**Contents**

**Abstract**

**1. Introduction-------------------------------------------------------------------- 1-2**

**2. Problem Statement------------------------------------------------------------- 2**

**3. Objectives------------------------------------------------------------------------ 2**

**4. Methodology / Development Process--------------------------------------- 3**

**5. Requirements------------------------------------------------------------------- 4**

**6. System Design------------------------------------------------------------------ 4-5**

**7. Timeline ------------------------------------------------------------------------- 6**

**8. Expected Outcome------------------------------------------------------------- 7**

**9. References----------------------------------------------------------------------- 7**

**Abstract**

Taskly is a web-based task management system that helps teams organize their work more efficiently. Managers can assign and track tasks, and employees can easily view and update their work. The system will include task CRUD operations, priority tagging, a dashboard, login features, and file uploads. It will be built with Django, HTML, Tailwind CSS, PostgreSQL. It is deployed on Render through GitHub integration. Taskly aims to provide a simple, secure, and responsive platform that improves collaboration and accountability within teams.

**1.Introduction**

**Background**

In any team setting—whether in a company or student group—managing tasks effectively is essential. Teams often struggle with unclear responsibilities and lack of visibility into task progress. A task management system provides structure, enabling clear task assignment, tracking, and accountability.

**Motivation**

As a student, I have experienced group projects where task distribution, priorities, and progress tracking were not always clear. Sometimes tasks weren’t properly assigned, and we didn’t know who was doing what. It often led to confusion and delays. That made me realize that even small teams need a structured way to stay organized and work efficiently.

This inspired me to propose **Taskly**—a simple task management system that will help teams stay organized and focused. I plan to design it with real teamwork in mind, introducing roles like Manager and Employee to reflect how tasks are handled in real-world workflows. In **Taskly**, Managers will be able to assign tasks, set priorities, and track overall progress, while Employees can easily view their assigned work, update statuses, and focus on what matters most. This clear separation of roles is expected to improve accountability and communication within teams.

**Scope**

**Included:**

• Dashboard with task statistics (Manager, Employee)

• Task creation, editing, and deletion

• Priority tagging and team assignment

• User registration and login

• Role-based access (Admin, Manager, Employee)

• File upload for task attachments

• Deployment to Render from GitHub

**Excluded:**

• Real-time messaging

• Mobile app version

• Advanced analytics or notifications

**2. Problem Statement**

Managing tasks without a proper system often leads to disorganization. There is no clear way to assign tasks, track progress, or manage roles. The proposed system, Taskly, aims to solve this problem by providing a structured, role-based web application. It will allow users to manage tasks, assign responsibilities, and monitor progress efficiently in one place.

**3. Objectives**

1. Build a full-stack web application for task management.
2. Enable CRUD operations for tasks and users.
3. Design a responsive dashboard with task statistics.
4. Implement secure login and role-based access control.
5. Allow file uploads for task attachments.
6. Deploy the system to Render using GitHub integration.

**4. Methodology / Development Process**

**System Overview:**

Taskly will follow Django’s MVT (Model-View-Template) architecture. The frontend will be developed with HTML and Tailwind CSS for a clean, responsive design. Django will handle backend logic such as task assignment and user roles, while data will be stored in a PostgreSQL database. GitHub will be used for version control, and the application will be deployed on Render with automatic updates through GitHub integration.

**A diagram of a diagram

AI-generated content may be incorrect.**

**Figure 01: Taskly System Architecture**

**Tools & Technologies**

* **Backend:** Django (Python)
* **Database:** PostgreSQL
* **Frontend:** HTML, Tailwind CSS
* **Version Control:** Git, GitHub
* **Deployment Platform:** Render

**5. Requirements**

**Functional Requirements**

1. Dashboard with task statistics
2. Task creation, editing, and deletion
3. Priority tagging (Low, Medium, High)
4. Role-based access (Admin, Manager, Employee)
5. User registration and login
6. Team assignment
7. File upload for task attachments

**Non-Functional**

* **Scalability:** Built with Django’s MVT architecture, which makes it easy to expand and maintain the system.
* **Availability:** The deployed version should have 99% uptime.
* **Security:** HTTPS is used for deployment to protect data.
* **Usability:** Responsive.
* **Performance:** Key pages will load within 2 seconds during normal use.

**6. System Design**

**The system will include three main features:**

1. **Dashboard:**
   * Displays overall task statistics.
   * Shows recent activity and updates.
2. **Task Management:**
   * Supports CRUD (Create, Read, Update, Delete) operations for tasks.
   * Allows setting task priorities (e.g., high, medium, low).
   * Enables assigning tasks to team members.
3. **Authentication:**
   * Manages user login and registration.
   * Controls access based on user roles (e.g., admin, member).

**Use Case Diagram**

**A diagram of a software system

AI-generated content may be incorrect.**

**Figure 02: Taskly Use Case Diagram**

**7. Project Timeline:**

|  |  |  |
| --- | --- | --- |
| **Week** | **Task / Milestone** | **Focus Area** |
| Week 1 | Project Idea Submission | Communication, Planning |
| Week 2 | Proposal Report Submission | Planning, Modeling |
| Week 3 | Environment Setup: Create Django project. | Setup |
| Week 4 | Set up GitHub repo, and configure database (PostgreSQL) and Tailwind CSS for responsive design. | Setup |
| Week 5 | Dashboard: Show task stats, recent activity, and  user-specific summary ( (Manager, Employee). | UI/UX |
| Week 6 | Improve Dashboard: Show task stats, activity logs, dynamically. | Frontend + Integration |
| Week 7 | |  | | --- | | Implement CRUD operations (task creation, editing, deletion) |  |  | | --- | |  | | Backend |
| Week 8 | Implement authentication system with role-based access (Admin, Manager, Employee). | Backend + Security |
| Week 9 | User Login and Logout implementation | Backend + Security |
| Week 10 | Create User Profile | Backend + UI |
| Week 11 | Testing & Debugging: fix bugs and refine UI/UX. | Quality Assurance |
| Week 12 | Deployment Setup: Deploy on Render, connect GitHub. | Deployment |
| Week 13-14 | Final Presentation & Demonstration: Present Taskly and show working prototype. | Deployment |

**8. Expected Outcome**

* A functioning task management portal that includes dashboard, task creation, and login.
* Role-based access for Admin, Manager, and Employee
* File upload
* Responsive design using Tailwind CSS
* Deployed version with a public access link
* GitHub repository with documented code

**9. References**

**1. Django Project : https://www.djangoproject.com**

**2. Tailwind CSS : https://tailwindcss.com**

**3. Render Deployment :** [**https://render.com**](https://render.com)