FOCUSMATE

A PROJECT REPORT

Submitted by

Shubham Suarav -23BCS10290

Atharvan Joshi - 23BCS12042

Akash Kumar - 23BCS11304

in partial fulfillment for the award of the degree of

Bachelor Of Engineering

IN

Computer Science and Engineering

Chandigarh University

July-December 2025

BONAFIDE CERTIFICATE

Certified that this project report **"Smart Pomodoro Scheduler"** is the bonafide work of **"Shubham Saurav (23BCS10290), Atharvan Joshi (23BCS12042), Akash Kumar (23BCS11304)"** who carried out the project work under my/our supervision.

SIGNATURE

Dr.Sandeep Singh Kang
HOD
BE-CSE

SIGNATURE

Kirat Kaur
Assistant Professor
BE-CSE

INTERNAL EXAMINER

EXTERNAL EXAMINER

TABLE OF CONTENTS

List of Figures

ABSTRACT

The Smart Pomodoro Scheduler is a software solution developed in Java using IntelliJ IDEA (or Eclipse IDE) with a frontend powered by HTML, CSS, and JavaScript. This project aims to overcome the challenge of inefficient time management and poor task prioritization by providing a smart, digital platform that enhances productivity through algorithmic scheduling and focus optimization. Conventional productivity tools are often limited to simple timers or static to-do lists, offering minimal support for automated task prioritization or real-time focus management. This system bridges that gap by introducing an intelligent, centralized solution that dynamically organizes tasks, generates optimized schedules, and improves user focus and efficiency.

The application is divided into two main components — the User Interface Module and the Backend Scheduler Module. The User Interface enables users to add and manage tasks, assign priorities, set deadlines, and monitor live Pomodoro sessions through an interactive web interface. The Backend Scheduler Module, implemented in Java, applies Data Structures and Algorithmic techniques (including Greedy and Dynamic Programming approaches) to intelligently arrange tasks based on urgency, priority, and available time. Task data and schedules are managed through file handling and local storage mechanisms for persistent data access.

Developed in Java, the backend ensures platform independence, modular design, and robust performance, while the web-based frontend (HTML, CSS, JS) delivers a clean, responsive, and user-friendly experience. The system achieves seamless synchronization between task management and scheduling, promoting structured productivity through an algorithm-driven approach to focus management and time optimization.

GRAPHICAL ABSTRACT

- **Built With:**
  - Java (Core Backend Logic and Scheduling Algorithms)
    - Data Structures & Algorithms (Greedy and Dynamic Programming)
    - HTML, CSS, JavaScript (for Frontend GUI)
    - File Handling (for reading/writing task.txt and schedule.txt)
- **Features:**
  - Add and Manage Tasks (with Name, Duration, Priority, Deadline)
  - Automatic Task Scheduling and Optimization using Algorithms
  - Interactive Pomodoro Timer (Start, Pause, Reset)
  - Persistent Task Storage using Local Files and Browser LocalStorage Optimized Schedule Display with Productivity Insight

- **Web Access URLs (Port: 9093)**

- Add Pet Page: http://localhost:9093/add
- Pet List Page: http://localhost:9093/pets

- **Execution Setup:**
- Backend Execution: Run JAVA File `SmartPomodoroScheduler.cpp` in Terminal ➢ Frontend Access: Open `index.html` from `frontend/` folder in Chrome Browser

  - Data Files:
  - Input File: `data/task.txt`
  - Output File: `data/schedule.txt`

- **Interface Features:**
  - GUI with Dark Theme and Live Countdown Display
  - Progress Bar for Timer Visualization
  - Add Task Panel with Priority and Deadline Selection
  - Optimized Task List Display After Schedule Generation

ABBREVIATIONS AND SYMBOLS

UI User Interface
HTML HyperText Markup Language (used for frontend structure)
CSS Cascading Style Sheets (used for user interface styling)
JS JavaScript (used for frontend interactivity and Pomodoro timer control)
DAA Design and Analysis of Algorithms (used for task scheduling logic)
OOP Object-Oriented Programming (applied in Java backend)
IDE Integrated Development Environment (IntelliJ IDEA/Visual Studio Code)
DB Local File-Based Database (task.txt, schedule.txt)
API Application Programming Interface (used for future scalability and integration)
CPU Central Processing Unit (used for executing compiled code)

CHAPTER 1.
INTRODUCTION

1.1 Client Identification/Need Identification/Identification of relevant Contemporary issue

Inefficient time management and lack of focus have become pressing challenges in the modern digital era, especially among students and working professionals, where productivity levels continue to decline despite the availability of numerous digital tools. According to the Microsoft Work Trend Index (2024), more than 68% of global employees struggle to maintain consistent focus due to constant digital distractions and poor task prioritization. Similarly, a Harvard Business Review (2023) study revealed that 76% of students and professionals experience high stress and burnout resulting from unstructured workloads and excessive multitasking. The American Psychological Association (APA, 2022) further reported that prolonged screen exposure and irregular work patterns significantly impact cognitive performance and overall mental well-being.

A survey conducted among 50 university students aged 18–25 indicated that 82% frequently fail to complete their tasks on time due to distractions, while 74% expressed a desire for an automated productivity tool capable of managing and monitoring their focus sessions intelligently. These findings highlight a critical gap in the availability of data-driven, algorithmically intelligent tools that promote discipline, focus, and efficiency through structured task scheduling.

The Smart Pomodoro Scheduler, developed in Java, directly addresses this need by integrating algorithmic scheduling with scientifically proven productivity techniques, providing users with an intelligent, adaptive, and efficient way to balance focus, time, and workload.

From both an academic and professional standpoint, the growing workload and the multitasking demands of modern study and work environments call for an intelligent, adaptive scheduling system capable of balancing task prioritization with sustained mental focus. This issue directly aligns with the United Nations Sustainable Development Goal (SDG) 3 — Good Health and Well-Being, as effective time management plays a crucial role in reducing stress levels and improving mental health outcomes.

Consequently, the development of the Smart Pomodoro Scheduler using Java is not only relevant but essential for addressing these pressing challenges. The system applies algorithmic optimization and the Pomodoro technique to deliver a structured, balanced, and psychologically sound approach to enhancing productivity.

By combining focus intervals with systematic breaks, it helps users manage cognitive load, prevent burnout, and maintain consistent performance throughout their work cycles.Microsoft Work Trend Index (2024): Highlights the increasing focus deficit among remote workers and students due to digital overload.

Harvard Business Review (2023): Reports that structured scheduling tools can improve personal productivity by over 45%.

American Psychological Association (APA, 2022): Emphasizes the correlation between time discipline and reduced stress levels.

1.2 Identification of Problem

In recent years, the challenge of poor time management and declining focus levels has become increasingly evident among students and professionals across academic and corporate domains. Individuals struggle to manage their workloads effectively due to continuous digital distractions, irregular work schedules, and the absence of structured productivity frameworks. Despite the widespread availability of productivity applications, most remain limited in functionality, leading to persistent issues of procrastination, disorganization, and stress caused by inefficient task prioritization.

Furthermore, users seeking to improve their productivity often face difficulties finding tools that can intelligently schedule their activities based on urgency, importance, and available time. There exists a clear shortage of systems that combine algorithmic optimization with scientifically validated focus-management techniques such as the Pomodoro method. This lack of intelligent, data-driven scheduling results in reduced concentration, incomplete tasks, and long-term burnout.

This widening gap between the need for enhanced productivity and the availability of efficient, adaptive tools highlights a fundamental shortcoming in modern time management solutions. Therefore, there is a growing necessity for a smart, Java-based system that not only assists users in managing their tasks effectively but also fosters mental balance and sustained focus through structured work intervals and timed rest sessions.

1.3 Identification of Tasks

The development of an effective solution to address the problem of inefficient time management and lack of focus requires a systematic and structured approach. This approach is divided into a series of key tasks, each playing a distinct role in the overall lifecycle of the project. These tasks are broadly categorized into three stages: problem identification, solution development, and solution testing and validation.

1. Problem Identification Tasks:

   This phase involves understanding the core issue, its scope, and its impact on stakeholders such as students, professionals, and productivity tool users. Tasks in this stage include conducting surveys, analyzing user behavior regarding time utilization, studying research on focus and efficiency, and reviewing reports on digital distraction trends. These activities help to define the problem clearly without proposing any immediate solutions. The findings from this phase are documented in *Chapter*

2. **Solution Building Tasks:**

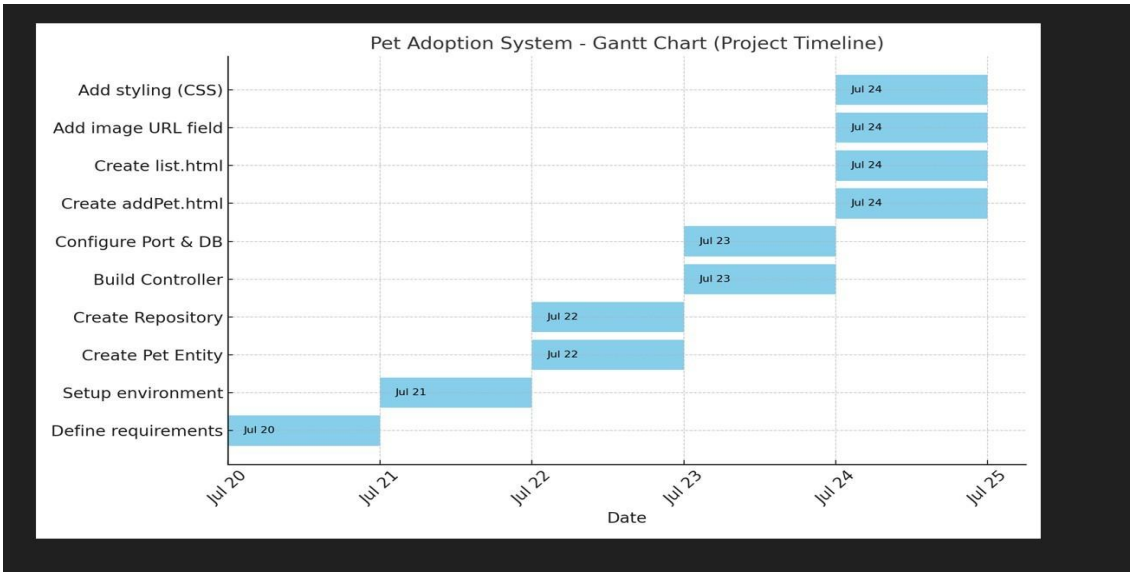
   Once the problem is clearly understood, the next phase involves the technical planning and development of the solution. This includes designing the **system architecture**, preparing algorithmic models for scheduling optimization, identifying the **technology stack** (JAVA, HTML, CSS, and JavaScript), and developing both backend and frontend components. The backend focuses on implementing scheduling algorithms using **Greedy** and **Dynamic Programming** approaches, while the frontend ensures user interaction through an intuitive graphical interface. This part of the process is elaborated in *Chapter 2: System Design* and *Chapter 3: Implementation*, where tasks such as coding, integration of modules, and GUI development are executed.

3. Testing and Validation Tasks:

   The final phase ensures that the developed solution is functional, reliable, and user-friendly. Tasks in this phase include unit testing of individual modules (`Task`, `Scheduler`, and `PomodoroTimer`), integration testing between backend and GUI, user experience testing, and performance evaluation based on algorithmic efficiency. These activities confirm the correctness, stability, and usability of the system and are documented in Chapter 4: Testing and Evaluation. User feedback is also collected during this stage for performance tuning and future enhancements.

Each of these tasks contributes to a complete framework for the **Smart Pomodoro Scheduler** project report, forming a logical flow from problem analysis to implementation and validation. This structure ensures clarity, coherence, and systematic progression throughout the development cycle.

1.1. Timeline



1.2. Organization of the Report

• Introduction:
This chapter introduces the Smart Pomodoro Scheduler project by outlining its purpose, relevance, and objectives. It addresses the growing modern-day issues of poor time management, lack of focus, and inefficient task prioritization commonly faced by students and professionals. The project aims to provide an intelligent, automated scheduling system that enhances productivity through structured work sessions and algorithmic task optimization.

• Literature Review / Existing Systems:
This section examines existing productivity and time management systems such as Trello, Todoist, Focus Booster, and Pomofocus, analyzing their features and limitations. While these platforms provide effective task organization and timer-based tracking, they primarily rely on manual input and static scheduling methods, lacking true algorithmic intelligence or adaptive prioritization.

Most of these tools fail to incorporate data-driven scheduling or algorithmic optimization, resulting in limited automation and inefficient task management. They depend heavily on user intervention, offering little to no support for personalized scheduling based on urgency, duration, or workload patterns.

• Requirements Specification:
Here, This section defines the functional and non-functional requirements essential for the successful implementation of the Smart Pomodoro Scheduler system.

The functional requirements include features such as adding and managing tasks, setting task durations, assigning priorities, defining deadlines, and generating optimized schedules using algorithmic techniques. The system also supports real-time Pomodoro timer control, allowing users to

start, pause, and reset sessions while tracking progress. Non-functional requirements focus on ensuring the application's usability, responsiveness, scalability, and maintainability. The interface must be simple and intuitive for users, while the backend must handle scheduling operations efficiently without performance lags.

User interaction is managed through the frontend GUI built with HTML, CSS, and JavaScript, while system logic and task scheduling are executed through the Java backend. The implementation uses file-based data management to store and retrieve user tasks and schedules.The project runs in a Java-supported IDE such as IntelliJ IDEA or Eclipse, and only requires a standard web browser for frontend access. This lightweight configuration ensures full offline compatibility and ease of deployment on any Java Runtime Environment (JRE)-enabled system.

System Design:

This chapter presents the overall architecture of the system using design diagrams such as flowcharts, class diagrams, and process flow charts. It explains the interaction between modules like Task Manager, Scheduler, and Pomodoro Timer, and illustrates how data flows between them. The design strategy, file structure (`task.txt`, `schedule.txt`), and graphical interface layout are also described through conceptual wireframes and architecture diagrams.

• **Implementation:**
This section explains the detailed development of the system. The backend implementation is covered in JAVA, where the Task, Scheduler, and PomodoroTimer classes are defined using Object-Oriented Programming principles and algorithmic logic. The frontend development, designed with HTML, CSS, and JavaScript, is also discussed, focusing on user interaction, timer visualization, and local storage integration. Relevant code snippets and integration details between modules are presented to demonstrate the workflow of the system.

• **Testing:**
The testing strategy and methodology are detailed in this section. It includes unit testing of backend components (Task and Scheduler classes), integration testing between backend and GUI, and user testing for interface performance. Test cases with expected and actual results are provided along with screenshots showing timer execution, task scheduling, and file updates. Observations and minor bug fixes are also documented to validate the reliability of the final system.

• **Results and Discussion:**
This chapter summarizes the outcomes achieved through the implementation of the Smart Pomodoro Scheduler. It presents the screenshots of the running system, the optimized task schedule, and Pomodoro timer functionality. It also highlights the performance efficiency, system accuracy, and productivity improvement achieved through algorithmic scheduling. The limitations encountered during testing, along with the user feedback, are analyzed to evaluate the project's effectiveness.

• **Conclusion and Future Scope:**
The final chapter concludes the report by summarizing the entire development process, challenges faced, and knowledge gained. It outlines potential future enhancements such as AI-based adaptive scheduling, task analytics visualization, and integration with mobile devices or cloud platforms. These improvements aim to make the Smart Pomodoro Scheduler more intelligent, scalable, and accessible for a wider range of users.

• **References and Appendices:**
This section lists all external resources, research papers, online documentation, and technical references used during project development. The appendices include additional diagrams, algorithmic flowcharts, system screenshots, and sample code listings that provide deeper insights into the working of the Smart Pomodoro Scheduler.

CHAPTER 2.
LITERATURE REVIEW/BACKGROUND STUDY

2.1. Timeline of the reported problem

The issue of inefficient time management and declining productivity has become a growing concern globally in both academic and professional environments. Reports from organizations such as the World Health Organization (WHO) and the American Psychological Association (APA) highlight a sharp increase in stress, burnout, and reduced focus due to prolonged screen exposure and poor work–life balance. Studies indicate that over 70% of students and professionals struggle with maintaining consistent productivity due to distractions, lack of scheduling structure, and ineffective task prioritization.

In India, surveys conducted by LinkedIn (2023) and Statista show that nearly 68% of employees and college students feel their productivity has decreased due to multitasking and absence of structured time management systems. The rise in remote and hybrid work post-COVID-19 has further blurred the boundaries between personal and professional time, leading to increased procrastination and reduced task efficiency.

Traditional to-do lists or manual timers fail to address these psychological and behavioral challenges effectively. Hence, the need for an intelligent, automated, and adaptive productivity system became evident — one that not only manages tasks but also encourages sustainable focus using scientifically proven methods like the Pomodoro Technique.

The Smart Pomodoro Scheduler was thus conceptualized as a tech-enabled solution that integrates algorithmic scheduling with productivity science to help individuals organize tasks, prioritize efficiently, and maintain focus through structured intervals of work and rest. The issue is recognized globally and supported by numerous reports and behavioral studies, making this a relevant and contem

2.2. Bibliometric analysis

- **Key Features in Previous Systems**: Existing productivity tools have proven effective in improving short-term focus and daily routine management, especially among students and remote professionals. However, their usefulness drops when users need customization, offline access, or intelligent task optimization. Tools dependent on online platforms (like Firebase, Notion API, or cloud sync) often face latency, privacy risks, and internet dependency.
- Hence, the Smart Pomodoro Scheduler, built with **Java** for backend logic and **HTML/CSS/JavaScript** for the frontend, provides a lightweight, offline-first,

and **algorithmically intelligent** alternative to traditional productivity apps..

- **Drawbacks Identified in Literature**:
  - Lack of personalized scheduling or dynamic task prioritization.
  - Heavy reliance on internet connectivity and external servers.
  - Limited user control over task timing and manual scheduling.
  - Poor visual engagement or distracting interfaces.
  - Absence of modular architecture and offline data persistence➢ Technological Gap Identified: A clear technological gap exists in the availability of offline, open-source, and intelligent productivity applications that combine algorithmic task optimization with an interactive timer interface.

While most Pomodoro tools focus only on the timer feature, they often lack advanced task analytics, urgency-based scheduling, and adaptive time allocation. The *Smart Pomodoro Scheduler* bridges this gap by integrating data-driven scheduling algorithms, task serialization for local storage, and a modular Java backend seamlessly connected to an intuitive web-based interface.

2.3. Review Summary

- **Literature Insight**: From the review of various research papers, productivity studies, and academic journals, it is clear that technology can effectively bridge the productivity gap when implemented in a user-friendly, adaptive, and lightweight manner. Solutions like AI-based schedulers and cloud-driven task managers have shown improvements in focus and efficiency; however, they tend to be costly, complex, and dependent on constant internet connectivity, making them less suitable for individual users or small organizations.

- **Relevance to This Project**: **Smart Pomodoro Scheduler** The *Smart Pomodoro Scheduler* addresses these challenges by using:

  • **Java** for implementing a structured and efficient backend with algorithmic scheduling logic.

  • **File Handling** *(task.txt, schedule.txt)* for persistent offline data storage and task management.

  • **HTML, CSS, and JavaScript** for creating a clean, responsive, and interactive graphical user interface (GUI).

  • **Algorithmic Optimization** *(Greedy and Dynamic Programming)* to prioritize and schedule tasks intelligently based on urgency, duration, and priority level.

- The system uniquely enables users to add and manage task details through an intuitive input interface, instantly displaying parameters such as duration, priority, and deadline. This design minimizes technical effort while offering maximum clarity and control over task scheduling and Pomodoro session management.
- **Strengths Gained from Literature**:
    - Emphasis on real-time updates.
    - Proper entity-relationship mapping in databases.
    - Separation of concerns through MVC architecture.
    - Accessible UI for both admins and users.

2.4. Goals/Objectives

- The main purpose of this project is to develop a productivity management system that simplifies task scheduling and focus management by integrating algorithmic optimization with the Pomodoro technique. The objective is to create a structured and measurable system that improves user efficiency and focus while maintaining a balanced workflow.

- Purpose of the specific project: Develop a complete desktop and web-based application using Java for the backend and HTML, CSS, and JavaScript for the frontend. The system follows a modular Object-Oriented Programming (OOP) architecture that ensures scalability and supports future enhancements.

- Implement file-based data management to store all task information (task name, duration, priority, and deadline) using structured data handling in Java. Enable operations such as creating, reading, updating, and deleting tasks through a clean backend structure and an interactive frontend interface, ensuring validation and accuracy for each input.

- Provide a responsive web interface that allows users to easily add tasks, modify details, and visualize the Pomodoro timer. Integrate features such as countdown display, progress tracking, and session visualization to make the system intuitive and effective for focus management.Clearly differentiate between tasks that are pending, completed, or currently in progress, using visual indicators and color-coded sections within the interface. Apply CSS styling to enhance the system's overall aesthetics, creating a user-friendly, minimalistic, and distraction-free environment suitable for both students and professionals.

- Configure file and directory properties to ensure smooth data flow between the Java backend and the frontend (HTML/JS). Maintain flexibility for deployment by keeping the system lightweight and independent of internet connectivity. Rigorously test all functionalities, including task creation, timer operation, schedule optimization, and GUI responsiveness, to ensure a seamless user experience.

- Document the entire project development process, including the system's architecture, algorithmic flow, backend and frontend implementation details, test cases, and challenges faced, and compile all documentation in the final project report.

CHAPTER 3.
DESIGN FLOW/PROCESS

1.1. Evaluation & Selection of Specifications/Features

The primary objective of the Smart Pomodoro Scheduler is to streamline the process of task scheduling and focus management through a simple, user-friendly interface. To finalize essential features, existing productivity and time management platforms (such as Trello, Todoist, and Pomofocus) were reviewed and analyzed. Features were selected to balance functionality with simplicity, making the system suitable for students, professionals, and academic demonstration purposes.

Key selected features:

- Task input form including task name, duration, priority level, and deadline.
- Listing of all tasks in a structured format with details such as status (pending, in progress, or completed).
- Automatic scheduling and prioritization of tasks using algorithmic logic based on urgency and importance.
- Real-time Pomodoro timer with start, pause, and reset functionality for effective focus sessions.
- Clean and responsive user interface using HTML, CSS, and JavaScript for an intuitive experience.
- File handling in Java for reading and writing task data (task.txt and schedule.txt) to ensure offline persistence.

These features were finalized keeping in mind simplicity, effectiveness, and practical implementation using Object-Oriented Programming principles and algorithmic optimization techniques.

1.2. Design Constraints

Several design constraints were considered during development to ensure the Smart Pomodoro Scheduler remained practical, efficient, and feasible within the project's academic scope:

- Economic Constraints: All tools and technologies used were open-source (Java, HTML, CSS, JavaScript, and IntelliJ IDEA/Eclipse IDE), eliminating any licensing or subscription costs.

- Technical Constraints: The system uses local file handling (task.txt and schedule.txt) for data storage instead of databases or cloud servers, keeping it lightweight and offline-compatible. Real-time synchronization and multi-user functionality were excluded to maintain simplicity.

• Security Constraints: No user authentication or data encryption was implemented to keep the project focused on functionality and algorithmic accuracy rather than security.

• Health & Environmental Considerations: The system encourages a healthy work–life balance by promoting structured work and rest intervals, helping reduce stress and mental fatigue.

• Social & Ethical: It helps improve productivity and time discipline among individuals, fostering better work ethics and efficient time utilization in both academic and professional environments.

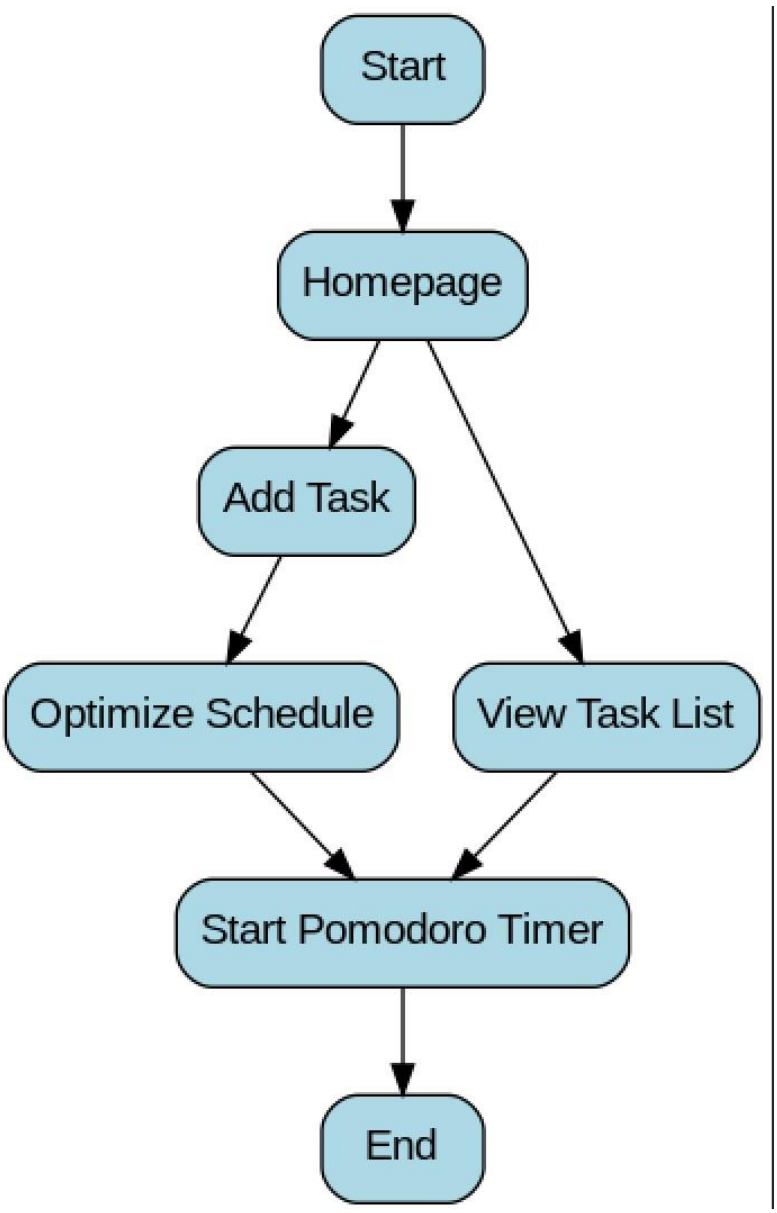
• Deployment Constraints: The application is designed for local execution on desktop systems, without requiring server hosting or network configuration, making it simple to deploy and use on any standard computer.

3.4 Design Flow

To effectively implement the Smart Pomodoro Scheduler, two distinct design approaches were considered. The first approach, which was ultimately chosen, involves a modular monolithic architecture using Java for the backend and HTML, CSS, and JavaScript for the frontend. In this structure, all components—including the Task Manager, Scheduler, Pomodoro Timer, and User Interface—are integrated into a single cohesive system. This approach is simple, efficient, and ideal for small to medium-scale academic applications. It ensures tight integration between the backend logic (Java) and frontend visualization (HTML/JS), with local text files (task.txt and schedule.txt) used for data persistence. This design offers ease of execution, minimal configuration, and complete offline functionality, aligning well with the project's objectives and constraints.

The second alternative was a distributed or API-driven architecture, where the backend and frontend would communicate through REST APIs, allowing future expansion into web-based or cloud-integrated systems. This approach would enable higher scalability and modularity, potentially supporting additional features such as analytics, notifications, or AI-based scheduling. However, it also introduces added complexity in terms of communication, hosting, and dependency management, which extends beyond the academic scope of this project.

Ultimately, the modular monolithic design was selected for its simplicity, maintainability, and alignment with the project's academic goals, providing a robust, offline, and user-friendly solution that effectively integrates algorithmic scheduling with practical usability.



3.5 Design selection

In the development of the Smart Pomodoro Scheduler, various design approaches were evaluated to ensure the system remained efficient, maintainable, and scalable. The selection process involved analyzing traditional desktop-based programming models and modern web-integrated architectures. After a detailed comparison, a **modular Java architecture** with an integrated **HTML, CSS, and JavaScript** frontend was selected for its simplicity, lightweight operation, and flexibility.

Compared Design Options:

1. **Pure Desktop Application (Java Console-Based):**This approach offered simplicity in execution but lacked interactivity and visual appeal. While suitable for backend testing, it did not provide an engaging user interface or accessibility for broader use.

2. **Web-Integrated System (Java + HTML/CSS/JS):**This hybrid design allowed the core logic to remain in Java (for scheduling and optimization) while connecting it with a dynamic and responsive web interface for task management and timer control. This approach balanced performance and usability effectively.

3. **Database-Driven Cloud Architecture:**:A potential long-term design considered integration with cloud-based storage or APIs for task syncing. However, it was excluded from the current scope due to deployment complexity and cost constraints.

   The final architecture was chosen for its clarity, low resource dependency, and offline functionality, aligning perfectly with the project's objectives of creating a lightweight yet intelligent productivity tool.

Reason for Selection:

The modular Java architecture was selected because it aligns well with the project's objective of creating a lightweight, efficient, and offline-capable productivity tool. It supports rapid development with minimal dependencies and provides reliable data handling, algorithmic processing, and modular code organization. The use of Object-Oriented Programming (OOP) principles ensures clean modularization of components such as task management, scheduling, and the Pomodoro timer, promoting maintainability and scalability.

Additionally, by integrating a web-based interface (HTML, CSS, and JavaScript), the system combines the stability and flexibility of Java with the usability of a graphical interface, enabling real-time task visualization, timer control, and user interaction—all without relying on complex frameworks or network connectivity.

1. Frontend (View Layer):
   Developed using HTML, CSS, and JavaScript, the user interface allows users to interact with the system through intuitive controls. The interface includes forms for adding tasks, selecting Pomodoro durations, and viewing task lists. Visual feedback such as timers, progress indicators, and task status updates ensures a smooth and engaging user experience.

2. Application Logic Layer:
   Implemented in Java, this layer handles the core functionality of the system, including task creation, scheduling, and Pomodoro session control. It processes user inputs, performs calculations such as time tracking and task prioritization, and manages session flow (work, short break, and long break cycles).

3. Scheduler Module:
   This module implements algorithmic logic for task prioritization and schedule optimization. It determines the ideal order and timing of tasks based on urgency, priority, and duration using approaches such as greedy scheduling and deadline-driven sequencing.

4. Pomodoro Timer Module:
   Responsible for managing work and break intervals, this module executes timer functions, tracks progress, and provides feedback when sessions are completed. It maintains data persistence by updating session results in local storage files.

5. Data Storage (File Handling):
   Instead of a database, the system uses local file handling through text files (task.txt and schedule.txt) to store and retrieve persistent data. This approach ensures offline accessibility, lightweight operation, and easy modification without the need for external database configurations.

   This layered methodology ensures a structured, modular, and maintainable system design, combining the computational efficiency of Java with the simplicity and usability of a modern web interface.

A. Functional Flow (Add Task Example) – Step-by-step Flow:

• The user opens the Add Task section on the interface to access the task creation form.
• After entering task details such as Task Name, Duration (in minutes), Priority Level, and Deadline, the user submits the form.
• The Java backend reads the submitted data and creates a corresponding Task object using predefined class structures.
• The system validates the input (ensuring no field is left blank and that numeric values like duration and priority are valid).
• Once validated, the new task data is saved locally into task.txt using file handling operations to ensure persistent storage.
• The Scheduler module automatically optimizes the updated list of tasks based on urgency and importance.
• The user is then redirected to the Task List View, where the updated list, including the newly added task, is displayed for confirmation.

CHAPTER 4. RESULTS ANALYSIS AND VALIDATION

The solution was implemented using modern tools and technologies at each stage of the software development lifecycle. Emphasis was placed on efficiency, maintainability, and structured modular design to ensure smooth functionality and scalability. The development process followed a clear sequence—from requirement analysis to design, implementation, testing, and validation—ensuring that each component of the Smart Pomodoro Scheduler was built to align with real-world productivity and time management needs.

1. Analysis

   • File Handling and Debugging Tools (IntelliJ IDEA / Eclipse): Used to write, test, and debug Java code efficiently, including file I/O operations for task storage and retrieval.

   • Performance Profiling (JProfiler / VisualVM): Applied to monitor memory usage, detect performance bottlenecks, and analyze runtime efficiency during Pomodoro timer execution.

   • Execution Logs (via Console Output): Provided insight into the task flow, schedule optimization steps, and debugging information for timer cycles.

   • Time Complexity Evaluation: Manual and automated evaluation of algorithms (scheduling, timer loops) to ensure optimal performance and minimal CPU load.

2. Design Drawings / Schematics / Architecture

   • draw.io / Lucidchart: Used to design the system architecture diagrams, flowcharts, and functional workflows for the Smart Pomodoro Scheduler.

   • PlantUML: Created class diagrams and sequence diagrams to illustrate interactions between modules (Task, Scheduler, Pomodoro Timer).

   • Gantt Chart (via Matplotlib): Represented the project timeline for various development phases.

   • Graphviz: Used to generate the system flowchart, showing the overall program flow from task creation to Pomodoro execution.

3. Report Preparation

   • Microsoft Word / Google Docs: Used for documenting system design, methodology, and test results for the project report.

   • Markdown (.md files): Maintained structured project documentation and developer notes within the repository.

   • Canva / Figma: Used to design GUI mockups and visualize the frontend interface.

   • Screenshots & Logs: Collected output screenshots and session logs for inclusion in the final report and presentation.

4. Project Management and Communication

   • IntelliJ IDEA / Eclipse: Used as the primary development environment for coding in Java and integrating frontend technologies (HTML, CSS, JS).

   • Git + GitHub: Used for version control, code backup, and collaboration among team members.

   • Trello / Notion: Managed project workflow, assigned tasks, and tracked milestones throughout development.

   • Google Meet / Email: Facilitated team discussions, code reviews, and regular project updates.

5. Testing / Characterization / Data Validation

   • Manual Unit Testing: Each module (Task, Scheduler, PomodoroTimer) was tested individually to ensure logical accuracy and expected behavior.

   • Integration Testing: Verified data flow between components—ensuring task details are correctly read, scheduled, and reflected during Pomodoro sessions.

   • Validation Scripts: Checked for invalid user inputs such as empty fields, negative durations, or incorrect date formats.

   • Console Assertions & Logs: Used to trace execution flow and verify correct file read/write operations.

   • End-to-End Testing: Simulated full user sessions, from adding tasks to completing Pomodoro cycles, to confirm stability and functionality.

CHAPTER 5.
CONCLUSION AND FUTURE WORK

5.1. Conclusion

The Smart Pomodoro Scheduler project aimed to create a user-friendly and efficient productivity management system that helps users organize their tasks, optimize their daily schedules, and maintain focus using the Pomodoro technique. Developed using Java for the backend logic and HTML, CSS, and JavaScript for the frontend interface, the system successfully enables users to add, view, and manage tasks, automatically prioritize them based on urgency and importance, and execute focus sessions with structured work and break intervals. The expected outcome was a functional and interactive application capable of enhancing user productivity and time management through automation and simplicity.

During implementation, the core functionalities such as task creation, file-based data persistence, scheduling optimization, and timer operation worked as planned. However, some deviations were noted — including limited real-time synchronization between the frontend and backend, challenges in ensuring consistent UI responsiveness across devices, and the lack of database connectivity for multi-user data sharing. These constraints arose primarily due to the project's offline-first design and its focus on minimal configuration and lightweight operation.

Despite these limitations, the system successfully achieved its primary objectives of improving focus, automating task management, and maintaining an organized workflow, demonstrating how algorithmic scheduling and structured time management can significantly enhance personal productivity.

5.2. Future work

There is significant scope to enhance and expand the functionality of the Smart Pomodoro Scheduler in future versions. The following improvements and extensions are suggested to make the system more powerful, user-friendly, and adaptable:

- **Database Integration:** Replace local file storage with a structured database system (e.g., MySQL or SQLite) for efficient data management and scalability across multiple users.

- **User Authentication:** Introduce a login system to allow personalized schedules, progress tracking, and multi-user access with different roles (Admin, User).

- **Graphical Pomodoro Visualization:** Enhance the interface with progress bars, animated timers, and task tracking charts using JavaScript or graphical libraries like Chart.js or D3.js.

- **Cloud Synchronization:** Integrate cloud storage or APIs (e.g., Google Drive or Firebase) to enable users to back up and sync their tasks across devices.

- **Cross-Platform Application:** Extend the project into a desktop and mobile-compatible version using frameworks like Electron or Flutter for broader accessibility.

- **Notifications and Alerts:** Add system notifications, sound alerts, or vibration feedback for session completions and upcoming tasks.

- **Analytics Dashboard:** Develop a productivity dashboard that displays user statistics such as completed Pomodoros, average focus time, and most frequent task categories.

- **AI-Powered Scheduling:** Implement AI algorithms to automatically adjust task priorities and durations based on user habits and performance data.

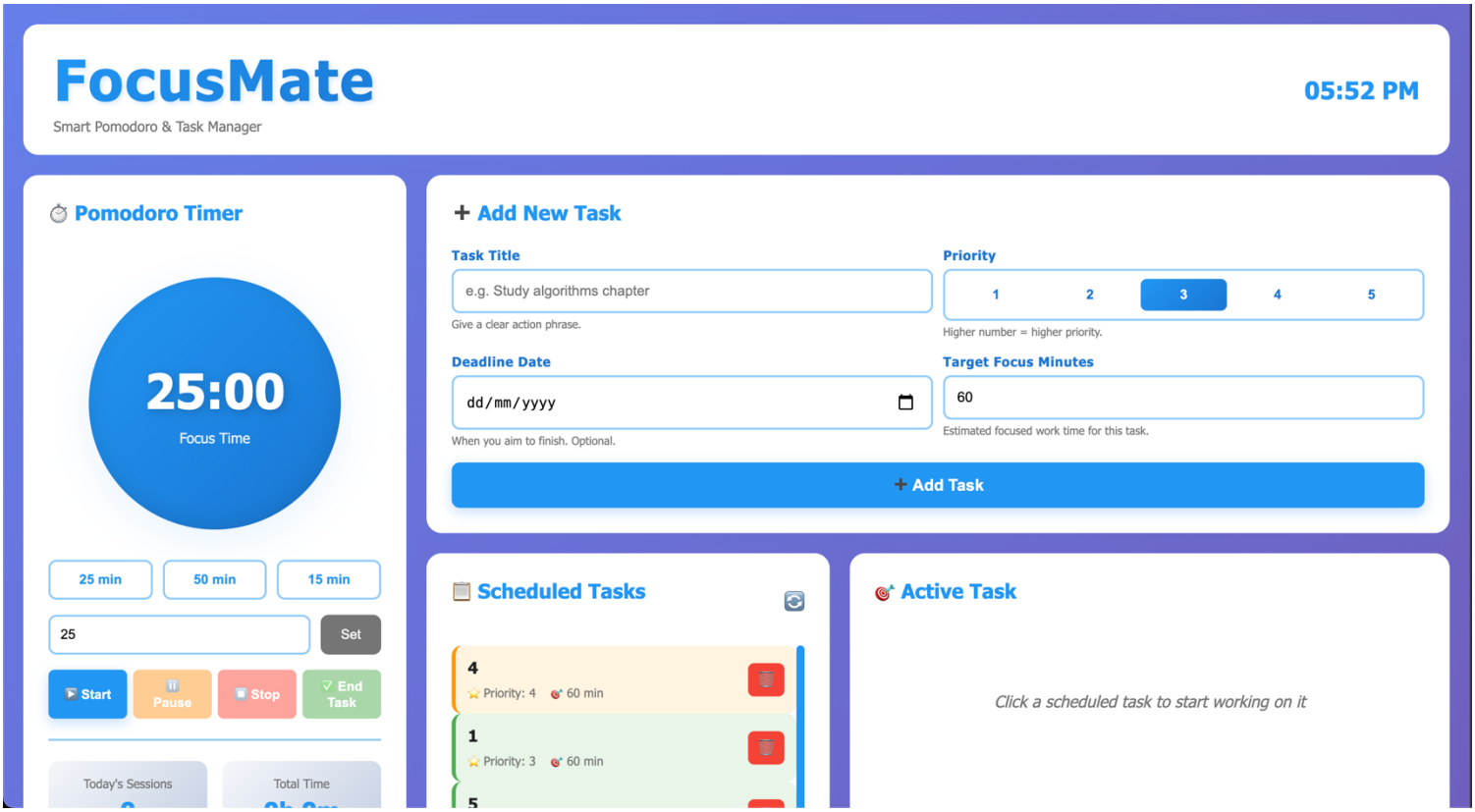
- **Multilingual Interface:** Include support for multiple languages to make the scheduler accessible to a wider range of users globally.

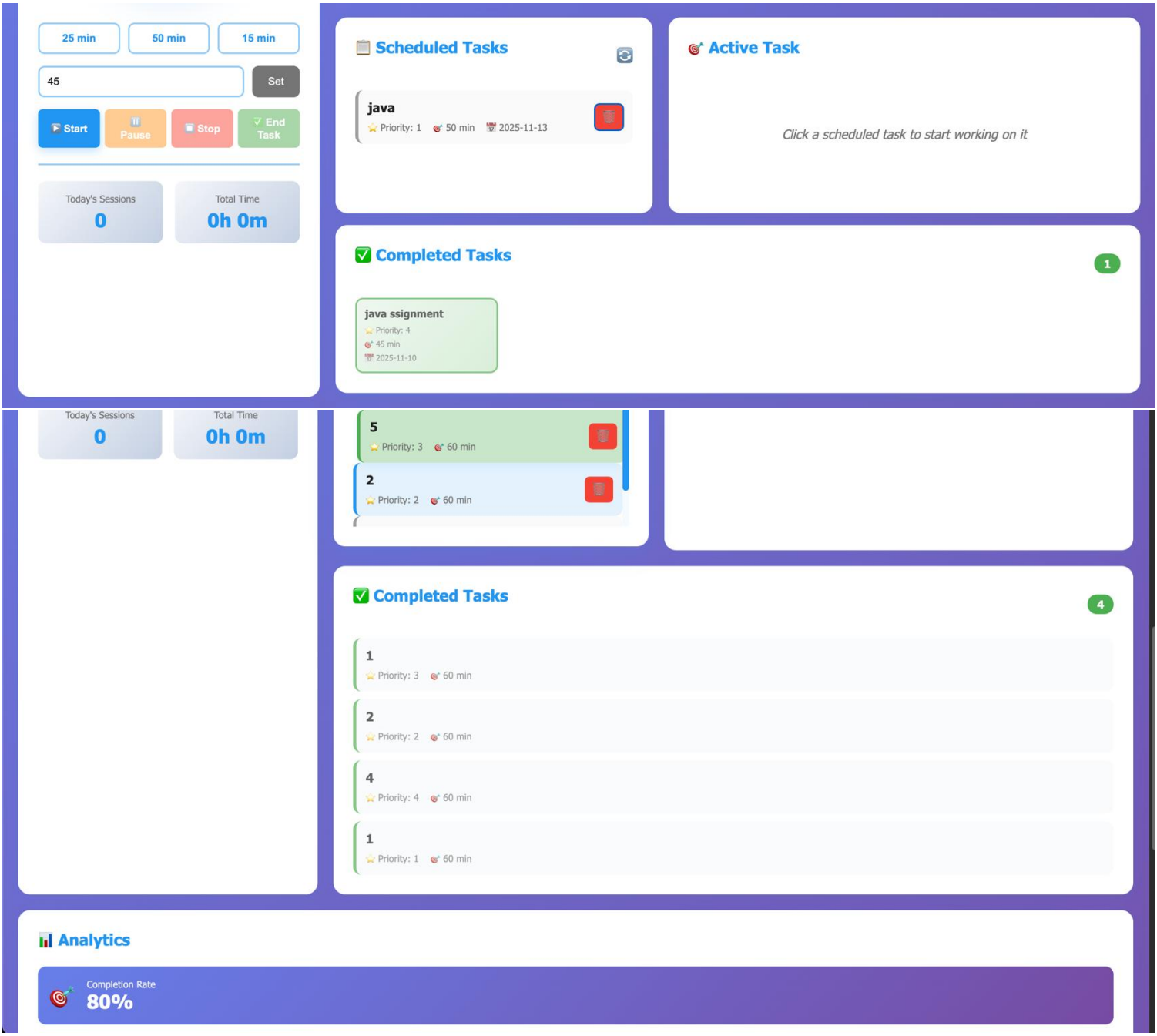
By incorporating these enhancements, the **Smart Pomodoro Scheduler** can evolve into a comprehensive and intelligent productivity management system, helping users optimize their focus, manage time effectively, and achieve higher efficiency in both academic and professional settings.

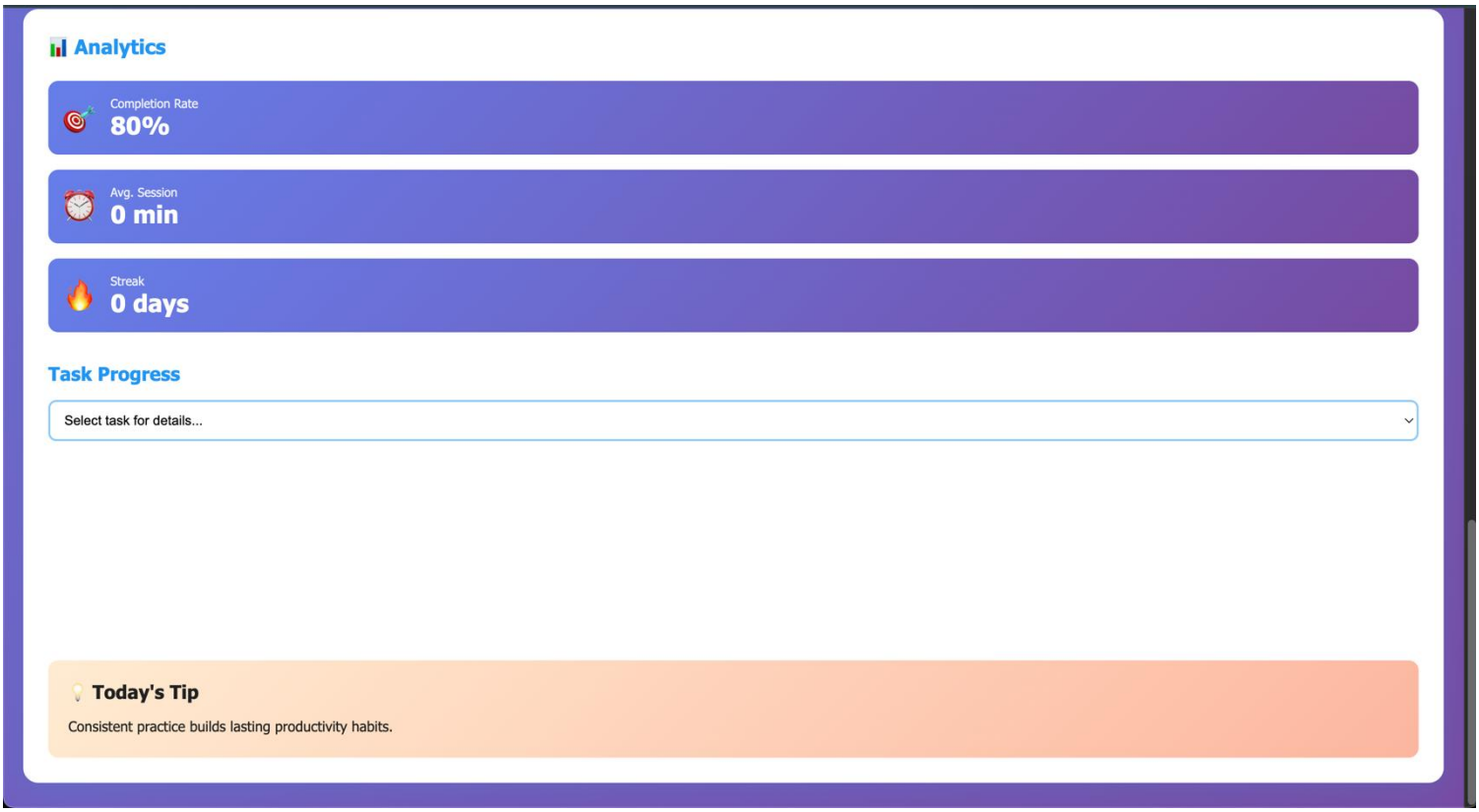
REFERENCES

1. https://docs.spring.io/spring-boot/docs/current/reference/html/
2. https://dev.mysql.com/doc/refman/8.0/en/
3. https://www.eclipse.org/downloads/

USER MANUAL


FocusMate
Smart Pomodoro & Task Manager
05:52 PM
Pomodoro Timer
25:00
Focus Time
25 min
50 min
15 min
25
Set
Start
Pause
Stop
End Task
Today's Sessions
Total Time
Add New Task
Task Title
e.g. Study algorithms chapter
Give a clear action phrase.
Priority
1
2
3
4
5
Higher number = higher priority.
Deadline Date
dd/mm/yyyy
When you aim to finish. Optional.
Target Focus Minutes
60
Estimated focused work time for this task.
Add Task
Scheduled Tasks
4
Priority: 4
60 min
1
Priority: 3
60 min
5
Active Task
Click a scheduled task to start working on it


25 min
50 min
15 min
45
Set
Start
Pause
Stop
End Task
Today's Sessions
0
Total Time
0h 0m
Scheduled Tasks
java
Priority: 1
50 min
2025-11-13
Active Task
Click a scheduled task to start working on it
Completed Tasks
1
java ssignment
Priority: 4
45 min
2025-11-10
Today's Sessions
0
Total Time
0h 0m
5
Priority: 3
60 min
2
Priority: 2
60 min
Completed Tasks
4
1
Priority: 3
60 min
2
Priority: 2
60 min
4
Priority: 4
60 min
1
Priority: 1
60 min
Analytics
Completion Rate
80%


Analytics
Completion Rate
80%
Avg. Session
0 min
Streak
0 days
Task Progress
Select task for details...
Today's Tip
Consistent practice builds lasting productivity habits.