

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.....	2
Definition of Software Engineering	3
Purpose of the Task Management System	3
What are you trying to solve with the app?.....	4
How are you solving it?.....	4
Design Choices and Implementation.....	4
Requirements Re-engineering.....	4
Design Diagrams	4
UML Use Case Diagram.....	4
UML Class Diagram	6
Wireframe Diagram.....	8
Mock-up of Application.....	21
User/System/functional/Non-Functional Requirements.....	36
Language and associated Paradigm/Database-For any identified choice, justify why you selected that language	38
Database Designs.....	47
Software Process Model Selected to come up with solution-List a few and justify your choice	50
High Level Design Choices.....	54
Layered Architecture	54
Client-Server Architecture	54
Monolithic Architecture	55
Microservice Architecture.....	55
Object Oriented Programming	55
Architectural Trade-offs.....	56
Software Antipatterns	56
Big Ball of Mud.....	56
Spaghetti Code.....	57
Golden Hammer.....	61
Continuous integration and continuous delivery/deployment aspects.....	62
Testing- Unit/Integration and system testing/ Acceptance testing	69
Unit Testing	69
Integration and System Testing	75

Acceptance Testing	80
Deployment	84
Release.....	97
Documentation	98
Security and reliability of the system	101
Disaster Recovery.....	119
Availability Design.....	119
Maintenance.....	120
Technical Debt.....	120
Inconsistent Coding Style.....	121
Design Smells	121
Outdated Documentation	122
Refactoring Opportunities.....	122
Ethics in Software Engineering.....	122
Lessons learnt in doing the technical solution implementing SE practices	123
References	124

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	https://github.com/AlexanderJohnRobertson/task-management-system-clean-install/wiki/Task-Management-System-(Clean-Install)-Documentation
Task Management System Documentation	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 render.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)

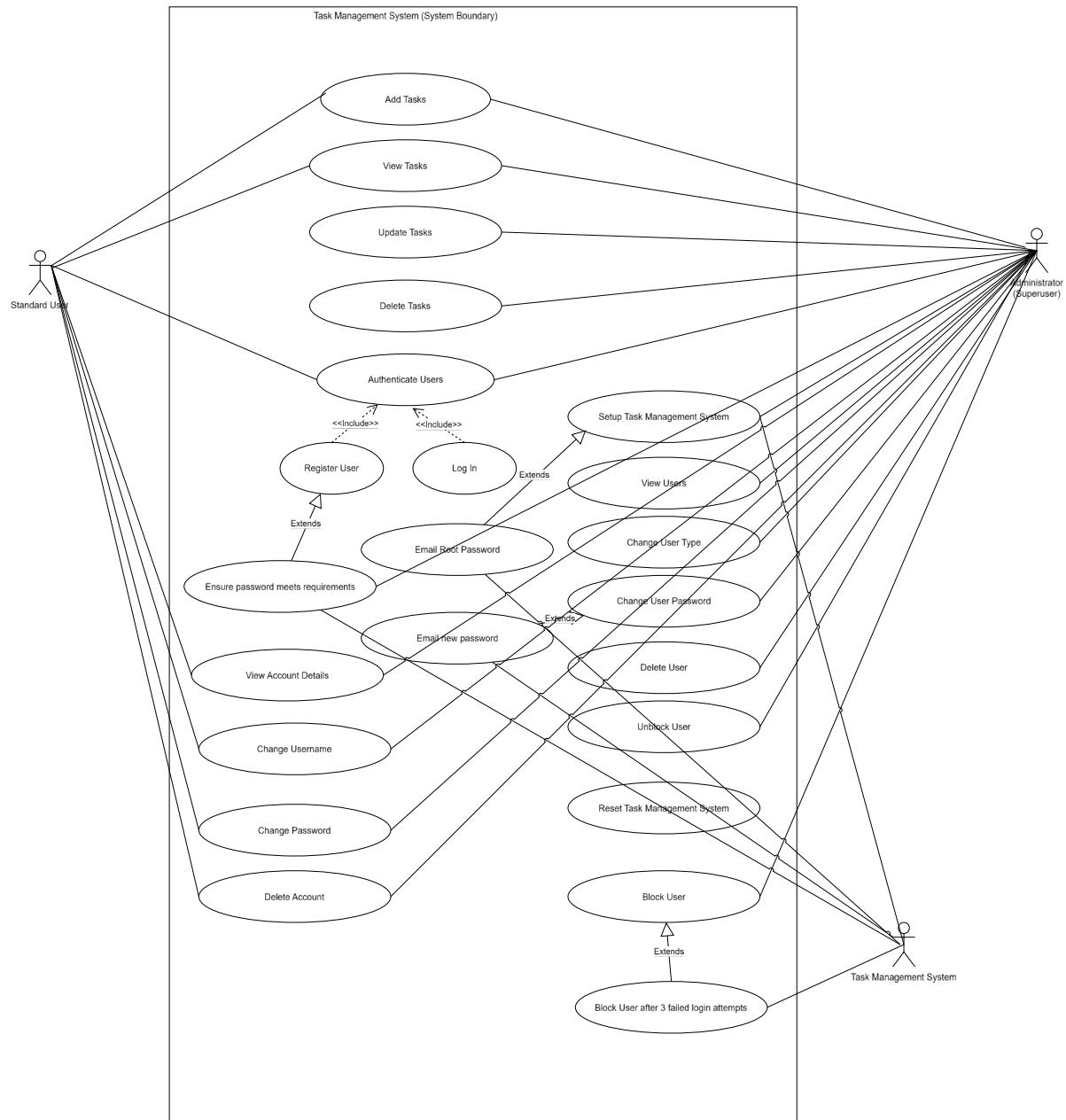


Figure 1 - Task Management System UML Use Case Diagram

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

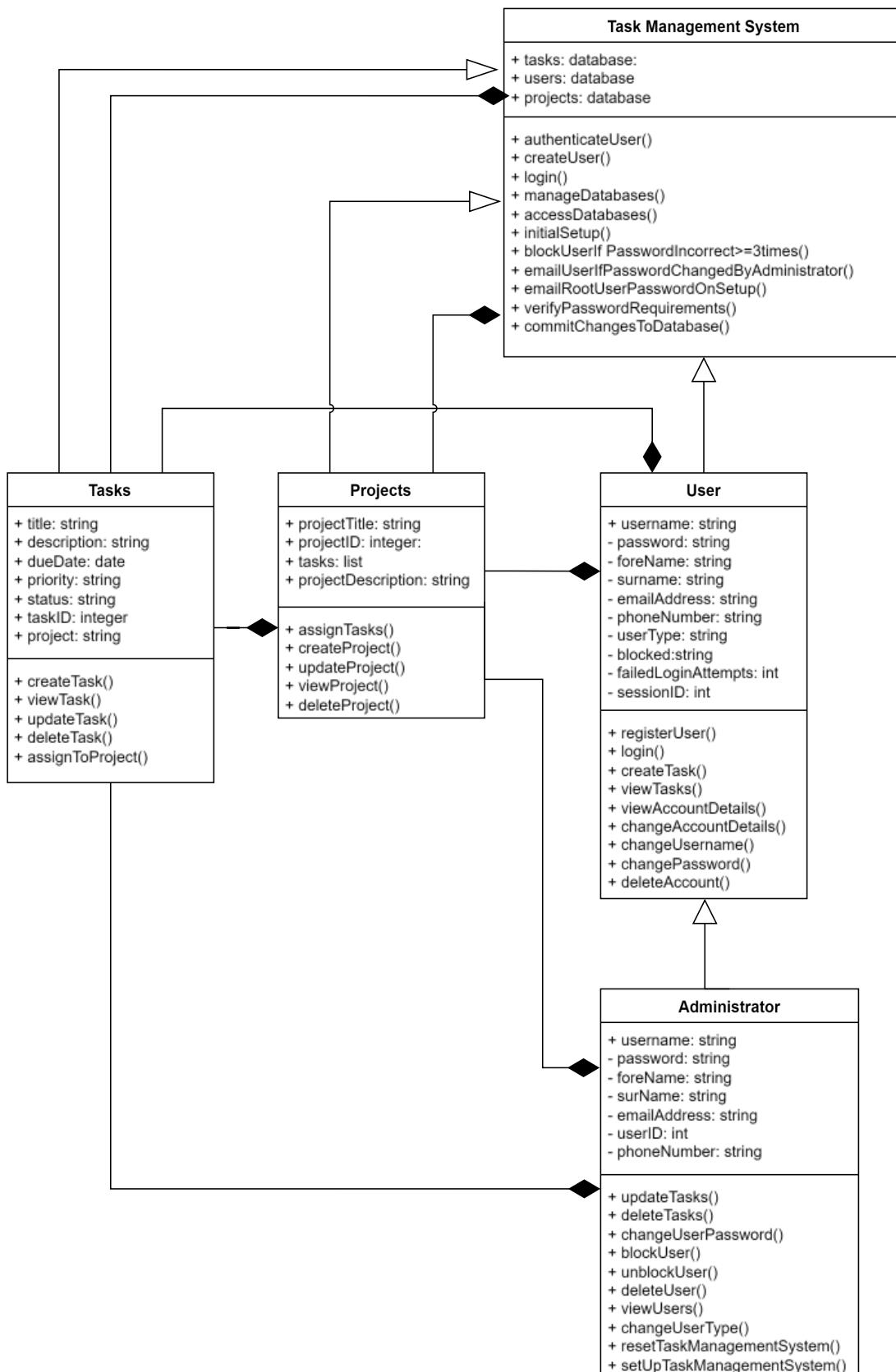


Figure 2 - Task Management System Class Diagram

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 for 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>

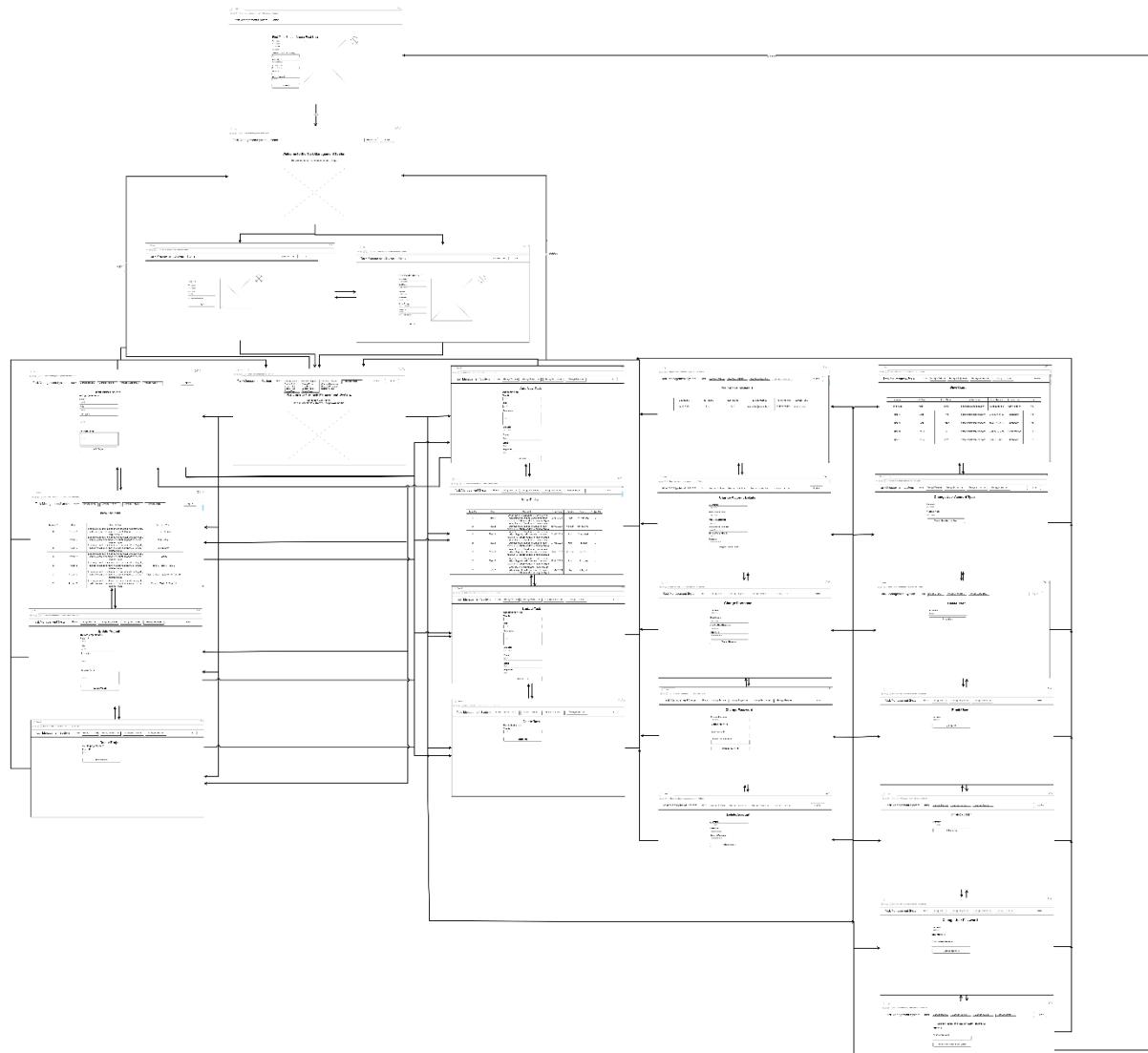


Figure 3 - Overall Task Management System Wireframe Diagram

S275931

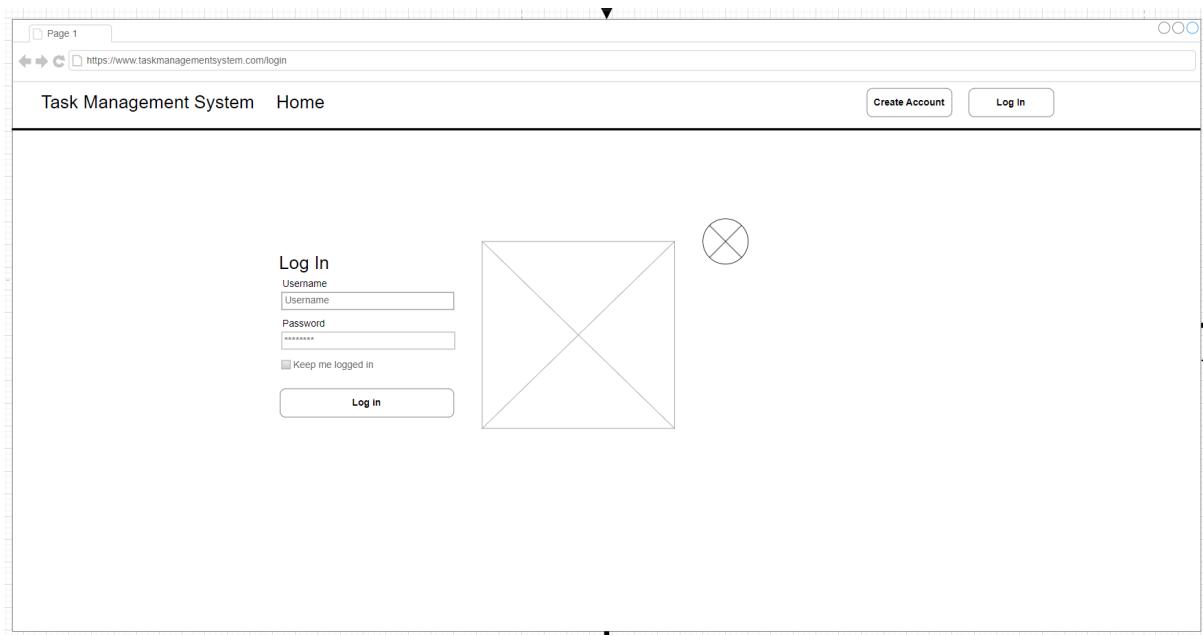
The wireframe shows a web browser window with the URL <https://www.taskmanagementsystem.com/createaccount>. The title bar says "Task Management System Home". The main content area is titled "First Time Setup - Create Root User". It contains several input fields: "First Name" (placeholder "First Name"), "Last Name" (placeholder "Last Name"), "Username (Cannot Be Changed)" (placeholder "Root"), "Email Address" (placeholder "Email Address"), "Phone Number" (placeholder "Phone Number"), "Password" (placeholder "*****"), and "Confirm Password" (placeholder "*****"). A large "X" is drawn over the entire form area. At the bottom right is a "Sign Up" button.

Figure 4 - Setup Wireframe

The wireframe shows a web browser window with the URL <https://www.taskmanagementsystem.com/>. The title bar says "Task Management System Home". On the right side, there are two buttons: "Create Account" and "Log In". The main content area has a heading "Welcome to the Task Management System." and a sub-instruction "Please log in or create an account to begin.". A large "X" is drawn over the entire content area.

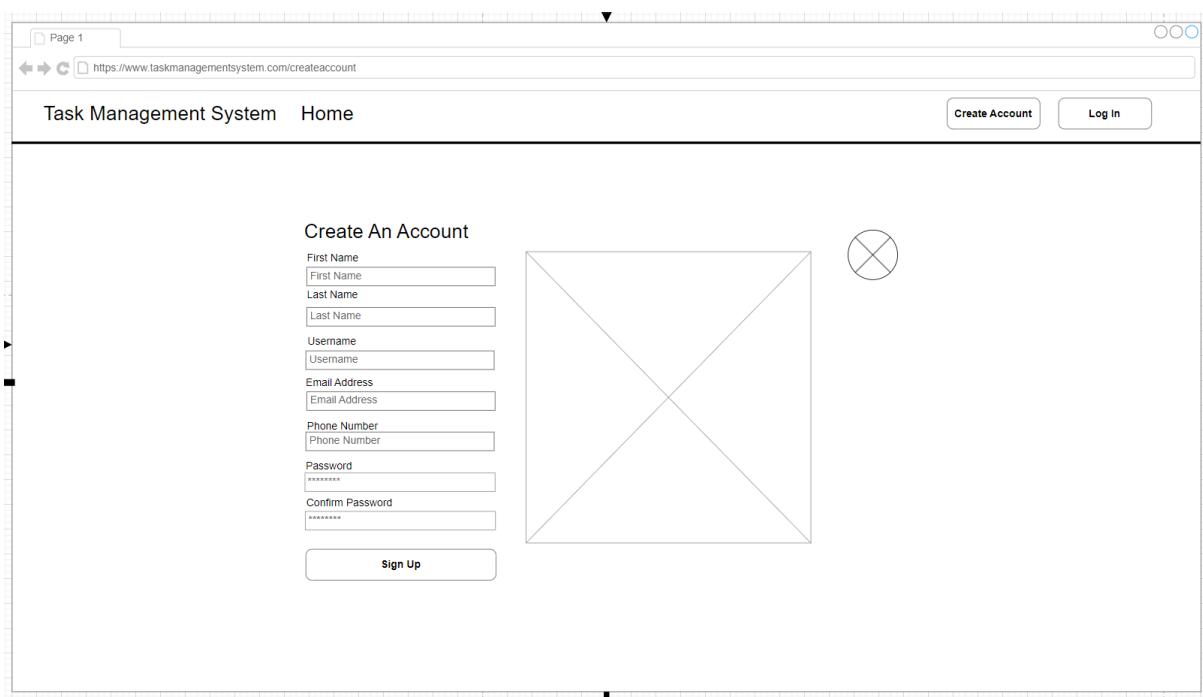
Figure 5 - Landing Page Wireframe

S275931



The wireframe shows a login interface for a Task Management System. At the top, there are browser navigation icons, a URL bar showing <https://www.taskmanagementsystem.com/login>, and a header with "Task Management System" and "Home" buttons. On the right are "Create Account" and "Log In" buttons. The main area contains a "Log In" form with fields for "Username" (with placeholder "Username") and "Password" (with placeholder "*****"). There is also a "Keep me logged in" checkbox and a "Log in" button. To the right of the form is a large red X mark inside a circle.

Figure 6 - Login Wireframe



The wireframe shows a "Create An Account" interface. At the top, there are browser navigation icons, a URL bar showing <https://www.taskmanagementsystem.com/createaccount>, and a header with "Task Management System" and "Home" buttons. On the right are "Create Account" and "Log In" buttons. The main area contains a "Create An Account" form with fields for "First Name" (placeholder "First Name"), "Last Name" (placeholder "Last Name"), "Username" (placeholder "Username"), "Email Address" (placeholder "Email Address"), "Phone Number" (placeholder "Phone Number"), "Password" (placeholder "*****"), and "Confirm Password" (placeholder "*****"). Below these is a "Sign Up" button. To the right of the form is a large red X mark inside a circle.

Figure 7 - Create Account Wireframe

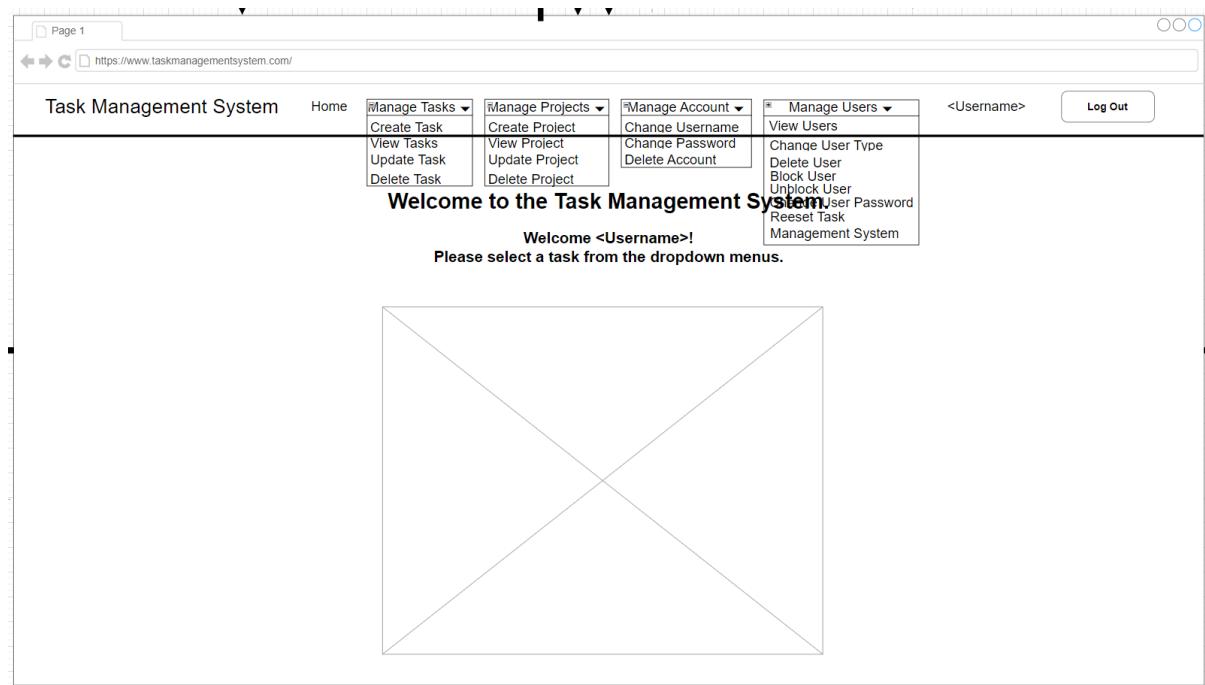


Figure 8 - User Home Wireframe

The wireframe shows a web browser window for the 'Task Management System' at the URL <https://www.taskmanagementsystem.com/addproject>. The header includes a 'Page 1' button, a refresh icon, and the URL. The main menu has 'Home' and four dropdown menus: 'Manage Tasks', 'Manage Projects', 'Manage Account', and 'Manage Users'. Below the menu is a section titled 'Add New Project' with the sub-instruction 'Add projects here.' It contains four input fields: 'Project ID' (with placeholder 'Line 1'), 'Title' (with placeholder 'Line 1'), 'Description' (with placeholder 'Line 1'), and 'Assigned Tasks' (with placeholder 'Line 1'). At the bottom is a 'Add Project' button.

Figure 9 - Add Project Wireframe

Project ID	Title	Description	Assigned Tasks
1	Project 1	Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.	Task 2, Task 7
2	Project 2	Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.	Task 3, Task 5
3	Project 3	Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.	Task 1, Task 4
4	Project 4	Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.	Task 6
5	Project 5	Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.	Task 7, Task 8, Task 9
6	Project 6	Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.	Task 10, Task 13, Task 15, Task 16
7	Project 7	Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.	Task 11, Task 12, Task 14

Figure 10 - View Projects Wireframe

Update Project

Update projects here.

Project ID
Line 1

Title
Line 1

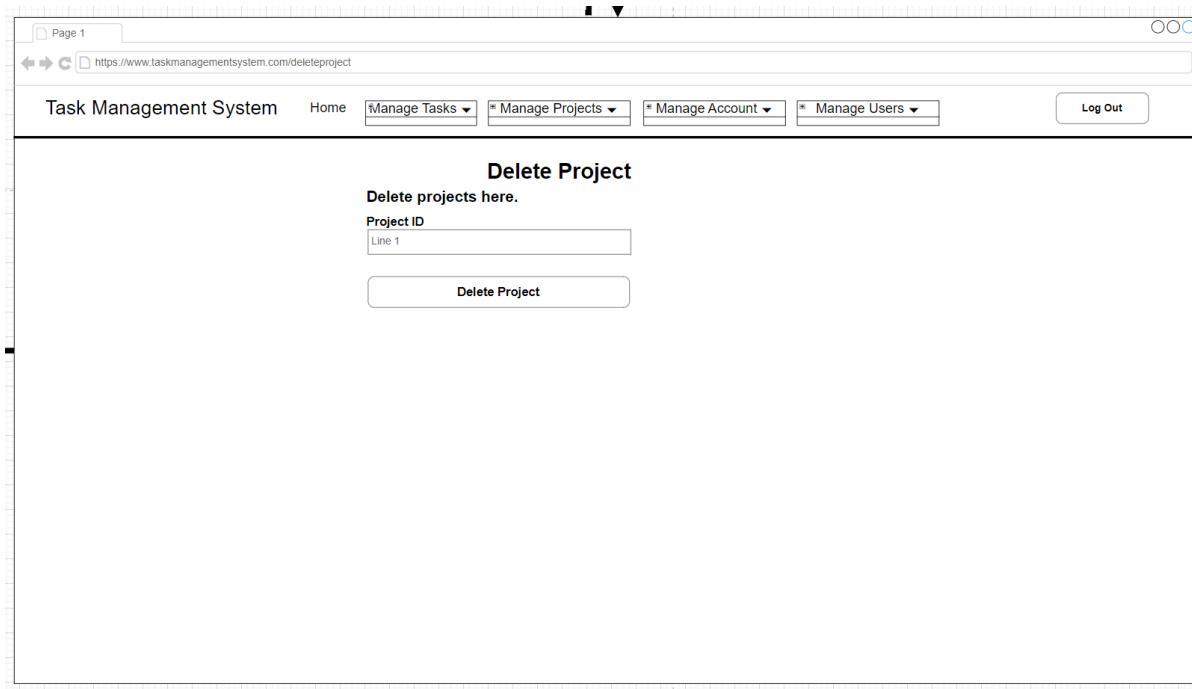
Description
Line 1

Assigned Tasks
Line 1

Update Project

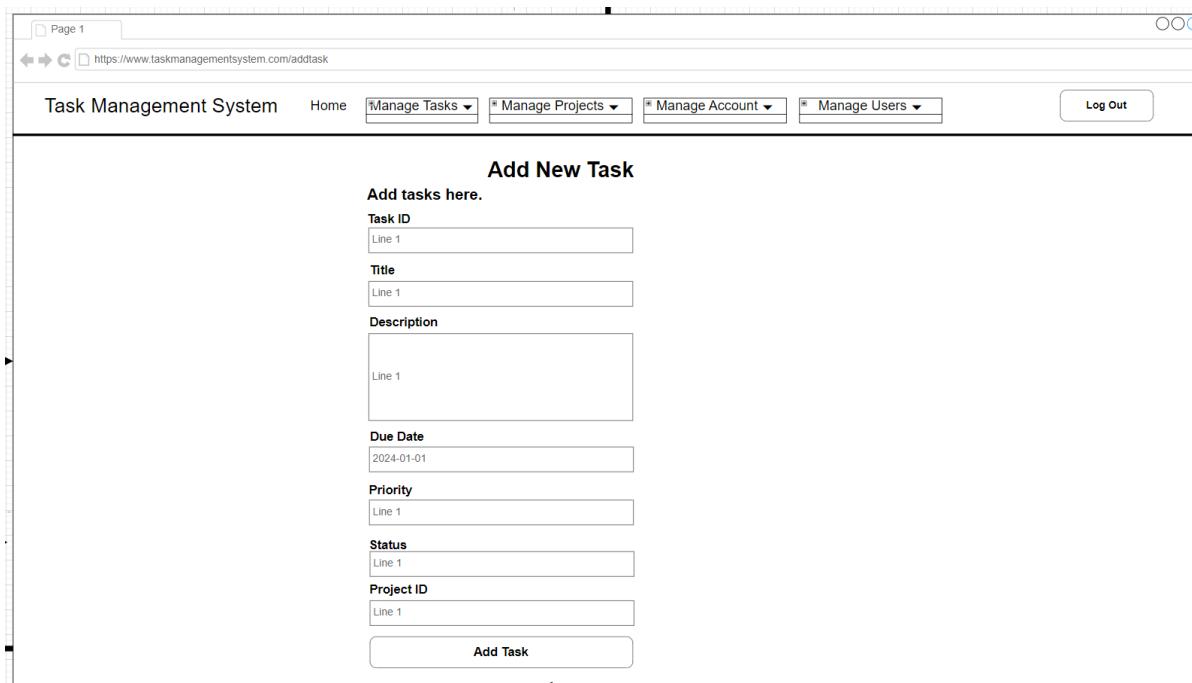
Figure 11 - Update Project Wireframe

S275931



The wireframe for the Delete Project page shows a header with 'Task Management System' and navigation links for Home, Manage Tasks, Manage Projects, Manage Account, Manage Users, and Log Out. The main content area is titled 'Delete Project' and contains the instruction 'Delete projects here.' Below this is a 'Project ID' input field containing 'Line 1'. A large 'Delete Project' button is centered at the bottom.

Figure 12 - Delete Project Wireframe



The wireframe for the Add New Task page shows a header with 'Task Management System' and navigation links for Home, Manage Tasks, Manage Projects, Manage Account, Manage Users, and Log Out. The main content area is titled 'Add New Task' and contains the instruction 'Add tasks here.' Below this are several input fields: 'Task ID' (containing 'Line 1'), 'Title' (containing 'Line 1'), 'Description' (containing 'Line 1'), 'Due Date' (containing '2024-01-01'), 'Priority' (containing 'Line 1'), 'Status' (containing 'Line 1'), and 'Project ID' (containing 'Line 1'). A large 'Add Task' button is centered at the bottom.

Figure 13 - Add Task Wireframe

The wireframe for the 'View Tasks' page features a header with a 'Page 1' link, a URL bar showing 'https://www.taskmanagementsystem.com/viewtasks', and a navigation menu with 'Task Management System' and links for 'Home', 'Manage Tasks', 'Manage Projects', 'Manage Account', 'Manage Users', and 'Log Out'. The main content area is titled 'View Tasks' and contains a table with columns: Task ID, Title, Description, Due Date, Priority, Status, and Project ID. The table has 7 rows, each representing a task with placeholder text for the description.

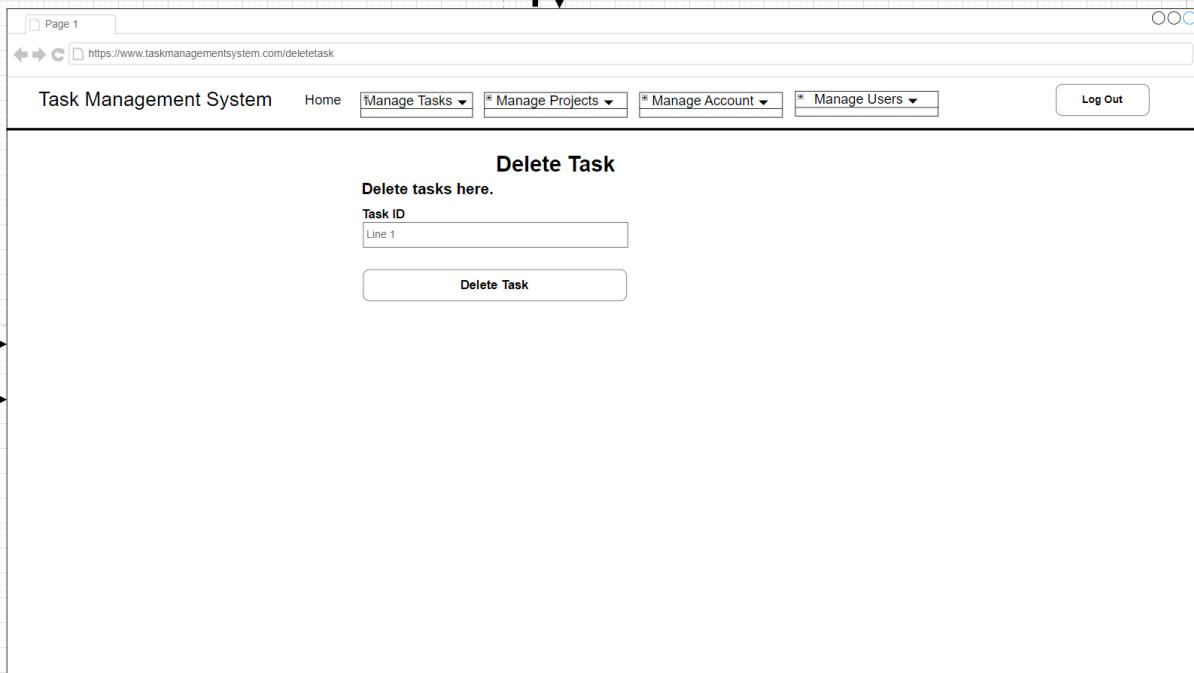
Task ID	Title	Description	Due Date	Priority	Status	Project ID
1	Task 1	Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.	06.03.2024	High	In Progress	3
2	Task 2	Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.	19.09.2024	Medium	Completed	1
3	Task 3	Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.	31.06.2024	Low	Cancelled	2
4	Task 4	Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.	01.05.2024	High	Planned	3
5	Task 5	Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.	09.01.2025	Medium	Over Due	2
6	Task 6	Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.	22.04.2024	Low	In Progress	4
7	Task 7	Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.	21.02.2024	High	Delayed	1

Figure 14 - View Tasks Wireframe

The wireframe for the 'Update Task' page features a header with a 'Page 1' link, a URL bar showing 'https://www.taskmanagementsystem.com/updatetask', and a navigation menu with 'Task Management System' and links for 'Home', 'Manage Tasks', 'Manage Projects', 'Manage Account', 'Manage Users', and 'Log Out'. The main content area is titled 'Update Task' and contains a form with fields for 'Task ID', 'Title', 'Description', 'Due Date', 'Priority', 'Status', and 'Project ID', along with a 'Update Task' button.

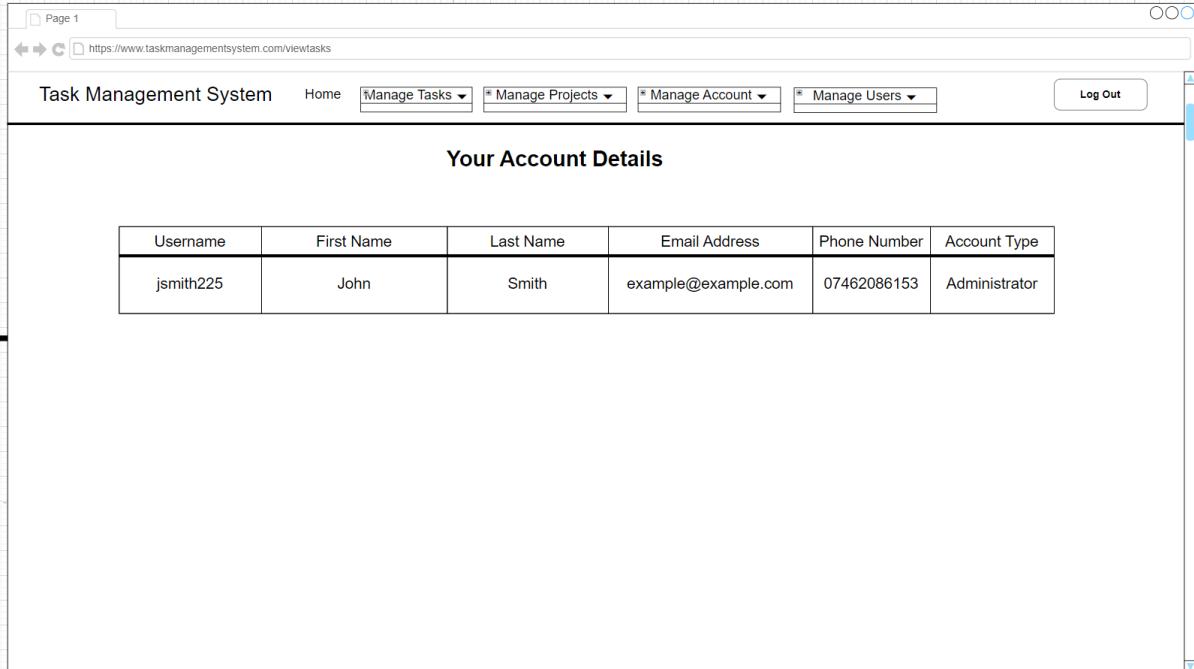
Figure 15 - Update Task Wireframe

S275931



The wireframe for the Delete Task page shows a header with the Task Management System logo, Home link, and four dropdown menus for Manage Tasks, Manage Projects, Manage Account, and Manage Users, along with a Log Out button. The main content area is titled "Delete Task" and contains the instruction "Delete tasks here." Below this is a "Task ID" input field containing "Line 1". A "Delete Task" button is positioned below the input field.

Figure 16 - Delete Task Wireframe

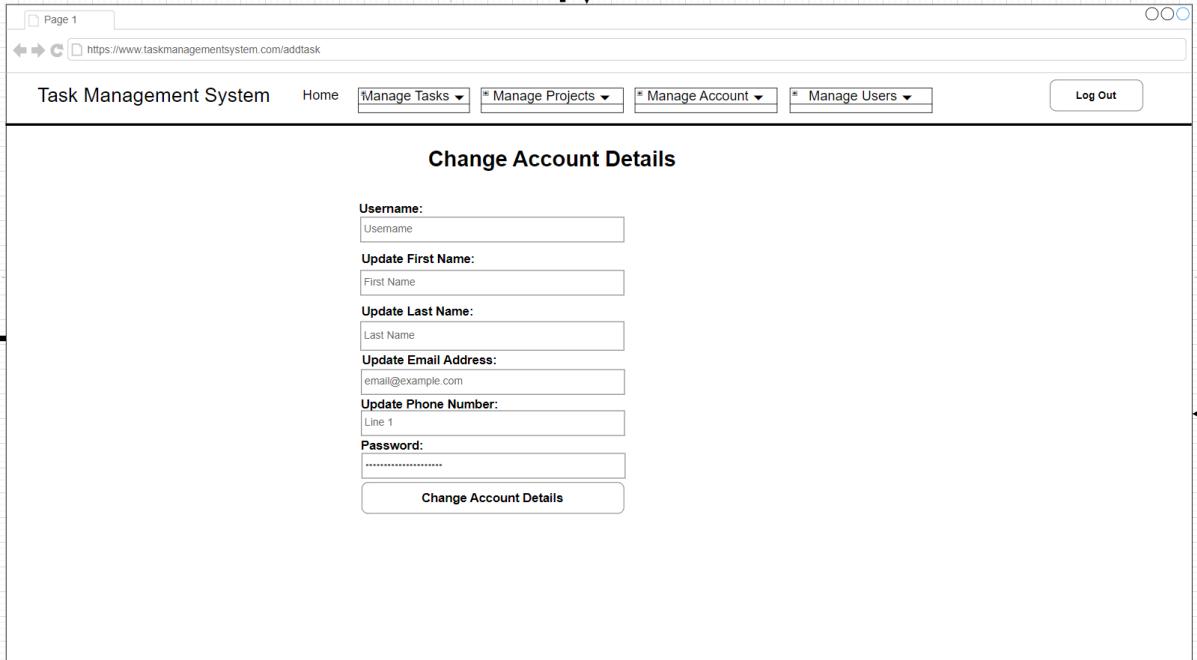


The wireframe for the Account Details page shows a header with the Task Management System logo, Home link, and four dropdown menus for Manage Tasks, Manage Projects, Manage Account, and Manage Users, along with a Log Out button. The main content area is titled "Your Account Details" and displays a table with account information:

Username	First Name	Last Name	Email Address	Phone Number	Account Type
jsmith225	John	Smith	example@example.com	07462086153	Administrator

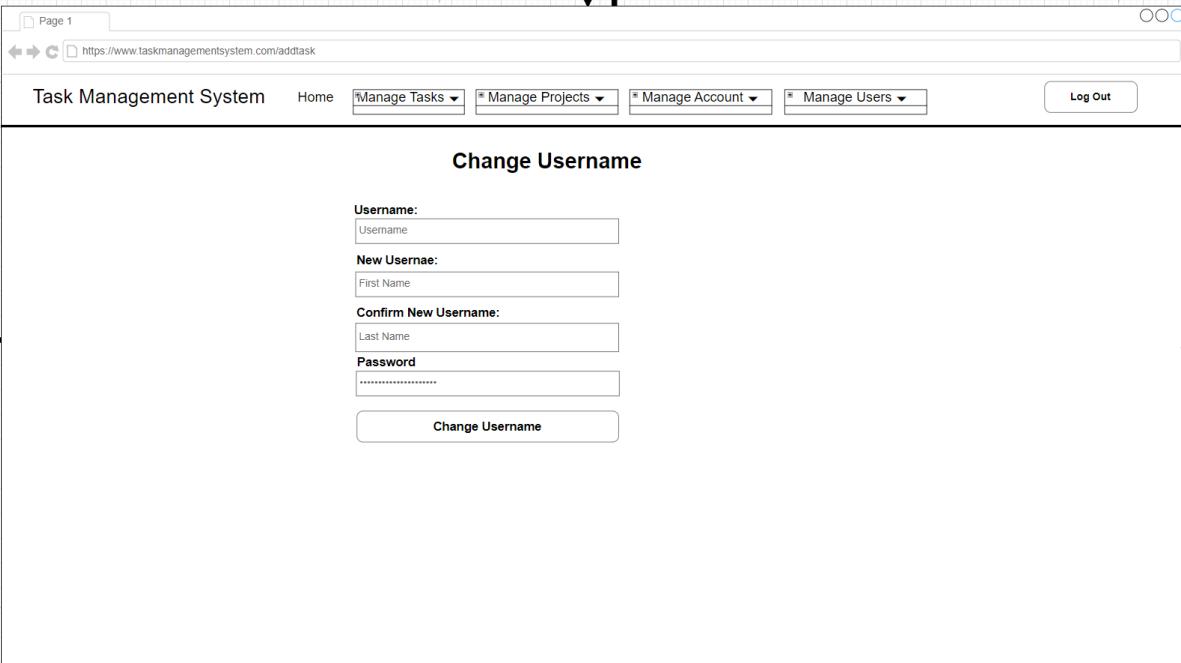
Figure 17 - Account Details Wireframe

S275931



The wireframe shows a web browser window for the 'Task Management System'. The address bar displays 'https://www.taskmanagementsystem.com/addtask'. The header includes the system name, a 'Home' link, and dropdown menus for 'Manage Tasks', 'Manage Projects', 'Manage Account', and 'Manage Users', along with a 'Log Out' button. The main content area is titled 'Change Account Details'. It contains several input fields: 'Username' (placeholder 'Username'), 'Update First Name' (placeholder 'First Name'), 'Update Last Name' (placeholder 'Last Name'), 'Update Email Address' (placeholder 'email@example.com'), 'Update Phone Number' (placeholder 'Line 1'), and 'Password' (placeholder '*****'). A 'Change Account Details' button is at the bottom.

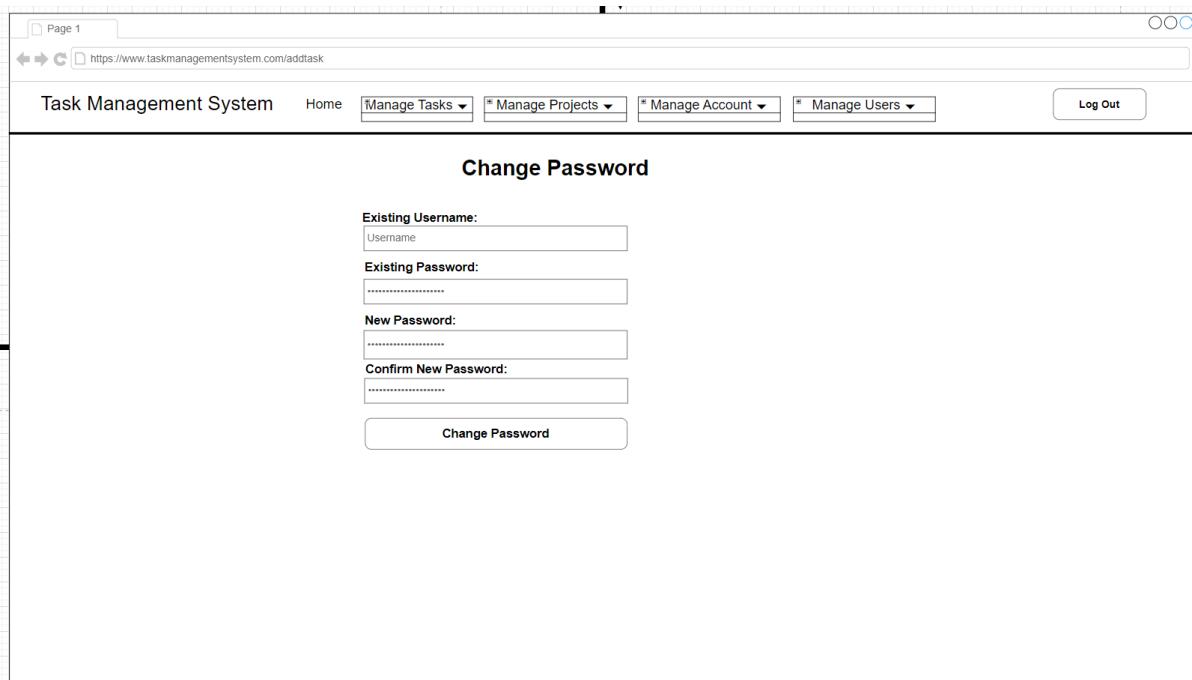
Figure 18 - Change Account details Wireframe



The wireframe shows a web browser window for the 'Task Management System'. The address bar displays 'https://www.taskmanagementsystem.com/addtask'. The header includes the system name, a 'Home' link, and dropdown menus for 'Manage Tasks', 'Manage Projects', 'Manage Account', and 'Manage Users', along with a 'Log Out' button. The main content area is titled 'Change Username'. It contains four input fields: 'Username' (placeholder 'Username'), 'New Username' (placeholder 'First Name'), 'Confirm New Username' (placeholder 'Last Name'), and 'Password' (placeholder '*****'). A 'Change Username' button is at the bottom.

Figure 19 - Change Username Wireframe

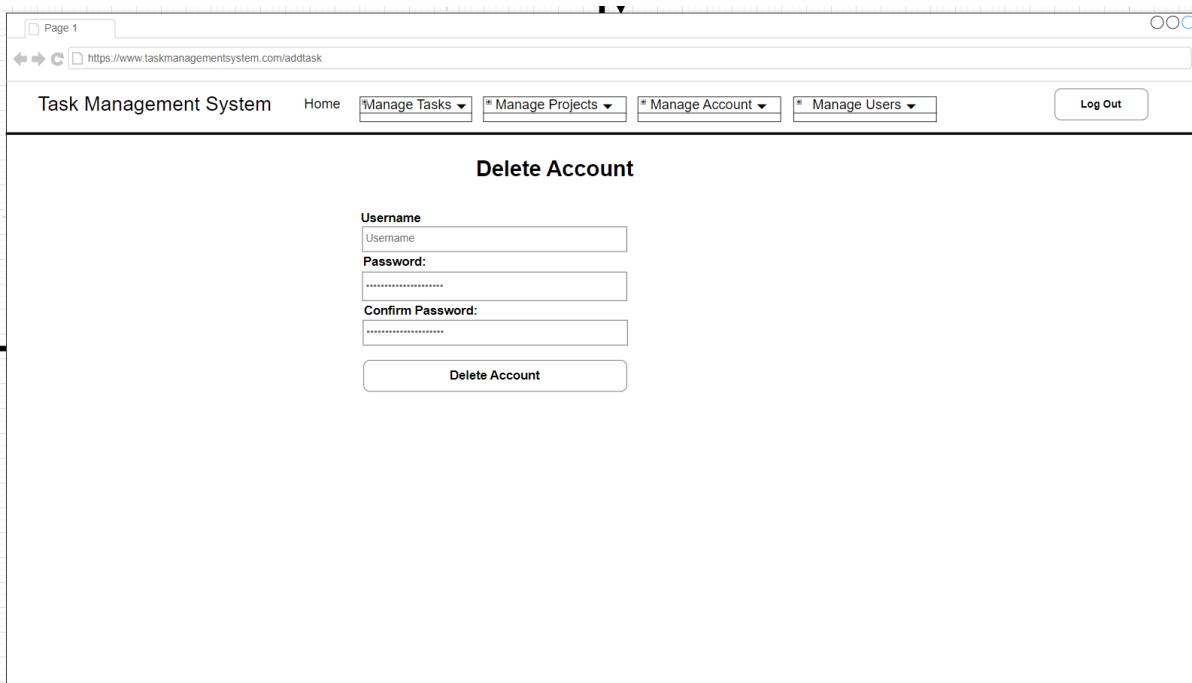
S275931



The wireframe shows a web browser window for the 'Task Management System'. The header includes a 'Page 1' button, a URL bar with 'https://www.taskmanagementsystem.com/addtask', and a 'Log Out' button. Below the header is a navigation bar with tabs: 'Home', 'Manage Tasks', '* Manage Projects', '* Manage Account', '* Manage Users', and 'Log Out'. The main content area is titled 'Change Password' and contains the following fields:

- Existing Username:**
- Existing Password:**
- New Password:**
- Confirm New Password:**
- Change Password**

Figure 20 - Change Password Wireframe



The wireframe shows a web browser window for the 'Task Management System'. The header includes a 'Page 1' button, a URL bar with 'https://www.taskmanagementsystem.com/addtask', and a 'Log Out' button. Below the header is a navigation bar with tabs: 'Home', 'Manage Tasks', '* Manage Projects', '* Manage Account', '* Manage Users', and 'Log Out'. The main content area is titled 'Delete Account' and contains the following fields:

- Username**
- Password:**
- Confirm Password:**
- Delete Account**

Figure 21 - Delete Account Wireframe

Page 1

https://www.taskmanagementsystem.com/viewtasks

Task Management System Home Manage Tasks * Manage Projects * Manage Account * Manage Users Log Out

View Users

Username	First Name	Last Name	Email Address	Phone Number	Account Type	Blocked
jsmith225	John	Smith	example@example.com	07462086153	Administrator	No
user2	User	Two	example@example.com	077632776482	Standard	No
user3	User	Three	example@example.com	07387648234	Standard	Yes
user4	User4	Four	example@example.com	074876232378	Administrator	No
user5	User5	Five	example@example.com	0723424493	Standard	No

Figure 22 - View Users Wireframe

Page 1

https://www.taskmanagementsystem.com/addtask

Task Management System Home Manage Tasks * Manage Projects * Manage Account * Manage Users Log Out

Change User Account Type

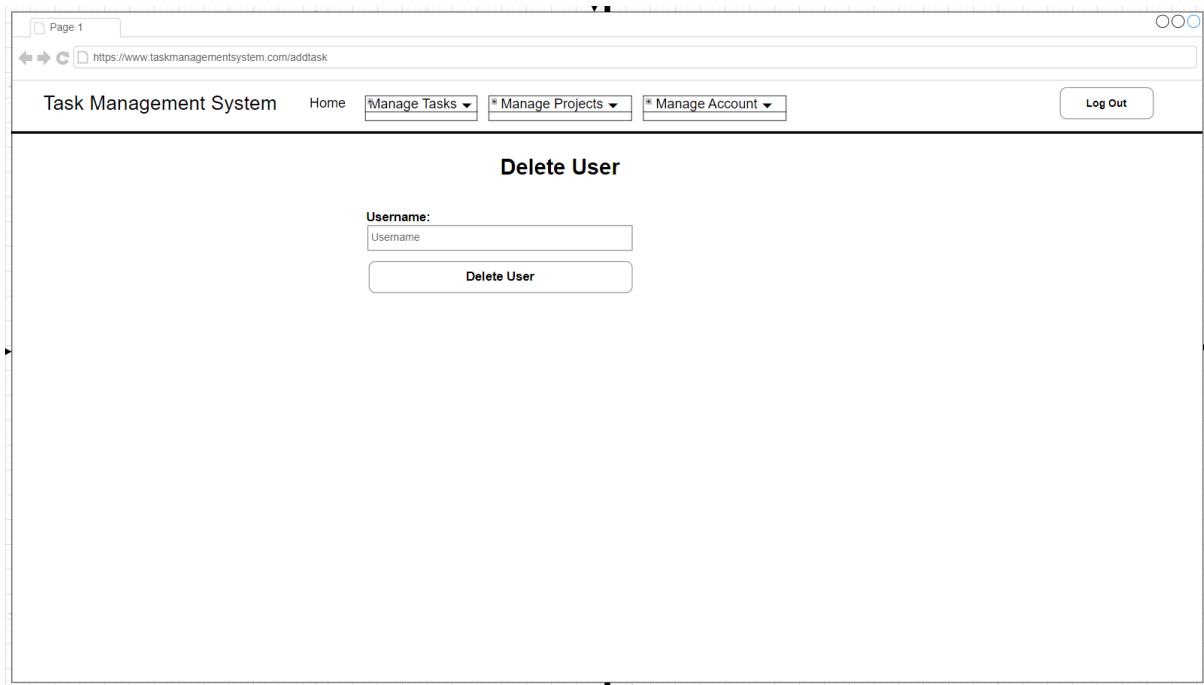
Username:

Account Type:

Change User Account Type

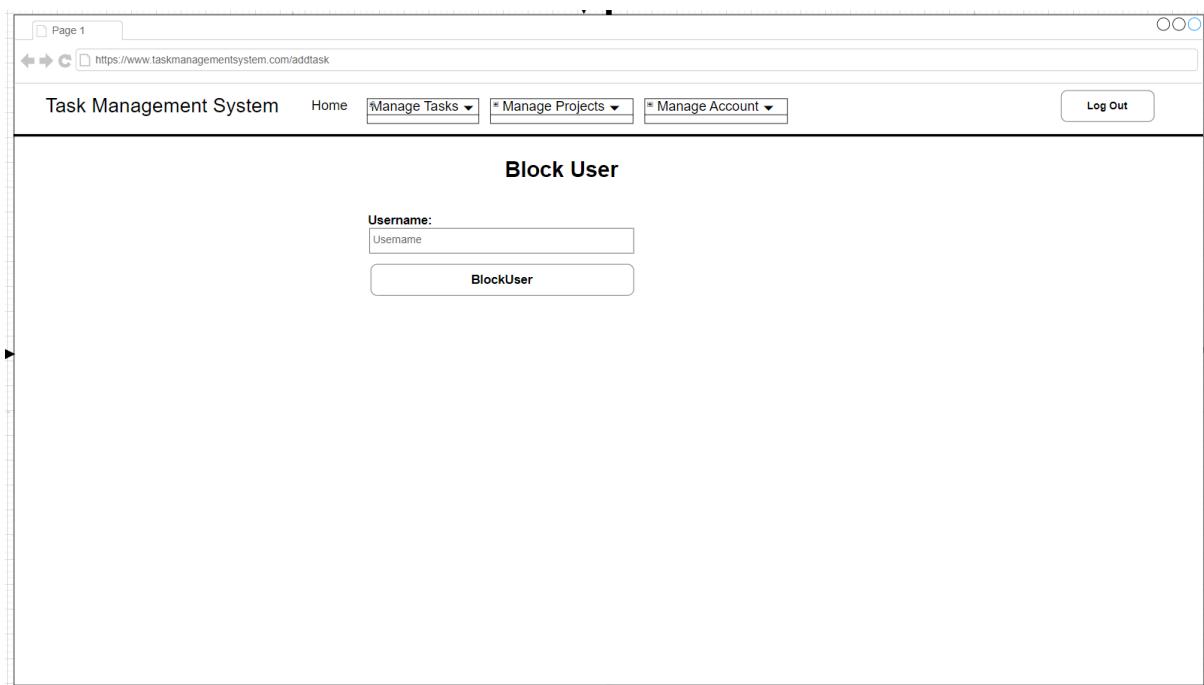
Figure 23 - Change User Account Type Wireframe

S275931



The wireframe for the Delete User page shows a header with 'Task Management System' and navigation links for 'Home', 'Manage Tasks', 'Manage Projects', 'Manage Account', and 'Log Out'. The main content area is titled 'Delete User' and contains a form field labeled 'Username:' with a placeholder 'Username' and a 'Delete User' button.

Figure 24 - Delete User Wireframe



The wireframe for the Block User page shows a header with 'Task Management System' and navigation links for 'Home', 'Manage Tasks', 'Manage Projects', 'Manage Account', and 'Log Out'. The main content area is titled 'Block User' and contains a form field labeled 'Username:' with a placeholder 'Username' and a 'BlockUser' button.

Figure 25 - Block User Wireframe

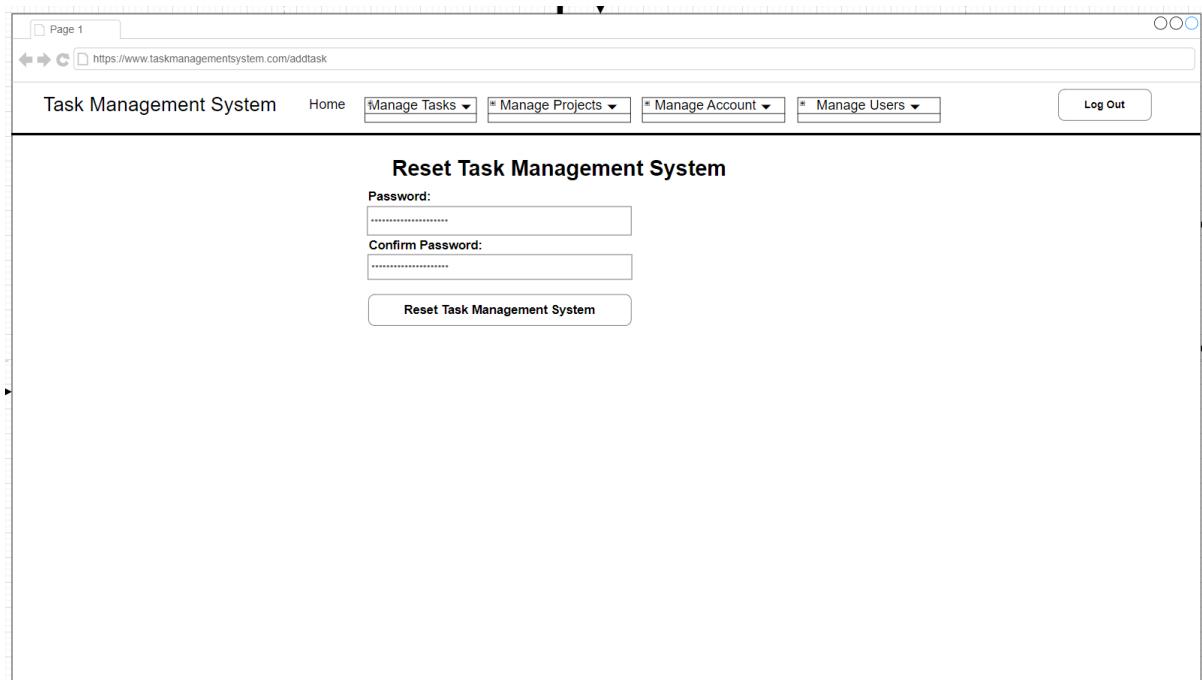
S275931

The wireframe shows a web browser window for the 'Task Management System' at the URL <https://www.taskmanagementsystem.com/addtask>. The header includes a 'Page 1' button, navigation icons, and a 'Log Out' button. The main menu has links for 'Home', 'Manage Tasks', 'Manage Projects', 'Manage Account', and 'Manage Users'. The title 'Unblock User' is centered above a form. The form contains a 'Username:' label and a text input field, followed by a large 'Unblock User' button.

Figure 26 - Unblock User Wireframe

The wireframe shows a web browser window for the 'Task Management System' at the URL <https://www.taskmanagementsystem.com/addtask>. The header includes a 'Page 1' button, navigation icons, and a 'Log Out' button. The main menu has links for 'Home', 'Manage Tasks', 'Manage Projects', 'Manage Account', 'Manage Users', and 'Manage Account'. The title 'Change User Password' is centered above a form. The form contains three text input fields labeled 'Username:', 'New Password:', and 'Confirm New Password:', each with a corresponding placeholder text ('Username', '.....', and '.....'). Below these is a large 'Change Password' button.

Figure 27 - Change User Password Wireframe



The wireframe shows a browser window with the URL https://www.taskmanagementsystem.com/addtask. The page title is "Task Management System". A navigation bar at the top includes "Home", "Manage Tasks", "Manage Projects", "Manage Account", "Manage Users", and "Log Out". The main content area is titled "Reset Task Management System" and contains fields for "Password" and "Confirm Password", both represented by redacted text boxes. A "Reset Task Management System" button is located below these fields.

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:

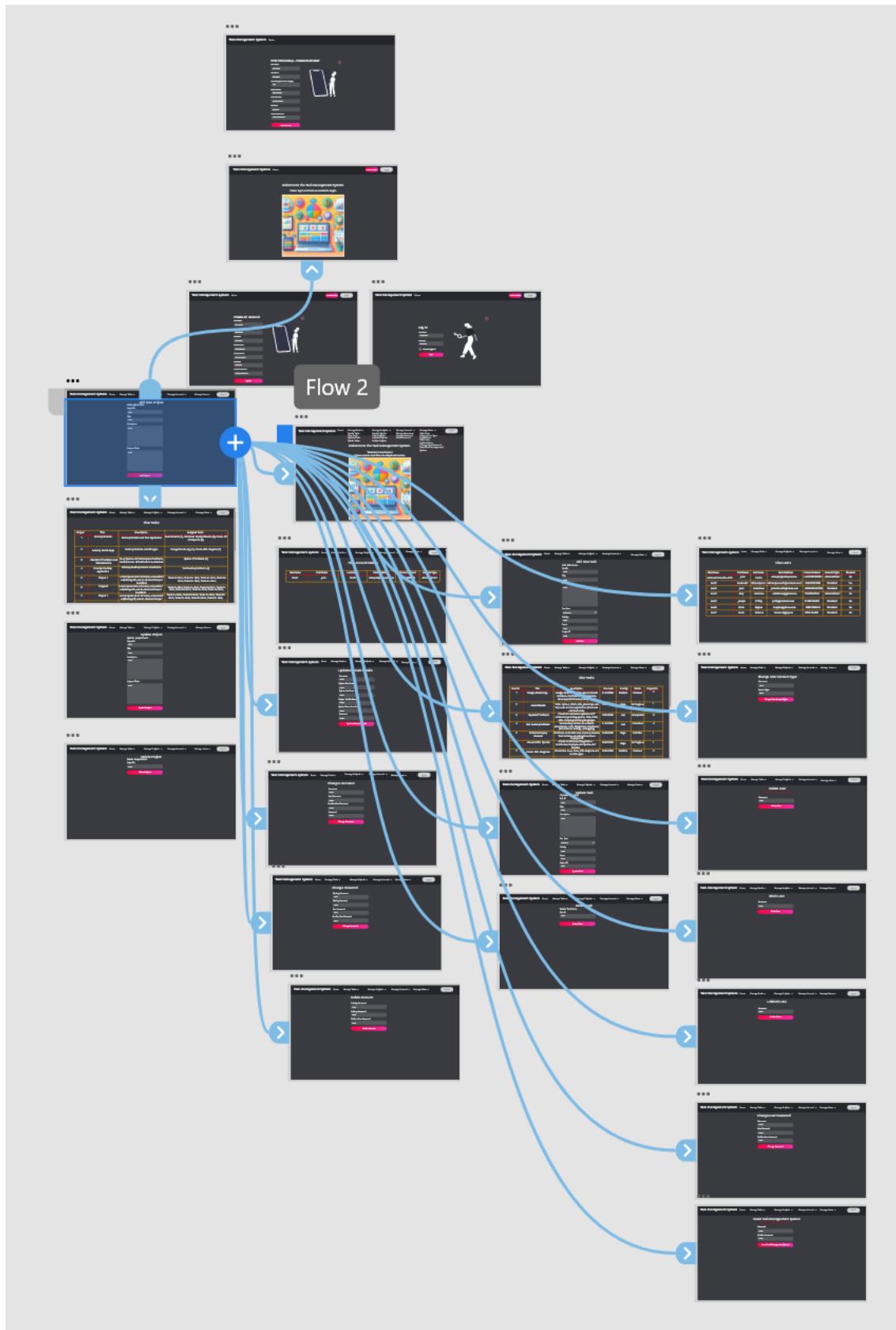


Figure 29 - Overall Mock-up of Task Management System showing how a webpage is linked to other webpages.

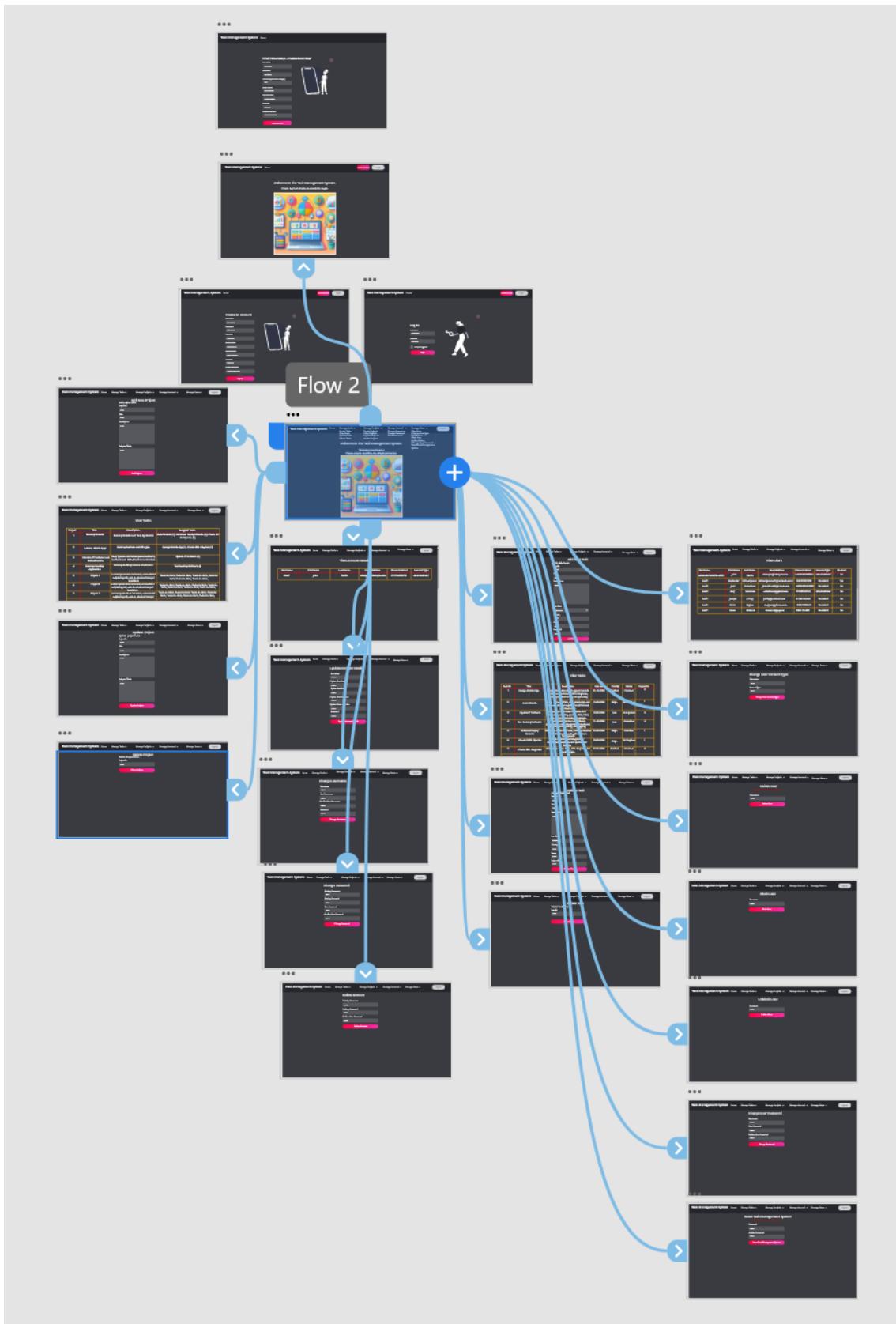


Figure 30 - Overall Mock-up of Task Management System showing how another webpage is linked to other webpages.

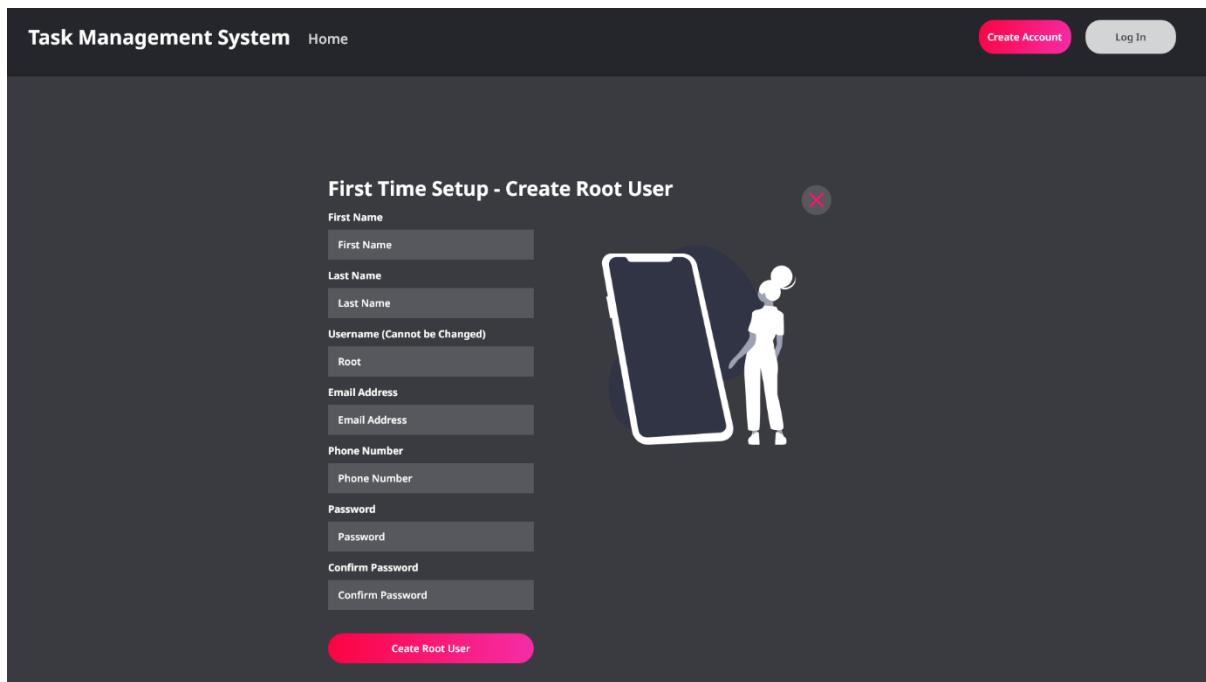


Figure 31 - Setup Page Mock-up.

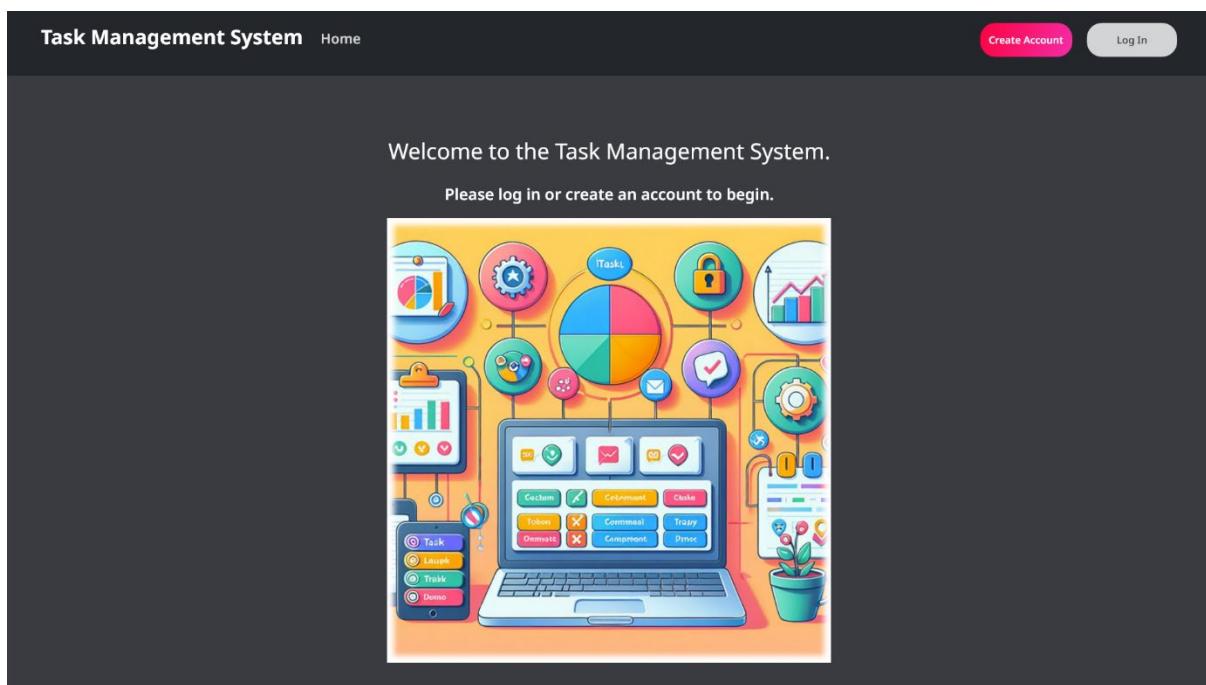


Figure 32 - Landing Page Mock-up.

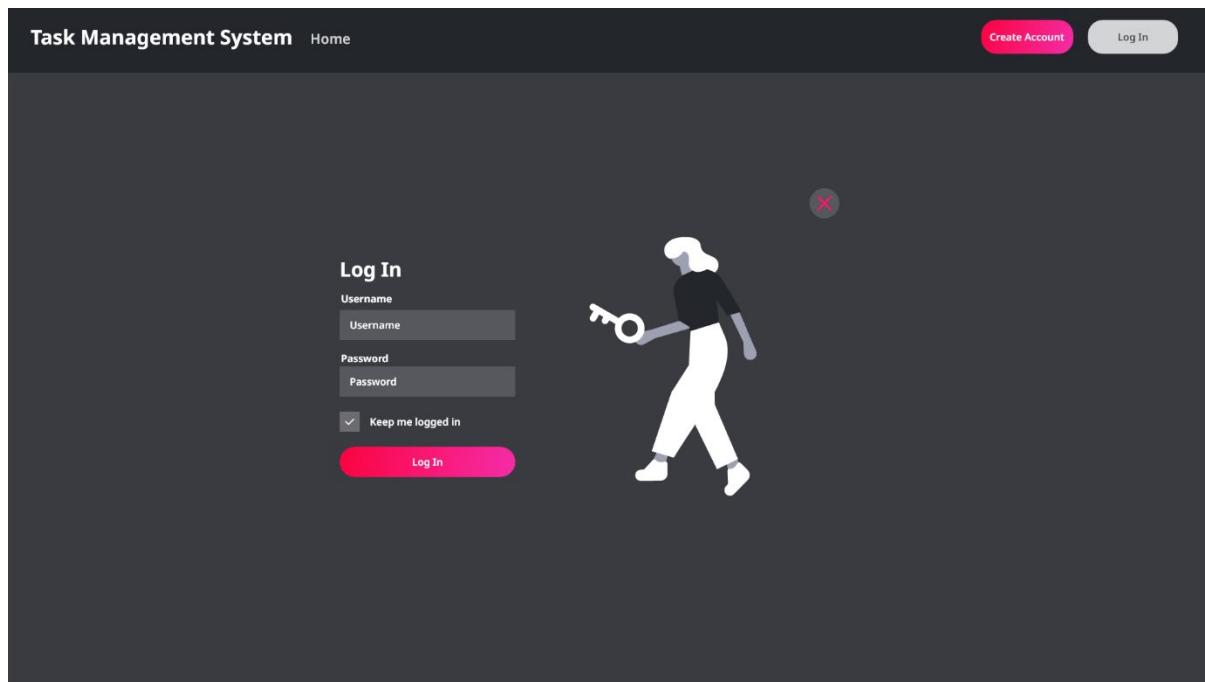


Figure 33 - Login Page Mock-up.

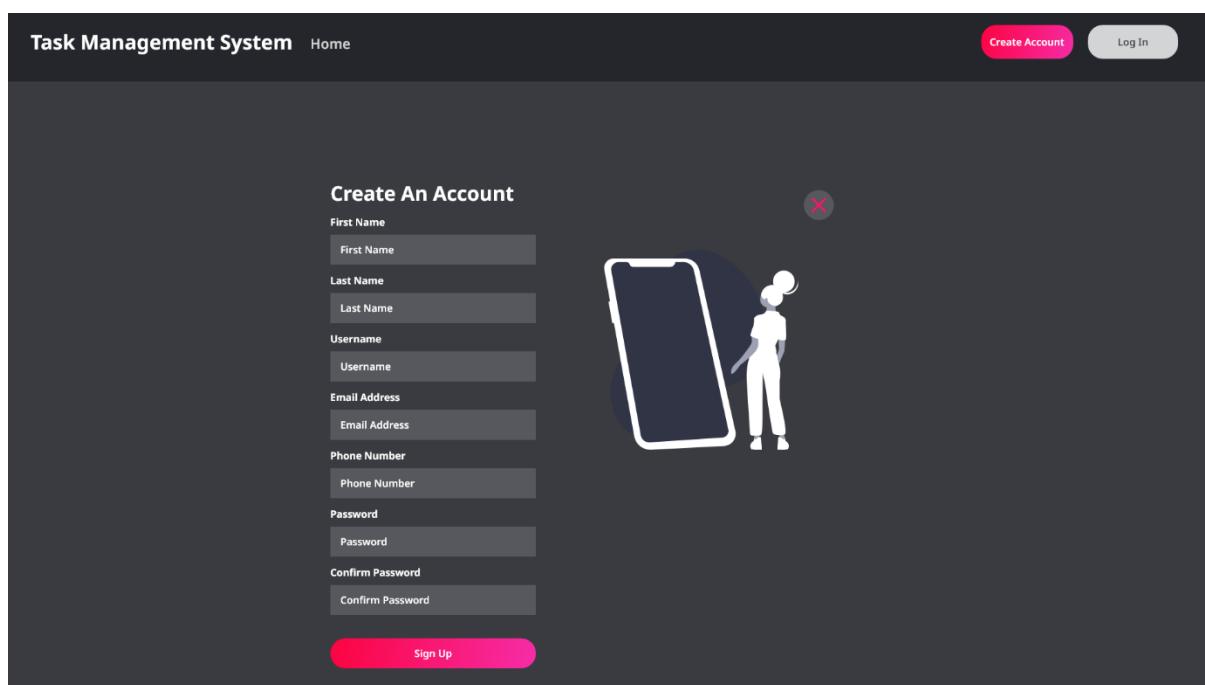


Figure 34 - Create Account Page Mock-up.



Figure 35 - User Home Page Mock-up.

A screenshot of the 'Add New Project' page. The title 'Add New Project' is at the top. Below it is a placeholder text 'Add project here.' There are four input fields: 'Project ID' (placeholder 'Value'), 'Title' (placeholder 'Value'), 'Description' (placeholder 'Value'), and 'Assigned Tasks' (placeholder 'Value'). At the bottom is a red 'Add Project' button.

Figure 36 - Add Project Page Mock-up.

View Projects				
Project	Title	Description	Assigned Tasks	
1	Develop Website	Develop Website and Web Application	Code Website (1), Build and Deploy Website (5), Create CI/CD Pipeline (6).	
2	Develop Mobile Apps	Develop Android and iOS apps.	Design Mobile App (1), Create UML Diagram (7)	
3	Maintain IT Software and Infrastructure.	Keep System and Development software, hardware and infrastructure maintained.	Update IT Software (3)	
4	Develop Desktop Application	Develop desktop version of software.	Test Desktop Software (4)	
5	Project 5	Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt	Tasks Go Here,	
6	Project 6	Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt	Tasks Go Here,	
7	Project 7	Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor	Tasks Go Here,	

Figure 37 - View Projects Page Mock-up.

Task Management System Home Manage Tasks ▾ Manage Projects ▾ Manage Account ▾ Manage Users ▾ Log Out

Update Project

Update project here.

Project ID
Value

Title
Value

Description
Value

Assigned Tasks
Value

Update Project

Figure 38 - Update Project Page Mock-up.

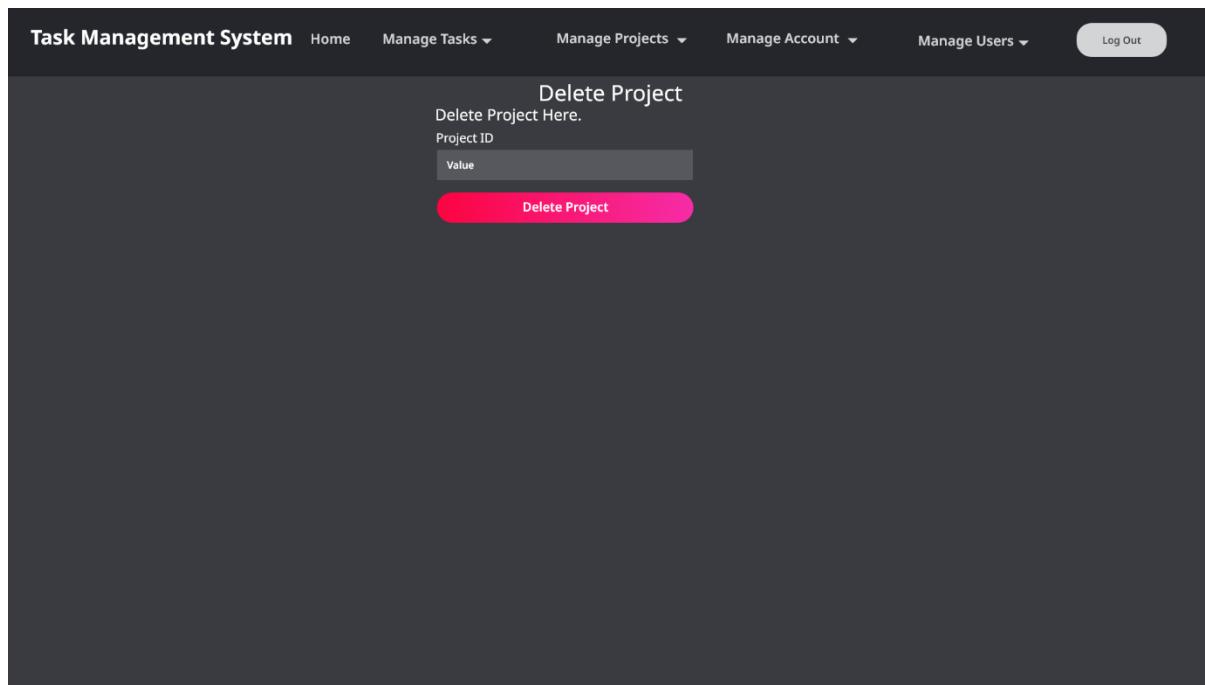


Figure 39 - Delete Project Page Mock-up.

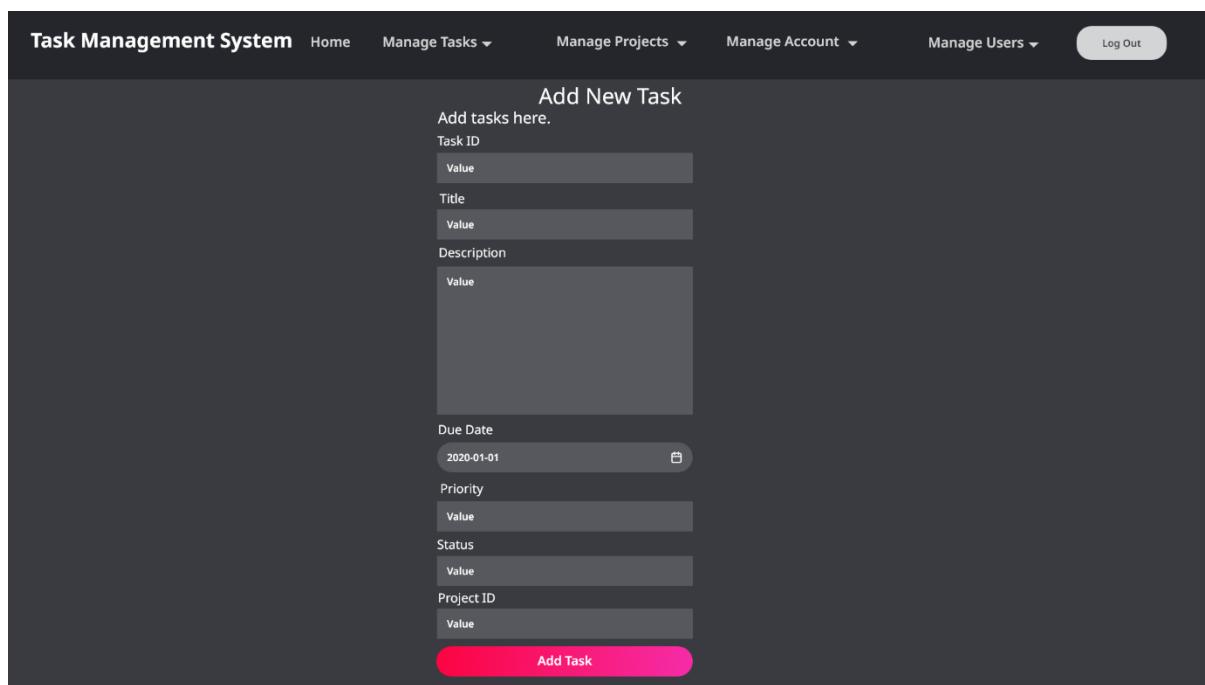


Figure 40 - Add Task Page Mock-up.

View Tasks						
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.

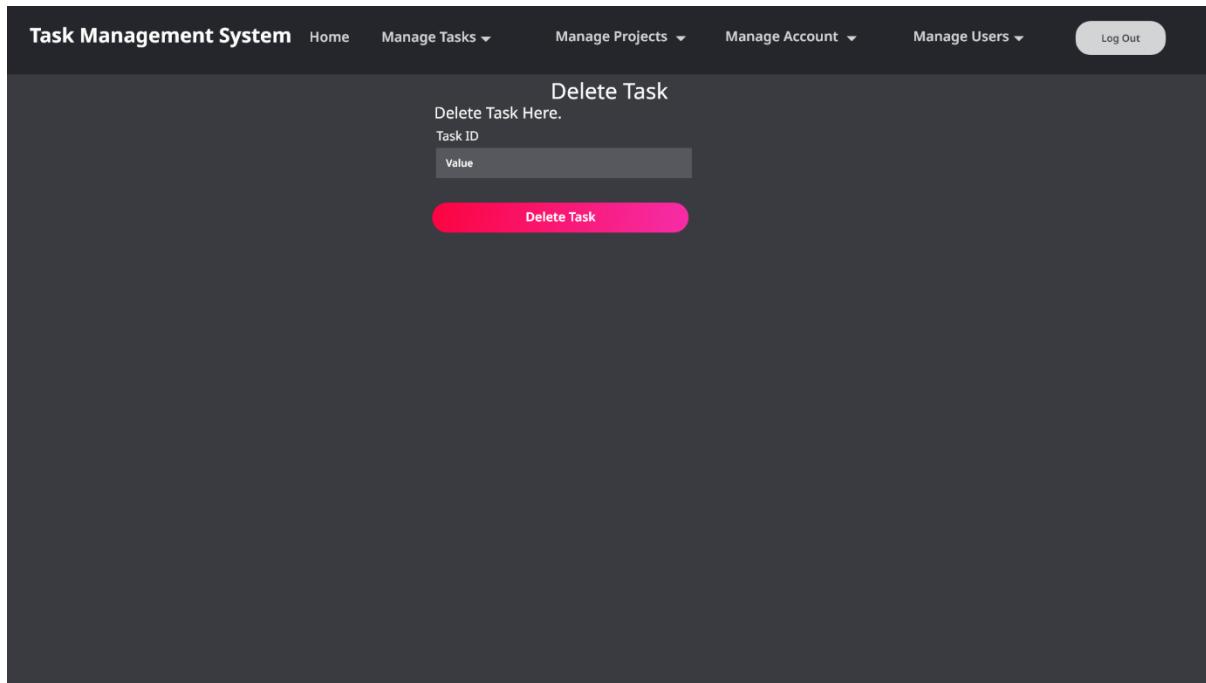
Task Management System Home Manage Tasks ▾ Manage Projects ▾ Manage Account ▾ Manage Users ▾ Log Out

Update Task

Update Task Here

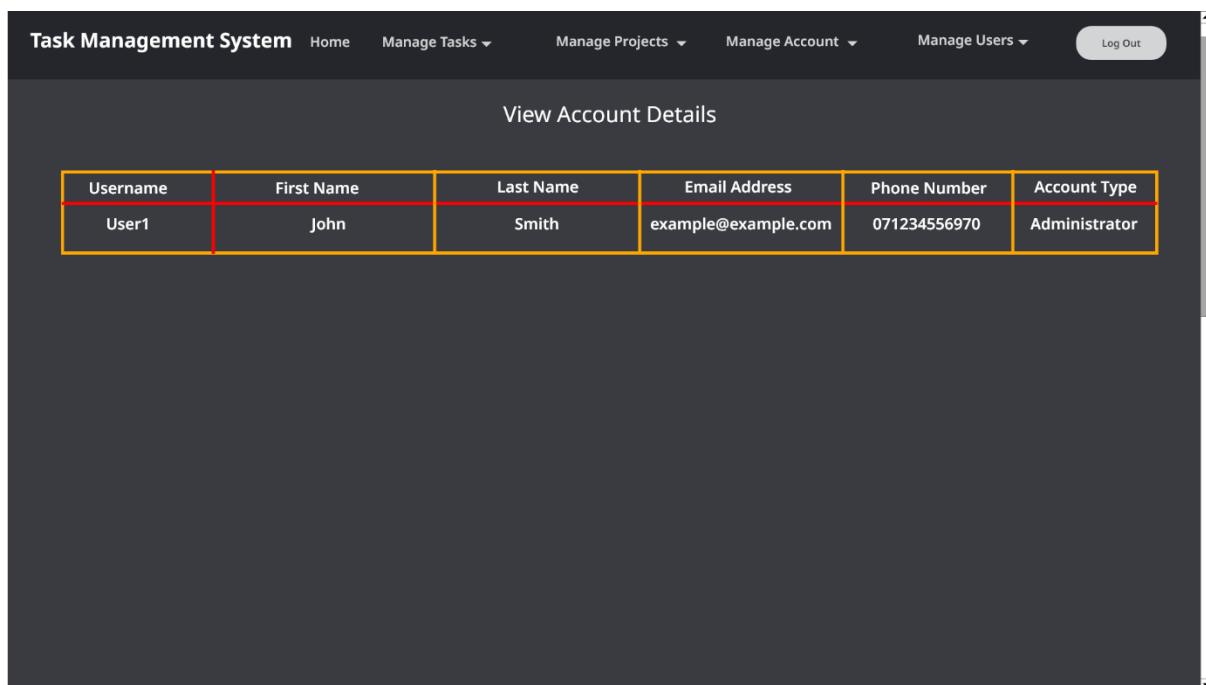
Task ID	<input type="text" value="Value"/>
Title	<input type="text" value="Value"/>
Description	<input type="text" value="Value"/>
Due Date	<input type="text" value="2020-01-01"/> <input type="button" value=""/>
Priority	<input type="text" value="Value"/>
Status	<input type="text" value="Value"/>
Project ID	<input type="text" value="Value"/>
<input type="button" value="Update Task"/>	

Figure 42 - Update Task Page Mock-up.



The screenshot shows a dark-themed web application interface for a Task Management System. At the top, there is a navigation bar with links for Home, Manage Tasks, Manage Projects, Manage Account, Manage Users, and Log Out. The main content area has a header "Delete Task" and a sub-header "Delete Task Here." Below this, there is a form field labeled "Task ID" with the placeholder "Value". A pink button labeled "Delete Task" is centered below the input field.

Figure 43 - Delete Task Page Mock-up.



The screenshot shows a dark-themed web application interface for a Task Management System. At the top, there is a navigation bar with links for Home, Manage Tasks, Manage Projects, Manage Account, Manage Users, and Log Out. The main content area has a header "View Account Details". Below this, there is a table displaying account details for a user named "User1". The table has columns for Username, First Name, Last Name, Email Address, Phone Number, and Account Type. The "Username" column is highlighted with a red border, and the entire row for "User1" is highlighted with an orange border.

Username	First Name	Last Name	Email Address	Phone Number	Account Type
User1	John	Smith	example@example.com	071234556970	Administrator

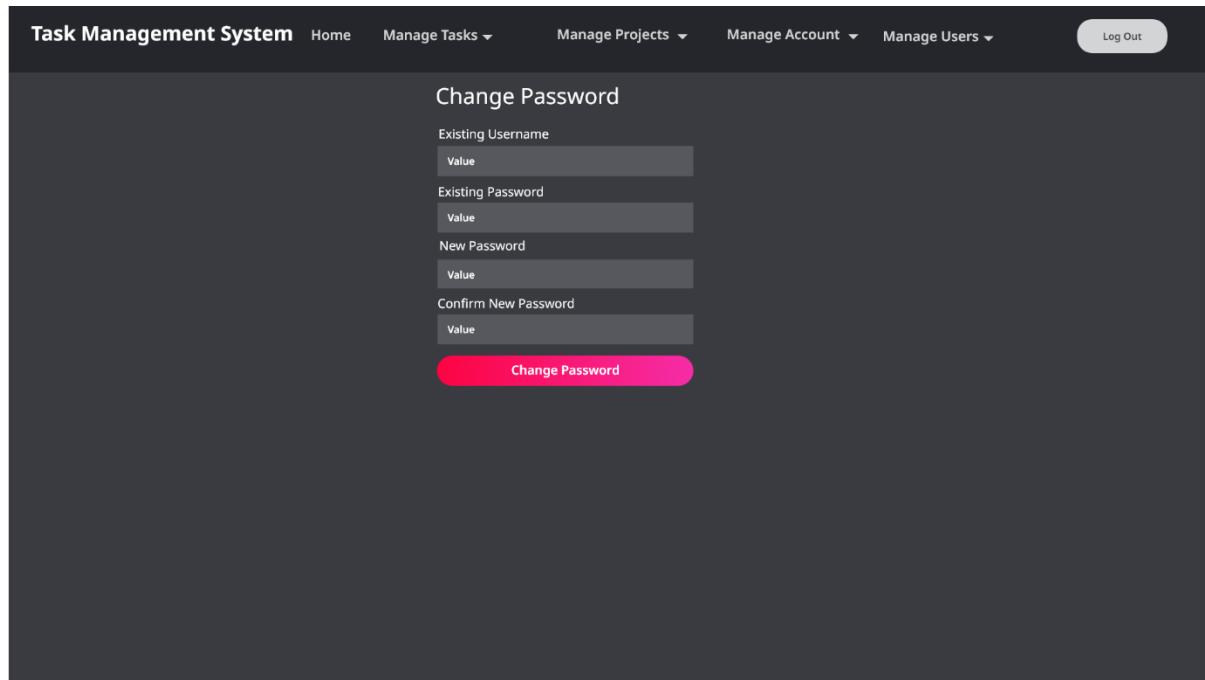
Figure 44 - View Account Details Page Mock-up.

The screenshot shows the 'Update Account Details' page of a web application. At the top, there is a dark header bar with the title 'Task Management System' and several navigation links: 'Home', 'Manage Tasks ▾', 'Manage Projects ▾', 'Manage Account ▾', 'Manage Users ▾', and a 'Log Out' button. Below the header, the main content area has a dark background and features a form titled 'Update Account Details'. The form contains seven input fields, each with a placeholder 'Value': 'Username', 'Update First Name', 'Update Last Name', 'Update Email Address', 'Update Phone Number', 'Password', and 'Confirm Password'. A pink rounded rectangular button labeled 'Update Account Details' is positioned at the bottom right of the form.

Figure 45 - Update Account Details Page Mock-up.

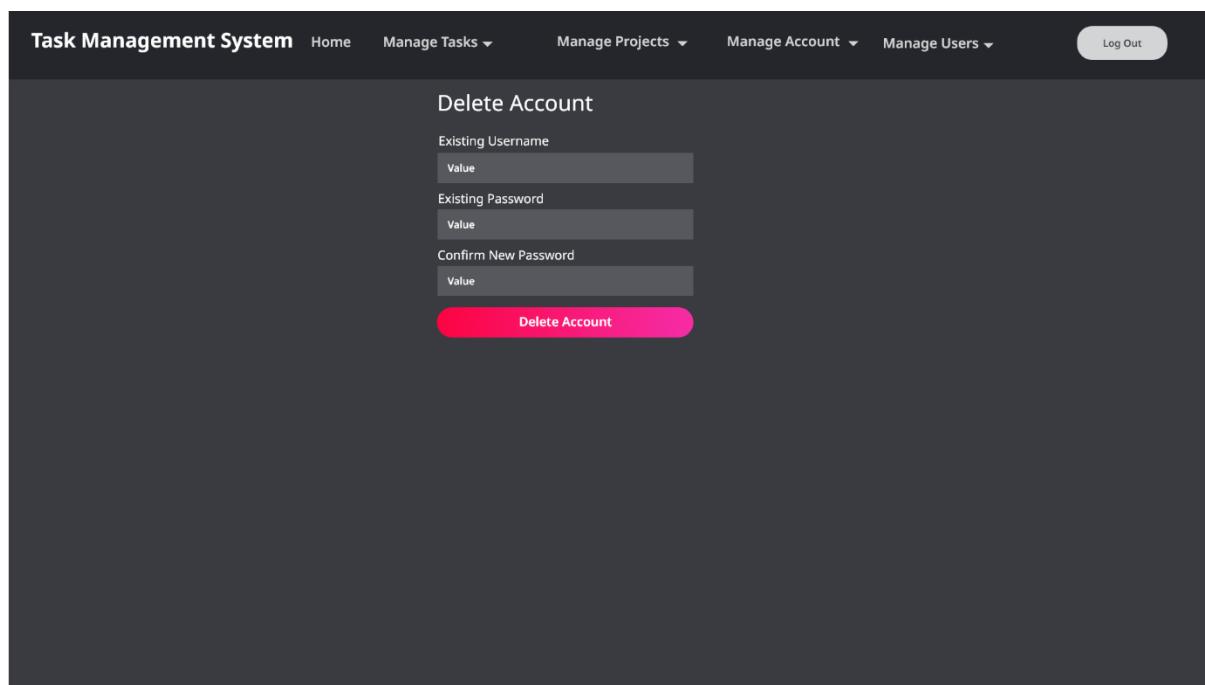
The screenshot shows the 'Change Username' page of a web application. The layout is identical to Figure 45, with a dark header bar and a dark main content area. The title 'Change Username' is centered above a form. This form includes five input fields with 'Value' placeholders: 'Username', 'New Username', 'Confirm New Username', 'Password', and 'Confirm Password'. A pink rounded rectangular button labeled 'Change Username' is located at the bottom right of the form.

Figure 46 - Change Username Page Mock-up.



The screenshot shows the 'Change Password' page of a Task Management System. The page has a dark grey header with the system name and navigation links. Below the header is a form titled 'Change Password' containing four input fields: 'Existing Username', 'Existing Password', 'New Password', and 'Confirm New Password'. Each field has a placeholder 'Value'. At the bottom is a pink button labeled 'Change Password'.

Figure 47 - Change Password Page Mock-up.



The screenshot shows the 'Delete Account' page of a Task Management System. The layout is identical to the 'Change Password' page, featuring a dark grey header, a form titled 'Delete Account', and four input fields for 'Existing Username', 'Existing Password', 'New Password', and 'Confirm New Password'. A pink 'Delete Account' button is at the bottom.

Figure 48 - Delete Account Page Mock-up.

View Users						
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.

Task Management System Home Manage Tasks ▾ Manage Projects ▾ Manage Account ▾ Manage Users ▾ Log Out

Change User Account Type

Username
Value

Account Type
Value

Change User Account Type

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

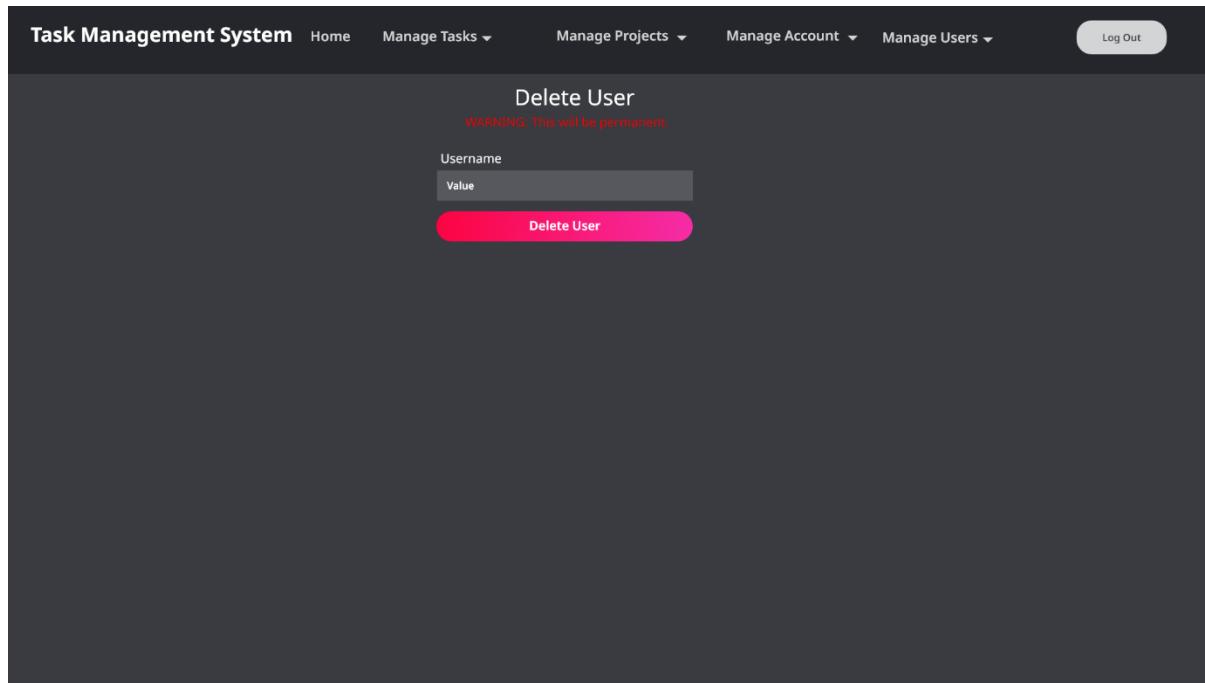


Figure 51 - Delete User Page Mock-up.

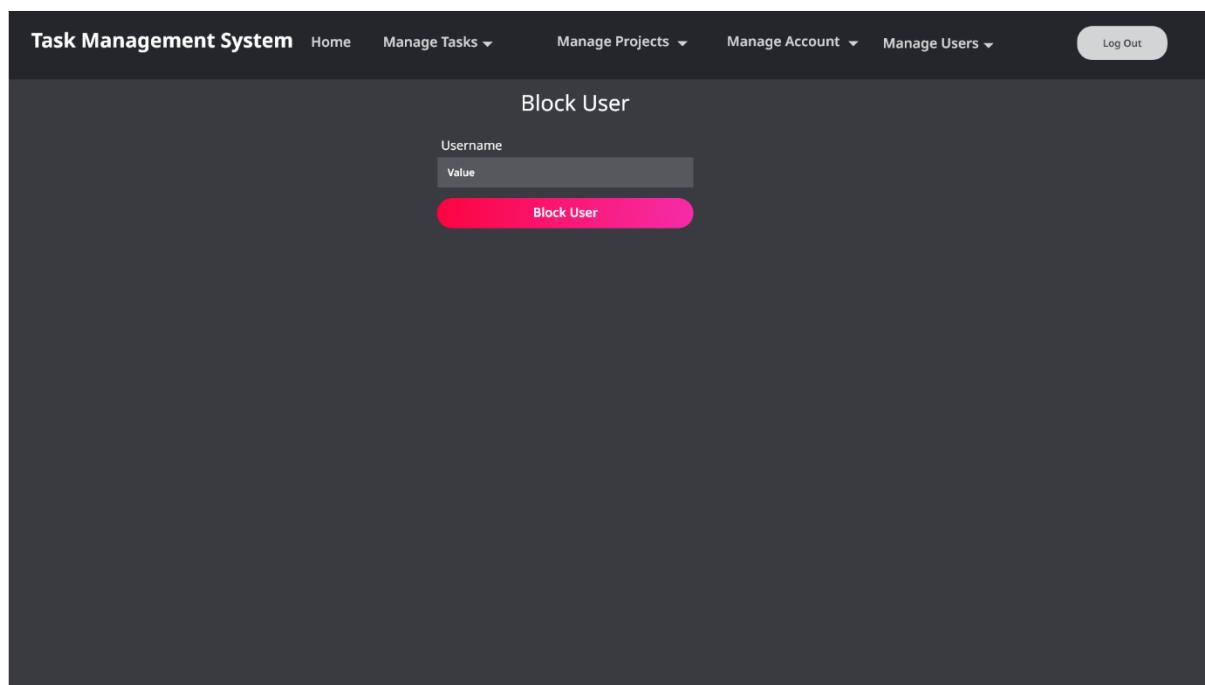


Figure 52 - Block User Page Mock-up.

The screenshot shows a dark-themed web application interface. At the top, there is a navigation bar with the title "Task Management System" and links for "Home", "Manage Tasks", "Manage Projects", "Manage Account", "Manage Users", and "Log Out". The main content area has a heading "Unblock User". Below the heading is a form field labeled "Username" with the placeholder "Value". At the bottom of the form is a pink button labeled "Unblock User".

Figure 53 - Unblock User Page Mock-up.

The screenshot shows a dark-themed web application interface. At the top, there is a navigation bar with the title "Task Management System" and links for "Home", "Manage Tasks", "Manage Projects", "Manage Account", "Manage Users", and "Log Out". The main content area has a heading "Change User Password". Below the heading are three form fields: "Username" (placeholder "Value"), "New Password" (placeholder "Value"), and "Confirm New Password" (placeholder "Value"). At the bottom of the form is a pink button labeled "Change Password".

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

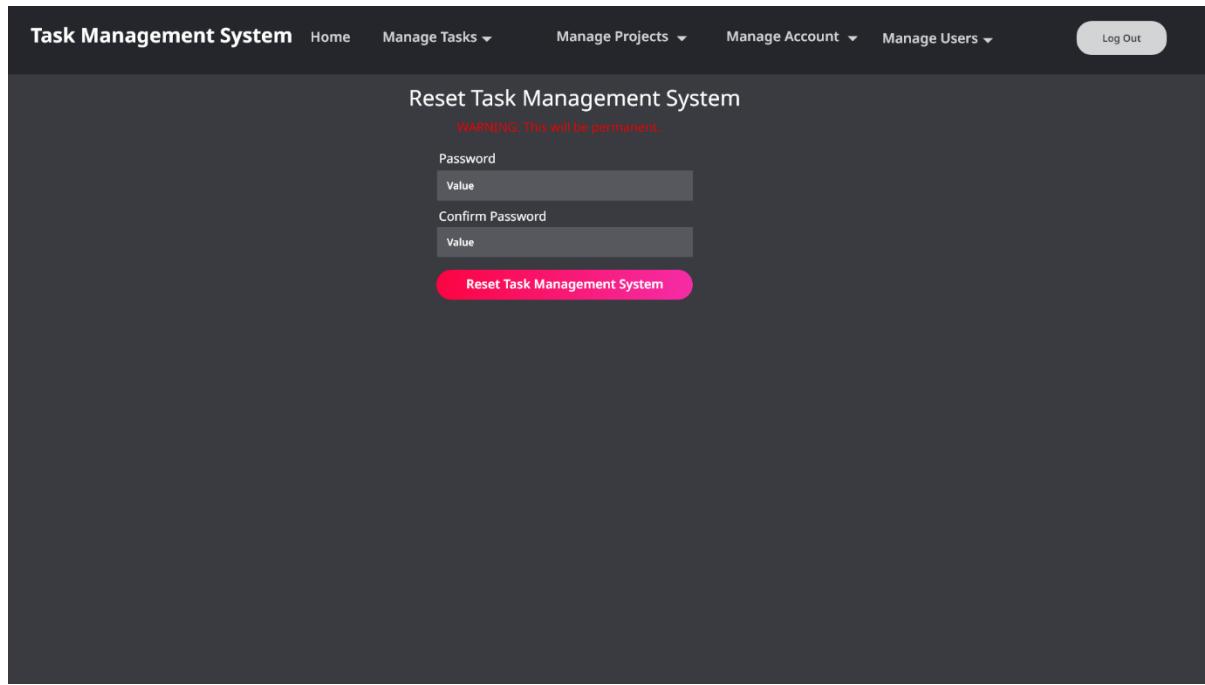


Figure 55 - Factory Reset Page Mock-up.

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

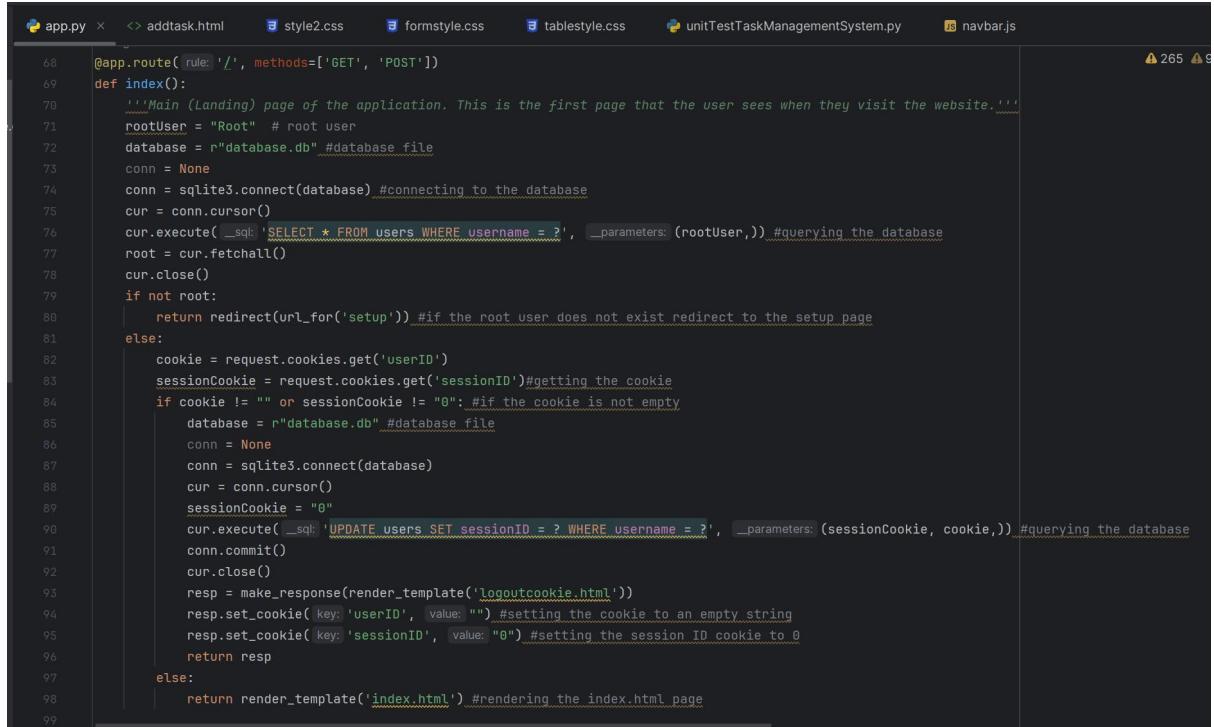
I have made the following decisions for the optional requirements:

- 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 easier 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 than 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:



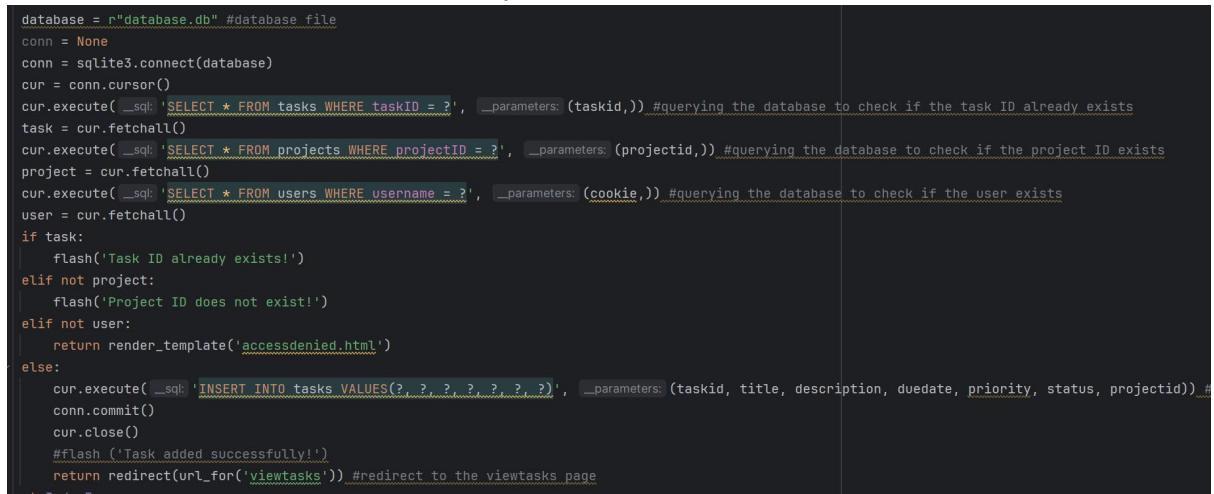
```

68     @app.route('/index', methods=['GET', 'POST'])
69     def index():
70         '''Main (Landing) page of the application. This is the first page that the user sees when they visit the website.'''
71         rootUser = "Root" # root user
72         database = r"database.db" #database file
73         conn = None
74         conn = sqlite3.connect(database) #connecting to the database
75         cur = conn.cursor()
76         cur.execute(_sql: 'SELECT * FROM users WHERE username = ?', _parameters: (rootUser,)) #querying the database
77         root = cur.fetchall()
78         cur.close()
79         if not root:
80             return redirect(url_for('setup')) #if the root user does not exist redirect to the setup page
81         else:
82             cookie = request.cookies.get('userID')
83             sessionCookie = request.cookies.get('sessionID')#getting the cookie
84             if cookie != "" or sessionCookie != "0": #if the cookie is not empty
85                 database = r"database.db" #database file
86                 conn = None
87                 conn = sqlite3.connect(database)
88                 cur = conn.cursor()
89                 sessionCookie = "0"
90                 cur.execute(_sql: 'UPDATE users SET sessionID = ? WHERE username = ?', _parameters: (sessionCookie, cookie,)) #querying the database
91                 conn.commit()
92                 cur.close()
93                 resp = make_response(render_template('logoutcookie.html'))
94                 resp.set_cookie(key: 'userID', value: "") #setting the cookie to an empty string
95                 resp.set_cookie(key: 'sessionID', value: "0") #setting the session ID cookie to 0
96                 return resp
97             else:
98                 return render_template('index.html') #rendering the index.html page
99

```

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

- SQLite Screenshot (SQLite embedded in Python):



```

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

- HTML Screenshot

```

1  - HTML source code for Create Task page -->
2  - Original HTML based on codepen.io responsive form template and modified by S275931 to suit the application. https://codepen.io/andstudio/pen/eYNeMM -->
3  link rel="stylesheet" href="/static/formstyle.css" >!-- Link to the CSS file for form -->
4  head>
5  extends 'base4.html' %} <!-- Extend from base4.html which is the HTML file for the responsive navbar once logged in -->
6
7  block content %} <!-- Block for the content of the page -->
8
9      <h1 class="heading1">{% block title %} Add New Task {% endblock %}</h1> <!-- Heading for the page -->
10     <div class="wpcf7" id="wpcf7-f156-p143-o1 formwrap">
11         <form method="post" class="wpcf7-form" novalidate="novalidate"> <!-- Form for the user to input the task details -->
12             <div style="...">
13                 <input type="hidden" name="_wpcf7" value="156">
14                 <input type="hidden" name="_wpcf7_version" value="3.7.2">
15                 <input type="hidden" name="_wpcf7_locale" value="en_US">
16                 <input type="hidden" name="_wpcf7_unit_tag" value="wpcf7-f156-p143-o1">
17                 <input type="hidden" name="_wpnonce" value="d1da331d93">
18             </div>
19             <p>
20                 <span class="wpcf7-form-control-wrap TaskID">
21                     <label class="text1">Task ID:</label>
22                     <input type="number" name="taskid" value="" size="40" class="textinput wpcf7-form-control wpcf7-text wpcf7-validates-as-required" aria-required="true" aria-invalid="false" />
23                 </span>
24                 <span class="wpcf7-form-control-wrap Title">
25                     <label class="text1">Title:</label>
26                     <input type="text" name="title" value="" size="40" class="textinput wpcf7-form-control wpcf7-text wpcf7-validates-as-required" aria-required="true" aria-invalid="false" />
27                 </span>
28                 <span class="wpcf7-form-control-wrap description">
29                     <label class="text1"><Description>Description</Description></label>
30                     <textarea class="md-textarea form-control rounded-0" id="description" rows="5" name="description" placeholder="Description text here."></textarea>
31                 </span>
32                 <span class="wpcf7-form-control-wrap DueDate">
33                     <label class="text1">Due Date:</label>
34                     <input type="date" name="duedate" value="" size="40" class="textinput wpcf7-form-control wpcf7-text wpcf7-validates-as-required" aria-required="true" aria-invalid="false" />
35                 </span>
36             </p>
37         </form>
38     </div>
39 
```

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

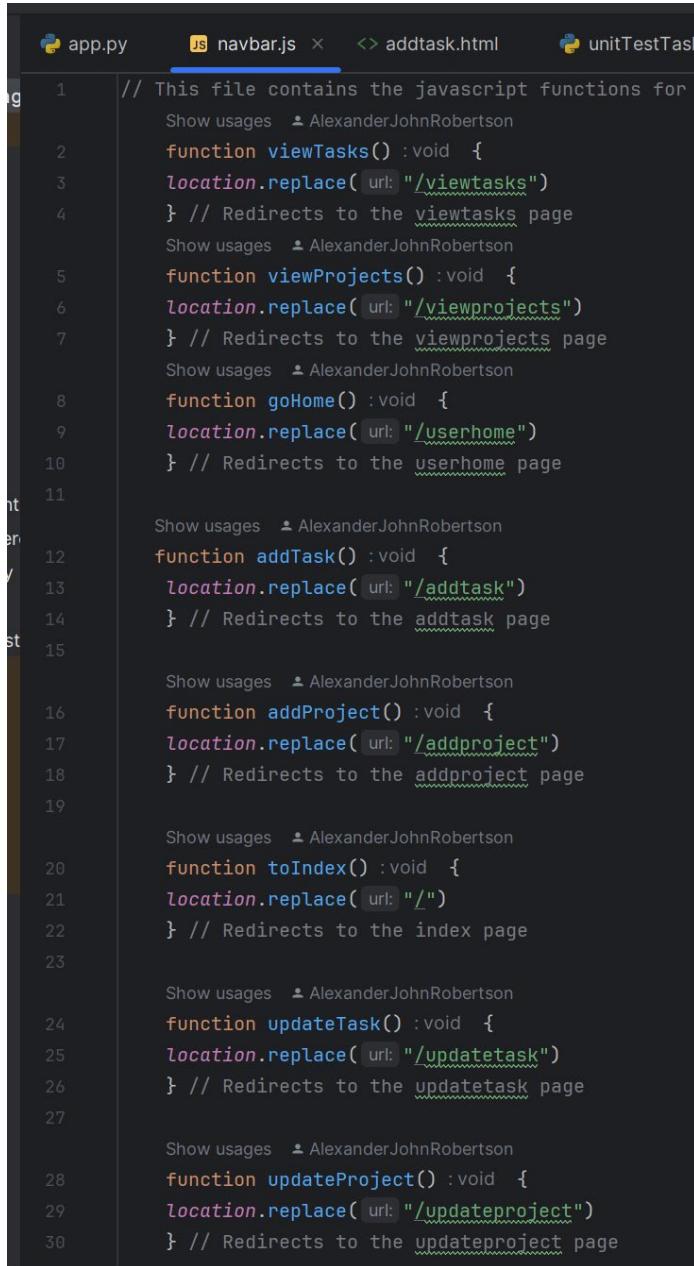
```

1  /* This is the CSS for responsive tables to display data from the database */
2  /* Original CSS sourced from codepen.io and modified by S275931 to suit the application.
3   *
4   * table {
5   *     border: 1px solid #3c3f3f;
6   *     border-collapse: collapse;
7   *     margin: 0;
8   *     padding: 0;
9   *     width: 100%;
10  *     table-layout: fixed;
11  *         overflow-y:scroll;
12  *         height: 100%;
13  *         display:block;
14  * } /* style for table */
15
16  table caption {
17  *     font-size: 1.5em;
18  *     margin: .5em 0 .75em;
19  *     color: #ff8900;
20  * } /* style for table caption */
21
22  table tr {
23  *     background-color: #3c3f3f;
24  *     border: 1px solid #fba600;
25  *     width: 100%;
26  *     padding: .35em;
27  * } /* style for table tr */
28
29  table th,
30  table td {
31  *     padding: .625em;
32  *     text-align: left;
33  *     color: #ff8900;
34  *     width: 100rem;
35  *     border: 1px solid #fd8800;
36  * }
37
38  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



The screenshot shows a code editor with a dark theme. The tab bar at the top has four tabs: 'app.py' (highlighted), 'navbar.js', 'addtask.html', and 'unitTestTask'. The code in the 'navbar.js' tab is as follows:

```

1 // This file contains the javascript functions for
2 Show usages ↗ AlexanderJohnRobertson
3     function viewTasks() :void {
4         location.replace( url: "/viewtasks")
5     } // Redirects to the viewtasks page
6 Show usages ↗ AlexanderJohnRobertson
7     function viewProjects() :void {
8         location.replace( url: "/viewprojects")
9     } // Redirects to the viewprojects page
10 Show usages ↗ AlexanderJohnRobertson
11     function goHome() :void {
12         location.replace( url: "/userhome")
13     } // Redirects to the userhome page
14
15 Show usages ↗ AlexanderJohnRobertson
16     function addTask() :void {
17         location.replace( url: "/addtask")
18     } // Redirects to the addtask page
19
20 Show usages ↗ AlexanderJohnRobertson
21     function addProject() :void {
22         location.replace( url: "/addproject")
23     } // Redirects to the addproject page
24
25 Show usages ↗ AlexanderJohnRobertson
26     function toIndex() :void {
27         location.replace( url: "/")
28     } // Redirects to the index page
29
30 Show usages ↗ AlexanderJohnRobertson
31     function updateTask() :void {
32         location.replace( url: "/updatetask")
33     } // Redirects to the updatetask page
34
35 Show usages ↗ AlexanderJohnRobertson
36     function updateProject() :void {
37         location.replace( url: "/updateproject")
38     } // Redirects to the updateproject page

```

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.

```

68 @app.route( rule: '/', methods=['GET', 'POST'])
69 def index():
70     '''Main (Landing) page of the application. This is the first page that the user sees when they visit the website....'''
71     rootUser = "Root" # root user
72     database = r"database.db" #database file
73     conn = None
74     conn = sqlite3.connect(database) #connecting to the database
75     cur = conn.cursor()
76     cur.execute(_sql: 'SELECT * FROM users WHERE username = ?', __parameters: (rootUser,)) #querying the database
77     root = cur.fetchall()
78     cur.close()
79     if not root:
80         return redirect(url_for('setup')) #if the root user does not exist redirect to the setup page
81     else:
82         cookie = request.cookies.get('userID')
83         sessionCookie = request.cookies.get('sessionID')#getting the cookie
84         if cookie != "" or sessionCookie != "0": #if the cookie is not empty
85             database = r"database.db" #database file
86             conn = None
87             conn = sqlite3.connect(database)
88             cur = conn.cursor()
89             sessionCookie = "0"
90             cur.execute( _sql: 'UPDATE users SET sessionID = ? WHERE username = ?', __parameters: (sessionCookie, cookie,)) #querying the database
91             conn.commit()
92             cur.close()
93             resp = make_response(render_template('logoutcookie.html'))
94             resp.set_cookie(key: 'userID', value: "") #setting the cookie to an empty string
95             resp.set_cookie(key: 'sessionID', value: "0") #setting the session ID cookie to 0
96             return resp
97         else:
98             return render_template('index.html') #rendering the index.html page
99

```

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.

```

app.py          unittestTaskManagementSystem.py  navbar.js      addtask.html
1 import unittest
2 from app import app
3
4 '''Unit test for the Task Management System'''
5
6 ▶ class TestRoutes(unittest.TestCase):
7     ▲ AlexanderJohnRobertson +1
8         def setUp(self): # Set up the app for testing
9             self.app = app.test_client()
10
11    ▲ AlexanderJohnRobertson +1
12        def test_login(self): # Test the login route
13            response = self.app.get('login')
14            self.assertEqual(response.status_code, second: 200)
15            self.assertIn( member: b'Log', response.data)
16
17    ▲ AlexanderJohnRobertson +1
18        def test_create_account(self): # Test the create account route
19            response = self.app.get('createaccount')
20            self.assertEqual(response.status_code, second: 200)
21            self.assertIn( member: b'Create', response.data)
22
23    ▲ AlexanderJohnRobertson +1
24        def test_add_task(self): # Test the add task route
25            response = self.app.get('addtask')
26            self.assertEqual(response.status_code, second: 200)
27            self.assertIn( member: b'Access Denied', response.data)
28
29    ▲ AlexanderJohnRobertson +1
30        def test_view_task(self): # Test the view task route
31            response = self.app.get('viewtask')
32            self.assertEqual(response.status_code, second: 200)
33            self.assertIn( member: b'Access Denied', response.data)
34
35    ▲ AlexanderJohnRobertson +1
36        def test_update_task(self): # Test the update task route
37            response = self.app.get('updatetask')
38            self.assertEqual(response.status_code, second: 200)
39

```

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.

```

app.py      JS navbar.js ×  Python unitTestTaskManagementSystem.py
g
1 // This file contains the javascript functions for th
2 Show usages ↗ AlexanderJohnRobertson
3     function viewTasks() :void {
4         location.replace( url: "/viewtasks")
5     } // Redirects to the viewtasks page
6 Show usages ↗ AlexanderJohnRobertson
7     function viewProjects() :void {
8         location.replace( url: "/viewprojects")
9     } // Redirects to the viewprojects page
10 Show usages ↗ AlexanderJohnRobertson
11     function goHome() :void {
12         location.replace( url: "/userhome")
13     } // Redirects to the userhome page
14
15 Show usages ↗ AlexanderJohnRobertson
16     function addTask() :void {
17         location.replace( url: "/addtask")
18     } // Redirects to the addtask page
19
20 Show usages ↗ AlexanderJohnRobertson
21     function addProject() :void {
22         location.replace( url: "/addproject")
23     } // Redirects to the addproject page
24
25 Show usages ↗ AlexanderJohnRobertson
26     function toIndex() :void {
27         location.replace( url: "/")
28     } // Redirects to the index page
29
30 Show usages ↗ AlexanderJohnRobertson
31     function updateTask() :void {
32         location.replace( url: "/updatetask")
33     } // Redirects to the updatetask page
34
35 Show usages ↗ AlexanderJohnRobertson
36     function updateProject() :void {
37         location.replace( url: "/updateproject")
38     } // Redirects to the updateproject page

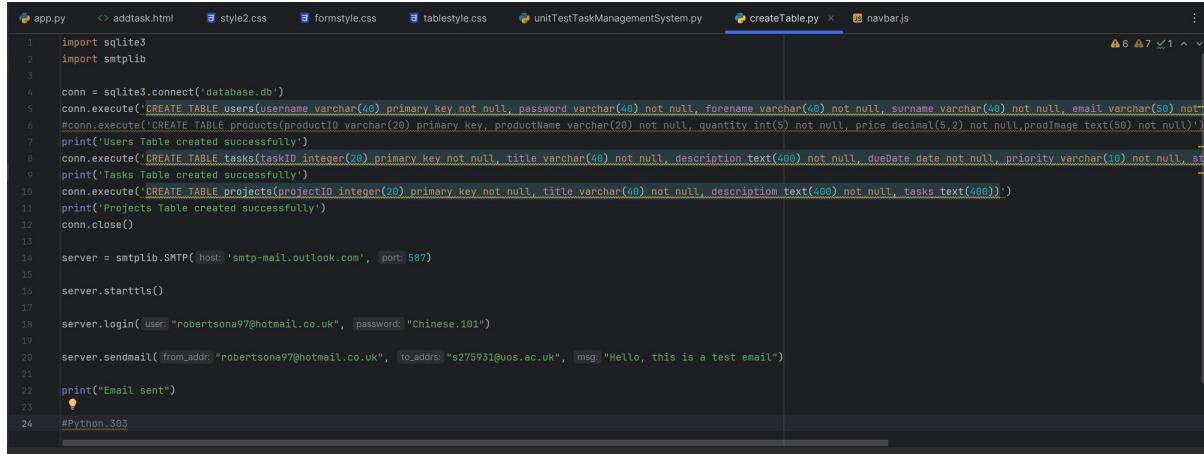
```

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 data 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.

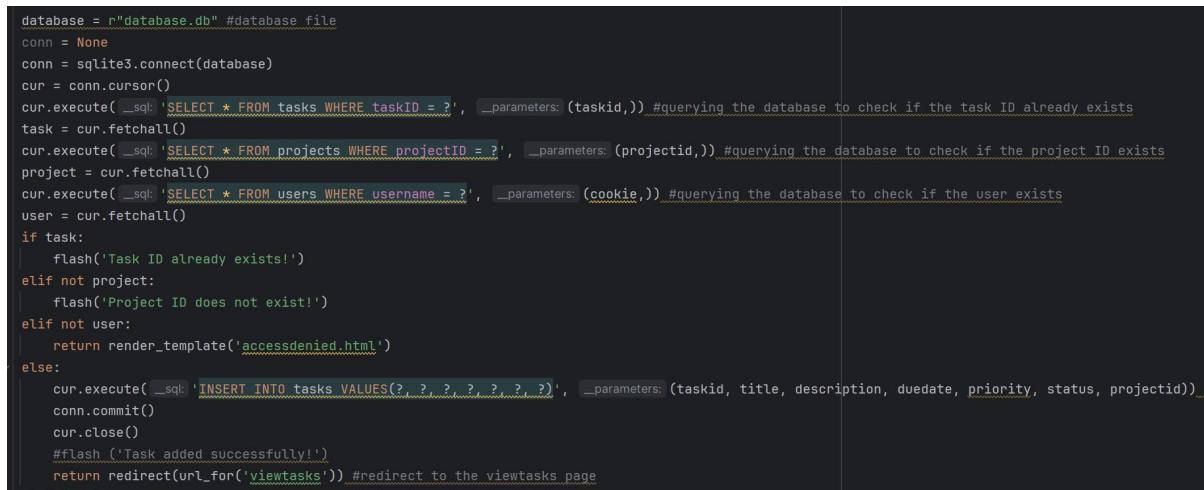


```

1 import sqlite3
2 import smtplib
3
4 conn = sqlite3.connect('database.db')
5 conn.execute('CREATE TABLE users(username varchar(40) primary key not null, password varchar(40) not null, forename varchar(40) not null, surname varchar(40) not null, email varchar(50) not null)')
6 #conn.execute('CREATE TABLE products(productID integer(20) primary key, productName varchar(20) not null, quantity int(5) not null, price decimal(5,2) not null, prodImage text(50) not null)')
7 print('Users Table created successfully')
8 conn.execute('CREATE TABLE tasks(taskID integer(20) primary key not null, title varchar(40) not null, description text(400) not null, dueDate date not null, priority varchar(10) not null, status varchar(10) not null)')
9 print('Tasks Table created successfully')
10 conn.execute('CREATE TABLE projects(projectID integer(20) primary key not null, title varchar(40) not null, description text(400) not null, tasks text(400))')
11 print('Projects Table created successfully')
12 conn.close()
13
14 server = smtplib.SMTP('host: smtp-mail.outlook.com', port: 587)
15
16 server.starttls()
17
18 server.login(user: "robertson97@hotmail.co.uk", password: "Chinese.101")
19
20 server.sendmail(from_addr: "robertson97@hotmail.co.uk", to_addr: "s275931@uos.ac.uk", msg: "Hello, this is a test email")
21
22 print("Email sent")
23
24 #Python_303

```

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.

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.

The screenshot shows the SQLite Database Browser interface. The left sidebar displays the database structure with 'database (SQLite 3)' expanded, showing 'Tables (3)' containing 'projects', 'tasks' (selected), and 'users'. The main area shows the 'Structure' tab for the 'tasks' table. The table has 7 columns: taskID (Primary Key, integer(20)), title (varchar(40)), description (text(400)), dueDate (date), priority (varchar(10)), status (varchar(20)), and projectID (integer(20)). The 'Collate' and 'Generated' columns are both set to NULL.

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

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

View Tasks						
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	adcascasdc	2024-02-13	Medium	Completed	1
9	Test Mobile app iPhone	Description Description Description 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	PAJMISDOXMOLWMDLOXNMCAwnjmodkjondxn	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

Task Management System Entity Relationship Diagram

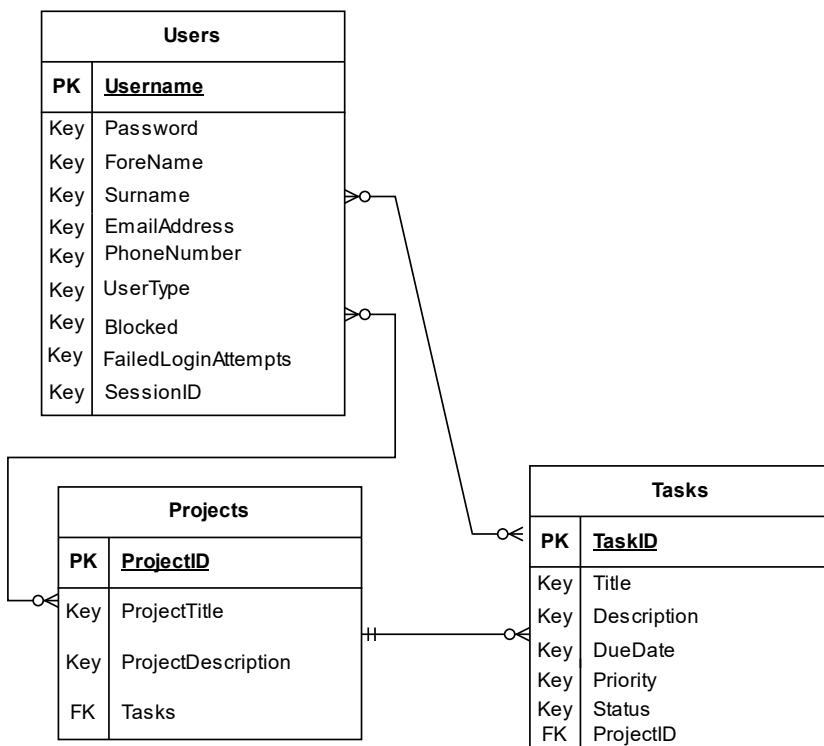


Figure 70 - Entity Relationship Diagram for Task Management System Database.

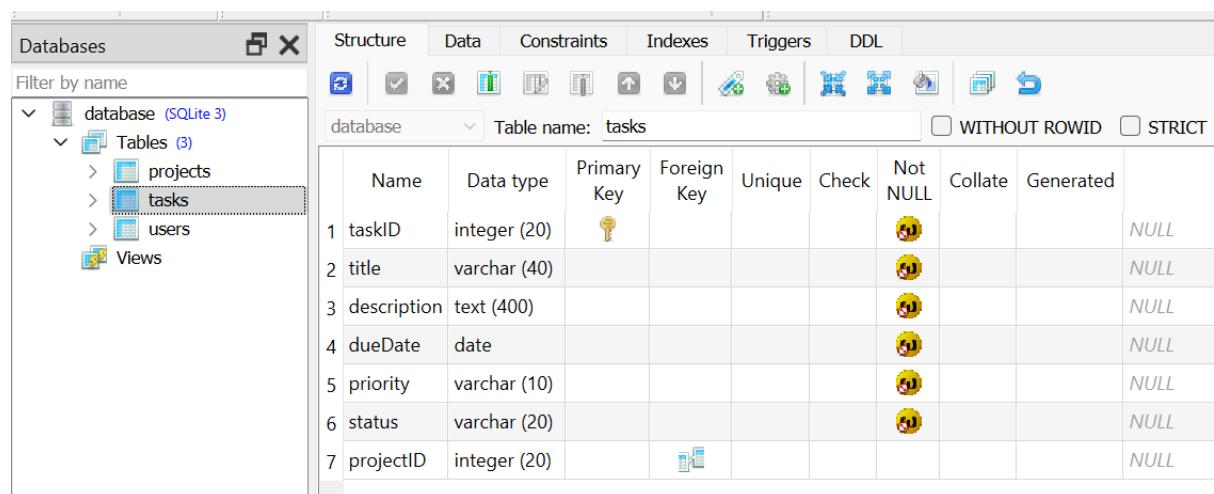


Figure 71 - Database structure viewed in SQLite Studio (tasks table)

S275931

The screenshot shows the SQLite Database Browser interface. The left sidebar lists databases and tables. Under 'Tables (3)', the 'projects' table is selected, indicated by a dotted border. The main area displays the table structure with columns: Name, Data type, Primary Key, Foreign Key, Unique, Check, Not NULL, Collate, and Generated. The 'Not NULL' column contains icons representing NULL (yellow smiley face) or NOT NULL (red frowny face). The 'Generated' column has 'NULL' entries. The 'Collate' column is empty.

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated
1	projectID	integer (20)	key				NULL		NULL
2	title	varchar (40)					w		NULL
3	description	text (400)					w		NULL
4	tasks	text (400)							NULL

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

The screenshot shows the SQLite Database Browser interface. The left sidebar displays the database structure with a tree view: 'database (SQLite 3)' containing 'Tables (3)' with entries 'projects', 'tasks', and 'users'. Below 'Tables' is a 'Views' section. The top navigation bar includes tabs for Structure, Data, Constraints, Indexes, Triggers, and DDL. Below the tabs is a toolbar with various icons. A search bar at the top right allows filtering by table name ('Table name: users') and includes checkboxes for 'WITHOUT ROWID' and 'STRICT'. The main area is a grid representing the 'users' table structure:

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated	
1	username	varchar (40)	key icon				error icon			NULL
2	password	varchar (40)					error icon			NULL
3	forename	varchar (40)					error icon			NULL
4	surname	varchar (40)					error icon			NULL
5	email	varchar (50)					error icon			NULL
6	phone	varchar (20)					error icon			NULL
7	userType	varchar (20)					error icon			NULL
8	blocked	VARCHAR (3)					error icon			NULL
9	failedLoginAttempt	INTEGER (1)					error icon			NULL
10	sessionID	VARCHAR (10)					error icon			NULL

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
3	3	Update Project 3	Update Project 3	task 1, task 2, task 3
4	5	project xxxxxxxx	This is project xxxx	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	Task 1, Task 2, Task 3, Task 4
20	1000000000	project gggg	Project Project project	t1, t2, t3

Figure 75 - Projects table content viewed in SQLite Studio.

	username	password	forename	surname	email	phone	userType	blocked	failedLogin	sessionID
1	Manchester000	491a0812d10f16546ca45ca12610224014028a204abe50f4c66dec0d4f6ea40f7ca8b3a8079aa08f4982...	Kane	Fenton	kfenlon@hotmail.com	42342342	Standard	No	0	0
2	L.FenlonVolcano	7aeffb087d2cb29efc3cf7651bcfab3f3e6cd059b7ae0d4c614e26407aeb19688062fcdb...	Lauren	Durury	ldurury@msn.com	234223424	Standard	No	0	0
3	GemmaWheels	4fde2fa1fa73742236b05d5abcb7d3ce58ae9612903e2bfa13e6614d5940ea8e7fe8465af1c24cd5a03b681cab7c...	Gemma	Bird	gbird@hotmail.com	2323466	Standard	No	0	0
4	kellyG	7aeffb087d2cb29efc3cf7651bcfab3f3e6cd059b7ae0d4c614e26407aeb0934fd070bbe148b19688062fcdb...	Kelly	Gates	kgates@outlook.com	78624668243	Standard	No	0	0
5	ethomas123	b48297422c02d0f05479bb9fdffdf121aaee6450b88533e7e023459bdfbd2f89ca9a7d42d6638a5a2ada1b02c1eb...	Ellie	Thomas	ethomas@icloud.com	836446834	Standard	No	0	0
6	ArasakaCEO123	2b8b3eeaca8d3bf644d493c3fd3d882700ee4d5c551e6076548188492ff6e3434748a02f233db6395b75175...	Bill	Thomas	bthomas@hotmail.com	01243476876	Standard	No	0	0
7	DandyLion000	295da93c654e2b52da7b9f97045027b39250b58accac07dd3d17461dd97463a6fb22ad704cc0ea799707b...	Daisy	Robinson	drobinson@icloud.com	9835973538	Administrator	No	0	0
8	elasticbands123	30a56dc2d1717646fb0bdc8b3895a37f0c8b145067236eb1bbdd0785a1d0e1e23df94c34d564d42625e64...	Sophie	Rivett	srivett@outlook.com	387548735	Standard	No	0	0
9	FlowerMeadow111	b5746f5ec139b2be8b54892da70031c44ce65a961200662d2204a8eeea2158bce4ed0163d10f01089a040...	Daisy	Yau	dyau@icloud.com	3546653476	Standard	No	0	0
10	FrankRivett3000	ac8c86fd77f852d1bba6d674e74752213fbf1333cc2cbb80c8ff8f67372673a48c290f517cd50507841552e35d...	Frank	Rivett	frivett@hotmail.com	93875937457	Standard	No	0	0
11	aGillick999	66f10705e970ca9a970f532d0fb4e1f2e2fb8e6664b9008017c7daac6952cfdfe73417b88870170f0ea84e4f1c...	Alex	Gillick	agillick@outlook.com	84428749287	Administrator	No	0	0
12	Birmingham888	583090a35a8383150907124359e93a9105d07740649812a2f3417876a57526c4526b679102b2d020834...	Linda	Bone	lbone97@hotmail.co.uk	0392743979	Standard	No	0	0
13	Root	bee44f1856e95d6de86a5ew8fd13b392b7f14e519aca7abcbe584343a4dd1ebac2a2b25368932a016c1187268f...	Alexandra	Robinson	30903853809	Administrator	No	0	9666523499	
14	barneyPurpleDinosaur	98e2ec6325b2bea65fad829be16e2c358007f3b0a216c45b7ffbab9085c3f7d87c91d63cbc7b3d67a823f5074c...	Alice	Robinson	robertson97@icloud.com	8734793738	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: <ol style="list-style-type: none"> 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)

<p>Process: Business Modelling → Data Modelling → Process Modelling → Application Generation → Testing and Turnover.</p>		
<p>Advantages:</p> <ul style="list-style-type: none"> • 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 <p>Disadvantages:</p> <ul style="list-style-type: none"> • 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) 	<p>Advantages:</p> <ul style="list-style-type: none"> • 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) <p>Disadvantages:</p> <ul style="list-style-type: none"> • 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) 	<p>Advantages:</p> <ul style="list-style-type: none"> • Efficient • Lower Risk • Enhanced Testing • Improved Quality • Improved Documentation <p>Disadvantages:</p> <ul style="list-style-type: none"> • 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) <p>Advantages:</p> <ul style="list-style-type: none"> • Good at-risk handling • Good for large projects • Customer satisfaction • Iterative and Incremental Approach • Focus on risk management • Improved quality • Improved communication <p>Disadvantages:</p> <ul style="list-style-type: none"> • 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: <ul style="list-style-type: none">• 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: <ul style="list-style-type: none">• Faster development and production time• Lower risk• More efficiency through automation	Disadvantages: <ul style="list-style-type: none">• Difficult to scale on large projects• Requires experienced team members with a lot of training	<ul style="list-style-type: none">• Advantages: Prepare software fast• Easy to make changes if requirements change or mistakes are made• Iterations provide risk handling support and reduce risk

<ul style="list-style-type: none"> Improved collaboration Easy to adapt changes if requirements change and fix mistakes <p>(Bigelow, 2022) (Altynpara, 2023)</p>	<ul style="list-style-type: none"> Only suitable for small teams – larger teams should use DevOps. Difficult to integrate with waterfall methodologies Requires radical changes within the organisation <p>(Vasiliauskas, 2023) (OS System, 2020) (Altynpara, 2023)</p>	<ul style="list-style-type: none"> Flexibility with scope Less expensive compared to other development models Transparency with client Easy to detect and identify errors •
Disadvantages: <ul style="list-style-type: none"> Expensive More complex DevOps can fail when goals are unrealistic Lack of performance metrics (Bigelow, 2022) (Altynpara, 2023) 		Disadvantages: <ul style="list-style-type: none"> 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 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.

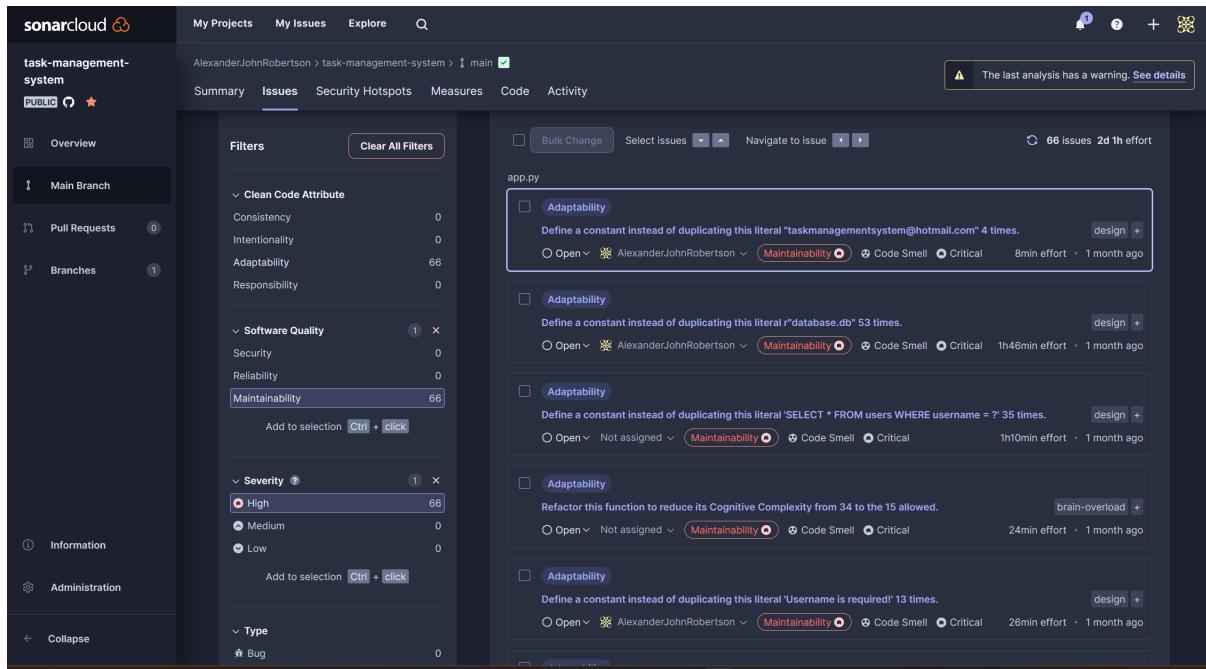


Figure 77 - Maintainability issues detected by SonarCloud during automated system testing - this is a sign of a Big Ball of Mud.

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.

```

248     flash('Password must contain at least one uppercase letter!')
249     elif not checkNumber:
250         flash('Password must contain at least one number!')
251     elif not checkSpecialChar:
252         flash('Password must contain at least one special character!')
253     elif len(password) < 10: # minimum password length is 10 characters
254         flash('Password must be at least 10 characters long!')
255     elif password == username:
256         flash('Password cannot be the same as the username!')
257     elif password == firstname:
258         flash('Password cannot be the same as the first name!')
259     elif password == lastname:
260         flash('Password cannot be the same as the last name!')
261     elif password == email:
262         flash('Password cannot be the same as the email!')
263     elif password == phonenumbers:
264         flash('Password cannot be the same as the phone number!')
265     elif password in global_var2(): #if the password is in the common passwords list flash an error message
266         flash('Password is too common!')
267     else:
268         try:
269             database = r"database.db" #database file
270             conn = None
271             conn = sqlite3.connect(database) #connecting to the database
272             cur = conn.cursor()
273             cur.execute(_sql: 'SELECT * FROM users WHERE username = ?', __parameters: (username,)) #querying the database
274             user = cur.fetchall()
275             cur.close()
276             if user: #if the username already exists flash an error message
277                 flash('Username already exists!')
278             else:
279                 if password != confirmPassword: #if the password and confirm password do not match flash an error message
280                     flash('Passwords do not match!')
281                 else:
282                     for i in range(10): #password hashing encryption
283                         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

```

211 @app.route(rule: '/createaccount', methods=['GET', 'POST'])
212 def createaccount():
213     '''This function allows the user to create an account.'''
214     if request.method == 'POST': #get form details
215         username = request.form['username']
216         password = request.form['password']
217         confirmPassword = request.form['confirmPassword']
218         firstname = request.form['firstname']
219         lastname = request.form['lastname']
220         email = request.form['email']
221         phonenumer = request.form['phonenumer']
222         accountType = "Standard" #set the account type to standard
223         blocked = "No" #set the blocked status to no
224         failedLoginAttempts = 0 #set the failed login attempts to 0
225         checkLowercase = re.search(pattern: r'[a-z]', password) #use regular expressions to validate the password
226         checkUppercase = re.search(pattern: r'[A-Z]', password)
227         checkNumber = re.search(pattern: r'[0-9]', password)
228         checkSpecialChar = re.search(pattern: r'[^A-Za-z0-9]', password)
229         if not firstname: #validate the form details
230             flash('First Name is required!')
231         elif not lastname:
232             flash('Last Name is required!')
233         elif not username:
234             flash('Username is required!')
235         elif not email:
236             flash('Email Address is required!')
237         elif not phonenumer:
238             flash('Phone Number is required!')
239         elif not username:
240             flash('Username is required!')
241         elif not password:
242             flash('Password is required!')
243         elif not confirmPassword:
244             flash('Confirm Password is required!')
245         elif not checkLowercase: #validate the password
246             flash('Password must contain at least one lowercase letter!')
247         elif not checkUppercase:

```

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

```

284         password = password.hexdigest()
285         sessionID = random.randint(a: 1000000000, b: 9999999999) #generate a random session ID
286         cur = conn.cursor()
287         cur.execute(sql: INSERT INTO users VALUES(?, ?, ?, ?, ?, ?, ?, ?, ?), parameters: (username, password, firstname, lastname,
288         conn.commit())
289         cur.close()
290         resp = make_response(render_template('readcookie.html')) #setting the cookie connected to the username
291         resp.set_cookie(key: 'userID', username)
292         resp.set_cookie(key: 'sessionID', str(sessionID))
293         return resp
294     except IndexError: #if there is an error caused by incorrect login details flash an error message
295         flash('Username or Password is incorrect!')
296     return render_template('createaccount.html') #render the createaccount.html page
297

```

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

```

10     <div class="wpcf7" id="wpcf7-f156-p143-o1 formwrap">
11     <form method="post" class="wpcf7-form novalidate" novalidate="novalidate"> <!-- Form for the user to input the task details -->
12     <div style="display: none;">
13         <input type="hidden" name="_wpcf7" value="156">
14         <input type="hidden" name="_wpcf7_version" value="3.7.2">
15         <input type="hidden" name="_wpcf7_locale" value="en_US">
16         <input type="hidden" name="_wpcf7_unit_tag" value="wpcf7-f156-p143-o1">
17         <input type="hidden" name="_wpnonce" value="d1da531d93">
18     </div>
19     <p>
20         <span class="wpcf7-form-control-wrap TaskID">
21             <label class="text1">Task ID:</label>
22             <input type="number" name="taskid" value="" size="40" class="textinput wpcf7-form-control wpcf7-text wpcf7-validates-as-required" aria-required="true">
23         </span>
24         <span class="wpcf7-form-control-wrap Title">
25             <label class="text1">Title:</label>
26             <input type="text" name="title" value="" size="40" class="textinput wpcf7-form-control wpcf7-text wpcf7-validates-as-required" aria-required="true">
27         </span>
28         <span class="wpcf7-form-control-wrap description">
29             <label class="text1"><u>Description

```

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.

```

/* 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 redeloys 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:

```

Terminal Local × + ∨
PS C:\Users\rober\OneDrive\Documents\Advanced_Computing_Suffolk\Advanced Software Engineering\Assignment\TaskManagementSystem2> git add .
PS C:\Users\rober\OneDrive\Documents\Advanced_Computing_Suffolk\Advanced Software Engineering\Assignment\TaskManagementSystem2> git commit -m "new commit"
[main fa579bf] new commit
 2 files changed, 1 insertion(+), 1 deletion(-)
PS C:\Users\rober\OneDrive\Documents\Advanced_Computing_Suffolk\Advanced Software Engineering\Assignment\TaskManagementSystem2> git push origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 32 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 493 bytes | 493.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To https://github.com/AlexanderJohnRobertson/task-management-system.git
   Sabbd424..fa579bf main -> main
PS C:\Users\rober\OneDrive\Documents\Advanced_Computing_Suffolk\Advanced Software Engineering\Assignment\TaskManagementSystem2>
  
```

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

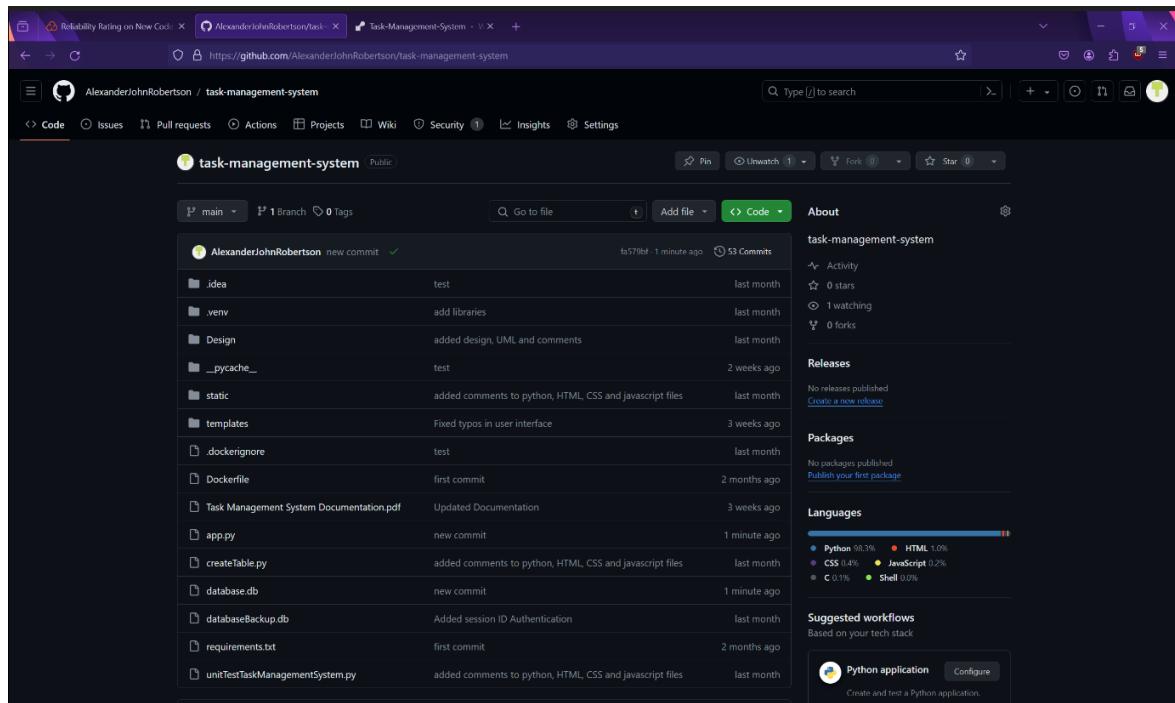


Figure 84 - The updated GitHub repository.

The screenshot shows the Render.com dashboard for a project named "Task-Management-System". The top navigation bar includes links for Dashboard, Blueprints, Env Groups, Docs, Community, Help, and a "New +" button. A user profile for Alexander Robertson is shown on the right. Below the header, there's a section for the GitHub repository "AlexanderJohnRobertson / task-management-system" with a "main" branch, a "Connect" button, and a "Manual Deploy" dropdown.

The main area displays a timeline of events:

- Events:** April 16, 2024 at 9:27 PM - "fa579bf new commit" (status: "in progress")
- Logs:** A detailed log viewer showing command-line output for the build process. Key logs include:
 - Apr 16 09:28:03 PM Using cached packaging-24.0-py3-none-any.whl (53 kB)
 - Apr 16 09:28:03 PM Using cached MarkupSafe-2.1.5-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (28 kB)
 - Apr 16 09:28:03 PM Installing collected packages: packaging, MarkupSafe, itsdangerous, click, blinker, Werkzeug, Jinja2, gunicorn, Flask
 - Apr 16 09:28:04 PM Successfully installed Flask-3.0.3 Jinja2-3.1.3 MarkupSafe-2.1.5 Werkzeug-3.0.2 blinker-1.7.0 click-8.1.7 gunicorn-21.2.0 itsdangerous-2.1.2 packaging-24.0
 - Apr 16 09:28:04 PM [notice] A new release of pip is available: 23.2.1 -> 24.0
 - Apr 16 09:28:04 PM [notice] To update, run: pip install --upgrade pip
 - Apr 16 09:28:09 PM ==> Uploading build...
 - Apr 16 09:28:27 PM ==> Deploying...
 - Apr 16 09:28:24 PM ==> Build uploaded in 9s
 - Apr 16 09:28:24 PM ==> Build successful 🎉
- Disks:** Environment, Shell, Previews, Jobs, Metrics, Scaling, Settings sections are visible on the left.

At the bottom, there are links for Feedback, Invite a Friend, and Contact Support.

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

This screenshot shows a detailed log viewer for the same CI/CD pipeline. The interface includes a "All logs" dropdown, a search bar, and various filter and control buttons (Live tail, GMT+1, etc.).

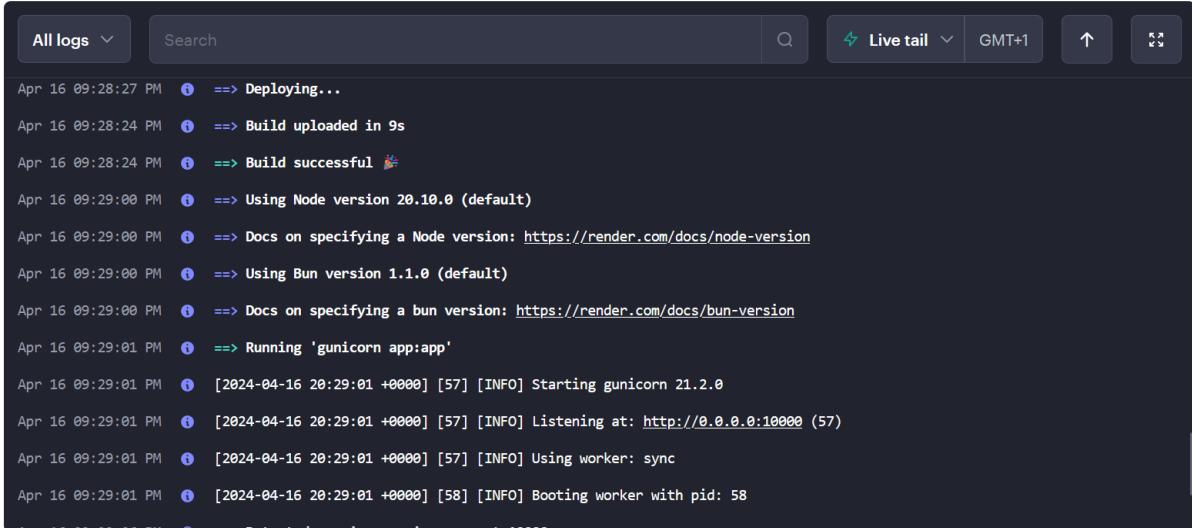
The log output is identical to Figure 85, showing the step-by-step process of the build and deployment:

```

Apr 16 09:28:03 PM ① Using cached packaging-24.0-py3-none-any.whl (53 kB)
Apr 16 09:28:03 PM ① Using cached MarkupSafe-2.1.5-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (28 kB)
Apr 16 09:28:03 PM ① Installing collected packages: packaging, MarkupSafe, itsdangerous, click, blinker, Werkzeug, Jinja2, gunicorn, Flask
Apr 16 09:28:04 PM ① Successfully installed Flask-3.0.3 Jinja2-3.1.3 MarkupSafe-2.1.5 Werkzeug-3.0.2 blinker-1.7.0 click-8.1.7 gunicorn-21.2.0
itsdangerous-2.1.2 packaging-24.0
Apr 16 09:28:04 PM ① [notice] A new release of pip is available: 23.2.1 -> 24.0
Apr 16 09:28:04 PM ① [notice] To update, run: pip install --upgrade pip
Apr 16 09:28:09 PM ① ==> Uploading build...
Apr 16 09:28:27 PM ① ==> Deploying...
Apr 16 09:28:24 PM ① ==> Build uploaded in 9s
Apr 16 09:28:24 PM ① ==> Build successful 🎉

```

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



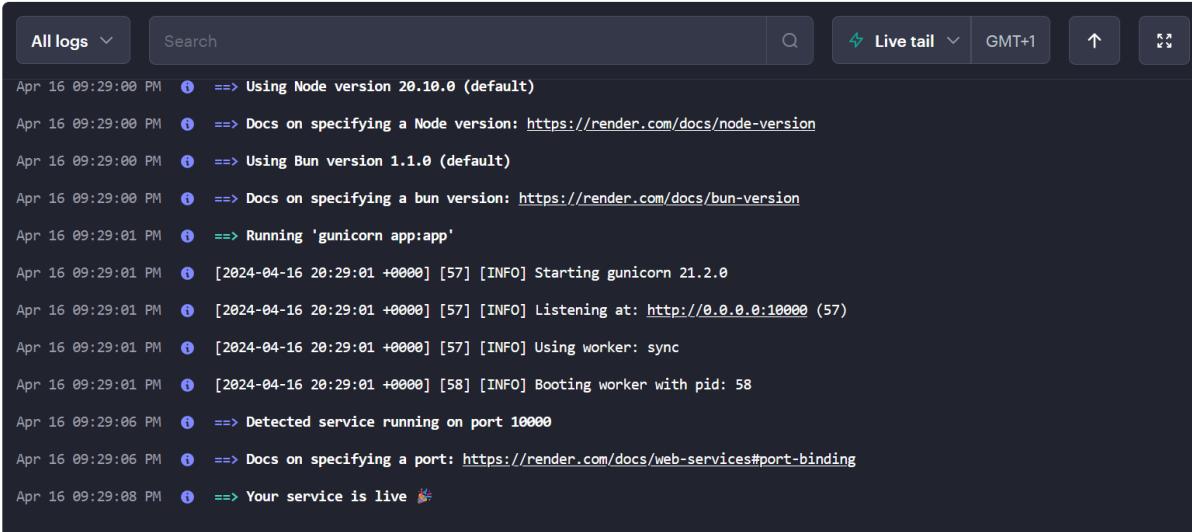
The screenshot shows a log viewer interface with the following log entries:

```

Apr 16 09:28:27 PM  i  ==> Deploying...
Apr 16 09:28:24 PM  i  ==> Build uploaded in 9s
Apr 16 09:28:24 PM  i  ==> Build successful 🎉
Apr 16 09:29:00 PM  i  ==> Using Node version 20.10.0 (default)
Apr 16 09:29:00 PM  i  ==> Docs on specifying a Node version: https://render.com/docs/node-version
Apr 16 09:29:00 PM  i  ==> Using Bun version 1.1.0 (default)
Apr 16 09:29:00 PM  i  ==> Docs on specifying a bun version: https://render.com/docs/bun-version
Apr 16 09:29:01 PM  i  ==> Running 'gunicorn app:app'
Apr 16 09:29:01 PM  i  [2024-04-16 20:29:01 +0000] [57] [INFO] Starting gunicorn 21.2.0
Apr 16 09:29:01 PM  i  [2024-04-16 20:29:01 +0000] [57] [INFO] Listening at: http://0.0.0.0:10000 (57)
Apr 16 09:29:01 PM  i  [2024-04-16 20:29:01 +0000] [57] [INFO] Using worker: sync
Apr 16 09:29:01 PM  i  [2024-04-16 20:29:01 +0000] [58] [INFO] Booting worker with pid: 58

```

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



The screenshot shows a log viewer interface with the following log entries, indicating the task management system is now live:

```

Apr 16 09:29:00 PM  i  ==> Using Node version 20.10.0 (default)
Apr 16 09:29:00 PM  i  ==> Docs on specifying a Node version: https://render.com/docs/node-version
Apr 16 09:29:00 PM  i  ==> Using Bun version 1.1.0 (default)
Apr 16 09:29:00 PM  i  ==> Docs on specifying a bun version: https://render.com/docs/bun-version
Apr 16 09:29:01 PM  i  ==> Running 'gunicorn app:app'
Apr 16 09:29:01 PM  i  [2024-04-16 20:29:01 +0000] [57] [INFO] Starting gunicorn 21.2.0
Apr 16 09:29:01 PM  i  [2024-04-16 20:29:01 +0000] [57] [INFO] Listening at: http://0.0.0.0:10000 (57)
Apr 16 09:29:01 PM  i  [2024-04-16 20:29:01 +0000] [57] [INFO] Using worker: sync
Apr 16 09:29:01 PM  i  [2024-04-16 20:29:01 +0000] [58] [INFO] Booting worker with pid: 58
Apr 16 09:29:06 PM  i  ==> Detected service running on port 10000
Apr 16 09:29:06 PM  i  ==> Docs on specifying a port: https://render.com/docs/web-services#port-binding
Apr 16 09:29:08 PM  i  ==> Your service is live 🎉

```

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

S275931

The screenshot shows the Render.com dashboard for the 'Task-Management-System' application. At the top, there's a header with the application name, Python 3, Starter, Connect, and Manual Deploy buttons. Below the header, it shows the GitHub repository information: AlexanderJohnRobertson / task-management-system, main branch, and the URL https://task-management-system-hnmb.onrender.com. A sidebar on the left lists various monitoring and configuration options: Events, Logs, Disks, Environment, Shell, Previews, Jobs, Metrics, Scaling, and Settings. The main area displays a log viewer titled 'All logs' with a 'Live tail' button. It shows log entries from April 16, 2024, at 9:27 PM, indicating the service is live and listening on port 10000.

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

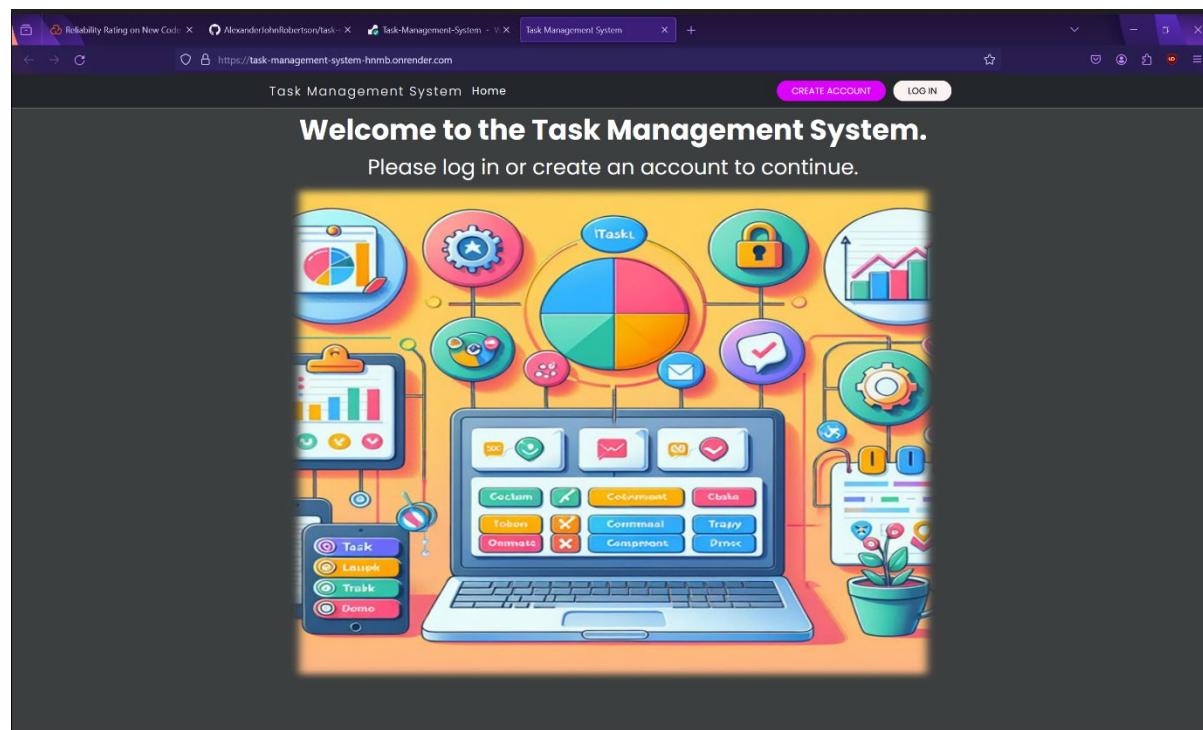


Figure 90 - Task Management System live landing page on Render.com.

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.

The screenshot shows the DockerHub interface for a repository named `alexjrobertson/taskmanagementsystem`. The repository is described as `General`, and its last update was 23 days ago. It has no description or category. The Docker commands section shows the command to push a new tag: `docker push alexjrobertson/taskmanagementsystem:tagname`. The Tags section lists one tag, `latest`, which is an image type pulled 22 days ago. The Automated Builds section explains that manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating. This feature is available with Pro, Team and Business subscriptions.

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

Automated Builds

Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.

Available with Pro, Team and Business subscriptions. [Read more about automated builds](#).

[Upgrade](#)

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

Plan	Price	Features
Personal	\$0	<ul style="list-style-type: none"> Docker Desktop Personal Unlimited public repositories Docker Engine + Kubernetes 200 image pulls per 6 hours 3 Scout-enabled repositories Local Scout analysis
Pro	\$5 / month	<p>Everything in Personal, plus:</p> <ul style="list-style-type: none"> Docker Desktop Pro Unlimited private repositories 5,000 image pulls per day 5 concurrent builds Synchronized file shares Local Scout analysis and remediation 5 day support response
Team	\$9 / seat / month	<p>Everything in Pro, plus:</p> <ul style="list-style-type: none"> Docker Desktop Team Unlimited teams 15 concurrent builds Add users in bulk Audit logs Up to 100 seats Role-based access control 2 day support response
Business	\$24 / seat / month	<p>Everything in Team, plus:</p> <ul style="list-style-type: none"> Docker Desktop Business Hardened Docker Desktop Single Sign-On (SSO) SCIM user provisioning VDI support Private Extensions Marketplace Image and Registry Access Management Purchase via invoice Priority case routing Proactive case monitoring 24-hour support response

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

Name	Tag	Status	Created	Size	Actions
alexrobertson/taskmanagementsystem	latest	In use	23 days ago	129.76 MB	▶ ⋮ 🔗
alexrobertson/taskmanagementsystemcleaninstall	latest	In use	23 days ago	129.4 MB	▶ ⋮ 🔗
<none>	<none>	In use (dangling)	23 days ago	129.4 MB	▶ ⋮ 🔗
<none>	<none>	In use (dangling)	23 days ago	129.76 MB	▶ ⋮ 🔗
<none>	<none>	In use (dangling)	25 days ago	106.81 MB	▶ ⋮ 🔗
<none>	<none>	In use (dangling)	25 days ago	107.17 MB	▶ ⋮ 🔗

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

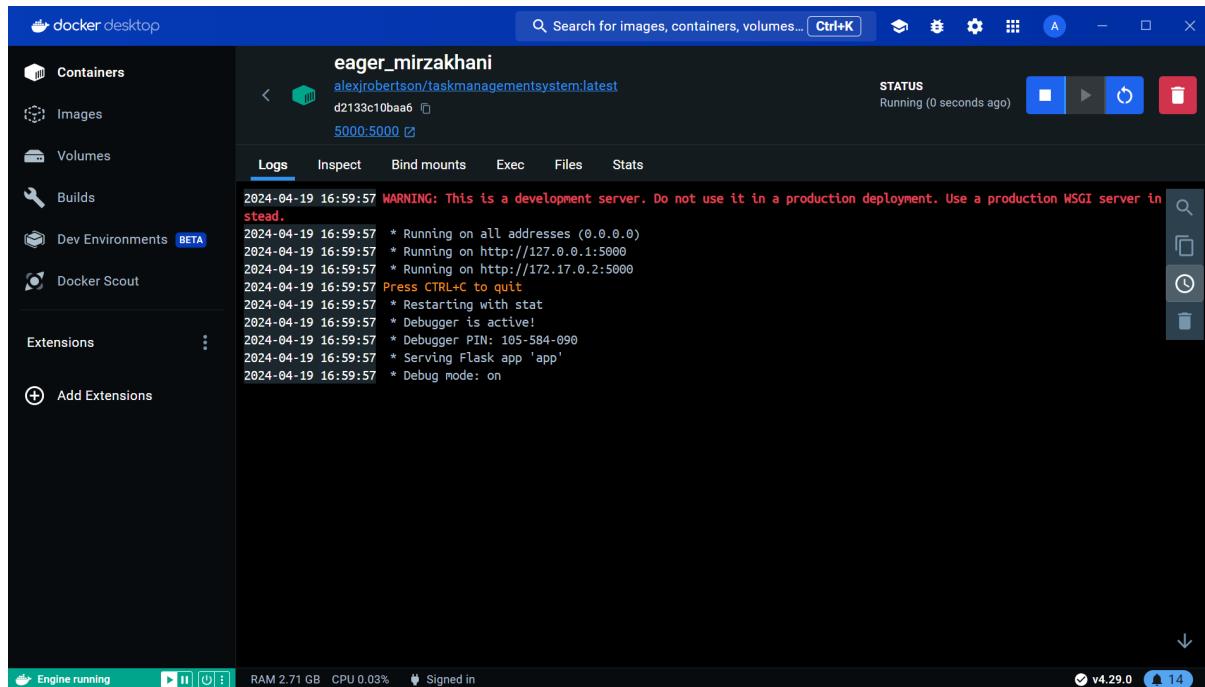


Figure 95 - Task Management System container deployed and running on Docker Desktop.

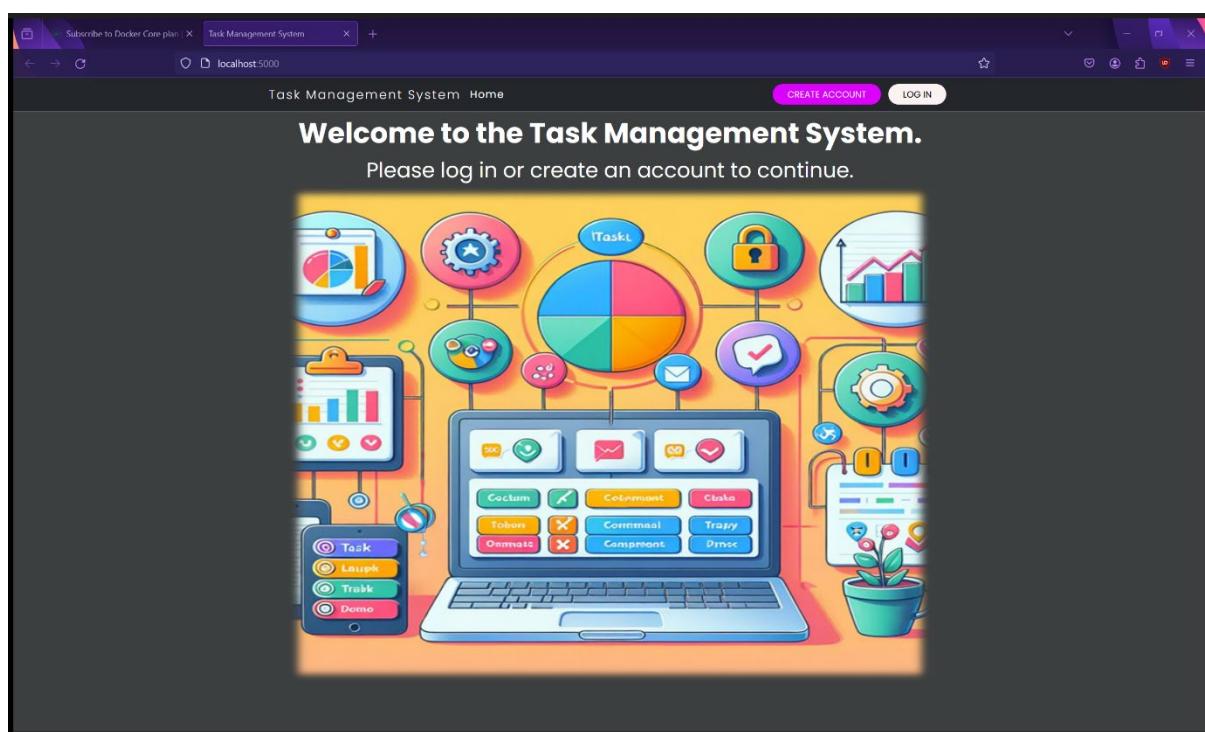
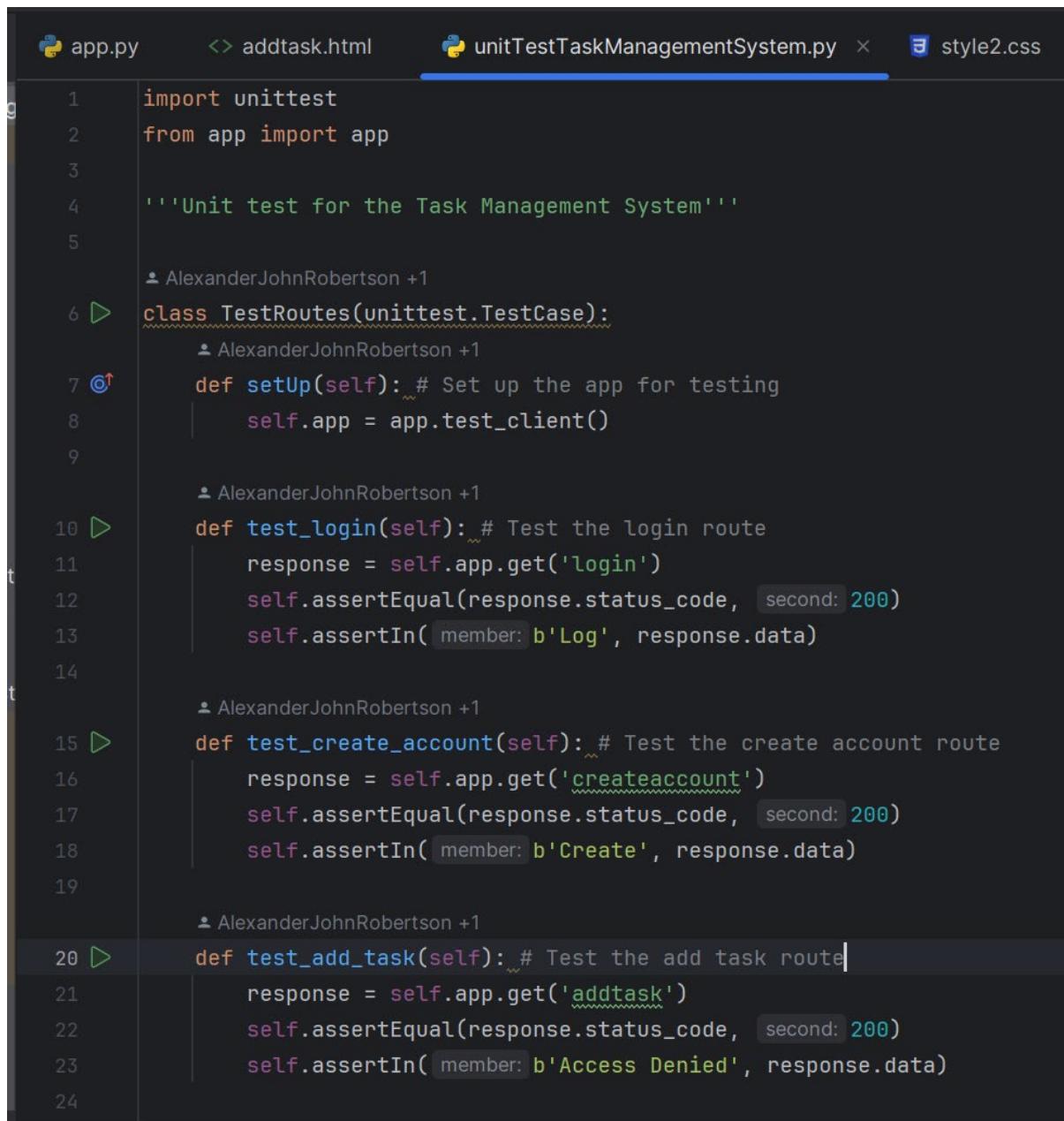


Figure 96 - Task Management System landing page accessible through a web browser from the Docker container.

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 work. 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
1 import unittest
2 from app import app
3
4 '''Unit test for the Task Management System'''
5
6 class TestRoutes(unittest.TestCase):
7     def setUp(self): # Set up the app for testing
8         self.app = app.test_client()
9
10    def test_login(self): # Test the login route
11        response = self.app.get('login')
12        self.assertEqual(response.status_code, 200)
13        self.assertIn(b'Log', response.data)
14
15    def test_create_account(self): # Test the create account route
16        response = self.app.get('createaccount')
17        self.assertEqual(response.status_code, 200)
18        self.assertIn(b'Create', response.data)
19
20    def test_add_task(self): # Test the add task route
21        response = self.app.get('addtask')
22        self.assertEqual(response.status_code, 200)
23        self.assertIn(b'Access Denied', response.data)
24

```

Figure 97 - Unit Testing Python source code.

```
Run Python tests in unitTestTaskManagementSystem.py
Test Results 42 ms
✓ Tests passed: 22 of 22 tests - 42 ms
"C:\Users\rober\OneDrive\Documents\Advanced_Computing_Suffolk\Advanced Software Engineering\Assignment\TaskManagementSystem2\.venv\Scripts\python.exe" "C:/U
Testing started at 16:16 ...
Could not find platform independent libraries <prefix>
Launching unittests with arguments python -m unittest C:\Users\rober\OneDrive\Documents\Advanced_Computing_Suffolk\Advanced Software Engineering\Assignment\

Ran 22 tests in 0.054s
OK
Process finished with exit code 0
```

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.

```
app.py      addtask.html      unitTestTaskManagementSystem.py      style2.css
g
6   class TestRoutes(unittest.TestCase):
22     self.assertEqual(response.status_code, second: 200)
23     self.assertIn( member: b'Access Denied', response.data)
24
25     ▶ def test_view_task(self): # Test the view task route
26       response = self.app.get('viewtasks')
27       self.assertEqual(response.status_code, second: 200)
28       self.assertIn( member: b'Access Denied', response.data)
29
30     ▶ def test_update_task(self): # Test the update task route
31       response = self.app.get('updatetask')
32       self.assertEqual(response.status_code, second: 200)
33       self.assertIn( member: b'Abcdefg Denied', response.data)
34
35     ▶ def test_delete_task(self): # Test the delete task route
36       response = self.app.get('deletetask')
37       self.assertEqual(response.status_code, second: 200)
38       self.assertIn( member: b'Access Denied', response.data)
```

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

```

Run Python tests in unitTestTaskManagementSystem.py ...
Test Results: 26ms
  ✓ Test Results: 26ms
    ✓ unitTestTaskManagementSystem: 26ms
      ✓ TestRoutes: 26ms
        ✘ test_Update.task: 6ms
          Tests failed: 1, passed: 21 of 22 tests - 26 ms
          "C:\Users\rober\OneDrive\Documents\Advanced_Computing_Suffolk\Advanced Software Engineering\Assignment\TaskManagementSystem2\.venv\Scripts\python.exe" "C:
          Testing started at 16:24 ...
          Could not find platform independent libraries <prefix>
          Launching unittests with arguments python -m unittest C:\Users\rober\OneDrive\Documents\Advanced_Computing_Suffolk\Advanced Software Engineering\Assignment\TaskManagementSystem2\unitTestTaskManagementSystem.py
          Failure
          Traceback (most recent call last):
            File "C:\Users\rober\OneDrive\Documents\Advanced_Computing_Suffolk\Advanced Software Engineering\Assignment\TaskManagementSystem2\unitTestTaskManagementSystem.py", line 1, in <module>
              self.assertIn(b'Abcdefg Denied', response.data)
          AssertionError: b'Abcdefg Denied' not found in b'<!-- HTML source code for Access Denied error page -->\n<!DOCTYPE html>\n<html lang="en">\n<head>\n<title>Access Denied</title>\n</head>\n<body>\n<h1>Access Denied</h1>\n<p>You do not have permission to view this page.</p>\n</body>\n</html>'

          Ran 22 tests in 0.039s
          FAILED (failures=1)

          Process finished with exit code 1

```

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.

```

Task_Management_System / Landing Page
GET http://127.0.0.1:5000/
Params Authorization Headers (7) Body Scripts Settings
Pre-request
1 pm.test("Response status code is 200", function () {
2   pm.expect(pm.response.code).to.equal(200);
3 });
4
5
6 pm.test("Title element is present in the HTML response", function () {
7   pm.expect(pm.response.text()).to.include('<title>');
8 });
9
10
11 pm.test("Response time is less than 200ms", function () {
12   pm.expect(pm.response.responseTime).to.be.below(200);
13 });
14
15
16 pm.test("Response Content-Type header is 'text/html'", function () {
17   pm.expect(pm.response.headers.get("Content-Type")).to.include("text/html");
18 });
19
20 var template = '';
Body Cookies (2) Headers (5) Test Results (4/4)
Pretty Raw Preview Visualize HTML
<-- HTML source code for index landing page -->
<!DOCTYPE html>
<html lang="en">
<link rel="stylesheet" href="/static/style2.css">!-- Link to the CSS file to style webpage -->
<head>
<meta charset="UTF-8">
<title>Task Management System</title>!-- Title of the webpage -->
</head>
<body>
<!-- HTML source code for base3 page for responsive navbars and basic webpage when logged out -->
<!-- Original HTML based on codepen.io template and modified by S275931 to suit the application, https://codepen.io/GilaniRabbu/ -->
<!DOCTYPE html>
<html lang="en">

```

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.

The screenshot shows the Postman interface with a collection named 'Task_Management_System'. A GET request is made to 'http://127.0.0.1:5000/' with a status of 200 OK. The response body contains the HTML code for a landing page, which is displayed in the 'Preview' tab. A context menu is open over the preview area, with the 'Visualize response as bar graph' option highlighted.

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

Element	Occurrence
Title	1
HTML source code	1
Redirect to home page	2

The screenshot shows the Postman interface with the 'Visualize' tab selected. The response body is visualized as a table with three rows: 'Title', 'HTML source code', and 'Redirect to home page'. The 'Occurrence' column indicates the count of each element found in the response. The status is 200 OK.

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

```

pm.test("Response status code is 200", function () {
    pm.expect(pm.response.code).to.equal(200);
});

pm.test("Title element is present in the HTML response", function () {
    pm.expect(pm.response.text()).to.include("title");
});

pm.test("Response time is less than 200ms", function () {
    pm.expect(pm.response.responseTime).to.be.below(200);
});

pm.test("Response Content-Type header is 'text/html'", function () {
    pm.expect(pm.response.headers.get("Content-Type")).to.include("text/html");
});

```

Test Results: 4/4

Test	Status
Response status code is 200	PASS
Title element is present in the HTML response	PASS
Response time is less than 200ms	PASS
Response Content-Type header is 'text/html'	PASS

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

```

var template =
<style type="text/css">
    .titable {font-size:14px;color:#333333;width:100%;border-width: 1px; border-color: #07ceeb; border-collapse: collapse;}
    .titable th {font-size:18px;background-color:#07ceeb; border-width: 1px;padding: 8px; border-style: solid; border-color: #07ceeb; text-align:left;}
    .titable tr {background-color:#ffffff;}
    .titable td {font-size:14px; border-width: 1px;padding: 8px; border-style: solid; border-color: #07ceeb; }
    .titable tr:hover {background-color:#e0ffff;}
</style>






```

Test Results: 4/5

Test	Status
Access Denied error page	Pretty
Access Denied error page	Raw
Access Denied error page	Preview
Access Denied error page	Visualize
Access Denied error page	HTML

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.

The screenshot shows the Postman interface with a collection named 'Task_Management_System'. A GET request is made to 'http://127.0.0.1:5000/userhome/'. The response body contains the following HTML code:

```

var template =
```
<style type="text/css">
 .ttable {font-size:14px;color:#333333;width:100%;border-width: 1px; border-color: #07ceeb; border-collapse: collapse;}
 .ttable th {font-size:18px;background-color:#07ceeb; border-width: 1px;padding: 8px; border-style: solid; border-color: #07ceeb; text-align:left;}
 .ttable tr {background-color:#ffffff;}
 .ttable td {font-size:14px; border-width: 1px;padding: 8px; border-style: solid; border-color: #07ceeb; background-color:#e0ffff; }
 .ttable tr:hover {background-color:#e0ffff; }
</style>

<table class="ttable" border="1">
 <tr>
 | <th>Error Message</th>
 </tr>
 <tr>
 | <td>Access Denied. You do not have permission to access the Task Management System.</td>
 </tr>
</table>
```

```

The response status is 200 OK.

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.

The screenshot shows the Postman interface with the same collection and request setup as Figure 106. The response body is visualized as a table with one row containing the message 'Access Denied. You do not have permission to access the Task Management System.'.

Figure 107 - Visualisation of the response.

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' containing various API endpoints for a 'Task_Management_System'. The main area shows a 'User Home' endpoint with a 'GET' request to 'http://127.0.0.1:5000/userhome/'. The 'Post-response' tab contains a script that extracts the response body. Below it, the 'Test Results' section shows four tests: 'Response status code is 200' (PASS), 'Response body contains the required HTML elements' (PASS), 'Response body includes the expected error message | AssertionException: expected 'cl-- HTML source code for Access Deni...' (FAIL), and 'Response time is less than 200ms' (PASS). A tooltip on the right indicates that the last test failed because the response body included the expected error message.

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.

The screenshot shows the SonarCloud interface for the 'task-management-system' project. The left sidebar has sections for 'Overview', 'Main Branch', 'Pull Requests' (0), 'Branches' (1), 'Information', 'Administration', and 'Collapse'. The main area is titled 'Issues' and shows a summary of 392 issues with a total effort of 5d 2h. It includes filters for 'Clean Code Attribute', 'Software Quality', 'Severity' (High, Medium, Low), 'Type', 'Resolution', and 'Status'. On the right, a list of specific issues is shown, each with details like title, severity, assignee, status, and creation date. For example, one issue is 'Don't disclose "Flask" secret keys.' (Severity: High, Status: Open, Created: 30min effort, 2 months ago).

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.

The screenshot shows the SonarCloud interface for a project named 'task-management-system'. The 'Security Hotspots' tab is selected. A prominent warning message at the top right states: "The last analysis has a warning. See details". On the left sidebar, there are sections for Overview, Main Branch, Pull Requests (0), Branches (1), Information, and Administration. The main content area displays a summary of security hotspots: "0.0% Security Hotspots Reviewed" (To review, Fixed, Safe), "50 Security Hotspots to review", and a breakdown by category: Authentication (1), Cross-Site Request Forgery (CSRF) (25), Permission (2), and Weak Cryptography (3). A specific hotspot is highlighted: "'SECRET_KEY' detected here, review this potentially hard-coded credential." This hotspot is categorized as 'Authentication' and 'Medium'. It includes a status bar: "Status: To Review", "Review priority: High", "Category: Authentication", and "Assignee: AlexanderJohnRober...". Below this is a code snippet from 'app.py' with line numbers 19 to 28. Line 23 contains the problematic code: `app.config['SECRET_KEY'] = 'hdry35b1dge65gcs1'`. A callout box highlights this line with the same error message: "'SECRET_KEY' detected here, review this potentially hard-coded credential." At the bottom of the code view, there are tabs for 'Where is the risk?', 'What's the risk?', 'Assess the risk', 'How can I fix it?', and 'Activity'.

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

The screenshot shows the SonarCloud interface for the same project 'task-management-system'. The 'Code' tab is selected. A warning message at the top right says: "The last analysis has a warning. See details". The left sidebar includes sections for Overview, Main Branch, Pull Requests (0), Branches (1), Information, and Administration. The main content area features a search bar and a table of code metrics: Lines of Code, Bugs, Vulnerabilities, Code Smells, Security Hotspots, Coverage, and Duplications. The table lists the following data:

| File/Folder | Lines of Code | Bugs | Vulnerabilities | Code Smells | Security Hotspots | Coverage | Duplications |
|---------------------------------|---------------|------|-----------------|-------------|-------------------|----------|--------------|
| task-management-system | | | | | | | |
| static | 1,048 | 4 | 0 | 1 | 0 | — | 0.0% |
| templates | 1,478 | 34 | 0 | 104 | 0 | — | 22.0% |
| app.py | 1,548 | 1 | 1 | 244 | 48 | — | 37.7% |
| createTable.py | 15 | 0 | 0 | 0 | 0 | — | 0.0% |
| Dockerfile | 9 | 0 | 0 | 0 | 2 | — | 0.0% |
| unitTestTaskManagementSystem.py | 100 | 0 | 0 | 3 | 0 | — | 0.0% |

At the bottom, a note says "6 of 6 shown". The footer includes links for Terms, Pricing, Privacy, Cookie Policy, Security, Community, Documentation, Contact us, Status, and About.

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

The screenshot shows the SonarCloud interface for the project 'task-management-system'. The left sidebar includes sections for Overview, Main Branch, Pull Requests (0), Branches (1), Information, Administration, and Collapse. The main area displays a 'Project Overview' card with Reliability (A), Overview, New Code, Bugs (0), Rating (A), Remediation Effort (0), Overall Code, Bugs (39), Rating (D), and Remediation Effort (3h 8min). Below this is a 'Security' section. On the right, a detailed view of the 'task-management-system' codebase shows a tree structure with 6 files: static, templates, app.py, createTable.py, Dockerfile, and unitTestTaskManagementSystem.py. A warning message at the top right states 'The last analysis has a warning. See details'.

Figure 112 - SonarCloud grading the quality of my code.

This screenshot is similar to Figure 112 but focuses on the 'static' directory. It shows a 'Reliability Rating' of D for the static files. The files listed are createaccountloginformstyle.css, formstyle.css, navbar.js, style.css, style2.css, and tablestyle.css. The same warning about a recent warning is present at the top right.

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.

The screenshot shows the SonarCloud interface for the project 'task-management-system'. The 'Measures' tab is selected. A warning message at the top right states: 'The last analysis has a warning. See details'. The main panel displays the 'Reliability Rating' for the file 'createaccountloginformstyle.css'. The rating is shown as a green circle with an 'A'. Below the rating, the code snippet is displayed:

```

1 146436... /* This file is used to style the create account, login and setup forms */
2 2 /* original CSS sourced from codenode.io and modified by S275931 to suit the application. https://codenode.io/
3 3 andstudio/po/eNwM */
4 4 * {
5 5 box-sizing: border-box;
6 6 } /* Box Sizing */
7 7
8 8 h2 {
9 9   text-align: center;
10 10   margin: 0px;
11 11   color: #444444;
12 12   font-size: 17px;
13 13   font-family: 'Quicksand', sans-serif;
14 14   @media(max-width:540px){
15 15     font-size: 28px;
16 16   margin: 30px 0;
17 17 }
18 18 } /* Heading 2 */
19 19
20 20 form {
21 21   padding: 0 10px;
22 22   position: relative;
23 23   width: calc(50% - 40px);
24 24   padding: 20px;
25 25   background: #e3e3f3;
26 26   margin: 20px;
27 27   border-radius: 5px;
28 28   & .left {
29 29     float: left;
30 30   & .right {
31 31     float: right;
32 32   }
33 33   input-container {
34 34     & .left {
35 35     & .right {
36 36       margin-left: 20px;
37 37     }
38 38   }
39 39 }
40 40 }
41 41 }
42 42 }
43 43 }
44 44 }
45 45 }
46 46 }
47 47 }
48 48 }
49 49 }
50 50 }
51 51 }
52 52 }
53 53 }
54 54 }
55 55 }
56 56 }
57 57 }
58 58 }
59 59 }
60 60 }
61 61 }
62 62 }
63 63 }
64 64 }
65 65 }
66 66 }
67 67 }
68 68 }
69 69 }
70 70 }
71 71 }
72 72 }
73 73 }
74 74 }
75 75 }
76 76 }
77 77 }
78 78 }
79 79 }
80 80 }
81 81 }
82 82 }
83 83 }
84 84 }
85 85 }
86 86 }
87 87 }
88 88 }
89 89 }
90 90 }
91 91 }
92 92 }
93 93 }
94 94 }
95 95 }
96 96 }
97 97 }
98 98 }
99 99 }
100 100 }
101 101 }
102 102 }
103 103 }
104 104 }
105 105 }
106 106 }
107 107 }
108 108 }
109 109 }
110 110 }
111 111 }
112 112 }
113 113 }
114 114 }
115 115 }
116 116 }
117 117 }
118 118 }
119 119 }
120 120 }
121 121 }
122 122 }
123 123 }
124 124 }
125 125 }
126 126 }
127 127 }
128 128 }
129 129 }
130 130 }
131 131 }
132 132 }
133 133 }
134 134 }
135 135 }
136 136 }
137 137 }
138 138 }
139 139 }
140 140 }
141 141 }
142 142 }
143 143 }
144 144 }
145 145 }
146 146 }
147 147 }
148 148 }
149 149 }
150 150 }
151 151 }
152 152 }
153 153 }
154 154 }
155 155 }
156 156 }
157 157 }
158 158 }
159 159 }
160 160 }
161 161 }
162 162 }
163 163 }
164 164 }
165 165 }
166 166 }
167 167 }
168 168 }
169 169 }
170 170 }
171 171 }
172 172 }
173 173 }
174 174 }
175 175 }
176 176 }
177 177 }
178 178 }
179 179 }
180 180 }
181 181 }
182 182 }
183 183 }
184 184 }
185 185 }
186 186 }
187 187 }
188 188 }
189 189 }
190 190 }
191 191 }
192 192 }
193 193 }
194 194 }
195 195 }
196 196 }
197 197 }
198 198 }
199 199 }
200 200 }
201 201 }
202 202 }
203 203 }
204 204 }
205 205 }
206 206 }
207 207 }
208 208 }
209 209 }
210 210 }
211 211 }
212 212 }
213 213 }
214 214 }
215 215 }
216 216 }
217 217 }
218 218 }
219 219 }
220 220 }
221 221 }
222 222 }
223 223 }
224 224 }
225 225 }
226 226 }
227 227 }
228 228 }
229 229 }
230 230 }
231 231 }
232 232 }
233 233 }
234 234 }
235 235 }
236 236 }
237 237 }
238 238 }
239 239 }
240 240 }
241 241 }
242 242 }
243 243 }
244 244 }
245 245 }
246 246 }
247 247 }
248 248 }
249 249 }
250 250 }
251 251 }
252 252 }
253 253 }
254 254 }
255 255 }
256 256 }
257 257 }
258 258 }
259 259 }
260 260 }
261 261 }
262 262 }
263 263 }
264 264 }
265 265 }
266 266 }
267 267 }
268 268 }
269 269 }
270 270 }
271 271 }
272 272 }
273 273 }
274 274 }
275 275 }
276 276 }
277 277 }
278 278 }
279 279 }
280 280 }
281 281 }
282 282 }
283 283 }
284 284 }
285 285 }
286 286 }
287 287 }
288 288 }
289 289 }
290 290 }
291 291 }
292 292 }
293 293 }
294 294 }
295 295 }
296 296 }
297 297 }
298 298 }
299 299 }
300 300 }
301 301 }
302 302 }
303 303 }
304 304 }
305 305 }
306 306 }
307 307 }
308 308 }
309 309 }
310 310 }
311 311 }
312 312 }
313 313 }
314 314 }
315 315 }
316 316 }
317 317 }
318 318 }
319 319 }
320 320 }
321 321 }
322 322 }
323 323 }
324 324 }
325 325 }
326 326 }
327 327 }
328 328 }
329 329 }
330 330 }
331 331 }
332 332 }
333 333 }
334 334 }
335 335 }
336 336 }
337 337 }
338 338 }
339 339 }
340 340 }
341 341 }
342 342 }
343 343 }
344 344 }
345 345 }
346 346 }
347 347 }
348 348 }
349 349 }
350 350 }
351 351 }
352 352 }
353 353 }
354 354 }
355 355 }
356 356 }
357 357 }
358 358 }
359 359 }
360 360 }
361 361 }
362 362 }
363 363 }
364 364 }
365 365 }
366 366 }
367 367 }
368 368 }
369 369 }
370 370 }
371 371 }
372 372 }
373 373 }
374 374 }
375 375 }
376 376 }
377 377 }
378 378 }
379 379 }
380 380 }
381 381 }
382 382 }
383 383 }
384 384 }
385 385 }
386 386 }
387 387 }
388 388 }
389 389 }
390 390 }
391 391 }
392 392 }
393 393 }
394 394 }
395 395 }
396 396 }
397 397 }
398 398 }
399 399 }
400 400 }
401 401 }
402 402 }
403 403 }
404 404 }
405 405 }
406 406 }
407 407 }
408 408 }
409 409 }
410 410 }
411 411 }
412 412 }
413 413 }
414 414 }
415 415 }
416 416 }
417 417 }
418 418 }
419 419 }
420 420 }
421 421 }
422 422 }
423 423 }
424 424 }
425 425 }
426 426 }
427 427 }
428 428 }
429 429 }
430 430 }
431 431 }
432 432 }
433 433 }
434 434 }
435 435 }
436 436 }
437 437 }
438 438 }
439 439 }
440 440 }
441 441 }
442 442 }
443 443 }
444 444 }
445 445 }
446 446 }
447 447 }
448 448 }
449 449 }
450 450 }
451 451 }
452 452 }
453 453 }
454 454 }
455 455 }
456 456 }
457 457 }
458 458 }
459 459 }
460 460 }
461 461 }
462 462 }
463 463 }
464 464 }
465 465 }
466 466 }
467 467 }
468 468 }
469 469 }
470 470 }
471 471 }
472 472 }
473 473 }
474 474 }
475 475 }
476 476 }
477 477 }
478 478 }
479 479 }
480 480 }
481 481 }
482 482 }
483 483 }
484 484 }
485 485 }
486 486 }
487 487 }
488 488 }
489 489 }
490 490 }
491 491 }
492 492 }
493 493 }
494 494 }
495 495 }
496 496 }
497 497 }
498 498 }
499 499 }
500 500 }
501 501 }
502 502 }
503 503 }
504 504 }
505 505 }
506 506 }
507 507 }
508 508 }
509 509 }
510 510 }
511 511 }
512 512 }
513 513 }
514 514 }
515 515 }
516 516 }
517 517 }
518 518 }
519 519 }
520 520 }
521 521 }
522 522 }
523 523 }
524 524 }
525 525 }
526 526 }
527 527 }
528 528 }
529 529 }
530 530 }
531 531 }
532 532 }
533 533 }
534 534 }
535 535 }
536 536 }
537 537 }
538 538 }
539 539 }
540 540 }
541 541 }
542 542 }
543 543 }
544 544 }
545 545 }
546 546 }
547 547 }
548 548 }
549 549 }
550 550 }
551 551 }
552 552 }
553 553 }
554 554 }
555 555 }
556 556 }
557 557 }
558 558 }
559 559 }
560 560 }
561 561 }
562 562 }
563 563 }
564 564 }
565 565 }
566 566 }
567 567 }
568 568 }
569 569 }
570 570 }
571 571 }
572 572 }
573 573 }
574 574 }
575 575 }
576 576 }
577 577 }
578 578 }
579 579 }
580 580 }
581 581 }
582 582 }
583 583 }
584 584 }
585 585 }
586 586 }
587 587 }
588 588 }
589 589 }
590 590 }
591 591 }
592 592 }
593 593 }
594 594 }
595 595 }
596 596 }
597 597 }
598 598 }
599 599 }
600 600 }
601 601 }
602 602 }
603 603 }
604 604 }
605 605 }
606 606 }
607 607 }
608 608 }
609 609 }
610 610 }
611 611 }
612 612 }
613 613 }
614 614 }
615 615 }
616 616 }
617 617 }
618 618 }
619 619 }
620 620 }
621 621 }
622 622 }
623 623 }
624 624 }
625 625 }
626 626 }
627 627 }
628 628 }
629 629 }
630 630 }
631 631 }
632 632 }
633 633 }
634 634 }
635 635 }
636 636 }
637 637 }
638 638 }
639 639 }
640 640 }
641 641 }
642 642 }
643 643 }
644 644 }
645 645 }
646 646 }
647 647 }
648 648 }
649 649 }
650 650 }
651 651 }
652 652 }
653 653 }
654 654 }
655 655 }
656 656 }
657 657 }
658 658 }
659 659 }
660 660 }
661 661 }
662 662 }
663 663 }
664 664 }
665 665 }
666 666 }
667 667 }
668 668 }
669 669 }
670 670 }
671 671 }
672 672 }
673 673 }
674 674 }
675 675 }
676 676 }
677 677 }
678 678 }
679 679 }
680 680 }
681 681 }
682 682 }
683 683 }
684 684 }
685 685 }
686 686 }
687 687 }
688 688 }
689 689 }
690 690 }
691 691 }
692 692 }
693 693 }
694 694 }
695 695 }
696 696 }
697 697 }
698 698 }
699 699 }
700 700 }
701 701 }
702 702 }
703 703 }
704 704 }
705 705 }
706 706 }
707 707 }
708 708 }
709 709 }
710 710 }
711 711 }
712 712 }
713 713 }
714 714 }
715 715 }
716 716 }
717 717 }
718 718 }
719 719 }
720 720 }
721 721 }
722 722 }
723 723 }
724 724 }
725 725 }
726 726 }
727 727 }
728 728 }
729 729 }
730 730 }
731 731 }
732 732 }
733 733 }
734 734 }
735 735 }
736 736 }
737 737 }
738 738 }
739 739 }
740 740 }
741 741 }
742 742 }
743 743 }
744 744 }
745 745 }
746 746 }
747 747 }
748 748 }
749 749 }
750 750 }
751 751 }
752 752 }
753 753 }
754 754 }
755 755 }
756 756 }
757 757 }
758 758 }
759 759 }
760 760 }
761 761 }
762 762 }
763 763 }
764 764 }
765 765 }
766 766 }
767 767 }
768 768 }
769 769 }
770 770 }
771 771 }
772 772 }
773 773 }
774 774 }
775 775 }
776 776 }
777 777 }
778 778 }
779 779 }
780 780 }
781 781 }
782 782 }
783 783 }
784 784 }
785 785 }
786 786 }
787 787 }
788 788 }
789 789 }
790 790 }
791 791 }
792 792 }
793 793 }
794 794 }
795 795 }
796 796 }
797 797 }
798 798 }
799 799 }
800 800 }
801 801 }
802 802 }
803 803 }
804 804 }
805 805 }
806 806 }
807 807 }
808 808 }
809 809 }
810 810 }
811 811 }
812 812 }
813 813 }
814 814 }
815 815 }
816 816 }
817 817 }
818 818 }
819 819 }
820 820 }
821 821 }
822 822 }
823 823 }
824 824 }
825 825 }
826 826 }
827 827 }
828 828 }
829 829 }
830 830 }
831 831 }
832 832 }
833 833 }
834 834 }
835 835 }
836 836 }
837 837 }
838 838 }
839 839 }
840 840 }
841 841 }
842 842 }
843 843 }
844 844 }
845 845 }
846 846 }
847 847 }
848 848 }
849 849 }
850 850 }
851 851 }
852 852 }
853 853 }
854 854 }
855 855 }
856 856 }
857 857 }
858 858 }
859 859 }
860 860 }
861 861 }
862 862 }
863 863 }
864 864 }
865 865 }
866 866 }
867 867 }
868 868 }
869 869 }
870 870 }
871 871 }
872 872 }
873 873 }
874 874 }
875 875 }
876 876 }
877 877 }
878 878 }
879 879 }
880 880 }
881 881 }
882 882 }
883 883 }
884 884 }
885 885 }
886 886 }
887 887 }
888 888 }
889 889 }
890 890 }
891 891 }
892 892 }
893 893 }
894 894 }
895 895 }
896 896 }
897 897 }
898 898 }
899 899 }
900 900 }
901 901 }
902 902 }
903 903 }
904 904 }
905 905 }
906 906 }
907 907 }
908 908 }
909 909 }
910 910 }
911 911 }
912 912 }
913 913 }
914 914 }
915 915 }
916 916 }
917 917 }
918 918 }
919 919 }
920 920 }
921 921 }
922 922 }
923 923 }
924 924 }
925 925 }
926 926 }
927 927 }
928 928 }
929 929 }
930 930 }
931 931 }
932 932 }
933 933 }
934 934 }
935 935 }
936 936 }
937 937 }
938 938 }
939 939 }
940 940 }
941 941 }
942 942 }
943 943 }
944 944 }
945 945 }
946 946 }
947 947 }
948 948 }
949 949 }
950 950 }
951 951 }
952 952 }
953 953 }
954 954 }
955 955 }
956 956 }
957 957 }
958 958 }
959 959 }
960 960 }
961 961 }
962 962 }
963 963 }
964 964 }
965 965 }
966 966 }
967 967 }
968 968 }
969 969 }
970 970 }
971 971 }
972 972 }
973 973 }
974 974 }
975 975 }
976 976 }
977 977 }
978 978 }
979 979 }
980 980 }
981 981 }
982 982 }
983 983 }
984 984 }
985 985 }
986 986 }
987 987 }
988 988 }
989 989 }
990 990 }
991 991 }
992 992 }
993 993 }
994 994 }
995 995 }
996 996 }
997 997 }
998 998 }
999 999 }
1000 1000 }
1001 1001 }
1002 1002 }
1003 1003 }
1004 1004 }
1005 1005 }
1006 1006 }
1007 1007 }
1008 1008 }
1009 1009 }
1010 1010 }
1011 1011 }
1012 1012 }
1013 1013 }
1014 1014 }
1015 1015 }
1016 1016 }
1017 1017 }
1018 1018 }
1019 1019 }
1020 1020 }
1021 1021 }
1022 1022 }
1023 1023 }
1024 1024 }
1025 1025 }
1026 1026 }
1027 1027 }
1028 1028 }
1029 1029 }
1030 1030 }
1031 1031 }
1032 1032 }
1033 1033 }
1034 1034 }
1035 1035 }
1036 1036 }
1037 1037 }
1038 1038 }
1039 1039 }
1040 1040 }
1041 1041 }
1042 1042 }
1043 1043 }
1044 1044 }
1045 1045 }
1046 1046 }
1047 1047 }
1048 1048 }
1049 1049 }
1050 1050 }
1051 1051 }
1052 1052 }
1053 1053 }
1054 1054 }
1055 1055 }
1056 1056 }
1057 1057 }
1058 1058 }
1059 1059 }
1060 1060 }
1061 1061 }
1062 1062 }
1063 1063 }
1064 1064 }
1065 1065 }
1066 1066 }
1067 1067 }
1068 1068 }
1069 1069 }
1070 1070 }
1071 1071 }
1072 1072 }
1073 1073 }
1074 1074 }
1075 1075 }
1076 1076 }
1077 1077 }
1078 1078 }
1079 1079 }
1080 1080 }
1081 1081 }
1082 1082 }
1083 1083 }
1084 1084 }
1085 1085 }
1086 1086 }
1087 1087 }
1088 1088 }
1089 1089 }
1090 1090 }
1091 1091 }
1092 1092 }
1093 1093 }
1094 1094 }
1095 1095 }
1096 1096 }
1097 1097 }
1098 1098 }
1099 1099 }
1100 1100 }
1101 1101 }
1102 1102 }
1103 1103 }
1104 1104 }
1105 1105 }
1106 1106 }
1107 1107 }
1108 1108 }
1109 1109 }
1110 1110 }
1111 1111 }
1112 1112 }
1113 1113 }
1114 1114 }
1115 1115 }
1116 1116 }
1117 1117 }
1118 1118 }
1119 1119 }
1120 1120 }
1121 1121 }
1122 1122 }
1123 1123 }
1124 1124 }
1125 1125 }
1126 1126 }
1127 1127 }
1128 1128 }
1129 1129 }
1130 1130 }
1131 1131 }
1132 1132 }
1133 1133 }
1134 1134 }
1135 1135 }
1136 1136 }
1137 1137 }
1138 1138 }
1139 1139 }
1140 1140 }
1141 1141 }
1142 1142 }
1143 1143 }
1144 1144 }
1145 1145 }
1146 1146 }
1147 1147 }
1148 1148 }
1149 1149 }
1150 1150 }
1151 1151 }
1152 1152 }
1153 1153 }
1154 1154 }
1155 1155 }
1156 1156 }
1157 1157 }
1158 1158 }
1159 1159 }
1160 1160 }
1161 1161 }
1162 1162 }
1163 1163 }
1164 1164 }
1165 1165 }
1166 1166 }
1167 1167 }
1168 1168 }
1169 1169 }
1170 1170 }
1171 1171 }
1172 1172 }
1173 1173 }
1174 1174 }
1175 1175 }
1176 1176 }
1177 1177 }
1178 1178 }
1179 1179 }
1180 1180 }
1181 1181 }
1182 1182 }
1183 1183 }
1184 1184 }
1185 1185 }
1186 1186 }
1187 1187 }
1188 1188 }
1189 1189 }
1190 1190 }
1191 1191 }
1192 1192 }
1193 1193 }
1194 1194 }
1195 1195 }
1196 1196 }
1197 1197 }
1198 1198 }
1199 1199 }
1200 1200 }
1201 1201 }
1202 1202 }
1203 1203 }
1204 1204 }
1205 1205 }
1206 1206 }
1207 1207 }
1208 1208 }
1209 1209 }
1210 1210 }
1211 1211 }
1212 1212 }
1213 1213 }
1214 1214 }
1215 1215 }
1216 1216 }
1217 1217 }
1218 1218 }
1219 1219 }
1220 1220 }
1221 1221 }
1222 1222 }
1223 1223 }
1224 1224 }
1225 1225 }
1226 1226 }
1227 1227 }
1228 1228 }
1229 1229 }
1230 1230 }
1231 1231 }
1232 1232 }
1233 1233 }
1234 1234 }
1235 1235 }
1236 1236 }
1237 1237 }
1238 1238 }
1239 1239 }
1240 1240 }
1241 1241 }
1242 1242 }
1243 1243 }
1244 1244 }
1245 1245 }
1246 1246 }
1247 1247 }
1248 1248 }
1249 1249 }
1250 1250 }
1251 1251 }
1252 1252 }
1253 1253 }
1254 1254 }
1255 1255 }
1256 1256 }
1257 1257 }
1258 1258 }
1259 1259 }
1260 1260 }
1261 1261 }
1262 1262 }
1263 1263 }
1264 1264 }
1265 1265 }
1266 1266 }
1267 1267 }
1268 1268 }
1269 1269 }
1270 1270 }
1271 1271 }
1272 1272 }
1273 1273 }
1274 1274 }
1275 1275 }
1276 1276 }
1277 1277 }
1278 1278 }
1279 1279 }
1280 1280 }
1281 1281 }
1282 1282 }
1283 1283 }
1284 1284 }
1285 1285 }
1286 1286 }
1287 1287 }
1288 1288 }
1289 1289 }
1290 1290 }
1291 1291 }
1292 1292 }
1293 1293 }
1294 1294 }
1295 1295 }
1296 1296 }
1297 1297 }
1298
```

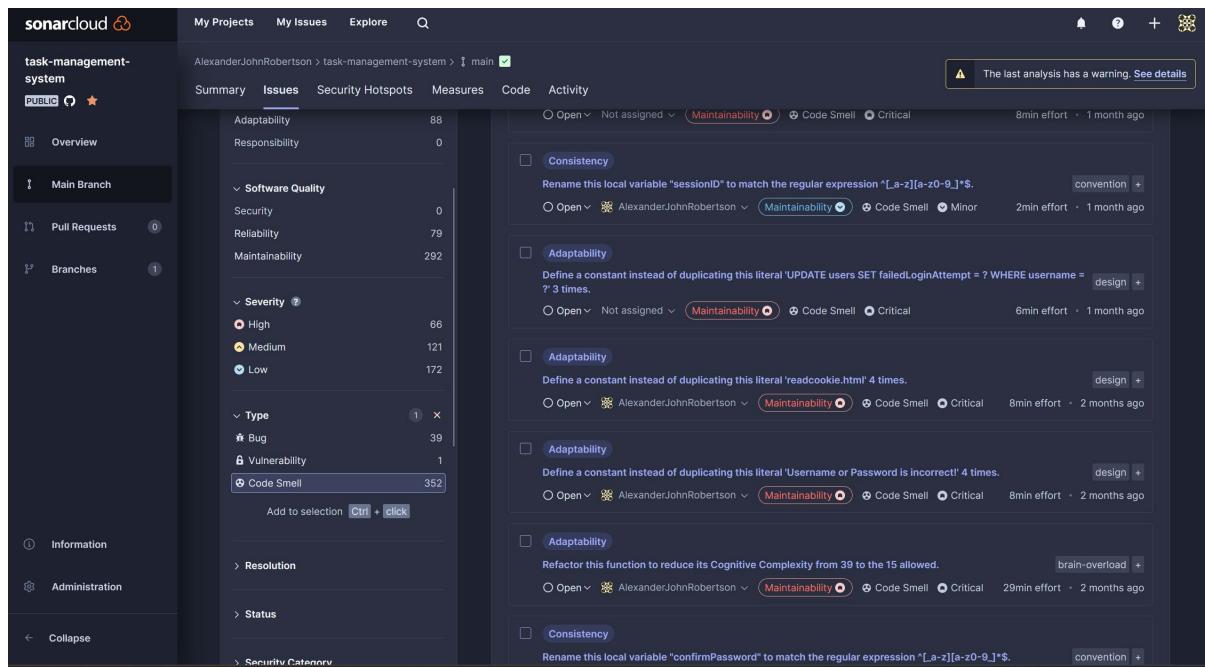


Figure 116 - SonarCloud has detected several code smells in the Task Management System.

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 | <ol style="list-style-type: none"> 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 | <ol style="list-style-type: none"> 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 | <ol style="list-style-type: none"> 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.

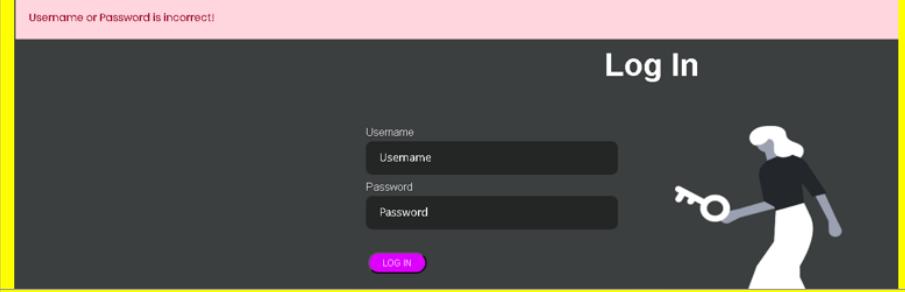
| | |
|---|--|
| Please put screenshots here: |  |
| Screen Size | 21.5" |
| Operating system on your device | Windows 10 |
| Browser(s) on your device used to access the Task Management System. | Google Chrome |
| Notes | |

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) | |

The screenshot shows the Render Dashboard's Overview page. At the top, there are tabs for 'Render' (selected), 'Dashboard', 'Blueprints', and 'Env Groups'. Below the tabs is a search bar labeled 'Search services'. A filter bar at the top right shows 'Active / 7', 'Suspended 0', and 'All 7'. The main table lists seven services:

| Service Name | Status | Type | Runtime | Region | Last Deployed | Actions |
|------------------------------|---------------|-------------|----------|-----------|---------------|---------|
| Task Management System (...) | Deployed | Web Service | Python 3 | Frankfurt | 3 days ago | ... |
| Task-Management-System | Deployed | Web Service | Python 3 | Frankfurt | 3 days ago | ... |
| TaskManagementSystemTest | Deployed | Web Service | Python 3 | Frankfurt | 3 days ago | ... |
| Responsive-Navbar-Test | Deployed | Web Service | Python 3 | Frankfurt | a month ago | ... |
| flast-test-4 | Deployed | Web Service | Python 3 | Frankfurt | 2 months ago | ... |
| flasktest2 | Deployed | Web Service | Python 3 | Frankfurt | 2 months ago | ... |
| flasktest | Failed deploy | Static Site | Static | Global | 2 months ago | ... |

At the bottom left are links for 'Feedback', 'Invite a Friend', and 'Contact Support'. On the bottom right are icons for 'Feedback', 'Invite a Friend', and 'Contact Support'.

Figure 123 - Render.com deployments

The screenshot shows the Render service creation interface. At the top, there are tabs for 'Docs', 'Community', and 'Help', and a 'New +' button. On the right, there is a user icon. The main area has a sidebar with 'TYP' and 'Web' filters, and a 'Web Service' section with a 'Learn more.' link. To the right, a large card lists service types:

- Static Site
- Web Service
- Private Service
- Background Worker
- Cron Job
- PostgreSQL
- Redis
- Blueprint

A blue 'Activ' button is visible on the right side of the card. At the bottom, there are sections for 'LA', '3 c', and '3 c'.

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

S275931

The screenshot shows the Render web interface. At the top, there's a navigation bar with links for Dashboard, Blueprints, Env Groups, Docs, Community, Help, and a 'New +' button. Below the navigation, the main title is 'Create a new Web Service'. A sub-instruction says 'Connect a Git repository, or use an existing image.' Underneath, a question asks 'How would you like to deploy your web service?'. Two options are shown: 'Build and deploy from a Git repository' (selected) and 'Deploy an existing image from a registry' (with an 'ADVANCED' link). Both options have a 'Connect' sub-instruction. At the bottom right is a large blue 'Next' button.

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

The screenshot shows the Render web interface. The top navigation bar includes 'Dashboard', 'Blueprints', 'Env Groups', 'Docs', 'Community', 'Help', a 'New +' button, and a user profile for 'Alexander Robertson'. The main title is 'Create a new Web Service' with the subtitle 'Connect your Git repository or use an existing public repository URL.' Below this, a section titled 'Connect a repository' features a search bar and a list of repositories from 'AlexanderJohnRobertson'. Each repository entry has a 'Connect' button. To the right, there are sections for 'GitHub' (listing 17 repos), 'GitLab' (with a '+ Connect account' link), and 'Bitbucket' (with a '+ Connect account' link).

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

You are deploying a web service for [AlexanderJohnRobertson/TaskManagementSystemDeployment](#).

You seem to be using Flask, so we've autofilled some fields accordingly. Make sure the values look right to you!

| | |
|----------------|---|
| Name | TaskManagementSystemDeployment |
| Region | Frankfurt (EU Central) |
| Branch | main |
| Root Directory | e.g. src |
| Runtime | Python 3 |
| Build Command | <code>\$ pip install -r requirements.txt</code> |

Figure 127 - Render.com deployment settings.

Instance Type

| | | |
|--|--|--|
| For hobby projects | Free
\$0 / month
512 MB (RAM)
0.1 CPU | Upgraded to enable more features
Free instances spin down after periods of inactivity. They do not support SSH access, scaling, one-off jobs, or persistent disks. Select any paid instance type to enable these features. |
| For professional use
For more power and to get the most out of Render, we recommend using one of our paid instance types. All paid instances support: | Starter
\$7 / month
512 MB (RAM)
0.5 CPU | Standard
\$25 / month
2 GB (RAM)
1 CPU |
| <ul style="list-style-type: none"> Zero Downtime SSH Access Scaling One-off jobs Support for persistent disks | Pro
\$85 / month
4 GB (RAM)
2 CPU | Pro Plus
\$175 / month
8 GB (RAM)
4 CPU |
| | Pro Max
\$225 / month
16 GB (RAM)
4 CPU | Pro Ultra
\$450 / month
32 GB (RAM)
8 CPU |

Need a [custom instance type](#)? We support up to 512 GB RAM and 64 CPUs.

Environment Variables Optional
Set environment-specific config and secrets (such as API keys), then read those values from your code. [Learn more](#).

| | | | |
|----------------------------|---------------|----------|--------|
| NAME_OF_VARIABLE | value | Generate | Remove |
| + Add Environment Variable | Add from .env | | |

Advanced ▾

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

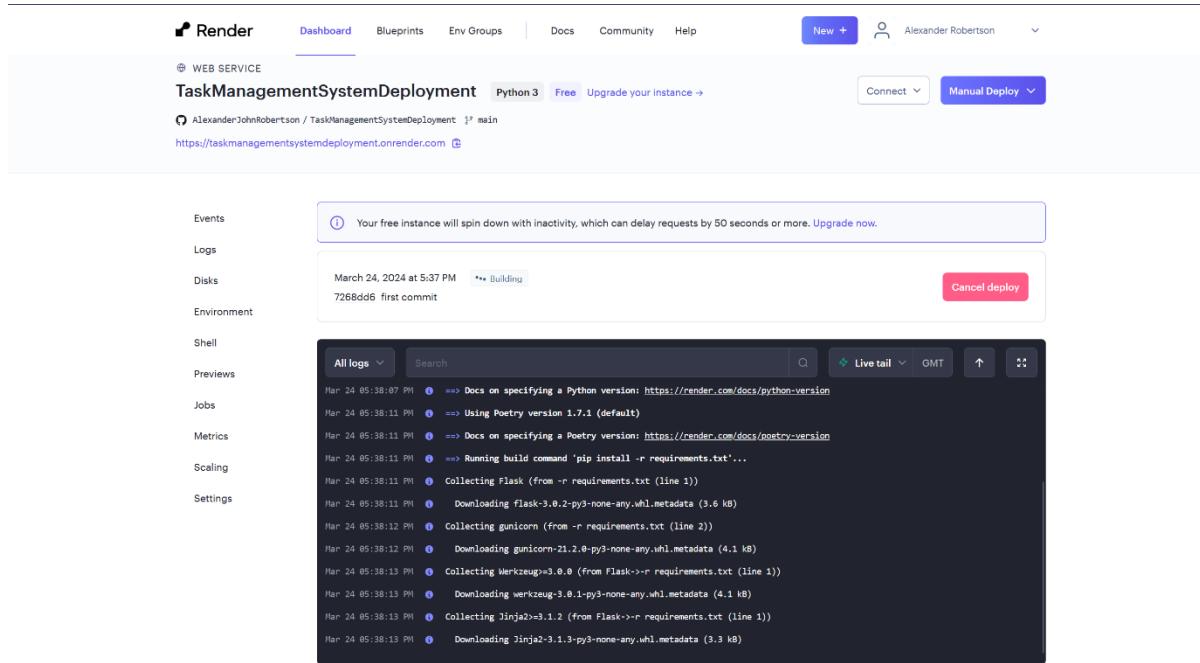


Figure 129 - Render.com is automatically building the Task Management System web service as a CI/CD pipeline.

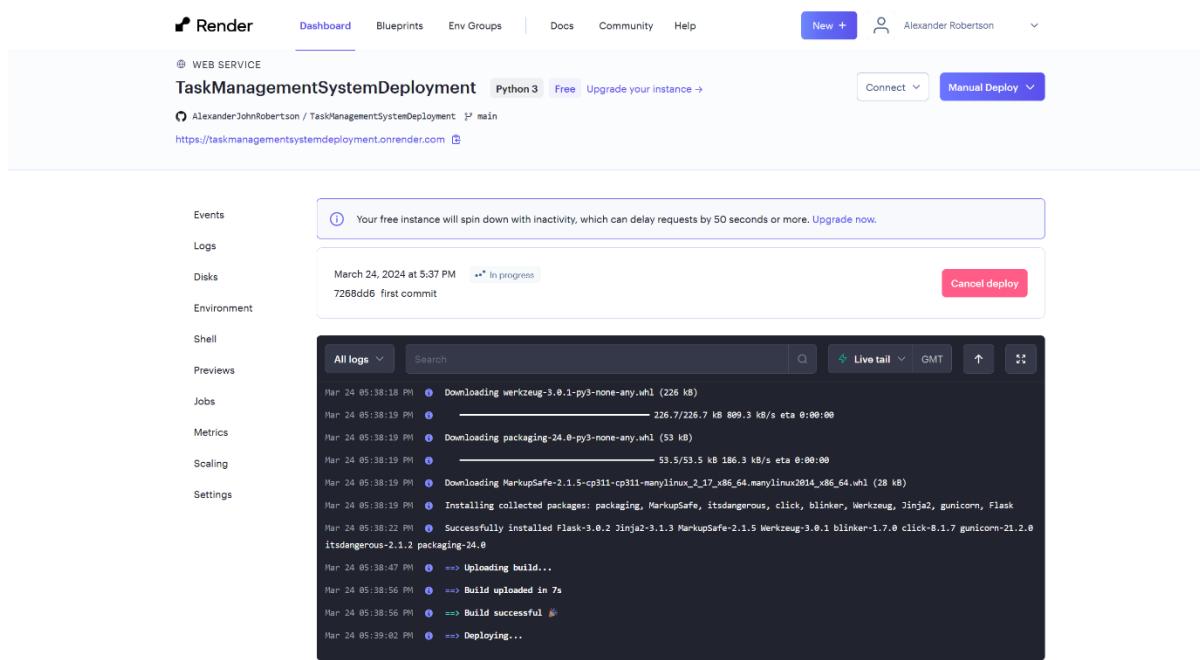


Figure 130 - Render.com is automatically deploying the Task Management System web service as a CI/CD pipeline.

S275931

The screenshot shows the Render web interface for the 'TaskManagementSystemDeployment' service. At the top, there's a navigation bar with 'Render', 'Dashboard', 'Blueprints', 'Env Groups', 'Docs', 'Community', 'Help', a 'New +' button, and a user profile for 'Alexander Robertson'. Below the navigation is a card for the 'TaskManagementSystemDeployment' service, which is running on Python 3 and is currently free. It includes a 'Connect' button and a 'Manual Deploy' button. The main area shows tabs for 'Events', 'Logs', 'Disks', 'Environment', 'Shell', 'Previews', 'Jobs', 'Metrics', 'Scaling', and 'Settings'. The 'Logs' tab is active, displaying a log viewer with a 'Live tail' button and a 'GMT' button. The log viewer shows several log entries from March 24, 2024, at 5:37 PM, including messages about the unicorn process starting and a worker booting up. A message in the log indicates that the instance will spin down if inactive for 50 seconds.

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

The screenshot shows the Render web interface for the 'Task-Management-System' service. At the top, there's a navigation bar with 'Render', 'Dashboard', 'Blueprints', 'Env Groups', 'Docs', 'Community', 'Help', a 'New +' button, and a user profile for 'Alexander Robertson'. Below the navigation is a card for the 'Task-Management-System' service, which is running on Python 3 and is currently a 'Starter' plan. It includes a 'Connect' button and a 'Manual Deploy' button. The main area shows tabs for 'Events', 'Logs', 'Disks', 'Environment', 'Shell', 'Previews', 'Jobs', 'Metrics', 'Scaling', and 'Settings'. The 'Events' tab is active, displaying a list of deployment events. One event is highlighted: 'Deploy live for fa579bf: new commit' on April 16, 2024, at 9:29 PM. Another event is shown: 'Deploy started for fa579bf: new commit' on April 16, 2024, at 9:27 PM, described as a 'New commit via Auto-Deploy'. There are also events for a test deployment on April 7, 2024, and a maintenance trigger on April 7, 2024. Two 'Rollback' buttons are visible next to the test deployment logs.

Figure 132 - Task Management System web service deployment history.

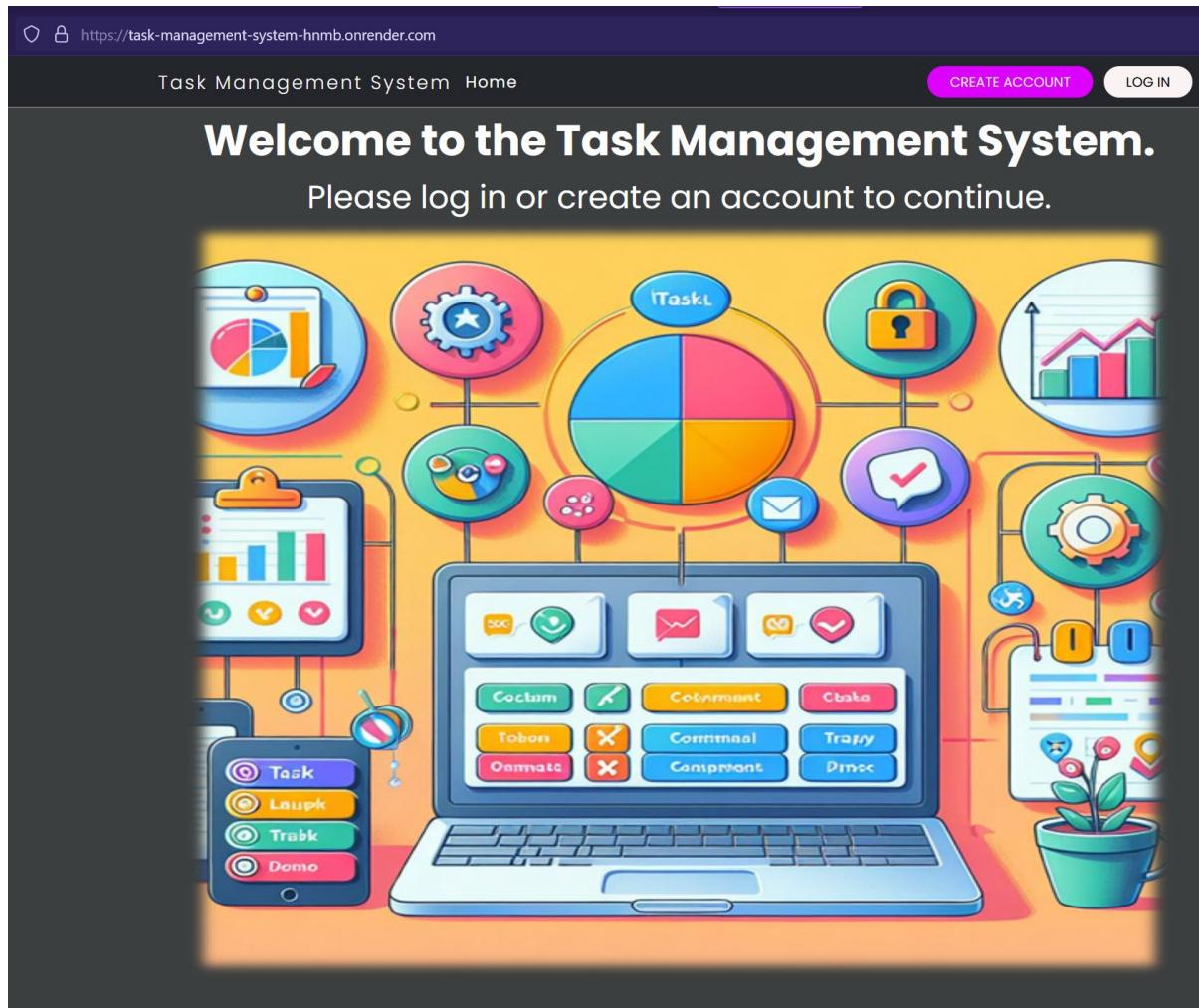


Figure 133 - Task Management System web service landing page.

S275931

The screenshot shows the Render web interface for a 'Task Management System (Clean Install)' service. The top navigation bar includes links for Dashboard, Blueprints, Env Groups, Docs, Community, Help, a 'New +' button, and a user profile for Alexander Robertson. The main content area displays the service details: 'WEB SERVICE' icon, name 'Task Management System (Clean Install)', language 'Python 3', and environment 'Starter'. It also shows the GitHub repository 'AlexanderJohnRobertson / task-management-system-clean-install' and the URL 'https://task-management-system-clean-install.onrender.com'. Below this, there are two buttons: 'Connect' and 'Manual Deploy'. On the left, a sidebar menu lists various monitoring and management options: Events (selected), Logs, Disks, Environment, Shell, Previews, Jobs, Metrics, Scaling, and Settings. The main panel on the right shows the deployment history for the 'fbe4c9d: test' branch. The events listed are:

- Deploy live for fbe4c9d: test (April 14, 2024 at 2:26 PM)
- Deploy started for fbe4c9d: test (Manually triggered by you via Dashboard, April 14, 2024 at 2:24 PM)
- Deploy live for fbe4c9d: test (April 7, 2024 at 1:29 AM)
- Deploy started for fbe4c9d: test (Triggered by Render as part of routine infrastructure maintenance, April 7, 2024 at 1:28 AM)
- Instance type changed from Free to Starter (April 3, 2024 at 7:00 PM)
- Deploy live for fbe4c9d: test (April 3, 2024 at 7:00 PM)

For the last two events, there are 'Rollback' buttons.

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

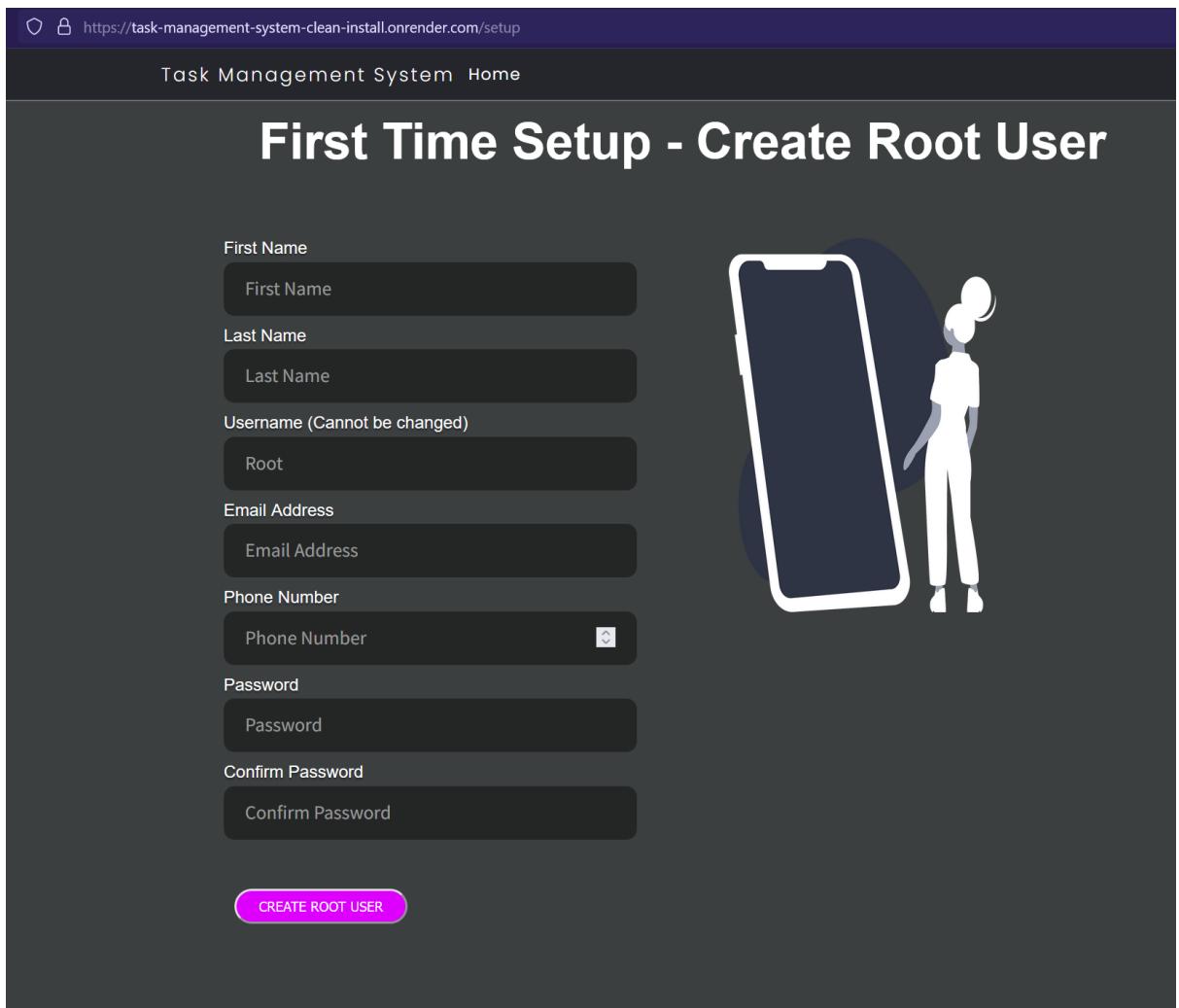


Figure 135 - Task Management System (Clean Install) setup page.

- 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) | |

```
PS C:\Users\rober\OneDrive\Documents\Advanced_Computing_Suffolk\Advanced Software Engineering\Assignment\TaskManagementS
ystem_Clean_Install> docker build -t alexjrobertson/taskmanagementsystemcleaninstall:latest .
[+] Building 0.9s (11/11) FINISHED                                            docker:default
=> [internal] load build definition from Dockerfile                         0.0s
=> => transferring dockerfile: 307B                                         0.0s
=> [internal] load metadata for docker.io/library/python:3-alpine3.12        0.7s
=> [internal] load .dockerrignore                                         0.0s
=> => transferring context: 568                                           0.0s
=> [1/6] FROM docker.io/library/python:3-alpine3.12@sha256:7f73901e568630443fc50e358b76603492e89c9bf330caf689e85 0.0s
=> [internal] load build context                                         0.1s
=> => transferring context: 131.97kB                                       0.0s
=> CACHED [2/6] WORKDIR ./app                                              0.0s
=> CACHED [3/6] ADD requirements.txt /app                                 0.0s
=> CACHED [4/6] COPY . /app                                               0.0s
=> CACHED [5/6] RUN export PYTHONPATH=/usr/bin/python                      0.0s
=> CACHED [6/6] RUN pip install -r requirements.txt                        0.0s
=> exporting to image                                                     0.0s
=> => exporting layers                                                    0.0s
=> => writing image sha256:7c4abaaaf5f6a74a111e4bf016e88eeba7ac6cf651fb2b0d614a62dc2e1cda73 0.0s
=> => naming to docker.io/alexjrobertson/taskmanagementsystemcleaninstall:latest 0.0s

What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Users\rober\OneDrive\Documents\Advanced_Computing_Suffolk\Advanced Software Engineering\Assignment\TaskManagementS
ystem_Clean_Install> docker container run -d -p 5000:5000 alexjrobertson/taskmanagementsystemcleaninstall:latest
57cc976e24363756cb7b32fe91d27338fcfa95ce7d3f71535d44e28d0f3f2e71a
PS C:\Users\rober\OneDrive\Documents\Advanced_Computing_Suffolk\Advanced Software Engineering\Assignment\TaskManagementS
ystem_Clean_Install> docker ps
CONTAINER ID IMAGE                                     COMMAND          CREATED          STATUS
PORTS              NAMES
57cc976e2436    alexjrobertson/taskmanagementsystemcleaninstall:latest   "/bin/sh -c 'python ...'"  5 seconds ago   Up 4 se
conds
0.0.0.5000->5000/tcp  loving_shirley
PS C:\Users\rober\OneDrive\Documents\Advanced_Computing_Suffolk\Advanced Software Engineering\Assignment\TaskManagementS
ystem_Clean_Install>
```

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

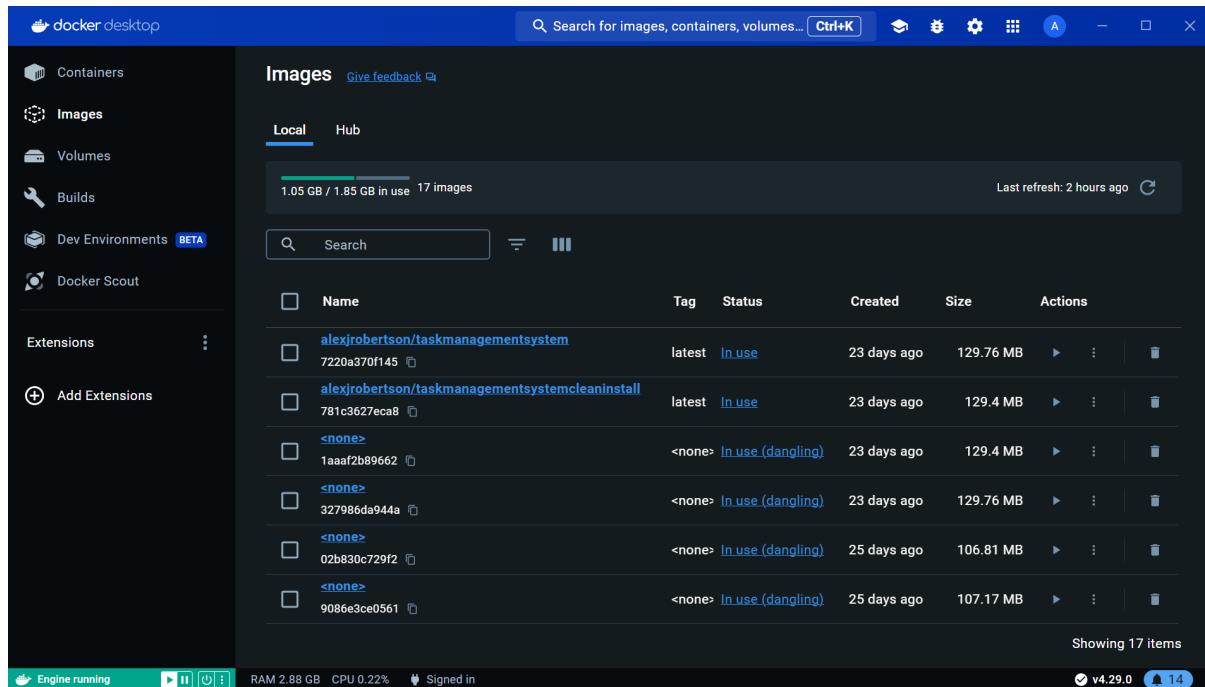


Figure 137 - Task Management System Docker Container images in Docker Desktop.

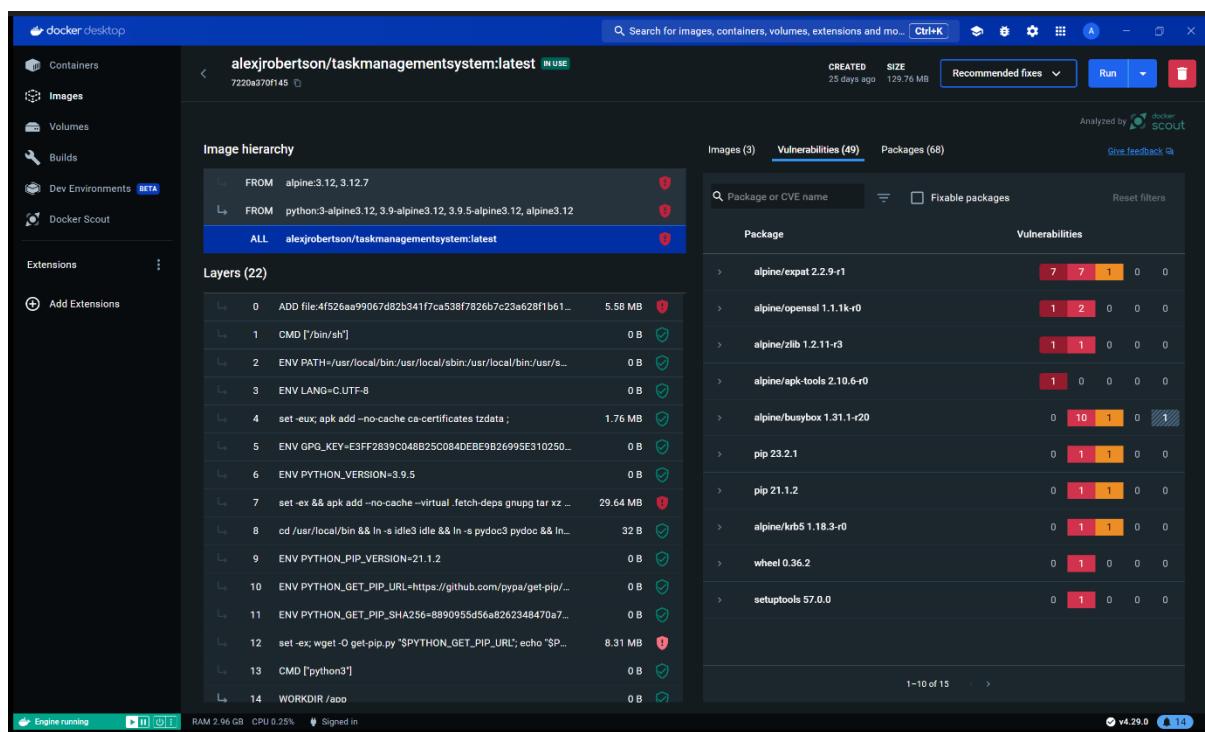


Figure 138 - Task Management System container in Docker Desktop. Docker has detected security vulnerabilities with Alpine Linux, which is required to run the Task Management System container.

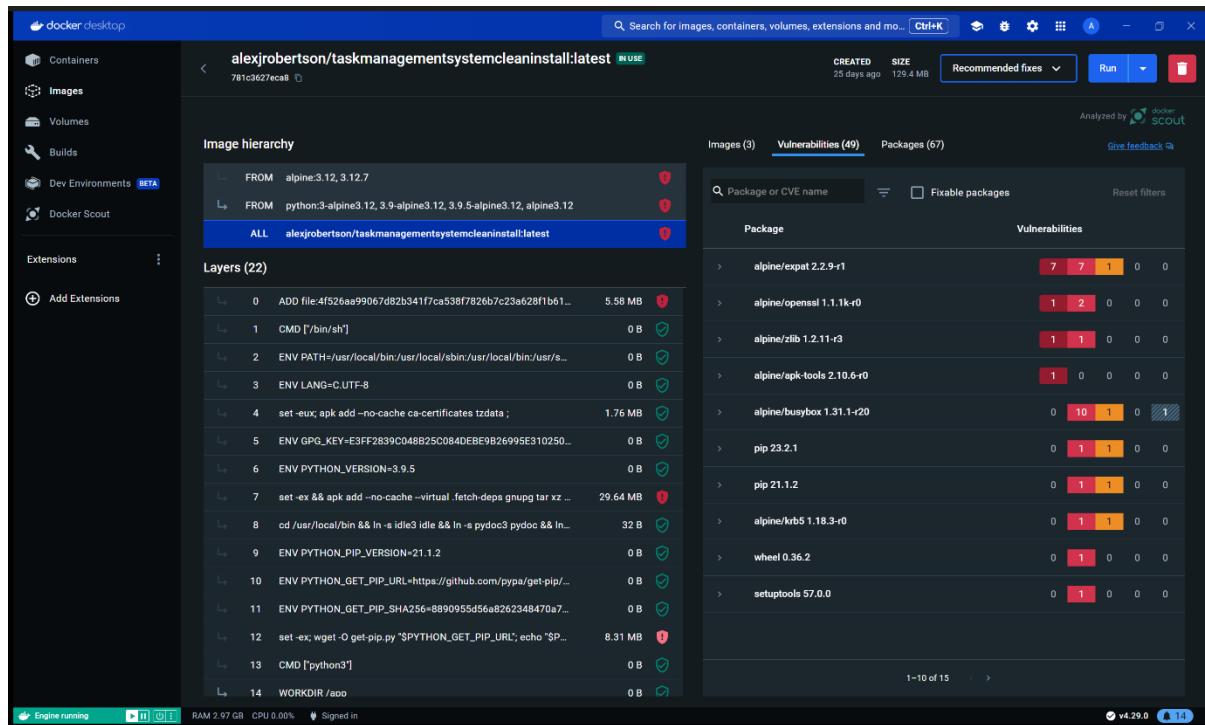


Figure 139 - Clean Install version of the Task Management System.

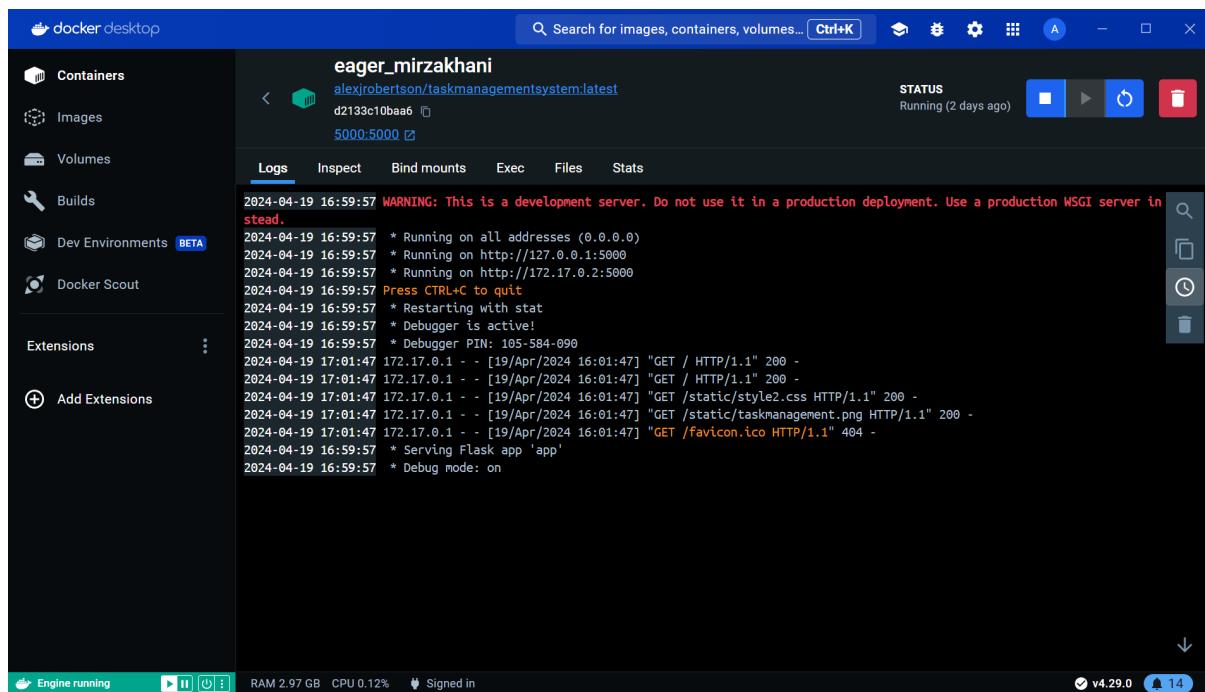


Figure 140 - Task Management System container running in Docker Desktop.

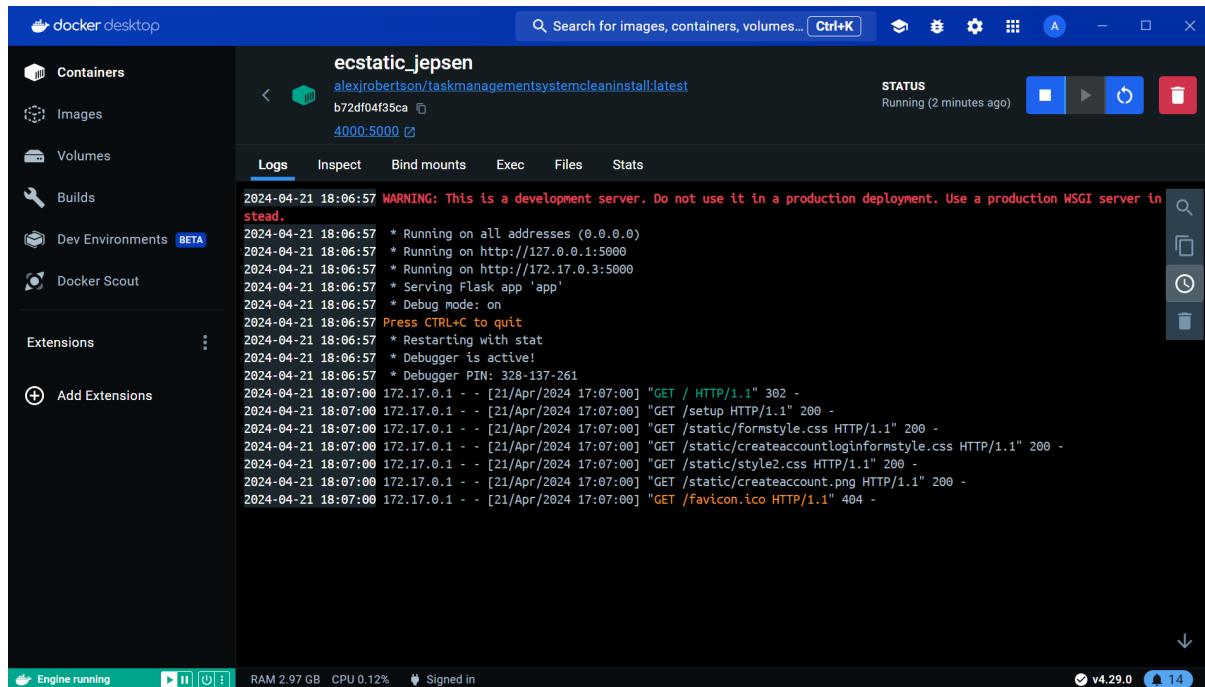


Figure 141 - Clean Install version of the Task Management System running in Docker Desktop.

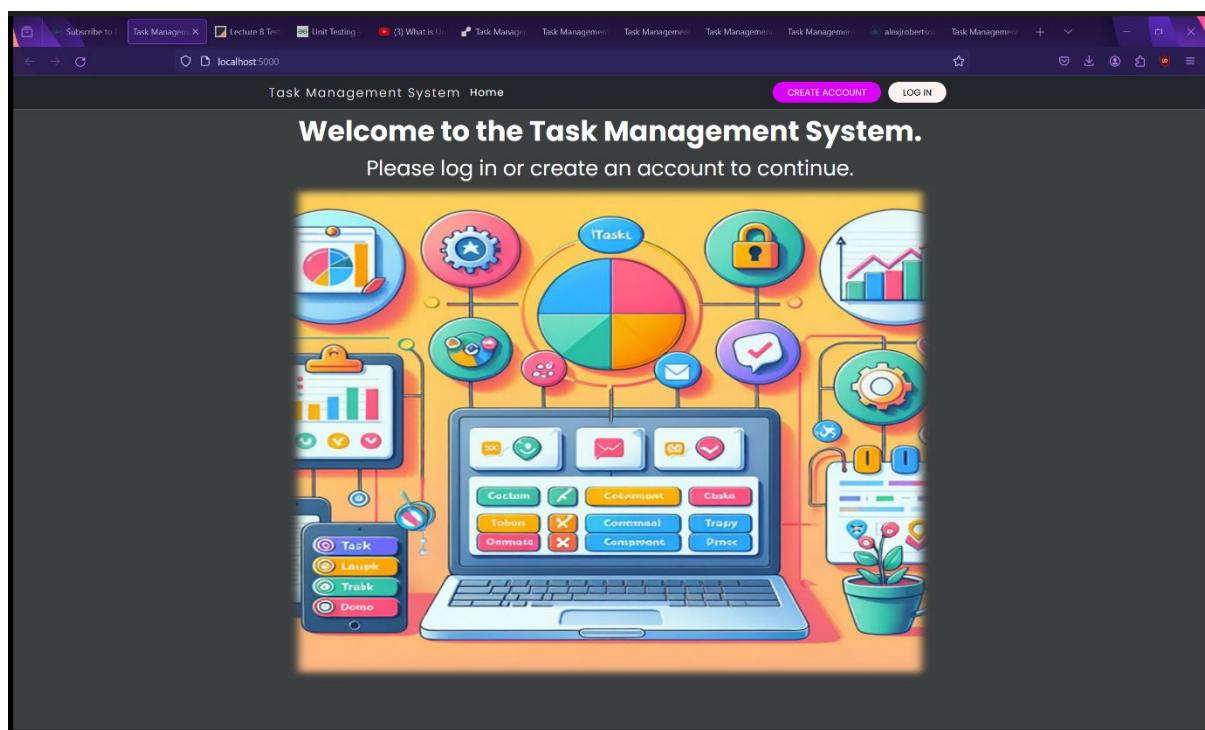


Figure 142 - Landing page of the Task Management System in a web browser, running in Docker as a container.

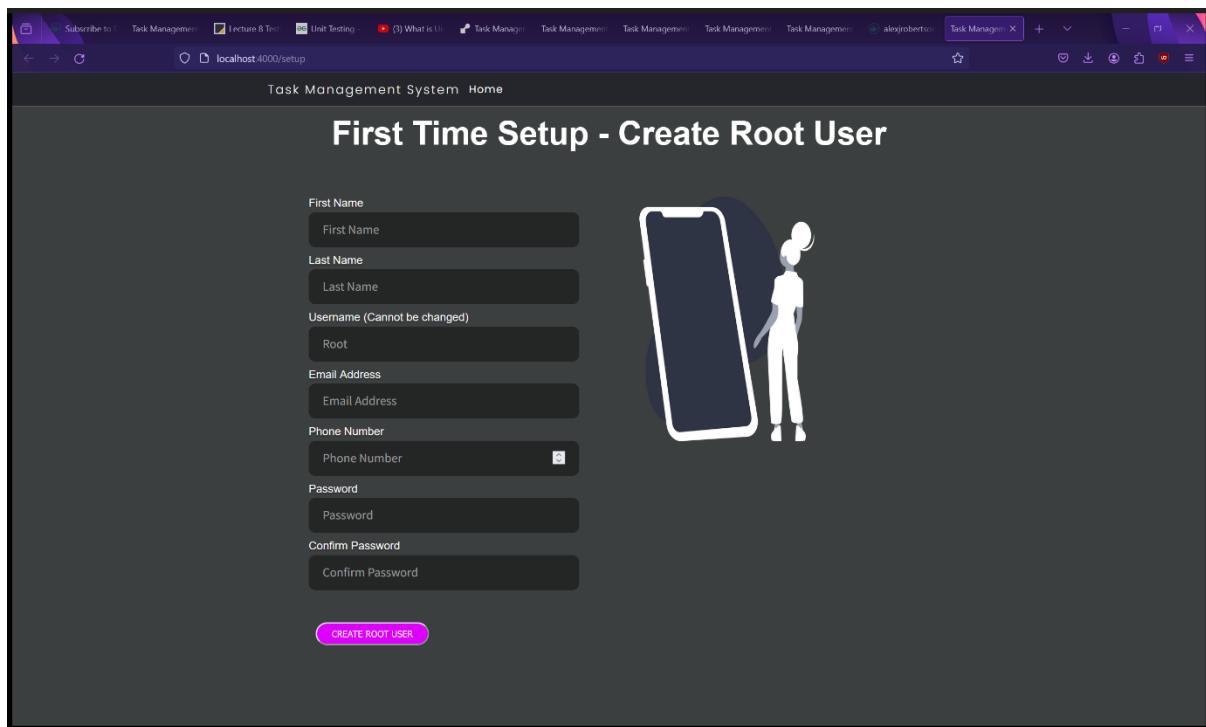


Figure 143 - Setup page of the Clean Install version of the Task Management System in a web browser, running in Docker as a container.

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 | |
| <u>Task Management System Documentation</u> | |
| Table of Contents | |
| Introduction | 2 |
| System Requirements | 3 |
| Docker Deployment (Windows):..... | 3 |
| Docker Deployment (macOS):..... | 4 |
| Docker Deployment (Linux):..... | 4 |
| Render.com deployment (online):..... | 5 |
| Building and deploying from source code (Windows):..... | 5 |
| Building and deploying from source code (macOS): | 5 |
| Building and deploying from source code (Linux):..... | 6 |
| Installation | 6 |
| Setup Render.com Deployment | 9 |
| Installation and Setup using Docker Deployment | 11 |
| Installation and Setup Using Source Code..... | 19 |
| Prerequisites | 20 |
| Downloading from the GitHub repository..... | 20 |
| Open the PyCharm Project..... | 21 |
| Setting up the PyCharm environment (Windows)..... | 24 |
| Setting up the PyCharm Environment (Linux) | 31 |
| Setting up the PyCharm environment (macOS)..... | 35 |
| Setting up the Flask Configuration | 40 |
| Run the Task Management System from PyCharm | 43 |
| Setting up and pushing to Git repository (Version Control)..... | 45 |
| Modifying Source code, Adding and Testing New Features..... | 52 |
| Suggested Modifications | 53 |
| Unit Testing the Task Management System Source Code | 55 |

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.

[Customise Documentation](#)

Figure 145 - Screenshot of the Task Management System documentation.

The screenshot shows a GitHub Wiki page titled "Task Management System Documentation". The main content area contains a "Table of Contents" section with several headings and sub-headings. The sidebar on the right is titled "Pages" and lists various pages related to the task management system, including "Home", "Task Management System Document...", and several deployment and setup guides.

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

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 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.

System Requirements

Docker Deployment (Windows)

- Computer must meet the system requirements to run Docker and a compatible web browser.
- 64 Bit Windows 10 Home, Professional, Enterprise or Education 21H2 (build 19044) or later or Windows 11 Home, Professional, Enterprise or Education version 22H2 or later (Docker, Inc., 2024).
- Docker Desktop and compatible web browser installed.
- Windows Subsystem for Linux (WSL) 2 feature enabled with a Linux operating system installed (Available from the Microsoft Store) or Hyper-V and Windows Containers features enabled (Docker, Inc., 2024).
- Professional and Enterprise editions of Windows are required to run Windows containers (Docker, Inc., 2024).
- 64-bit processor (CPU) with Second Level Address Translation (SLAT) (Docker, Inc., 2024) (Extended Page Table (EPT) in Intel processors, Rapid Virtualisation Indexing (RVI) in AMD processors) (Gibb, 2011).
- 4GB system RAM (Docker, Inc., 2024) (More may be required if application is scaled).

Delete Tasks – Administrators and Root users only.

Managing Your Account

[View Your Account Details](#)

[Changing your Account Details](#)

[Changing your Username](#)

[Changing your Password](#)

[Deleting your Account](#)

Managing Users and Administrative Tasks – Administrators and Root User Only

[View the List of Users as a Table \(Administrators and Root User Only\)](#)

[Change User Account Type \(Administrators and Root Users Only\)](#)

[Delete a User \(Administrators and Root Users Only\)](#)

[Blocking a User \(Administrators and Root Users Only\)](#)

[Unblocking a User \(Administrators and Root Users Only\)](#)

[Changing Another User's Password \(Administrators and Root Users Only\)](#)

[Factory Resetting the Task Management System \(Administrators and Root Users Only\)](#)

Task Management System on Mobile Devices

Security Features

References

+ Add a custom sidebar

Clone this wiki locally

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.

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.

```

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 user is not blocked.

```

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 == phonenumer:
    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 | |
| alillick999 | Alex | Gilllick | lgilllick@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 | robertson97@icloud.com | 983749739378 | 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 | friwett@hotmail.com | 93875937457 | Standard | No | |
| oGillick999 | 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 | robertson97@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 | 491a0812d10f16546ca5ca12610224014028a204abe50f4c66dec04616ea40f7ca8a9d64ac88b3a9079aa0814982.. | Kane | Fenlon | kfenlon@hotmail.com | 42342342 | Standard | Yes | 3 | 0 |
| 2 LfenlonVolcano | 7a6efbb087d2cb2b2efc3cf7651bcfa13f36cd69bf8sead4c61a6e2640b..aeab0934fd070bbe148bf19688062fc8.. | Lauren | Durury | ldurury@msn.com | 234223424 | Standard | No | 0 | 0 |
| 3 GemmaWheels | 4dfe2afaf1af7342236b05cb657d3ce58bae9612903e2bf13e6614d5940eab7e9e4fa5f1c24cd5a0f3b681ca7c6.. | Gemma | Bird | gbird@hotmail.com | 2323466 | Standard | No | 0 | 0 |
| 4 kellyG | 7a6efbb087d2cb2b2efc3cf7651bcfa13f36cd69bf8sead4c614a62640b..aeab0934fd070bbe148bf19688062fc8.. | Kelly | Gates | kgates@outlook.com | 78624868243 | Standard | No | 0 | 0 |
| 5 ethomas123 | b48297d22c02d065479b69cf0ffdd112ae8e6450b0853367e023459bdff2ff03ca9d742066385a52da1b202c1eb.. | Ellie | Thomas | ethomas@icloud.com | 836446834 | Standard | No | 0 | 0 |
| 6 ArasakaCEO123 | 2bb5b3eecc8a3bf8ff44d4f2933d1d38b2700e445c551e9e60765481884d926fe34347d8a0f2233b6b395b75175.. | Bill | Thomas | bthomas@hotmail.com | 012434376876 | Standard | No | 0 | 0 |
| 7 DandyLion000 | 295da93c654e2b52da7b9f7045027b39250b58aeacc07d3d4117d61dd97463a6fb226ad704cc0e79907bd.. | Daisy | Robinson | drobinson@icloud.com | 9835973538 | Administrator | No | 0 | 0 |
| 8 elasticbands123 | 30a56dc2d7177646f0b0bdc3895a370c8c145067236eb1bbfdc80785a9d0e1e23df94c3d564d42625e84.. | Sophie | Rivett | srivett@outlook.com | 387548735 | Standard | No | 0 | 0 |
| 9 FlowerMeadow111 | b5746f5e139b2b8eb548972a7003134cce65a0961200662d2f204a88ea2f158b1e8d0163d30f30f89a40400.. | Daisy | Yau | dyau@icloud.com | 3546653476 | Standard | No | 0 | 0 |
| 10 FrankRivett3000 | a6c86fd77852dd1bb64674e74745221f9ff1333c2eabbcb0ff08673726734848..2905ff7dc5050784155b2e36d.. | Frank | Rivett | friwett@hotmail.com | 93875937457 | Standard | No | 0 | 0 |
| 11 oGillick999 | 66f1070b5e070e3aa970f3320fbae12e22fb9e96694b090817cfdaec952cfde734127b98870170f6e8a4f1c.. | Alex | Gillick | lgillick@outlook.com | 843428749287 | Administrator | No | 0 | 0 |
| 12 Birmingham888 | 5830903a5a83315d90f712b453e9a9e105d07740649812af7341c876a57c26cc45266b79102b2dd02834e.. | Linda | Bone | lbone97@hotmail.co.uk | 0392743979 | Standard | No | 0 | 0 |
| 13 Root | bee041856e95dede886a5e8ff3b292b74e519ca7tabcbe584343add1eba2c2a28253689323a016c1187268.. | Alexandra | Robinson | s275931@uos.ac.uk | 30903853809 | Administrator | No | 0 | 0 |
| 14 barneyPurpleDinosaur | 98e2ec6325b2bea65fadb029be162c3580073ba0216ce45b7ffbab9085c37d8791d63c3bc71b3d67a023f5074c.. | Alice | Robinson | robertson97@icloud.com | 983749739378 | Administrator | No | 0 | 0 |

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

```

def login():
    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: 999999999) #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,) ) # updating the failed login attempts in the database
                conn.commit()
                cur.close()
            except:
                flash('Username or Password is incorrect!')

```

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,) ) # 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:
        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.

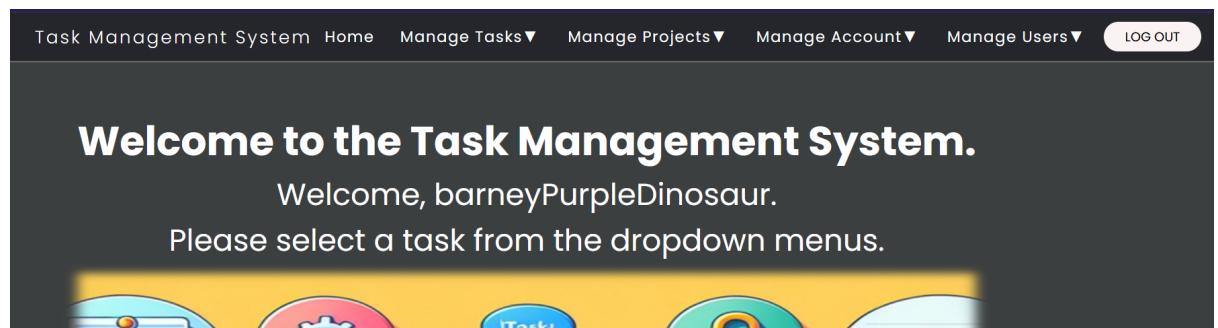


Figure 165 - User Home page after logging in. You will now be granted access to Task Management System webpages as your access is legitimate and has been authenticated.

| PROJECT ID | TITLE | DESCRIPTION | ASSIGNED TASKS |
|------------|------------------|--|---|
| 1 | Create Wrb App | Develop Web Application | Plan Web App, Create UML diagrams, Code Website, Test Website |
| 2 | project | project project | project project project project project project |
| 3 | Update Project 3 | Update Project 3Update Project 3Update Project 3 | task 1, task 2, task 3 |
| 5 | project xxxxxxxx | This is project xxxx | t1, t2, t3 |
| 6 | project | project | project |

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 = "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 to 0
    cur.execute(_sql: 'UPDATE users SET sessionID = ? WHERE username = ?', _parameters: (sessionID, username, )) #update the session ID in the database
    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.

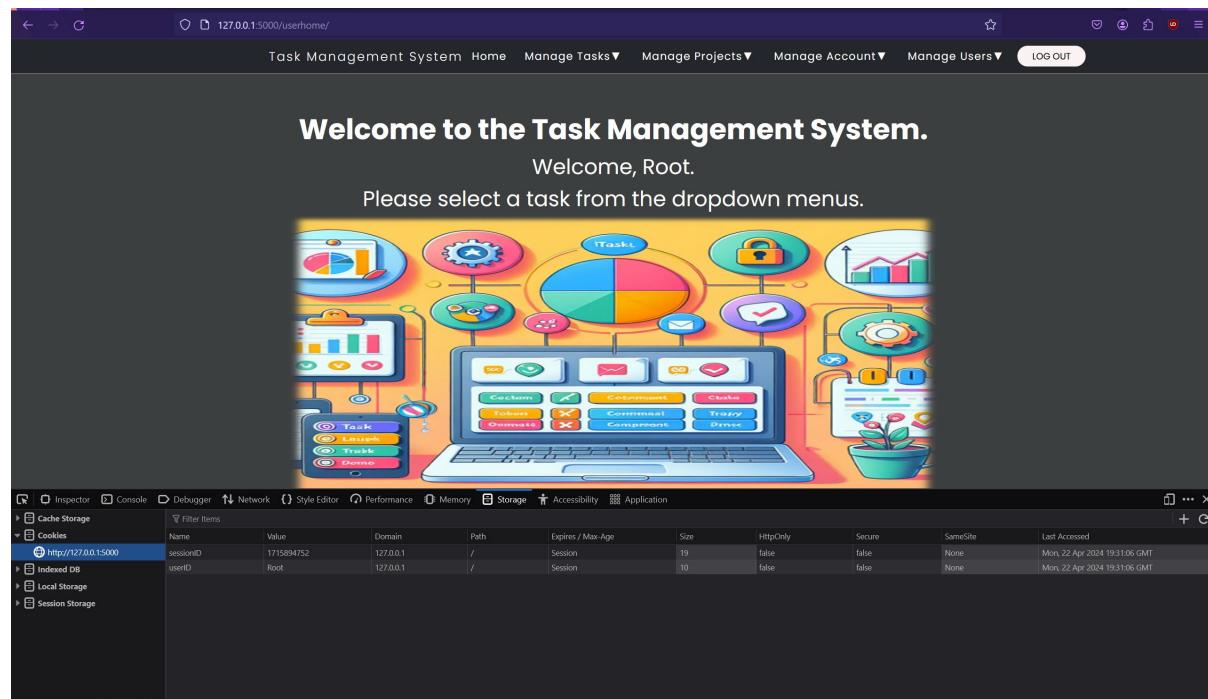


Figure 168 - UserID and SessionID cookies after logging in.

S275931

The screenshot shows the browser developer tools Storage tab. Under the Cookies section, there are two entries for the domain http://127.0.0.1:5000: 'sessionID' with value '1715894752' and 'userID' with value 'Root'. Other sections like Cache Storage, Indexed DB, Local Storage, and Session Storage are also visible.

Figure 169 - UserID and SessionID cookies after logging in.

This screenshot is identical to Figure 169, showing the browser developer tools Storage tab with the same two cookies ('sessionID' and 'userID') for the domain http://127.0.0.1:5000.

Figure 170 - UserID and SessionID cookies after logging in.

| username | password | forename | surname | email | phone | userType | blocked | failedLogi | sessionID |
|-------------------------|--|-----------|----------|-------------------------|--------------|---------------|---------|--------------|-----------|
| 1 Manchester000 | 491a0812d10f16546ca45ca12610224014028a204abe50f4c66decfd64f6ea0f7ca8a9d64ac88b3a9079aa08f4982... | Kane | Fenlon | ktenlon@hotmail.com | 42342342 | Standard | No | 0 0 | |
| 2 L'FenorVolcano | 7a6efbf087dc2b28efc3cf7651bcfab3f36cd6c9b8f5ea4c614af2640b7eb0934fd40a70b8ec148bf19688062fc8... | Lauren | Dunury | ldunury@msn.com | 234223424 | Standard | No | 0 0 | |
| 3 GemmaWheels | 41de2af1a1f7342236b05bc67d3ce58ae9612903e2bf413e6614d5940ea8e7fe846a5f1c24cd5a0f3b681cab7cb6... | Gemma | Bird | gbird@hotmail.com | 2323466 | Standard | No | 0 0 | |
| 4 kellyG | 7a6efbf087dc2b28efc3cf7651bcfab3f36cd6c9b8f5ea4c614af2640b7eb0934fd40a70b8ec148bf19688062fc8... | Kelly | Gates | kgates@outlook.com | 78624868243 | Standard | No | 0 0 | |
| 5 ethomas123 | b48297d22c02f065479b9cfdffdf12ae8e86450b8853367e023459b9fdff2ff99ba0d74246638a5a2d1b202c1eb... | Ellie | Thomas | ethomas@icloud.com | 836446834 | Standard | No | 0 0 | |
| 6 ArasakaCE0123 | 2bb83ee8483bf844d442933cd1d38b2700e44d551e960765481884d926ffe34347d8a0f02233b6b395b75175... | Bill | Thomas | bthomas@hotmail.com | 012434376876 | Standard | No | 0 0 | |
| 7 DandyLion000 | 295d9a3c654a2b52da7b9f97045027b39250b58aeacc07dcf3d117d61dd97463a66fb226ad70c4cc0e799f07bd... | Daisy | Robinson | drobinson@icloud.com | 983597358 | Administrator | No | 0 0 | |
| 8 elasticbands123 | b5056cd2d21171646fb0b0cd3895a37f0c8b145067236eb1bb0ff80785a91d0e1e23d994c34d564d42625e84... | Sophie | Rivett | rivett@outlook.com | 387548735 | Standard | No | 0 0 | |
| 9 FlowerMeadow111 | b57465ec139b2b8eb54892da70031c34cce65a0f961200662d2104a88eee2f158bc1e80d0f6d3b0f309a0400... | Daisy | Yau | dyau@icloud.com | 3546653476 | Standard | No | 0 0 | |
| 10 FrankRivett3000 | a8c86fd717852dd1bb64674e7f475221f3f9133cc2e8bbc0ff3673726734a48c290f57dc8050784155b2e36d... | Frank | Rivett | rivett@hotmail.com | 93875937457 | Standard | No | 0 0 | |
| 11 aGillick999 | 66f1070b5e970e3aa97097320fbae12e22ff8e96694b908017fdacac6952cfd6734127b888701706ea8a4e4ff1c... | Alex | Gillick | lgillick@outlook.com | 843428749287 | Administrator | No | 0 0 | |
| 12 Birmingham888 | 583090a35a9f838315d90f712b435e9a39e105d07740649812af2341c876a57c26cc4526b7910c2b2d02834... | Linda | Bone | lbone97@hotmail.co.uk | 039274397 | Standard | No | 0 0 | |
| 13 Root | bee04f1856e95dede886a5e8ff3b292b2f74e51f9ca7fabcbc584343ad4d1eba2c2a28253689233a0f6c1187268f... | Alexandra | Robinson | s275931@uos.ac.uk | 30903853809 | Administrator | No | 0 1715894752 | |
| 14 barneyPurpleDinosaur | 98e2ec6325b2baa65fad829be16e2c358007f3ba0216ce45b7ffbab9085c3f7d8791d63cbc71b3d67a823f5074c... | Alice | Robinson | robertson497@icloud.com | 983749739378 | 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.

This screenshot shows the browser developer tools Storage tab. After logging out and reusing the session and user cookies, the session ID has changed to '1715894752' and the user ID is still 'Root'. The other sections like Cache Storage, Indexed DB, Local Storage, and Session Storage are also visible.

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.

| Name | Value | Domain | Path |
|-----------|------------|-----------|------|
| sessionID | 1715894752 | 127.0.0.1 | / |
| userID | Root | 127.0.0.1 | / |

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 | 491a0812d10f16546ca5ca12610224014028a204abe50f4c66decd64f6ea40f7ca8a9d64ac88b3a9079aa0f4982... | Kane | Fenlon | kfenlon@hotmail.com | 42342342 | Standard | No | 0 | 0 |
| 2 | lFenlonVolcano | 7a6efb0b087d2cb28efc3cf7651bcfa3f36cd9c9fb5ead4c614a62640b7eab0934fd070bbec148bf19688062fc... | Lauren | Durury | ldurury@msn.com | 234223424 | Standard | No | 0 | 0 |
| 3 | GemmaWheels | 4fdfe2a1a1af7342236bd5bcd6743ce5bae9612903e2b4143e6614d5940ea8e7fe846a5f1c24cd5a0f3b61cab7cb6... | Gemma | Bird | gbird@hotmail.com | 2323466 | Standard | No | 0 | 0 |
| 4 | kellyG | 7a6efb0b087d2cb28efc3cf7651bcfa3f36cd9c9fb5ead4c614a62640b7eab0934fd070bbec148bf19688062fc... | Kelly | Gates | kgates@outlook.com | 78624868243 | Standard | No | 0 | 0 |
| 5 | ethomas123 | b48297d22c02f065479b5cfdf1f12ae8e6450b8853367e023459bdf2f89cb9d742d6d638a5a2da1b202c1eb... | Ellie | Thomas | ethomas@icloud.com | 836446834 | Standard | No | 0 | 0 |
| 6 | ArasakaCEO123 | 2bb8b3ecc8d3bf844df4293c3d1d38b2700ee4d5c551e960765481884926ff34347d8a0f2233b6b395675175... | Bill | Thomas | bthomas@hotmail.com | 012434376876 | Standard | No | 0 | 0 |
| 7 | DandyLion000 | 295da93c554a2b52da7b9f7045027b39250b58aceacd07cf3d117d61dd97463a66fb226ad704cc0ea79907... | Daisy | Robinson | drobinson@icloud.com | 9835973538 | Administrator | No | 0 | 0 |
| 8 | elasticbands123 | 30a5fd6c2d171646f0b0bdc3b395a370c81a5067236eb1bbdf8d0785a91d0e123d94c34d564d42625684... | Sophie | Rivett | srivett@outlook.com | 387548735 | Standard | No | 0 | 0 |
| 9 | FlowerMeadow111 | b5746f5ec139b5b6eb54892da70031c3cce65a9f961200662d2f204a88ee2f158bc1e8d0f16d30f3089a0400... | Daisy | Yau | dyau@icloud.com | 3546653476 | Standard | No | 0 | 0 |
| 10 | FrankRivett3000 | a8c86fd7f7852dd1bb64674e4f74f5213f91333cc2eb8b8c08f3673726734a48c290f57dcb50930785a91d0e... | Frank | Rivett | frivett@hotmail.com | 93875937457 | Standard | No | 0 | 0 |
| 11 | oGillick999 | 66f1070b5e970e3a970f3320bae1f2e2fbb9e666949e908017cfdaac6952f6de734127b88701706e8a44f1c... | Alex | Gillick | lgillick@outlook.com | 843428749287 | Administrator | No | 0 | 0 |
| 12 | Birmingham888 | 583090a35a838315d90f712b435e9a39e105d077406f49812a2f2341c876a657c26cc4526b67910c2b2dd02834e... | Linda | Bone | lbone97@hotmail.co.uk | 0392743979 | Standard | No | 0 | 0 |
| 13 | Root | bee04f1856e2685dede868a5e8f3b292b274e5191ca7abce584343add1eba22a28253689323a016c11187268f... | Alexandra | Robinson | s275931@uos.ac.uk | 30903853809 | Administrator | No | 0 | 0 |
| 14 | barneyPurpleDinosaur | 98e2ec6325b2beab5f5db82b82be16e2c358007fbab0216c45b7ffbab9085c5f7a87c91d63c9c71b3d7a7b23f5074c... | Alice | Robinson | robertson97@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 |
| oGillick999 | 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 | robertson97@icloud.com | 983749739378 | Administrator | No |

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

Task Management System

Home Manage Tasks▼ Manage Projects▼ Manage Account▼ Manage Users▼ LOG OUT

Add New Task

Task ID:

Must be a whole number.

Title:

Title text here.

Description:

Description text here.

Due Date:

dd / mm / yyyy

Priority:

Type "High", "Medium", or "Low" (Case Sensitive).

Status:

Type "Not Started", "Planned", "In Progress", "Completed", "Overdue" or "Cancelled" (Case Sensitive)

Project ID:

Must be a whole number.

ADD TASK

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

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

The screenshot shows a dark-themed web application. At the top, there is a navigation bar with links: 'Task Management System', 'Home', 'Manage Tasks▼', 'Manage Projects▼', 'Manage Account▼', 'Manage Users▼', and 'LOG OUT'. Below the navigation bar, the main title 'Delete Task' is centered. The first form field is labeled 'Task ID:' with a placeholder 'Must be a whole number.' and a numeric input field. The second form field is labeled 'Title:' with a placeholder 'Title text here.' and a text input field. A pink button at the bottom left of the form area is labeled 'DELETE TASK'.

Figure 178 - Only administrators can delete tasks and projects.

The screenshot shows a dark-themed web application. At the top, there is a navigation bar with links: 'Task Management System', 'Home', 'Manage Tasks▼', 'Manage Projects▼', 'Manage Account▼', 'Manage Users▼', and 'LOG OUT'. Below the navigation bar, the main title 'Change User Account Type' is centered. The first form field is labeled 'Username:' with a placeholder 'Enter username of account type to be changed.' and a text input field. The second form field is labeled 'User Account Type:' with a placeholder 'Type Administrator or Standard (Case Sensitive).' and a text input field. A pink button at the bottom left of the form area is labeled 'CHANGE USER ACCOUNT TYPE'.

Figure 179 - Only administrators can manage other users.

The screenshot shows a dark-themed web application. At the top, there is a navigation bar with links: 'Task Management System', 'Home', 'Manage Tasks▼', 'Manage Projects▼', 'Manage Account▼', 'Manage Users▼', and 'LOG OUT'. Below the navigation bar, the main title 'View Users' is centered above a table. The table has a header row with columns: 'USERNAME', 'FIRST NAME', 'LAST NAME', 'EMAIL ADDRESS', 'PHONE NUMBER', 'ACCOUNT TYPE', and 'BLOCKED'. There are 15 data rows, each representing a user with their corresponding details. The last two rows are highlighted in pink.

| 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 | 01243476876 | 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 | frigett@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.

Task Management System Home Manage Tasks▼ Manage Projects▼ Manage Account▼ Manage Users▼ LOG OUT

Reset Task Management System

WARNING: This will be permanent.

Administrator Password:

Confirm Password:

Confirm password for factory reset.

RESET TASK MANAGEMENT SYSTEM

Figure 181 - Only administrators can perform administrative tasks.

| USERNAME | FIRST NAME | LAST NAME | EMAIL ADDRESS | PHONE NUMBER | ACCOUNT TYPE |
|----------|------------|-----------|-------------------|--------------|---------------|
| Root | Alexandra | Robinson | s275931@uos.ac.uk | 30903853809 | Administrator |

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

| USERNAME | FIRST NAME | LAST NAME | EMAIL ADDRESS | PHONE NUMBER | ACCOUNT TYPE |
|---------------|------------|-----------|-----------------------|--------------|--------------|
| Birmingham888 | Linda | Bone | lbone97@hotmail.co.uk | 0392743979 | Standard |

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

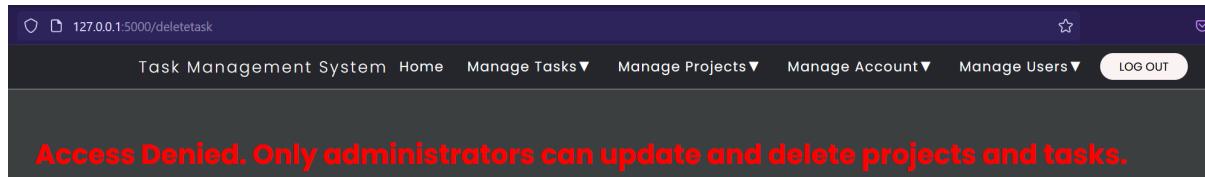


Figure 184 - Standard users cannot update and delete tasks and projects. They will receive an Access Denied error instead.

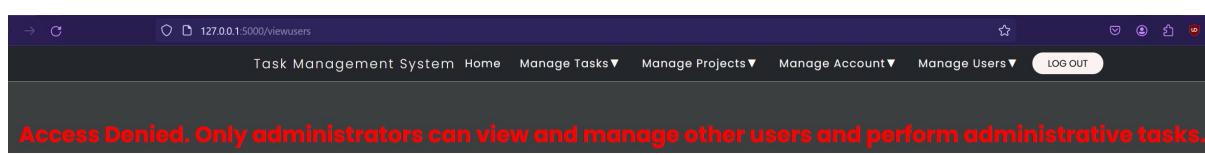


Figure 185 - Standard users cannot manage other users and perform administrative tasks. They will receive an Access Denied error instead.

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.

Task Management System Home Manage Tasks▼ Manage Projects▼ Manage Account▼ Manage Users▼ LOG OUT

Block User

Username:

BLOCK USER

Figure 186 - Administrators can manually block other users.

Task Management System Home Manage Tasks▼ Manage Projects▼ Manage Account▼ Manage Users▼ LOG OUT

User blocked successfully!

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 | Daiy | 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 | igillick@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 | robertson97@icloud.com | 983749739378 | Administrator | No |

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

Task Management System Home CREATE ACCOUNT LOG IN

Log In

Username

Password



LOG IN

Figure 188 - Blocked user attempts to log in.

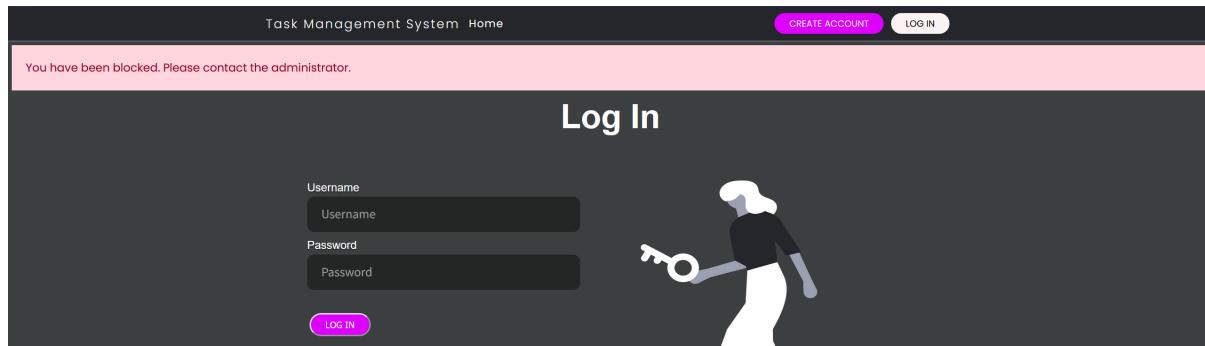


Figure 189 - Login attempt fails because the user is blocked.

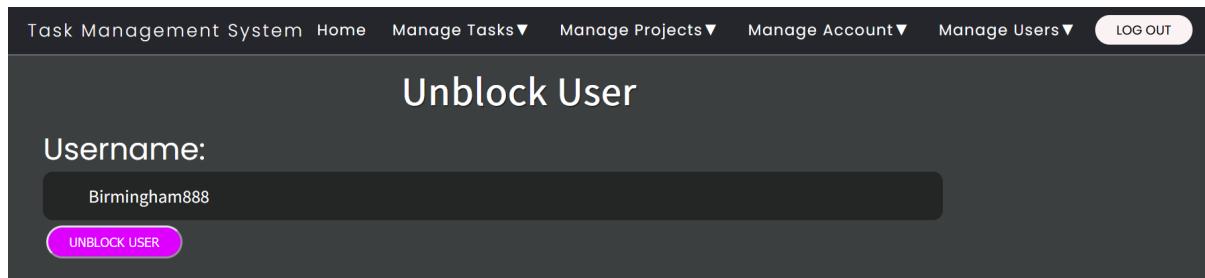


Figure 190 - Administrators can unblock blocked users.

The screenshot shows the 'View Users' page. The top navigation bar includes 'Task Management System Home', 'Manage Tasks▼', 'Manage Projects▼', 'Manage Account▼', 'Manage Users▼', and 'LOG OUT'. A pink banner at the top says 'User unblocked successfully!'. Below is a table with columns: USERNAME, FIRST NAME, LAST NAME, EMAIL ADDRESS, PHONE NUMBER, ACCOUNT TYPE, and BLOCKED. The table lists various users, including 'Birmingham888' whose status has been updated from 'Yes' to 'No'.

| USERNAME | FIRST NAME | LAST NAME | EMAIL ADDRESS | PHONE NUMBER | ACCOUNT TYPE | BLOCKED |
|----------------------|------------|-----------|-------------------------|--------------|---------------|---------|
| Manchester000 | Kane | Fenlon | kfenlon@hotmail.com | 42342342 | Standard | Yes |
| LFentonVolcano | 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 |
| gGillick999 | Alex | Gillick | lgillick@outlook.com | 843428749287 | Administrator | No |
| Birmingham888 | Linda | Bone | lbone97@hotmail.co.uk | 0392743879 | 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.

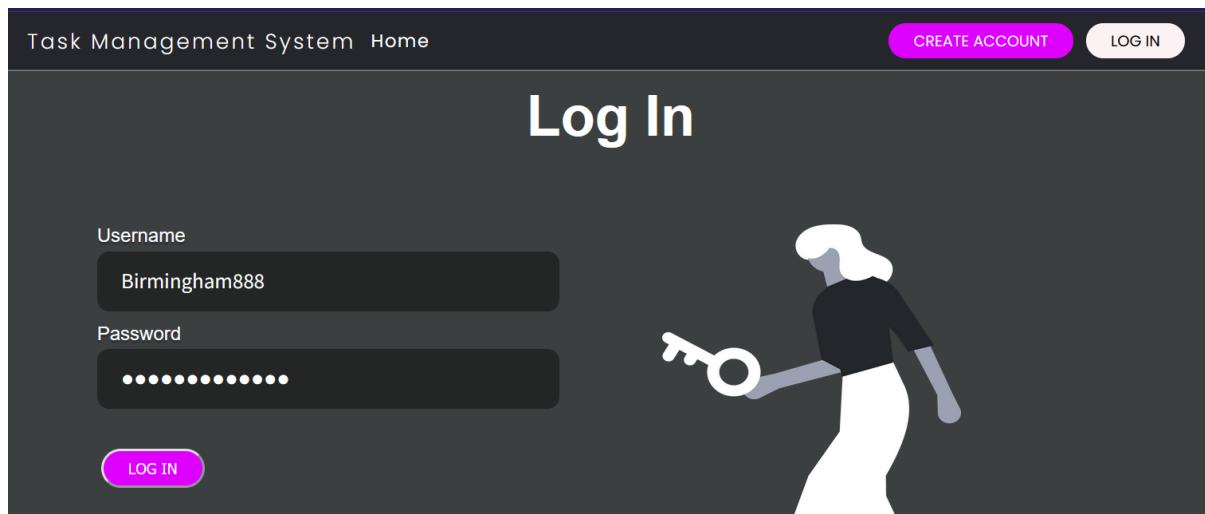


Figure 192 - Unblocked user attempts to log in.



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.

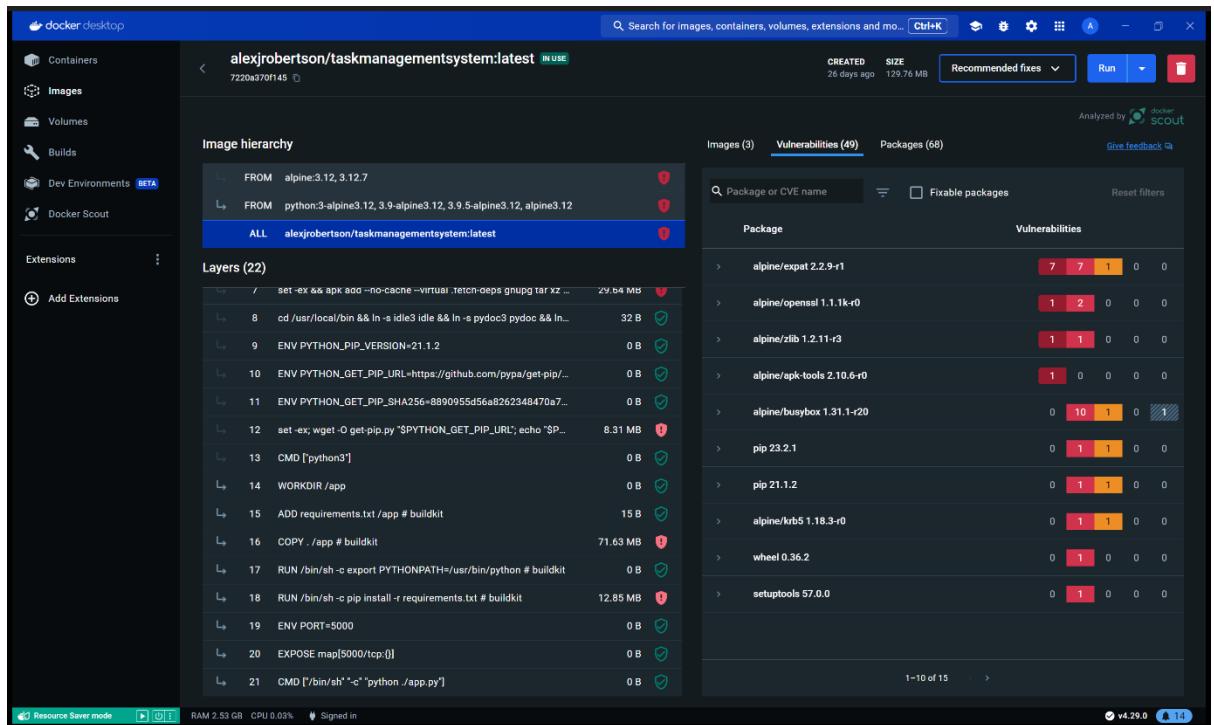


Figure 194 - Docker has detected vulnerabilities with the Docker container deployment of the Task Management System to do with Alpine Linux and pip. This is a security threat.

- 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.

The screenshot shows the SonarCloud interface for a project named 'task-management-system'. The left sidebar includes links for Overview, Main Branch, Pull Requests, Branches, Information, and Administration. The main area displays a 'Security Hotspots' section with a summary: 0.0% Security Hotspots Reviewed, 50 Security Hotspots to review, and a Review priority set to High. A specific hotspot is highlighted: "'SECRET_KEY'" detected here, review this potentially hard-coded credential. This hotspot is categorized under Authentication and is marked as a Hard-coded credential that is security-sensitive (python:\$2068). The status is 'To Review'. A 'Review' button is present. Below this, code snippets from 'app.py' are shown, with line 23 circled in red: 'app.config['SECRET_KEY'] = 'hdyrasbjdgce85gcsl''. A tooltip for this line states: "'SECRET_KEY'" detected here, review this potentially hard-coded credential.

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.

The screenshot shows the 'Pick an Instance Type' section of the Render.com interface. On the left, there's a sidebar with links: Events, Logs, Disks, Environment, Shell, Previews, Jobs, Metrics, Scaling, and Settings. The main area is titled 'For hobby projects' and contains a 'Free' plan (\$0/month) with 512 MB RAM and 0.1 CPU. Below it, under 'For professional use', are several paid plans: 'Starter' (\$7/month), 'Standard' (\$25/month), 'Pro' (\$85/month), 'Pro Plus' (\$175/month), 'Pro Max' (\$225/month), and 'Pro Ultra' (\$450/month). A note at the bottom states: 'Need a [custom instance type](#)? We support up to 512 GB RAM and 64 CPUs.' At the bottom right are 'Cancel' and 'Save Changes' buttons.

| Plan | Price | RAM | CPU |
|-----------|---------------|--------------|---------|
| Free | \$0 / month | 512 MB (RAM) | 0.1 CPU |
| Starter | \$7 / month | 512 MB (RAM) | 0.5 CPU |
| Standard | \$25 / month | 2 GB (RAM) | 1 CPU |
| Pro | \$85 / month | 4 GB (RAM) | 2 CPU |
| Pro Plus | \$175 / month | 8 GB (RAM) | 4 CPU |
| Pro Max | \$225 / month | 16 GB (RAM) | 4 CPU |
| Pro Ultra | \$450 / month | 32 GB (RAM) | 8 CPU |

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.

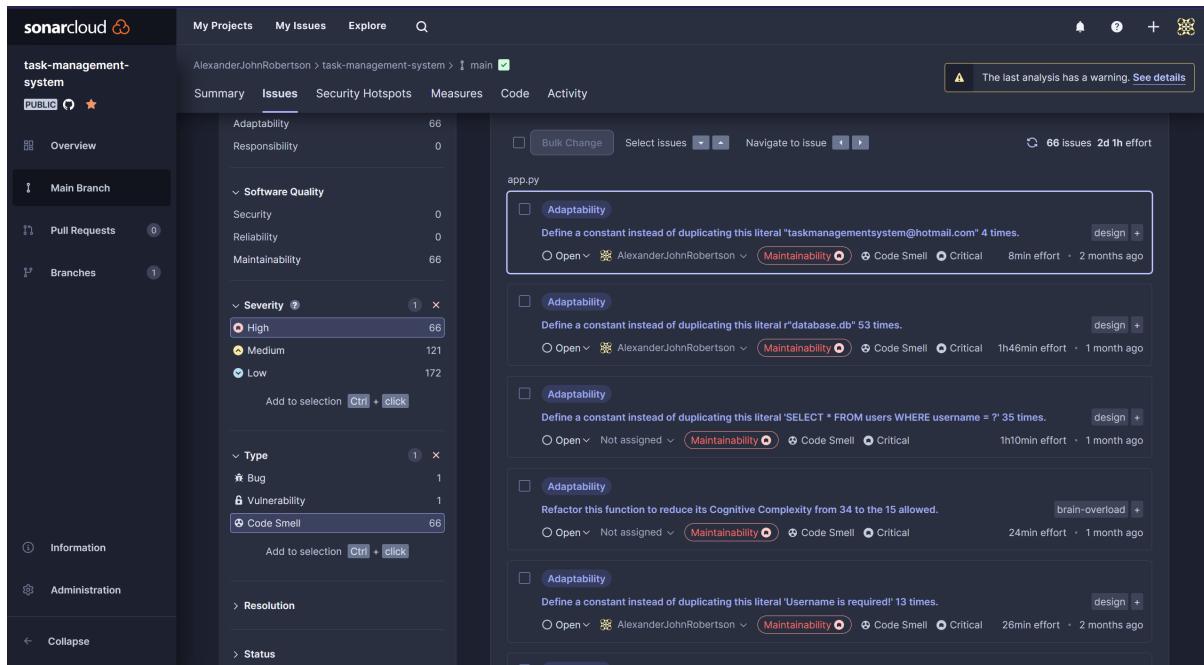


Figure 198 - Code smells have been detected by SonarCloud.

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 (SEPP), 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).