DLMCSPCSP01 Project: Computer Science Project

Research Report

Task Master

Author: Sandeep

Enrollment Number: 102210527

Study Program: Master’s in Computer Science

Table of Contents

List of Figures

Task Master

1. Introduction

In this 21st century, the time play an important part in the individual's life. In this fast-paced world, an individual is surrounding with several daily tasks or work. It is challenging for an individual to manage the time efficiently. In simple words, for a person who is dealing with his professional life is challenging to manage the personal and the professional life together. Similarly, for a person who is studying also have a hard part to manage the studies and personal life together. Moreover, the person who is studying and working all together, for him to manage the time is very challenging to deal with the time or manage the time efficiently. As this is the digital world, and the society is running very fast, an individual needs to manage the time perfectly. For this rapid change, an individual demands the tools that are simple, easy to use, and the tools would save the time for managing the daily, weekly, or monthly tasks. The traditional tools of time management are becoming less effective because they do not understand the nature of individual daily life according to the current time or situation. The solution is to create a simple and user-friendly tool that help the individuals to manage their time efficiently. Moreover, the digital tools would help in this situation in more accurate manner because as this generation is always stick with the digital tools like smartphones, laptops, etc. This is the context, in which we can see the stage of the problem.

Now the problem in this context is this that how can an individual manage the time or the daily tasks. The nature of the problem is very complex because of the uncertainty of the daily routines. For example, a university student has 6 exams that must clear in one semester and one semester runs for 6 months. If student decided that 2 exams, he must clear in 2 months that means 60 or 61 days for 2 exams. Then, 60 or 61 days are multiple to the 1440 or 1464 hours. Now, let's assume that the student consumed 480 hours for sleeping 360 hours for cooking 60 hours for grocery shopping and 60 hours for washing or etc. After consuming these hours, a student is left with 480 hours and that he is giving to his study but the uncertainty here is that there are no personal tasks is involved in this schedule, for example, roaming, clubbing etc. This is just a simple example related to the student but if same scenario would apply to the professional individual, then we find number of uncertainties like family problem, or business problems or a job-related tasks or problems. These types of uncertainties make the individuals lives more complex and stressed. To resolve these types of problems individuals may need the digital tools that deal with these problems and individual can adjust the time as per the need. They need the system that left behind the traditional boundaries and provide the modern solution to deal with these types of uncertainties and the problems. In this context of problem, in which individuals are finding hard way to manage the time, we develop the project to deal with this problem.

In this modern time, we cannot deny the relevance of the Task Management System (Task Master) that we developed. In this era, to manage the personal and professional life is a challenging part. If the important tasks are not completed on the time, then it leads to the stress and anxiety. It may then impact on economic and social spheres. In the professional context, the poor time management can lead to the missed deadlines and may lead to the financial loss also. In the same way, in other context like academic life or both the academic and working life, the tasks not completed then it may lead to submit the assignments not on time that then may lead the delay in the results. If individuals manage both personal and professional tasks through unified system, then it leads to the better mental health and productivity. Individuals then may be able to resolve the complex tasks easily and will have the leadership qualities if they learn how to manage the time systematically. Here, we are just giving the importance to the individuals but if we dive deeper then this Task Management System (Task Manager) can be used by the organizations as well. Through this system they easily can divide the tasks of the organizations by giving the name to the relevant person or group who working on it. Anyhow, the system is not only addressing the prevalent problem but also set a new notion that how a personal and professional task can be managed in a digital context.

The motivation behind developing this Task Management System (Task Master) origin from the personal and the professional experience. The experience motivates to create an impactful technological solution. After having online research and doing surveys with the friends, it provides the need of the system that helps to manage the time systematically. If the daily routine is managed or grouped systematically then it produces the productivity and the brain would run according to the nature, rather to feel much stress. The solution, from this motivation, that we developed is not just another tool but also the solution, in which people can easily divide their daily, weekly, or monthly routines into the groups or tasks, so they can easily complete these tasks perfectly and get no stress. By addressing the challenge, the aim to use the technology in the manner so the individuals can manage or balance the personal and the professional life together.

This research report is divided into number of sections and all the sections will provide the deep understanding of the Task Management System (Task Manager). The report provides the information about the related work that who else is trying to solve this time managing problem. Then the technical backgrounds, that which technology is used to develop the Task Management System. After this we will see the methodological approach, the implementation, and the testing sections of the research report. The conclusion section will provide the information related to the relevant findings, reflect on the project's impact and limitations, and outline future enhancements. The references that help to create the research report also would be mentioned in it. In the end, in this research report reader will understand the need of the project and how this project is resolving the problem of individual's modern life.

2. Main Body

2.1 Related Work

This section is divided into four following subsections:

Overview of Existing Task Management System, Analysis of Limitations in Current Systems, Review of Technological Innovations, and Comparative Analysis. One by one we see these points to understand the “related works” on this Task Management System.

2.1.1 Overview of Existing Task Management Systems

Nowadays, the Task Management Systems play a pivotal role in this mega trend digitalization and these systems help individuals to manage their daily routines effectively. These systems are simple task-based system in which individuals can easily add their daily task to stay align with their work. The digital age helps individuals to move forward from paper-based management to using software to create a kind of digital timetable to follow the daily routines. This digital innovation is a kind of solution that shift the focus of the people to manage their personal and professional responsibilities together.

There are number of examples that show us how the digital platforms help individuals to manage their time. For example, Asana and Trello are the well-known tools that deal with the task management. These tools become integral to modern workflow management. According to the Jones (2020), Asana allows for extensive project customization which enhances team collaboration and productivity by up to 30%. But according to the Smith (2019), These tools are exceled at professional project management (often overlook the integration of personal task management) that can lead to a disconnection between personal and professional life. So, the gap underscores the need for a more integrated approach. According to Lee (2018), a successful task management system must cater equally to both personal and professional dimensions to enhance overall user efficiency. It means that the task management system should be a kind of system that have features in which individual easily can add the tasks to manage both personal and professional work. There is another tool that deal with the task management system is Microsoft To-Do. It integrates with the Office 365. The users who already embedded with the Microsoft ecosystem for them it is a more perfect experience to use it. This Microsoft To-Do emphasizes on ease of use and integration.

These are some existing task management systems that allows users to add their daily tasks, but these systems may have some limitations. The system that is required for this modern time should be user friendly and as simple as possible. Because the goal of the task management system is to save the time of the individual so if the system is as simple as possible that means individual will spend less time over the application and simply just add the required tasks that he/she wants to complete or necessary for them to complete on time.

2.1.2 Analysis of Limitations in Current Systems

If we analyse the current systems, then we find that these systems have some limitations and gaps in the functionality. The systems may be rich in features and may have multiple features but if it has complexity in use then it becomes hard for users to use it and they may consume more time to use the application only. The rule is to save the time as much as possible. The main concern is to adapt the user need perfectly and these features maybe fail in that condition. Because user need maybe change according to the different context or uncertainties. There are number of task management tools that are designed with the one-size fit for all approaches and that may fail to address the specific requirement of the user related to personal preferences or professional environment. For example, Asana as it offers extensive project management features and that make complexity for the individual users who are seeking the simplicity for daily tasks. Similarly, the Microsoft-To Do is also the task management tool that may not like Asana but that only provide the feature to add task without any description of the task and due date and there is no option to remove the task or update the task as per user need. These types of usually deal with the organizations those looking professional tasks to be complete and they distribute among their employees. But the main task management system should be compatible for organizations as well as for the individuals also.

It is really a challenging part to integrate the personal and professional task management in single platform. If user use different tool for different need, then it would consume more time and the extra load will build over the individual. For example, for personal tasks user use Microsoft To-Do and for professional tasks user use Asana then it be complex to manage both the application together and it would build more pressure or stress over individual. These tools somewhere may be able to manage the professional life of the individuals but the other scenarios like family gathering, family parties or events, personal appointments, or even health-related reminders for these scenarios these tools have no features or lack in these features. These are the small but crucial tasks maybe not deal with daily routine but very important in individuals' life. The current tools are specifically designed for the business point of view and that is a good move by any of the developer. But the core or the base of that designed tool should meet the criteria or the need of the user.

These limitations of the current systems provide the need to develop the system that would deal with the personal and professional tasks of the individuals. Furthermore, the tools like Asana, Microsoft To-Do, Airtable, ClickUp, Trello, etc all do not provide specific features related to the academic tasks, but an individual can add any academic task in it but may not get that feeling of that feature he may looking for. The purpose of task management system should not only be focused on organizations and their daily task but also may deal with the professional task all together for better productivity.

2.1.3 Review of Technological Innovations

In recent years, the development in technology has powerful impact on creating the task management systems. It enhances their functionalities and user interface. The integration of cloud computing provide the meaningful impact and it is transformative. It allows users to use the application in various devices and platforms. It allows users to access their tasks anywhere and at any time. According to Smith (2021), the cloud computing has fundamentally changed how tasks are managed, moving away from static lists to dynamic, and accessible databases that are continuously updated in real-time. Furthermore, the upgradation in smartphone technology provides the wings to the task management applications. It offers the notifications and the reminders related to the tasks. It integrates task management in daily life more seamlessly than ever before in earlier times. The technological enhancements would improve the efficiency of task management systems and it also align with the modern user's mobile-centric lifestyle.

The role of Artificial Intelligence and Machine Learning algorithms have the meaningful impact in enhancements of task management systems. Artificial Intelligence and Machine Learning helps the system to be more intuitive and responsive to the user needs. The use of Artificial Intelligence is to capture the user interaction or behaviour with the system and the tasks that user adding every day or week. Then it may help users to understand his pattern of daily routine and help system to predict the users' daily routine. In the same manner, Machine Learning helps the system to check or analyse the past behaviour of the user with the system then it able to update the system according to the user needs. The Artificial Intelligence is more ahead too able the predict of user daily routine but also can provide the feature of voice recognition and adjust the time or task without need of the touch of user with the device. Simply, the user can speak over device and can add the tasks as he wants, these technological innovations may help the individual to manage the tasks or save the time efficiently.

The project that we developed in the initial phase has the simple functionality to add the tasks manually but with the use of Artificial Intelligence and Machine Learning would upgrade the project for more user-friendly environment. The review of the technological innovations provides the idea to extend the project according to the need of the users. As the role of the task management system is deal with the time and saving time is the main priority of the project.

Now, we move forward to comparative analysis the related work. With this we get more clarity with the nature of the project that we developed that it would in real the solution of the real-world problem or not. As, with the comparative analysis will provide the valuable insights of the project in the deep manner. It then helps to align the research report with the project and that is the base of the research report.

2.1.4 Comparative Analysis

In this sub-section the focus is on the solution versus existing tools and the unique features. As we identify already some of the features that other tools have but may not specifically created for the actual need of the individuals. We see that what new features we have in the project that make it difference from other available tools. In actual the task management landscape is already populated with various solutions but provide lacks flexibility and user-centric design. These characteristics ‘flexibility’ and ‘user-centric design’ is the central part of the project that we developed.

Other tools are more organizations centric tools that mostly design for the professional meetings point of view. In which company provide some tasks to the employees and employees are working on it and they have deadlines to complete it. Unlike this, the application that we developed is more individual centric in which individual may add the task according to his time and priority. Because, if individual add the task according to his capability to complete the task, then the productivity will automatically come out from the task.

This feature sets our solution or project apart from other applications like Asana and Trello. As these two applications are more focused on manual task entry and static scheduling. Although the project or solution that we developed may deal with manual task entry but in further development can be use the Artificial Intelligence to ease the solution more effectively. Moreover, the developed application provides one interface in which easily can add the tasks related to the need rather to navigate the interface here and there to understand the webpage first and consume the unnecessary time in it. The sophisticated and the simple task management system is the main priority of the project.

The Task Management System (Task Master) that have been developed by us has the unique features that missing in other platforms. The first feature the simplicity of the interface that make it different from other platforms. It simply provides the adding task feature in which individuals simply add their tasks, describe about it and remove or update according to the need. Moreover, they also add the due date and immediately after completion of the task in which task would be the next task to complete and remove option also available that is not given in another platform like Microsoft To-Do.

Here we find that in this section that our project contributes new insights and improvements. The new insights like voice recognition tools may developed by Artificial Intelligence and the prediction for future tasks of the individuals. Moreover, the improvements can be using the Machine Learning algorithms so can assume or track the past behaviour of the user to add his/her task so for future update him/her to update his tasks according to his/her need. Now, we move forward to see the technical background of the project.

2.2 Technical Background

In this section the focus is on the techniques that have been used to develop the Task Master. In the development phase of the project, we mentioned the main points of the technical background. For example, technologies (HTML, CSS, and Python), frameworks (Flask), architecture (Client-Server model), and database (SQLite). Now, we move forward in this section with the detailed description of the technical backgrounds of the project. This section is divided into four sub-sections, and it provide us the detailed description of the technical backgrounds.

2.2.1 System Architecture Overview

The architecture of the project is based on three main technologies. The front-end is built with HTML and CSS. The back-end is built with Python using the Flask framework, and a SQLite database for data management. In the system, the front-end communicates the user inputs to the Flask-powered back-end. Then, the Flask-powered back-end processes the user inputs or requests and interacts with the SQLite database to retrieve or update the data. It also ensures that the data flow without any pause and user experience should be consistent. To handle the session management and authentication, the application utilizes Flask middleware. It ensures secure and powerful interactions between user interface and the database.

As we know now, that the front-end of the project is developed by using HTML and CSS, so it provides the clean and intuitive user interface so it can facilitate easy navigation and interaction for users. The back-end as developed using the Python language, so it utilizes the Flask framework. This framework handles all server-side logic (data processing, request routing, and session management) to ensure secure and efficient operation of the application. Moreover, for the database SQLite is being used in the project. It handles the data storage, retrieval, and management without the need of separate server setup.

The components in the system communicate via HTTP requests. At the first, through forms and API calls, the front-end sends user inputs to the back-end, then the back-end process the user inputs and interacts with SQLite database to retrieve or store the data. For the secure data transmission, the system utilizes the HTTPS protocols. It with the help of Flask's built-in security features for data encryption and session management safe the user data from unauthorized access and ensure while transmitting the data integrity should be there.

The architecture of the system is designed to be scalable (Flask's ability to handle increasing loads by integrating with webservers) to enhance the overall system performance. The modular nature of Flask allows for future adaptation, to enhance scalability and maintenance efficiency.

The main reason to choose this architecture for the system for its simplicity and effectiveness. As, it is effective in small to medium-scale projects. Because Flask provides a lightweight

framework that is suitable for rapid development and easier maintenance. The main benefit of choosing this architecture is that system remains adaptable and efficient.

Now, in below we can see some of the UML diagrams that would show us about the system architecture, and we get the overview of the system:

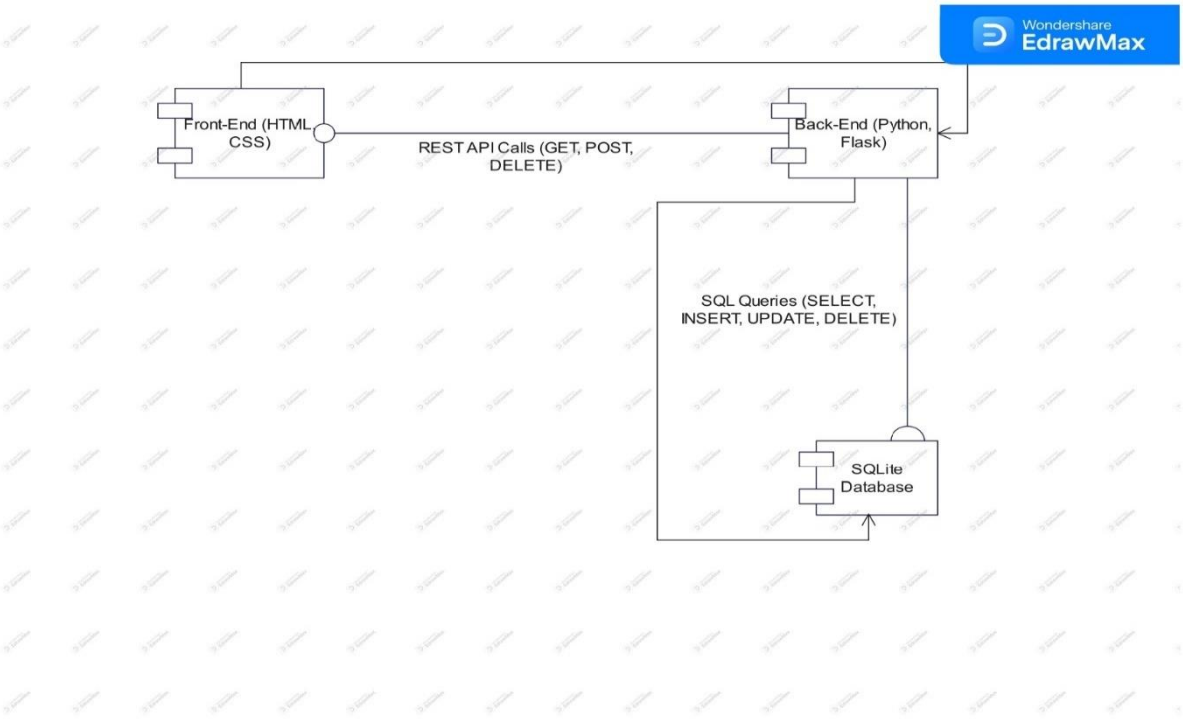

Figure 1: UML Diagram of System Architecture

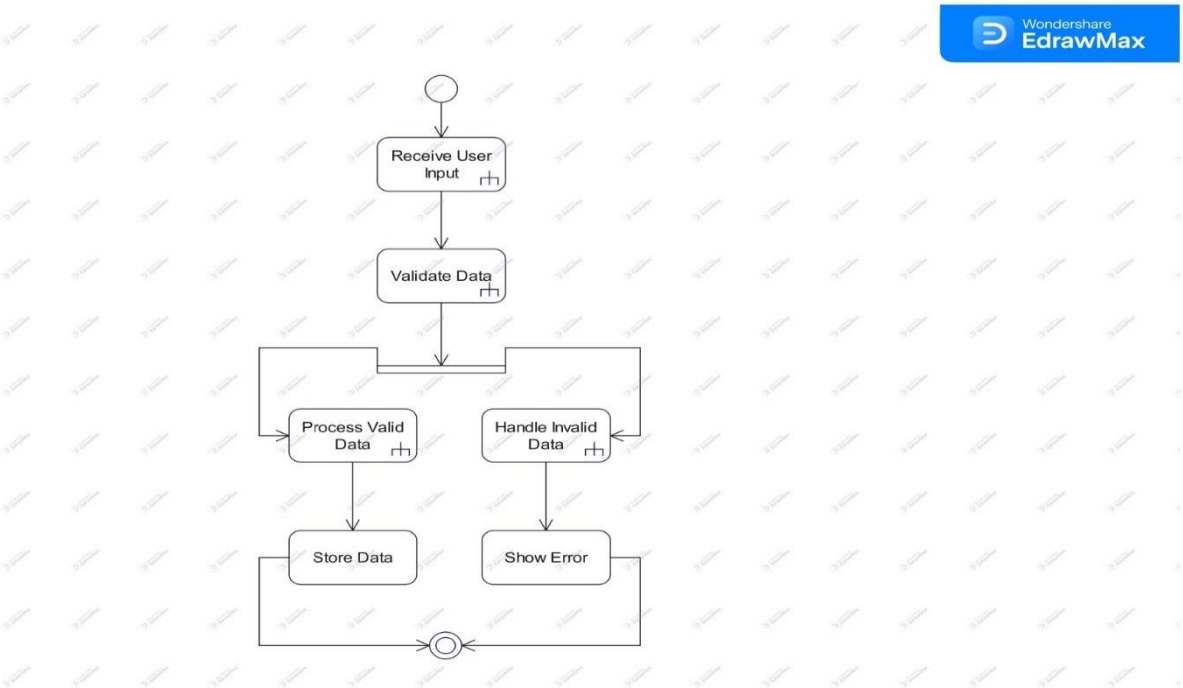

Figure 2: UML Activity Diagram

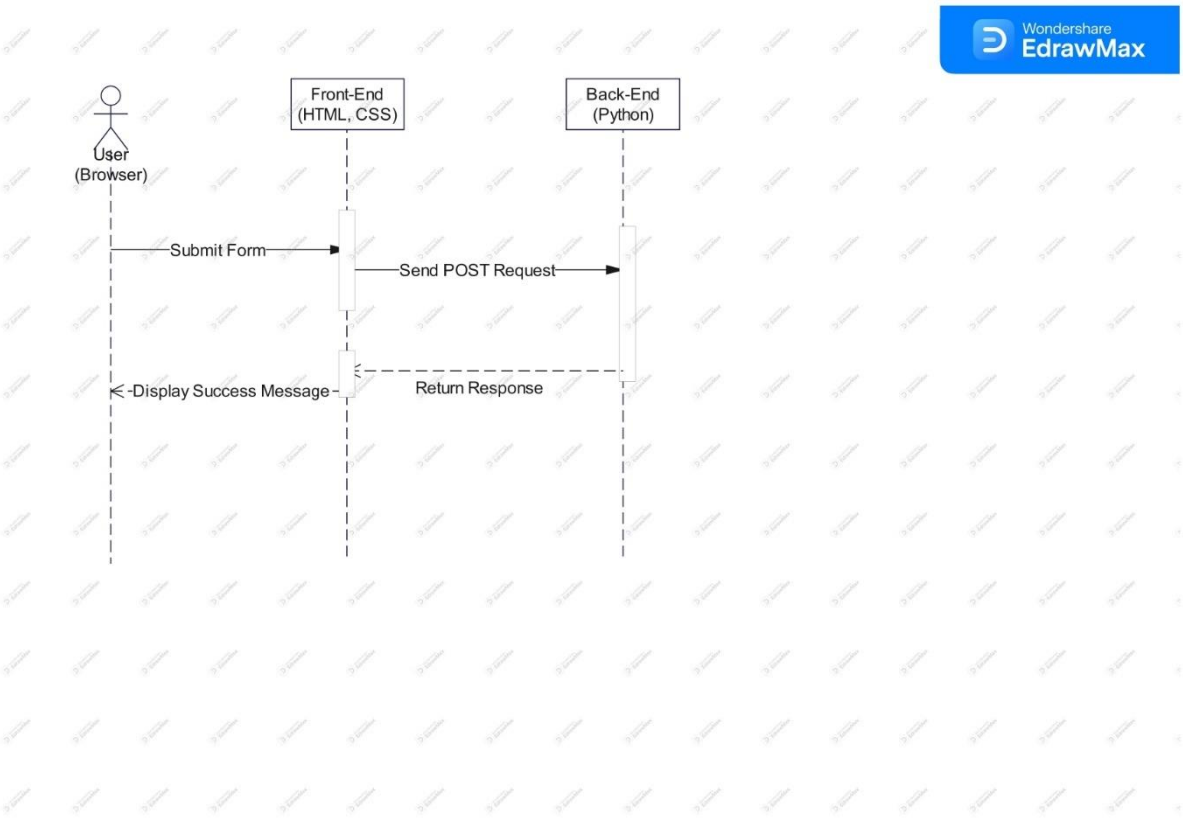

Figure 3: UML Sequence Diagram

The three above diagrams are more related with the System of the project that how the requests are handled by the system. These diagrams are just showing the overview of the system that define the technical background of the system. Now, we move forward to see the front-end technologies.

2.2.2 Front-End Technologies

For the project, Task Master, the front-end of the web application is developed by using the HTML and CSS. HTML (HyperText Markup Language) provides the structural framework of the web pages. With elements like headers, lists, and forms, HTML organizes the content into a logical hierarchy (Gauchat, 2028). This structural framework complements by CSS (Cascading Style Sheets), it adds aesthetic styles and layout controls to enhance the user experience and visual appeal. This pairing provides the experience to user to easily navigate the web application rather to roam over web page to find the correct tab to click. The simplicity of the web application engages the user to save the time and click on the needful tabs that in actual user is looking for. These two core technologies ensure fast loading times across a wide range of devices (Schengili-Roberts, 2004). This approach in real suit the straight nature of the Task Management System (Task Master).

If we see the rationale of the front-end technologies, then we can say that the reason of choosing HTML and CSS for the front-end because of the universal compatibility and ease of

use. These technologies can provide the solid foundation for a responsive design. The designs can adapt the different screen sizes and it is crucial for the users who want to access the application in various devices. The simplicity of HTML and CSS allows easy development and maintenance so for further update for the web application can be possible easily without the need of the expertise coding. There is another benefit of using these technologies is that these technologies support all major browsers that ensure that the web application then can reach the broad audience without any compatibility issues. The main part of the project is that we may be able to provide the user-friendly interface to the user so user may will not face any discrepancy while using the web application on any platform of the network.

2.2.3 Back-End Technologies

For the back-end development of the web application (Task Master), Python as the coding language is being used. The system employs the Flask as its server-side framework. It provides flexibility and lightweight structure. It is ideal for the medium-sized applications. It offers manageable way to set up web servers. This feature of Flask simplifies the debugging. Flask framework has the versatility because it can support extensions that can add functionalities (database integration, migration tools, authentication, and authorization). Flask framework makes the development process of web application is so easy and appealing and all the tools function correctly (Maia, 2015).

SQLite has been used to develop the project because of its reliability and efficient storage solution. As the project required lightweight database and for this the SQLite is the best option to use and develop the web application. The best feature of SQLite is that it runs directly within the application, and it enhance the performance by eliminating the need to communicate over a network. SQLite provides the storage that data (username, password, task details, required data) can be stored in the database management easily. SQLite does not require separate database servers and the use of SQLite facilities easy deployment and scaling (Kreibich, 2010). The files then easily shared among development environments.

2.2.4 Security and Data Handling

The security is the crucial part for the web application because when the user enters the website, he/she creates an account and add his/her personal details and to securing those details are an important part. To save the user data and maintain confidentiality, in first place secure the HTTP(HTTPS) is employed throughout the application. It ensures that all the data transmitted between the front-end and the server is encrypted. It then gets the protection against interception by unauthorized users. Data encryption is an important part because user data would be encrypted and stored as in the SQLite database. It means that the strong cryptographic algorithm is employed before saving the data in the database.

Data Handling is an important part because if data is not handled correctly then it may lead developer against the law. According to the GDPR (General Data Protection Regulation), user data cannot be collected without the consent of the user, and it is the strict guidelines. Moreover, the web application should have the feature in which user can deactivate and remove the information whenever he wants. The system that we developed under all the guidelines so user may not be in any trouble related to his/her data.

2.3 Method

This section of method is divided into three main subsections (Project Planning and Requirements Analysis, Design and Development, Testing and Deployment) in which we can see the approaches and methodologies in detail that have been employed to address the problem. As in the development phase, we only provide the main points about the method of the project development. For example, as we used the sequential methodology, the core functionalities (task creation, editing, and deletion), implementation (HTML, CSS, and Python). But, here we go in detailed description of the method that have been used to develop the project.

2.3.1 Project Planning and Requirements Analysis

The project's initial plan is to create the user-friendly tool that make the process simple to manage the personal and professional tasks. The initial plan of the project should be related to the need of the project so it will stay align till reach the target of the project. firstly, the need is to find the existing problems with the same tools those are developed already related to the project. For example, other task management tools or platforms have some missing features specially the task related to the personal tasks. The developed tools are more focused with organizational tasks rather to related with academic or personal tasks. We address this issue in the project that we developed and make it as simple as possible to interact the user and save his/her time for that the main reason of developing this project.

To achieve this goal to develop the user-friendly platform that easily can address the problem, the requirements need to be gathered. To gather the requirements, we search over internet the available tools first those are related with task management systems. For example, Asana, Trillo, Microsoft To-Do, etc. Moreover, from personal experience and having words with the people who are doing jobs and study both together, we gather the information and the requirements to develop the task management platform. The requirement provides the detail that to manage the time is a near actual need in today's environment.

In the requirement analysis, we analysis the gathered requirements. We try to sort and prioritize the main features those are required for the development of the project. The features may like to add the tasks, update the tasks, and remove the tasks. These are the essential

features for the platform. Moreover, in updated version of the web application may include the reminders of the task, email notifications. etc. but that may stay align with the project requirement and user expectations. Now, we see a bit in detail about the design and the development in the below sub-section.

2.3.2 Design and Development

Another methodology that we employed to develop the project is design and development of the web application. The design is the crucial part of the web pages because the interface should be the appealing for the user. Whenever a new user enters the website then the webpage should be catching so user easily like to navigate the website. CSS helps for the interface that align the tabs in the appropriate manner on the page so for a user no need to explore and waste time to find it.

For the development process of the project, the Agile methodology is being adopted. Although the Agile methodology is suitable for the big teams in which scrum master, developer, product manager, etc would follow the rules and do the given tasks accordingly. Anyhow for the single developer of the project we can adopt this methodology. This approach is flexible and reliable, as the development can be divided into the sprints and later for the further advancement of the project can be do easily (Sutherland, 2019). As discussed earlier in this report, for the development of the project, HTML is being used and CSS for the front-end for the website and Python is being used for the back-end server. In Python the Flask framework is used for its lightweight nature and ease of use that create the scalable application. Additionally, SQLite is being used for the data management because of its seamless integration with Flask.

Later, when the design and development process completed by following the Agile approach, easily can upgrade the web application design according to the user's feedback. It is the best part of the Agile methodology that provides the flexibility in system development. This can be fall under the category of iterative development of the project. Moreover, this iterative process is enhanced the functionality and usability of the system. Additionally, it ensures that the final product may stay align with the user needs.

2.3.3 Testing and Deployment

Testing and the deployment are the important part in web application development. The starting part is with unit testing to ensure that each component of the web page function correctly. After unit testing there is an integration testing that verify the different parts of the system worked together. And the final test is the user acceptance testing (UAT) that confirm if all the user needs and expectations are met. Moreover, before the final deployment must check if is there any issue left while doing the testing of the web application.

The deployment process is involved with the number of steps that ensure the application may run perfectly at the time of live use. In this final phase of the web application, we can test the application in the last time and check whether it is ready for the deployment over internet or not. If it is correctly ready, then we can deploy the web application after all testing over internet so the users all over the world can access it.

After the post-deployment, we can monitor the system by using the tools like Google Analytics for web performance. Moreover, the server health can be tracked via system logs and performance monitoring software. We can put the feedback mechanism so easily we can get the information or data related add new features in the application.

2.4 Implementation

In this implementation section, we see that how the solution can be realized about the development of the project. it includes the technical specifications and development process. Although, it starts with the overview of the implementation strategy, front-end implementation, back-end implementation, testing and quality assurance, and deployment process.

2.4.1 Overview of Implementation Strategy

The strategy of implementation approach is a quite simple and straightforward. We use the HTML and CSS for the front-end and Python and Flask for the back-end. In which front-end is responsible for the design of the web application so it can be run over mobile browser and desktop browser easily. Moreover, the use of Python for its simple coding language and Flask that ensure a smooth integration of server-side logic.

The scope of the project is to create a user-friendly task management system that can run various devices. In one interface, users can easily add the personal and professional tasks. The data security should be there, so the data of the users are not harmed or steal. The design of the web application is simple and easy to navigate by the users.

2.4.2 Front-End Implementation

Now here we can see the technological stack of the front-end implementation in which we choose to used HTML and CSS for the flexibility. Additionally, it creates the interface that run across various devices and web application adjust in the screen according to the size. These technologies provide the advanced styling of the web pages and help to enhance the user experience, in addition it keeps the website lightweight and fast loading.

As this project deals with the Task Management System, and the website that we developed is the initial phase of the project. It is slightly simple and better to understand the project in first place. For the reason being, we make it simple so in first hand will see if this project is aligned

with the project need or requirement. Later phase may try to develop this front-end with the help of bootstrap.

The technology is helping to process the design of the web application so it can be accessible for all the users in various devices. The main goal first to implement the front-end for the better understanding of the project. Let's see the screenshots of the web application of Task Management System (Task Master) to see the front-end of the web application:

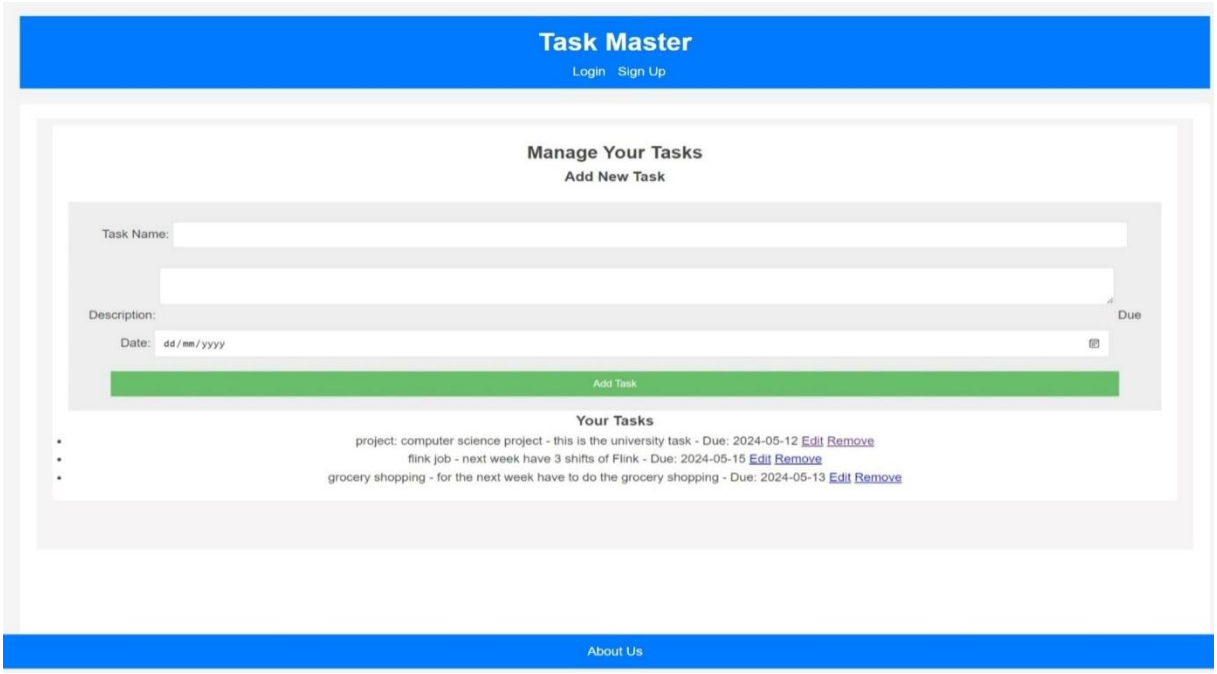

Figure 4: Screenshot of home page of the web application (Task Management System)

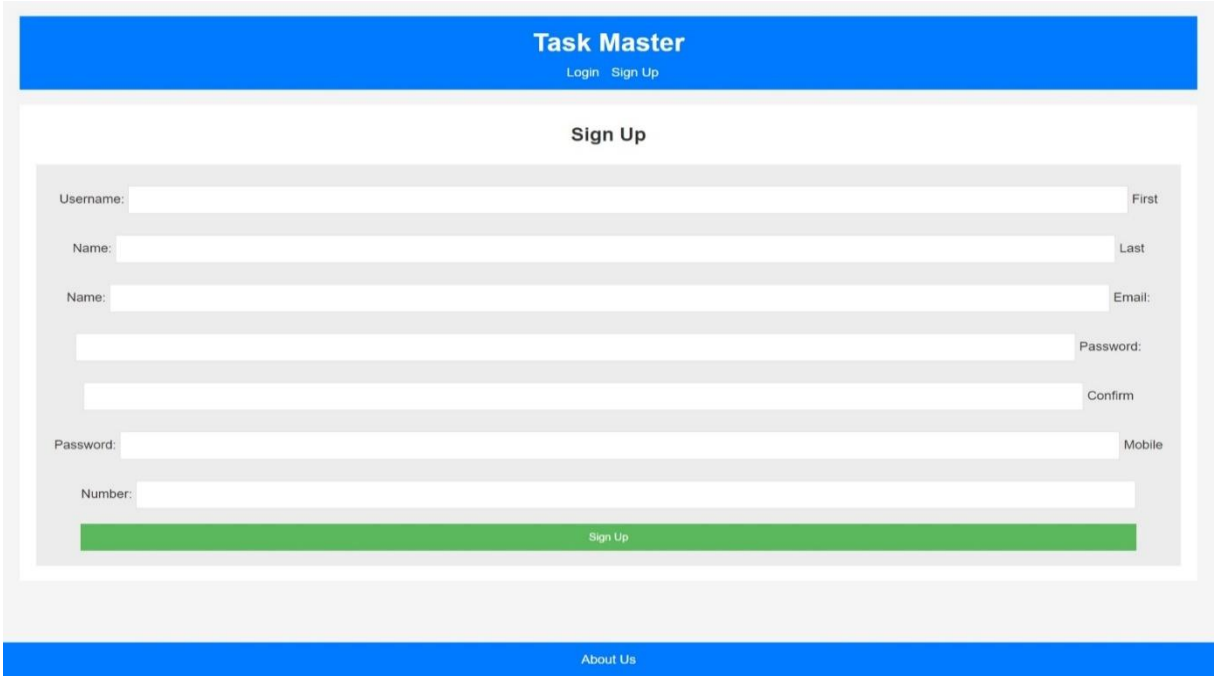

Figure 5: Screenshot of Sign Up of the web application (Task Management System)

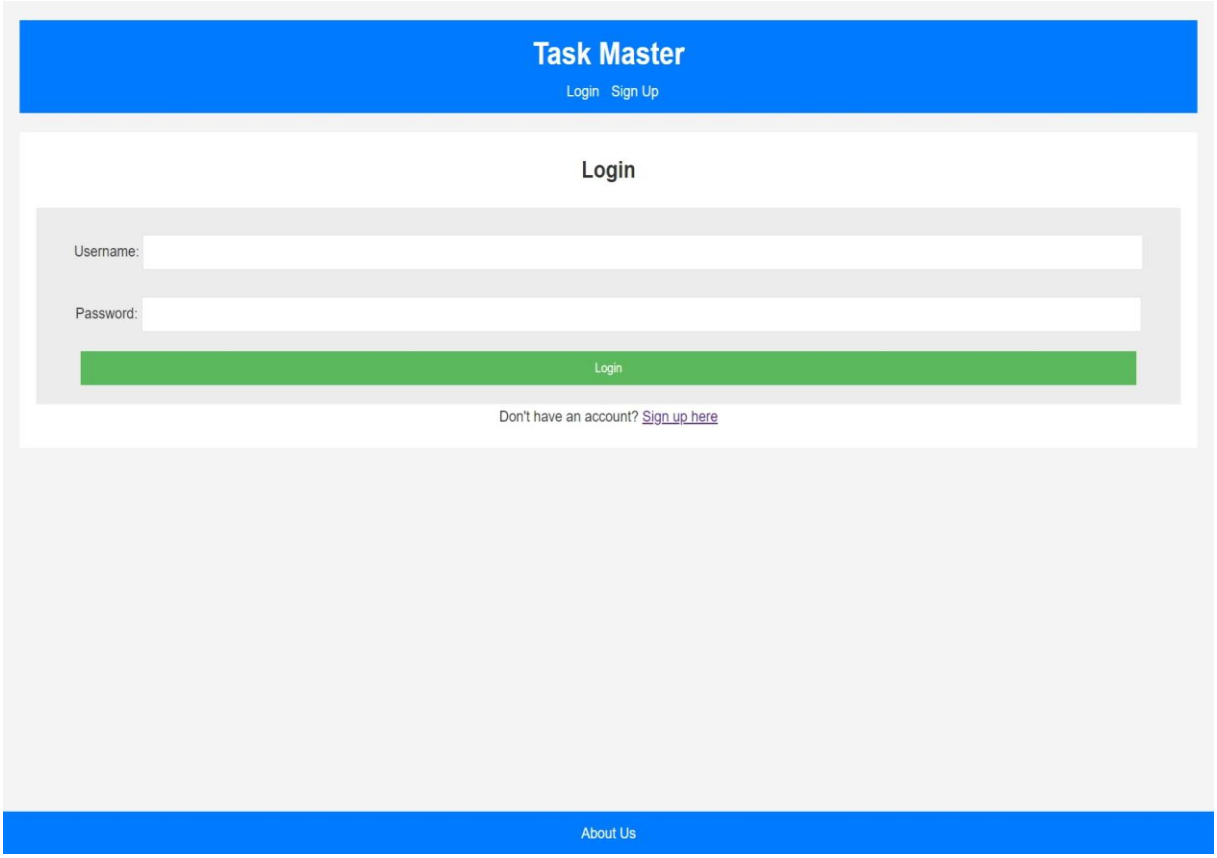

Figure 6: Screenshot of Login of the web application (Task Management System)

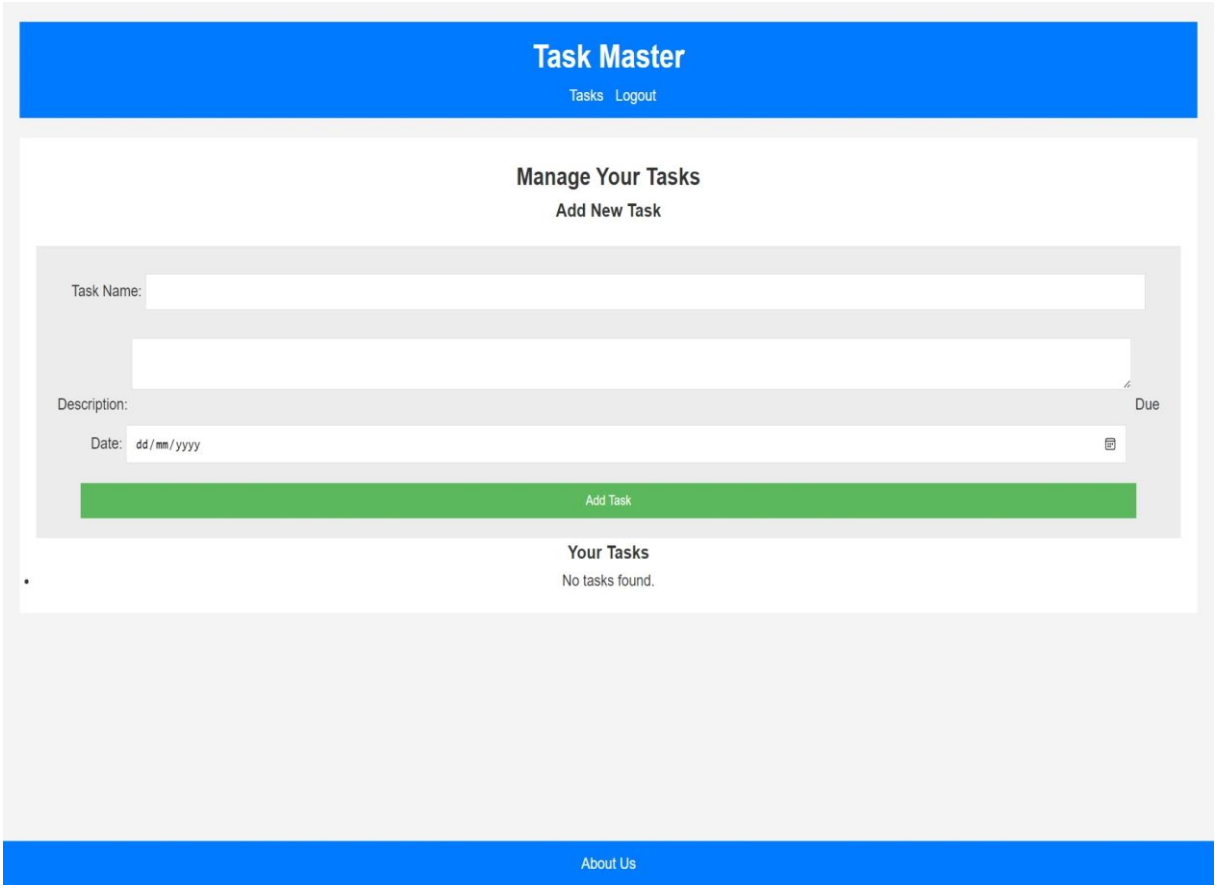

Figure 7: Screenshot of Add Task of the web application (Task Management System)

2.4.3 Back-End Implementation

For the back-end development of the Task Management System (Task Master), we implement server-side technology Python and Flask. Both technologies are being used for its simplicity and efficiency. It has capabilities to handle requests, responses, and session management. Flask framework is the lightweight framework, but it is a powerful framework that fulfil the requirements that we need to develop the Task Management System.

For the data integration, we implement the SQLite database. It has the serverless nature and it is a lightweight. It is being used in a careful manner that is designed to optimize data flow and storage. Moreover, we implement the comprehensive security measures to protect the sensitive user data. It is because of the HTTPS for secure data transmission and encryption techniques.

We can see the code snippet for Flask Route and SQLite Database interaction below:

```
@app.route('/submit-new-task', methods=['POST'])
def submit_new_task():
    if 'username' not in session:
        flash("You must be logged in to add tasks.", "warning")
        return redirect(url_for('login'))

    task_name = request.form['task_name']
    task_description = request.form['task_description']
    task_date_str = request.form['task_date']

    try:
        task_date = datetime.strptime(task_date_str, '%Y-%m-%d').date()
    except ValueError:
        flash("Invalid date format. Please use YYYY-MM-DD format.", "error")
        return redirect(url_for('tasks'))

    user = User.query.filter_by(username=session['username']).first()
    if not user:
        flash("User not found.", "error")
        return redirect(url_for('login'))

    new_task = Task(
        name=task_name,
        description=task_description,
        due_date=task_date,
        user_id=user.id
    )
```

```
    db.session.add(new_task)
    try:
        db.session.commit()
        flash('Task added successfully!', 'success')
    except Exception as e:
        db.session.rollback()
        flash("Error adding task: " + str(e), 'error')

    return redirect(url_for('tasks'))
```

Figure 8: Code Snippet Example for Flask Route

```
class Task(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(150), nullable=False)
    description = db.Column(db.String(500), nullable=False)
    due_date = db.Column(db.Date, nullable=False)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'))
```

Figure 9: Code Snippet Example for SQLite Database Interaction

2.4.4 Testing and Quality Assurance

The testing is an important part of the development of the project to check whether all the individual components would work efficiently. We will see the testing part in detail of the project in the next section of testing. But in this implementation section the testing part involved related to a brief overview about the unit and integration testing. In integration testing, to check the flow of the entire web application from front-end to back-end.

The tool and the framework for the project development is Python and Flask. In which Python provides the *unittest* framework that deals with the unit testing. It ensures that core components would meet their design specifications. Moreover, Flask's testing tools are for simulate the requests to the application and access the responses. It also provides the assurance of the quality of the web application, that the web application is fall under the condition that we think should have before starting the project.

2.4.5 Deployment Process

In this part we can see that how we can deploy the web application. It is simple as need to prepare the environment first. Heroku platform is the better choice for this as its support with Python and Flask applications. It simplifies the deployment process. We also can use the Docker to containerize the web application.

There are some challenges also can be occurred during deployment in which configuration of the Docker environment. It is a quite complex to manage the persistent data with SQLite across

container restarts. But if the SQLite files can be stored outside the container, then the issue can be resolved. It then maintains the data integrity and availability.

2.4.6 Documentation

In this part of implementation, the documentation is required so the user easily can read that how to navigate the web application. It includes the user manuals that outlines the operational procedures and system documentation that helps to navigate and troubleshoot the system. In the appendix section, we will see the documentation of the code, the code that we created to run the web application. In actual, the documentation related with the readme file of the project so user can read about the web application that how to operate in actual.

Now, we will move forward from this implementation section to the testing section, in which we get the detail overview about the testing of the web application. Additionally, it is also the important part of the web application.

2.5 Testing

In this section, we discussed about testing, and this is the important part of the Task Management System that have been developed. This section of testing is divided into four sub-sections, overview of Testing, details of Testing procedures, performance and security testing, and user acceptance testing:

2.5.1 Overview of Testing

The main thing is when we develop any software then the testing part is very crucial. The main work of testing to check the features of the developed software is functioning correctly or not. It ensures that the user experience while using the web application should be the same that structured before starting to develop the software or web application.

The testing levels that implemented for the system development are unit testing, integration testing, and system testing. As we already discussed above in method and implementation section that unit testing is used to validate individual components for correctness. Then, the integration testing make it sure that the components of the web applications are working together without any trouble. The system testing is used when the web application is completed and need to check all the application is working correctly and efficiently that imagine before development of the web application.

2.5.2 Details of Testing Procedures

In this sub section, we will see all three testing in details. Firstly, in Unit Testing, the focus is on the individual component of the web application. Here we can see the code snippet of the unittest framework:

```
import unittest
from main import app, db, User, Task, datetime

class TestCase(unittest.TestCase):
    def setUp(self):
        self.app = app.test_client()
        self.app.testing = True
        app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///test.db'
        db.create_all()

    def tearDown(self):
        db.session.remove()
        db.drop_all()

    def test_task_creation(self):
        """Test task creation functionality"""
        user = User(username='testuser', first_name='Test', last_name='User',
email='test@example.com')
        user.set_password('securepassword')
        db.session.add(user)
        db.session.commit()
        self.app.post('/login', data=dict(username='testuser', password='securepassword'))
        response = self.app.post('/submit-new-task', data=dict(task_name='New Task',
task_description='Test Description', task_date='2022-12-31'))
        self.assertIn(b'Task added successfully!', response.data)

    def test_task_deletion(self):
        """Test task deletion functionality"""
        user = User(username='testuser', first_name='Test', last_name='User',
email='test@example.com')
        user.set_password('securepassword')
        db.session.add(user)
        db.session.commit()
        task = Task(name='Delete Task', description='Task to be deleted', due_date=datetime.utcnow(),
user_id=user.id)
        db.session.add(task)
        db.session.commit()
        self.app.post('/login', data=dict(username='testuser', password='securepassword'))
        response = self.app.get(f'/delete_task/{task.id}')
        self.assertIn(b'Task removed successfully!', response.data)
```

```
if __name__ == '__main__':
    unittest.main()
```

Figure 10: Code Snippet Example for Unit Testing

It checks the validity of the user inputs, the correctness of task creation and the functionalities of the deletion. Moreover, the proper handling of data and time calculations within the system.

The second testing is fall under the category of Integration testing. In this integration testing, units are combined into groups and then testing them in complete subsystem. It can be done by using the tool like *pytest* to validate the interactions and data flow between the components. The main components are like authentication flow and task management operations. The focus is on to check the user session management and database interactions worked together and able to detect any integration issues.

The third testing is the system testing in which the role of this testing to evaluate the entire application. The combined components are tested all together and validate the overall system behaviour against the specific requirements. The main tests that are involved are like the loading time of the webpage in different devices like mobile browser or desktops to ensure the consistent functionalities and user experience.

2.5.3 Performance and Security Testing

Firstly, for the performance testing, we simply run the web application and monitor the application's performance using available tools like the Flask development server's logs. It provides the insights into the responses times and error rates under user loads. Moreover, through manual testing, we also observed that how the web application is handling the different tasks like added, deleted, updated ensure how it work or responses under different user scenarios.

Secondly, the security testing is also the important part so the data of the users would be saved and not be tampered under certain situations. Flask's built-in protections against threat like SQL Injection were in mind while the time of developing the Task Management System. We ensure that the data would be transmitted over HTTPS, so it enhanced the security of the user data. Additionally, we also conduct the manual security checks to prevent the unauthorized access and the data breaches.

2.5.4 User Acceptance Testing

As, we already discussed about the User Acceptance Testing in the Method section, so here we discuss it in a bit of detail that provide how this testing work. This testing is conducting by inviting number of users all together. These users then run the web application and access it accordingly and then we observe is there an error while accessing any user facing or not. This step provides that how the application would be run in the practical use.

3. Conclusion

This Task Management System (Task Master) is the suitable web application that address the real-world problem to adjust or manage the time according to the individual needs. It provides the user-friendly interface for adding, updating, and removing the tasks. After test the application and compare with other available task-based application, the develop application (Task Master) can satisfied the user need that assumed before staring the development of the application. The web application provides the significant platform to the individuals to organize the time efficiently and more effectively. Additionally, the performance testing confirms that the application operates correctly and fast under the user loads. It ensures the reliability and responsiveness of the application.

The Task Management System (Task Master) is integrated with the secure HTTP and data encryption that prove that it is effective in protecting user data. It follows the standard data security protocols. There are some challenges also faced while developing the application for example, ensure the cross-browser compatibility and it is addressed successfully during development of the application. However, there is a limitation include the lack of integration with external calendar applications and advanced task prioritization algorithms. Additionally, as a user perspective there could be a feature like task support, or the notification related tasks are under consideration. Moreover, for the future enhancements, the focus is on integrating the machine learning algorithms to predict task urgency and suggest optimal task schedules. The use of the Artificial Intelligence for predictive scheduling can enhance the productivity for the individual's time by automatically prioritizing the task based on user's behaviour and added tasks accordingly.

In the end, there are some future developments of the web application is under consideration. For example, develop a mobile application so users can easily access it on mobile without the need of the browser. the system should have the integration capabilities with productive tools like Google Calendar and Microsoft Outlook that can provide the seamless task management experience. As, the main aim of the project is completed to create the simple and usable task management system is completed. The development of the "Task Master" is in real a great learning experience. It shows that creating a digital solution is the effective tools that in real deal with the real-world problems.

Bibliography and List of References

Introduction to HTML: Learn how to create the structure of your website with HTML and HTML5. (2018). (n.p.): J.D Gauchat.

Jonas, A. (2020). The Impact of Task Management Systems on Organizational Productivity. Journal of Business Technology, 24(2), 158-172.

Kreibich, J. (2010). Using SQLite. Germany: O'Reilly Media.

Lee, C. (2018). Bridging the Gap: Integrating Personal and Professional Task Management. Journal of Personal Productivity, 14(1), 34-35.

Maia, I. (2015). Building Web Applications with Flask. United Kingdom: Packt Publishing.

Schengili-Roberts, K. (2004). Core CSS: Cascading Style Sheets. Vereinigtes Königreich: Prentice Hall PTR.

Smith, B. (2019). The Disconnect in Task Management Tools. Technology in Life Journal, 19(4), 200-210.

Smith, J. (2021). Impact of Cloud Technology on Task Management. Journal of Business Technology, 15(2), 34-45.

Sutherland, J. (2019). The Scrum Fieldbook: A Master Class on Accelerating Performance, Getting Results, and Defining the Future. United States: Crown.

https://flask.palletsprojects.com/en/3.0.x/quickstart/#deploying-to-a-web-server

Appendix

1. main.py

```
from flask import Flask, render_template, request, session, redirect, url_for, flash
from flask_sqlalchemy import SQLAlchemy
from werkzeug.security import generate_password_hash, check_password_hash
from datetime import datetime

app = Flask(__name__)
app.config['SECRET_KEY'] = 'your_secret_key'
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///users.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False

db = SQLAlchemy(app)

class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(80), unique=True, nullable=False)
    first_name = db.Column(db.String(100), nullable=False)
    last_name = db.Column(db.String(100), nullable=False)
    email = db.Column(db.String(120), unique=True, nullable=False)
    password_hash = db.Column(db.String(128))
    mobile_number = db.Column(db.String(20), nullable=False)

    def set_password(self, password):
        self.password_hash = generate_password_hash(password)

    def check_password(self, password):
        return check_password_hash(self.password_hash, password)

class Task(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(150), nullable=False)
    description = db.Column(db.String(500), nullable=False)
    due_date = db.Column(db.Date, nullable=False)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'))

@app.route('/')
def home():
    return render_template('index.html')
```

```
@app.route('/signup', methods=['GET'])
def signup():
    return render_template('signup.html')

@app.route('/submit-signup-form', methods=['POST'])
def submit_signup_form():
    username = request.form['username']
    first_name = request.form['first_name']
    last_name = request.form['last_name']
    email = request.form['email']
    password = request.form['password']
    confirm_password = request.form['confirm_password']
    mobile_number = request.form['mobile_number']

    if password != confirm_password:
        flash('Passwords do not match. Please try again.', 'error')
        return redirect(url_for('signup'))

    existing_user = User.query.filter_by(email=email).first()
    if existing_user:
        flash('Email already in use. Please choose another one.', 'error')
        return redirect(url_for('signup'))

    new_user = User(
        username=username,
        first_name=first_name,
        last_name=last_name,
        email=email,
        mobile_number=mobile_number
    )
    new_user.set_password(password)
    db.session.add(new_user)
    db.session.commit()

    session['username'] = username
    flash('Successfully signed up and logged in!', 'success')
    return redirect(url_for('home'))

@app.route('/login', methods=['GET'])
def login():
    return render_template('login.html')
```

```python
@app.route('/submit-login-form', methods=['POST'])
def submit_login_form():
    username = request.form['username']
    password = request.form['password']

    user = User.query.filter_by(username=username).first()
    if user and user.check_password(password):
        session['username'] = username
        flash('You have successfully logged in!', 'success')
        return redirect(url_for('home'))
    else:
        flash('Invalid username or password', 'error')
        return redirect(url_for('login'))

@app.route('/tasks')
def tasks():
    if 'username' not in session:
        flash("You must be logged in to view this page.", "warning")
        return redirect(url_for('login'))

    user = User.query.filter_by(username=session['username']).first()
    if user is None:
        flash("User not found.", "error")
        return redirect(url_for('login'))

    user_tasks = Task.query.filter_by(user_id=user.id).all()
    return render_template('task.html', tasks=user_tasks)

@app.route('/submit-new-task', methods=['POST'])
def submit_new_task():
    if 'username' not in session:
        flash("You must be logged in to add tasks.", "warning")
        return redirect(url_for('login'))

    task_name = request.form['task_name']
    task_description = request.form['task_description']
    task_date_str = request.form['task_date']

    try:
        task_date = datetime.strptime(task_date_str, '%Y-%m-%d').date()
    except ValueError:
```

```
        flash("Invalid date format. Please use YYYY-MM-DD format.", "error")
        return redirect(url_for('tasks'))

    user = User.query.filter_by(username=session['username']).first()
    if not user:
        flash("User not found.", "error")
        return redirect(url_for('login'))

    new_task = Task(
        name=task_name,
        description=task_description,
        due_date=task_date,
        user_id=user.id
    )
    db.session.add(new_task)
    try:
        db.session.commit()
        flash('Task added successfully!', 'success')
    except Exception as e:
        db.session.rollback()
        flash("Error adding task: " + str(e), 'error')

    return redirect(url_for('tasks'))

@app.route('/delete_task/<int:task_id>')
def delete_task(task_id):
    if 'username' not in session:
        flash("You must be logged in to perform this action.", "warning")
        return redirect(url_for('login'))

    task = Task.query.get(task_id)
    if task:
        db.session.delete(task)
        db.session.commit()
        flash('Task removed successfully!', 'success')
    else:
        flash('Task not found.', 'error')
    return redirect(url_for('tasks'))

@app.route('/edit_task/<int:task_id>', methods=['GET'])
def edit_task(task_id):
```

```
    if 'username' not in session:
        flash("You must be logged in to perform this action.", "warning")
        return redirect(url_for('login'))

    task = Task.query.get(task_id)
    if task:
        return render_template('edit_task.html', task=task)
    else:
        flash('Task not found.', 'error')
        return redirect(url_for('tasks'))

@app.route('/update_task/<int:task_id>', methods=['POST'])
def update_task(task_id):
    if 'username' not in session:
        flash("You must be logged in to perform this action.", "warning")
        return redirect(url_for('login'))

    task = Task.query.get(task_id)
    if task:
        task.name = request.form['name']
        task.description = request.form['description']
        task.due_date = datetime.strptime(request.form['due_date'], '%Y-%m-%d').date()
        db.session.commit()
        flash('Task updated successfully!', 'success')
        return redirect(url_for('tasks'))
    else:
        flash('Task not found.', 'error')
        return redirect(url_for('tasks'))

@app.route('/about')
def about():
    return render_template('about.html')

@app.route('/logout')
def logout():
    session.pop('username', None)
    flash('You have been logged out.', 'success')
    return redirect(url_for('home'))

if __name__ == '__main__':
    with app.app_context():
```

```
        db.create_all()
    app.run(debug=True)
```

2. index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Task Master</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>
    <header>
        <h1><a href="{{ url_for('home') }}">Task Master</a></h1>
        <nav>
            <ul>
                {% if 'username' in session %}
                <li><a href="{{ url_for('tasks') }}">Tasks</a></li>
                <li><a href="{{ url_for('logout') }}">Logout</a></li>
                {% else %}
                <li><a href="{{ url_for('login') }}">Login</a></li>
                <li><a href="{{ url_for('signup') }}">Sign Up</a></li>
                {% endif %}
            </ul>
        </nav>
    </header>
    <main>
        <section>
            <div class="slideshow">
                <img src="{{ url_for('static', filename='demo-task-image.jpeg') }}" alt="stay ahead always"
style="width: 100%; height: auto;">
                <img src="{{ url_for('static', filename='second-image.jpeg') }}" alt="stay ahead always"
style="width: 100%; height: auto;">
                <img src="{{ url_for('static', filename='third-image.jpeg') }}" alt="stay ahead always"
style="width: 100%; height: auto;">
            </div>
        </section>
    </main>
    <footer>
        <a href="{{ url_for('about') }}">About Us</a>
```

```
    </footer>
</body>
</html>
```

3. requirements.txt

```
Flask~=3.0.3
Flask-SQLAlchemy~=3.0.5
Werkzeug~=3.0.2
```

4. docker-compose.yml

```
version: '3.8'

services:
  web:
    build: .
    ports:
      - "5000:5000"
    volumes:
      - .:/app
    environment:
      - FLASK_APP=main.py
    command: flask run --host=0.0.0.0
```

5. Dockerfile

```
# Use an official Python runtime as a parent image
FROM python:3.9-slim

# Set the working directory in the container
WORKDIR /app

# Copy the current directory contents into the container at /app
COPY . /app

# Install any needed packages specified in requirements.txt
RUN pip install --no-cache-dir -r requirements.txt

# Make port 5000 available to the world outside this container
EXPOSE 5000

# Define environment variable
ENV FLASK_APP=main.py
```

```
# Run app.py when the container launches
CMD ["flask", "run", "--host=0.0.0.0"]
```

6. README.md

```
# Task Management System
This project is a web application developed as part of the Project Computer Science Project course. It showcases the design and implementation skills in creating a web-based time management solution using Flask, a popular Python web framework. The application aims to provide a simple yet functional interface for users to manage their tasks.

## Features
- User registration and authentication
- Task creation, editing, and deletion
- Task viewing for logged-in users
- Web interface built with Flask
- Integration with a database using Flask-SQLAlchemy for data persistence
- User-friendly interface designed with Jinja2 templates

## Getting Started
These instructions will guide you through setting up and running the project locally using Docker. Ensure you have Docker installed on your system before proceeding.

### Prerequisites
- Docker
- Git (for cloning the repository)

### Installation
1. **Clone the repository**
   Open a terminal and run:
   ```bash
   git clone https://github.com/ichsandeep/task-management-system.git
```

GitHub Repository Link

https://github.com/ichsandeep/PCSproject

https://github.com/ichsandeep/PCSproject/tree/main/Sandeep_Sandeep_102210527_DLMCSPCSP01
```