SDD Document
SEM II 20222023

e-Munakahat System (EMUN)

Group Name

1. Muhammad Amir Bin Mohamed Ali [CB21060]
2. Ahmad Suffian Bin Md Noor Suhaime [CB21137]
3. Chua Kian Pheng [CB21106]
4. Nik Alia Syafiqah Binti Nik Azuri [CB21035]
5. Shammene A/P Muneesvaran [CB21018]

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Purpose

This software design document (SDD) serves as a blueprint for the development of the e-Munakahat System (EMUN). It outlines the software design process, including the software's architecture, modules, interfaces, and data structures. The primary purpose of an SDD is to ensure that the software development team and stakeholders understand the design of the software application and how it will function.

The software design document (SDD) is a communication tool between the development team and stakeholders, ensuring everyone is on the same page regarding the software's design and implementation. It will also help the stakeholders to understand any limitations or assumptions that may impact the system's performance or usability, which will help the stakeholders to make informed decisions and better plan for the project. This will also help to ensure that the final product meets the needs of the users and the jurisdiction.

1.2 System Identification

The Software Design Document (SDD) belongs to the “e-Munakahat System” (EMUN).

Table 1.1 Document Identity.

<table>
<tr><th>System title</th><th>e-Munakahat System</th></tr>
<tr><td>System abbreviation</td><td>EMUN</td></tr>
<tr><td>System identification number</td><td>SDD_EMUN_QC_2023</td></tr>
<tr><td>Subsystem Title</td><td>IkatanCinta</td></tr>
<tr><td>Subsystem Abbreviation</td><td>IC</td></tr>
<tr><td>Package ID</td><td>SDD-REQ-100
Meanings for terms use:
<table>
<tr><td>SDD</td><td>Software Design Document</td></tr>
<tr><td>REQ</td><td>Requirement</td></tr>
<tr><td>100</td><td>Package number</td></tr>
</table></td></tr>
<tr><td>Use Case ID</td><td>UC101-EMUN-001
Meanings for terms used:
<table>
<tr><td>UC</td><td>Use Case</td></tr>
<tr><td>1</td><td>Number of the system module</td></tr>
<tr><td>01</td><td>Number of use case within a module in the subsystem</td></tr>
<tr><td>EMUN</td><td>e-Munakahat System (System name)</td></tr>
<tr><td>001</td><td>Document release number</td></tr>
</table></td></tr>
<tr><td>Requirement Traceability ID</td><td>UC101-EMUN-001
Meanings for terms used:
<table>
<tr><td>UC</td><td>Use Case</td></tr>
<tr><td>1</td><td>Number of the system module</td></tr>
<tr><td>01</td><td>Number of use case within a module in the system</td></tr>
<tr><td>EMUN</td><td>e-Munakahat System (System name)</td></tr>
<tr><td>001</td><td>Document release number</td></tr>
</table></td></tr>
<tr><td>Document Identification System</td><td>SDD_EMUN_IC_2023_V1.0</td></tr>
</table>

Symbol/Number	Meaning
EMUN	It represents the system name which is e-Munakahat System (EMUN)
SDD	It represents the software design document (SDD)
QC	It represents the company name which is Quantum Corp SDN. BHD.
2023	It represents the year where the system has been released.
V1.0	V represent the word “Version” of the system while 1.0 is the general version of the system. The number will increase by 0.1 if there any updates or bugs fixed.

1.3 System Overview

Our system is a web-based system to support the Pahang Wedding management system. The purpose of the system is to provide a solution for managing wedding registration and operations efficiently. Users can use this system for marriage registration and information retrieval. It also eases the government to collect granular data on weddings in Pahang and store the information for future use. This system is handled by the Quantum Corp company. This system contains various functionalities such as user registration, marriage application, marriage registration, marriage preparation course, proof of payment, request for marriage, marriage card, marriage consultation, and special incentives for the bride and groom.

There are five modules in this system which are:

1. Registration and handle user profile

The initial stage of using the system is applicant registration, which is required for applicants to enter the system. It contains private information such as an Identity Card number, phone number, email address, and password. Since registration is based on the user's IC number, a unique number with no duplicates, each user has just one account for the system. Once the applicant's registration is successful, they can log in to access the system's contents after completing the registration process. The admin can register new staff through the admin dashboard. Once registration for staff is successful, they will be granted access to the system. The system also allows users (applicants & staff) to modify their profile details to update their information. In case the applicant has forgotten their password, they will be able to reset their password. Apart from that, the admin can update staff and applicant details through the admin dashboard.

2. Attend the marriage preparation course with proof of payment and to request a marriage.

   After users register and login into the account, the users can proceed to the next step where they can apply for the marriage registration course. As a marriage registration course is an important step for the e-Munakahat system will update the name list of applicants. The system will show if the user has already paid for the course or not. So, then the staff can print the proof of payment and send the receipt to the applicants. The system shall allow the admin to manage the marriage preparation course details such as date, time, and where the course will be conducted. While the staff will handle the things that are related to the user or applicants. Approve the applications in the system and print the applicants' name list for the course for every session. If the applicant presents on the course day, the staff can approve their marriage course certificate. After the course ends, the applicants will be represented with the certificate. The staff can update the applicant's name from the system for obtaining the marriage course certificate. Next step for the applicant is request their marriage from E-Munakahat system.

3. An application to register a marriage within or outside the country, as well as a voluntary marriage, and to produce the marriage card or certificate with proof of payment.

   Marriage registration is split into two processes which are voluntary and authorized registration. User needs to pass a pre-marriage course to be able to register their marriage. Voluntary registration is for the older generation who live far away from town and did not register when they married. For authorized registration, the user must prepare various forms and information in which the template is provided in the system. After staff from JAIP has approved the marriage registration, it will notify the user so that they know their marriage registration has been approved.

4. Register for a marriage consultation with a service advisor.

   A consultation module is a module where the user can make an appointment to seek advice or to get a divorce. This module requires the users to enter their personal information and also their spouse information including their marriage information and upload related documents as proof. Staff can view the application made by the user and will decide whether to approve the appointment and responsible to assign consultant and slot to the consultation session. Staff are also responsible for managing the list of consultants and their information in this module. The staff can edit, add, and delete the consultants' information in the system.

5. Application for special incentive for the bride and groom

The marriage registration system features a special incentive module that enables users to apply for incentives, provided they meet the necessary prerequisites. This module allows individuals to input their personal information and upload necessary documents, making the process more streamlined. Additionally, the staff side of the platform allows for the review of applications and ensures that the process is efficient and user-friendly. The special incentive module in the marriage registration system also includes a notification feature that keeps applicants informed about the status of their applications. Once the staff reviews the application, the applicant will receive a notification, either confirming their approval or outlining the reasons for the denial. This ensures that applicants are aware of the status of their application in a timely and efficient manner."

1.4 Referenced Document

1. Azma, B. A., et al (2023) BCS2243 Final Assessment Question Sem II 20222023.

2. Amir.A, et al (2023) SOFTWARE REQUIREMENT SPECIFICATION (SRS).

3. Layered Architecture. (2021, November 11). Retrieved May 2, 2023, from https://www.baeldung.com/cs/layered-architecture.

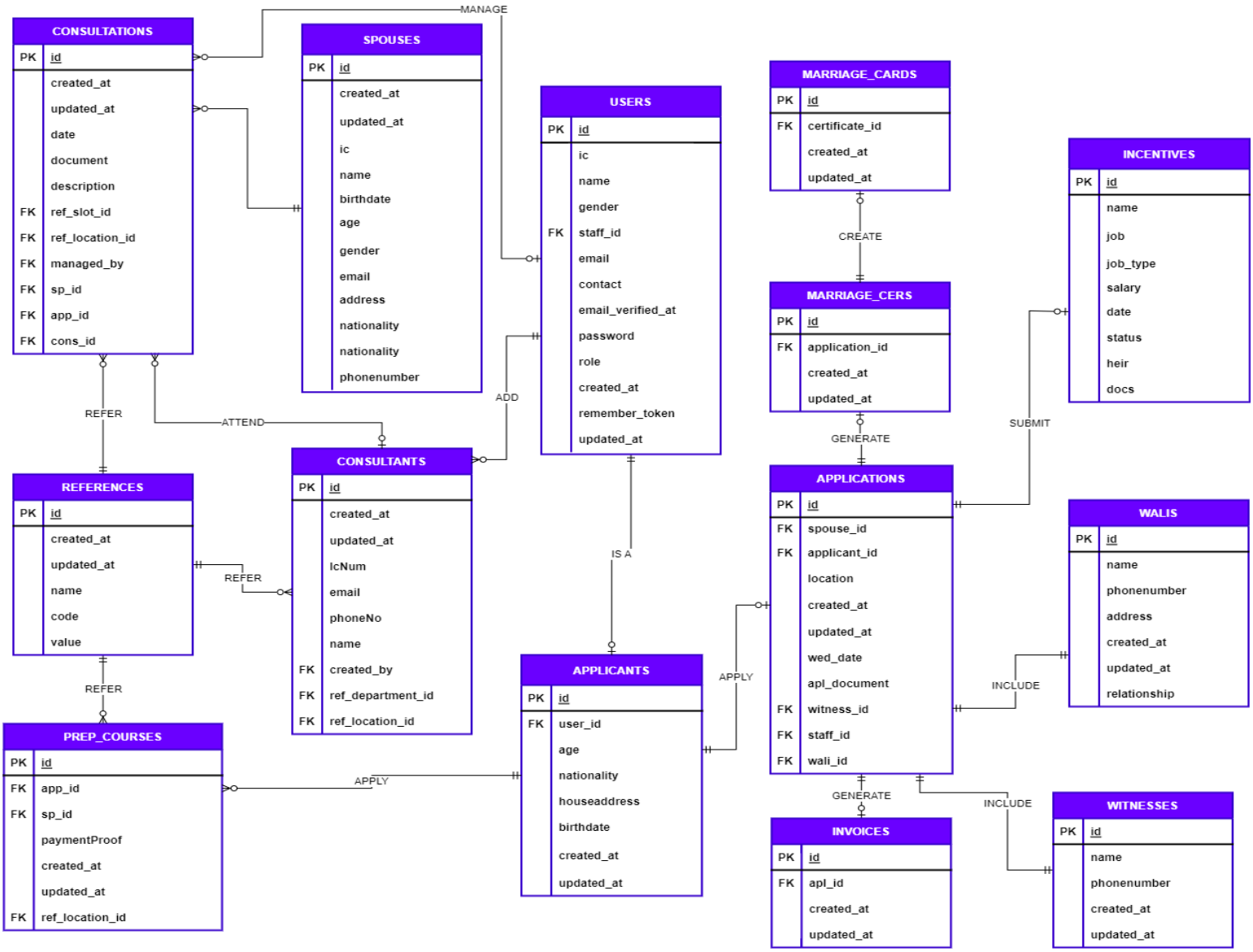
4. How to build a simple PHP MVC framework. (2021, May 30). Retrieved May 28, 2023, from https://www.giuseppemaccario.com/how-to-build-a-simple-php-mvc-framework/.

5. Simple PHP MVC Framework Example. (2023, May 26). Retrieved May 28, 2023, from https://phpflow.com/php/simple-mvc-architecture-example-in-php/.

6. What Is Middleware? Definition, Architecture, and Best Practices. (2022, February 18). Retrieved May 28, 2023, from https://www.spiceworks.com/tech/cloud/articles/what-is-middleware/.

2. DATA DESIGN

2.1 Entity Relationship Diagram (ERD)



2.2 Data Dictionary

2.2.1 USERS

Field Name	Description	Data Type	Constraint
id	User's ID	BIGINT	PK
ic	User's IC	VARCHAR (12)	NOT NULL, UNIQUE
name	User's name	VARCHAR (255)	NOT NULL
gender	User's gender	VARCHAR (8)	NOT NULL
staff_id	Staff's ID	VARCHAR (10)	
email	User's email	VARCHAR (255)	NOT NULL, UNIQUE
contact	User's contact	VARCHAR (15)	NOT NULL
email_verified_at	User's email verification dates	TIMESTAMP	NOT NULL
password	User's password	VARCHAR (255)	NOT NULL
role	User's role	TINYINT	NOT NULL
remember_token	Users remember token	VARCHAR (100)	
created_at	User's created date	TIMESTAMP	NOT NULL
updated_at	User's updated date	TIMESTAMP	

2.2.2 APPLICANTS

Field Name	Description	Data Type	Constraint
id	Applicant's ID	BIGINT(20)	PK
user_id	User ID	BIGINT(20)	Not null
age	Applicant's age	INT(10)	
nationality	Applicant's nationality	VARCHAR(255)	
houseaddress	Applicant's house address	VARCHAR (255)	
birthdate	Applicant's birthdate	DATE	
created_at	Applicant's created time	TIMESTAMP	Not null
updated_at	Applicant's updated time	TIMESTAMP	

2.2.3 CONSULTATIONS

Field Name	Description	Data Type	Constraint
id	Consultation ID number	BIGINT (20)	PK
created_id	Consultation created date	TIMESTAMP	NOT NULL
updated_at	Consultation update date	TIMESTAMP	
date	Consultation appointment date	DATE	NOT NULL
ref_slot_id	Consultation slot	BIGINT (20)	NOT NULL
description	Consultation description	TEXT	NOT NULL
document	Consultation document file path	VARCHAR (255)	NOT NULL
ref_location_id	Consultation location	BIGINT (20)	NOT NULL
managed_by	Staff who managed the consultation	BIGINT (20)	
sp_id	Spouse ID	BIGINT (20)	NOT NULL
app_id	Applicant ID	BIGINT (20)	NOT NULL
cons_id	Consultant ID	BIGINT (20)	

2.2.4 CONSULTANTS

Field Name	Description	Data Type	Constraint
id	Consultation's id	BIGINT (20)	PK
created_at	Consultation's created time	TIMESTAMP	NOT NULL
updated_at	Consultation 's updated time	TIMESTAMP	
IcNum	Consultant's IC number	VARCHAR (255)	NOT NULL
name	Consultant's name	VARCHAR (255)	NOT NULL
email	Consultant's email address	VARCHAR (255)	NOT NULL
ref_department_id	Consultant's departments	BIGINT (20)	FK, NOT NULL
ref_location_id	Consultant's working location	BIGINT (20)	FK, NOT NULL
phoneNo	Consultant's phone number	VARCHAR (255)	NOT NULL
created_by	Staff who register the consultant	BIGINT (20)	FK, NOT NULL

2.2.5 SPOUSES

Field Name	Description	Data Type	Constraint

id	Spouse's ID	BIGINT (255)	PK
created_at	Spouse's created time	TIMESTAMP	NOT NULL
updated_at	Spouse's updated times	TIMESTAMP	
ic	Spouse's IC	VARCHAR (12)	NOT NULL
name	Spouse's name	VARCHAR (255)	NOT NULL
birthdate	Spouse's birthdate	DATE	NOT NULL
age	Spouse's age	INT	NOT NULL
gender	Spouse's gender	VARCHAR (8)	NOT NULL
nationality	Spouse's nationality	VARCHAR (15)	NOT NULL
address	Spouse's address	VARCHAR (255)	NOT NULL
email	Spouse's email address	VARCHAR (255)	NOT NULL
phonenumber	Spouse's phone number	VARCHAR (15)	NOT NULL

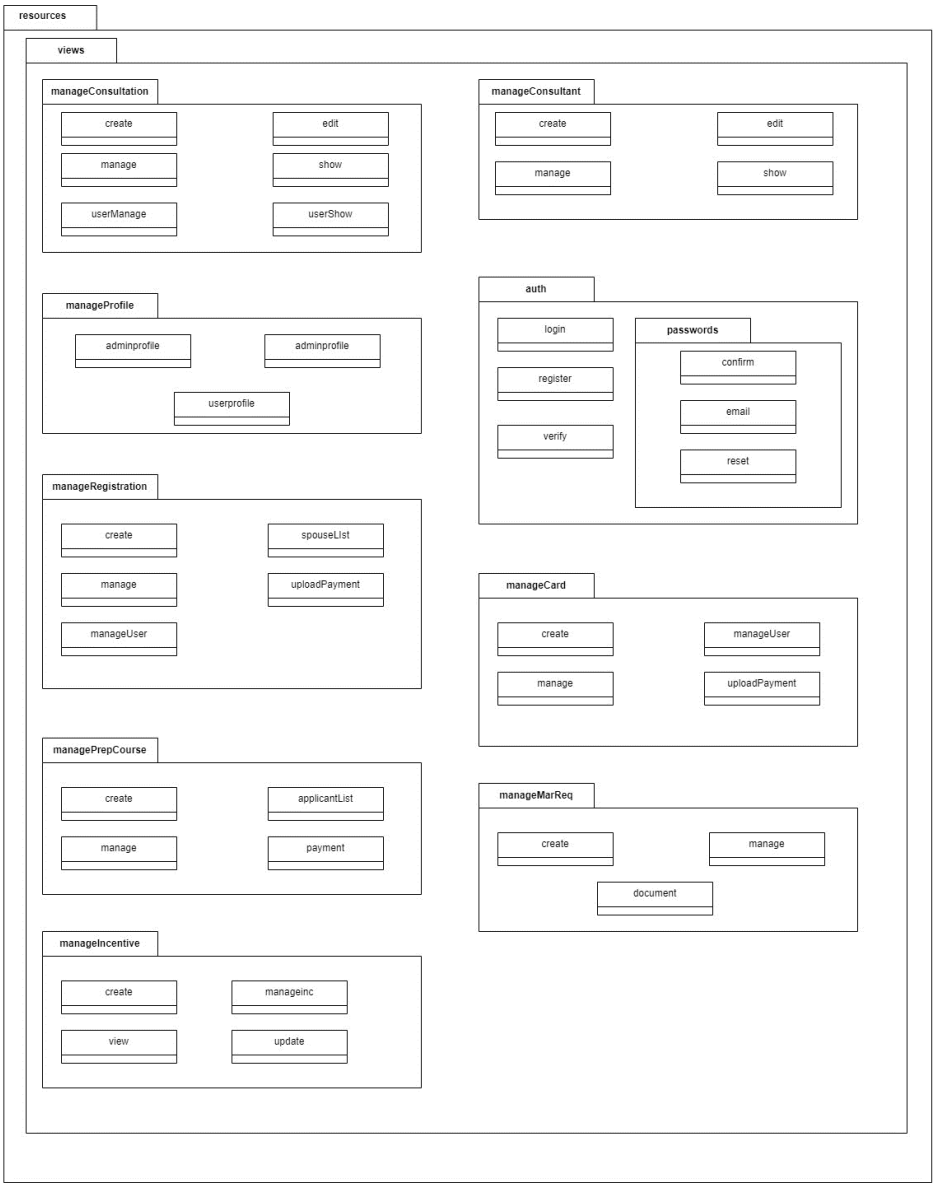
2.2.6 REFERENCES

Field Name	Description	Data Type	Constraint
id	References id	BIGINT (20)	PK
created_at	References created date	TIMESTAMP	
updated_at	References updated date	TIMESTAMP	
name	References name	VARCHAR (255)	NOT NULL
code	References code	VARCHAR (255)	NOT NULL
value	References value	VARCHAR (255)	NOT NULL

3. GENERAL ARCHITECTURE

3.1 Layered Architecture

3.1.1 Application Layer



3.1.1.1 Manage Profile [SDD-REQ-100]

Class Name	Description
adminprofile	A view class for admin to register staff in the EMUN system
staffprofile	A view class for staff to manage their profile in the EMUN system

userprofile	A view class for user to manage their profile in the EMUN system

3.1.1.2 Auth [SDD-REQ-200]

Class Name	Description
login	A view class for user, staff and admin login into the EMUN system
register	A view class for user to register into the EMUN system
verify	A view class for user to verify the account in the EMUN system
confirm	A view class for user to confirm the account in the EMUN system
email	A view class for user to enter the email to reset password in the EMUN system
reset	A view class for user to reset password in the EMUN system.

3.1.1.3 Manage Preparation Course [SDD-REQ-300]

Class Name	Description
create	A view class for applicant to register the marriage preparation course by fill in the form
manage	A view class for applicant display the information regarding to marriage preparation course
applicantList	A view class for JAIP staff to see the list of applicants that apply marriage preparation course
payment	A view class for applicant to submit proof of payment

3.1.1.4 Manage Marriage Request [SDD-REQ-400]

Class Name	Description
create	A view class for applicant to register the marriage application by fill in the form
manage	A view class for applicant edit their information
document	A view class for applicant to submit document

3.1.1.5 Manage Registration [SDD-REQ-500]

Class Name	Description
spouseList	A view class that displays the spouse's information and allow the user to search, edit, delete, print and submit
create	A view class that allows the user to create a marriage registration application
manage	A view class that displays the user's marriage registration application and allow staff to accept or reject
manageUser	A view class that displays the marriage registration application
uploadPayment	A view class that displays payment receipt request and allow the user to upload their receipt

3.1.1.6 Manage Card [SDD-REQ-600]

Class Name	Description
create	A view class allow the user to request for the marriage card
manage	A view class that displays the user's marriage card application and allow staff to accept or reject
manageUser	A view class that displays the marriage card application
uploadPayment	A view class that displays payment receipt request and allow the user to upload their receipt

3.1.1.7 Manage consultation [SDD-REQ-700]

Class Name	Description
create	A view class to display form to allow the user to fill information details
show	A view class to allow JAIP staff to view the information of the application

edit	A view class to allow JAIP staff to fill user appointment detail and approve their application
userManage	A view class to allow user to check the list of consultation applications made
userShow	A view class to allow user to view the information of the application made
manage	A view class to display the list of application and allow JAIP staff to manage the consultation application

3.1.1.8 Manage consultant [SDD-REQ-800]

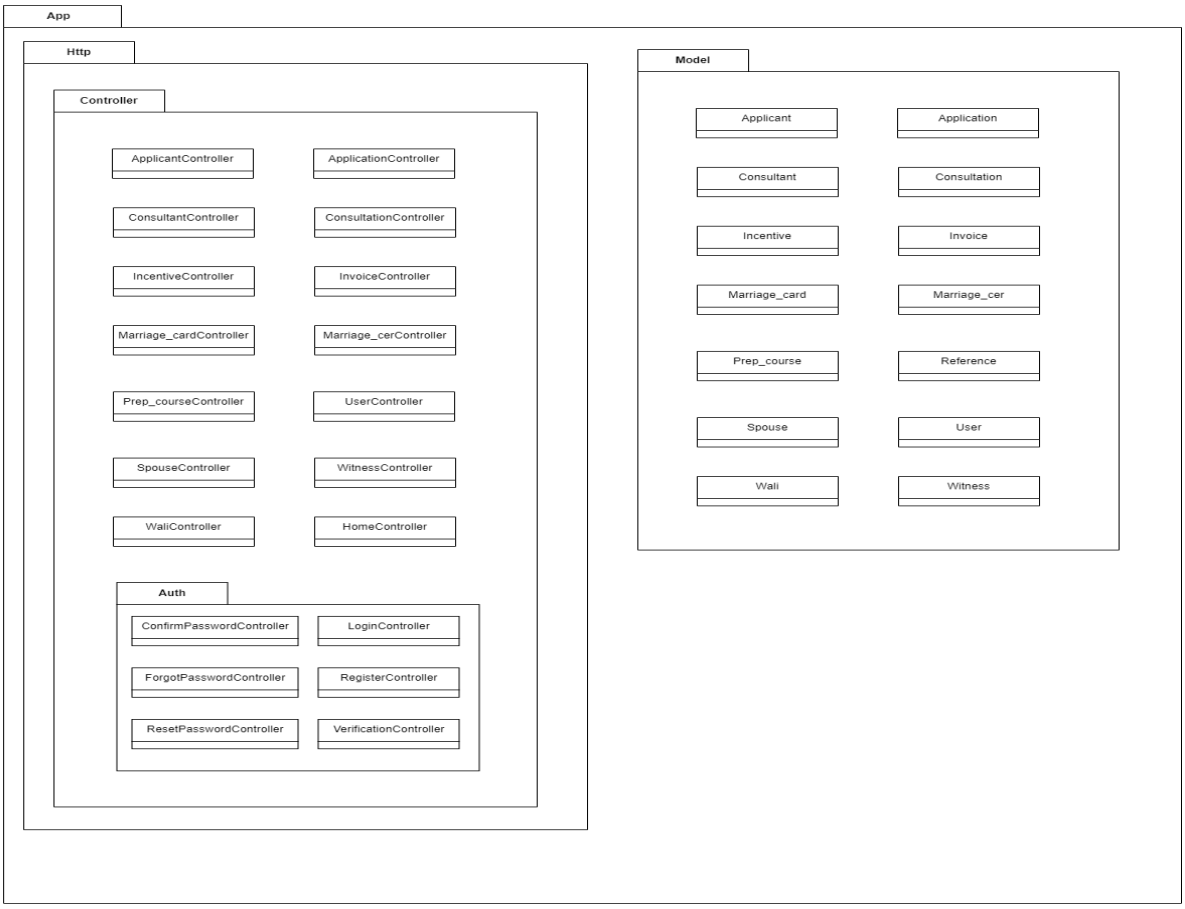
Class Name	Description
edit	A view class to allow JAIP staff to update the consultant's information
manage	A view class to allow JAIP staff to manage the list of consultants
create	A view class to allow JAIP staff to enter information of new consultant
show	A class that handles the consultant information

3.1.1.9 Manage Incentive [SDD-REQ-900]

Class Name	Description
create	A view class that displays the apply incentive page upon request which includes an application form
view	A view class that displays the user's incentive application details, hence information in a specified format
update	A view class that displays the user's information and allows the user to edit their information
delete	A view class that displays the user's information and allows the user to delete their information

3.1.2 Business Services Layer

3.1.2.1 Controller [SDD-REQ-1000]



Class Name	Description
ApplicantController	Handles the logic and actions related to managing applicants in the application. Contains methods for creating, updating, and deleting applicant records, as well as retrieving applicant information.
ApplicationController	Manages the application-related functionality within the application. Contains methods for handling the creation, editing, deletion, and retrieval of application records.
ConsultantController	Responsible for handling the actions and operations related to managing consultants in the application. Contains

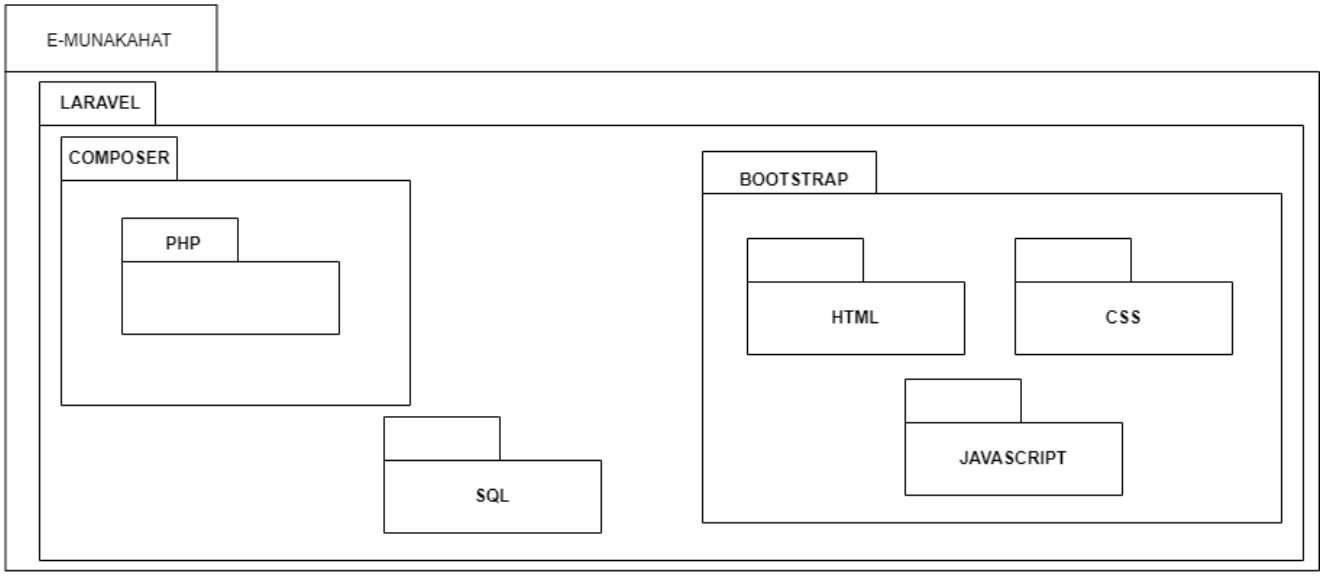
	methods for creating, updating, deleting, and retrieving consultant information.
ConsultationController	Handles the actions and operations associated with managing consultations in the application. Contains methods for scheduling consultations, updating consultation details, and retrieving consultation information.
IncentiveController	Manages the logic and actions related to incentives in the application. Contains methods for creating, updating, deleting, and retrieving incentive records.
InvoiceController	Handles the actions and operations related to managing invoices within the application. Contains methods for creating, updating, deleting, and retrieving invoice records.
Marriage_cardController	Manages the functionality associated with marriage cards within the application. Contains methods for creating, updating, deleting, and retrieving marriage card records.
Marriage_cerController	Handles the actions and operations related to managing marriage ceremonies in the application. Contains methods for creating, updating, deleting, and retrieving marriage ceremony records.
Prep_courseController	Responsible for managing the functionality related to preparation courses in the application. Contains methods for creating, updating, deleting, and retrieving preparation course records.
SpouseController	Manages the functionality and operations related to spouses within the application. Contains methods for creating, updating, deleting, and retrieving spouse records.
UserController	Handles the actions and operations related to managing users within the application. Contains methods for user registration, authentication, updating user information, and retrieving user details.
WaliController	Manages the logic and actions associated with managing walis (guardians) within the application. Contains methods for creating, updating, deleting, and retrieving wali records.

WitnessController	Handles the actions and operations related to managing witnesses within the application. Contains methods for creating, updating, deleting, and retrieving witness records.
HomeController	Handles the main functionality and actions related to the home page or main dashboard of the application. Contains methods for retrieving and displaying relevant information on the home page.
ConfirmPasswordController	This controller is responsible for handling password confirmations and uses a simple trait to include the behaviour
ForgotPasswordController	This controller is responsible for handling password reset emails and includes a trait that assists in sending these notifications from your application to your users.
LoginController	This controller handles authenticating users for the application and redirecting them to your home screen.
RegisterController	This controller handles the registration of new users as well as their validation and creation.
ResetPasswordController	This controller is responsible for handling password reset requests and uses a simple trait to include this behaviour.
VerificationController	This controller is responsible for handling email verification for any user that recently registered with the application.

3.1.2.2 Model [SDD-REQ-1100]

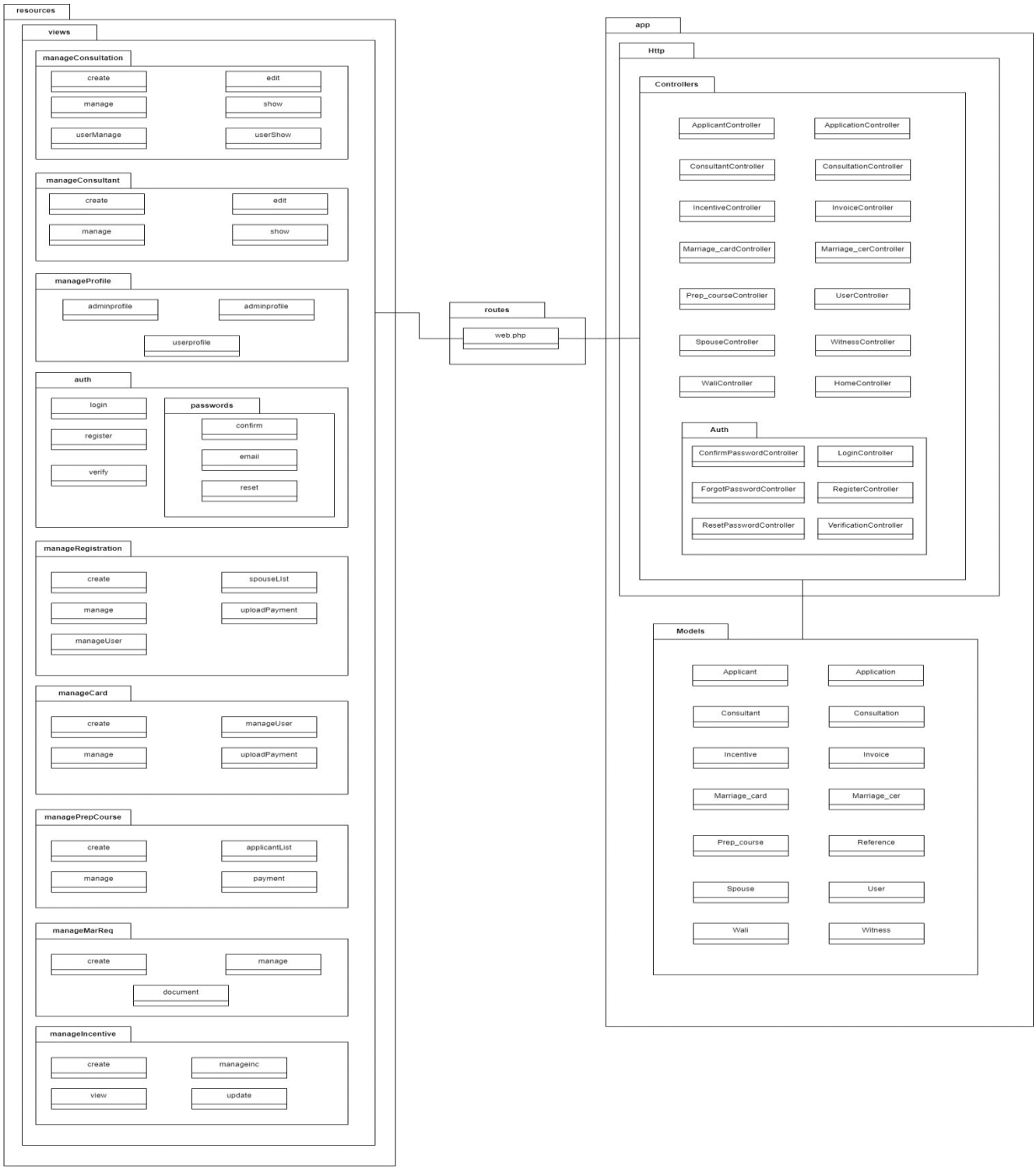
Class Name	Description
Applicant	This class retrieve and store applicant details
Application	This class retrieve and store application details
Consultant	This class retrieve and store consultant details
Consultation	This class retrieve and store consultation details
Incentive	This class retrieve and store incentive details
Invoice	This class retrieve and store invoice details
Marriage_card	This class retrieve and store marriage card details
Marriage_cer	This class retrieve and store marriage certificate details
Prep_course	This class retrieve and store preparation course details
Reference	This class retrieve and store reference details
Spouse	This class retrieve and store spouse details
User	This class retrieve and store user details
Wali	This class retrieve and store wali details
Witness	This class retrieve and store witness details

3.1.3 Middleware Layer



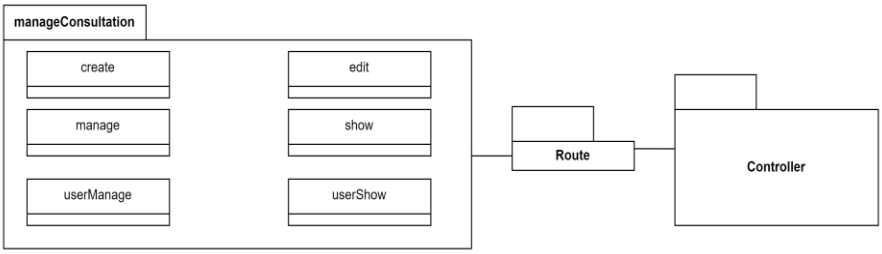
Package Name	Description
HTML	The package that contains of creating and design a website page
CSS	The package that contains of styling of the HTML package
JAVASCRIPT	The package that contains of enhancing the interactivity and functionality of website pages
LARAVEL	PHP web application framework. It contains various packages and classes for building web application.
PHP	The package that contains of code that builds dynamic web applications
SQL	The package that contains the database management

3.2 MVC Package Relationship



4. DETAIL DESIGN

4.1 manageConsultation [SDD-REQ-700]



4.1.1 create [SDD-REQ-701]

Class Type	Boundary class	
Responsibility	This interface allows user to fill the consultation application form	
Attributes	Attributes Name	Attributes Type
	$user $applicant $spouse $slots $locations	User Applicant Spouse Array Array
Methods	Method Name	Description
	store ()	Method to create new Consultation and store the data.
	userManage()	Method to redirect user to userManage class in manageConsultation package
Algorithm	1) Start 2) Display form 3) If save button is clicked a) Call store() 4) End if 5) If consultation button is clicked a) Call userManage() 6) End if 7) End	

4.1.2 manage [SDD-REQ-702]

Class Type	Boundary class	
Responsibility	This interface allows JAIP Staff to manage the consultation application list.	
Attributes	Attributes Name	Attributes Type
	$datas	Consultation
Methods	Method Name	Description
	edit($id)	Method to redirect JAIP Staff to edit class in manageConsultation package
	show($id)	Method to redirect JAIP Staff to show class in manageConsultation package
	decline($id)	Method to update the status of consultation to decline
Algorithm	1) Start 2) List all consultation 3) If edit icon is clicked a) Call edit() 4) End if 5) If show icon is clicked a) Call show() 6) End if 7) If decline icon is clicked a) Call decline() 8) End if 9) End	

4.1.3 userManage [SDD-REQ-703]

Class Type	Boundary class	
Responsibility	This interface allows user to manage the consultation application list.	
Attributes	Attributes Name	Attributes Type

	$datas	Consultation
Methods	Method Name	Description
	userShow($id)	Method to redirect user to userShow class in manageConsultation package
	create()	Method to redirect user to create class in manageConsultation package
Algorithm	1) Start 2) List all consultations 3) If show icon is clicked a) Call userShow() 4) End if 5) If new button is clicked a) Call create() 6) End if 7) End	

4.1.4 edit [SDD-REQ-704]

Class Type	Boundary class	
Responsibility	This interface allows JAIP Staff to update the consultation application	
Attributes	Attributes Name	Attributes Type
	$id $consultants $datas	Bigint Consultant Consultation
Methods	Method Name	Description
	update(Request $request, $id)	Method to update the Consultation's details.
	decline($id)	Method to update the status of consultation to decline
	manage()	Method to redirect JAIP Staff to manage class in manageConsultation package
Algorithm	1) Start 2) Display form with previous details	

	3) If update button is clicked a) Call update() 4) End if 5) If back button is clicked a) Call manage() 6) End if 7) If decline button is clicked a) Call decline() 8) End if 9) End

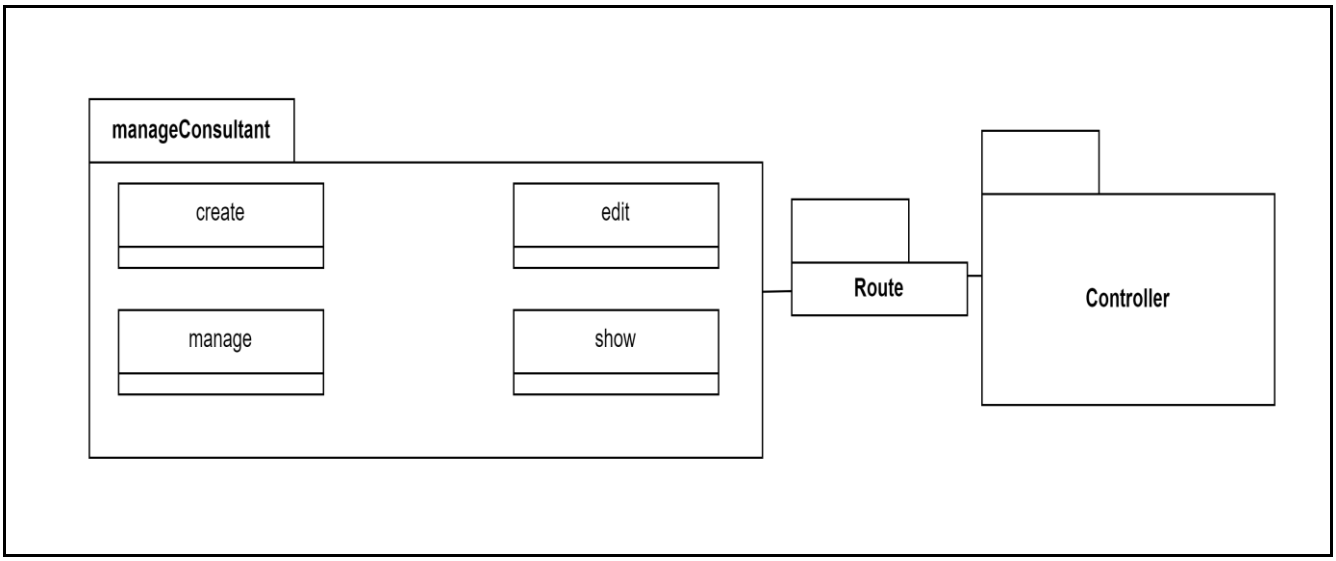
4.1.5 show [SDD-REQ-705]

Class Type	Boundary class	
Responsibility	This interface allows JAIP Staff view the consultation application details	
Attributes	Attributes Name	Attributes Type
	$data	Consultation
Methods	Method Name	Description
	displayFile($fileName)	Method to display file
	manage()	Method to redirect JAIP Staff to manage class in manageConsultation package
Algorithm	1) Start 2) Display consultation details 3) If file button is clicked a) Call displayFile() 4) End if 5) If back button is clicked a) Call manage() 6) End if 7) End	

4.1.6 userShow [SDD-REQ-706]

<table>
<tr><td>Class Type</td><td colspan="2">Boundary class</td></tr>
<tr><td>Responsibility</td><td colspan="2">This interface allow user to view the consultation application details</td></tr>
<tr><td rowspan="2">Attributes</td><td>Attributes Name</td><td>Attributes Type</td></tr>
<tr><td>$data</td><td>Consultation</td></tr>
<tr><td rowspan="3">Methods</td><td>Method Name</td><td>Description</td></tr>
<tr><td>displayFile($fileName)</td><td>Method to display file</td></tr>
<tr><td>userManage()</td><td>Method to redirect user to userManage class in manageConsultation package</td></tr>
<tr><td>Algorithm</td><td colspan="2">1) Start
2) Display consultation details
3) If file button is clicked
a) Call displayFile()
4) End if
5) If back button is clicked
a) Call userManage()
6) End if
7) End</td></tr>
</table>

4.2 manageConsultant [SDD-REQ-800]



4.2.1 create [SDD-REQ-801]

Class Type	Boundary class	
Responsibility	This interface allow JAIP Staff to create new consultant in the system.	
Attributes	Attributes Name	Attributes Type
	$departments	Reference
	$locations	Reference
Methods	Method Name	Description
	store(Request $request)	Method to create new Consultant and store the data.
	manage()	Method to redirect JAIP Staff to manage class in manageConsultant
Algorithm	1) Start 2) Display form 3) If save button is clicked a) Call store() 4) End if 5) If save button is clicked a) Call manage() 6) End if 7) End	

4.2.2 edit [SDD-REQ-802]

Class Type	Boundary class	
Responsibility	This interface allow JAIP Staff to edit consultant's detail.	
Attributes	Attributes Name	Attributes Type
	$departments $locations $consultant	Reference Reference Consultant
Methods	Method Name	Description
	manage()	Method to redirect JAIP Staff to manage class in manageConsultant
	update(Request $request)	Method to update the Consultant's details.
Algorithm	1) Start 2) Display details 3) If update button is clicked a) Call update() 4) End if 5) If back button is clicked a) Call manage() 6) End if 7) End	

4.2.3 manage [SDD-REQ-803]

Class Type	Boundary class	
Responsibility	This interface allow JAIP Staff to manage list of consultants.	
Attributes	Attributes Name	Attributes Type
	$datas	Consultant
Methods	Method Name	Description
	create()	Method to redirect JAIP Staff to create class in manageConsultant

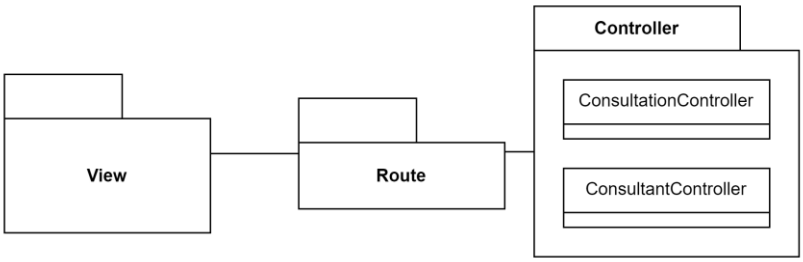
	edit($id)	Method to redirect JAIP Staff to edit class in manageConsultant
	destroy($id)	Method to delete record of Consultant in the database
	show($id)	Method to redirect JAIP Staff to show class in manageConsultant
Algorithm	1) Start 2) Display consultants list 3) If edit icon is clicked a) Call edit() 4) End if 5) If new button is clicked a) Call create() 6) End if 7) If delete icon is clicked a) Call destroy() 8) End if 9) If show icon is clicked a) Call show() 10) End if 11) End	

4.2.4 show [SDD-REQ-804]

Class Type	Boundary class	
Responsibility	This interface allow JAIP Staff to view consultant's detail.	
Attributes	Attributes Name	Attributes Type
	$consultant	Consultant
Methods	Method Name	Description
	manage()	Method to redirect JAIP Staff to manage class in manageConsultant
Algorithm	1) Start 2) Display consultant details	

	3) If back button is clicked a) Call manage() 4) End if 5) End

4.3 Controller [SDD-REQ-1000]



4.3.1 ConsultantController [SDD-REQ-1003]

Class Type	**Controller Class**	
Responsibility	This controller controls activities between manageConsultant interfaces and model class related to consultants.	
Attributes	Attributes Name	Attributes Type
	None	None
Methods	Method Name	Description
	index ()	Method to redirect user to staff.consultant.manage route
	manage()	Method to redirect JAIP Staff to manage class in manageConsultant
	create()	Method to redirect JAIP Staff to create class in manageConsultant
	store(Request $request)	Method to create new Consultant and store the data.
	update(Request $request)	Method to update the Consultant's details.
	edit($id)	Method to redirect JAIP Staff to edit class in manageConsultant
	show($id)	Method to redirect JAIP Staff to show class in manageConsultant

	destroy($id)	Method to delete record of Consultant in the database
Algorithm	1. Start 2. Redirect to the 'staff.consultant.manage' route. 3. Define the 'manage' function: • Retrieve consultant data with location and department information, paginated by 10. • Display the 'manageConsultant.manage' view, passing the 'datas' variable. 4. Define the 'create' function: • Retrieve locations and departments from the 'Reference' model, ordered by code. • Display the 'manageConsultant.create' view, passing the 'locations' and 'departments' variables. 5. Define the 'store' function with a request parameter: • Create a new consultant record using the request data. • Redirect to the 'staff.consultant.manage' route with a success message. 6. Define the 'update' function with request and id parameters: • Find the consultant with the given id and update its information using the request data. • Redirect to the 'staff.consultant.manage' route with a success message. 7. Define the 'edit' function with an id parameter: • Retrieve the consultant with the given id, including its location and department information. • Retrieve locations and departments from the 'Reference' model, ordered by code. • Display the 'manageConsultant.edit' view, passing the 'locations', 'departments', and 'consultant' variables. 8. Define the 'show' function with an id parameter: • Retrieve the consultant with the given id, including its location and department information.	

	• Display the 'manageConsultant.show' view, passing the 'consultant' variable. 9. Define the 'destroy' function with an id parameter: • Find the consultant with the given id. • If the consultant does not exist, redirect back with an error message. • Delete the consultant. • Redirect to the 'staff.consultant.manage' route with a success message. 10. End

4.3.2 ConsultationController [SDD-REQ-1004]

<table>
<tr><td>Class Type</td><td colspan="2">Controller Class</td></tr>
<tr><td>Responsibility</td><td colspan="2">This controller controls activities between manageConsultation interfaces and model class related to consultations.</td></tr>
<tr><td rowspan="2">Attributes</td><td>Attributes Name</td><td>Attributes Type</td></tr>
<tr><td>None</td><td>None</td></tr>
<tr><td rowspan="11">Methods</td><td>Method Name</td><td>Description</td></tr>
<tr><td>userManage()</td><td>Method to redirect user to userManage class in manageConsultation package</td></tr>
<tr><td>manage()</td><td>Method to redirect JAIP Staff to manage class in manageConsultation package</td></tr>
<tr><td>create()</td><td>Method to redirect user to create class in manageConsultation package</td></tr>
<tr><td>store(Request $request)</td><td>Method to create new Consultation and store the data.</td></tr>
<tr><td>update(Request $request, $id)</td><td>Method to update the Consultation’s details.</td></tr>
<tr><td>edit($id)</td><td>Method to redirect JAIP Staff to edit class in manageConsultation package</td></tr>
<tr><td>show($id)</td><td>Method to redirect JAIP Staff to show class in manageConsultation package</td></tr>
<tr><td>decline($id)</td><td>Method to update the status of consultation to decline</td></tr>
<tr><td>userShow($id)</td><td>Method to redirect user to userShow class in manageConsultation package</td></tr>
<tr><td>displayFile($fileName)</td><td>Method to display file</td></tr>
<tr><td>Algorithm</td><td colspan="2">1. Start
2. Define the 'manage' function:
• Retrieve consultations with related data, paginated by 3.
• Display the 'manageConsultation.manage' view with 'datas'.
3. Define the 'userManage' function:
• Retrieve the applicant associated with the authenticated user.</td></tr>
</table>

- Retrieve consultations with related data where 'app_id' matches the applicant's id, paginated by 9.
- Display the 'manageConsultation.userManage' view with 'datas'.

4. Define the 'create' function:

- Retrieve locations and slots.
- Retrieve the authenticated user.
- Display the 'manageConsultation.create' view with 'user', 'locations', and 'slots'.

5. Define the 'store' function with a request parameter:

- Retrieve the uploaded file.
- Generate a unique file name.
- Store the file.
- Update user information.
- Update the applicant's details.
- Check and create/update spouse information.
- Create a consultation record.
- Redirect to the 'user.consultation.manage' route with a success message.

6. Define the 'edit' function with an id parameter:

- Retrieve consultation data with related spouse, applicant's user, slot, and location.
- Retrieve consultants.
- Display the 'manageConsultation.edit' view with 'data', 'consultants', and 'id'.

7. Define the 'update' function with request and id parameters:

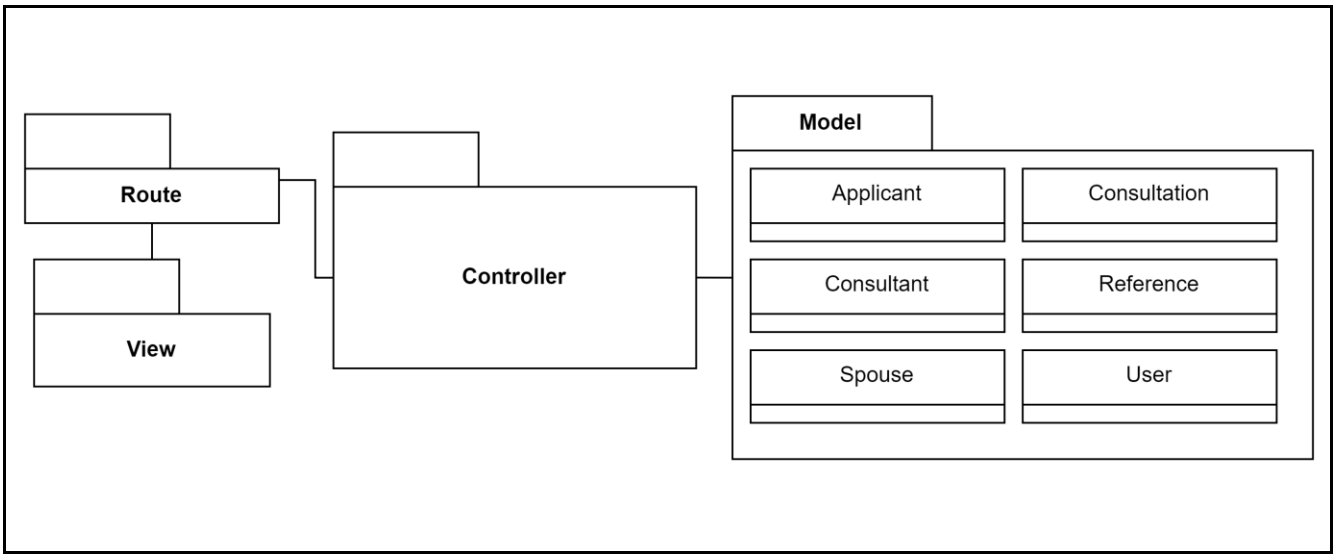
- Set the 'ref_status_id' field to 3 in the request.
- Update the consultation record with the given id using the request data.
- Redirect to the 'staff.consultation.manage' route with a success message.

8. Define the 'decline' function with an id parameter:

- Set the 'ref_status_id' field to 2 in the status array.

	• Update the consultation record with the given id using the status array. • Redirect to the 'staff.consultation.manage' route. 9. Define the 'show' function with an id parameter: • Retrieve consultation data with related spouse, applicant's user, location, slot, and status. • Display the 'manageConsultation.show' view with 'data' and 'id'. 10. Define the 'displayFile' function with a fileName parameter: • Create the file path. • Check if the file exists. • Retrieve the file content and MIME type. • Return a response with the file content, MIME type, and inline filename. • Abort with a 404 error if the file doesn't exist. 11. End

4.4 Model [SDD-REQ-1100]



4.4.1 Applicant [SDD-REQ-1101]

Class Type	**Model Class**	
Responsibility	This model manage the retrieving and storing data from applicants table and also manage its relationship with other tables.	
Attributes	Attributes Name	Attributes Type
	$fillable	Array
Methods	Method Name	Description
	user()	Method to link the applicant model with user model
	consultation()	Method to link the applicant model with consultation model
Algorithm	1. Start 2. Define the "Applicant" class and make it extend the "Model" class. 3. Define the fillable attributes of the "Applicant" model: "id", "user_id", "birthdate", "age", "nationality", and "houseaddress". 4. Define a relationship method named "user": • The method should return a belongsTo relationship with the "User" model. 5. Define a relationship method named "consultation":	

	• The method should return a hasMany relationship with the "Consultation" model. 6. End

4.4.2 Consultant [SDD-REQ-1103]

Class Type	**Model Class**	
Responsibility	This model manage the retrieving and storing data from consultants table and also manage its relationship with other tables.	
Attributes	Attributes Name	Attributes Type
	$fillable	Array
Methods	Method Name	Description
	created_by()	Method to link the consultant model with user model
	location()	Method to link the consultant model with reference model
	department()	Method to link the consultant model with reference model
Algorithm	1. Start 2. Define the fillable attributes of the "Consultant" model: "id", "created_by", "IcNum", "name", "email", "ref_department_id", "ref_location_id", and "phoneNo". 3. Define a relationship method named "created_by": • The method should return a belongsTo relationship with the "User" model, using the "created_by" foreign key. 4. Define a relationship method named "location": • The method should return a belongsTo relationship with the "Reference" model, using the "ref_location_id" foreign key. 5. Define a relationship method named "department": • The method should return a belongsTo relationship with the "Reference" model, using the "ref_department_id" foreign key. 6. End	

4.4.3 Consultations [SDD-REQ-1104]

<table>
<tr><td>Class Type</td><td colspan="2">Model Class</td></tr>
<tr><td>Responsibility</td><td colspan="2">This model manage the retrieving and storing data from consultations table and also manage its relationship with other tables.</td></tr>
<tr><td rowspan="2">Attributes</td><td>Attributes Name</td><td>Attributes Type</td></tr>
<tr><td>$fillable</td><td>Array</td></tr>
<tr><td rowspan="8">Methods</td><td>Method Name</td><td>Description</td></tr>
<tr><td>managed_by()</td><td>Method to link the consultation model with user model</td></tr>
<tr><td>spouse()</td><td>Method to link the consultation model with spouse model</td></tr>
<tr><td>applicant()</td><td>Method to link the consultation model with applicant model</td></tr>
<tr><td>consultant()</td><td>Method to link the consultation model with consultant model</td></tr>
<tr><td>location()</td><td>Method to link the consultation model with reference model</td></tr>
<tr><td>slot()</td><td>Method to link the consultation model with reference model</td></tr>
<tr><td>status()</td><td>Method to link the consultation model with reference model</td></tr>
<tr><td>Algorithm</td><td colspan="2">1. Start
2. Define the fillable attributes of the "Consultation" model: "id", "date", "ref_slot_id", "description", "document", "ref_location_id", "managed_by", "sp_id", "app_id", "cons_id", and "ref_status_id".
3. Define a relationship method named "managed_by":
• The method should return a belongsTo relationship with the "User" model, using the "managed_by" foreign key.
4. Define a relationship method named "spouse":
• The method should return a belongsTo relationship with the "Spouse" model, using the "sp_id" foreign key.
5. Define a relationship method named "applicant":</td></tr>
</table>

	• The method should return a belongsTo relationship with the "Applicant" model, using the "app_id" foreign key. 6. Define a relationship method named "consultant": • The method should return a belongsTo relationship with the "Consultant" model, using the "cons_id" foreign key. 7. Define a relationship method named "location": • The method should return a belongsTo relationship with the "Reference" model, using the "ref_location_id" foreign key. 8. Define a relationship method named "slot": • The method should return a belongsTo relationship with the "Reference" model, using the "ref_slot_id" foreign key. 9. Define a relationship method named "status": • The method should return a belongsTo relationship with the "Reference" model, using the "ref_status_id" foreign key. 10. End

4.4.4 Reference [SDD-REQ-1110]

Class Type	**Model Class**	
Responsibility	This model manage the retrieving and storing data from references table and also manage its relationship with other tables.	
Attributes	Attributes Name	Attributes Type
	$fillable	Array
Methods	Method Name	Description
	None	None
Algorithm	1. Start 2. Define the fillable attributes of the "Reference" model: "id", "name", "code", and "value". 3. End	

4.4.5 Spouse [SDD-REQ-1111]

Class Type	**Model Class**

Responsibility	This model manage the retrieving and storing data from spouses table and also manage its relationship with other tables.	
Attributes	Attributes Name	Attributes Type
	$fillable	Array
Methods	Method Name	Description
	None	None
Algorithm	1. Start 2. Define the fillable attributes of the "Spouse" model: "id", "name", "birthdate", "email", "ic", "gender", "phonenumber", "nationality", "address", and "age". 3. End	

4.4.6 User [SDD-REQ-1112]

Class Type	**Model Class**	
Responsibility	This model manage the retrieving and storing data from users table and also manage its relationship with other tables.	
Attributes	Attributes Name	Attributes Type
	$fillable $hidden $cast	Array Array Array
Methods	Method Name	Description
	role()	Method to returns an Attribute object representing the role attribute and provides a mapping functionality for converting input values to user roles from a predefined array. It ensures controlled access to the role attribute within the class and its subclasses.
Algorithm	1. Start 2. Create a class named "User" and make it extend the "Authenticatable" class. 3. Implement the "MustVerifyEmail" interface. 4. Use the traits "HasApiTokens", "HasFactory", and "Notifiable" in the "User" class.	

	5. Define the fillable attributes of the "User" model: "ic", "staff_id", "role", "name", "gender", "contact", "email", and "password". 6. Define the hidden attributes of the "User" model: "password" and "remember_token". 7. Define the casts for the "User" model: "email_verified_at" is cast as "datetime", and "password" is cast as "hashed". 8. Define a method named "role" that returns an instance of the "Attribute" class. The method should accept a value and use a lambda function to map the value to the corresponding role ("user", "staff", or "admin"). 9. End

5. REQUIREMENT TRACEABILITY

5.1 Manage Marriage Consultation [SRS-REQ-400]

CHUA KIAN PHENG [CB21106]

Requirement ID	Description	Design ID
SRS-REQ-UC-400	**Manage Consultation** The system provides the user with the capability to apply for consultation, search for existing consultation application.	SDD-REQ-701 SDD-REQ-703
SRS-REQ-UC 400-1	**Print Application** The system provides the capability for the user to print the consultation application made details by clicking the print icon.	SDD-REQ-706
SRS-REQ-UC 400-2	**Delete Record** The system provides the capability to the user to delete the information stored as long as it not submitted yet by clicking the print icon.	SDD-REQ-703
SRS-REQ-UC 400-3	**Update Information** The system provides the capability to the user to update the information that they entered by clicking the edit icon.	
SRS-REQ-UC 400-4	**View Application Made** The system provides the user with the capability to view the consultation applications made details by clicking the view icon.	SDD-REQ-706
SRS-REQ-UC 400-5	**Approve Application** The system provides the capability to the user to approve the consultation application made by <> button	SDD-REQ-704
SRS-REQ-UC 400-6	**Search for Application**	SDD-REQ-702

	The system provides the capability to the user to search for specific consultation application made by entering the applicant's IC number of the desired application details.	
SRS-REQ- UC 400-7	**The Information is Not Complete** The system can verify that the compulsory information entered by the user are not completed.	SDD-REQ-701
SRS-REQ- UC 400-8	**The Identification Card Number's Length is Not Valid** The system can verify that the length of the IC number entered by the user is not valid either it is more or less than 12.	SDD-REQ-701

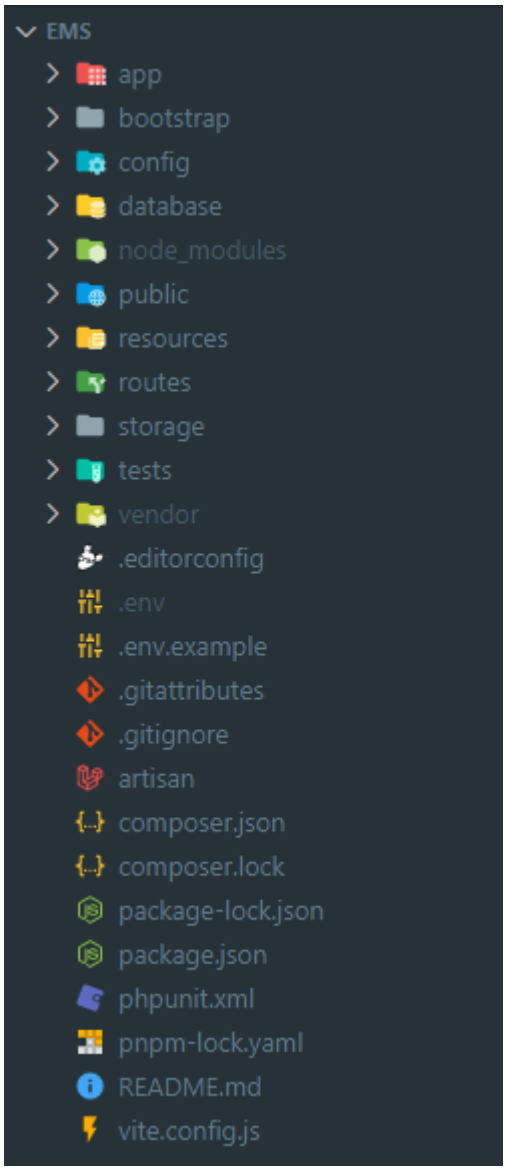
APPENDIX

APPENDIX A

3. Requirement Traceability

Requirements	Description
SRS-REQ-UC-400	**Manage Consultation** The system provides the user with the capability to apply for consultation, search for existing consultation application.
SRS-REQ-UC-400-1	**Print Application** The system provides the capability for the user to print the consultation application made details by clicking the print icon.
SRS-REQ-UC-400-2	**Delete Record** The system provides the capability to the user to delete the information stored as long as it not submitted yet by clicking the print icon.
SRS-REQ-UC-400-3	**Update Information** The system provides the capability to the user to update the information that they entered by clicking the edit icon.
SRS-REQ-UC-400-4	**View Application Made** The system provides the user with the capability to view the consultation applications made details by clicking the view icon.
SRS-REQ-UC-400-5	**Approve Application** The system provides the capability to the user to approve the consultation application made by <<Sahkan Permohonan>> button
SRS-REQ-UC-400-6	**Search for Application** The system provides the capability to the user to search for specific consultation application made by entering the applicant's IC number of the desired application details.
SRS-REQ-UC-400-7	**The Information is Not Complete** The system can verify that the compulsory information entered by the user are not completed.
SRS-REQ-UC-400-8	**The Identification Card Number's Length is Not Valid** The system can verify that the length of the IC number entered by the user is not valid either it is more or less than 12.

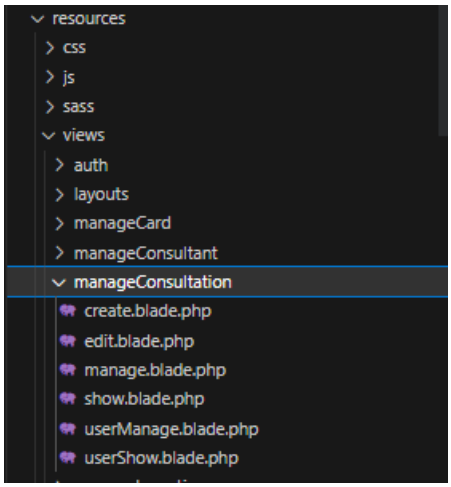
APPENDIX B



System Name	Layer		
EMS	resources	views	
	app	Http	Controllers
		Models	
	routes		

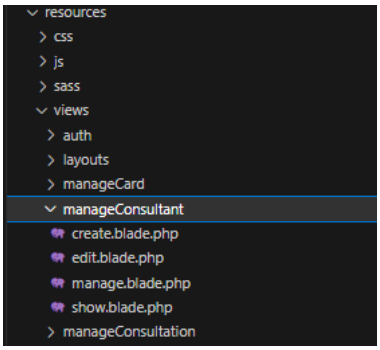
APPENDIX C1

manageConsultation - Views Layer



Layer	Package	Class
views	manageConsultation	create
		manage
		userManage
		show
		userShow
		edit

manageConsultant - Views Layer

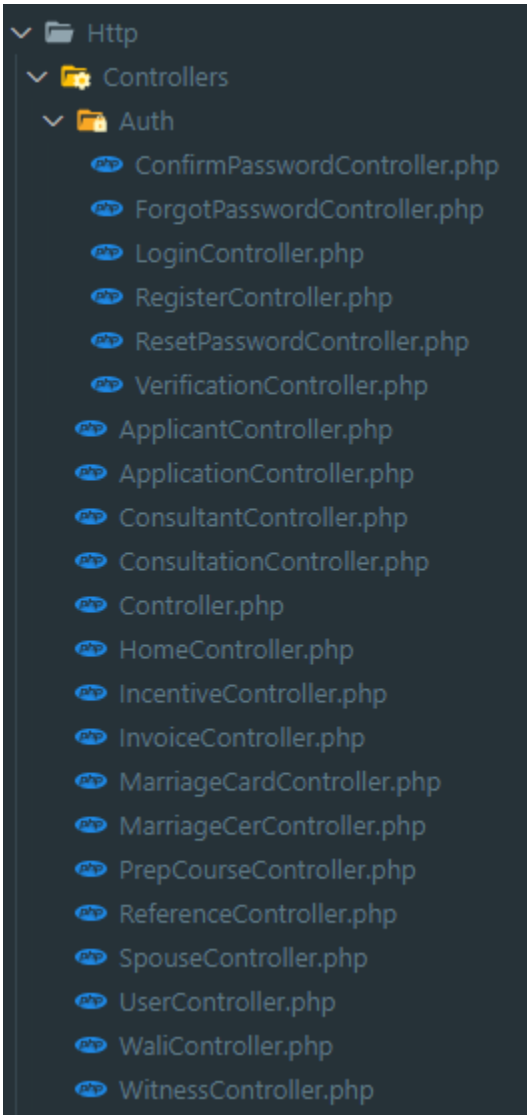


Layer	Package	Class
views	manageConsultant	edit
		create
		show
		manage

APPENDIX C2

APPENDIX C3

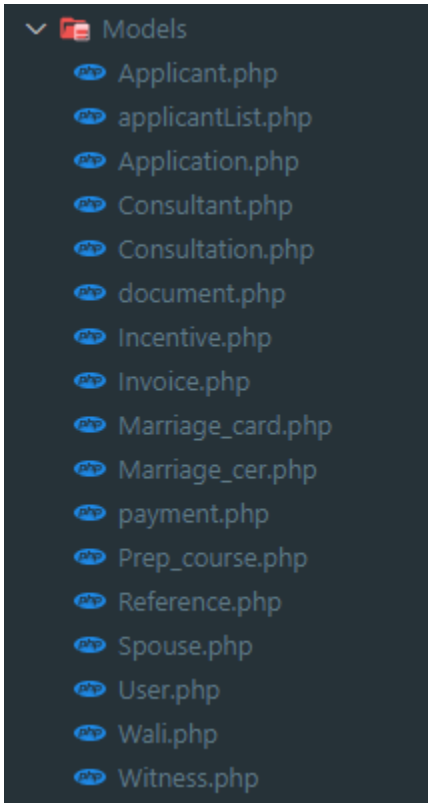
Controller Layer



Layer	Class
Controllers	ConsultantController ConsultationController

APPENDIX C4

Model Layer



Layer	Class
Models	User Applicant Consultant Consultation Reference Spouse