# 2.5 Design and Implementation Constraints

## 2.5.1 Design Constraints

1. **User Interface Consistency:**

The user interfaces must be designed consistently and be easy to use for all types of users (students, teachers, and parents), with a responsive layout that adapts to various devices (desktops, tablets, and mobile phones).

1. **Technology Restriction:**

The system is required to use React.js for the web frontend and Flutter for the mobile application to ensure high performance and a unified user experience across platforms.

1. **Accessibility and Usability:**

The design must follow accessibility principles regarding color contrast, font sizes, and visual hierarchy to accommodate different age groups and accessibility needs.

1. **Data Synchronization:**

Both the web and mobile applications must integrate seamlessly with the same backend REST API to ensure consistent and synchronized data across all platforms.

1. **Integration Flexibility:**

The system should be designed to allow easy integration with external services such as Google Meet API and AI Chatbot APIs without the need to rebuild core system components.

1. **Security by Design:**

Security principles must be considered from the early design phase, especially for handling sensitive data such as passwords and student grades.

## 2.5.2 Implementation Constraints

1. **Backend Framework Restriction:**

The backend must be developed using Node.js and Express.js to ensure compatibility with RESTful architecture and handle concurrent requests efficiently.

1. **Database Limitation:**

The relational database PostgreSQL is used for structured data (e.g., users, exams, grades), while MongoDB stores unstructured data such as videos and PDFs.

1. **Security Standards:**

Passwords must be encrypted using bcrypt, authentication and session management handled via JWT (JSON Web Token), and file uploads restricted and validated using multer.

1. **Performance and Resource Constraints:**

Since the system may be hosted on limited-resource servers (e.g., free-tier or academic environments), performance optimization should be prioritized through query optimization, caching, and reduced heavy computations.

1. **Development Environment Constraints:**

Only open-source tools and frameworks should be used. Code must follow clean code principles and maintainable architecture to support future scalability.

1. **Mobile Integration Constraint:**

The mobile application must be developed exclusively using Flutter and must adhere to the same design and data consistency standards as the web platform.