Project Ideas

1. **StudyCircle** – Match students with compatible peers to form effective study groups.
2. **RoomMatch** – Find compatible roommates based on lifestyle habits and preferences.
3. **Flight Booking** – A flight booking management system that organizes reservations and flight data in one place (that's the project we did in Intro to Databases).
4. **AI Trip Planner** – Generate personalized itineraries using AI to optimize routes, costs, and activities.
5. **Seatly** – A ticket management platform for concerts, movies, and events, similar to SeatGeek but simplified.
6. **MeetingTime?**– A smarter alternative to When2Meet that finds the best meeting times across busy schedules.
7. **LinkUp** – A student-focused "LinkedIn lite" for showcasing skills, projects, and connecting with peers.
8. **CVInsight** – A resume analyzer that scores alignment with job postings and suggests improvements.
9. **EchoWall** – An anonymous feedback wall where students or teams can post thoughts and ratings safely.
10. **Calorie Tracker** – Take a picture of your food, and AI will estimate how many calories in it.
11. **RateMyStudent** – Like "RateMyProfessor," but for students to review teammates on group projects.
12. **Cover Letter Maker**– An AI-assisted cover letter generator tailored to each job description.

Project Specification: [PROJECT NAME]

Course: CS-UY 4513 - Software Engineering
Professor: Dr. DePasquale
Project Title: [PROJECT NAME]
Date: [Date]

1.0 Project Overview
[COMPLETE THIS SECTION]

2.0 Core Requirements
The project must adhere to the mandatory characteristics outlined in the project specification template document.

2.1 User-Based SystemD SUMMARIZE]

[DESCRIBE THE REQUIREMENT AND HOW YOU WILL MEET IT]

2.2 User Roles (Minimum 3)
[DEFINE EACH ROLE AND SUMMARIZE ITS FUNCTIONALITY]:

- **ROLE 1:** [DESCRIBE AND SUMMARIZE]
- **ROLE 2:** [DESCRIBE AND SUMMARIZE]
- **ROLE 3:** [DESCRIBE AN### 2.3 Persistent Storage
[DESCRIBE THE REQUIREMENT AND HOW YOU WILL MEET IT]

Proposed Database Schema (max 10 tables):
- **[table_name]:** [describe the purpose of the table]

2.4 Modular Architecture (3-5 Modules)
The system must be logically divided into a minimum of 3 and a maximum of 5 modules. These modules must demonstrate clear dependencies and rely on each other for full functionality.

Proposed Modules:
1. **[MODULE NAME]**
   - [SUMMARIZE THE FUNCTIONALITY OF THE MODULE IN A LIST]
   - **Dependency:** [DESCRIBE ANY DEPENDENCIES TO/FROM THIS MODULE]

2.5 API Interfaces
Each module must expose a RESTful API interface. This is crucial for enabling the modules to communicate with each other and for potential future integrations. The API should follow standard REST conventions.

Example Endpoints:
- **User & Identity Management API:**

  - [LIST EACH TYPE AND …]

- **Course & Assignment Management API:**
  - `GET /api/courses`
  - `POST /api/courses` (Requires Instructor/Admin role via User API)
  - `GET /api/courses/:id/assignments`

3.0 Technical Stack
- **Language:** [LIST AND DESCRIBE IF NECESSARY]
- **Framework:** [LIST AND DESCRIBE IF NECESSARY]
- **Database:** [LIST AND DESCRIBE IF NECESSARY]
- **Testing:** [LIST AND DESCRIBE IF NECESSARY]

4.0 Deliverables
The final submission must include:

1. **Complete Source Code:**
   A well-structured and commented codebase, including all necessary configuration files.

2. **Project Documentation:**
   A `README.md` file that explains the system's architecture, setup instructions, and how to run the application.

3. **API Documentation:**
   A separate document or file detailing all API endpoints, their expected parameters, and response formats.

4. **Presentation:**
   A brief presentation and demonstration of the application's core features.