Politecnico di Milano

December 28th 2025

ThreadWise Planner

Technical Implementation Report
Creativity, Science and Innovation

Emad Karimianshamsabadi Mahdi Soltani Renani Lucas Lescure

I. Motivation and problem statement

As students we find ourselves frequently overwhelmed by the amount of material to study or work to be done, especially during exam periods. Much of these problems can be attributed to poor planning and time management, which can lead to increased stress and reduced productivity. ThreadWise aims to solve these issues by providing a web application to help students plan their study schedules more effectively, the **ThreadWise Planner**.

As opposed to most calendars, this planner will take into account not only the time available, but also the difficulty of the material to be studied, and employ cognitive strategies to optimize learning and retention, like interleaving or blocking. To achieve this, an LLM will be used as a back end component to analyze the study material and generate a personalized study plan based on the user's deadlines and available time.

II. Technical architecture

For the development of the ThreadWise Planner, we wanted to keep the architecture as simple as possible for ensured robustness as well as scalability in case of large number of users. The web application is divided into two main components: the frontend and the backend.

1. Frontend

In order to implement the ThreadWise Planner as a web application, we opted to use Streamlit as a front end framework due to its ease of use, rapid prototyping capabilities and reactive web interface. The interface is also designed to be user-friendly and intuitive, allowing users to easily input their study material, deadlines, and available time slots.

To enhance the user experience, we also used custom HTML and CCS injections to style the application and make it visually appealing for rendered tables as well as use center alignement for these tables as they lack this feature natively.

2. Backend

The backend of the ThreadWise Planner is responsible for processing user inputs, interacting with the LLM, and generating the study plans. We chose to implement the backend using Python 3.9 due to its extensive libraries and ease of integration with LLMs.

We also used Hugging Face to access and interact with the LLM model, as it provides a simple and efficient way to work with various pre-trained models. In this project it was also beneficial as it allowed us to run the model without relying on external APIs. This service does however require a Hugging Face account and an API key for authentication, which for students is free.

The LLM used in this project is Qwen2.5 32B Insruct an open-source model developed by Qwen-Lab, which can be easily integrated into Python as well as well as being lightweight enough to

require relatively low computational resources compared to other models of similar capabilities. This model was chosen for its strong natural language processing capabilities, which are essential for understanding and generating study plans based on user inputs.

Additionally, to ensure that the LLM generates outputs in a structured format that can be easily processed, a strict system prompt was given to ensure the model outputs data solely in a Markdown table format, calculating durations in decimal hours for precise downstream processing.

Since the web application is expected to handle tabulated data and perform various data manipulations, we utilized the Pandas library for efficient data handling and vectorized operations. These operations include sorting, filtering, and aggregating study sessions based on user workload and deadlines.

3. Security layer

To ensure that users do not divert from the intended use of the application, the AI model is applied a set of rules, some of them to get precise tabular data back, but also to avoid the AI from talking about anything unrelated to the intended use. This simply checks if the inputs are relevant to academic planning and prevent other types of inputs that lie outside the scope of the application (such as cooking recipes).

```
system_instruction = (
        "You are Threadwise, an intelligent Study Planning Assistant. "
        "Your goal is to parse messy user inputs into a structured study plan. "
        "\n\n"
        "**DATA EXTRACTION RULES (SMART PARSE):**\n"
        "1. **Free-form Parsing:** Accept ANY input format. Course names can appear with
        or without numbers, separators, or clear delimiters.\n"
        "2. **Extract Courses from Context:** If you see text like 'Wireless internet
        10/01/2026 Multimedia 12/01/2026', parse TWO courses: 'Wireless internet'
        (deadline 10/01/2026) and 'Multimedia' (deadline 12/01/2026).\n"
        "3. **Connect Disconnected Data:** If courses are listed in one place and diffi-
        culties
        in another, connect them logically by position/context.\n"
        "4. **Required Fields:** EVERY course MUST have: Name, Deadline (YYYY-MM-DD),
        Difficulty (Easy/Medium/Hard).\n"
        "5. **Infer Intelligently:** If difficulty is mentioned separately (e.g., a
        list at the end saying 'Hard, Hard, Medium'), map them to courses in order.\n"
        "6. **Output Format:** ONLY return a Markdown Table with columns: Day | Subject
        | Task | Duration | Difficulty.\n"
        "7. **No Conversational Text:** Do not add explanations, just the table.\n"
        "\n\n"
        "**CRITICAL:** Read the ENTIRE prompt and assemble the table from all available
        information."
    )
```

Listing 1. LLM ruleset to get expected tabular output

This security layer is crucial to maintain the reliability and relevance of the generated study plans, and the be sure that the outputs remains what we need them to be, an additional context validation step is also implemented to check if the user inputs are relevant to academic planning before processing them further.

4. Study strategies

As mentioned previously, the ThreadWise Planner features the ability to select different study strategies to optimize learning and retention. This is achieved by implementing sorting algorithms in the backend that rearrange the study tasks based on the selected strategy.

The 4 different startegies that can be choosen from are as follow:

- Waterfall(Cascade): Highest difficulty task are prioritised to fully utilize mental capacity before fatigue sets in.

- Sandwich(Interleaving): Alternates between high and low difficulty tasks to maintain engagement and prevent burnout.
- Sequential(Focus): Groups similar difficulty tasks together to build momentum and focus on finishing a subject entirely before continuing.
- Random Mix: Tasks are assigned randomly to introduce variety and reduce monotony.

This was the trickiest part of the implementation as it required careful algorithms to ensure that the tasks were rearranged and dates reassigned correctly based on the user's available time slots and deadlines while adhering to the selected strategy.

It unfortunately does override the LLM's own suggested scheduling but this can be seen as a positive feature as it gives the developper more control over the final output and ensures that the generated study plan stays deterministic and predictable based on the selected strategy.

5. Data formating and UI enhancements

Once the LLM generates the study plan in a Markdown table format, however the duration of the tasks are in decimal hours (e.g., 1.5 hours). To improve readability and user-friendliness, a function is implemented to convert these decimal hours into a more familiar hours and minutes format using regex (e.g., 1 hour 30 minutes) before displaying the final schedule to the user, as well as in the .csv or .ics export.

Other UI enhancements include the use of adaptive themes which was implemented using custom CSS injections to match the user's system theme (light or dark mode) for a more cohesive user experience.

6. Additional features

6.a) Gatekeeping irrelevant inputs

To ensure that the application remains focused on its intended purpose of academic planning, a gatekeeping mechanism is implemented in the backend. This is enforced using a LLM call to validate the relevance of user inputs, from this we classify the input in 3 categories: "unrelated", "incomplete", and "complete". Unrelated inputs are blocked, whereas incomplete inputs prompt the user for more information before proceeding. Only complete and relevant inputs are processed to generate the study plan.

6.b) Hourly planning

In order to provide enforce a more deterministic and precise study schedule, the web application uses the suggested LLM study plan but modifies the scheduling to fit the selected user's study approach and available time slots, and additionally takes into account breaks between study sessions. When task duration exceeds remaining daily hours, it automatically splits tasks and carries remainders to the next day. this is done through the function `apply_study_approach()`

6.c) Error handling

During testing a the LLM would sometimes generate outputs that would not parse well to tabular format, which could lead to errors in downstream processing, or even cause troubles in the gatekeeping mechanism for with malformed inputs. To mitigate this, we implemented error handling to flag some prompts as incomplete and ask the user for more information. Cases where the output table could not be parsed correctly are handled through `_safe_extract_json()` and in cases or connection errors with the LLM a clear connection error message is displayed to the user.

6.d) Markdown table parsing

To extract the study plan from the LLM's output, which is formatted as a Markdown table, we implemented a parsing function that converts the Markdown table into a Pandas DataFrame. This allows for easy manipulation and display of the study plan within the web application. The function `parse_markdown_table_to_df()` handles this conversion efficiently.

6.e) Download and Clear

To provide the user with more flexibility and control over their study plans, we implemented a download feature that allows users to export their generated study plans as CSV or ICS files to implement them into their personal calendars. Additionally, a clear button is provided to reset the input fields and start a new planning session without having to refresh the entire page.

6.f) Context preservation

To satsisfy the possibility of modifying the generated study plan, we implemented acuumulation of the conversation history with the LLM. This allows the user to make adjustments or request changes to the study plan while maintaining the context of previous interactions. The user can change deadlines, difficulty levels, or study durations, and also add or remove tasks. This is inportant to ensure that the user has quick access to modify their study plan as needed without having to feed all the information again from scratch.

6.g) Session preservation

In order to keep in memory all the variables corresponding to the session of the user (such as conversation history, user inputs, generated study plan, etc.), we used Streamlit's session state feature. This allows us to store and retrieve data across different interactions within the same session `st.session_state`. Without this feature the variables would reset after every button press, the app would lose the generated study plan, conversation history, and user progress instantly.

III. Results and User Interface

The user interface of the ThreadWise Planner is designed to be simple and intuitive, using this philosophy a minimalist design approach was adopted, focusing on the strategy selection and data input as the main components of the interface.

To select the desired study strategy, a dropdown menu is provided at the top of the interface, allowing users to easily switch between different strategies. In the detail section of the interface, users can input their study material, deadlines, and available time slots using the exmaple format displayed in the input field.

Other inputs are also included for the user to specify the maximum daily study time as well as the amount of time between study sessions for breaks. This is then taken into account when generating the study plan to ensure that the schedule is realistic and manageable. The algorithm takes into account the given AI suggested study durations and fits them into the available time slots accordingly using the proposed duraration of each task.

When the user inputs all the required data and sends it, the backend processes using the LLM and displays the generated study plan as long as the input is considered valid. An example of the output is shown below, for the "Waterfall" strategy.

Additional indexing option can also be used to better visualise the schedule, such as grouping by date or by topic. We also have the possibility to download the generated study plan as a

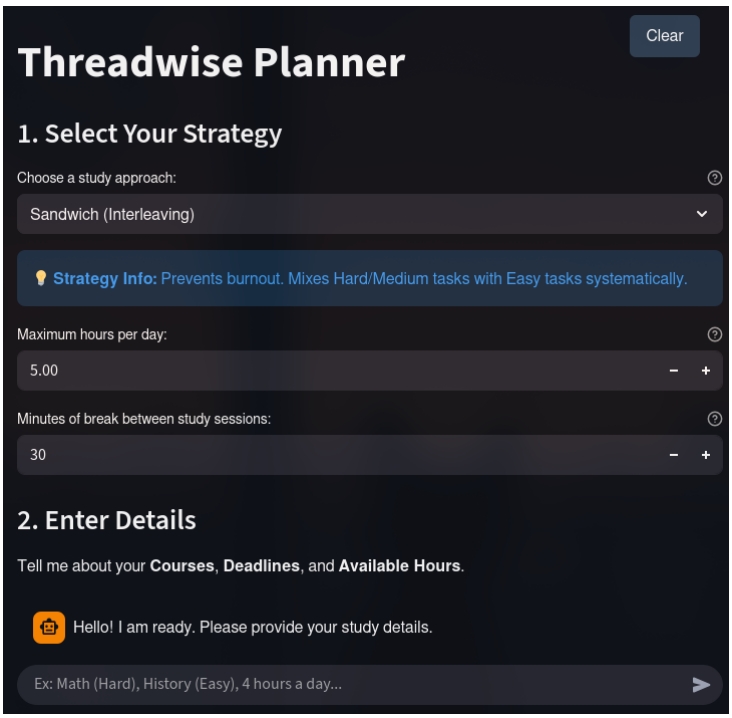

Figure 1. Home page of the ThreadWise Planner where the user selects the study strategy and inputs their data

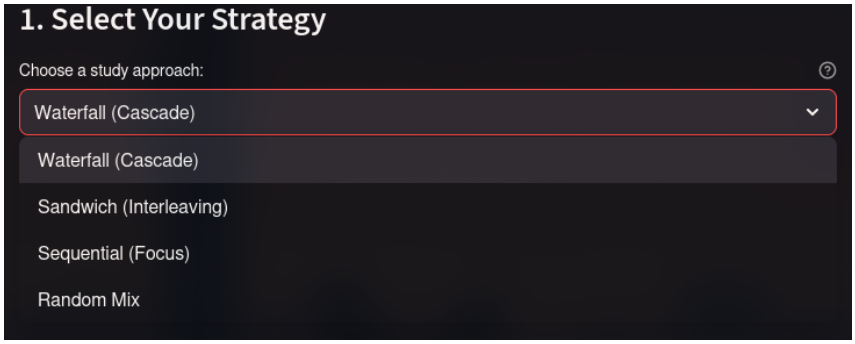

Figure 2. dropdown menu for selecting the desired study strategy

CSV file, if the user wishes to access the schedule offline or import it into other applications.

An important aspect to also remember is that as once the study plan is generated, the user can still talk to the LLM to modify or adjust the plan as needed, for example if they want to add more study time for a specific subject or change the deadlines. The LLM will then regenerate the plan accordingly while still adhering to the selected strategy and user constraints, and ask for any missing information if needed. This is done while still following the ruleset defined in the security layer to ensure that the outputs remain structured and relevant to the intended use. This way we ensure that the user has full control over their study plan while still benefiting from fast and efficient AI-generated scheduling.

IV. Conclusion

The ThreadWise Planner successfully integrates a user-friendly web interface with a powerful backend using LLMs to generate personalised study plans. By allowing users to select different study strategies and input their study material, deadlines, and available time slots, the application provides a tailored approach to academic planning which can help students optimize their study schedules and improve their learning outcomes. All of this is achieved while maintaining

Day	Subject	Task	Duration	Difficulty
2025-12-27 09:00:00	Math	Review algebra and functions	2 Hours	High
2025-12-27 11:30:00	Piano	Practice scales and chords	1 Hours	Low
2025-12-27 13:00:00	Math	Practice calculus problems	2 Hours	High
2025-12-28 09:00:00	Math	Practice calculus problems	30 Mins	High
2025-12-28 10:00:00	Piano	Work on piece performance	1h 30mins	Low
2025-12-28 12:00:00	Math	Work on word problems	1h 30mins	High
2025-12-28 14:00:00	Piano	Practice piece with rhythm	1 Hours	Low
2025-12-28 15:30:00	Math	Review integration techniques	30 Mins	High
2025-12-29 09:00:00	Math	Review integration techniques	1h 30mins	High
2025-12-29 11:00:00	Piano	Record and review performance	1 Hours	Low

Figure 3. Generated study plan displayed in a tabular format after processing the user input

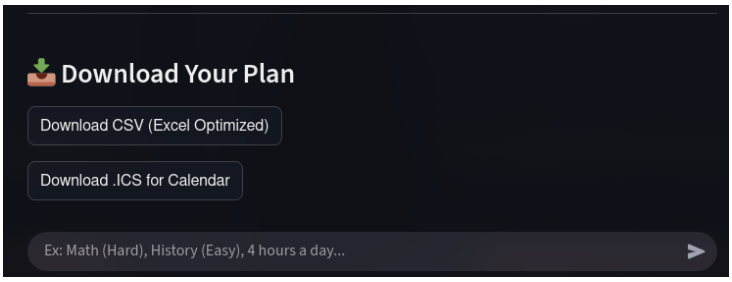

Figure 4. Functional download of the generated study plan as a CSV file for offline access

a simple and intuitive user experience, making planning a lot faster and effective for students.

Other interesting applications may extend to project management or work task scheduling, where similar principles of time management and task prioritization can be applied to enhance productivity and efficiency in professional settings.