Software Architecture Specification

for

FOES Data Management System

Version 1.0

Prepared by

Lee Chian Hui, Ng Jing Ru, James Wong Siew Huei, Yeum Kyu Seok

Curtin University Malaysia

26 April 2022

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
All Members	26/4/2022	Initial Draft	1.0

1. Introduction

Contributor(s): Kyu Seok yeum

1.1 Purpose of this Document

This document is to guide with the architectural overview of FOES Data Management System project.

The target audience includes engineers involved in developing the project, any manager responsible for software maintenance. Anyone interested in understanding the architectural overview of FOES Data management project should read this document.

1.2 Document Scope

This document specifies the technical stacks and structural communications within the components. All the high-level architectural structure used in the project will be presented, and major components will be explained in detail.

1.3 Related Documents

Document Name	Source
Laravel Official Document	https://laravel.com/docs/8.x/readme
MongoDB Official Document	https://www.mongodb.com/docs/
HTTPS Transfer Protocol Document	https://www.w3.org/Protocols/rfc2616/rfc2616.html
Blade Template Document	https://laravel.com/docs/9.x/blade
Laravel Analytics Document	https://github.com/spatie/laravel-analytics

2. Architecture Overview

Contributor(s): Kyu Seok yeum

2.1 Identification

This document explains the software architecture of the FOES Database Management System, and the high-level overview of the project to help with the development and maintenance.

2.2 Goals and Objectives

The goal of this document is to provide a clear description with detailed information of software architecture, communications between the components, and the interfaces that the project implements. This document will explain the high-level design of the APIs and not contain the exact logics of the APIs.

2.2.1 Compatibility and Interoperability

The architecture uses industry standard interfaces for communications between the forming modules and distributed layers. By implementing industry standard interfaces, the software is expected to be compatible with the third-party software. For example, the application implements W3C web standard body, for the network transactions of any HTTPS request, which makes the software accessible regardless of OS and device on web browsers. ACID will be implemented for database transactions to guarantee data validity on error, power failures, and other mishaps.

2.2.2 Extensibility

The architecture is designed with extensibility features by using Laravel framework. The architecture may extend any functionality of application, backend logic, and addition of the third-party software into the project.
This allows the customization of applications in a particular user's needs and requests.

2.2.3 Portability and Platform Independence

The architecture is designed portable regardless of OS for both Windows and Linux and independent of specific deployment environments on either cloud computing or local hosting. The application runs on top of a web browser, which is platform independent on any device.

3. Architecture Model

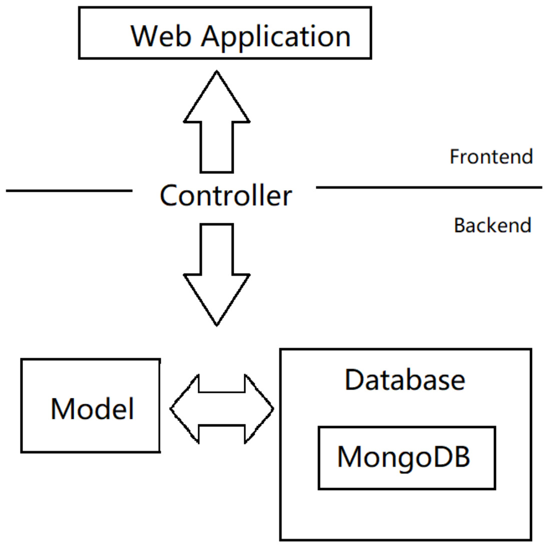
Contributor(s): Jing Ru Ng

The architectural model used to describe the architecture utilizes functional block diagrams.

3.1 Architectural Concepts and Representations

In the functional block diagram, the frontend of the application is represented at the top of the diagram while the backend of the application is shown at the bottom of the diagram. Type of the database is displayed in a separate box under the database box.

3.2 Architecture Block Diagram



3.3 Architecture Block Diagram Description

Web Application - Interfaces that user interacts with the data information.

Database - Stores the information that is sent from the web application by user and fetches the information requested by the user back to the application.

4. Architectural Specification - Home Screen

4.1 Architectural Specification - User Account

Contributor(s): Kyu Seok yeum

4.1.1 Introduction

This section will discuss how the user will create the admin for the database management system.

4.1.2 Component View

When a user enters Admin Dashboard from the navigation bar, The application displays the table list of the admin in the system. The table consists of the information of ID, Name, Department, Office.

From here, the user may add a new admin by clicking on "Create Admin" button on the top right and load the create admin page with input fields. After the user inputs sufficient data and presses the "Create" button to create a new admin, the page will be redirected to the Admin Dashboard, and the list will contain the newly created admin.

To update any information of the admin, the user may click on the update button under the action column, and the edit admin page will be loaded. The user may change the data in the column, and by pressing the "Edit" button after right input is made, the application will update the edited data of an admin.

For deletion of an admin, the user may click on the delete button under the action column. Once the button is clicked, the application will prompt a confirmation message to proceed with the deletion. If the user wants to proceed and click the "Yes" button, the data of the admin will be deleted from the database.

The user may also search for admin using the search bar on the top right

4.1.3 Theory of Operations

The application only proceeds with the Admin Dashboard if the logged in user is a super user. If a super user is verified, the application loads the list of admins in the system by getting a request to the database.

Before any writing request is made from the frontend to the backend, validation will be done. The validity check message will be prompted to the users on create and update operations, to check if the input data consists of sufficient information with the correct type. The application will carry out the request to the database only if the input data is valid.

For deletion of an admin, the application will delete a admin data of matching ID from the database once the request is made from the frontend.

If the filter entered the search bar meets any results in the database, the result will be shown on the dashboard.

4.2 Architectural Specification - Staff Information

Contributor(s): James Siew Huei Wong

4.2.1 Introduction

This section will discuss how the staff information such as their details are stored and accessed.

4.2.2 Component View

When the user enters the dashboard of the database management system, they will be shown the list of tables, be it admin details (if they are a super admin), employee details or asset details. To access the staff information, they can navigate to the employee tab on the navigation bar on the left under the dashboard category.

Once there, the user can either add a new employee by clicking on the button on the top right called “New Employee Entry”, and the user will be brought to a new page where they can enter the details of the employee.

After the details of the employee have been entered, the user can click the “Create” button to save it into the database.

To update the employee’s details, the user can click on the edit button under the edit column, and they will be brought to a page similar to the page where they add a new employee, but with the data of that employee prefilled. After the new data has been entered, they can press the “Edit” button and it will be saved into the database.

For deleting an employee, the user can click on the delete button under the action column. Once that button is clicked, the application will prompt an alert message to make sure of the user’s action. If they are sure of what they are doing, they can click on the “Yes” button on the alert message, and the data of the employee will be deleted from the database.

They can also search for the employee by using the search bar on the top right.

4.2.3 Theory of Operations

As the details of the employee are passed into the backend, validation will be done to make sure that all the details that have been entered by the user are correct and do not contain any characters that are not specified to prevent SQL injection attack.

If one of the information is not correct / validation is not true, an error message will be prompted under that field which contains the error, and the user would have to make amendments to that field. If this is not done, nothing will be saved into the database.

Similar to creating a new employee, updating an existing employee’s information works with the same principle. The newly entered details will first be undergoing a validation check before being saved into the database. Once it has been validated, if it is true, it will be saved into the database, else it will prompt an error message under the field that contains the error.

For deleting an existing employee's information, it will make sure that the user has clicked yes on the alert message prompt on the application, and once it has been clicked, the system will proceed with the deletion of the employee.

If the filter entered the search bar meets the results stored in the database, it will be shown to the user on the dashboard. It will search for the related information, for example, by typing "a", any name containing the letter "a" will be returned. In the case where the user is trying to perform a SQL injection attack, the filter entered will be first filtered in the backend to check for any special characters being used. An error message will be shown if such a condition is met.

4.3 Architectural Specification – Asset Information

Contributor(s): Jing Ru Ng

4.3.1 Introduction

This section will describe how the user accesses, creates, edits, deletes and searches the asset information.

4.3.2 Component View

There will be a "Dashboard" tab under the navigation bar, as the user clicks on the tab, "Asset Dashboard" will be shown below the tab. When the user selects the "Asset Dashboard", the list of the asset table will be shown at the page view. The user can view, create, edit, delete and search the asset information under the table "Asset Dashboard".

The user can create a new asset entry by clicking the "Create Asset" button and it will lead the user to a new view to enquire the user to enter the data information of the asset. The user can also edit or delete the asset data entry by clicking on the buttons beside the selected entry.

The user can search the asset information by entering the search term into the search box. The view will only display the search result that requests from the user.

4.3.3 Theory of Operations

When the user clicks on the "Asset Dashboard", the system will return the list of asset data entries and display them on the view.

The system will perform validation checking while the user creates a new asset entry and edits an existing asset entry. If the action performed is validated as successful, the system will update the information in the database and update it at the view, else the system will prompt an error message at the failure section.

For the delete button, the system will prompt a window to ask confirmation on deleting the data to the user.

As the user enters a search term in the search box, the system will return and display the data information according to the given search term. If the result doesn’t exist, it will prompt a related error message on the view.

4.4 Architectural Specification - Outreach and Community Engagement Activities

Contributor(s): James Siew Huei Wong

4.4.1 Introduction

This section will discuss how the outreach and community engagement activities would be stored and accessed.

4.4.2 Component View

As the user clicks on the “Activities” tab under the navigation bar, they will be brought to the dashboard of the activities. For adding a new activity, the user can click on the button “Add Activity” on the top right, and they will be brought to the page where they can enter the details of the new activity. Once finished, the user can click on the “Add Event” button to store it into the database.

For editing an existing activity, the user can click on the edit button on the action column, and they will be brought to a page similar to the add activity page, but with the existing data prefilled. After changes are made, the user can click on the “Edit” button and reflect the changes to the database.

Same as to the edit of an activity, the user can click on the delete button on the action column to delete an existing activity. Once the user has clicked on the delete button, an alert message will appear on top of the screen to make sure that the user is aware of their actions. After “Yes” is pressed in the alert box, it will be deleted.

As to searching for a specific activity, the user can enter the keyword into the search bar and the related result will be shown.

4.4.3 Theory of Operations

All the data entered by the user, be it create, update or search, will be validated when the data is sent to the backend. If any of the data is proved to be wrong or does not match the validation based on the condition set, an error message will be shown under that specific field.

Creating a new activity will only work after the validation has been completed, and no error was found, then only the data will be stored into the database.

Updating an existing activity works similarly with the creation of a new activity. The newly entered details will be validated, and once it is true, it will be updated in the database.

For deleting an existing activity, it will make sure that the user has clicked yes on the alert message prompt on the application, and once it has been clicked, the system will proceed with the deletion of the activity.

Since the keyword entered in the search bar would be validated, SQL injection attack would be minimized. Only the related results would be shown to the user. For example, the keyword “school” will show the results containing the word “school”, be it in front, in between or at the end of the name of the activity.

4.5 Architectural Specification - Generation of Report

Contributor(s): Chian Hui Lee

4.5.1 Introduction

This section will discuss how users generate reports of different tables from the database management system.

4.5.2 Component View

There will be a “Data Export” button on the sidebar. Users can click on the button, it will lead to the Data Export Page.

In the Data Export Page, users can select the data he/she wants to export. After selecting the tables, user can select a format to export the file, such as PDF or CSV. After that, users can click on the “Preview” button to preview what the exported PDF/CSV looks like, or click the “Export” button to have the PDF/CSV download on the user’s device. User can also export all of the table by clicking on the “Export All” button.

4.5.3 Theory of Operations

After the user selects the data he/she wants to export and clicks either the “Preview” or “Export” button, the system will perform data validation checking. If the user does not check any of the checkboxes, the “Preview” and the “Export” button will be disabled, which means that the user will not be able to click on the button.

If the user checks one of the checkboxes and clicks on the “Preview” button, the system will generate a preview of the PDF/CSV, containing the data of the selected table.

If the user selects a table and clicks on the “Export” button, the system will have the data of the selected table generated into PDF/CSV and download it on the user’s device.

4.6 Architectural Specification - Import of Data Automatically

Contributor(s): Chian Hui Lee

4.6.1 Introduction

This section will discuss how users import the data of the staff automatically.

4.6.2 Component View

There will be a “Data Import” button on the sidebar. Users can click on the button, it will lead to the Data import page.

In the data import page, the user can select a table they want to import data into, and select the CSV file from their device by clicking on the “Select” button. The selected file’s name will appear under the “Select” button for the user to double confirm it is the correct file.

After that, the user can click on the “Upload” button to upload the file and import the data into the particular table in the database.

4.6.3 Theory of Operations

The system will perform file checking on the selected file. If the format does not match, or the user does not select a file, the “Upload” button will not be enabled.

If the user selects a valid file, the system will read the filename and display it under the “Select” button for the user’s double-checking purpose. The system will read the content of the file and perform data validation to prevent injection. If there is nothing malicious, the system will save the data from the CSV file into the database.

4.7 Architectural Specification - Table Customization

Contributor(s): Jing Ru Ng

4.7.1 Introduction

This section will describe how users perform customization on the selected table.

4.7.2 Component View

There will be a “Table Customization” tab under the navigation bar, as the user clicks on the tab, the list of the existing tables will be shown below. The user can select the table that he or she wants to be edited or deleted.

After selecting a table, the current column of the table will be displayed on the view. There will be a “Add New” button that leads the user to a new view that requires the user to enter the column name that he or she wants to add and a “Add” button to create the column.

4.7.3 Theory of Operations

When the user clicks on the edit button, it allows the user to edit the column name. The system will perform validation checking for the edited column name and update the view. For the delete button, rather than deleting all data information of the column, it allows the user to disable the data view of the selected column.

When the user clicks on the “Add” button in the add column page, the system will perform validation checking on the input data. The action will be failed when the submitting text is empty. The column view will be updated when the system has successfully added a new column into the selected table.

4.8 Architectural Specification - Dashboard View Customization

Contributor(s): James Wong Siew Huei

4.8.1 Introduction

This section will describe how the user can perform customization on the view of the dashboard.

4.8.2 Component View

In each of the dashboards, there will be a button called “Customize View” on the top right of the page. Once the user has clicked on that button, a dropdown of checkboxes would be displayed, and the user can select which columns they would like to show on the dashboard. Once the selection has been made, they can click on the “Apply Changes” button to save the changes.

4.8.3 Theory of Operations

When the user clicks on the “Apply Changes” button, the backend would check what columns have been selected. Those columns would have a variable whereby it would keep track if it has to be shown on the dashboard or not. If it is true, that column will be shown, and vice versa.

4.9 Architectural Specification - Database Backup & Restore

Contributor(s): Kyu Seok yeum

4.9.1 Introduction

This section will discuss how the backup feature of the database will be carried out by the system.

4.9.2 Component View

Under the Database Backup & Restore view, there are “Backup” and “Restore” buttons. By clicking the “Backup” button, the user may save the current state of the database system as a backup. The “restore” button provides the restore feature of the backup database file to the system, so that the user may always restore the previous state of the database system. The drop down bar in the database restore feature provides the options of selecting the backup file to restore.

4.9.3 Theory of Operations

On the backup request by the user, the system will store all the state of the database into a bson file in the local system. The file will contain all the data, database collections, and the date time of the backup file. All the process will be done internally with the mongoDB commands which will be embedded in the application.

The system will restore the selected file of the database on the restore request by the user. On successful restore, the system will contain all the data and the collections with the previous state of the database.

4.10 Architectural Specification - Mobility

Contributor(s): Kyu Seok yeum

4.10.1 Introduction

This section will discuss how the staff and student mobility data will be stored and accessed

4.10.2 Component View

When the mobility tab is clicked under activities, “Mobility” dashboard will be displayed in the application. The dashboard will contain a table with the list of mobility data and the user may view, create, edit, delete or search the mobility information in the dashboard.

The user may create a new mobility entry by clicking the “Create Activity” button which will lead the user to the data input form page where the user may input a new entry. The user may also edit or delete the mobility data entry by clicking on the buttons beside the selected entry.

The user may search the mobility information by entering the search term into the search box. The view will only display the search result that requests from the user.

4.10.3 Theory of Operations

When the user clicks on the “Mobility” dashboard, the system will return the list of mobility data entries and display them on the view.

The system will perform validation checking while the user creates a new mobility entry and edits an existing mobility entry. If the action performed is validated as successful, the system will update the information in the database and update it at the view, else the system will prompt an error message at the failure section.
For the delete button, the system will prompt a window to ask confirmation on deleting the data to the user.

As the user enters a search term in the search box, the system will return and display the data information according to the given search term. If the result does not exist, it will prompt a related error message on the view.

4.11 Architectural Specification - Research Awards Table

Contributor(s): Jing Ru Ng

4.4.4 Introduction

This section will describe how the user accesses, creates, edits, deletes and searches the research awards information.

4.4.5 Component View

There will be a “Dashboard” tab under the navigation bar, as the user clicks on the tab, “Research Awards Dashboard” will be shown below the tab. When the user selects the “Research Awards Dashboard”, the list of the research awards table will be shown at the page view. The user can view, create, edit, delete and search the award information under the table “Research Awards Dashboard”.

The user can create a new award entry by clicking the “Create Research Awards” button and it will lead the user to a new view to enquire the user to enter the data information of the award. The awards data will also be shown in the selected staff data from the Staff Dashboard. The user can also edit or delete the award data entry by clicking on the buttons beside the selected entry.

The user can search the awards information by entering the search term into the search box. The view will only display the search result that requests from the user.

4.4.6 Theory of Operations

When the user clicks on the “Research Awards Dashboard”, the system will return the list of award data entries and display them on the view.

The system will perform validation checking while the user creates a new award entry and edits an existing award entry. If the action performed is validated as successful, the system will update the information in the database and update it at the view, else the system will prompt an error message at the failure section.

For the delete button, the system will prompt a window to ask confirmation on deleting the data to the user.

As the user enters a search term in the search box, the system will return and display the data information according to the given search term. If the result doesn’t exist, it will prompt a related error message on the view.

4.12 Architectural Specification - MOU & MOA Information Table

Contributor(s): Jing Ru Ng

4.4.7 Introduction

This section will describe how the user accesses, creates, edits, deletes and searches the MOU and MOA information.

4.4.8 Component View

There will be a “Activities” tab under the navigation bar, as the user clicks on the tab, “MOU/MOA Information” will be shown below the tab. Under the tab, there will be two sub-tab for it, which are “Current List” and “Inactive List”.

When the user selects the “Current List”, the current list in both MOU & MOA categories will be shown at the page view. When the user selects the “Inactive List”, the inactive list in both MOU & MOA categories will be shown at the page view. The user can view, create, edit, delete and search the list information under the table “MOU/MOA Information”.

The user can create a new award entry by clicking the “Create Activity” button and it will lead the user to a new view to enquire the user to enter the data information. The user can select the program category of “MOU” or “MOA” in the create view. The user can also edit or delete the program data entry by clicking on the buttons beside the selected entry.

The user can search the program information by entering the search term into the search box. The view will only display the search result that requests from the user.

4.4.9 Theory of Operations

When the user clicks on the tab “Current List” and “Inactive List” which are under “MOU/MOA Information”, the system will return the list of information data entries and display them on the view.

The system will perform validation checking while the user creates a new information list entry and edits an existing information list entry. If the action performed is validated as successful, the system will update the information in the database and update it at the view, else the system will prompt an error message at the failure section. The system will detect the program category of the entry and store the information accordingly in the database.

For the delete button, the system will prompt a window to ask confirmation on deleting the data to the user.

As the user enters a search term in the search box, the system will return and display the data information according to the given search term. If the result doesn’t exist, it will prompt a related error message on the view.

5. Architectural Specification - Backend

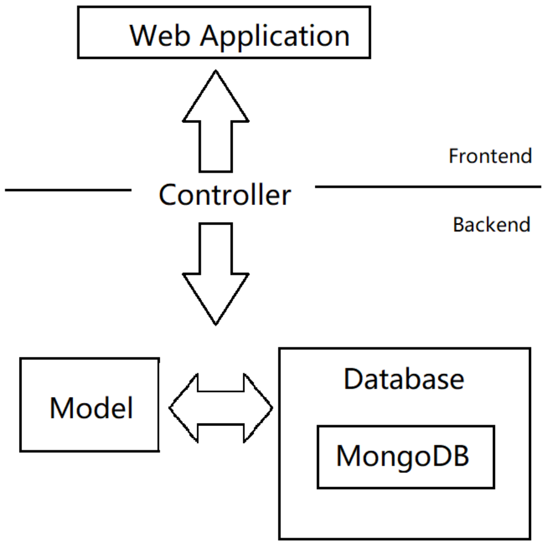
Contributor(s): Kyu Seok yeum

5.5 Introduction

This section describes the architectural specifications of the backend

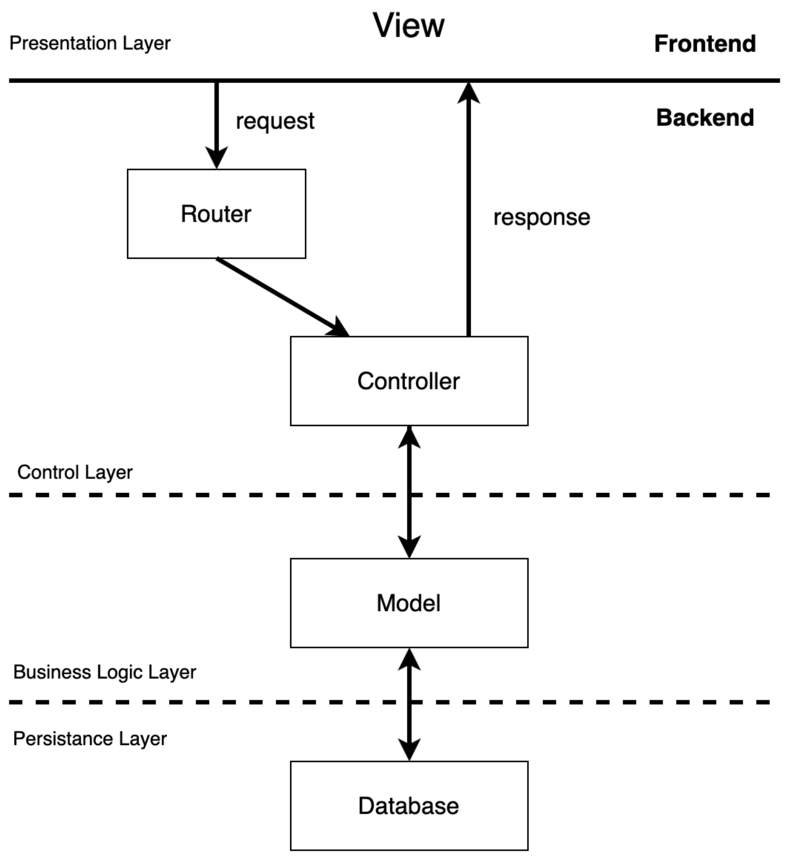
5.6 System Structural Overview

5.6.1 Relationship between Backend and the Web Application



The application implements MVC architectural pattern using Laravel Framework. The architecture is divided into three main parts, Model, View, Controller which separates the layers of business logic and the presentation layer. Controller and the Model lie under the backend system, which Model stores data and the database related logic, and the Controller contains interfaces between the model and view.

5.6.2 Software Component Overview



The backend architecture can be divided into 3 layers, Control, Business Logic, and Persistence. By implementing separate layers in the backend system, the development process is faster, easier to maintain and prone to error as End-to-End testing is possible.

5.7 Theory of Operation

When there is any request from the frontend application to the backend, the requests will go through the routing table in the router. If there is no matching route, the backend will return an exception to the front, and if there is a matching route, the middleware will filter http requests to the controller. Depending on what the request is in http, the controller executes the declared function and returns new components to the view, or controller may communicate with the model for CRUD operation to the database. After the request is handled completely, the controller returns the http response back to the frontend with updated UI components.

6. Architectural Specification - Database

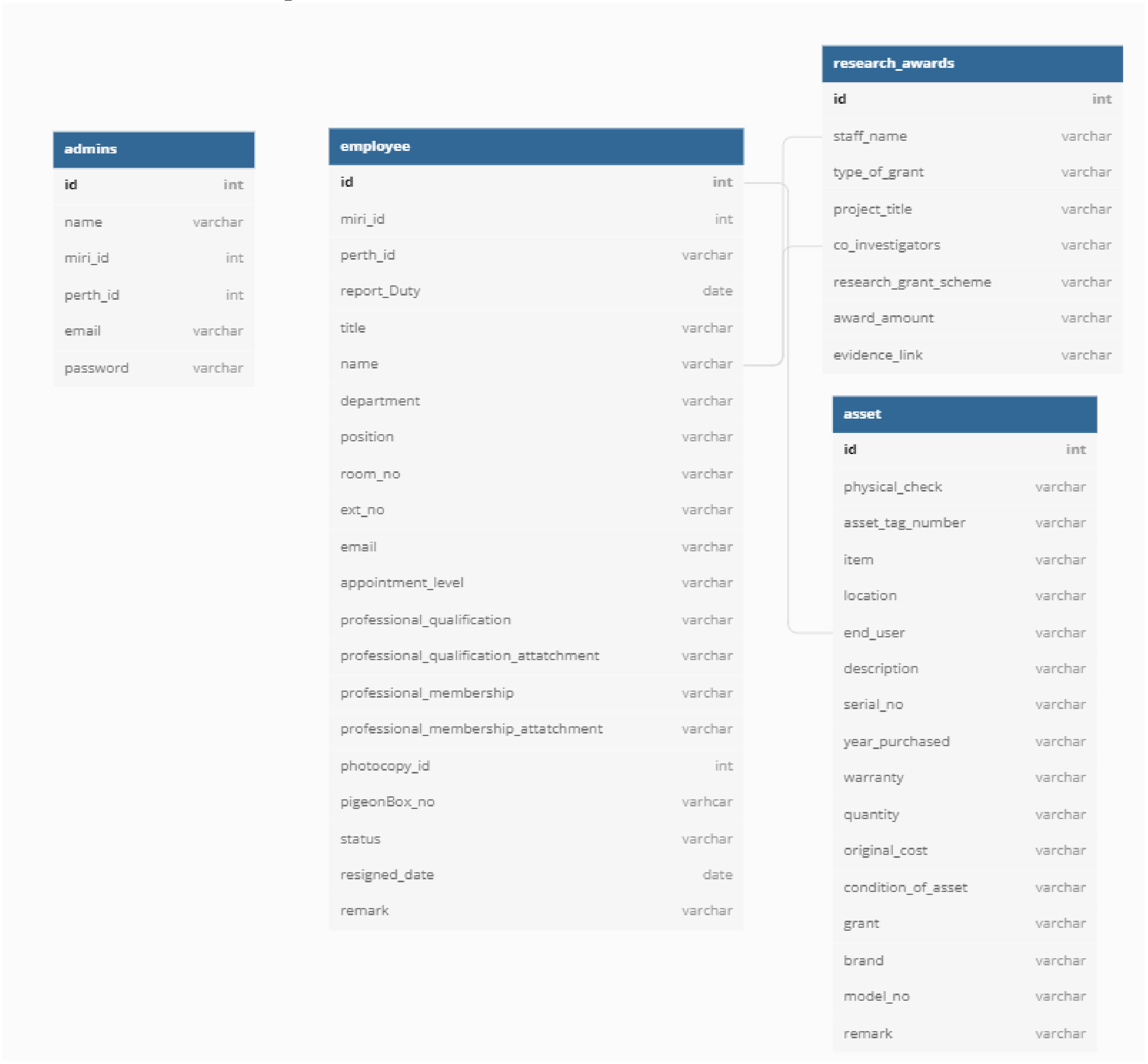
Contributor(s): Chian Hui Lee

6.5 Introduction

In this section, the schema of the database will be defined.

6.6 System Structural Overview

6.6.1 Software Component Overview

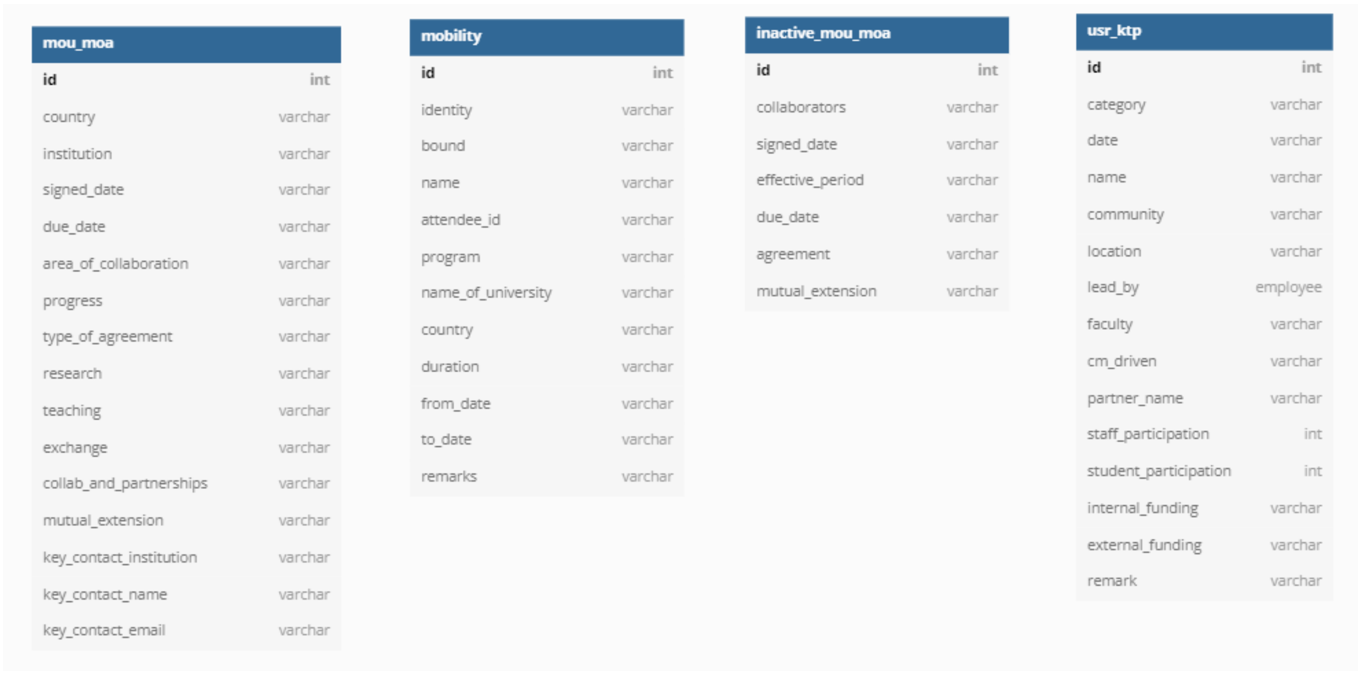


admins	
id	int
name	varchar
miri_id	int
perth_id	int
email	varchar
password	varchar

employee	
id	int
miri_id	int
perth_id	varchar
report_Duty	date
title	varchar
name	varchar
department	varchar
position	varchar
room_no	varchar
ext_no	varchar
email	varchar
appointment_level	varchar
professional_qualification	varchar
professional_qualification_attatchment	varchar
professional_membership	varchar
professional_membership_attatchment	varchar
photocopy_id	int
pigeonBox_no	varhcar
status	varchar
resigned_date	date
remark	varchar

research_awards	
id	int
staff_name	varchar
type_of_grant	varchar
project_title	varchar
co_investigators	varchar
research_grant_scheme	varchar
award_amount	varchar
evidence_link	varchar

asset	
id	int
physical_check	varchar
asset_tag_number	varchar
item	varchar
location	varchar
end_user	varchar
description	varchar
serial_no	varchar
year_purchased	varchar
warranty	varchar
quantity	varchar
original_cost	varchar
condition_of_asset	varchar
grant	varchar
brand	varchar
model_no	varchar
remark	varchar



mou_moa	
id	int
country	varchar
institution	varchar
signed_date	varchar
due_date	varchar
area_of_collaboration	varchar
progress	varchar
type_of_agreement	varchar
research	varchar
teaching	varchar
exchange	varchar
collab_and_partnerships	varchar
mutual_extension	varchar
key_contact_institution	varchar
key_contact_name	varchar
key_contact_email	varchar

mobility	
id	int
identity	varchar
bound	varchar
name	varchar
attendee_id	varchar
program	varchar
name_of_university	varchar
country	varchar
duration	varchar
from_date	varchar
to_date	varchar
remarks	varchar

inactive_mou_moa	
id	int
collaborators	varchar
signed_date	varchar
effective_period	varchar
due_date	varchar
agreement	varchar
mutual_extension	varchar

usr_ktp	
id	int
category	varchar
date	varchar
name	varchar
community	varchar
location	varchar
lead_by	employee
faculty	varchar
cm_driven	varchar
partner_name	varchar
staff_participation	int
student_participation	int
internal_funding	varchar
external_funding	varchar
remark	varchar

7. Debugging, Error Logging, Performance Monitoring

Contributor(s): James Siew Huei Wong

7.5 Instrumentation

7.5.1 Performance Monitoring

Since Laravel does not have built-in performance monitoring capabilities, a third-party library could be used, which uses Google Analytics called Laravel Analytics (spatie, n.d.).

The developer console from different browsers works well too to assist in detecting which assets take the longest amount of time to load, in turn helps the developer to find out the issue and fix it before the deployment of the project.

7.6 Debug

Laravel provides a debug mode during the development. Once it is enabled in the environment file, errors would be displayed once the web application crashes unexpectedly on the web page itself. It can be used to trace back to where the error occurs, and which variable causes that error.

Basic code errors could also be debugged, as IDE (integrated development environment) such as PHPStorm would be used in this project. Since an IDE is being used, it will automatically detect errors while writing the code, such as any syntax errors and typing errors will be displayed as warning and red lines under the specific codes.

Methods such as dd() (Dump and Die) is also a helpful method used for debugging purposes, as it will display the data being displayed, and it will kill the page, stopping it from loading further.

Third party libraries such as Laravel Debugger (barryvdh, n.d.) would be of great help in this case as well, as it includes the basic things for logging and debugging.

7.7 Error logging

Laravel has a built-in logging tool, and same goes to the debug part, it can be turned on in the environment file. It has multiple variables whereby you can decide how long to keep the log files for. There are a lot more variables which can be altered based on the needs for the applications, but for FOES Database Management System, a simple daily logging would be sufficient.

Reference

barryvdh. n.d. “barryvdh/laravel-debugbar: Laravel Debugbar (Integrates PHP Debug Bar).” GitHub. Accessed May 3, 2022. https://github.com/barryvdh/laravel-debugbar.

Laravel. n.d. “Actions Handled By Resource Controller.” Laravel. Accessed May 6, 2022. https://laravel.com/docs/9.x/controllers#actions-handled-by-resource-controller.

spatie. n.d. “spatie/laravel-analytics: A Laravel package to retrieve pageviews and other data from Google Analytics.” GitHub. Accessed May 3, 2022. https://github.com/spatie/laravel-analytics.

Dependency Matrix

Contributor(s): Jing Ru Ng

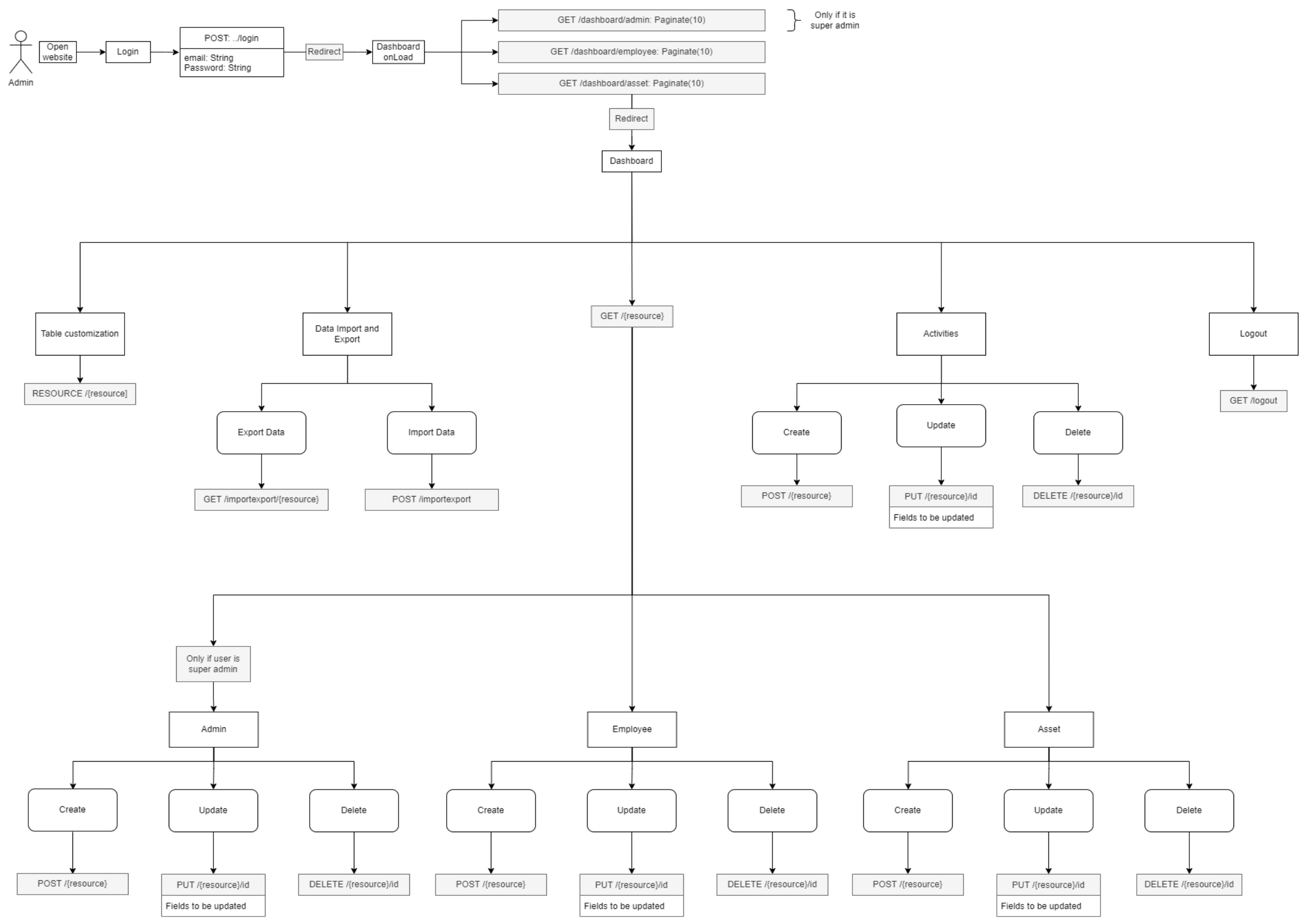
	Home Screen	Admin Dashboard	Employee Dashboard	Asset Dashboard	Research Awards Dashboard	MOU/MOA Program - Current List	MOU/MOA Program - Inactive List	KTP/USR Program Activities	Mobility Program - Staff Inbound	Mobility Program - Staff Outbound	Mobility Program - Student Inbound	Mobility Program - Student Outbound	Data Resource*	Dashboard View Customization	Data Import	Data Export	Table Customization	Data Backup & Restore
Home Screen	▨	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			✓	✓		✓
Admin Dashboard	✓	▨											✓	✓			✓	
Employee Dashboard	✓		▨										✓	✓			✓	
Asset Dashboard	✓			▨									✓	✓			✓	
Research Awards Dashboard	✓				▨								✓	✓			✓	
MOU/MOA Program - Current List	✓					▨							✓	✓			✓	
MOU/MOA Program - Inactive List	✓						▨						✓	✓			✓	
KTP/USR Program Activities	✓							▨					✓	✓			✓	
Mobility Program - Staff Inbound	✓								▨				✓	✓			✓	
Mobility Program - Staff Outbound	✓									▨			✓	✓				
Mobility Program - Student Inbound	✓										▨		✓	✓				
Mobility Program - Student Outbound	✓											▨	✓	✓				
Data Resource*		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	▨					
Dashboard View Customization		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		▨				
Data Import	✓														▨			
Data Export	✓															▨		
Table Customization		✓	✓	✓	✓	✓	✓	✓	✓			✓					▨	
Data Backup & Restore	✓																	▨

The dependency matrix above is based on the dependency between the model views which are the functional requirement of the application. By going through the rows, for example, admin dashboard, employee dashboard, activities program, data import & export and table customization are dependent on the home screen, hence the checkmark.

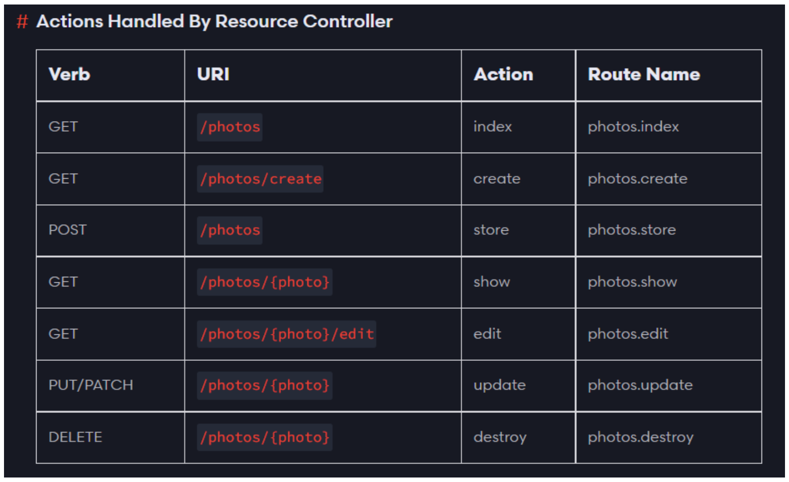
Data Resource* means create, read, update, delete and search operation for the data.

Component Interaction Design

Contributor(s): James Siew Huei Wong



The RESOURCE HTML method under the table customization is a method used by Laravel, which consists of all the HTML methods such as GET, POST, PUT and more (Laravel, n.d.). The image below shows all the methods being handled by RESOURCE:


Actions Handled By Resource Controller

Verb	URI	Action	Route Name
GET	`/photos`	index	photos.index
GET	`/photos/create`	create	photos.create
POST	`/photos`	store	photos.store
GET	`/photos/{photo}`	show	photos.show
GET	`/photos/{photo}/edit`	edit	photos.edit
PUT/PATCH	`/photos/{photo}`	update	photos.update
DELETE	`/photos/{photo}`	destroy	photos.destroy

Taken from https://laravel.com/docs/9.x/controllers#actions-handled-by-resource-controller