Project Report

Student Academic Goal & Progress Tracker

Name: Raviraj Jain **Registration No: 25BAI11023**

Introduction:

Every student dream of improving their academic performance, yet most struggle to stay consistent, maintain goals, and keep track of their progress. We often rely on scattered notes, reminders, or last-minute planning, which eventually leads to stress and inefficiency.

This project was created as a practical attempt to solve that everyday problem.
The *Student Academic Goal & Progress Tracker* is a simple, functional, and thoughtfully designed web application that helps students manage their goals, record marks, follow their study habits, and visually understand how well they are progressing.

Built using **Python (Flask)**, **MySQL**, and **HTML/CSS**, the project demonstrates how basic tools taught in class can be combined to build a complete real-world solution.

Problem Statement:

Students rarely get a structured way to track academic growth.
Common difficulties include:

- No clear system to set and monitor goals
- Inconsistent study routines
- No centralized place to store subject performance
- Lack of visual insights (charts, trends, patterns)
- No motivation or gamification to maintain consistency

The absence of such a system directly affects consistency, planning, and performance.

Hence, there is a strong need for a **simple, reliable, and student-friendly platform** to help learners organize their academic journey.

Functional Requirements:

Major Modules

1. Goal Management

- Students can create short-term and long-term academic goals.
- They can update progress, mark goals as completed, and view ongoing tasks.
- Deadlines help maintain discipline.

2. Subject & Performance Tracking

- Subjects can be added for a semester or exam.
- Marks from tests or assignments can be entered.
- Performance is converted into charts to make the progress crystal clear.
- Helps students identify strong and weak areas.

3. Study Streak & Gamification

- Tracks how regularly the student logs in.
- Displays daily streaks in a calendar view.
- Awards badges like *Consistency King* or *Goal Master* for motivation.

Additional Features

- Secure login system
- Dashboard showing goals + recent marks + streak status
- Task reminders
- Clean user interface

The absence of such a system directly affects consistency, planning, and performance.

Hence, there is a strong need for a **simple, reliable, and student-friendly platform** to help learners organize their academic journey.

Non-Functional Requirements:

1. Usability

The interface is intentionally kept minimal so that students can use it without confusion.

2. Performance

Queries and routes are optimized to load quickly, even with larger data.

3. Security

Passwords are hashed, and input data is validated before storing.

4. Maintainability

The application is built in small modules so future upgrades are easy.

5. Reliability

Session management ensures stable performance across sessions.

6. Scalability

With slight modifications, it can be extended for multiple users or even institutions.

7. Error Handling

User-friendly messages guide students if they enter wrong or incomplete data.

8. Resource Efficiency

Uses lightweight frameworks ensuring smooth functioning on simple systems.

System Architecture:

The system follows a straightforward flow:

Client (Browser)

↓

Frontend (HTML/CSS)

↓

Flask Server (Python)

↓

MySQL Database

↓

Processed Data → Charts/Output

↓

Displayed on Dashboard

This architecture keeps the system simple yet robust.

Process / Workflow

1. Student logs in
2. Dashboard loads with summary
3. User selects a module
4. System processes the request
5. Output (goal list, marks graph, streak count, etc.) is shown

Sequence Overview (Example: Adding Marks)

User → Performance Page → Flask route → Database Insert → Graph Updated → Redirect to Dashboard

Database Design

The database includes these main tables:

- **users**
- **lookup**
- **goals**
- **progress_logs**
- **study_sessions**
- **badges**
- **todo_tasks**
- **motivation_messages**

Each table is connected using foreign keys for structured storage.

Design Decisions & Rationale:

- **Flask** was chosen as it is easy to work with, especially for beginners.
- **MySQL** is suitable for structured, relational academic data.
- **Matplotlib** is used to generate clean and simple graphs without advanced tools.
- The UI is deliberately basic because the objective is clarity and practicality, not heavy design.
- The project architecture uses separate modules to keep the system organized.

The goal was to build a *solid*, *functional* system rather than an overly complicated one.

Implementation Details:

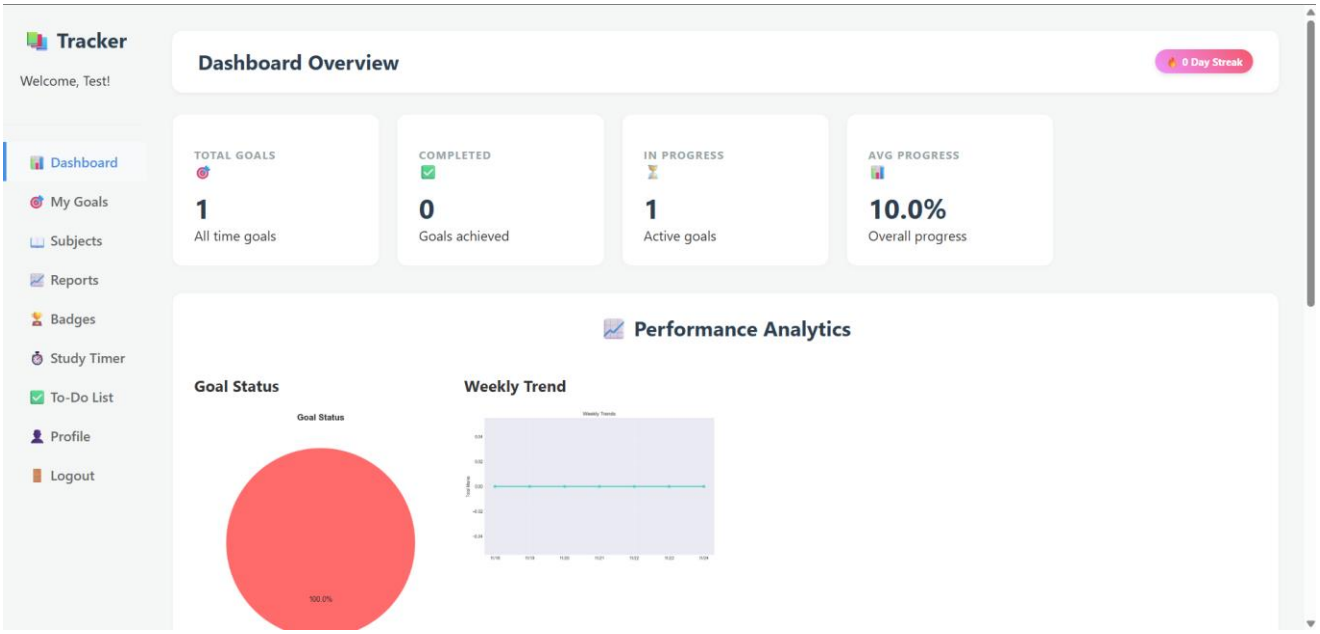
The application is divided into small modules:

- `auth.py` – Login, registration, session handling
- `goals.py` – Adding and managing goals
- `subjects.py` – Subject CRUD operations
- `reports.py` – Performance analytics
- `charts.py` – Graph generation
- `exports.py` – Exporting data
- `db.py` – Database connection
- `templates/` – HTML pages
- `static/css/` – Stylesheets

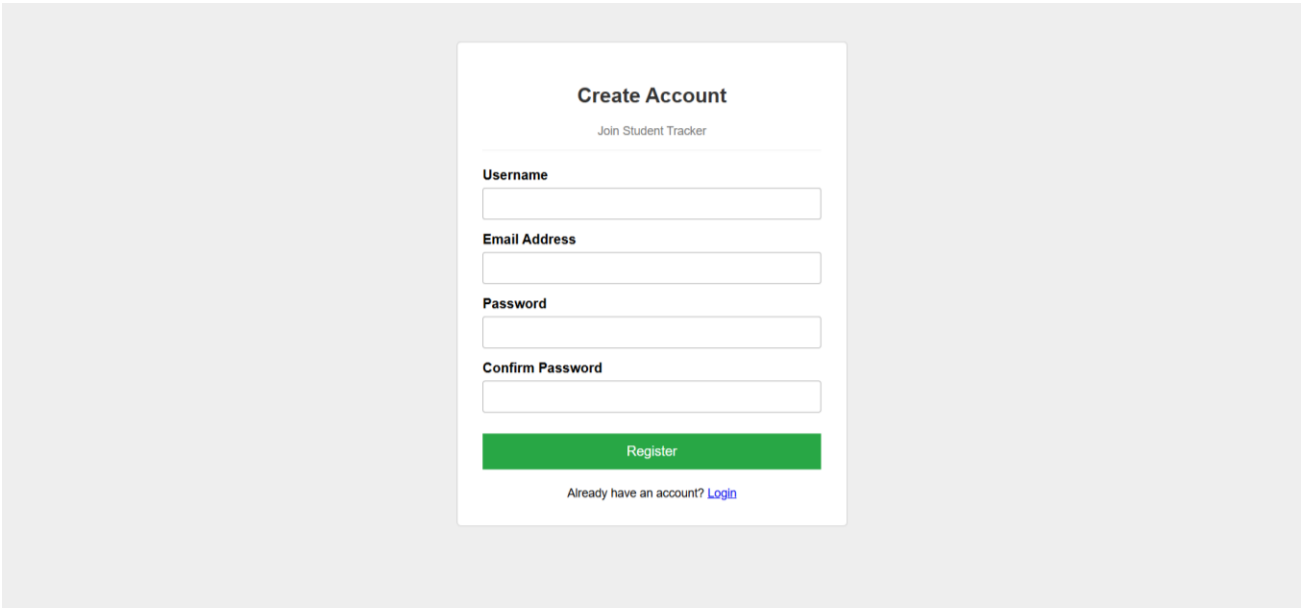
The modular layout makes the code easier to understand and improve.

Screenshots / Results:

The main dashboard showing progress:

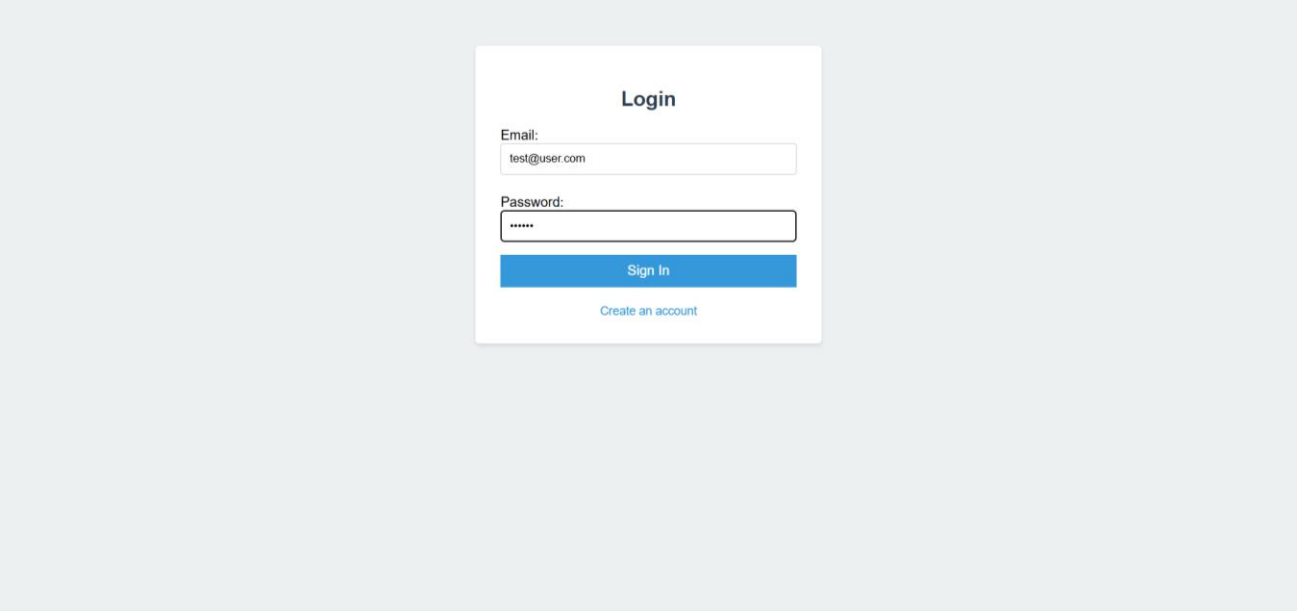


The registration interface:

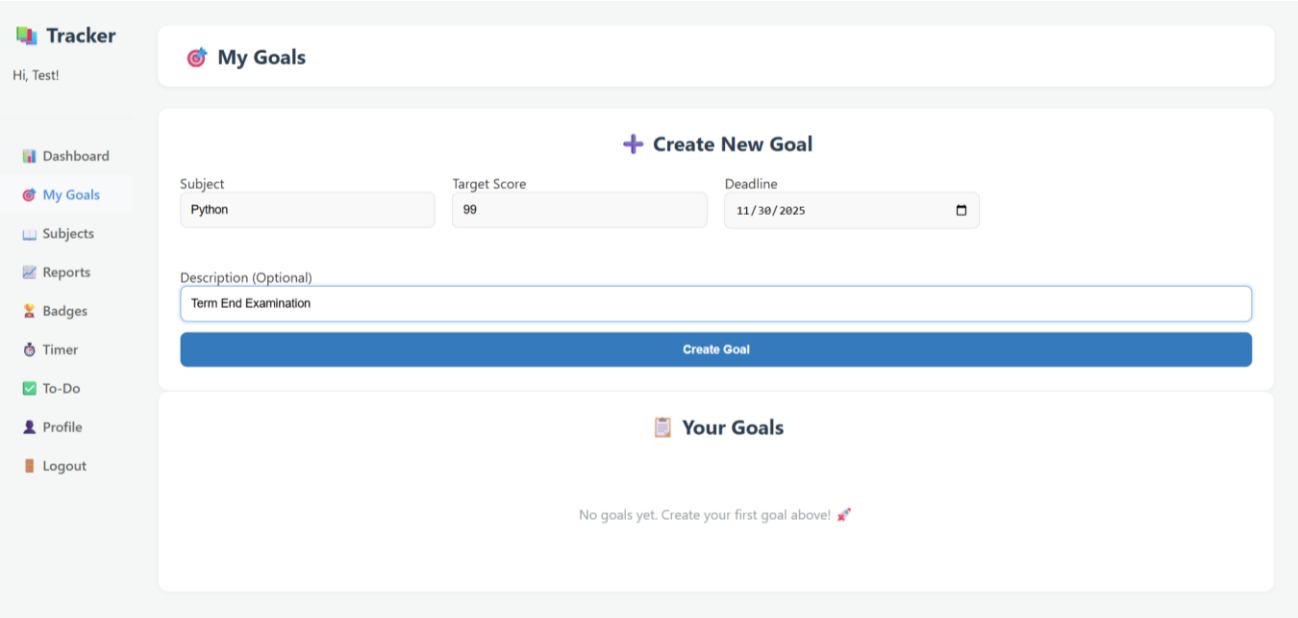


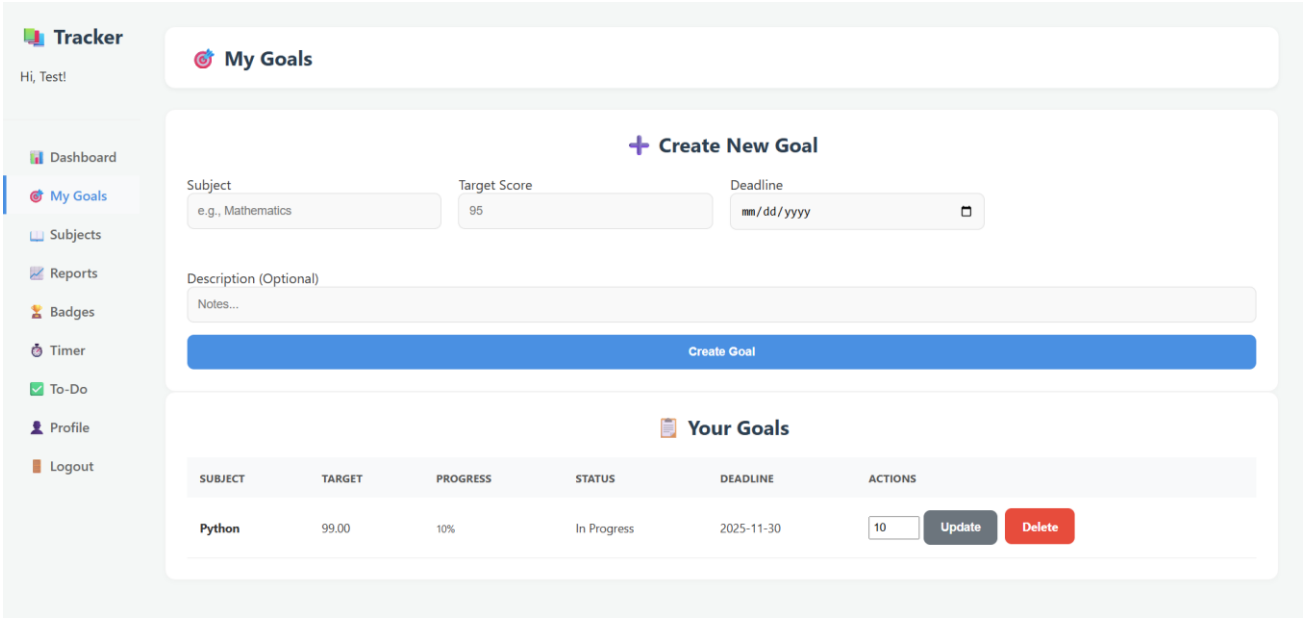
Name: Raviraj Jain Registration No: 25BAI11023

The login interface:

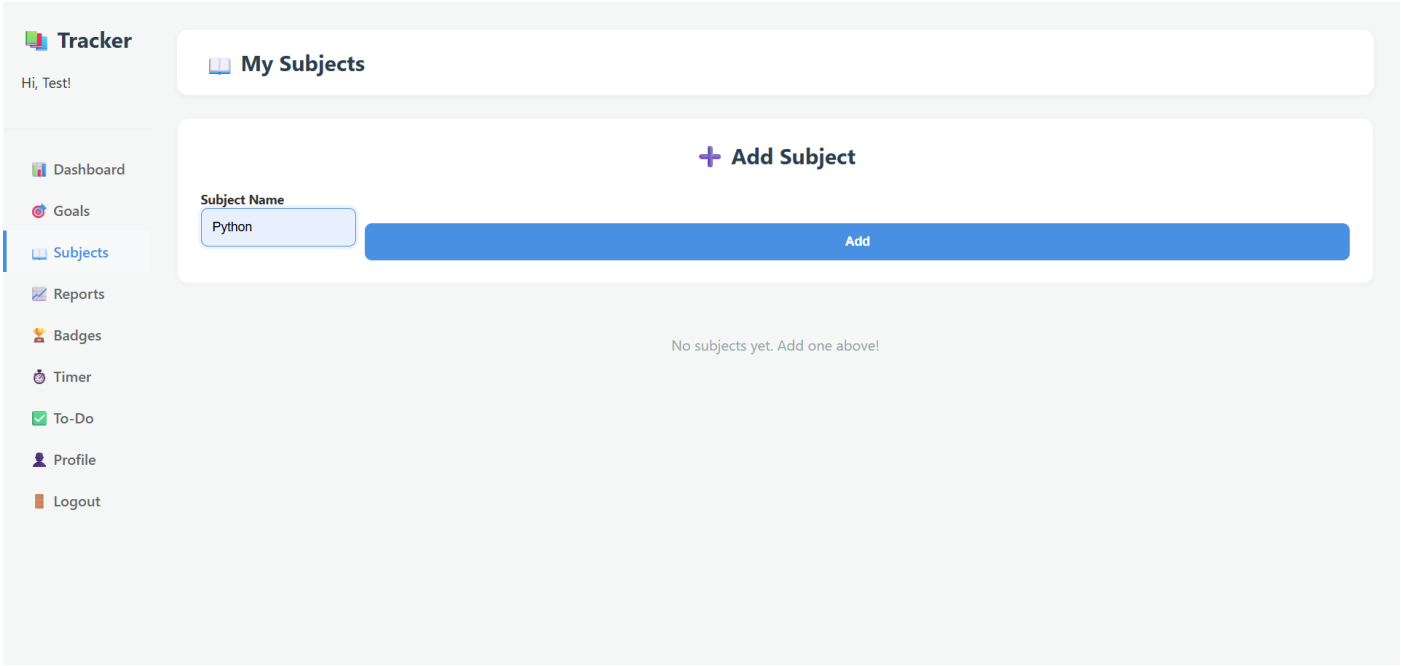


The goal management interface:



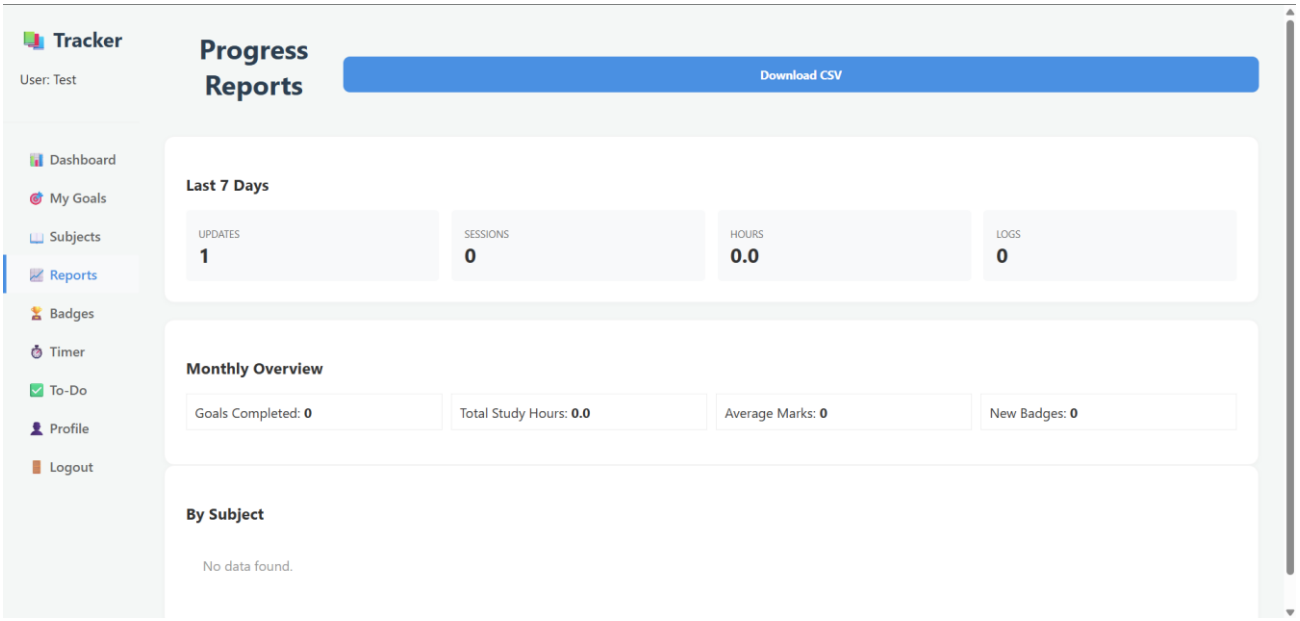


The subject management interface:

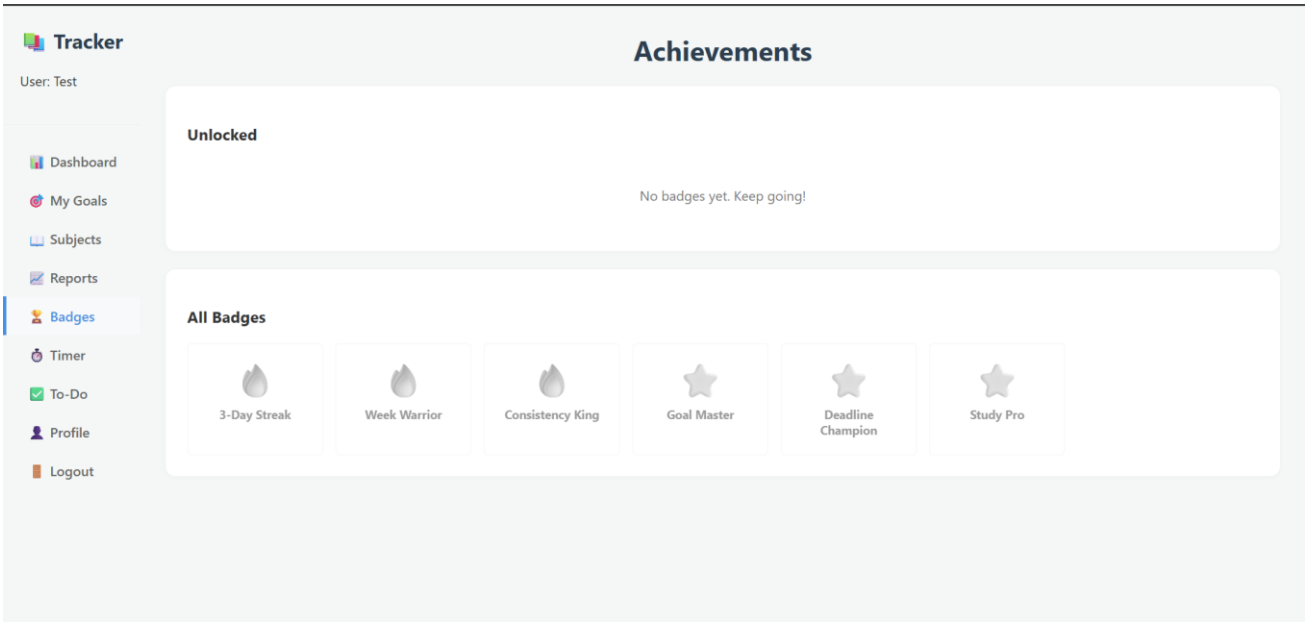


Name: Raviraj Jain Registration No: 25BAI11023

The report management interface:

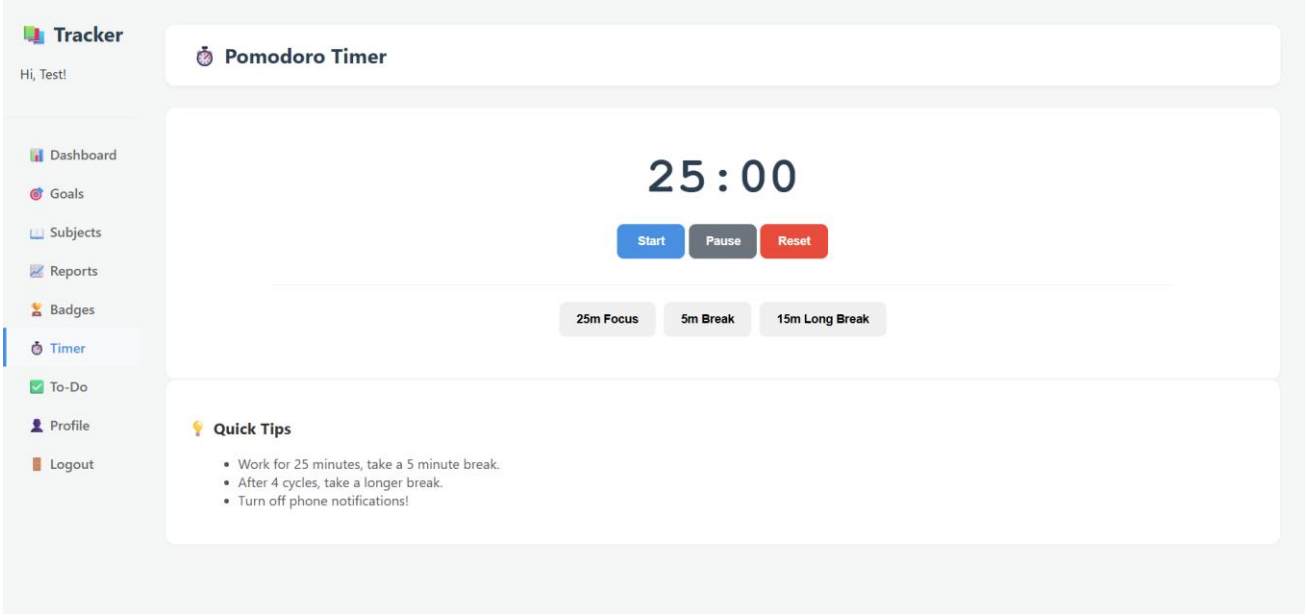


The badges management interface:

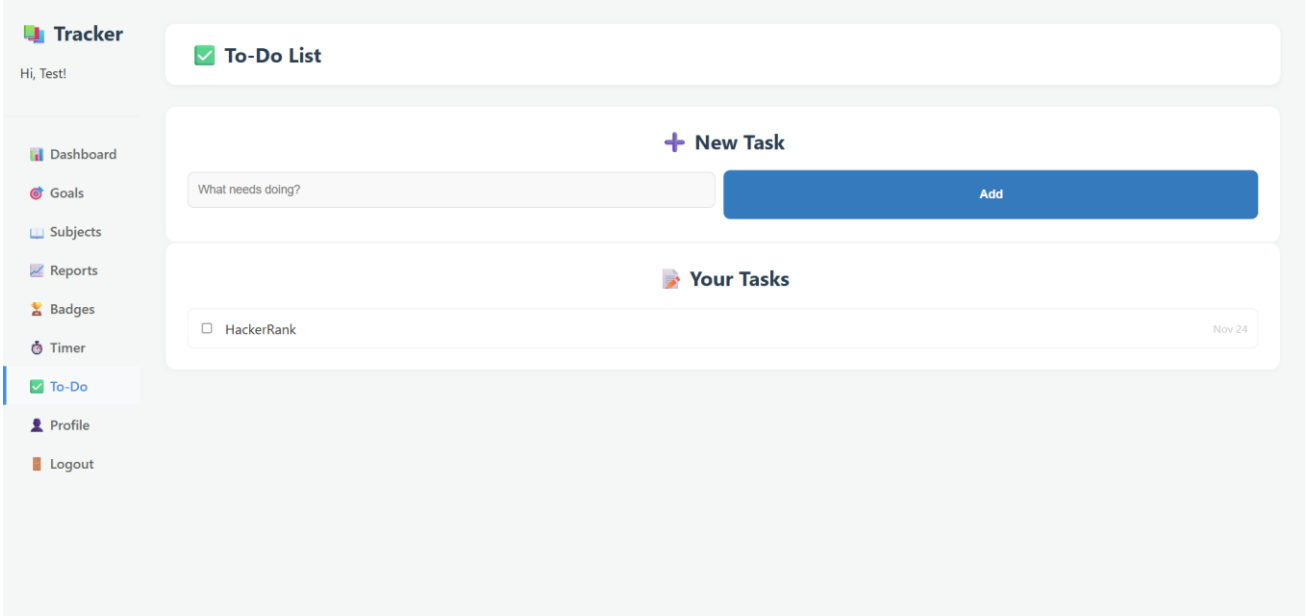


Name: Raviraj Jain Registration No: 25BAI11023

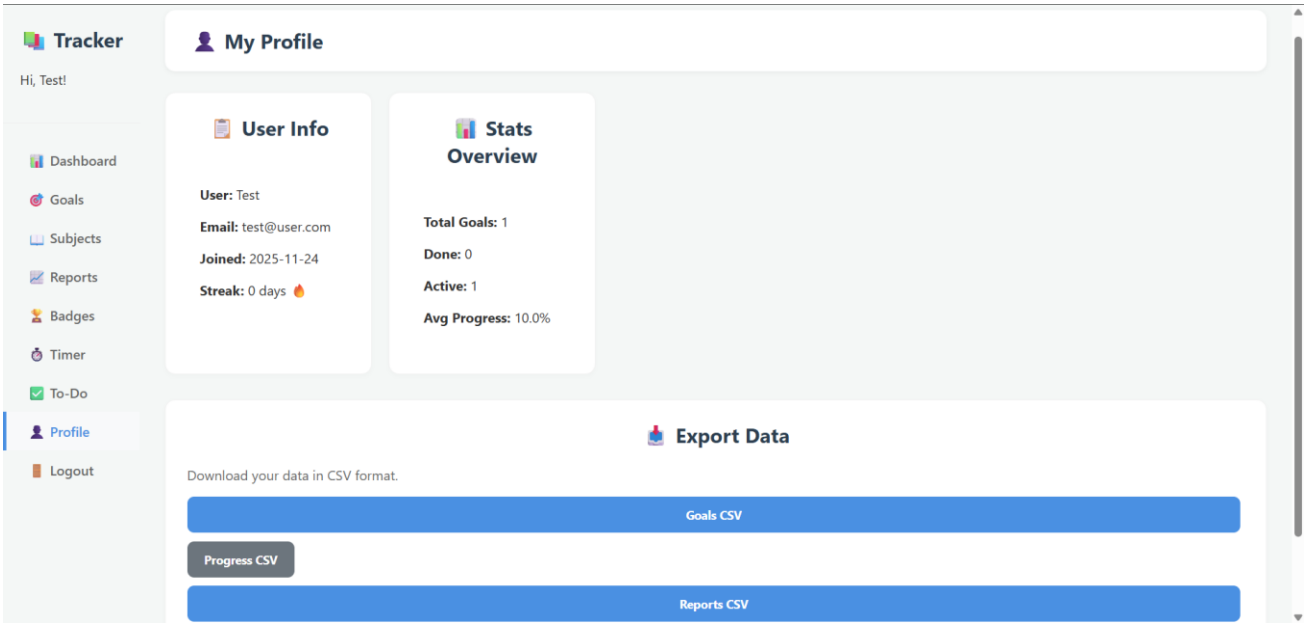
The timer interface:

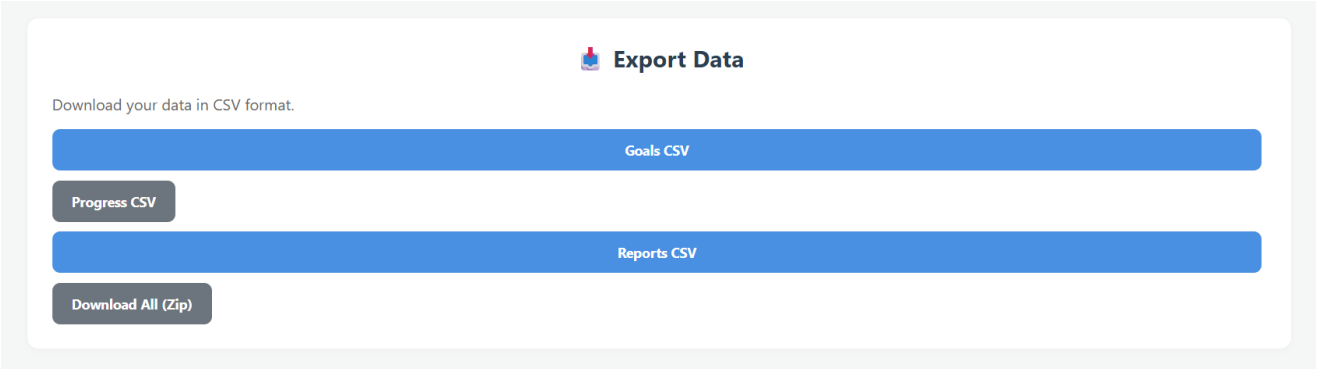


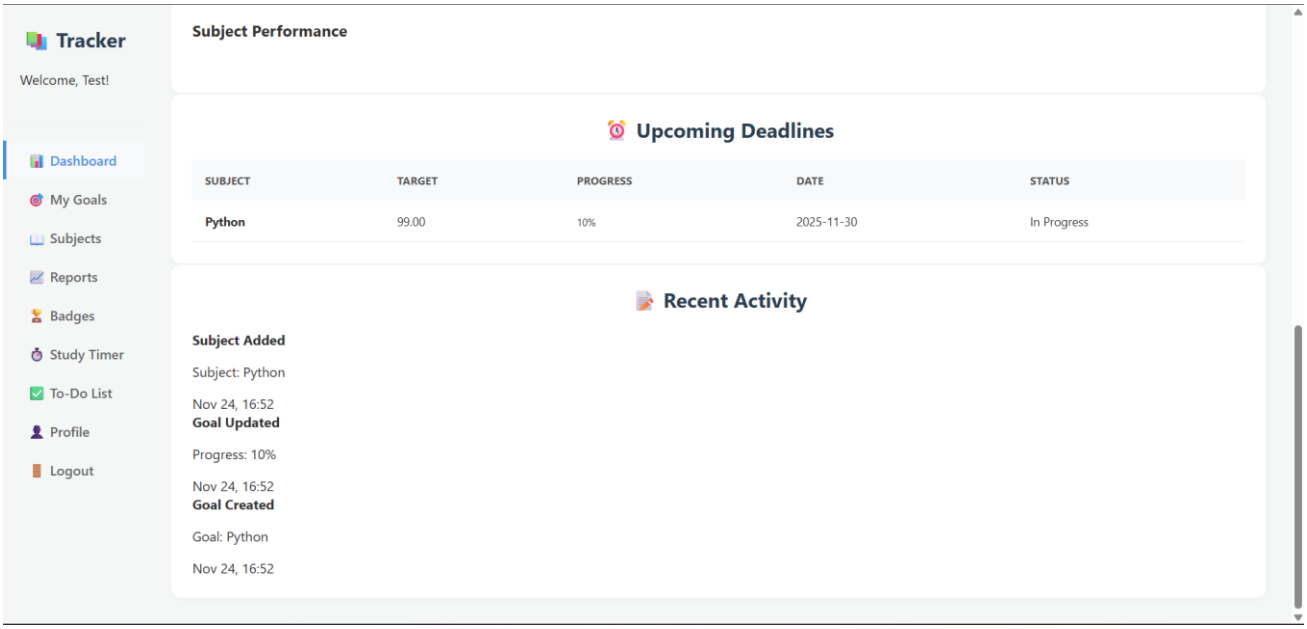
The To-Do management interface:



The profile management interface:







Testing Approach:

1. Functional Testing

Each feature was tested individually:

- Login
- Goal creation
- Marks entry
- Charts

2. Validation Testing

- Blank fields
- Invalid marks
- Incorrect passwords

3. UI Testing

Ensured that pages load properly and buttons respond correctly.

4. Basic Unit Tests

Checked database insertions and streak calculations.

Challenges Faced:

- Setting up a clean and scalable database structure
- Linking matplotlib charts to Flask routes
- Ensuring streak logic resets correctly after inactivity
- Designing a simple UI that still looks clean
- Managing data flow between modules

These challenges helped deepen understanding of real-world development.

Learnings & Key Takeaways:

- How to build a complete end-to-end web application
- Database structuring and CRUD operations

- Role of modular coding in Python
- Handling user sessions
- Generating charts dynamically
- Planning documentation and diagrams
- Importance of error handling and usability

This project strengthened both conceptual and practical skills.

Future Enhancements:

In future versions, the system can include:

- A study timetable generator
- Daily motivational reminders
- Weekly performance reports
- Mobile app interface
- Cloud backup
- AI-based performance predictions
- Pomodoro study timer

These additions would make the system more smart, modern, and helpful.

References:

- VITyarthi Lectures
- Official Flask documentation
- MySQL Reference Manual
- W3Schools (HTML/CSS basics)
- Python Matplotlib Guide
- Classroom lecture notes
- Online articles for best development practices