Master of Technology in Artificial Intelligence Systems

PROJECT REPORT

AI-Powered Scheduling System

Group 10		
Name	**Student ID**	**Email**
Jin Keyi	e1133134@u.nus.edu	A0276819L
Ko Hung-Chi	e1539175@u.nus.edu	A0327344E
Sun Yuchen	e1538079@u.nus.edu	A0326248B
Zhang Yuxuan	e1216649@u.nus.edu	A0285664N
Zhao Jiahui	e1554179@u.nus.edu	A0329852U

Contents

1 Executive Summary

The AI-Powered Scheduling System is an intelligent, human-centered productivity platform designed to optimize time management through adaptive task planning and personalized recommendations. Unlike traditional calendar or to-do list applications that rely solely on manual input, this system integrates artificial intelligence—specifically neural network-based task evaluation and rule-based adaptive scheduling—to dynamically generate optimized daily plans. It accounts for individual user factors such as task difficulty, energy levels, stress, and historical performance to balance productivity and well-being.

The system operates through a dual-model architecture: a **Task Evaluation Model** that predicts energy expenditure and psychological pressure based on task features and historical context, and a **Scheduler** that automatically allocates tasks and rest periods in a feasible timetable. By leveraging intelligent reasoning and learning from user feedback, it supports real-time adaptation to unexpected events, shifting priorities, and personal rhythms.

Through user surveys and competitor analysis, the project identified a growing demand for personalized, adaptive scheduling tools—particularly among students and professionals balancing diverse responsibilities. The developed system not only demonstrates the feasibility of integrating reasoning algorithms into daily task management but also contributes to research in intelligent decision support, adaptive human-AI interaction, and behavioral modeling.

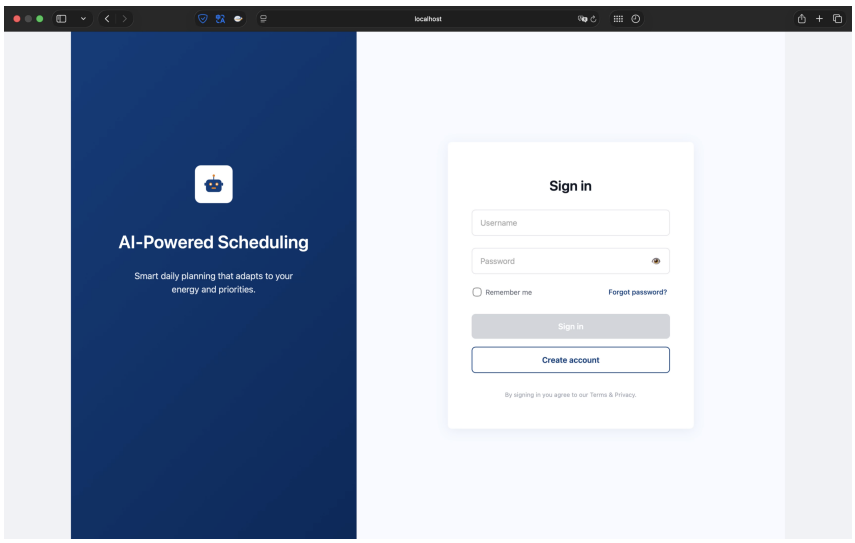

Figure 1: AI-Powered Scheduling System

2 Problem Description

Effective time management is a persistent challenge in the modern era, where individuals are increasingly burdened by overlapping professional, academic, and personal demands. Traditional task management tools—such as calendars and to-do lists—provide static interfaces that rely heavily on user discipline and manual updates. They lack contextual awareness and cannot adapt dy-

namically to changes in user workload, energy, or motivation. This often leads to inefficiency, missed deadlines, and heightened stress.

The problem lies in the absence of intelligent reasoning and adaptive personalization in existing scheduling tools. Most applications fail to interpret a user's behavioral patterns or optimize task order based on fluctuating mental and physical states. Furthermore, users frequently experience decision paralysis—a state in which they have available time but cannot decide what to prioritize—resulting in unproductive downtime.

To address these limitations, this project introduces an AI-Powered scheduling system that leverages data-driven task evaluation and reasoning-based scheduling. The system learns from user history—including completion rates, fatigue patterns, and time usage—to produce personalized daily plans. By balancing workload and recovery periods, it supports sustainable productivity and mental well-being.

From a broader perspective, the system contributes to ongoing research in human-centered artificial intelligence and intelligent decision-making systems. It exemplifies how computational reasoning can enhance everyday productivity tools, making them more adaptive, context-aware, and aligned with human cognitive and emotional needs.

3 Solutions

Our **AI-Powered Scheduling System** is designed to provide personalized, adaptive, and intelligent time management. It integrates artificial intelligence reasoning and user-centered design to help individuals plan tasks efficiently while maintaining well-being.

The system is composed of two major parts:

1. **Front-End Interface**

    - Features an intuitive weekly timetable and task board for visualizing, adding, and editing both fixed and flexible tasks through an interactive calendar view.
    - Integrates an AI chatbot that understands natural language commands, enabling users to create, update, or delete tasks conversationally.
    - Supports real-time feedback, task-status transitions, and adaptive rescheduling through intelligent scheduler execution.
    - Provides analytical dashboards (daily and weekly reports) with dynamic charts that visualize productivity, energy, and stress trends.

2. **Back-End Intelligence**
   The back-end intelligence comprises two key reasoning modules:

    - **Task Evaluation Model** – a neural network that predicts each task's energy cost and pressure level based on user data and task history. It introduces a Historical Impact

Weight (HIW) mechanism to model how past tasks influence current performance.

- **Scheduler** – a rule-based engine that arranges tasks into a feasible timetable by prioritizing deadlines and user-defined importance. It ensures workload balance and automatically allocates rest periods to prevent fatigue.

The system achieves several functions:

1. **Intelligent Task Ranking**: Tasks are evaluated and ranked according to urgency, priority, and predicted energy/pressure levels.
2. **Adaptive Scheduling**: Automatically assigns tasks to optimal time slots, considering fixed appointments and dynamic updates.
3. **Rest Time Optimization**: Determines ideal rest intervals using an evolutionary algorithm to enhance recovery and long-term efficiency.
4. **Natural Language Interaction**: (Optional LLM integration) Users can describe tasks in plain language, and the system extracts relevant parameters (e.g., deadline, duration).
5. **Personalized Learning**: The system refines its scheduling strategy based on user feedback and past behavior patterns.

3.1 Task Evaluation Model

The **Task Evaluation Model** serves as the system's predictive intelligence component. Its primary objective is to estimate how each task influences a user's energy consumption and mental pressure, thereby enabling intelligent scheduling and ranking based on predicted physical and cognitive load.

The model takes as input structured task attributes—such as type, duration, and difficulty—and outputs two key indicators:

$$(\Delta E, \Delta P) = f(\text{type}, \text{duration}, \text{difficulty})$$

where ΔE denotes the predicted change in energy (energy loss), and ΔP represents the predicted change in psychological pressure (pressure increase).

3.1.1 Data Preparation

To train the predictive model, two primary datasets are used:

- **stats_check.csv** — continuous biometric logs collected throughout the day, recording each user's **Energy** and **Pressure** levels at specific timestamps.
- **schedule.csv** — task logs detailing each activity's user, task type, start/end time, and difficulty rating.

A dedicated preprocessing script (**data_preprocessing.py**) combines these two sources through time-aligned interpolation. For each task instance, the system:

1. Locates the user's Energy and Pressure data on the same date.
2. Interpolates the values at both the task's start and end timestamps.
3. Computes the deltas:

$$\Delta E = E_{end} - E_{start}, \quad \Delta P = P_{end} - P_{start}$$

These deltas represent the physical and cognitive effects associated with performing the task. The preprocessing procedure robustly handles edge cases such as:

- Missing or incomplete biometric records,
- Single-point or sparse daily logs,
- Cross-midnight or invalid time ranges.

The resulting dataset (**new_tasks_df.csv**) forms a structured learning table, where each task is paired with its observed energy and pressure variations.

3.1.2 Feature Engineering

Before model training, several features are extracted and numerically encoded:

Feature	Description	Type
Task type	Encoded categorical label (via LabelEncoder)	Categorical (int)
Difficulty	User-defined perceived difficulty (0–5 scale)	Numeric
Duration	Computed task duration (in minutes)	Numeric

Additional derived labels include:

- **energy_loss** — difference in Energy between task end and start,
- **pressure_increase** — difference in Pressure between task end and start.

3.1.3 Model Training and Evaluation

Two independent regression models are trained to predict Energy Loss and Pressure Increase respectively. Both use the same input features `[type, difficulty, duration]`.

Model Choice. A **Random Forest Regressor** is adopted for both targets due to its robustness to noise and missing data, nonlinear feature interaction handling, and interpretability through feature importance analysis.

Training Process.

1. The dataset is split 80/20 into training and test sets.
2. Separate models are trained for each target:

$$f_E(x) \to \widehat{\Delta E}, \quad f_P(x) \to \widehat{\Delta P}$$

3. Model performance is evaluated using:
   - MAE (Mean Absolute Error)
   - MSE (Mean Squared Error)
   - R^2 Score (Goodness of fit)

Evaluation output:

```
=== Energy Loss Prediction ===
MAE: 0.6154976455054124
MSE: 0.7161091879304533
R²: 0.05477522739025942

=== Pressure Increase Prediction ===
MAE: 0.6326830109599901
MSE: 0.7067960090389993
R²: -0.13040842378934858
```

Both models are then persisted as serialized `.pkl` files using `joblib`:

- `energy_loss_model.pkl`
- `pressure_increase_model.pkl`
- `task_label_encoder.pkl`

3.1.4 Task Scoring and Application

Once trained, the model is applied to new or upcoming tasks (from `new_tasks_df.csv`) to predict:

$$\text{Predicted Load} = \{\widehat{\Delta E}, \widehat{\Delta P}\}$$

These predicted values are subsequently integrated into the Scheduler and Task Ranking modules to:

- Prioritize low-fatigue time slots for high-energy tasks,
- Avoid consecutive high-pressure activities,
- Recommend rest intervals when cumulative predicted load exceeds a threshold.

Through this integration, the system achieves both predictive awareness and adaptive scheduling, ensuring an optimal balance between productivity and user well-being.

3.2 Scheduler

The **Scheduler** is the central reasoning engine of the AI-powered Scheduling System. Its main objective is to transform a collection of user tasks—both fixed and flexible—into a feasible, adaptive, and optimized timetable that balances productivity with well-being. It embodies the system's intelligent reasoning capability, enabling real-time adaptation to user behavior, task constraints, and time availability.

The design of the Scheduler integrates **rule-based logic**, **heuristic optimization**, and **data-driven evaluation**. It not only arranges tasks efficiently but also ensures the user receives adequate rest, minimizing cognitive fatigue and maintaining consistent performance throughout the day. The workflow of the scheduler is shown in Figure 2.

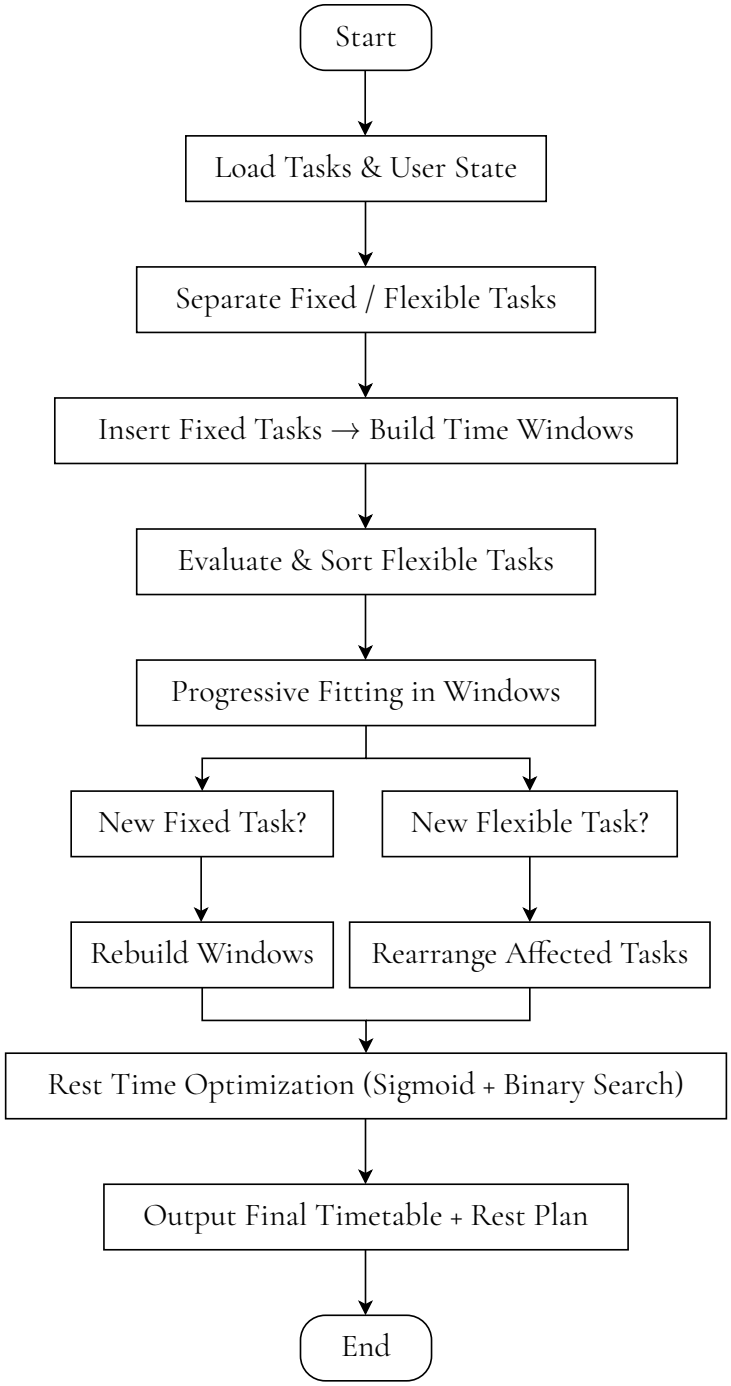

Figure 2: Scheduler Workflow

3.2.1 Core Architecture and Logic

1. **Task Categorization**

   All incoming tasks are divided into two primary types:

   - **Fixed Tasks**: Events that have immutable start and end times (e.g., meetings, classes).
   - **Flexible Tasks**: Tasks with adjustable time windows that the system can optimize (e.g.,

studying, project work).

This distinction allows the Scheduler to maintain temporal integrity (fixed commitments first) while using flexible intervals to adaptively optimize user time.

2. **Scheduling Pipeline**

   The Scheduler operates through a structured multi-stage process:

   - **Initialization**: All fixed tasks are inserted into the timeline in chronological order. The system identifies available windows between fixed tasks.
   - **Task Sorting**: Flexible tasks are sorted by earliest deadline, priority, and expected duration. This ensures time-critical and high-importance tasks are handled first.
   - **Progressive Fitting**: For each flexible task, the Scheduler traverses available windows sequentially. It checks three constraints: **1.** The task cannot be scheduled before the current time. **2.** It must finish before its deadline. **3.** It must not exceed the available duration window (with a 25% buffer margin to absorb unexpected changes). Tasks violating any constraint are skipped or reported as unscheduled.
   - **Conflict Handling and Reallocation**: If inserting a new fixed task disrupts existing ones, the Scheduler automatically extracts affected tasks, rebuilds free windows, and reinserts displaced tasks using the same priority-based fitting logic. This guarantees adaptive reorganization without manual intervention.

3.2.2 Rest Time Distribution

To ensure sustainable performance, the Scheduler allocates **optimal rest intervals** between tasks:

1. The system uses a **Sigmoid rest-feedback function** to model the diminishing returns of rest time (i.e., a short rest boosts performance sharply, but excessive rest yields less benefit).
2. Using Lagrange multiplier optimization and binary search, the Scheduler distributes the available rest budget among multiple rest slots such that:

   $$\sum_i r_i(\lambda) = R_{total}$$

   where r_i represents the rest time allocated to slot i under a global rest-time constraint R_{total}.
3. The method guarantees convergence and fairness among all rest periods, maximizing overall energy recovery and stress reduction.

This process mirrors a **water-filling optimization** approach used in resource allocation problems.

3.2.3 Historical Impact Integration

To make scheduling context-aware, the Scheduler considers the historical influence of previous tasks:

1. It computes **task similarity** based on attribute vectors (type, difficulty, duration, etc.) using **cosine similarity**.
2. The **Historical Impact Weight (HIW)** mechanism then adjusts the predicted energy and pressure of a task based on residual fatigue from similar prior tasks:

$$\text{Adjusted Energy}_t = E_t + \text{Similarity} \times E_{t-1}$$

This approach ensures that performing several similar tasks consecutively increases cumulative fatigue, encouraging more balanced task sequences.

3.2.4 Computational Efficiency

1. Task Sorting: $O(N \log N)$
2. Task Placement: $O(N \times W)$, where W is the number of windows.
3. Rest Optimization: $O(K \log(1/\epsilon))$, where K is the number of rest slots and ϵ is the convergence tolerance.

Overall, the algorithm achieves practical real-time performance for individual users while maintaining theoretical rigor in optimization.

3.3 System Development

The system development can be divided into three main parts: the **database construction**, the **backend development** and **frontend design**.

3.3.1 Database

Our system uses **MongoDB** as the core database for storing users, tasks, and scheduling information. As a **NoSQL document database**, MongoDB organizes data in a flexible **JSON-like structure**, allowing different users' tasks and records to have dynamic fields such as deadlines, priorities, and predicted energy or stress levels. This flexibility is essential for an adaptive AI scheduling system, where task attributes and user data frequently evolve.

We chose MongoDB because it offers high scalability, schema flexibility, and fast query performance. Its JSON/BSON format integrates seamlessly with our Python-based backend, enabling

efficient data exchange between the AI reasoning modules and the web interface. Moreover, MongoDB's real-time update and indexing features allow the scheduler to react immediately to user input and changes, ensuring smooth, responsive performance even under dynamic workloads.

For our system, we build a database with three collections: users, fixed-tasks and flexible-tasks. The users collection stores user account information, including fields such as: { user_id, user_name, password }. The fixed-tasks collection stores each user's fixed-time task records, with fields: { user_id, task_name, task_type, task_start_time, task_end_time, task_duration, expected_difficulty, task_location, status, created_at }. The flexible-tasks collection stores user tasks that can be rescheduled within a flexible time window, in contrast to the strictly timed tasks in fixed-tasks. Compared to fixed-tasks collection, flexible-tasks adds adaptive and predictive attributes (predicted_energy, predicted_pressure, task_priority), but doesn't has explicit start time or end time.

3.3.2 Backend

The backend of our system is implemented using **FastAPI**, which provides strong support for data validation, dependency injection, and asynchronous I/O, which makes it particularly well-suited for managing real-time interactions between the user interface and the AI scheduling engine.

The backend architecture follows a modular and service-oriented design, organized into distinct router modules under the `routers` directory. Each router is responsible for a specific subsystem of the application, ensuring high maintainability, scalability, and clarity in the overall system structure. This modular organization allows independent development and updates to individual modules without affecting the stability of the entire system.

The major router modules are summarized as follows:

- **login.py** – Handles user authentication and session management, including login and credential verification.
- **users.py** – Manages user-related operations such as registration and profile retrieval
- **tasks.py** – Supports CRUD (Create, Read, Update, Delete) operations for both fixed and flexible tasks. It interacts directly with MongoDB to manage user task data efficiently.
- **scheduler.py** – Implements the scheduling logic that determines optimal time slots for flexible tasks based on user availability, task priority, and AI-predicted stress or energy levels.
- **stats_recording.py** – Collects and records user activity and performance statistics, supporting adaptive learning and system optimization.
- **chatbot.py** – Integrates with the OpenAI API to provide an intelligent conversational interface. Through this module, users can interact with the chatbot.

In addition to the router modules, the backend also includes a **utils** package containing supporting scripts such as **scheduler_updated.py**, which defines the scheduler class.

All API endpoints are defined using FastAPI's routing system and return data in JSON format

to the frontend. The backend manages real-time communication with MongoDB, allowing instant data synchronization when users create, modify, or delete tasks. The combination of FastAPI's asynchronous processing and MongoDB's dynamic schema design ensures efficient, low-latency system performance, even under heavy user interaction and concurrent task updates.

Overall, the backend serves as the system's computational and integration core. It coordinates user input, AI reasoning, and data persistence, forming the foundation of the adaptive and intelligent scheduling experience provided by the platform.

3.3.3 Frontend (System Introduction)

1. **Login**

   At the login page, users should type in their username and password, then click the "Sign in" button to login. The system will check if the username and password is correct, if correct, the system will redirect the page to the main timetable, as shown in Figure 3, otherwise it will show the problem to the user, as shown in Figure 4 & 5.

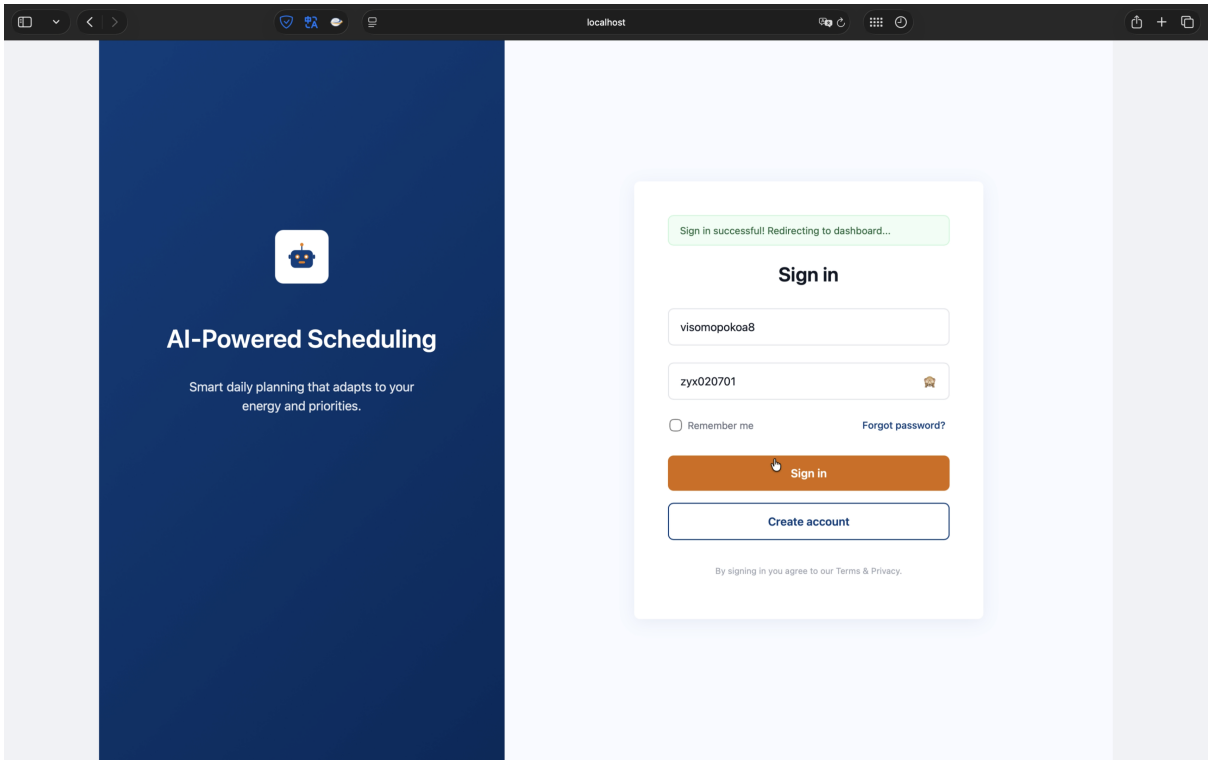

Figure 3: Login Successfully

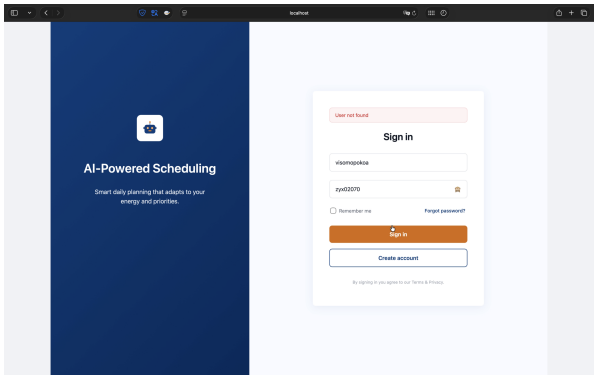

Figure 4: User Not Found

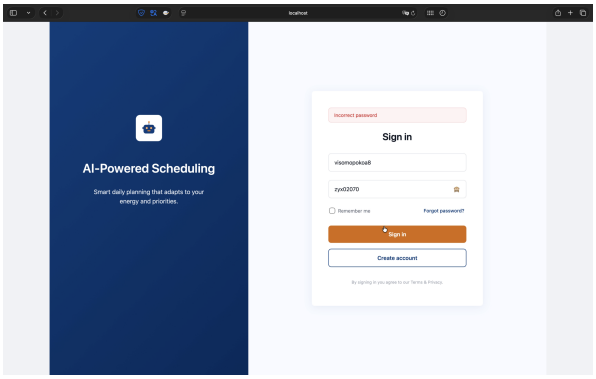

Figure 5: Incorrect Password

2. **Timetable**

After signing in, the first view is the timetable, as Figure 6, which will list all the registered tasks in it. As show in Figure 7 to 14, clicking on the white space, the users can add new tasks, clicking on the task block, the users can edit the tasks and even delete them.

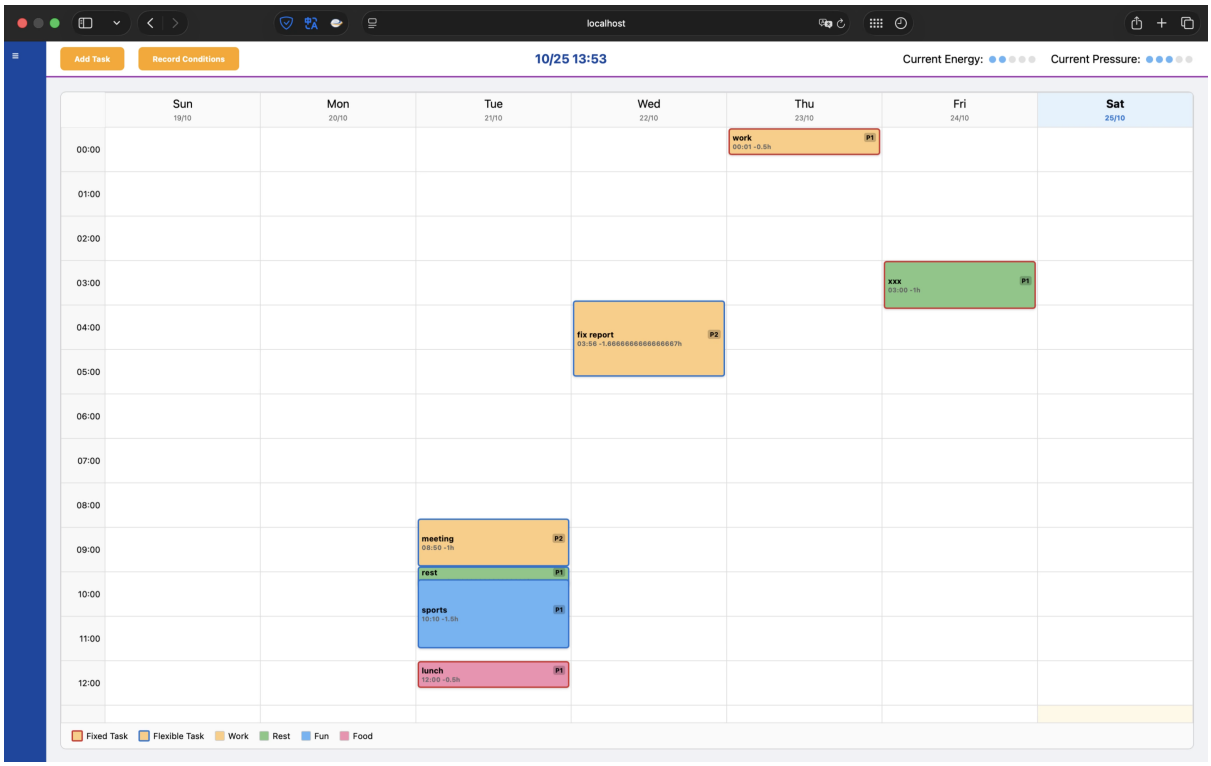

Figure 6: Timetable

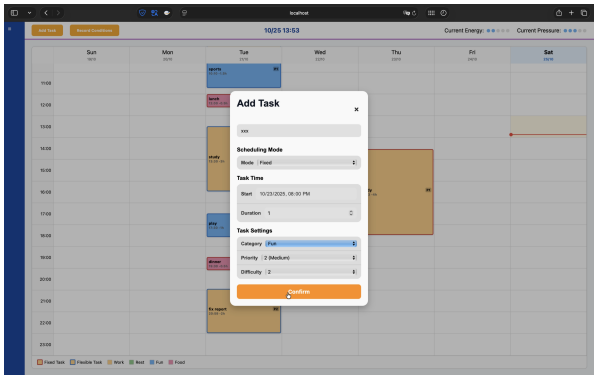

Figure 7: Add Tasks

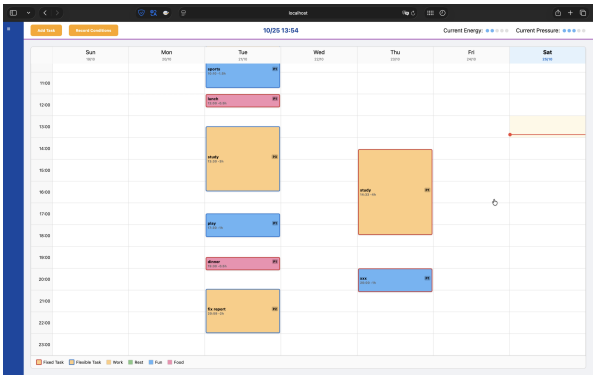

Figure 8: After Adding Tasks

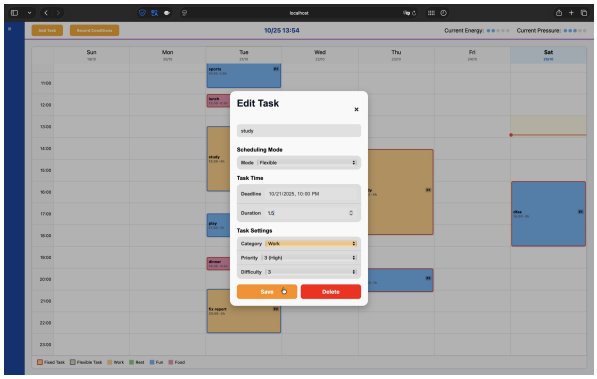

Figure 9: Edit Duration of Tasks

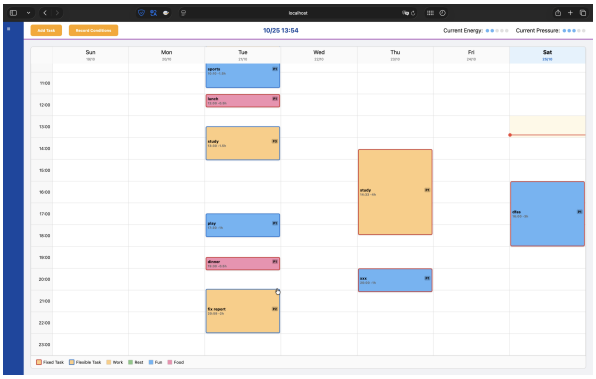

Figure 10: After Editing Duration

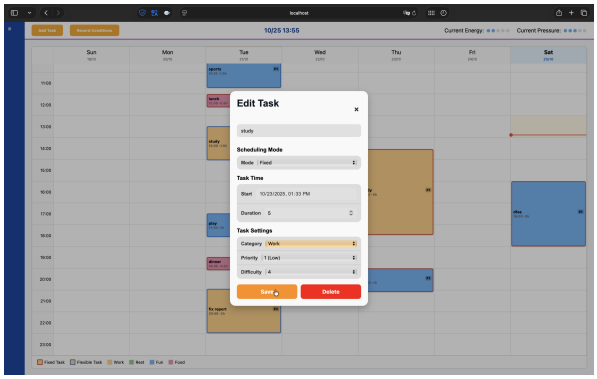

Figure 11: Edit Start Time of Tasks

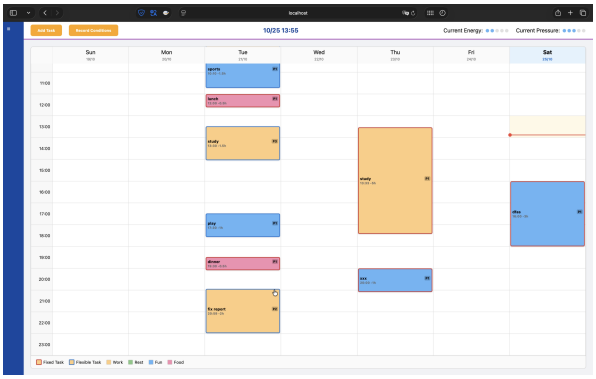

Figure 12: After Editing Start Time

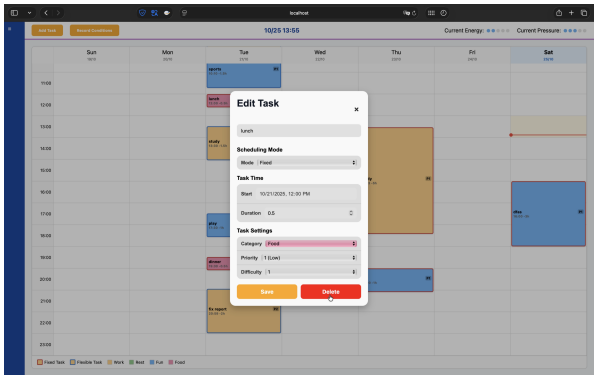

Figure 13: Delete Tasks

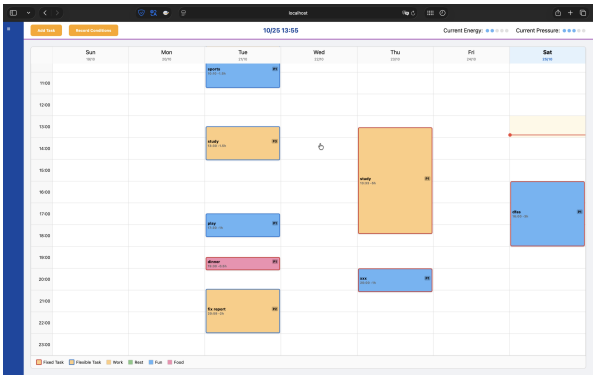

Figure 14: After Deleting

3. **AI Scheduling**

   xxx

4. **Intelligent Report**

   The **Intelligent Report** module provides an analytical visualization of the user's daily and weekly performance, bridging the reasoning engine with interpretable insights. It is designed not only to display completed tasks and statistics but also to reflect patterns in user behavior, energy consumption, and psychological pressure trends over time. By combining data analytics and adaptive visualization, this component transforms raw scheduling data into actionable self-awareness.

   - **Daily Report**

     As shown in Figure 15, the daily report panel presents an overview of the user's performance on a specific day. It consists of three main sections:

     (a) **Task Status Summary** — Displays total number of tasks, number of completed ones, and the completion rate. This section offers a quantitative reflection of user productivity.

     (b) **Task Type Distribution** — A pie chart categorizing tasks into work, food, rest and fun. This distribution helps the user identify whether their schedule maintains a healthy balance between professional and leisure activities.

     (c) **Energy & Pressure Trends** — Line charts that visualize predicted energy and stress levels throughout the day. These curves are derived from the AI Task Evaluation Model, showing how task arrangement impacts the user's well-being.

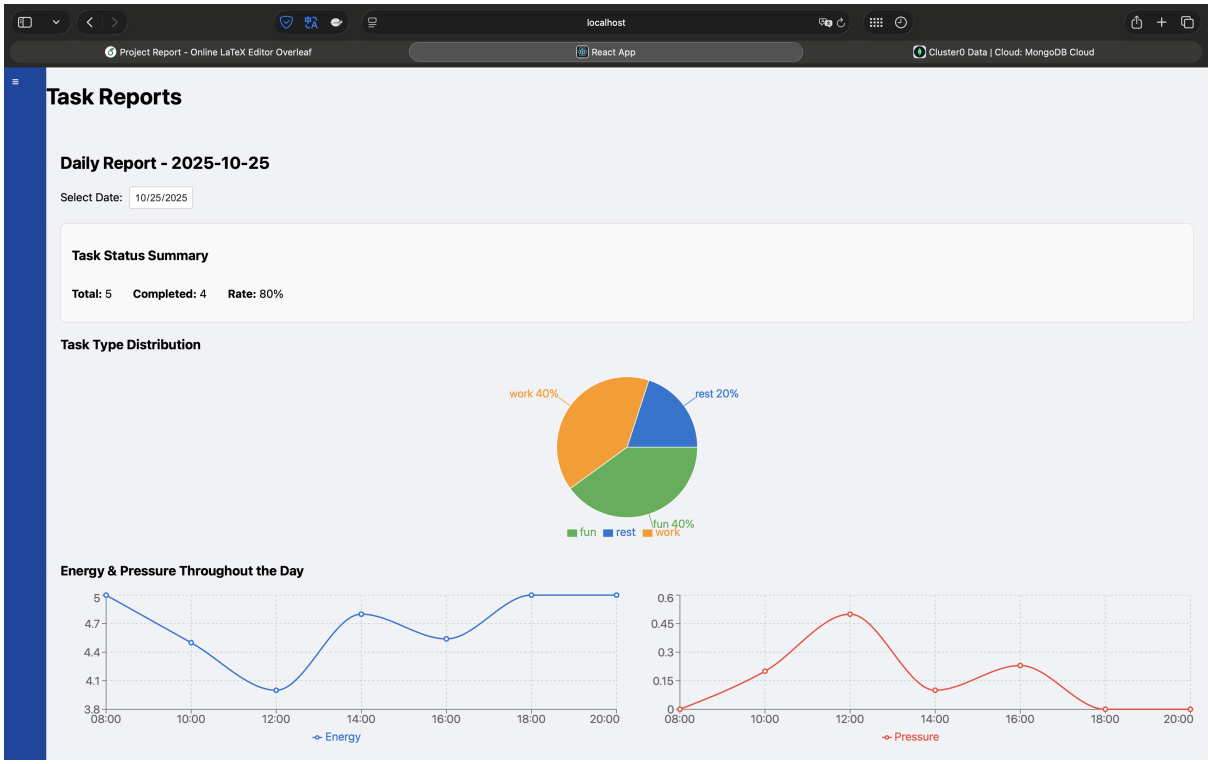

Figure 15: Daily Report Interface

- **Weekly Report**

  The weekly report (Figure 16) provides a longitudinal view of performance over seven days, summarizing total task counts, completion ratios, and changes in energy-pressure patterns. Key features include:

  (a) **Weekly Summary**: Summarizes the total number of tasks and completion rate across the week.

  (b) **Daily Task Count**: A bar chart comparing total tasks per day, revealing workload fluctuations and potential over-scheduling days.

  (c) **Task Type Distribution**: Visualizes proportions of different task types (work, food, fun and rest), indicating the overall lifestyle balance across the week.

  (d) **Energy & Pressure Trends**: Plots average energy and stress over the week, highlighting days with excessive fatigue or pressure peaks.

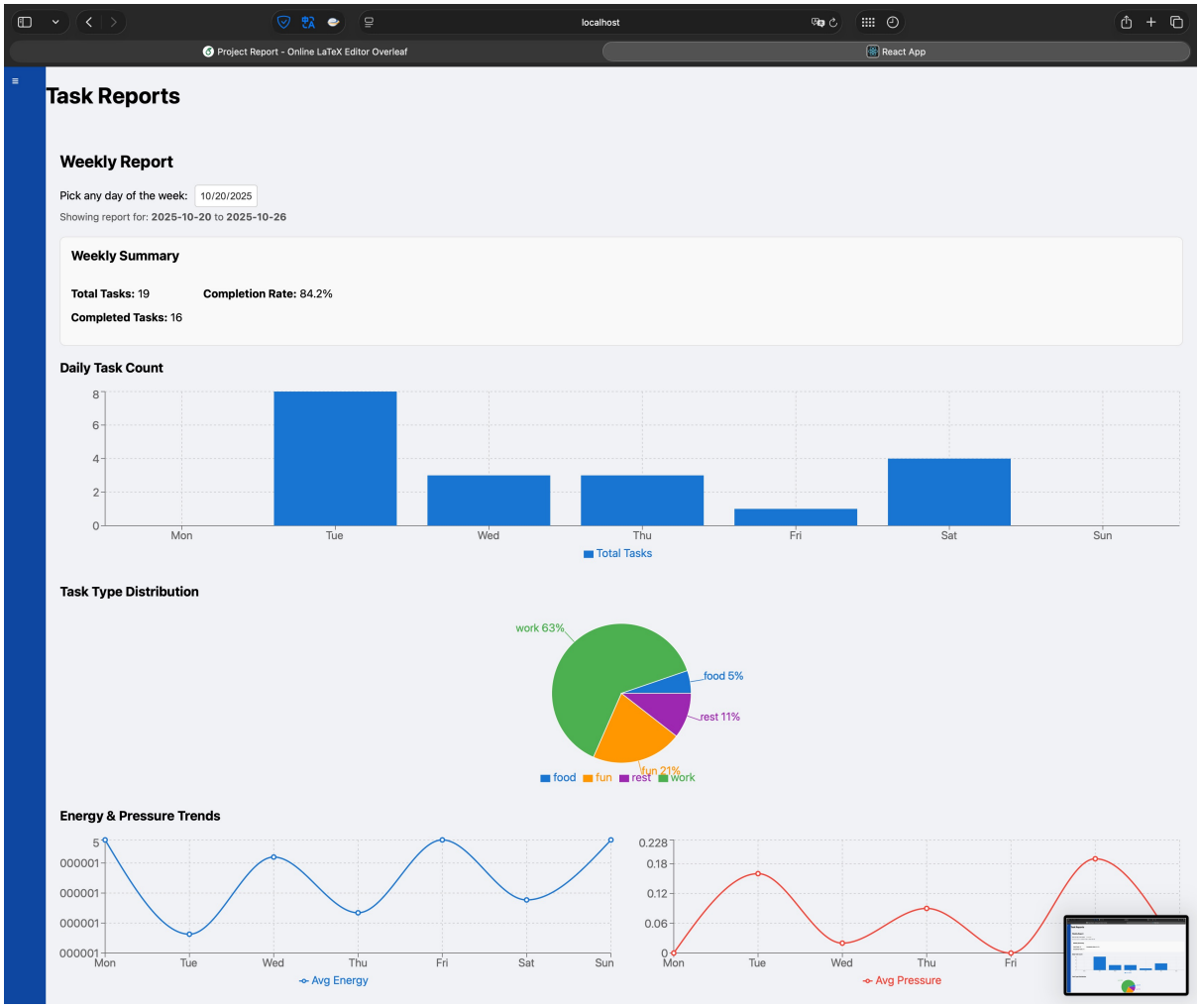

Figure 16: Weekly Report Overview

- **Analytical Insights.**

  By combining daily and weekly reports, users can understand:

  (a) **Productivity Patterns**: How the total workload and completion efficiency evolve over time.

  (b) **Lifestyle Balance**: Whether "work" tasks dominate excessively or leisure activities are underrepresented.

  (c) **Well-being Trends**: How energy depletion and pressure accumulation fluctuate throughout the week.

These visualizations are dynamically generated based on real-time data queried from MongoDB and computed by the backend scheduler. They help users not only to reflect on past performance but also to adjust future schedules. The module demonstrates how AI-driven reasoning can be complemented by intuitive data visualization to support self-management, reflection, and long-term behavioral optimization.

5. **LLM Chatbot**

   The **AI Task Chatbot** serves as a conversational interface that bridges natural language understanding and intelligent task automation. Instead of filling structured forms, users can directly describe their tasks in plain language, and the chatbot automatically interprets, structures, and adds them to the scheduling system. This component demonstrates the integration of Large Language Models (LLMs) with task reasoning, providing a natural, efficient, and human-centered interaction experience.

   - **Natural Language Input**

     As shown in Figure 17, users can simply type a message such as:

     > *"Group meeting, 2025/10/25 at 10pm, 30 minutes, work."*

     The chatbot employs an embedded LLM to parse and extract structured task attributes, including:

     (a) **Task name** ("group meeting")

     (b) **Task mode** (fixed)

     (c) **Start time** (2025-10-25 22:00)

     (d) **Duration** (30 minutes)

     (e) **Task type** (work)

     If the LLM detects that the input explicitly includes a start time, the task is automatically categorized as a **fixed task**. Otherwise, when the description lacks a specific starting time, it is interpreted as a **flexible task** whose scheduling window will later be optimized by the system.

     After determining the task mode, the chatbot performs entity extraction to identify all available attributes (time, duration, type, etc.) and prepares the structured task object for backend processing.

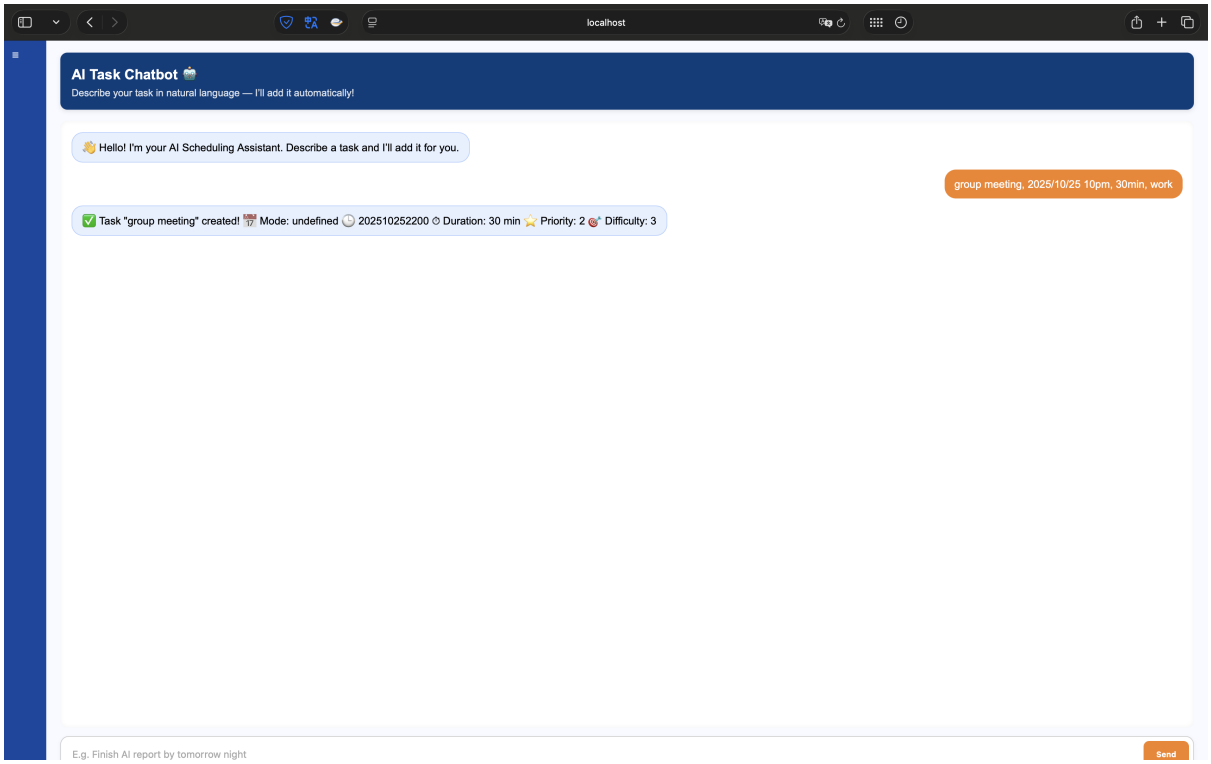

Figure 17: AI Chatbot

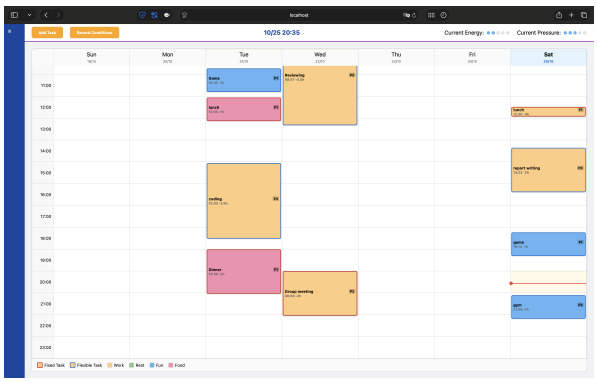

Figure 18: Before AI Assignment

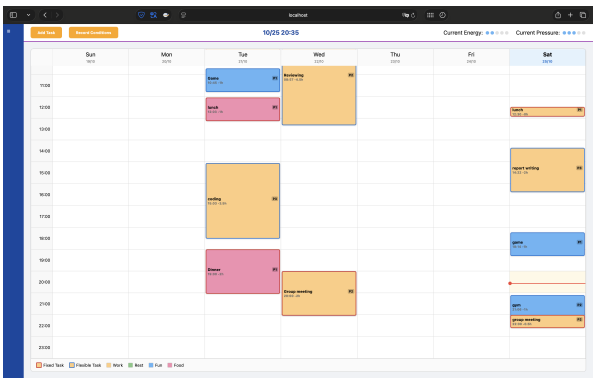

Figure 19: After AI Assignment

- **Task Creation and Confirmation**

  Once the LLM successfully interprets the natural language input, the chatbot generates a structured task object and communicates with the backend API to store it in the MongoDB database. A confirmation message is then returned to the user, listing all parsed parameters (name, duration, time, difficulty, priority) for verification. This allows transparent interaction between users and the system, ensuring accuracy and trust in automated scheduling.

- **Integration with Scheduler**

  After task creation, the system instantly synchronizes with the intelligent scheduling

engine. Figures 18 and 19 illustrate the dynamic scheduling process: the left figure shows the timetable **before** adding the new task, while the right one shows the **updated schedule after** the chatbot's AI-driven assignment. The scheduler automatically re-evaluates available windows and integrates the new task into an optimized time slot—without requiring manual refresh or adjustment.

- **User Experience and Significance**

  This conversational interface demonstrates how modern large language models can enhance usability and accessibility in AI-driven scheduling systems. It lowers cognitive barriers for non-technical users and allows fluid, human-like communication with the system. Beyond convenience, it embodies the project's core philosophy — integrating intelligent reasoning with empathetic human-AI interaction.

4 Conclusion & References

In conclusion, the **AI-Powered Scheduling System** successfully demonstrates how intelligent reasoning and data-driven modeling can be applied to real-world time management. By integrating a neural network–based Task Evaluation Model and a rule-based Scheduler, the system dynamically adapts to users' changing priorities, energy levels, and workloads. It goes beyond traditional calendar tools by offering personalized recommendations, optimized rest-time allocation, and adaptive rescheduling in response to unexpected events.

Through this project, we illustrate the potential of combining **artificial intelligence** and **human-centered design** to improve productivity and well-being simultaneously. The system not only enhances users' ability to plan and execute daily tasks efficiently but also encourages sustainable work habits that prevent burnout. Although challenges such as limited data availability and user subjectivity remain, the framework establishes a solid foundation for future extensions—such as collaborative scheduling, richer behavioral analytics, and integration with wearable or productivity-tracking devices.

Overall, this work contributes both practically and academically to the development of intelligent personal assistants, marking a step toward more adaptive, context-aware, and empathetic AI systems in everyday life.