System Architecture Plan

**Frontend**

* **Technology**: React.js
  + **Reason**: A robust and scalable framework for building dynamic and responsive user interfaces.
  + **Responsibilities**:
    - User registration/login forms.
    - Application management interface (CRUD operations).
    - Dynamic dashboards for visualizing job application progress.
    - Collaboration tools for teams.
    - Filters and advanced search functionalities.

**Backend**

* **Technology**: Django (Python)
  + **Reason**: A secure and scalable framework with built-in features for authentication, routing, and API management.
  + **Responsibilities**:
    - API endpoints for managing applications, reminders, and user interactions.
    - Integration with AI-powered recommendation models.
    - Business logic for user permissions and collaboration.
    - Automated status updates via job portal and email parsing.

**Database**

* **Technology**: PostgreSQL
  + **Reason**: A powerful, open-source relational database system that supports advanced queries and data integrity.
  + **Schema Design**:
    - Tables for users, job applications, reminders, AI insights, and team collaborations.
    - Relationships via foreign keys (e.g., user ownership of applications).

**AI Layer**

* **Technology**: TensorFlow or Scikit-learn
  + **Reason**: Suitable for building and training lightweight predictive models.
  + **Responsibilities**:
    - Analyze trends in job applications.
    - Recommend job opportunities, companies, or certifications.
    - Provide actionable insights via APIs.

**Integration Layer**

* **Communication Protocol**: REST API
  + **Endpoints**:
    - /signup and /login for user authentication.
    - /applications for job application CRUD operations.
    - /dashboard/summary for data visualization.
    - /ai/recommendations for AI insights.

**Deployment**

* **Frontend Hosting**: Vercel or Netlify
* **Backend Hosting**: AWS or GCP
* **Database Hosting**: AWS RDS or GCP Cloud SQL
* **Additional Features**:
  + Secure communication via HTTPS.
  + Scalable deployment to handle growth.

**Architecture Diagram**

* A high-level diagram showing:
  + **Frontend**: React.js components connecting to backend APIs.
  + **Backend**: Django handling logic and interfacing with AI and the database.
  + **Database**: PostgreSQL managing user and application data.
  + **AI Layer**: Exposed via backend endpoints for insights and recommendations.
  + **Deployment**: Hosted on cloud platforms with secure communication.