Detailed MVP Document: AI Agent for Strategy

# Overview

The AI Agent for Strategy MVP is designed to streamline OKR management, meeting notetaking (with Microsoft Teams integration), and task tracking, enhanced with AI capabilities for transcription and intelligent suggestions. This version includes company and user management for better organization and scalability.

# Core Features

**1. Company and User Management**

* **Features**:
  + Allow company creation.
  + Associate users with specific companies.
  + Users receive passwords to log in after being added by the admin.
* **Schema Design**:
  + **Company Schema**:



* + **User Schema**:
  + const teams = require('./path/to/teams.json');
  + const UserSchema = new mongoose.Schema({
  + name: { type: String, required: true },
  + email: { type: String, required: true, unique: true },
  + company: { type: mongoose.Schema.Types.ObjectId, ref: 'Company', required: true },
  + role: { type: String, enum: ['admin', 'user'], default: 'user' },
  + l1Team: { type: String, enum: teams.l1Teams, required: true },
  + l2Team: { type: String, enum: teams.l2Teams, required: true },
  + createdAt: { type: Date, default: Date.now },
  + updatedAt: { type: Date, default: Date.now }

});

* **Workflow**:
  + **Company Creation**:
    - An admin creates a company.
    - The admin is automatically added as the first user.
  + **User Management**:
    - Admins can add users to their company.
    - Users receive email invites with initial login credentials.

**2. User Authentication**

* **Description**:
  + Secure login and registration for users.
  + Session management to control access.
* **Implementation**:
  + Use passport and passport-local-mongoose for authentication.
  + Store user credentials in **MongoDB** or **SQLite**.
* **Endpoints**:
  + /register (POST): User registration.
  + /login (POST): User login.
  + /logout (GET): User logout.

**3. OKR Management**

* **Features**:
  + Create, edit, and manage objectives and key results.
  + Provide feedback if the KR is not SMART.
* **Implementation**:
  + **Objective Schema**:
  + const ObjectiveSchema = new mongoose.Schema({
  + objective: { type: String, required: true },
  + keyResults: [{ type: mongoose.Schema.Types.ObjectId, ref: 'KeyResult' }],
  + owner: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },
  + company: { type: mongoose.Schema.Types.ObjectId, ref: 'Company', required: true },
  + fromDate: { type: Date, required: true },
  + toDate: { type: Date, required: true },
  + createdAt: { type: Date, default: Date.now },
  + updatedAt: { type: Date, default: Date.now }

});

* + **Key Result Schema**:
  + const KeyResultSchema = new mongoose.Schema({
  + description: { type: String, required: true },
  + tasks: [{ type: mongoose.Schema.Types.ObjectId, ref: 'Task' }],
  + status: { type: String, enum: ['not started', 'in progress', 'completed'], default: 'not started' },
  + fromDate: { type: Date, required: true },
  + toDate: { type: Date, required: true },
  + progress: { type: Number, default: 0, min: 0, max: 100 },
  + atRisk: { type: Boolean, default: false },
  + objective: { type: mongoose.Schema.Types.ObjectId, ref: 'Objective', required: true },
  + createdAt: { type: Date, default: Date.now },
  + updatedAt: { type: Date, default: Date.now }
  + });
  + KeyResultSchema.pre('save', function (next) {
  + if (this.fromDate < this.objective.fromDate || this.toDate > this.objective.toDate) {
  + return next(new Error('Key Result dates must be within the Objective\'s date range.'));
  + }
  + next();

});

* **Endpoints**:
  + /okr/create (POST): Save a new OKR.
  + /okr/view (GET): Retrieve all OKRs for a company.
  + /okr/update/:id (PUT): Update an existing OKR.
  + /okr/delete/:id (DELETE): Delete an OKR.

**4. Meeting Notes Management with Microsoft Teams Integration**

* **Features**:
  + Fetch meeting recordings from Teams using Microsoft Graph API.
  + Transcribe audio to text using **Whisper** (open-source).
  + Summarize notes to highlight key points and action items.
  + Link meeting notes to OKRs and provide tagging suggestions.
  + Auto-tag meeting notes to Objectives if no manual tag is provided.
  + Add manual text entry for meeting notes.

**5. Action Items**

* **Task Schema**:
* const TaskSchema = new mongoose.Schema({
* description: { type: String, required: true },
* assignee: { type: mongoose.Schema.Types.ObjectId, ref: 'User' },
* status: { type: String, enum: ['not started', 'in progress', 'completed'], default: 'not started' },
* deadline: { type: Date },
* keyResult: { type: mongoose.Schema.Types.ObjectId, ref: 'KeyResult', default: null },
* company: { type: mongoose.Schema.Types.ObjectId, ref: 'Company', required: true },
* atRisk: { type: Boolean, default: false },
* createdAt: { type: Date, default: Date.now },
* updatedAt: { type: Date, default: Date.now }

});

**Technical Architecture**

**Frontend:**

* **HTML/CSS/JavaScript**:
  + Build simple forms for company management, OKR creation, meeting notes, and action items.
  + Dashboard with tabs for key features.

**Backend:**

* **Node.js + Express**:
  + Serve as the API layer for data operations and integrations.
  + Integrate Microsoft Graph API for meeting recordings.

**Database:**

* **Primary**: MongoDB community edition for cloud support.
* **Alternative**: SQLite for lightweight local use.

**NLP and AI:**

* **Whisper (Open Source)**: For audio-to-text transcription.
* **Hugging Face**: For summarization and tagging suggestions.

**Automation:**

* **Nodemailer**: Send reminders and notifications.
* **Agenda.js**: Schedule tasks like sending automated emails.

**Implementation Timeline**

**Week 1–2:**

* Set up backend, database, and user authentication.
* Implement company and user management.

**Week 3:**

* Integrate Microsoft Teams API.
* Implement transcription using Whisper.

**Week 4:**

* Add NLP for summarization and tagging.
* Build chatbot logic for OKR suggestions.

**Week 5:**

* Test and debug all features.
* Optimize backend and database queries.

**Week 6:**

* Deploy the app, gather feedback, and finalize based on user input.

**Cost-Saving Measures**

* Use **open-source tools** like Whisper and Hugging Face.
* Leverage free-tier cloud services (e.g., MongoDB Atlas, Render).
* Delay premium API usage until post-MVP.