Case Study Brief

Objective

Build an AI-powered recruitment assistant system for evaluating blockchain developer candidates. The system should automatically analyze CVs and project submissions to assess candidate fit against job requirements, using standardized rubrics and semantic search capabilities.

Context/Background

The recruitment process for blockchain developers requires careful evaluation of both technical expertise (smart contract development, cryptography, distributed systems) and practical implementation skills. Manual review of CVs and technical projects is time-consuming and prone to inconsistency. This system aims to standardize and automate the initial screening phase while maintaining high-quality candidate assessment.

Core Logic & Data Flow

1. **Document Ingestion**: Upload and parse CVs (PDF format) and extract structured information
2. **Semantic Indexing**: Store job descriptions, evaluation rubrics, and case study briefs in a vector database (ChromaDB) for semantic retrieval
3. **CV Evaluation**: Compare candidate profiles against job requirements using AI-driven semantic matching
4. **Project Assessment**: Evaluate technical deliverables based on code quality, correctness, and documentation
5. **Scoring & Ranking**: Generate standardized scores and provide detailed feedback for each candidate

Deliverables

Backend Service

- RESTful API endpoints for CV upload and evaluation
- Integration with cloud storage (AWS S3) for document management
- Database layer (PostgreSQL + Prisma ORM) for persisting evaluation results
- Authentication and authorization mechanisms

Evaluation Pipeline

- AI-powered CV parsing and information extraction
- Semantic search using ChromaDB with Google Gemini embeddings
- Automated scoring based on predefined rubrics
- Support for both CV evaluation and project deliverable assessment

Standardized Evaluation Parameters

- CV Match Evaluation: Technical skills, experience level, achievements, cultural fit
- Project Evaluation: Code quality, correctness, error handling, documentation, creativity

Requirements

Functional Requirements

- Upload CVs in PDF format
- Extract candidate information (skills, experience, education, projects)
- Evaluate CVs against job description using semantic similarity
- Score candidates based on standardized rubrics
- Store and retrieve evaluation results
- Generate detailed evaluation reports

Non-Functional Requirements

- Response time < 30 seconds for CV evaluation
- Support concurrent evaluations
- Secure handling of candidate data
- Scalable architecture for batch processing
- Comprehensive error handling and logging

Technical Stack

Backend

- **Runtime**: Node.js with NestJS framework
- **Language**: TypeScript
- **Database**: PostgreSQL with Prisma ORM
- **Vector Database**: ChromaDB for semantic search
- **Cloud Storage**: AWS S3 for document storage
- **AI/LLM**: Google Gemini API for embeddings and content generation

Development Tools

- **Package Manager**: pnpm
- **API Testing**: Postman/REST Client
- **Environment Management**: dotenv
- **Code Quality**: ESLint, Prettier

Job Description - Blockchain Developer 2025

About the Job

We are seeking a talented Blockchain Developer to join our innovative team building next-generation decentralized applications. You will be responsible for designing, developing, and deploying smart contracts and blockchain-based solutions that solve real-world problems.

Key Responsibilities

- Design and develop smart contracts using Solidity for Ethereum and EVM-compatible chains
- Build and maintain decentralized applications (dApps) with Web3 integration
- Implement security best practices and conduct smart contract audits
- Optimize gas efficiency and transaction throughput
- Collaborate with frontend developers to integrate blockchain functionality
- Research and implement emerging blockchain technologies and protocols
- Write comprehensive technical documentation and test cases

About You

Required Skills & Experience

- 3+ years of software development experience
- 2+ years of hands-on experience with Solidity and smart contract development
- Strong understanding of Ethereum, EVM, and blockchain fundamentals
- Experience with Web3.js, ethers.js, or similar libraries
- Proficiency in testing frameworks (Hardhat, Truffle, Foundry)
- Knowledge of cryptography, consensus mechanisms, and distributed systems
- Experience with Git version control and CI/CD pipelines

Preferred Qualifications

- Experience with Layer 2 solutions (Polygon, Arbitrum, Optimism)
- Familiarity with DeFi protocols (AMMs, lending platforms, yield farming)
- Knowledge of NFT standards (ERC-721, ERC-1155)
- Experience with cross-chain bridges and interoperability
- Contributions to open-source blockchain projects
- Understanding of smart contract security vulnerabilities (reentrancy, overflow, etc.)
- Experience with backend development (Node.js, TypeScript)

Personal Attributes

- Strong problem-solving and analytical skills
- Attention to detail and security-conscious mindset
- Excellent communication and collaboration abilities
- Self-motivated and eager to learn new technologies
- Passion for decentralization and blockchain innovation

Benefits & Perks

- Competitive salary and equity/token compensation
- Flexible remote work arrangements
- Professional development budget for conferences and courses
- Health insurance coverage
- Annual team retreats and hackathons
- Access to cutting-edge blockchain infrastructure
- Collaborative and innovative work environment

Requirements

- Bachelor's degree in Computer Science, Engineering, or related field (or equivalent experience)
- Portfolio of deployed smart contracts or blockchain projects
- Strong understanding of software engineering principles
- Ability to work in a fast-paced, agile environment
- Legal authorization to work in the specified location

CV Match Evaluation Rubric

Parameter 1: Technical Skills Match (Weight: 40%)

Description: Evaluate the candidate's blockchain-specific technical skills against job requirements, including smart contract development, Web3 technologies, and relevant programming languages.

Scoring Guide: - **5 (Excellent)**: Expert-level proficiency in Solidity, smart contract development, and multiple blockchain platforms. Deep understanding of security best practices, gas optimization, and advanced concepts (Layer 2, DeFi protocols, cross-chain). Active contributions to blockchain open-source projects. - **4 (Good)**: Strong proficiency in Solidity and smart contract development. Experience with major blockchain platforms (Ethereum, Polygon). Solid understanding of Web3.js/ethers.js, testing frameworks, and security considerations. Some experience with DeFi or NFTs. - **3 (Satisfactory)**: Moderate proficiency in Solidity and basic smart contract development. Familiar with Ethereum and Web3 libraries. Understanding of blockchain fundamentals and

basic security practices. Limited production experience. - **2 (Below Average)**: Basic knowledge of blockchain concepts and minimal Solidity experience. Limited hands-on smart contract development. Weak understanding of security and optimization practices. - **1 (Poor)**: No relevant blockchain development experience. Lacks fundamental knowledge of smart contracts, Web3, or blockchain architecture.

Parameter 2: Experience Level (Weight: 25%)

Description: Assess the candidate's years of relevant experience in blockchain development and overall software engineering background.

Scoring Guide: - **5 (Excellent)**: 4+ years of blockchain development experience with 6+ years total software development. Led multiple successful blockchain projects from conception to production. Experience architecting complex decentralized systems. - **4 (Good)**: 2-3 years of blockchain development with 4-6 years total software development. Contributed to several production blockchain projects. Experience with full development lifecycle. - **3 (Satisfactory)**: 1-2 years of blockchain development with 3-4 years total software development. Worked on a few blockchain projects, possibly not all in production. Growing expertise. - **2 (Below Average)**: Less than 1 year blockchain development or 2-3 years total software development. Limited project experience. Primarily junior-level work. - **1 (Poor)**: No blockchain development experience or less than 2 years total software development. No substantial project involvement.

Parameter 3: Relevant Achievements (Weight: 20%)

Description: Evaluate the candidate's track record of achievements in blockchain development, including deployed contracts, contributions to notable projects, hackathon wins, publications, or certifications.

Scoring Guide: - **5 (Excellent)**: Multiple deployed smart contracts with significant TVL (Total Value Locked) or user adoption. Winner of major blockchain hackathons. Published security audits or research papers. Recognized contributor to major blockchain protocols. - **4 (Good)**: Several deployed smart contracts in production. Participation in blockchain hackathons with notable results. Contributions to well-known open-source blockchain projects. Relevant certifications (e.g., Certified Blockchain Developer). - **3 (Satisfactory)**: At least one deployed smart contract or dApp. Some involvement in blockchain community (conferences, meetups). Basic portfolio of blockchain projects. Some open-source contributions. - **2 (Below Average)**: Limited demonstrated achievements. Mostly personal or learning projects. Minimal community involvement or public portfolio. - **1 (Poor)**: No demonstrable blockchain achievements. No portfolio of projects or community involvement.

Parameter 4: Cultural/Collaboration Fit (Weight: 15%)

Description: Assess the candidate's alignment with team values, communication skills, collaborative mindset, and passion for decentralization and blockchain technology.

Scoring Guide: - **5 (Excellent)**: Exceptional communication skills with clear passion for blockchain and decentralization. Strong evidence of collaborative work (team projects, open-source collaboration). Demonstrates leadership and mentoring abilities. Aligns perfectly with company values and mission. - **4 (Good)**: Good communication skills and genuine interest in blockchain technology. Clear examples of effective teamwork. Positive attitude toward collaboration. Good cultural alignment. - **3 (Satisfactory)**: Adequate communication skills. Some evidence of teamwork. Interest in blockchain but not deeply passionate. Reasonable cultural fit. - **2 (Below Average)**: Weak communication skills or limited evidence of collaboration. Appears primarily interested in compensation rather than technology. Questionable cultural fit. - **1 (Poor)**: Poor communication skills. No evidence of teamwork or collaboration. No apparent interest in blockchain mission. Poor cultural fit.

Project Deliverable Evaluation Rubric

Parameter 1: Correctness (Prompt & Chaining) (Weight: 30%)

Description: Evaluate whether the solution correctly implements the required functionality, properly uses AI prompts and chain-of-thought reasoning, and produces accurate results.

Scoring Guide: - **5 (Excellent)**: All requirements fully implemented and working flawlessly. AI prompts are expertly crafted with clear reasoning chains. Edge cases handled properly. Output is consistently accurate and reliable. System logic is sound and well-thought-out. - **4 (Good)**: Most requirements implemented correctly with minor issues. AI prompts are well-structured with good reasoning. Handles common cases effectively. Output is generally accurate with occasional minor discrepancies. - **3 (Satisfactory)**: Core requirements implemented but with some issues. AI prompts work but could be improved. Some edge cases not handled. Output is mostly correct but has noticeable gaps. - **2 (Below Average)**: Partial implementation with significant functionality missing. AI prompts are poorly structured or ineffective. Many edge cases fail. Output is unreliable or frequently incorrect. - **1 (Poor)**: Minimal functionality implemented. AI prompts are ineffective or missing. Core requirements not met. Output is mostly incorrect or system doesn't work.

Parameter 2: Code Quality & Structure (Weight: 25%)

Description: Assess code organization, readability, adherence to best practices, proper use of TypeScript/NestJS patterns, and overall software architecture.

Scoring Guide: - **5 (Excellent)**: Exceptional code quality with clear architecture. Follows all best practices and design patterns. Well-organized modules and services. Excellent TypeScript typing. Clean, readable, and maintainable code. Professional-grade structure. - **4 (Good)**: Good code quality with solid structure. Follows most best practices. Proper use of NestJS patterns. Good TypeScript usage. Code is readable and well-organized with minor areas for improvement. - **3 (Satisfactory)**: Acceptable code quality. Basic structure in place but inconsistent. Some best practices followed. TypeScript usage is adequate but not optimal. Code is understandable but could be cleaner. - **2 (Below Average)**: Poor code quality with weak structure. Many best practices ignored. Inconsistent patterns. Weak TypeScript usage or type safety issues. Code is hard to follow or maintain. - **1 (Poor)**: Very poor code quality. No clear structure. Best practices not followed. Minimal or incorrect TypeScript usage. Code is messy, unorganized, and difficult to understand.

Parameter 3: Resilience & Error Handling (Weight: 20%)

Description: Evaluate how well the system handles errors, validates inputs, manages failures gracefully, and provides meaningful error messages.

Scoring Guide: - **5 (Excellent)**: Comprehensive error handling throughout the application. All inputs validated properly. Graceful failure recovery with meaningful error messages. Proper logging and monitoring. Handles all edge cases and unexpected scenarios elegantly. - **4 (Good)**: Good error handling in most areas. Input validation in place. Errors are caught and handled appropriately. Error messages are helpful. Most edge cases covered. - **3 (Satisfactory)**: Basic error handling implemented. Some input validation. Errors are caught but handling could be improved. Error messages are present but not always clear. Some edge cases handled. - **2 (Below Average)**: Minimal error handling. Little to no input validation. Errors often crash the application or produce unclear messages. Many edge cases not handled. - **1 (Poor)**: No error handling. No input validation. Application crashes frequently. No meaningful error messages. Edge cases cause failures.

Parameter 4: Documentation & Explanation (Weight: 15%)

Description: Assess the quality of code comments, README documentation, API documentation, and overall explanation of design decisions and implementation approach.

Scoring Guide: - **5 (Excellent)**: Comprehensive documentation including detailed README, API docs, architecture diagrams, and design decisions. Code

is well-commented where needed. Setup instructions are clear and complete. Explains trade-offs and reasoning. - **4 (Good)**: Good documentation with clear README and setup instructions. API endpoints documented. Key design decisions explained. Code comments where helpful. Minor areas could use more detail. - **3 (Satisfactory)**: Basic documentation present. README exists with setup instructions. Some code comments. Minimal explanation of design decisions. Enough to understand and run the project. - **2 (Below Average)**: Minimal documentation. README is incomplete or unclear. Few code comments. Design decisions not explained. Difficult to understand or set up the project. - **1 (Poor)**: No meaningful documentation. No README or it's severely lacking. No code comments. No explanation of implementation. Very difficult to understand the project.

Parameter 5: Creativity / Bonus (Weight: 10%)

Description: Reward innovative approaches, additional features beyond requirements, clever optimizations, or exceptional implementation quality.

Scoring Guide: - **5 (Excellent)**: Highly innovative solution with creative approaches. Multiple value-added features beyond requirements. Exceptional optimizations or unique insights. Demonstrates deep understanding and original thinking. - **4 (Good)**: Some creative elements or additional features. Good optimizations. Shows initiative in going beyond basic requirements. Thoughtful implementation choices. - **3 (Satisfactory)**: Meets requirements without significant creativity. Standard implementation approach. No major additional features but solid execution. - **2 (Below Average)**: Very basic implementation with no creative elements. Barely meets minimum requirements. No additional effort or optimization. - **1 (Poor)**: Incomplete basic implementation. No creativity or additional effort. Below minimum expectations.