CharityQ

Technical Manual

Capstone-004 Spring 2018

MANAGEMENT INFORMATION SYSTEMS | UNIVERSITY OF NEBRASKA AT OMAHA

Table of Contents

Introduction

The purpose of this document is to describe the architecture, design, technical dependencies, key configurations and specifications used to develop and deploy the CharityQ application. Here the recipient is assumed to be proficient in administering and maintaining the application developed.

Purpose

CharityQ is a platform designed to save business owners their time and money by assisting with tasks related to charitable giving. The application is intended to contain the following functionality:

- **Standardized Requests** - The application is intended to standardize and organize donation requests to make them easier to process. A standard donation form is attached to the business website where donation requesters can submit their requests.
- **Assisted Decisions** – The application also assists the business user in processing the requests to decide which organizations they prefer to donate to. Business rules are determined by the admin user and automatically sort requests based on the business preferences.
- **Communication** – Once a decision is made, the system generates an email to the donation requestor. Email templates for approved and rejected requests are available and may be edited by the admin user.
- **Organization of Donation Information** – With all donation requests available within one application it is easy for a business owner to confirm the business is helping the organizations that align with the company's vision. By setting a monthly budget, a business owner is also able to quickly determine when the company can assist additional organizations, and when to pass on a donation request.

Terms/Definitions

Terms	Definition
Agile	A software development project management methodology that is based on iterative development; it emphasizes on evolving solutions and client interaction
AWS S3	Cloud service provided by Amazon Web Service used for file storage.
CharityQ	The application organizations can use to organize and track donation requests and response to donation requestors.
Composer	Used to manage Laravel dependencies
GIT	version control system
Heroku	cloud platform based on a managed container system, with integrated data services and a powerful ecosystem, for deploying and running modern apps.
PHP Storm	standard development IDE
Pull	Send code from the online code repository to the developer's local machine
Push	Send code from the developer's local system to the online code repository
Selenium	Open source programs used to automate the testing of a web application.
Stripe	A third-party payment processing service used by the application

User	Any individual signing in to the CharityQ application
WAMP	Open source applications, combined with Microsoft Windows, which are commonly used in Web server environments. The WAMP stack provides developers with the four key elements of a Web server: an operating system, database, Web server, and Web scripting software.

System Requirements

This section describes the software and hardware required for an individual to administrator the CharityQ application.

Hardware Requirements

CharityQ is a web browser-based application which can be accessed either through a browser on a desktop, laptop or a mobile device. It is recommended that the browsers being used are compatible with HTML5 and CSS3.

Commonly used browsers and the recommended versions:

Internet Explorer: Version 9 and above

Mozilla Firefox: Version 64 or above

Google Chrome: Version 60 or above

Safari: Version 5 or above

Commonly used mobile browsers and the recommended versions:

Safari: Version 5 or above

Google Chrome: 65 or above

For developing the application, an application, database and a web server are needed. To run these server's, it is recommended developers have the following requirements installed or updated on the development environment:

Operating Systems: Windows 8 or higher, Ubuntu 12.0.4, MacOS 10 or higher

A 64-bit computer processor

Software Requirements

For developing the CharityQ application these are some of the required software's and their versions:

Server

PHP: Version 7.0

Apache Server: Version 2.4.23

MySQL: Version 5.7.14

Framework

Laravel: Version 5.5

Version Control

GitHub Desktop: Version 1.0.9

GitHub: Version 2.11

Development Environment

PHP Storm: Version 2017.3

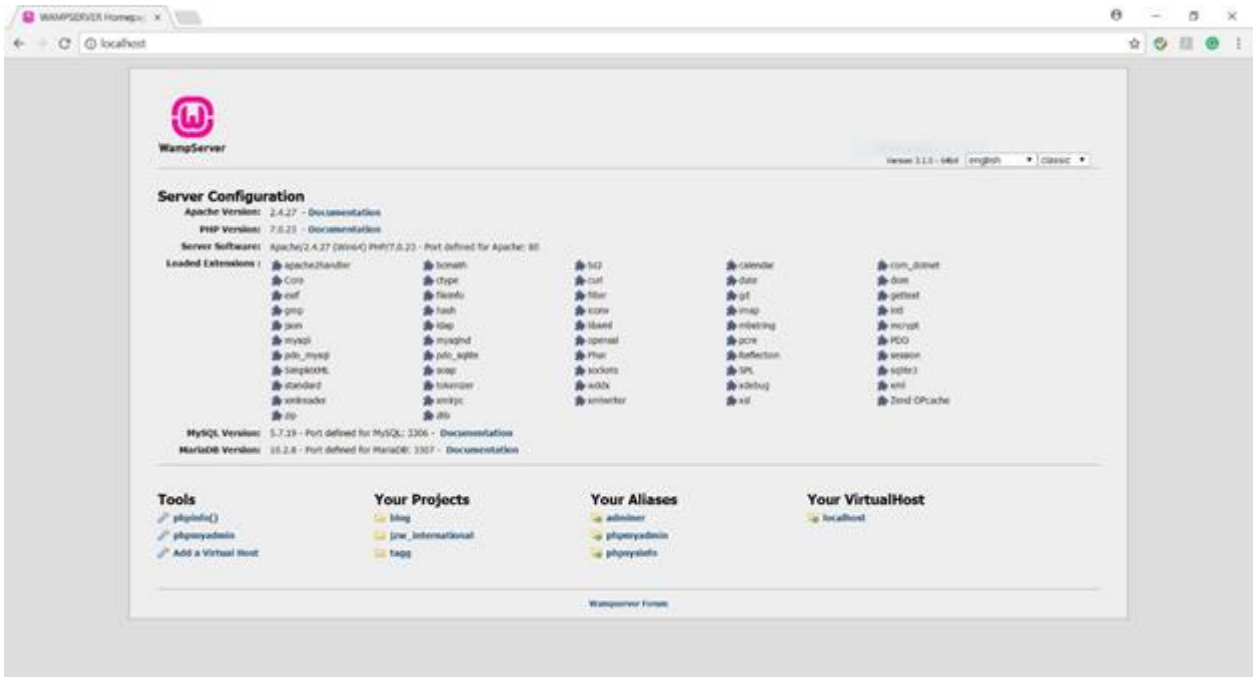
Local Environment Installation

The following guide is intended for developers at TAGG to help them get their local environment working and verify they can commit changes to their team repository.

Install WAMP/MAMP

WAMP (MAMP or IOS) will be used for the local environment which can be obtained from the link: http://www.wampserver.com/en/ . The WAMP server will install Apache2, PHP, MySQL database, and PhpMyAdmin to manage the local database. WAMP server must be started and its status can be verified by checking the system tray for the "Green WAMP" indicator.

WAMP installation can also be verified by going to http://localhost, under WAMP settings.



Click on the phpMyAdmin. This tool to manages the database on the local system. The opening page should look like this:

By default, the username is root and there is no password. Click on Go to access the database of the application.

PHPStorm

PhpStorm by JetBrains will be used as the standard development IDE(*Integrated Development Environment*) which can be obtained from the link: https://www.jetbrains.com/phpstorm/ .

Install GIT

By far, the most widely used modern version control system in the world today is Git. Git is a mature, actively maintained open source project originally developed in 2005 by Linus Torvalds, the famous creator of the Linux operating system kernel. A staggering number of software projects rely on Git for version control, including commercial projects as well as open source. Developers who have worked with Git are well represented in the pool of available software development talent and it works well on a wide range of operating systems and IDEs (*Integrated Development Environments*).

- Download the latest Git from https://git-for-windows.github.io/
- After successful installation, the Git Setup wizard screen appears. Follow the Next and Finish prompts to complete the installation using the default options.
- Open Command Prompt (or Git Bash if during installation if it was selected not to use Git from the Windows Command Prompt).
- Run the following commands to configure the Git username and email using the following commands, replacing Emma's name with the name on the account. These details will be associated with any commits that are create:

```
$ git config --global user.name "Emma Paris"
$ git config --global user.email “emma@gmail.com”
```

Optional: Install the Git credential helper on Windows

Bitbucket supports pushing and pulling over HTTP to the remote Git repositories on Bitbucket. When interacting with the remote repository, a username/password combination is required. Credentials can be stored with the Git Credential Manager for Windows.

Install Composer

The installer will download composer and set up the PATH environment variable, so the local system can simply call composer from any directory. The composer can be downloaded from the link: https://getcomposer.org/. There are several methods to download the global composer, follow the instructions from Download button.

Install Laravel

The Laravel framework has the following server requirements:

- ***v. PHP >= 7.0.0***
- ***v. OpenSSL PHP Extension***
- ***v. PDO PHP Extension***
- ***v. Mbstring PHP Extension***
- ***v. Tokenizer PHP Extension***
- ***v. XML PHP Extension***

Laravel utilizes Composer to manage its dependencies. So, before using Laravel, make sure Composer is installed on the machine.

First, download the Laravel installer using Composer:

```
composer global require "laravel/installer"
```

Place composer's system-wide vendor bin directory in the $PATH so the Laravel executable can be located by the system. This directory exists in different locations based on the operating system; however, some common locations include:

v. MacOS: $HOME/.composer/vendor/bin

v. GNU / Linux Distributions: $HOME/.config/composer/vendor/bin

Once installed, the Laravel new command will create a fresh Laravel installation in the directory specified. For instance, Laravel new blog will create a directory named blog containing a fresh Laravel installation with all of Laravel's dependencies already pre-installed:

```
laravel new blog
```

Alternatively, install Laravel by issuing the Composer create-project command in the terminal:

```
composer create-project --prefer-dist laravel/laravel blog
```

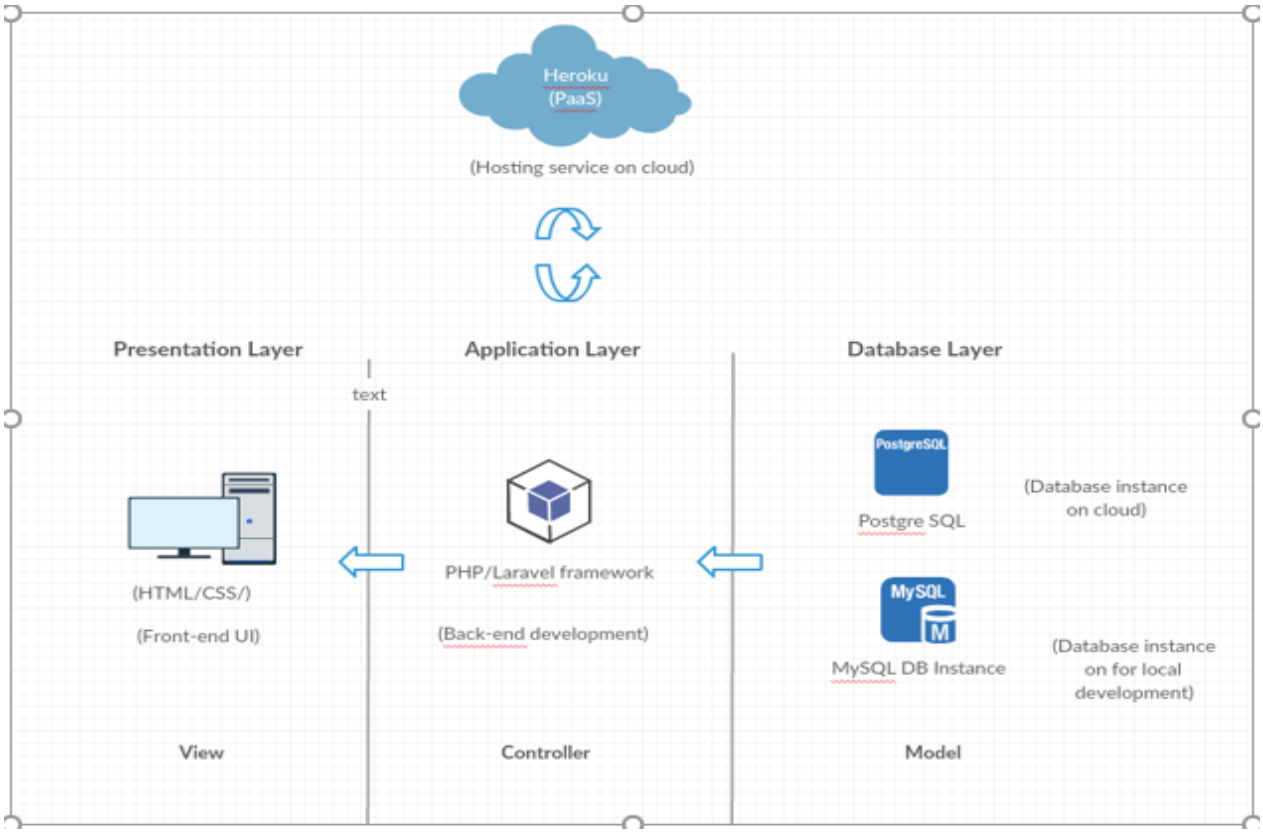
Application Architecture

This Laravel project follows MVC architectural pattern, comprises separation of business logic from presentation associated with GUI.

Three components of MVC pattern:

- **Model**: It is about the data related logic that a user works with. The data being transferred between view and controller. It enforces all the business rules on the related data, by putting the implementation of business rules in model, we can make sure that nothing in application can make our data invalid.

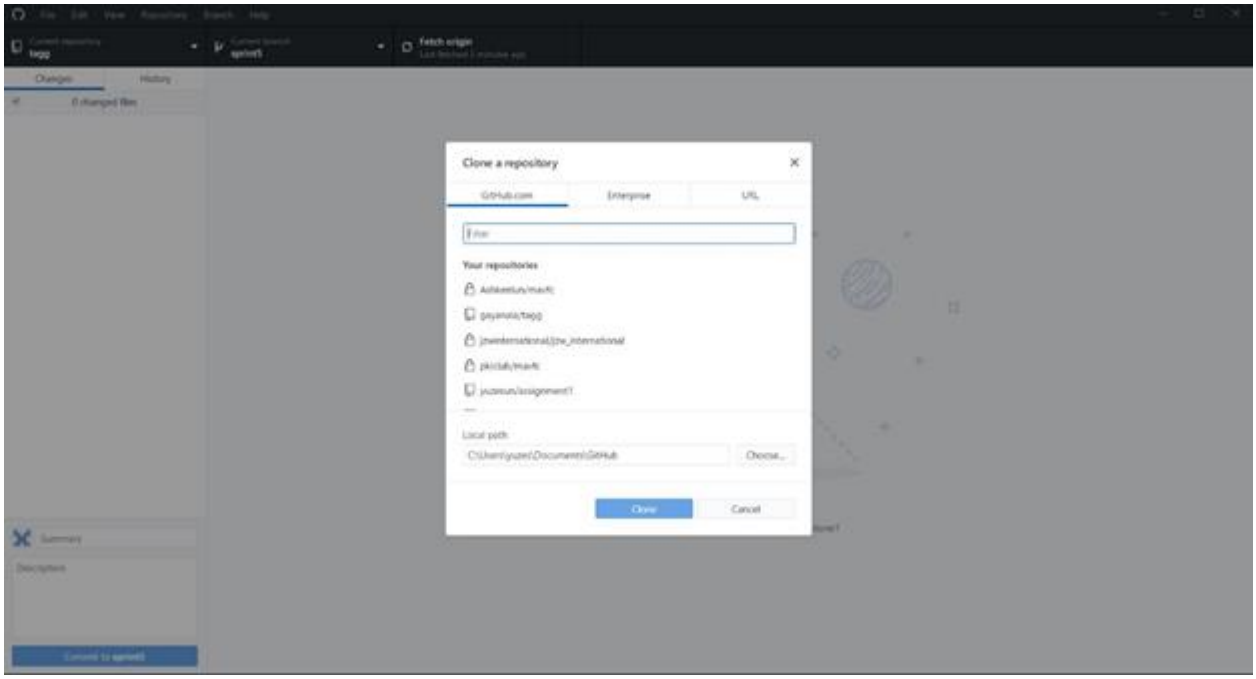
- **View**: It comprises UI logic of the application. For instance, UI components like textbox, checkboxes, etc., that a user interacts with. It never handles incoming data but displays it once available.
- **Controller**: It acts as interface between Model and View to process all the inputs, act on the model and decide what action to perform next. It can be anything such as rendering a view or redirecting to another page.



Set Up TAGG with GitHub Desktop

There are many methods to review, commit and push the code. GitHub Desktop is a convenient method.

- Download the GitHub Desktop form link: https://desktop.github.com/ to install it on a local machine.
- Open the application and add a repository.
- If a local repository is already established on GitHub, click on File -> Clone Repository.

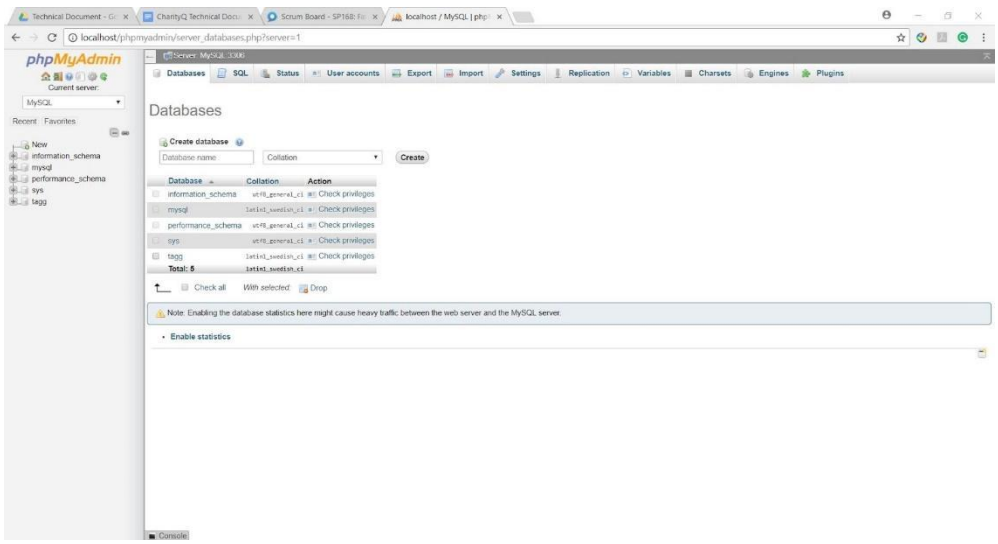


- Select the repository and change the local path to C:\wamp64\www.
- The local path must be in the www folder to connect the project to the local database.

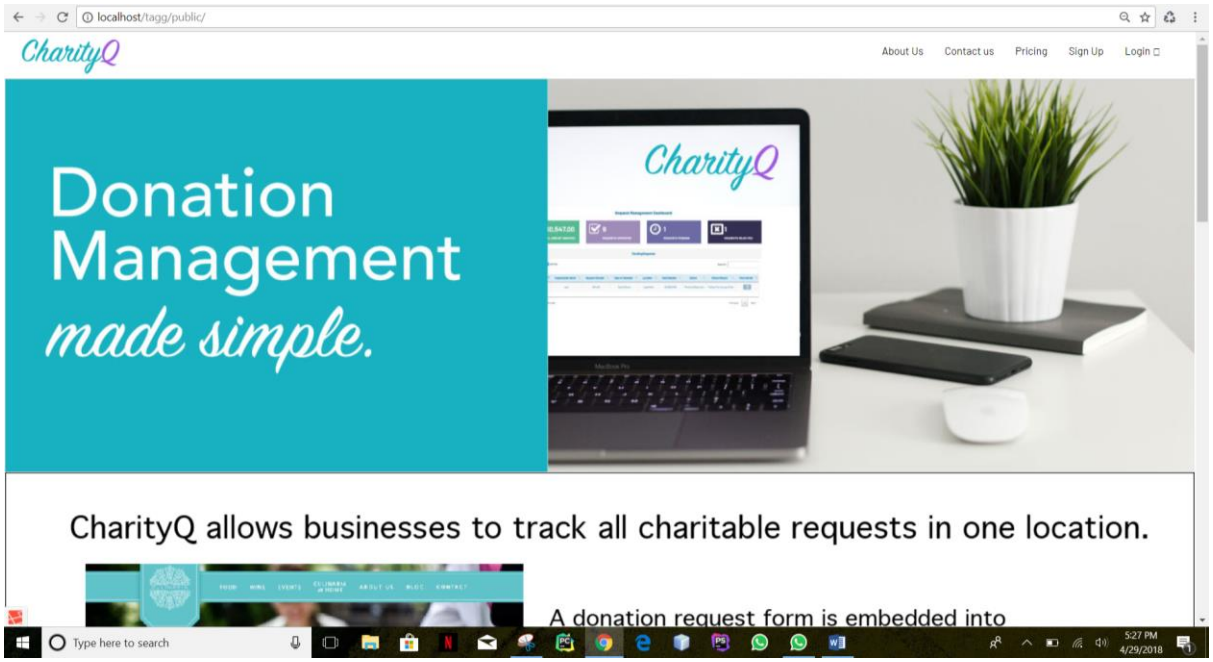
GitHub Desktop allows the user to choose a branch to work on and create a new branch; To commit the code, user can simply add comments then click the 'commit to' button on the bottom left and click on Fetch Origin towards the centre on the top, then click the same button again to push the changes.

Local Database & Website

Log into phpMyAdmin and create **TAGG** database.



The local website will be https://localhost/tagg/public. This is the home page of CharityQ.



Development Environment

Software Name	Version
WAMP Server	3.1.0 64bit
Apache Server	2.4.27
Php	7.0.23
MySQL	5.7.19
Laravel	5.5.40
Git	2.14.1. windows.1
GitHub Repo	https://github.com/gayanala/tagg

Production Environment

The current production environment connected with Heroku. A cloud platform based on a managed container system, with integrated data services and a powerful ecosystem, for deploying and running modern apps. The Heroku developer experience is an app-centric approach for software delivery, integrated with today's most popular developer tools and workflows.

Production URL

The production URL will be https://tagg-uno.herokuapp.com/

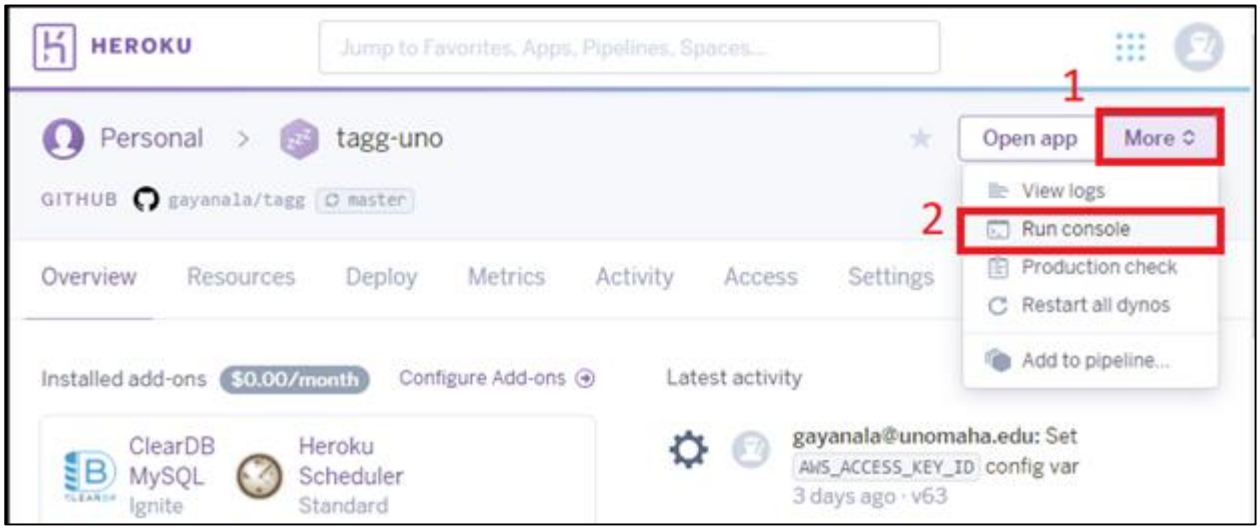
Local Website

The URL to access the application on the local machine is https://localhost/tagg/public

Database Migrations and Seedings

Access the Heroku production environment for database migration and seeding:

- Log into Heroku at https://dashboard.heroku.com/apps/tagg-uno using an email address and password.
- Select **More**
- Select **Run console**



- In the command window, type 'bash'
- Click the **Run** button

Heroku should be connected to the Dyno.

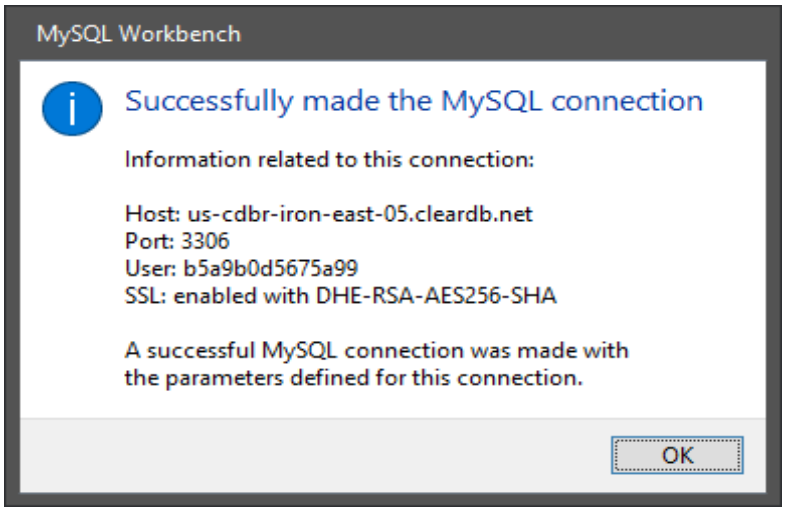
Migrations
Type `php artisan migrate` in the console window to run a migration,

Seeding
Type `php artisan db:seed` to seed the database with the initial setup information. NOTE: seeding the database will delete ***ALL*** existing data in the database!!

Database Access

- MySQL Workbench Client is required to access the database or view/update the data model. Download the software from https://dev.mysql.com/downloads/installer/.
- Open MySQL Workbench
- Click on the + icon next to "MySQL Connections" on the home tab to create a new database connection.
- On the Setup New Connection screen, enter in the following information:
  Connection Name: tagg-uno
  Hostname: us-cdbr-iron-east-05.cleardb.net
  Port: 3306
  Username: b5a9b0d5675a99
  Password à Select "Store in Vault" and enter: 759dfc4f
  Default Schema: heroku_22e68d30d35242e

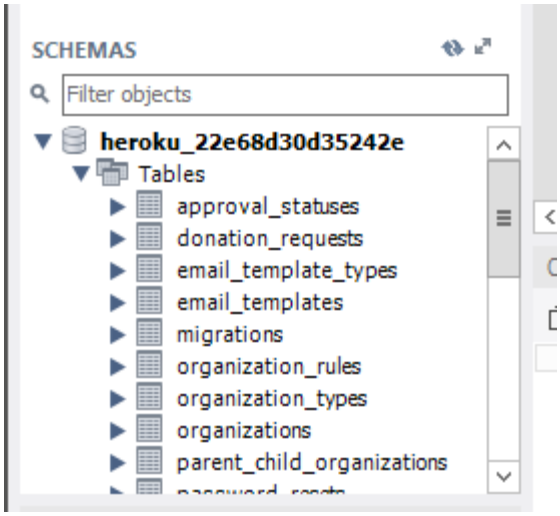
Pressing "Test Connection" should result in the following message:



If the Test Connection was successful, click OK and then OK again.

A connection should now be listed on the Home tab with the name **tagg-uno**.

Click **tag-uno** to connect to the database. All tables will be available on the left side in the Navigator panel, under SCHEMAS:



Integration with Stripe

Functionality

Stripe Integration will help the business registered to pay for the service based on the number of locations, choice of plan monthly or annually, coupon (if any) and the credit/debit card details. The plan auto renews based on the type of plan the business pays for at the time of payment. The plan auto renews monthly or annually based on the type of plan selected.

Configuration

- With reference to the cashier documentation provided by Laravel for version 5.5 the required Cashier package provided by Laravel, created the necessary Database Migrations for the payment system to work.

- On Stripe dashboard, under the API tab a set of test keys and live keys are included to enable the development and production environment
- These are available in the. env file (environment file) 09640964**Plans and business locations**

A unique combination of plan and business locations creates a subscription plan on the stripe dashboard. A coupon can be created on stripe dashboard which can be provided to the businesses for promotion purposes.

Integration with Amazon Web Service

AWS S3:

Amazon web services provides a cloud storage named S3. This can be accessed either through an AWS management console/dashboard or integrated with a web application to store, read, write and delete files easily. Specifically, for the CharityQ app it helps in saving the storage of the app itself and provides an easy way to access files.

AWS S3 Configuration:

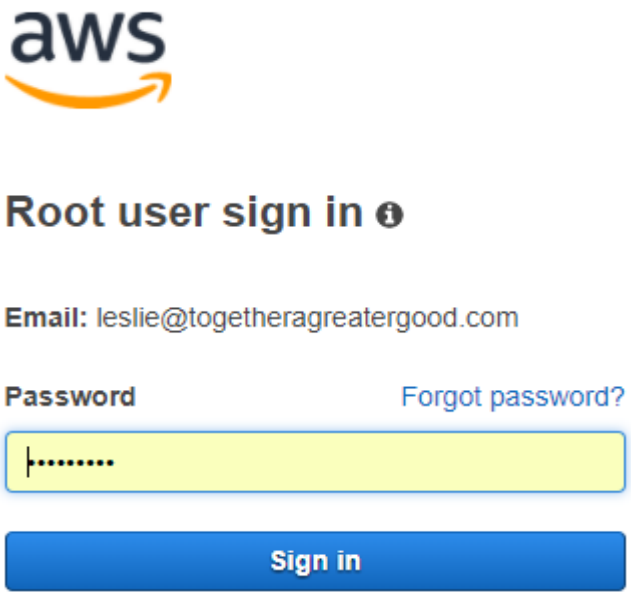
Laravel's Filesystem component makes it very easy to work with cloud storage drivers, and the documentation provided by Amazon does an excellent job of covering how the Storage facade works. A package must be installed via composer.

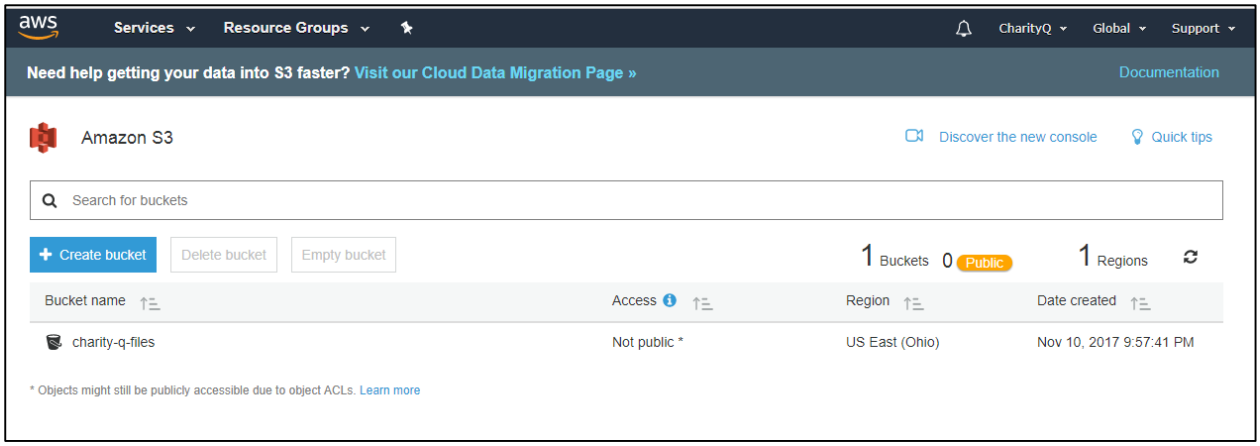
Before using the S3 driver, Install the appropriate package via Composer:

- league/flysystem-aws-s3-v3 ~1.0

On the AWS side, set up

- An S3 bucket
- An IAM user
- An IAM policy attached to that user to let it use the bucket
- The AWS Key and Secret belonging to the IAM user

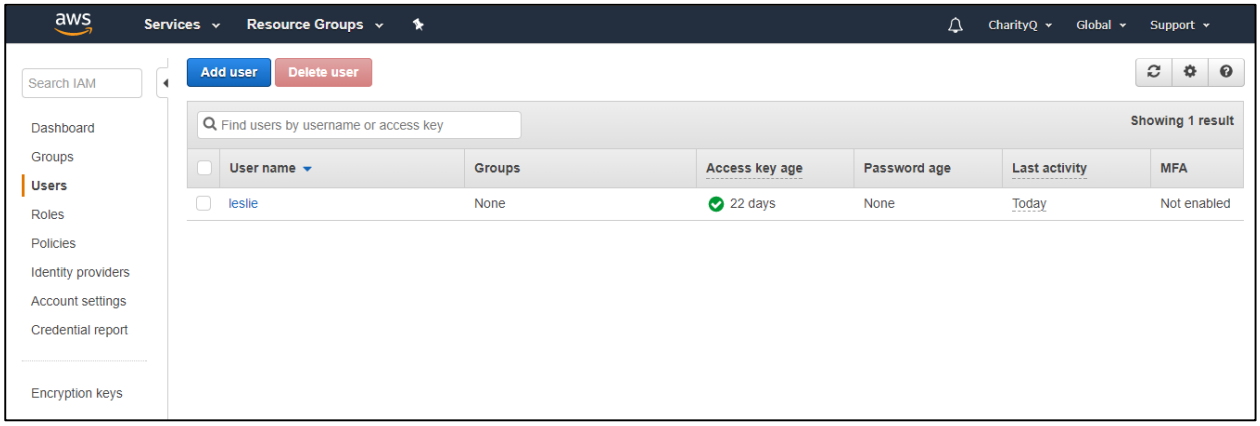




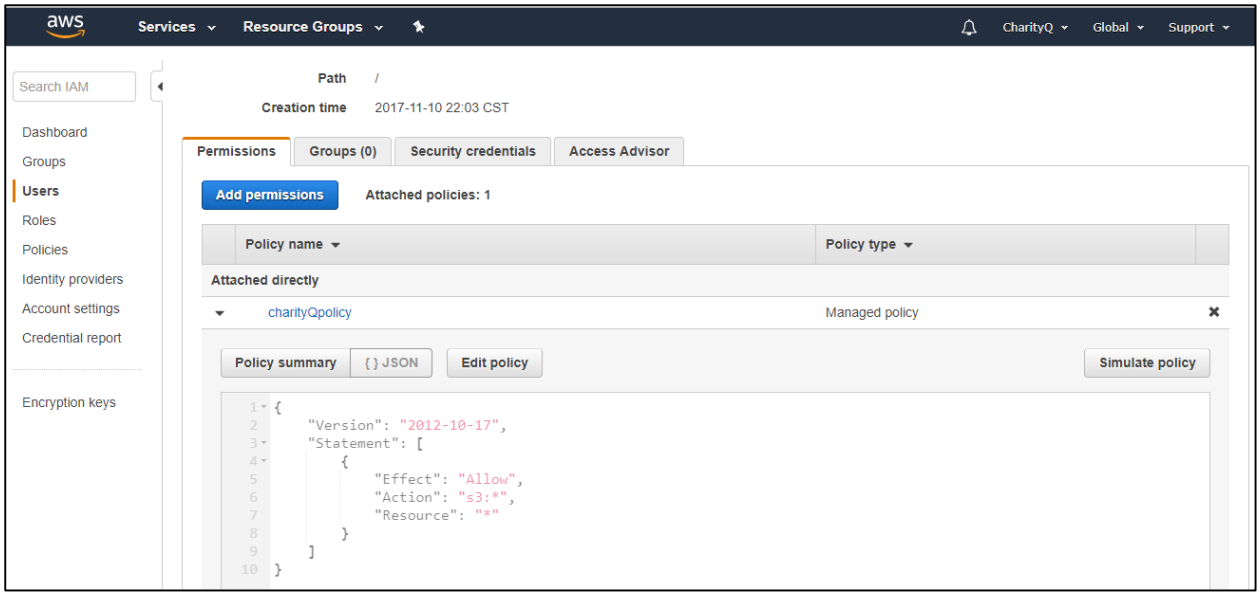
The above screen shows the S3 console, where buckets are created to store files.

- Bucket Name: *charity-q-files*
- Select the region intended to use the bucket. *Example: US East (Ohio)*
- Set permission levels for the bucket. *out of scope for this project*
- Specify an 'I AM User' to access all the content of the bucket.

In the below screen shot 'Leslie' is the user.



Create a basic policy for the user to have the desired level of access. In this example, the user has complete/full access.



- When a new user is created an Access key ID and a Secrete Key (password) are required. This application has one user 'Leslie'
- The secrete key and password can be accessed in settings.
- This informant should not be changed because it is used in the. env and config files of the application.

Quality Assurance

Automation Environment

Installation

All the Automation process from installation to setting up tests is specified below. The following order needs to be followed for installation and using it to perform QA tests.

Step 1 - Download and install JDK and JRE

http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

Step 2 - Install Eclipse IDE for Java EE Developers

http://www.eclipse.org/downloads/eclipse-packages/

Step 3 - Download the Selenium Client for Java, Standalone Server and Driver Software

http://www.seleniumhq.org/download/

Step 4 -Extract all the files to a specific folder on your local machine

Step 5 - Create a path in C drive as "C/Drivers" and place all the driver software in it

Step 6 – GitHub link for TestTagg project

https://github.com/msreeperumbuduru/TestTAGG.git

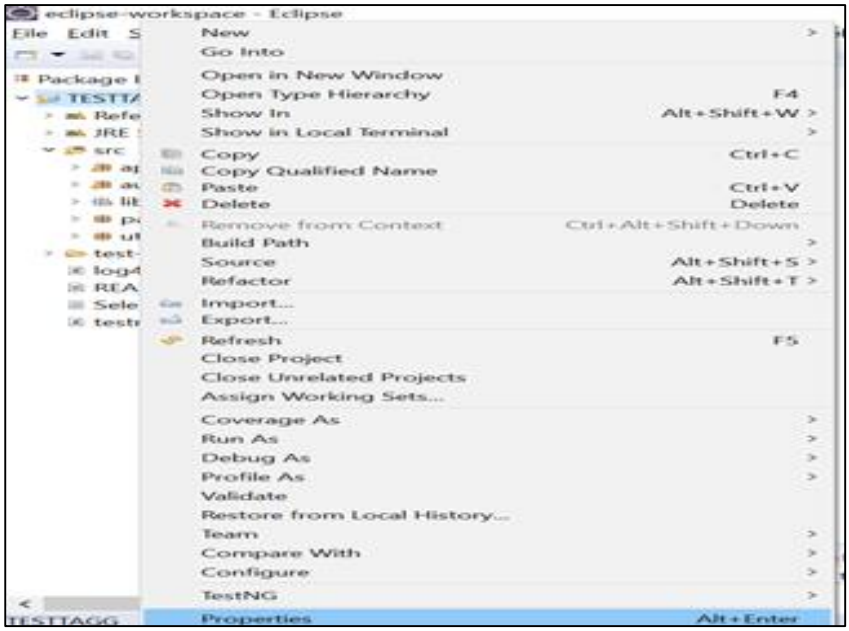
Setup

Make sure all the software listed through Step 1 through Step 3 is configured accurately on your machine. Open the GitHub link mentioned in Step 6 and download the project into your eclipse workspace on local machine.

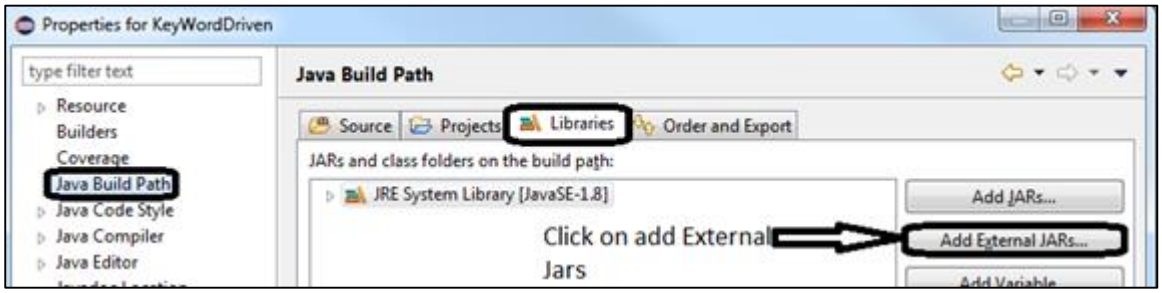
Selenium Web driver

Step 1 - In the IDE, create a new Java Project and import the downloaded project.

Step 2 - Right click on the project and click on properties

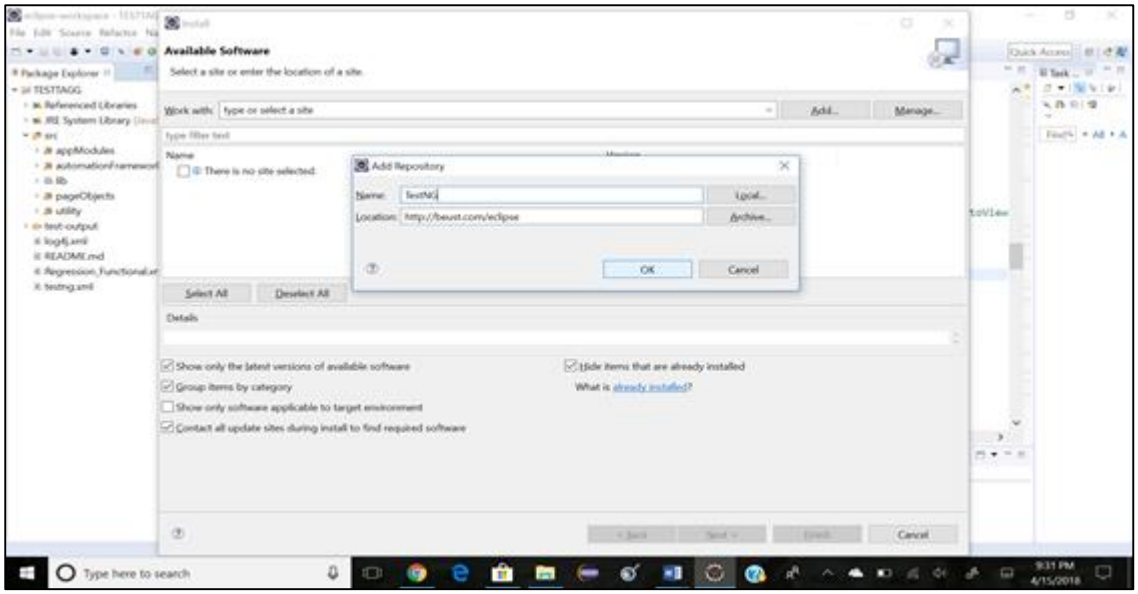


Step 3 - Go to Java Build Path -> Libraries -> Add External JAR's

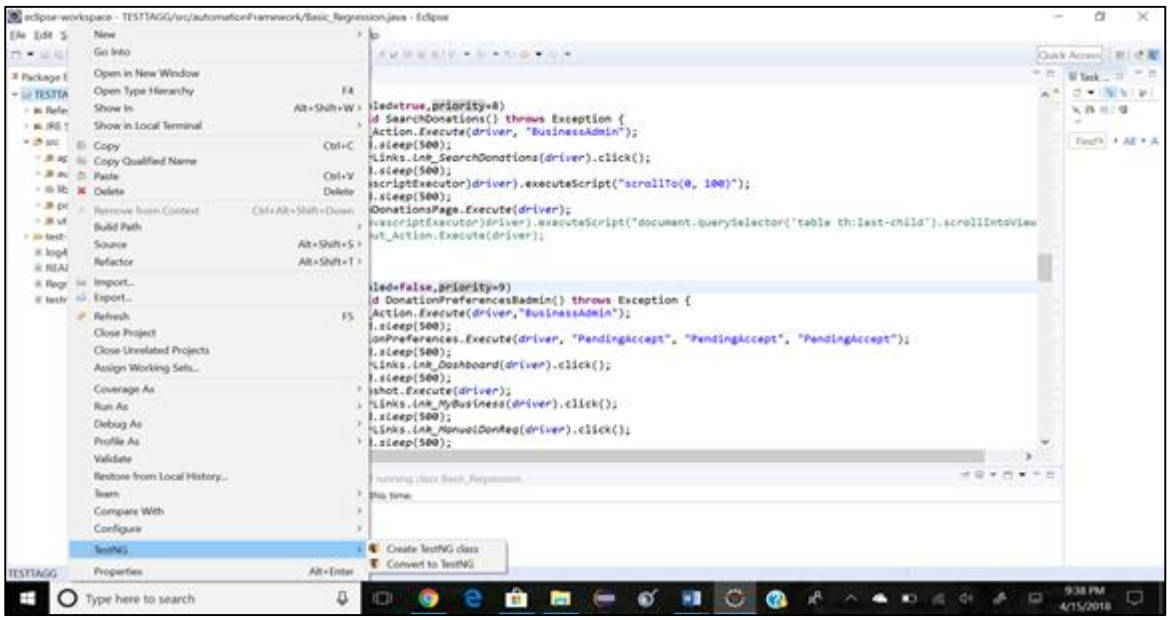


Navigate to the Folder that has all the JAR files, select all the JAR files and add to the project. Ensure all the required jars are in place and project shows no errors due to lack of jar files.

Step 4 - Go to Help -> Install New software from the Eclipse menu items



Enter Name and Locations as “TestNG”, “http://beust.com/eclipse” and navigate till finish. Allow Eclipse to restart after installation.
Right click on the project and check for TestNG option. If TestNG is installed successfully, TestNG will appear in the list as in below screenshot.



Select “Convert to” and select “TestNG Project”. Click finish on the new window. A new testing.xml file will be generated into the project.

Framework

Step 1 - Basic folder structure is created using Page Object Model.

a. appModules: Contains the Modular action classes for each modular functionality of the application which can be reused.
b. automation Framework: Contains the customized classes designed for each sprint, utilizing page objects and modular classes.

c. page Objects: Object repository is made for the project for entire application. This makes adding or deleting of new elements into the application in an easy way.

Step 2 - Constant elements that need to run in the project are declared in the Constant class. Constant elements in the application are declared single time at one place.

Step 3 - TestNG will allow the code re-utilization efficiently. All the required classes that needs to run for a sprint or regression can be run in one go. Also, multiple regression suites can be run at a time and multiple testNG files also can be added to the application.

Step 4 – Select the testNG.xml file, add required classes and methods that needs to run and click run from Eclipse menu.

Step 5 - Automation tests will run. The results are found in the test-output folder and on Console.