

Project Report

On

Task Manager

ANUDIP FOUNDATION

Project Title

“Task Manager”

Name	Enrollment No
Babar Atharva Mohan	AF0481890
Babar Shubham Pralhad	AF0481892

**Under Guidance
Of**

Rajashri Thete

Index

Sr.no	Topic	Page no
1	Title of Research	1
2	Certificate	2
3	Acknowledgement	4
4	Introduction	6
5	<u>Problem definition</u>	7
6	Project Planning	9
7	Implementation Details	10-14
8	System design /UML Diagrams	15-23
9	Screen shots	24-31
10	Reports	32-33
11	Limitations	34-35
12	Conclusion	36
13	Biography	37
14	References	38

ACKNOWLEDGEMENT

The project “**Task Manager**”

Name	Enrollment No
Babar Atarva Mohan	AF0481890
Babar Shubham Pralhad	AF0481892

Under the Guidance.

I am thankful to my project guide for providing valuable guidance, constant support, and encouragement throughout the completion of this project. His suggestions and advice have been extremely helpful in shaping the project and preparing this project report.

I am also grateful to my family members and friends for their continuous support, encouragement, and motivation, which helped me successfully complete the project.

Lastly, I thank almighty, my parents and friends for their constant encouragement without which this project would not be possible.

Abstract

The project called **Task Manager Web Application** is about creating a responsive platform that allows users to manage their daily tasks quickly, efficiently, and in an organized way. This system replaces the traditional manual method of tracking tasks with a digital platform where users can easily add, update, delete, and categorize their tasks based on priority and status.

This application lets users view tasks in a clean interface, set deadlines, and track progress in real-time. It includes user authentication for secure access, and all data is stored in a MySQL database, making the system reliable and scalable.

The system is simple, user-friendly, and works well across devices. It offers real-time updates, clean task categorization, and efficient status tracking. It helps individuals and teams increase productivity, stay organized, and manage time better. This platform focuses on ease of use, reliability, and performance for both personal and professional task management.

1. Introduction

In today's age of Information Communication and Technology, we rely on digital systems to manage both our personal and professional lives. From organizing schedules to tracking deadlines, managing tasks is a crucial aspect of productivity. Traditional methods like paper-based planners or isolated mobile apps are limited in flexibility, accessibility, and efficiency.

One of the most important developments of this technological era is the advancement of web applications. These applications allow users to interact with data in real-time, from anywhere, and at any time, with minimal cost and greater convenience.

This project focuses on developing a **Task Manager Web Application** that enables users to create, manage, and track their daily tasks. The system is built using the Django framework with Python for backend logic and MySQL for secure, scalable data storage. The platform offers user authentication, real-time task updates, status tracking, and a clean, responsive UI. Whether for students, professionals, or team members, this application simplifies the task management process, improves time management, and increases productivity.

1.1 Objective of the Present Work

The main objectives of this project are as follows:

- To develop a web-based Task Manager application using Django and MySQL.
- To allow users to create, update, delete, and view tasks.
- To include priority levels (Low, Medium, High) and status categories (To Do, In Progress, Completed).
- To implement user authentication and session handling for secure access.
- To provide a simple and intuitive user interface accessible on mobile and desktop.
- To store and retrieve task data efficiently using a relational database.
- To offer a scalable and maintainable system that can support additional features in the future.

3.1 PROBLEM DEFINITION

Many individuals and organizations struggle to manage tasks efficiently using traditional methods such as paper planners or unstructured tools. These methods do not offer real-time updates, task prioritization, or progress tracking, which leads to inefficiencies and missed deadlines.

This project solves these problems by providing a centralized, web-based task management system. It allows users to securely log in, manage tasks, update their status, assign priority levels, and track deadlines. The system improves organization, ensures timely completion of tasks, and offers a clear and intuitive user interface.

3.2 Preliminary Investigation

Purpose

The **Task Manager Web Application** is designed to help users manage their tasks effectively and efficiently through a responsive web platform. It replaces manual task tracking methods with a structured system that offers real-time updates, task categorization, and status management.

Benefits

The system offers several advantages:

- **Efficient Task Management:** Users can create, update, delete, and view tasks easily.
- **Real-Time Status Tracking:** Tasks can be updated with statuses like To-Do, In Progress, and Completed.
- **Priority Assignment:** Tasks can be marked with priority levels—Low, Medium, or High—for better time management.
- **Secure Access:** Each user has an authenticated login to protect their task data.
- **Clean Interface:** The dashboard offers a clear and user-friendly layout accessible from multiple devices.

Proposed System

The proposed Task Manager system is designed to be reliable, structured, and user-friendly:

- Users can log in, manage tasks, and view their progress in real-time.
- The system stores task data in a MySQL database, ensuring data persistence and integrity.
- A responsive frontend ensures a seamless experience across desktop and mobile devices.

3.3 Feasibility Study

This study determines whether the **Task Manager Web Application** can be successfully built and deployed with available resources, technologies, and constraints.

Types of Feasibility Analysis

Technical Feasibility

- Developed using Python's Django framework for robust, scalable backend operations.
- Frontend built with HTML, CSS, and Bootstrap for a clean and responsive interface.
- MySQL database used for secure and consistent data storage.
- Django's built-in features simplify authentication, admin interface, and session handling.

Economic Feasibility

- Entirely based on open-source tools, which eliminates licensing costs.
- Reduces dependence on paid task management tools or commercial software.
- Designed to be lightweight and hostable on low-cost servers or even free tiers.

Operational Feasibility

- Offers a highly intuitive interface that requires no technical knowledge to use.
- Task creation and tracking are designed to be simple, reducing learning curves.
- Application can be used individually or adapted for team use with slight modifications.

Schedule Feasibility

- Development was planned in well-defined phases, making the timeline realistic.
- Modular development enabled faster coding, testing, and debugging cycles.
- Tasks were distributed across planning, design, coding, testing, and deployment.

Social Feasibility

- Encourages disciplined task tracking and promotes productivity among users.
- Can be beneficial for students, professionals, freelancers, and small teams.
- Promotes time management habits, reducing procrastination and missed deadlines.

3.4 Project Planning

Purpose of Project Planning

Project planning is a critical phase in software development that defines the foundation for successful project execution. It ensures that each development stage is conducted in a structured, goal-oriented manner. Proper planning allows for efficient use of resources, minimizes risks, establishes realistic timelines, and keeps the project aligned with its defined objectives.

For the **Task Manager Web Application**, the planning phase provided a roadmap for the development team to follow. It facilitated coordination among tasks, set priorities, tracked progress, and ensured quality control at each phase of the project lifecycle.

Phases Covered in the Plan

To ensure a streamlined and efficient development cycle, the entire project was divided into the following well-defined phases:

◆1.Preliminary Investigation

This phase involved identifying the problem statement and understanding the need for a task management system. Key deliverables included:

- Requirement collection
- Identification of user pain points
- Setting scope and goals of the project

◆2.SystemAnalysis

Here, the functional and non-functional requirements were defined. The development team analyzed how users would interact with the system and what features would be essential.

- Feature listing (task CRUD, status tracking, priority tagging)
- Technical feasibility study
- Data and workflow modeling

◆3.System Design

In this phase, the structural framework of the application was built. The team designed the architecture, database schema, and user interface.

- Creation of ER diagrams and wireframes
- Selection of technology stack (Django, MySQL, Bootstrap)
- Designing navigation and UI layout for responsiveness

◆4.Coding

Actual implementation of the backend and frontend components was done in this phase. Coding followed a modular approach, allowing separate development of views, models, and templates.

- Developed using Django's MVT pattern
- MySQL was integrated for structured data storage
- HTML/CSS/Bootstrap used for responsive UI

◆5.Security

This critical phase focused on securing user data and application integrity.

- User authentication and session management
- Input validation to prevent SQL injection or XSS attacks
- Password encryption using Django's built-in security modules

6.Testing

To ensure the application performs as expected, thorough testing was performed.

- Unit testing of individual modules
- Integration testing for combined workflows
- User Acceptance Testing (UAT) for real-world usability

7.Implementation

In the final phase, the system was deployed and made available for end users. A test run was conducted on local and cloud environments.

- Final deployment using Django's server or external hosting
- Creation of user documentation for easy onboarding
- Backup and migration scripts prepared for database

This well-planned approach ensured that the **Task Manager Web Application** was delivered efficiently, with minimal delays and high performance.

3.6 Software Requirement Specification (SRS)

The Software Requirement Specification (SRS) defines the essential functional and non-functional requirements of the **Task Manager Web Application** to ensure its effectiveness, reliability, and maintainability. It provides a detailed description of the system's architecture, modules, required tools, and hardware environment.

System Overview

The **Task Manager Web Application** is a web-based productivity tool that helps users manage and track tasks in a structured and efficient way. It allows users to create, edit, delete, and monitor the progress of their tasks with features like priority setting, status tracking, and deadline alerts.

The system is organized into two main modules:

1. **User Module** – Allows users to:
 - Register and log in securely
 - Add new tasks with title, description, deadline, and priority
 - Update or delete existing tasks
 - Change task status (To-Do, In Progress, Completed)
 - View tasks in a categorized and user-friendly interface
2. **Admin Module (Optional)** – Enables an administrator to:
 - Manage user accounts (create, delete, or modify users)
 - View all user tasks (if multi-user support is implemented)
 - Maintain application data integrity and system logs

Backend Development:

- **Python 3.8 or higher**
 - Core programming language used for server-side logic.
 - Supports object-oriented design and integrates seamlessly with Django.
- **Django 5.0.2**
 - A high-level web framework that enables rapid development and clean design.
 - Offers built-in authentication, session management, admin interface, form handling, and ORM (Object-Relational Mapping).
- **MySQL Server**
 - Relational database used to persist user and task data.
 - Ensures fast querying, indexing, and ACID compliance.
- **PyMySQL**
 - Python library that allows Django to communicate with MySQL.
- **python-dotenv**
 - Used for managing sensitive environment variables such as database credentials securely.

◆ Frontend Development:

- **HTML5 & CSS3**
 - Markup and styling languages used to create the layout and structure of the web pages.
- **Bootstrap 5**
 - A responsive frontend framework that allows UI components to adjust across screen sizes.
- **JavaScript (optional)**
 - Used for dynamic frontend features (like form validations or interactivity, if implemented).

◆ Development Tools:

- **Visual Studio Code / PyCharm / Sublime Text**
 - Code editors for writing and managing source code.
 - Recommended extensions: Python, Django, Git, Prettier.
- **Git (Version Control)**
 - Used to manage source code changes and collaborate with other developers.
- **XAMPP or MySQL Workbench (for managing databases)**
 - GUI tools to visualize tables, run SQL queries, and backup databases.
 -

◆ *Browser Compatibility:*

- Fully tested on modern browsers including:
 - Google Chrome
 - Mozilla Firefox
 - Microsoft Edge

Hardware Requirements

Depending on the user type (developer, end user, or deployment server), the hardware requirements may vary. Below is a categorized list.

◆ *Developer Machine (for building the project):*

- **Processor:** Intel Core i3 or higher (i5/i7 recommended for multitasking)
- **RAM:** Minimum 4 GB (8 GB or more recommended)
- **Hard Disk:** 500 MB for project source code and dependencies
- **Display:** HD (720p) or Full HD screen
- **Network:** Internet connection required for installing packages and using version control (GitHub)

◆ *End-User Requirements (for using the app in a browser):*

- **Device:** Smartphone, Tablet, Laptop, or Desktop
- **Processor:** Any modern CPU (ARM or x86)
- **RAM:** 2 GB or more
- **Browser:** Updated version of Chrome, Firefox, or Edge
- **Connectivity:** Internet access for interacting with the server

◆ *Deployment Server (for hosting):*

- **OS:** Ubuntu 20.04 LTS / CentOS / Windows Server
- **Processor:** Dual-core or higher
- **RAM:** 2–4 GB minimum
- **Storage:** SSD recommended, 10 GB or more free space
- **Web Server:** Gunicorn, Nginx, or Apache (for production)
- **Database:** MySQL installed and configured
- **Security:** SSL certificates (if deployed online), firewall enabled

3.7 Functional Requirements

1. User Module

Regular users can register and manage their tasks efficiently. The core functionalities available to users include:

- **Secure User Registration and Login**
Users can create an account and securely log in using their credentials.
- **Task Creation**
Users can add new tasks by providing the task name, description, deadline, priority level (Low, Medium, High), and initial status.
- **Task Editing**
Users can modify any part of the task, including updating the due date or changing the priority and status.
- **Task Deletion**
Users can remove tasks that are no longer needed.
- **Status Management**
Tasks can be marked as:
 - **To-Do** (yet to start)
 - **In Progress** (ongoing)
 - **Completed** (finished)
- **Dashboard Access**
The user is presented with a clean dashboard that visually separates tasks by status or priority, helping them stay organized.
- **Mobile Compatibility**
The web interface is responsive and adjusts gracefully across screen sizes, enabling task management from smartphones or tablets.

2. Admin Module

In implementations where an administrative layer is included, an admin user has elevated privileges to manage the overall application and its data.

- **Admin Authentication**
Secure login system allowing only authorized administrators to access admin functionalities.
- **User Management**
View registered users and optionally delete inactive accounts or manage permissions.
- **System Monitoring**
Access logs or records of task creation and deletion for audit purposes.
- **Data Integrity Checks**
Ensure that tasks are correctly linked to users and maintain database hygiene.

(Note: This module is optional and can be extended for multi-user or enterprise use cases.)

3.8 Software Engineering Paradigm

The development of the **Task Manager Web Application** follows a structured and methodical software engineering paradigm to ensure efficiency, clarity, maintainability, and reliability. The adopted approach allows for linear development with iterative feedback to refine system components based on evolving insights and user testing.

3.9 Data Flow Diagram:

A **Data Flow Diagram (DFD)** visually represents the movement of data through the system. It outlines how user inputs are processed, stored, and displayed. In the context of this project, DFDs depict the flow of information between the user, task management system, and database.

Purpose of DFD

The DFD provides:

- A clear, visual understanding of system boundaries and processes
- A communication tool between developers and stakeholders
- A blueprint for logical data interaction

Levels of DFD for Task Manager

⚡ Zero-Level DFD (Context Level)

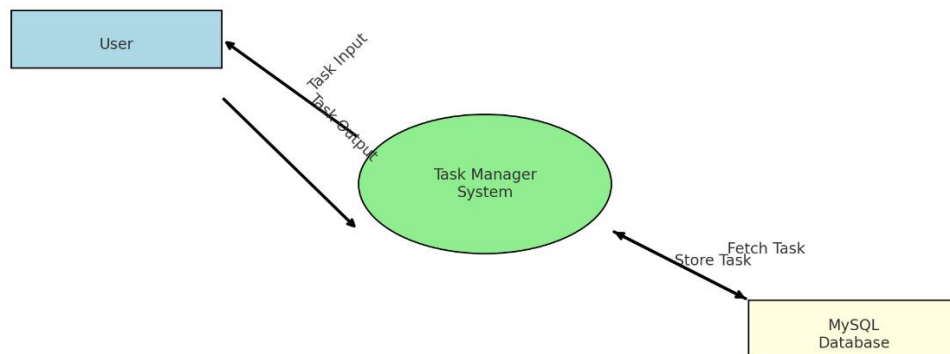
This level presents an overview of the entire system:

- Users interact with the system through task-related functions: Add Task, Update Task, Delete Task, View Dashboard.
- The system processes and stores all data in the central MySQL database.

Entities Involved:

- User
- Task Management System
- MySQL Database

Zero-Level DFD: Task Manager Web Application



◆ *First-Level DFD*

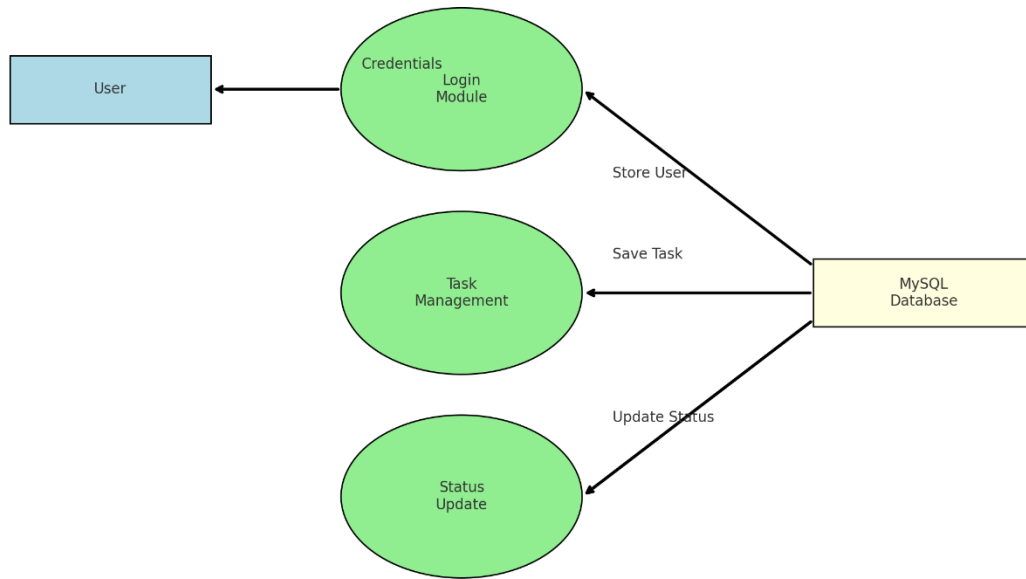
Breaks the system into more specific functions:

- Login and Authentication
- Task CRUD (Create, Read, Update, Delete) operations
- Dashboard retrieval
- Status update processes

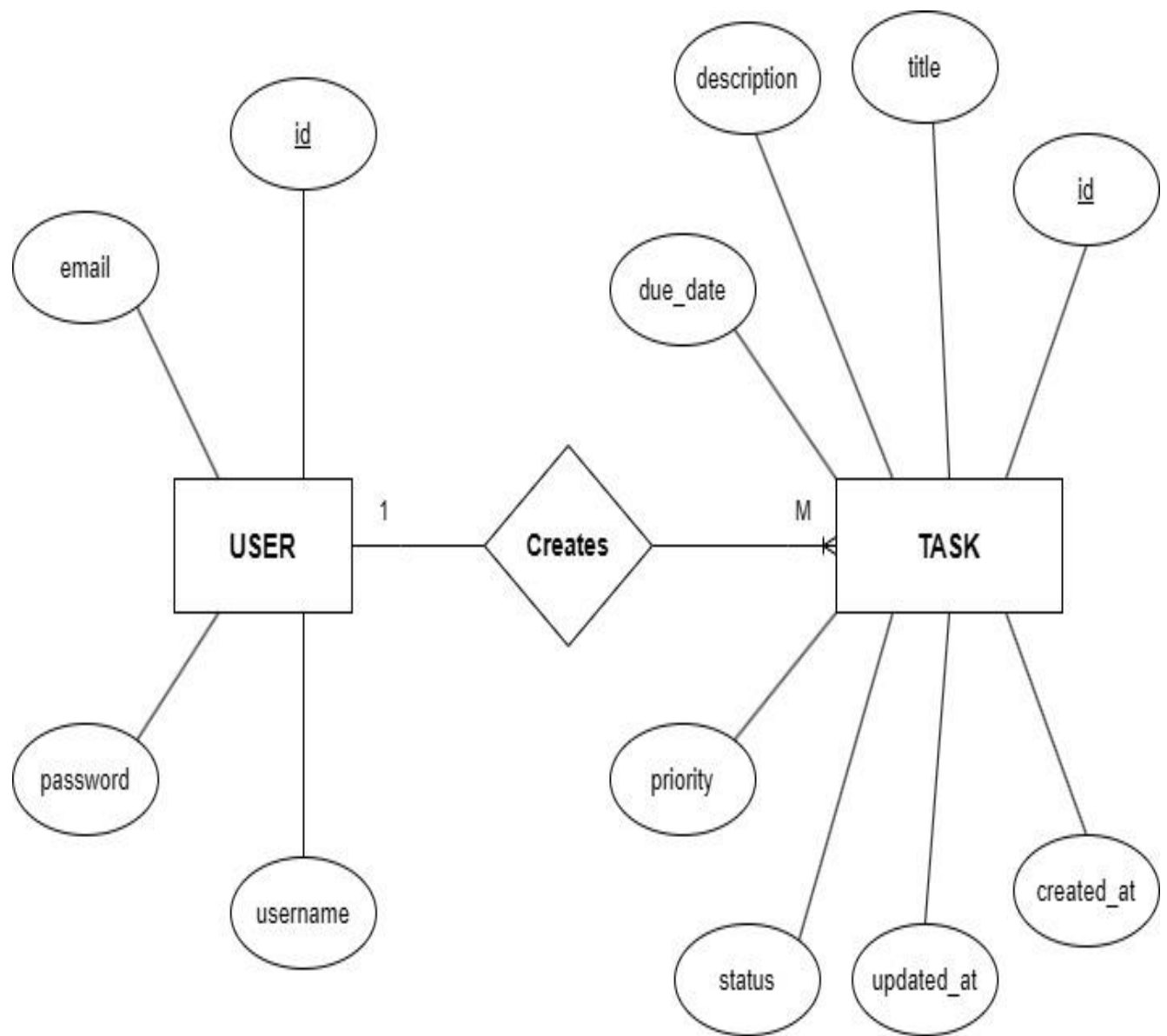
Modules:

- User Login/Registration
- Task Management
- Data Persistence

First-Level DFD: Task Manager Web Application

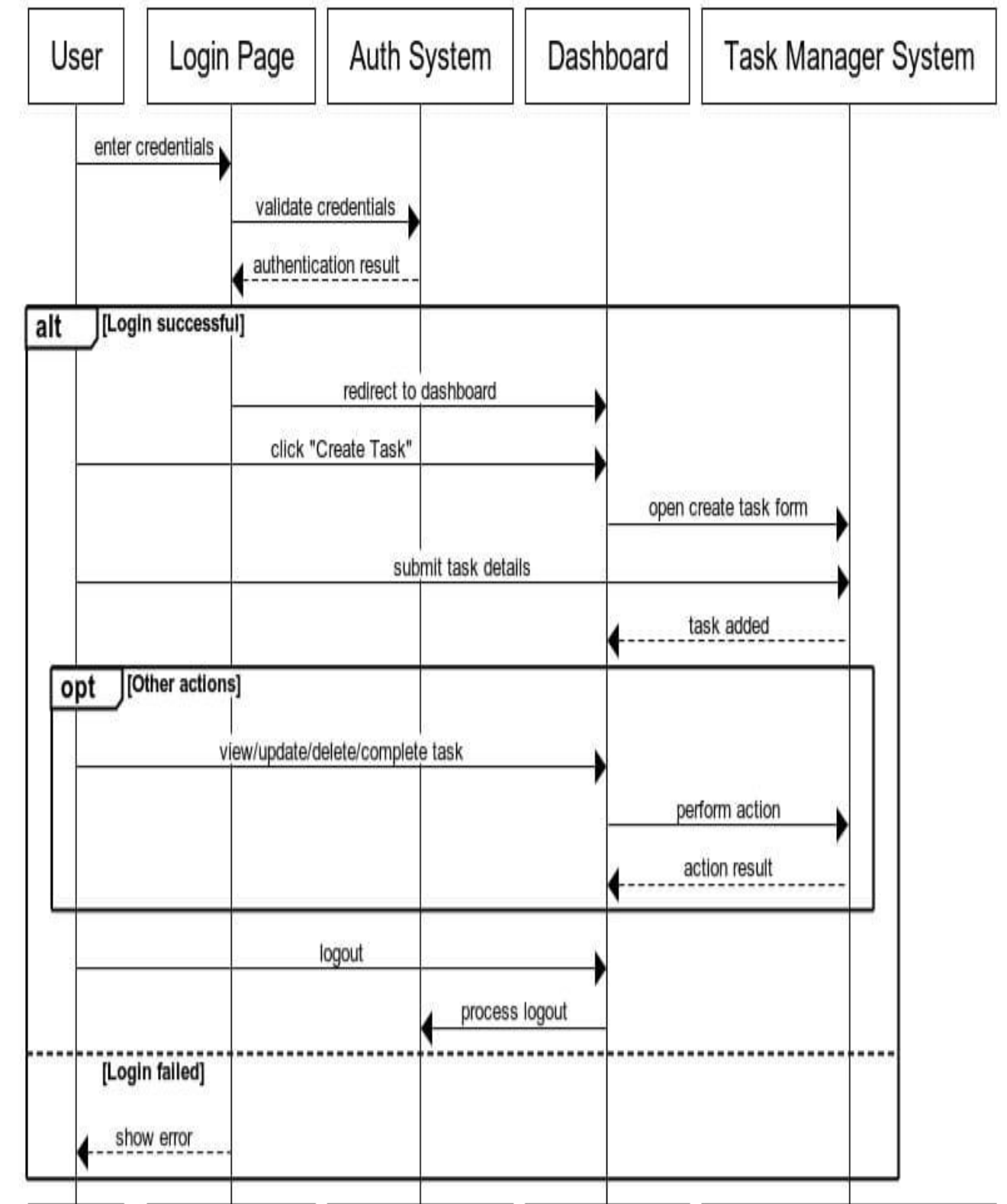


ER diagram

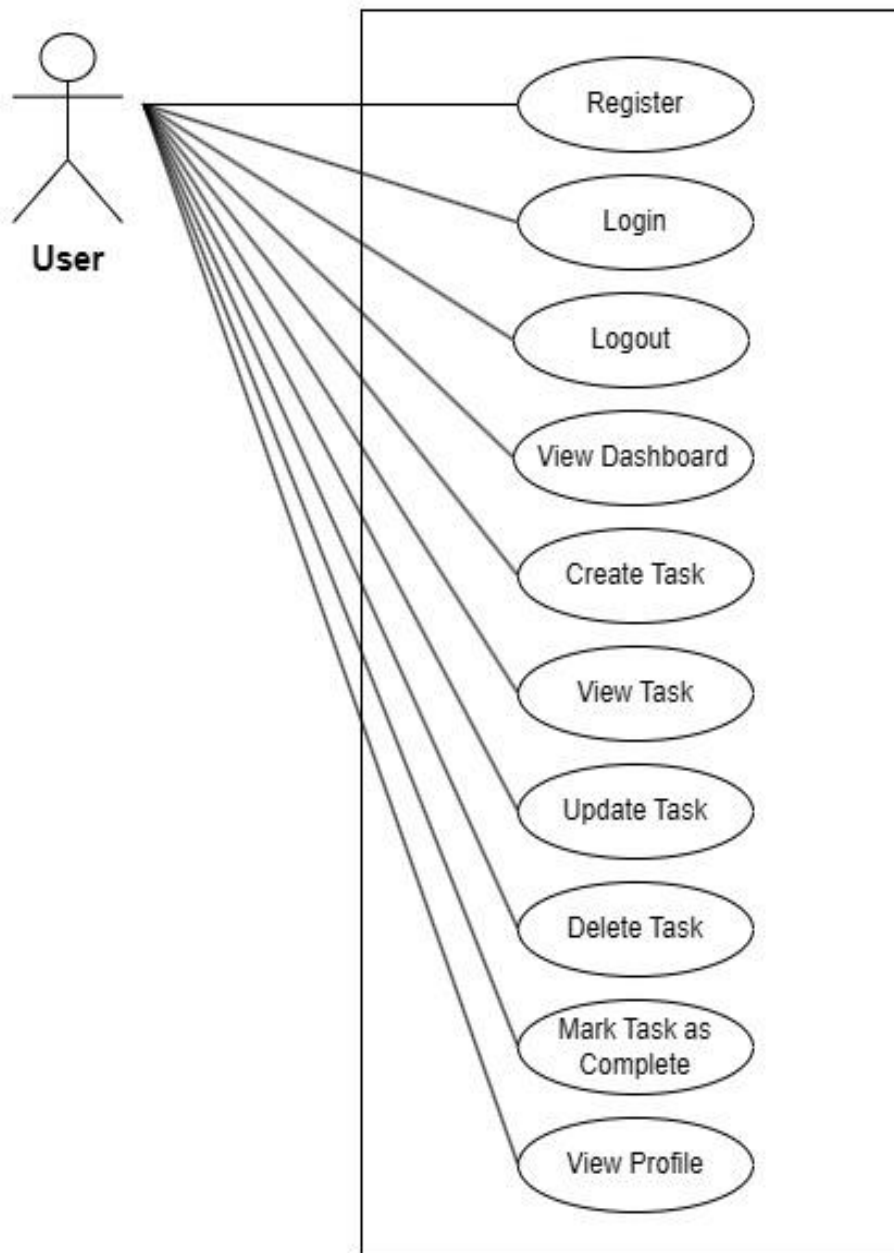


Entity Relationship (ER) Diagram

SEQUENCE DIAGRAM

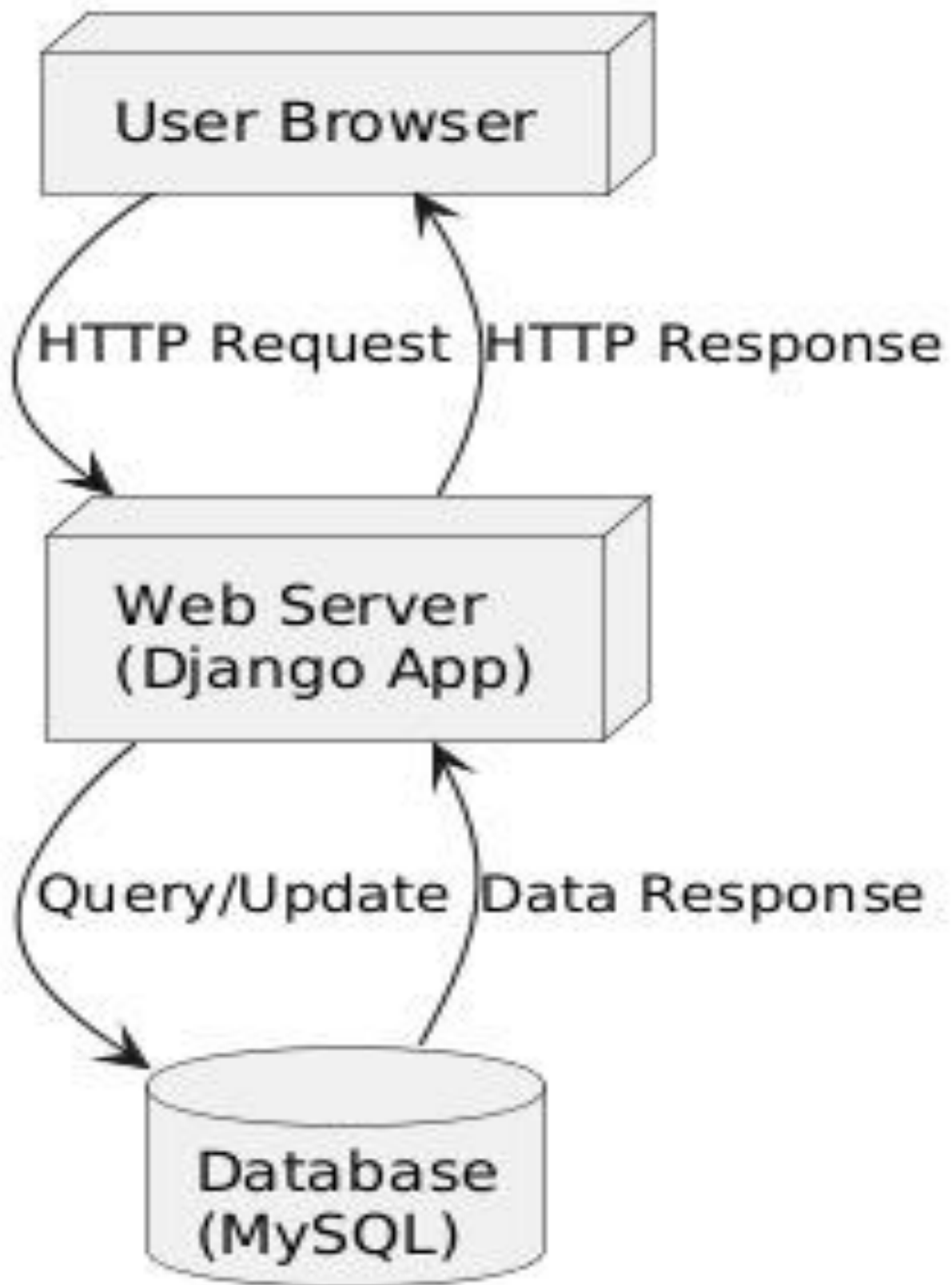


Use Case DIAGRAM

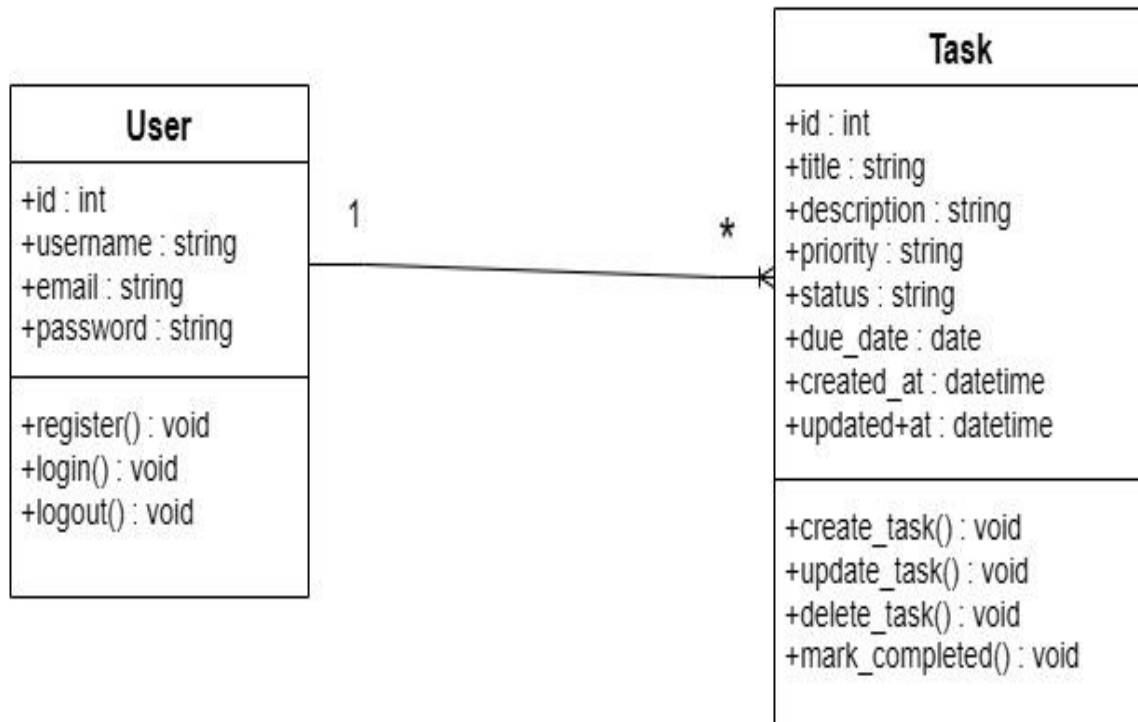


Use Case Diagram

Deployment Diagram:

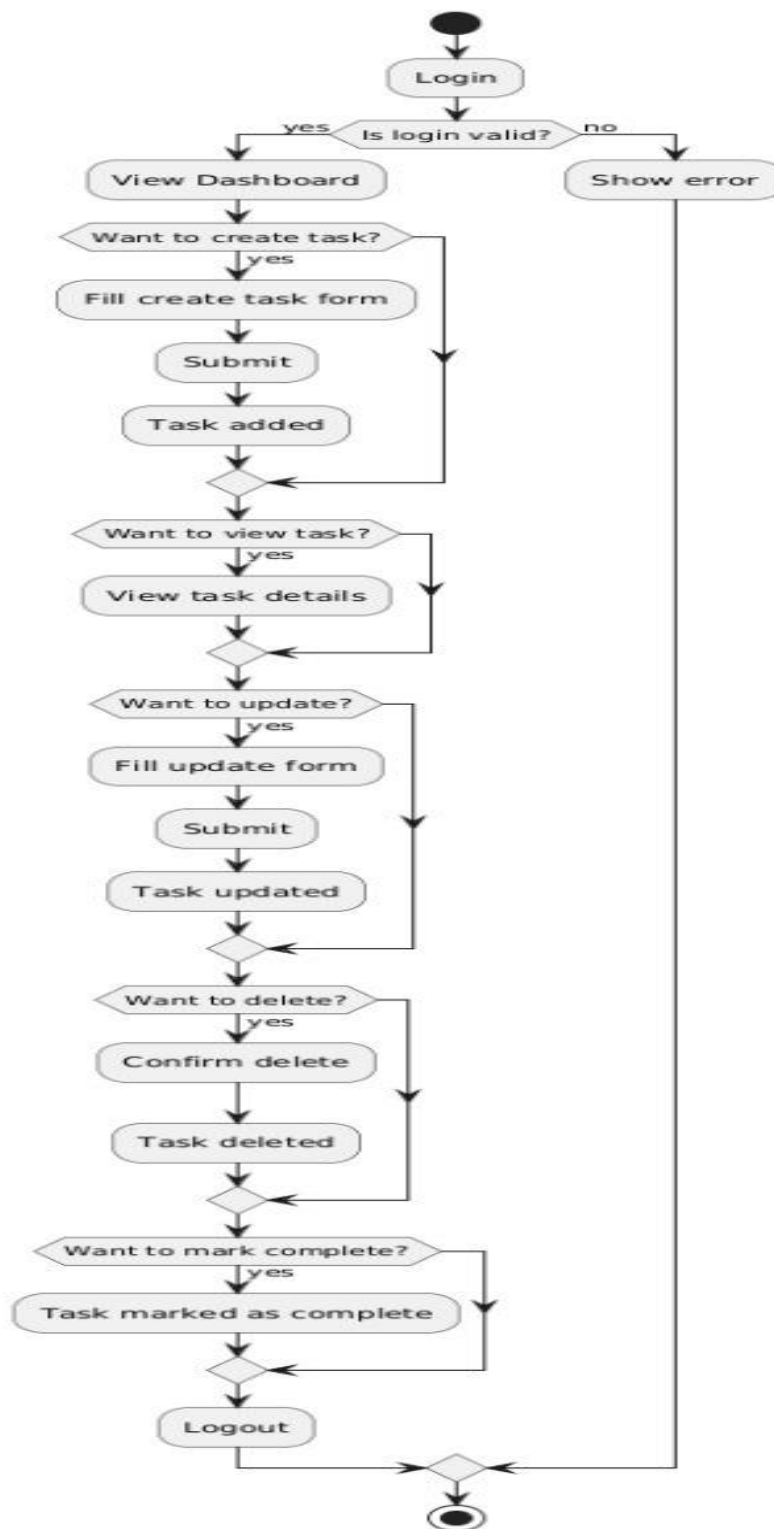


CLASS DIAGRAM



Class Diagram

Activity Diagram:



4. System design

4.1 Modules

The **Task Manager Web Application** is logically divided into the following modules. Each module handles a specific set of responsibilities to keep the application organized, modular, and efficient.

➤ 1. User Module

Users are the primary operators of the system. They can:

- **Register and Log In** securely using email and password.
- **Create Tasks** by entering a title, description, deadline, and priority level.
- **Edit or Update Tasks** as needed.
- **Delete Tasks** that are no longer relevant.
- **Change Task Status** to To-Do, In Progress, or Completed.
- **View All Tasks** in a clean dashboard, categorized by status or priority.
- **Log Out** safely after completing work.

The user module emphasizes simplicity, accessibility, and security for personal task management.

➤ 2. Admin Module (Optional – for team-based usage)

The admin has full control over the system in cases where multiple users are managed centrally:

- **Secure Login System:** Admins must log in to access admin features.
- **User Management:** Admins can view user activity, delete inactive users, and monitor logs.
- **System Monitoring:** Track system health, logs, and active session data.
- **Backup and Maintenance:** Ensure database safety and export functionalities (future scope).

This module can be extended for multi-user or enterprise versions of the application.

4.2 DATA STRUCTURE OF ALL MODULES:

We have structured a **MySQL database** for the Task Manager Web Application. It is organized using both **custom tables** (built for the task manager's functionality) and **default tables** (auto-generated by Django for authentication and permissions). This database can be accessed directly or sequentially by authenticated users and ensures organized, secure, and scalable data management.

✓ Customized Tables Details

✦ *User Table*

Table name: taskapp_customuser

This table stores user registration and login details including username, email, and encrypted password.

Field Name	Data Type	Description
id	Integer (PK)	Unique User ID
username	Varchar	Username
email	Varchar	User's email
password	Varchar	Hashed password
is_staff	Boolean	Admin access flag
is_active	Boolean	Account status
date_joined	DateTime	Registration timestamp

✦ *Task Table*

Table name: taskapp_task

This table stores individual task records created by users.

Field Name	Data Type	Description
id	Integer (PK)	Unique Task ID
user_id	ForeignKey	Linked user ID
title	Varchar	Title of the task

Field Name	Data Type	Description
description	Text	Detailed description
due_date	Date	Deadline of the task
created_at	DateTime	Creation timestamp
updated_at	DateTime	Last updated time

◆ *Priority Table*

Table name: taskapp_priority

This table defines available priority levels.

Field Name	Data Type	Description
id	Integer (PK)	Priority ID
name	Varchar	Priority name (Low, etc.)

◆ *Status Table*

Table name: taskapp_status

This table defines the task statuses.

Field Name	Data Type	Description
id	Integer (PK)	Status ID
name	Varchar	Task status (To-Do, Completed)

⚙ Default Tables Details

◆ *auth_user*

Django's default table to store authentication-related data for users.

◆ *auth_group*

Used to define groups of users.

◆ *auth_group_permissions*

Manages group-to-permission relationships.

◆ *auth_permission*

Stores all permission definitions.

◆ *django_admin_log*

Stores records of changes made via the Django admin interface.

◆ *django_content_type*

Stores metadata about each model in the Django project.

◆ *django_migrations*

Tracks applied migrations for consistent database versioning.

◆ *django_session*

Stores session data for user login sessions

Screenshots

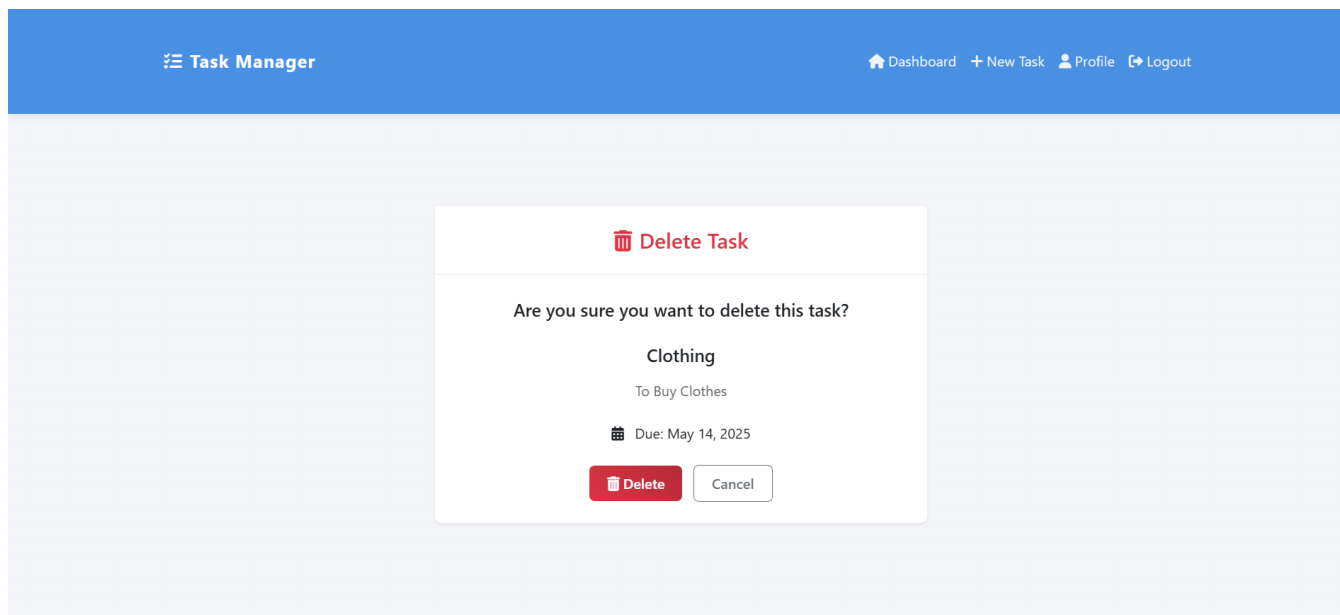
Add Task Page

The screenshot shows the 'Add Task Page' of the Task Manager application. The page has a blue header with the 'Task Manager' logo and navigation links: 'Dashboard', '+ New task', 'Profile', and 'Logout'. The main content area is light gray. A white modal form titled '+ New Task' is centered on the screen. It includes a 'Back to Dashboard' button in the top right corner. The form has three input fields: 'Title' (with placeholder 'Enter task title'), 'Description' (with placeholder 'Enter task description'), and 'Priority' (a dropdown menu currently set to 'Medium'). To the right of the 'Priority' field is a 'Due Date' field with a calendar icon and placeholder 'dd-mm-yyyy'. At the bottom of the form is a blue button labeled '+ Create Task'.

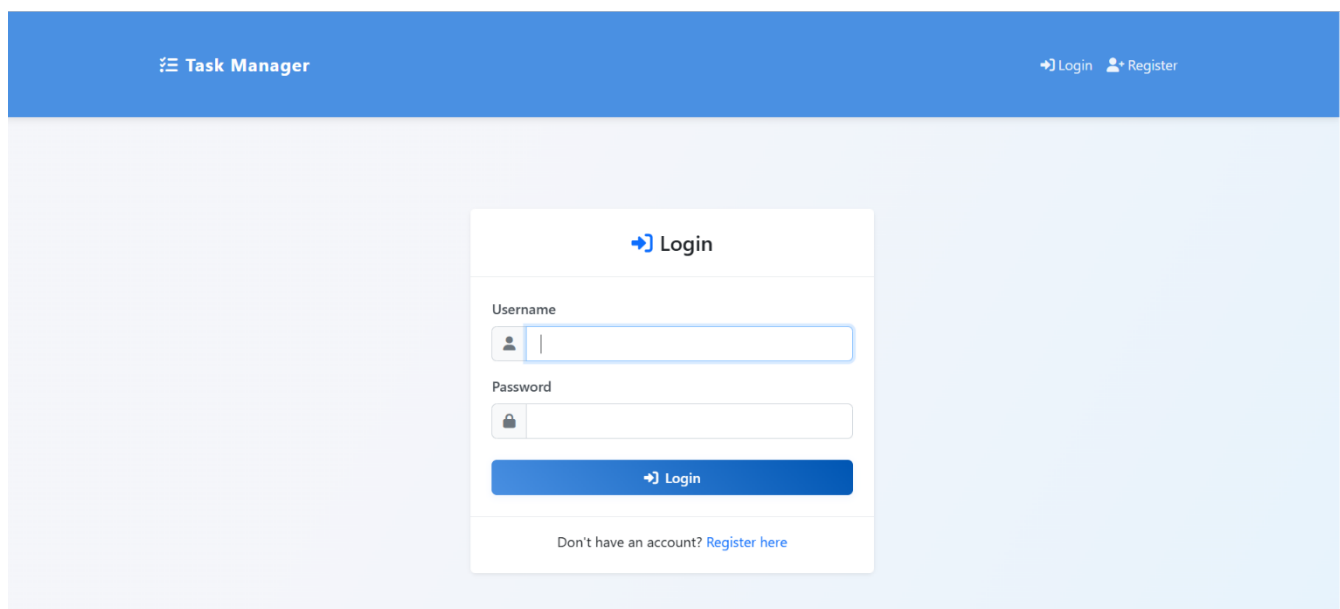
Dashboard:

The screenshot shows the 'Dashboard' of the Task Manager application. The page has a blue header with the 'Task Manager' logo and navigation links: 'Dashboard', '+ New Task', 'Profile', and 'Logout'. The main content area is light gray. At the top, there are four colored cards: 'Total Tasks' (blue, 2), 'Completed' (green, 1), 'Pending' (yellow, 1), and 'Overdue' (red, 0). Below these cards is a filter section with three dropdown menus: 'Priority' (set to 'All Priorities'), 'Status' (set to 'All Status'), and 'Due Date' (set to 'dd-mm-yyyy'). To the right of these dropdowns is a blue button labeled 'Filter'. Below the filter section is a section titled 'Your Tasks' with a '+ New Task' button in the top right corner. It contains a table with the following columns: 'Title', 'Priority', 'Status', 'Due Date', and 'Actions'. The table has one row with the title 'shopping', priority 'High', status 'Pending', and an 'Actions' column containing three icons: a pencil, a trash can, and a checkmark. The bottom of the screenshot shows a Windows taskbar with various application icons and a system tray with the date '13-05-2025' and time '23:45'.

Delete Task :




Edit Task Page:




Profile Page:


Task Manager

Dashboard + New Task Profile Logout




Shubham

 shubhambabar1022@gmail.com

 Member since May 13, 2025


View Tasks

Task Statistics




2

Total




1

Completed



1

Pending



0

Overdue

Task Completion Rate


50%


Register Page:


Task Manager


Login Register

Register









Register

30

Task Details:

Task Manager

DashboardNew TaskProfileLogout

Task Details

EditDelete

Clothing

MediumCompleted

Due Date

May 14, 2025

Created

Description

To Buy Clothes

Back to Dashboard

Task List Page:

3
Total Tasks

1
Completed

2
Pending

0
Overdue

Priority

Status

Due Date

Medium

All Status

22-05-2025

Filter

Your Tasks

+ New Task

Title	Priority	Status	Due Date	Actions
Study	Medium	Pending	May 22, 2025	<div>EditDeleteCheck</div>
shopping	High	Pending	May 22, 2025	<div>EditDeleteCheck</div>
Clothing	Medium	Completed	May 14, 2025	<div>EditDelete</div>

Reports

The **TaskManager** task management system provides essential reports that help users and administrators monitor productivity, user behavior, and system performance. These reports enhance user experience and support data-driven decisions for system improvements and future enhancements.

1. Task Activity Report

- Displays the **total number of tasks created**, completed, and pending over a selected time period.
- Highlights **task trends**, such as weekly task completion rates or overdue patterns.
- Useful for users to assess their productivity and adjust workload distribution.

2. User Engagement Report

- Tracks **new user registrations**, login frequency, and total tasks per user.
- Helps identify **active users**, dormant accounts, and usage trends.
- Supports future feature development and targeted notifications based on activity levels.

3. Task Distribution Report

- Categorizes tasks by **priority** (High, Medium, Low) and **status** (Pending, In Progress, Completed).
- Visual dashboards can be integrated to display pie charts or bar graphs for easy analysis.
- Assists users in identifying areas where work is being delayed or stacked.

4. Overdue Tasks Report

- Lists all tasks that are **past their due date**.
- Includes user, task title, due date, and days overdue.
- Useful for prioritizing workload and ensuring important deadlines are met.

6. Audit and Log Report (Optional / Future Scope)

- Logs **user actions** such as task creation, edits, deletions, and logins.
- Useful for **security auditing**, rollback in case of user errors, and tracking usage patterns.
- May be extended in the future to support **exportable logs** and version tracking.

Limitations

While **TaskManager** provides essential features for task tracking and productivity, the current version has certain limitations that may affect its suitability for large-scale or collaborative environments:

1. Scalability and Performance

The platform is currently optimized for **individual users or small teams**. Performance may degrade with a high number of concurrent users or tasks without further optimization strategies such as **database indexing, caching, or asynchronous task handling**.

2. No Team Collaboration Support

TaskManager is a **single-user-based system**. There is no functionality for **sharing tasks**, assigning tasks to other users, or managing group projects, which limits its use in team or organization-wide scenarios.

3. Limited Notification System

There is **no integration with email or push notification services**. Users do not receive alerts for upcoming deadlines, overdue tasks, or completed tasks, which could improve time management and user engagement.

4. Basic UI/UX Design

Although the interface is functional and mobile-responsive via Bootstrap, it lacks **advanced front-end features** such as dynamic filtering, drag-and-drop tasks, dark mode, or accessibility enhancements for differently-abled users.

5. No Calendar or External Tool Integration

The current version does not support integration with external tools like **Google Calendar**, Trello, or Slack, which would allow better task visualization and ecosystem connectivity.

6. Security Enhancements Needed

While Django's authentication provides basic security, advanced measures such as **Two-Factor Authentication (2FA)**, **session timeout**, or **login history tracking** are not yet implemented.

7. Limited Reporting Features

The reporting system provides **basic insights** only. There is no support for **exporting reports (PDF/CSV)**, filtering by date ranges, or analyzing productivity trends in depth.

These limitations do not hinder the core purpose of TaskManager but highlight areas for **future enhancement** as the platform evolves toward a more collaborative and feature-rich solution.

Conclusion

The **TaskManager** project is a fully functional, database-driven task management system built using the Django web framework and MySQL. It provides users with a platform to efficiently create, manage, and track tasks based on priority, status, and due dates. The system emphasizes modular development, clean backend architecture, and user-focused functionality, showcasing a strong understanding of full-stack web development principles.

Throughout this project, key concepts such as user authentication, role-based access control, dashboard reporting, and data validation have been effectively implemented. The use of Django models, views, forms, and signals demonstrates how complex features can be structured in a scalable and maintainable manner.

While the current system is tailored for individual users or small-scale usage, it lays a strong foundation for future enhancements. Planned improvements include collaborative features, notification systems, mobile app integration, and advanced analytics.

In conclusion, TaskManager not only meets its core objectives of task organization and productivity enhancement, but also serves as a robust framework for further development into a comprehensive task and project management solution.

Bibliography

1. **Django Documentation** – Official guide for Django web framework.
<https://docs.djangoproject.com>
2. **MySQL Developer Guide** – Reference for relational database integration and SQL commands.
<https://dev.mysql.com/doc/>
3. **Python Official Documentation** – Detailed explanation of Python libraries, syntax, and core modules.
<https://docs.python.org>
4. **W3Schools Django Tutorial** – Beginner-friendly resources on Django models, views, and templates.
<https://www.w3schools.com/django/>
5. **MDN Web Docs** – Front-end technologies reference (HTML, CSS, JavaScript).
<https://developer.mozilla.org/>
6. **GeeksforGeeks** – Tutorials and explanations for Django, Python, and database design.
<https://www.geeksforgeeks.org>
7. **Stack Overflow** – Community-driven solutions for debugging and code optimization.
<https://stackoverflow.com>
8. **Real Python** – Practical tutorials and advanced Python/Django articles.
<https://realpython.com>

References

1. **"Python Crash Course"** by Eric Matthes
 - A hands-on, project-based introduction to Python.
2. **"Two Scoops of Django"** by Audrey Roy Greenfeld & Daniel Roy Greenfeld
 - Best practices for building Django applications.
3. **"Learning Python"** by Mark Lutz
 - In-depth reference for mastering Python programming.
4. **"Head First HTML and CSS"** by Elisabeth Robson and Eric Freeman
 - A beginner-friendly guide to frontend development.
5. **"MySQL: The Complete Reference"** by Vikram Vaswani
 - Comprehensive guide to MySQL queries, data design, and management.
6. **"Django for Beginners"** by William S. Vincent
 - Step-by-step tutorial book for building Django web apps.
7. **"Clean Code"** by Robert C. Martin
 - Teaches writing readable, maintainable, and clean code (general software practice).
8. **"Web Development with Django Cookbook"** by Jake Kronika
 - Recipes and tips for solving Django web application problems.
9. **Django Official Documentation** – <https://docs.djangoproject.com/>
10. **W3Schools HTML/CSS/JavaScript Reference** – <https://www.w3schools.com/>