15-Day Assessment Plan – AIML Developer

Objective

To assess the candidate's ability to:

- Build AI/ML pipelines for data processing and classification.
- Integrate pre-trained/free ML/DL models into APIs.
- Handle real-world data scenarios (resumes, images).
- Deliver usable APIs for integration with web/mobile apps.
- Demonstrate structured coding, documentation, and problem-solving.

Project Theme: Smart Resume Classifier + Beard Style AI API

Mini Web/Service Modules

1. **Automatic Resume Bifurcation**
    - HR uploads job post + multiple resumes (PDF/DOC).
    - System parses resumes and classifies candidates by match score (experience, skills, tech stack, expected salary).
2. **AI Beard Style Generator API**
    - User uploads face image.

    - System applies multiple beard style filters (from free/pre-trained models).
    - Exposes REST API for mobile app developers to consume.

Detailed 15-Day Plan

Phase 1 – Setup & Fundamentals (Day 1–3)

- **Project Setup** → Setup Python (FastAPI/Flask) backend + MongoDB/SQLite for metadata. Push initial repo to GitHub.
- **Resume Parsing R&D** → Research libraries (PyMuPDF, Spacy, Transformers, resume-parser). Document findings.
- **Data Schema** → Design schema for job posts, resumes, and bifurcation results.
- **Face Detection & Style R&D** → Research free/pre-trained beard/face models (e.g. Mediapipe, OpenCV, StyleGAN variants). Document feasibility.

Phase 2 – Resume Bifurcation (Day 4–7)

- **Resume Extractor** → Implement parser to extract skills, experience, education, salary expectations from resumes.
- **Job Post Parser** → Define keywords/requirements (tech stack, min exp, salary range). Store in DB.
- **Matching Algorithm** → Implement scoring system (e.g. cosine similarity / BERT embeddings) to compare resumes vs. job post.
- **API Development** → Create API endpoints for HR → upload job post, upload resumes, get bifurcation results.
- **Frontend (Minimal)** → Simple UI (React/HTML) to upload job + resumes and show bifurcation results.

Phase 3 – Beard Style AI API (Day 8–12)

- **Model Setup** → Integrate free/pre-trained model for beard style overlay.
- **Image Processing** → User uploads face → system detects face + applies beard filter.
- **Multiple Styles** → Offer multiple categories (light stubble, medium, full beard, etc.).
- **API Creation** → Build REST API: `/apply-beard?style=stubble`. Return processed image URL/base64.
- **Mobile Integration Demo** → Provide sample Postman collection or mobile integration doc.

Phase 4 – Testing & Documentation (Day 13–15)

- **Testing** → Test resume bifurcation accuracy with 10+ resumes. Test beard API with sample faces.
- **Debugging** → Fix common parsing/image errors.
- **Documentation** → README with setup steps, API docs, screenshots, and architecture diagram.
- **Short Report** → 1-page summary: Resume parsing approach + Beard API approach.

Deliverables

- GitHub Repository (Python backend + any frontend demo).
- Two Working APIs:
    - **Resume Bifurcation API**
    - **Beard Style Generator API**
- README.md including:

    - Setup steps
    - API routes + sample input/output
    - Architecture diagram
- **Short Report (1 page)** summarizing:
    - Resume parsing approach
    - Beard API integration

Tech Stack

Area	Tools / Frameworks
Backend	Python (FastAPI/Flask)
Database	MongoDB/SQLite
Resume Parsing	PyMuPDF, Spacy, HuggingFace Transformers
Matching/Scoring	NLP similarity (BERT, TF-IDF, cosine similarity)
Image Processing	OpenCV, Mediapipe, Pre-trained beard models
Deployment (Optional)	Render / Vercel / Colab
Docs	Markdown (README), PPT (optional)

Evaluation Criteria

Category	Weightage	What to Check
Functionality (Resume + Beard API)	30%	APIs working end-to-end
Code Structure & Logic	20%	Modular, clean, documented
ML/AI Model Integration	20%	Correctly applied, usable results
Problem Solving & Research	15%	Documented approaches, alternatives

API Usability & UI	10%	Clear, easy to test
Documentation & Demo	5%	Completeness of README + report

Optional Bonus Tasks

- **Resume Ranking** → Rank candidates with percentage match + recommendations.
- **Cloud Upload** → Store processed resumes/images in Cloudinary/AWS S3.
- **Sentiment Analysis** → Analyze tone of candidate resume summary.
- **Multiple AI Filters** → Add hairstyles or mustaches in addition to beard styles.

Note for Candidate: You are expected to push your code regularly to GitHub, maintain a clean commit history, and document all research and approaches tried.