Technical Overview of the Study Helper Desktop Application

General Description

The Study Helper is an offline desktop application designed to assist users in managing their study routines effectively. It is built with a focus on portability, security, and user customization. The application uses a Python backend to manage local data operations and a React-based frontend with Electron for a modern and responsive user interface. Tailwind CSS is integrated into the frontend for efficient and consistent styling. The application functions entirely offline, requiring no Wi-Fi, and communicates between the frontend and backend through local API calls.

Key Features

1. **Task Management**
    - Users can create, edit, delete, and organize study tasks.
    - Tasks are prioritized based on user-defined categories.
2. **Pomodoro Timer**
    - Integrated focus timer with customizable work and break durations.
    - Notifications to remind users of session transitions.
3. **Flashcards**
    - Support for creating, categorizing, and reviewing flashcards.
    - Includes quiz modes for self-testing.
4. **Progress Tracking**
    - Tracks total study time, completed tasks, and reviewed flashcards.
    - Visual representation of progress over time.
5. **Data Management**
    - Persistent storage of tasks, flashcards, and progress in JSON files.
    - File encryption to secure user data.
    - Option to export all data to a ZIP file and import data to restore or replace existing files.

Technical Implementation

1. **Backend**
    - Built with **FastAPI** for managing local API calls.
    - Data stored in JSON files within a local directory for simplicity and portability.
    - AES-based encryption implemented using the `cryptography` library to secure all stored data.

    - Default storage location is OS-specific, but users can select a custom directory during setup or in the settings menu.
2. **Frontend**
    - Developed with **React** and **TypeScript** for a modular, scalable, and type-safe user interface.
    - Integrated with **Electron** to package the frontend as a desktop application.
    - Styled with **Tailwind CSS**, offering utility-first classes for rapid UI development and consistent design.
    - Communicates with the Python backend via local API calls.
    - Includes a settings menu to manage storage location, export, and import options.
3. **Storage and Encryption**
    - Default storage paths:
        - Windows: `%AppData%\StudyApp`
        - macOS: `~/Library/Application Support/StudyApp`
        - Linux: `~/.studyapp`
    - Allows users to override the default path and store data in a custom directory.
    - Data is stored in encrypted JSON files for security and is only decrypted during runtime.
4. **Export and Import**
    - Data export bundles all JSON files into a ZIP archive.
    - Data import validates and replaces the current storage files with those from a ZIP archive.
    - Accessible through a user-friendly interface in the settings menu.
5. **Styling with Tailwind CSS**
    - Tailwind's utility classes simplify the styling process.
    - Allows for responsive and accessible design with minimal custom CSS.
    - Ensures consistent look and feel across the application.
6. **Packaging and Deployment**
    - The backend is bundled into a single executable using **PyInstaller**.
    - The frontend is packaged with Electron Builder into a single `.exe` file.
    - Final distribution includes both the backend and frontend, ensuring seamless communication and offline functionality.

System Requirements

- **Platform**: Windows (initial focus, extendable to macOS and Linux).
- **Dependencies**: No external server or internet connection required after installation.
- **Data Security**: Encrypted storage and optional backup/restore ensure privacy and safety.

Unique Selling Points

- Fully offline with no internet dependency.
- Secure, encrypted local data storage.
- User-controlled data storage location.
- Modern, cross-platform interface with responsive design.
- Styled with Tailwind CSS for rapid development and consistent UI.