Full Stack Assignment

Build Core Modules for SARAL AI Recruitment Platform

Goal
Build two production-ready modules of the SARAL AI recruitment platform. You will receive a working demo link and product flow notes. You must implement these modules end-to-end with clean architecture, solid UI, and a stable backend.
Link: PROTOTYPE DESIGN - LIVE

Module 1: AI Candidate Search and Result Window

What to build
- Natural language search input with example placeholders
- Add Filter and Upload JD buttons (UI only)
- Run Search button
- Credit confirmation modal
- Suggestion chips

Search flow
Simulate multi-step AI processing with four stages Loader:
1 Fetch profiles
2 Semantic search and LLM match
3 Ranking and scoring
4 Preparing insights
Each stage must show pending, loading, and completed states. Minimum two seconds per stage. Handle errors.

Results grid
Three-column card layout on desktop.

Cards show:
Image placeholder
Name and availability dot
Title, company
Experience years, icons, and location
Match percent with color codes

Icons for like, dislike, and shortlist
Pagination and tabs for Matches, Shortlisted

Candidate detail modal
Opens on card click.

Includes:
Header with image, name, title, match percent
Strengths
Areas to Probe
AI Verdict banner
Career timeline
Work experience list
About section
Contact locked with Unlock Credits button
Education
Scrollable modal

Tech stack
Frontend: ReactJS with TypeScript. State manager of your choice. Responsive layouts. Skeleton loading. Error boundaries.
Backend: Node Express or FastAPI. REST APIs. PostgreSQL

Database: Candidates, search history, credits, and shortlisted.

Search endpoint must support pagination and credit deduction.
Mock at least 40 candidates.

Module 2: Email Sequence Builder with Analytics

Dashboard
Tabs for LinkedIn Campaigns and Email Sequences.
Cards showing delivery, open rate, reply rate, plus trends.
Campaign list table with status, steps, sent, replied, and actions.

Tech stack
Frontend: React with TypeScript. Rich text editor. Validation. Charts with Recharts.
Backend: Node Express or FastAPI.
Database: Campaigns, steps, analytics events, recipients.
Endpoints for create, update, delete, list, and analytics.
Template parser for variables.

Shared Requirements

Code quality
TypeScript. ESLint and Prettier. Component-driven structure. Custom hooks.
Strong error handling. Validation. Input sanitization.

Performance
Lazy loading. Memoization. API caching.
Bundle under 500KB.
Efficient DB queries.

Security
SQL injection protection
Rate limits.

Deployment
Docker. Env variables. Database migrations.
README with setup and API docs.
Live on free Apps

Evaluation Criteria

- Architecture and design quality
- Technical depth
- UI clarity and responsiveness
- Edge case handling
- Performance
- Tests and documentation

Deliverables

- GitHub repo with a clean structure and commits
- README with setup and API docs
- FULLY DEPLOYED LIVE Platform at free hosting.
- Short summary of approach and decisions