inputs and connects the two. This structure is ideal for the scheduling application since it For this project, the Model-View-Controller (MVC) architecture is the best choice due the View, which manages the user interface; and the Controller, which processes user application into three components: the Model, which handles data and business logic; allows independent updates to the UI, scheduling logic, and external integrations like to its clear separation of concerns, flexibility, and maintainability. MVC organizes the Google and Outlook calendars.

Other architectural styles, such as Data-Flow and Main Program/Subprogram, are less Main Program/Subprogram structure, while good for hierarchical designs, tightly couples suitable. The Data-Flow model works best for continuous data processing but lacks the flexibility needed for an interactive scheduling tool where user input is dynamic. The logic, making updates and UI modifications more difficult.

Al-based recommendations, and output optimized schedules and reminders. A database The application will handle task data, user preferences, AI-generated scheduling, and calendar synchronization. It will input user tasks, store scheduling preferences, process is required to store user information, scheduled tasks, and historical data. A relational database like PostgreSQL or MySQL would be best for structured data, while NoSQL could handle Al learning patterns.

scheduling algorithms, enhance the UI, and add integrations without disrupting the entire MVC ensures scalability, maintainability, and easy updates. Developers can refine system. This modular approach keeps the software efficient, user-friendly, and adaptable to future improvements.

