**Agent-Based Automation Platform: Project Documentation**

**Table of Contents**

1. **Overview**
2. **Core Components and Features**
3. **Tech Stack**
4. **Phase-wise Development Plan**
5. **Database Models (MongoDB)**
6. **Initial Seed Data**
7. **Authentication and RBAC**
8. **Front-End UI Design**
9. **Milestones and Timeline**

**1. Overview**

This document outlines the complete project plan, core components, technology stack, and phased milestones for the Agent-Based Automation Platform. It serves as the blueprint for all phases of development, starting with foundational modules (Phase 1) and scaling up to agent-based workflows, chat modules, and LLM integration (Phase 2).

**Goals**

1. Build secure, scalable components that include user identity, authentication, and CRUD functionalities.
2. Develop dynamic, multi-theme UI with a minimalistic design similar to OpenAI/ChatGPT.
3. Implement a structured, modular architecture to accommodate future integrations (LLMs, LangChain, agent workflows).
4. Phase-wise development focusing on incremental, testable deliverables.

**2. Core Components and Features**

The following table summarizes the core components and their key features:

| **Component** | **Description** |
| --- | --- |
| **Agent Management** | Create, manage, and configure agents with modular capabilities and workflows. |
| **Capabilities Engine** | Manage capabilities like summarization, generation, and custom tasks. |
| **AI Model Management** | Integrate API-based and hosted LLMs (e.g., OpenAI, Qwen) with LangChain. |
| **Chat Module** | Advanced chat interface for agent collaboration and dynamic responses. |
| **Authentication & RBAC** | Secure login system, user management, and role-based permissions. |
| **Dynamic UI (Multi-Theme)** | Multi-theme, minimalistic frontend with mobile responsiveness. |
| **API Integration** | Secure APIs for communication between microservices. |
| **External Systems Integration** | Support Slack, Teams, and Discord for external workflows. |
| **Containerization** | Dockerize all services to ensure consistent deployments. |
| **Monitoring and Logs** | (Planned for future) Track usage, logs, and system performance. |

**3. Tech Stack**

| **Layer** | **Technology** |
| --- | --- |
| **Backend** | Node.js (Express.js) |
| **AI Services** | Python (FastAPI) |
| **Database** | MongoDB |
| **Authentication** | JWT (encrypted) |
| **Frontend** | Angular + TailwindCSS |
| **Containerization** | Docker |
| **Caching** | Redis |
| **Communication** | Kafka |
| **Deployment** | Local/Docker Compose |

**4. Phase-wise Development Plan**

**Phase 1: Foundational Modules**

**1. Database Models and Services**

* Design MongoDB models for foundational modules:
  + **Users**
  + **Roles and Permissions**
  + **Agents**
  + **Capabilities**
  + **AI Models**
  + **Activity Logs**
* Create controllers and services for full CRUD functionality.
* Expose REST APIs for these services.

**2. Initial Seed Data**

* Implement a **Node.js seed script** to populate MongoDB with initial data:
  + Default roles (Admin, User).
  + Sample agents, capabilities, and models.
  + Basic user accounts.

**3. Authentication and RBAC**

* Secure microservices using **encrypted JWT tokens**.
* Implement Role-Based Access Control (RBAC):
  + Roles: Admin, User.
  + Permissions: CRUD access to modules.
* Create:
  + **Login** API.
  + **Sign-Up** form for user onboarding.

**4. Front-End Development**

* Build Angular frontend with:
  + **Login and Sign-Up Pages**.
  + **Dynamic Multi-Theme Support** (dark/light themes).
  + **RBAC Integration**: Users can only access allowed features.
  + **CRUD UI**:
    - Agents
    - Capabilities
    - AI Models
    - User management.
* Ensure full **mobile responsiveness**.

**5. Containerization**

* Containerize Node.js backend, Python services, and MongoDB using **Docker Compose**.

**Phase 2: Advanced Modules**

* Develop a **Chat Module** with a rich, interactive UI.
* Integrate LangChain and LLM services (API-based and hosted models).
* Build agent workflows with multi-agent collaboration capabilities.
* Add external integrations for Slack, Discord, and Teams.
* Introduce monitoring and usage logs (if required).

**5. Database Models (MongoDB)**

**1. Users**

{

"\_id": ObjectId, // Unique user ID

"name": "John Doe", // Full name

"email": "john@example.com", // User email

"password": "hashed\_password", // Hashed password for security

"role": "Admin", // Role: Admin or User

"createdAt": ISODate(), // Account creation timestamp

"updatedAt": ISODate() // Last updated timestamp

}

**2. Roles**

{

"\_id": ObjectId, // Unique role ID

"name": "Admin", // Role name

"permissions": [ // List of permissions

"create",

"read",

"update",

"delete"

]

}

**3. Agents**

{

"\_id": ObjectId, // Unique agent ID

"name": "Summarizer Agent", // Agent name

"description": "Agent for summarizing documents", // Description

"model": "GPT-4", // Associated AI model

"capabilities": ["summarize", "generate"] // List of capabilities

}

**4. Capabilities**

{

"\_id": ObjectId, // Unique capability ID

"name": "Summarization", // Capability name

"description": "Summarizes text inputs", // Description

"tools": ["LangChain", "OpenAI"] // Tools used for capability

}

**5. AI Models**

{

"\_id": ObjectId, // Unique model ID

"name": "GPT-4", // Model name

"type": "API-Based", // Type: API-Based or Hosted

"endpoint": "api.openai.com", // API endpoint (if applicable)

"description": "OpenAI GPT-4 API" // Model description

}

**6. Initial Seed Data**

* **Roles**: Admin, User.
* **Users**: Default admin account.
* **Sample Agents**: Summarizer Agent, Generator Agent.
* **Capabilities**: Summarization, Text Generation.
* **AI Models**: GPT-4, GPT-3.5-turbo.

**7. Authentication and RBAC**

* **JWT Authentication**:
  + Encrypted tokens for microservice protection.
* **RBAC**:
  + Permissions assigned based on roles.
  + Secure API endpoints with middleware validation.

**8. Front-End UI Design**

* **Technologies**: Angular + TailwindCSS.
* **Features**:
  + **Dynamic Multi-Themes**: Dark/Light themes via TailwindCSS.
  + **Responsive Design**: Mobile-friendly layout.
  + **Minimalistic UI**: Similar to OpenAI/ChatGPT.
* **Core Pages**:
  + Login/Sign-Up
  + Dashboard
  + Agents Management (CRUD)
  + Capabilities Management (CRUD)
  + User Management

**9. Milestones and Timeline**

**Phase 1 Milestones**

| **Milestone** | **Expected Completion** |
| --- | --- |
| MongoDB Models and CRUD APIs | Week 1 |
| Initial Seed Data Script | Week 1 |
| Authentication and RBAC Implementation | Week 2 |
| Front-End: Login, Sign-Up, Theme Support | Week 2 |
| Agents, Capabilities CRUD Front-End | Week 3 |
| Containerization with Docker | Week 3 |
| Integration Testing and Debugging | Week 4 |

**Phase 2 Milestones *(For Future Reference)***

| **Milestone** | **Expected Completion** |
| --- | --- |
| Chat Module with Multi-Agent Invocation | TBD |
| LangChain LLM Integration | TBD |
| Real-Time Agent Collaboration | TBD |
| External Systems Integration | TBD |

**Conclusion**

This document provides the complete project overview, core components, database models, and a detailed Phase 1 plan. Subsequent phases will focus on advanced features like chat modules, agent collaboration, and LLM integration. This document serves as the foundational reference for development and future discussions.