Job Interview Prep Assistant - Development Plan

Project Overview

Build an AI-powered conversational agent that helps CS students prepare for technical interviews through mock interviews, coding challenges, and personalized feedback.

Day 1-2: Setup & Core Architecture

Google Cloud Setup

- Create Google Cloud Project
- Enable required APIs:
  - Vertex AI API
  - Dialogflow CX API
  - Cloud Functions API
  - Cloud Storage API
- Set up authentication and service accounts

Tech Stack Decision

Backend:

- Google Cloud Functions (serverless)
- Vertex AI for conversational AI
- Cloud Firestore for data storage
- Python/Node.js for backend logic

Frontend:

- React web app or Flutter mobile app
- Simple chat interface
- Progress tracking dashboard

Database Schema Design

```
Users:
- user_id, email, name, skill_level
- interview_history, progress_metrics

Interview_Sessions:
- session_id, user_id, type, difficulty
- questions_asked, responses, scores
- feedback, duration, timestamp

Question_Bank:
- question_id, category, difficulty
- question_text, expected_approach
- sample_solutions, evaluation_criteria
```

Day 3-4: Core Features Implementation

1. Question Bank Creation

Technical Questions Categories:

- Data Structures (Arrays, Linked Lists, Trees, Graphs)
- Algorithms (Sorting, Searching, Dynamic Programming)
- System Design (for senior students)
- Behavioral Questions
- Company-specific questions (Google, Microsoft, Amazon)

Implementation:

- Create JSON files with 50-100 questions per category
- Include difficulty levels (Easy, Medium, Hard)
- Add expected solution approaches and time complexity
- Store in Cloud Storage or embed in code

2. Conversational Flow Design

Using Dialogflow CX:

- Welcome intent and user profiling
- Interview type selection (coding, behavioral, system design)
- Question delivery and response collection

- Feedback and scoring system
- Progress tracking and recommendations

Sample Conversation Flow:

```
Agent: "Hi! I'm your interview prep assistant. What type of interview would you like to
practice?"
User: "Technical coding interview"
Agent: "Great! What's your experience level and preferred language?"
User: "Intermediate, Python"
Agent: "Perfect! Let's start with a medium-level array problem..."
```

3. Code Evaluation System

For Coding Questions:

- Use Cloud Functions to execute and test code
- Implement basic test cases for each problem
- Check for:
  - Correctness
  - Time complexity
  - Code style and readability
  - Edge case handling

Day 5-6: Advanced Features

1. AI-Powered Feedback System

Using Vertex AI:

- Analyze user responses for behavioral questions
- Provide personalized improvement suggestions
- Track progress over multiple sessions
- Generate detailed performance reports

2. Adaptive Difficulty

- Start with user's comfort level
- Adjust question difficulty based on performance
- Mix easy wins with challenging problems

- Track improvement over time

3. Mock Interview Simulator

Real-time Interview Experience:

- Timer for each question (30-45 minutes for coding)
- Pressure simulation with countdown
- Multiple rounds (screening, technical, behavioral)
- Final score and detailed feedback

Day 7: Polish & Demo Preparation

1. User Interface Enhancement

- Clean, professional chat interface
- Code editor integration (Monaco Editor)
- Progress visualization charts
- Mobile-responsive design

2. Testing & Bug fixes

- Test all conversation flows
- Verify code execution security
- Performance optimization
- Error handling improvement

3. Demo Preparation

Key Demo Points:

- Live mock interview session
- Show adaptive difficulty adjustment
- Demonstrate code evaluation
- Display progress tracking
- Highlight Google Cloud integration

Technical Implementation Details

Cloud Functions Structure

python

```python
# main.py - Cloud Function entry point
import vertexai
from google.cloud import firestore

def interview_agent(request):
    # Parse user input
    # Query appropriate question from bank
    # Generate conversational response
    # Store session data
    # Return structured response
```

Vertex AI Integration

python

```python
# Using Vertex AI Conversation API
from vertexai.preview.generative_models import GenerativeModel

model = GenerativeModel("gemini-pro")
response = model.generate_content(
    f"Evaluate this coding solution: {user_code}. "
    f"Problem: {question}. Provide feedback on correctness, "
    f"efficiency, and suggest improvements."
)
```

Security Considerations

- Sandbox code execution environment
- Input validation and sanitization
- Rate limiting for API calls
- User authentication and session management

Evaluation Criteria & Scoring

Coding Questions (70% weight)

- Correctness: 40%
- Time Complexity: 20%
- Space Complexity: 10%
- Code Quality: 20%

- Edge Cases: 10%

Behavioral Questions (30% weight)

- Structure (STAR method): 25%
- Relevance: 25%
- Communication: 25%
- Specific examples: 25%

Bonus Features If Time Permits

- Integration with GitHub for code portfolio review
- Company-specific interview tracks
- Peer comparison and leaderboards
- Interview scheduling with calendar integration
- Resume analysis and suggestions

Demo Script Outline

1. **Introduction** (2 minutes)
    - Problem statement and solution overview
    - Google Cloud services used
2. **Live Demo** (5 minutes)
    - User onboarding flow
    - Technical interview simulation
    - Real-time code evaluation
    - Feedback generation
3. **Technical Deep Dive** (2 minutes)
    - Architecture overview
    - Google Cloud integration highlights
    - Scalability considerations
4. **Results & Impact** (1 minute)
    - Performance metrics
    - User feedback simulation
    - Future enhancement plans

Success Metrics

- Interview completion rate
- User satisfaction scores
- Improvement in practice scores over time
- Code execution accuracy
- Response time for feedback generation

Resources Needed

- Google Cloud free tier credits
- Sample interview questions dataset
- Code execution environment setup
- UI/UX design tools (Figma for mockups)
- Testing devices/browsers