A Project
SmartTask

CSE-318 : System Analysis and Design Lab

Submitted By:

Name	ID	Intake	Section
MD.Jahidul Islam Shihab	20234103347	52	09
MD. Al Nasir Uddin Siam	20234103349	52	09
Munim Halder	20234103351	52	09
Yeasir Ibna Hasibur Rahman	20234103358	52	09

Supervised By :

Shefayatuj Johara Chowdhury

Lecturer

*Department of **CSE***

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BANGLADESH UNIVERSITY OF BUSINESS AND TECHNOLOGY

(BUBT)

ABSTRACT

We presents an advanced SmartTask platform, designed to streamline personal and professional time management through a unified interface with multiple functional modules. The system features task creation, scheduling, priority setting, reminders, and progress tracking, enabling users to efficiently organize their daily activities. Administrators can manage user profiles and monitor task completion across multiple users, while individuals benefit from real-time notifications, task categorization, and performance analytics. With centralized database integration and intuitive user controls, SmartTask ensures secure storage of task data, seamless synchronization across devices, and insightful reports on productivity patterns. Overall, the platform enhances time utilization, reduces procrastination, and supports effective planning and goal achievement.

DECLARATION

I hereby declare that the project entitled “Game Hive” submitted for the degree of Bachelor of Science in Computer Science and Engineering in the faculty of Computer Science and Engineering of Bangladesh University of Business and Technology (BUBT), is our original work. I affirm that all the content presented in this project is created by us and does not contain any materials previously published or written by any other person. Furthermore, we assert that this project does not include any content that has been accepted for the award of the degree or diploma to any other candidate. Any external sources or references utilized in this project have been duly cited and acknowledged.

Md. Jahidul Islam Shihab
ID: 20234103347
Intake-52
Section-9

MD. Al Nasir Uddin Siam
ID: 20234103349
Intake-52
Section-9

Munim Halder
ID: 20234103351
Intake-52
Section-9

Yeasir Ibna Hasibur Rahman
ID: 20234103358
Intake-52
Section-9

CERTIFICATION

This project report titled “SmartTask” submitted by MD.Jahidul Islam Shihab ,Md. Al Nasir Uddin Siam ,Munim Halder and Yeasir Ibna Hasibur Rahman students of the Department of Computer Science and Engineering, Bangladesh University of Business and Technology (BUBT), under the supervision of Shefayatuj Johara Chowdhury , Lecturer, Department of Computer Science and Engineering, has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Engineering.

Shefayatuj Johara Chowdhury
Lecturer
Department of CSE

Md. Saifur Rahman
Assistant Professor
&
Chairman
Department of CSE

DEDICATION

Dedicated to our parents for all their love,support and inspiration.I would love to dedicate our

well wishers who helped us and supported us when needed.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to Shefayatuj Johara Chowdhury, Lecturer and my project supervisor, for her guidance, expertise, and encouragement, which were instrumental in the successful completion of this project.

I also thank all the lecturers and instructors of the Department of Computer Science and Engineering, Bangladesh University of Business and Technology (BUBT), for their valuable guidance and support in teaching the fundamentals of coding and software development.

Finally, I am grateful to my family and friends for their continuous encouragement throughout this journey.

APPROVAL

This project report titled “SmartTask” submitted by MD.Jahidul Islam Shihab (ID:20234103347) ,Md. Al Nasir Uddin Siam (ID:20234103349) ,Munim Halder (ID:20234103351) and Yeasir Ibna Hasibur Rahman (ID:20234103358) students of the Department of Computer Science and Engineering (CSE), Bangladesh University of Business and Technology (BUBT), under the supervision of Shefayatuj Johara Chowdhury, Lecturer, Department of CSE, has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science (B.Sc.) in Computer Science and Engineering. This report has been approved as to its style and contents.

Supervisor:
Shefayatuj Johara Chowdhury
Lecturer
Department of Computer Science and Engineering (CSE)
Bangladesh University of Business and Technology (BUBT)
Mirpur-2, Dhaka-1216, Bangladesh

Chairman:
Md. Saifur Rahman
Assistant Professor and Chairman
Department of Computer Science and Engineering (CSE)
Bangladesh University of Business and Technology (BUBT)
Mirpur-2, Dhaka-1216, Bangladesh

Abbreviations and Nomenclature

STM – Smart Task Manager

DB – Database

UI – User Interface

UX – User Experience

HTML – HyperText Markup Language

PY- Python

CRUD – Create, Read, Update, Delete

API – Application Programming Interface

AP – APScheduler

DFD – Data Flow Diagram

ERD – Entity-Relationship Diagram

BDT – Bangladeshi Taka

TABLE OF CONTENTS

List of Figures

List of Tables

List of Snippets

Chapter-1: Introduction

1.1 Introduction

Managing daily tasks effectively is essential for productivity. This project aims to create a web-based Daily Task Planner that lets users schedule their tasks and get notified on time. The software will save tasks in a MySQL database and provide functionalities to view, edit, and delete tasks. Recurring tasks etc will also be supported.

1.2 Problem Statement

1.Users often forget important daily tasks due to lack of reminders.

2. Most existing apps are either too complex or cluttered.

3. Simple apps lack useful features like task repetition or alerts.

4. Manual task tracking wastes time and causes disorganization.

5. There's a need for a lightweight, efficient, and user-friendly reminder system.

1.3 Objectives

1. Provide a simple web-based interface to add, schedule, edit, delete, and manage recurring tasks.
2. Send timely reminders to users for their scheduled tasks.
3. Maintain a persistent task list stored securely in a database for easy access anytime.

1.4 Functionalities

- **Add Task:** Input task name and time.
- **View Tasks:** Display tasks in a list with status.
- **Edit/Delete Task:** Update or remove tasks.
- **Recurring Tasks:** Set tasks to repeat daily.
- **Reminder Alerts:** Notify users on task time.
- **Data Persistence:** Store all tasks in MySQL.

1.5 Implementation Details

The system will be based on a classic web application architecture:

1. **Frontend:** HTML forms for task input, tables to display tasks, and buttons for editing/deleting.
2. **Backend**: PHP handles CRUD operations for tasks and serves pages.
3. **Database:** MySQL stores task details including name, time, status, and recurrence.
4. **Reminder/Notification:** PHP script with time comparison logic paired with JavaScript for front-end notifications.

1.6 Agile Model

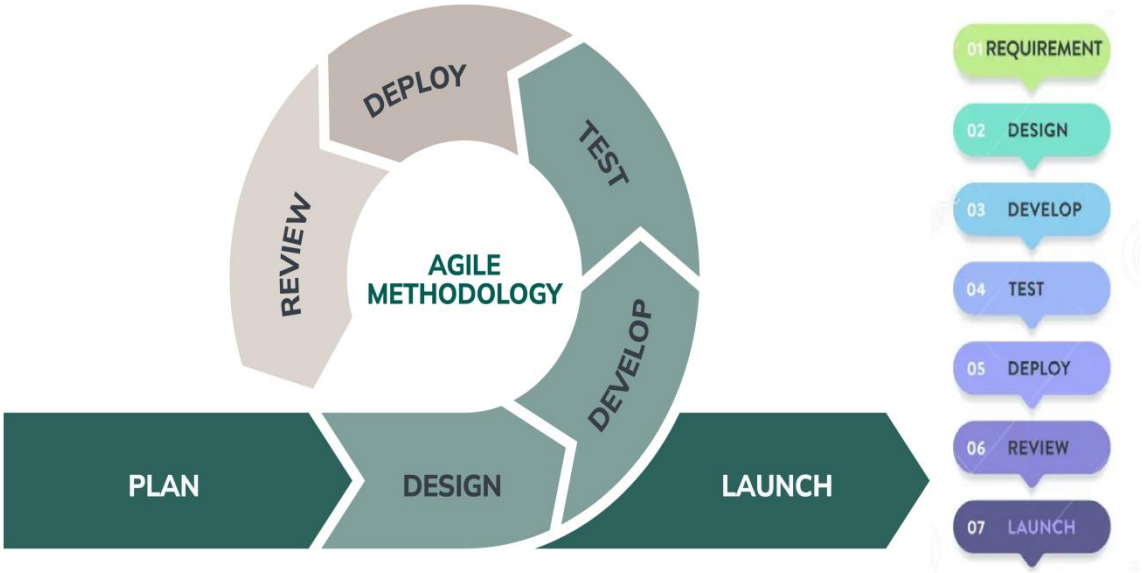


Figure-1: Agile Model

1.6.1 Agile Model Justification

1. Flexible and adapts to changing requirements
2. Encourages frequent customer collaboration and feedback
3. Delivers working software quickly in short iterations
4. Identifies and reduces risks early
5. Improves product quality through continuous testing
6. Empowers teams to be productive and self-organized
7. Provides transparency and progress visibility

1.7 Expected Outcome / Conclusion

1. Users will be able to easily add, edit, and delete their daily tasks through a simple interface.

2. The system will send timely reminders so users don't miss important tasks.

3. All tasks will be safely saved, ensuring no data loss after closing or restarting the app.

4. Recurring tasks can be set up to repeat daily, making routine management easier.

5. Being a web app, users can access their tasks from any device with an internet connection.

6. The project will provide a good foundation for adding more features in the future, like user accounts or better notifications.

7. Overall, it will help users manage their time better and stay organized.

8. This project will also help us improve our programming, database, and web development skills.

Chapter -2: Stakeholder Interview Preparation & Questionnaire Design

1. 1 Admin Feedback

1. What is your role as an admin?
(Short answer)_____

2. How often do you check the task logs and reminders?

- Daily
- Weekly
- Only when issues arise
- Rarely/Never

3. Is it easy to manage tasks created by users in the system?
Multiple choice

- Very Easy
- Somewhat Easy
- Neutral
- Somewhat Difficult
- Very Difficult

4. Do you find the scheduler system (reminders for due tasks) reliable?

- Not Good
- Good
- Very Good
- Excellent

***5.* Which features would you like to see added in the Admin dashboard?**
(Checkboxes)

- ■ Task history log
- ■ Email notifications
- ■ Role-based access
- ■ Bulk task editing
- ■ Analytics dashboard

- ■ Others (add below)

6. Any additional suggestions or comments?
(Paragraph)_____

2.2 User Feedback

1. How often do you use the Smart Task Manager?(multple choice)

- ● Daily
- ● A few times a week
- ● Occasionally
- ● Rarely
- ● First time

2. How easy was it to sign up and log in?(Multiple choice grid)

	Very Easy	**Easy**	**Neutral**	**Difficult**	**Very Difficult**
Sign Up	○	○	○	○	○
Login	○	○	○	○	○

3. Which features do you use the most?(Checkboxes)

- ■ Adding /Editing /Marking tasks
- ■ Giving Feedback
- ■ Built in schedules
- ■ Viewing all tasks
- ■ Logging out
- ■ Others (add below)

4. How satisfied are you with the task management features?

- ● ★
- ● ★ ★
- ● ★ ★ ★
- ● ★ ★ ★ ★

- ★★★★★

5. Did you face any issues while using the application?

- Yes
- No

If yes, please describe*:* (Short answer)_____

7. What features would you like to see in the future?
(Paragraph)_____

2.3 Developers Feedback

1. What part of the system are you primarily responsible for?
(Short answer)_____

2. How clear and well-defined were the project requirements?(Multiple choice)

- Very clear
- Somewhat clear
- Neutral
- Unclear
- Very unclear

3.How would you rate the overall system architecture and code structure?

(Multiple choice)

- Excellent
- Good
- Fair
- Poor
- Very poor

4. What were the most challenging technical aspects during development?
(Paragraph)_____

5. How effectively does the current tech stack support system functionality?

(Multiple choice)

- Fully supports
- Mostly supports
- Somewhat supports
- Poorly supports
- Not suitable at all

6. Did you face any integration or deployment issues? If yes, explain briefly.
(Paragraph)_____

7. How would you rate the system's performance (e.g., load speed, responsiveness)?

- ★
- ★★
- ★★★
- ★★★★
- ★★★★★

8. Are there any known bugs or limitations still present in the system?

- Yes
- No

If yes, describe: (Short answer):_____

9. What improvements would you suggest for future versions?
(Paragraph)_____

10. Do you feel the current codebase is maintainable and scalable? Why or why not?
(Paragraph)_____

Chapter-3: Stakeholders and System Features

3.1 Stakeholders

End Users: Individuals or professionals needing organized task tracking.

Administrators: Manage user accounts and monitor system logs.

Developers: Maintain and improve the platform over time.

3.2 System Features (Table-1)

Feature	Description
Login/Signup System	Secure authentication using hashed passwords (werkzeug.security).
User Dashboard	Responsive HTML5 dashboard to add, edit, and view tasks.
Task Categorization	Tasks classified into Pending, Missed, or Completed via dropdown.
Recurring Task Support	Checkbox-based toggle for marking tasks as recurring.
Reminder Engine	Background scheduler (APScheduler) runs every minute to check for due tasks and mark them.
Data Storage	MySQL-backed storage for user and task records.
Responsive UI	Styled interface with video background, blur overlays, and intuitive layout.
Session Management	Uses Flask sessions to track authenticated users.
Logout Functionality	Clean session termination on logout.

Chapter- 4: Feasibility Study

A feasibility study was conducted to evaluate the viability of the proposed Smart Task Manager system. This analysis assesses whether the project is technically, economically, and operationally achievable. The findings confirm that the project is not only possible but also presents a strategic advantage, offering a clear path to more efficient task management and user productivity. The results of this study are detailed below.

4.1 Technical Feasibility

4.2 The project is highly feasible from a technical perspective. The selected technologies are modern, well-supported, and widely used in web application development.

- **Frontend:** The user interface is built using HTML, CSS, and JavaScript, which are standard technologies for creating responsive, user-friendly, and interactive web pages. This ensures compatibility across all major browsers and device types, providing a seamless experience for users managing tasks.

- **Backend:** The backend is developed using Python with the Flask framework, a lightweight and flexible choice well-suited for RESTful API design and session management. This architecture supports scalability and integration with frontend components efficiently.

- Database: The system employs MySQL as the database management system. MySQL is a robust, secure, and widely adopted relational database with excellent Python integration via libraries such as flask_mysqldb. It supports the complex queries and data integrity requirements of task storage, user management, and status tracking.

4.2 Economic Feasibility

The economic feasibility of the Smart Task Manager is strong, with a focus on long-term efficiency gains and cost savings.

- **Development Cost:** Utilizing open-source frameworks and languages like Python and Flask significantly reduces licensing costs. The initial development investment includes setup and coding, estimated within a moderate budget suitable for small teams or academic projects.

- **Long-Term Savings:** By automating key task management processes—such as scheduling, reminders, status updates, and recurring task tracking—the system minimizes manual workload and human errors. This leads to improved staff productivity and reduced administrative overhead, which offsets the initial development costs over time.

4.3 Operational Feasibility

The system is operationally feasible, designed with usability and user acceptance in mind.

- Tranning Requirements: The intuitive and clean user interface requires minimal training. Users can quickly learn to add, edit, and track tasks. Supporting materials such as a short user guide or onboarding tutorial will ensure rapid adoption.

- User Acceptance: Staff and end-users are likely to embrace the system as it simplifies daily task management. Features like task status categorization (pending, missed, completed), recurring task options, and real-time reminders enhance productivity and reduce missed deadlines. The mobile-friendly design increases accessibility, supporting flexible work environments.

4.4 SWOT (Strengths, Weaknesses, Opportunities, and Threats) Analysis

Strengths:

1. Functional user-based login and session management
2. Live task reminders with a background job.
3. Minimalist, responsive, and modern interface.
4. Robust backend integration with MySQL and Flask.
5. Easily scalable for future features (like multi-user collaboration).

Weaknesses:

1. Lacks email/SMS notification support.
2. No admin dashboard yet for overseeing all users.
3. No current analytics/dashboard view.
4. Reminder system only logs to console (no visual alerts).
5. No user profile customization or role-based UI differences.

Opportunities:

1. Expand to include team collaboration features.
2. Integrate with calendar APIs (e.g., Google Calendar).
3. Add a mobile app frontend (using Flutter or React Native).
4. Introduce export/import features (PDF, Excel).
5. Implement progressive web app (PWA) support for offline access.

Threats:

1. Data loss if no regular database backup is configured.
2. Potential performance limitations under high concurrent usage.
3. Security concerns if HTTPS and CSRF protections aren't enforced.
4. Dependency on Flask dev server .
5. External changes in MySQL or Python packages could cause breakage.

4.5 System Request

Smart Task:

Project Sponsor: Head of Development Team

Business Need:

Managing tasks manually is inefficient. Users need an automated system to track tasks with reminders and submit feedback easily.

Business Requirements:

- User authentication (login/signup).
- Create, edit, delete, and view tasks with status and recurrence options.
- Automated reminders for pending tasks.
- Feedback submission and display interface.
- Modern UI with video background and overlays.

Business Value:

- Save user time by automating task management.
- Improve productivity and user satisfaction.
- Enable better feedback collection.

Special Issues or Constraints:

- Must comply with data privacy policies.

- Reminders should trigger on time.
- Deployment target by December 2025.

Chapter-5: Requirement Analysis

The requirements for Smart Task, a task management web platform, were gathered through open-ended stakeholder interviews and structured questionnaires. Key stakeholders included Admin, User, and System Administrators or Developers. The objective was to identify current pain points in task delegation, collaboration, and progress tracking in real-time environments.

5.1 Functional Requirements

These define what the system must do to manage and track tasks effectively:

1) Task Management:

- Add, edit, delete tasks
- Set priorities and due dates
- Assign tasks to users
- Create subtasks

2) User Roles and Permissions:

- Register/login users
- Differentiate between admin, manager, and staff roles

3) Collaboration Tools:

- Real-time task updates
- Comments on tasks
- File attachments

4) Progress Tracking:

- Visual task status (To-Do, In Progress, Done)
- Personal and team dashboards
- Notification System:
- Email or in-app reminders for deadlines
- Alerts for task assignment or status change

5) Analytics & Reports:

- Generate weekly productivity reports
- Filter and export task data

5.2 Non-Functional Requirements

These describe how the system should behave:

1) Usability:

- Clean and intuitive user interface
- Minimal training required
- Performance:
- Fast loading pages
- Capable of handling up to 100 users concurrently

2) Security:

- Password hashing, session management
- Role-based access control
- Reliability:
- 99.9% system uptime
- Robust error handling

3) Scalability:

- Easy to extend for future features like calendar sync or mobile app

5.3 User Interface Requirements

Key screens and their layout:

1) Login Page:

- Simple form with email & password
- Forgot password link

2) Dashboard:

- Summary of pending, completed, and overdue tasks
- Recent activity feed

3) Task View Page:

- List of tasks with filters (date, priority, status)
- Status update buttons (drag-drop or select menu)

4) Admin Panel:

- Add/edit users
- Role assignment and permission configuration

5) Reports Page:

- Graphs and charts showing task completion rate
- Export to PDF/CSV

5.4 Assumptions & Constraints

Assumptions are the conditions we believe to be true for the project to work smoothly, like having trained staff and stable internet. Constraints are the fixed boundaries like budget, time, and hardware that limit how the project is planned and executed.

1) Assumptions:

- Users have stable internet access
- Staff will attend one training session
- Hosting is available for deployment
- End-users will provide constructive feedback

2) Constraints:

- The project must finish within the given timeline.
- The project must finish within budget.
- Must support at least 50 concurrent users on launch
- UI must be responsive across devices

5.5 Gantt Chart (Table-2)

Phase / Task	Week 1-2	Week 2-3	Week 3-4	Week 4-5	Week 5-6	Week 6-7	Week 8-9	Week 10-11
Project Planning & Requirements	■							
Feasibility Study		■						
UI/UX Design			■	■				
Backend Development (Flask/API)					■	■		
Frontend Integration (HTML/CSS/ JS)				■	■			
Database Setup (MySQL)						■		
Task Management Module						■	■	
Testing (Unit & Functional)							■	
Deployment & Hosting								■

5.6 Budget (Table-3)

Item	Description	Estimated Cost (BDT)
Development	Basic backend & frontend coding	10,000
Design	Simple UI/UX	2,000
Testing	Basic manual testing	1,500
Hosting & Domain	Basic hosting + domain (annual)	2,000
Database Setup	Basic setup & backups	1,000
Maintenance	Bug fixes & small updates (1 month)	2,000
Contingency	Unexpected costs/Reserv cost	1,500
Total Estimated Cost		**20,000 BDT**

Chapter-6 :System Analysis & Design

6.1 Data Flow Diagrams (DFDs) for the Smart Task System

The Smart Task System DFDs illustrate how users and admins interact with the platform to manage tasks, reminders, and feedback. Data flows through processes like authentication, dashboards, and reporting, while being stored in dedicated databases. Each level of the DFD moves from a simple overview to detailed operations, giving a structured picture of how the system handles data efficiently.

6.1.1DFD Level - 0: Context Diagram for the Smart Task

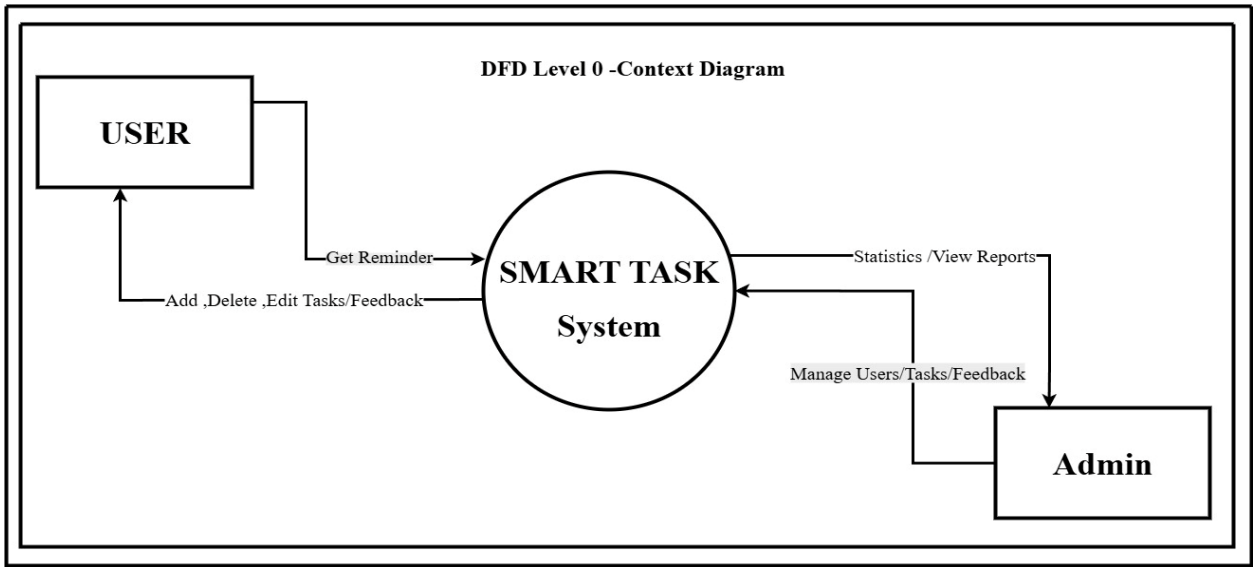


Figure-2: Level-0 - Context DFD

1. **Purpose:**
   - To show the system as a single entity interacting with external entities.
   - Provides a high-level overview without showing internal processes.
   - Helps stakeholders quickly understand what the system does and how it exchanges data.
2. **Process:**
   - System Management – represents the entire Smart Task app as a single process.
3. **External Entities:**
   - User – interacts with the system by providing input (login, tasks, feedback) and receiving output (task lists, reminders, feedback confirmation).
   - Admin – interacts with the system by managing statistics,users, and their feedback.
4. **Data Flow :** Data moves between users, admin, and the system.

6.1.2 DFD Level - 1: Main Processes of the Smart Task

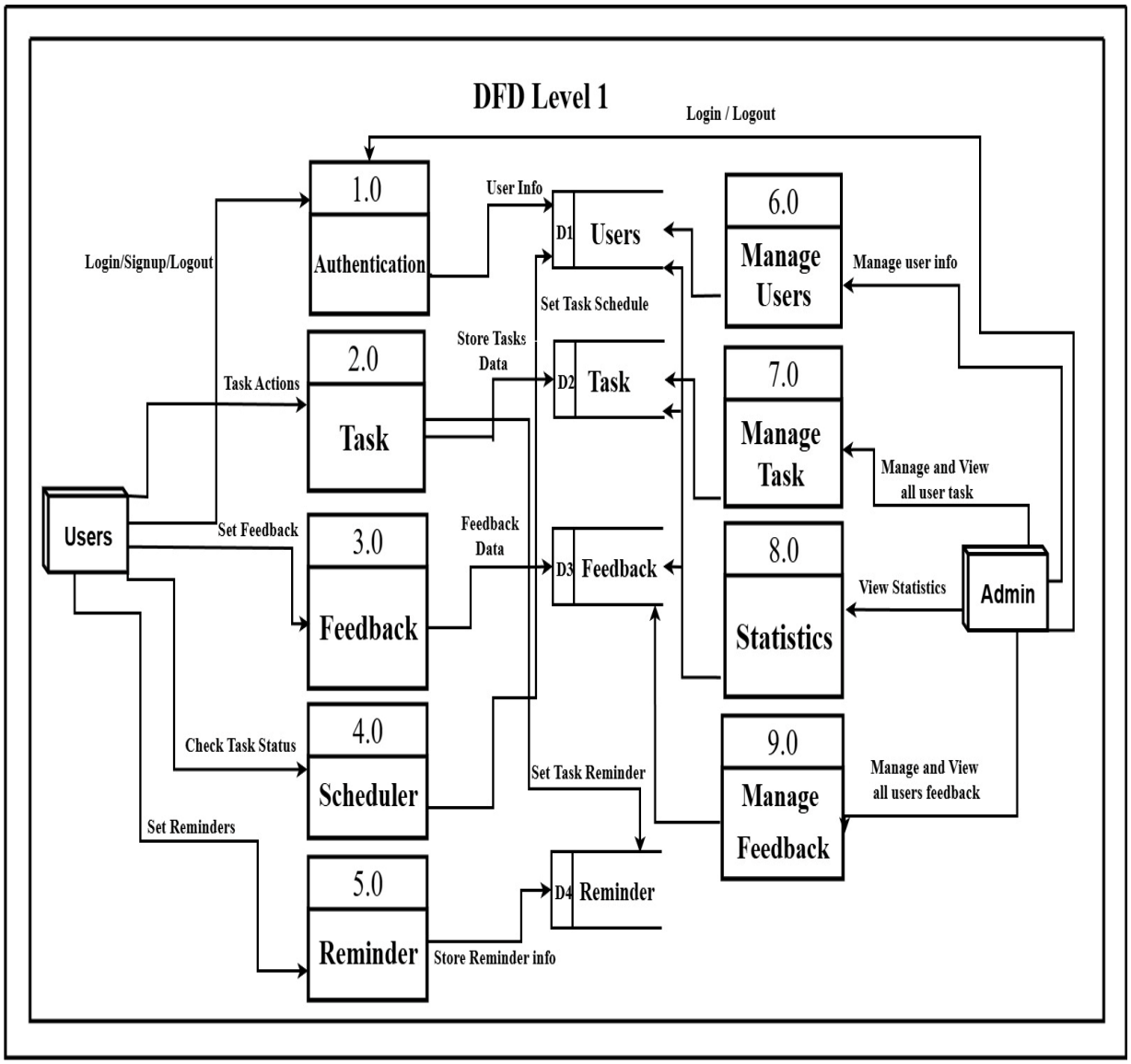


Figure-3: DFD Level - 1: Main Processes

1. Purpose:

- To illustrate the main processes of the Smart Task system.
- To show how users and admins interact with the system.
- To depict data flow between processes and databases.
- To identify the datastores and the type of data they hold.
- To help understand system structure, workflow, and interactions for development

Components:

- External Entities: User, Admin
- Processes: Login & Authentication (1.0), Task Management (2.0), Feedback Handling (3.0), Admin Dashboard (4.0)
- Data Stores: D1 (Users DB), D2 (Tasks DB), D3 (Reminder DB), D4 (Feedback DB)

2. Tasks and Data Flow:

- User → Login & Authentication (1.0): Sends login credentials.
- Login & Authentication (1.0) → D1 (Users DB): Verifies credentials.
- User → Task Management (2.0): Submits task details.
- Task Management (2.0) → D2 (Tasks DB): Saves or updates tasks.
- Task Management (2.0) → D3 (Reminder DB): Schedules reminders.
- User → Feedback Handling (3.0): Submits feedback.
- Feedback Handling (3.0) → D4 (Feedback DB): Stores feedback.
- Admin → Admin Dashboard (4.0): Requests reports/statistics.
- Admin Dashboard (4.0) → D1, D2, D3, D4: Retrieves data for reports and insights.

6.1.3 DFD Level - 2: Decomposing a Process for the Smart Task

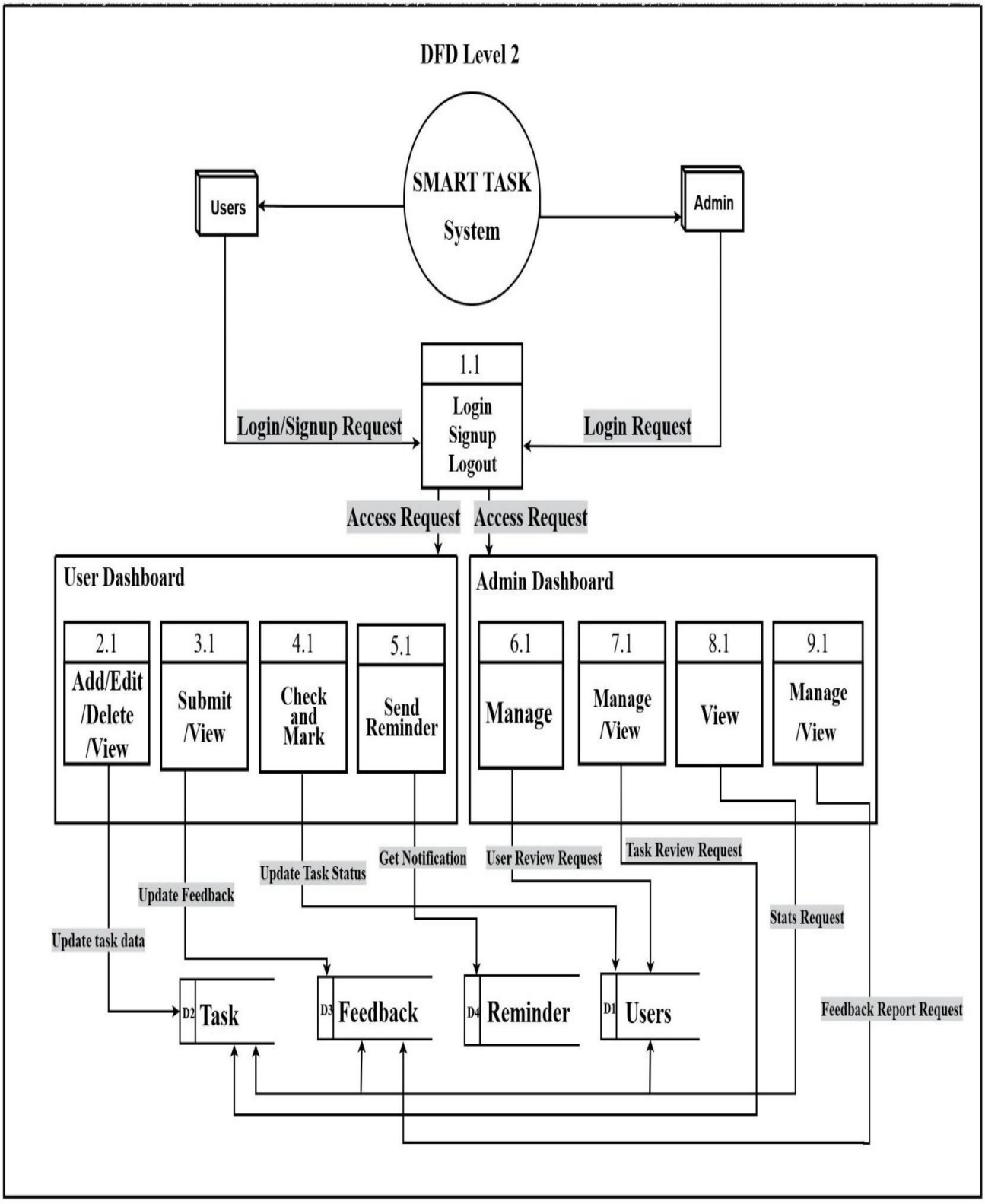


Figure-4: DFD Level - 2: Decomposed the Process

1. Purpose:

- To show the detailed internal processes of the system, including login/authentication, task management, feedback handling, and admin dashboard.
- To illustrate how users and admins interact with specific sub-processes.
- To depict data flow between processes and databases (Users, Tasks, Reminder, Feedback).
- To highlight authentication as the gateway for secure access to system functionalities.
- To help stakeholders understand task operations, feedback flow, and admin monitoring at a detailed level.

2. Components:

- External Entities: User, Admin

Processes:

- Login / Authentication (1.x)
- Task Management (2.x) – Add, Delete, View, Edit Tasks
- Feedback Handling (3.x) – Submit Feedback, View Feedback
- Admin Dashboard (4.x) – Manage Tasks, View Feedbacks, View Statistics

3. Data Stores:

- Users DB (D1)
- Tasks DB (D2)
- Reminder DB (D3)
- Feedback DB (D4)

4. Tasks and Data Flow:

- User → Login / Authentication (1.x): Sends login credentials.
- Login / Authentication → D1 (Users DB): Verifies credentials.
- Login / Authentication → Task Management (2.x), Feedback Handling (3.x), Admin Dashboard (4.x): Grants access if authenticated.
- User → Task Management (2.x): Adds, deletes, views, or edits tasks.
- Task Management → D2 (Tasks DB): Saves, updates, or deletes tasks.

- Task Management → D3 (Reminder DB): Stores reminders for tasks.
- User → Feedback Handling (3.x): Submits feedback or views feedback.
- Feedback Handling → D4 (Feedback DB): Stores or retrieves feedback.
- Admin → Admin Dashboard (4.x): Manages tasks, views all feedbacks, views statistics.

6.2 Difference between Level 0 , Level 1 and Level 2 DFD (Table-4)

Aspect	Level 0 DFD	Level 1 DFD	Level 2 DFD
Purpose	Shows the entire system as one single process	Breaks down the system into main processes	Further decomposes main processes into detailed sub-processes
Focus	External entities and overall data flow	High-level processes and data stores	Detailed functions and specific data flow
Processes Shown	1 (whole system as a black box)	Few (Login, Task Mgmt, Feedback, Admin Dashboard)	Many (Add/Edit/Delete/View Task, Submit/View Feedback, Manage Tasks, View Stats, etc.)
Data Stores	Not shown	Introduced (Users DB, Tasks DB, Reminder DB, Feedback DB)	Fully detailed with exact interactions
Entities	User, Admin	User, Admin	User, Admin
Detail Level	Very high-level (abstract)	Moderate (overview of system modules)	Very detailed (step-by-step operations)

6.3 Usecase Diagram

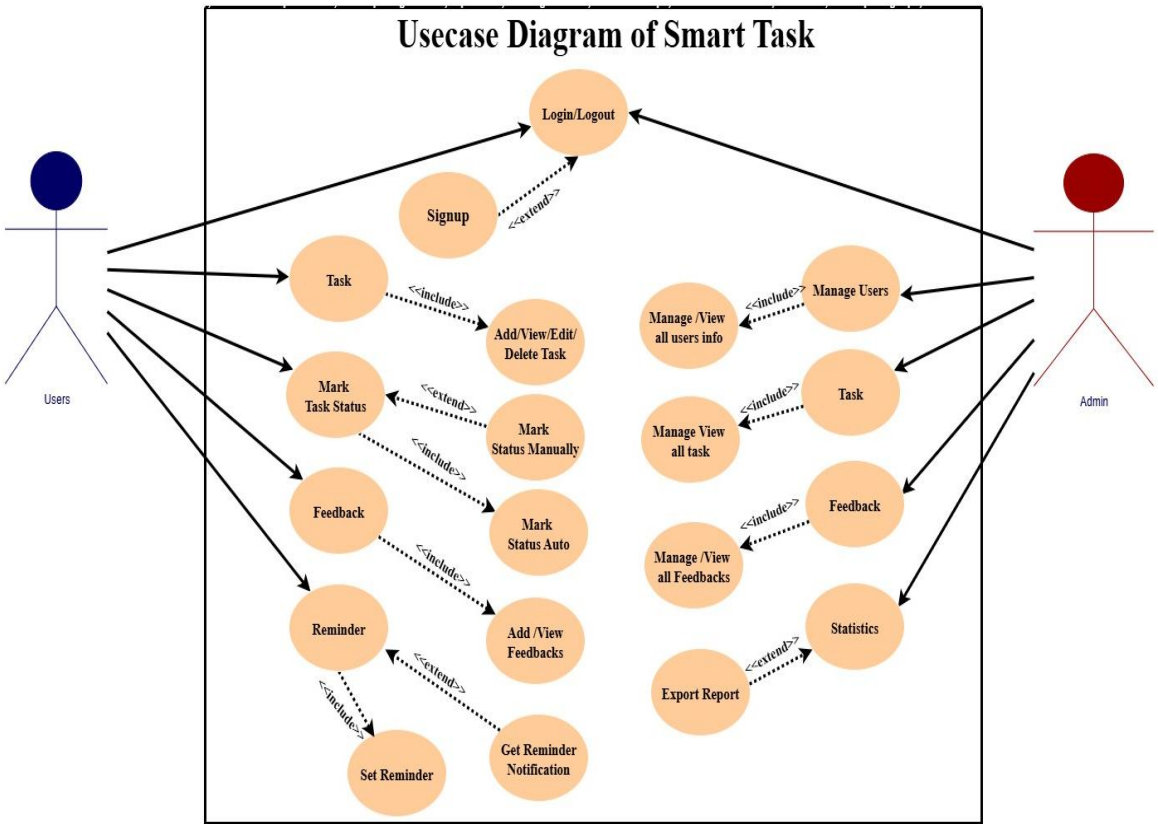


Figure-5: Use Case Diagram of Smart Task

Actors:

1. **Users (left side):**Represent normal system users who create and manage tasks.
2. **Admin (right side):**Represents administrators who manage users, tasks, and system-wide data.

Explanation:

1. This use case diagram describes how the Smart Task System works:
2. Users can register, log in, manage personal tasks, set reminders, mark task status, and provide feedback.
3. Admins can log in, manage users, oversee all tasks and feedback, generate reports, and view system statistics.

In short: The diagram clearly separates user-level functions (personal task management) and admin-level functions (system management and reporting).

6.4 Entity Relationship Diagram (ER Diagram)

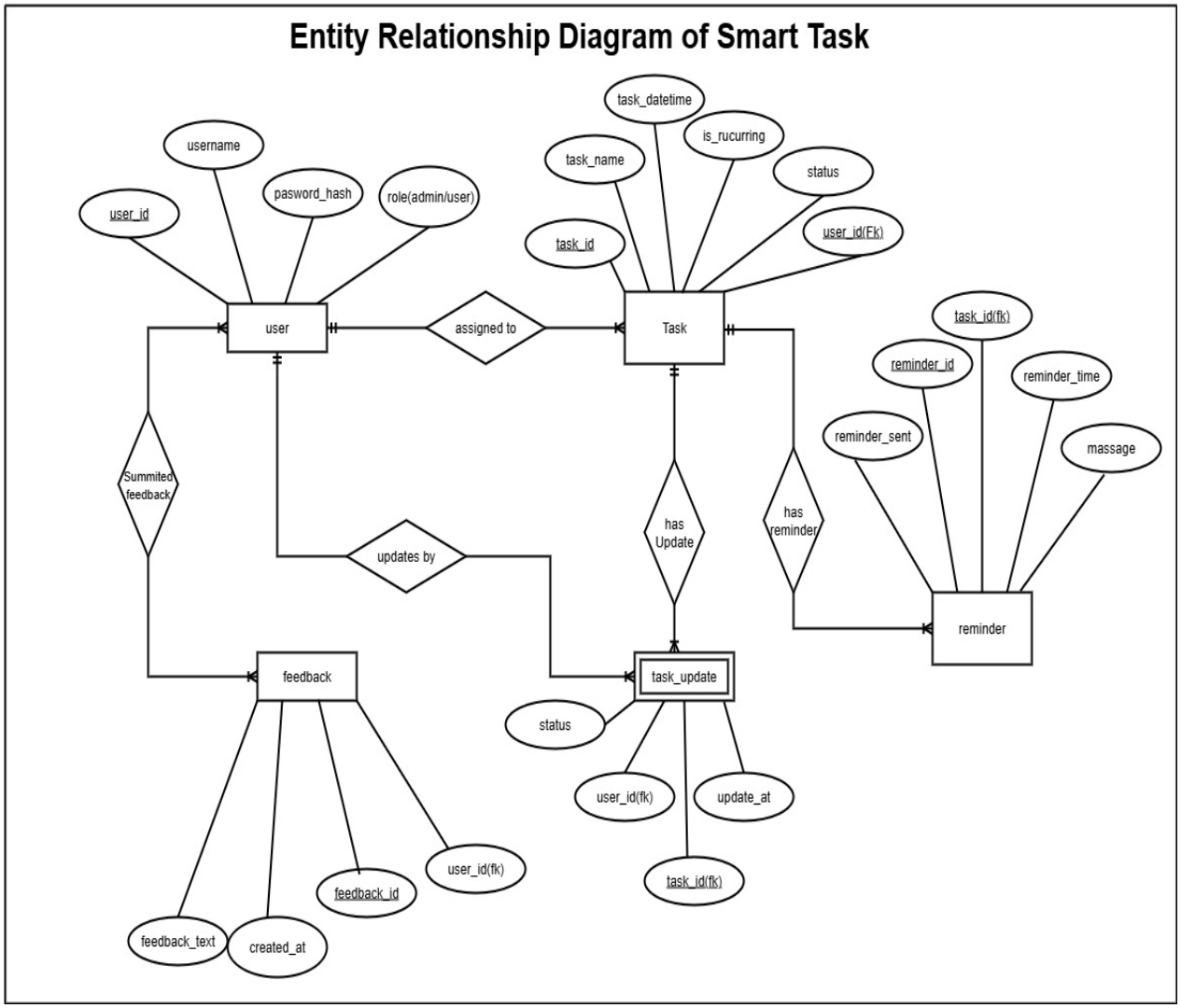


Figure-6: Entity Relation Diagram of Smart Task

Overall Explanation:

The ER diagram models a Smart Task Management System where:

1. Users manage tasks.
2. Each task can have reminders to alert users.
3. Task progress is tracked through updates.
4. Users can submit feedback about tasks or the system.

In short: This system keeps track of users, tasks, reminders, task updates, and feedback to help manage tasks efficiently.

6.5 Sequence diagram

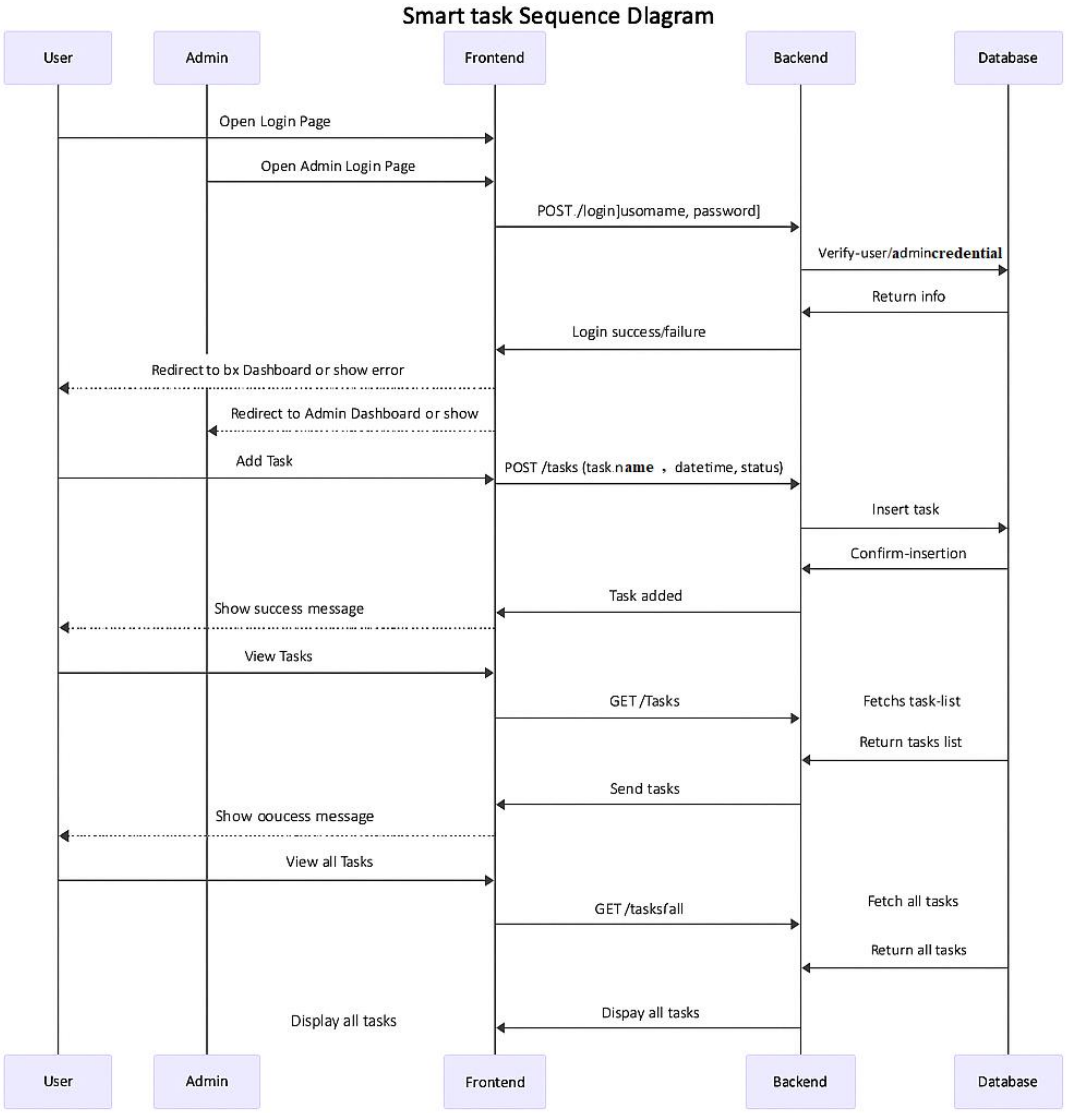


Figure-7: Sequence diagram of Smart Task

The SmartTask Sequence Diagram shows how users and admins interact with the system through the frontend, backend, and database.

Login: User/Admin logs in → frontend sends credentials → backend verifies → success or error returned.

Add Task (User): User adds a task → backend stores it in the database → confirmation sent back.

View Tasks (Admin/user): Admin/user requests tasks → backend fetches users tasks → frontend displays them.

View All (Admin): Admin requests all tasks/feedback → backend retrieves and returns them from the database.

6.6 Activity Diagram

An activity diagram of the Smart Task is a visual flow showing how users/admin interact with the system, from logging in, creating tasks, updating status, to viewing stats or feedback, including decisions and workflow steps.

6.6.1 Activity Diagram of Users

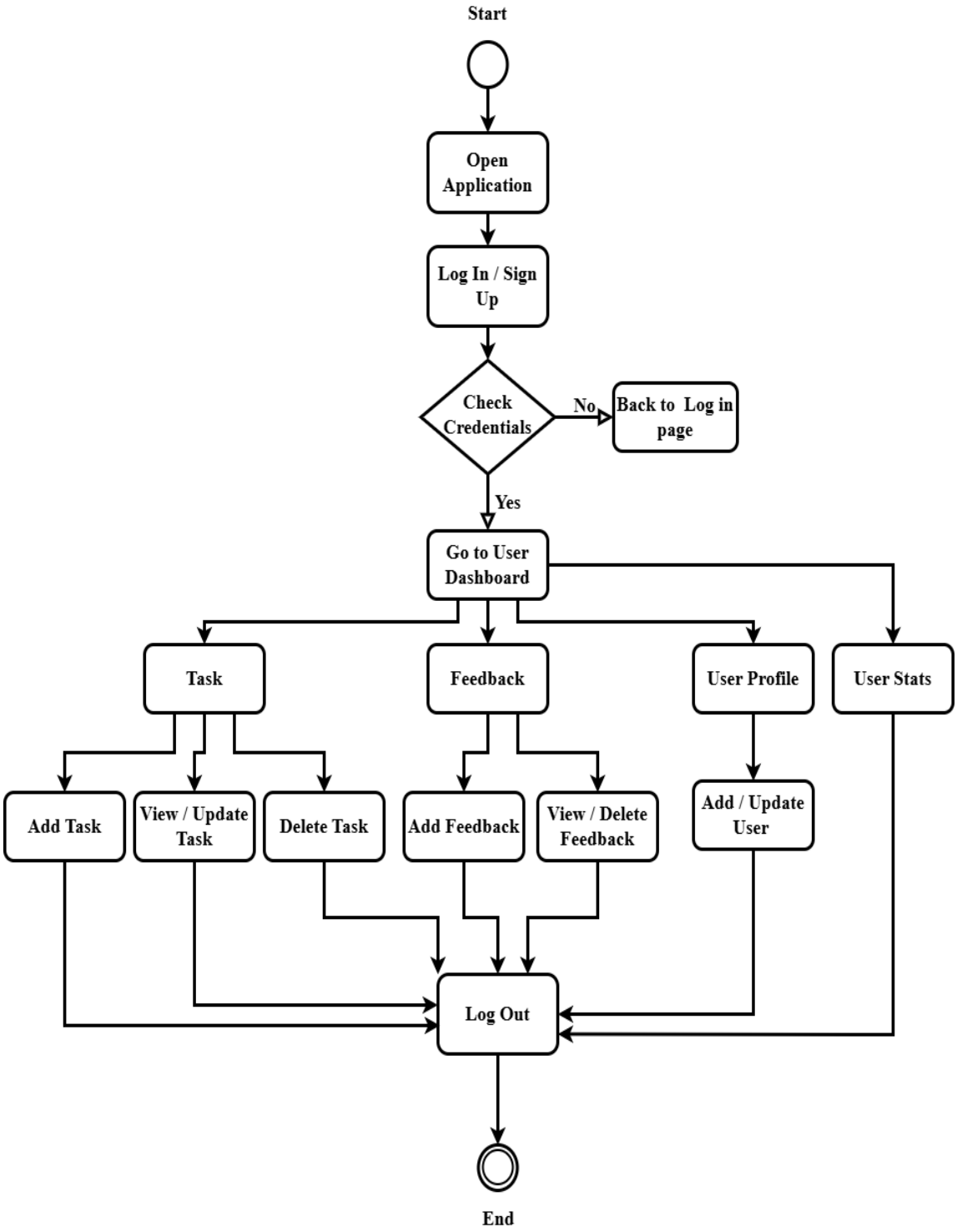


Figure-8: Activity flow of a users

6.6.2 Activity Diagram of Admin

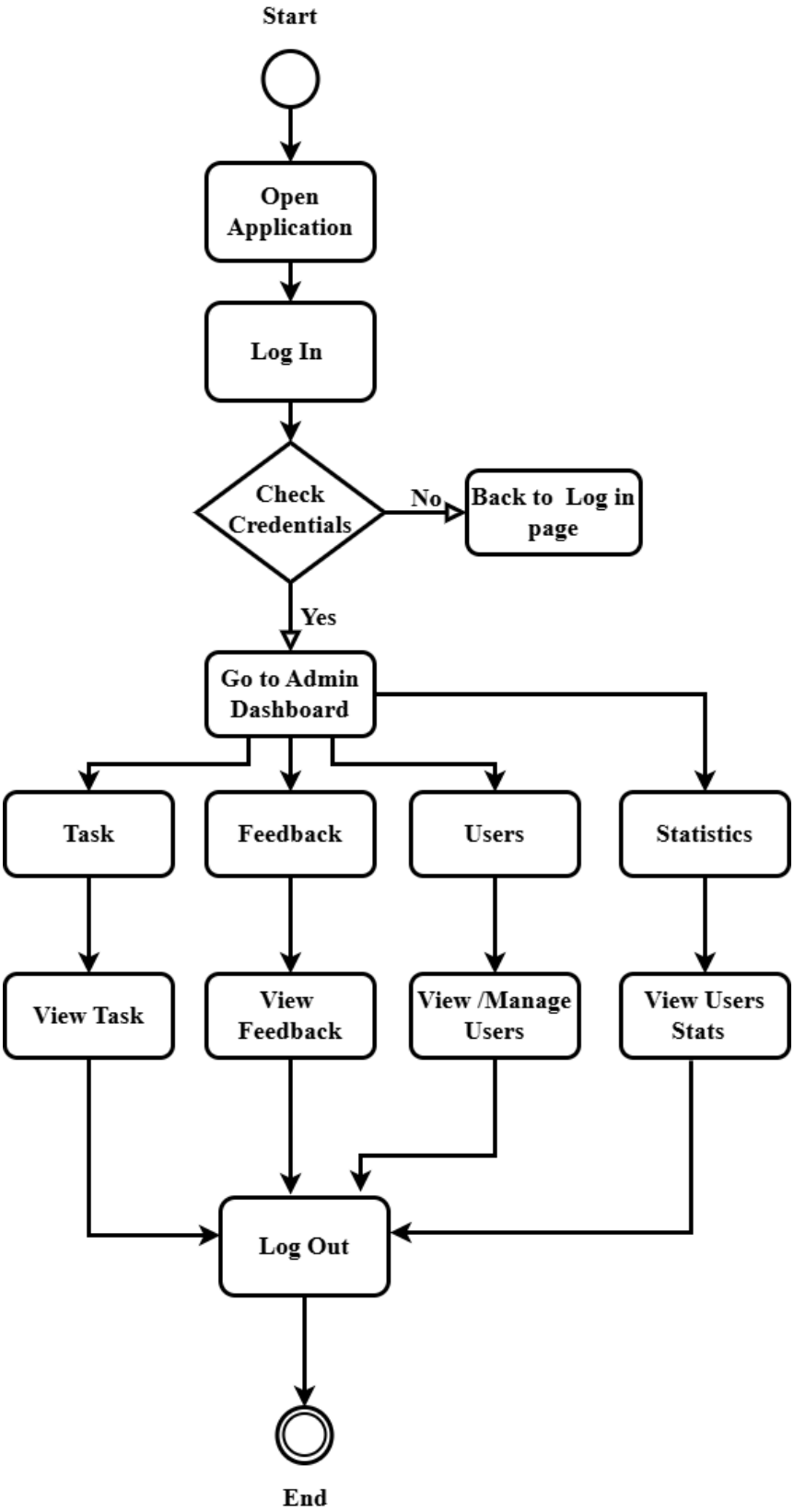
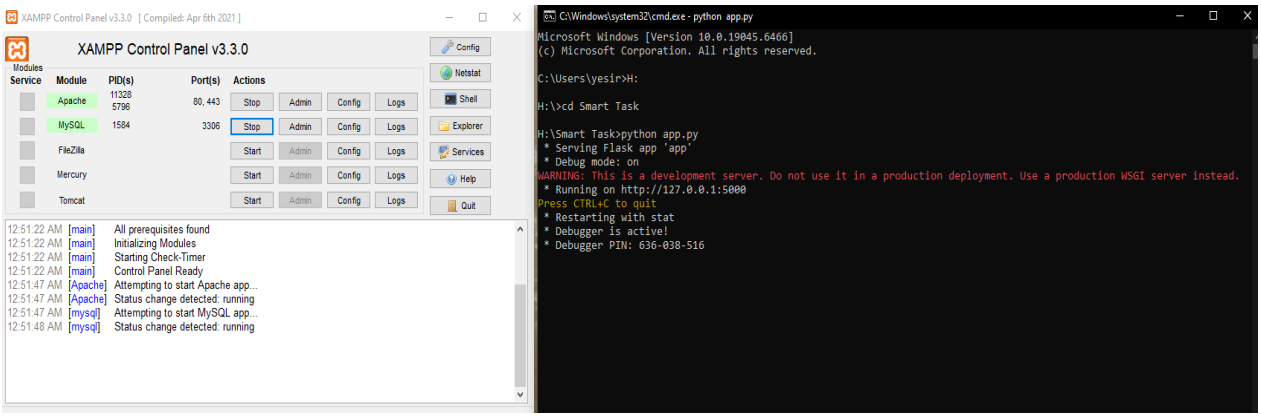


Figure-9: Activity flow of a Admin

6.7 Visual Process of Smart Task

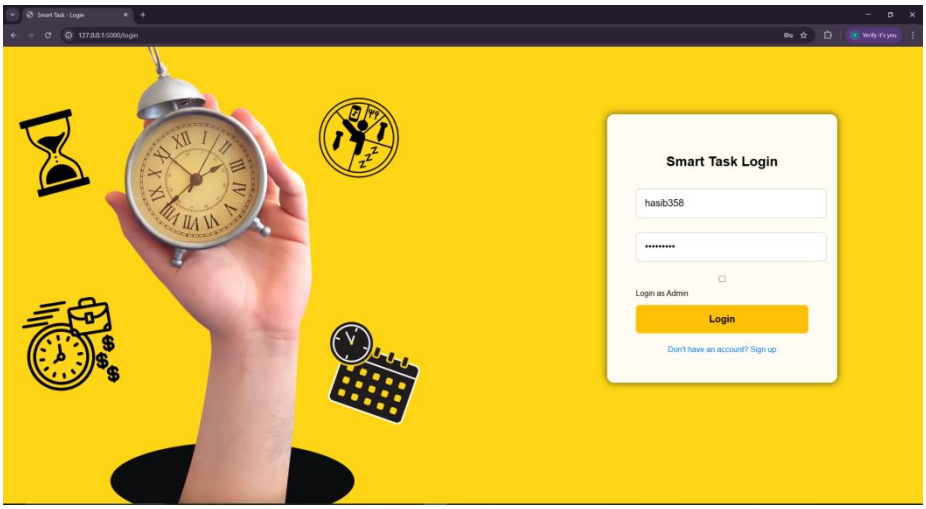
1. First we will start Xampp to connect localhost then we will use cmd promt to connect backend with frontend .



Snippet-1: Connecting Backend with Frontend

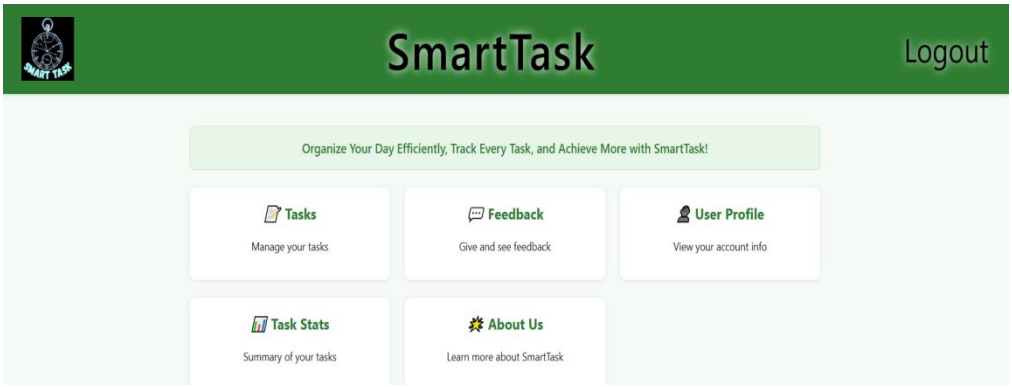
Here the message shows the backend successfully connected to frontend

2. Next we will enter login page for users/admin both



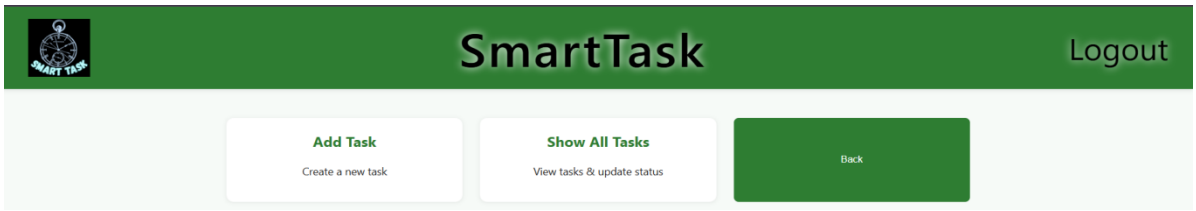
Snippet-2: Entering login/signup window

3. If we enter user credentials then we will enter user dashboard



Snippet-3: Window of User Dashboard

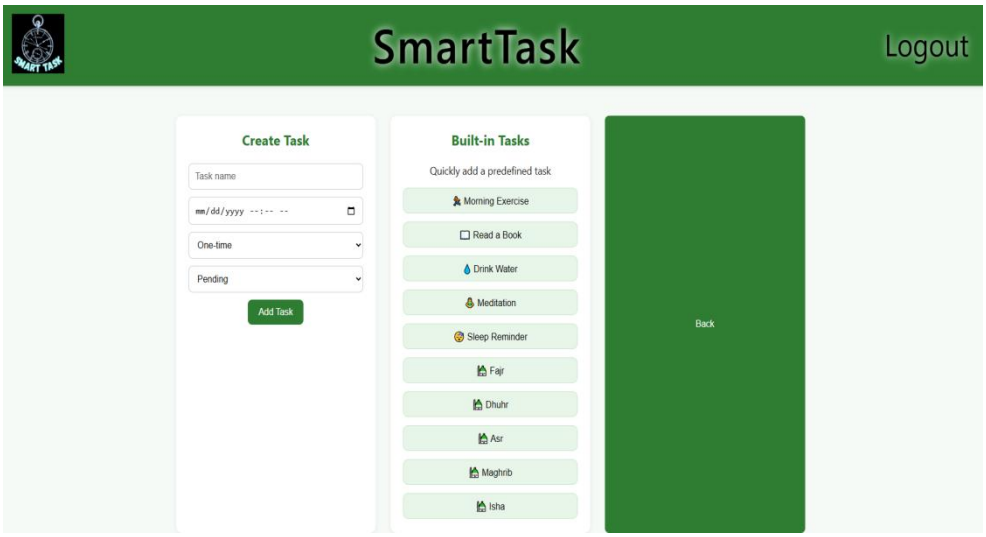
3.1 Here we can Add ,view and update task-



Snippet-4: Window of Tasks

Then,

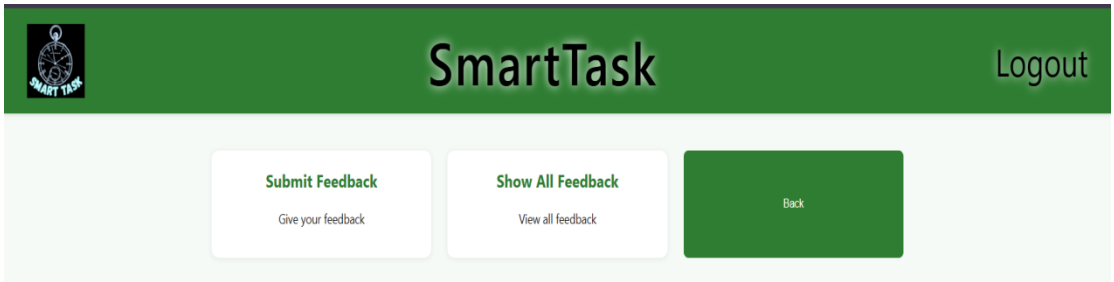
1. Add task→



2. View / Update Task (Show all task) →

Snippet-5: Windows of add/view/update tasks

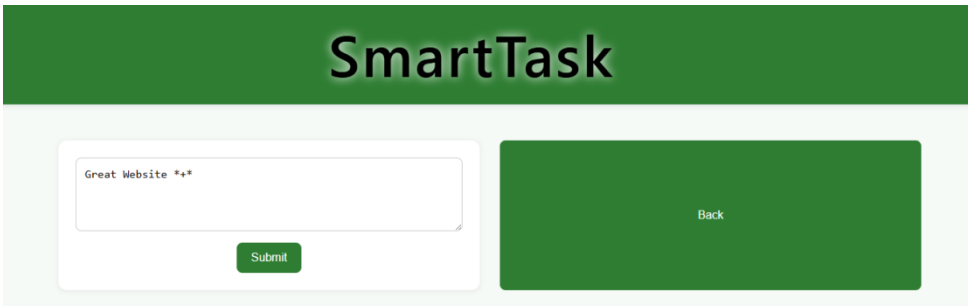
3.2 Now we want to add feedback and view/update feedback-



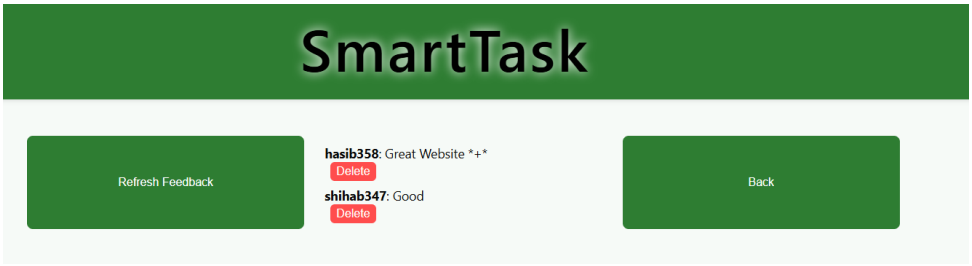
Snippet-6: Window of Feedback

Then,

1. Add feedback →



2. View / Update feedback (Show all task) →

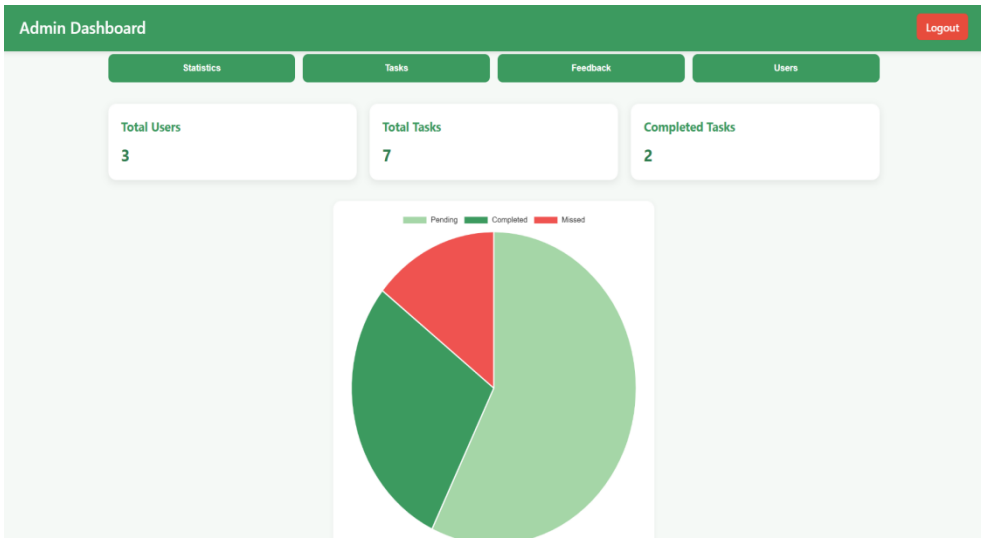


Snippet-7: Windows of add/view/delete feedback

3.2 we can add and update user info -

Snippet-8: Window of User Profile

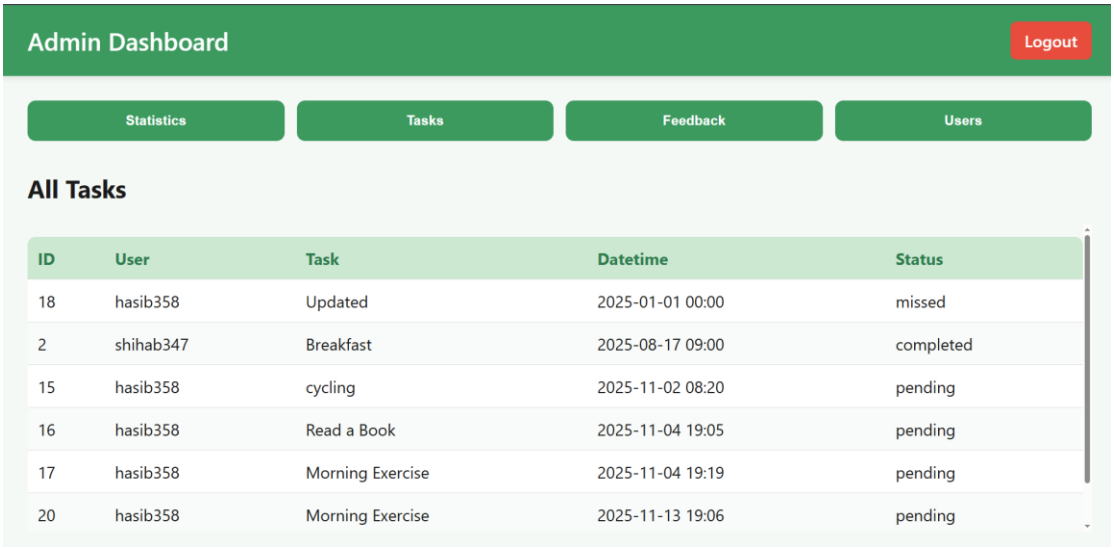
4. If admin enter admin credentials then admin will enter admin dashboard



Snippet-9: Window of Admin Dashboard

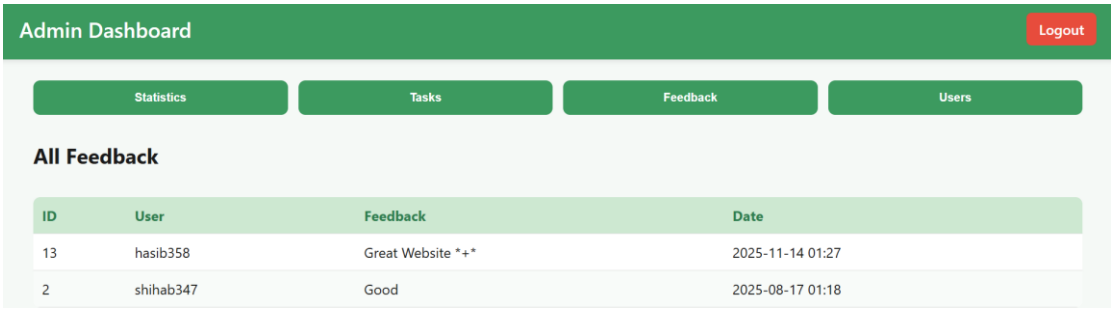
4.1 Admin can View task,feedback and user info of all users -

Task-



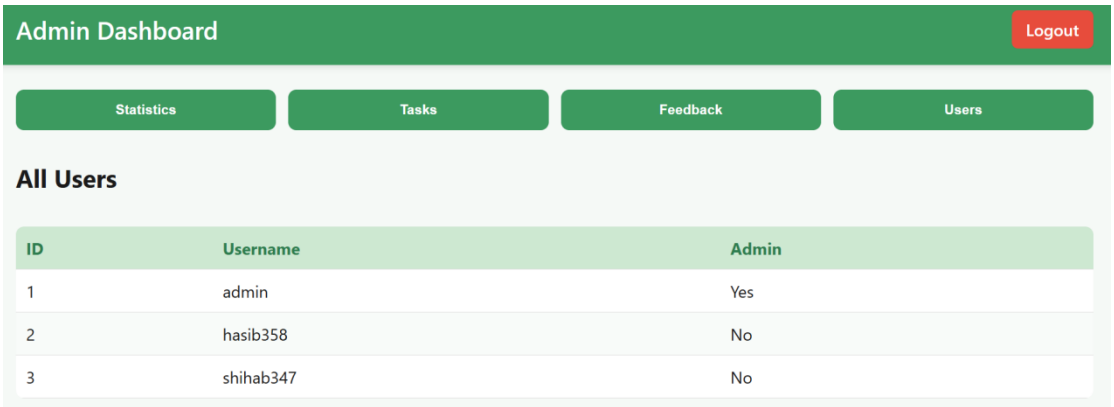
Snippet-10: Window of Admin view task

Feedback-



Snippet-11: Window of Admin view feedback

Users-



Snippet-12: Window of Admin view users

This is the overall backend and frontend visual working process of Smart Task.

Chapter 7: Conclusion

7.1 Conclusion

The Smart Task Manager project successfully achieves its goal of providing a web-based platform for managing daily tasks efficiently. Users can add, edit, delete, and view tasks through a simple and intuitive interface. The system supports recurring tasks and timely reminders, ensuring that users do not miss important deadlines. By integrating a MySQL database, all task information is stored securely, offering persistent access across sessions. Overall, the project improves productivity, organization, and task tracking for users while providing a foundation for future enhancements.

7.2 Limitation

Despite its functionality, the current system has several limitations:

1. The reminder system is limited to email or SMS alerts and does not support in-app notifications
2. The application is accessible only via web browsers; a mobile app version is not yet available.
3. User roles and permissions are basic, with no role-based customization in the interface.
4. The system lacks advanced collaboration features such as task sharing or team management.

7.3 Future Works

Future enhancements can significantly improve the system's capabilities:

1. Implement in-app notifications for better reminders.
2. Introduce mobile app support using technologies like React Native or Flutter.
3. Add team collaboration features, including task sharing, comments, and file attachments.
4. Incorporate analytics and reporting features for better productivity insights.
5. Enable integration with calendar APIs (Google Calendar, Outlook) for advanced scheduling.