1. Project Title

SkillSync – AI-Powered Career Path Navigator for Students & Fresh Graduates

2. Problem Statement

Many students and fresh graduates struggle to find the right career path due to a lack of proper guidance. They often have **scattered skills** that don't directly match a clear career, making it difficult to **choose the right field**. Traditional career counseling is limited and doesn't adapt to **personal skills and strengths**.

SkillSync solves this by using **AI-driven skill analysis** to suggest **tailored career paths**, learning roadmaps, and **additional skills** needed to transition into new fields.

---

3. Project Objectives

✅ **Analyze students' skills** (via resume or manual input) and match them with career paths.
✅ **Provide structured career roadmaps** (steps needed to reach a goal).
✅ **Suggest additional skills** to explore new career paths.
✅ **Indicate "career distance"** (how far a student is from a chosen path).
✅ **Provide learning resources** (courses, mentorship, certifications).
✅ **Help students with no clear skills** find potential fields based on their interests.

---

4. Target Users

Primary Users:

- University students exploring career options.
- Fresh graduates seeking career direction.
- Students with diverse but unstructured skills.

Secondary Users:

- Career coaches or mentors.
- Universities & career centers.(Future work)

---

5. Technologies & Frameworks

Mobile App Development (Frontend)

Language & Framework:

- **Flutter (Dart) → Cross-platform, UI-rich, and widely used.**

UI & State Management:

- **Material Design → Built-in with Flutter for modern UI.**
- **Provider → Simple and beginner-friendly state management.**

Backend Development (API & Business Logic)

Backend Framework:

- **Node.js (Express.js) → Lightweight, easy to learn, and scalable.**

API Design:

- **RESTful API → Standard approach for connecting frontend and backend.**
- **JSON Web Tokens (JWT) → Secure user authentication.**

Hosting Options:

- **Render / Railway → Free and easy hosting for Node.js apps.**

Database (Storing User Data & Career Paths)

Options:

- **Firebase Firestore (NoSQL) → If you want real-time updates and easy integration.**
- **PostgreSQL (SQL-based alternative) → If you need structured queries and relational data.**

AI & Machine Learning (For Skill & Career Matching)

ML Frameworks & Tools:

- **Hugging Face Transformers → Best for NLP-based resume & skill analysis.**
- **Scikit-learn → For career path recommendation models.**

Use Cases:

- **Extract skills from resume text.**
- **Match skills to predefined career paths.**
- **Suggest additional skills based on industry trends.**

Authentication & User Management

Authentication Provider:

- **Firebase Authentication → Google, email, and phone-based login (easy setup).**

Cloud & Hosting

Frontend & Database Hosting:

- **Firebase Hosting → Best for beginners, auto-deploy for Flutter & Firestore.**

Backend Hosting:

- **Render / Railway → Free and simple hosting for Node.js API.**

DevOps & Deployment

Version Control & CI/CD:

- **GitHub → Version control for code.**
- **GitHub Actions → Automate testing & deployment.**

Optional Deployment:

- **Docker (Optional) → If you want to containerize your backend later.**

6.Design Patterns for SkillSync

1. MVC (Model-View-Controller) – For Flutter & Node.js Backend

Why?

- **Keeps the codebase organized, modular, and easy to maintain.**
- **Separates UI (View), logic (Controller), and data (Model) for clean architecture.**

How?
Mobile App (Flutter):

- **Model → Represents user data, career paths, and skill mapping.**
- **View → UI screens (Flutter widgets).**

- **Controller → Handles user interactions and API calls.**

Backend (Node.js/Express.js):

- **Model → Database schema for users, jobs, and skills.**
- **Controller → Handles API requests, processes data, and returns responses.**
- **View → Not needed since it's an API-based backend.**

---

2. Repository Pattern – For Data Management in Flutter

Why?

- **Decouples business logic from Firebase/PostgreSQL.**
- **Makes it easier to switch databases or APIs later.**

How?

- **Repositories act as a bridge between API and UI.**
- **UI doesn't directly interact with the database but calls the repository instead.**

---

3. Singleton Pattern – For Managing Authentication & Global Services

Why?

- **Ensures only one instance of an object is created (e.g., `AuthService`).**
- **Prevents multiple redundant API connections.**

How?

- **Implements a single global instance of `AuthService` and `ApiClient`.**

---

4. Factory Pattern – For Creating User & Job Objects

Why?

- **Standardizes object creation for Users & Job Listings.**
- **Prevents redundant "if-else" logic when creating objects.**

How?

- **Factory class creates users or jobs dynamically.**

---

5. Observer Pattern – For Real-Time Updates (Job Alerts, Notifications)

Why?

- **Allows UI components to update automatically when new job alerts arrive.**
- **Ideal for real-time notifications & Firebase Firestore updates.**

How?

- **Implements Streams & Firebase Firestore listeners.**
- **Ensures UI updates instantly when a new job is posted.**

---

7. Initial High-Level System Overview

- **User registers/logs in.**
- **User uploads a resume OR manually enters skills.**
- **AI analyzes skills and suggests career paths.**
- **System provides a structured roadmap for each suggested path.**
- **App recommends additional skills to explore new career fields.**
- **User can access courses and resources to gain missing skills.**
- **System updates progress as user learns new skills.**
