Project Document: NicheTalentDB (Beta Version)

1. Project Objective

To build a centralized database system to store, parse, and update resumes of non-technical or niche candidates (sales, operations, etc.) with a user-friendly dashboard for internal management.

Key Goals:

- - Store resumes securely in a structured database.
- - Parse resumes automatically to extract candidate information (education, experience, skills, etc.).
- - Track last update and support bi-annual refreshes.
- - Provide a spreadsheet-like visualization for quick filtering and insights.
- - Enable manual addition of market data (e.g., salary, availability).
- - Maintain resume attachments and allow versioning/replacement.

2. Target Users

- - Internal HR/recruiting teams.
- - Businesses or startups looking to buy curated candidate databases.
- - Users managing niche profiles where public ATS data is limited.

3. Version 1 – Core Features

3.1 Dashboard

Candidate Table (Spreadsheet View)

- - Columns: Name, Email, Phone, Location, Primary Skill, Experience, Current Salary, Resume Last Updated, Availability.
- - Search & filter functionality (by skill, location, experience, etc.).
- - Sortable columns.

Candidate Popup / Detail View

- - Full resume data (parsed fields).
- - Resume attachment (PDF, DOCX).
- - Salary & market information (manual input).
- - Last update date.
- - Option to update/replace resume.

Add Resume Section

- - Drag-and-drop upload or browse files.
- - Automatic parsing:
- - Name, contact info, education, work experience, skills, certifications.
- - System checks:
- - If candidate exists → Option to replace/update if >6 months.
- - If new → Add as a new entry.

Resume Reupdate Reminder
- - Flag candidates whose resume has not been updated in >6 months.
- - Notification in dashboard for follow-up.

3.2 Backend – Database Structure

Candidate Table
- - candidate_id (primary key)
- - name
- - email
- - phone
- - location
- - skills (JSON array)
- - education (JSON array)
- - work_experience (JSON array)
- - last_resume_update (timestamp)
- - salary_info (JSON array: {period, salary, market_reference})
- - availability_status (available/not available)
- - resume_file_link (path or blob)

Resume Version Table
- - version_id
- - candidate_id
- - resume_file_link
- - parsed_data_snapshot (JSON)
- - uploaded_on (timestamp)

Market Data Table
- - skill
- - average_salary
- - market_demand
- - last_updated

3.3 Resume Parsing

- - Extract:
- - Personal Info: Name, email, phone, location
- - Education: Degree, institution, graduation year
- - Work Experience: Role, company, duration
- - Skills & Certifications
- - Store parsed data as structured JSON for easy visualization and reporting.

3.4 Versioning / Updates

- - Check if candidate exists.
- - If yes:
- - Check last update date.

- - If >6 months → Suggest update.
- - Keep previous version in Resume Version Table.
- - If no → Add new entry.
- - Manual override for replacing resume or updating market data.

3.5 Frontend – Beta Dashboard

- - Web-based, responsive UI.
- - Components:
- 1. Spreadsheet View

2. Add Resume Modal
3. Candidate Detail Modal
4. Update Reminder Widget

- - Optional: Export to Excel/CSV.

3.6 Security & Access

- - User login with roles (admin, read-only).
- - Resume storage encrypted.
- - Audit log for resume updates and uploads.

4. Technology Stack (Version 1)

- - **Frontend:** React.js or Vue.js
- - **Backend:** Python (Flask/FastAPI) or Node.js
- - **Database:** PostgreSQL or MongoDB (for flexible JSON storage)
- - **Resume Parsing:** Python NLP libraries (spaCy, PyPDF2, pdfminer, docx) or Resume Parser API
- - **File Storage:** AWS S3 / Azure Blob / Google Cloud Storage
- - **Deployment:** Dockerized for quick prototype testing

5. Team Requirements

- - **Frontend Developer (1-2)**: React/Vue, UI/UX design
- - **Backend Developer (1-2)**: Python/Node.js, REST API development
- - **Database Engineer (1)**: SQL/NoSQL, schema design, indexing
- - **Data Engineer/NLP Specialist (1)**: Resume parsing, JSON structure
- - **DevOps Engineer (1)**: Cloud deployment, Docker, CI/CD
- - **QA Engineer (1)**: Testing of uploads, parsing, UI validation

6. Cloud & Storage Platforms

- - AWS S3 / Azure Blob / GCP Storage for resume files
- - PostgreSQL or MongoDB hosted on AWS RDS / Atlas
- - Docker for containerized deployment
- - Optional: Cloud monitoring tools (AWS CloudWatch / Azure Monitor)

7. Milestones for Beta Version

1. **Week 1–2:** Database setup & schema design
2. **Week 2–3:** Resume upload + parsing functionality
3. **Week 3–4:** Dashboard spreadsheet view + candidate popup
4. **Week 4–5:** Versioning logic & 6-month update reminders
5. **Week 5–6:** Testing, bug fixes, initial demo

8. Future Enhancements

- - Analytics dashboard (skills in demand, salary trends)
- - API access for clients
- - Automated salary benchmark updates
- - Role-based access control and subscription model
- - Integration with LinkedIn or public directories