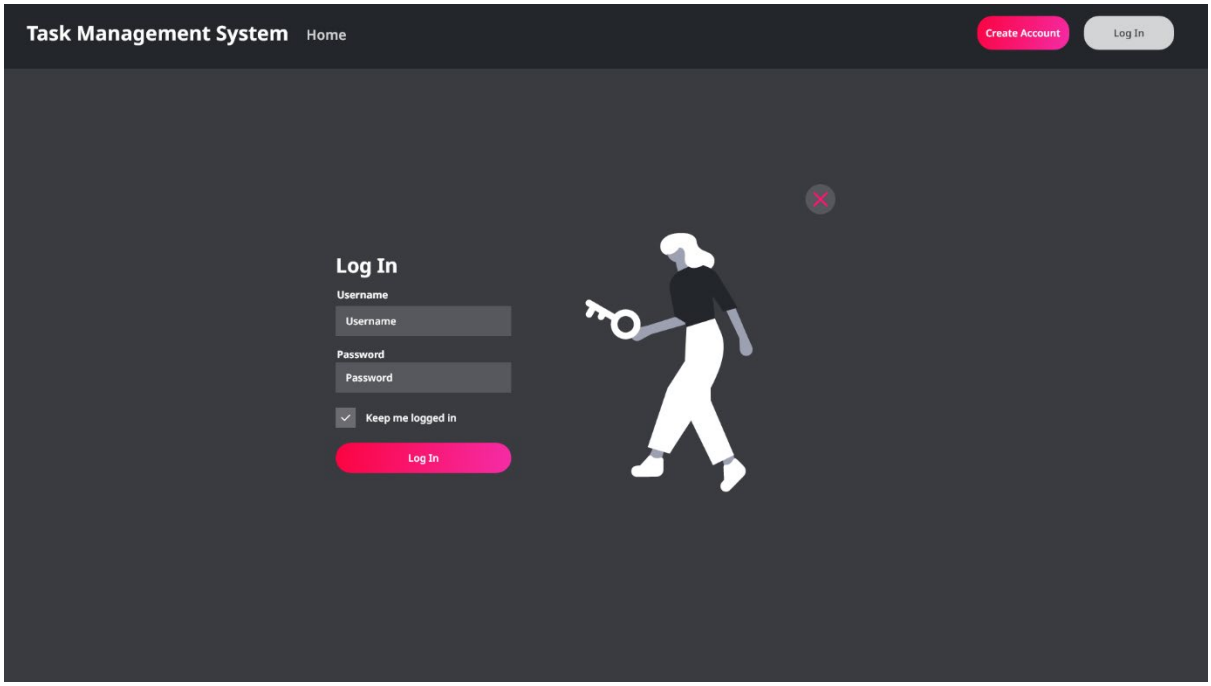


Figure 33 - Login Page Mock-up.

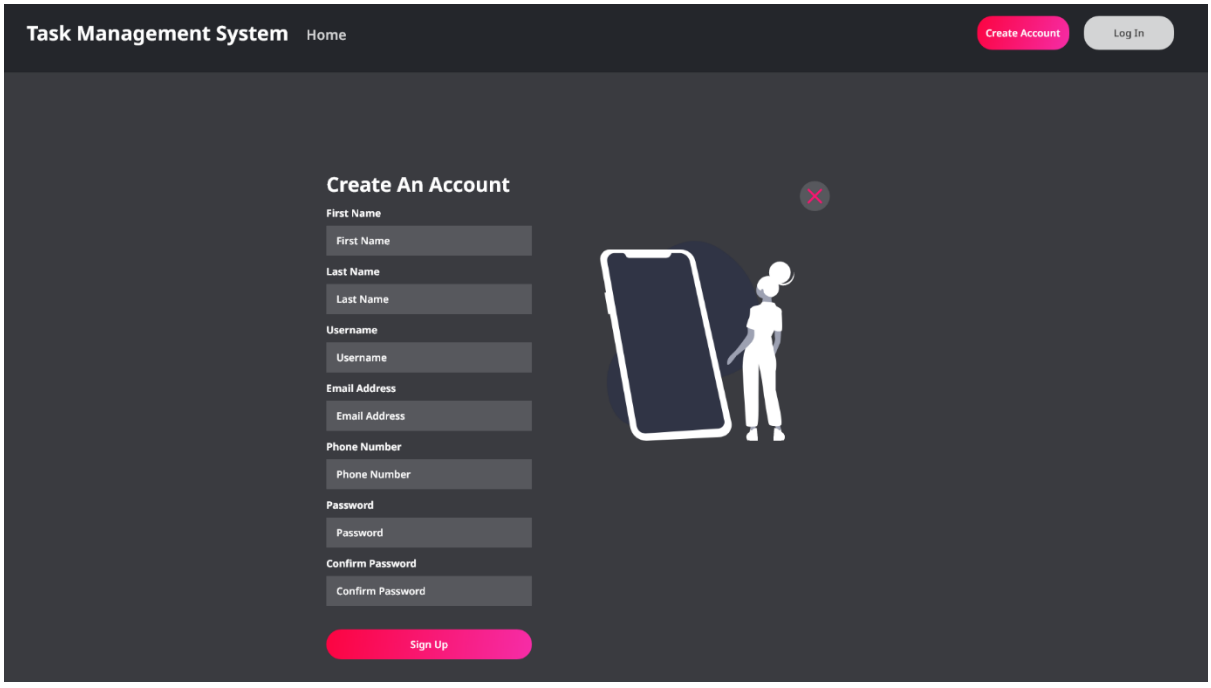


Figure 34 - Create Account Page Mock-up.

Figure 35 - User Home Page Mock-up.

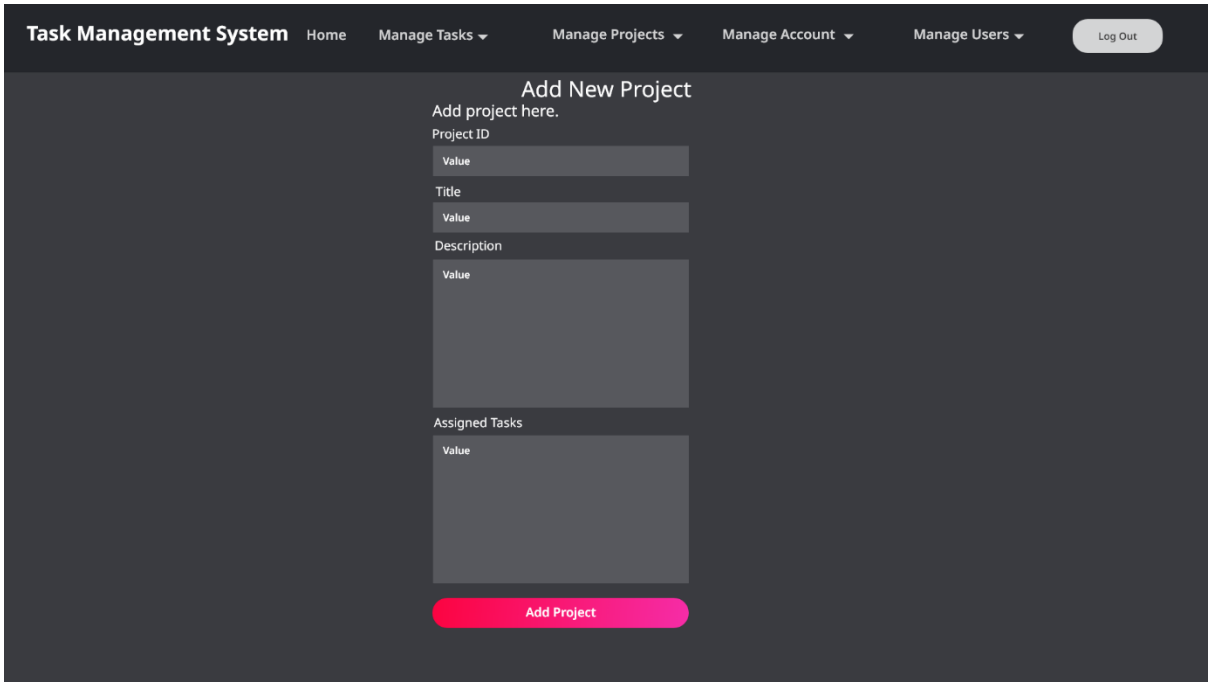


Figure 36 - Add Project Page Mock-up.

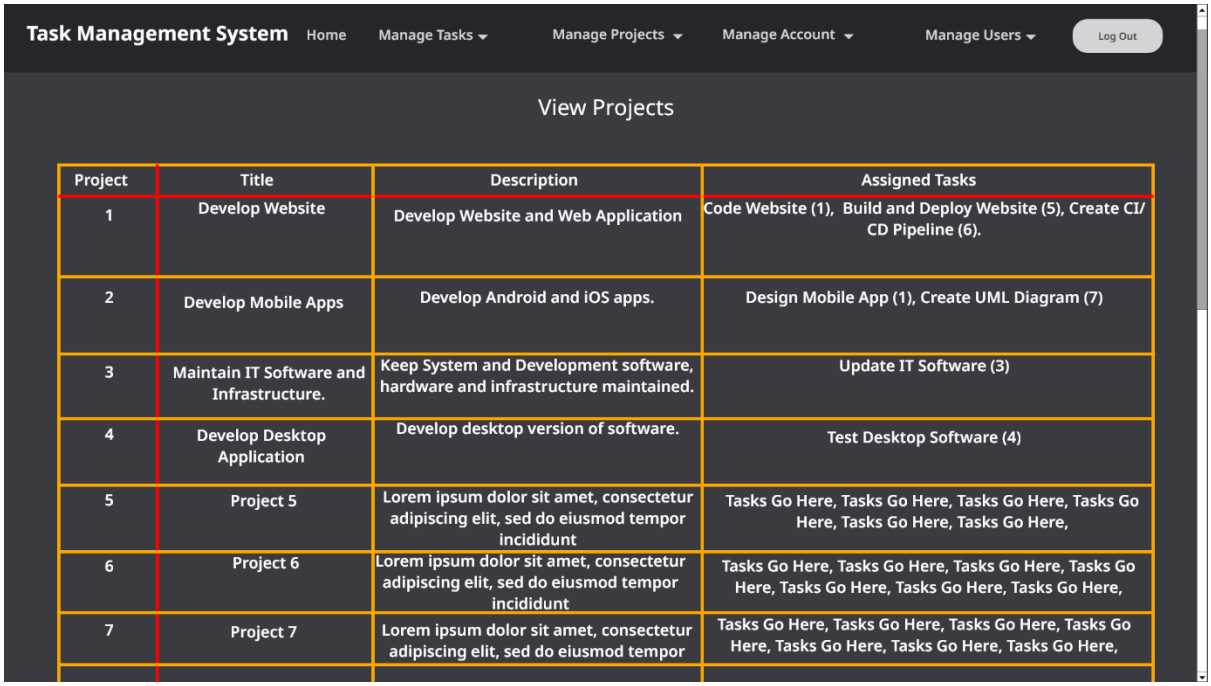


Figure 37 - View Projects Page Mock-up.

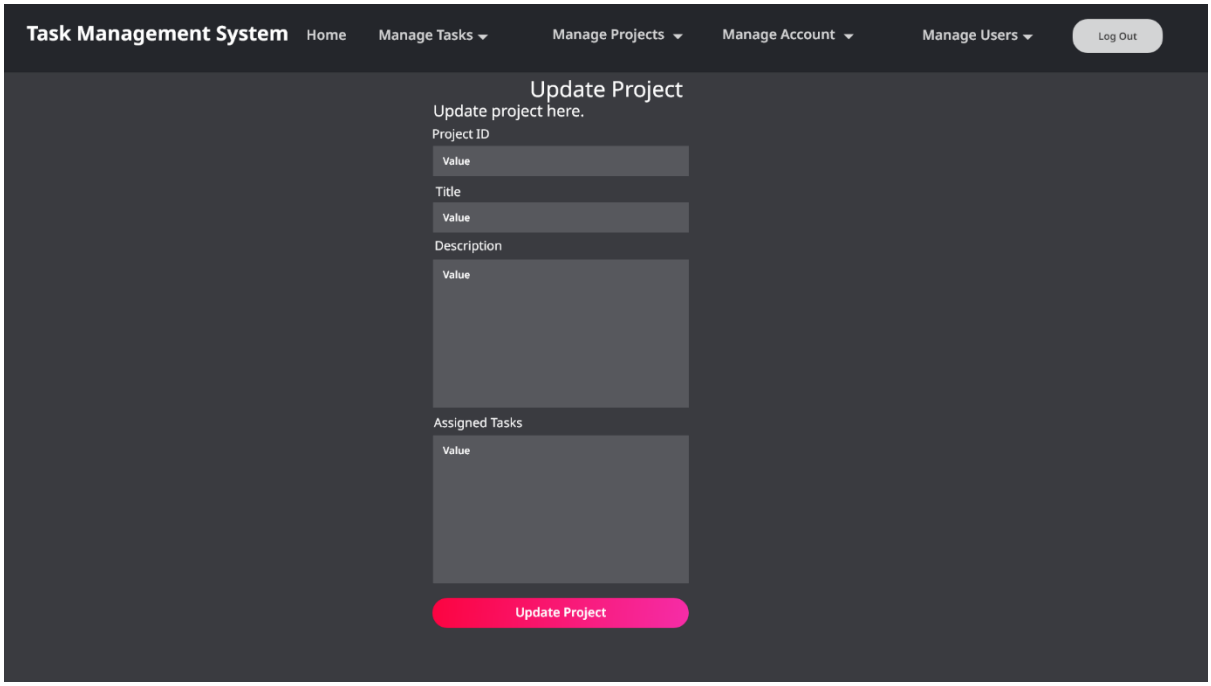


Figure 38 - Update Project Page Mock-up.

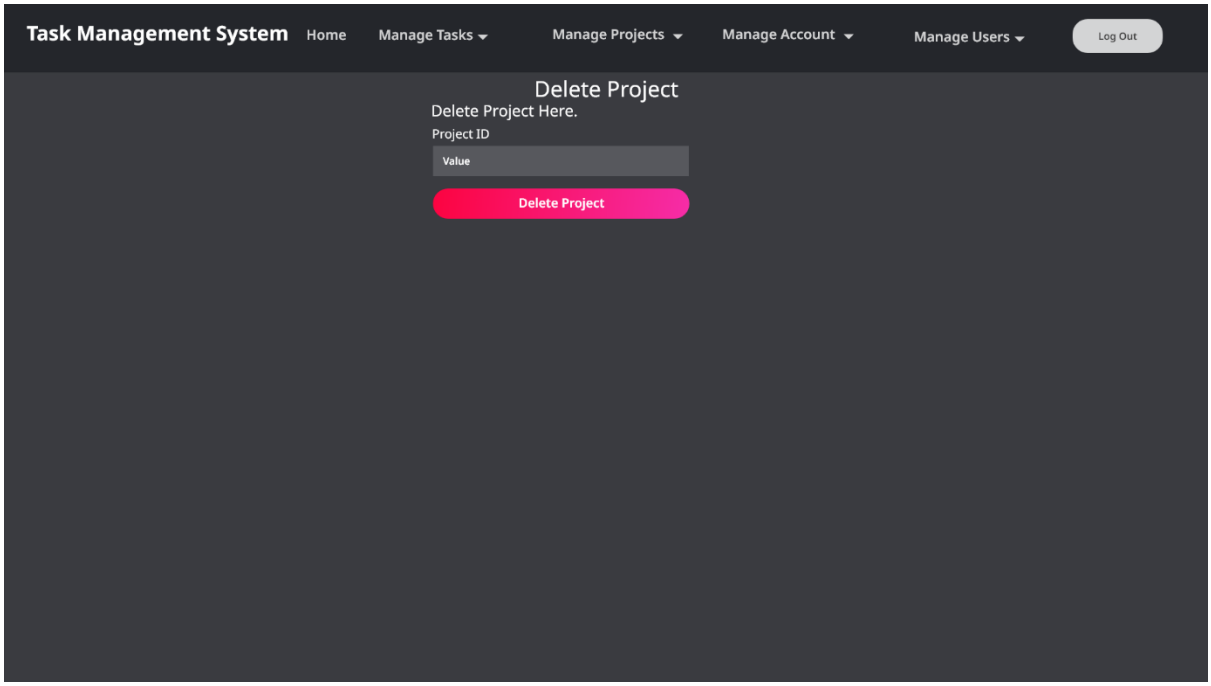


Figure 39 - Delete Project Page Mock-up.

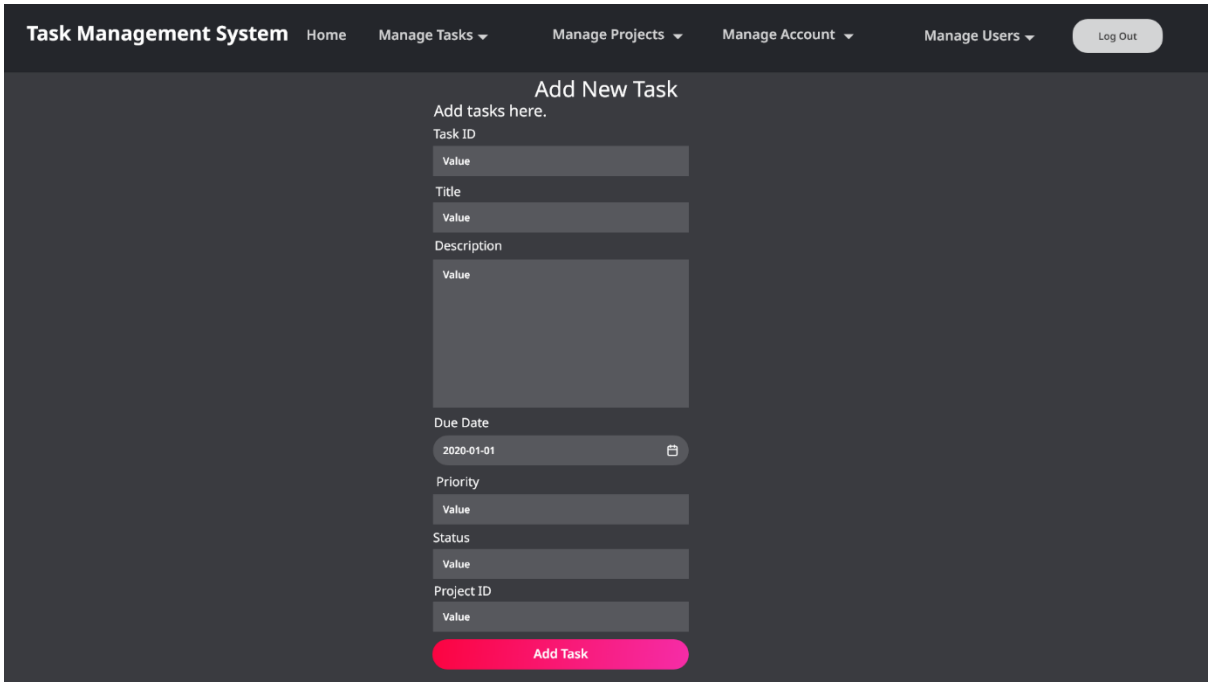


Figure 40 - Add Task Page Mock-up.

Task ID	Title	Description	Due Date	Priority	Status	Project ID
1	Design Mobile App	Design Android and iOS apps for mobile versions of software (UML Diagrams, Mock-ups, Wireframes, Storyboards)	31.05.2024	Medium	Planned	2
2	Code Website	Write Python, HTML, CSS, JavaScript and SQL code for web application (front end and back end).	15.08.2024	High	In Progress	1
3	Update IT Software	Check for and install updates to IT software (operating system, IDEs, SDKs, APIs and programming languages	10.02.2024	Low	Completed	3
4	Test Desktop Software	Test desktop version of software (Acceptance, unit, integration, functional, performance testing , debugging)	11.03.2024	Low	Cancelled	4
5	Build and Deploy Website	Build web on Render.com, Docker, Amazon Web Services etc using Blue-Green Deployment	25.08.2024	High	Overdue	1
6	Create CI/CD Pipeline	Create Continuous Integration / Continuous Deployment Pipelines for Website	20.08.2024	High	In Progress	1
7	Create UML Diagrams	Create Use Case, Class, ERD diagrams for mobile apps.	22.05.2024	Medium	Planned	2

Figure 41 - View Tasks Page Mock-up.

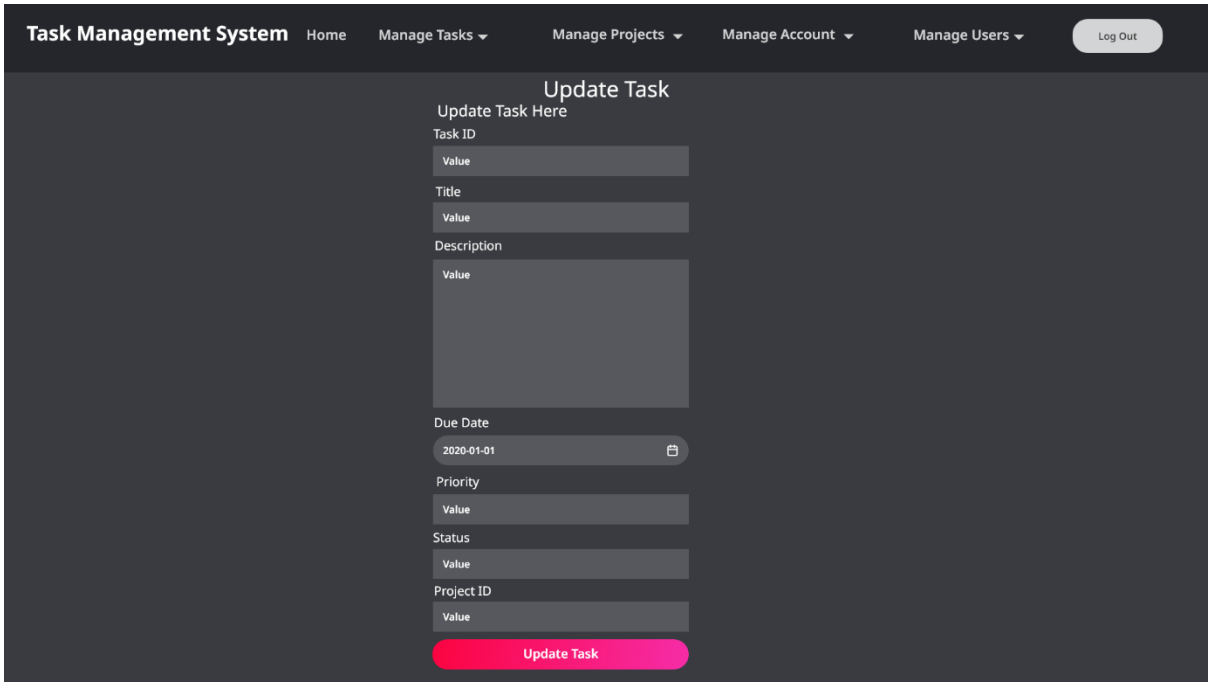


Figure 42 - Update Task Page Mock-up.

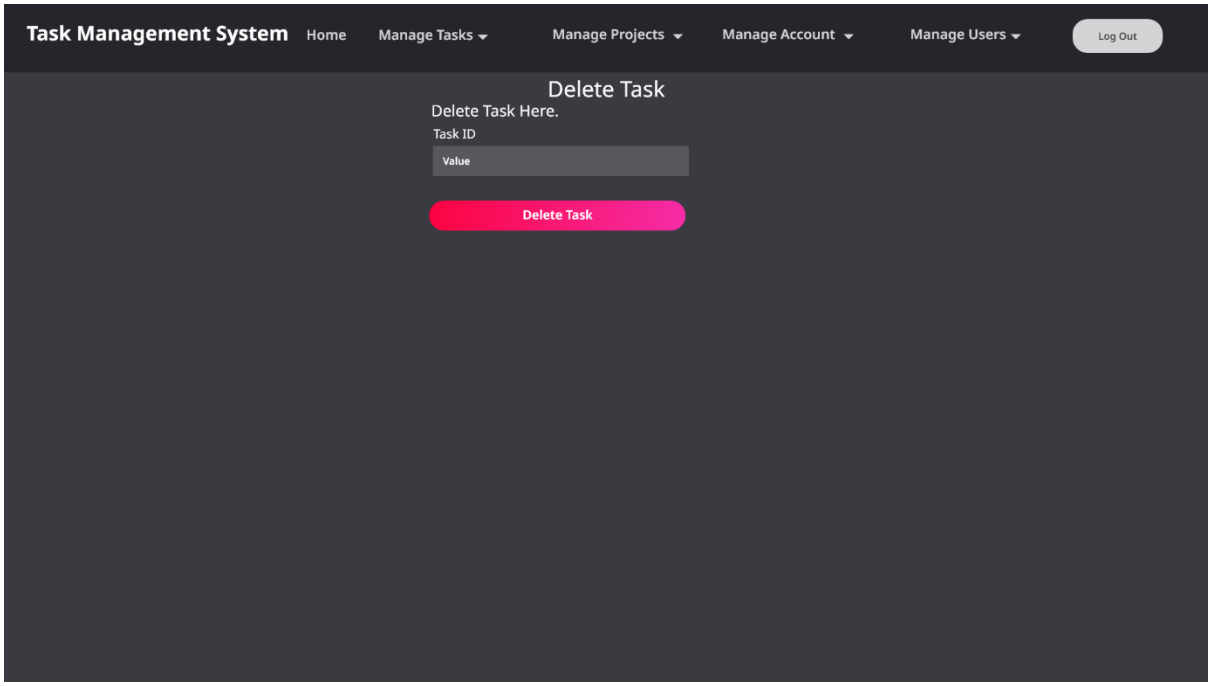


Figure 43 - Delete Task Page Mock-up.

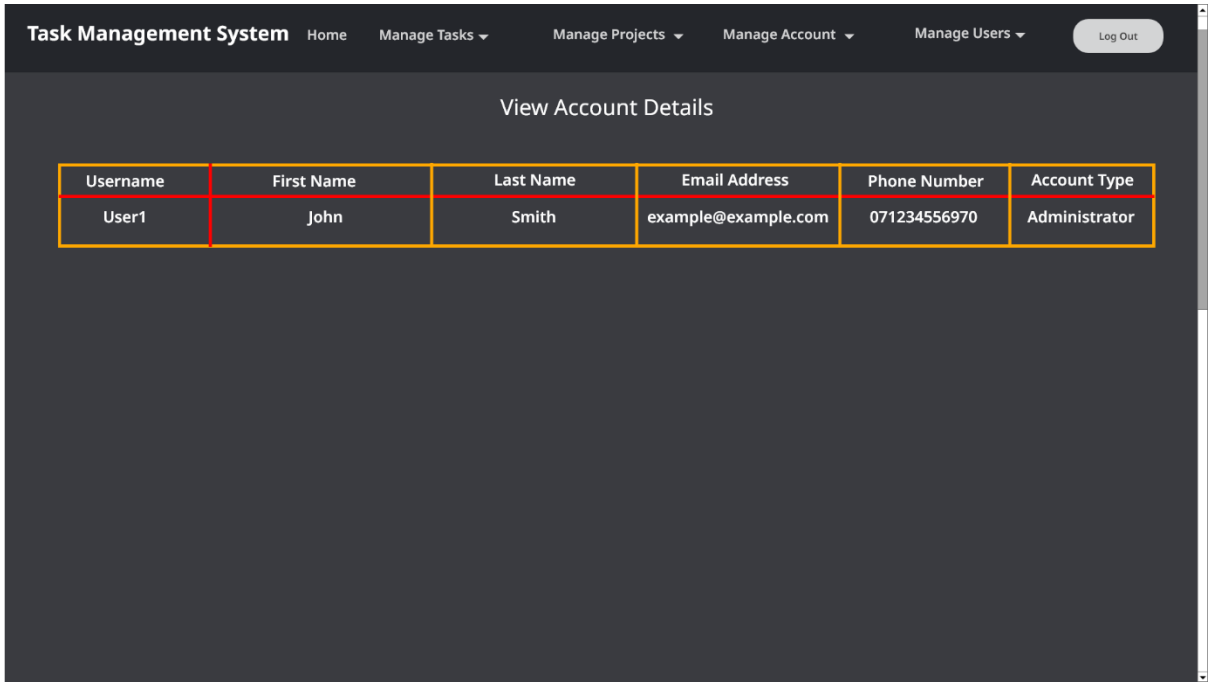


Figure 44 - View Account Details Page Mock-up.

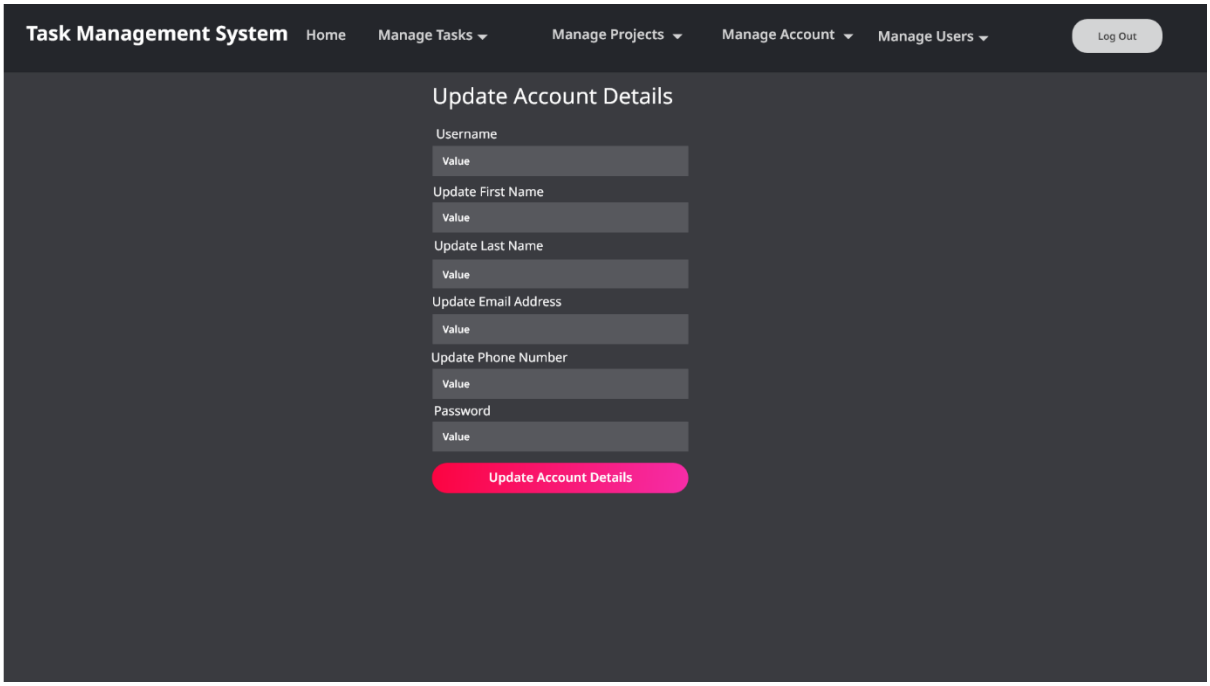


Figure 45 - Update Account Details Page Mock-up.

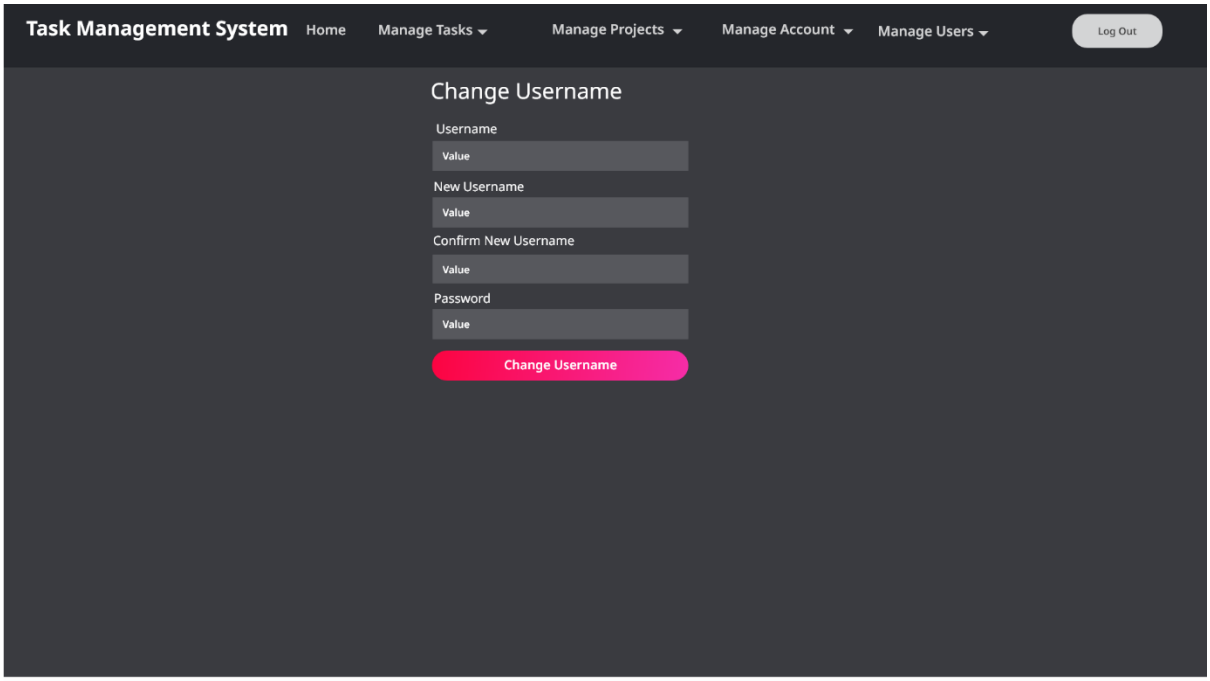


Figure 46 - Change Username Page Mock-up.

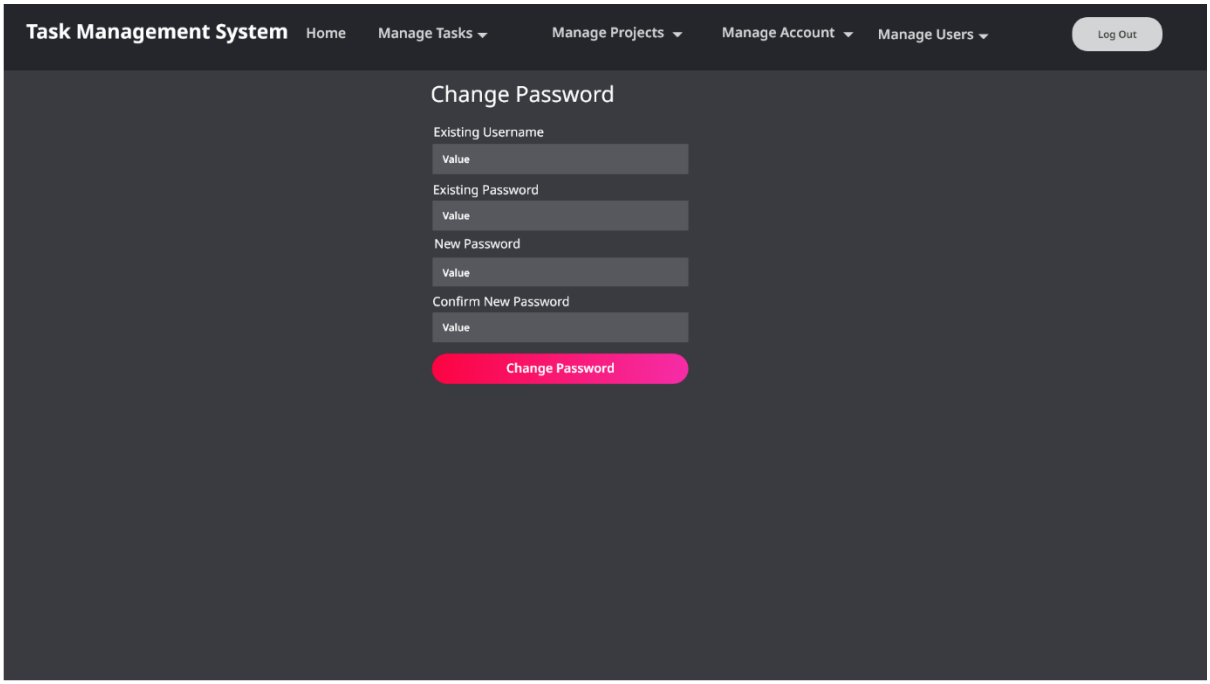


Figure 47 - Change Password Page Mock-up.

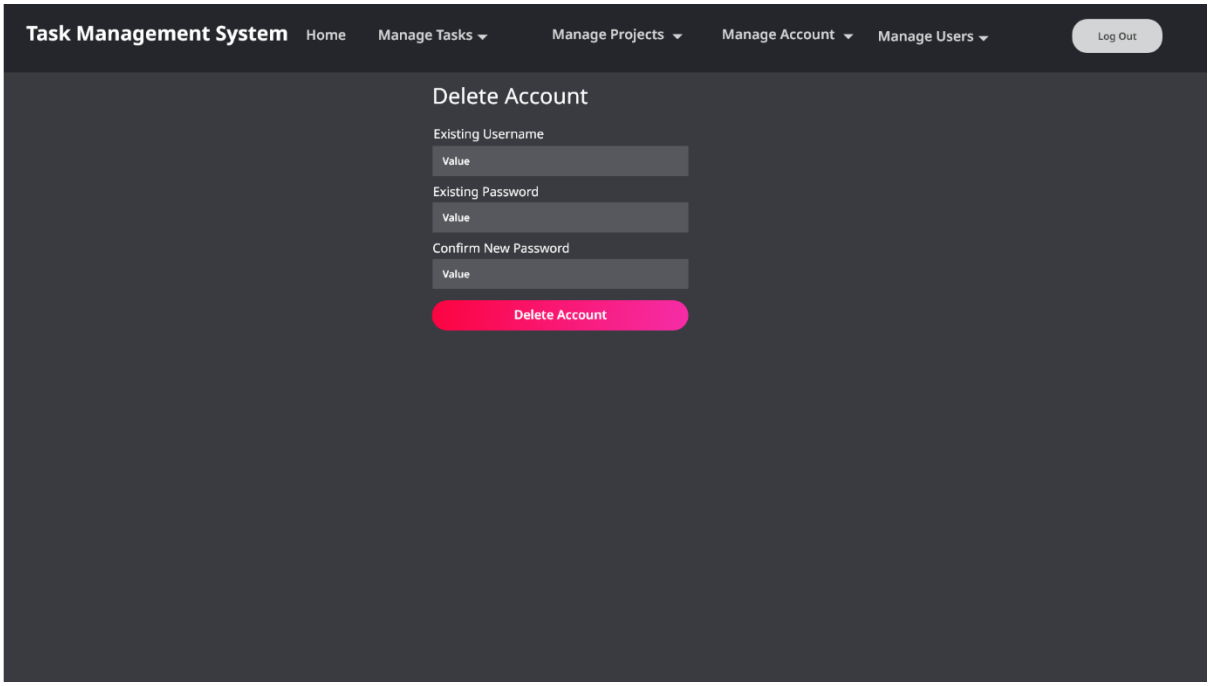
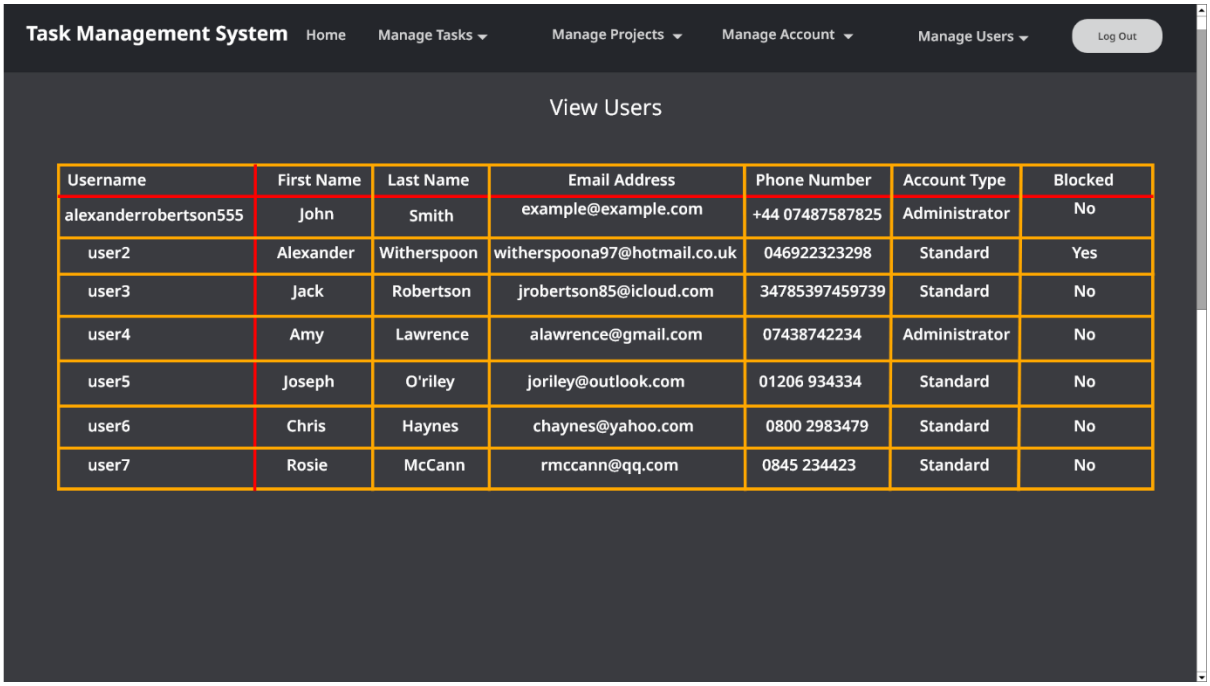


Figure 48 - Delete Account Page Mock-up.



Username	First Name	Last Name	Email Address	Phone Number	Account Type	Blocked
alexanderrobertson555	John	Smith	example@example.com	+44 07487587825	Administrator	No
user2	Alexander	Witherspoon	witherspoona97@hotmail.co.uk	046922323298	Standard	Yes
user3	Jack	Robertson	jrobertson85@icloud.com	34785397459739	Standard	No
user4	Amy	Lawrence	alawrence@gmail.com	07438742234	Administrator	No
user5	Joseph	O'riley	joriley@outlook.com	01206 934334	Standard	No
user6	Chris	Haynes	chaynes@yahoo.com	0800 2983479	Standard	No
user7	Rosie	McCann	rmccann@qq.com	0845 234423	Standard	No

Figure 49 - View Users Page Mock-up.

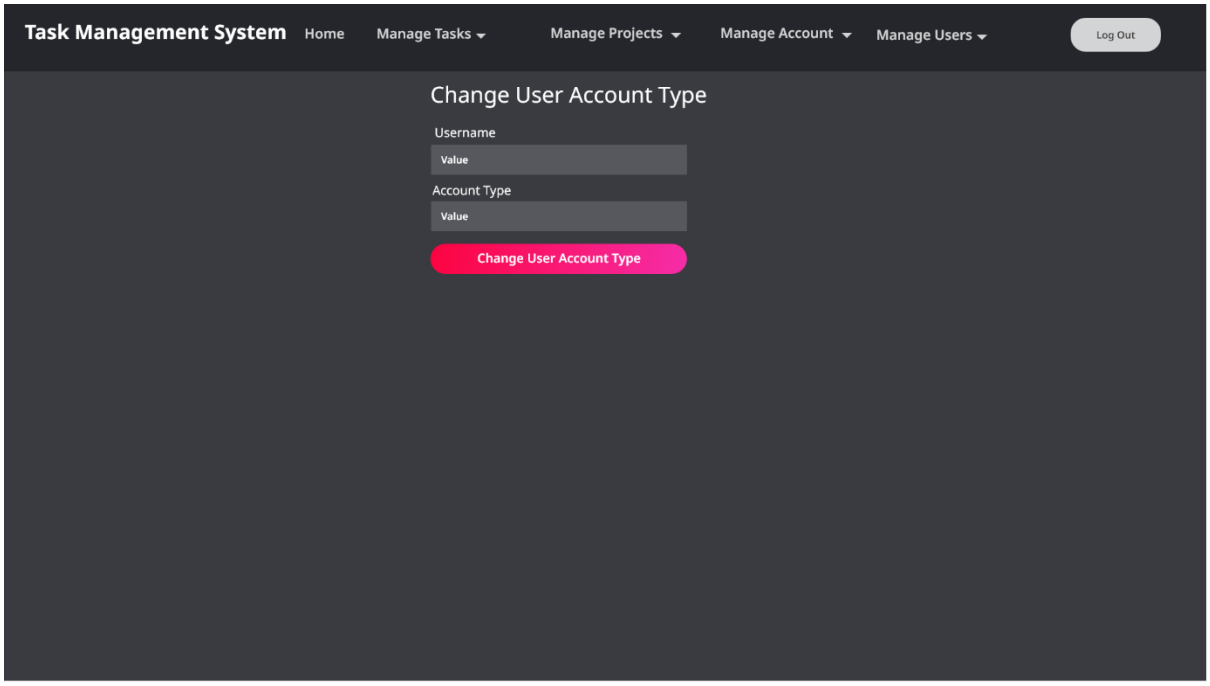


Figure 50 - Change User Account Type Page Mock-up.

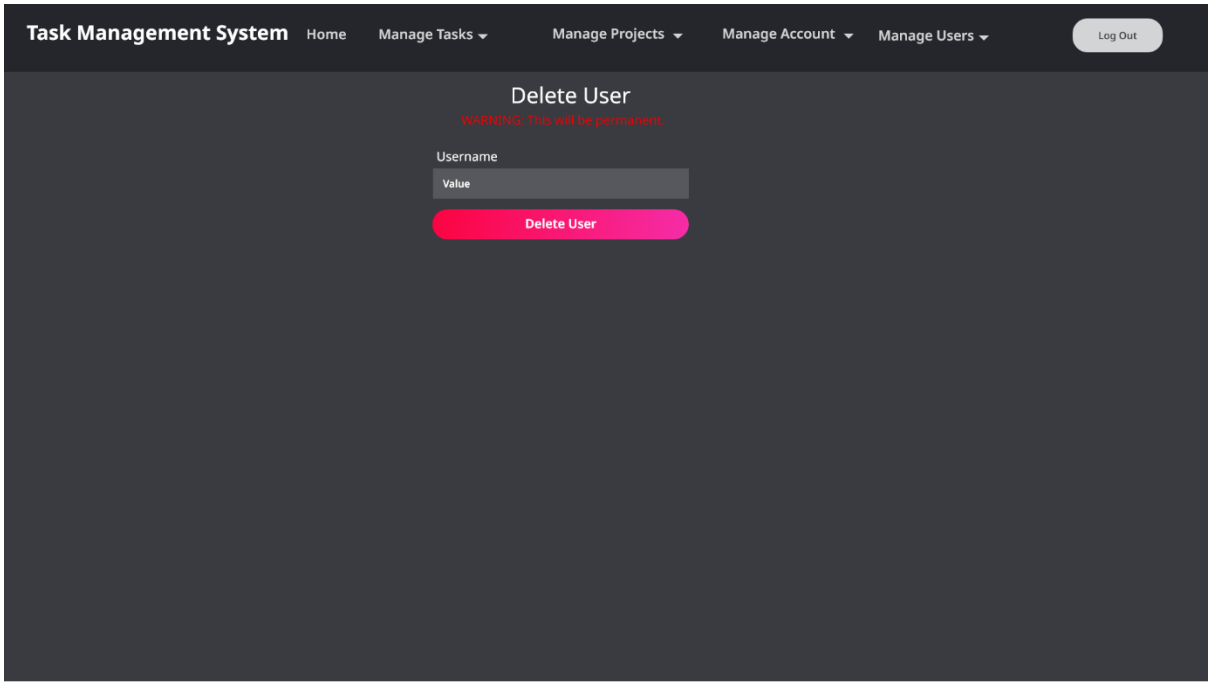


Figure 51 - Delete User Page Mock-up.

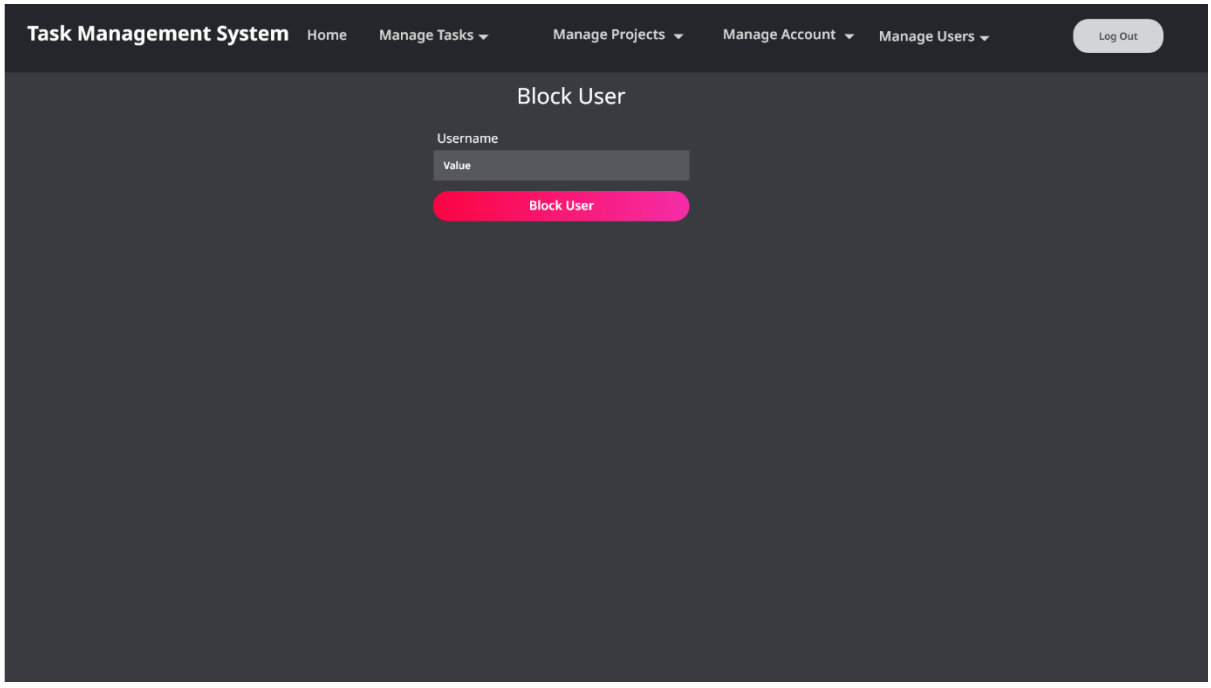


Figure 52 - Block User Page Mock-up.

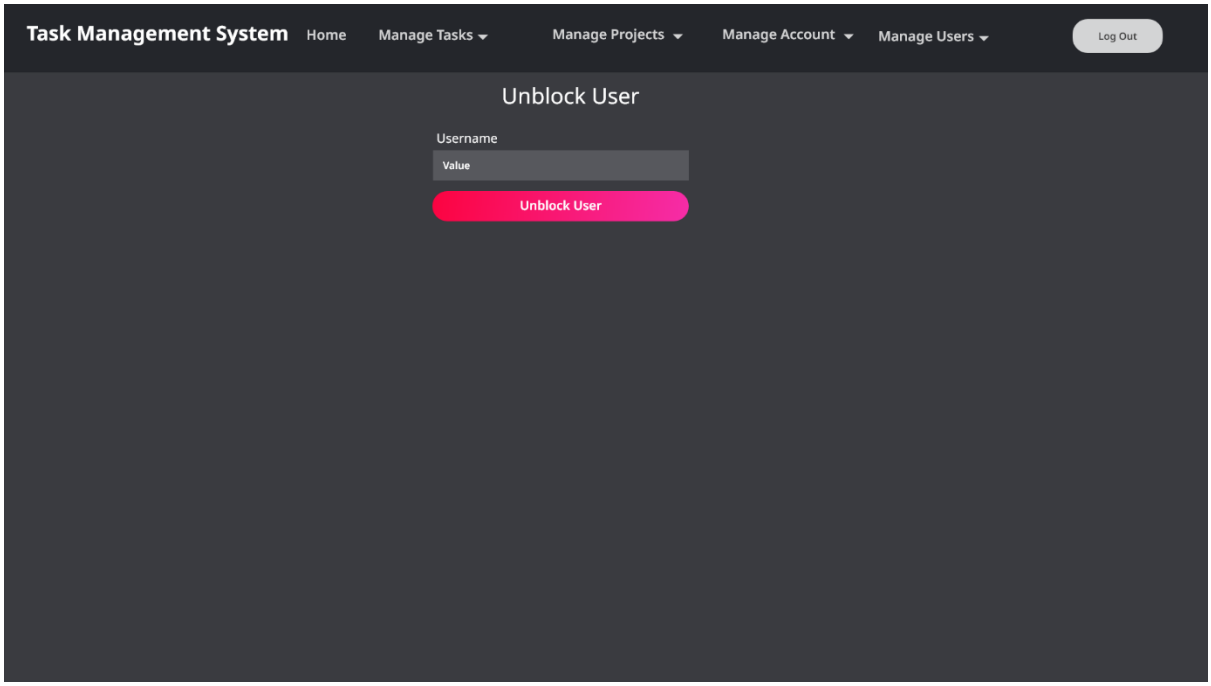


Figure 53 - Unblock User Page Mock-up.

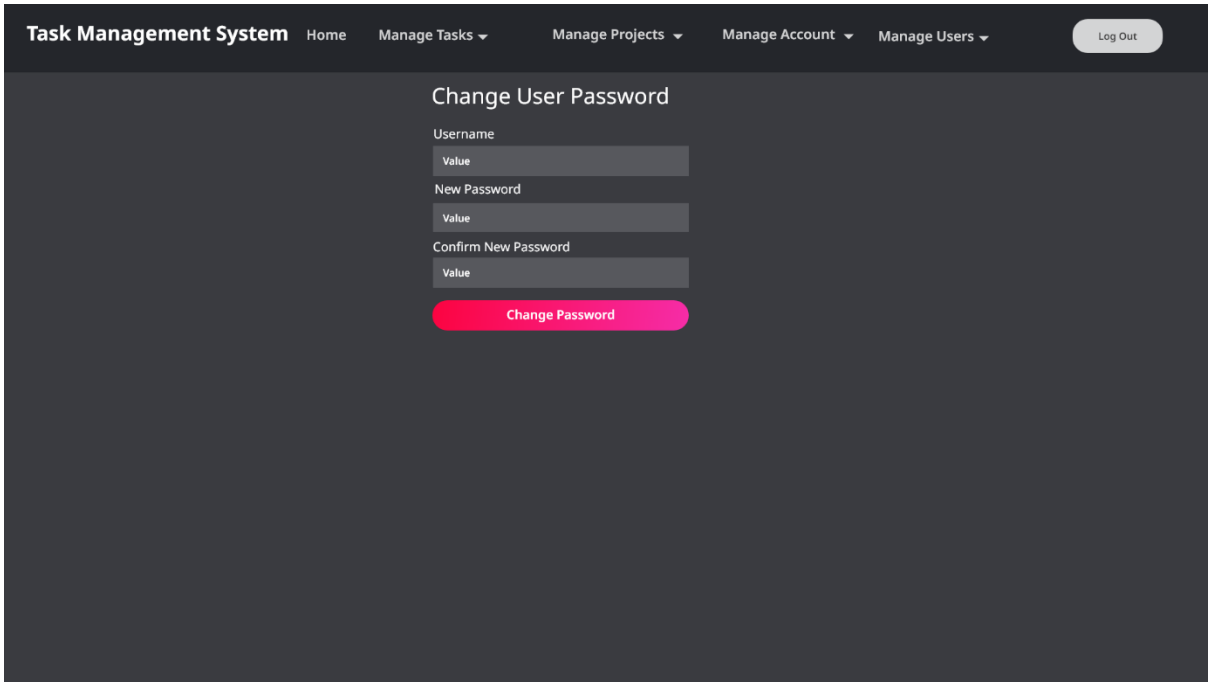


Figure 54 - Change Another User's Password Page Mock-up.

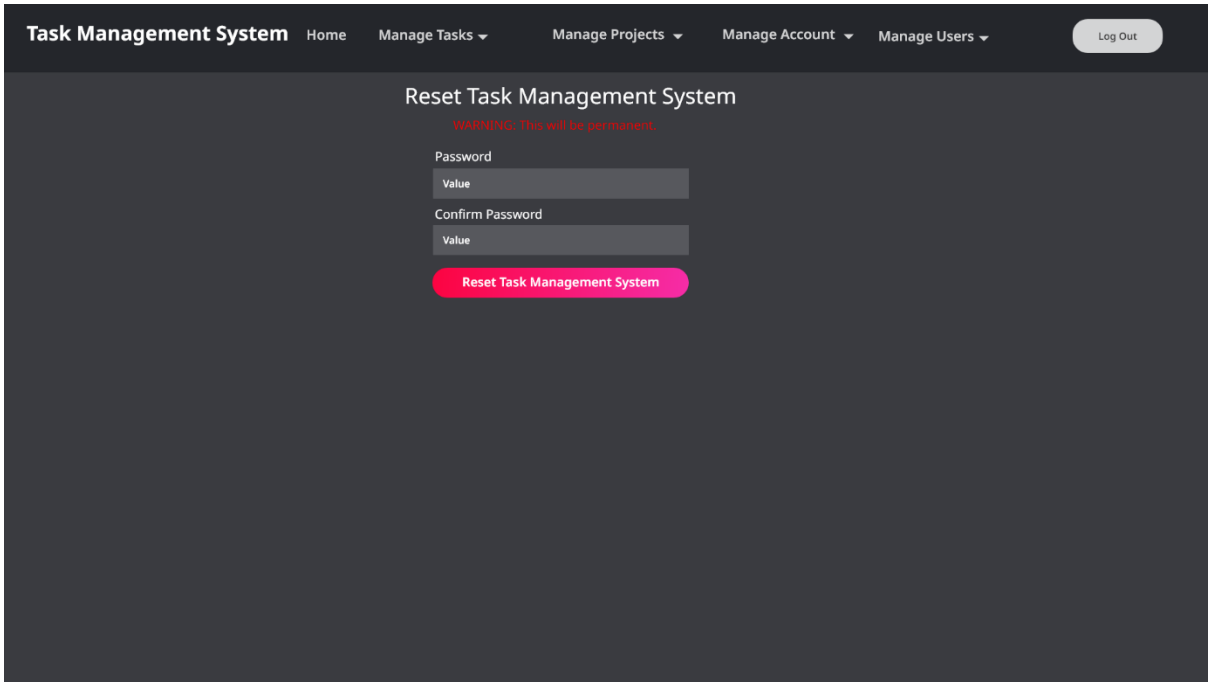


Figure 55 - Factory Reset Page Mock-up.

User/System/functional/Non-Functional Requirements

I have created a separate requirements analysis document for the Task Management System which has specified the mandatory requirements that the client has stated (numbered list) and optional requirements where I had to make my own decisions (in bullet points). See the screenshot below of the requirements document.

Requirements Definition:

1. Project management application
2. Web based
3. Track tasks online
4. Must have responsive user design
5. User authentication (login, registration)
6. Two user roles: Standard User, Administrator
7. Standard Users: view all tasks and add new tasks
8. Administrators: Perform all CRUD operations (Create, Read, Update and Delete) tasks
9. App must have a landing page and functional navbar.
10. Each task should have a title, description, due date, priority, and status.
11. Tasks should be categorized into different projects.
12. Tasks and users to be stored on a database.
13. App must be tested.

- Architecture not specified – choose own architecture.
- Programming language, IDE, framework, APIs not specified - choose own.
- Deployment platform not specified – choose own.
- UI / UX design not specified – design own.

Figure 56 - Screenshot of the Requirements Analysis Document

I have made the following decisions for the optional requirements:

- Architecture – Monolithic as it is easier to develop, code and implement when you first create your application and is less time consuming and more simple to develop for small scale software engineering projects however, I will consider microservice architecture in the future as more features are added to the application and the application becomes larger and more scalable as microservice architecture is more scalable and easer to develop and debug on larger software engineering projects.
- Programming Languages:
    - Python for the backend Flask application
    - SQL for the database
    - HTML for structuring the webpages
    - CSS for styling the webpages
    - JavaScript for adding functionality to the navbar.
- IDE –PyCharm Professional
- Framework and APIs – Flask for backend webpage API.
- A Graphical User Interface (GUI) was chosen as they are easier to use and require less experience and computer literacy then Command Line Interfaces (CLI).

Language and associated Paradigm/Database-For any identified choice, justify why you selected that language

This is a list of languages and associated paradigms and databases used to create the Task Management System:

- Python Screenshot:

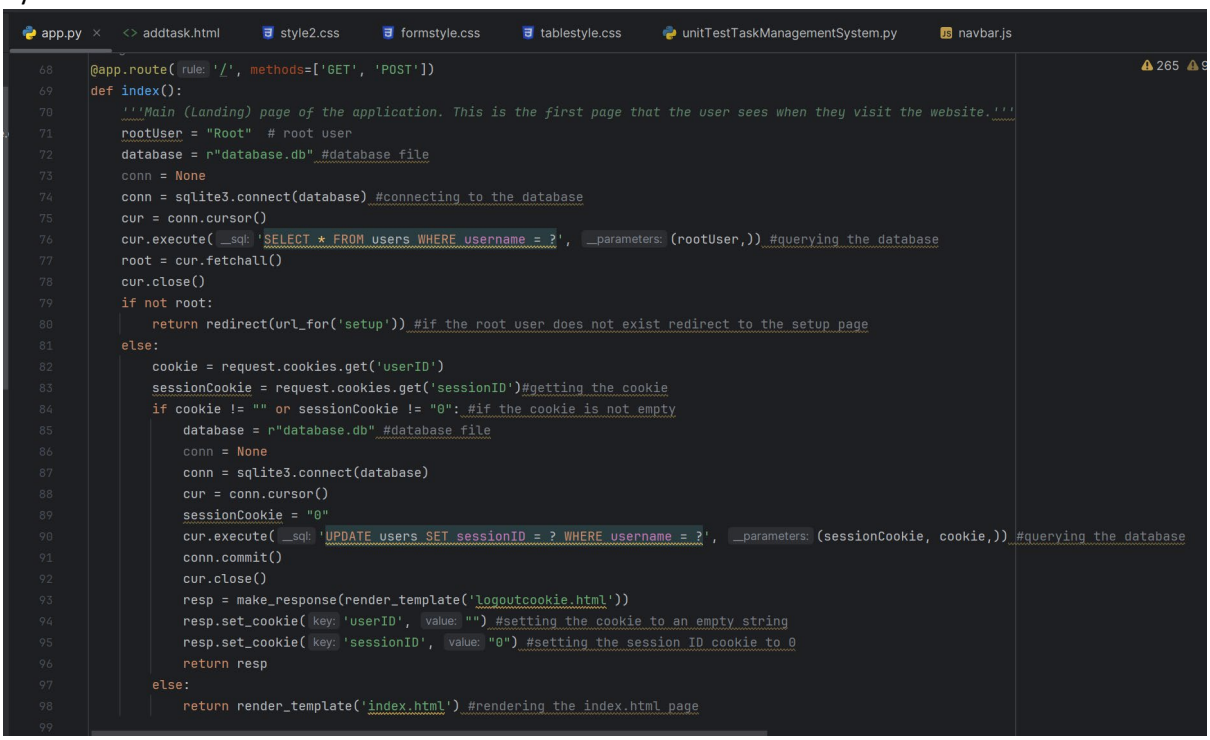


Figure 57 - Screenshot of the Task Management System's main backend Python source code (app.py)

- SQLite Screenshot (SQLite embedded in Python):

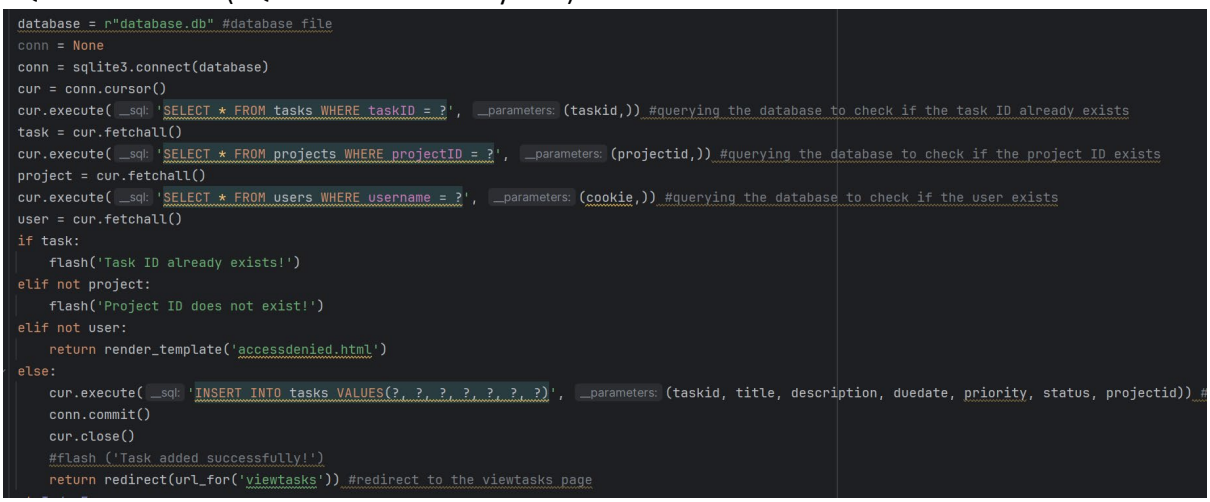
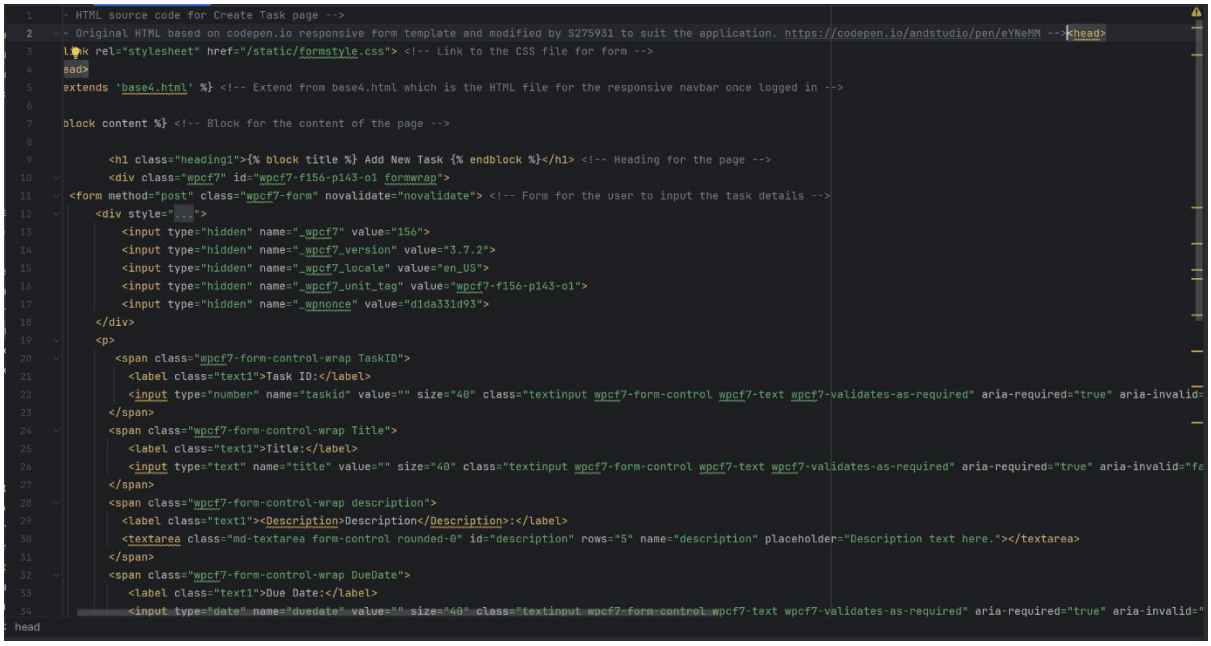


Figure 58 - SQL statements being executed from within the backend Python code to interact with the relational database.

- HTML Screenshot



```
<!-- HTML source code for Create Task page -->
<!-- Original HTML based on codepen.io responsive form template and modified by S275931 to suit the application. https://codepen.io/andstudio/pen/eYNeMM --><head>
<link rel="stylesheet" href="/static/formstyle.css"> <!-- Link to the CSS file for form -->
</head>
{% extends 'base4.html' %} <!-- Extend from base4.html which is the HTML file for the responsive navbar once logged in -->

{% block content %} <!-- Block for the content of the page -->

      <h1 class="heading1">{% block title %} Add New Task {% endblock %}</h1> <!-- Heading for the page -->
      <div class="wpcf7" id="wpcf7-f156-p143-o1 formwrap">
<form method="post" class="wpcf7-form" novalidate="novalidate"> <!-- Form for the user to input the task details -->
    <div style="...">
        <input type="hidden" name="_wpcf7" value="156">
        <input type="hidden" name="_wpcf7_version" value="3.7.2">
        <input type="hidden" name="_wpcf7_locale" value="en_US">
        <input type="hidden" name="_wpcf7_unit_tag" value="wpcf7-f156-p143-o1">
        <input type="hidden" name="_wpnonce" value="d1da331d93">
    </div>
    <p>
      <span class="wpcf7-form-control-wrap TaskID">
        <label class="text1">Task ID:</label>
        <input type="number" name="taskid" value="" size="40" class="textinput wpcf7-form-control wpcf7-text wpcf7-validates-as-required" aria-required="true" aria-invalid=
      </span>
      <span class="wpcf7-form-control-wrap Title">
        <label class="text1">Title:</label>
        <input type="text" name="title" value="" size="40" class="textinput wpcf7-form-control wpcf7-text wpcf7-validates-as-required" aria-required="true" aria-invalid="fa
      </span>
      <span class="wpcf7-form-control-wrap description">
        <label class="text1"><Description>Description</Description>:</label>
        <textarea class="md-textarea form-control rounded-0" id="description" rows="5" name="description" placeholder="Description text here."></textarea>
      </span>
      <span class="wpcf7-form-control-wrap DueDate">
        <label class="text1">Due Date:</label>
        <input type="date" name="duedate" value="" size="40" class="textinput wpcf7-form-control wpcf7-text wpcf7-validates-as-required" aria-required="true" aria-invalid="
head
```

Figure 59 - Frontend HTML code for Add Task form webpage. The CSS and JavaScript files are attached to the HTML files for styling and functionality. HTML defines the structure of webpages.

- CSS Screenshot

app.py | addtask.html | style2.css | formstyle.css | tablestyle.css | unitTestTaskM

```css
/* This is the CSS for responsive tables to display data from the database */
/* Original CSS sourced from codepen.io and modified by S275931 to suit the application.
table {
  border: 1px solid #3c3f3f;
  border-collapse: collapse;
  margin: 0;
  padding: 0;
  width: 100%;
  table-layout: fixed;
   overflow-y:scroll;
  height: 100%;
  display:block;
} /* style for table */

table caption {
  font-size: 1.5em;
  margin: .5em 0 .75em;
    color: #ff8900;
} /* style for table caption */

table tr {
  background-color: #3c3f3f;
  border: 1px solid #fba600;
  width: 100%;
  padding: .35em;
} /* style for table tr */

table th,
table td {
  padding: .625em;
  text-align: left;
    color: #ff8900;
  width: 100rem;
    border: 1px solid #fd8800;
```

media screen and (max-width: 600px) › table tr

Figure 60 - Frontend CSS code for styling the Task Management System's webpages, adding responsiveness and layout.

- JavaScript Screenshot

Figure 61 - Frontend JavaScript code for navbar functionality.

There are two main programming paradigms I have used to create the Task Management System:

1. Functional – The main python program file (app.py) uses functional programming paradigms this is supported by the Flask framework for backend web development as each webpage has its own function which is preceded by a Flask decorator which specifies the endpoint URL for the page e.g. `@app.route('/createaccount', methods=['GET', 'POST'])` which specifies the URL endpoint for the create account page. The function contains the code that provides functionality for the specified webpage. For example, the create account function contains the code for creating the account including obtaining information from the Create Account form using the POST method, validating the content in the form, using regular expressions to check that the password meets the password requirements, query the database using SQL to verify the account doesn't already exist, encrypt the password using

SHA3-512 bit encryption to store the passwords securely as hashes, use SQL to add the new account to the database and generate userID and SessionID cookies to authenticate the session. Finally, the function renders the webpage template in the user's web browser (front-end). Functional programming with Python is also easy to understand the code.

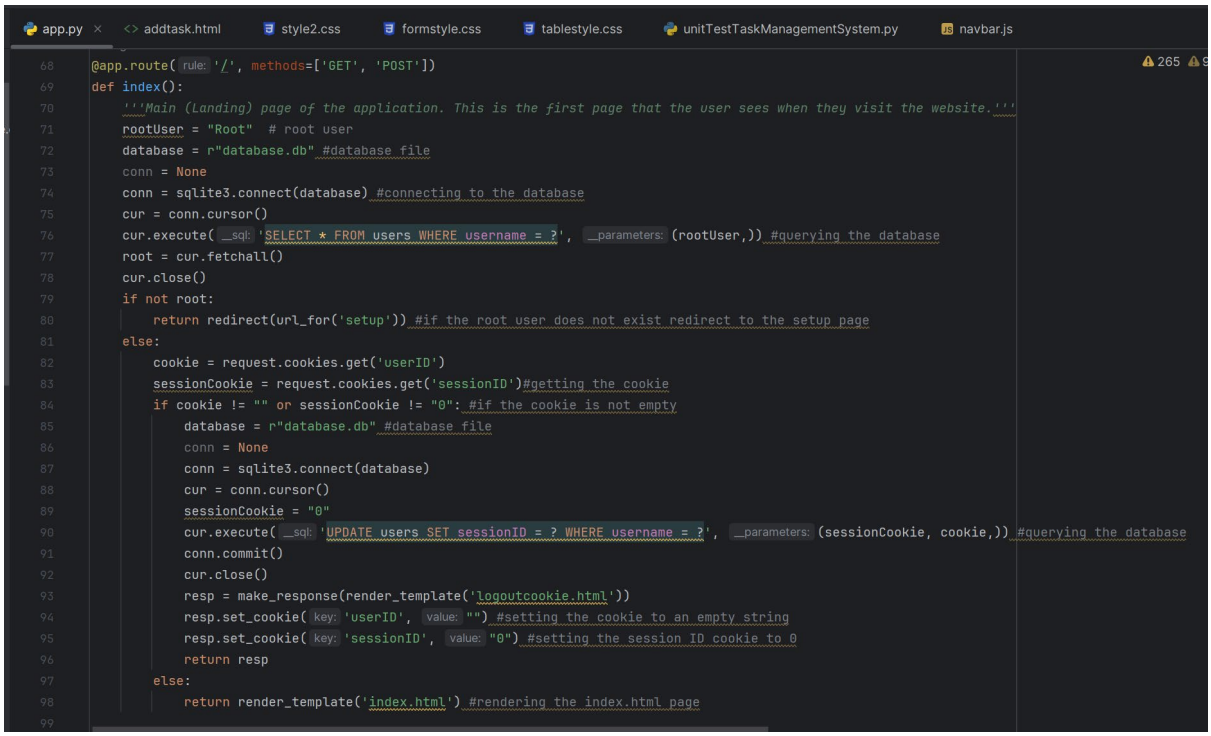


Figure 62 – Functional Programming - Flask decorator and Python function to navigate to the landing page endpoint, render the landing page (index.html) template and add functionality

2. Object-oriented programming – The unit testing Python code uses object-oriented programming to test the route to the Task Management System's web pages. The unit testing code contains a TestRoutes(unittest.testcase) class that contains a setup method to set up the Task Management System for testing and methods to unit test each webpage within the TestRoutes class.

Figure 63 - Object Oriented Programming - Class to unit test endpoint routes encapsulating functions to test each webpage.

3. The JavaScript for the navbar's navigation functionality is also functional as there are individual functions for each item in the navbar dropdown menus that redirect the user to the specified page when they click on the items in the dropdown menus.

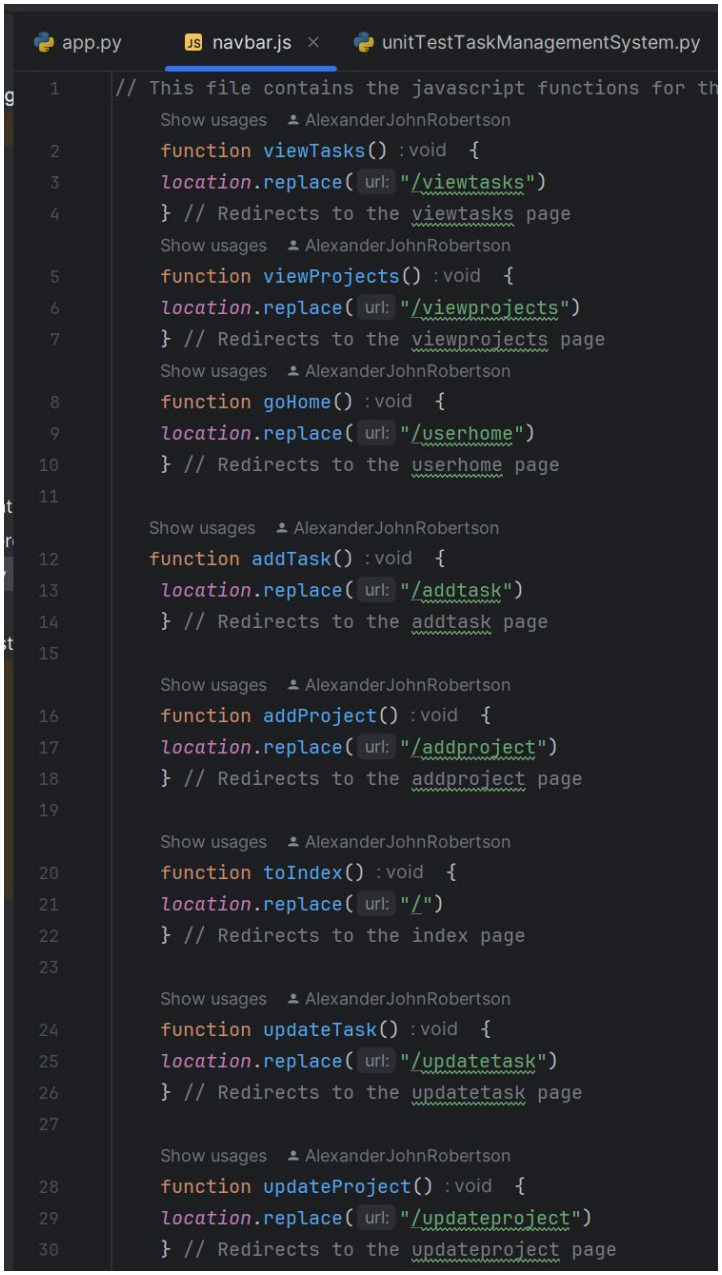


Figure 64 - Functions in JavaScript code for navbar - One function for each navbar item.

For the Task Management System's database, I have chosen to use a locally stored SQLite relational database that has three tables to store the dada for users, tasks, and projects for the following reasons:

- SQLite uses SQL code which is supported by Python and Flask and the SQL statements can be executed directly from within the Python code.
- SQLite is lightweight and does not require powerful computer hardware to use.
- SQLite is easy to code and use.
- SQLite requires no installation or setup; it just needs a database file and SQL code to interact with the database.
- SQLite is portable and cross-platform like Python.
- I have chosen a relational database because it is a simple database to setup and use compared to other NoSQL database types.

- Relational databases are fast due to their simplicity.
- Relational databases support multiple users.
- Relational databases use primary and foreign keys in each table to allow the tables to relate to each other and provide constraints for data entry.

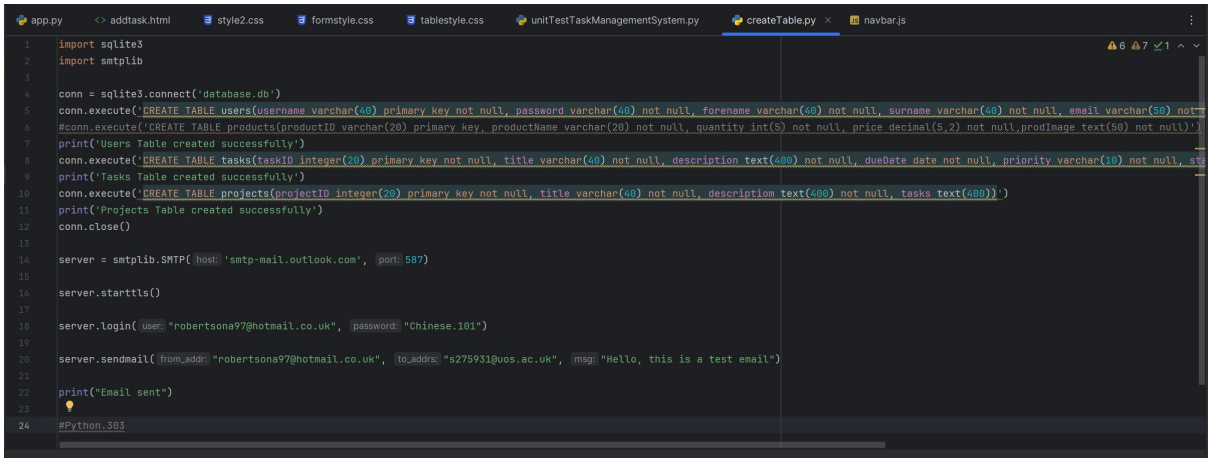
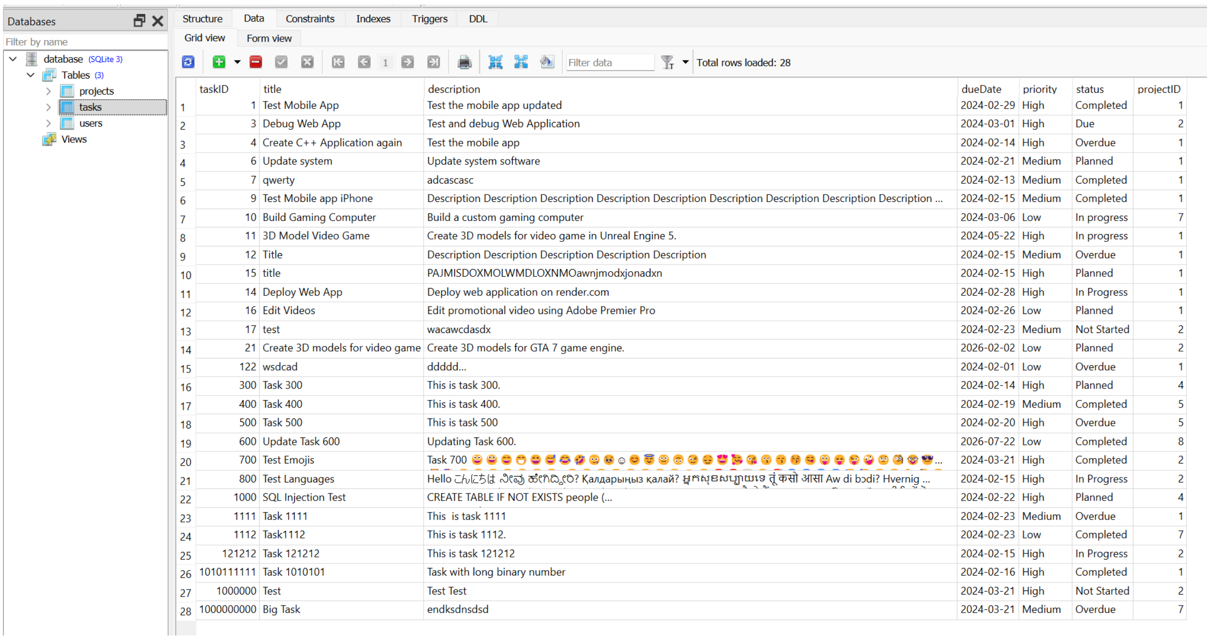


Figure 65 - Create Table SQL code for Task Management System database (separate Python file).

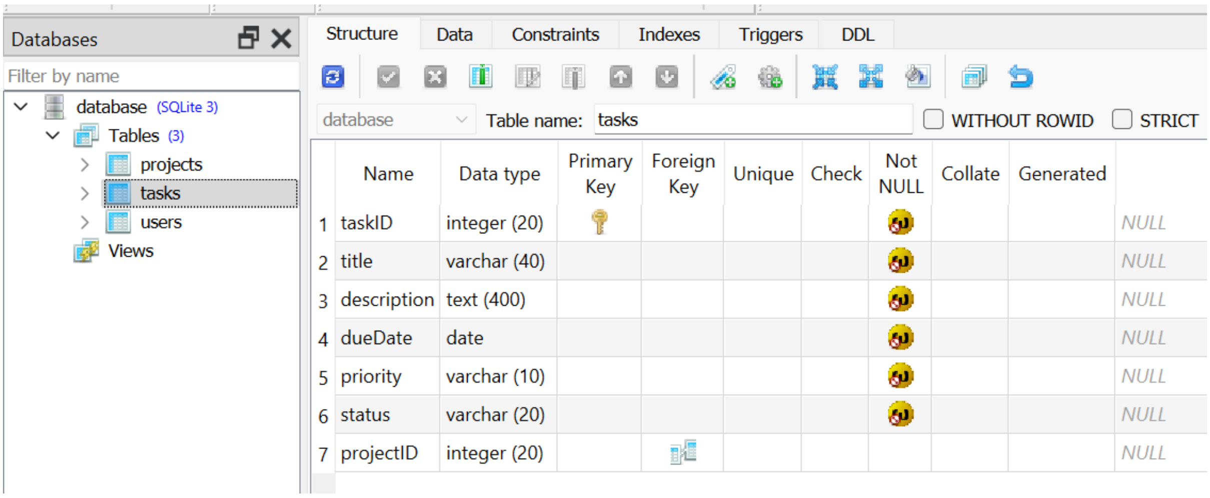
```
database = r"database.db" #database file
conn = None
conn = sqlite3.connect(database)
cur = conn.cursor()
cur.execute( __sql: 'SELECT * FROM tasks WHERE taskID = ?', __parameters: (taskid,)) #querying the database to check if the task ID already exists
task = cur.fetchall()
cur.execute( __sql: 'SELECT * FROM projects WHERE projectID = ?', __parameters: (projectid,)) #querying the database to check if the project ID exists
project = cur.fetchall()
cur.execute( __sql: 'SELECT * FROM users WHERE username = ?', __parameters: (cookie,)) #querying the database to check if the user exists
user = cur.fetchall()
if task:
    flash('Task ID already exists!')
elif not project:
    flash('Project ID does not exist!')
elif not user:
    return render_template('accessdenied.html')
else:
    cur.execute( __sql: 'INSERT INTO tasks VALUES(?, ?, ?, ?, ?, ?, ?)', __parameters: (taskid, title, description, duedate, priority, status, projectid)) #
    conn.commit()
    cur.close()
    #flash ('Task added successfully!')
    return redirect(url_for('viewtasks')) #redirect to the viewtasks page
```

Figure 66 - SQL code executed by Python code in the main application file app.py to interface with the Task Management System database.



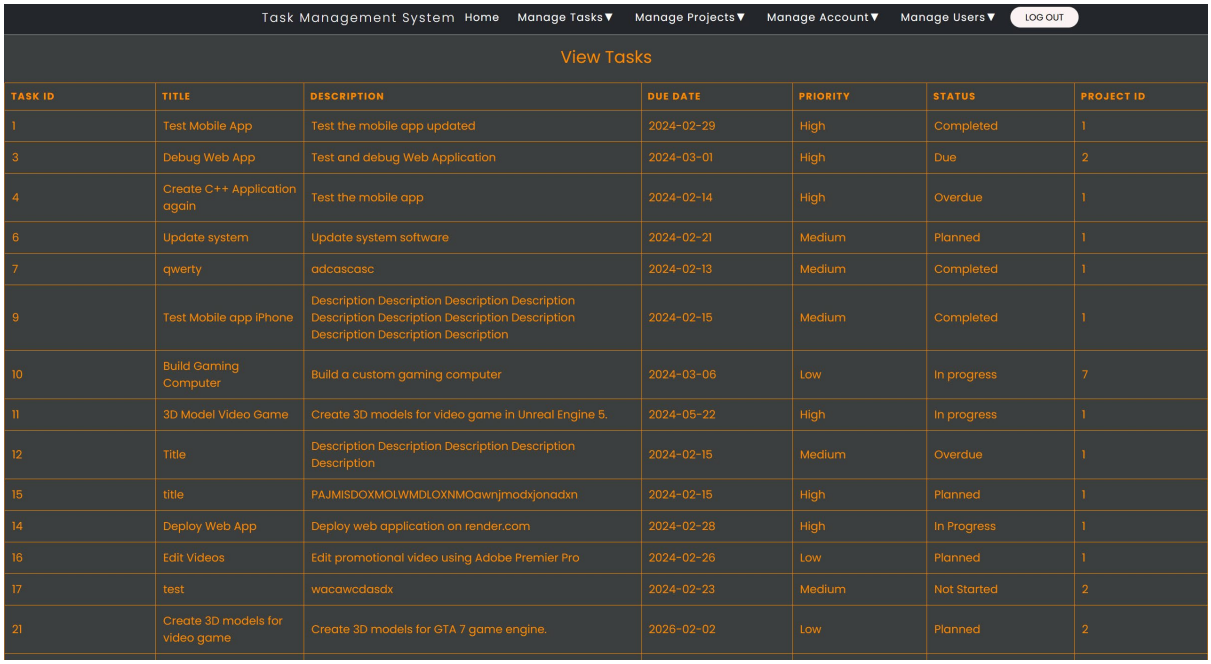
	taskID	title	description	dueDate	priority	status	projectID
1	1	Test Mobile App	Test the mobile app updated	2024-02-29	High	Completed	1
2	3	Debug Web App	Test and debug Web Application	2024-03-01	High	Due	2
3	4	Create C++ Application again	Test the mobile app	2024-02-14	High	Overdue	1
4	6	Update system	Update system software	2024-02-21	Medium	Planned	1
5	7	qwerty	adcascasc	2024-02-13	Medium	Completed	1
6	9	Test Mobile app iPhone	Description Description Description Description Description Description Description Description Description ...	2024-02-15	Medium	Completed	1
7	10	Build Gaming Computer	Build a custom gaming computer	2024-03-06	Low	In progress	7
8	11	3D Model Video Game	Create 3D models for video game in Unreal Engine 5.	2024-05-22	High	In progress	1
9	12	Title	Description Description Description Description Description	2024-02-15	Medium	Overdue	1
10	15	title	PAJMISDOXMOLWMDLOXNMOawnjmodxjonadxn	2024-02-15	High	Planned	1
11	14	Deploy Web App	Deploy web application on render.com	2024-02-28	High	In Progress	1
12	16	Edit Videos	Edit promotional video using Adobe Premier Pro	2024-02-26	Low	Planned	1
13	17	test	wacawcdasdx	2024-02-23	Medium	Not Started	2
14	21	Create 3D models for video game	Create 3D models for GTA 7 game engine.	2026-02-02	Low	Planned	2
15	122	wsdcad	ddddd...	2024-02-01	Low	Overdue	1
16	300	Task 300	This is task 300.	2024-02-14	High	Planned	4
17	400	Task 400	This is task 400.	2024-02-19	Medium	Completed	5
18	500	Task 500	This is task 500	2024-02-20	High	Overdue	5
19	600	Update Task 600	Updating Task 600.	2026-07-22	Low	Completed	8
20	700	Test Emojis	Task 700 [illegible] ...	2024-03-21	High	Completed	2
21	800	Test Languages	Hello こんにちは ನೀವು ಹೇಗಿದ್ದೀರಿ? Қалдарыңыз қалай? អ្នកសុខសប្បាយទេ तूं कसो आसा Aw di bɔdi? Hvernig ...	2024-02-15	High	In Progress	2
22	1000	SQL Injection Test	CREATE TABLE IF NOT EXISTS people (...	2024-02-22	High	Planned	4
23	1111	Task 1111	This is task 1111	2024-02-23	Medium	Overdue	1
24	1112	Task1112	This is task 1112.	2024-02-23	Low	Completed	7
25	121212	Task 121212	This is task 121212	2024-02-15	High	In Progress	2
26	1010111111	Task 1010101	Task with long binary number	2024-02-16	High	Completed	1
27	1000000	Test	Test Test	2024-03-21	High	Not Started	2
28	1000000000	Big Task	endksdnsdsd	2024-03-21	Medium	Overdue	7

Figure 67 - SQLite Studio displaying the Tasks table in the Task Management System's database. SQLite Studio makes it easier to manage, edit and develop the database during the development phase.



	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated	
1	taskID	integer (20)								NULL
2	title	varchar (40)								NULL
3	description	text (400)								NULL
4	dueDate	date								NULL
5	priority	varchar (10)								NULL
6	status	varchar (20)								NULL
7	projectID	integer (20)								NULL

Figure 68 - Managing the structure of Task Management System database tables in SQLite Studio.



TASK ID	TITLE	DESCRIPTION	DUE DATE	PRIORITY	STATUS	PROJECT ID
1	Test Mobile App	Test the mobile app updated	2024-02-29	High	Completed	1
3	Debug Web App	Test and debug Web Application	2024-03-01	High	Due	2
4	Create C++ Application again	Test the mobile app	2024-02-14	High	Overdue	1
6	Update system	Update system software	2024-02-21	Medium	Planned	1
7	qwerty	adcascasc	2024-02-13	Medium	Completed	1
9	Test Mobile app iPhone	Description Description Description Description Description Description Description Description Description Description Description Description	2024-02-15	Medium	Completed	1
10	Build Gaming Computer	Build a custom gaming computer	2024-03-06	Low	In progress	7
11	3D Model Video Game	Create 3D models for video game in Unreal Engine 5.	2024-05-22	High	In progress	1
12	Title	Description Description Description Description Description	2024-02-15	Medium	Overdue	1
15	title	PAJMISDOXMOLWMDLOXNMOawnjmodxjonadxn	2024-02-15	High	Planned	1
14	Deploy Web App	Deploy web application on render.com	2024-02-28	High	In Progress	1
16	Edit Videos	Edit promotional video using Adobe Premier Pro	2024-02-26	Low	Planned	1
17	test	wacawcdasdx	2024-02-23	Medium	Not Started	2
21	Create 3D models for video game	Create 3D models for GTA 7 game engine.	2026-02-02	Low	Planned	2

Figure 69 - View Tasks table viewed in the web browser on the Task Management System's View Tasks Page. The backend Python code executes SQL code to obtain data from the SQLite database then uses GET RESTful API to send the data to the frontend webpages. The HTML and CSS then display the data as a table on the webpage.

Database Designs

I have created a UML Entity Relationship (ERD) Diagram to design the Task Management System's database. There are three different entities in the database – Users, Projects, and Tasks. There are three different tables for the three entity types (Users table, Projects table and Tasks table). The Users entity has the Username attribute as the primary key, has a many to many relationship with the Projects and Tasks entities as many users can create, read (view), update and delete (CRUD) many tasks and projects

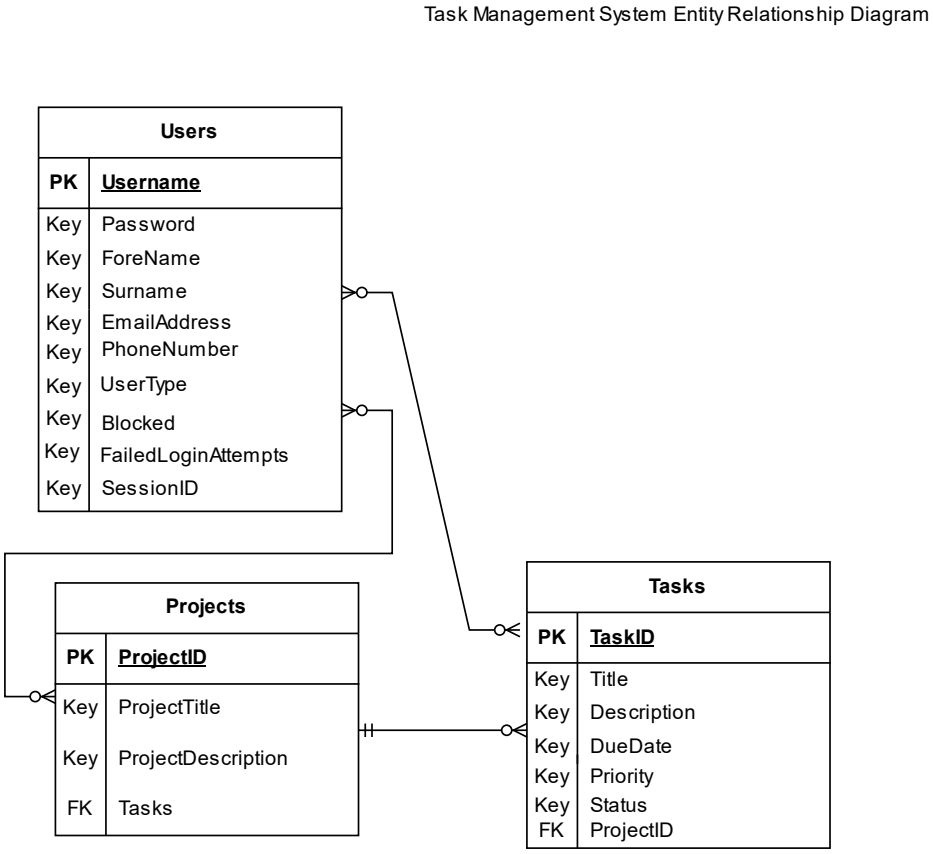


Figure 70 - Entity Relationship Diagram for Task Management System Database.

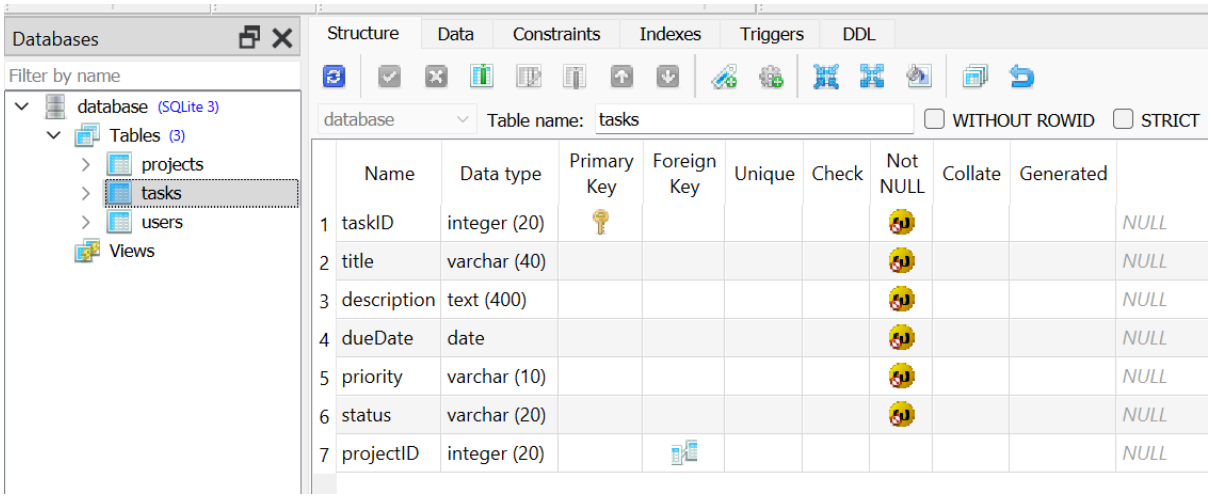


Figure 71 - Database structure viewed in SQLite Studio (tasks table)

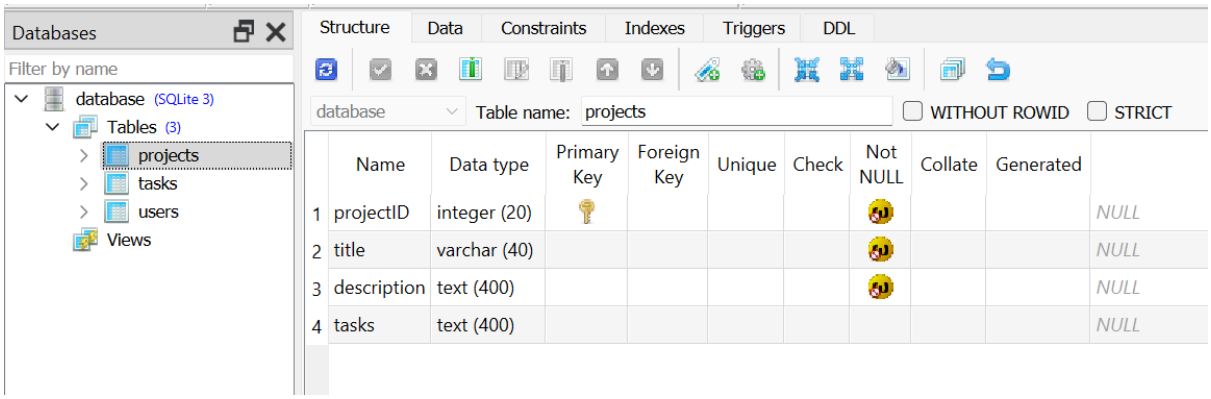


Figure 72 - Database structure viewed in SQLite Studio (projects table)

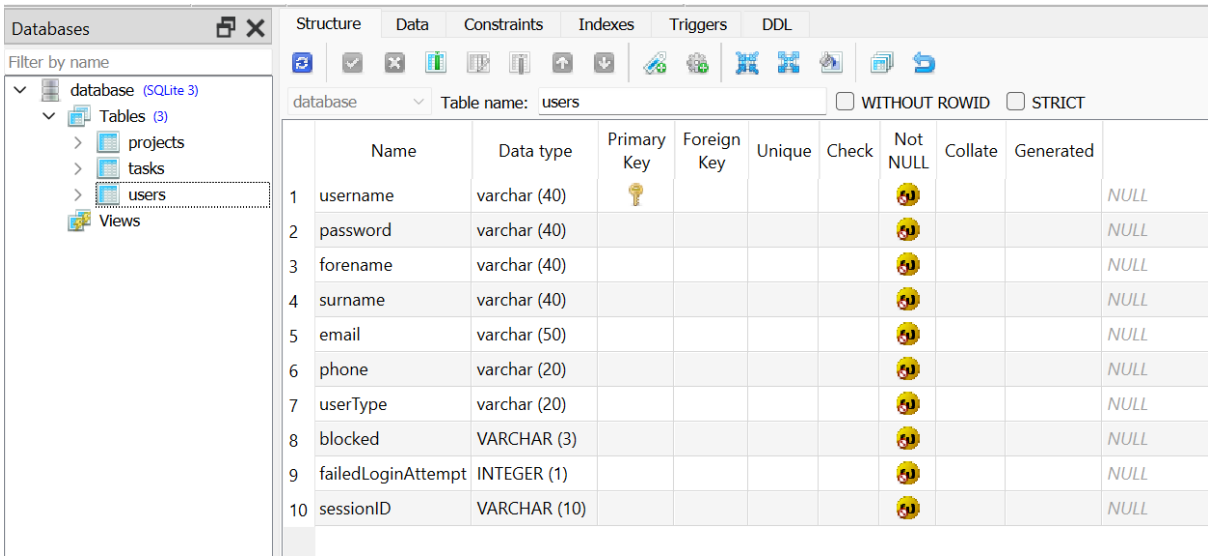


Figure 73 - Database structure viewed in SQLite Studio (users table)

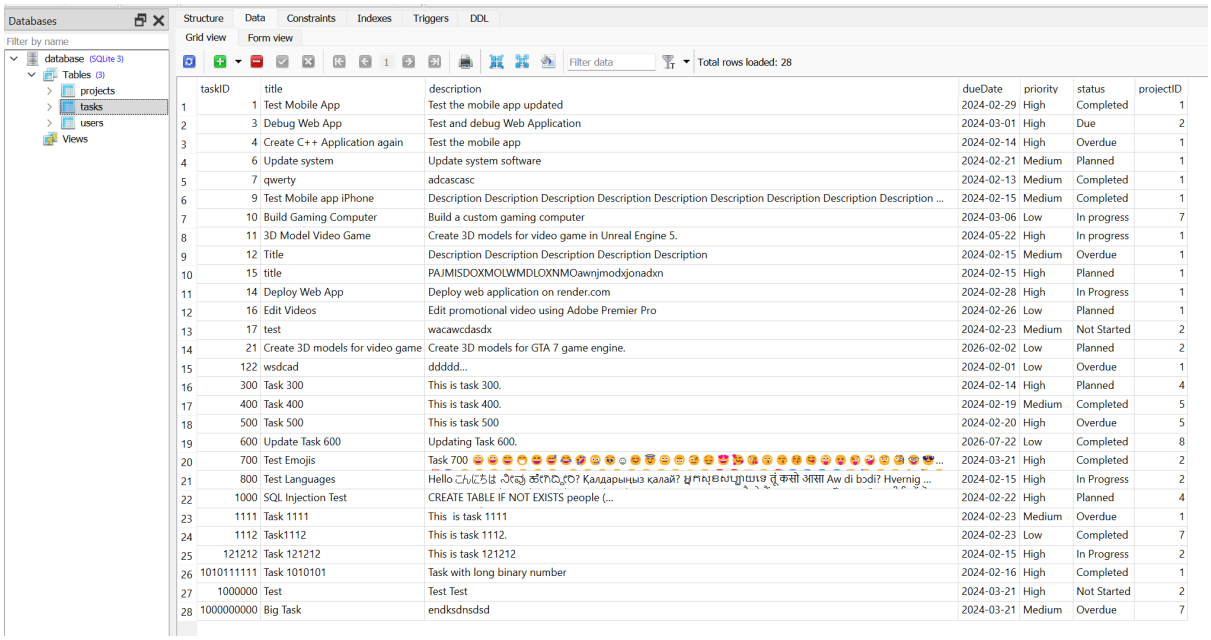
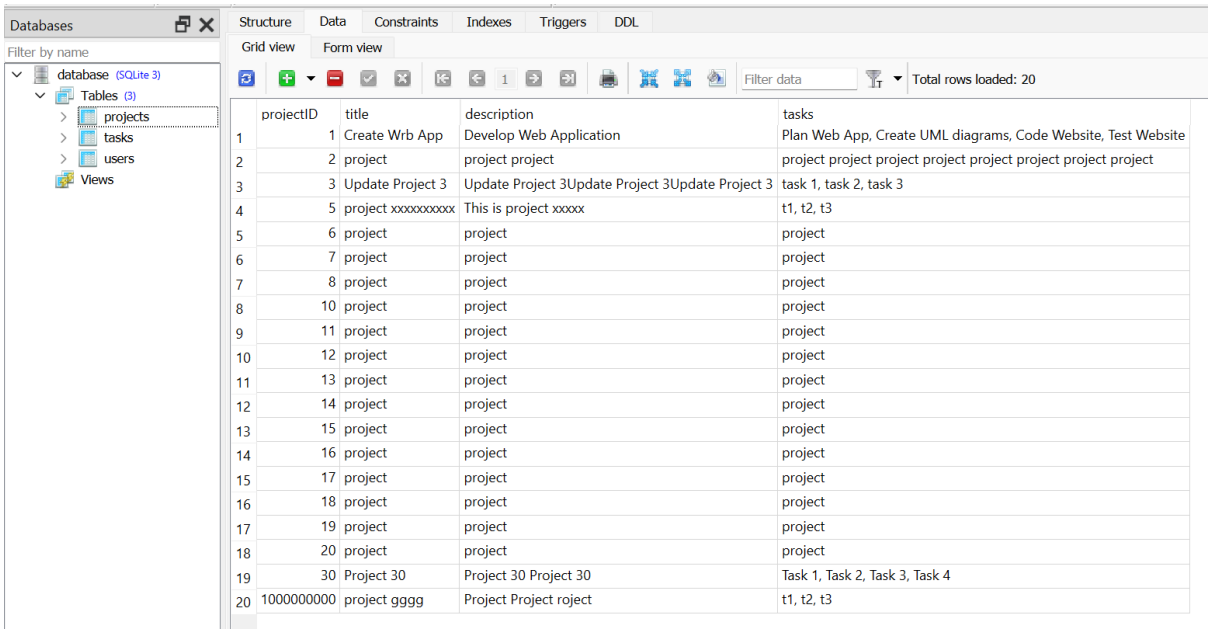
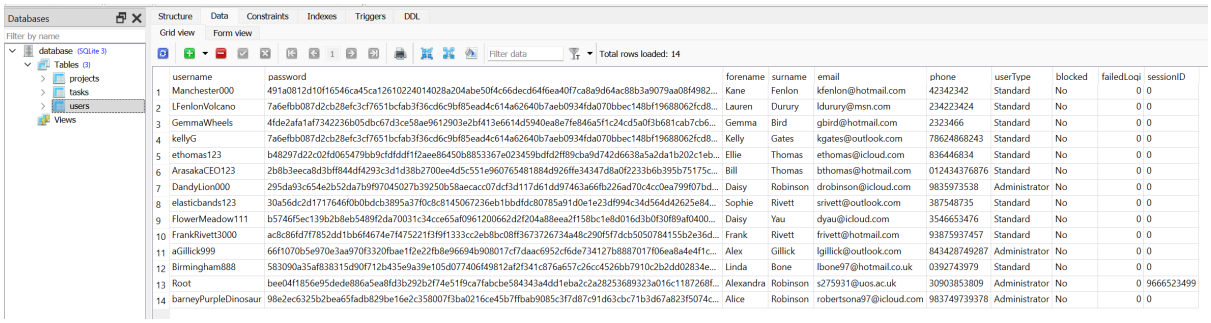


Figure 74 - Tasks table content viewed in SQLite Studio.



	projectID	title	description	tasks
1	1	Create Wrb App	Develop Web Application	Plan Web App, Create UML diagrams, Code Website, Test Website
2	2	project	project project	project project project project project project project project
3	3	Update Project 3	Update Project 3Update Project 3Update Project 3	task 1, task 2, task 3
4	5	project xxxxxxxxxx	This is project xxxxx	t1, t2, t3
5	6	project	project	project
6	7	project	project	project
7	8	project	project	project
8	10	project	project	project
9	11	project	project	project
10	12	project	project	project
11	13	project	project	project
12	14	project	project	project
13	15	project	project	project
14	16	project	project	project
15	17	project	project	project
16	18	project	project	project
17	19	project	project	project
18	20	project	project	project
19	30	Project 30	Project 30 Project 30	Task 1, Task 2, Task 3, Task 4
20	1000000000	project gggg	Project Project roject	t1, t2, t3

Figure 75 - Projects table content viewed in SQLite Studio.



	username	password	forename	surname	email	phone	userType	blocked	failedLogi	sessionID
1	Manchester000	491a0812d10f16546ca45ca1261022401402 8a204abe50f4c66decd64f6ea40f7ca8a9d64ac88b3a9079aa08f4982...	Kane	Fenlon	kfenlon@hotmail.com	42342342	Standard	No	0	0
2	LFenlonVolcano	7a6efbb087d2cb28efc3cf7651bcfab3f36cd6c9bf85ead4c614a62640b7aeb0934fda070bbec148bf19688062fcd8...	Lauren	Durury	ldurury@msn.com	234223424	Standard	No	0	0
3	GemmaWheels	4fde2afa1af7342236b05dbc67d3ce58ae9612903e2bf413e6614d5940ea8e7fe846a5f1c24cd5a0f3b681cab7cb6...	Gemma	Bird	gbird@hotmail.com	2323466	Standard	No	0	0
4	kellyG	7a6efbb087d2cb28efc3cf7651bcfab3f36cd6c9bf85ead4c614a62640b7aeb0934fda070bbec148bf19688062fcd8...	Kelly	Gates	kgates@outlook.com	78624868243	Standard	No	0	0
5	ethomas123	b48297d22c02fd065479bb9cfdfddf1f2aee86450b8853367e023459bdfd2ff89cba9d742d6638a5a2da1b202c1eb...	Ellie	Thomas	ethomas@icloud.com	836446834	Standard	No	0	0
6	ArasakaCEO123	2b8b3eeca8d3bff844df4293c3d1d38b2700ee4d5c551e960765481884d926ffe34347d8a0f2233b6b395b75175c...	Bill	Thomas	bthomas@hotmail.com	012434376876	Standard	No	0	0
7	DandyLion000	295da93c654e2b52da7b9f97045027b39250b58aecacc07dcf3d117d61dd97463a66fb226ad70c4cc0ea799f07bd...	Daisy	Robinson	drobinson@icloud.com	9835973538	Administrator	No	0	0
8	elasticbands123	30a56dc2d1717646f0b0bdcb3895a37f0c8c8145067236eb1bbdfdc80785a91d0e1e23df994c34d564d42625e84...	Sophie	Rivett	srivett@outlook.com	387548735	Standard	No	0	0
9	FlowerMeadow111	b5746f5ec139b2b8eb5489f2da70031c34cce65af096120066 2d2f204a88eea2f158bc1e8d016d3b0f30f89af0400...	Daisy	Yau	dyau@icloud.com	3546653476	Standard	No	0	0
10	FrankRivett3000	ac8c86fd7f7852dd1bb6f4674e7f475221f3f9f1333cc2eb8bc08ff3673726734a48c290f5f7dcb5050784155b2e36d...	Frank	Rivett	frivett@hotmail.com	93875937457	Standard	No	0	0
11	aGillick999	66f1070b5e970e3aa970f3320fbae1f2e22fb8e96694b908017cf7daac6952cf6de734127b8887017f06ea8a4e4f1c...	Alex	Gillick	lgillick@outlook.com	843428749287	Administrator	No	0	0
12	Birmingham888	583090a35af838315d90f712b435e9a39e105d077406f49812af2f341c876a657c26cc4526bb7910c2b2dd02834e...	Linda	Bone	lbone97@hotmail.co.uk	0392743979	Standard	No	0	0
13	Root	bee04f1856e95dede886a5ea8fd3b292b2f74e51f9ca7fabcbe584343a4dd1eba2c2a28253689323a016c1187268f...	Alexandra	Robinson	s275931@uos.ac.uk	30903853809	Administrator	No	0	9666523499
14	barneyPurpleDinosaur	98e2ec6325b2bea65fadb829be16e2c358007f3ba0216ce45b7ffbab9085c3f7d87c91d63cbc71b3d67a823f5074c...	Alice	Robinson	robertsona97@icloud.com	983749739378	Administrator	No	0	0

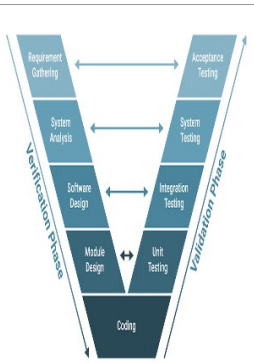
Figure 76 - Users table content viewed in SQLite Studio.

Software Process Model Selected to come up with solution-List a few and justify your choice

I have chosen the Rapid Application Development (RAD) methodology because the Task Management System software needs to be developed urgently as it is required as a university assignment therefore I only had a few weeks to completely develop the entire application from scratch from requirements analysis to design with UML Use Case, Class and ERD diagrams, Wireframes an mock-ups to developing, coding and debugging the application to testing the application to releasing and deploying the application and receiving feedback and maintenance.

Table to compare RAD to other development methodologies:

Rapid Application Development (My Choice)	**Waterfall**	**V-Model**	**Spiral Model**
Develop software at high speed and deploy as soon as possible	Oldest methodology.	Verification and Validation	Contains iterative and waterfall elements
More reliable software due to constant debugging and client feedback	All phases including planning, development, testing, and deployment are done sequentially.	Each development phase is complimented with a testing phase forming a V shaped diagram	Resembles a spiral with unspecified number of loops.
Sequential Linear development methodology	Cannot make changes after the planning phase therefore changes cannot be made after planning if requirements change or if mistakes are found.	Verification evaluates plans, designs, code, requirements, and specifications	Split into four quadrants: 1. Objective Setting 2. Risk Assessment and Reduction 3. Development and Validation 4. Review and Planning
Prioritise functionality over UI/UX	Rigid methodology – development objectives are set at planning stage.	Validation tests the application for compliance with the organisation's needs such as unit testing, system testing, component testing, integration testing, acceptance testing.	Systematic and iterative approach
Easy to change requirements and fix mistakes	Client involved at the planning stage only.	Sequential – next phase starts after the previous phase completes	Each spiral is a phase of the development cycle.
Interact with client throughout all development phases	Process: Feasibility Study → Requirements Analysis → Design → Coding → Testing → Maintenance.	Process: Business Requirements specs (Acceptance Testing) → System Requirements analysis (System Testing) → Software Design (Integration Testing) → Module Design (Unit Testing) → Coding → Unit Testing (Module Design) → Integration testing (Software Design) → System Testing (System Requirements Analysis) → Acceptance Testing (Requirement Gathering)	Process: Planning → Risk analysis → Engineering → Evaluation → Planning (repeat for next spiral)

Process: Business Modelling → Data Modelling → **Process Modelling** → Application Generation → **Testing and Turnover.**			
Advantages: • **Fast Application development** • **Easy to accommodate changes and fix mistakes** • **Reduced cost** • **Reusability of components** • **Easy to measure project progress** • **Customer/client feedback at all stages** **Disadvantages:** • **Require highly skilled professionals** • **Not suitable for all applications** • **Not meant for small scale projects** • **Only suitable for modular systems (Team Kissflow, 2024) (Geeks for Geeks, 2024) (Altynpara, 2023)**	Advantages: • Defines milestones and deadlines • Simple to understand and follow • Understanding, following, and arranging tasks is simplified • Coding after design encourages good coding habits • Results in a structured and disciplined organisation (OS System, 2020) Disadvantages: • No room for error • Testing is delayed until end of lifecycle • Project phases cannot overlap • Reduced overall efficiency • Not suitable for complex, high risk or ongoing projects • No working product until end of lifecycle. (Lewis, 2022) (OS System, 2020) (Altynpara, 2023)	Advantages: • Efficient • Lower Risk • Enhanced Testing • Improved Quality • Improved Documentation Disadvantages: • Very Rigid – little room for changes if requirements change or mistakes are detected. • Time consuming • Limited Agility • Too much focus on testing • Resource intensive. (Oppermann, 2023)	Advantages: • Good at-risk handling • Good for large projects • Customer satisfaction • Iterative and Incremental Approach • Focus on risk management • Improved quality • Improved communication Disadvantages: • More complex than other SDLC modes • Expensive therefore not suitable for small projects • Relies too much on risk analysis • Time consuming and difficulty with time management • Complex compared to other SDLC models (Geeks for Geeks, 2024)

DevOps and DevSecOps	Agile (Scrum)	Incremental Model
Includes the operations team as well as the development team	Agile Framework	Incremental software improvements are built as repeated, separate timelines
DevSecOps also includes the security team	Incremental development	Two types: Staged and Parallel
Incorporates agile practises	Scrum team consists of: 1. Scrum Master 2. Product Owner 3. Developers 4. Review team	Staged – Each phase is completed sequentially Parallel – Some stages are completed simultaneously
Development, Security and Operations teams collaborate to create a software product. (Hall, 2024) (Altynpara, 2023)	Process: Product backlog→ Sprint planning meeting → Development sprints every 1-4 weeks with daily scrums → Finished work. (Scrum.org, 2024)	Process: Requirements analysis → Version 1 → Design → Code → Test → Deploy → Version 2 → Design → Code → Test → Deploy → version n → Design → Code → Test → Deploy → repeat
Process: Plan → Code → **Build** → Test → **Release** → Deploy → **Operate** → Monitor → **Plan (repeat cycle)**	Advantages: • Adaptable and flexible – easy to make changes if requirements change or if mistakes are made. • Emphasises innovation, new ideas and creativity • Faster time to market • Cheaper than other methodologies • Higher product quality due to more transparency • Increased worker motivation and satisfaction • Improved customer satisfaction from continuous feedback (Vasiliauskas, 2023) (OS System, 2020)	System development divided into several smaller projects
Advantages: • **Faster development and production time** • **Lower risk** • **More efficiency through automation**	Disadvantages: • Difficult to scale on large projects • Requires experienced team members with a lot of training	• Advantages: Prepare software fast • Easy to make changes if requirements change or mistakes are made • Iterations provide risk handling support and reduce risk

• **Improved collaboration** • **Easy to adapt changes if requirements change and fix mistakes** **(Bigelow, 2022) (Altynpara, 2023)**	• Only suitable for small teams – larger teams should use DevOps. • Difficult to integrate with waterfall methodologies • Requires radical changes within the organisation (Vasiliauskas, 2023) (OS System, 2020) (Altynpara, 2023)	• Flexibility with scope • Less expensive compared toother development models • Transparency with client • Easy to detect and identify errors •
Disadvantages: • **Expensive** • **More complex** • **DevOps can fail when goals are unrealistic** • **Lack of performance metrics (Bigelow, 2022) (Altynpara, 2023)**		Disadvantages: • Good team and planning are required • Cost increases as the number of iterations increase • Time consuming if error needs to be fixed in all units. • Requirements must be gathered upfront. (Geeks for Geeks, 2024)

High Level Design Choices

Layered Architecture

The Task Management System has a layered architecture consisting of a presentation layer, application layer, domain layer and a database layer:

- The presentation layer is the HTML, CSS, and JavaScript code for the Task Management System's front-end webpages with the UI consisting of forms, tables, navbars and responsiveness. This is where the users' interface with the application.
- The application layer is the Task Management System's main backend app.py file containing that Python Flask code to render the webpages, allow POST to send user data to the Task Management System's backend code and GET method for the user to receive data from the backend. This layer also enforces form validation.
- Domain layer – Form validation such as using regular expressions to make sure the password meets the requirements, password encryption, random number generation for session IDs, user authentication and Python code that executes SQL code to manage the database.
- Database Layer – the database.db SQLite file containing the Task Management System's relational database consisting of the users, tasks and projects tables and the SQL code that is executed by the Python code in app.py to manage the database. (Baeldung, 2021)

Client-Server Architecture

As the Task Management System is a web application, it has a client-server architecture where the backend code for core functionality (app.py), its supporting files, Flask framework, virtual environment, database, and image/container (docker deployments only) are all executed on a server either online (such as render.com) or locally on localhost/Docker Desktop. The front-end code

(HTML, CSS, and JavaScript) for the Task Management System's webpages are executed on the client's computer (web browser). POST and GET requests are used to communicate data between the client's computer and the server over the internet or local network. The front-end webpages (templates) are also stored on the server but are sent to the client's computer using HTTP or HTTPS protocol. (Geeks for Geeks, 2024)

Monolithic Architecture

The Task Management System is currently a monolithic application as they are quicker and easier to create in the early stages of software development. All the application's functions are executed in the main app.py file but has multiple functions (one for each web page to manage users, tasks, projects, and administrative tasks). Debugging, testing and deployment is also easier when you work with only one file than multiple files as is the case in microservice architecture.

However, there are some downsides with monolithic architecture:

- Unable to scale individual components if the demand increases
- Reliability issues – the entire Task Management System can fail if one critical component fails.
- Large monolithic applications are slower and harder to develop as the number of lines of code in the file increases.
- Difficult to adapt new technologies. (Harris, 2024)

Microservice Architecture

Although the Task Management System is currently a monolithic application and does not have microservice architecture, I might implement microservice architecture in the future if I had more time. This is because there are several advantages to microservice architecture:

- More scalable – individual components are easy to scale based on need.
- More maintainable and testable than monolithic architecture ad easier to experiment with new features.
- More reliable as the entire application does not fail if one component fails.
- Better support for Continuous Integration / Continuous Deployment (CI/CD) pipelines.
- Agility and faster development times for large projects. (Harris, 2024)

Object Oriented Programming

Although the main Task Management System does not use object-oriented programming, the unit testing Python code does use object-oriented programming as it has a TestRoutes class with a setup method and methods for each unit test.

Architectural Trade-offs

Monolithic Architecture	Layered Architecture	Client-Server Architecture
Unable to scale individual components if the demand increases	Framework does not allow for growth therefore scalability is difficult.	Requires internet connection
Reliability issues – the entire Task Management System can fail if one critical component fails.	Difficult to maintain as application is a single unit.	Task Management System is unavailable during server downtime
Large monolithic applications are slower and harder to develop as the number of lines of code in the file increases.	Each layer is dependent on the other layer.	Network congestion if too many users use it at once and server is not powerful enough or internet connection is too slow.
Difficult to adapt new technologies.	Not suitable for parallel processing.	
Entire application needs to be redeployed if a small change to a feature is made, which wastes time.		
Monolithic architecture is not flexible. (Harris, 2024)		

Software Antipatterns

Big Ball of Mud

Although I have tried to make the Task Management System's code maintainable there are issues with the code resulting in it being defined as a Big Ball of Mud as the application is monolithic and running code through automated system testing on Sonarcloud.io has detected a number of maintainability issues such as not using a constant variable for the database instead of using the database.db literal for the database file multiple times as this will have to be changed 53 times if the database changes or is renamed instead of doing it once. Other maintainability concerns have also been raised by Sonarcloud.io resulting in a Big Ball of Mud.

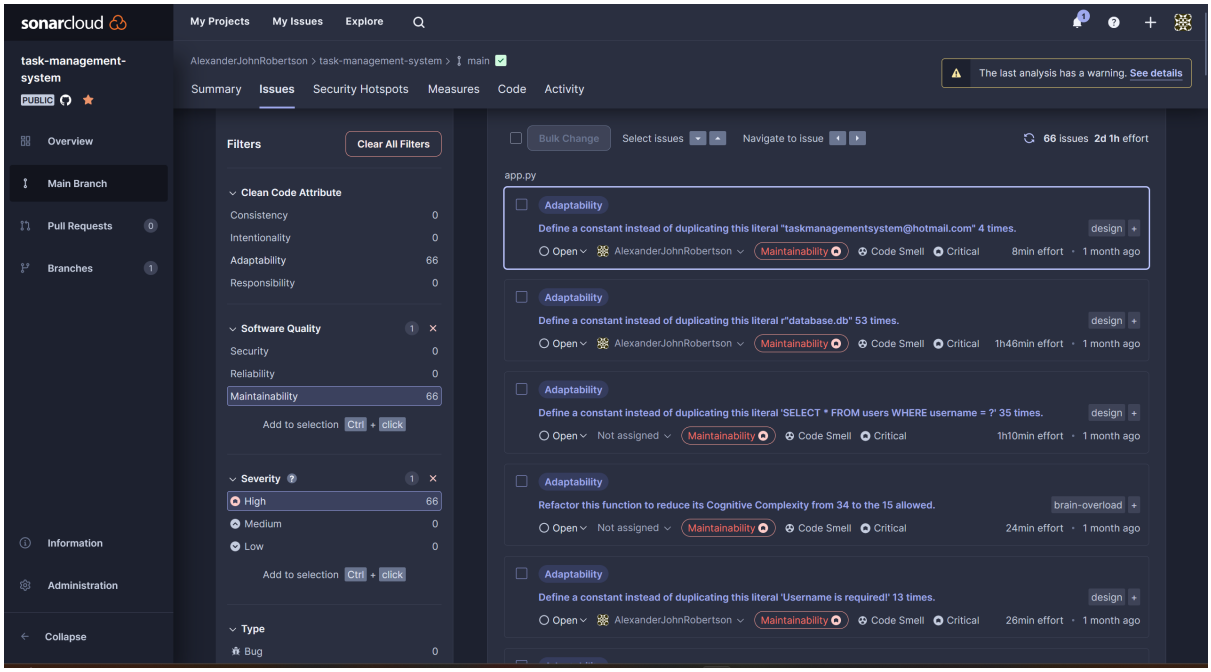


Figure 77 - Maintainability issues detected by SonarCloud during automated system testing - this is a sign of a Big Ball of Mud.

Spaghetti Code

There are some spaghetti code aspects in the Task Management System such as:

- Not using object orient programming in the main application file (app.py),
- only one class in the unit testing file, duplication in many functions in app.py,
- many if – elif – else statements after each other,
- comments could be better,
- lack of comments in HTML and CSS code,
- Lack of effort making HTML and CSS code understandable compared to the backend app.py code as the core functionality was my main priority over the user interface and due to time constraints.
- HTML and CSS code is harder to understand than the Python code.
- Code duplication in Python, CSS, HTML
- Conflicting code in CSS files
- Some functions are very long.

```
            flash('Password must contain at least one uppercase letter!')
        elif not checkNumber:
            flash('Password must contain at least one number!')
        elif not checkSpecialChar:
            flash('Password must contain at least one special character!')
        elif len(password) < 10: # minimum password length is 10 characters
            flash('Password must be at least 10 characters long!')
        elif password == username:
            flash('Password cannot be the same as the username!')
        elif password == firstname:
            flash('Password cannot be the same as the first name!')
        elif password == lastname:
            flash('Password cannot be the same as the last name!')
        elif password == email:
            flash('Password cannot be the same as the email!')
        elif password == phonenumber:
            flash('Password cannot be the same as the phone number!')
        elif password in global_var2(): #if the password is in the common passwords list flash an error message
            flash('Password is too common!')
        else:
            try:
                database = r"database.db" #database file
                conn = None
                conn = sqlite3.connect(database) #connecting to the database
                cur = conn.cursor()
                cur.execute( __sql: 'SELECT * FROM users WHERE username = ?', __parameters: (username,)) #querying the database
                user = cur.fetchall()
                cur.close()
                if user: #if the username already exists flash an error message
                    flash('Username already exists!')
                else:
                    if password != confirmPassword: #if the password and confirm password do not match flash an error message
                        flash('Passwords do not match!')
                    else:
                        for i in range(10): #password hashing encryption
                            password = hashlib.sha3_512(password.encode('utf-8'))
```

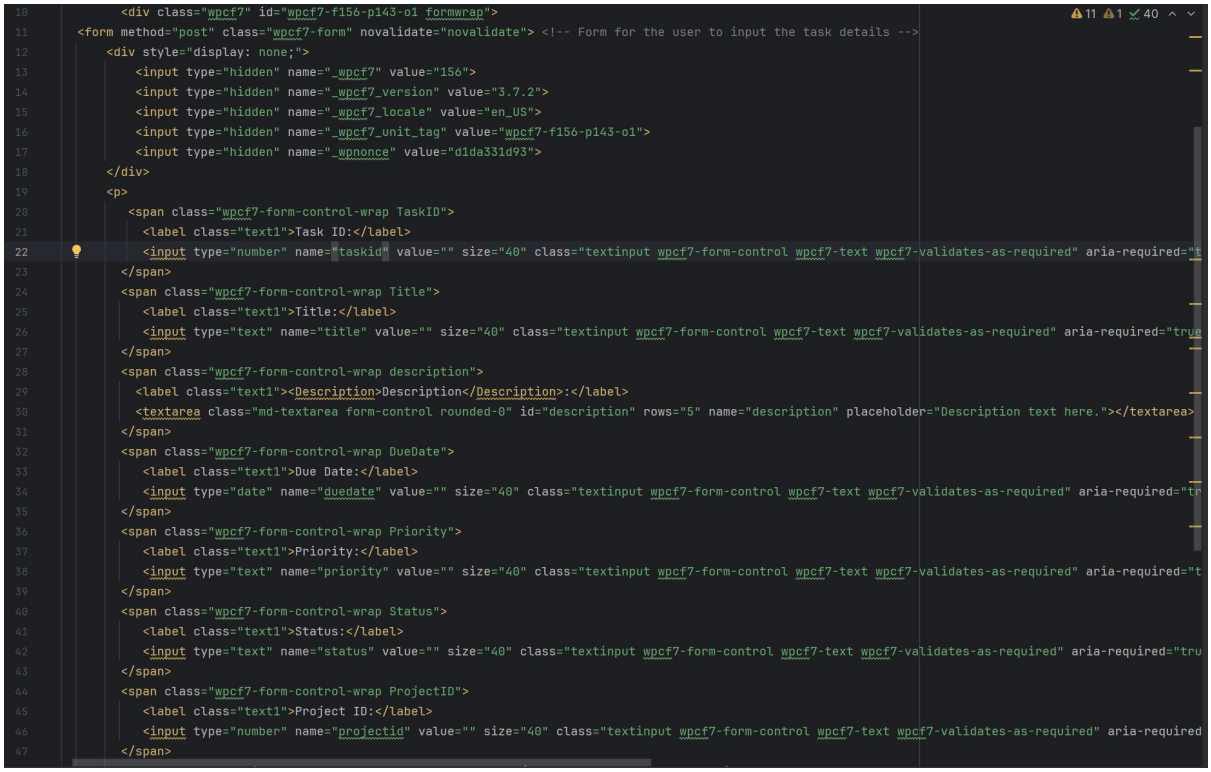
Figure 78 - Multiple if-elif-else statements in series and code duplication are often considered spaghetti code

```python
@app.route( rule: '/createaccount', methods=['GET', 'POST'])
def createaccount():
    '''This function allows the user to create an account.'''
    if request.method == 'POST': #get form details
        username = request.form['username']
        password = request.form['password']
        confirmPassword = request.form['confirmPassword']
        firstname = request.form['firstname']
        lastname = request.form['lastname']
        email = request.form['email']
        phonenumber = request.form['phonenumber']
        accountType = "Standard" #set the account type to standard
        blocked = "No" #set the blocked status to no
        failedLoginAttempts = 0 #set the failed login attempts to 0
        checkLowercase = re.search( pattern: r'[a-z]', password) #use regular expressions to validate the password
        checkUppercase = re.search( pattern: r'[A-Z]', password)
        checkNumber = re.search( pattern: r'[0-9]', password)
        checkSpecialChar = re.search( pattern: r'[^A-Za-z0-9]', password)
        if not firstname: #validate the form details
            flash('First Name is required!')
        elif not lastname:
            flash('Last Name is required!')
        elif not username:
            flash('Username is required!')
        elif not email:
            flash('Email Address is required!')
        elif not phonenumber:
            flash('Phone Number is required!')
        elif not username:
            flash('Username is required!')
        elif not password:
            flash('Password is required!')
        elif not confirmPassword:
            flash('Confirm Password is required!')
        elif not checkLowercase: #validate the password
            flash('Password must contain at least one lowercase letter!')
        elif not checkUppercase:
```

Figure 79 - Multiple if-elif-else statements in series and code duplication are often considered spaghetti code.

```python
                    password = password.hexdigest()
                sessionID = random.randint( a: 1000000000, b: 9999999999) #generate a random session ID
                cur = conn.cursor()
                cur.execute( _sql: 'INSERT INTO users VALUES(?, ?, ?, ?, ?, ?, ?, ?, ?, ?)', _parameters: (username, password, firstname, lastname,
                conn.commit()
                cur.close()
                resp = make_response(render_template('readcookie.html')) #setting the cookie connected to the username
                resp.set_cookie( key: 'userID', username)
                resp.set_cookie( key: 'sessionID', str(sessionID))
                return resp
        except IndexError: #if there is an error caused by incorrect login details flash an error message
            flash('Username or Password is incorrect!')
    return render_template('createaccount.html') #render the createaccount.html page
```

Figure 80 - Multiple if-elif-else statements in series and code duplication are often considered spaghetti code



```
      <div class="wpcf7" id="wpcf7-f156-p143-o1 formwrap">
<form method="post" class="wpcf7-form" novalidate="novalidate"> <!-- Form for the user to input the task details -->
    <div style="display: none;">
        <input type="hidden" name="_wpcf7" value="156">
        <input type="hidden" name="_wpcf7_version" value="3.7.2">
        <input type="hidden" name="_wpcf7_locale" value="en_US">
        <input type="hidden" name="_wpcf7_unit_tag" value="wpcf7-f156-p143-o1">
        <input type="hidden" name="_wpnonce" value="d1da331d93">
    </div>
    <p>
       <span class="wpcf7-form-control-wrap TaskID">
         <label class="text1">Task ID:</label>
         <input type="number" name="taskid" value="" size="40" class="textinput wpcf7-form-control wpcf7-text wpcf7-validates-as-required" aria-required="t
       </span>
       <span class="wpcf7-form-control-wrap Title">
         <label class="text1">Title:</label>
         <input type="text" name="title" value="" size="40" class="textinput wpcf7-form-control wpcf7-text wpcf7-validates-as-required" aria-required="true
       </span>
       <span class="wpcf7-form-control-wrap description">
         <label class="text1"><Description>Description</Description>:</label>
         <textarea class="md-textarea form-control rounded-0" id="description" rows="5" name="description" placeholder="Description text here."></textarea>
       </span>
       <span class="wpcf7-form-control-wrap DueDate">
         <label class="text1">Due Date:</label>
         <input type="date" name="duedate" value="" size="40" class="textinput wpcf7-form-control wpcf7-text wpcf7-validates-as-required" aria-required="tr
       </span>
       <span class="wpcf7-form-control-wrap Priority">
         <label class="text1">Priority:</label>
         <input type="text" name="priority" value="" size="40" class="textinput wpcf7-form-control wpcf7-text wpcf7-validates-as-required" aria-required="t
       </span>
       <span class="wpcf7-form-control-wrap Status">
         <label class="text1">Status:</label>
         <input type="text" name="status" value="" size="40" class="textinput wpcf7-form-control wpcf7-text wpcf7-validates-as-required" aria-required="tru
       </span>
       <span class="wpcf7-form-control-wrap ProjectID">
         <label class="text1">Project ID:</label>
         <input type="number" name="projectid" value="" size="40" class="textinput wpcf7-form-control wpcf7-text wpcf7-validates-as-required" aria-required
       </span>
```

Figure 81 - The HTML code for the frontend webpages is spaghetti code as I find it less understandable than the Python code and due to a lack of comments.

```css
/* This is the CSS for styling the responsive CRUD forms */
/* Original CSS sourced from codepen.io and modified by S275931 to suit the application.
@import url(https://fonts.googleapis.com/css?family=Source+Sans+Pro); /* Import fonts */
body { /* style for body */
      background: #3c3f3f;
    -webkit-background-size: cover;
      -moz-background-size: cover;
      -o-background-size: cover;
      background-size: cover;
    padding-top: 0px;
}

h1 { /* style for h1 */
  color: #fff;
   text-shadow: 1px 1px 0 rgba(0,0,0,0.4);
   padding-top: 10px;
   font-size: 100px;
   font-weight: 700;
   text-align: center;
   font-family: 'Source Sans Pro', sans-serif;
   margin: 0px;
}

form { /* style for form */
    margin-left:auto;
    margin-right:auto;
    width: 965px;
    height: 100%;
    padding:10px;
    -moz-border-radius: 10px;
    -webkit-border-radius: 10px;
    border-radius: 10px;
    -moz-background-clip: padding;
    -webkit-background-clip: padding-box;
    background-clip: padding-box;
    overflow: scroll;
}
```

Figure 82 - The CSS code is also spaghetti code as it is harder for me to understand than the Python and HTML code, due to a lack of comments, conflicting code. The code was originally from Codepen then modified by me to suit the application however the original was also spaghetti code as I had a lot of difficulty understanding it and getting it to work.

Golden Hammer

The Task Management System is very reliant on the Flask Framework, the backend is primarily written in Python, relies heavily on SQLite for database management. Frontend relies heavily on HTML, CSS, and RESTful APIs.

Continuous integration and continuous delivery/deployment aspects

The Task Management System incorporates a Continuous Integration / Continuous Deployment (CI/CD) Pipeline for the online Render.com deployment. The CI/CD pipeline is illustrated below:

Make changes to code in PyCharm → Commit and push to the GitHub repository using Git from within the PyCharm Terminal → The changes are uploaded to the GitHub repository → Render.com automatically detects the new push to the GitHub repository → Render.com automatically rebuilds the Task Management System web app from the GitHub repository → Render.com automatically redeploys the Task Management System after rebuilding it → The new version of the Task Management System is now live and ready for production and use. Therefore, I have implemented the Plan (Wireframes, Mockups, UML diagrams, requirements), Code (PyCharm, Python, HTML, CSS, SQL, JavaScript, GitHub), Build (Docker, Render.com), Test (PyCharm, Python, Postman, SonarCloud, Acceptance Testing), Release and Deploy (Render.com, Docker, DockerHub) and Operate (Use) phases of the DevOps cycle.

Screenshots below:

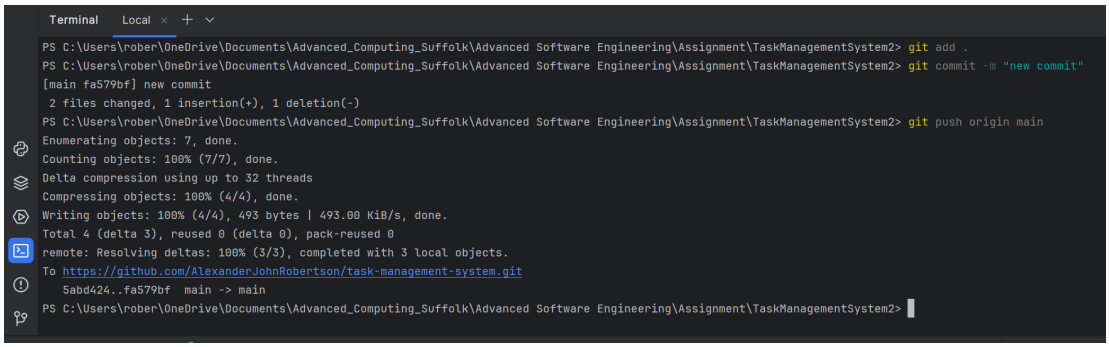


Figure 83 - Committing to the GitHub repository using Git commands in the terminal in PyCharm.

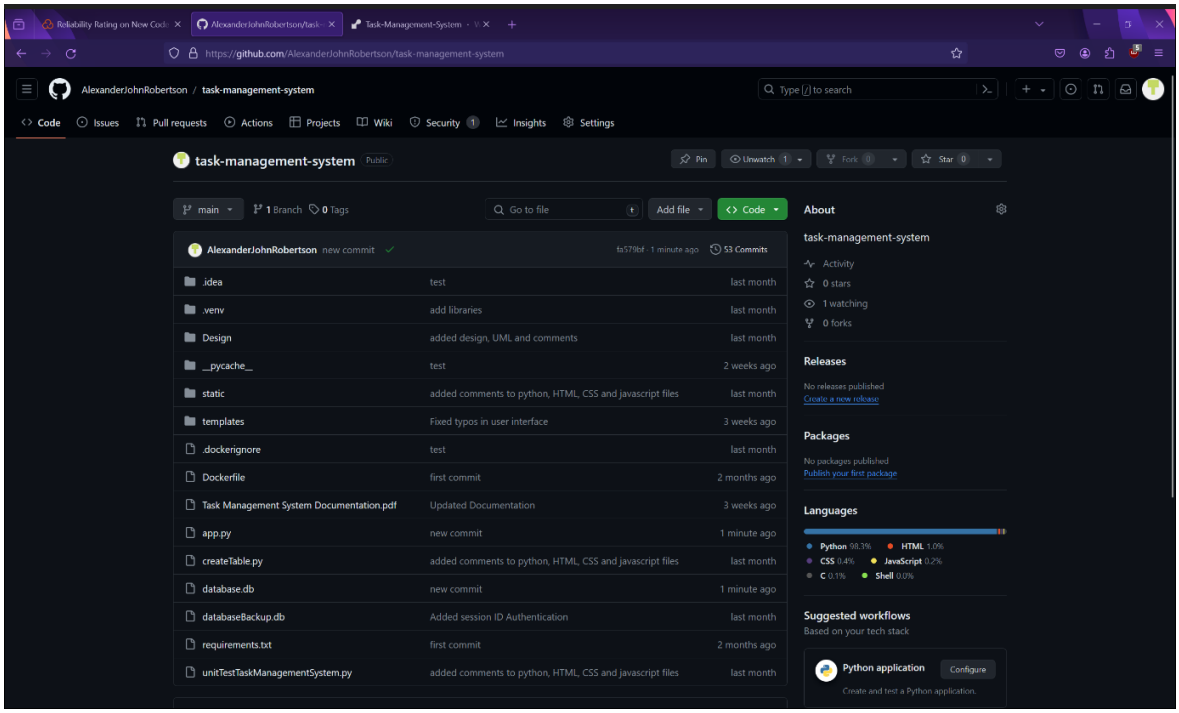


Figure 84 - The updated GitHub repository.

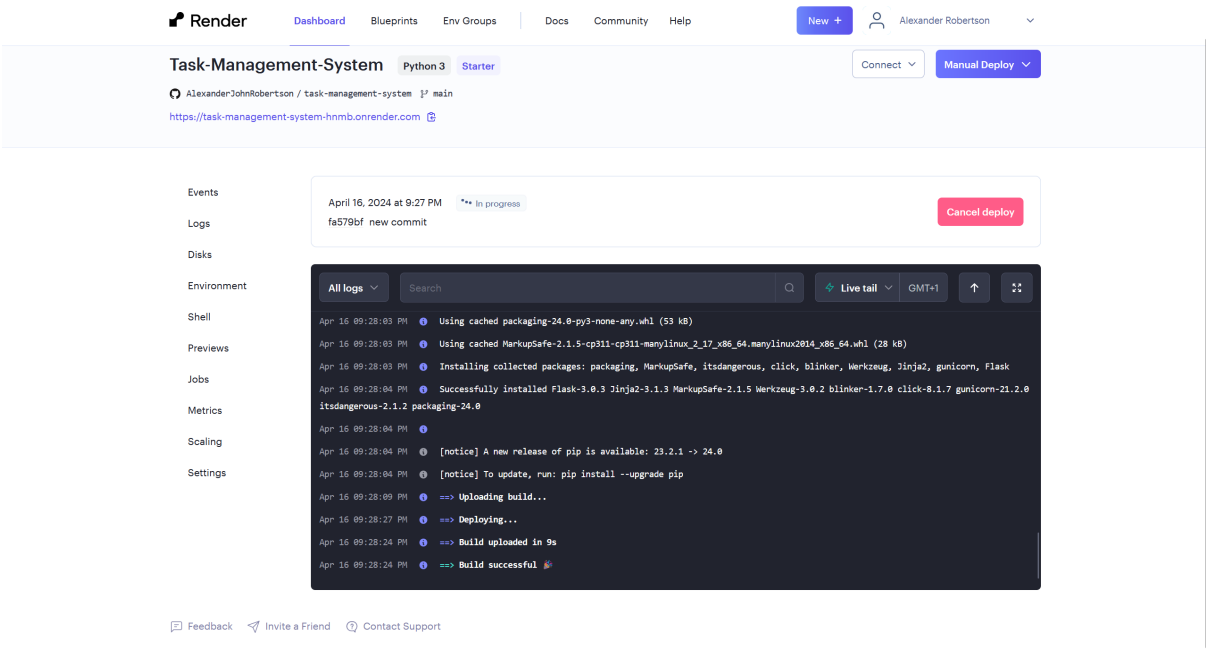


Figure 85 - CI/CD Pipeline - Render.com automatically detects the new push to the GitHub repository and automatically rebuilds the Task Management System website.

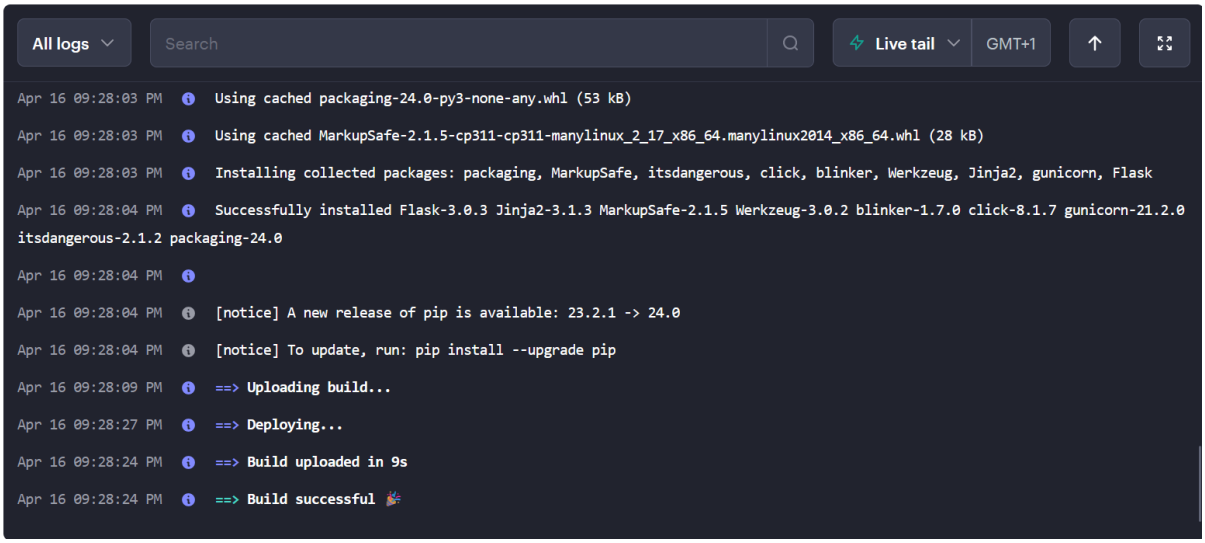


Figure 86 - Render.com automatically rebuilding the Task Management System.

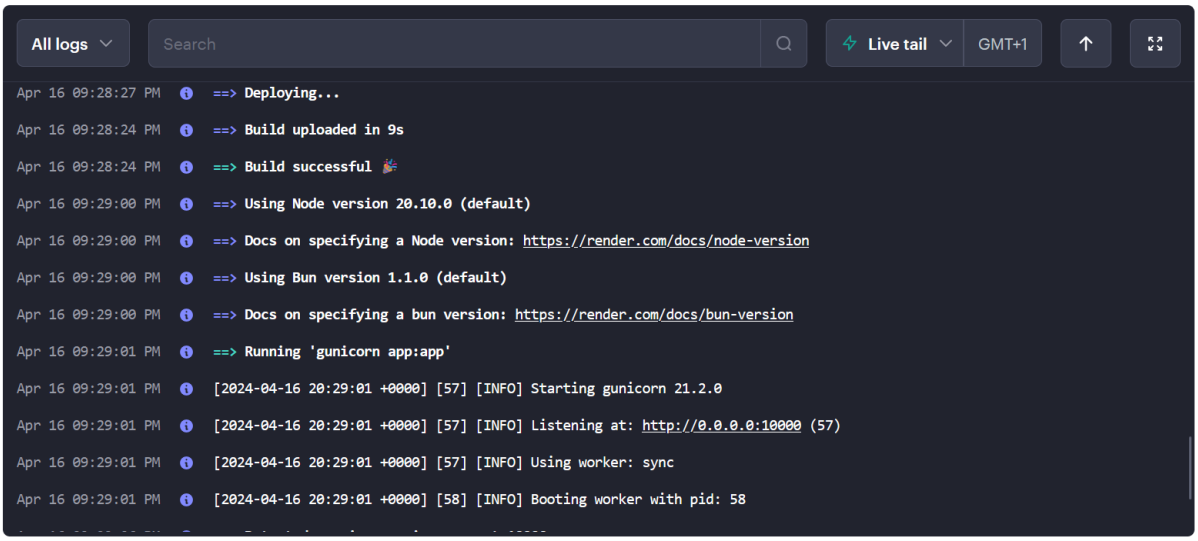


Figure 87 - Render.com automatically redeploying the Task Management System.

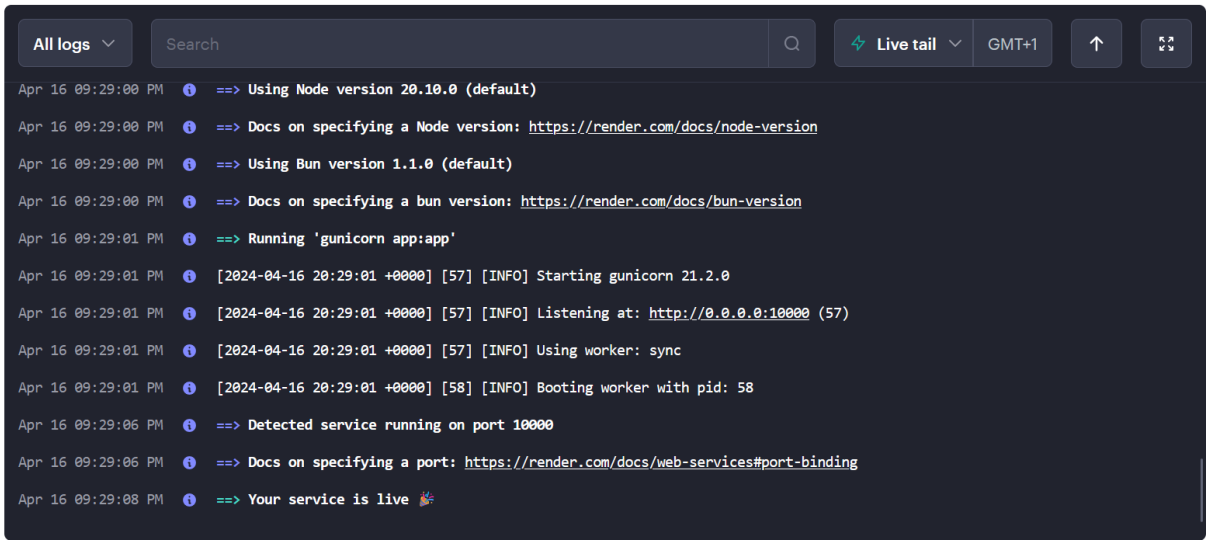


Figure 88 - Automatic redeploy completed and latest Task Management System is now live

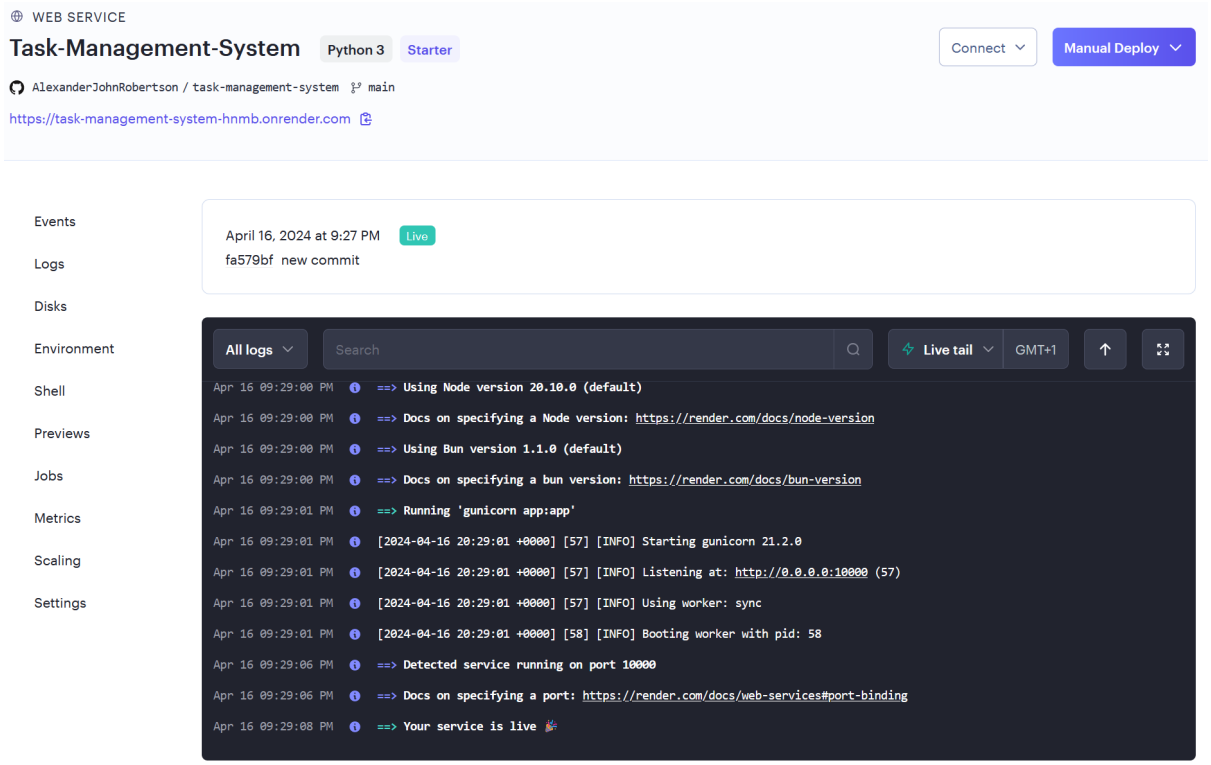


Figure 89 - Automatic redeploy completed and latest Task Management System is now live.

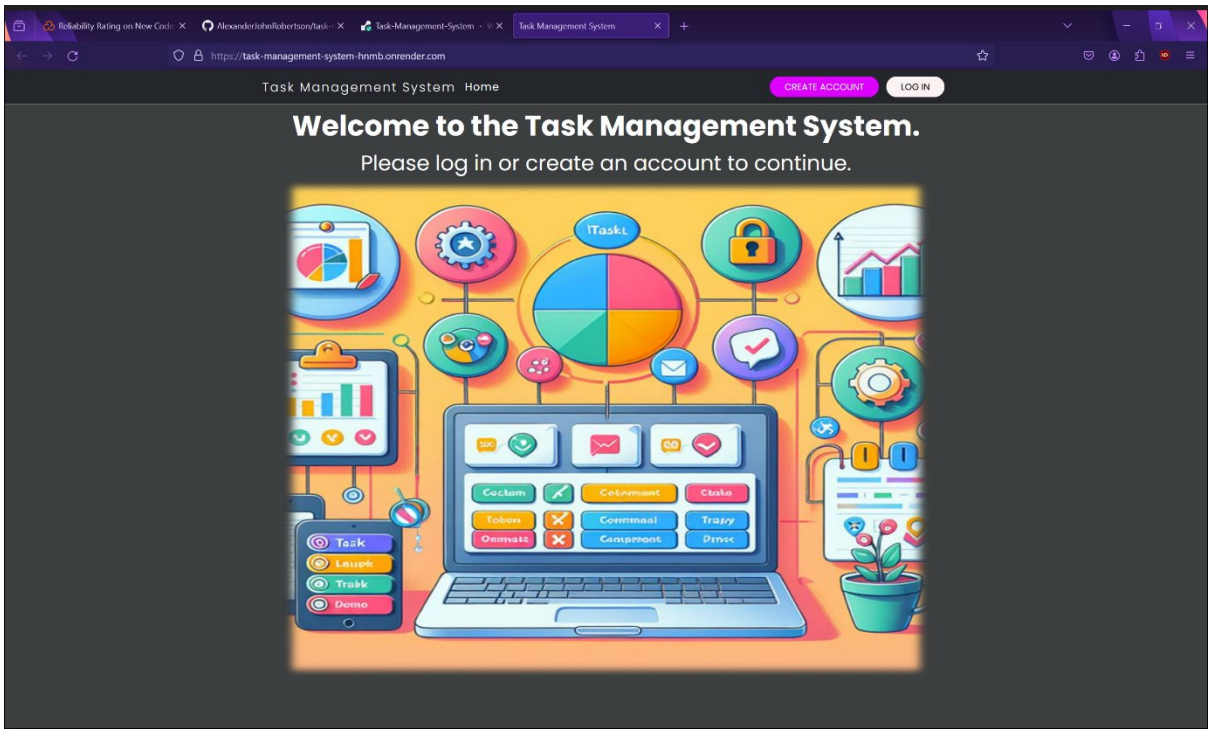


Figure 90 - Task Management System live landing page on Render.com.

I am also deploying the Task Management System as Docker images/containers however I am deploying the application manually as I am using the free version of Docker and CI/CD pipelines are only available in paid versions of Docker.

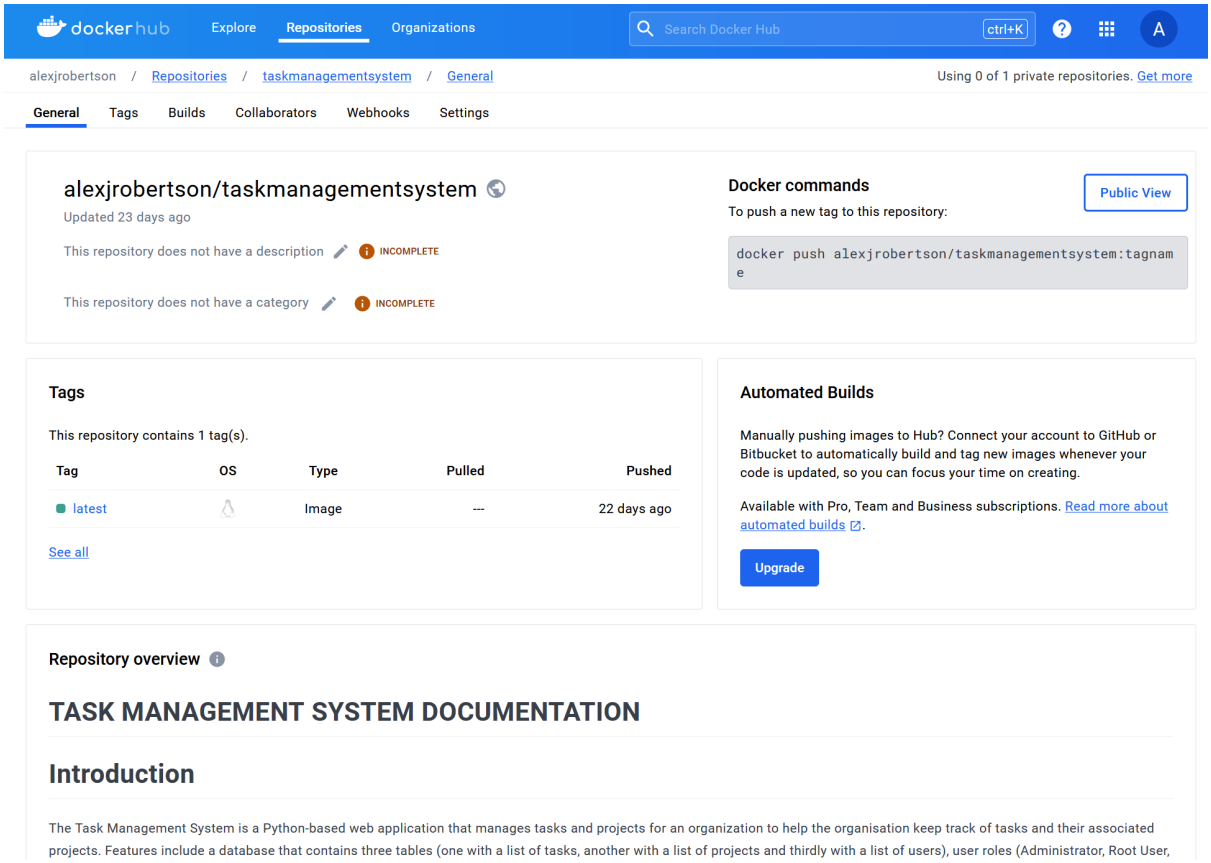


Figure 91 - Task Management System deployed on DockerHub as a container image.

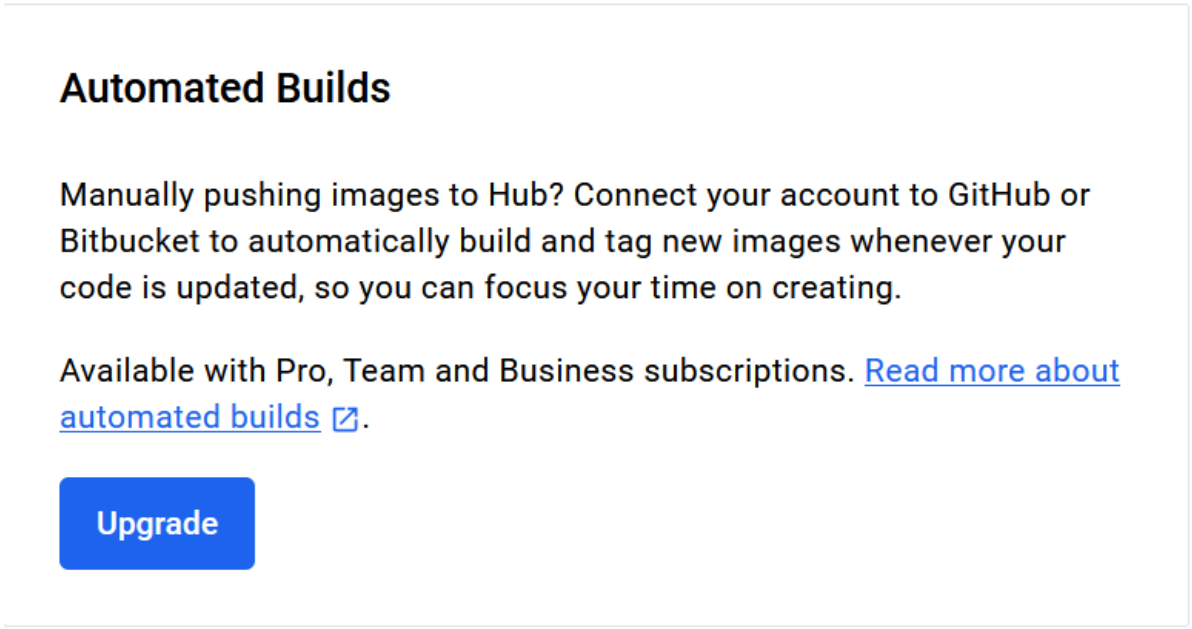


Figure 92 - You must pay a fee to use automated CI/CD pipelines on DockerHub.

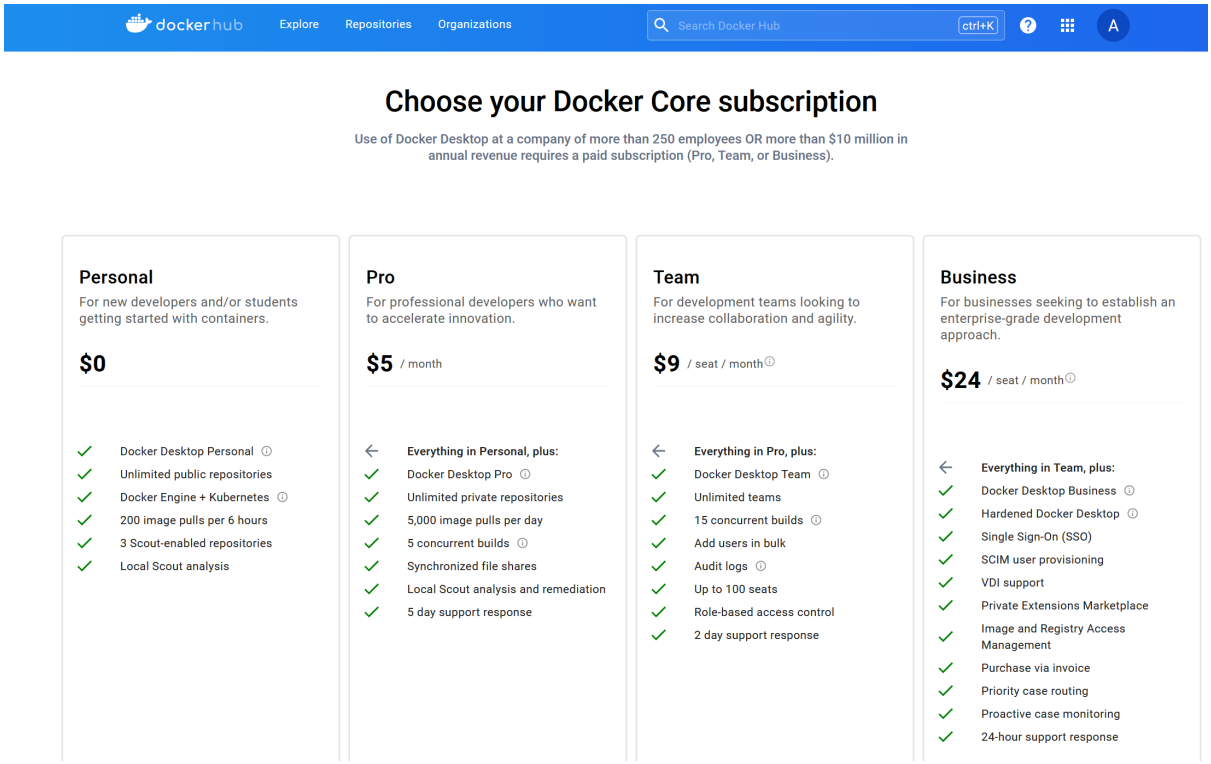


Figure 93 - You must pay a subscription fee to use automated CI/CD pipelines on DockerHub.

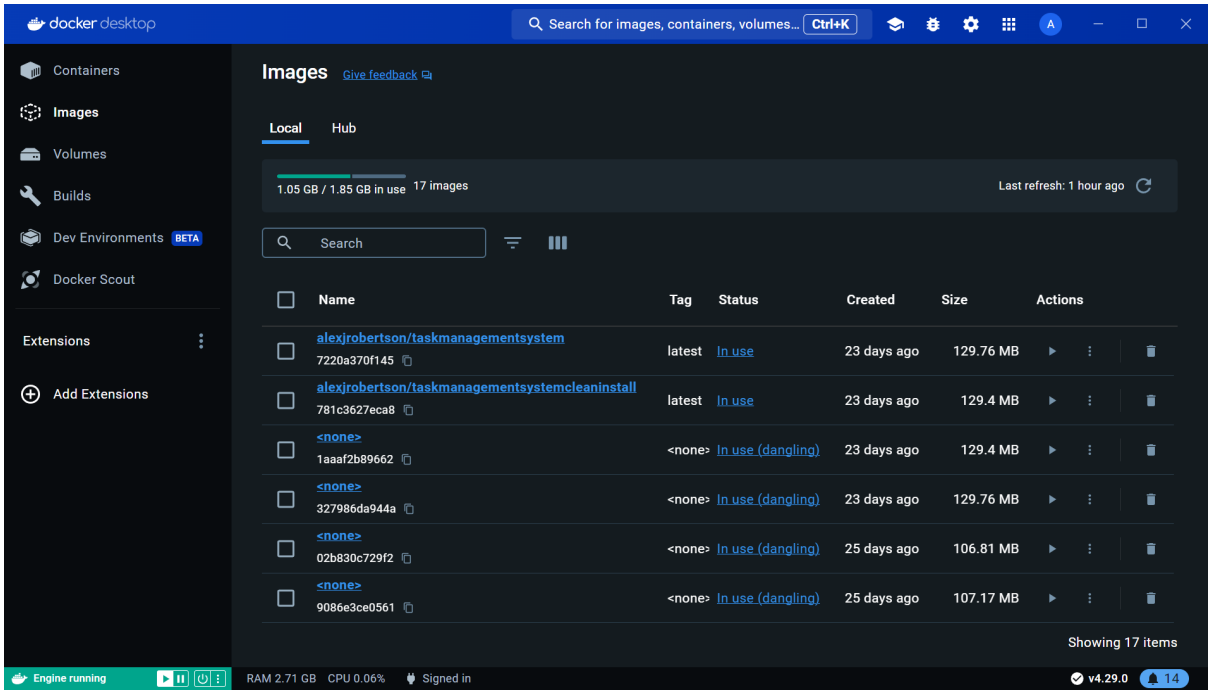


Figure 94 - Task Management System container images built on Docker Desktop.

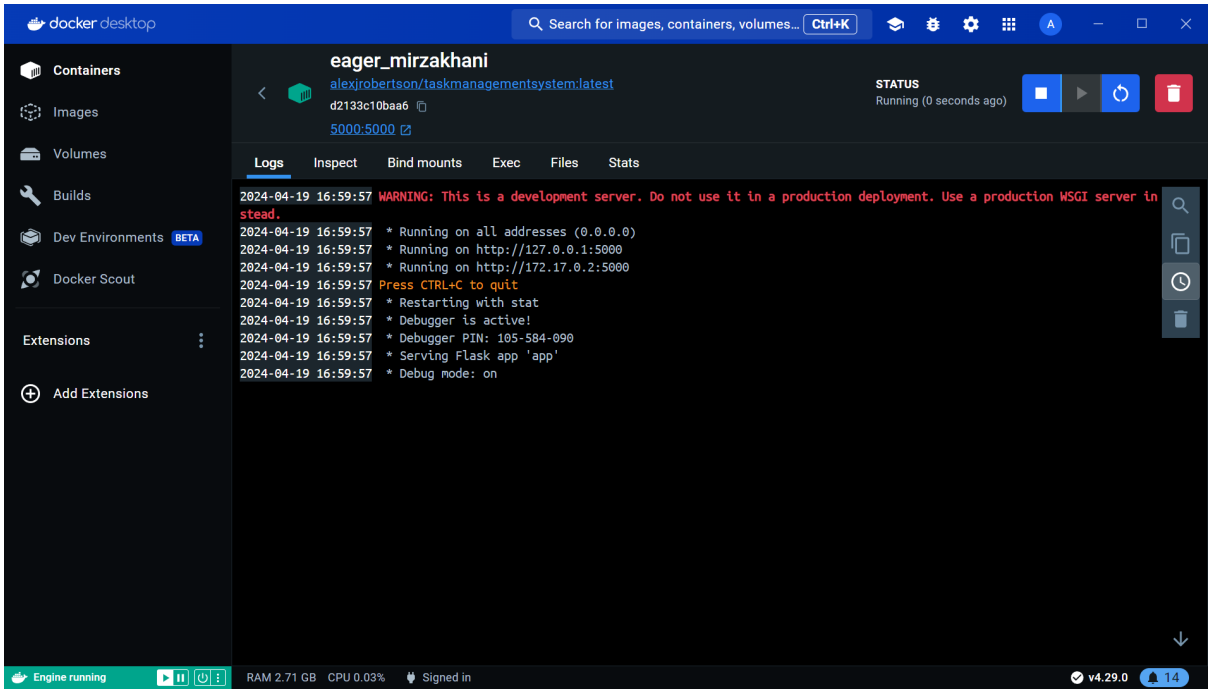


Figure 95 - Task Management System container deployed and running on Docker Desktop.

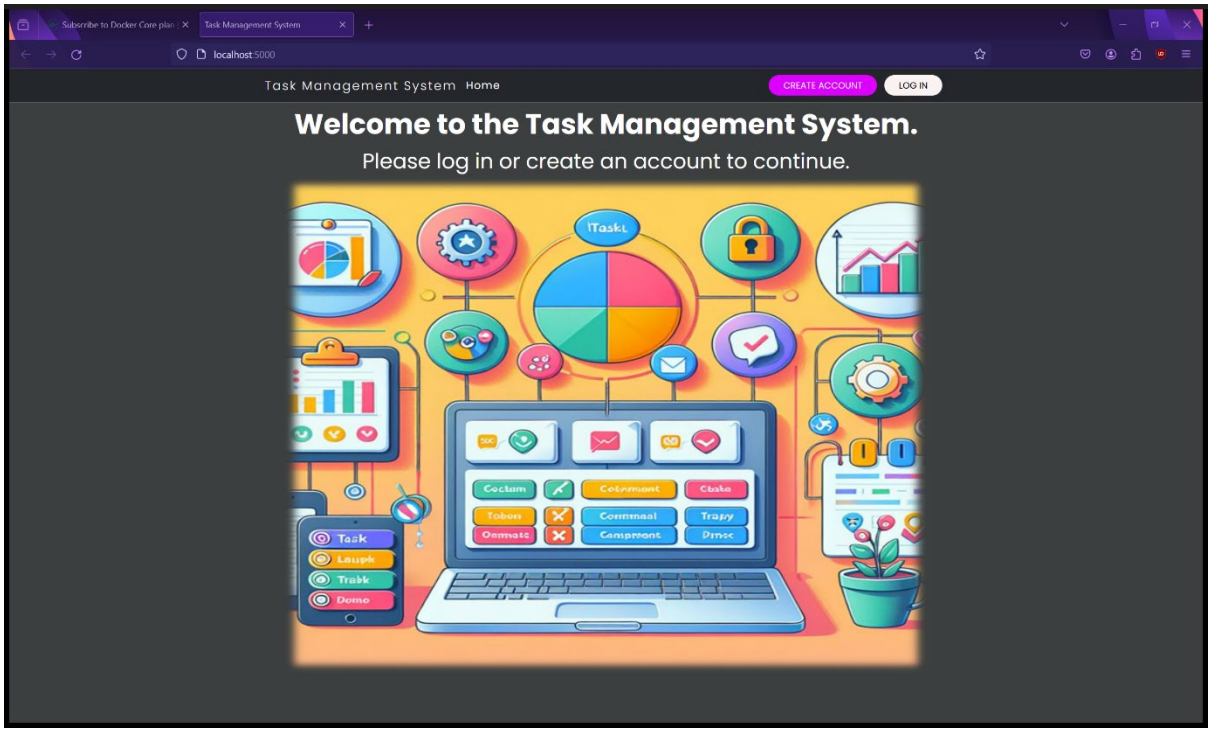


Figure 96 - Task Management System landing page accessible through a web browser from the Docker container.

Testing- Unit/Integration and system testing/ Acceptance testing

Unit Testing

I have created a separate Python file with code to perform unit testing on the main Task Management System Python file (app.py) to test that all the webpages are available by testing that the endpoint routes are available. The unit test file has a class to test the routes and, a setup method to set up the app for testing and individual methods within the class to test each route for each webpage / function individually. However, most of the unit test functions test for "Access Denied" instead of the normal content on these webpages as you must be logged in to access the webpages or else an access denied error appears as the test code is not able to log in or create an account however this still proves that the endpoints word. The Task Management System passed all the unit tests as the actual outcome matches the expected outcome.

app.py | addtask.html | unitTestTaskManagementSystem.py | style2.css

```python
import unittest
from app import app

'''Unit test for the Task Management System'''

AlexanderJohnRobertson +1
class TestRoutes(unittest.TestCase):
    AlexanderJohnRobertson +1
    def setUp(self): # Set up the app for testing
        self.app = app.test_client()

    AlexanderJohnRobertson +1
    def test_login(self): # Test the login route
        response = self.app.get('login')
        self.assertEqual(response.status_code, second: 200)
        self.assertIn( member: b'Log', response.data)

    AlexanderJohnRobertson +1
    def test_create_account(self): # Test the create account route
        response = self.app.get('createaccount')
        self.assertEqual(response.status_code, second: 200)
        self.assertIn( member: b'Create', response.data)

    AlexanderJohnRobertson +1
    def test_add_task(self): # Test the add task route
        response = self.app.get('addtask')
        self.assertEqual(response.status_code, second: 200)
        self.assertIn( member: b'Access Denied', response.data)

```

Figure 97 - Unit Testing Python source code.

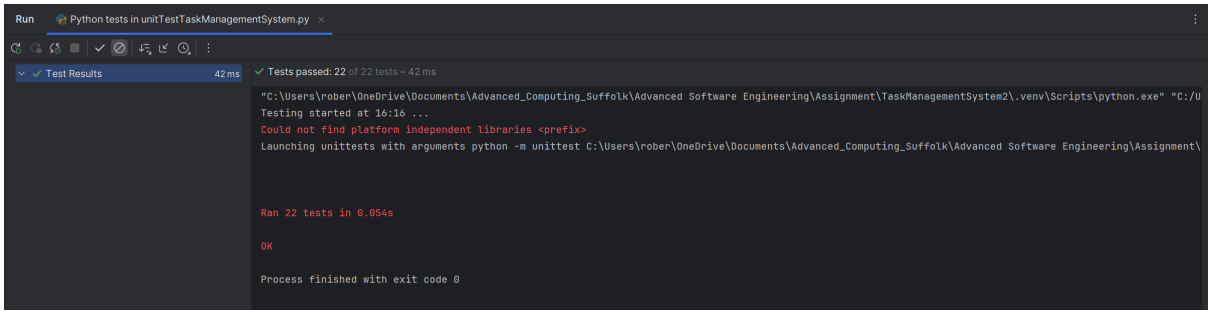


Figure 98 - Task Management System passes unit testing as the actual outcome is the same as the expected outcome as defined by the unit testing code.

When the unit testing fails when the actual outcome does not meet the expected outcome (I changed the expected outcome on one of the unit test functions to induce a failure) a FAILED error message appears with a description of the failure and where it is located. A tree directory appears in the test results pane directing to the unit that failed.

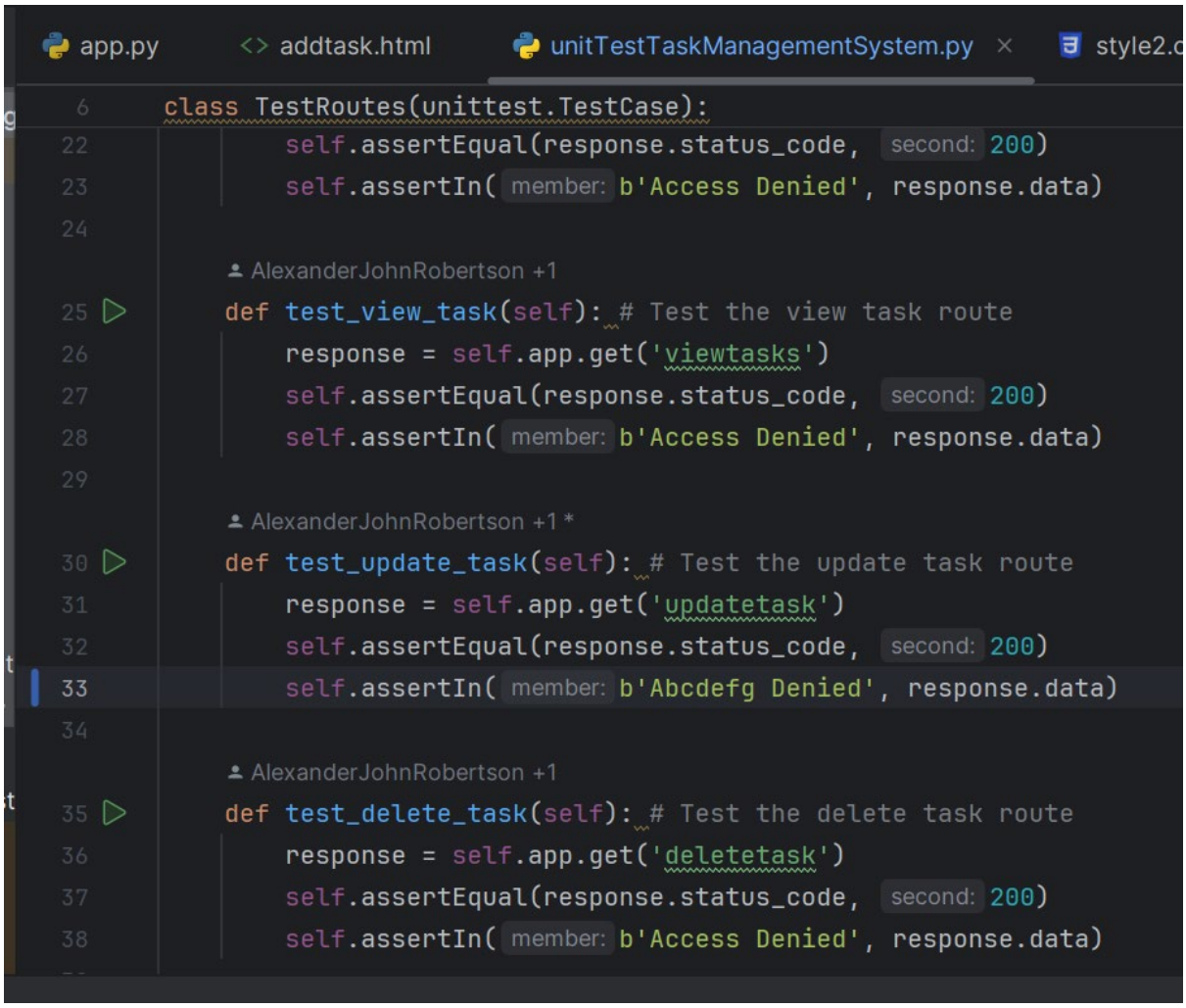


Figure 99 - Modifying the unit testing source code to change the expected outcome will cause the Task Management System to fail unit testing.

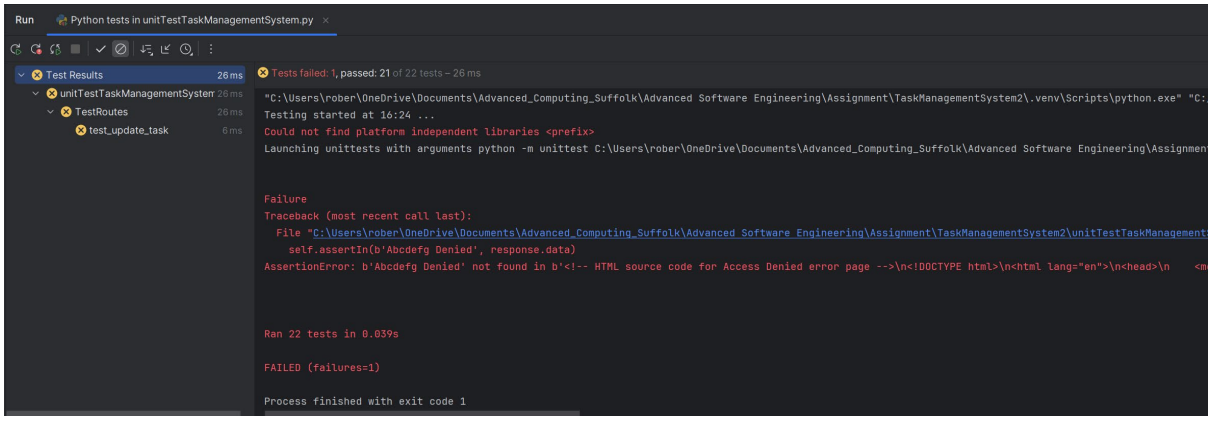


Figure 100 - Task Management System has now failed unit testing as the actual outcome no longer matches the expected outcome. A tree is provided directing the developer to the test that failed.

I have also performed RESTful GET API unit testing with Postman for each Task Management System webpage. Postman shows the HTML code that appears when a GET request is sent to the webpages' URLs, provides a preview of the webpage without the CSS, provides a bot generated visualisation of the response and automated testing. When testing pages that require authentication using Postman, an "Access Denied" error is returned instead of the actual content as Postman is attempting to access these webpages without being logged in which is not allowed due to security measures.

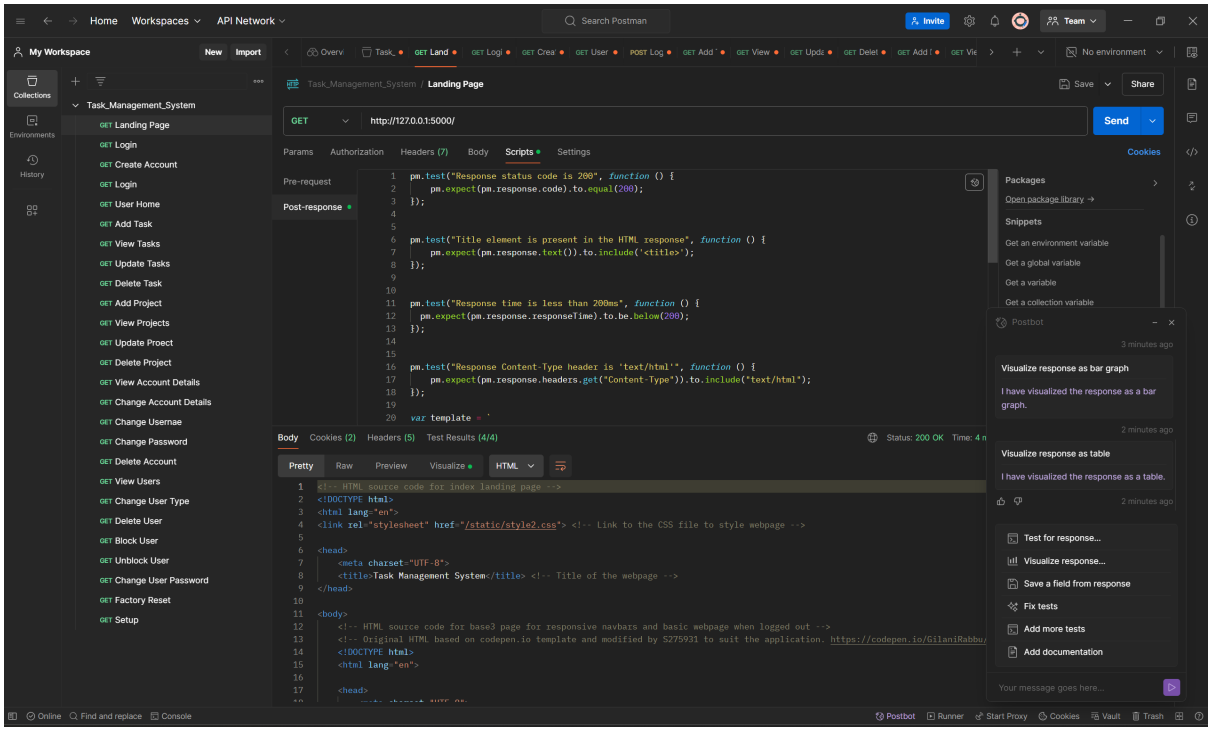


Figure 101 - Using Postman to perform RESTful API testing for GET requests. This test is for the landing page and returns the Task Management System's landing page HTML code.

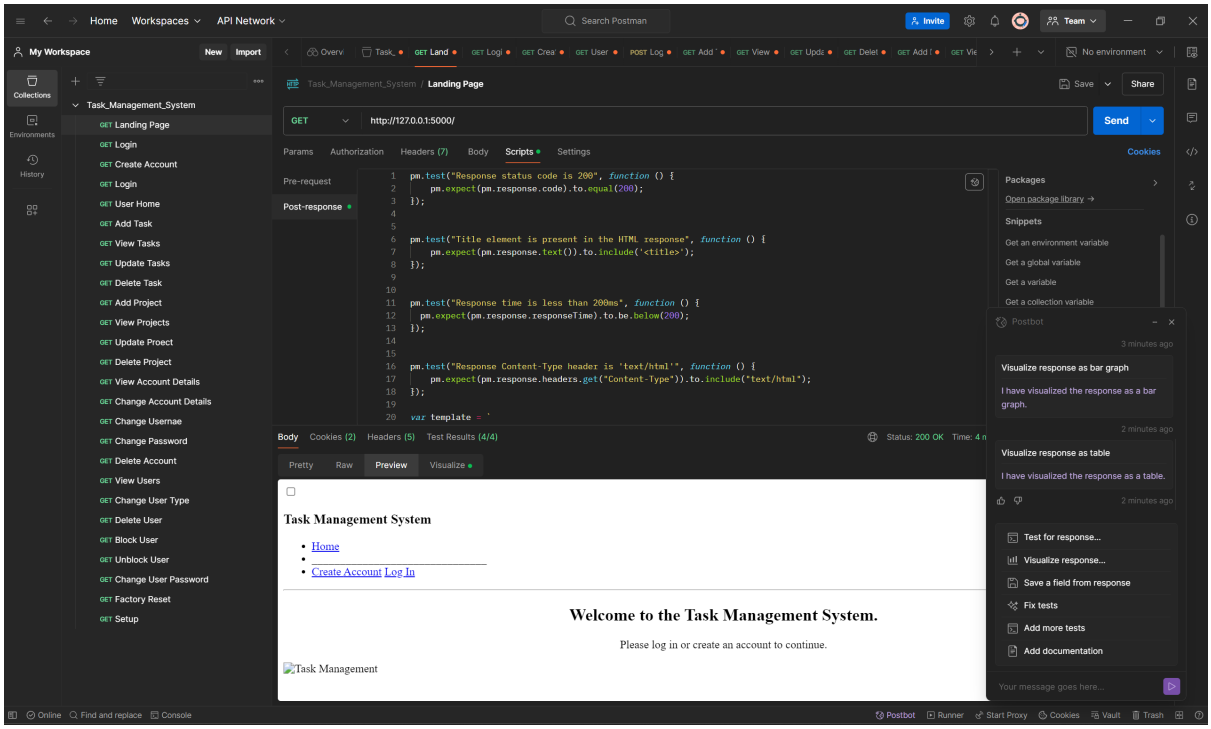


Figure 102 - Postman generating a preview of the HTML code from the returned response however the CSS is not supported.

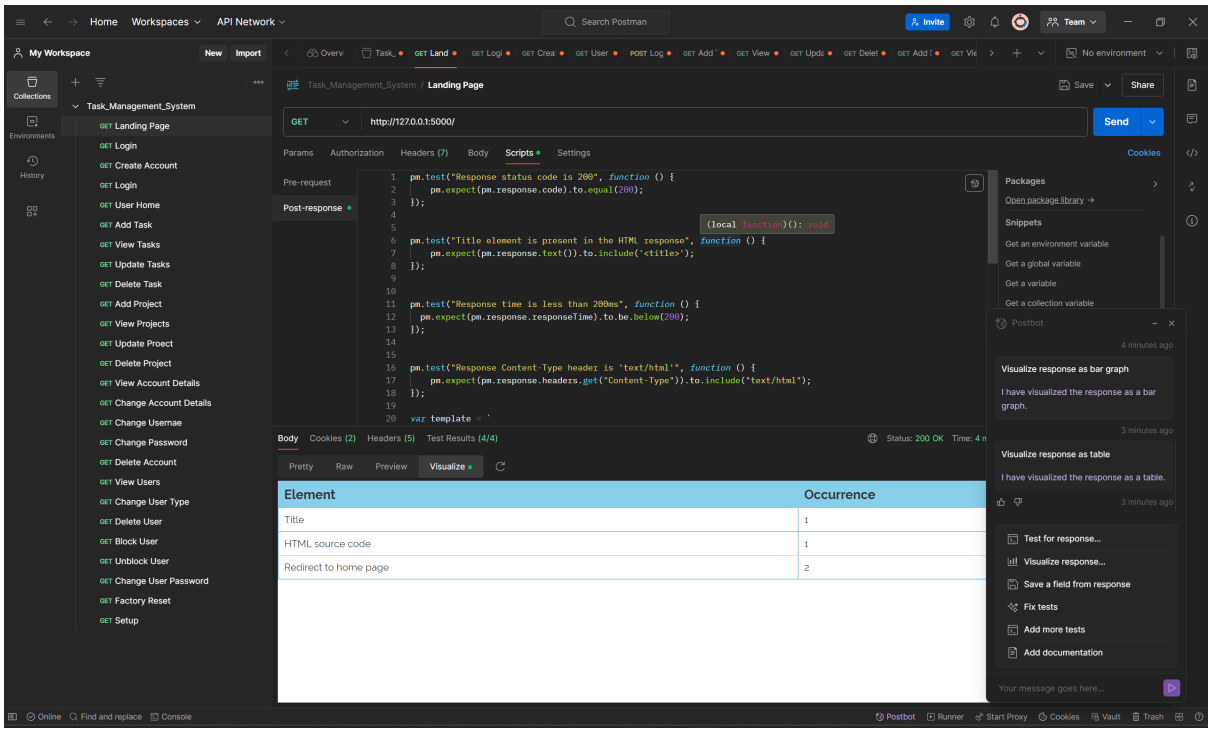


Figure 103 - Visualisation of the response from the Task Management System during the API test by Postman.

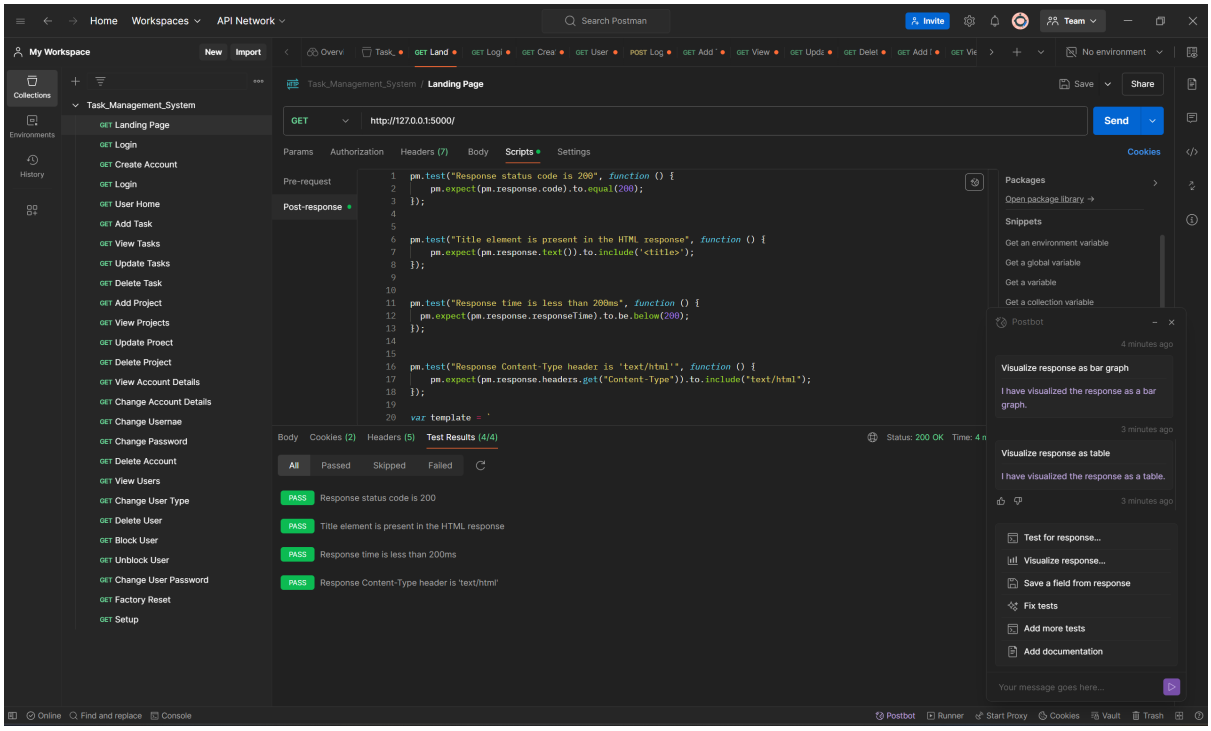


Figure 104 - Running automated testing for landing page (tests automatically generated by PostBot). All tests have passed.

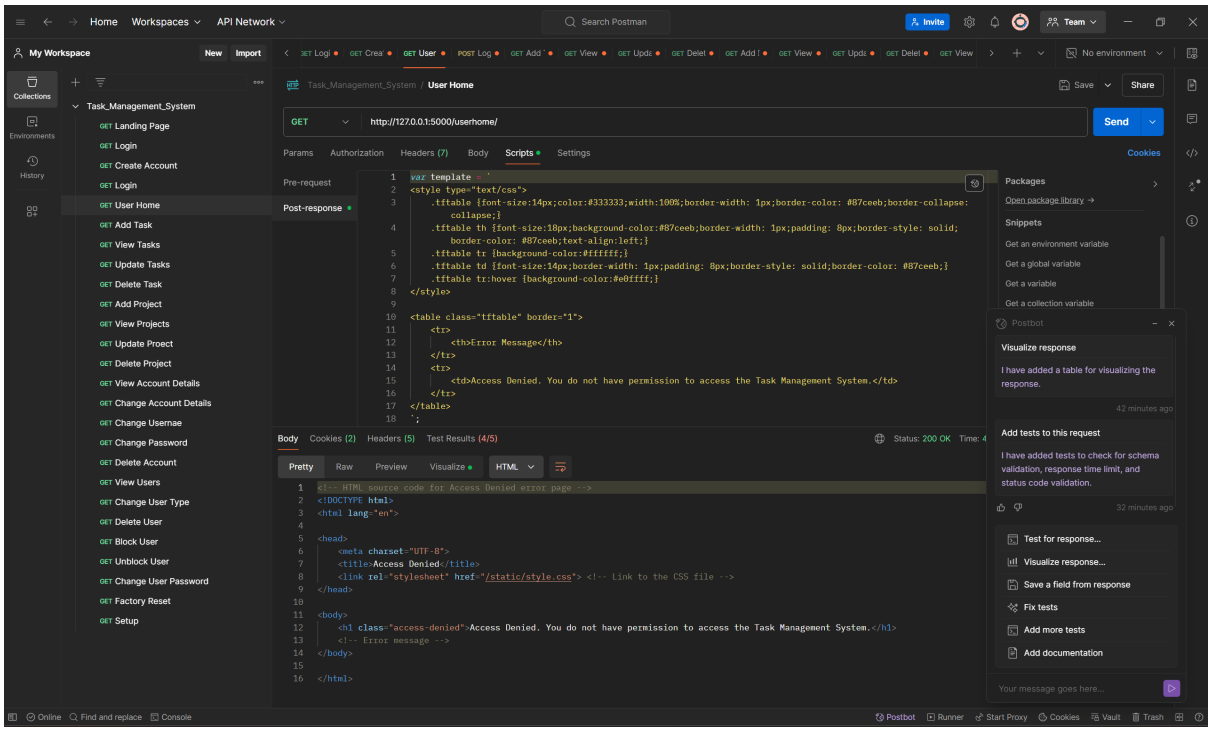


Figure 105 - Testing the User Home page using Postman however it returns the Access Denied error page as a response as authentication is required by logging in therefore the Task Management System's security code detects Postman as unauthorised access.

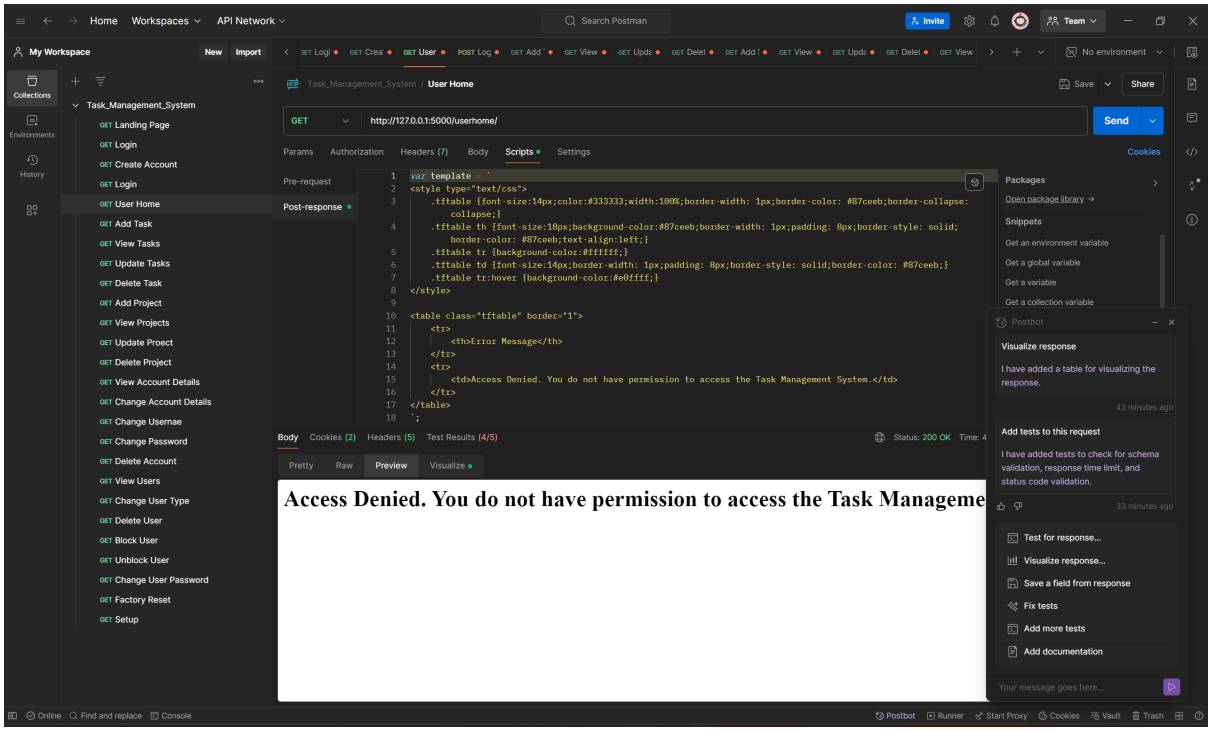


Figure 106 - Preview of Access Denied error page instead of User Home page because the access by Postman is unauthorised however this proves that the authentication system is working.

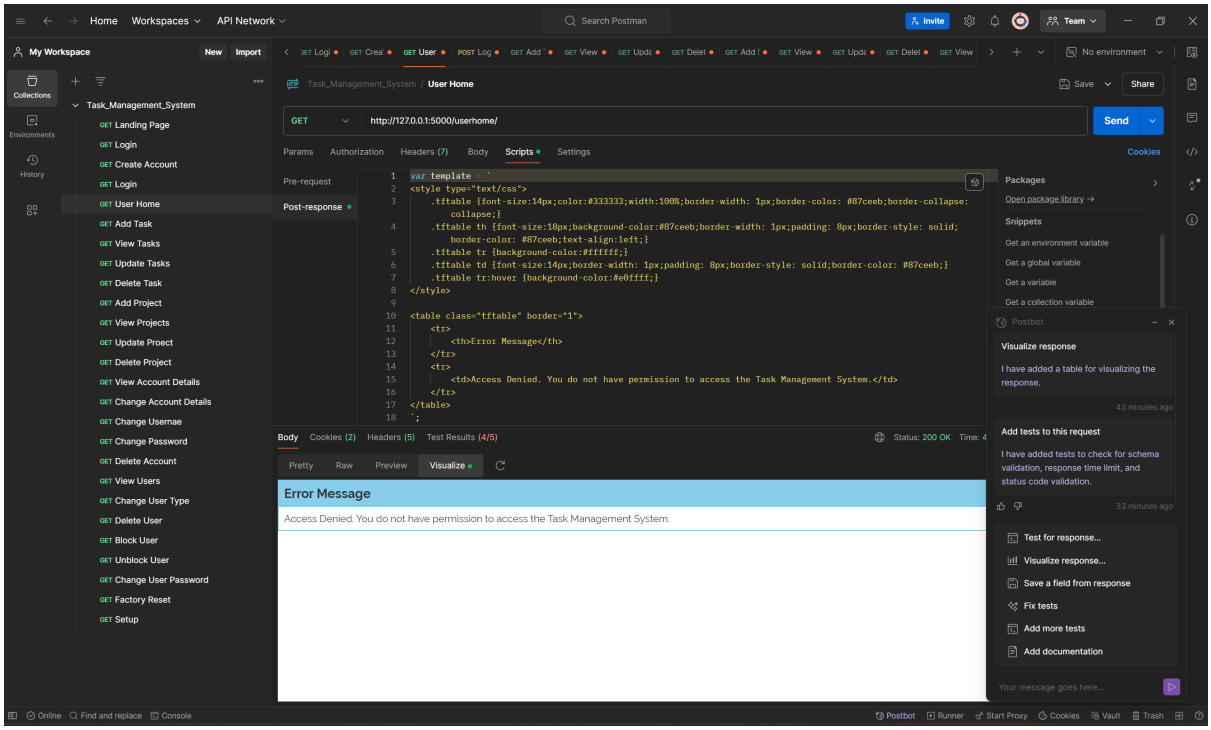


Figure 107 - Visualisation of the response.

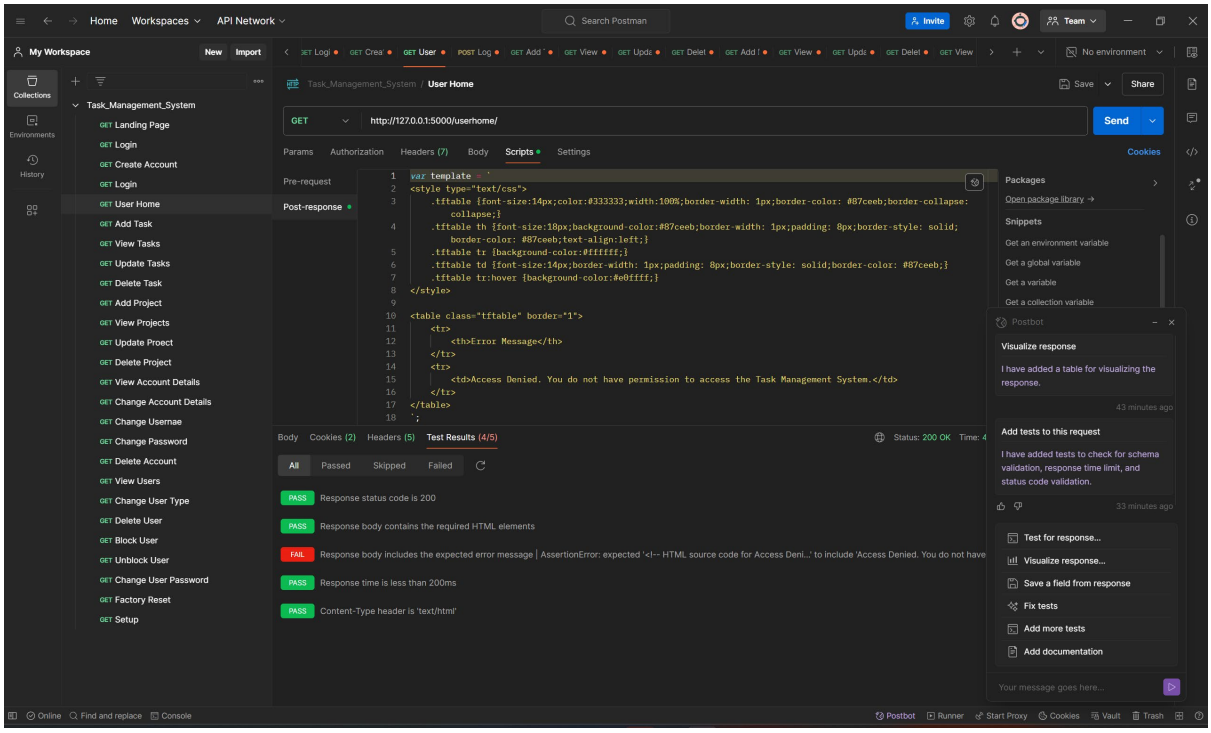


Figure 108 - One of the Postman tests fails because it gave an Access Denied error instead of allowing access to the actual page.

Integration and System Testing

I have used SonarCloud (an online DevOps testing tool that uses a CI/CD pipeline to link to the GitHub repository) to perform system testing on the Task Management System. This included automated testing for security threats, vulnerabilities, code smells, software quality, bugs, consistency, reliability, maintainability, antipatterns, spaghetti code etc. SonarCloud then shows where these errors are then suggests how to remediate these errors.

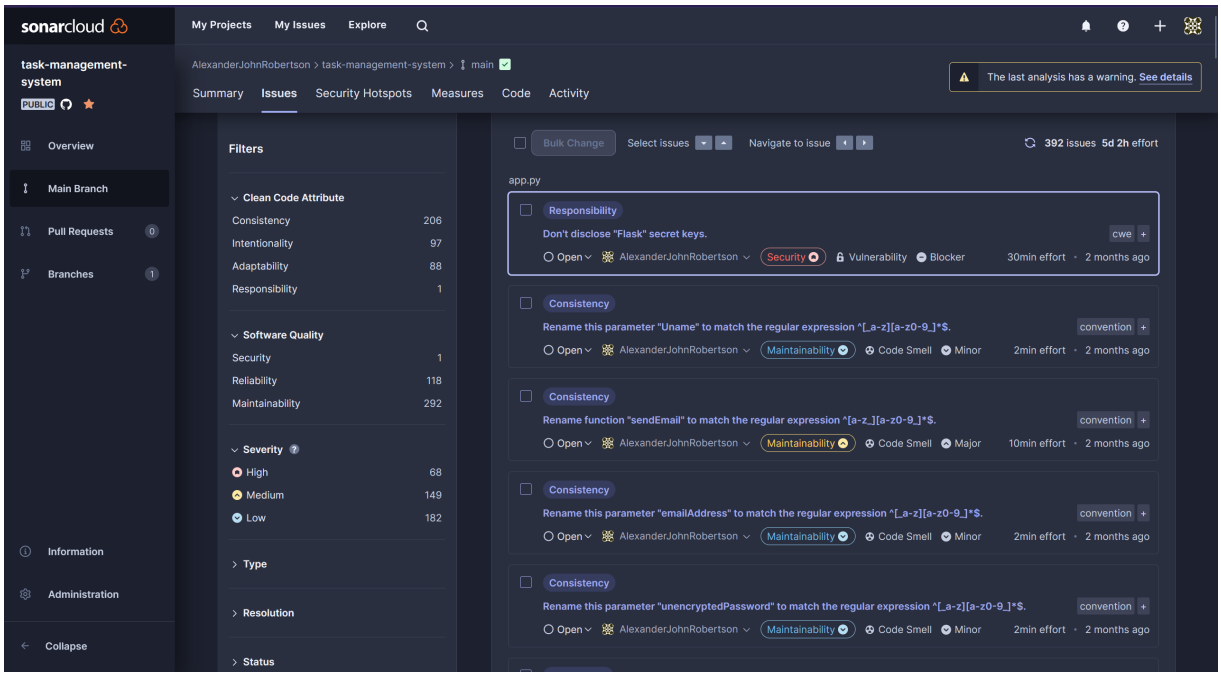


Figure 109 - Using SonarCloud to perform automated system testing to detect security issues, vulnerabilities, bugs, code smells, software quality issues, clean code, antipatterns, maintainability and reliability issues.

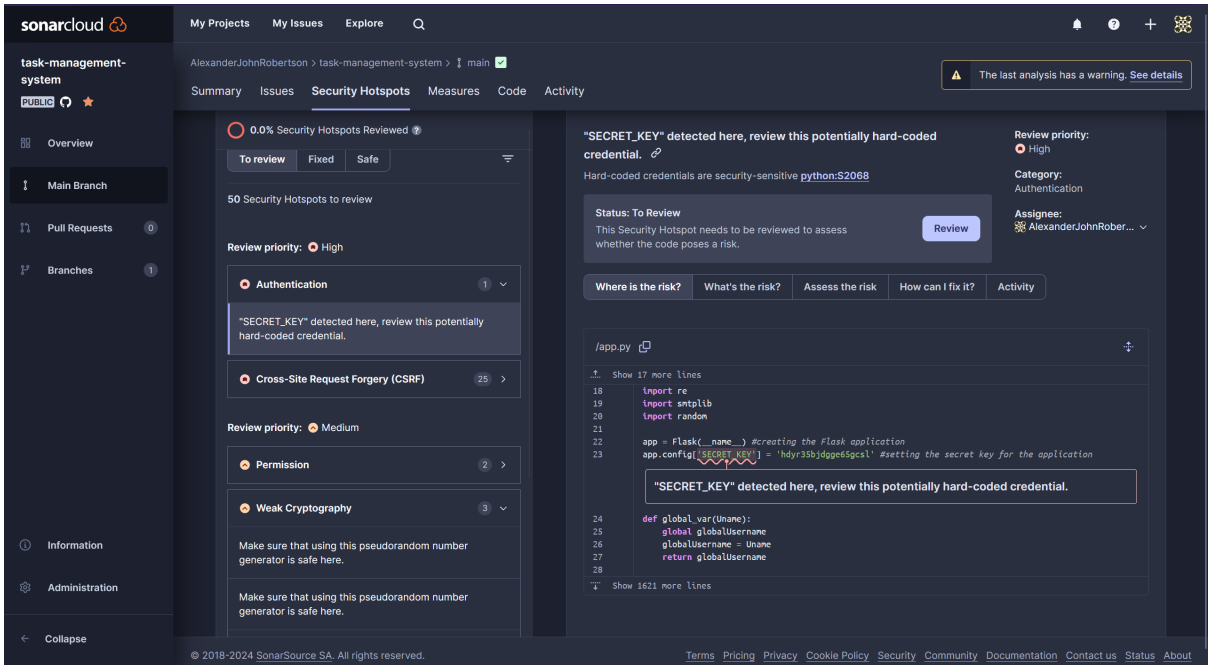


Figure 110 - SonarCloud has detected a security vulnerability as the secret key for flash messages is hardcoded into the application.

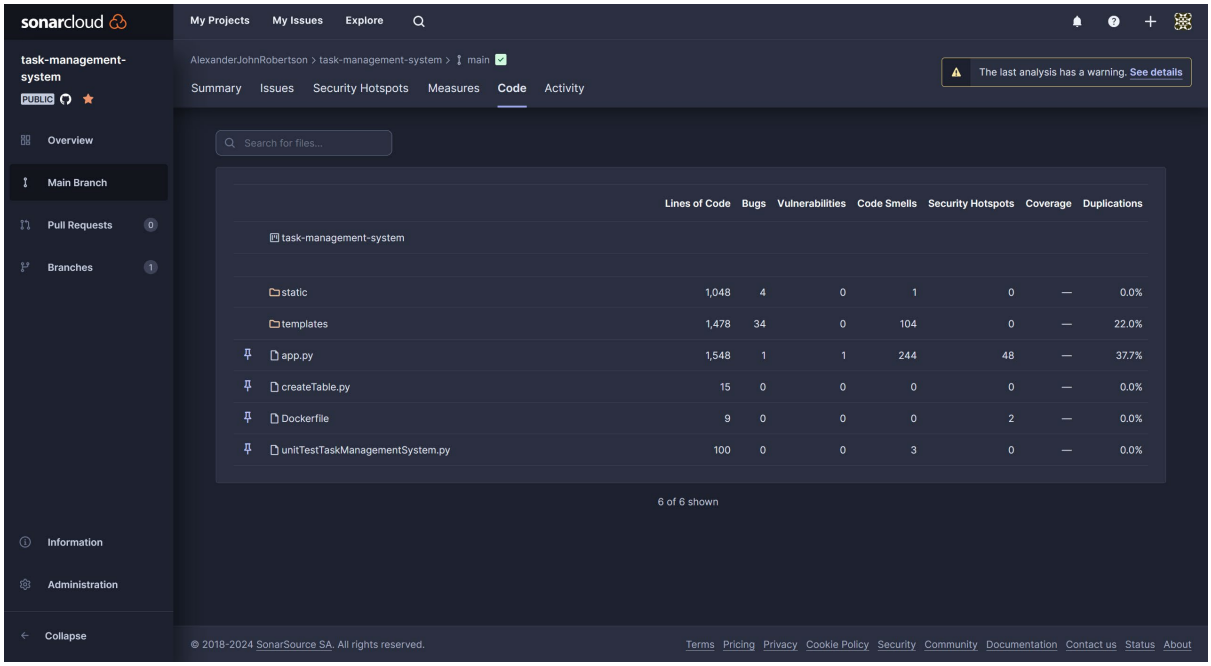


Figure 111 - SonarCloud showing the files and folders where the code issues are.

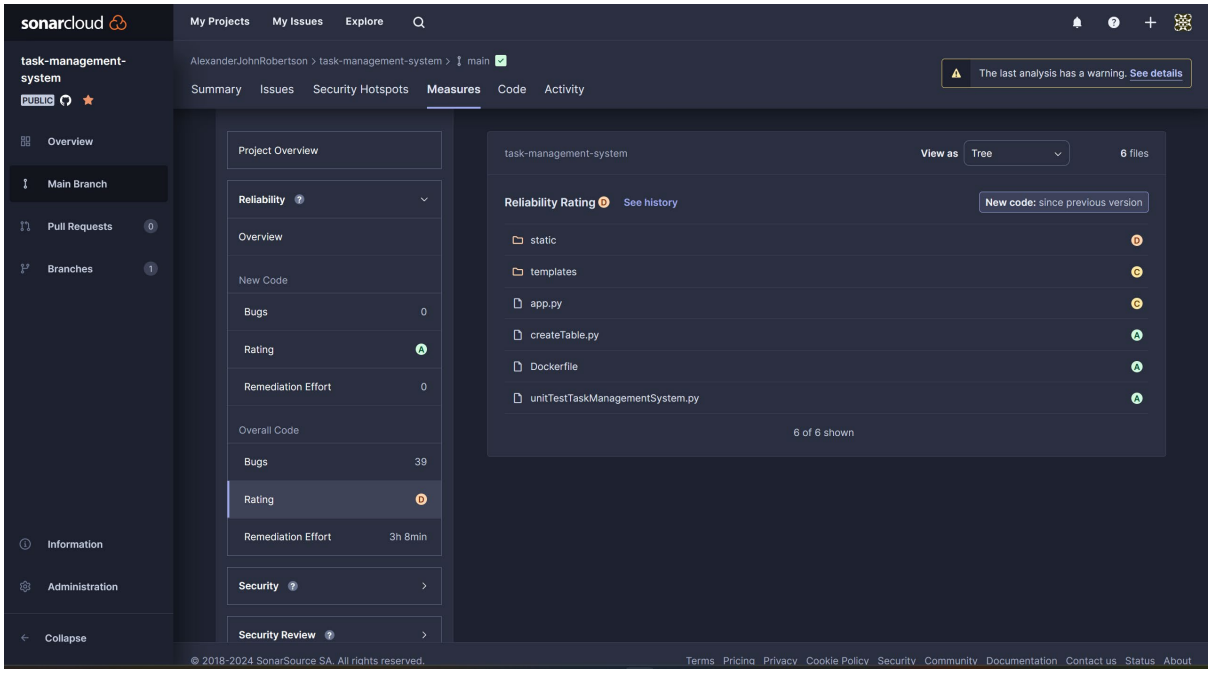


Figure 112 - SonarCloud grading he quality of my code.

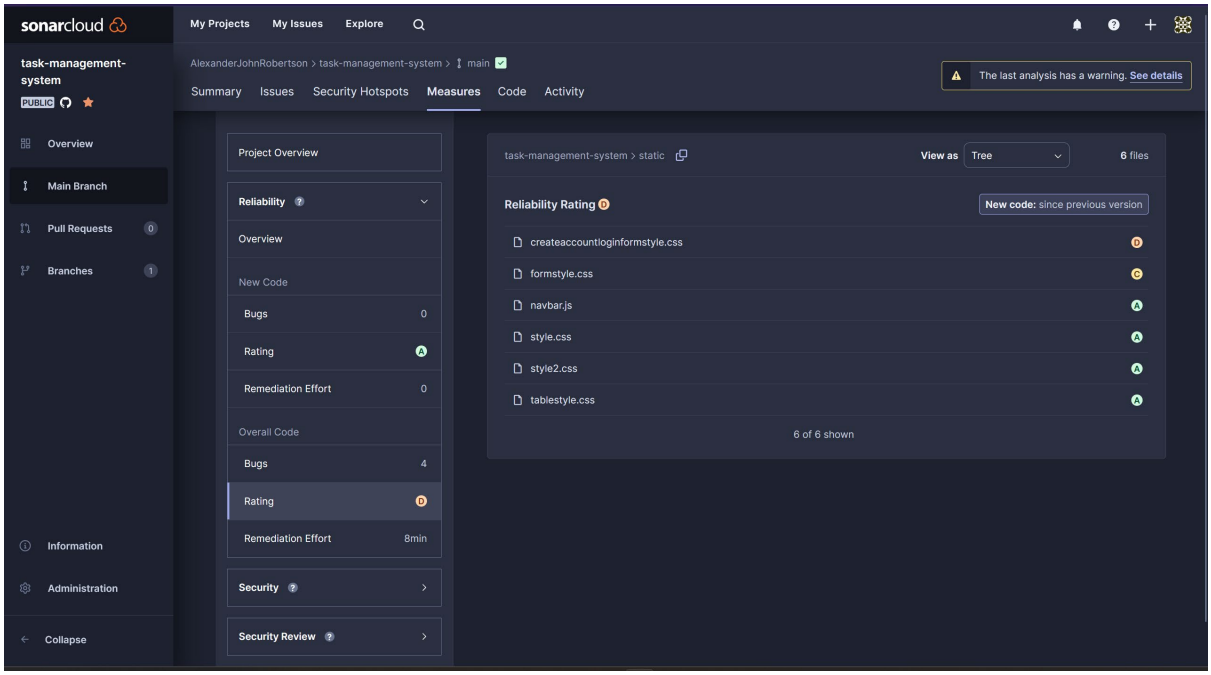


Figure 113 - SonarCloud has detected that the CSS files for forms have the worst code quality, the most smells and the most likely to be spaghetti code.

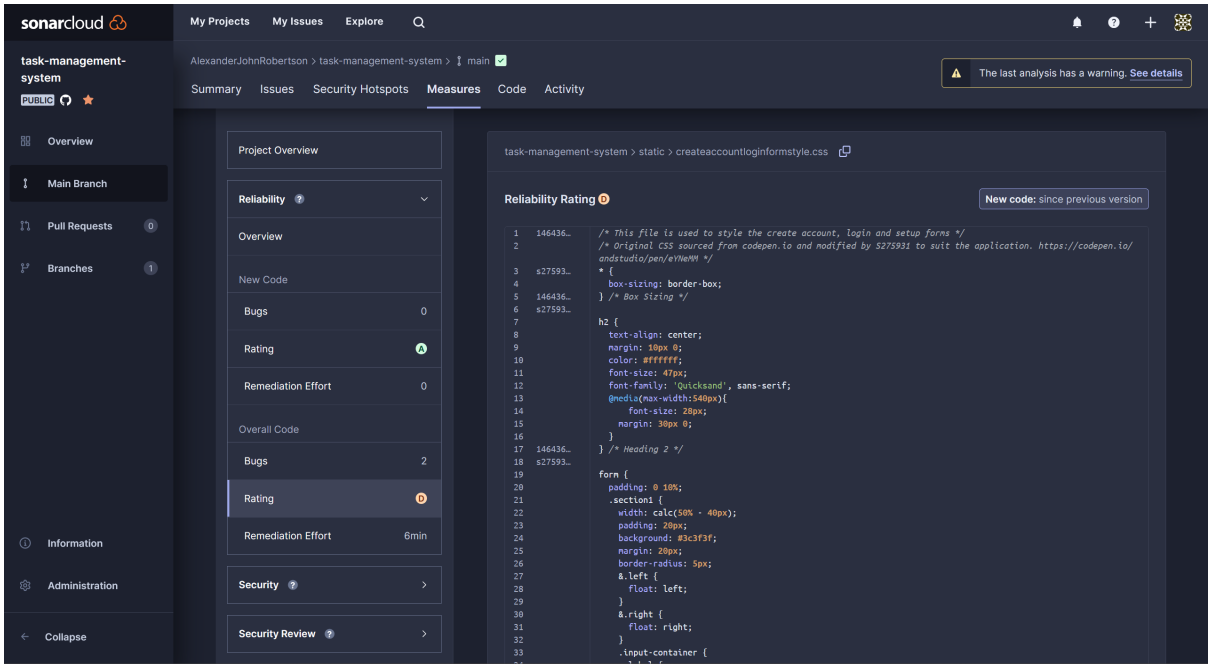


Figure 114 - The CSS for the create account and login forms is bad for reliability.

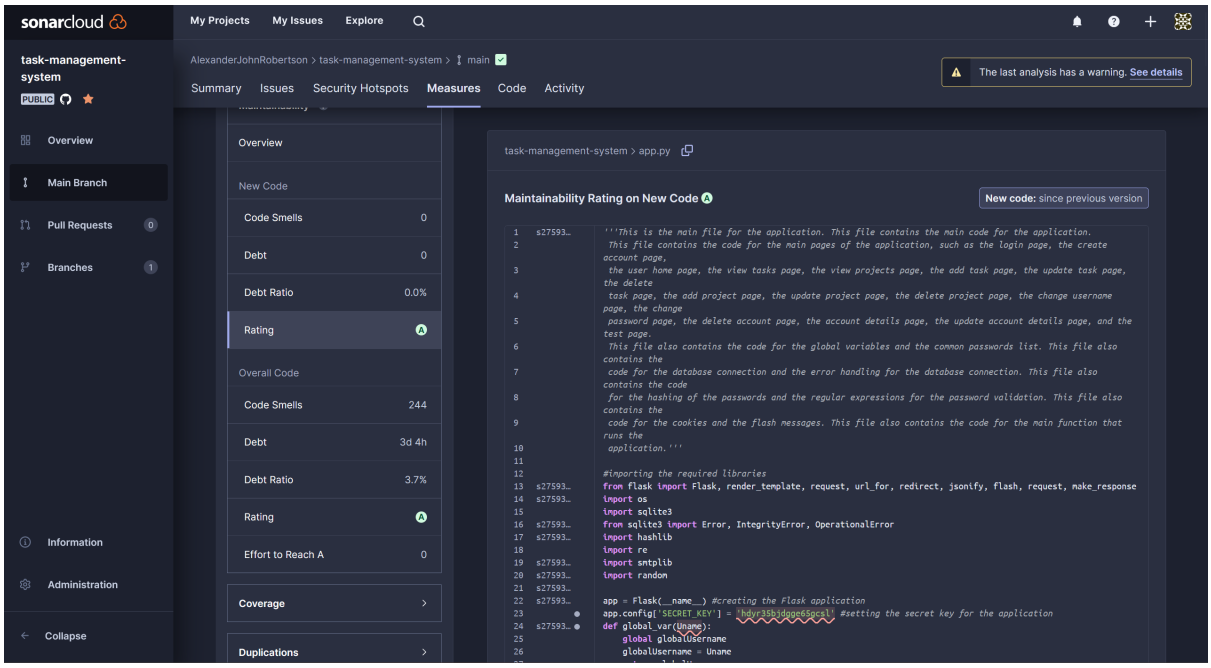


Figure 115 - The app.py Python code has good reliability.

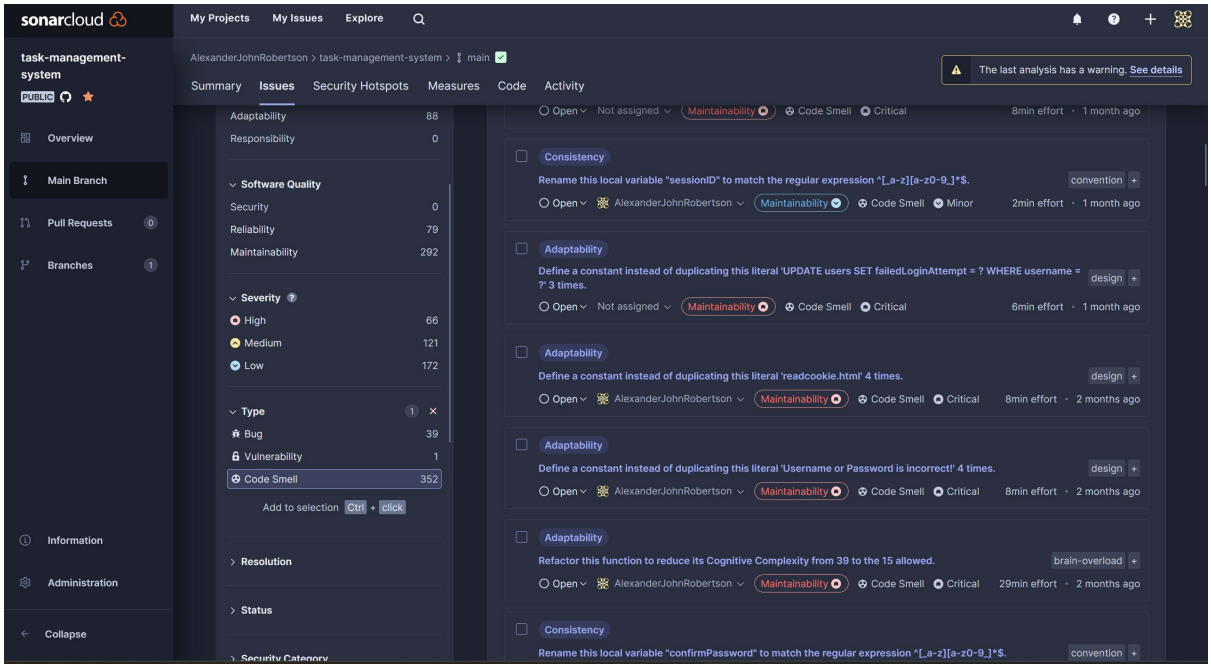


Figure 116 - SonarCloud has detected several code smells in the Task Management System.

Acceptance Testing

I have created a user acceptance testing form with a series of test cases for other users to test the Task Management System and give feedback of their user experience.

The user acceptance testing is in separate documents.

Task Management System – User Acceptance Testing

This is a form to fill in User Acceptance Testing of the Task Management System. Please fill in the sections of each table highlighted in yellow. Please refer to the User Manual section of this document for full instructions on how to use the Task Management System.

Setup

Label	Value
Name	Setup
Description	Perform First Time Setup and Create Root User.
Reporter (Your Name)	
Completion Date	
Test Case Steps	1. Go to https://task-management-system-clean-install.onrender.com/setup 2. Fill in Setup form to set up the Task Management System
Expected Result	Successfully set up the Task Management System by filling in the Setup form and clicking “Create Root User” button to submit the form. The user is redirected to their User Home page (in this case it will be “Root” and have access to all Task Management System features accessible through the navbar or hamburger menu. An automated email will be sent to the user (email address submitted in the Setup form) containing the password that was entered in the Setup form.
Actual Outcome (Your answer here)	
Status (Pass or Fail)	
URL	https://task-management-system-clean-install.onrender.com/setup
Please put screenshots here:	
Screen Size	
Operating system on your device	
Browser(s) on your device used to access the Task Management System.	
Notes	

Figure 117 - User Acceptance Testing Form - Incomplete

Task Management System – User Acceptance Testing

This is a form to fill in User Acceptance Testing of the Task Management System. Please fill in the sections of each table highlighted in yellow. Please refer to the User Manual section of this document for full instructions on how to use the Task Management System.

Setup

Label	Value
Name	Setup
Description	Perform First Time Setup and Create Root User.
Reporter (Your Name)	S275931
Completion Date	04/04/2024
Test Case Steps	1. Go to https://task-management-system-clean-install.onrender.com/setup 2. Fill in Setup form to set up the Task Management System
Expected Result	Successfully set up the Task Management System by filling in the Setup form and clicking "Create Root User" button to submit the form. The user is redirected to their User Home page (in this case it will be "Root" and have access to all Task Management System features accessible through the navbar or hamburger menu. An automated email will be sent to the user (email address submitted in the Setup form) containing the password that was entered in the Setup form.
Actual Outcome (Your answer here)	Accessed Setup page through the link. Filled in and submitted form. Setup was successful. Able to load the User Home page and be logged in as the root user. Password email was received.
Status (Pass or Fail)	Pass
URL	https://task-management-system-clean-install.onrender.com/setup

Figure 118 - Completed User Acceptance Testing form where the test case has passed.

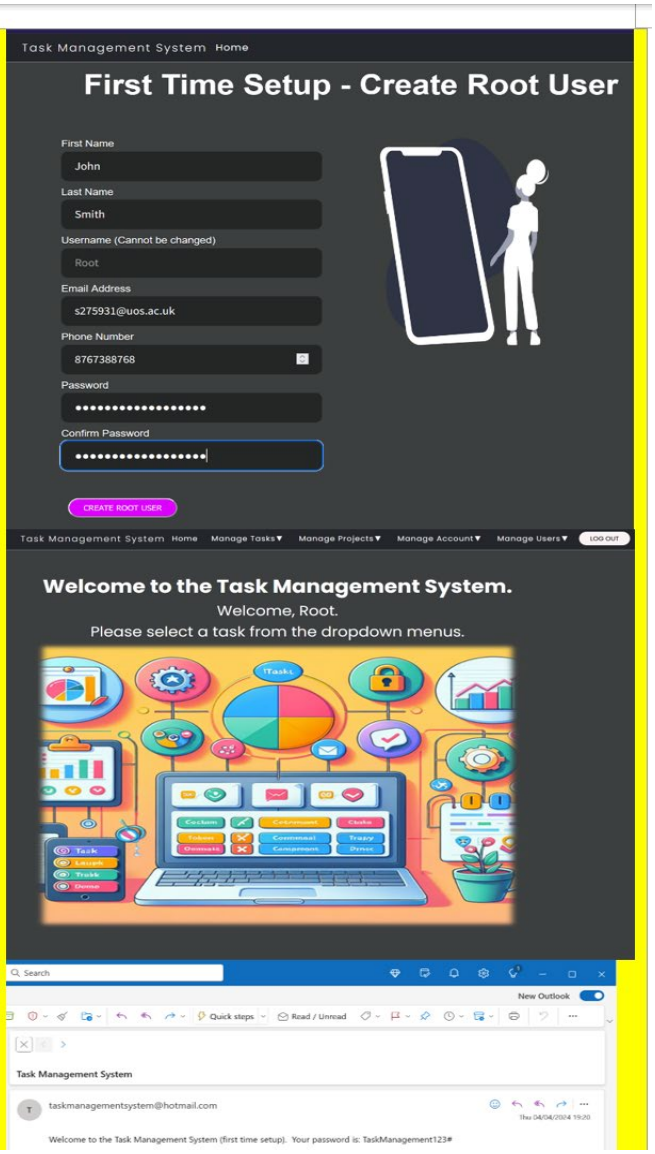
Please put screenshots here:	

Figure 119 - Completed User Acceptance Testing form where the test case has passed.

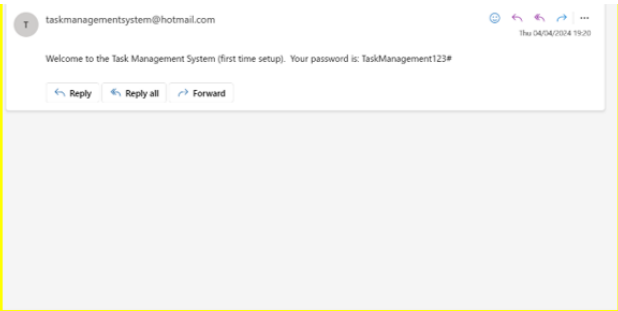
	
Screen Size	2560 x 1600, 17 Inch Laptop
Operating system on your device	Windows 11 Pro
Browser(s) on your device used to access the Task Management System.	Mozilla Firefox
Notes	

Figure 120 - Completed User Acceptance Testing form where the test case has passed.

Create Account, Log In and Log Out

Login details for pre-set-up Task Management System

User Role	Username	Password
Root User	Root	TaskManagement123#
Administrator	barneyPurpleDinosaur	BabyBop123#
Standard	Birmingham888	SouthSide123@

Label	Value
Name	Log In
Description	Log in to Task Management System.
Reporter (Your Name)	Heyser
Completion Date	09/04/2024
Test Case Steps	1. Go to https://task-management-system-clean-install.onrender.com/login or https://task-management-system-hnmb.onrender.com/login 2. Use the log in form to log in either with the details in the previous table or with an account you have created or the Root User account you created during setup.
Expected Result	After logging in, the user is redirected to the User Home page displaying text saying, "Welcome to the Task Management System.", "Welcome, <your-username> and a task management cartoon featuring a laptop. There is a navigation bar or hamburger menu (depending on your device) that gives you access to all the Task Management System's features.
Actual Outcome (Your answer here)	The pre-set login details didn't work
Status (Pass or Fail)	Fail
URL	https://task-management-system-clean-install.onrender.com/login or https://task-management-system-hnmb.onrender.com/login

Figure 121 - Completed User Acceptance Testing form where the test case has failed as they were unable to log in using the login details provided. This means that there is room for improvement.

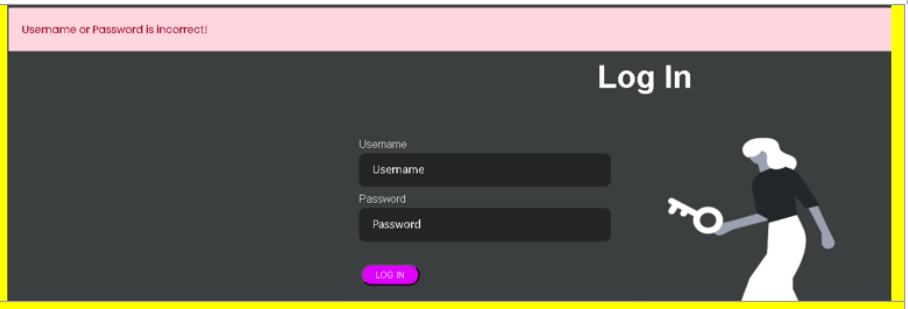
Please put screenshots here:	
Screen Size	21.5”
Operating system on your device	Windows 10
Browser(s) on your device used to access the Task Management System.	Google Chrome
Notes	

Figure 122 - Completed User Acceptance Testing form where the test case has failed as they were unable to log in using the login details provided. This means that there is room for improvement.

Deployment

I have deployed the Task Management System web application on two platforms:

- Render.com – I have deployed the Task Management System online using the webhosting service render.com as a Python web service. The deployments are available here: https://task-management-system-hnmb.onrender.com/ and https://task-management-system-clean-install.onrender.com/setup
- The Render.com deployment uses a CI/CD pipeline that automatically builds and deploys the application from the codebase on the GitHub repository.
-

Advantages of Render.com	Disadvantages of Render.com
Very easy to set up and use	Free instances spin down due to inactivity – must pay for instance to be constantly available.
Automatic CI/CD pipelines from Git repositories – even on free instances.	Internet connection required (Acosta, 2023)
Easy deployment and building	
Supports Python and Flask web apps	
No development skills required	
Smaller learning curve compared to Docker	
Application available on any device with a supported web browser and an internet connection without the need for Docker or any other software to be installed. (Acosta, 2023)	

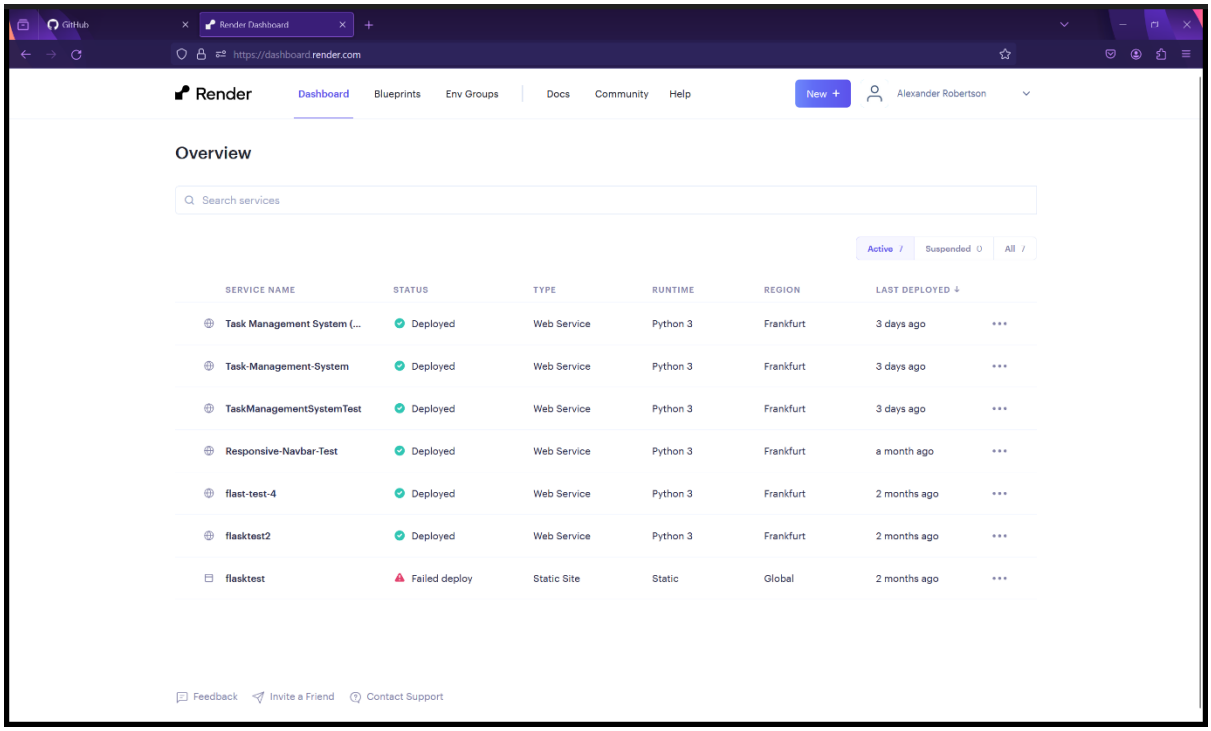


Figure 123 - Render.com deployments

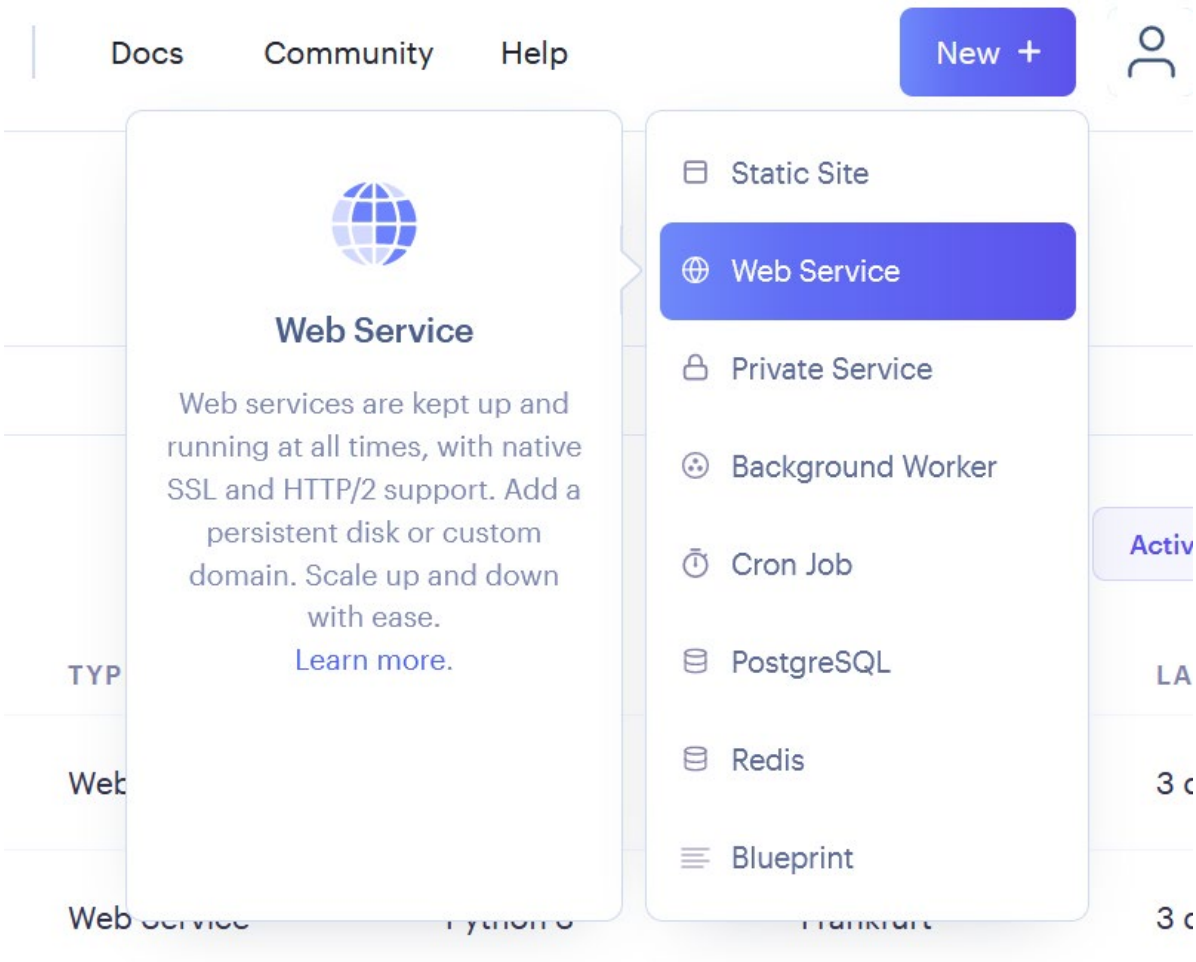


Figure 124 - Creating a web service on Render.com for deployment.

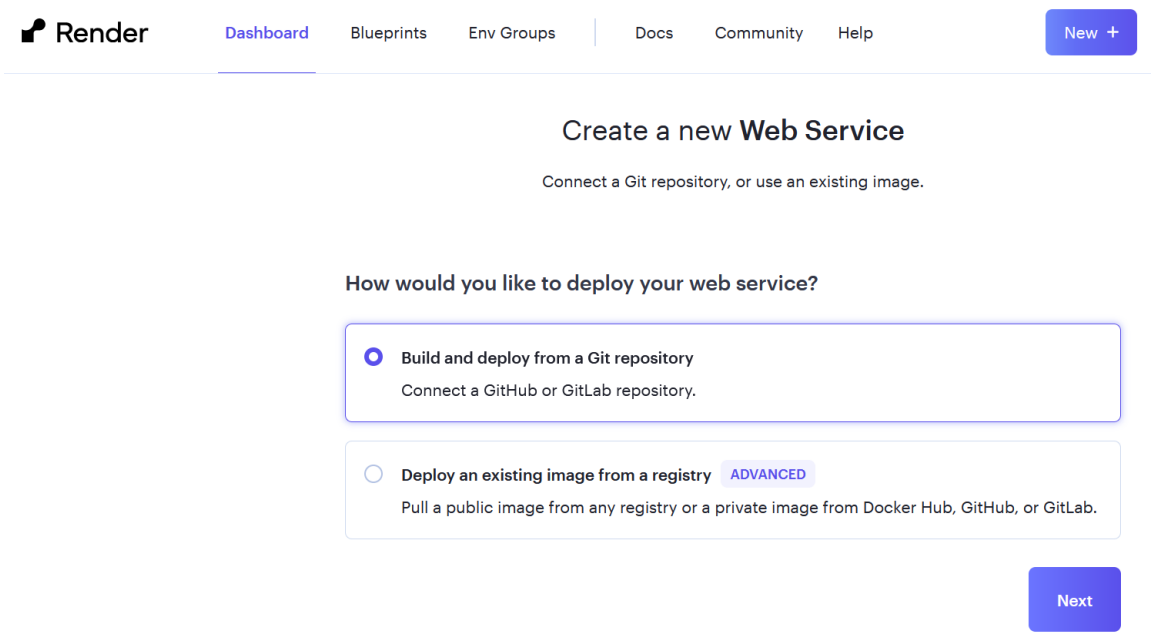


Figure 125 - Connecting to the Task Management System's Git repository to create a CI/CD pipeline.

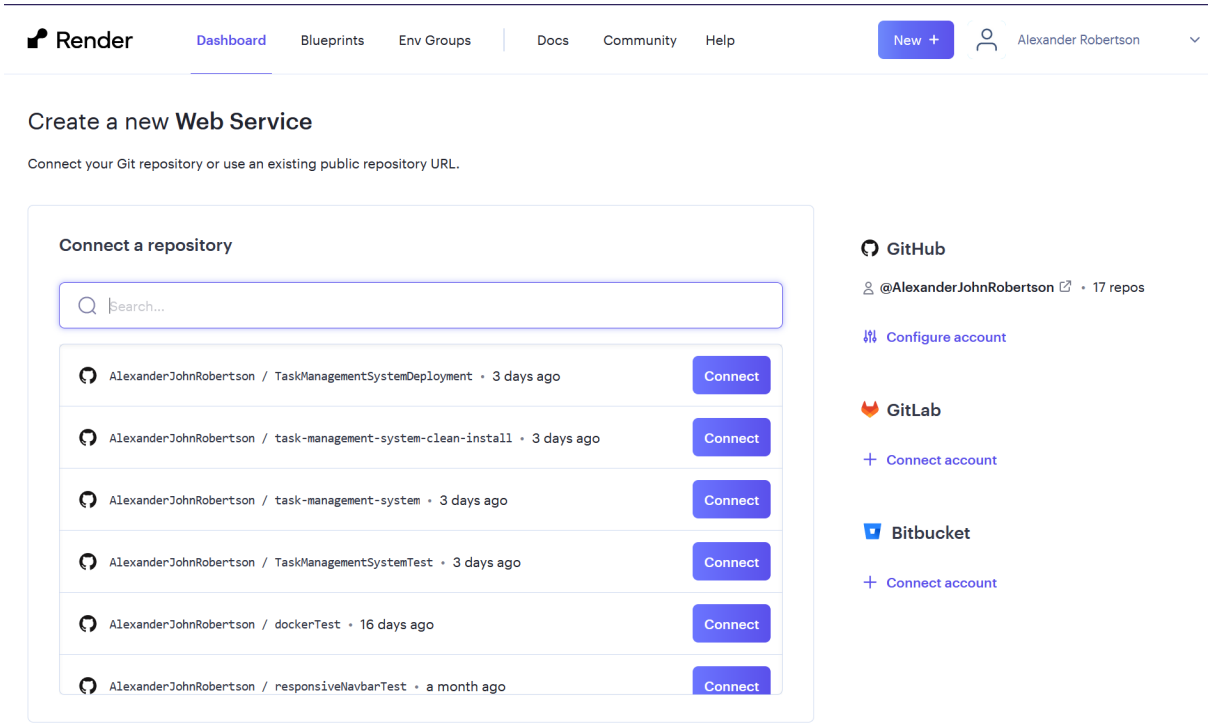


Figure 126 - Connecting to the Task Management System's Git repository to create a CI/CD pipeline.

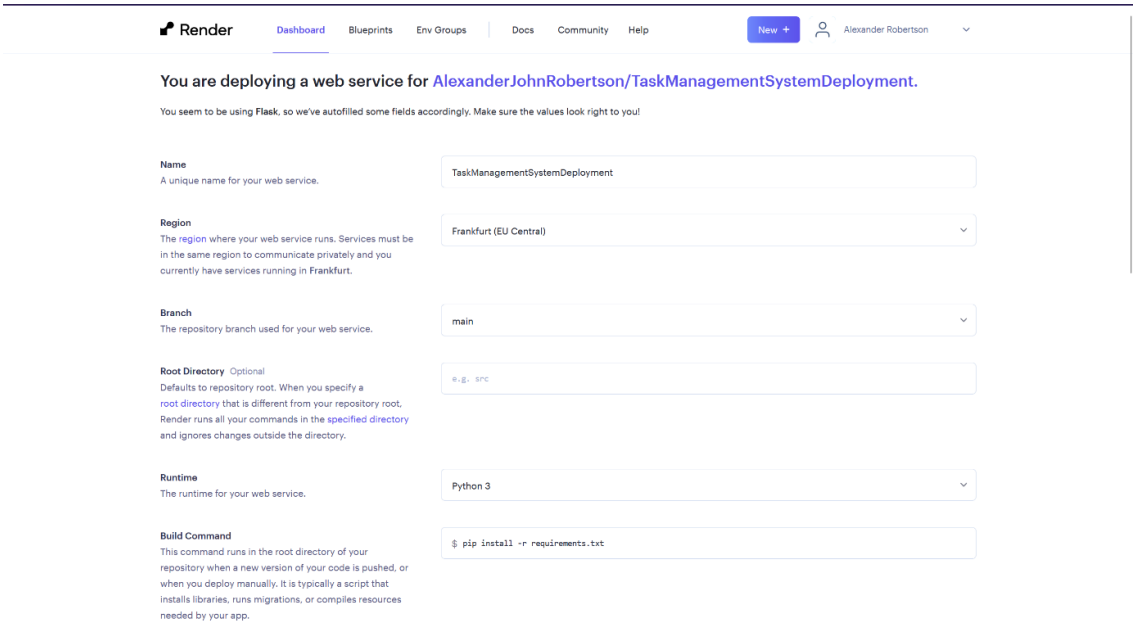


Figure 127 - Render.com deployment settings.

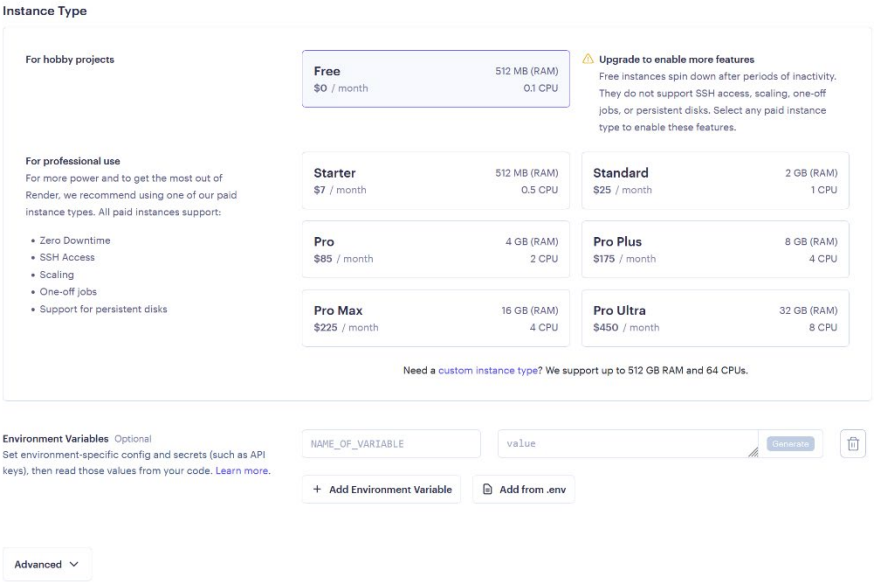


Figure 128 - Choosing a render.com subscription for the web service.

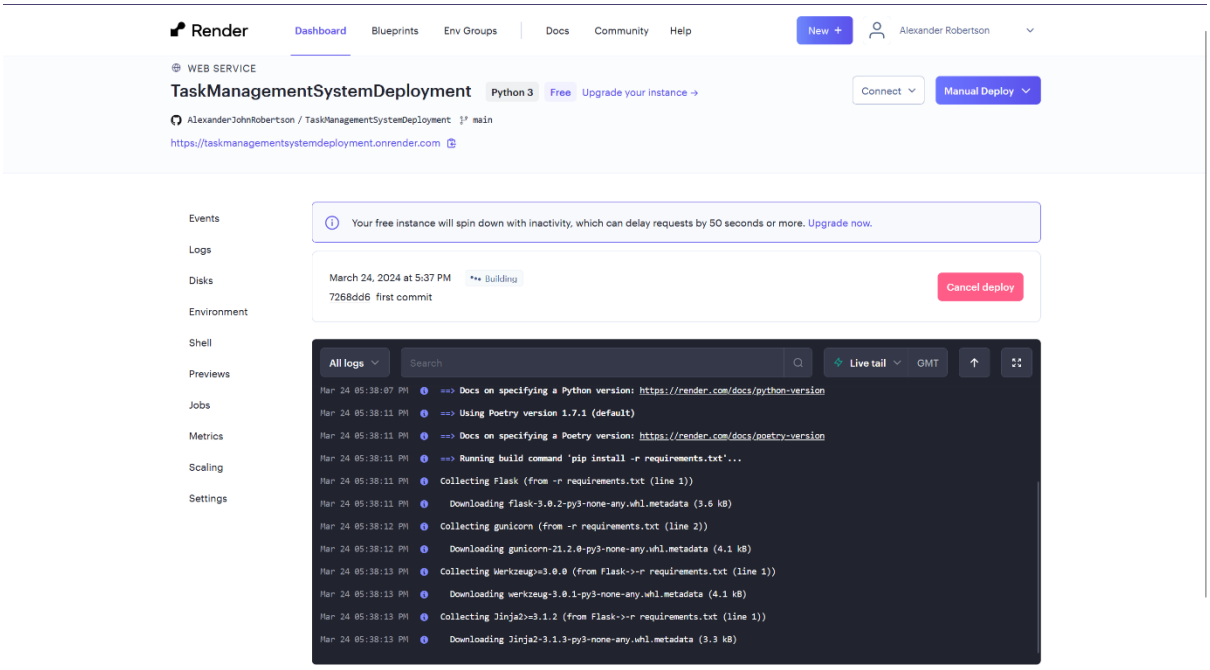


Figure 129 - Render.com is automatically building the Task Management System web service as a CI/CD pipeline.

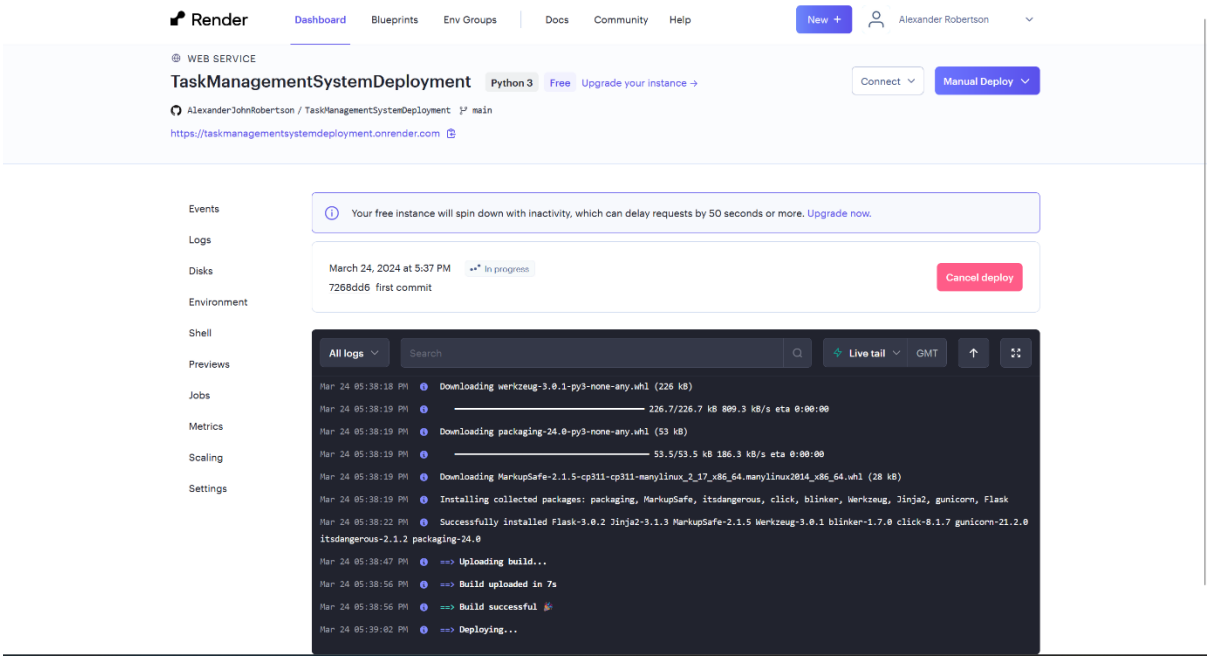


Figure 130 - Render.com is automatically deploying the Task Management System web service as a CI/CD pipeline.

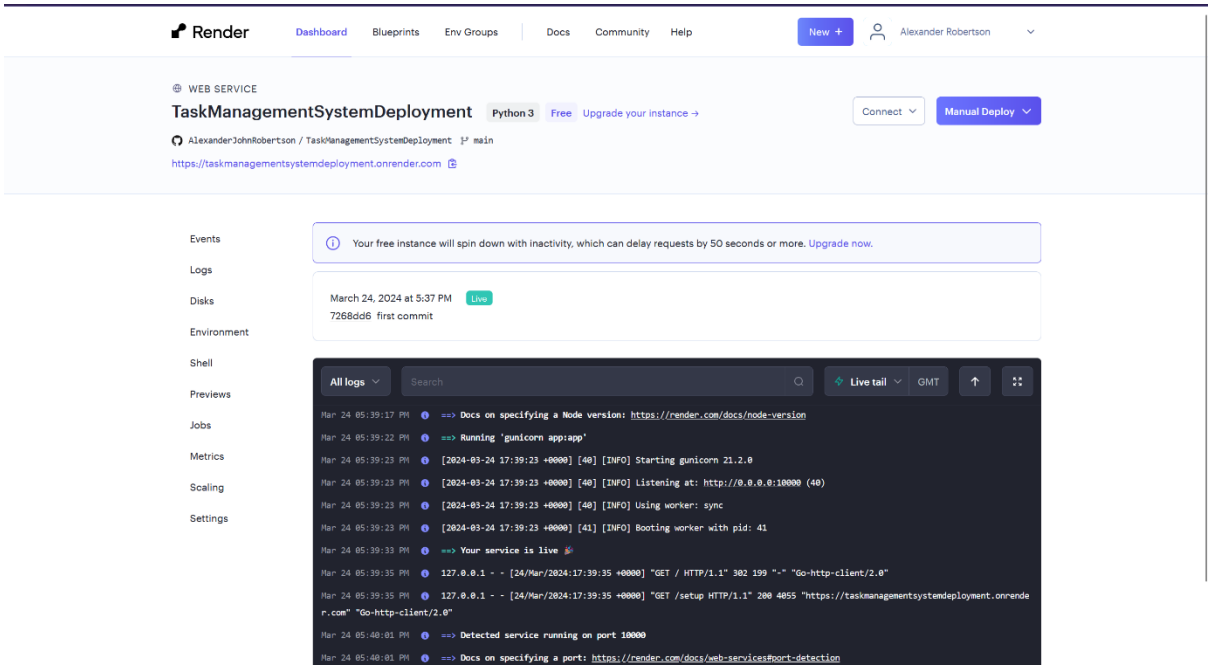


Figure 131 - The Task Management System web service is now live.

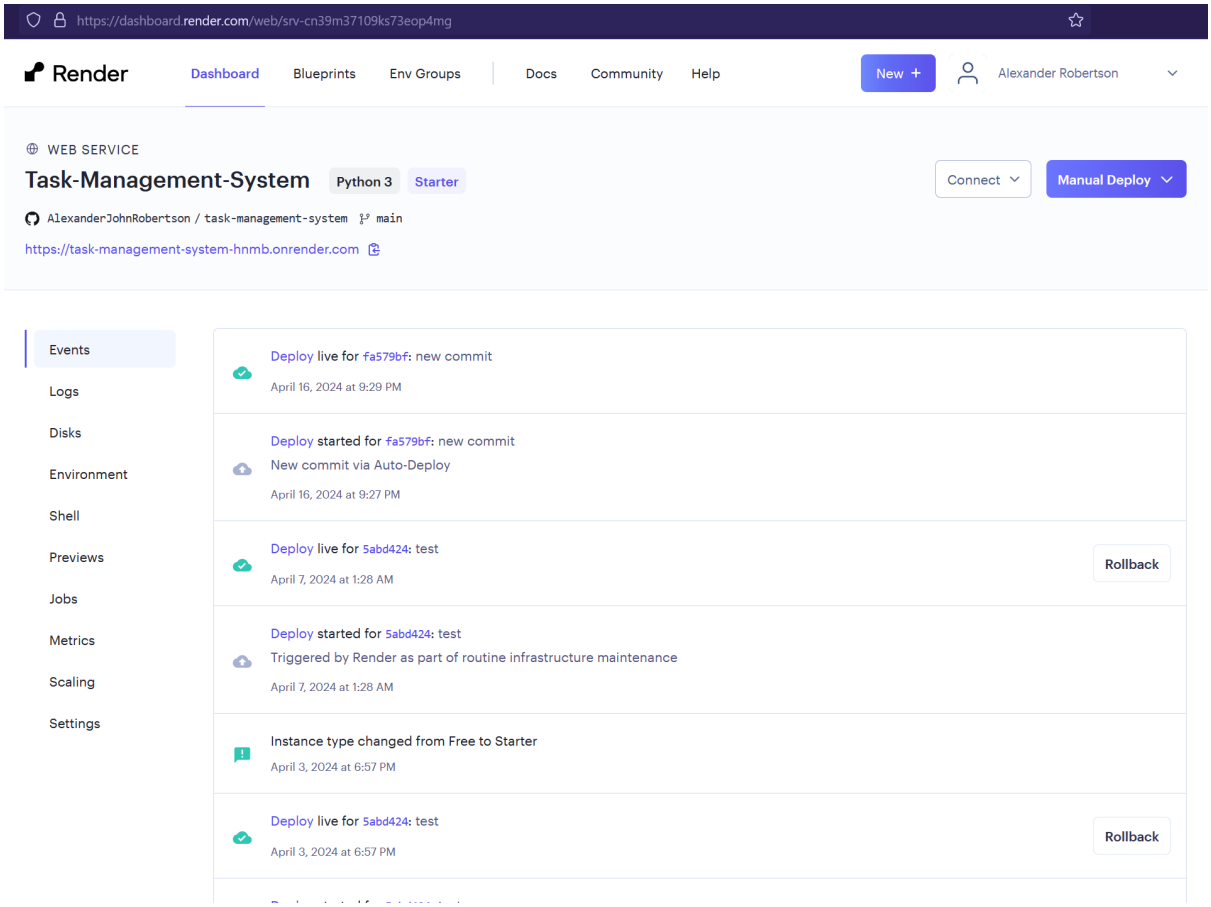


Figure 132 - Task Management System web service deployment history.

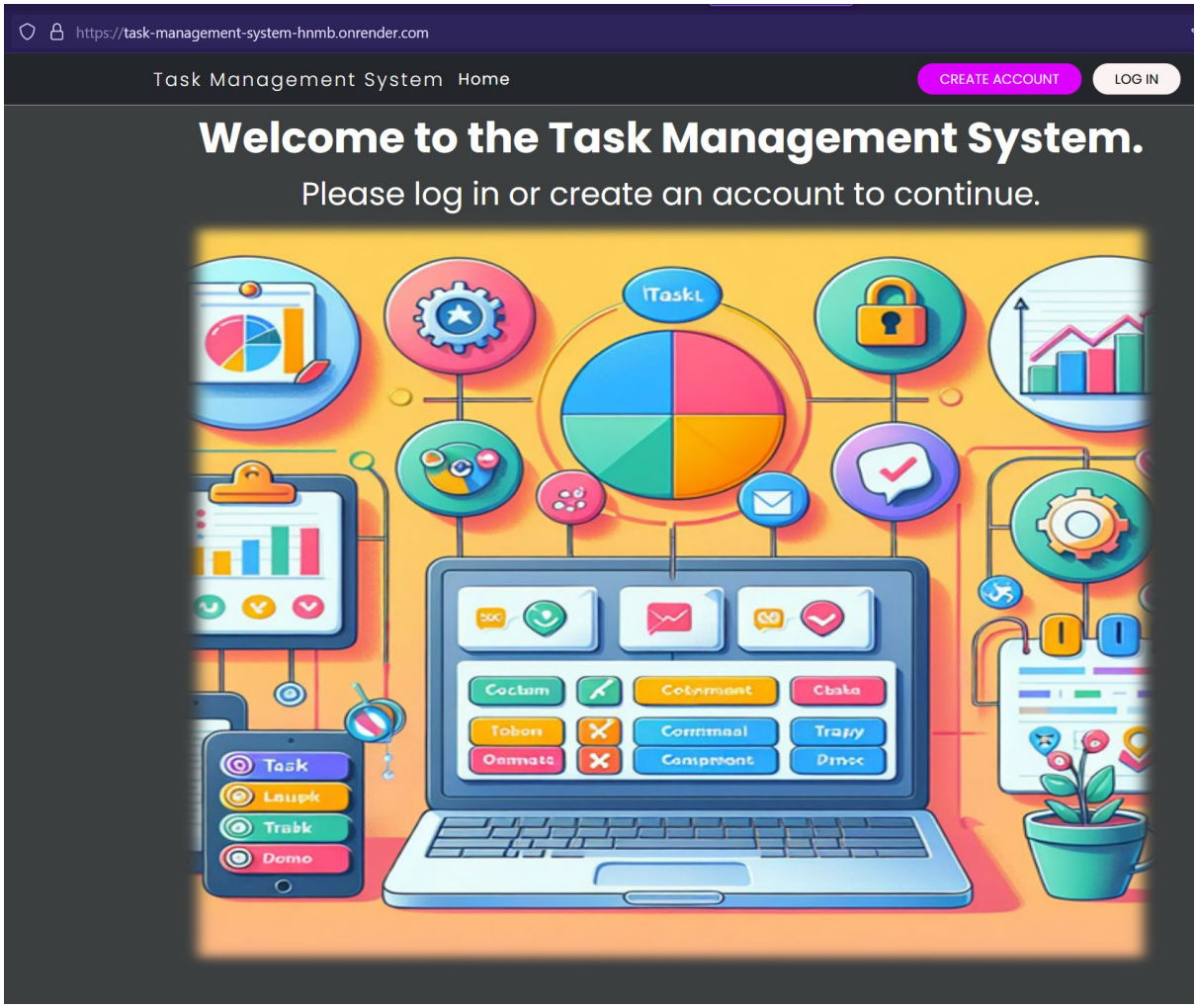


Figure 133 - Task Management System web service landing page.

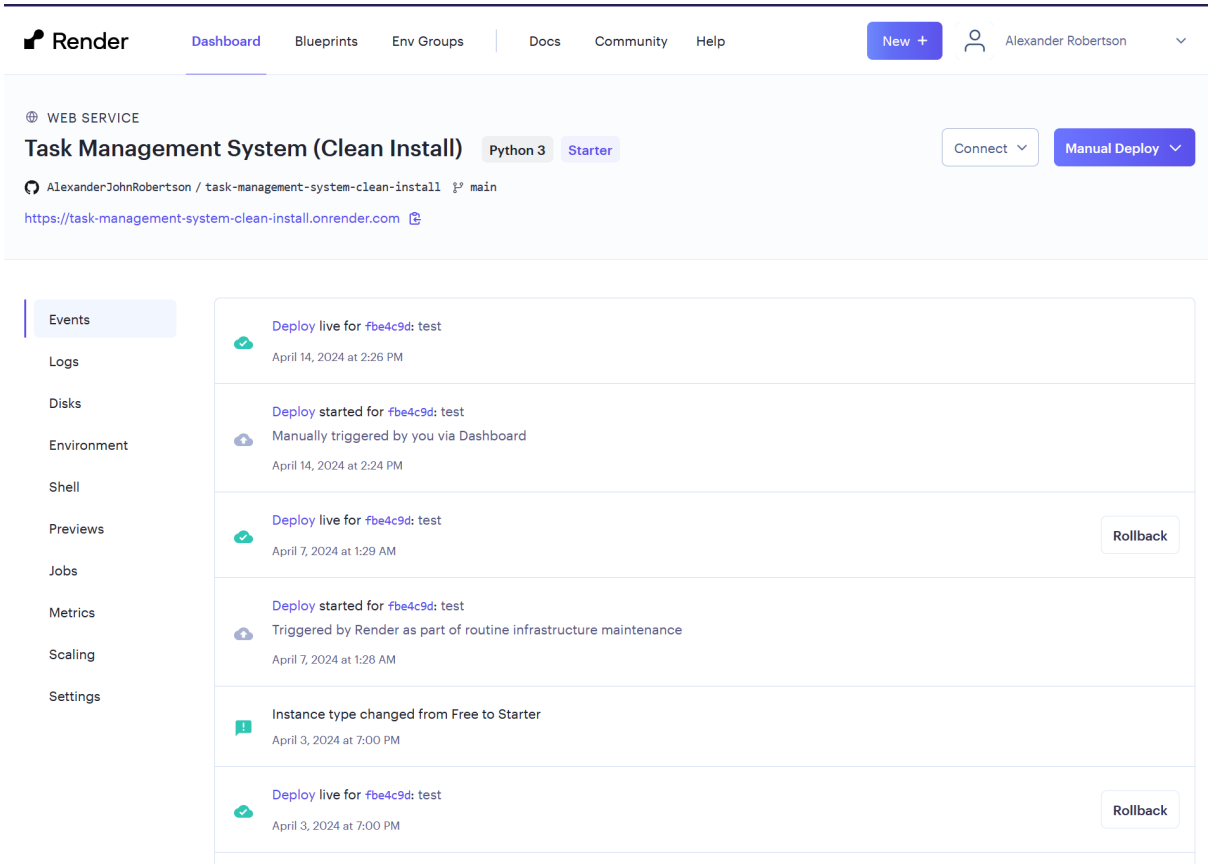


Figure 134 - Task Management System (Clean Install) web service deployment history

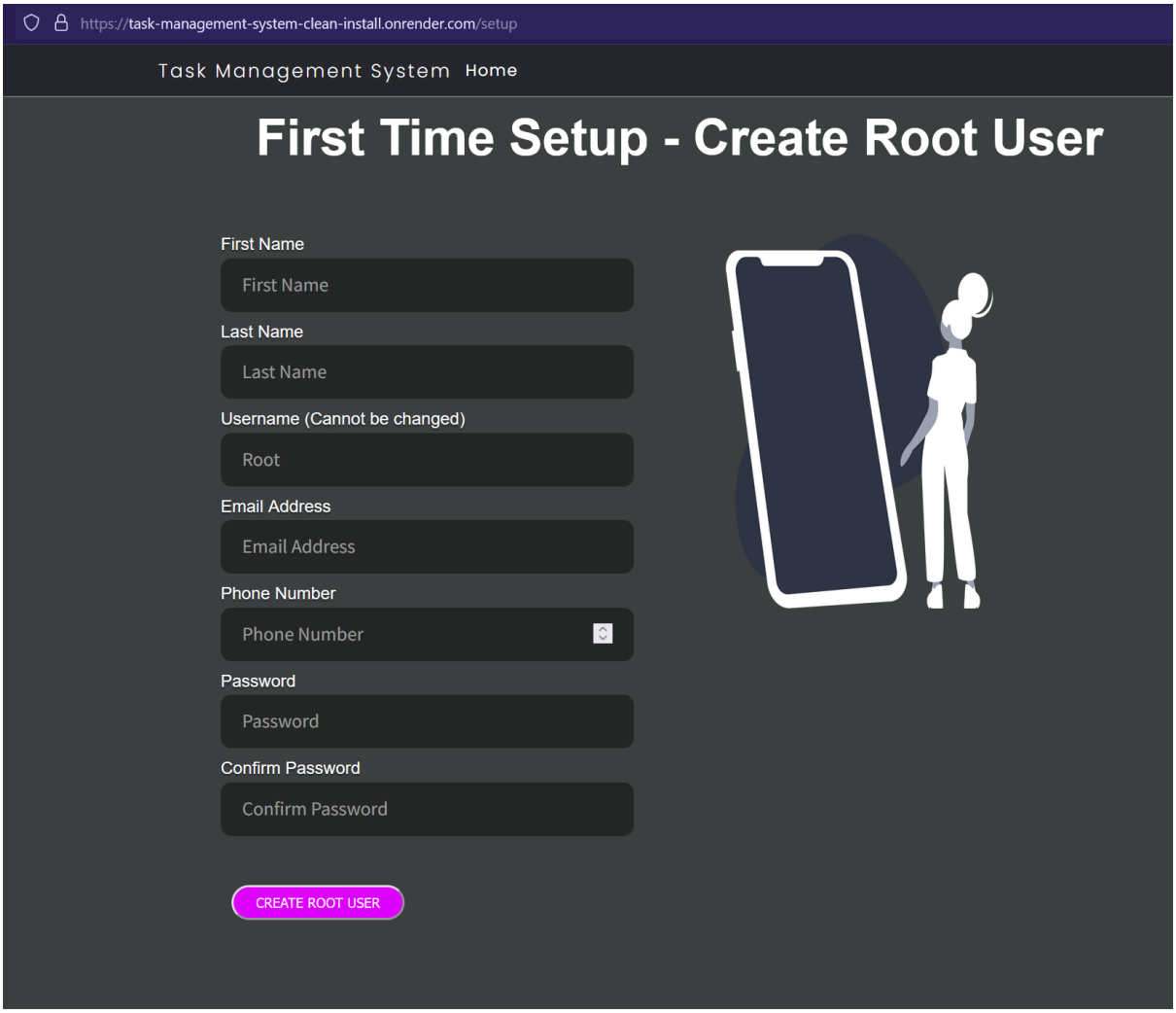


Figure 135 - Task Management System (Clean Install) setup page.

- Docker containerisation - I have deployed the Task Management System as a Docker image/container using Docker Desktop. The containerisation means that the Task Management System is cross-platform can be installed and run on any computer or server (Windows/Linux/macOS) with Docker installed. The container images has been uploaded to DockerHub and are available here: https://hub.docker.com/r/alexjrobertson/taskmanagementsystem and https://hub.docker.com/r/alexjrobertson/taskmanagementsystemcleaninstall and can be pulled (downloaded) to your computer using these commands: `docker pull alexjrobertson/taskmanagementsystem` and `docker pull alexjrobertson/taskmanagementsystemcleaninstall`
- Currently the Docker deployment is run locally on a computer or server and is accessible in a web browser through localhost. The Docker deployment has the potential to be run on cloud computing platforms such as AWS as a Docker container on an EC2 instance or on AWS Elastic Container Services.

Advantages of Docker	Disadvantages of Docker
Cross-Platform software deployment – Docker allows your application to run on Windows, macOS and Linux.	Steep learning curve – Docker is complex, and containerisation requires the developer to learn new concepts, skills, and tools.
Containerisation is more efficient and lightweight than virtual machines.	Difficult to debug – When I used Docker to deploy the Task Management System, it gave generic error messages instead of detailed error messages making it difficult for me to identify the problem and fix it.
Docker containers run consistently on multiple devices without the need to modify the Task Management System's source code.	Containers have overhead therefore used more resources and less efficient than natively run applications.
Faster deployment times	Outdated Documentation – not keeping up with updates. (DuploCloud, 2023) (AspiringYouths, 2024)
Isolation of applications – if one app encounters a software failure it doesn't affect other applications.	
Easy to build and deploy Python applications (my experience)	
Available on cloud computing platforms such as AWS. (DuploCloud, 2023) (AspiringYouths, 2024)	

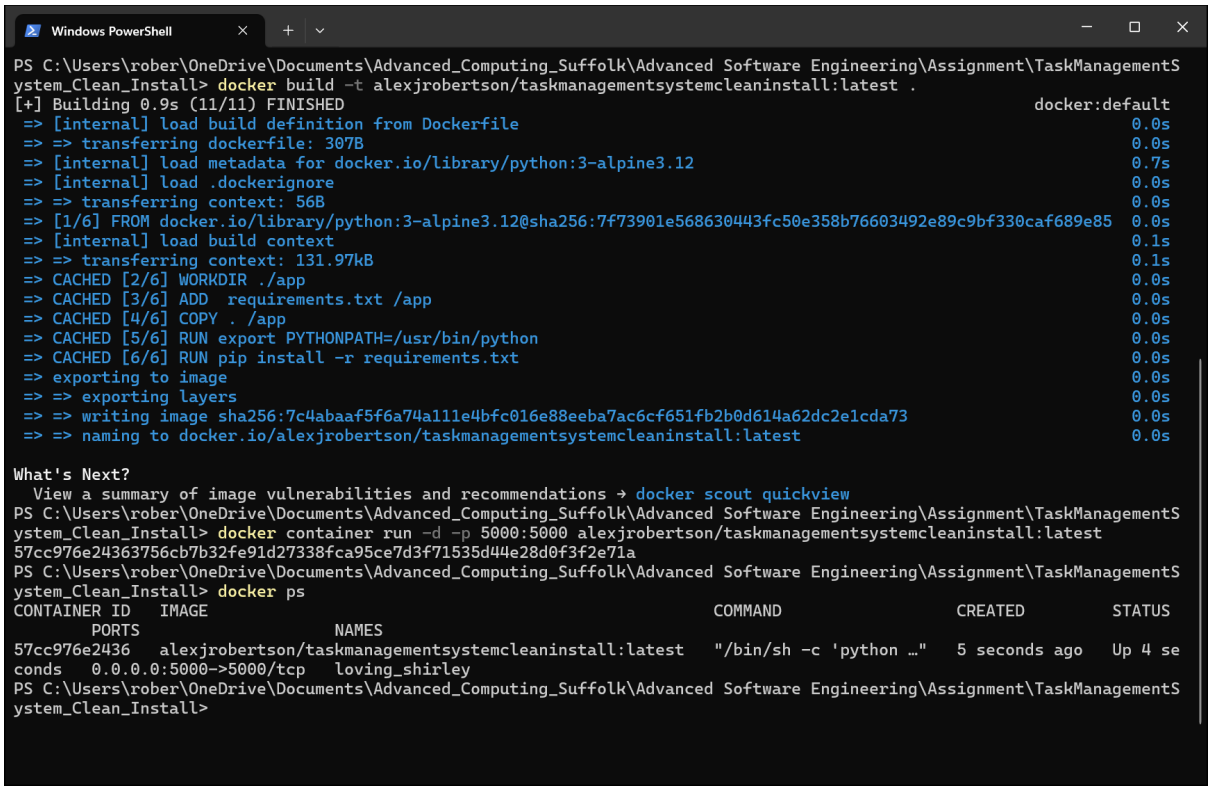


Figure 136 - Building the Docker container images for the Task management System using Docker commands in Windows PowerShell then running the containers.

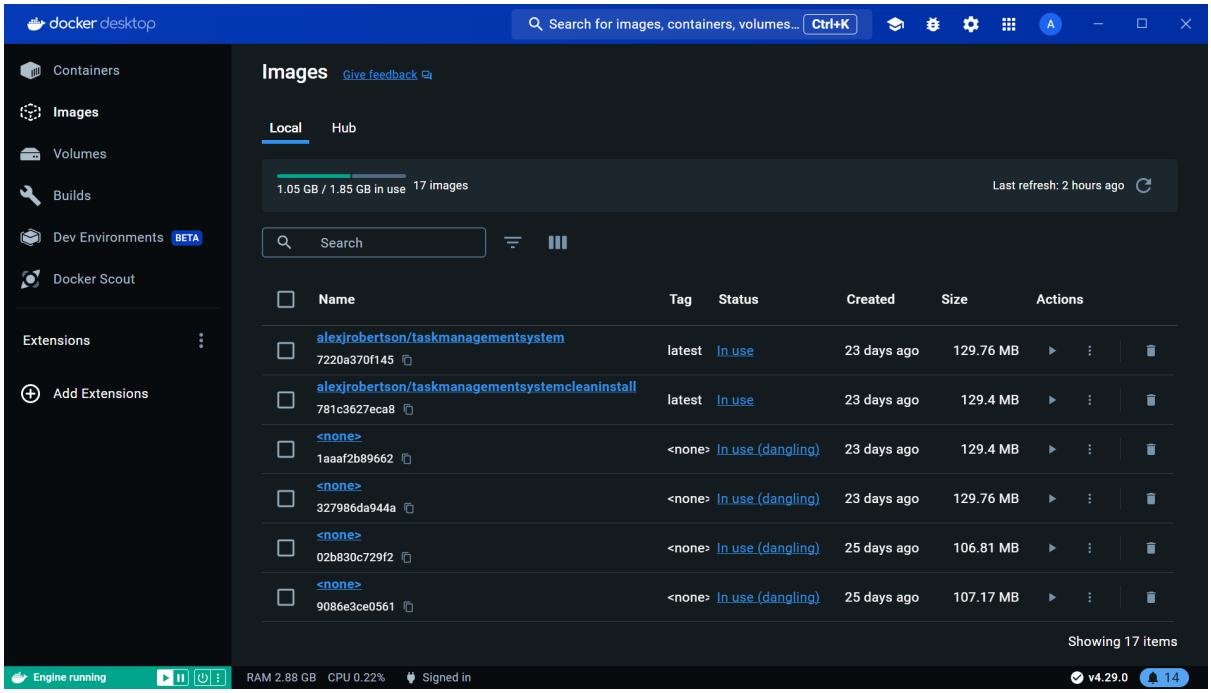


Figure 137 - Task Management System Docker Container images in Docker Desktop.

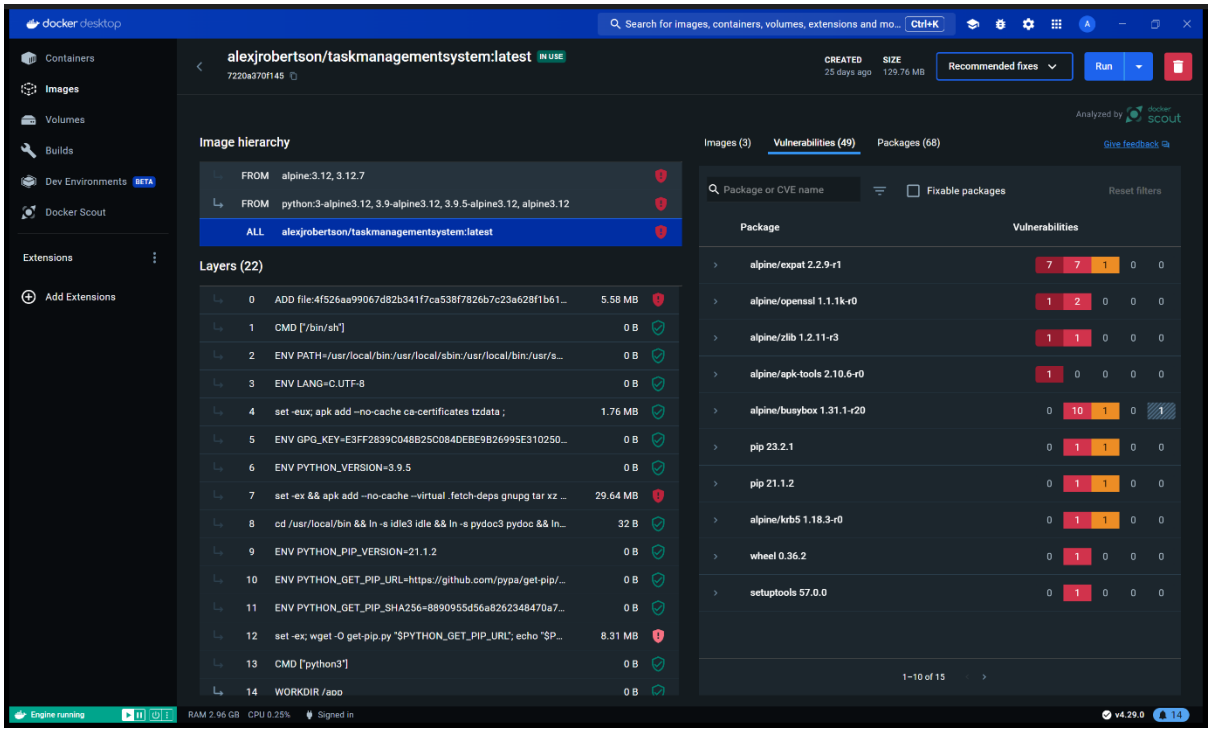


Figure 138 - Task Management System container in Docker Desktop. Docker has detected security vulnerabilities with Alpine Linux, which is required to run the Task Management System container.

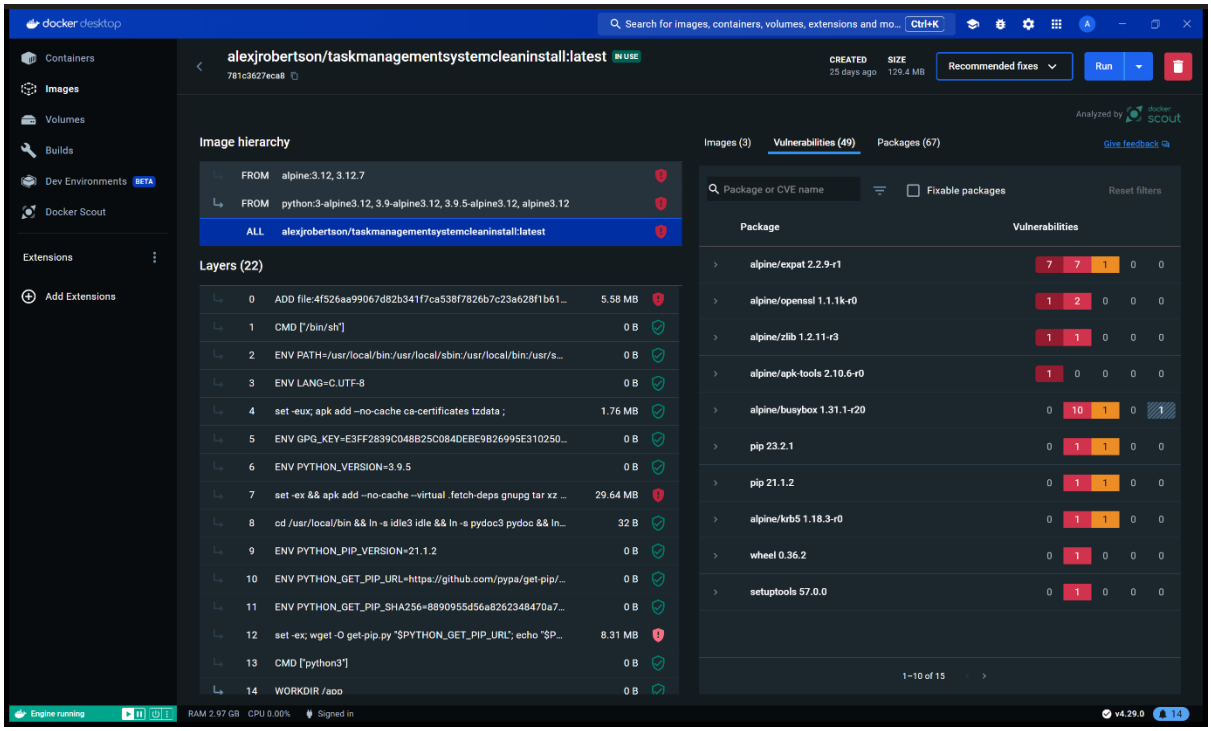


Figure 139 - Clean Install version of the Task Management System.

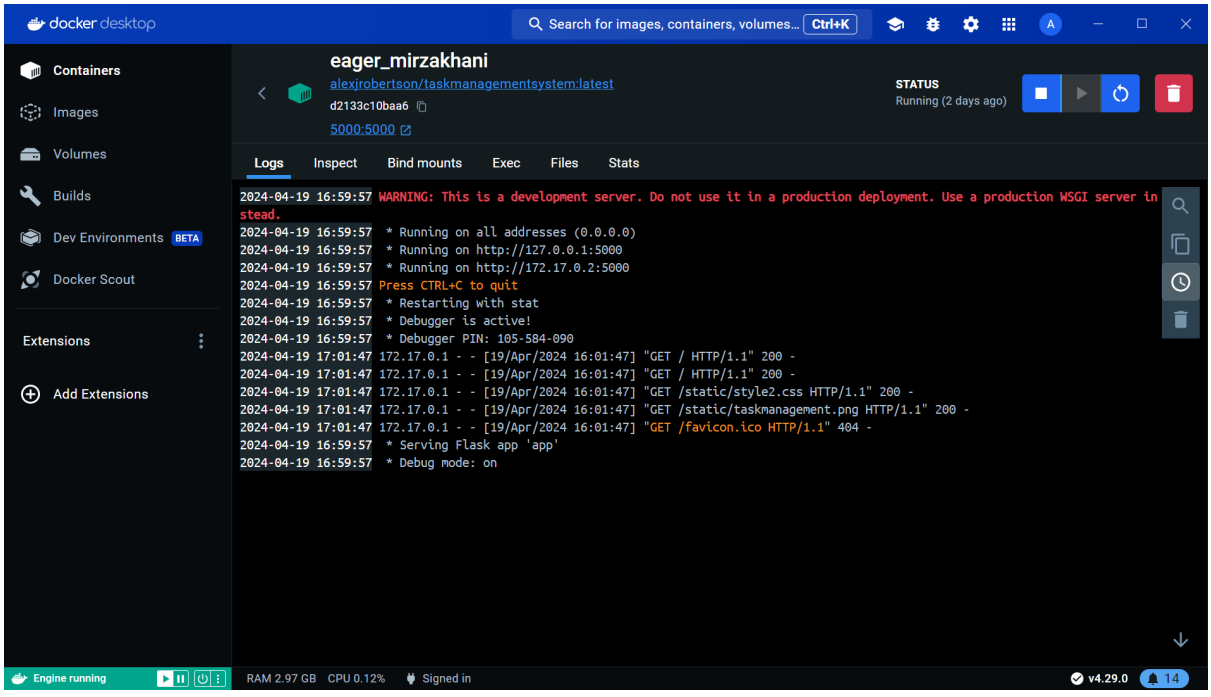


Figure 140 - Task Management System container running in Docker Desktop.

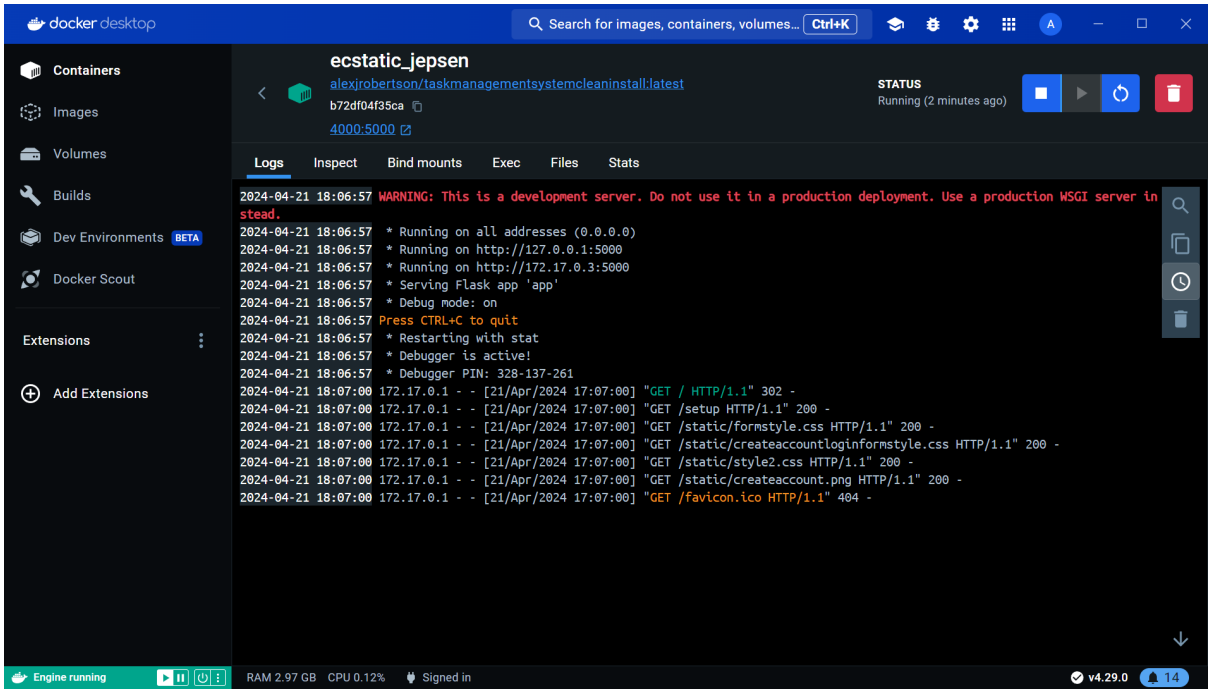


Figure 141 - Clean Install version of the Task Management System running in Docker Desktop.

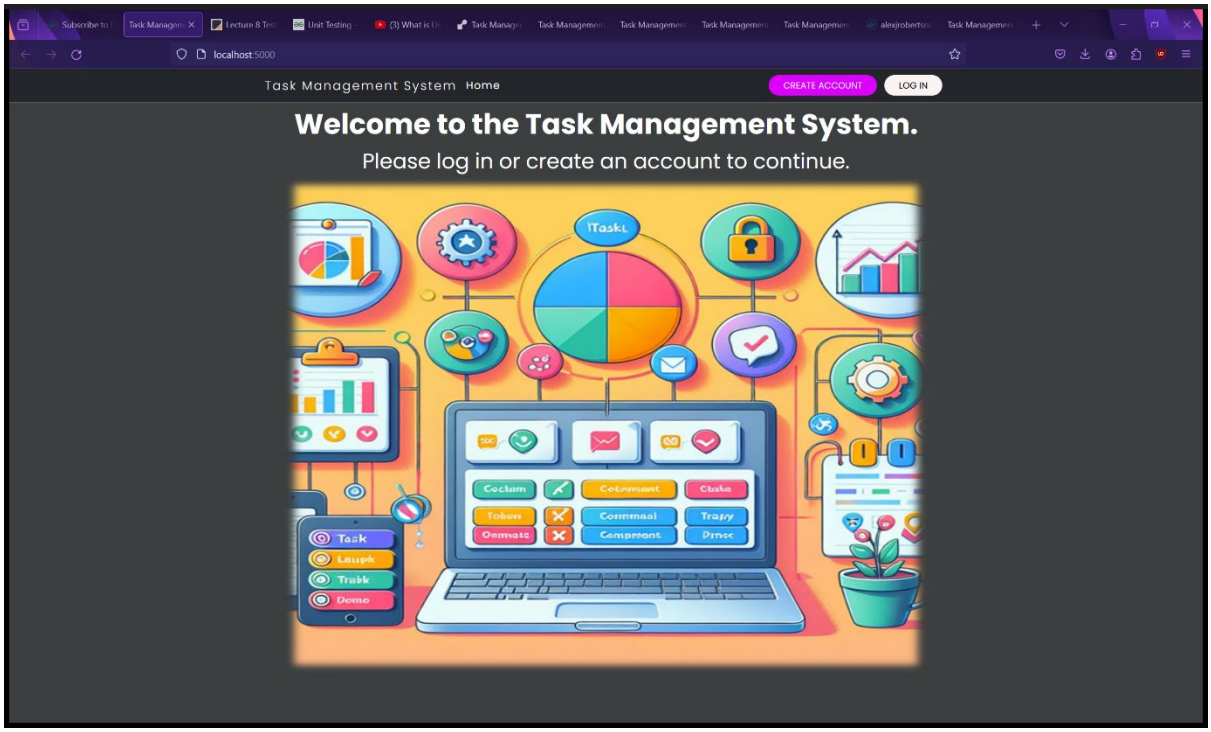


Figure 142 - Landing page of the Task Management System in a web browser, running in Docker as a container.

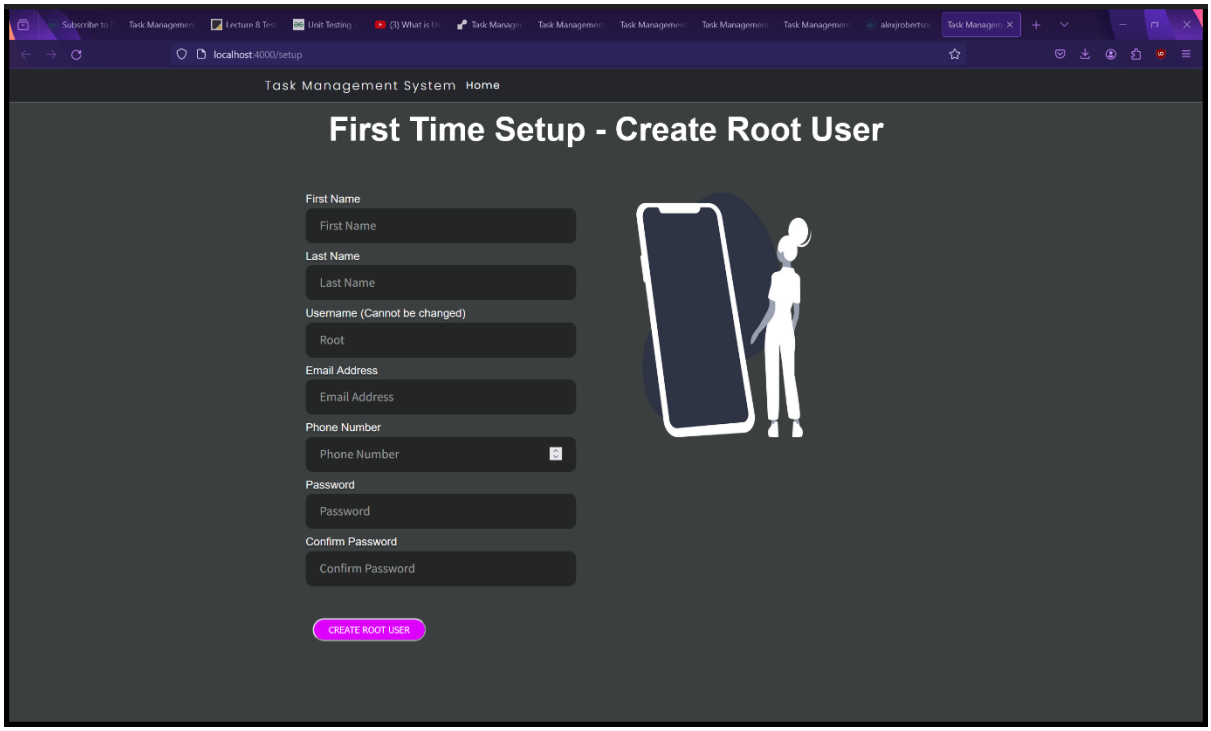


Figure 143 - Setup page of the Clean Install version of the Task Management System in a web browser, running in Docker as a container.

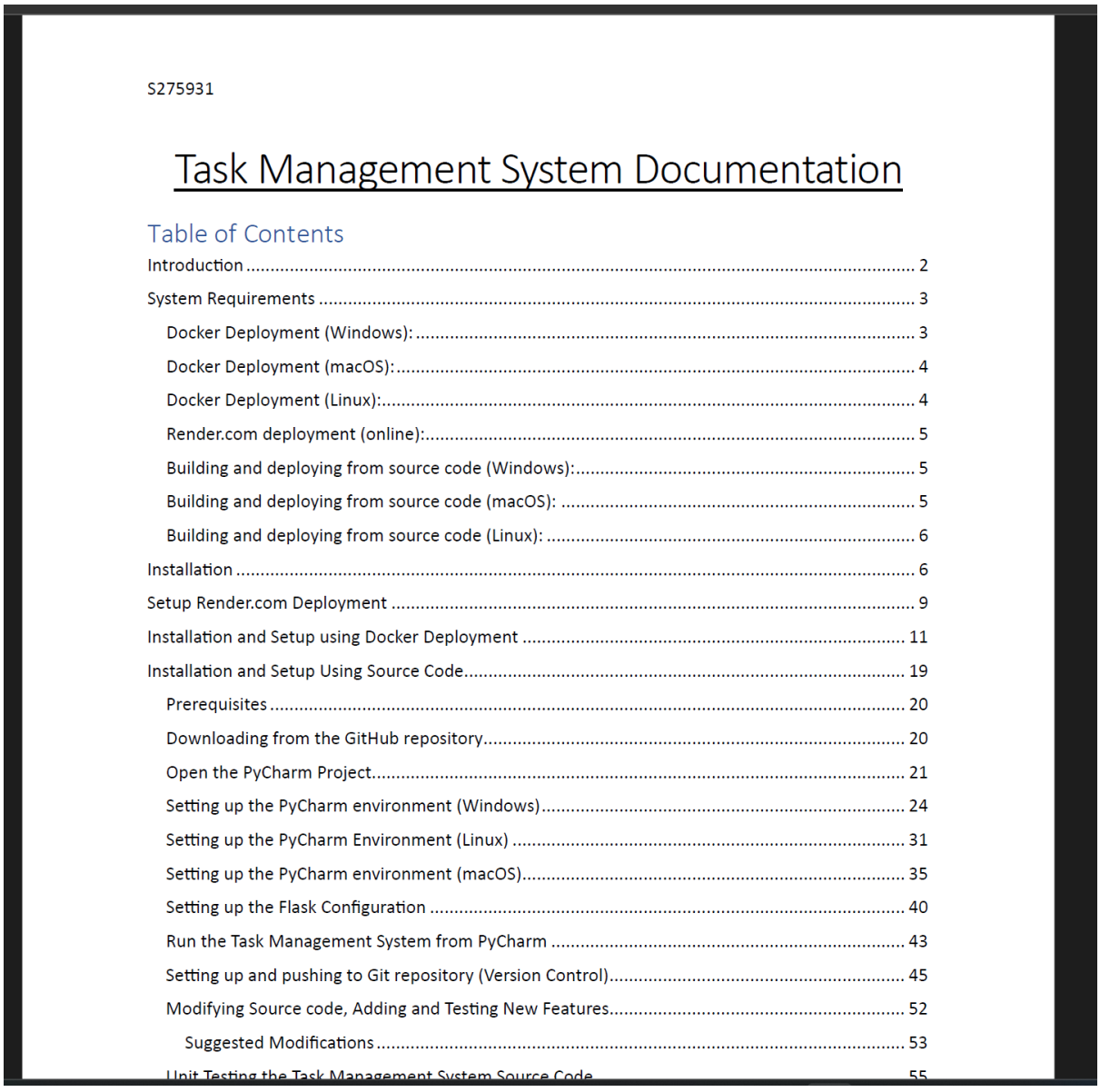
Release

I have released the Task Management System on the same platforms it has been deployed on including Render.com and DockerHub.

Documentation

The documentation is a separate document containing the Task Management System's system requirements, links to third party software documentations, license, terms of use, installation instructions, deployment instructions and user manual with screenshots. The documentation is available as a Word document, PDF file and as a Wiki on the Task Management System's GitHub repositories. A smaller documentation is available on the DockerHub repositories.

Task Management System Documentation Wikis are available here: https://github.com/AlexanderJohnRobertson/task-management-system/wiki/Task-Management-System-Documentation and https://github.com/AlexanderJohnRobertson/task-management-system-clean-install/wiki/Task-Management-System-%28Clean-Install%29-Documentation .



S275931

Task Management System Documentation

Table of Contents

Figure 144 - Screenshot of the Task Management System documentation.

Introduction

The Task Management System is a Python-based web application that manages tasks and projects for an organization to help the organisation keep track of tasks and their associated projects. Features include a database that contains three tables (one with a list of tasks, another with a list of projects and thirdly with a list of users), user roles (Administrator, Root User, Standard User), Create, Read, Update and Delete functionality for tasks, projects and users. Administrators have full control of the

2

S275931

Task Management system (Add, Update, View and Delete tasks and projects, manage other users (change account type and password, view details of all users, delete users, block and unblock users), factory reset the Task Management System, and manage their own account. Standard users are restricted therefore can only add and view tasks and projects, view and manage their own account. Standard users cannot update or delete tasks and projects, view and manage other users or factory reset the Task Management system. For security, authentication is required for all users and all accounts are password protected. The passwords are encrypted and stored as hashes in the database in the event the application or database gets hacked, randomly generated session IDs authenticate each session, Passwords are required to have minimum length of 10 characters and must contain at least one uppercase letter, one lowercase letter, one number, one special character and must not be a common password. The Task Management System is cross-platform and can run as a container on a computer or server running Windows, Mac OS or Linux with Docker installed, hosted online as a web service on hosting sites such as Render.com linked to a GitHub repository (It could also be hosted on other web hosting platforms that are compatible with Python 3.12, Flask and Gunicorn). You can also deploy the Task Management System directly from its source code by cloning its Git repository, run app.py in a suitable Python IDE such as PyCharm and have the Python 3.12 interpreter installed on your computer, then deploy on a containerisation platform of your choice such as Docker (others may require the Dockerfile to be replaced with their own containerisation file) or on a web hosting platform such as Render.com.

Figure 145 - Screenshot of the Task Management System documentation.

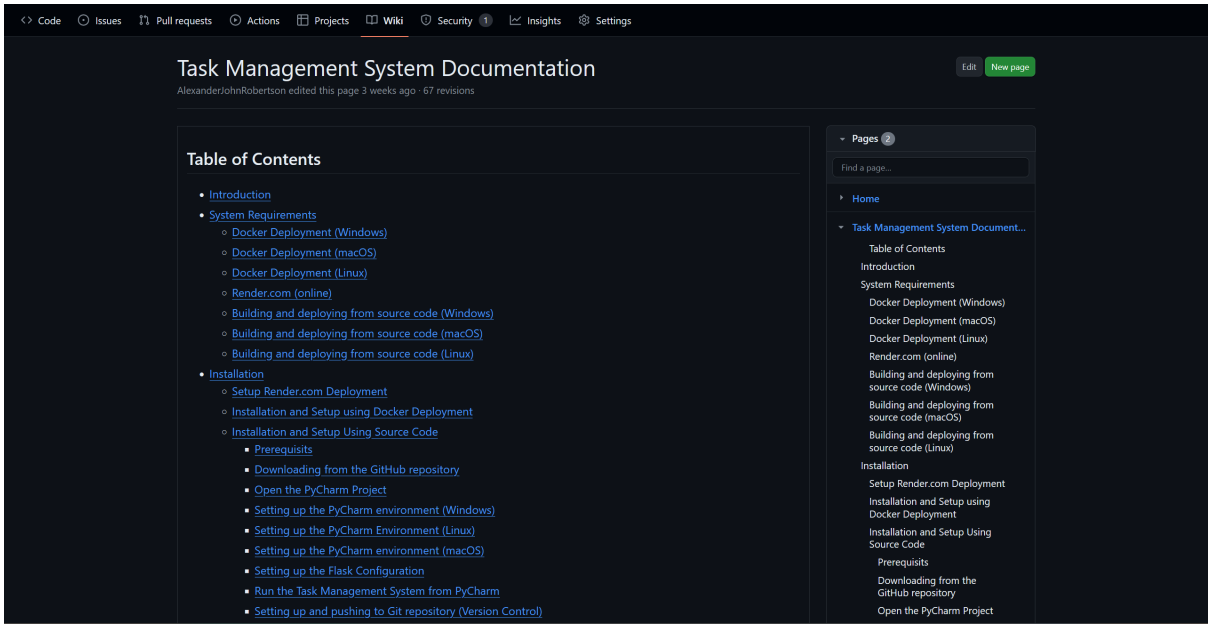


Figure 146 - Task Management System documentation on the GitHub Wiki.

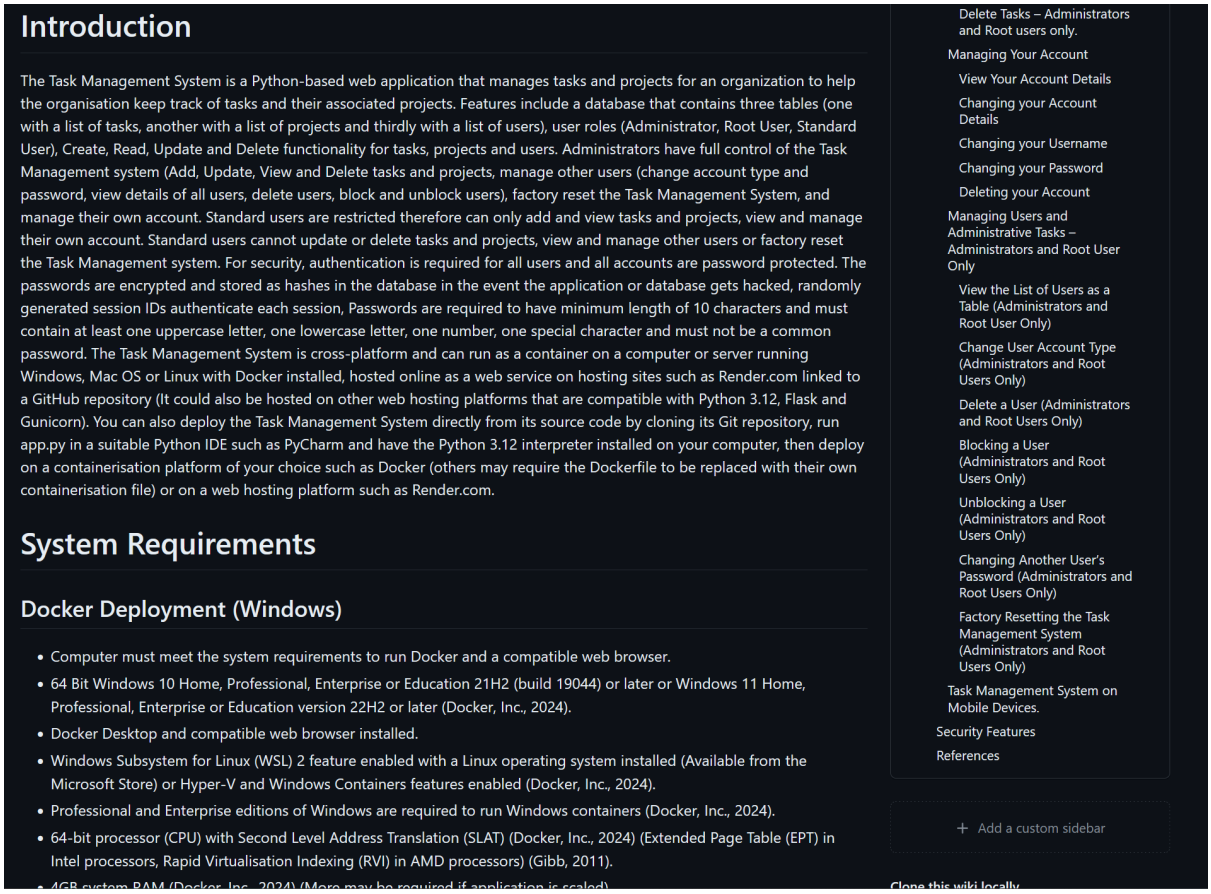


Figure 147 - Task Management System documentation on the GitHub Wiki.

Security and reliability of the system

The Task Management System has several security features for data protection and to prevent unauthorised access, hacking and misuse:

1. Password Encryption using SHA3-512-bit encryption to store the passwords as hashes in the database. Therefore, if the database gets hacked the hacker will be unable to enter the password stored in the database and will get an error instead.

```
if password != confirmPassword: #if the password and confirm password do not match flash an error message
    flash('Passwords do not match!')
else:
    for i in range(10): #password hashing encryption
        password = hashlib.sha3_512(password.encode('utf-8'))
        password = password.hexdigest()
```

Figure 148 - SHA3-512-bit encryption Python code for password encryption.

```
try:
    for i in range(10): #password hashing encryption
            password = hashlib.sha3_512(password.encode('utf-8'))
            password = password.hexdigest()
    database = r"database.db" #database file
    conn = None
    conn = sqlite3.connect(database) #connecting to the database
    cur = conn.cursor()
    cur.execute( __sql: 'SELECT * FROM users WHERE username = ? AND password = ?', __parameters: (username, password)) #querying the database
    user = cur.fetchall()
    cur.close()
    User = user[0]
    Uname = User[0]
    Pword = User[1]
    blocked = User[7]
    failedLoginAttempts = User[8]
    if Uname == username and Pword == password: #if the username and password are correct
```

Figure 149 - SHA3-512-bit encryption Python code for password encryption.

Structure | Data | Constraints | Indexes | Triggers | DDL

Grid view | Form view

Filter data | Total rows loaded: 14

	username	password	forename	surname	email	phone	userType	blocked	failedLogi	sessionID
1	Manchester000	491a0812d10f16546ca45ca12610224014028a204abe50f4c66decd64f6ea40f7ca8a9d64ac88b3a9079aa08f4982d59a43f73e8bef8753d89a1c05ad2c8f5bd	Kane	Fenlon	kfenlon@hotmail.com	42342342	Standard	No	0	0
2	LFenlonVolcano	7a6efbb087d2cb28efc3cf7651bcfab3f36cd6c9bf85ead4c614a62640b7aeb0934fda070bbec148bf19688062fcd8bc146e262c7b6ea07ee386d4916b5c8236	Lauren	Durury	ldurury@msn.com	234223424	Standard	No	0	0
3	GemmaWheels	4fde2afa1af7342236b05dbc67d3ce58ae9612903e2bf413e6614d5940ea8e7fe846a5f1c24cd5a0f3b681cab7cb6e896e3f250f276c40cb1a4ad5e9b8d67d0a	Gemma	Bird	gbird@hotmail.com	2323466	Standard	No	0	0
4	kellyG	7a6efbb087d2cb28efc3cf7651bcfab3f36cd6c9bf85ead4c614a62640b7aeb0934fda070bbec148bf19688062fcd8bc146e262c7b6ea07ee386d4916b5c8236	Kelly	Gates	kgates@outlook.com	78624868243	Standard	No	0	0
5	ethomas123	b48297d22c02fd065479bb9cfdfddf1f2aee86450b8853367e023459bdfd2ff89cba9d742d6638a5a2da1b202c1eb1bbc2be5fe842328f5ea7be9f659cef25e0	Ellie	Thomas	ethomas@icloud.com	836446834	Standard	No	0	0
6	ArasakaCEO123	2b8b3eeca8d3bff844df4293c3d1d38b2700ee4d5c551e960765481884d926ffe34347d8a0f2233b6b395b75175c031c02ed8a449ca164c2f3b015fc752ca3b8	Bill	Thomas	bthomas@hotmail.com	0124343767 6	Standard	No	0	0
7	DandyLion000	295da93c654e2b52da7b9f97045027b39250b58aecacc07dcf3d117d61dd97463a66fb226ad70c4cc0ea799f07bd540254bae9de92fde0494f87c67fe0192263	Daisy	Robinson	drobinson@icloud.com	9835973538	Administrator	No	0	0
8	elasticbands123	30a56dc2d1717646f0b0bdcb3895a37f0c8c8145067236eb1bbdfdc80785a91d0e1e23df994c34d564d42625e84b6af2ceb460ed730ed0d851330a75316e89ca	Sophie	Rivett	srivett@outlook.com	387548735	Standard	No	0	0
9	FlowerMeadow111	b5746f5ec139b2b8eb5489f2da70031c34cce65af0961200662d2f204a88eea2f158bc1e8d016d3b0f30f89af04004ff87343b8a29822b69ba4cb91feed2fbec	Daisy	Yau	dyau@icloud.com	3546653476	Standard	No	0	0
10	FrankRivett3000	ac8c86fd7f7852dd1bb6f4674e7f475221f3f9f1333cc2eb8bc08ff3673726734a48c290f5f7dcb5050784155b2e36df3278a1a2109ccdc03dd4c13f7e0ef984	Frank	Rivett	frivett@hotmail.com	93875937457	Standard	No	0	0
11	aGillick999	66f1070b5e970e3aa970f3320fbae1f2e22fb8e96694b908017cf7daac6952cf6de734127b8887017f06ea8a4e4f1cc3413184e402e35eb3924b71a5bde21cad	Alex	Gillick	lgillick@outlook.com	843428749287	Administrator	No	0	0
12	Birmingham888	583090a35af838315d90f712b435e9a39e105d077406f49812af2f341c876a657c26cc4526bb7910c2b2dd02834eb52dd7a3a2a4fc43ffa493bc4cae0d3ec7f6	Linda	Bone	lbone97@hotmail.co.uk	0392743979	Standard	No	0	0
13	Root	bee04f1856e95dede886a5ea8fd3b292b2f74e51f9ca7fabcbe584343a4dd1eba2c2a28253689323a016c1187268fdeede98018b720bbf00049acdf4cfda509b	Alexandra	Robinson	s275931@uos.ac.uk	30903853809	Administrator	No	0	9666523499
14	barneyPurpleDinosaur	98e2ec6325b2bea65fadb829be16e2c358007f3ba0216ce45b7ffbab9085c3f7d87c91d63cbc71b3d67a823f5074c910760cb372d1df61308af905aa7534ad6b	Alice	Robinson	robertsona97@icloud.com	983749739378	Administrator	No	0	0

Figure 150 - User passwords encrypted and stored as hashes in the Task Management System database (users table). You cannot log in by copying these hashes from the database into the Task Management System's login page.

2. Minimum password requirements – The application uses regular expressions to validate that the passwords meet the minimum requirements when setting up the application, creating an account and changing the password. These requirements are:
    a. Minimum of 10 characters
    b. At least one uppercase letter
    c. At least one lowercase letter
    d. At least one number
    e. At least one special character
    f. Must not be in the common passwords blacklist.

```python
if request.method == 'POST': #get form details
    username = request.form['username']
    password = request.form['password']
    confirmPassword = request.form['confirmPassword']
    firstname = request.form['firstname']
    lastname = request.form['lastname']
    email = request.form['email']
    phonenumber = request.form['phonenumber']
    accountType = "Standard" #set the account type to standard
    blocked = "No" #set the blocked status to no
    failedLoginAttempts = 0 #set the failed login attempts to 0
```

Figure 151 - Using POST RESTful API method to send account details to Python program and verifying that the use is not blocked.

```python
checkLowercase = re.search( pattern: r'[a-z]', password) #use regular expressions to validate the password
checkUppercase = re.search( pattern: r'[A-Z]', password)
checkNumber = re.search( pattern: r'[0-9]', password)
checkSpecialChar = re.search( pattern: r'[^A-Za-z0-9]', password)
if not firstname: #validate the form details
    flash('First Name is required!')
elif not lastname:
    flash('Last Name is required!')
elif not username:
    flash('Username is required!')
elif not email:
    flash('Email Address is required!')
elif not phonenumber:
    flash('Phone Number is required!')
elif not username:
    flash('Username is required!')
elif not password:
    flash('Password is required!')
elif not confirmPassword:
    flash('Confirm Password is required!')
elif not checkLowercase: #validate the password
    flash('Password must contain at least one lowercase letter!')
elif not checkUppercase:
```

Figure 152 - Using Regular Expressions to make sure the passwords meet the minimum requirements of one uppercase letter, one lowercase letter, one number and one special character. If statements validating the account creation form details.

```
elif not checkUppercase:
    flash('Password must contain at least one uppercase letter!')
elif not checkNumber:
    flash('Password must contain at least one number!')
elif not checkSpecialChar:
    flash('Password must contain at least one special character!')
elif len(password) < 10: # minimum password length is 10 characters
    flash('Password must be at least 10 characters long!')
elif password == username:
    flash('Password cannot be the same as the username!')
elif password == firstname:
    flash('Password cannot be the same as the first name!')
elif password == lastname:
    flash('Password cannot be the same as the last name!')
elif password == email:
    flash('Password cannot be the same as the email!')
elif password == phonenumber:
    flash('Password cannot be the same as the phone number!')
elif password in global_var2(): #if the password is in the common passwords list flash an error message
    flash('Password is too common!')
else:
    try:
        database = r"database.db" #database file
```

Figure 153 - If statements validating that the password meets the requirements and making sure that the password is no less than 10 characters in length.

```
def global_var2():
    """This function returns a list of common passwords."""
    global commonPasswords #declaring the global variable
    commonPasswords = ["Hello", "Password","Password123@", "123456", "123456789", "qwerty", "password",
                       "1234567", "12345678", "12345", "iloveyou", "111111", "123123", "abc123", "qwerty123",
                       "1q2w3e4r", "admin", "qwertyuiop", "654321", "555555", "lovely", "7777777", "welcome",
                       "888888", "princess", "dragon", "password1", "123qwe", "666666", "1qaz2wsx", "121212",
                       "123654", "superman", "qazwsx", "1234qwer", "asdf", "zxcvbnm", "qwe123", "123qweasd",
                       "admin123", "1234567890", "123456a", "123456q", "123456789a", "123456789q", "123456789z",
                       "123456789x", "123456789c", "123456789v", "123456789b", "123456789n", "123456789m",
                   "123456", "123456789", "12345", "12345678", "111111", "123123", "1234567890", "234567","qwerty123"
                   "000000", "1q2w3e", "aa12345678", "abc123", "password1", "  1234", "qwertyuiop", "123321", "password123"]
    return commonPasswords
```

Figure 154 - Common password blacklist. Users are not allowed to use these passwords.

3. Maximum failed login attempts – if the user enters the wrong password three times consecutively the Task Management System automatically blocks the user and can only be unblocked by an administrator to prevent guessing the password and brute force attacks. The failed login counter resets if the user successfully logs in before getting blocked.

View Users

USERNAME	FIRST NAME	LAST NAME	EMAIL ADDRESS	PHONE NUMBER	ACCOUNT TYPE	BLOCKED
Manchester000	Kane	Fenlon	kfenlon@hotmail.com	42342342	Standard	No
LFenlonVolcano	Lauren	Durury	ldurury@msn.com	234223424	Standard	No
GemmaWheels	Gemma	Bird	gbird@hotmail.com	2323466	Standard	No
kellyG	Kelly	Gates	kgates@outlook.com	78624868243	Standard	No
ethomas123	Ellie	Thomas	ethomas@icloud.com	836446834	Standard	No
ArasakaCEO123	Bill	Thomas	bthomas@hotmail.com	012434376876	Standard	No
DandyLion000	Daisy	Robinson	drobinson@icloud.com	9835973538	Administrator	No
elasticbands123	Sophie	Rivett	srivett@outlook.com	387548735	Standard	No
FlowerMeadow111	Daisy	Yau	dyau@icloud.com	3546653476	Standard	No
FrankRivett3000	Frank	Rivett	frivett@hotmail.com	93875937457	Standard	No
aGillick999	Alex	Gillick	lgillick@outlook.com	843428749287	Administrator	No
Birmingham888	Linda	Bone	lbone97@hotmail.co.uk	0392743979	Standard	No
Root	Alexandra	Robinson	s275931@uos.ac.uk	30903853809	Administrator	No
barneyPurpleDinosaur	Alice	Robinson	robertsona97@icloud.com	98374939378	Administrator	No

Figure 155 - List of user details showing that no users are blocked.

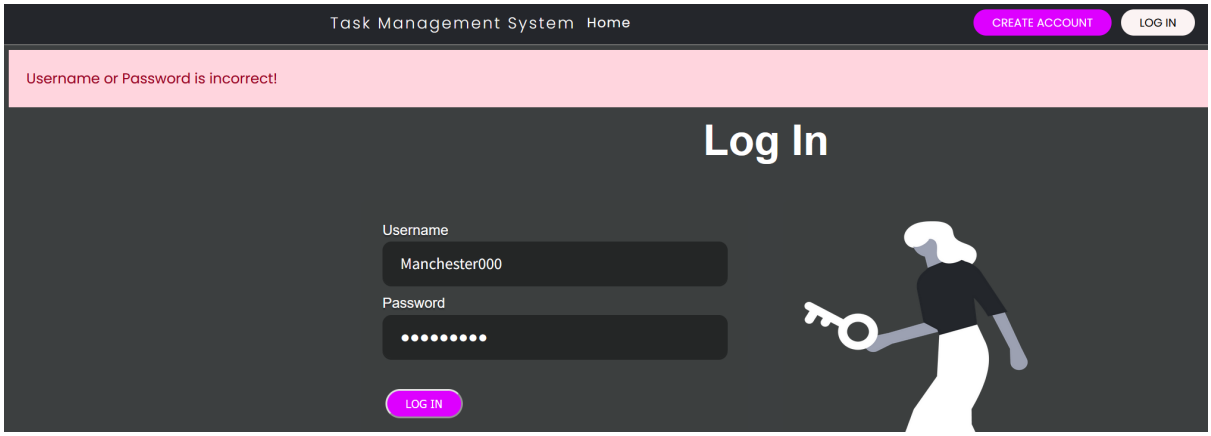


Figure 156 - User enters incorrect password.

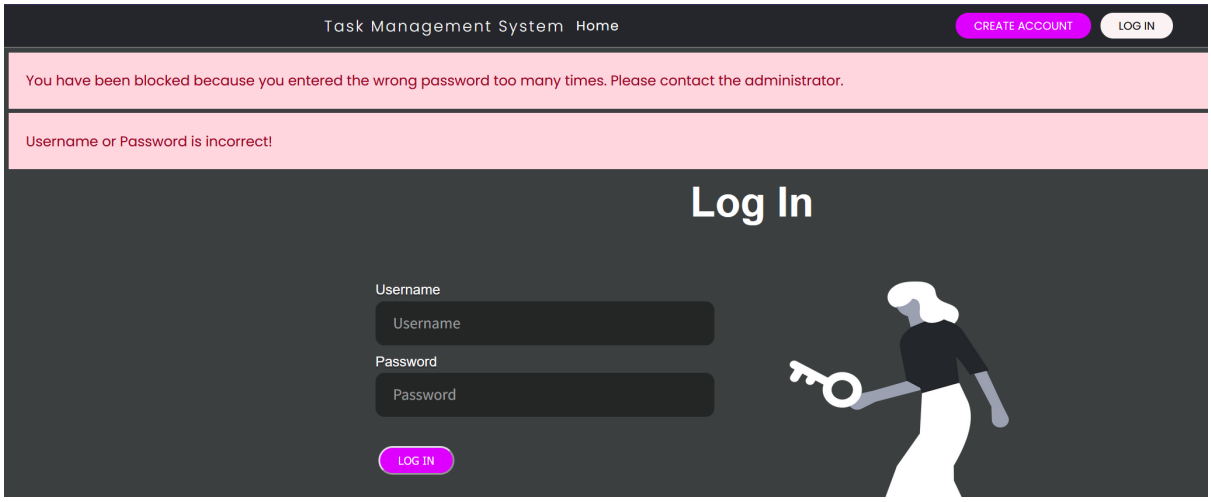


Figure 157 - User is automatically blocked because they entered the wrong password three times.

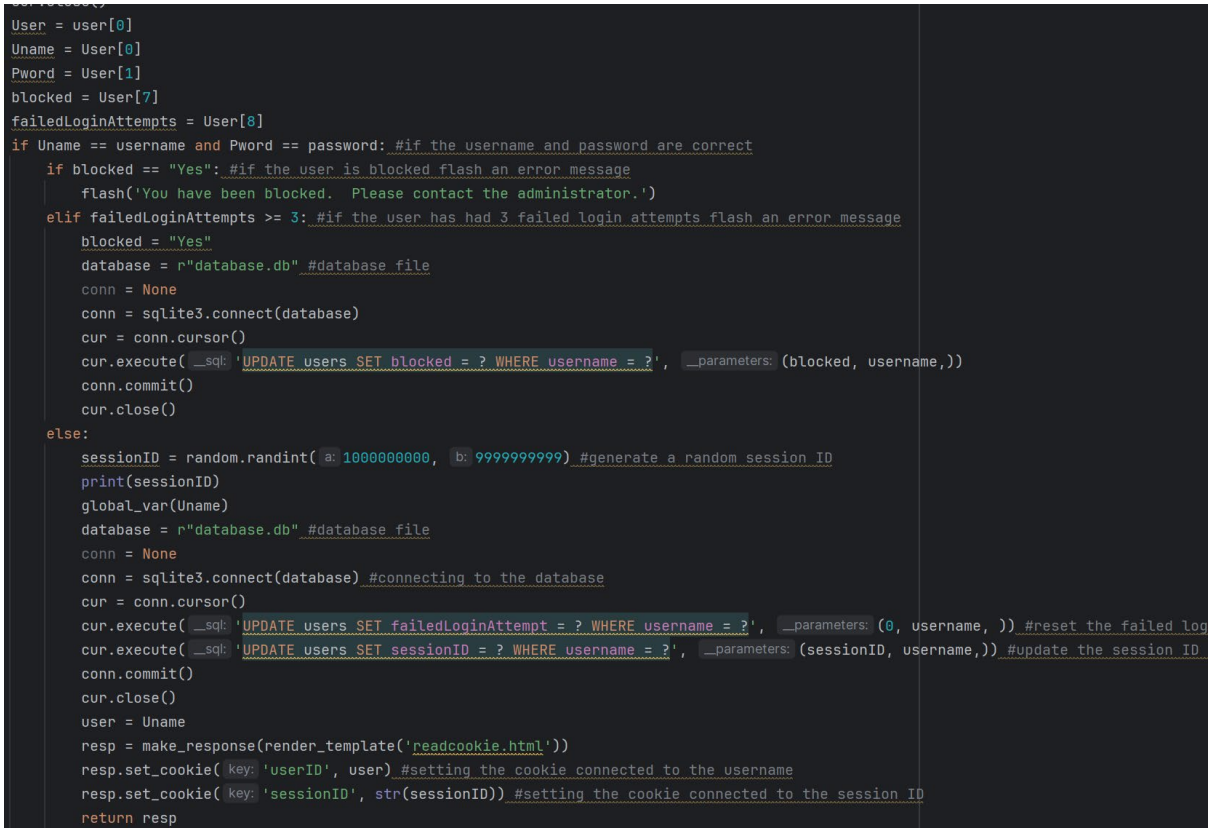
View Users

USERNAME	FIRST NAME	LAST NAME	EMAIL ADDRESS	PHONE NUMBER	ACCOUNT TYPE	BLOCKED
Manchester000	Kane	Fenlon	kfenlon@hotmail.com	42342342	Standard	Yes
LFenlonVolcano	Lauren	Durury	ldurury@msn.com	234223424	Standard	No
GemmaWheels	Gemma	Bird	gbird@hotmail.com	2323466	Standard	No
kellyG	Kelly	Gates	kgates@outlook.com	78624868243	Standard	No
ethomas123	Ellie	Thomas	ethomas@icloud.com	836446834	Standard	No
ArasakaCEO123	Bill	Thomas	bthomas@hotmail.com	012434376876	Standard	No
DandyLion000	Daisy	Robinson	drobinson@icloud.com	9835973538	Administrator	No
elasticbands123	Sophie	Rivett	srivett@outlook.com	387548735	Standard	No
FlowerMeadow111	Daisy	Yau	dyau@icloud.com	3546653476	Standard	No
FrankRivett3000	Frank	Rivett	frivett@hotmail.com	93875937457	Standard	No
aGillick999	Alex	Gillick	lgillick@outlook.com	843428749287	Administrator	No
Birmingham888	Linda	Bone	lbone97@hotmail.co.uk	0392743979	Standard	No
Root	Alexandra	Robinson	s275931@uos.ac.uk	30903853809	Administrator	No
barneyPurpleDinosaur	Alice	Robinson	robertsona97@icloud.com	983749739378	Administrator	No

Figure 158 - User now shows up in the database as blocked on View Users page.

	username	password	forename	surname	email	phone	userType	blocked	failedLoginAttempt	sessionID
1	Manchester000	491a0812d10f16546ca45ca12610224014028a204abe50f4c66decd64f6ea40f7ca8a9d64ac88b3a9079aa08f4982...	Kane	Fenlon	kfenlon@hotmail.com	42342342	Standard	Yes	3	0
2	LFenlonVolcano	7a6efbb087d2cb28efc3cf7651bcfab3f36cd6c9bf85ead4c614a62640b7aeb0934fda070bbec148bf19688062fcd8...	Lauren	Durury	ldurury@msn.com	234223424	Standard	No	0	0
3	GemmaWheels	4fde2afa1af7342236b05dbc67d3ce58ae9612903e2bf413e6614d5940ea8e7fe846a5f1c24cd5a0f3b681cab7cb6...	Gemma	Bird	gbird@hotmail.com	2323466	Standard	No	0	0
4	kellyG	7a6efbb087d2cb28efc3cf7651bcfab3f36cd6c9bf85ead4c614a62640b7aeb0934fda070bbec148bf19688062fcd8...	Kelly	Gates	kgates@outlook.com	78624868243	Standard	No	0	0
5	ethomas123	b48297d22c02fd065479bb9cfdfddf1f2aee86450b8853367e023459bdfd2ff89cba9d742d6638a5a2da1b202c1eb...	Ellie	Thomas	ethomas@icloud.com	836446834	Standard	No	0	0
6	ArasakaCEO123	2b8b3eeca8d3bff844df4293c3d1d38b2700ee4d5c551e960765481884d926ffe34347d8a0f2233b6b395b75175c...	Bill	Thomas	bthomas@hotmail.com	012434376876	Standard	No	0	0
7	DandyLion000	295da93c654e2b52da7b9f97045027b39250b58aecacc07dcf3d117d61dd97463a66fb226ad70c4cc0ea799f07bd...	Daisy	Robinson	drobinson@icloud.com	9835973538	Administrator	No	0	0
8	elasticbands123	30a56dc2d1717646f0b0bdcb3895a37f0c8c8145067236eb1bbdfdc80785a91d0e1e23df994c34d564d42625e84...	Sophie	Rivett	srivett@outlook.com	387548735	Standard	No	0	0
9	FlowerMeadow111	b5746f5ec139b2b8eb5489f2da70031c34cce65af0961200662d2f204a88eea2f158bc1e8d016d3b0f30f89af0400...	Daisy	Yau	dyau@icloud.com	3546653476	Standard	No	0	0
10	FrankRivett3000	ac8c86fd7f7852dd1bb6f4674e7f475221f3f9f1333cc2eb8bc08ff3673726734a48c290f5f7dcb5050784155b2e36d...	Frank	Rivett	frivett@hotmail.com	93875937457	Standard	No	0	0
11	aGillick999	66f1070b5e970e3aa970f3320fbae1f2e22fb8e96694b908017cf7daac6952cf6de734127b8887017f06ea8a4e4f1c...	Alex	Gillick	lgillick@outlook.com	843428749287	Administrator	No	0	0
12	Birmingham888	583090a35af838315d90f712b435e9a39e105d077406f49812af2f341c876a657c26cc4526bb7910c2b2dd02834e...	Linda	Bone	lbone97@hotmail.co.uk	0392743979	Standard	No	0	0
13	Root	bee04f1856e95dede886a5ea8fd3b292b2f74e51f9ca7fabcbe584343a4dd1eba2c2a2825368932 3a016c1187268f...	Alexandra	Robinson	s275931@uos.ac.uk	30903853809	Administrator	No	0	0
14	barneyPurpleDinosaur	98e2ec6325b2bea65fadb829be16e2c358007f3ba0216ce45b7ffbab9085c3f7d87c91d63cbc71b3d67a823f5074c...	Alice	Robinson	robertsona97@icloud.com	983749739378	Administrator	No	0	0

Figure 159 - User now shows up in the database as blocked (SQLite Studio).



```
User = user[0]
Uname = User[0]
Pword = User[1]
blocked = User[7]
failedLoginAttempts = User[8]
if Uname == username and Pword == password: #if the username and password are correct
    if blocked == "Yes": #if the user is blocked flash an error message
        flash('You have been blocked.  Please contact the administrator.')
    elif failedLoginAttempts >= 3: #if the user has had 3 failed login attempts flash an error message
        blocked = "Yes"
        database = r"database.db" #database file
        conn = None
        conn = sqlite3.connect(database)
        cur = conn.cursor()
        cur.execute( __sql: 'UPDATE users SET blocked = ? WHERE username = ?',  __parameters: (blocked, username,))
        conn.commit()
        cur.close()
    else:
        sessionID = random.randint( a: 1000000000,  b: 9999999999) #generate a random session ID
        print(sessionID)
        global_var(Uname)
        database = r"database.db" #database file
        conn = None
        conn = sqlite3.connect(database) #connecting to the database
        cur = conn.cursor()
        cur.execute( __sql: 'UPDATE users SET failedLoginAttempt = ? WHERE username = ?',  __parameters: (0, username, )) #reset the failed log
        cur.execute( __sql: 'UPDATE users SET sessionID = ? WHERE username = ?',  __parameters: (sessionID, username,)) #update the session ID
        conn.commit()
        cur.close()
        user = Uname
        resp = make_response(render_template('readcookie.html'))
        resp.set_cookie( key: 'userID', user) #setting the cookie connected to the username
        resp.set_cookie( key: 'sessionID', str(sessionID)) #setting the cookie connected to the session ID
        return resp
```

Figure 160 - Python code that automatically blocks users after 3 incorrect passwords entered during failed login attempts. Code also denies access to the user if they have been blocked.

```
            return resp
    else: #if the username and password are incorrect flash an error message
        database = r"database.db"  # database file
        conn = None
        conn = sqlite3.connect(database)  # connecting to the database
        cur = conn.cursor()
        cur.execute( _sql: 'SELECT failedLoginattempt FROM users WHERE username = ?',
                     _parameters: (username,))  # querying the database
        failedLoginAttempts = cur.fetchall()
        failedLoginAttempts = failedLoginAttempts[0][0]
        failedLoginAttempts = failedLoginAttempts + 1 #increment the failed login attempts if password is incorrect
        print(failedLoginAttempts)
        cur.execute( _sql: 'UPDATE users SET failedLoginAttempt = ? WHERE username = ?',
                     _parameters: (failedLoginAttempts, username,)) # updating the failed login attempts in the database
        conn.commit()
        cur.close()
        if failedLoginAttempts >= 3: #block the user if there are 3 failed login attempts
            blocked = "Yes"
            database = r"database.db"
            conn = None
            conn = sqlite3.connect(database)
            cur = conn.cursor()
            cur.execute( _sql: 'UPDATE users SET blocked = ? WHERE username = ?', _parameters: (blocked, username,))
            conn.commit()
            cur.close()
            flash(
                'You have been blocked because you entered the wrong password too many times. Please contact the administrator.')
        flash('Username or Password is incorrect!')
except IndexError: #if there is an error caused by incorrect login details flash an error message
    try:
        database = r"database.db"  # database file
        conn = None
        conn = sqlite3.connect(database)  # connecting to the database
        cur = conn.cursor()
        cur.execute( _sql: 'SELECT failedLoginattempt FROM users WHERE username = ?', _parameters: (username,))  # querying the database
        failedLoginAttempts = cur.fetchall()
        failedLoginAttempts = failedLoginAttempts[0][0]
        failedLoginAttempts = failedLoginAttempts + 1 #increment the failed login attempts if password is incorrect
        cur.execute( _sql: 'UPDATE users SET failedLoginAttempt = ? WHERE username = ?',  _parameters: (failedLoginAttempts, username,)) #updatin
```

Figure 161 - Python code that automatically blocks users after 3 incorrect passwords entered during failed login attempts. Code also denies access to the user if they have been blocked.

```
            try:
                database = r"database.db"  # database file
                conn = None
                conn = sqlite3.connect(database)  # connecting to the database
                cur = conn.cursor()
                cur.execute( _sql: 'SELECT failedLoginattempt FROM users WHERE username = ?', _parameters: (username,))  # querying the database
                failedLoginAttempts = cur.fetchall()
                failedLoginAttempts = failedLoginAttempts[0][0]
                failedLoginAttempts = failedLoginAttempts + 1 #increment the failed login attempts if password is incorrect
                cur.execute( _sql: 'UPDATE users SET failedLoginAttempt = ? WHERE username = ?', _parameters: (failedLoginAttempts, username,)) #upd
                conn.commit()
                cur.close()
                if failedLoginAttempts >= 3: #block the user if there are 3 failed login attempts
                    blocked = "Yes"
                    database = r"database.db"
                    conn = None
                    conn = sqlite3.connect(database)
                    cur = conn.cursor()
                    cur.execute( _sql: 'UPDATE users SET blocked = ? WHERE username = ?', _parameters: (blocked, username,))
                    conn.commit()
                    cur.close()
                    flash('You have been blocked because you entered the wrong password too many times. Please contact the administrator.')
                flash('Username or Password is incorrect!')
            except IndexError:
                flash('Username does not exist!')
    return render_template('login.html') #render the login.html page
```

Figure 162 - Python code that automatically blocks users after 3 incorrect passwords entered during failed login attempts. Code also denies access to the user if they have been blocked.

4. Authentication – The user must be logged in to perform task and project management operations, manage their account and perform administrative tasks. Unauthorised access to these URLs will result in an Access Denied error.

Figure 163 - If you try to access the Task Management System webpages by bypassing the login, then your access will be blocked, and you will receive an Access Denied error message instead of accessing the Task Management System webpages.

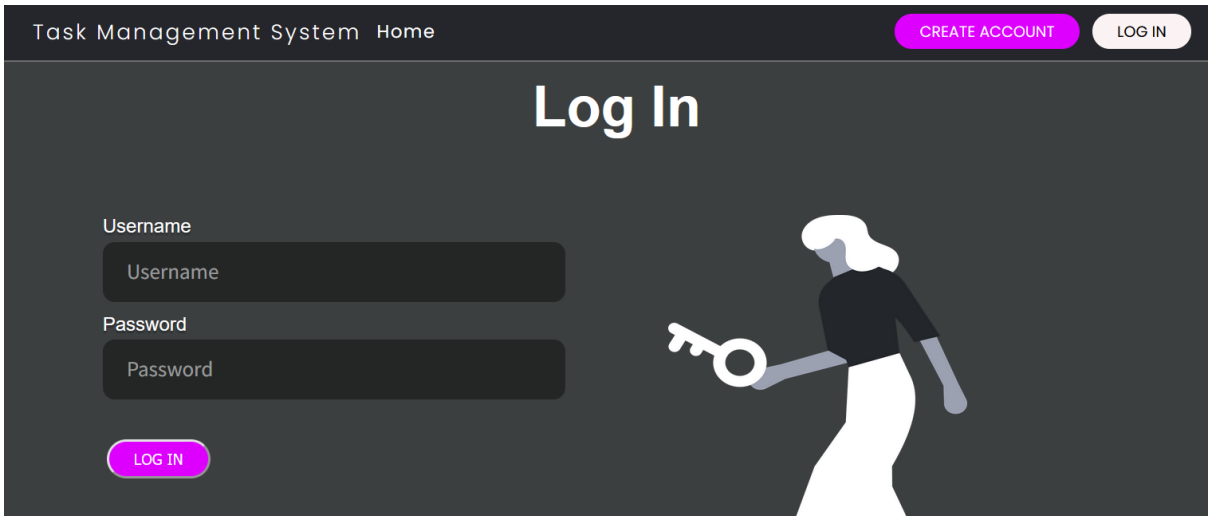


Figure 164 - Login page and form.

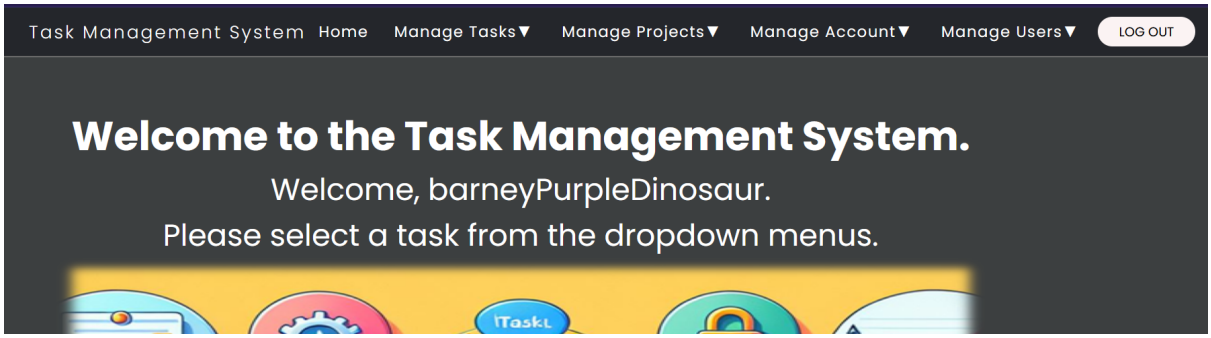


Figure 165 - User Home page after logging in. You will now be granted access to Task Management System webpages as your access is legitimate and has been authenticated.

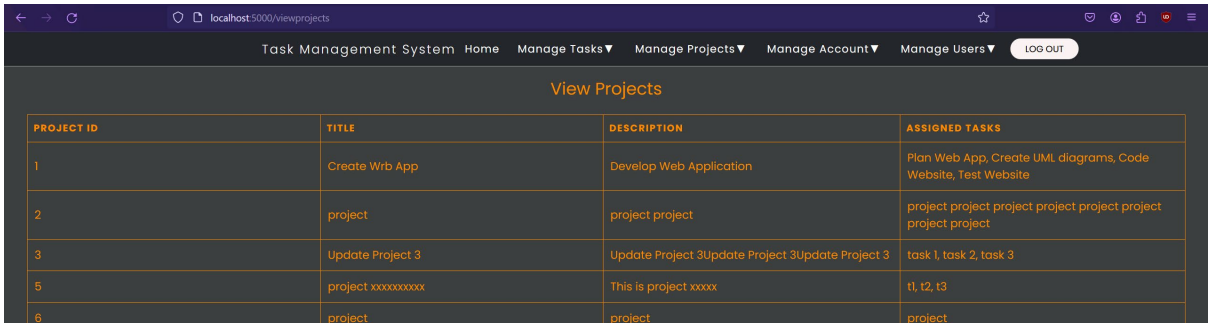


Figure 166 - User Home page after logging in. You will now be granted access to Task Management System webpages as your access is legitimate and has been authenticated.

5. Session ID and User ID cookies – The Task Management System uses cookies in the web browser containing a sessionID and the username for authentication. The username cookie contains the user's username when logged in and the backend code for each webpage checks that the username exists in the database to grant access to the pages. This also determines the roles of the user depending on the role stored in the database (administrator or standard). The SessionID is a randomly generated integer that is created when the user logs in or an account is created and is stored as a cookie in the web browser and in temporarily in the database users table. The username and sessionID in the cookies and the database must match to grant access to the Task Management System or else there will be an Access Denied error.

```
else:
    sessionID = random.randint( a: 1000000000, b: 9999999999) #generate a random session ID
    print(sessionID)
    global_var(Uname)
    database = r"database.db" #database file
    conn = None
    conn = sqlite3.connect(database) #connecting to the database
    cur = conn.cursor()
    cur.execute( _sql: 'UPDATE users SET failedLoginAttempt = ? WHERE username = ?', _parameters: (0, username, )) #reset the failed login attempts t
    cur.execute( _sql: 'UPDATE users SET sessionID = ? WHERE username = ?', _parameters: (sessionID, username,)) #update the session ID in the databa
    conn.commit()
    cur.close()
    user = Uname
    resp = make_response(render_template('readcookie.html'))
    resp.set_cookie( key: 'userID', user) #setting the cookie connected to the username
    resp.set_cookie( key: 'sessionID', str(sessionID)) #setting the cookie connected to the session ID
    return resp
```

Figure 167 - Python code that generates the UserID cookie and the randomly generated SessionID cookie and database entry. The UserID cookie identifies which user you are and the SessionID authenticates your session as genuine and prevent bypassing the login process.

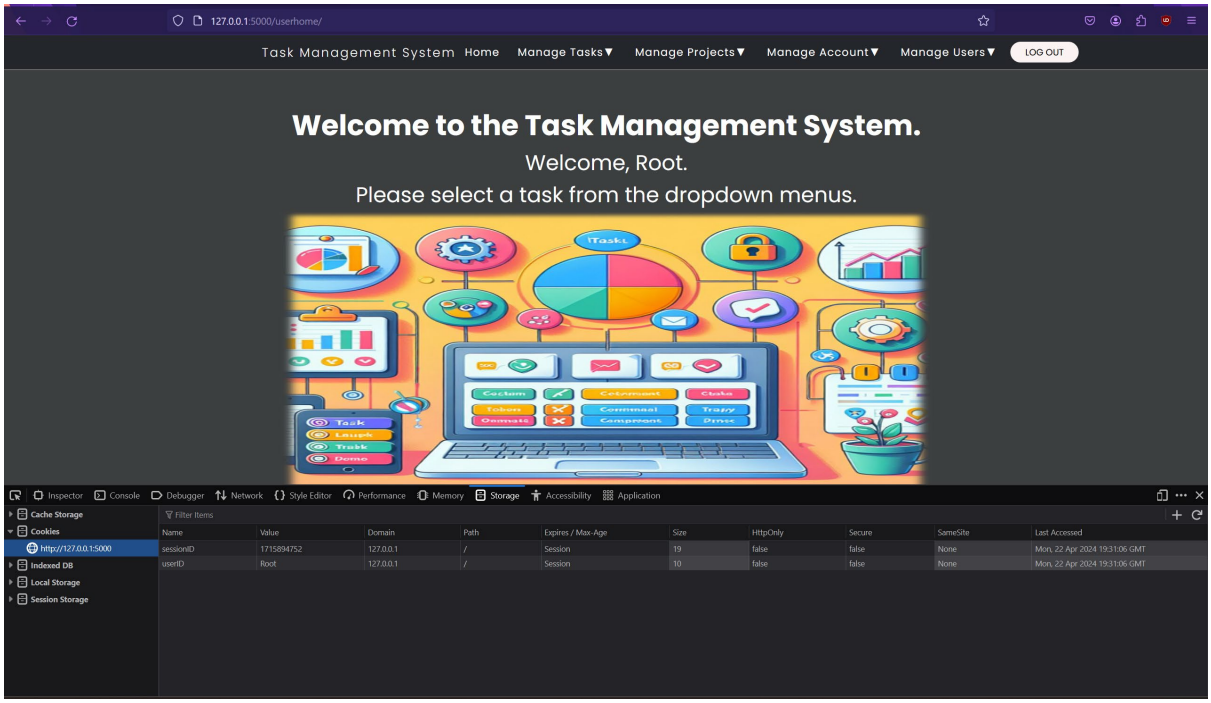


Figure 168 - UserID and SessionID cookies after logging in.

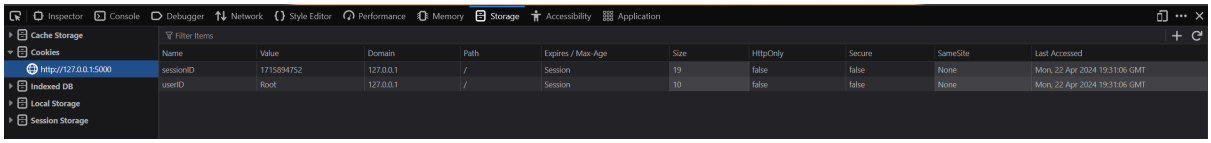


Figure 169 - UserID and SessionID cookies after logging in.

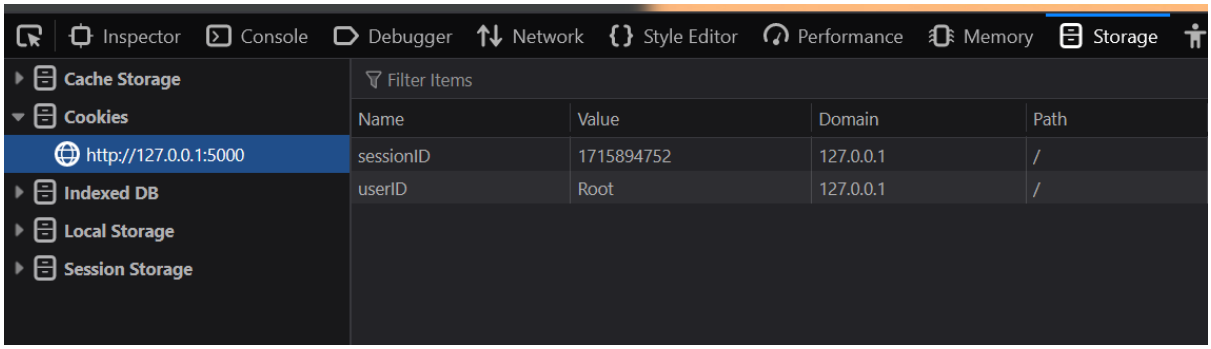


Figure 170 - UserID and SessionID cookies after logging in.

	username	password	forename	surname	email	phone	userType	blocked	failedLogi	sessionID
1	Manchester000	491a0812d10f16546ca45ca12610224014028a204abe50f4c66decd64f6ea40f7ca8a9d64ac88b3a9079aa08f4982...	Kane	Fenlon	kfenlon@hotmail.com	42342342	Standard	No	0	0
2	LFenlonVolcano	7a6efbb087d2cb28efc3cf7651bcfab3f36cd6c9bf85ead4c614a62640b7aeb0934fda070bbec148bf19688062fcd8...	Lauren	Durury	ldurury@msn.com	234223424	Standard	No	0	0
3	GemmaWheels	4fde2afa1af7342236b05dbc67d3ce58ae9612903e2bf413e6614d5940ea8e7fe846a5f1c24cd5a0f3b681cab7cb6...	Gemma	Bird	gbird@hotmail.com	2323466	Standard	No	0	0
4	kellyG	7a6efbb087d2cb28efc3cf7651bcfab3f36cd6c9bf85ead4c614a62640b7aeb0934fda070bbec148bf19688062fcd8...	Kelly	Gates	kgates@outlook.com	78624868243	Standard	No	0	0
5	ethomas123	b48297d22c02fd065479bb9cfdfddf1f2aee86450b8853367e023459bdfd2ff89cba9d742d6638a5a2da1b202c1eb...	Ellie	Thomas	ethomas@icloud.com	836446834	Standard	No	0	0
6	ArasakaCEO123	2b8b3eeca8d3bff844df4293c3d1d38b2700ee4d5c551e960765481884d926ffe34347d8a0f2233b6b395b75175c...	Bill	Thomas	bthomas@hotmail.com	012434376876	Standard	No	0	0
7	DandyLion000	295da93c654e2b52da7b9f97045027b39250b58aecacc07dcf3d117d61dd97463a66fb226ad70c4cc0ea799f07bd...	Daisy	Robinson	drobinson@icloud.com	9835973538	Administrator	No	0	0
8	elasticbands123	30a56dc2d1717646f0b0bdcb3895a37f0c8c8145067236eb1bbdfdc80785a91d0e1e23df994c34d564d42625e84...	Sophie	Rivett	srivett@outlook.com	387548735	Standard	No	0	0
9	FlowerMeadow111	b5746f5ec139b2b8eb5489f2da70031c34cce65af0961200662d2f204a88eea2f158bc1e8d016d3b0f30f89af0400...	Daisy	Yau	dyau@icloud.com	3546653476	Standard	No	0	0
10	FrankRivett3000	ac8c86fd7f7852dd1bb6f4674e7f475221f3f9f1333cc2eb8bc08ff3673726734a48c290f5f7dcb5050784155b2e36d...	Frank	Rivett	frivett@hotmail.com	93875937457	Standard	No	0	0
11	aGillick999	66f1070b5e970e3aa970f3320fbae1f2e22fb8e96694b908017cf7daac6952cf6de734127b8887017f06ea8a4e4f1c...	Alex	Gillick	lgillick@outlook.com	843428749287	Administrator	No	0	0
12	Birmingham888	583090a35af838315d90f712b435e9a39e105d077406f49812af2f341c876a657c26cc4526bb7910c2b2dd02834e...	Linda	Bone	lbone97@hotmail.co.uk	0392743979	Standard	No	0	0
13	Root	bee04f1856e95dede886a5ea8fd3b292b2f74e51f9ca7fabcbe584343a4dd1eba2c2a28253689323a016c118726f8f...	Alexandra	Robinson	s275931@uos.ac.uk	30903853809	Administrator	No	0	1715894752
14	barneyPurpleDinosaur	98e2ec6325b2bea65fadb829be16e2c358007f3ba0216ce45b7ffbab9085c3f7d87c91d63cbc71b3d67a823f5074c...	Alice	Robinson	robertsona97@icloud.com	98374973978	Administrator	No	0	0

Figure 171 - SessionID stored in the Task Management System's database.

Access Denied. You do not have permission to access the Task Management System.

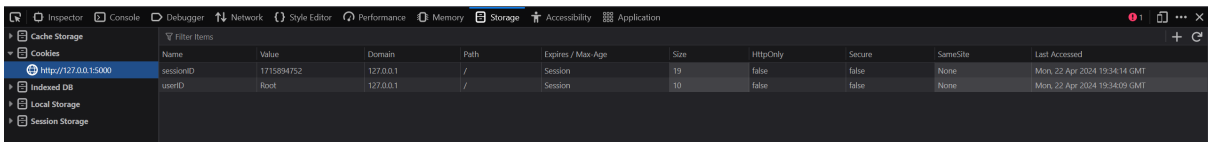


Figure 172 - Copying the UserID and SessionID then logging out then reusing them will result in an Access Denied error as the SessionID does in the cookie does not match the database entry, as it changes each time and must match. Logging out clears the sessionID cookie and database entry as well as the UserID cookie.

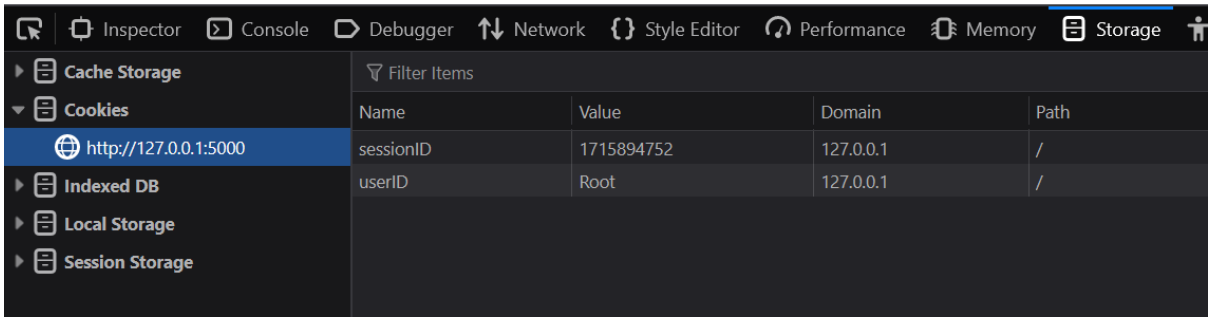


Figure 173 - Copying the userID and sessionID cookies will result in a failed login attempt as it no longer matches the sessionID in the database after logging out.

	username	password	forename	surname	email	phone	userType	blocked	failedLogi	sessionID
1	Manchester000	491a0812d10f16546ca45ca12610224014028a204abe50f4c66decd64f6ea40f7ca8a9d64ac88b3a9079aa08f4982...	Kane	Fenlon	kfenlon@hotmail.com	42342342	Standard	No	0	0
2	LFenlonVolcano	7a6efbb087d2cb28efc3cf7651bcfab3f36cd6c9bf85ead4c614a62640b7aeb0934fda070bbec148bf19688062fcd8...	Lauren	Durury	ldurury@msn.com	234223424	Standard	No	0	0
3	GemmaWheels	4fde2afa1af7342236b05dbc67d3ce58ae9612903e2bf413e6614d5940ea8e7fe846a5f1c24cd5a0f3b681cab7cb6...	Gemma	Bird	gbird@hotmail.com	2323466	Standard	No	0	0
4	kellyG	7a6efbb087d2cb28efc3cf7651bcfab3f36cd6c9bf85ead4c614a62640b7aeb0934fda070bbec148bf19688062fcd8...	Kelly	Gates	kgates@outlook.com	78624868243	Standard	No	0	0
5	ethomas123	b48297d22c02fd065479bb9cfdfddf1f2aee86450b8853367e023459bdfd2ff89cba9d742d6638a5a2da1b202c1eb...	Ellie	Thomas	ethomas@icloud.com	836446834	Standard	No	0	0
6	ArasakaCEO123	2b8b3eeca8d3bff844df4293c3d1d38b2700ee4d5c551e9607654818840926ffe34347d8a0f2233b6b395b75175c...	Bill	Thomas	bthomas@hotmail.com	012434376876	Standard	No	0	0
7	DandyLion000	295da93c654e2b52da7b9f97045027b39250b58aecacc07dcf3d117d61dd97463a66fb226ad70c4cc0ea799f07bd...	Daisy	Robinson	drobinson@icloud.com	9835973538	Administrator	No	0	0
8	elasticbands123	30a56dc2d1717646f0b0bdcb3895a37f0c8c8145067236eb1bbdfdc80785a91d0e1e23df994c34d564d42625e84...	Sophie	Rivett	srivett@outlook.com	387548735	Standard	No	0	0
9	FlowerMeadow111	b5746f5ec139b2b8eb5489f2da70031c34cce65af0961200662d2f204a88eea2f158bc1e8d016d3b0f30f89af0400...	Daisy	Yau	dyau@icloud.com	3546653476	Standard	No	0	0
10	FrankRivett3000	ac8c86fd7f7852dd1bb6f4674e7f475221f3f9f1333cc2eb8bc08ff3673726734a48c290f5f7dcb5050784155b2e36d...	Frank	Rivett	frivett@hotmail.com	93875937457	Standard	No	0	0
11	aGillick999	66f1070b5e970e3aa970f3320fbae1f2e22fb8e96694b908017cf7daac6952cf6de734127b8887017f06ea8a4e4f1c...	Alex	Gillick	lgillick@outlook.com	843428749287	Administrator	No	0	0
12	Birmingham888	583090a35af838315d90f712b435e9a39e105d077406f49812af2f341c876a657c26cc4526bb7910c2b2dd02834e...	Linda	Bone	lbone97@hotmail.co.uk	0392743979	Standard	No	0	0
13	Root	bee04f1856e95dede886a5ea8fd3b292b2f74e51f9ca7fabcbe584343a4dd1eba2c2a2825368932 3a016c1187268f...	Alexandra	Robinson	s275931@uos.ac.uk	30903853809	Administrator	No	0	0
14	barneyPurpleDinosaur	98e2ec6325b2bea65fadb829be16e2c358007f3ba0216ce45b7ffbab9085c3f7d87c91d63cbc71b3d67a823f5074c...	Alice	Robinson	robertsona97@icloud.com	983749739378	Administrator	No	0	0

Figure 174 - Copying the userID and sessionID cookies will result in a failed login attempt as it no longer matches the sessionID in the database after logging out.

6. User Roles – There are three user roles to prevent unauthorised changes and misuse:
    a. Standard – Able to create and view tasks and projects and manage their own account. Default account type upon account creation
    b. Administrator – Able to create, view, update and delete tasks and projects, manage their own account, manage other users, and perform administrative tasks.
    c. Root – Mandatory administrator account created during first time setup. Full administrator rights but with the following restrictions due to technical reasons:
        i. Username cannot be changed
        ii. Cannot be deleted
        iii. Cannot be blocked
        iv. Cannot be changed to standard user.

USERNAME	FIRST NAME	LAST NAME	EMAIL ADDRESS	PHONE NUMBER	ACCOUNT TYPE	BLOCKED
Manchester000	Kane	Fenlon	kfenlon@hotmail.com	42342342	Standard	Yes
LFenlonVolcano	Lauren	Durury	ldurury@msn.com	234223424	Standard	No
GemmaWheels	Gemma	Bird	gbird@hotmail.com	2323466	Standard	No
kellyG	Kelly	Gates	kgates@outlook.com	78624868243	Standard	No
ethomas123	Ellie	Thomas	ethomas@icloud.com	836446834	Standard	No
ArasakaCEO123	Bill	Thomas	bthomas@hotmail.com	012434376876	Standard	No
DandyLion000	Daisy	Robinson	drobinson@icloud.com	9835973538	Administrator	No
elasticbands123	Sophie	Rivett	srivett@outlook.com	387548735	Standard	No
FlowerMeadow111	Daisy	Yau	dyau@icloud.com	3546653476	Standard	No
FrankRivett3000	Frank	Rivett	frivett@hotmail.com	93875937457	Standard	No
aGillick999	Alex	Gillick	lgillick@outlook.com	843428749287	Administrator	No
Birmingham888	Linda	Bone	lbone97@hotmail.co.uk	0392743979	Standard	No
Root	Alexandra	Robinson	s275931@uos.ac.uk	30903853809	Administrator	No
barneyPurpleDinosaur	Alice	Robinson	robertsona97@icloud.com	983749739378	Administrator	No

Figure 175 - View Users page showing the user roles Standard and Administrator in the table of users' details.

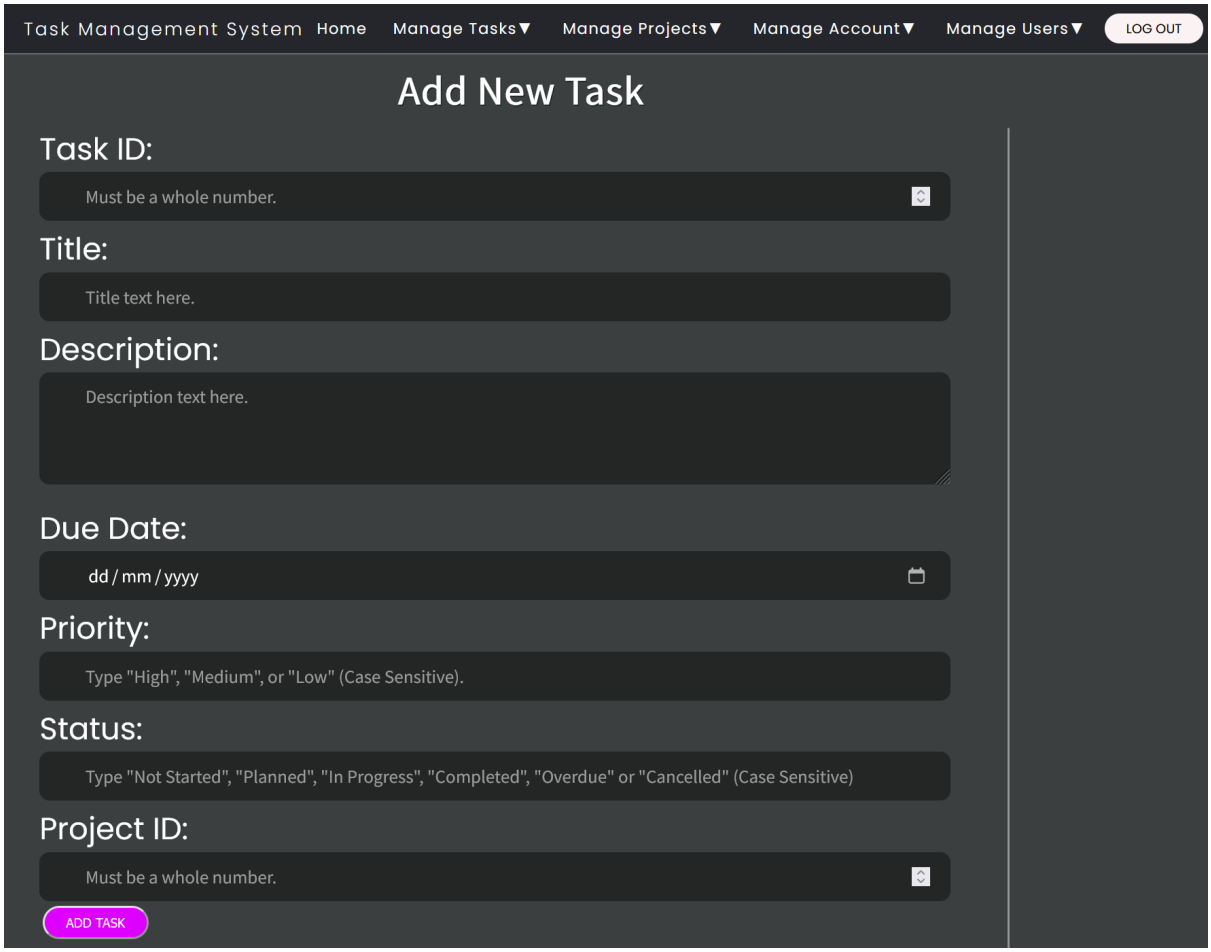
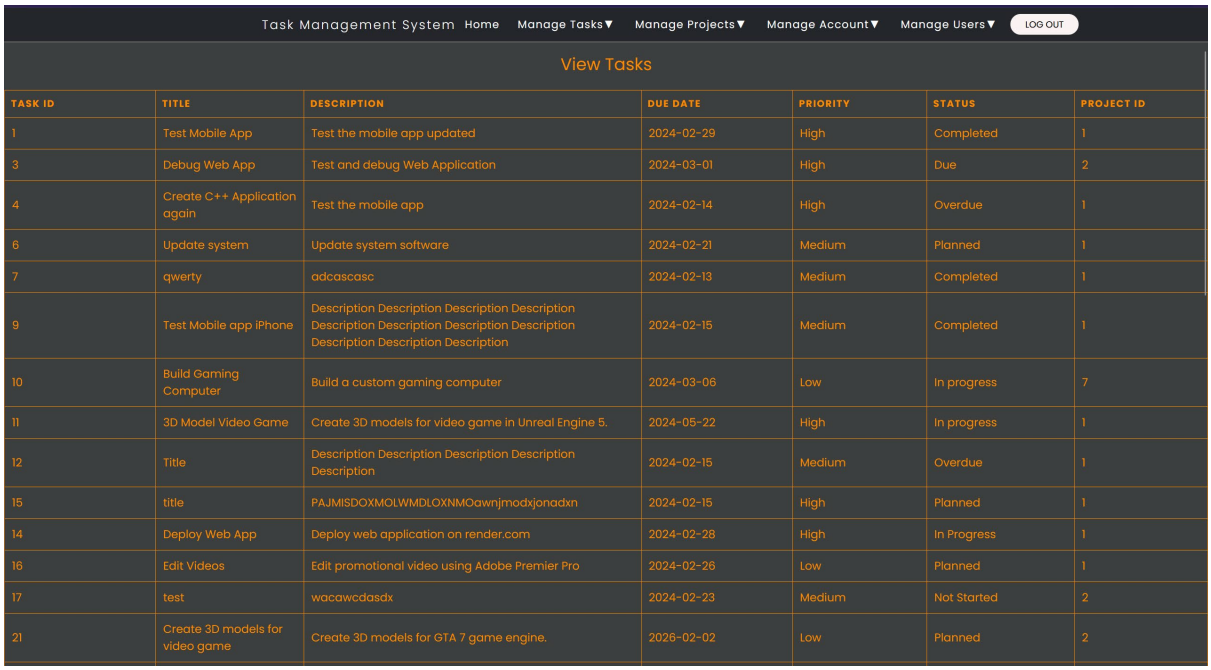


Figure 176 - Standard users and administrators can add tasks and projects.



TASK ID	TITLE	DESCRIPTION	DUE DATE	PRIORITY	STATUS	PROJECT ID
1	Test Mobile App	Test the mobile app updated	2024-02-29	High	Completed	1
3	Debug Web App	Test and debug Web Application	2024-03-01	High	Due	2
4	Create C++ Application again	Test the mobile app	2024-02-14	High	Overdue	1
6	Update system	Update system software	2024-02-21	Medium	Planned	1
7	qwerty	adcascasc	2024-02-13	Medium	Completed	1
9	Test Mobile app iPhone	Description Description Description Description Description Description Description Description Description Description Description	2024-02-15	Medium	Completed	1
10	Build Gaming Computer	Build a custom gaming computer	2024-03-06	Low	In progress	7
11	3D Model Video Game	Create 3D models for video game in Unreal Engine 5.	2024-05-22	High	In progress	1
12	Title	Description Description Description Description Description	2024-02-15	Medium	Overdue	1
15	title	PAJMISDOXMOLWMDLOXNMOawnjmodxjonadxn	2024-02-15	High	Planned	1
14	Deploy Web App	Deploy web application on render.com	2024-02-28	High	In Progress	1
16	Edit Videos	Edit promotional video using Adobe Premier Pro	2024-02-26	Low	Planned	1
17	test	wacawcdasdx	2024-02-23	Medium	Not Started	2
21	Create 3D models for video game	Create 3D models for GTA 7 game engine.	2026-02-02	Low	Planned	2

Figure 177 - Standard users and administrators can view tasks and projects.

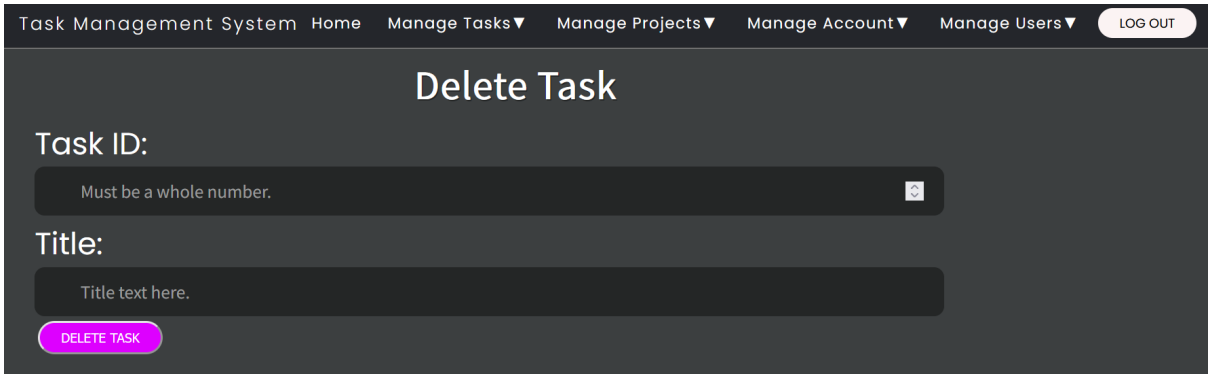


Figure 178 - Only administrators can delete tasks and projects.

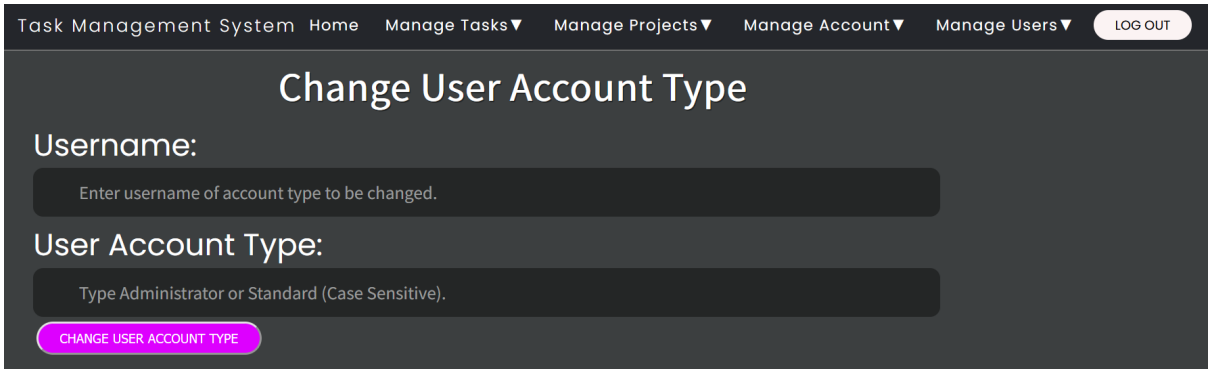
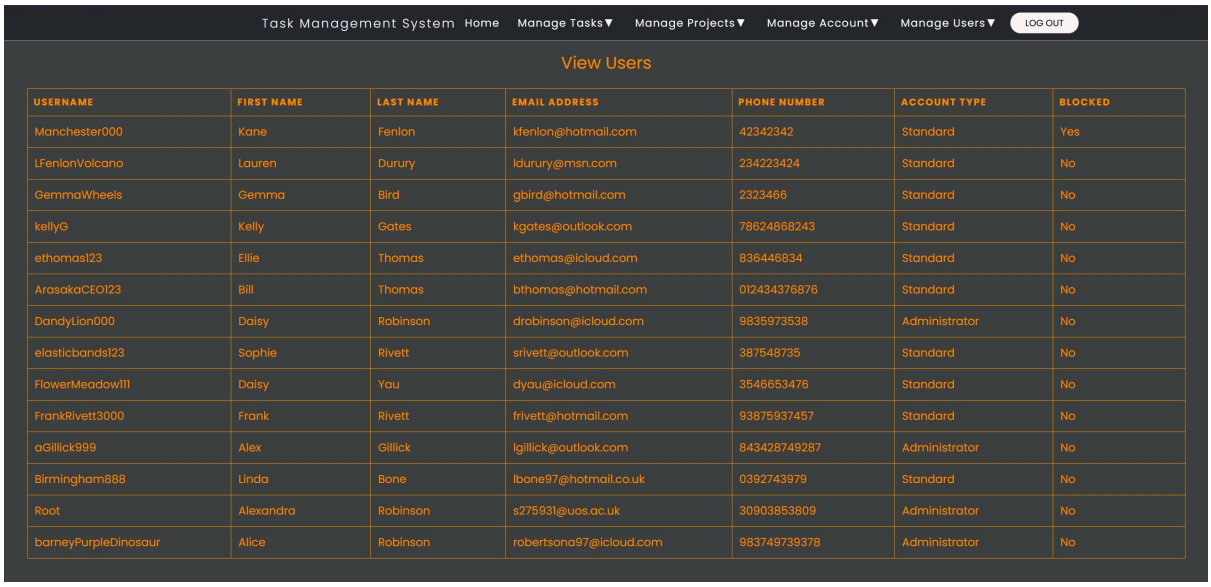


Figure 179 - Only administrators can manage other users.



USERNAME	FIRST NAME	LAST NAME	EMAIL ADDRESS	PHONE NUMBER	ACCOUNT TYPE	BLOCKED
Manchester000	Kane	Fenlon	kfenlon@hotmail.com	42342342	Standard	Yes
LFenlonVolcano	Lauren	Durury	ldurury@msn.com	234223424	Standard	No
GemmaWheels	Gemma	Bird	gbird@hotmail.com	2323466	Standard	No
kellyG	Kelly	Gates	kgates@outlook.com	78624868243	Standard	No
ethomas123	Ellie	Thomas	ethomas@icloud.com	836446834	Standard	No
ArasakaCEO123	Bill	Thomas	bthomas@hotmail.com	012434376876	Standard	No
DandyLion000	Daisy	Robinson	drobinson@icloud.com	9835973538	Administrator	No
elasticbands123	Sophie	Rivett	srivett@outlook.com	387548735	Standard	No
FlowerMeadow111	Daisy	Yau	dyau@icloud.com	3546653476	Standard	No
FrankRivett3000	Frank	Rivett	frivett@hotmail.com	93875937457	Standard	No
aGillick999	Alex	Gillick	lgillick@outlook.com	843428749287	Administrator	No
Birmingham888	Linda	Bone	lbone97@hotmail.co.uk	0392743979	Standard	No
Root	Alexandra	Robinson	s275931@uos.ac.uk	30903853809	Administrator	No
barneyPurpleDinosaur	Alice	Robinson	robertsona97@icloud.com	983749739378	Administrator	No

Figure 180 - Only administrators can view other users.

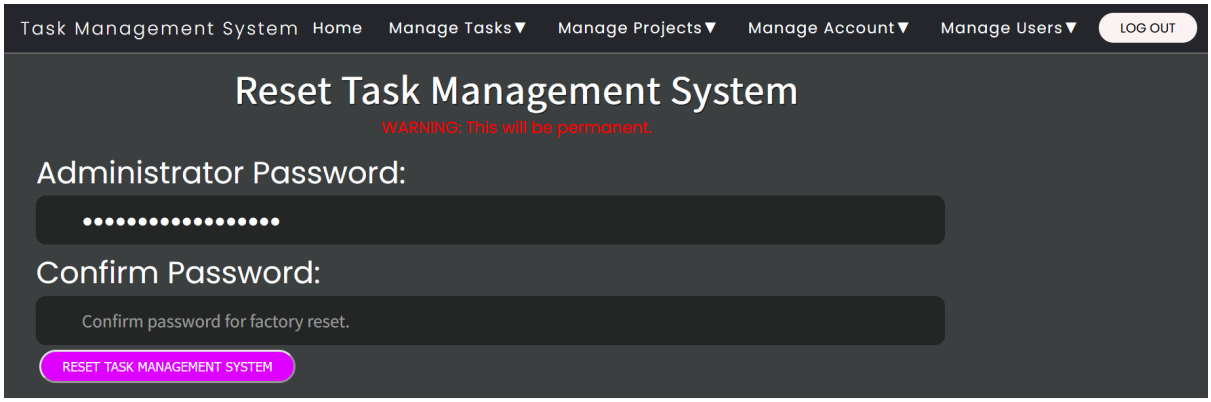


Figure 181 - Only administrators can perform administrative tasks.

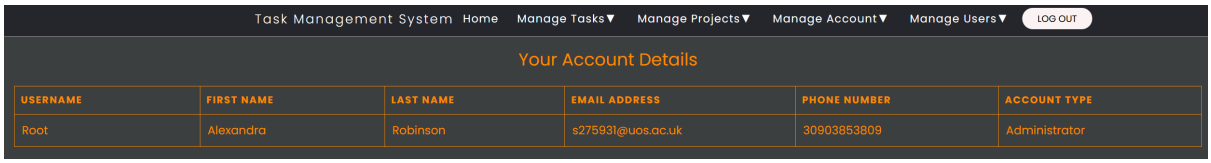


Figure 182 - Your Account Details page showing user's account type as Administrator.

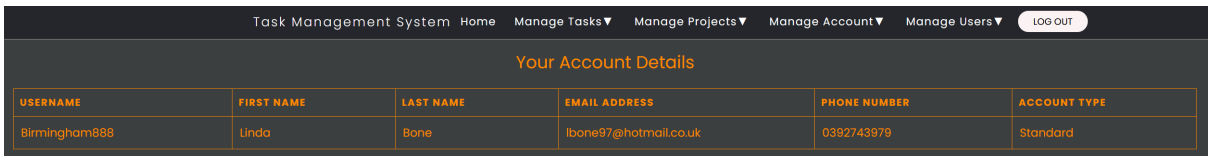


Figure 183 - Your Account Details page showing user's account type as Standard.

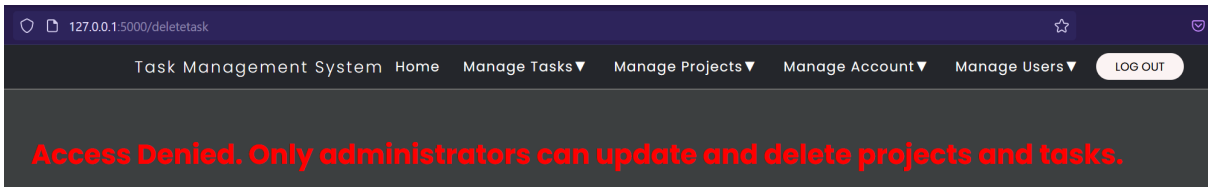


Figure 184 - Standard users cannot update and delete tasks and projects. They will receive an Access Denied error instead.

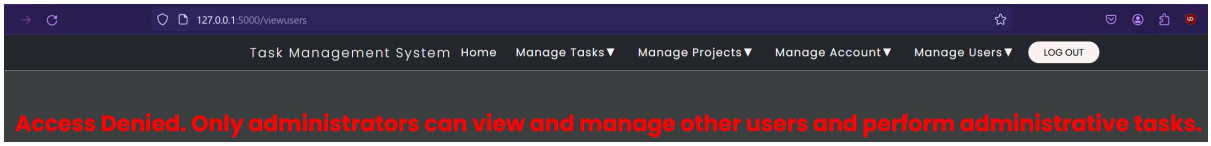


Figure 185 - Standard users cannot manage other users and perform administrative tasks. They will receive an Access Denied error instead.

7. User Blocking – Administrators can block users to prevent abuse if they misuse the Task Management System or if their account is compromised. Administrators can also unblock users too.

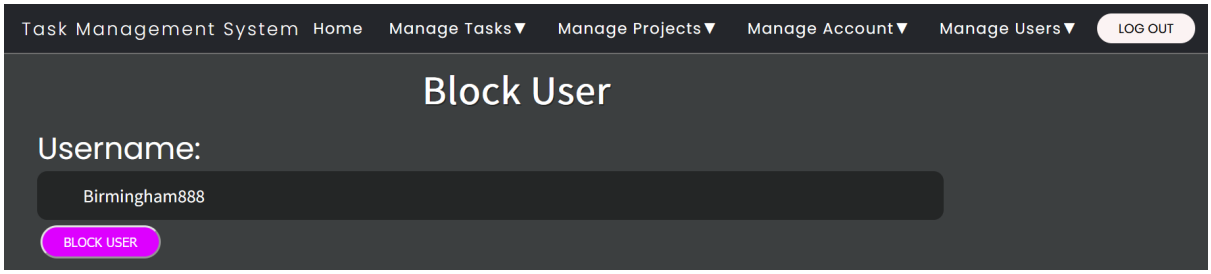
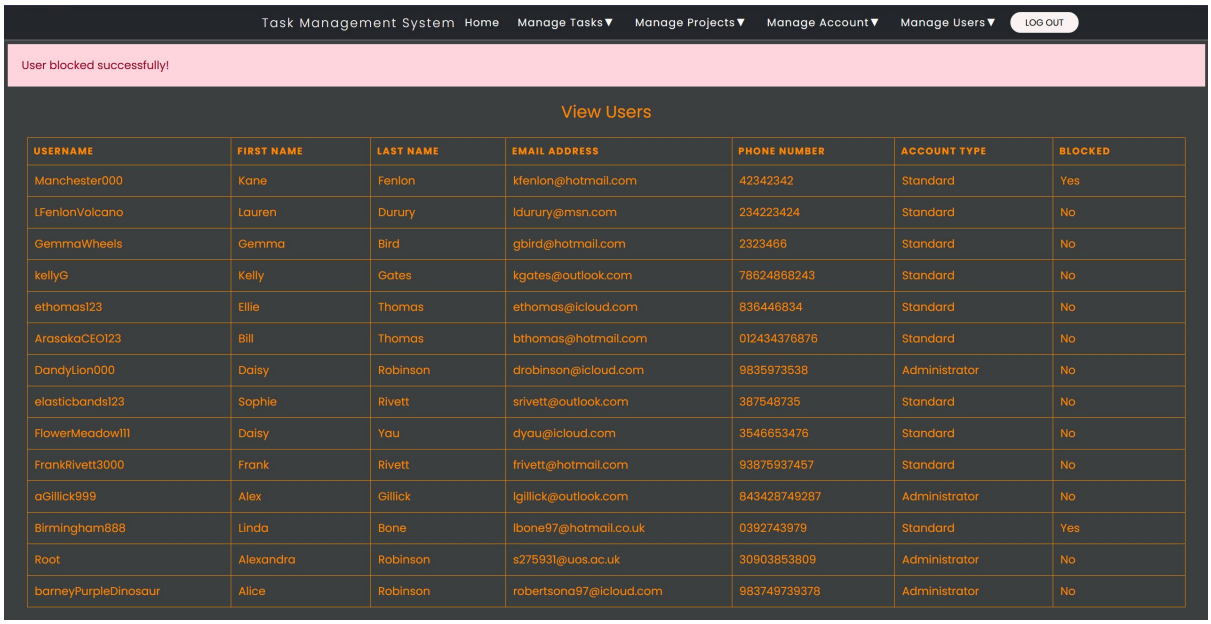


Figure 186 - Administrators can manually block other users.



USERNAME	FIRST NAME	LAST NAME	EMAIL ADDRESS	PHONE NUMBER	ACCOUNT TYPE	BLOCKED
Manchester000	Kane	Fenlon	kfenlon@hotmail.com	42342342	Standard	Yes
LFenlonVolcano	Lauren	Durury	ldurury@msn.com	234223424	Standard	No
GemmaWheels	Gemma	Bird	gbird@hotmail.com	2323466	Standard	No
kellyG	Kelly	Gates	kgates@outlook.com	78624868243	Standard	No
ethomas123	Ellie	Thomas	ethomas@icloud.com	836446834	Standard	No
ArasakaCEO123	Bill	Thomas	bthomas@hotmail.com	012434376876	Standard	No
DandyLion000	Daisy	Robinson	drobinson@icloud.com	9835973538	Administrator	No
elasticbands123	Sophie	Rivett	srivett@outlook.com	387548735	Standard	No
FlowerMeadow111	Daisy	Yau	dyau@icloud.com	3546653476	Standard	No
FrankRivett3000	Frank	Rivett	frivett@hotmail.com	93875937457	Standard	No
aGillick999	Alex	Gillick	lgillick@outlook.com	843428749287	Administrator	No
Birmingham888	Linda	Bone	lbone97@hotmail.co.uk	0392743979	Standard	Yes
Root	Alexandra	Robinson	s275931@uos.ac.uk	30903853809	Administrator	No
barneyPurpleDinosaur	Alice	Robinson	robertsona97@icloud.com	983749739378	Administrator	No

Figure 187 - User's blocked status has changed to Yes.

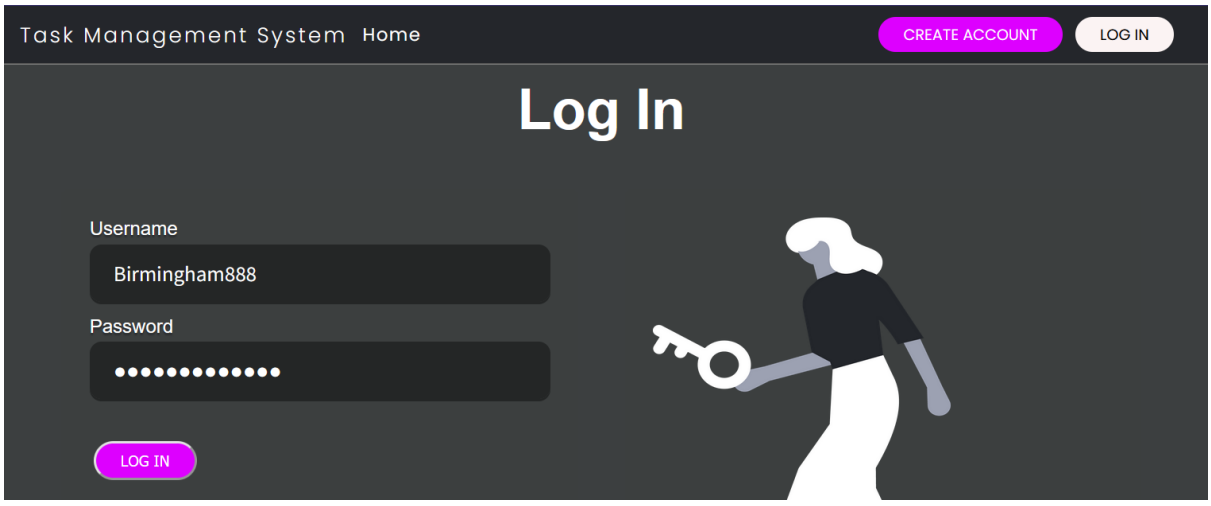


Figure 188 - Blocked user attempts to log in.

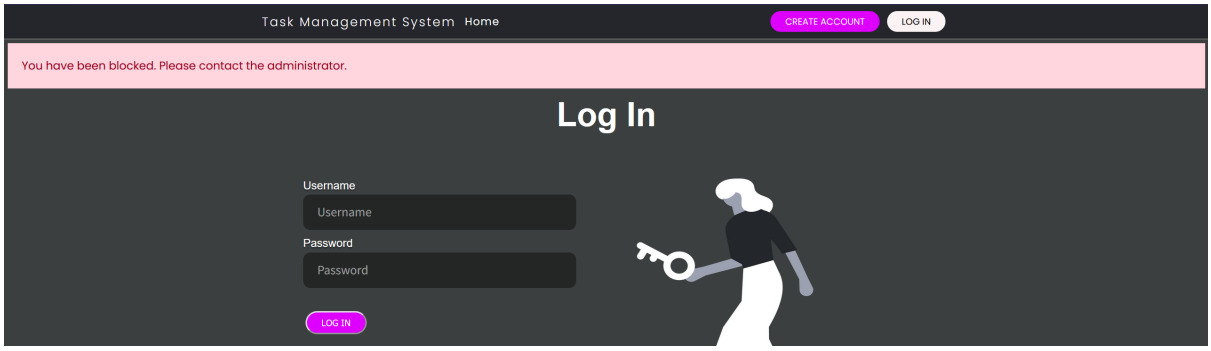


Figure 189 - Login attempt fails because the user is blocked.

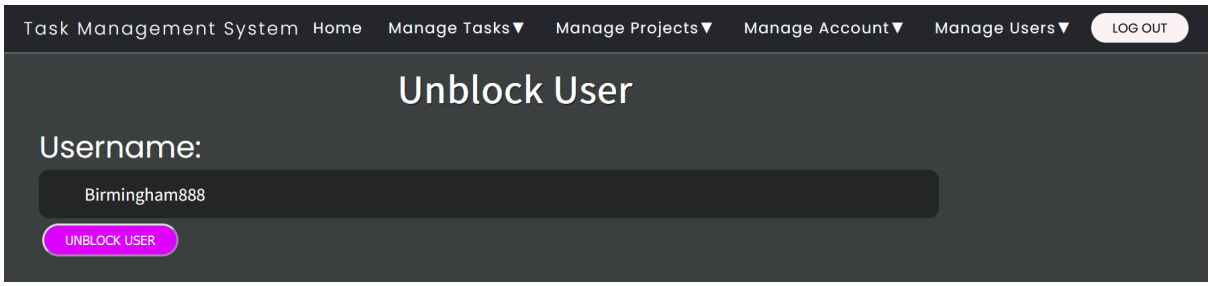
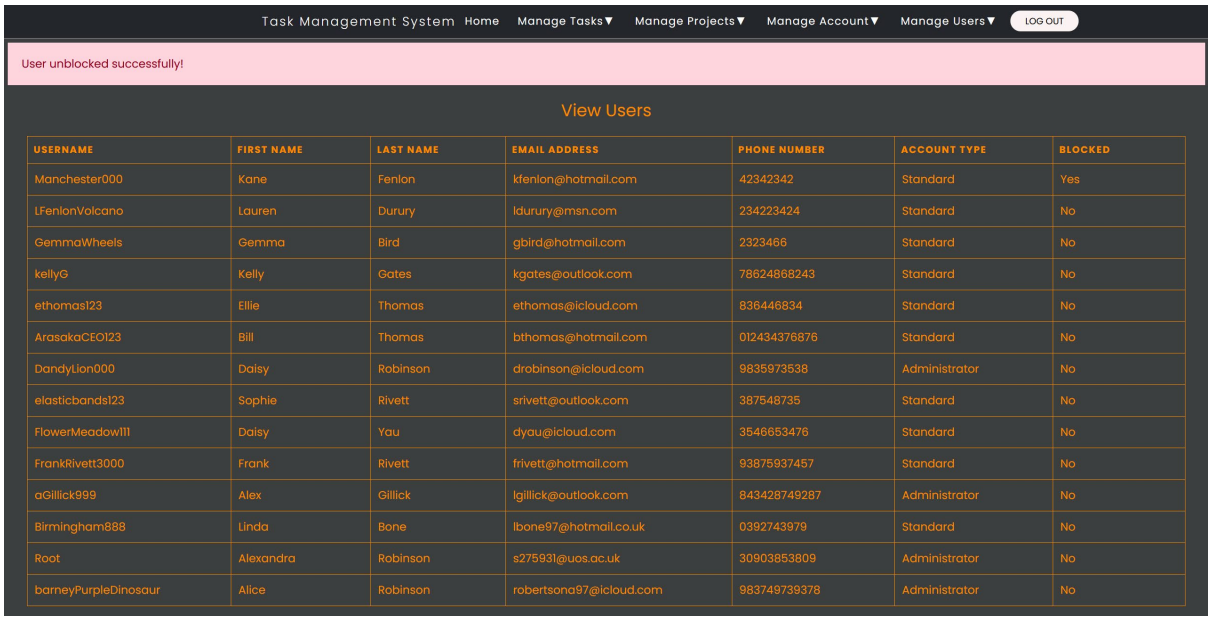


Figure 190 - Administrators can unblock blocked users.



USERNAME	FIRST NAME	LAST NAME	EMAIL ADDRESS	PHONE NUMBER	ACCOUNT TYPE	BLOCKED
Manchester000	Kane	Fenlon	kfenlon@hotmail.com	42342342	Standard	Yes
LFenlonVolcano	Lauren	Durury	ldurury@msn.com	234223424	Standard	No
GemmaWheels	Gemma	Bird	gbird@hotmail.com	2323466	Standard	No
kellyG	Kelly	Gates	kgates@outlook.com	78624868243	Standard	No
ethomas123	Ellie	Thomas	ethomas@icloud.com	836446834	Standard	No
ArasakaCEO123	Bill	Thomas	bthomas@hotmail.com	012434376876	Standard	No
DandyLion000	Daisy	Robinson	drobinson@icloud.com	9835973538	Administrator	No
elasticbands123	Sophie	Rivett	srivett@outlook.com	387548735	Standard	No
FlowerMeadow111	Daisy	Yau	dyau@icloud.com	3546653476	Standard	No
FrankRivett3000	Frank	Rivett	frivett@hotmail.com	93875937457	Standard	No
aGillick999	Alex	Gillick	lgillick@outlook.com	843428749287	Administrator	No
Birmingham888	Linda	Bone	lbone97@hotmail.co.uk	0392743979	Standard	No
Root	Alexandra	Robinson	s275931@uos.ac.uk	30903853809	Administrator	No
barneyPurpleDinosaur	Alice	Robinson	robertsona97@icloud.com	983749739378	Administrator	No

Figure 191 - User's blocked status has changed to No.

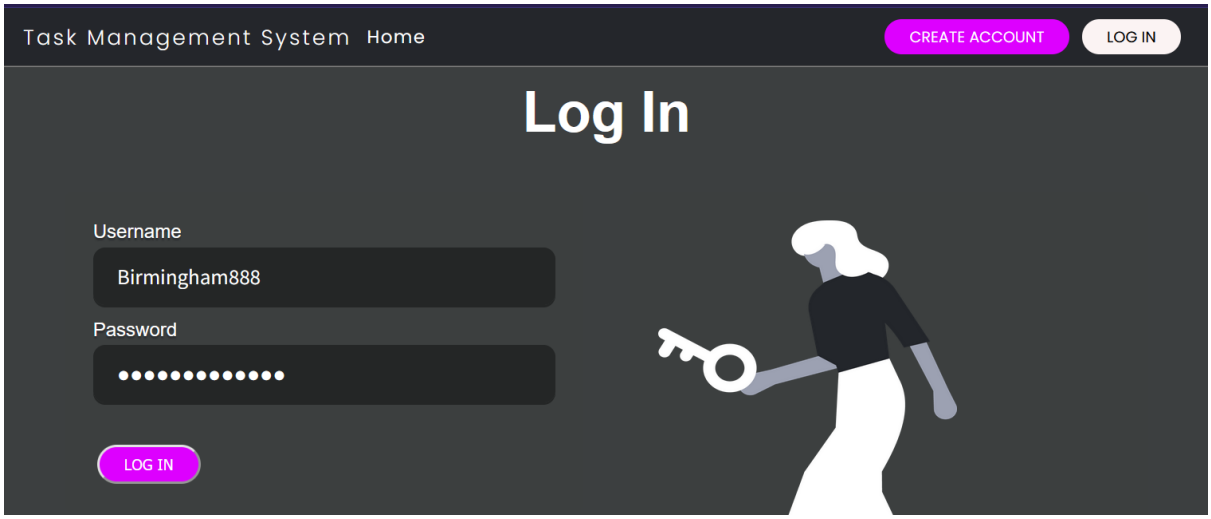


Figure 192 - Unblocked user attempts to log in.

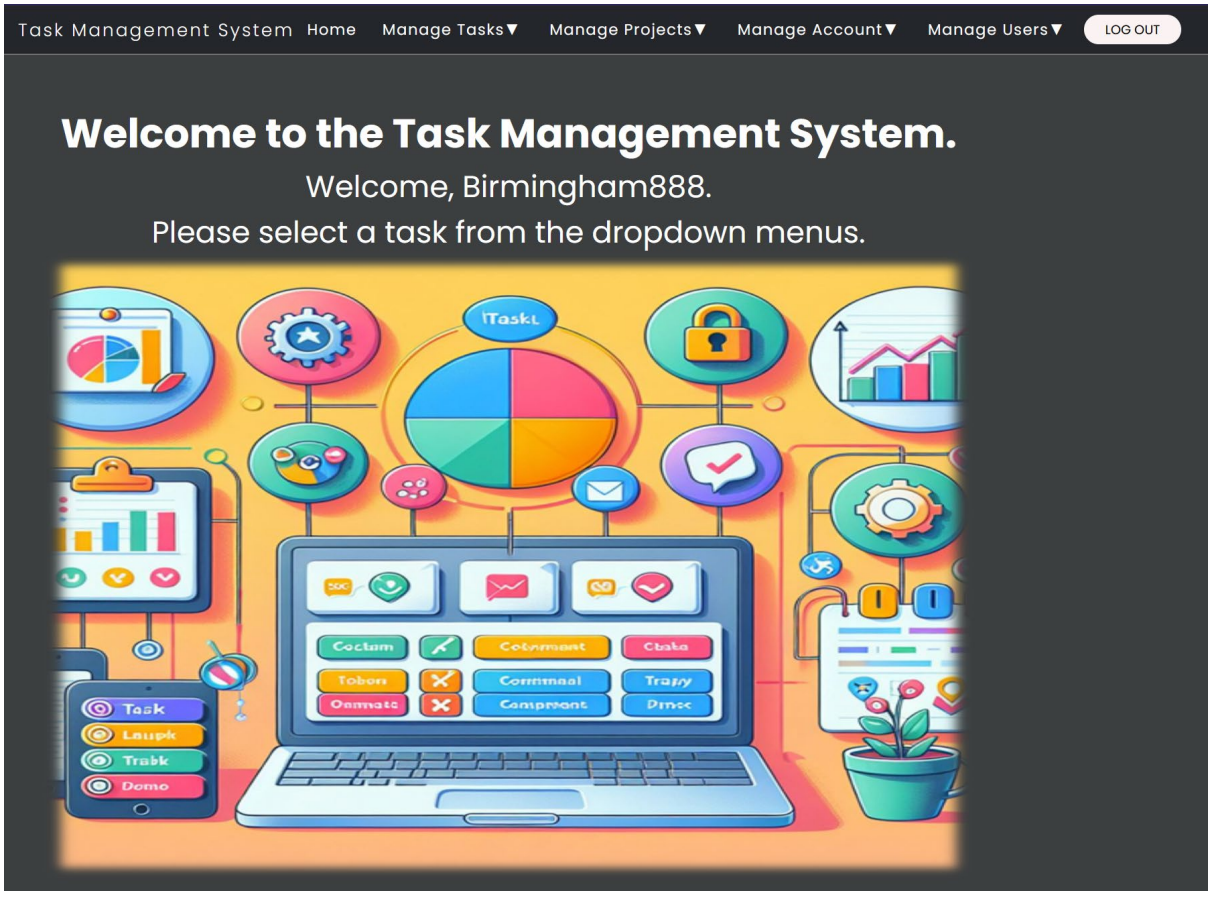


Figure 193 - Unblocked user can now successfully log in.

8. Despite having security measures in place, Docker discovered vulnerabilities in my code when it analysed the container image. It discovered vulnerabilities with the Alpine, pip, wheel, and setup tools packages, which are required to run the Task Management System containers.

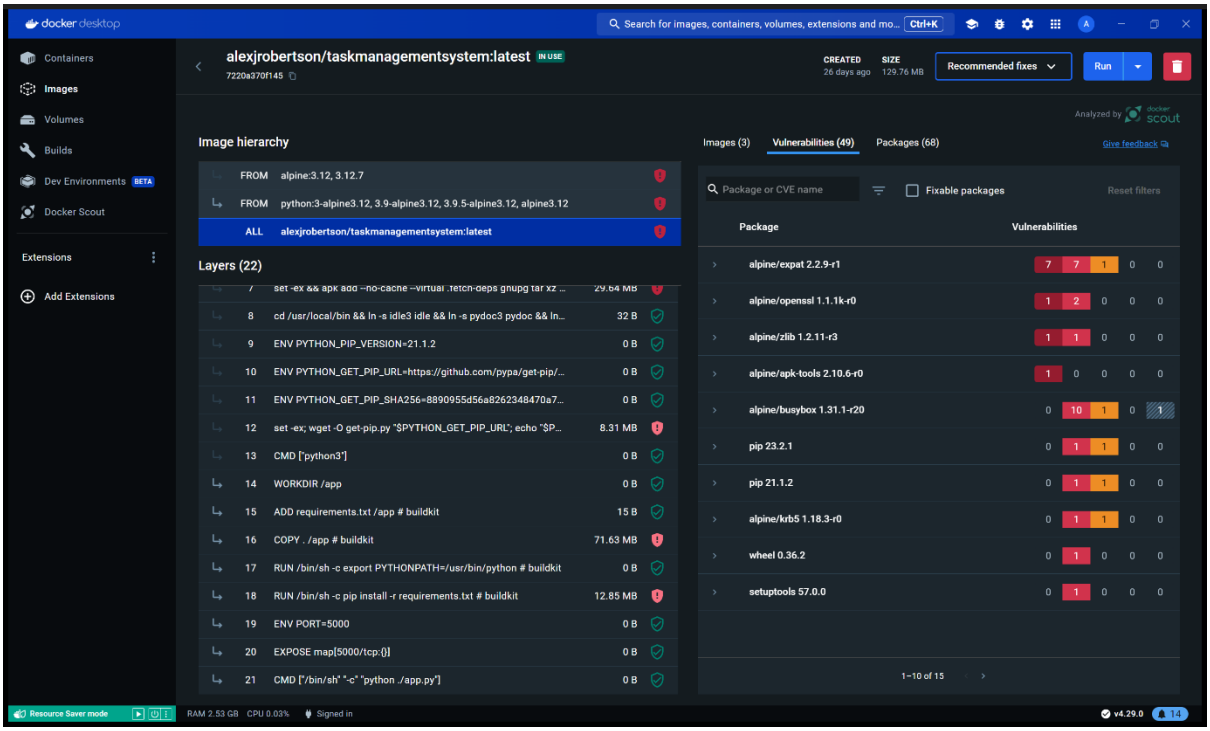


Figure 194 - Docker has detected vulnerabilities with the Docker container deployment of the Task Management System to do with Alpine Linux and pip. This is a security threat.

9. SonarCloud also discovered vulnerabilities with the Task Management System's source code during testing. These vulnerabilities included the secret key for flash messages, risk of Cross-Site Request Forgery, considers the random number generation for sessionIDs as weak cryptography however I use strong SHA3-512-bit cryptography to encrypt the passwords, COPY . /app in the Dockerfile can add sensitive data to the container and insecure configuration for the port.

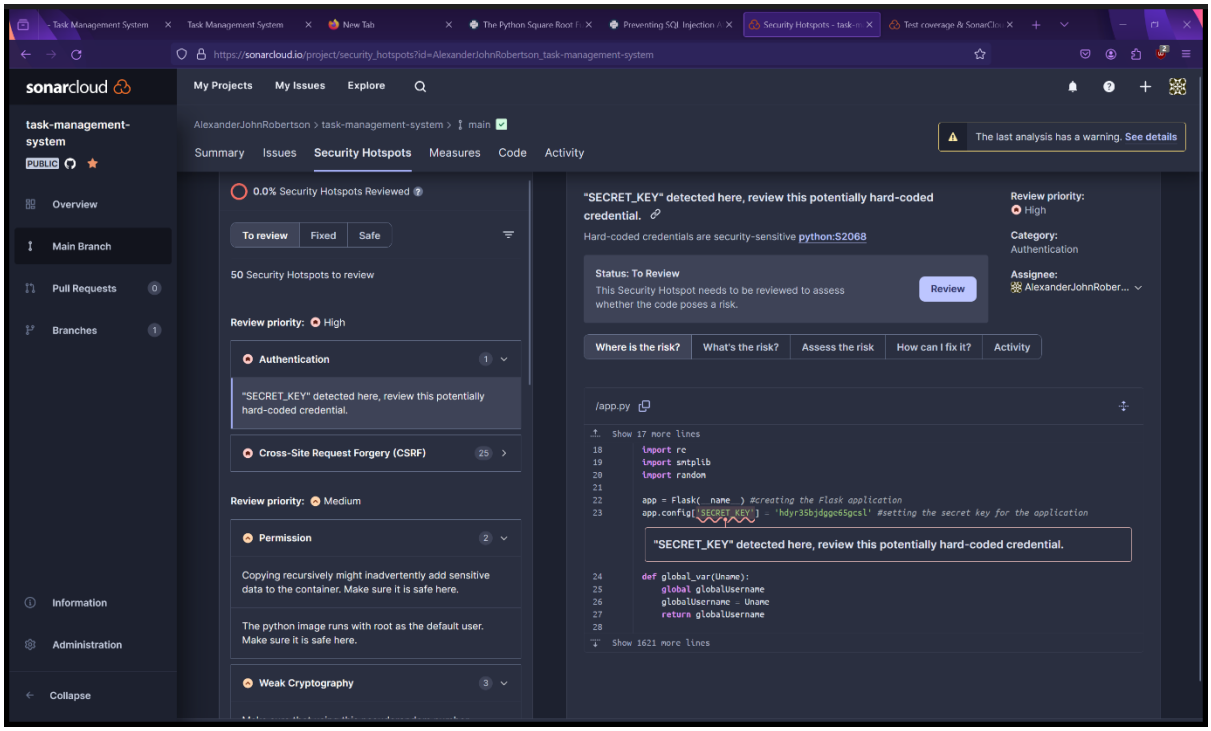


Figure 195 - SonarCloud has detected a security vulnerability with the secret key for the flash messages as the credential is hard-coded.

Disaster Recovery

To reduce the risk and impact of disasters affecting the Task Management System and its development, I have implemented the following measures:

1. Version control using GitHub
2. Backups – The working directory on the computer for development is a OneDrive directory
3. Multiple Deployment platforms – I am using Render.com and Docker for deployment if one of them goes down.

Availability Design

For the online deployment on Render.com, I am using the $7 per month subscription for the Task Management System's instance type as paid instances on Render.com have zero downtime, are scalable, supports SSH and persistent disks. Render.com is known for its reliability.

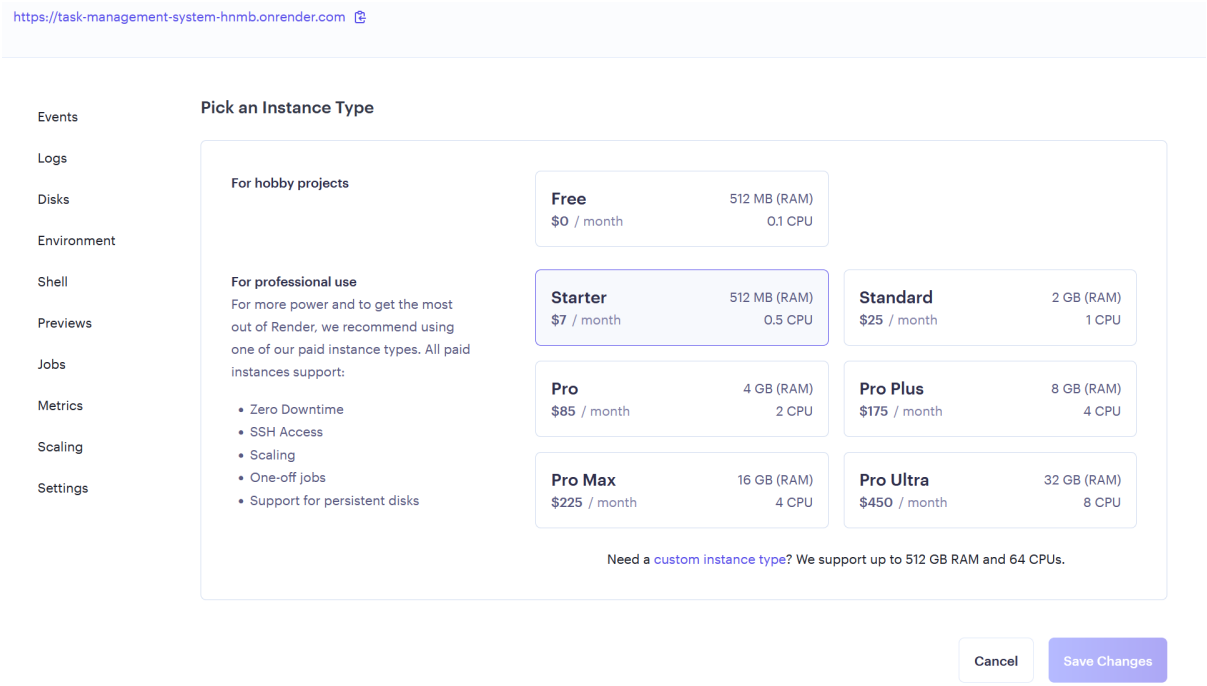


Figure 196 - The $7 per month subscription is the cheapest Render.com deployment that has zero downtime as the free version will spin down due to inactivity and is very slow to load the web app when this happens.

Maintenance

Technical Debt

Technical debt (where I found a quick fix for a bug/issue but might cause bigger problems later) occurred in this project in several places:

- Didn't create a session ID cookie and database column, only a username cookie for authentication. This led to unauthorised access by copying the username from the cookie before logging out, copying it back into the cookie and overriding the authentication system. This was later fixed.
- Monolithic application – A monolithic application is quicker and easier for me to start building than a microservice application but will be more difficult to scale and maintain in the future.
- Automated email bug – no error handling if email domain to send automated emails during setup and changing passwords if it gets blocked for triggering a spam filter.
- Not using a constant variable for the database file as it is repeated many times (detected by SonarCloud) throughout the application, will be problematic if renaming the database file or using another database file.
- Not using Object-Oriented Programming in the main application.
- Using Antipatterns such as Spaghetti code HTML and CSS and some Python, Big Ball of Mud, Golden Hammer.
- Inexperience with CSS compared to Python and HTML
- Lack of comments in some code.
- Design and Code smells

Inconsistent Coding Style

SonarCloud has detected inconsistencies with the code during automated system testing including several consistency issues with the HTML code tags.

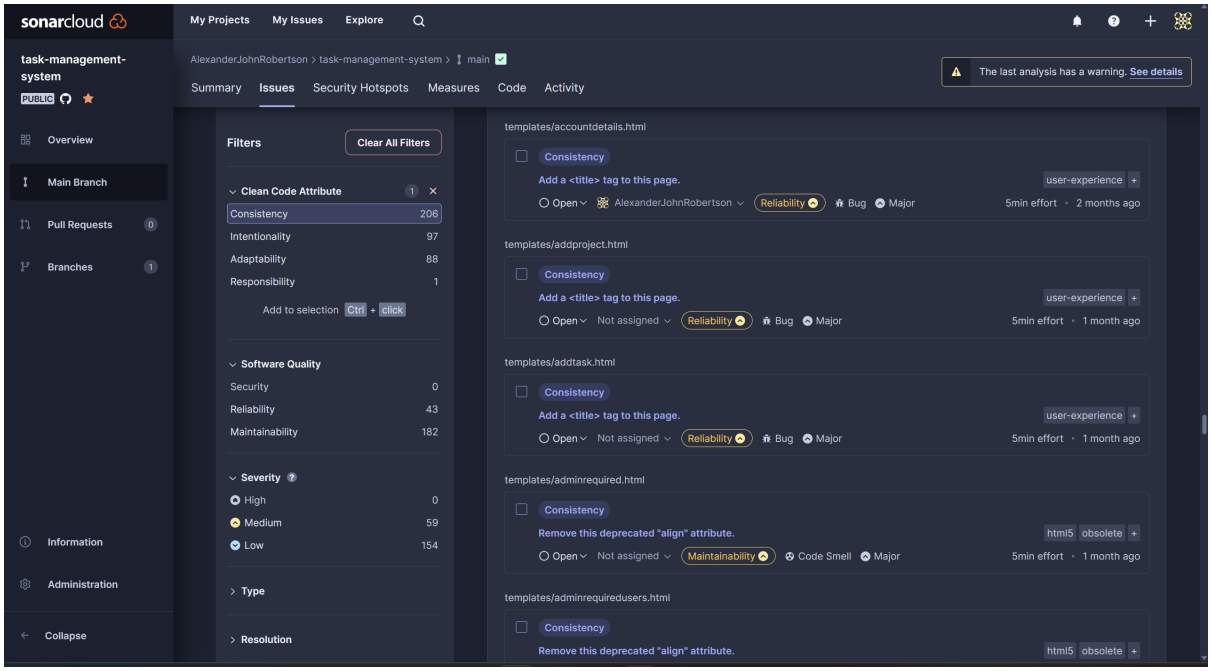


Figure 197 - SonarCloud has detected many coding inconsistencies mainly with the frontend HTML code.

Design Smells

During development the Task Management System encountered some major design smells that have since been fixed:

- The original navbar was unresponsive when I tested it on a mobile device and resized the browser on a desktop. This was rectified by using a responsive navbar of a similar design from CodePen and heavily modified to suit the application.
- The forms and tables were also unresponsive and only worked in a desktop browser. This was also fixed by using responsive form and table templates from CodePen and modifying them.
- No interface to manage other users, perform first-time setup and reset the application. An interface was developed to fix this.
- Poor security and authentication – No fixed by using SHA3-512 encryption for passwords, random SessionID cookies and database entries, user blocking, automatic blocking after 3 incorrect password attempts.
- Unattractive landing and home pages – Added an AI generated image related to task management to them.
- Several code smells were detected by SonarCloud during automated system testing.

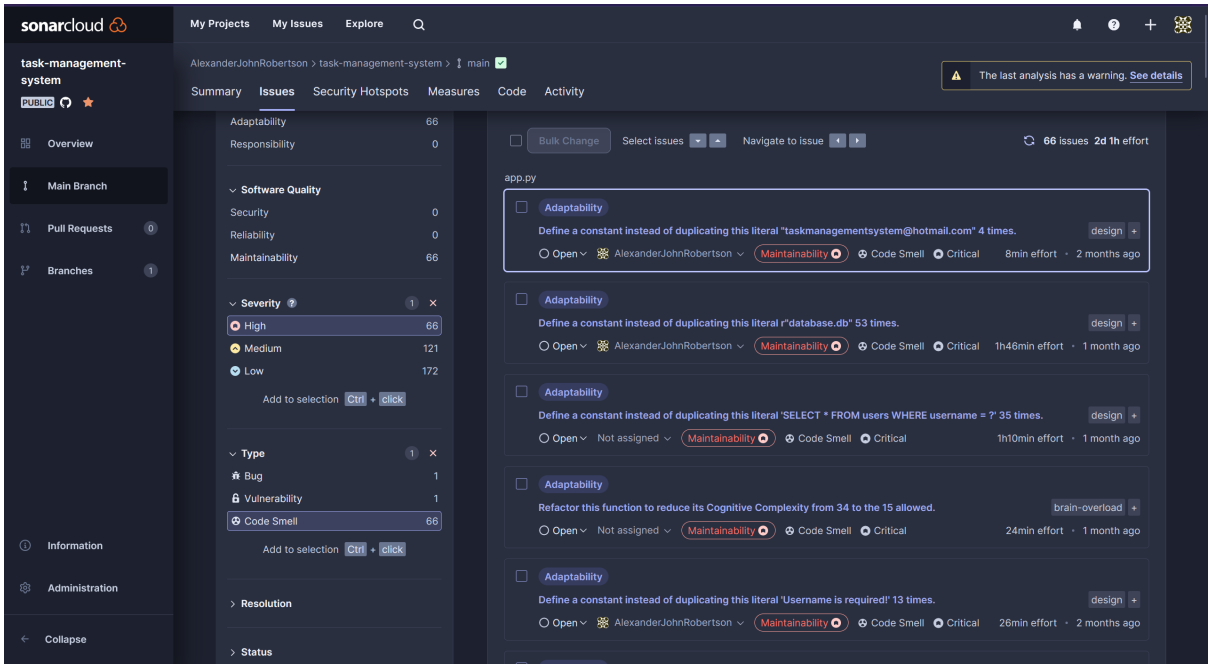


Figure 198 - Code smells have been detected by SonarCloud.

Outdated Documentation

The Task Management System is likely to be updated as technology advances, requirements change, license type changes, and new features are developed. Due to limitations, I will prioritise making the new versions of the software work before updating the documentation. Third-party software documentation may also be outdated however I have no control over this.

Refactoring Opportunities

Refactoring Opportunities include:

- Migrating to microservice application architecture.
- Fixing up code smells and technical dets
- Fixing up software antipatterns.
- Improving maintainability.
- Clearer instructions in documentation.
- Further improvements to responsiveness and UI/UX
- Use variable for database file
- Switch to OOP

Ethics in Software Engineering

I have taken ethics into consideration when developing the Task Management System including adherence to the eight principles of software ethics. These included:

1. Public – Accepting full responsibility for my work and only releasing it if it is safe, meets requirements and passes tests in the testing phase.
2. Client and Employer – not using counterfeit, pirated or illegal software, data protection compliance.

3. Product – Produce a high-quality product for the client, following specifications and requirements for the product.
4. Judgement – Must maintain integrity and independence in my internal judgement such as not trying to sell the Task Management System through deceptive means.
5. Management – Manage development and maintenance of the software ethically.
6. Profession – I have promoted public knowledge of my software engineering profession through user acceptance testing, with classmates and submitting it to the examiner as this is a university assignment.
7. Colleagues – This is an individual assignment therefore this principle didn't apply here.
8. Self – I have a lifelong learning responsibility to adapt to advances in technology and requirement changes in software engineering.

Additionally, my software does not contain any malicious code and it is against the Terms of Use to insert any malicious code. I have tried to comply with data protection with password encryption and user authentication. (IEEE-CS/ACM joint task force on Software Engineering Ethics and Professional Practices (SEEPP), 1999)

Lessons learnt in doing the technical solution implementing SE practices

Lessons I have learnt from this project include:

- Improving my coding skills with Python, SQL, HTML, CSS, and JavaScript.
- Learn how to use Docker containerisation and write DockerFiles
- Learn about advanced software engineering practises, principles, and methodologies.
- Implement CI/CD Pipelines using GitHub, Render.com and SonarCloud.
- Use UML to draw use case, class, and ERD diagrams.
- Design mock-ups and wireframes of user interfaces.
- Learn how to make responsive cross-platform software and UIs
- Learn how to use the Flask framework and APIs for web development using Python.
- Learn how to test software:
- Awareness of software antipatterns such as:
- Maintainability issues such as:
- How to write documentations and licensing.

Improvements and future plans:

- Use microservice architecture
- Use object-oriented programming
- Better comments
- Improved code consistency
- Get rid of the technical debts, code and design smells listed in the Maintenance section of this report.
- Clean up antipatterns such as spaghetti code and big ball of mud.
- Further improvements to UI/UX
- Clearer documentation

References

Acosta, D. (2023) *Render.*
Available at: https://stackshare.io/render
(Accessed: 04 05 2024).

Altynpara, E. (2023) *Popular Software Development Methodologies Comparison: Full Overview.*
Available at: https://www.cleveroad.com/blog/software-development-methodologies/
(Accessed 04 05 2024).

AspiringYouths (2024) *Advantages and Disadvantages of Docker Containers.*
Available at: https://aspiringyouths.com/advantages-disadvantages/docker-containers/
(Accessed 04 05 2024).

Baeldung (2021) *Layered Architecture.*
Available at: https://www.baeldung.com/cs/layered-architecture
(Accessed: 04 05 2024).

Bigelow, S. J. (2022) *Carefully weigh these DevOps pros and cons.*
Available at: https://www.techtarget.com/searchitoperations/tip/Carefully-weigh-these-DevOps-pros-and-cons
(Accessed: 04 05 2024).

DuploCloud (2023) *Docker Advantages and Disadvantages: What You Need to Know Before You Switch.*
Available at: https://duplocloud.com/blog/docker-advantages-and-disadvantages/
(Accessed: 04 05 2024).

Geeks for Geeks (2024) *Client-Server Model.*
Available at: https://www.geeksforgeeks.org/client-server-model/
(Accessed: 04 05 2024).

Geeks for Geeks (2024) *Incremental Process Model – Software Engineering.*
Available at: https://www.geeksforgeeks.org/software-engineering-incremental-process-model/
(Accessed: 04 05 2024).

Geeks for Geeks (2024) *Rapid application development model (RAD) – Software Engineering.*
Available at: https://www.geeksforgeeks.org/software-engineering-rapid-application-development-model-rad/
(Accessed: 04 05 2024).

Geeks for Geeks, 2024. *Spiral Model – Software Engineering.*
Available at: https://www.geeksforgeeks.org/software-engineering-spiral-model/
(Accessed: 04 05 2024).

Hall, T. (2024) *DevOps vs. Agile.*
Available at: https://www.atlassian.com/devops/what-is-devops/agile-vs-devops
(Accessed: 04 05 2024).

Harris, C., (2024) *Microservices vs. monolithic architecture.*
Available at: https://www.atlassian.com/microservices/microservices-architecture/microservices-vs-monolith
(Accessed: 04 05 2024).

IEEE-CS/ACM joint task force on Software Engineering Ethics and Professional Practices (SEEPP) (1999) *Code of Ethics.*
Available at: https://www.computer.org/education/code-of-ethics
(Accessed: 04 05 2024).

Lewis, B. L. &. S. L. (2022) *waterfall model.*
Available at: https://www.techtarget.com/searchsoftwarequality/definition/waterfall-model
(Accessed: 04 05 2024).

Lucidchart (2024) *UML Use Case Diagram Tutorial.*
Available at: https://www.lucidchart.com/pages/uml-use-case-diagram
(Accessed: 04 05 2024).

Oppermann, A. (2023) *What Is the V-Model in Software Development?.*
Available at: https://builtin.com/software-engineering-perspectives/v-model
(Accessed: 04 05 2024).

OS System (2020) *Top 4 Software Development Methodologies: Comparison, Differences, Pros and Cons.*
Available at: https://os-system.com/blog/top-software-development-methodologies-comparison-differences-pros-and-cons/
(Accessed: 04 05 2024).

Scrum.org (2024) *What is Scrum?.*
Available at: https://www.scrum.org/learning-series/what-is-scrum/
(Accessed: 04 05 2024).

Team Kissflow (2024) *What is Rapid Application Development (RAD)? An Ultimate Guide for 2024.*
Available at: https://kissflow.com/application-development/rad/rapid-application-development/
(Accessed: 04 05 2024).

Vasiliauskas, V. (2023) *14 Scrum Advantages and Disadvantages in 2024.*
Available at: https://teamhood.com/agile/scrum-advantages-disadvantages/
(Accessed: 04 05 2024).

Visual Paradigm (2024) *What is Class Diagram?.*
Available at: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/
(Accessed: 04 05 2024).

Visual Paradigm, 2024. *What is Entity Relationship Diagram (ERD)?.* [Online]
Available at: https://www.visual-paradigm.com/guide/data-modeling/what-is-entity-relationship-diagram/
(Accessed: 04 05 2024).

Yasar, K. (2024) *software engineering.*
Available at: https://www.techtarget.com/whatis/definition/software-engineering
(Accessed: 04 05 2024).