# AI-Driven Automated Bench Sales System - MVP Project Plan

## Executive Summary (MVP Focus)

The MVP for the AI-Driven Automated Bench Sales System will establish a foundational platform for core bench sales activities. Over 30 days, we will deliver:

1. **User Management**: Secure user registration, login, and basic role-based access for Admin, Profile Managers, and Job Seekers.
2. **Basic Job Discovery**: Automated scraping from a *single* job board (Indeed for MVP) with keyword filtering and storage.
3. **Basic Contact Extraction**: Automated extraction of contact names and emails *directly from scraped job postings* (no external enrichment APIs in MVP).
4. **Manual Email Outreach**: Ability for Profile Managers to send *single, personalized emails* to extracted contacts (no sequences or AI-generated content in MVP).
5. **Core Data Storage**: Robust PostgreSQL database for all system entities.
6. **Essential UI**: Basic dashboards for users to view relevant data and interact with core features.
7. **System Health**: Basic monitoring endpoints.

This MVP prioritizes core workflow automation and data persistence, laying the groundwork for future AI and advanced automation features.

## Assumptions & Constraints

* **Team Size**: 2-3 dedicated developers (1-2 Backend, 1 Frontend).
* **Technology Stack**: FastAPI (Python) for Backend, React.js (TypeScript) for Frontend, PostgreSQL for Database. Docker for containerization.
* **Cloud Provider**: AWS/GCP (pre-configured accounts assumed).
* **External APIs**: Only a single job board (Indeed) scraper and one transactional email API (e.g., SendGrid/Mailgun) will be integrated for the MVP. No external contact enrichment APIs (Apollo, PDL, Hunter) or AI APIs (OpenAI for optimization/generation) will be used in the MVP, as these are Phase 2 features.
* **Scope Freeze**: No new features will be added during the 30-day MVP period.
* **Testing**: Focus on unit and basic integration tests. End-to-end and advanced performance testing will be in later phases.
* **Deployment**: Basic CI/CD, likely manual deployment to a single environment for MVP.

## Team Structure

* **Project Lead / Architect**: Oversees overall technical direction, critical design decisions.
* **Backend Developer(s)**: Responsible for FastAPI APIs, database interactions, core logic (scraping, contact extraction, email sending).
* **Frontend Developer(s)**: Responsible for React UI, API integration, user experience.

## Backlog Stories (MVP Focus)

**Epic 1: User Management & Authentication (High Priority)**

* **US 1.1: User Registration**
  + **Story**: As a new user, I want to register for an account with a username, email, and password so that I can access the system.
  + **Tasks**:
    - BE: Design User table schema (id, username, email, password\_hash, roles, created\_at, updated\_at).
    - BE: Implement password hashing (Bcrypt).
    - BE: Create POST /api/v1/users/register endpoint (FastAPI model UserCreate).
    - FE: Design Registration form UI.
    - FE: Implement API call for registration.
    - FE: Handle registration success/failure.
* **US 1.2: User Login & Session**
  + **Story**: As a registered user, I want to log in using my credentials so that I can access my personalized dashboard and features.
  + **Tasks**:
    - BE: Implement JWT generation logic (access token).
    - BE: Create POST /api/v1/auth/token endpoint (FastAPI model UserLogin, OAuth2PasswordBearer).
    - BE: Implement JWT verification dependency.
    - FE: Design Login form UI.
    - FE: Implement API call for login.
    - FE: Store JWT securely (e.g., HTTP-only cookies, localStorage for MVP).
    - FE: Implement client-side routing protection.
* **US 1.3: User Profile & Roles**
  + **Story**: As an authenticated user, I want to view my profile details, and as an Admin, I want to ensure roles are assigned correctly.
  + **Tasks**:
    - BE: Implement GET /api/v1/users/me endpoint.
    - BE: Implement basic RBAC logic (Depends on role checks for endpoints).
    - FE: Design basic User Profile display UI.
    - FE: Display user's assigned role.

**Epic 2: Basic Job Discovery (MVP)**

* **US 2.1: Manual Job Scraping Trigger**
  + **Story**: As a profile manager, I want to manually trigger job scraping from Indeed with specific keywords and locations so that I can find initial job opportunities.
  + **Tasks**:
    - BE: Design Job table schema (id, title, description, company, location, url, source, posted\_date, created\_at, updated\_at).
    - BE: Implement basic Indeed scraper logic (Python requests + BeautifulSoup/Playwright for simple DOM parsing).
    - BE: Create POST /api/v1/jobs/scrape endpoint (input: source, keywords, location).
    - BE: Store scraped jobs in Job table.
    - BE: Implement basic job deduplication (e.g., by URL).
    - FE: Design simple "Trigger Scraping" UI with input fields for keywords, location, source.
    - FE: Implement API call for scraping trigger.
* **US 2.2: View Scraped Jobs**
  + **Story**: As a profile manager, I want to view a list of all scraped jobs so that I can assess their relevance for my candidates.
  + **Tasks**:
    - BE: Implement GET /api/v1/jobs endpoint with basic filters (keywords, location).
    - FE: Design "Jobs List" UI with table to display job title, company, location, source.
    - FE: Implement data fetching for job list.

**Epic 3: Basic Contact Extraction (MVP)**

* **US 3.1: Extract Basic Contact Info**
  + **Story**: As a profile manager, I want the system to automatically extract basic contact (poster) information (name, email) directly from job posting HTML where available, so I don't have to manually search.
  + **Tasks**:
    - BE: Design Contact table schema (id, name, email, company, title, source, created\_at, updated\_at).
    - BE: Enhance scraper to look for common email/name patterns within job description HTML.
    - BE: Store extracted contacts linked to the Job.
    - FE: Enhance Job details view to show extracted contact (if any).
* **US 3.2: View Contacts**
  + **Story**: As a profile manager, I want to view a list of extracted contacts so that I can manage them.
  + **Tasks**:
    - BE: Implement GET /api/v1/contacts endpoint.
    - FE: Design "Contacts List" UI with table to display contact name, email, company.

**Epic 4: Basic Email Outreach (MVP)**

* **US 4.1: Send Single Email**
  + **Story**: As a profile manager, I want to manually compose and send a single email to a specific contact so that I can initiate communication.
  + **Tasks**:
    - BE: Integrate with transactional Email API (SendGrid/Mailgun).
    - BE: Create POST /api/v1/campaigns/send\_single\_email endpoint (input: contact\_id, subject, body).
    - BE: Implement email sending logic via API.
    - FE: Design "Send Email" modal/form on contact details page.
    - FE: Implement API call for sending email.

**Epic 5: Infrastructure & System Health (MVP)**

* **US 5.1: Core Infrastructure Setup**
  + **Story**: As a developer, I want to have the core development and deployment infrastructure ready so that I can build and run the application.
  + **Tasks**:
    - INFRA: Setup AWS/GCP project & IAM.
    - INFRA: Provision PostgreSQL DB instance.
    - INFRA: Configure Dockerfiles for Backend (FastAPI) and Frontend (React).
    - INFRA: Setup Docker Compose for local development.
    - INFRA: Basic Nginx/Traefik configuration for reverse proxy (if applicable for deployment strategy).
    - INFRA: Initial CI/CD pipeline setup (e.g., GitHub Actions for linting/tests).
* **US 5.2: System Health Check**
  + **Story**: As an administrator, I want a quick way to check if the core system services are running and accessible.
  + **Tasks**:
    - BE: Implement GET /api/v1/health endpoint (checks DB connection, basic external API reachability).
    - FE: Create a simple "System Status" page (Admin view only).

## Day-by-Day MVP Plan (30 Business Days)

**Phase 1: Foundation**

| Day | Date | Epic/Story/Task | Owner | Est. Hours | Notes |
| --- | --- | --- | --- | --- | --- |
| **Day 1** | 2025-07-07 | **INFRA: Project Setup & DB** | PL/BE | 8 | Project repo, FastAPI skeleton, React skeleton, initial Dockerfiles. PostgreSQL DB provision. |
| **Day 2** | 2025-07-08 | **INFRA: Docker Compose & Initial Dev Env** | BE/FE | 8 | Docker Compose for BE/FE, local DB connection, basic migrations. |
| **Day 3** | 2025-07-09 | **US 1.1: BE - User Schema & Hashing** | BE | 8 | Define User model, implement password hashing. |
| **Day 4** | 2025-07-10 | **US 1.1: BE - Register Endpoint & US 1.2: BE - Login Endpoint (No JWT)** | BE | 8 | POST /register, POST /login (simple username/password check for now). |
| **Day 5** | 2025-07-11 | **US 1.2: BE - JWT Implementation** | BE | 8 | Add JWT token generation and verification. Update POST /login to return JWT. |
| **Day 6** | 2025-07-14 | **US 1.1 & 1.2 & 1.3: FE - Auth Forms** | FE | 8 | Login/Register UI. Basic routing. Store/use token. |
| **Day 7** | 2025-07-15 | **US 1.3: BE - GET /me & Basic RBAC** | BE | 8 | GET /api/v1/users/me endpoint. RBAC decorator concept (e.g., role\_required). |

**Phase 2: Core Features**

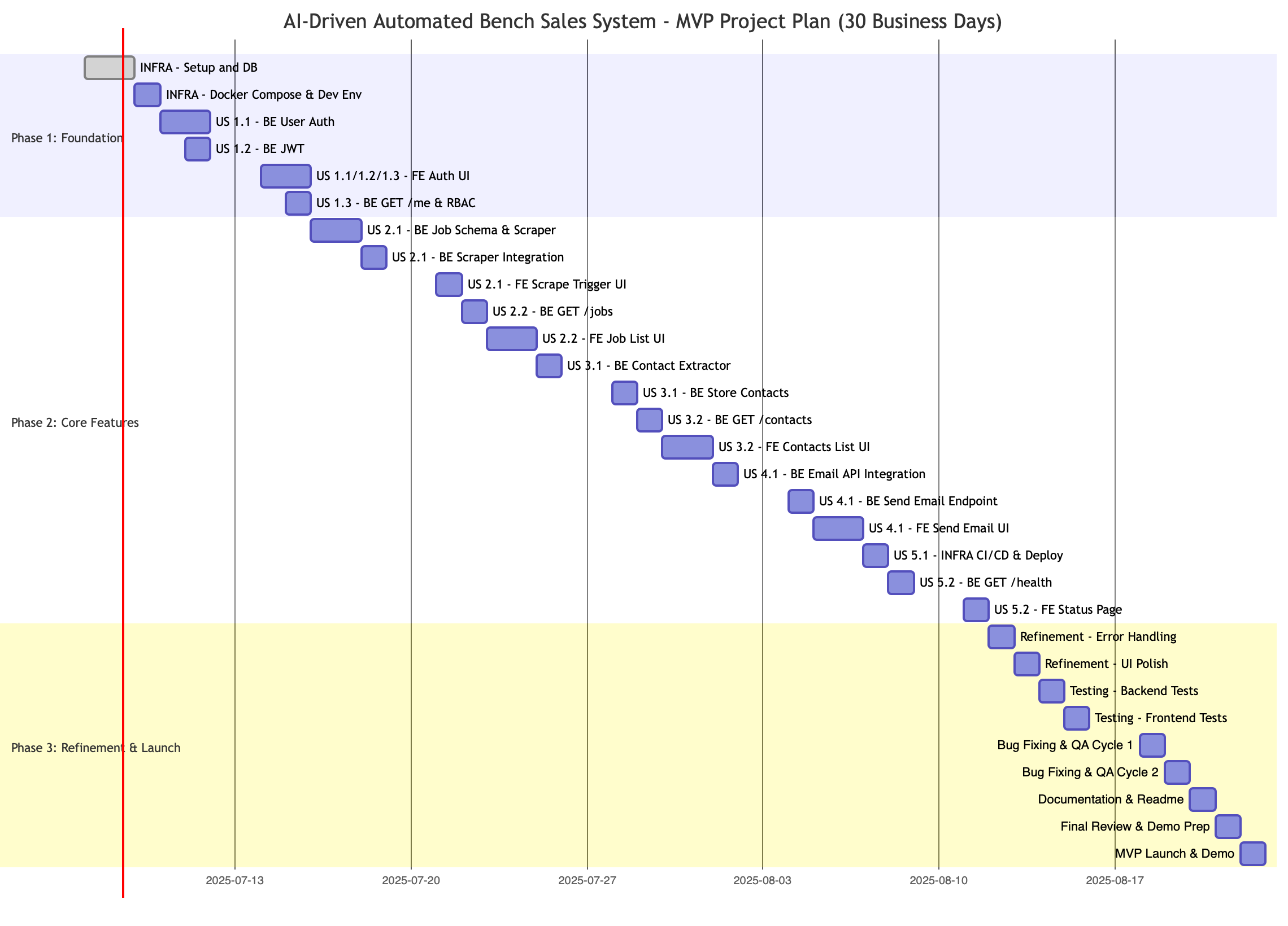
| Day | Date | Epic/Story/Task | Owner | Est. Hours | Notes |
| --- | --- | --- | --- | --- | --- |
| **Day 8** | 2025-07-16 | **US 2.1: BE - Job Schema & Indeed Scraper (Core Logic)** | BE | 8 | Define Job model. Start building scraper for Indeed (requests + BeautifulSoup). |
| **Day 9** | 2025-07-17 | **US 2.1: BE - Scraper Integration & Deduplication** | BE | 8 | Integrate scraper with FastAPI, implement POST /jobs/scrape. Simple URL dedupe. |
| **Day 10** | 2025-07-18 | **US 2.1: FE - Scrape Trigger UI** | FE | 8 | Build UI for manual scraping (keywords, location). |
| **Day 11** | 2025-07-21 | **US 2.2: BE - GET /jobs Endpoint** | BE | 8 | Implement GET /api/v1/jobs with basic filtering. |
| **Day 12** | 2025-07-22 | **US 2.2: FE - Job List Display** | FE | 8 | Build Jobs List UI component to display scraped jobs. |
| **Day 13** | 2025-07-23 | **US 3.1: BE - Contact Schema & Extraction** | BE | 8 | Define Contact model. Enhance scraper to extract contact email/name from job HTML. |
| **Day 14** | 2025-07-24 | **US 3.1: BE - Store Extracted Contacts** | BE | 8 | Store extracted contacts in DB, link to job. |
| **Day 15** | 2025-07-25 | **US 3.2: BE - GET /contacts Endpoint** | BE | 8 | Implement GET /api/v1/contacts. |
| **Day 16** | 2025-07-28 | **US 3.2: FE - Contacts List Display** | FE | 8 | Build Contacts List UI component. |
| **Day 17** | 2025-07-29 | **US 4.1: BE - Email API Integration** | BE | 8 | Integrate with SendGrid/Mailgun (API key setup, basic send functionality). |
| **Day 18** | 2025-07-30 | **US 4.1: BE - send\_single\_email Endpoint** | BE | 8 | Implement POST /api/v1/campaigns/send\_single\_email. |
| **Day 19** | 2025-07-31 | **US 4.1: FE - Send Email UI** | FE | 8 | Build simple modal/form to compose and send email to a contact. |
| **Day 20** | 2025-08-01 | **US 5.1: INFRA - Basic CI/CD & Deployment Prep** | PL/BE/FE | 8 | Set up GitHub Actions for linting/tests. Manual deployment guide for a staging env. |
| **Day 21** | 2025-08-04 | **US 5.2: BE - GET /health Endpoint & FE - Status Page** | BE/FE | 8 | Implement health check. Basic Admin status page. |

**Phase 3: Refinement & MVP Launch**

| Day | Date | Epic/Story/Task | Owner | Est. Hours | Notes |
| --- | --- | --- | --- | --- | --- |
| **Day 22** | 2025-08-05 | **Refinement: Error Handling & Validation** | BE | 8 | Implement consistent error responses for all endpoints. Pydantic validation. |
| **Day 23** | 2025-08-06 | **Refinement: UI Polish & UX Improvements** | FE | 8 | Ensure smooth navigation, basic styling. Loading states. |
| **Day 24** | 2025-08-07 | **Testing: Unit & Integration Tests (BE)** | BE | 8 | Write unit tests for models, endpoints. Integration tests for core flows. |
| **Day 25** | 2025-08-08 | **Testing: Component & Integration Tests (FE)** | FE | 8 | Write basic tests for React components and API integration. |
| **Day 26** | 2025-08-11 | **Bug Fixing & QA Cycle 1** | ALL | 8 | Identify and fix bugs reported from internal testing. |
| **Day 27** | 2025-08-12 | **Bug Fixing & QA Cycle 2** | ALL | 8 | Final bug fixes and overall system sanity checks. |
| **Day 28** | 2025-08-13 | **Documentation & Readme Update** | PL/BE/FE | 8 | Update API docs (FastAPI Swagger UI), READMEs for projects. |
| **Day 29** | 2025-08-14 | **Final Review & Preparation for Demo** | ALL | 8 | Review all features against MVP goals. Prepare demo script. |
| **Day 30** | 2025-08-15 | **MVP Launch & Demo** | ALL | 8 | Deploy to production-like environment. Conduct internal demo. |

## 

## Project Gantt



## 

## MVP Success Criteria

By the end of this 30-day MVP, we expect to meet the following criteria:

**Technical Metrics**:

* **Database**: All core models (User, Job, Contact) are defined, persisted, and accessible via API.
* **API Uptime**: 95%+ for core endpoints during testing.
* **Scraping**: Successfully scrapes 100+ job postings from Indeed daily with keyword filtering.
* **Email Sending**: Successful delivery of single emails via integrated API.

**Business Metrics**:

* **User Accounts**: 5+ test user accounts (Admin, Profile Manager, Job Seeker roles).
* **Job Discovery**: Demonstrable list of scraped jobs relevant to defined keywords.
* **Contact Extraction**: Demonstrable extraction of basic contact info (name, email) from at least 50% of scraped jobs where present.
* **Outreach**: Successful manual sending of single emails to test contacts.

**User Experience Metrics**:

* **Core Workflow Completion**: Ability for a Profile Manager to log in, trigger a scrape, view jobs, view contacts, and send a single email to a contact.
* **Basic UI Usability**: Forms are intuitive, lists are readable, and core actions are clear.
* **System Responsiveness**: Page loads generally under 3 seconds.