**MRA-TaxNet Applicaton**

**Development Guide**

Allan

Chafukira

**MRA-TAXNET APPLICATION DEVELOPMENT**

**INTRODUCTION**

MRA-TaxNet is an application being developed by Allan chafukira as part of SDO (System Development Officer) interviews -phase two at Malawi revenue authority- MRA. The application is being developed with the intension to help MRA expand its tax network by registering new tax payers.

The application is still in the development stage as of 26th November 2021, the application is being developed using the PHP programming language. The application development is following a System Development Life Cycle (SDLC) methodology called the Rapid Application Development (RAD). Under this SDLC, a technique called prototyping has been adopted.

**DEVELOPMENT STAGES**

Note: The application can be developed using any programming language. This document gives a structure on how to go with your development.

1. **REQUIRMENT GATHERING AND ANALYSIS**

The first thing is to understand the problem given. This will help in determining the requirements for the project. In this stage we look at core functionalities of the proposed application in order to determine what data and outputs will be application be using, who are the intended users of the application, and what tools will be used to develop the application. After collecting all the requirements, do analysis to determine if the application is developable under the given requirements and other factors.

1. **DESIGN**

In order to understand how the application will work, the design phase had three deliverables. The **user interface design**, **the architecture**, and **the database/endpoint** design. These designs will help in producing a clear understanding of how the to-be application will operate and require in terms of software, hardware, and the network infrastructure.

**User interface design.**

The user interface design determines how the application will interact with the users (tax officers). It defines how the users will interact and perform different functionalities through the new application. The interface of the application adopted an administrative interface theme called AdminLTE. The them was selected due to its dashboard features.

The other interface designing part mainly involved the creation of forms (login and add-taxpayer forms), and identifying the type of input and output that the application will accept and send back as feedback. Mainly the user interface design involved the input mechanism design, which specified what kind of data the application will accepts (data input validation rules) and how the application will capture that data. The output mechanism design defined how the application will provide information or communicate to the application users.

**Architecture design.**

The architecture design mainly determined what type of hardware equipment to be used and how the software applications will be distributed across the chosen hardware equipment. In our case, we have an API end point, which requires our application to always be online, hence the need for network devices.

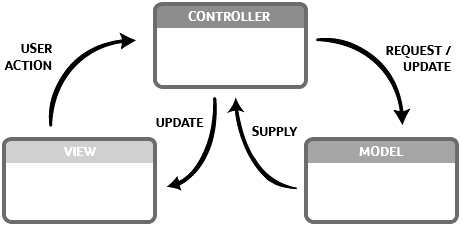
**Database design.**

MRA-TaxNet is uses a web service API end point provided by the MRA. In this case the core database is designed already by the MRA team. As a result, in this designing phase, we only focus on what type of data should we be sending to the endpoint and how to handle responses form the endpoint.

1. **DEVELOPMENT/PROGRAMMING**

The software application is being developed using the **Model View Controller (MVC)** approach using the PHP programming language. This approach separates the application into **three** sections namely; the **models**, the **views**, and the **controllers**.

* The model section holds all the business logic and data access code. This is the code that the application uses when interacting with the endpoint/database. In our case it’s the section responsible for all request and response handling to and from the end point.
* The view section holds all the display logic code (which is mostly the application interface features).
* The controller section is used as an interface between the model section and the view section. In our application the controller is also responsible for the server side-data data validation sanitizing.

Figure 1 gives a summary of the MVC approach.

The application is following the client-server architecture. In this type of architecture all the presentation and application logic reside on the client side (client machine). All the data access and data storage logic are on the server side (server machine). In this project, both the client side and server-side ware configured to be running on the same machine. In this case, Google chrome browser (or any other web browser) is our client side, while XAMPP (Cross-Platform, Apache, MariaDB, PHP, and Perl) is server side.

**Development Tools**

MAR-tax-Net is being developed with the help of the following tools:

* The PHP programming language
* HTML
* CSS
* JavaScript
* Bootstrap
* Visual Studio code.

1. **APPLICATION TESTING**

Since the application is still under development, white box testing is being used by the application developer to test the inner functionality modules of the application. This testing mainly focuses of on putting the applications actual code into test, for example, testing how code **validates**, **flirters**, and **sanitize** input data.

This kind of testing, tests the inputs that the code is meant to accept to produce the needed output.

Note: Snice the application is interacting with an API endpoint, all testing should be done online for better response and modifications.

1. **DEPLOYMENT**

Snice the application is being developed for MRA; we assume that it will be hosted /installed on a MRA’s local servers upon completion. From there it can be accessed by the tax officers through the devices with communication link to the servers.

**Summary**

MRA-TaxNet is a tax payer registration application being developed to assist MRA widen its tax network. The application can be developed using any programming language by following the outlined steps above.