COLLEGE CODE:[8223]

COLLEGE NAME: [vandayar engineering college]

DEPARTMENT:[Computer Science And Engineering]

STUDENT NM-ID:[1BD6CE474A7EEC39FC9493EE700D5A95]

ROLL NO:[822323104014]

DATE: [19.09.2025]

Completed the project named as

TECHNOLOGY PROJECT NAME:STUDENT GRADING SYSTEM

SUBMITTED BY,

NAME: [Mariyalivisha .A]

MOBILE NO:[9360409811]

1. Introduction

The Solution Design and Architecture phase is one of the most critical stages in software development. It bridges the gap between requirement gathering (Phase 1) and actual implementation (Phase 3). The purpose of this phase is to decide which technologies will be used (Tech Stack), how data will be handled, what the User Interface (UI) will look like, and how system components interact.

A well-structured design ensures that the project will be scalable, maintainable, and efficient. Poor design at this stage will cause high rework costs later.

---

2. Tech Stack Selection

Choosing the right technology stack (programming languages, frameworks, tools, and databases) determines the success of the system.

Criteria for Tech Stack Selection:

Scalability → Can the system handle more users in the future?

Performance → Is the technology fast enough?

Community Support → Active developer community for updates.

Security → Strong built-in security features.

Ease of Learning & Maintenance → Team's familiarity with stack.

Example for Student Grading System:

Frontend (UI) → React.js or Angular

Backend → Node.js with Express OR Python Django

Database → MySQL / MongoDB

Authentication → JWT (JSON Web Tokens)

Version Control → Git & GitHub

Deployment → AWS / Heroku

Diagram Example:

[React.js UI] → [Node.js Backend] → [MySQL Database]

2.UI Structure / API SchemaDesign

The UI structure represents how screens are arranged for different users (Students, Teachers, Admins, Parents). API schema defines how data will be transferred between frontend and backend.

UI Structure:

Login Page (Student/Teacher/Admin)

Student Dashboard → View Grades, Download Report Card

Teacher Dashboard → Upload Marks, Generate Reports

Admin Panel → Manage Users, Define Grading Rules

API Schema Design:

POST /api/auth/login – User login

POST /api/marks – Teacher enters marks

GET /api/grades/:studentId – Fetch student grades

GET /api/reports/class/:classId – Class report

This ensures clear communication between frontend & backend.

---

4. Data Handling Approach

Data is the backbone of any grading system. A clear data handling strategy ensures accuracy, security, and availability.

Storage Strategy:

Local State → Temporary (e.g., session data on frontend).

Database → Permanent storage of student records, marks, grades.

Data Flow Example:

1. Teacher enters marks → Stored in Database.

2. System applies grading rules → Calculates grade.

3. Student requests grade → API fetches data → Displayed in dashboard.

Security Measures:

Encrypt passwords (bcrypt).

Use role-based access control.

Backup database regularly.

---

5. Component / Module Diagram

Breaking down the system into smaller modules makes development manageable.

EXAMPLE MODULES:

Authentication Module (Login/Signup, JWT)

Marks Entry Module (Teacher uploads marks)

Grade Calculation Module (Apply rules → GPA/CGPA)

Reporting Module (PDF/Excel generation)

Admin Module (User Management, Grading Policy Setup)

DIAGRAM (TEXTUAL):

[Authentication]

↓

[Marks Entry] → [Grade Calculation] → [Reports]

↓

[Admin Module]

---

6. Basic Flow Diagram

A flow diagram explains the step-by-step process of how the system works.

Example Flow:

1. User logs in.

2. System verifies credentials.

3. Teacher uploads marks.

4. System calculates grades.

5. Student views results.

6. Reports generated & downloadable.

Flow Representation:

[Login] → [Authentication] → [Dashboard]

→ (If Teacher) → [Upload Marks] → [Grade Calculation]

→ (If Student) → [View Grades] → [Download Report]

→ (If Admin) → [Manage Users / Rules]

---

7. Advantages of Proper Solution Design

Reduces errors in later stages.

Improves system scalability.

Provides clear roadmap for developers.

Ensures better user experience (UI/UX).

Helps in easier debugging and testing.