Non-Functional Requirements

# Performance

- The frontend should respond within 2 seconds for each user interaction.

- React and Flutter interfaces must load quickly even on slow internet connections (≤ 3 Mbps).

- Backend API must handle at least 500 requests per minute without noticeable delays.

- Use caching mechanisms (e.g., Redis) to improve data retrieval speed.

- Uploaded files (videos, PDFs) should be compressed to optimize load and storage time.

# Security

- User passwords must be encrypted using bcrypt before being stored in the database.

- Use JWT (JSON Web Tokens) for secure authentication and session management.

- Implement role-based access control (RBAC) to restrict actions by user type (Student, Teacher, Parent).

- Protect the system against SQL Injection, XSS, and CSRF attacks.

- All communication between client and server must occur over HTTPS.

- Validate uploaded files and restrict file types to safe formats (e.g., PDF, MP4).

# Scalability

- Backend should allow horizontal scaling as the number of users increases.

- Architecture should support modular or microservice expansion for future features.

- System should handle simultaneous requests from multiple users efficiently.

# Reliability

- Ensure at least 99.5% uptime.

- Implement proper error handling to avoid crashes and provide clear feedback to users.

- Automate daily backups for PostgreSQL and MongoDB databases.

- Failure of one service must not affect the operation of others.

# Usability

- Interface should be intuitive, responsive, and easy to navigate for users of all ages.

- Design should be mobile-friendly and support multiple screen sizes.

- Flutter app interfaces must be optimized for Android and iOS.

- Support both Arabic and English languages.

# Maintainability

- Follow modular, clean, and well-documented coding practices.

- Use Git/GitHub for version control and collaboration.

- Separate frontend and backend for easier updates and maintenance.

- Maintain a dedicated testing environment before deploying new features.

# Compatibility

- Web app should be compatible with major browsers (Chrome, Edge, Firefox, Safari).

- Mobile app should run on both Android and iOS platforms.

- RESTful API should be interoperable with third-party applications (e.g., Google Meet, AI APIs).

# Testability

- Provide a testing environment separate from production.

- Write unit tests for backend using Jest or Mocha.

- Use React Testing Library for frontend testing.

- Conduct performance and security tests before deployment.

# AI and Advanced Features Reliability

- Ensure AI-related modules do not degrade system performance.

- Store AI analysis results in separate collections or databases.

- Monitor API usage and resource consumption regularly to maintain stability.