



May 2020

Refactory

Catalyst 004

Final Technical Assessment

REFACTORY CATALYST4 FINAL TECHNICAL ASSESSMENT

Introduction:

The world is currently faced with a pandemic of Coronavirus Disease (COVID-19), a new virus that spreads so fast through droplet infection especially in crowded places and causes illness. It is spread from person to person through sneezing or coughing (droplet infection), human to human contact and contact with contaminated surfaces. – [Ministry of Health Uganda] --

As the world grapples with the coronavirus, public health of course must be the first level of concern, with focus being on preventative and containment measures as well as equipping and preparing the global health care systems' capacity to confront the pandemic. – [PwC Uganda] --

Challenge:

In answer to the call for public safety, the ministry of Health Proposed an electronic Information system to smoothen their work of trying to ease the effect of corona virus disease in Uganda. The proposal, was approved, the system was documented, planned, and designed as well, and now all efforts are directed towards implementation of the system. You are one of the selected qualified and technically competent full stack Software developers and You have been assigned the "Virus-Test Registration form" module to be used for registration of whoever is to be tested for Corona virus. Prove your competence by honoring and fully addressing the challenge presented to you.

Proposed Module development time:

Total Duration: 10Hours

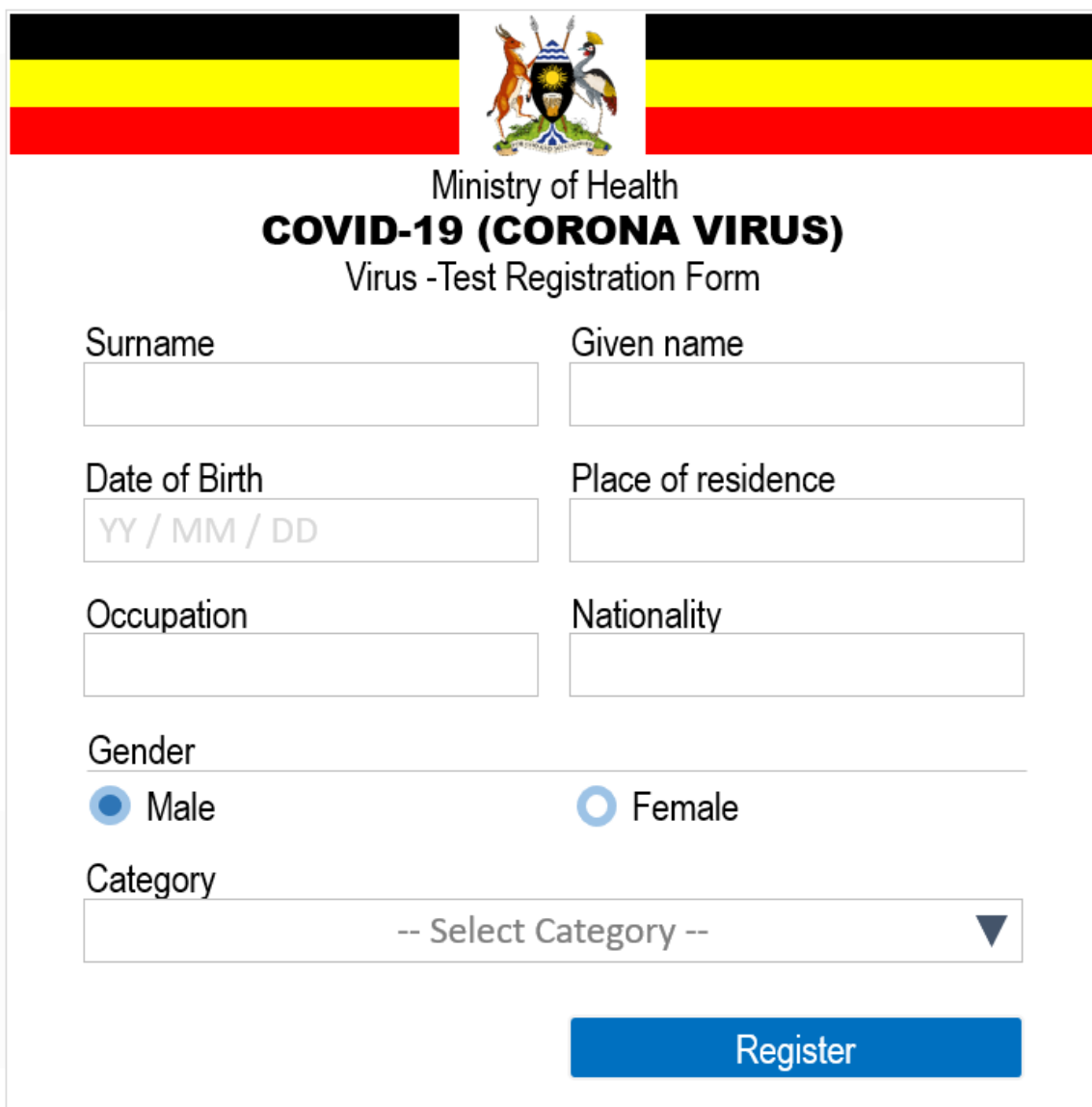
Requirements [What to be Implemented]:

- System should render the registration form upon request.
- User can enter details of Patient into the registration form.
- User can submit the registration form (with Patient details)
- Upon registration-form submission, the system should validate all fields.
- Upon registration-form validation, if any field is invalid, the system should Inform the user about the invalidity and terminate the submission process.
- After registration form validation, the system should submit a form only if all fields are valid.

- After registration-form submission, the system's Resource server should receive the form data and store it in the database.
- After a successful registration form submission, Validation and data storage, the system should redirect to the same form or reset the form and notify the user with a success message.
- The success message should disappear upon close or upon focus of any of the form elements.

Proposed User Interfaces according to the form designs:

According to the designs of the proposed system, after requesting for the Registration form from the server, the form is expected to look as in the Figure below after it's initial rendering.






The form is titled "Ministry of Health COVID-19 (CORONA VIRUS) Virus -Test Registration Form". It features a header with the Indonesian flag and the national emblem. The form contains several input fields for user registration:




- Surname**: A text input field.
- Given name**: A text input field.
- Date of Birth**: A text input field with a placeholder "YY / MM / DD".
- Place of residence**: A text input field.
- Occupation**: A text input field.
- Nationality**: A text input field.
- Gender**: A section with two radio buttons, "Male" (selected) and "Female".
- Category**: A dropdown menu with the placeholder "-- Select Category --" and a downward arrow.

A blue "Register" button is located at the bottom right of the form.

Below is the proposed look for each of the invalid fields after form validation.

				
<p>Ministry of Health COVID-19 (CORONA VIRUS) Virus -Test Registration Form</p>				
<p>Surname</p> <div></div> <p>This field is required</p>		<p>Given name</p> <div></div> <p>This field is required</p>		
<p>Date of Birth</p> <div>YY / MM / DD</div> <p>Select Date of Birth</p>		<p>Place of residence</p> <div></div> <p>This field is required</p>		
<p>Occupation</p> <div></div> <p>This field is required</p>		<p>Nationality</p> <div></div> <p>This field is required</p>		
<p>Gender</p> <div><input checked="" type="radio"/> Male <input type="radio"/> Female</div>				
<p>Category</p> <div>-- Select Category --</div> <p>Select Patient Category</p>				


Below is the proposed look for the Registration form immediately after a successful Patient registration.



Ministry of Health

COVID-19 (CORONA VIRUS)

Virus -Test Registration Form

Registration was successful ! 

Surname

Given name

Date of Birth

YY / MM / DD

Place of residence

Occupation

Nationality


Gender

☒ Male

☐ Female

Category

-- Select Category --



Register

Form Validation Guidelines:

Validation of the Form Fields Should be done basing on the following guidelines:

1. Surname Field:

It should be between 1 to 16 alpha-bet characters (limits exclusive)

2. Given Name Field:

It should be between 1 to 16 alpha-bet characters (limits exclusive)

3. Date of Birth Field:

The test can only be carried out to a Patient who is at least One year old and at most 150 years old.

4. Place of Residence Field:

It should be between 1 to 20 alpha-bet characters (limits exclusive)

5. Occupation Field:

It should be between 5 to 50 alpha-bet characters (limits exclusive)

6. Nationality Field:

It should be between 5 to 20 alpha-bet characters (limits exclusive)

7. Gender Field:

Patient can be Male or Female but neither none of the two nor both. though set Male as default

8. Category field:

This field represents a category of a patient being registered, which is to be selected from the availed list of categories as below.

-- Select Category – should be used as a place holder (Automatically set value as default value) and hence should not be considered as a valid Category input value.

A valid Category input value can only be one and only one of the following options.

- Returnee
- Contact
- Alert
- Volunteer

Programming Languages, Libraries, Frameworks and tools to be used:

1. HTML
2. CSS
 - Bootstrap (optional)
3. JavaScript
 - JQuery (optional)
4. Nodejs
 - Vue.js (optional)
 - Express
 - Body-Parser
 - Pug (optional)
 - Mongoose
 - Passport.js (optional)
 - bcrypt (optional)
5. Git
 - GitHub
 - GitHub Desktop
6. MongoDB
 - MongoDB Compass

Text Editors and IDEs:

- Visual Studio Code (VScode).

Preparation Guidelines:

Note: Assessment Starts right from here

These are the steps you ought to take to get ready to start building the project

Step 1: Check for whether you have Git installed on the computer you are to use.

Step 2: Login to your GitHub account (Create one if you don't have).

Step 3: Follow this link <https://github.com/tech-refactory/Refactory-Catalyst0004-Final-Assessment> and fork the repository named **Refactory-Catalyst004-Final-Technical-Assessment**.

Step 4: Now clone a fork (copy) of the above-mentioned repository to any desired location (directory) on your computer (like on the desktop).

Step 5: Open the **Cloned repository** in a text editor and a terminal or Console input and output window (Most preferably VSCode with Git Bash as the integrated terminal) Otherwise if you are to use a text editor with no integrated Console Terminal, Open the Cloned repository in any terminal availed by your Operating system, i.e. **CMD, POWERSHELL LINUX TERMINAL, etc.**

Note: Every time you are to stage changes, commit changes or push changes to a remote repository, ensure that the path is pointing to this cloned repository.

Note: Endeavor not to Initialize any repository within this repository.

Note: Ensure to have the gitignore file (if not create one) right at the start of this repository. Use it to ignore all the un-necessary files and Directories within this repository so you don't make them part of your commits and pushes.

Step 6: Open that cloned copy on your computer and then within it create an empty folder and name it with your First and Last name (or Sur and Given name).

Note: You may receive Instructions from your Invigilator or Instructor in a refined way from what is in this step (Step 6)

Step 7: Within that folder (the one with your name) is where you should have your project done. Now start and work on your project as you add, commit, and push to update your remote repository.

Note: The repository you push changes to should be (a fork) the one you attained after forking the central repository (as in step3). Don't Push to (or try to update) the central repository during development.

Note: You are expected to make only one pull request and that is at the end of your assessment. All pull requests will be merged once at the end of the assessment by the Administrator of the central repository.

Success wishes from
The Refactory Technical Department
On behalf of Refactory

