SOFTWARE ENGINEERING PROJECT DOCUMENTATION TEMPLATE (STAGE 6)

Title Page

- **Project Title**:Student Study Planner(SSP)
- **Student Name**:Eden Chung
- **Date**:July 4th 2025
- **Course**: Software Engineering Stage 6
- **GitHub URL**:

1. Identifying and Defining

1.1 Problem Statement

What problem are you solving?

Students often feel overwhelmed managing multiple assessments and balancing study, extracurriculars, and leisure. Time management and task awareness are crucial, especially during assessment blocks. This project addresses that need with a digital planner that combines scheduling tools and AI support to reduce stress and improve academic productivity.

Who is affected?

- The core focus is on senior high school students.
- The planner is designed to assist students who struggle to stay organised.
- It also benefits students who are already organised but need a more efficient and updated method.
- University students can use the interactive calendar to manage studies and responsibilities.
- Professionals can use it to track tasks and maintain productivity.
- Busy adults can benefit from it by managing time effectively in a fast-paced lifestyle.
- The interactive calendar promotes an organised and productive lifestyle for users beyond just students.

1.2 Project Purpose and Boundaries

What is the project trying to achieve?

This project aims to help users take control of their time and responsibilities through a practical, user-friendly digital calendar. By recording tasks and deadlines, it reduces stress, improves productivity, and supports better study and work habits. The app encourages goal-setting and routine consistency, helping users complete tasks before they're due. Ultimately, it promotes independence, discipline, and time management skills essential not only in high school but throughout life. Building these habits early can significantly improve the academic performance of disorganised students.

What is in and out of scope?

In Scope:

- Interactive calendar with day/week/month views
- Task creation and tracking
- AI assistant (OpenAI API)
- Public holiday highlighting (Calendarific API)
- Secure, per-user data storage

Out of Scope:

- Mobile/web versions
- Real-time collaboration
- Notifications and reminders
- External calendar syncing
- Automatic timetable integration

1.3 Stakeholder Requirements

Who are your stakeholders?

Primary stakeholders:

- High school and university students
- Educators and parents
- Anyone needing to stay organised

What do they need from this software?

- Reliable time/task management
- Clean, intuitive UI
- Consistent tracking and long-term planning support

1.4 Functional Requirements

- Authentication: Sign-up/login with SHA-256 hashed passwords; guest mode available
- Calendar: Switchable day/week/month views, task counters, public holidays
- Tasks: Add/edit/delete all-day and hourly tasks with dialogs and visual feedback
- Storage: JSON-based local saving per user; guest data not saved
- APIs: OpenAI for AI assistant; Calendarific for holidays; includes error handling

1.5 Non-Functional Requirements

- Security: Encrypted passwords; no plaintext or SQL vulnerabilities
- Persistence: Task data saved locally with error handling
- Usability: CustomTkinter GUI, clear prompts, and warnings
- Performance: UI updates dynamically; APIs run in threads
- Maintainability: Modular class structure with constants for easy updates
- Scalability: Future additions (e.g., recurring tasks, smarter AI) supported
- Offline Support: Most features work offline; AI/holidays show warning when unavailable
- Limitations: No notifications, mobile/cloud sync, or analytics

1.6 Constraints

Time, budget, technical limits

Time:

- Limited timeframe required focusing on core features over advanced enhancements.

Budget:

- No external funding available.
- All tools were free or personally funded (e.g., ChatGPT API purchased by developer).

Technical Limits:

- Desktop-only application using Python and CustomfTkinter.
- No mobile or web support.
- Internet required for API features (AI assistant and public holidays).

2. Research and Planning

2.1 Development Methodology

Explain your approach (e.g. Agile, Waterfall) and why you chose it.

For this project, the AGILE method was utilized as it allowed for continuous testing, evaluation, and refinement of features, again, making it the ideal approach for a user experience centered application. It also supported quicker improvements and adaptability, which allowed me to make changes (minor through to major) relatively quickly and with minimal problems later on.

2.2 Tools and Technology

Languages:

- Python

IDEs:

- Visual Studio Code

Libraries, frameworks used

GUI:

- CustomTkinter: Modern themed GUI with cleaner design and customizable widgets.
- Tkinter: Used via CustomTkinter for message boxes.

Python Modules:

- json: Saves and loads user data/tasks.
- os: Handles file paths and directories.
- hashlib: Hashes passwords for security.
- threading: Prevents GUI freezing during AI responses.
- datetime, calendar, timedelta: Manage dates and times.

External APIs & Tools:

- Calendarific API: Retrieves Australian public holidays.
- OpenAI API: Powers the AI assistant for study support.
- requests: Makes API calls to Calendarific and OpenAI.

2.3 Gantt Chart / Timeline

Stage	Start Week	End Week	Estimated	Milestone
Planning	1	2	5	Proposal Complete
Design	2	3	5	Design Mockups Complete
Development	4	7	15	Core Features Built
Development	4	7	15	Additional Features Added
Testing & Degbugging	8	9	10	Testing Complete
Evaluation and Maintenance	9	10	5	Project Submission

Timeline	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10
Planning										
Design										
Development Sprint 1										
Development Sprint 2										
Testing & Debugging										
Evaluation & Deployment										

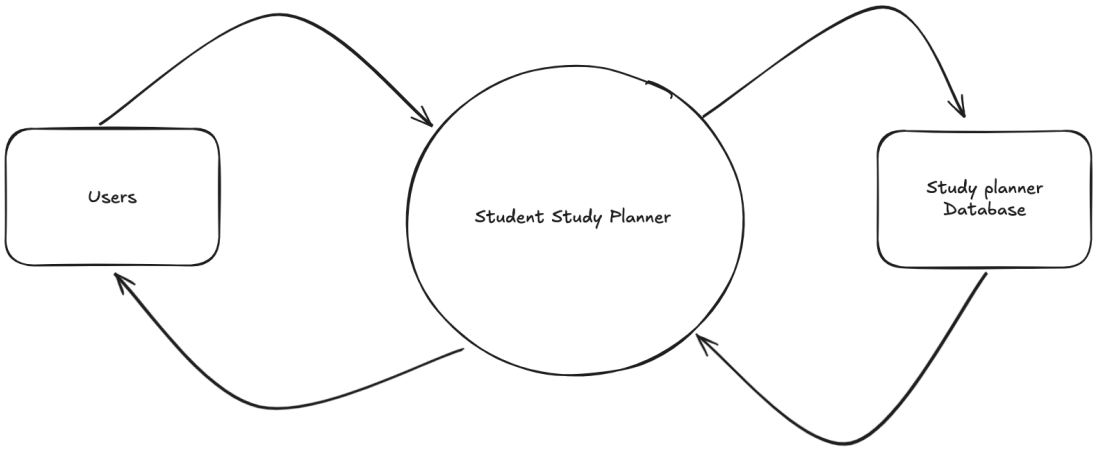
2.4 Communication Plan

How will you engage with your client or simulate feedback?

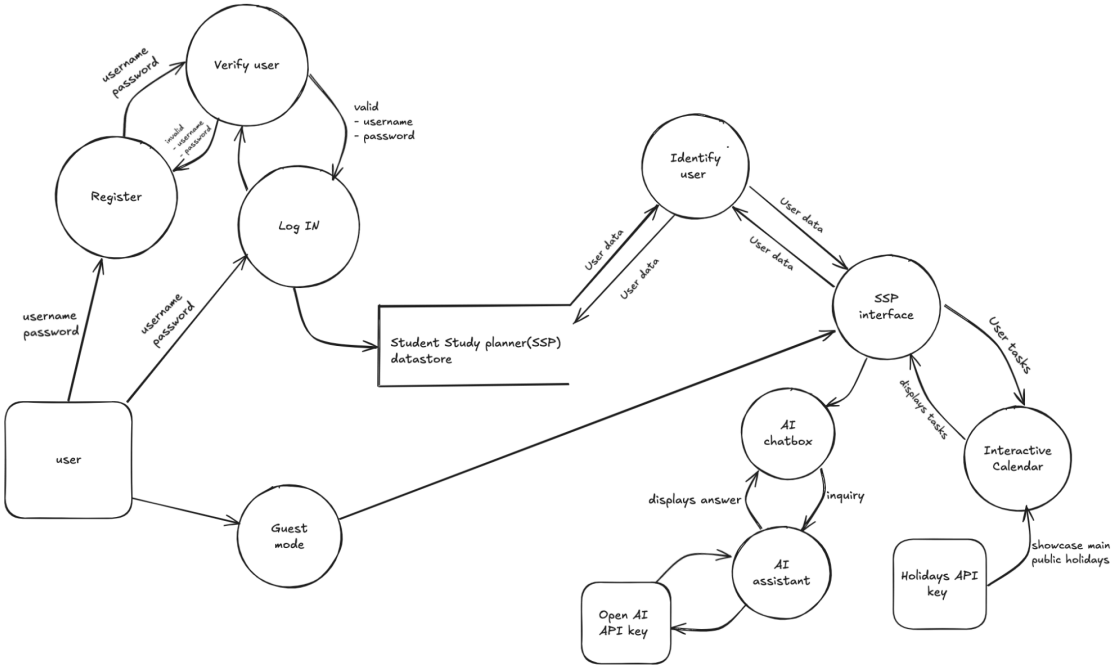
I will engage with client and simulate feedback by implementing additional features that clients may request, as well as fixing any bugs or issues that may arise

3. System Design

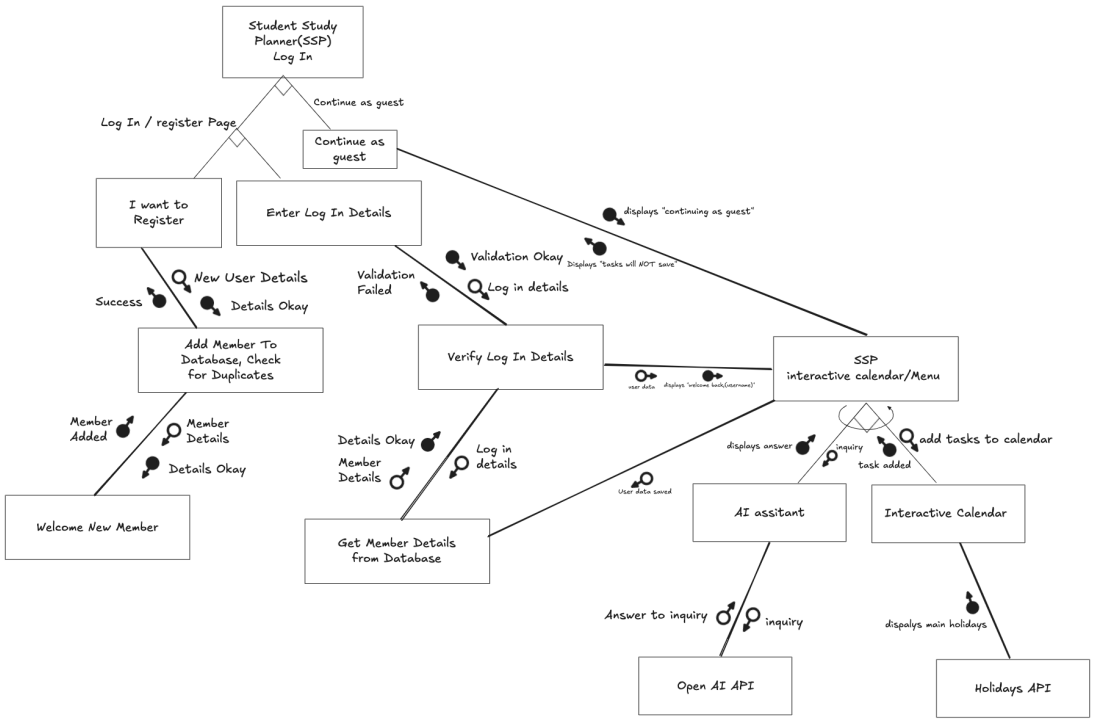
3.1 Context Diagram



3.2 Data Flow Diagrams (Level 0 and 1)



3.3 Structure Chart



3.4 IPO Chart

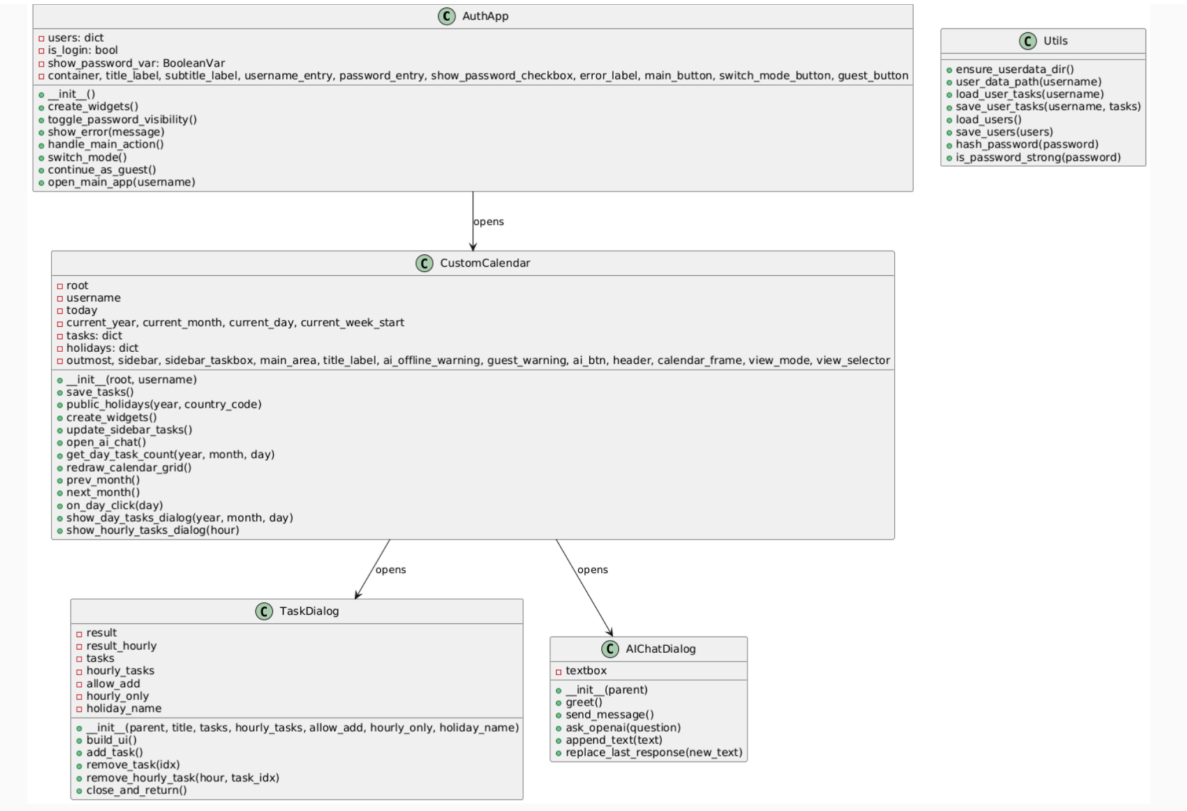
Input	Process	Output
- Username and password entered by user	- Authenticate user using hashed password - Check user existence or register new account	- Access granted to main calendar interface - New users created (if signing up)
- Task title and time/date selected	- Save task in json under correct date or hour - Update sidebar with current months all day tasks	- Task added to calendar and listed in sidebar
- Task removal request	- Locate and remove selected task from stored task list	- Task deleted and interface updated
- date/month/week navigation (button click)	- Calculate appropriate date context - Redraw Calendar grid - Fetch new public holidays if year changes	- Updated calendar view showing relevant days and holidays
- OpenAI API question typed by user	- Send questions to OpenAI API in a background thread - Wait for response - Display answer in AI chat	- AI generated response shown in the chat window
- User toggles between Day,Week and Month views	- Dynamically generate appropriate grid layout - Show relevant tasks or time slots accordingly	- Day/Week/Month calendar view displayed
- Application start	- Ensures directories exist	- Ready to use login/signup

	- Load any saved user or task data - Connect to APIs - Launch sign in screen	screen, then calendar app
- Guest mode selected	- Skip data saving - Open calendar in temporary session	- Non persistent calendar session with warning displayed
- Public holiday data for year	- Call calendarific API - Parse response JSON - Save relevant holidays	- Holidays highlighted on calendar with 🎉 emoji

3.5 Data Dictionary

Name	Type	Description
username	string	Stores the current user's login name or "guests"
users	dictionary	Stores all registered usernames and their hashed passwords
tasks	dictionary	Holds all task data for the current user,keyed by date or datetime strings
hourly_tasks	List of tuples	Holds tasks tied to specific hours,each tuple contains (hour,task description)
holidays	Dictionary	Maps (year,month, day) tuples to public holiday names fetched from API
view_mode	StringVar	Tracks the current calendar view mode:"Day","Week", or "Month"

3.6 UML Class Diagram (if OOP)



4. Producing and Implementing

4.1 Development Process

How did you approach the building of your solution?

Development followed the AGILE approach with a focus on security.

Started by experimenting with CustomTkinter to create widgets and buttons for:

- Signup page
- Calendar interface

Defined and built core features:

- User authentication
- Task management
- Calendar view

Each component was built and tested independently for stability and easier debugging.

Implemented secure data storage for user tasks and preferences.

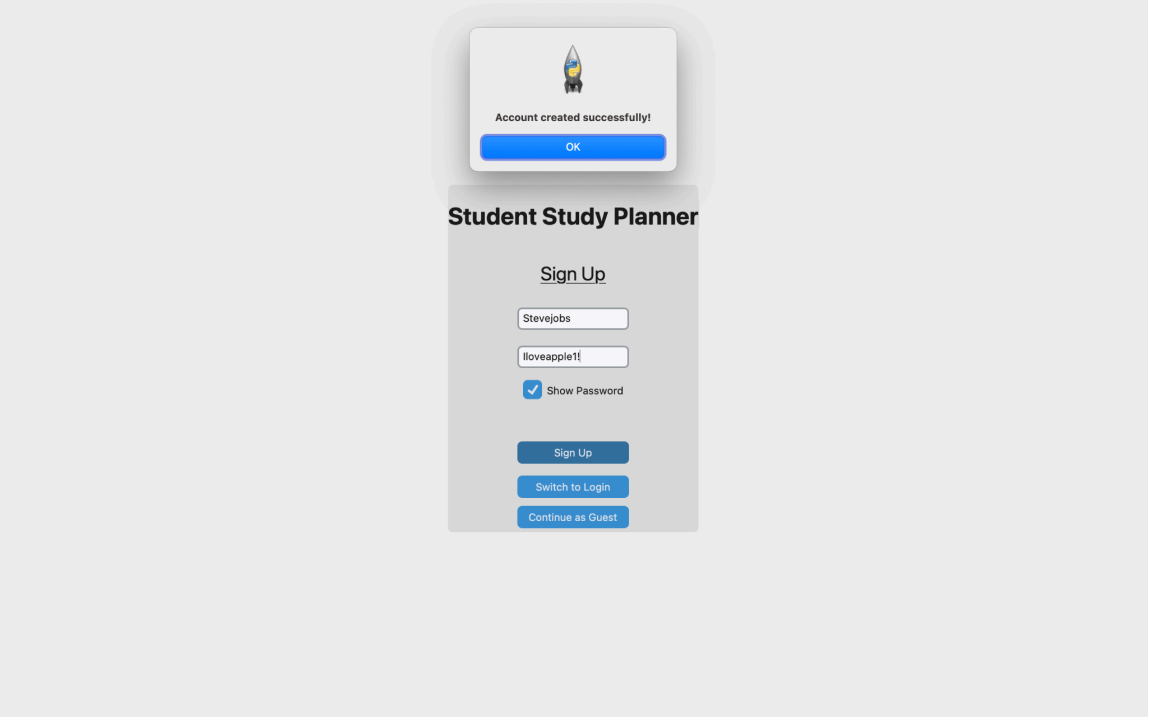
4.2 Key Features Developed

Describe and justify the core features.

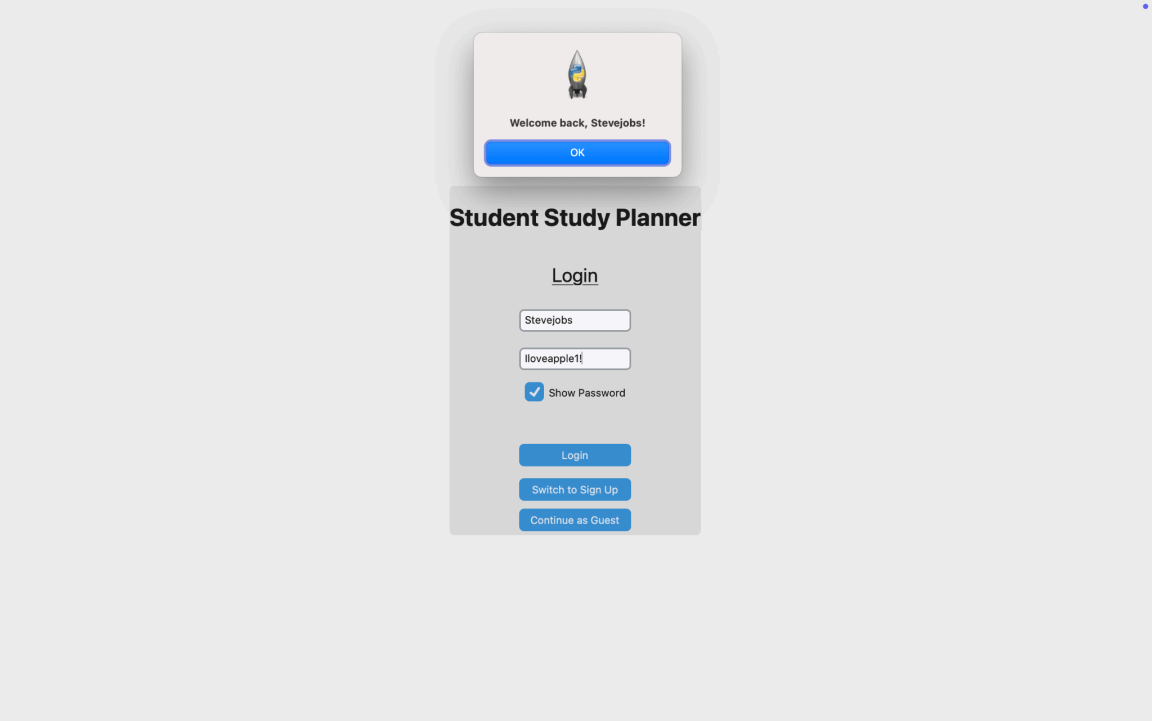
1. User Authentication:
- Secure sign-up and login system
- Passwords are hashed to protect user data
2. Interactive Calendar Views:
- Day, week, and month view options
- Supports both high-level and detailed scheduling
3. Task Management:
- Add, view, and remove all-day and hourly tasks
- Accommodates varied study routines and priorities
4. AI Study Assistant:
- Integrated via OpenAI API
- Provides interactive chat for study support and questions
- Enhances engagement and productivity
5. Persistent Data Storage:
- Tasks and user info stored locally and securely
- Data retained across sessions

4.3 Screenshots of Interface

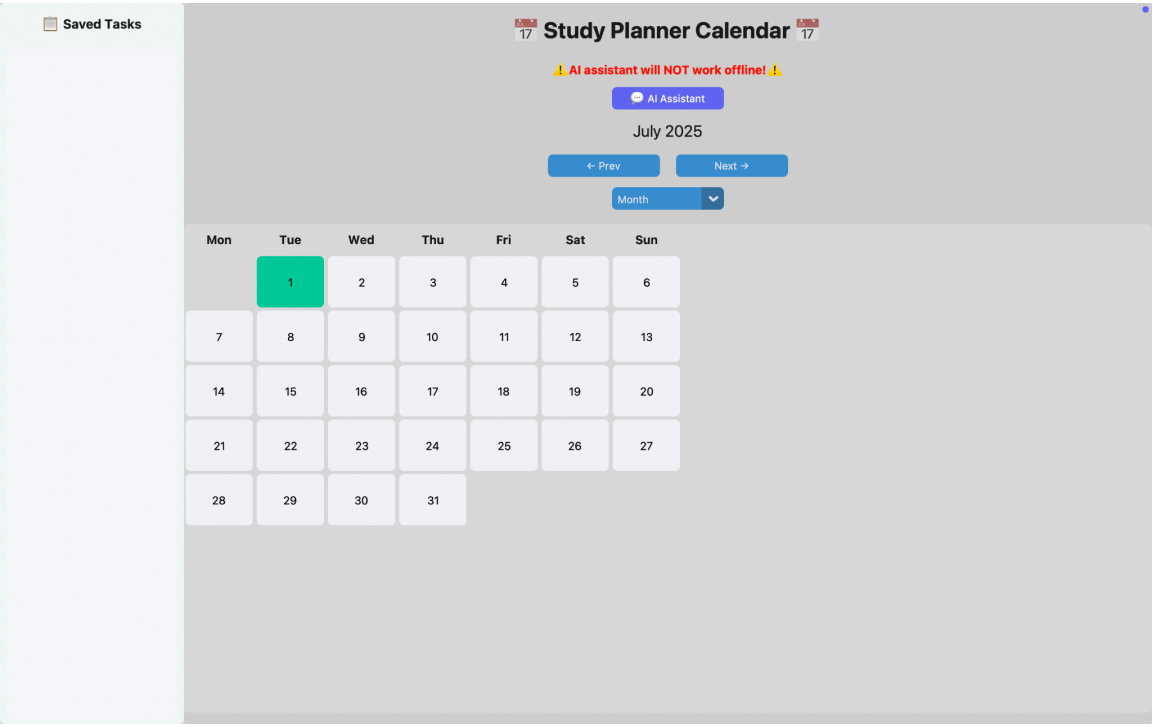
(Full screen ver)



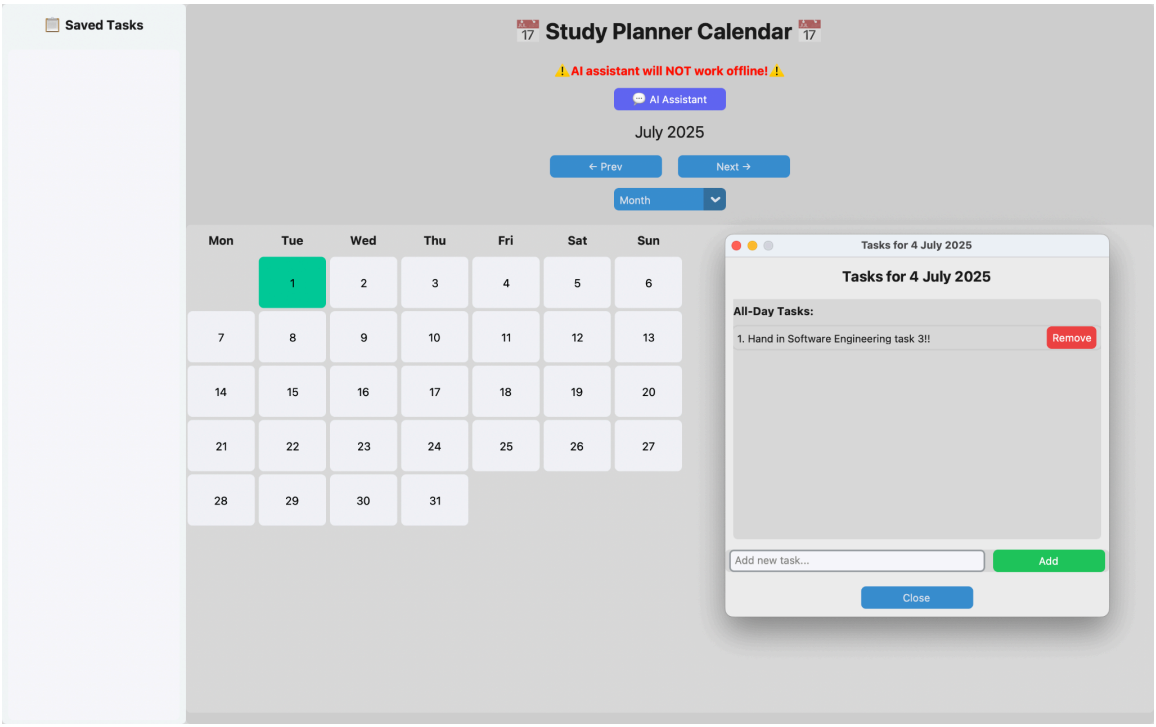
When program is run signup page comes up



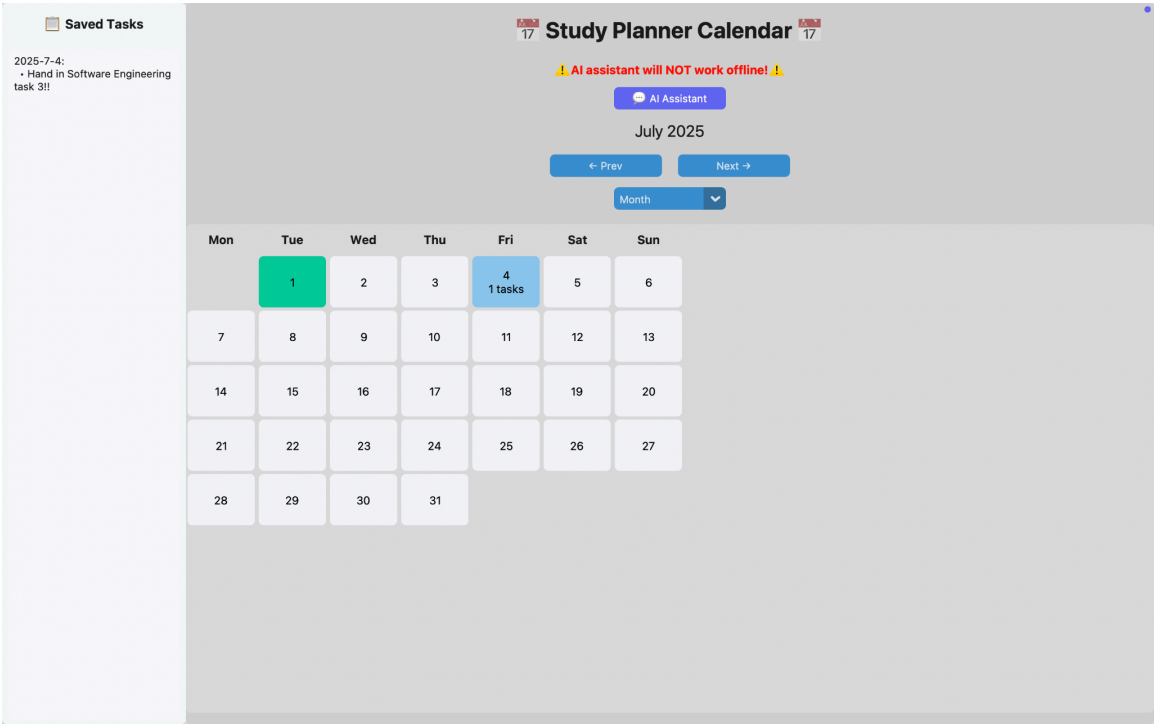
Once user successfully signs up, page automatically changes to Login



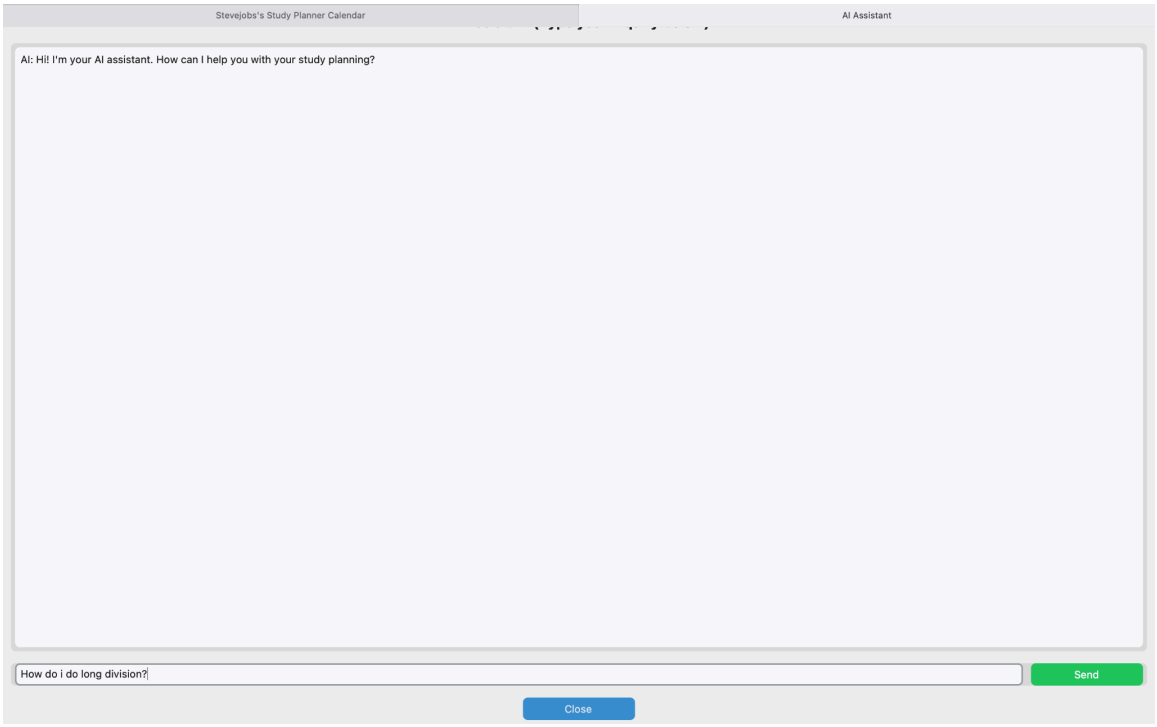
This is the “main menu” once users successfully create an account and or login, displaying the interactive calendar and AI assistant.



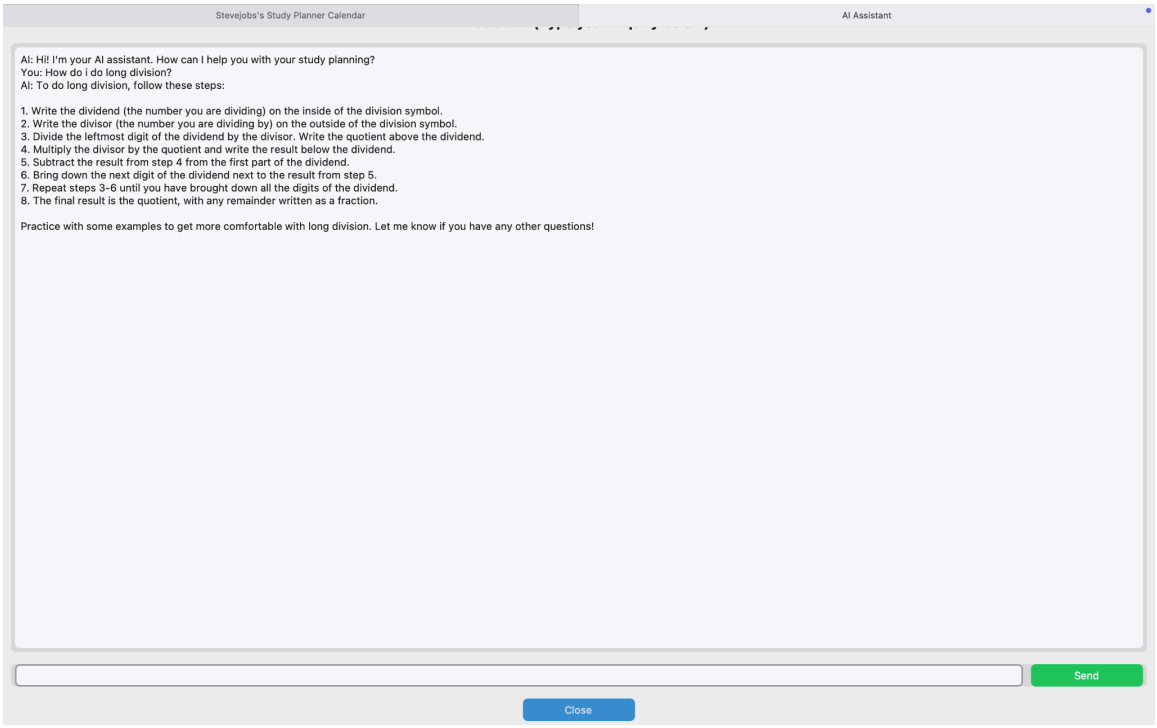
Once the user clicks on a specific date on the calendar, this mini screen pops up allowing users to save an all day task for a selected day.



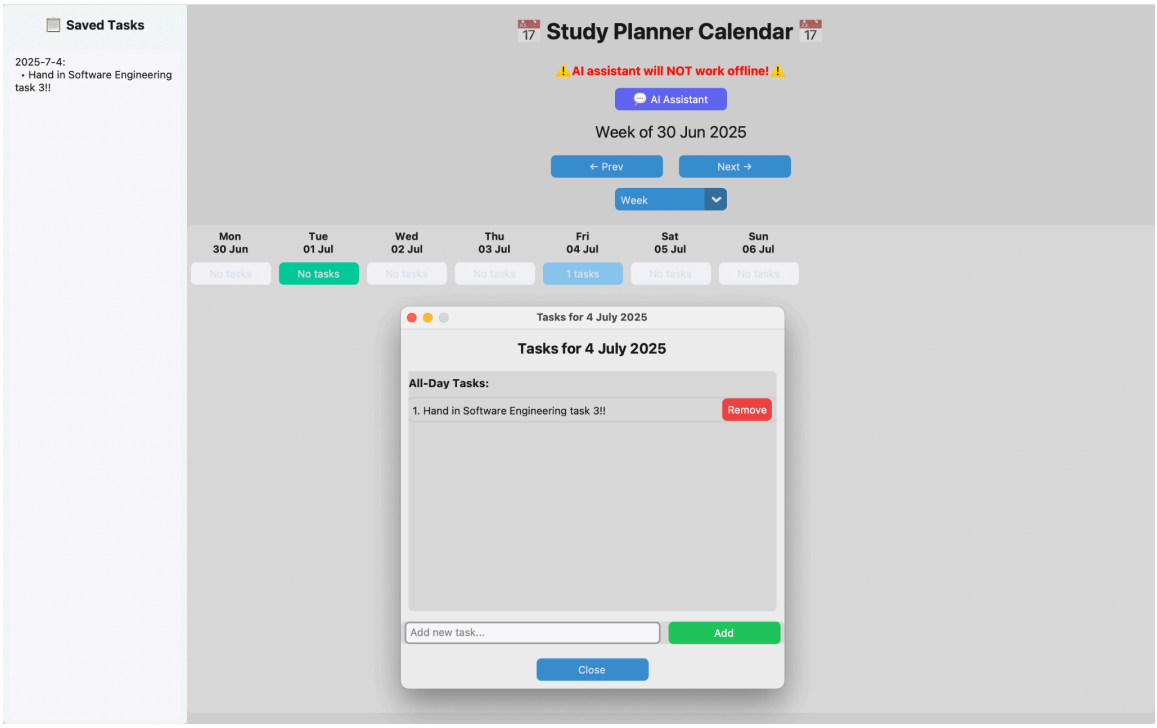
Once the user closes the little window, on the selected date, the colour changes and “(number) tasks” is saved under the date. The specific all day task is also displayed on the “saved tasks” section on the left that shows the user specifically what all day task is set on what date.



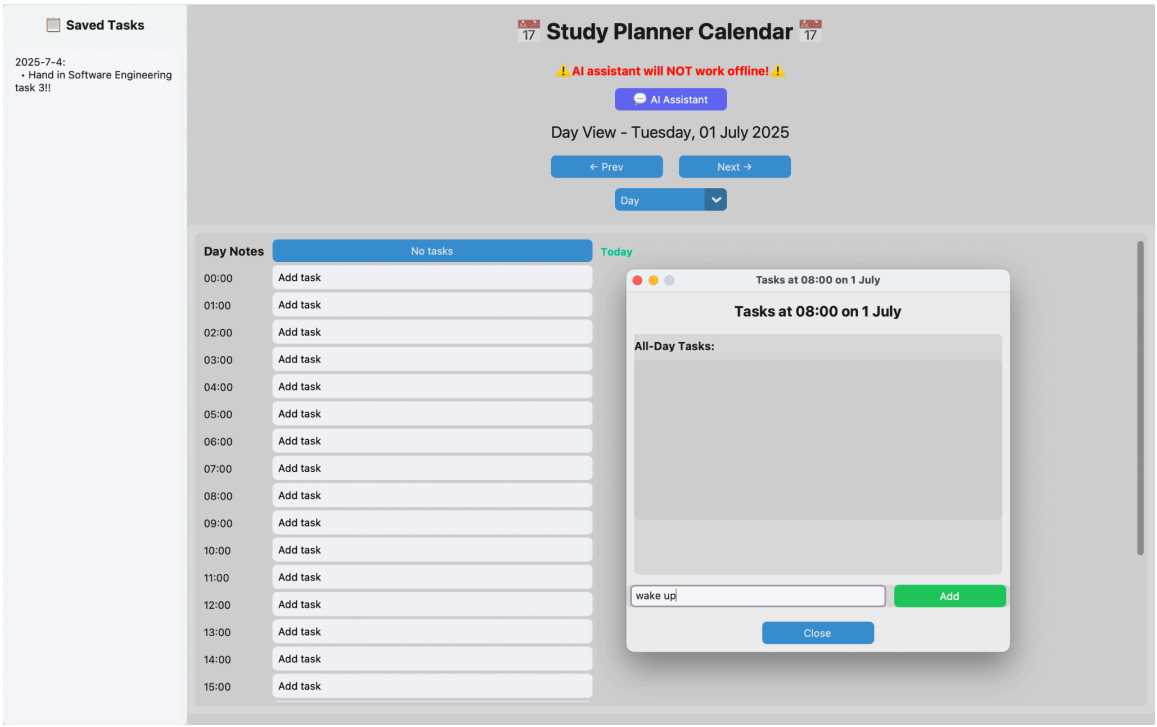
if the user clicks on the purple button that reads “AI assistant” this screen shows up allowing users to ask any inquiry.



And the AI answers accordingly, increasing users' experience.



The same applies for weekly tasks, once the user switches to the week view mode, the week is displayed with the current day coloured differently as well as any tasks set on a different date that week, which is then displayed on the sidebar.



If the user clicks on the Day View this screen is shown where users can set a specific task at a specific time, However this will not show up on the saved tasks bar on the left.

5. Testing and Evaluation

5.1 Testing Methods Used

Unit testing, user testing, automated testing, etc.

- Manual Testing: tested by interacting with the application's features ensureing that all core functionalities worked smoothly.
- Unit Testing: Key functions related to data handling were tested to ensure data integrity and security.
- API Response Testing: Verify correct handling of responses from external APIs, including error handling for network failures or invalid data.

5.2 Test Cases and Results

Test ID	Description	Expected Result	Actual Result	Pass/Fail
TC01	Login with valid user (functional testing)	Success message and access granted	Success message and access granted	Pass
TC02	Login with invalid username (functional testing)	Error message "User not found"	Error message "User not found"	Pass
TC03	Login with incorrect password (functional testing)	Error message "incorrect username or password"	Error message "incorrect username or password"	pass
TC04	Sign up with new username (functional testing)	Error message "incorrect username or password"	Error message "incorrect username or password"	Pass
TC05	Signup with new username (functional testing)	Account creation success message	Account creation success message	Pass

TC06	Signup with existing username (functional testing)	Error message “Username already exists”	Error message “Username already exists”	Pass
TC07	Add all day task on a date (functional testing)	Task added and displayed in sidebar	Task not added not added on sidebar	Fail
TC08	Remove task from day (functional testing)	Task removed and sidebar updated	Task removed and sidebar updated	Pass
TC09	Switch between day, week and month views (functional/UI testing)	Calendar updates accordingly	Calendar updates accordingly	Pass
TC10	Flick through days,months and months on calendar (functional/UI testing)	Calendar updates accordingly with minimal lag	Calendar takes too long to load	Fail
TC11	Fetch public holidays via API (API response testing)	Holidays displayed on calendar	Holidays don't show up on calendar	Fail
TC12	Use Ai Assistant with query (API response / system testing)	Ai provides relevant study response	AI does not respond	Fail
TC13	Continue as Guest (Functional testing)	Tasks not saved after restart	Tasks not saved after restart	Pass
TC14	Test if user data saves (persistence/D	Tasks saved exactly as intended	Tasks saved exactly as intended	Pass

	ata testing)			
TC15	Test hash_password() (Unit testing)	Hash is generated correctly and consistently	Hash is generated correctly and consistently	Pass
TC16	Test is_password_strong()(Unit testing)	Weak passwords are rejected, strong ones pass	Weak passwords are rejected, strong ones pass	Pass
TC17	Test save_user_tasks()(Unit testing)	Tasks are correctly saved to and loaded from JSON file	Tasks are correctly saved however the JSON file did not show.	Fail

5.3 Evaluation Against Requirements

How well does the solution meet goals?

Technical

Goal	Level of achievement	Comment
Secure password storage	Met	SHA-256 hashing used, no plain text stored
Offline support	met	App works offline except for AI and holidays
error handling	met	Robust handling for file and API errors
Performance	met	Minor lag when navigating calendar
Usability UI	Partially met	UI is clean and easy to navigate
Modular code structure	Met	Code organized in classes for easy maintenance

Functional

Goal	Level of achievement	Comment
Easy login	Met	Easy and straightforward for users
Task management	Met	Users can add/edit/delete task successfully
Calendar navigation	Met	Day/Week/Month views work as intended
Guest access	Met	Guest mode works with clear warnings
Ai assistant integration	Partially met	Falls slow sometimes, does not work online
Public holiday highlight	Partially met	Fails to display some less popular holidays
Data persistence	Met	Tasks save/load correctly for registered users

Summary

Most of the core functionality was met providing users with a user friendly interactive calendar that supports academic organization and productivity, fulfilling its primary purpose of improving time management and supporting academic planning for its target audience securely.

5.4 Improvements and Future Work

What would you improve or add if you had more time?

Add:

- Stopwatch/timer -allows students to track and record study time directly on the calendar. Prevents distractions from using a stopwatch on users phones etc.
- Notification function - sends notifications to remind users of due dates, prevents missed deadlines.
- “Forgot password” feature - allows users to recover their account instead of making a new account potentially losing all of their set tasks.

Improve: Works but could be better

- General calendar functionality
    - Reduce delays when flicking through different views and dates.

    - Ensure AI assistant provides answers much faster
    - Add shortcuts such as “esc” to minimize the task setting window
- A more “aesthetically pleasing” GUI
    - Use code instead of python modules to enhance GUI

6. (Client) Feedback and Reflection

6.1 Summary of Client Feedback

List of client comments or peer feedback.

Positive: Intuitive layout, useful views, helpful AI, modern UI

Negative: No recurring tasks, slow AI loading, lacks notifications

6.2 Personal Reflection

What did you learn? What skills did you gain?

I learned how to design and develop a secure, user-friendly Python application using custom tkinter. I improved my skills in UI design, data handling, API integration, and code organisation. The project also helped me understand the full development process, from planning to testing, while enhancing my problem-solving and time management abilities which was a new experience.

7. Appendices

- Update Gantt Chart

Stage	Start Week	End Week	Estimated	Milestone
Planning	1	2	5	Proposal Complete
Design	2	3	5	Design Mockups Complete
Development	4	7	15	Core Features Built
Development	4	7	15	Additional Features Added
Testing & Degbugging	8	9	10	Testing Complete
Evaluation and Maintenance	9	10	5	Project Submission

Timeline	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10
Planning										
Design										
Development Sprint 1										
Development Sprint 2										
Testing & Debugging										
Evaluation & Deployment										

- Exemplar Code Snippets

```python
def hash_password(password):
    return hashlib.sha256(password.encode()).hexdigest()

def is_password_strong(password):
    return (
        len(password) >= 8 and
        any(c.isupper() for c in password) and
        any(c.islower() for c in password) and
        any(c.isdigit() for c in password) and
        any(c in "!@#$%^&*()_+-=[]{}|;':\",.<>?/" for c in password)
    )
```

Ensures passwords are sophisticated,, furthermore hashes secure passwords.

```python
class AIChatDialog(ctk.CTkToplevel):
    def ask_openai(self, question):
            url = API_URL
            headers = {
                "Authorization": f"Bearer {OPENAI_API_KEY}",
                "Content-Type": "application/json"
            }
            data = {
                "model": "gpt-3.5-turbo",
                "messages": [
                    {"role": "system", "content": "You are a helpful study planning assistant."},
                    {"role": "user", "content": question}
                ],
                "max_tokens": 500,
                "temperature": 0.7,
            }
            response = requests.post(url, headers=headers, json=data, timeout=20)
            if response.status_code == 200:
                resp = response.json()
                msg = resp["choices"][0]["message"]["content"].strip()
                self.after(0, lambda: self.replace_last_response(f"AI: {msg}\n"))
            else:
                error_msg = f"AI: [API error {response.status_code}] {response.text}\n"
                self.after(0, lambda: self.replace_last_response(error_msg))
        except Exception as e:
            self.after(0, lambda: self.replace_last_response(f"AI: [Error: {str(e)}]\n"))
```

Sends user questions to open AI API, then displays answers
Allows for me to test the API