COLLEGE CODE: 9126

COLLEGE NAME: SRI BHARATHI ENGINEERING COLLEGE FOR WOMEN

DEPARTMENT : BE-CSE

STUDENT-NM-ID: **au912623104007**

au912623104011

au912623104013

ROLL.NO : **912623104007**

912623104011

912623104013

DATE :

COMPLETED THE PROJECT NAMED AS : IBM-NJ-ONLINE QUIZ APPLICATION

PHASE-TECHNOLOGY PROJECT NAME: NODE JS

SUBMITTED BY,

NAME: V.BRINTHA

R.DHARSHINI

R.GAYATHIRI

MOBILE NO: 9787026737

6382996269

7845079586

Phase2: Solution design and Architecture

Then stack selection

Project Requirements

- Does the app need **real-time features** (e.g., timers, live updates)?
- What types of questions and quizzes are planned?
- User roles and access control complexity.
- Scalability expectations (number of users).

2. Frontend

- **User Experience**: Choose a framework that supports responsive, clean UI (e.g., React, Vue, Angular).
- **Ease of Development**: Popular frameworks with large communities have lots of libraries and tutorials.
- **Performance**: Fast load times and smooth interactivity.
- **Mobile Friendly**: Support responsive or mobile-first design.

3. Backend

- **Language & Framework**: Node.js (Express), Python (Django/Flask), Ruby (Rails), Java (Spring) — choose what your team is comfortable with.
- **API Design**: RESTful or GraphQL APIs for flexible frontend integration.
- **Real-time Support**: If timers or live updates needed, consider WebSocket support (e.g., Socket.io with Node.js).

UI structure/API scheme design

Authentication

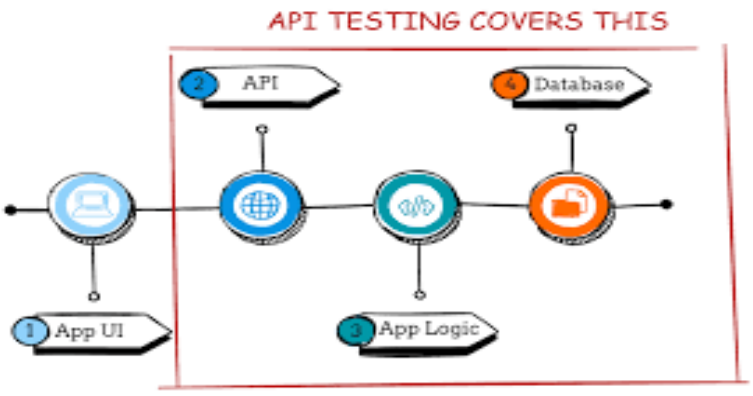
- Login Page
  - Email Input
  - Password Input
  - Login Button
  - Register Link
- Register Page
  - Name, Email, Password Inputs
  - Register Button

2. Dashboard

- **Teacher View**
  - Create Quiz Button
  - List of Quizzes (Title, Status, Actions: Edit, Delete, View Results)
- **Student View**
  - List of Available Quizzes (Title, Status, Take Quiz Button, View Score)

Authentication

Method	Endpoint	Description	Payload	Response
POST	/api/auth/register	Register a new user	{ name, email, password, role }	User info + token
POST	/api/auth/login	Login existing user	{ email, password }	User info + token
GET	/api/auth/me	Get current user info	(auth token)	User info



Data handling approach

1.Data Flow Overview

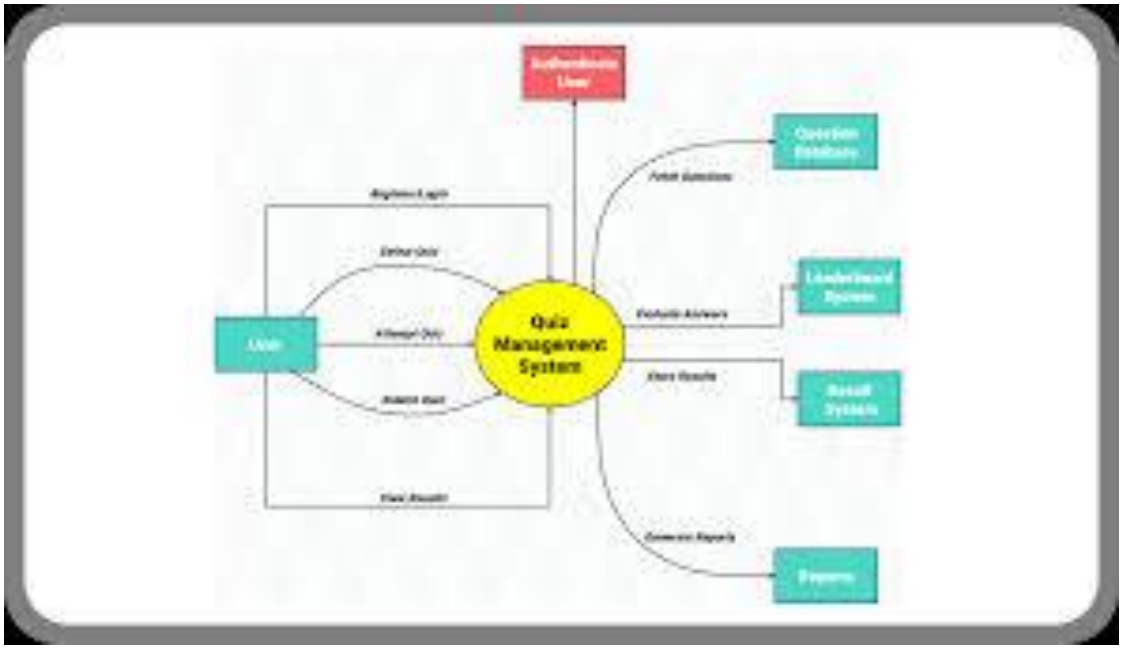
- **User Input → Frontend Validation → API Request → Backend Validation → Database Storage → Response → Frontend Display**

2. Data Types to Handle

- **User Data:** name, email, password (hashed), role (student/teacher)
- **Quiz Data:** quiz title, description, time limit, status (draft/published)
- **Question Data:** question text, type (MCQ/True-False), options, correct answer
- **Attempt Data:** user answers, start/end time, score
- **Result Data:** computed score, stats for teachers

3. Frontend Data Handling

- **Form Validation:** Ensure required fields are filled, valid formats (email, time limit numeric), option uniqueness for MCQs.
- **State Management:** Use React state or context to manage quiz/question creation flow.
- **Timer Handling:** Store remaining time in local state; sync with backend on submission.
- **Secure Data:** Don't expose correct answers in frontend before quiz completion.
- **Error Handling:** Show clear validation errors from backend or network issues.



Component/module diagram

UI Layer (Frontend)

- Built in React/Vue
- Role-based dashboards
- Pages:
  - Login/Register
  - Create Quiz
  - Take Quiz
  - View Results

2. Auth Module

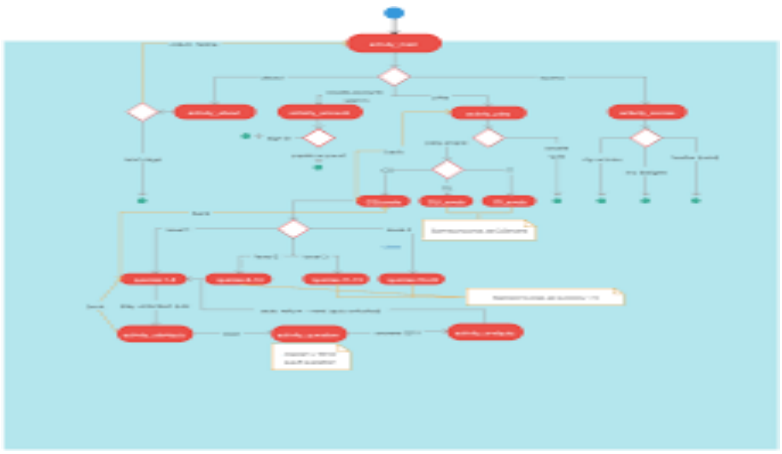
- Handles:
  - Register/Login
  - JWT Token generation/validation
  - Role-based access (Student/Teacher)

3. Quiz Module

- Teachers can:
  - Create, Edit, Delete quizzes
  - Add questions (MCQ, T/F)
- Quizzes are stored with status: draft or published

4.Question Submodule

- Nested under quizzes
- Supports:
  - Adding question text, type, options, correct answer



Basic for diagram

1.Authentication Module

- Register & Login (JWT-based)
- User roles: Student or Teache

2.Quiz Module

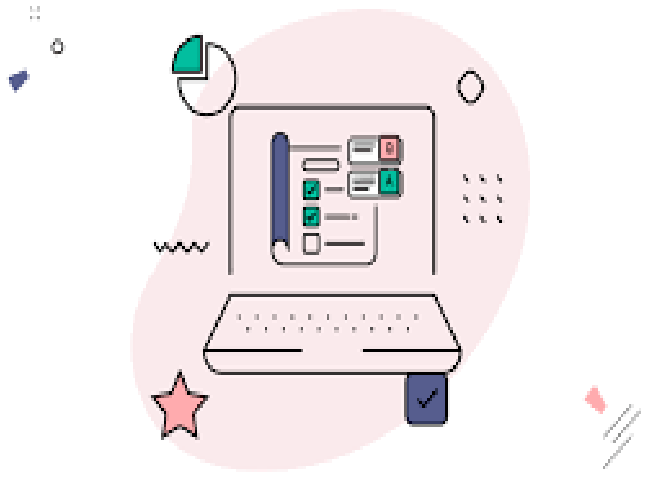
- Teachers can:
  - Create/Edit/Delete quizzes
  - Add MCQ or T/F questions
  - Publish quizzes

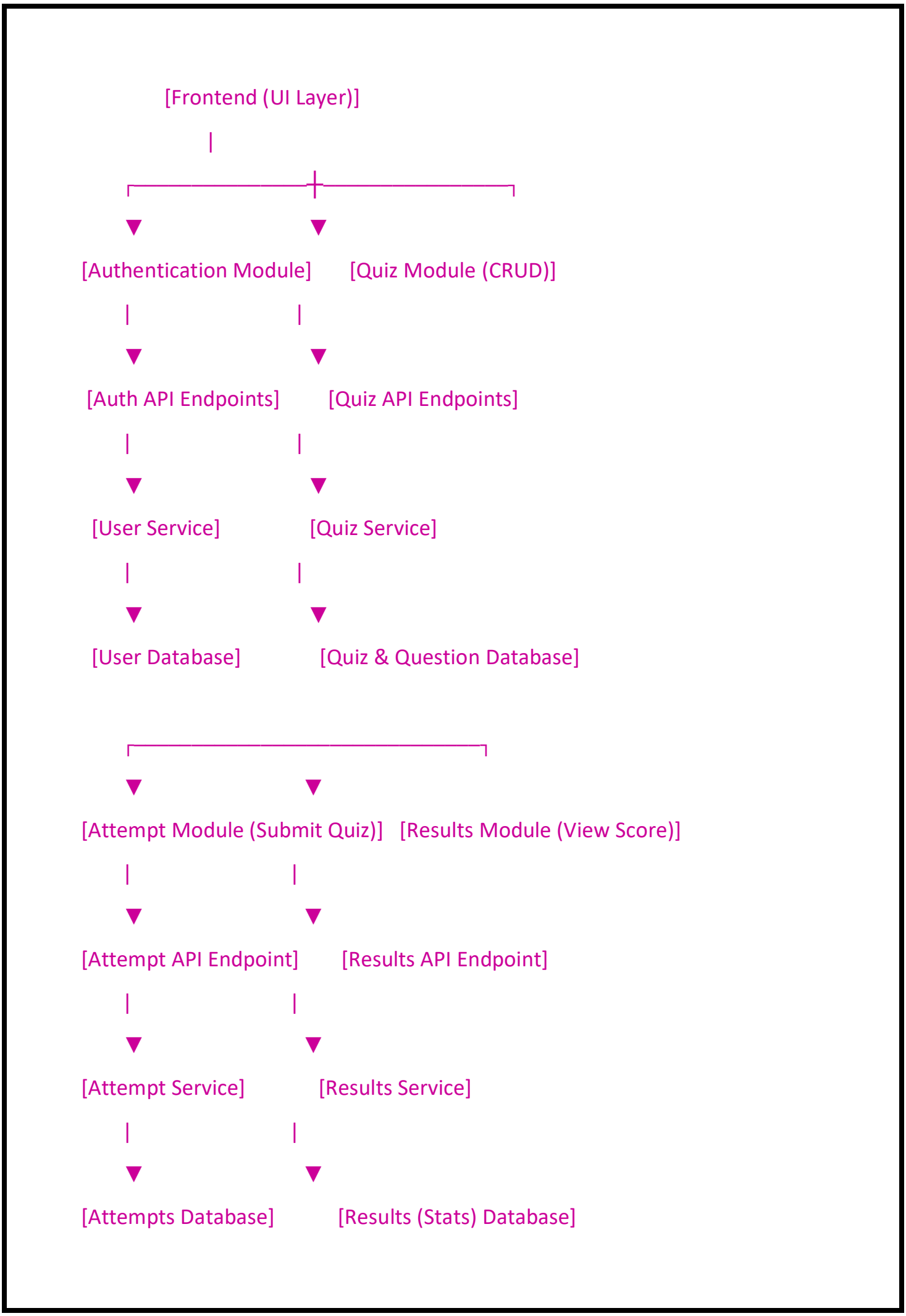
3. Attempt Module

- Students:
  - View available quizzes
  - Take and submit quiz
  - Timer-controlled submission

4. Results Module

- Auto-grading for objective questions
- Score calculation
- Stats for teachers (optional in MVP)




[Frontend (UI Layer)]
[Authentication Module]
[Quiz Module (CRUD)]
[Auth API Endpoints]
[Quiz API Endpoints]
[User Service]
[Quiz Service]
[User Database]
[Quiz & Question Database]
[Attempt Module (Submit Quiz)]
[Results Module (View Score)]
[Attempt API Endpoint]
[Results API Endpoint]
[Attempt Service]
[Results Service]
[Attempts Database]
[Results (Stats) Database]