Project Report

Author

- AMULYA VERMA
- 21f1003701
- 21f1003701@student.onlinedegree.iitm.ac.in
- My Name is Amulya Verma having almost 10 years of experience in the IT industry, to be specific the experience is more onto the Digital Communication and Software QA Automation side. This is the first time that I am having a hands on the web technologies(both Frontend and Backend) and I am pretty much loving the fact that what I always wanted to learn (technology missing in my CV), IIT Madras is helping in delivering the same via means of Online classes.

Description

- Design Kanban Dashboard which translates to Ticket Management System. I have taken somewhat freedom to explore the different Dashboards available (viz Jira, Asana etc), and then design the Dashboard according to the requirements.

Technologies Used

- Python Flask -> Web Framework Design
- SQLAlchemy -> Database Querying and modification
- Flask RESTful -> Inbuilt flask framework for building REST APIs
- Matplotlib -> For plotting Graphs to get a better picture of task insights.
- Werkzeug -> Mainly to handle HTTP Exception to handle standard HTTP non-200 response

DB Schema Design

-> Two type of tables in the DB Schema:

Tasks Table

SNo.	Columns	Description	Constraints
1.	ID	Is the unique identifier for the tasks.	Integer, Primary Key, AI
2.	Summary	A short summary/heading for the tasks.	String, Not Null
3.	Description	Description about the task	String
4.	Status	Status of the task i.e. Blocked, Pending, In Progress and Closed,	String, Not Null

Users Table			
SNo.	**Columns**	**Description**	**Constraints**
1.	User_ID	Is the unique identifier for the Users.	Integer, Primary Key, AI
2.	username	Unique username for the users.	String, Not Null, Unique
3.	Password	Description about the task	String, Not Null
4.	f_name	First Name of the User	String, Not Null
5.	l_name	Last Name of the User	String

API Design

- REST APIs are designed mainly to fetch and post the Task related information to the Backend server/Database.
- 4 types of API Designed:

    1. **GET** -> To get the task information from the backend, ID is used to filter the tasks and fetch the information.
    2. **POST** -> To add a new Ticket/task. Additional logic is defined based upon different constraints set on Task table attributes.
    3. **PUT** -> Fetch the task and do an update in the task. ID is used to filter the tasks and fetch the information.
    4. **DELETE** -> To Delete the task entry from the Database.

Architecture and Feature

- The entire framework is designed using MVC Design Pattern wherein the controllers(/application/controllers.py) are used wherein the business logic is defined i.e. redirection pages, error handling and User Login.
- Templates(/templates/..) formulates the View/Presentation layer for the application. There are many templates which handle the main pages and error pages.
- API(/application/api.py) the API file is where the REST API logic and working is defined.
- Models(/application/models.py) forms the backbone of the framework and defines the Database schema.
- Default Features -> Ticket Management i.e. Addition of new ticket, Updation of existing ticket and Deletion of existing tickets. Also the app contains user login feature and a summary Page.
- Additional Feature -> User Registration, Logout feature.

Video