**Solutions Architect / Development Officer**

***Practical Evaluation/ Test***

**Test Overview;**

This is a scenario-based practical evaluation to assess your competence and skills for the position of Solutions Architect / Development Officer.

The test duration is 8 hours (from 9am to 5pm). Response should be submitted via responding the same email. Submissions received after 5pm will not be considered.

**Background Information;**

Pakistan Polio Program conducts vaccination campaigns of children under 5 years. Union Council is the lowest geographical operational unit, and the frontline vaccinators (Polio Teams) perform household level vaccination activities by giving polio drops to the children. This activity comprises of 3 days (Intra-Campaign Days Coverage) of vaccination activities by visiting all planned houses of a team, and then additional 2 days activity (Catchup Days Coverage) to cover the children remained missed during initial 3 days of the campaign by revisiting the houses. The team carries a hard copy format (tally sheet) to record the number of children they vaccinated during a particular day and the number of children remained unvaccinated (missed children) in two categories (i.e. Children Not Available at the time of team’s visit “NA Children” and the Children whose parents has refused for vaccinating “Refusals Children”).

All tally sheets data is compiled in a Union Council (UC) level daily compilation sheet by aggregating all vaccination coverage information.

**Task Description:**

You are required to develop a centralized information system with a web admin panel and create a campaign activity of “SNID January 2025” by defining its geographical scope for all UCs of 7 districts (i.e. Abbotabad, Bannu, Dukki, Gwadur, Jehlum, Kasur, and Sukkur). Enter data of about 10 to 15 UCs falling under various districts which captures UC level aggregated coverage values from daily compilation sheets. For the intra-campaign days coverage reporting user web application form, and for the catchup days coverage reporting use the mobile application form. Finally, you are supposed to develop a dashboard with few useful information insights to see coverage results and missed children status at various geographic levels.

Perform your task as per the specified requirements on following pages and submit your work according to the guidelines by responding at the same email within given timeframe.

**Q1: Building a Vaccination Information Management System following the requirements.**

**Scenario Overview:**

You are tasked with developing a comprehensive Vaccination Information Management System for both a Web Application and a Android base Mobile Application. The system will collect and store Polio campaign data, including details about the vaccination coverage, geographic scope, target population (<5year children) etc. Data must be accessible through APIs for cross-platform integration, supporting both web and mobile platforms. The collected data will be stored in a PostgreSQL database.

**Technical Requirements**

The system should implement a robust authentication process for both the mobile and web applications, ensuring that only authenticated users can insert data. Admin users, upon successful authentication, should be granted the ability to create **new campaigns** **(Web App Admin Reqs [1])**, including specifying the year, month, campaign type, **geographic scope** **(Geodata [2])**, and other campaign-specific information through the web interface. Non-admin/ field users (UC level) on both the web and mobile applications should have restricted access based on their role. Web application users, once authenticated, will be allowed to input **intra-campaign data** **(Web App User Reqs [3])**, such as vaccination coverage and other campaign metrics, while mobile application users will be permitted to submit **catchup coverage data** **(Mobile App User Reqs [4])**. All data must be fed into the backend through APIs to ensure seamless synchronization between the mobile and web platforms, and all data entries should be securely stored in the database after proper validation/ authorization checks are performed and must be securely processed through a well-designed API. The API should go beyond basic CRUD operations to include data validation, authorization, and error handling. It must support role-based access control (RBAC) and ensure seamless, real-time data synchronization across platforms while maintaining data integrity and performance.

**Technology Stack followed:**

|  |  |
| --- | --- |
| **Backend Implementation** | **Flask / Django / Java / Javascript / AJAX** |
| **Front End Implementation** | **HMTL/ CSS / Jquery / Bootstrap / XML** |
| **API Implementation** | **Flask / Django / RestAPI** |
| **Database Requirements** | **PostgreSQL (Preferred), MySQL** |
| **CI / CD Requirements** | **GitHub Actions, Docker** |
| **Testing Requirements** | **Postman or Pytest** |
| **Software Requirements** | **Pycharm, PostMan, Android Studio** |

\* Ref: 1, 2, 3, 4 (Follow the attached excel sheet)

**Supporting Specifications:**

**Part 1: Front-End Development (Web Interface)**

1. **HTML & CSS:**

Create a form for the **Campaign Creation** feature. The form should allow the admin to input the following details as mentioned in Excel “(**Web App Admin Reqs [1])**”.

* + Use Bootstrap for responsiveness.

1. **JavaScript & jQuery** Implement validation for the **Campaign Creation** form. For example:
   * Ensure that the “Start Date” is earlier than the “End Date.”
   * Define and validate mandatory fields are filled in before submission.
   * Design the web application giving enterprise and professional look.

**Part 2: Backend Development (API Integration & Database)**

**Flask/Django API Endpoints** You are required to build an API that will allow the web admin, users (web/ mobile) to submit the vaccination campaign data to the backend. This API should:

* + Accept POST requests with the form data (e.g. Campaign Name, Geographic Scope, etc.).
  + Insert the data into a PostgreSQL database.
  + Return a response indicating success or failure.

**Database Design (PostgreSQL)** Design a PostgreSQL database schema for storing campaign data. You need to create tables that reflect the following:

* + Create tables considering the database normalization best practices.
  + Follow the excel sheet for creating multiple database tables and variables.
  + Use foreign keys to link tables where appropriate.

**Part 3: Android Mobile App Development (XML & Java)**

**Mobile App (XML & Java)** The mobile app will collect vaccination campaign catchup data from field workers. Implement an XML-based form to collect data and then send it to the server via an API.

**Part 4: Integration & API Testing**

1. **Cross-Platform Data Synchronization** Ensure that both the web and mobile platforms can feed data into the backend seamlessly. Both platforms should communicate with the same API, and the data must be stored consistently in the database.

**Part 5: CI/CD and Version Control**

1. **CI/CD Pipeline (GitHub Actions, Docker, etc.)** You need to set up a CI/CD pipeline that automates deployment for both the web and mobile platforms. The pipeline should trigger on new commits, run tests, and deploy to a staging or production environment.

Dataset Requirements

**[\*]: Please review ‘Solution Architect - Test Data Sheets - 11-1-25 Final.xlsx’ File.**

**Q2: Developing a Data Visualization Dashboard for Vaccination Insights**

**Scenario Overview:**

The goal is to develop an interactive and visually insightful vaccination dashboard for monitoring and visualizing key metrics related to polio vaccination campaign coverages status across various regions. The dashboard will display crucial vaccination data such as vaccination coverage, target achievement on real-time data from various districts, tehsils, and union councils (UC). This dashboard should integrate continuous deployment (CI/CD) workflows to ensure seamless updates and deployments. The system will leverage GitHub for version control and CI/CD pipelines to automate the testing and deployment process.

**Technical Requirements:**

1. **Dashboard Design**:
   * The dashboard should include following visualizations but not limited to it, and you are open to include any further visualization idea:
     + Overall Vaccination Coverage out of target (%age shown through charts) and coverage breakdown by geographical locations
     + Coverage status of missed (unvaccinated) children covered during same day and catchup day revisits (%age of covered vs recorded missed children “NA+Refusals”).
     + Still Missed Children (# and % children left after revisits)

**Supporting Specifications:**

1. **Technology Stack:**

* **Backend**: Flask/Django for APIs & Backend.
* **Frontend**: HTML, CSS, JavaScript, AJAX, Jinja For Front End
* **CI/CD**: GitHub, Jenkins/GitHub Actions.
* **Data Storage**: PostgreSQL/MySQL/SQLite.

1. **Backend Services:**

* RESTful APIs to process vaccination data.
* Data aggregation and caching with Redis.

1. **Responsive Design:**

* Mobile-friendly with Bootstrap or media queries.

1. **Visualizations:**

* Use D3.js, Chart.js, or Plotly for interactive charts.
* Filters for exploring data.

1. **Deployment & CI/CD:**

* GitHub version control.
* Automated testing and deployment with CI/CD.

**DevOps Requirements**

* Deploy Project on any Linux distribution on your local machine (VMware Workstation, Virtual Box etc.)
* Optimize the solution for 1000 number of concurrent users to access the system
* Implement security best practices while deployment

**Submission Requirements:**

1. **GitHub Repository (Share via GitHub Repository link):**
   * Initialize a GitHub repository for version control.
   * Include all web, dashboard & mobile application code, files, APIs, DB backup, Queries, everything related to work must be commit in your GitHub Repository.
   * Commit all your changes frequently and follow Git best practices:
   * Use meaningful commit messages
   * Ensure proper branching (e.g., main for production-ready code, dev for ongoing work)
   * Regularly push changes to GitHub
   * Consider using .gitignore to exclude unnecessary files (e.g., node\_modules, .env, etc.)
2. **Demo Video (share via google drive link):**
   * Create a demonstration video using any screen recording tool. The video must showcase your work, with a clear view of your code and explanation of its functionality.
   * Ensure that the video content is directly related to the work you have completed and not pre-recorded or unrelated content.
   * The demo video should be included in both the Web Application and REST API folder, as well as in the Android Application folder.