### **3.3 Non-Functional Requirements**

The application can deliver a reliable and user-friendly experience that empowers users to effectively manage their time, organize their tasks, and enhance their learning through flashcards. These NFRs provide a strong foundation for developing a comprehensive productivity tool that caters to the diverse needs of users

#### 3.3.1 Performance

The pomodoro timer functionality shall exhibit minimal lag and ensure a smooth user experience. The application should respond to user actions, such as starting, pausing, and resetting the timer, within 1 second. Additionally, the visual and auditory cues marking the beginning and end of intervals should trigger precisely at the designated times, with a tolerance of no more than one minute. This performance requirement applies to all supported devices and operating systems, even under moderate system load, to guarantee an uninterrupted flow during focused work sessions and breaks.

The application should maintain consistent performance even during peak usage periods. This ensures users can add tasks, set timers, and access flashcards seamlessly.

#### 3.3.2 Reliability

The application should implement robust data recovery mechanisms to safeguard user information in case of unexpected shutdowns or device malfunctions. This ensures minimal data loss and protects critical details like scheduled events, timers and remainders.

#### 3.3.3 Availability

Upon restart after a crash or bug, the application should automatically recover the most recent data. This allows users to resume their tasks, pick up where they left off with their schedules, and continue studying with their flashcards without manually re-entering lost information. By implementing these data persistence and recovery mechanisms, the application guarantees user productivity and minimizes disruptions caused by unforeseen issues.

#### 

#### 

#### 3.3.4 Security

The application should undergo regular security testing and patching to identify and address potential vulnerabilities promptly. This proactive approach minimizes the risk of cyberattacks and safeguards user data from exploitation. By adhering to these security NFRs, the application fosters user trust and empowers them to confidently manage their tasks, schedules, and learning without compromising their privacy

#### 3.3.5 Maintainability

Well-documented code with comments and explanations will enhance understanding for developers who maintain the application in the future. This reduces time spent deciphering the code's logic and streamlines the maintenance process

The application should be built using a modular architecture, where functionalities like the pomodoro timer, scheduler, and flashcards are separate modules. This simplifies future modifications and reduces the risk of unintended consequences when changing one section of the code.

#### 3.3.6 Portability

The application should adhere to relevant industry standards and best practices for mobile development. This promotes code clarity and maintainability, making it easier to port the application to new platforms that follow similar standards. By following these portability NFRs, the application gains flexibility for future expansion and reaches a wider user base across different mobile ecosystems

### **3.4 Inverse Requirements**

* **Unnecessary Complexity:** The application should avoid overly complex interfaces or functionalities that overwhelm users. A clear and intuitive design is essential for users to efficiently manage their tasks, schedules, and flashcards.
* **Data Leakage:** The application should not collect or store any user data beyond what's necessary for core functionalities. This protects user privacy and avoids potential security risks associated with unnecessary data collection.
* **Battery Drain:** The application should be optimized for efficient battery usage. Excessive background processes or resource-intensive features can quickly drain battery life.
* **Internet Reliance :** If the core functionalities (pomodoro timer, scheduler, flashcards) can function offline, the application should strive to minimize its dependence on an internet connection. This ensures users can be productive even in situations with limited or no internet access.

By adhering to these inverse requirements, the application prioritizes a user-friendly experience, minimizes distractions, and safeguards user privacy. This focus on what the application should not do complements the traditional NFRs, ultimately leading to a well-rounded and effective productivity tool.

### 

### **3.6 Logical Database Requirements**

The success of this productivity application hinges on a robust and secure local database to manage user information. Since the application utilizes SQLite as its embedded database engine, the following non-functional requirements (NFRs) are crucial for ensuring data integrity and accessibility.

The local SQLite database should persistently store all user-generated data associated with the application's core functionalities. This includes information related to pomodoro timers (duration, history), schedules (events, tasks), and flashcards (decks, content, progress).By persisting this data, the application ensures users don't lose their work due to application crashes, device restarts, or unforeseen shutdowns. Users can confidently rely on the application to retain their progress and pick up where they left off.