Access Point Analytics - Developer Manual

Table of Contents:

Introduction

Purpose

The purpose of this document is to provide the technical steps to set up the Web Application and necessary services, and provide insight surrounding the system to allow for maintainability in the future.

Project Overview

The Access Point Analytics project is first and foremost a *Web Application*. There are a few secondary services that are being utilized, but the web application ties together all of these services.

Summary

The primary goal of this project is to ingest, analyze, and aggregate data regarding the various Access Points across TCU's campus. These insights are presented via a Web Application interface to TCU Network Service employees and broken down into several *Key Performance Indicators* (KPIs) and presented through various data visualization mediums such as tables and graphs.

Services

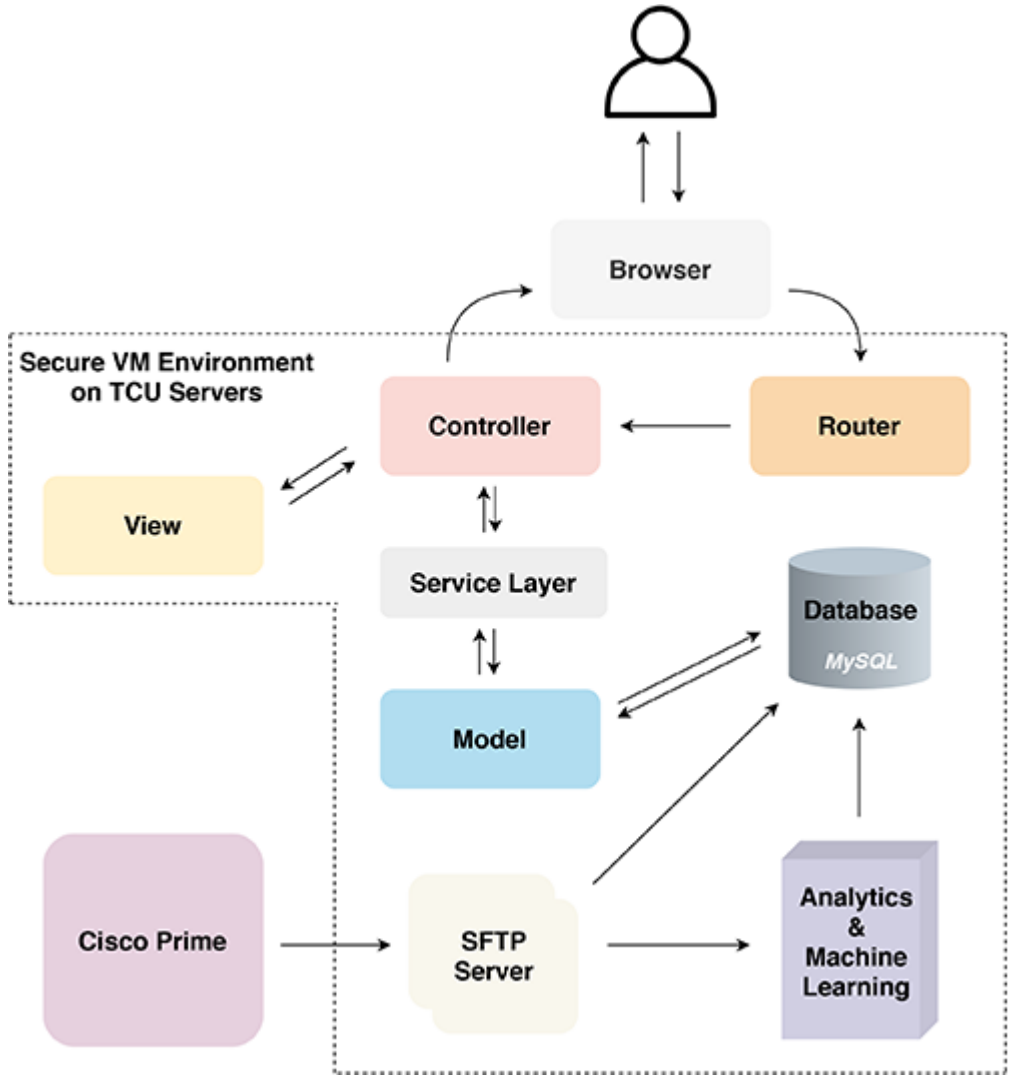
- **Web Application** - The .NET Core Web Application that employees of TCU Network Services will interact with. The code for the web application is stored in the KPI Analytics repository.
- **Cisco Prime Server** - The enterprise application that sends the access point related CSV data files for consumption via *SSH File Transfer Protocol* (SFTP).
- **Data Import Scripts** - Ingestions scripts written in C# that execute based on a cron job schedule on the server. Each KPI will have its own respective ingestion script. After these are executed, data should be stored in the MySQL instance. The code for the import scripts is stored in the Data Scripts repository.
- **MySQL Server** - The system database that ties together the *Data Import Scripts* and the *Web Application*. All application data is stored here from raw KPI data, down to user data.

Tech Stack

Tool	Version	Documentation Resource
C#	8.0	https://docs.microsoft.com/en-us/dotnet/csharp/
.NET Core Framework	3.1	https://docs.microsoft.com/en-us/dotnet/

Tool	Version	Documentation Resource
jQuery	3.3.1	https://api.jquery.com/
Chart.js	2.9.3	https://www.chartjs.org/docs/latest/
jQuery tablesorter	2.31.1	https://mottie.github.io/tablesorter/docs/
Bootstrap	4.3.1	https://getbootstrap.com/docs/4.1/getting-started/introduction/
FontAwesome	5.11.2	https://fontawesome.com/how-to-use/on-the-web/referencing-icons/basic-use
MySQL	8.0.17	https://dev.mysql.com/doc/

System Architecture



Setting up the Development Environment

In this section, we will cover setting up a local development environment for adding features/developing on the web application.

Download Docker Desktop

Follow this link to install Docker Desktop.

Clone the Data Scripts Repository

Navigate to the Data Scripts repository.

Clone the repository:

```
git clone git@gitlab.com:ap-analytica/data-scripts.git
git pull
```

Deploy the Docker Instance

Navigate to `LocalDBMigration/`

```
cd data-scripts/LocalDBMigration/
```

Build, create, and start the container

```
docker-compose up
```

This will load the MySQL server with the following information:

```
{
    "user": "root",
    "password": "tcuit",
    "host": "127.0.0.1",
    "port": 3308,
    "schema": "kpi"
}
```

Load in the Schema

Using the provided MySQL file that contains the base schema, functions, and data; run the following:

```
mysql -h 127.0.0.1 -P 3308 -u root -p < [Name of the schema file]
[ENTER PASSWORD]
```

Confirm that there are now ~3,200 entries in the `access_point` table:

```
use kpi;
SELECT COUNT(*) FROM access_point;
```

Creating a User

If there are no base users for the web application, you can create one with the following command:

```
INSERT INTO `user` (username, password) VALUES ("[USERNAME]", sha2("[PASSWORD]",
512));
```

Running the Web Application

Clone the Repository

Clone the KPI Analytics repository using `git clone git@gitlab.com:ap-analytica/kpi-analytics.git`.

Download .NET Core 3.1

If .NET Core 3.1 is not installed on the machine you are using, you can download it here.

Open the project in Visual Studio

In the root of the project, there is a `KPI-Analytics.sln`, if you are using a flavor of Visual Studio, launch Visual Studio, and click `Open a Project`, then select this file.

Add your `appsettings.Development.json` file

For local development, you need to specify your database connection strings to match a locally set up MySQL environment (steps can be found above). The production connection strings are the default in the application code found in the `appsettings.json` file.

In the `kpi-analytics/KPI-Analytics/` directory, create a file named `appsettings.Development.json`, and paste the following:

```
{
  "ConnectionStrings": {
    "DefaultConnection": "server=127.0.0.1;port=[PORT];user=[USERNAME];password=
[PASSWORD];database=kpi;"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Debug",
      "System": "Information",
      "Microsoft": "Information"
    }
  }
}
```

Note: If you are using the Docker installation from the tutorial, your port will be `3308`, otherwise, your port should be `3306`. Replace the `user` field and `password` field with your respective credentials from the database user you created above. This user is not the user you creating in the `user` database table, but is the user used to log in to MySQL.

Locally Deploying the Application

Press the play button in Visual Studio, or run:

```
dotnet build
dotnet run
```

in terminal.

When the application runs in the browser, you can login with the user credentials you inserted into the `user` table.

Glossary of the Solutions Contents (by subproject)

`kpi-analytics/`

This is the glossary for the KPI Analytics repository, AKA the web application.

- **`.gitignore`** - defines filepaths for files/subdirectories that you don't want uploaded to git. Should include any file with sensitive information, and includes all of the unnecessary files generated by Visual Studio.
- **`KPI_Analytics.sln`** - a Visual Studio Solution file that organizes the project, project items, and solution items in the web-app. Open this when opening the project in Visual Studio.
- **`README.md`** - general info about the project.

`kpi-analytics/KPI-Analytics/`

- **`Dependencies/`** - a list of installed NuGet packages used within the project. Only includes C# packages, all JS/CSS packages are downloaded into `wwwroot/`.
- **`Controllers/`** - handles and responds to user interaction. The controller should remain thin, gather information for the view, and render the view. Shouldn't be overly complicated. Business logic should be put into the service layer. Routes to the controller are defined in `Startup.cs`.
- **`Interfaces/`** - interfaces that describe the service layers of the application.
- **`Models/`** - set of classes that describe, and allow us to interact with the data. Should be a reflection of the current state of the database. See this page for more information. Each table in the database should have a model, and all relationships are described in `Models/KPIContext.cs`.
- **`Properties/`** - internal build and debugging information.
- **`Services/`** - injectable services for the web application. Used for all the business logic and gathering of data from the Models. Sits between the Controller and Model. Handles accessing MySQL through the context.

- **`Views/`** - handles the UI presentation of data as a result of a request received from the controller.
- **`wwwroot/`** - holds static files that will be publicly accessible from the web-app. Has subdirectories to put css stylesheets, JavaScript files, and libraries such as jQuery, Bootstrap, Chart.JS, etc.
- **`KPI-Analytics.csproj`** - contains list of files to be compiled as a part of the project, and general project information.
- **`Program.cs`** - contains the main method for execution.
- **`Startup.cs`** - configures services and the app's request pipeline, specifies when the app's host is built. Contains the routes and services declarations.
- **`appsettings.Development.json`** - configuration settings for the development site. Read if the `ASPNETCORE_ENVIRONMENT` environment variable is set to `Development`. Contains connection strings, etc.
- **`appsettings.json`** - holds configuration settings for the site (read by default) for production. See this link for further explanation.
- **`Constants.cs`** - a file containing project wide constants to eliminate disconnect.

`kpi-analytics/KPI-Analytics.Documentation/`

- **`KPI-Analytics.xml`** - the autogenerated XML documentation for the `KPI-Analytics` project.

`kpi-analytics/KPI-Analytics.Tests/`

- **`KPI-Analytics.csproj`** - contains list of files to be compiled as a part of the project, and general project information.

General Breakdown of Testing Suite

Each KPI has its own directory for testing its controller actions and service layer. These are depicted by `[KPI NAME].Tests`.

In the root level of theses sub-directories is a `[KPI NAME]MockData.cs`. This is the file that generates the Mock Data for each test suite.

There is also a `Controller.Tests/` folder, which contains tests files. Each test file is a test for a respective controller action.

`data-scripts/`

This is the glossary for the Data Scripts repository, AKA the import scripts.

- **`.gitignore`** - defines filepaths for files/subdirectories that you don't want uploaded to git. Should include any file with sensitive information, and includes all of the unnecessary files generated by Visual Studio.

`data-scripts/ImportScripts/`

- **`Data/`** - contains sample CSV imports generated from the Cisco Prime server, per KPI.
- **`scripts/`** - each KPI has an import project that will ingest data from a CSV from its respective data folder. These can be run using the `import-script.bat`. Documentation is pretty extensive for each of these projects.
- **`scripts/import-script.bat`** - the batch file that runs each KPI import program. Note, this will likely need to be changed when the system is migrated.
