PURPLE ELEPHANT VENTURES

Software Engineer

Technical Assessment

Nairobi, Kenya | Remote Available

1. Overview

This document outlines the second round interview exercise for software engineering candidates at Purple Elephant Ventures. The core requirement is to build a web application and a backend application that integrates with the FBI Wanted API (https://www.fbi.gov/wanted/api).

The exercise is intended to evaluate the candidate's technical and problem-solving skills in both frontend and backend development.

1.1. Objective

Build a full-stack application (frontend + backend) that consumes and manages information from the FBI Wanted API.

1.2. Tech Stack Choices

1. Frontend: React.js or Vue.js
2. Backend: Java, Kotlin, Go, PHP, or Node.js

1.3. Expected Deliverable

A functioning web application with key features implemented, demonstrating your ability to integrate external data, handle it on the server, and present it effectively on the frontend.

2. Key Areas of Focus

2.1. Integration with External API

1. Understanding of RESTful API concepts.
2. Ability to fetch data from the FBI Wanted API endpoints.
3. How errors and edge cases are handled (e.g., rate-limiting, malformed requests, interruptions).

2.2. Backend Development

1. Creating RESTful endpoints or GraphQL resolvers (whichever is preferred) to securely and efficiently handle data from the FBI Wanted API.
2. Data transformation and caching strategies (if applicable).
3. Proper error handling and logging on the server side.

2.3. Frontend Development

1. Building a responsive interface using React.js or Vue.js.
2. Proper component structure, state management, and data flow (e.g., Redux, Vuex, context APIs, or composition APIs).
3. Good UI/UX principles, including consistent styling and a clear navigation scheme.

2.4. Testing

1. Unit testing and/or integration testing of critical functionality both on the frontend and backend.
2. Understanding of testing frameworks and best practices for the chosen languages.

2.5. Documentation & Code Quality

1. Clear and concise documentation for setup and usage.
2. Code readability, maintainability, and adherence to style guidelines.

3. Detailed Requirements

3.1. Must-Have Features

3.1.1. Fetching Data

1. At minimum, fetch a list of wanted persons from the FBI Wanted API and display relevant information (e.g., name, description, image).
2. Implement pagination or infinite scrolling when displaying the list.

3.1.2. Search & Filter

1. Implement a search feature that allows users to search for wanted persons based on parameters supported by the API (e.g., name, keywords, subjects, etc.).
2. Allow users to filter the results (e.g., by hair color, eye color, race, etc.) if such data is provided by the API.

3.1.3. Detailed View

1. Enable users to select an item from the list to view more details (e.g., personal details, descriptions, alerts, caution, field offices).
2. Include an “additional information” panel that might include external links, if provided in the API response.

3.1.4. Backend Integration

1. Expose a simple backend endpoint (**e.g., /api/wanted**) that the frontend consumes. This backend endpoint should proxy or transform data from the FBI Wanted API.
2. Implement at least one form of caching or data storage mechanism (optional to store the results in memory or a small database) to illustrate how your app handles repeated requests.

3.1.5. Error Handling

1. Display meaningful error messages on the frontend (e.g., “Unable to fetch data,” “Search returned no results,” etc.).
2. Handle service failures gracefully.

3.2. Nice-to-Have Features

3.2.1. Authentication

A simple login flow (session or token-based) to demonstrate secure routes or personalized data.

3.2.2. Advanced Filtering

More granular filter options if the API supports additional filtering parameters.

3.2.3. Performance Optimizations

Caching strategy or local data store for offline capability or faster UI.

3.2.4. Automated Testing

Frontend: Unit testing with Jest, Mocha, or equivalent.

Backend: Using JUnit/TestNG (Java/Kotlin), Go testing libraries, PHPUnit, Jest (Node.js), or another relevant testing framework.

3.2.5. Containerization

Use Docker to containerize the application for easier deployment.

4. Interview Breakdown

4.1. Initial Discussion (15-20 mins)

1. The interviewer will discuss your approach, overall architectural decisions, and design choices.
2. Be ready to explain your reasoning for technology choice, data flow, and structure.

4.2. Implementation Walkthrough (20-30 mins)

1. You will walk through your code on both the frontend and backend.
2. Highlight API integration, data parsing, caching strategy, and UI design.
3. Discuss any challenges encountered and how you dealt with them.

4.3. Live Testing / Feature Demonstration (10-15 mins)

1. Show the key features mentioned above.
2. Demonstrate how the application responds to errors and edge cases (e.g., invalid search queries).

4.4. Q&A on Enhancements (10-15 mins)

1. The interviewer might ask how you would improve or extend the application (scalability, more advanced functionalities, etc.).
2. This portion tests your ability to plan iterative improvements.

5. Evaluation Criteria

5.1. Technical Competence

1. Correct usage of the FBI Wanted API.
2. Well-structured code with clear separation of concerns (frontend vs. backend).
3. Efficient data handling, including pagination and filtering requests.

5.2. Code Quality

1. Readable, maintainable code with a logical directory structure.
2. Compliance with language-specific best practices and style guides.

5.3. Problem-Solving & Decision-Making

1. Sound architectural decisions and justifications.
2. Effective handling of errors, edge cases, and potential bottlenecks.

5.4. UI/UX

1. Clean interface, intuitive navigation, and responsiveness.
2. User-friendly search, filter, and details features.

5.5. Communication

1. Ability to explain your approach, trade-offs, and results clearly.
2. Clarity in discussing challenges, shortcomings, or future improvements.

6. Expected Deliverables

6.1. Source Code Repository

Hosting on GitHub, GitLab, or similar (include any build instructions or scripts).

6.2. Documentation

README file containing:

1. How to run the frontend application.
2. How to run the backend application.
3. Configuration details, environment variables, or placeholders.
4. Any dependencies and installation steps.

6.3. Instructions on Testing (If Implemented)

Steps to run automated tests on both frontend and backend.

6.4. Brief Deployment Guide (If you containerize)

Steps to run Docker containers (if you use Docker or any other containerization method).

7. Final Notes

1. **Don’t Overcomplicate**: Focus on demonstrating that you can build a solid and maintainable application in the time given. You do not need to implement every possible feature if time is limited.
2. **Focus on Usability**: Make sure the application is easy to navigate and all main features are discoverable.
3. **Demonstrate Best Practices**: Show you understand good coding and architectural principles for a production-like environment.

We look forward to seeing your approach and discussing solutions during the interview. Good luck!