

PS 1: Collaborative Task Management with AI Workflow

Brief & Case Scenario:

Case Scenario: Modern remote teams struggle with 'Task Fragmentation' where work is scattered across emails and chat apps. A typical mid-sized team loses 20% of its productive hours simply trying to figure out who is working on what and when a task is due. Traditional static lists fail to account for the dynamic nature of developer availability and the varying complexity of technical tickets.

The Challenge: Develop a real-time collaborative platform that enables teams to create, assign, and track tasks simultaneously. You must integrate an AI-driven workflow management engine that optimizes task assignments by considering team availability, skill levels, and task priority. The system must utilize WebSockets to ensure that updates are instantly visible to all stakeholders without the need for page refreshes.

Expected Outcome:

An advanced task management platform featuring real-time collaboration and AI-driven task allocation to improve team productivity and efficiency. The platform must include code management features such as check-in/check-out, forking, and version control for project documentation.

Tools & Technologies:

MERN Stack: MongoDB, Express.js, React.js, and Node.js.

Integration of WebSockets is required for real-time features, along with specialized AI/ML libraries for task optimization algorithms.

Technical Constraints & Requirements:

- Real-time collaboration using WebSockets
- Secure Role-Based Access Control (RBAC)
- AI-assisted task allocation based on priority and skillset
- Automated notifications for deadlines and task progress

Focus: Full-Stack Development, AI Integration, and Real-Time Systems.

PS 2: AI-Powered Job Portal with Resume Gap Analysis

Brief & Case Scenario:

Case Scenario: The job market is flooded with generic resumes, making it impossible for employers to find the right fit manually. Simultaneously, candidates often lack the specific niche skills required for high-paying roles but have no clear roadmap on how to bridge that gap. This mismatch leads to high unemployment in some sectors while tech roles remain vacant.

The Challenge: Build an intelligent job portal that uses AI to parse resumes and match them against job descriptions with high precision. The platform must go beyond simple matching; it should perform a 'Skill Gap Analysis' for the user and provide customized learning suggestions (courses/certifications) based on the specific job roles they are targeting.

Expected Outcome:

A fully functional job portal where employers can post openings and receive AI-ranked candidate lists. For users, the platform provides an interactive dashboard showing their 'Hireability Score' and a personalized learning path to reach their dream role.

Tools & Technologies:

MERN Stack (MongoDB, Express, React, Node) and Python-based NLP libraries for resume parsing.

Technical Constraints & Requirements:

- AI-driven resume screening and feedback generation
- Personalized learning recommendations based on gap analysis.
- Secure user profiles and employer job-posting portals.
- Automatic candidate ranking for recruiters.

Focus: Full-Stack Development, AI Integration, and Real-Time Systems.

PS 3: Gen-AI Personal Productivity Assistant

Brief & Case Scenario:

Case Scenario: Professionals are overwhelmed by 'Information Overload,' managing multiple calendars, thousands of emails, and endless meeting notes. Valuable time is wasted in administrative overhead-summarizing discussions or drafting routine correspondence-leaving little room for deep, creative work.

The Challenge: Create a Generative AI-powered personal assistant that integrates into a user's daily workflow. The application must be capable of summarizing long documents, taking meeting notes via NLP, and drafting professional emails based on brief user prompts. It should act as a proactive 'Day Planner' that sets reminders and manages schedules through natural language interactions.

Expected Outcome:

A smart assistant web app that offers personalized recommendations based on past user behavior and preferences. It must feature a clean interface for text and voice-based interactions to help users organize their professional life efficiently.

Tools & Technologies:

Python, Flask/Django, TensorFlow, NLTK/SpaCy, and Generative AI Frameworks.

Technical Constraints & Requirements:

- NLP for conversational task management.
- AI model training for personalizing user suggestions.
- Summarization of documents and meeting notes.
- Integration of email drafting and schedule management.

Focus: Full-Stack Development, AI Integration, and Real-Time Systems.

PS 4: AI-Generated Adaptive Quiz & Learning Platform

Brief & Case Scenario:

Case Scenario: Static online courses often lead to low engagement because the assessment (quizzes) is the same for every student, regardless of their proficiency. High-performing students find them boring, while struggling students find them discouraging, leading to high drop-out rates in traditional e-learning environments.

The Challenge: Develop an innovative learning platform where the curriculum is static but the assessments are 'Dynamic'. Use Generative AI to create unique sets of quiz questions every time a user takes a test, ensuring that no two students get the same questions. The AI should manipulate question difficulty based on the user's progress and previous quiz performance.

Expected Outcome:

An e-learning portal with a real-time progress dashboard and an AI engine that generates unique quizzes on-the-fly. The system must provide instant feedback and adapt the learning path based on quiz results.

Tools & Technologies:

MERN Stack and Generative AI Models (GPT-API or similar)

Technical Constraints & Requirements:

- Dynamic quiz generation and question manipulation.
- Progress tracking with personalized learning paths.
- Support for multiple courses with diverse topics.
- Interactive UI with timers and instant result feedback.

Focus: Full-Stack Development, AI Integration, and Real-Time Systems.

PS 5: Smart Complaint Management with Chatbot Integration

Brief & Case Scenario:

Case Scenario: Customer service departments are bogged down by repetitive 'Level 1' complaints (e.g., 'Where is my order?' or 'Reset my password'). These simple queries delay the resolution of complex, high-priority issues, leading to poor customer satisfaction and high operation costs.

The Challenge: Build a sophisticated complaint management system integrated with an AI chatbot that acts as the first point of contact. The chatbot should attempt to resolve the issue using NLP; if it fails, it must automatically generate a ticket and categorize it (e.g., Technical, Billing) using AI. The system must intelligently prioritize and assign these tickets to agents based on the 'Severity Level' detected in the user's language.

Expected Outcome:

A seamless end-to-end support system where users can track their ticket status in real-time via the chatbot. It must include an Admin Dashboard for managing tickets and viewing analytics on resolution times.

Tools & Technologies:

MERN Stack, NLP Libraries (Dialogflow/Rasa/SpaCy), and ML Libraries (TensorFlow).

Technical Constraints & Requirements:

- Chatbot-based complaint registration using NLP.
- AI-driven ticket prioritization and dynamic assignment.
- Real-time status tracking for users (Registered, In Progress, Resolved).
- Automated escalation mechanism for high-priority unresolved issues.

Focus: Full-Stack Development, AI Integration, and Real-Time Systems.

PS 6: Enterprise Knowledge Base with Semantic Search

Brief & Case Scenario:

Case Scenario: Large organizations have vast amounts of internal documentation (HR policies, technical manuals, brand kits) spread across different portals. Employees often waste hours searching for a single document because keyword-based searches are inaccurate and don't understand the 'intent' behind the search query.

The Challenge: Develop a centralized 'Knowledge Hub' that uses Semantic Search (AI that understands context) to retrieve information. The application should allow admins to upload documents (PDFs/Docs) and allow users to ask questions in natural language, receiving direct answers extracted from the company's specific knowledge base instead of generic web results.

Expected Outcome:

A secure internal portal with a Google-like search interface that understands organization-specific terminology. The system should provide a 'Source Citation' for every answer it gives to ensure reliability and trust among employees.

Tools & Technologies:

Python (Django/Flask), Vector Databases (Pinecone/Milvus), MERN for the UI, and LLM Frameworks (LangChain).

Technical Constraints & Requirements:

- Support for multiple document formats (PDF, DOCX, TXT).
- Semantic search capabilities (Context-aware results).
- User authentication and role-based access for sensitive documents.
- Admin dashboard for document management and search analytics.

Focus: Full-Stack Development, AI Integration, and Real-Time Systems.