TECHNICAL REQUIREMENT

For an **Airline Management System** college project, the technical requirements will define the technologies, tools, and frameworks needed to build the system. Here’s a breakdown of the essential technical requirements:

**1. Programming Language**

* **C++/Java**: If you’re focusing on a backend-heavy project, C++ or Java would be good choices for building core functionalities like flight management, passenger handling, and booking.
* **Python**: Python can be used for a simpler, quicker implementation, especially if using libraries like Flask or Django for a web-based version.
* **JavaScript**: If you’re building a web-based interface, JavaScript (along with frameworks like React or Angular) would be useful for the frontend.

**2. Database**

* **MySQL/PostgreSQL**: For relational databases, where flight details, passenger information, and bookings will be stored in tables. MySQL and PostgreSQL are popular, easy-to-use options.
* **SQLite**: For smaller-scale projects, SQLite can be used as an embedded database.
* **MongoDB**: If you need a NoSQL database, MongoDB can store documents and provide flexibility in data management.

**3. Backend Development**

* **Java (Spring Boot)**: For a more enterprise-grade airline management system, Spring Boot is a good framework for building REST APIs and handling complex data.
* **Node.js (Express)**: If you prefer JavaScript, Node.js with Express can be used for building the backend, especially if you are comfortable with JavaScript.
* **Flask/Django (Python)**: These Python frameworks are great for handling backend logic and making a system scalable and secure.

**4. Frontend Development**

* **HTML/CSS/JavaScript**: Basic web development tools to create a user-friendly interface for passengers and admins.
* **React/Angular**: Modern frontend frameworks for building dynamic, interactive user interfaces (UIs). They help in creating responsive designs for booking pages, flight schedules, and ticket management.
* **JavaFX**: If you are working with Java, JavaFX can be used for building a desktop-based graphical user interface (GUI).

For an **Airline Management System** college project, the technical requirements will define the technologies, tools, and frameworks needed to build the system. Here’s a breakdown of the essential technical requirements:

**1. Programming Language**

* **C++/Java**: If you’re focusing on a backend-heavy project, C++ or Java would be good choices for building core functionalities like flight management, passenger handling, and booking.
* **Python**: Python can be used for a simpler, quicker implementation, especially if using libraries like Flask or Django for a web-based version.
* **JavaScript**: If you’re building a web-based interface, JavaScript (along with frameworks like React or Angular) would be useful for the frontend.

**2. Database**

* **MySQL/PostgreSQL**: For relational databases, where flight details, passenger information, and bookings will be stored in tables. MySQL and PostgreSQL are popular, easy-to-use options.
* **SQLite**: For smaller-scale projects, SQLite can be used as an embedded database.
* **MongoDB**: If you need a NoSQL database, MongoDB can store documents and provide flexibility in data management.

**3. Backend Development**

* **Java (Spring Boot)**: For a more enterprise-grade airline management system, Spring Boot is a good framework for building REST APIs and handling complex data.
* **Node.js (Express)**: If you prefer JavaScript, Node.js with Express can be used for building the backend, especially if you are comfortable with JavaScript.
* **Flask/Django (Python)**: These Python frameworks are great for handling backend logic and making a system scalable and secure.

**4. Frontend Development**

* **HTML/CSS/JavaScript**: Basic web development tools to create a user-friendly interface for passengers and admins.
* **React/Angular**: Modern frontend frameworks for building dynamic, interactive user interfaces (UIs). They help in creating responsive designs for booking pages, flight schedules, and ticket management.
* **JavaFX**: If you are working with Java, JavaFX can be used for building a desktop-based graphical user interface (GUI).

**5. Security Features**

* **Authentication & Authorization**: Implement security protocols to protect user data and admin control. Tools like **OAuth**, **JWT (JSON Web Tokens)**, or **Spring Security** (for Java projects) can be used.
* **Encryption**: Use SSL/TLS encryption for secure data transmission, especially for sensitive information like personal details or payments.
* **Password Hashing**: Implement password hashing (e.g., using **bcrypt**) to store user credentials securely.

**6. Payment Gateway Integration**

* **Payment APIs**: Integrate third-party payment gateways like **PayPal**, **Stripe**, or **Razorpay** for handling online payments.
* **Currency Conversion API**: For international flights, use APIs to manage fare calculations in different currencies.