Project Title: AVPolyQuiz – Mobile Quiz Application

---

1. Introduction

1.1 Purpose of the Document

This Software Requirements Specification (SRS) document provides a complete and detailed description of the **AVPolyQuiz** mobile application. The purpose of this document is to clearly define the system requirements, functional behavior, non-functional constraints, system interfaces, and overall design considerations of the application. This document is intended for developers, project guides, examiners, testers, and future maintainers of the system.

The SRS serves as a formal agreement between stakeholders and developers, ensuring that the application is built according to defined expectations and academic standards.

1.2 Scope of the Project

AVPolyQuiz is a mobile-based quiz application developed using **React Native** and **Firebase**. The application is designed primarily for educational institutions such as polytechnics and colleges to conduct quizzes digitally.

The system supports two main roles:

- **Teacher** – creates quizzes and adds questions
- **Student** – attempts quizzes and views results

The application allows teachers to create quizzes with a time limit and multiple-choice questions (MCQs). Students can view available quizzes, attempt them in real time, and receive scores instantly.

The system aims to:

- Replace manual or paper-based quizzes
- Reduce evaluation time
- Improve transparency and accuracy
- Enable remote quiz participation

1.3 Definitions, Acronyms, and Abbreviations

Term	Description
AVPolyQuiz	Name of the quiz application
SRS	Software Requirements Specification
UI	User Interface
UX	User Experience
RTDB	Firebase Realtime Database
Auth	Firebase Authentication
MCQ	Multiple Choice Question
CRUD	Create, Read, Update, Delete

1.4 References

- Firebase Documentation
- React Native Documentation
- Google Material Design Guidelines
- IEEE 830 SRS Standard

1.5 Overview of the Document

This document is structured as follows:

- Section 2 describes the overall system
- Section 3 defines functional requirements
- Section 4 explains non-functional requirements
- Section 5 covers system architecture
- Section 6 explains database design
- Section 7 describes UI requirements
- Section 8 explains security requirements
- Section 9 covers future enhancements

---

2. Overall Description

2.1 Product Perspective

AVPolyQuiz is a standalone mobile application that operates independently using cloud-based services provided by Firebase. It follows a client-server architecture where:

- **Client**: React Native mobile application
- **Backend Services**: Firebase Authentication + Firebase Realtime Database

The app does not require a custom backend server, reducing complexity and maintenance cost.

2.2 Product Functions

The major functions of AVPolyQuiz include:

- User authentication (Student / Teacher)
- Quiz creation by teachers
- Question management (manual & JSON upload)
- Quiz listing for students
- Quiz attempt and navigation
- Automatic score calculation
- Result storage

2.3 User Classes and Characteristics

2.3.1 Teacher

- Technically proficient
- Responsible for creating quizzes
- Adds MCQ questions
- Controls quiz publication

2.3.2 Student

- Basic smartphone user
- Attempts quizzes
- Views score after submission

2.4 Operating Environment

- Platform: Android (primary), iOS (secondary)
- Development Framework: React Native CLI
- Backend: Firebase
- Database: Firebase Realtime Database
- Authentication: Firebase Auth (Email/Password)

2.5 Design and Implementation Constraints

- Internet connection required
- Firebase free-tier limitations
- Android permissions required
- Limited offline functionality

2.6 Assumptions and Dependencies

- Users have valid email IDs
- Firebase services are available
- Mobile device has internet access

---

3. Functional Requirements

3.1 User Authentication

3.1.1 Login

- Users must log in using email and password
- Authentication handled by Firebase Auth

3.1.2 Signup

- New users can register
- Role selection (Student / Teacher)

3.2 Teacher Module

3.2.1 Create Quiz

- Teacher can create a quiz
- Fields:
    - Quiz Title
    - Subject
    - Time Limit

3.2.2 Add Questions

- Manual question entry
- JSON file upload
- Each question contains:
    - Question text
    - 4 options
    - Correct option index

3.3 Student Module

3.3.1 View Quiz List

- Students can view available quizzes
- Only published quizzes are visible

3.3.2 Attempt Quiz

- Students can attempt one quiz at a time
- Navigation between questions

3.3.3 Submit Quiz

- Automatic submission on last question
- Score calculation

3.4 Result Management

- Results stored in Firebase
- Each result includes:
    - Quiz ID
    - Student ID
    - Score
    - Total questions
    - Timestamp

---

4. Non-Functional Requirements

4.1 Performance

- Quiz loading time < 3 seconds
- Real-time data synchronization

4.2 Security

- Firebase Authentication
- Secure token-based database access

4.3 Usability

- Simple and intuitive UI
- Mobile-friendly layout

4.4 Reliability

- Cloud-based backend ensures availability

---

5. System Architecture

5.1 Architecture Diagram (Logical)

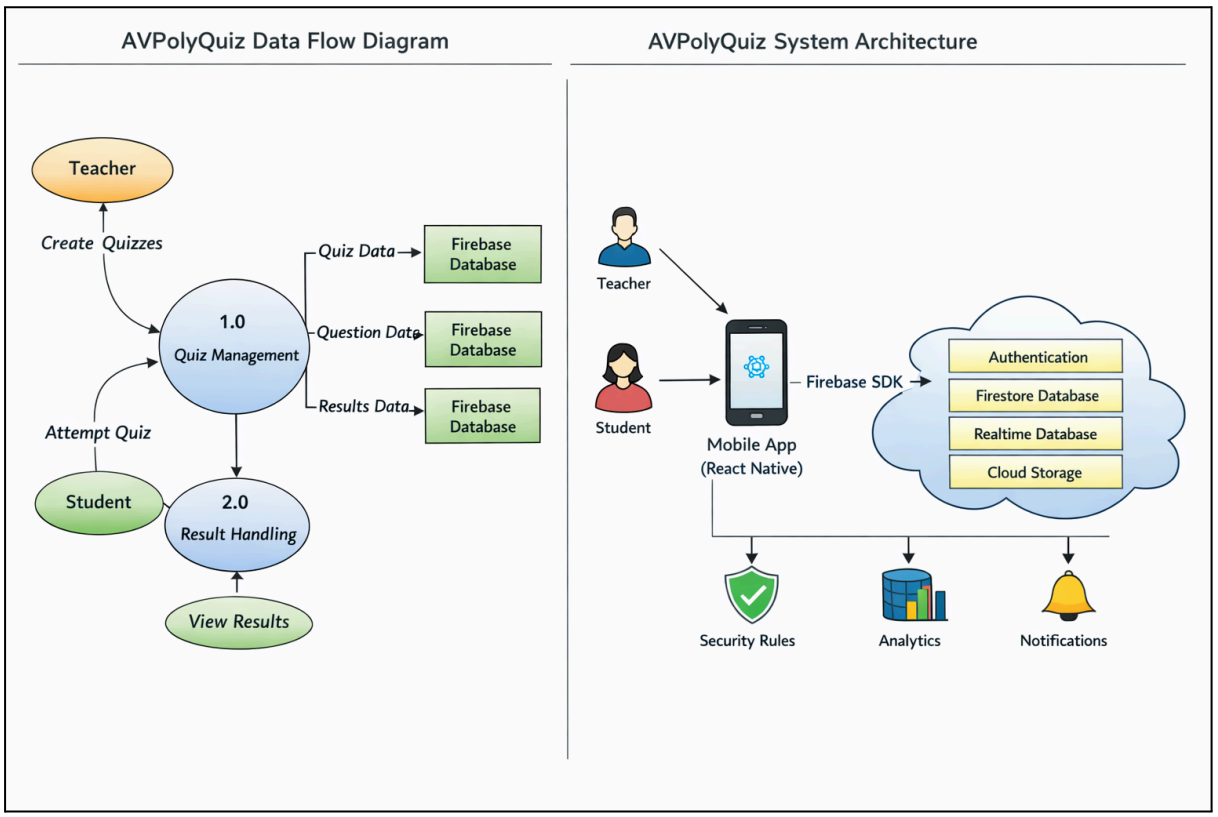
Client (React Native)

↓

Firebase Auth

↓

Firebase Realtime Database



5.2 Technology Stack

Layer	Technology
Frontend	React Native
Backend	Firebase
Database	Firebase RTDB
Auth	Firebase Auth

6. Database Design

6.1 Firebase Structure

users/{userId}
quizzes/{quizId}
questions/{quizId}/{questionId}
results/{quizId}/{userId}

---

7. User Interface Requirements

- Login Screen
- Signup Screen
- Quiz List Screen
- Quiz Attempt Screen
- Result Screen

---

8. Security Requirements

- Authentication required for all actions
- Token-based access control

---

9. Future Enhancements

- Timer-based quiz
- Leaderboard
- Admin panel
- Analytics dashboard
- Offline mode

---

10. Conclusion

AVPolyQuiz is a robust, scalable, and secure mobile quiz application suitable for academic use. The system meets modern educational requirements by providing digital assessments, real-time evaluation, and ease of use. This SRS document ensures clarity and serves as a foundation for development, testing, and future upgrades.