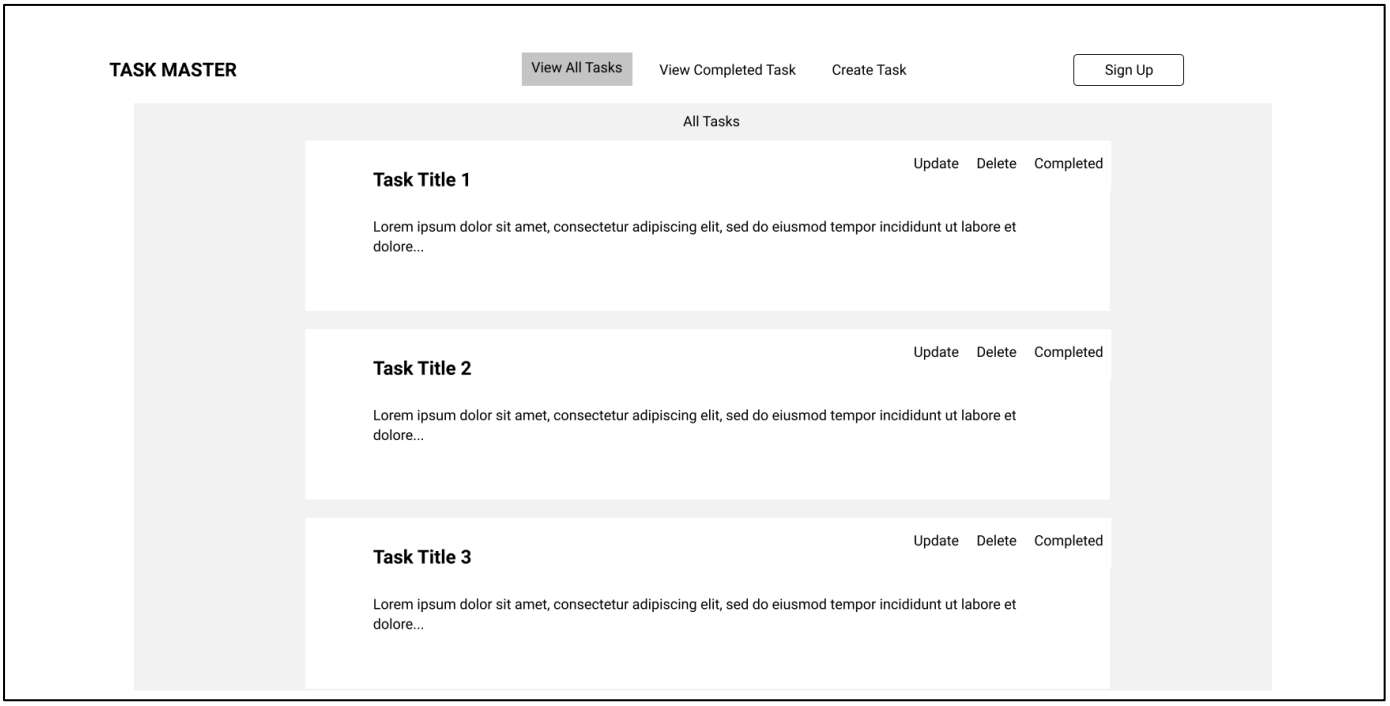
WEB APPLICATION DEVELOPMENT

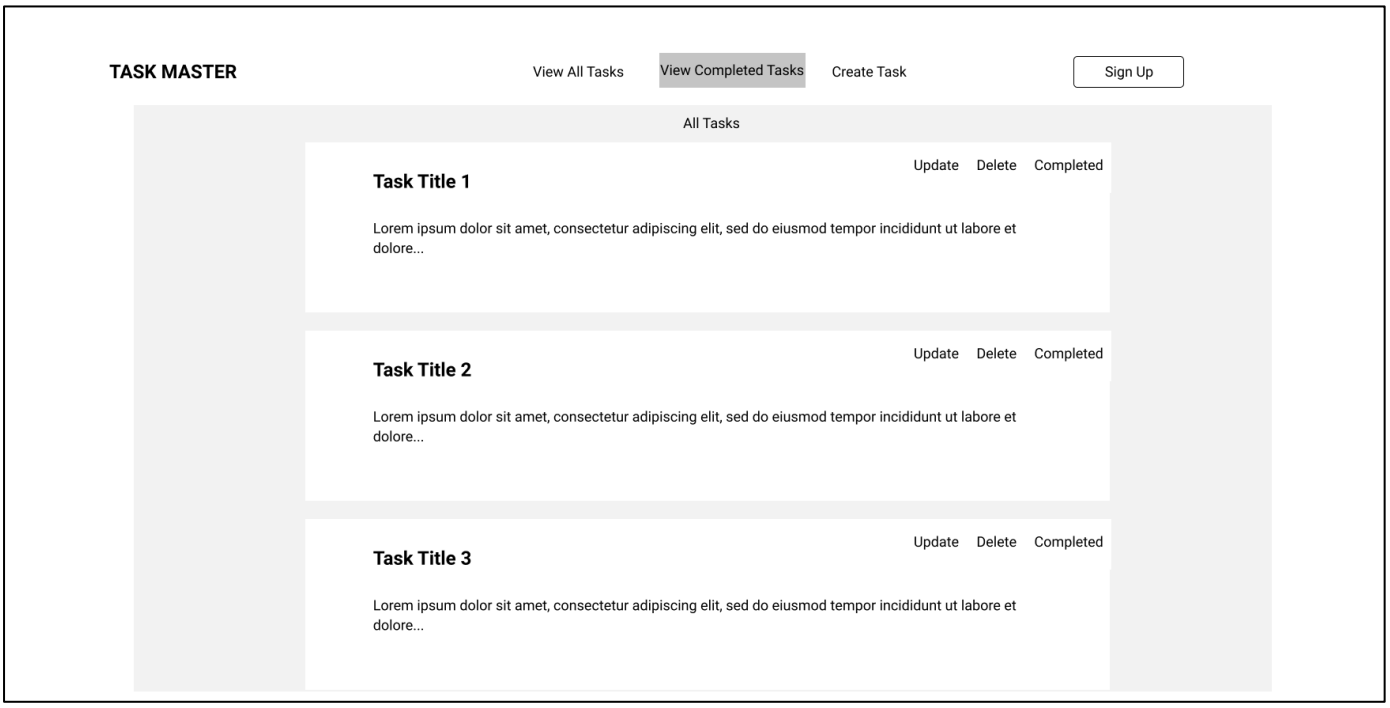
COURSEWORK-1(TODO LIST)

1.WIREFRAME MODELS:

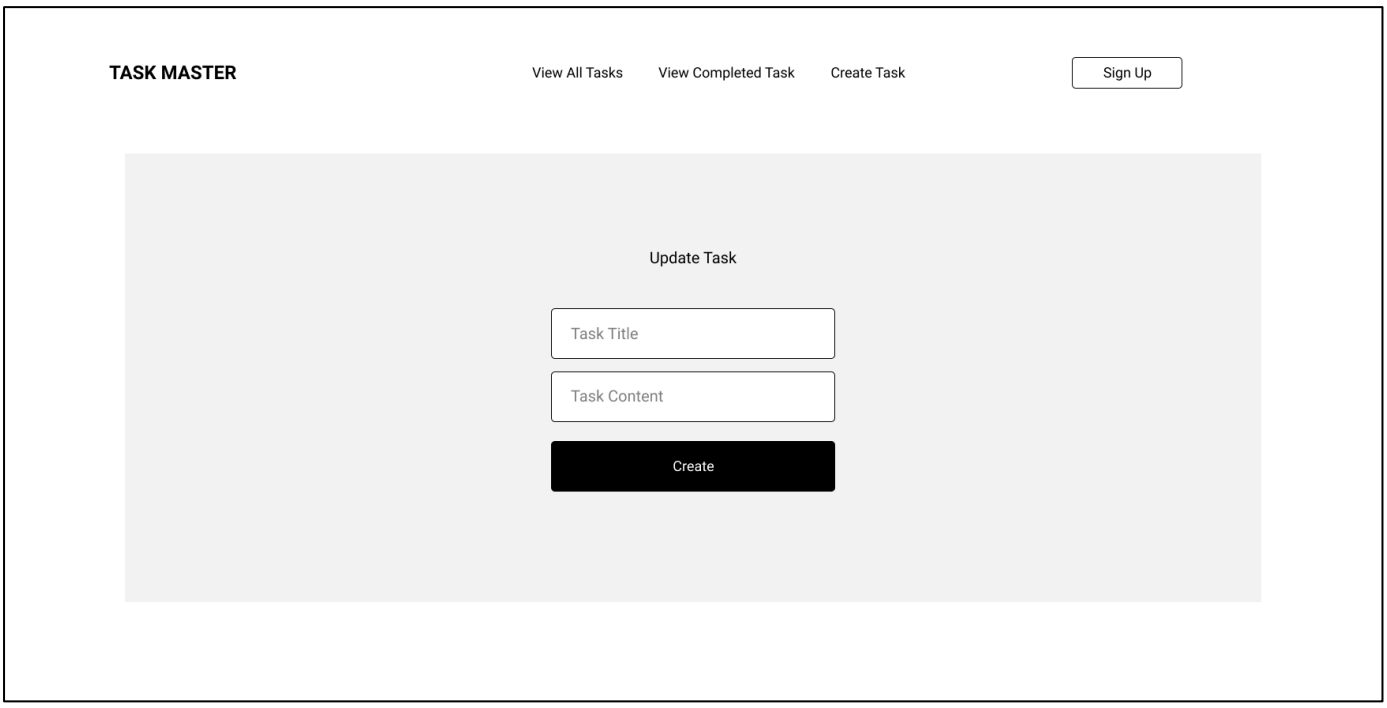
1.1 HOME PAGE(index.html)



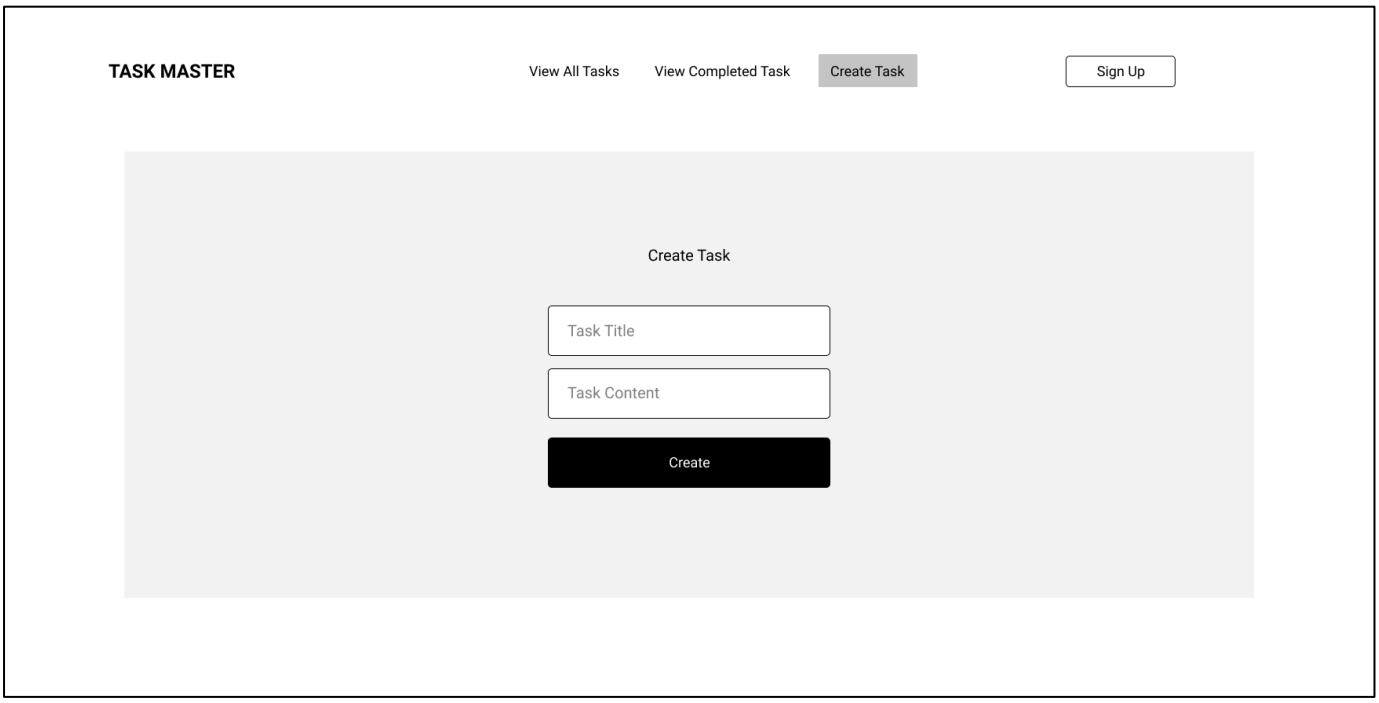
1.2 COMPLETED TASKS PAGE (completed.html)



1.3 UPDATE TASK PAGE (update.html)



1.4 CREATE TASK PAGE (create.html)



2. JUSTIFICATION FOR THE CHOICE OF LAYOUT:

I have chosen a page layout that display the tasks and their respective description in the form of cards which is different from a typical presentation of tasks in a tabular form.

Displaying information in forms of cards makes the web application look more visual appealing and easy to use. I have also chosen not to include too many stylistic formats and colors as a simplistic view makes the application easier to adapt and learn. I have also chosen to include commonly used icons to navigate to or perform certain functionalities. This is because icons occupy lesser space there by preventing the page from looking too bulky and unimportant texts divert the users view/attention from the primary content of the page which are the tasks and their descriptions.

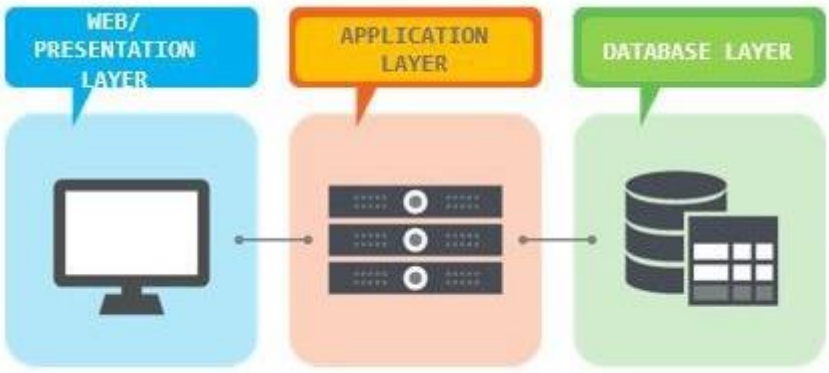
When the user click on the edit icon of a card that he wishes to edit/update , the user gets redirected to a update task page where the details of he task can be changed .In this page I choose to use form to take input of the user as it makes the job of adding and committing a task detail to the database.

I have also chosen to use if blocks and else blocks in the index and completed task page because , This lets me display “There is no new task” or “ No tasks are completed” if there is nothing to display in that page. I chose to adapt a dark theme as it cuts glare and reduces blue light thereby reducing the strain on eyes especially in a low light condition.

3. DESCRIPTION OF WEB APPLICATION’S ARCHITECTURE:

My application adapts a 3-tier architecture system. The three tiers in this architecture are presentation layer, application layer and database layer.

- **Presentational**: This is the front end layer in the 3-tier system and consists of the user interface. This tier, which is built with HTML5, cascading style sheets (CSS) and JavaScript, is deployed to a computing device through a web-based application. The presentation tier communicates with the other tiers through application program interface (API) calls.
- **Application**: The application tier, which may also be referred to as the logic tier, is written in Python and contains the business logic that supports the application’s core functions. The underlying application tier establishes connection with the database thereby adding , deleting and committing data to it based on the access rights give to each function .
- **Database**: The data tier comprises of the database/data storage system and data access layer which is responsible for communicating with the database , in this case I have chosen to use sqlalchemy as my SQL tool kit s it is more compatible an easy to use for a flask application. Data is accessed by the application layer via API calls.



A 3 tier architecture improves horizontal scalability, performance and availability. With three tiers, each part can be developed concurrently coded in different languages. Because the programming for a tier can be changed or relocated without affecting the other tiers, the 3-tier model makes it easier to continually evolve an application as new needs and opportunities arise

4.DESCRIPTION OF REQUEST HANDLING AND HTTP FEATURE:

I made use of 2 API calls for this application, which are 'GET' ,'POST' and 'DELETE' .

- **GET**
  The GET method requests a representation of the specified resource. Requests using GET only retrieves data, I have thus used the GET method in the index function, update function and other functions to get data from the database to display.
- **POST**
  The POST method is used to submit an entity to the specified resource, often causing a change in state or side effects on the server. I thus have used the POST method create new tasks and update task as this request allows the function to update data in the database.
  This request Is crucial in displaying the data and rendering the final template.

5. EVALUATION AND IMPORTANCE OF TESTING:

In software development, everything needs to double-check before releasing the product. It is important to have a tester team who will test software and invest effort and knowledge to make sure that product is defect-free.

A **primary purpose of testing** is to detect software failures so that defects may be discovered and corrected. **Testing** cannot establish that a product functions properly under all conditions, but only that it does not function properly under specific conditions. Software testing will point out the errors that occur during the development phases. It makes sure that the application's performances are adequate and that customers are satisfied with it. When the delivered product is of quality, it helps in gaining the confidence of the customers.

Software testing with strict test execution assures lower maintenance cost. There cannot be any failures because it can be very expensive in the later stages of the development.

For staying in business, having a software testing is imperative. A critical defect left undetected can cause losses in business. Every serious company will invest in a tester team and find experts who will be capable of detecting problems and flaws in a software product.

In software development, debugging involves locating and correcting code errors in a computer program. Majorly performed while the software testing process, debugging is anintegral part of the entire software development cycle. The process of debugging starts as soon as as the code of the software is written and continues in successive stages as code is combined with other units of programming to form a software product. Some of the other benefits of debugging are mentioned below:

- **Reports error condition immediately,** which allows earlier detection of an error and makes the process of software development stress-free and unproblematic.
- It provides **maximum useful information** of data structures and allows its **easy interpretation.**

- Assists the developer in **minimizing useless and distracting information.**
- Through debugging the developer can **avoid complex one-use testing code**, which helps the developer save time and energy in software development.

6. SECURITY:

There are no major security threats to this application. This is because of multiple reasons, firstly this application was created for a single user thus this app does not have any authentication features that can be hacked by anyone. This app is not built for multiple users thus avoiding the common security risks faced such as hijacking an authenticated session etc.

Also, this application does not face the issue of Cross-site scripting as I have made sure that all the variables used in this application are initialized. This application does not collect any personal details thus leaving the risk of exposure of sensitive content behind. I have also made sure to implement as many mitigation techniques as possible which were explained in the web app course website.

7. INCLUDING ACCESSIBILIY IN THE WEB APPLICATION:

The Web is fundamentally designed to work for all people, whatever their hardware, software, language, location, or ability. When the Web meets this goal, it is accessible to people with a diverse range of hearing, movement, sight, and cognitive ability.

A web application can be made accessible to by making the website:

- **Perceivable**: Users need to be able to use your application and access all content; you must provide text alternatives for any non-text content and make it easier for visually or audibly impaired users to access your content.
- **Operable:** Make your entire web app navigable by keyboard and establish a clear information architecture.
- **Understandable:** Make sure your content is understandable (content-wise and visually).
- **Robust:** Ensure compatibility with user agents, such a screen reader.

To implement the above features the web application must make use of semantic HTML, all content must be keyboard accessible, provide text alternatives for media content and finally use accessible font & color.

8.COMMENT ON WHAT HAS BEEN LEARNT AND ITS PROCESS:

I was learning on how to work with a few web frameworks like Vue.js and Django during the summer break, which were useful for large applications. But these frameworks are too big and complex to develop small scale projects like my coursework for Web application development, which was to implement a To-do list using Flask.

Having not worked with Flask before it was surprisingly an easy and fun framework to work with. Implementing the Backend part of a web application seemed simple and straightforward. Especially creating a database and declaring variables in them.

9.REFERENCES:

https://www.tutorialspoint.com/flask/index.htmGuan

https://www.geeksforgeeks.org/flask-creating-first-simple-application/

http://www.cs.toronto.edu/~mashiyat/csc309/Lectures/Web%20App%20Architectures.pdf

https://www.dreamhost.com/blog/make-your-website-accessible/