Software Design Documentation

For

Workshop Management System

Version 1.0 approved

Prepared by

CB23116 AMALA KARTHIGAYAN

CB222141 NUR ALYA SYAKIRAH BINTI NASARUDIN

CB22139 VARSHINI JAGARAJAN

CB23101 NUR NABIHAH FATINY BINTI ISMAIL

26/4/2025

Table of Contents

1.0 Introduction

1.1 Purpose

The process of software design involves converting user requirements into a structured plan that assists programmers during the coding and implementation phases. It provides a clear and detailed overview of how the system will be constructed, ensuring development progresses with a strong understanding of the intended functionality and structure. This design document is produced during the software design stage and is based on the specifications outlined in the Software Requirements Specification (SRS).

This particular Software Design Document (SDD) outlines a simple system that acts as a proof of concept, aiming to demonstrate the system's feasibility for broader production deployment. The design activities described here focus on the system’s core components and foundational framework. Special attention is given to the creation, handling, and management of documents within the system.

The system discussed in this document is intended to operate alongside existing platforms and will mainly offer a document management interface that simplifies object handling and document operations. Additionally, the system architecture section provides a high-level overview of the system’s design, illustrating its relationship with other external systems. This helps readers and users of the document to quickly understand the broader design context before exploring detailed components.

1.2 System Overview

The Workshop Management System is designed to simplify and improve the daily operations of vehicle repair workshops. It provides an integrated platform that allows workshop owners, foremen, and customers to interact efficiently while ensuring smooth business operations.

Workshop owners can manage essential tasks such as tracking inventory, scheduling employees, requesting spare parts from other workshops, processing payroll, and updating personal profiles. Foremen have access to a scheduling system where they can select available work slots, ensuring fair task distribution and minimizing conflicts. Customers can share their experiences through a rating and review system, providing valuable feedback that helps improve service quality.

The system consists of several key features. The profile management feature ensures that user information is always up to date for effective communication. Inventory management prevents shortages and overstocking by allowing real-time tracking of spare parts and tools. The inventory request feature enables workshop owners to obtain necessary parts from other workshops quickly, reducing downtime. The foreman scheduling module helps allocate tasks effectively, while the scheduling selection feature allows foremen to manage their work shifts efficiently. Payroll processing is automated, ensuring accurate salary calculations based on work hours, deductions, and bonuses. Lastly, the rating and review system captures customer feedback, helping workshops enhance their services and build trust with clients.

By integrating these features, the Workshop Management System enhances efficiency, reduces delays, and improves overall customer satisfaction. The platform is designed to be user-friendly and scalable, ensuring it meets the evolving needs of modern workshops

Here are the modules for our system and explanation in detail:

i. **Manage Profile (NUR ALYA SYAKIRAH BINTI NASARUDIN CB22141)**
   In this module, Workshop Owners and Foremen can manage their profiles by updating personal information, credentials and other relevant details. Users are allowed to edit

and save changes to their profiles with input validation to maintain data accuracy. They can view and update details such as names, contact information, address and login credentials when needed. The system also applies role-based access control where users have specific permissions based on their role, helping to protect sensitive information. This module helps keep user data organized, accurate and easily manageable within the Workshop Management System.

ii. Manage Shop Inventory (VARSHINI JAGARAJAN, CB22139)

This module allows the Workshop Owner to effectively oversee and manage the shop's inventory. The Workshop Owner has the ability to add new inventory items, modify details of current items, and independently monitor stock levels. When specific items fall to low stock levels, the Workshop Owner can directly initiate a new inventory request using the Request Inventory extension module. All inventory operations, such as addition, modification, and stock changes, are logged and stored in the database for later tracking and reference. In conclusion, this module enables only the Workshop Owner to monitor and handle inventory availability, guaranteeing the seamless and ongoing function of the workshop.

iii. Request Inventory (VARSHINI JAGARAJAN, CB22139)

This extension module enables the Workshop Owner to request extra inventory items when stock levels are running low. The Workshop Owner is able to create and send a request detailing the needed items and amounts. When submitted, the request is logged in the system, and the Workshop Owner can manually adjust the inventory when the requested stock is received. This Request Inventory module helps the Workshop Owner in proactively managing inventory shortages, ensuring a well-stocked and efficient workshop setting.

iv. Manage Foreman Schedule (Amala Karthigayan, CB23116)

Manage Foreman Schedule module is for the Workshop Owners to possess an easy control and management of the allocation of available gig slots to foremen. Under this module, owners can create, read, update, and delete gig slots, such as setting the date, setting the time, setting the type of gig, and setting the number of foremen required. It supports dynamic scheduling requirements adjustments. Foremen are notified upon slot

updates. This module plays a critical role in resource utilization at the best level and balanced distribution of workload among the workshop workforce.

v. **Select Available Schedule (Amala Karthigayan, CB23116)**

Select Available Schedule module allows Foremen to browse a list of available work schedules and select a preferrable gig slot. The module gives an easy-to-use interface showing only valid, non-conflicting gig slots. Once a slot has been selected by a foreman, the system checks real-time availability to prevent over-allocation. Once verified, the system validates the selected schedule and updates the status accordingly. This mechanism ensures effective task allocation, minimizes schedule conflicts, and facilitates transparent open-task allocation among Foremen and Workshop Owners.

vi. **Manage Payroll (NUR NABIHAH FATINY BINTI ISMAILCB23101)**

This module allows the Workshop Owner to manage all payment-related activities in the workshop system. The owner can monitor, track, and handle every payment of staff salaries. The workshop owner can add the payment amount meanwhile the foreman, however, has limited access. They can only view their own payroll details payment history. This module helps keep all financial records organized, secure, and easy to manage, ensuring smooth and transparent payment operations within the workshop.

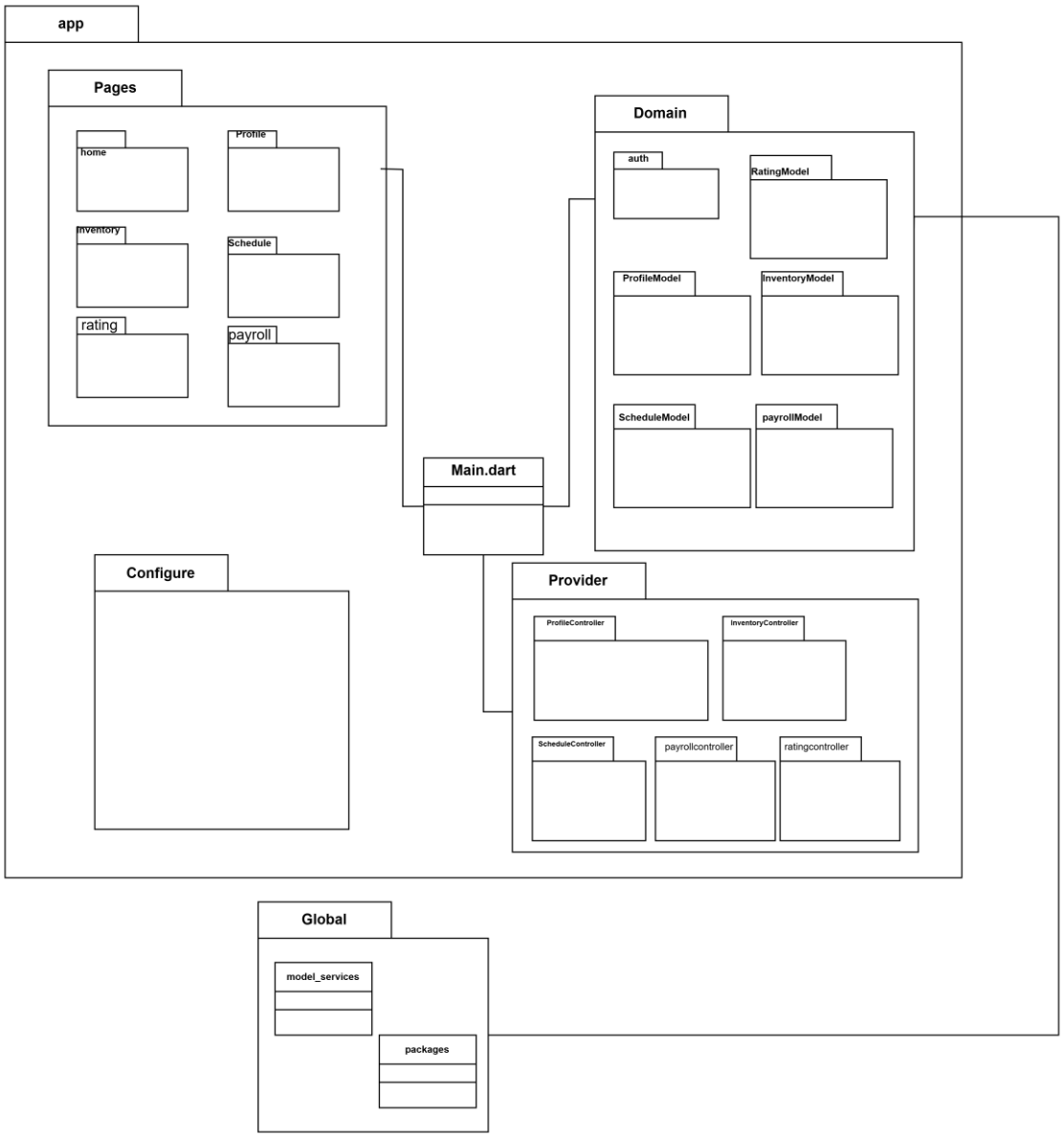
vii. **Manage Rating (NUR NABIHAH FATINY BINTI ISMAIL CB23101)**

In this module, Customers are allowed to rate the services provided by the workshop. They can give feedback based on their experience, which helps the workshop understand how well they are performing. The Workshop Owner and Foreman only can view all the ratings submitted by customers. This module helps improve service quality by collecting honest feedback and allows the workshop team to identify areas that need improvement.

2.0 Referenced Document

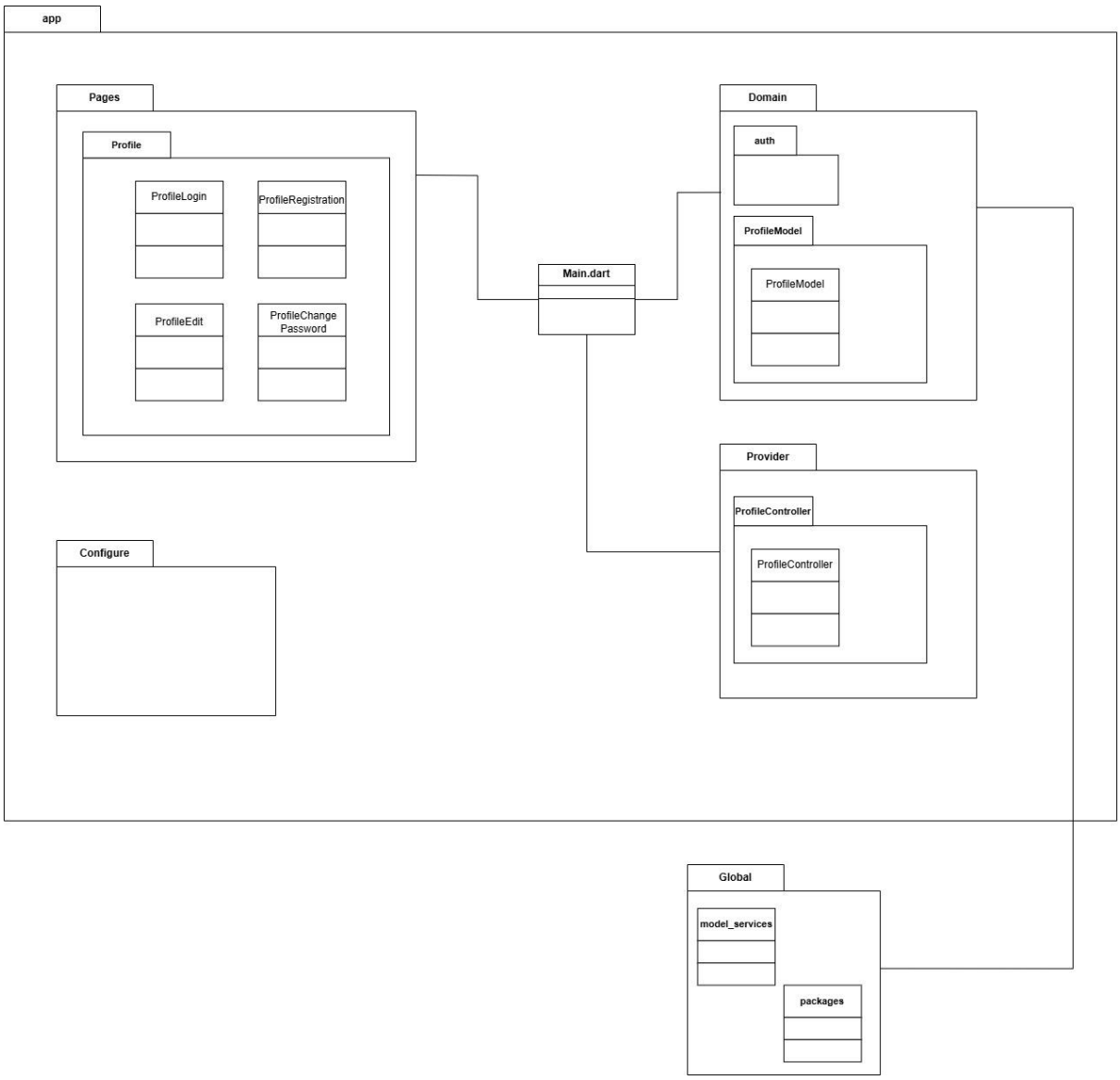
[1] IEEE 830-1998, IEEE Recommended Practice for Software Requirements Specifications, IEEE Computer Society, 1998.
[2] IEEE 1016-2009, IEEE Standard for Information Technology—Systems Design—Software Design Descriptions, IEEE Computer Society, 2009.
[3] IEEE 1058-1998, IEEE Standard for Software Project Management Plans, IEEE Computer Society, 1998.
[4] IEEE 1028-1997, IEEE Standard for Software Reviews, IEEE Computer Society, 1997.
[5] ISO/IEC/IEEE 29148:2011, Systems and Software Engineering – Life Cycle Processes – Requirements Engineering, ISO/IEC/IEEE, 2011.
[6] "Workshop Management System – POMEN." Accessed: Mar. 30, 2025. [Online]. Available: https://www.pomen.io/workshop-management-system/

3.0 General Architecture



1.1 Application Layer

1.1.1 Manage Profile (NUR ALYA SYAKIRAH BINTI NASARUDIN CB22141)



Pages [PKG-WMS-2025-101]

Class Name	Description
ProfileLogin	Login screen for Workshop Owners and Foremen to access their accounts.

ProfileRegistration	Registration form for new users to create a Workshop Owner or Foreman account.
ProfileEdit	Displaying the user's existing profile information and allowing users to update their personal details and workshop information.
ProfileChangePassword	Interface to change the user's password by verifying their old password and entering a new password.

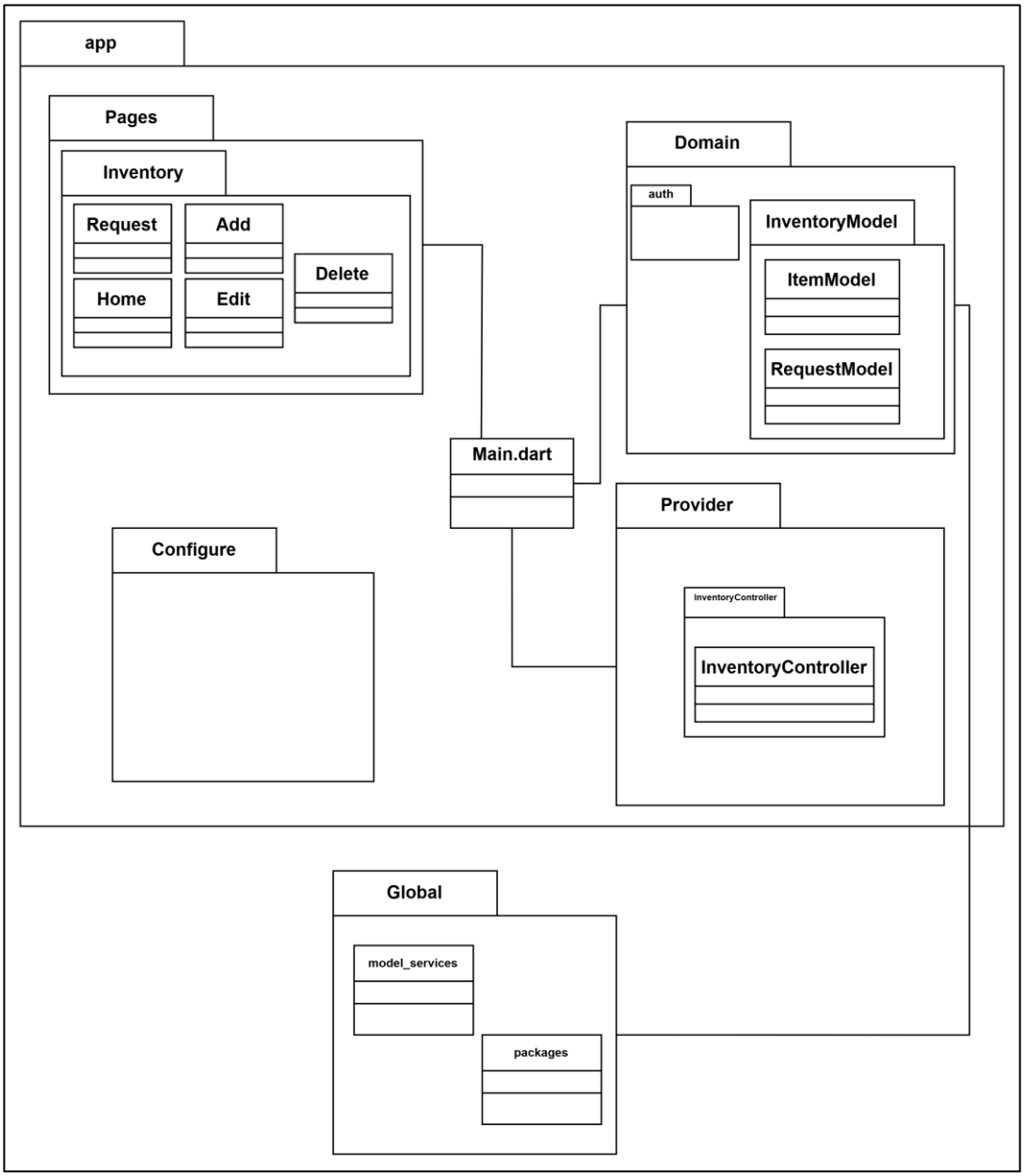
Domain [PKG-WMS-2025-102]

Class Name	Description
auth	Manages user authentication and session control.
ProfileModel	The model that handles the storage, retrieval and updating of user profile data from the database, ensuring data consistency and security.

Provider [PKG-WMS-2025-103]

Class Name	Description
ProfileController	Handles registration, login, update, validation and password change logic.

1.1.2 Manage Shop Inventory (Varshini Jagarajan, CB22139)



Pages [PKG-WMS-2025-201]

Class Name	Description
Home	The main page that shows the inventory list to the workshop owner.
Add	UI that enables the workshop owner to add a new inventory item.
Edit	UI display for modifying current inventory item information.
Delete	UI that allows the workshop owner to delete the existing item information.
Request	UI for the workshop owner to order inventory that is currently unavailable.

Domain [PKG-WMS-2025-202]

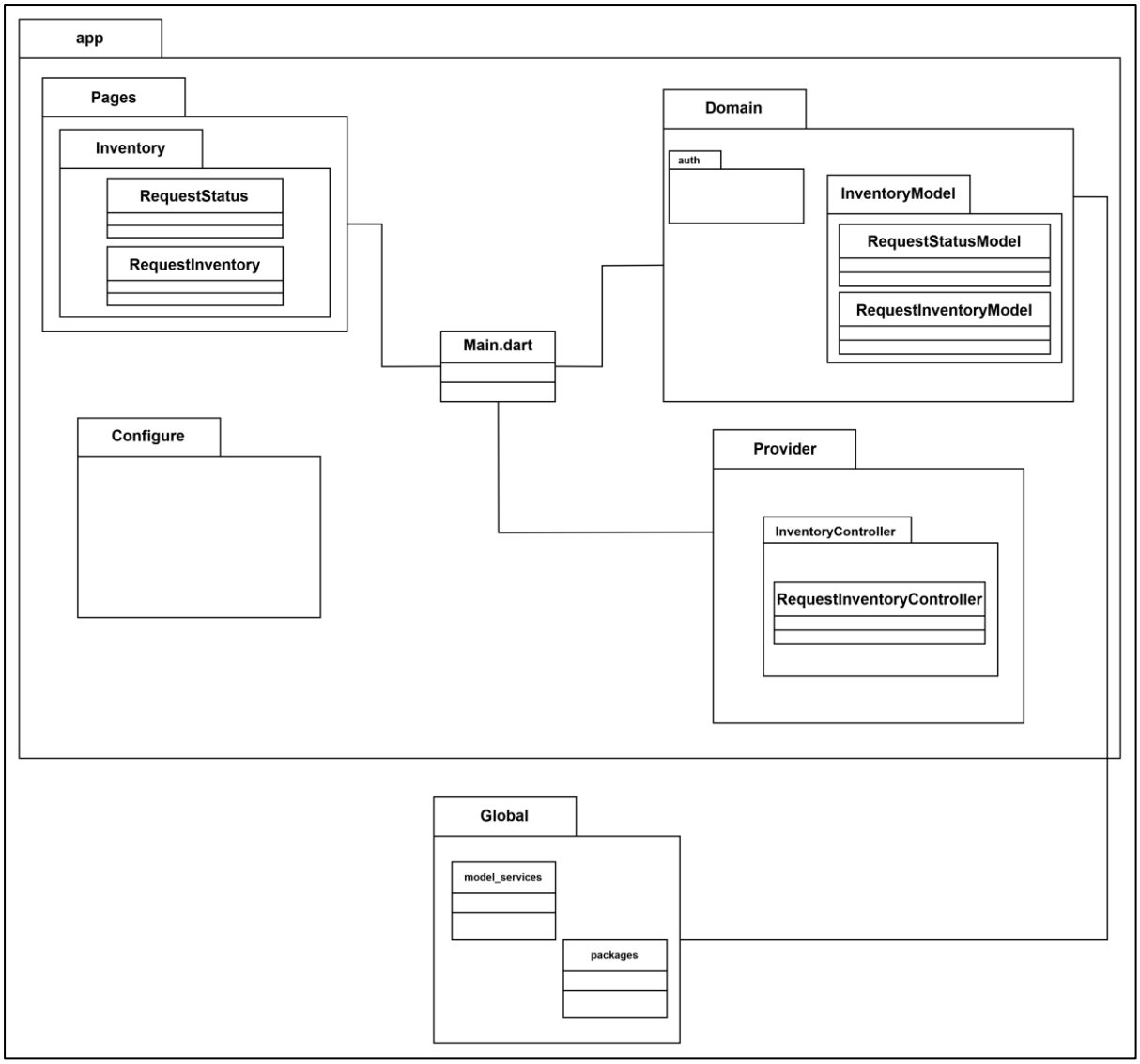
Class Name	Description
Auth	To preserve user authentication.
ItemModel	Represents an inventory item entity such as item name, quantity, and location.
RequestModel	Represents a request to add or restock inventory items.

Provider [PKG-WMS-2025-203]

Class Name	Description
InventoryController	Handles business logic and state related with inventory management, including CRUD operations.

1.1.3 Request Inventory (Varshini Jagarajan, CB22139)

Individual Package Diagram



Pages [PKG-WMS-2025-301]

Class Name	Description
RequestInventory	User interface screen that allows the workshop owner to create a new inventory request for items that are currently unavailable.
RequestStatus	UI screen showing the status such as pending, approved, or rejected, of all prior inventory requests submitted.

Domain [PKG-WMS-2025-302]

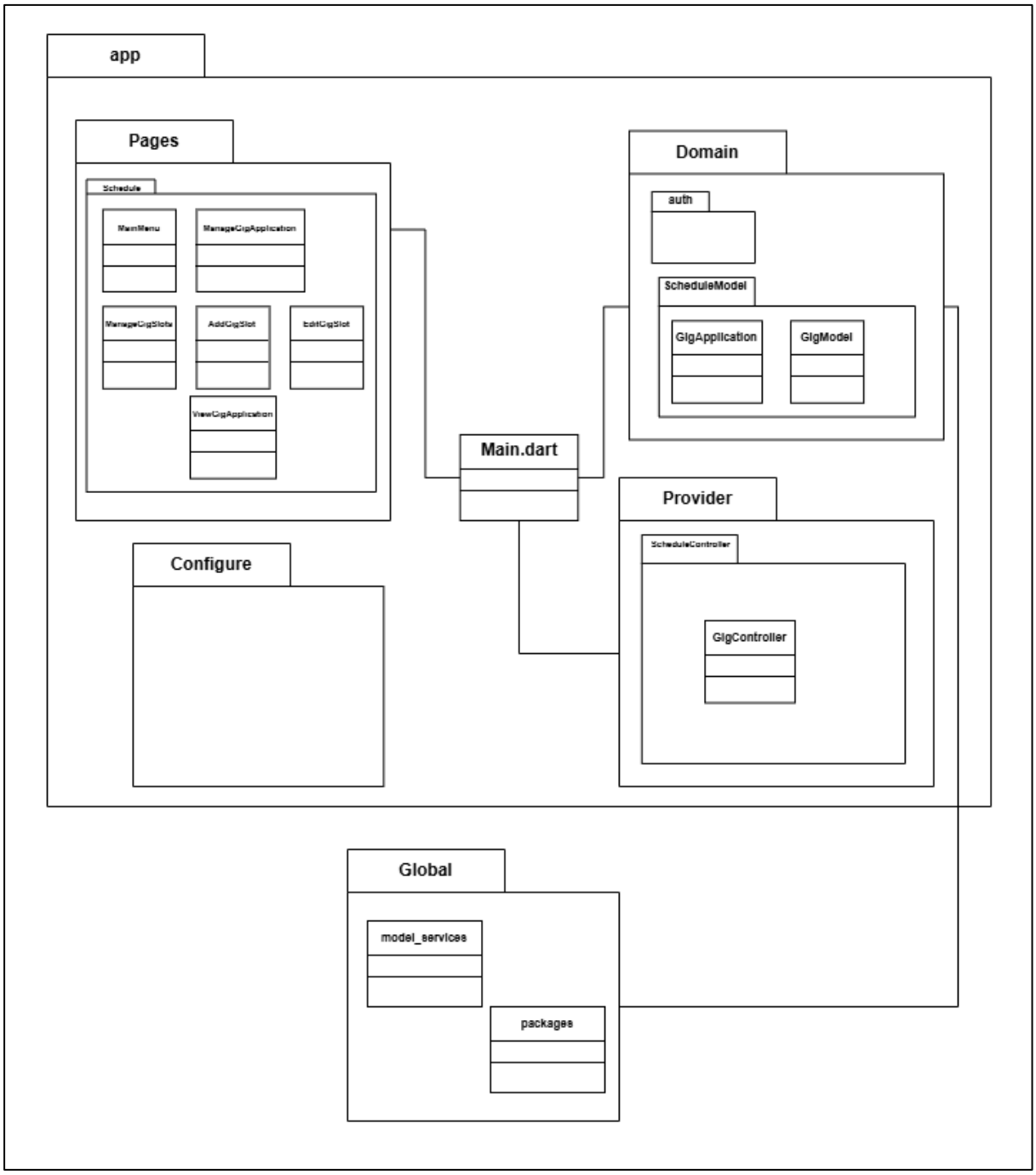
Class Name	Description
Auth	To preserve user authentication.

RequestStatusmodel	A model that indicates the status of an inventory request, such as "Pending," "Approved," or "Rejected."
RequestInventoryModel	Data model representing the inventory request made by the workshop owner, including item details and the date of the request.

Provider [PKG-WMS-2025-303]

Class Name	Description
RequestInventoryController	Manages the logic for submitting, verifying, and modifying the status of inventory requests; interacts with backend services.

1.1.4 Manage Foreman Schedule (Amala Karthigayan, CB23116)



Pages [PKG-WMS-2025-401]

Class Name	Description

MainMenu	Main menu for choosing between Manage Gig Slot and Manage Application
ManageGigSlots	Displays a list of all gig slots created by the workshop owner.
AddGigSlot	Form interface that allows the workshop owner to add new gig slots.
EditGigSlot	Form interface that allows editing of existing gig slot details.
ManageGigApplication	Displays a list of applications submitted by foremen for various gig slots.
ViewGigApplication	Detailed view of a specific foreman's application, including action buttons for approve/reject.

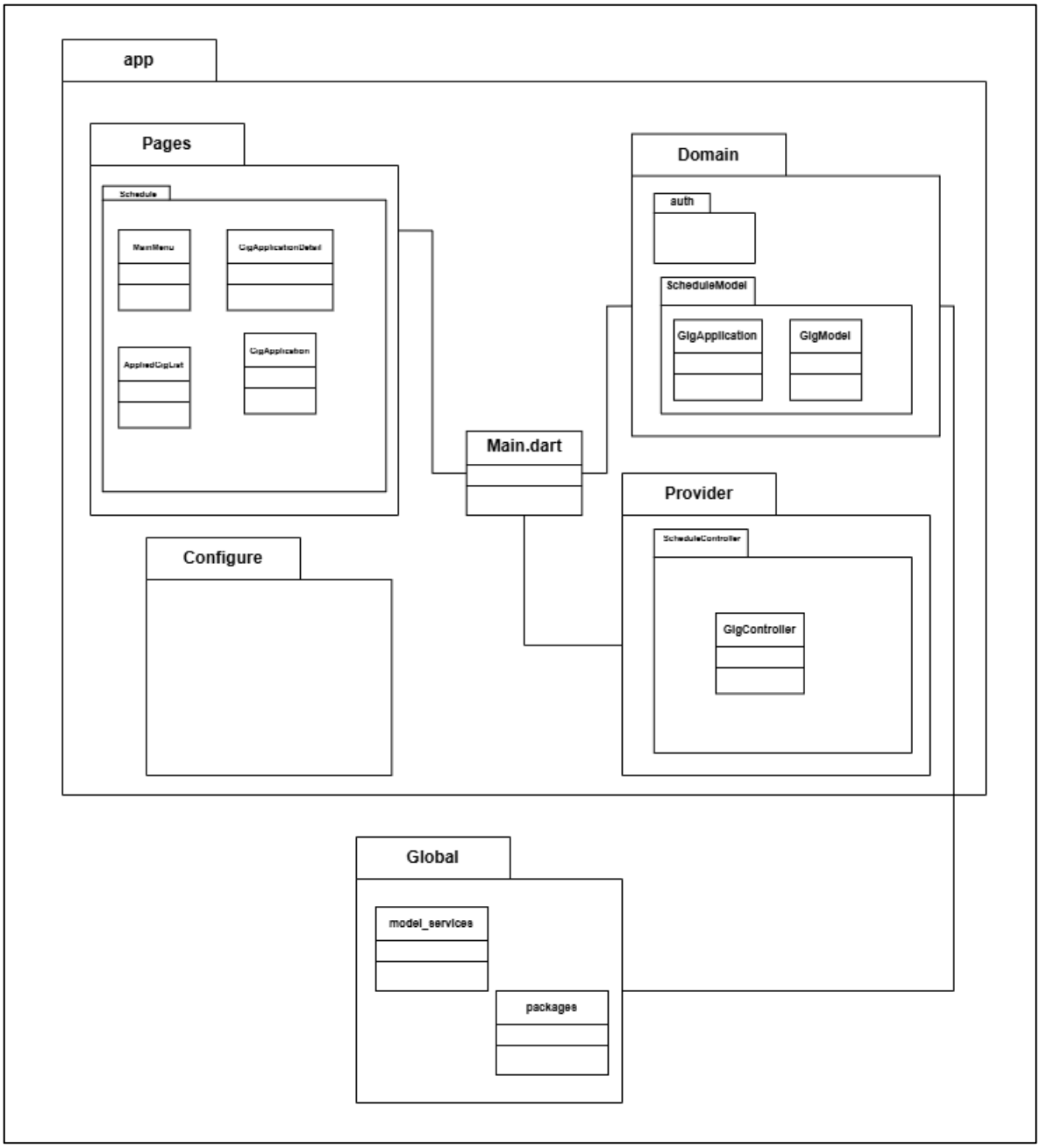
Domain [PKG-WMS-2025-402]

Class Name	Description
Auth	To preserve user authentication.
GigModel	Represents a gig slot with attributes such as date, time, location, and required foreman count.
GigApplication	Represents an application submitted by a foreman to a specific gig slot.

Provider [PKG-WMS-2025-403]

Class Name	Description
GigController	• Manages gig slot operations including add, edit, delete, and validation. • Handle gig approval and rejection logic. • Validates gig availability and notifications.

1.1.5 Select Available Schedule (Amala Karthigayan, CB23116)



Pages [PKG-WMS-2025-501]

Class Name	Description
MainMenu	Main menu for choosing between Manage Gig Slot and Manage Application.
GigApplicationInterface	For viewing available gig slots.
AppliedGigList	For listing gigs the foreman has applied to.
GigApplicationDetail	A detailed view per gig (optional)

Domain [PKG-WMS-2025-502]

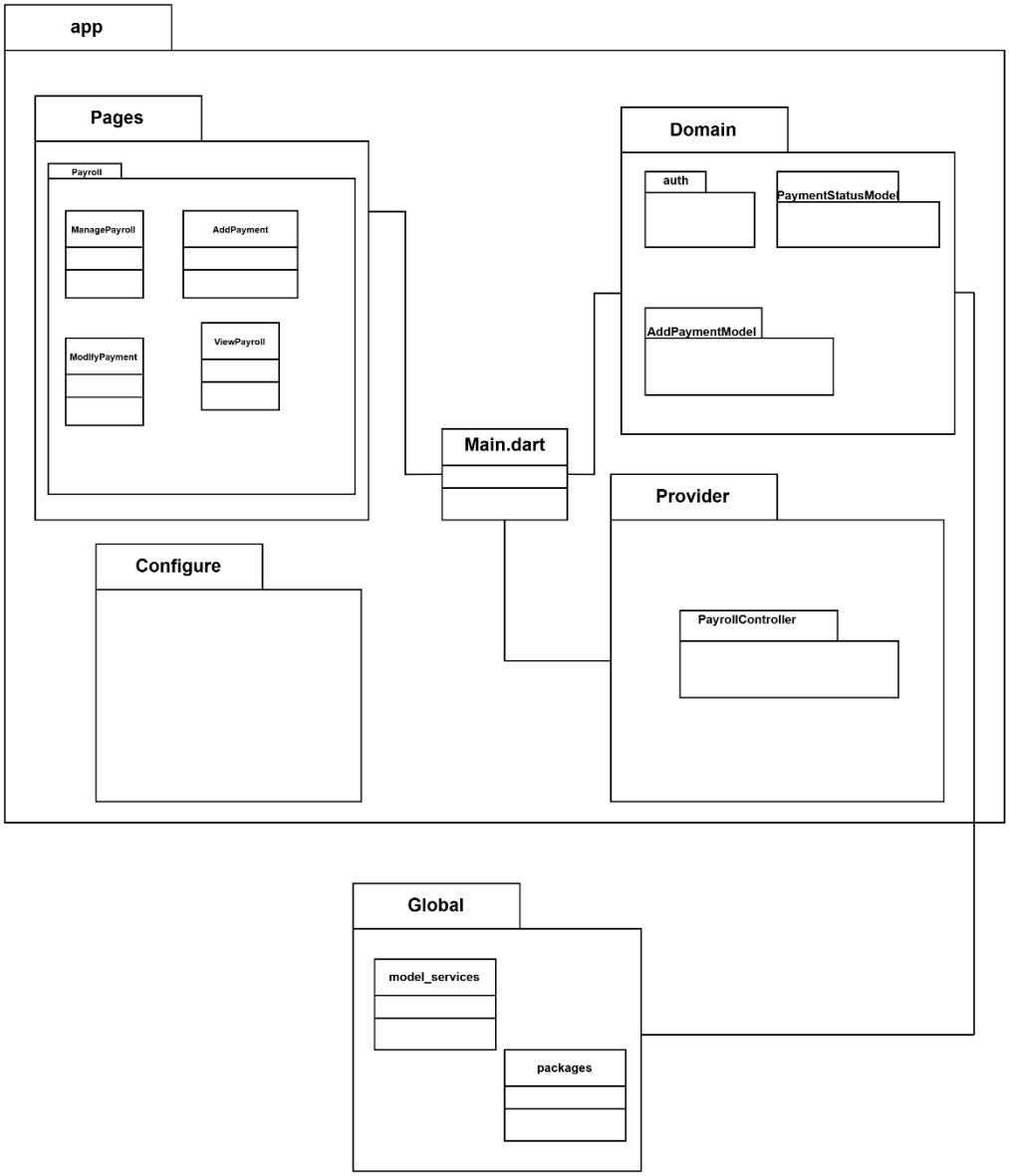
Class Name	Description
Auth	To preserve user authentication.
GigModel	Represents a gig slot with attributes such as date, time, location, and required foreman count.
GigApplication	Represents an application submitted by a foreman to a specific gig slot.

Provider [PKG-WMS-2025-503]

Class Name	Description
GigController	• Retrieves available gig slots and validates availability. • Handles applying and cancelling gigs. • Updates schedules upon approval or cancellation.

1.1.6 Manage Payroll (NUR NABIHAH FATINY BINTI ISMAIL CB23101)

Individual Package Diagram



Page [PKG-WMS-2025-601]

Class Name	Description
ManagePayroll	UI screen showing a list of Foremen with pending payments, and allows navigation to payment actions

AddPayment	Interactive form where the Workshop Owner fills in payment details with form validation.
ModifyPayment	To modify the payment amount before confirming the payment.
ViewPayroll	Read-only UI for Foremen to view payroll records and payment status.

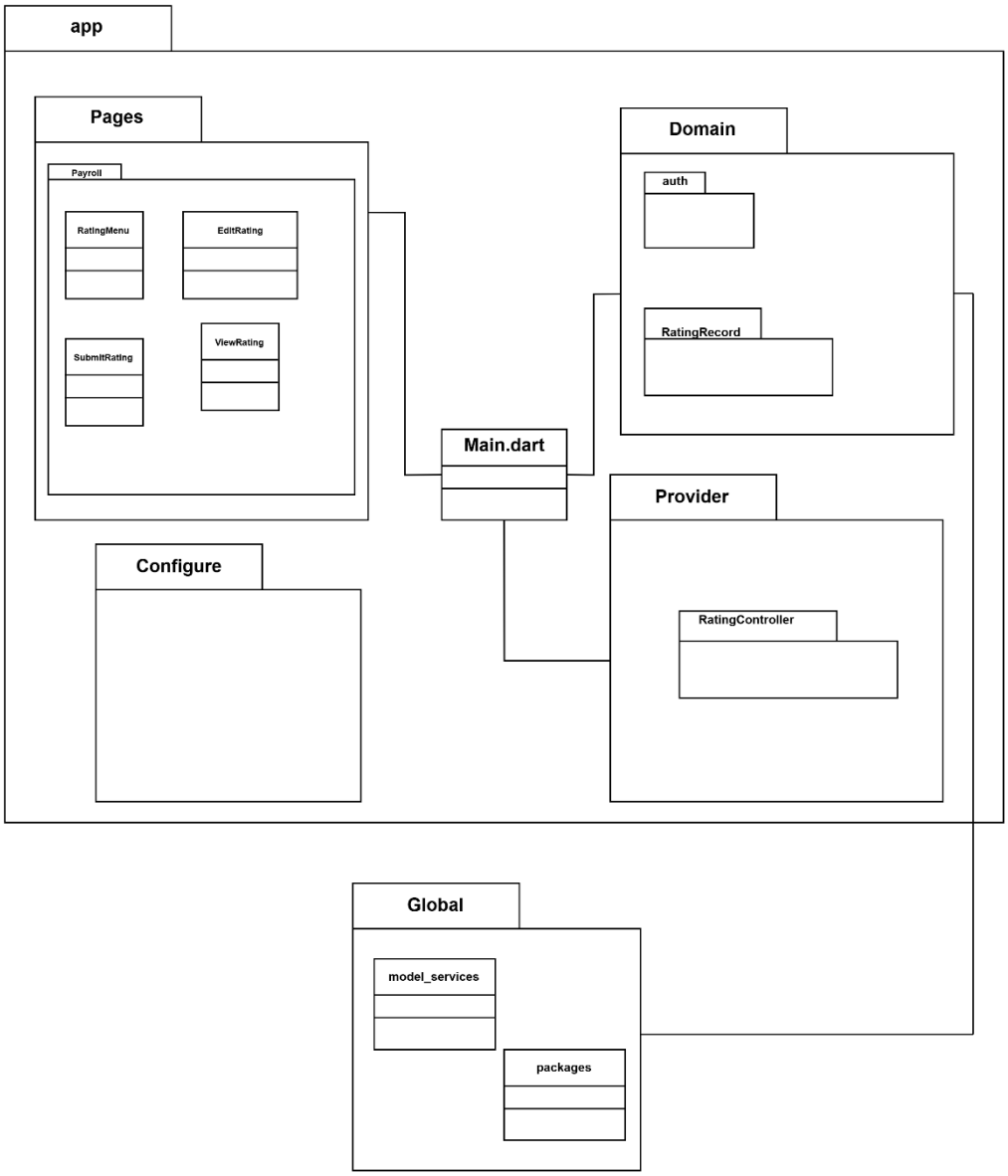
Domain [PKG-WMS-2025-602]

Class Name	Description
Auth	To preserve user authentication.
AddPaymentModel	Represents the data entered in the payment form before submission such as the details and date.
PaymentStatusModel	Model indicating the status of payment “Pending”, “Paid”, “Rejected”

Provider [PKG-WMS-2025-603]

Class Name	Description
PayrollController	Handles payroll logic. fetching pending payments, add payments, modifying amounts, and retrieving payment history. Interfaces between pages and backend services.

1.1.7 Manage Rating (NUR NABIHAH FATINY BINTI ISMAIL CB23101)



Page[PKG-WMS-2025-701]

Class Name	Description

RatingMenu	Displays all submitted/unsubmitted ratings and provides navigation to submit/edit options.
SubmitRating	UI form for the customer to submit a new rating and feedback.
ViewRating	Read-only UI for Workshop Owner and Foreman to view all submitted ratings.

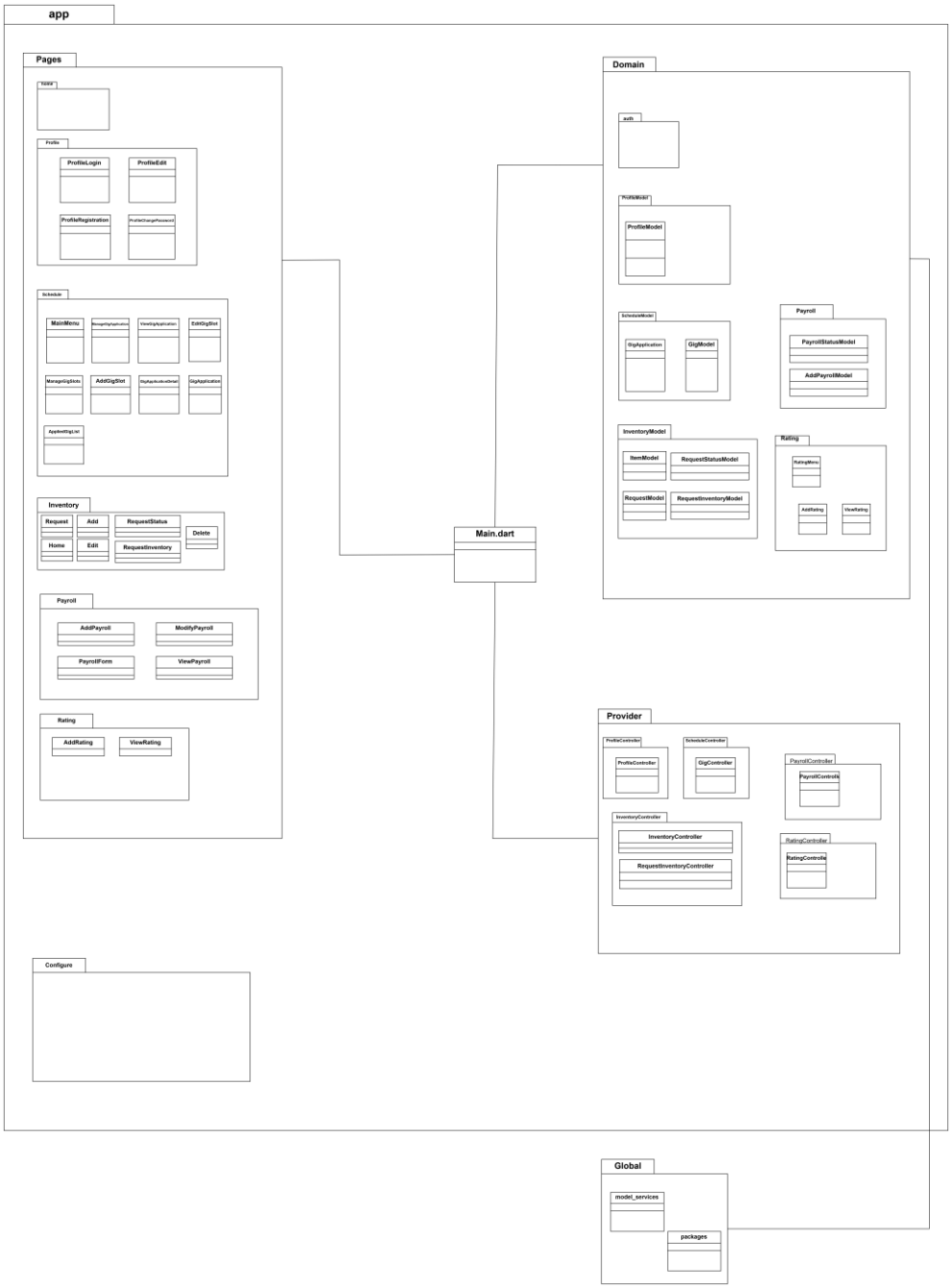
Domain [PKG-WMS-2025-702]

Class Name	Description
Auth	To preserve user authentication.
RatingRecord	Model class that represents the rating data. Handles basic data structure and validation if necessary. Comments and rating star

Provider[PKG-WMS-2025-703]

Class Name	Description
RatingController	Handles rating submission, editing, fetching all ratings, and notifies the UI of state changes.

3.2 Package Module



1.1 Data Dictionary

1.1.1 Manage Profile (NUR ALYA SYAKIRAH BINTI NASARUDIN CB22141)

Field Name	Description	Data Type	Constraint
RegistrationID	Unique registration ID for profile table	NUMBER	PK
UserID	Internal system identifier for the user profile	VARCHAR(10)	FK
Name	Full name of the user (Workshop Owner or Foreman)	VARCHAR(50)	
Email	Email address of the user, used for login authentication	VARCHAR(50)	UNIQUE
Contact	Contact number of the user	VARCHAR(15)	
Address	Residential or workshop address of the user	VARCHAR(100)	
Position	User's position (Workshop Owner or Foreman)	VARCHAR(30)	
Skills	User’s skills or specializations	VARCHAR(100)	
Experience	User’s working experience in years	NUMBER	
Password	Encrypted password for user authentication	VARCHAR(100)	

1.1.2 Manage Shop Inventory (Varshini Jagarajan, CB22139)

Field Name	Description	Data Type	Constraint
Item_ID	Unique identifier for each inventory item	VARCHAR (10)	PK

Item_Name	Name of the inventory item	VARCHAR (50)	
Category	Category to which the item belongs (e.g., tools, parts)	VARCHAR (30)	
Quantity	Number of items available in stock	INT	
Unit	Unit of measurement (e.g., pcs, boxes, liters)	VARCHAR (10)	
Storage_location	Storage location or shelf number	VARCHAR (50)	
Created_at	Date when the item was added to the inventory	DATE	
Updated_at	Date when the inventory record was last updated	DATE	

1.1.3 Request Inventory (Varshini Jagarajan, CB22139)

Field Name	Description	Data Type	Constraint
Request_ID	Unique identifier for each inventory request	VARCHAR (10)	PK
Item_ID	Foreign key referencing the requested item in inventory	VARCHAR (10)	FK
Request_Date	Date the inventory request was submitted	DATE	
Requested_Quantity	Quantity of item being requested	INT	
Status	Status of the request (e.g.,	VARCHAR (20)	

	Pending, Approved, Rejected)		
Status_date	Date the request status was last updated	DATE	

1.1.4 Manage Foreman Schedule (Amala Karthigayan, CB23116)

GigModel

Field Name	Description	Data Type	Constraint
gigId	Unique identifier for the gig slot	String	Primary Key, Auto-generated
date	Scheduled date of the gig	Date	Required, Must be in future
time	Scheduled time of the gig	Time	Required, 24-hour format (e.g., "14:00")
location	Location where the gig will take place	String	Required, Max 100 characters
jobDescription	Description of the job/task to be performed	String	Required, Max 255 characters
requiredForemenCount	Number of foremen required for the gig	Integer	Required, Minimum: 1
paymentDetails	Payment information or rate for the gig	Double(4,3)	Required
status	Status of the gig (e.g., Open, Filled)	String	Default: ‘Open’

GigApplication

Field Name	Description	Data Type	Constraint
applicationId	Unique identifier for each application	String	Primary Key, Auto-generated
gigId	ID of the gig the application is tied to	String	Foreign Key → GigModel.gigId
foremanId	ID of the applying foreman	String	Foreign Key → User.id
applicationDate	Date the application was submitted	DateTime	Not Null
status	Application status (Pending, Approved, Rejected)	String	Default: 'Pending'

1.1.5 Select Available Schedule (Amala Karthigayan, CB23116)

Same as Manage Foreman Schedule:

GigModel

Field Name	Description	Data Type	Constraint
gigId	Unique identifier for the gig slot	String	Primary Key, Auto-generated
date	Scheduled date of the gig	Date	Required, Must be in future
time	Scheduled time of the gig	Time	Required, 24-hour format (e.g., "14:00")
location	Location where the gig will take place	String	Required, Max 100 characters

jobDescription	Description of the job/task to be performed	String	Required, Max 255 characters
requiredForemenCount	Number of foremen required for the gig	Integer	Required, Minimum: 1
paymentDetails	Payment information or rate for the gig	Double(4,3)	Required, Format: text or currency string
status	Status of the gig (e.g., Open, Filled)	String	Default: ‘Open’

GigApplication

Field Name	Description	Data Type	Constraint
applicationId	Unique identifier for each application	String	Primary Key, Auto-generated
gigId	ID of the gig the application is tied to	String	Foreign Key → GigModel.gigId
foremanId	ID of the applying foreman	String	Foreign Key → User.id
applicationDate	Date the application was submitted	DateTime	Not Null
status	Application status (Pending, Approved, Rejected)	String	Default: 'Pending'

1.1.6 Manage Payroll (NUR NABIHAH FATINY BINTI ISMAIL CB23101)

Field Name	Description	Data Type	Constraint

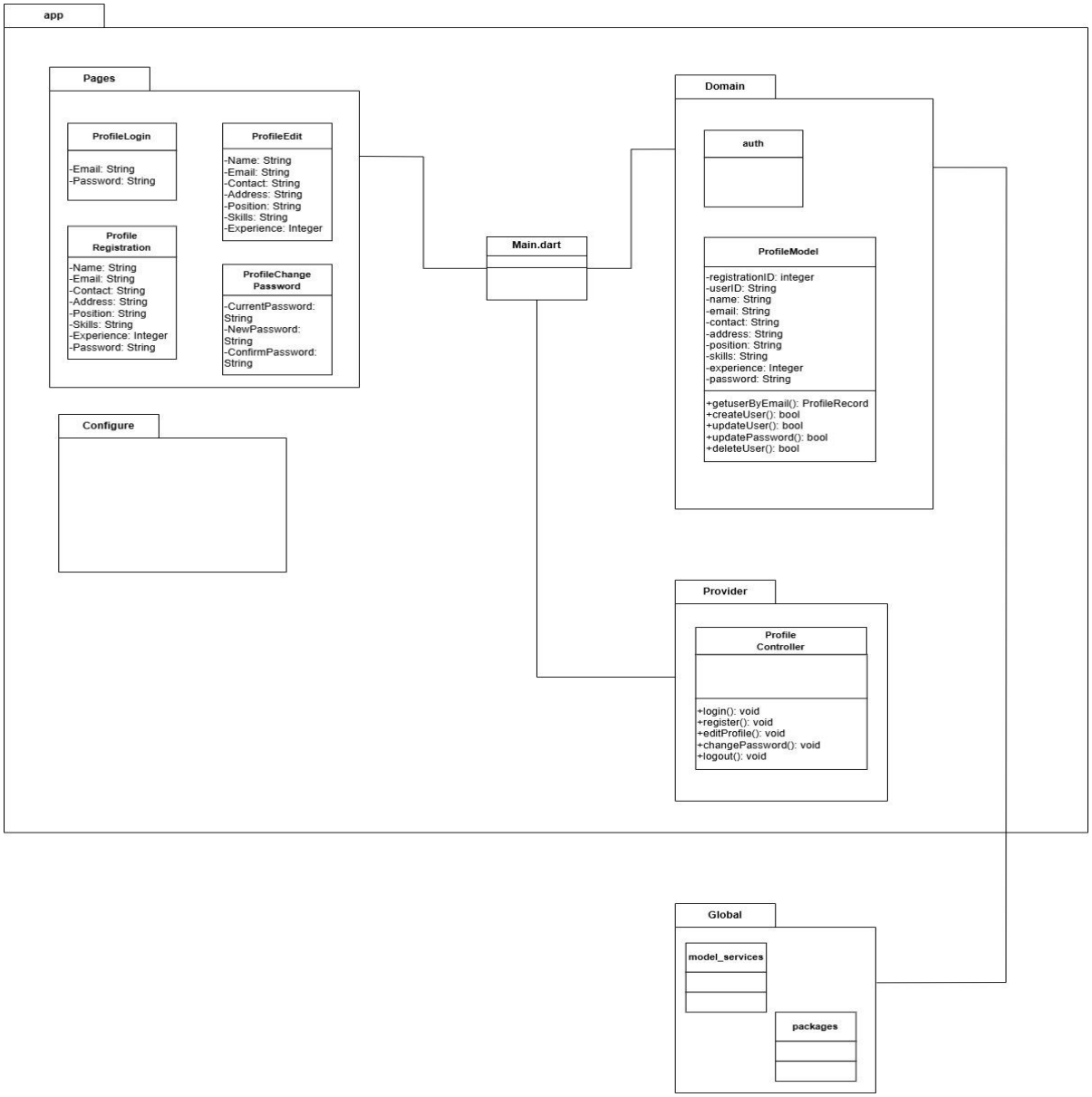
Payroll_ID	Unique ID for each payroll record	VARCHAR(10)	PK
Foreman_ID	ID of the Foreman	VARCHAR(8)	FK
Foreman_Name	Name of the Foreman	VARCHAR(30)	
Bank_Name	Name of the bank	VARCHAR(30)	
Account_Number	Foreman's bank account number	VARCHAR(20)	
Payment_Amount	Amount to be paid	INT	
Payment_Date	Date of the payment	DATE	
Payment_Reference	Reference number or note for payment	VARCHAR(30)	
Payment_Status	Status of payment	VARCHAR(15)	

1.1.7 Manage Rating (NUR NABIHAH FATINY BINTI ISMAIL CB23101)

Field Name	Description	Data Type	Constraint
Rating_ID	Unique ID for each rating	VARCHAR(10)	PK
Customer_ID	ID of the Customer who submitted the rating	VARCHAR(8)	FK
Workshop_ID	ID of the related Workshop	VARCHAR(10)	FK
Star_Rating	Star rating given by the customer (1 to 5)	INT	
Feedback_Comment	Customer's comment on the service	VARCHAR(255)	
Rating_Date	Date the rating was submitted	DATE	

2.0 Detail Design

4.1 Manage Profile (NUR ALYA SYAKIRAH BINTI NASARUDIN CB22141)



ProfileLogin [CLS-WMS-2025-101-01]

Class Type	Boundary class

Responsibility	This class is responsible for displaying the login interface for users to enter their email and password.	
Attributes	Attributes Name	Attributes Type
	Email Password	String String
Methods	Method Name	Description
	Not applicable	Not applicable
Algorithm	Not applicable	

ProfileRegistration [CLS-WMS-2025-102-01]

Class Type	Boundary Class	
Responsibility	This class is responsible for displaying the registration form to create a new profile account with detailed information.	
Attributes	Attributes Name	Attributes Type
	Name Email Contact Address Position Skills Experience Password	String String String String String String Integer String
Methods	Method Name	Description
	Not applicable	Not applicable
Algorithm	Not applicable	

ProfileEdit [CLS-WMS-2025-103-01]

Class Type	Boundary Class

Responsibility	This class is responsible for displaying the profile edit form allowing users to update their personal and professional information.	
Attributes	Attributes Name	Attributes Type
	Name Email Contact Address Position Skills Experience	String String String String String String Integer
Methods	Method Name	Description
	Not applicable	Not applicable
Algorithm	Not applicable	

ProfileChangePassword [CLS-WMS-2025-104-01]

Class Type	Boundary Class	
Responsibility	This class is responsible for displaying the change password form where users can update their existing password securely.	
Attributes	Attributes Name	Attributes Type
	CurrentPassword NewPassword ConfirmPassword	String String String
Methods	Method Name	Description
	Not applicable	Not applicable
Algorithm	Not applicable	

ProfileController [CLS-WMS-2025-105-01]

Class Type	Control Class

<table>
<tr><td>Responsibility</td><td colspan="2">This controller manages all operations related to user profiles including login, registration, profile editing and password changes.</td></tr>
<tr><td rowspan="2">Attributes</td><td>Attributes Name</td><td>Attributes Type</td></tr>
<tr><td>Name
Email
Contact
Address
Position
Skills
Experience
Password</td><td>String
String
String
String
String
String
Integer
String</td></tr>
<tr><td rowspan="2">Methods</td><td>Method Name</td><td>Description</td></tr>
<tr><td>login(): void
register(): void
editProfile(): void
changePassworrd
(): void
logout(): void</td><td>Handles user authentication
Registers a new user profile
Updates user profile information
Updates user account password

Logs the user out of the system</td></tr>
<tr><td>Algorithm</td><td colspan="2">login()
START
User submits login form
Check if email and password match existing profile
If valid, redirect to dashboard
Else, display error message
END

register()
START
User submits registration form
Validate all fields
Create a new ProfileRecord</td></tr>
</table>

	Redirect to login page END editProfile() START User accesses edit profile page Load existing profile data Allow user to modify data Save updated data into ProfileRecord END changePassword() START User submits change password form Validate current password Update password to new password in ProfileRecord Confirm password change to user END logout() START User clicks logout Clear user session Redirect to login page END

ProfileModel [CLS-WMS-2025-106-01]

Class Type	Entity Class
Responsibility	This class is responsible for interacting with the database to manage user profile data such as creating, updating, deleting and retrieving user information.

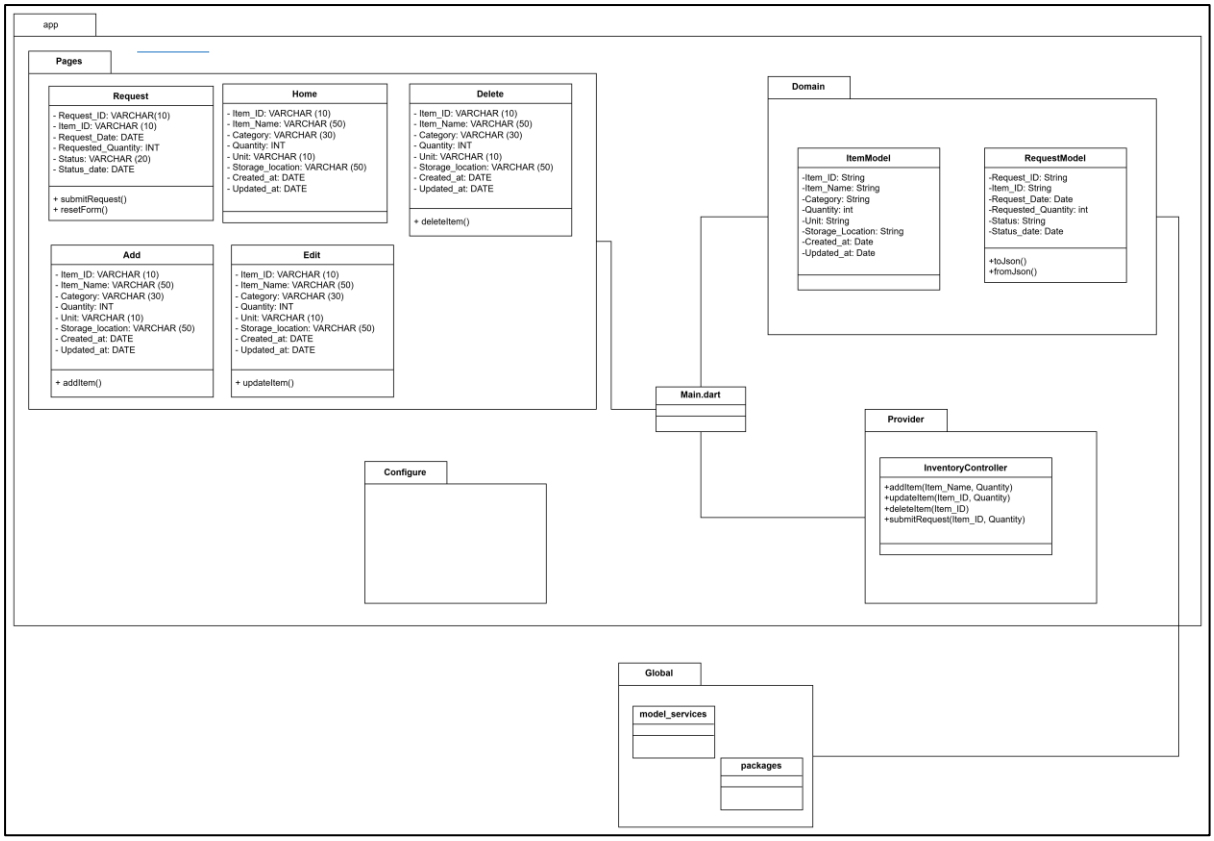
<table>
<tr><td rowspan="11">Attributes</td><td>Attributes Name</td><td>Attributes Type</td></tr>
<tr><td>RegistrationID</td><td>Integer</td></tr>
<tr><td>UserID</td><td>String</td></tr>
<tr><td>Name</td><td>String</td></tr>
<tr><td>Email</td><td>String</td></tr>
<tr><td>Contact</td><td>String</td></tr>
<tr><td>Address</td><td>String</td></tr>
<tr><td>Position</td><td>String</td></tr>
<tr><td>Skills</td><td>String</td></tr>
<tr><td>Experience</td><td>Integer</td></tr>
<tr><td>Password</td><td>String</td></tr>
<tr><td rowspan="6">Methods</td><td>Method Name</td><td>Description</td></tr>
<tr><td>GetUserByEmail():ProfileRecord</td><td>Retrieves a user profile based on email address</td></tr>
<tr><td>createUser(): bool</td><td>Creates a new user profile in the database</td></tr>
<tr><td>updateUser(): bool</td><td>Updates an existing user profile by ID</td></tr>
<tr><td>UpdatePassword(): bool</td><td>Updates password for a user</td></tr>
<tr><td>DeleteUser(): bool</td><td>Deletes a user profile by ID</td></tr>
<tr><td>Algorithm</td><td colspan="2">GetUserByEmail()
START
Receive email input
Query the database for a matching email
Return the user profile record if found
END

CreateUser()
START
Receive user profile data
Insert data into the profile table</td></tr>
</table>

	Return true if successful, else false END UpdateUser() START Receive ID and updated data Update the profile record in the database Return true if successful END UpdatePassword() START Receive user ID and new password Encrypt the new password Update password in the database Return true if successful END DeleteUser() START Receive ID of the profile to delete Delete the record from database Return true if successful END

4.2 Manage Shop Inventory (Varshini Jagarajan, CB22139)

Package Diagram with classes including attributes and function



Home [CLS-WMS-2025-201-01]

Class Type	Boundary class	
Responsibility	An interface to navigate to Inventory actions.	
Attributes	Attributes Name	Attributes Type
	Item_ID	VARCHAR (10)
	Item_Name	VARCHAR (50)
	Category	VARCHAR (30)
	Quantity	INT
	Unit	VARCHAR (10)
	Storage_Location	VARCHAR (10)
	Created_at	DATE
	Updated_at	DATE
Methods	Method Name	Description
	None	None
Algorithm	None	

Add [CLS-WMS-2025-202-01]

Class Type	Boundary class	
Responsibility	An interface to add a new inventory item.	
Attributes	Attributes Name	Attributes Type
	Item_ID	VARCHAR (10)
	Item_Name	VARCHAR (50)
	Category	VARCHAR (30)
	Quantity	INT
	Unit	VARCHAR (10)
	Storage_Location	VARCHAR (10)
	Created_at	DATE
	Updated_at	DATE
Methods	Method Name	Description
	addItem()	Adds new item to controller.
Algorithm	addItem(): if Item_Name is not empty and Quantity> 0 call InventoryController.addItem(Item_Name, Quantity) Else show error	

Edit [CLS-WMS-2025-203-01]

Class Type	Boundary class	
Responsibility	An interface to edit existing inventory.	
Attributes	Attributes Name	Attributes Type
	Item_ID	VARCHAR (10)
	Item_Name	VARCHAR (50)
	Category	VARCHAR (30)
	Quantity	INT
	Unit	VARCHAR (10)
	Storage_Location	VARCHAR (10)
	Created_at	DATE

	Updated_at	DATE
Methods	Method Name	Description
	updateItem()	Updates selected inventory.
Algorithm	updateItem(): call InventoryController.updateItem(Item_ID, Quantity)	

Delete [CLS-WMS-2025-204-01]

Class Type	Boundary class	
Responsibility	An interface to delete existing inventory.	
Attributes	Attributes Name	Attributes Type
	Item_ID	VARCHAR (10)
	Item_Name	VARCHAR (50)
	Category	VARCHAR (30)
	Quantity	INT
	Unit	VARCHAR (10)
	Storage_Location	VARCHAR (10)
	Created_at	DATE
	Updated_at	DATE
Methods	Method Name	Description
	deleteItem()	Deletes item from inventory.
Algorithm	deleteItem(): call InventoryController.deleteItem(Item_ID)	

Request [CLS-WMS-2025-205-01]

Class Type	Boundary class	
Responsibility	An interface for shop staff to make inventory requests.	
Attributes	Attributes Name	Attributes Type
	Item_ID	VARCHAR (10)
	Request_ID	VARCHAR (10)

	Request_Date Requested_Quantity Status Status_date	DATE INT VARCHAR (20) DATE
Methods	Method Name	Description
	submitRequest() resetForm()	Submits the request form to controller. Clears form fields.
Algorithm	submitRequest(): if Requested_Quantity > 0 and Item_ID is valid call InventoryController.handleRequest(Item_ID, Requested_Quantity) else show error	

ItemModel [CLS-WMS-2025-206-01]

Class Type	Entity class	
Responsibility	Represents inventory item data.	
Attributes	Attributes Name	Attributes Type
	Item_ID Item_Name Category Quantity Unit Storage_Location Created_at Updated_at	VARCHAR (10) VARCHAR (50) VARCHAR (30) INT VARCHAR (10) VARCHAR (10) DATE DATE
Methods	Method Name	Description
	None	None
Algorithm	None	

RequestModel [CLS-WMS-2025-207-01]

Class Type	Entity class	
Responsibility	Represents the structure of a shop inventory request.	
Attributes	Attributes Name	Attributes Type
	Request_ID Item_ID Request_Date Requested_Quantity Status Status_date	VARCHAR (10) VARCHAR (10) DATE INT VARCHAR (20) DATE
Methods	Method Name	Description
	toJson() fromJson()	Converts request to JSON for API Parses request data from JSON
Algorithm	None	

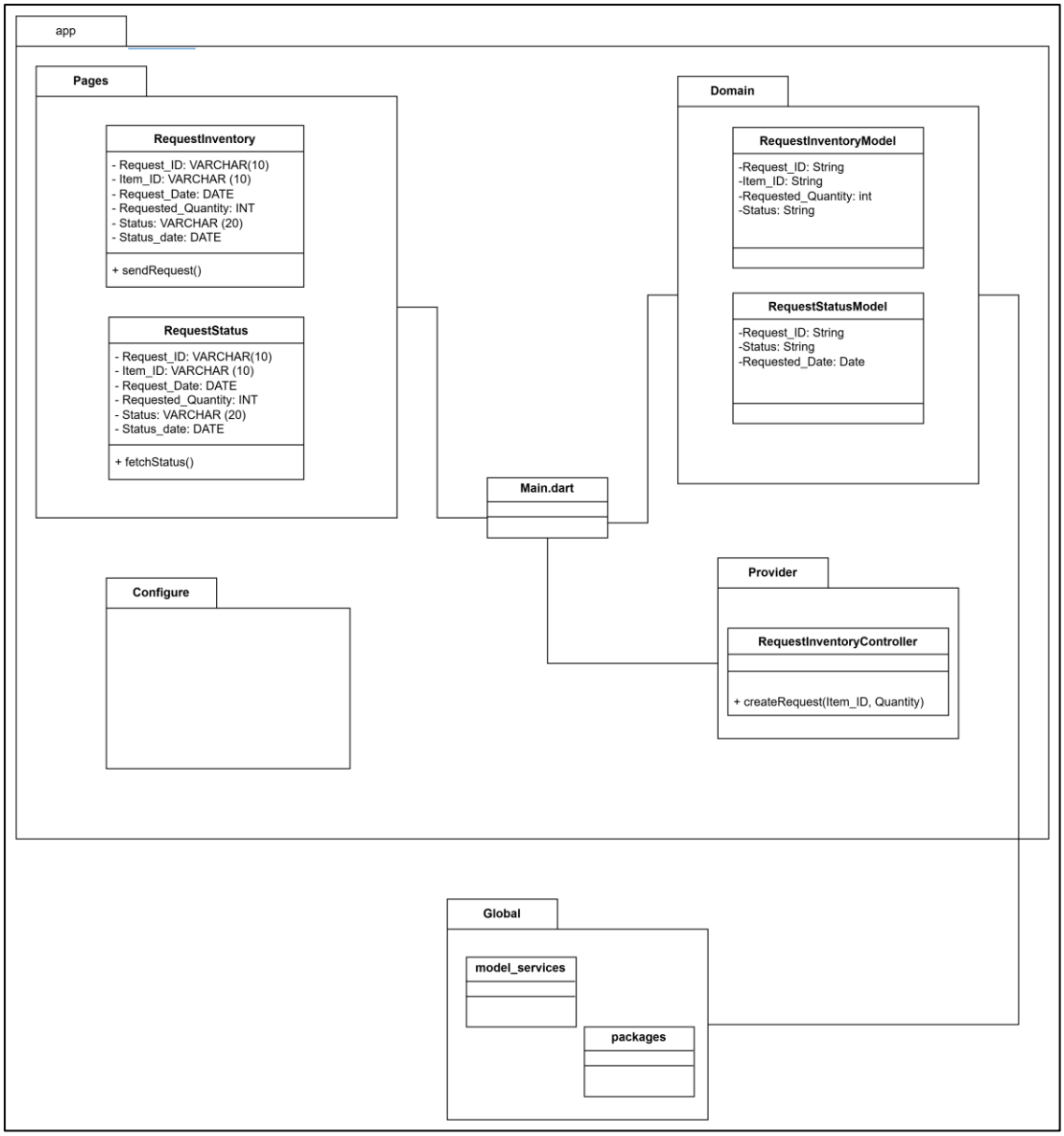
InventoryController [CLS-WMS-2025-208-01]

Class Type	**Control class**	
Responsibility	Business logic for inventory operations.	
Attributes	Attributes Name	Attributes Type
	None	None
Methods	Method Name	Description
	addItem() updateItem() deleteItem() submitRequest()	- - - -
Algorithm	addItem(Item_Name, Quantity): item = new ItemModel(Item_Name, Quantity)	

	model_services.save(item) updateItem(Item_ID, Quantity): item = model_services.findById(Item_ID) item.Quantity = Quantity model_services.update(item) deleteItem(Item_ID): model_services.delete(Item_ID) submitRequest(Item_ID, Quantity): request = new RequestModel(Item_ID, Quantity) model_services.save(request)

4.3 Request Inventory (Varshini Jagarajan, CB22139)

Package Diagram with classes including attributes and function



RequestInventory [CLS-WMS-2025-301-01]

Class Type	Boundary class
Responsibility	An interface to request items.

Attributes	Attributes Name	Attributes Type
	Request_ID Item_ID Request_Date Requested_Quantity Status Status_date	VARCHAR (10) VARCHAR (10) DATE INT VARCHAR (20) DATE
Methods	Method Name	Description
	sendRequest()	Submit item request.
Algorithm	sendRequest(): if Requested_Quantity > 0 call RequestInventoryController.createRequest(Item_ID, Requested_Quantity) else show error	

RequestStatus [CLS-WMS-2025-302-01]

Class Type	**B**oundary class	
Responsibility	An interface to show request history and status.	
Attributes	Request_ID Item_ID Request_Date Requested_Quantity Status Status_date	VARCHAR (10) VARCHAR (10) DATE INT VARCHAR (20) DATE
	Method Name	Description
Methods	Method Name	Description
	fetchStatus()	Gets list of previous requests.
Algorithm	None	

RequestInventoryModel [CLS-WMS-2025-303-01]

Class Type	Entity class	
Responsibility	Data structure for item requests.	
Attributes	Attributes Name	Attributes Type
	Request_ID Item_ID Requested_Quantity Status	VARCHAR (10) VARCHAR (10) INT VARCHAR (20)
Methods	Method Name	Description
	None	None
Algorithm	None	

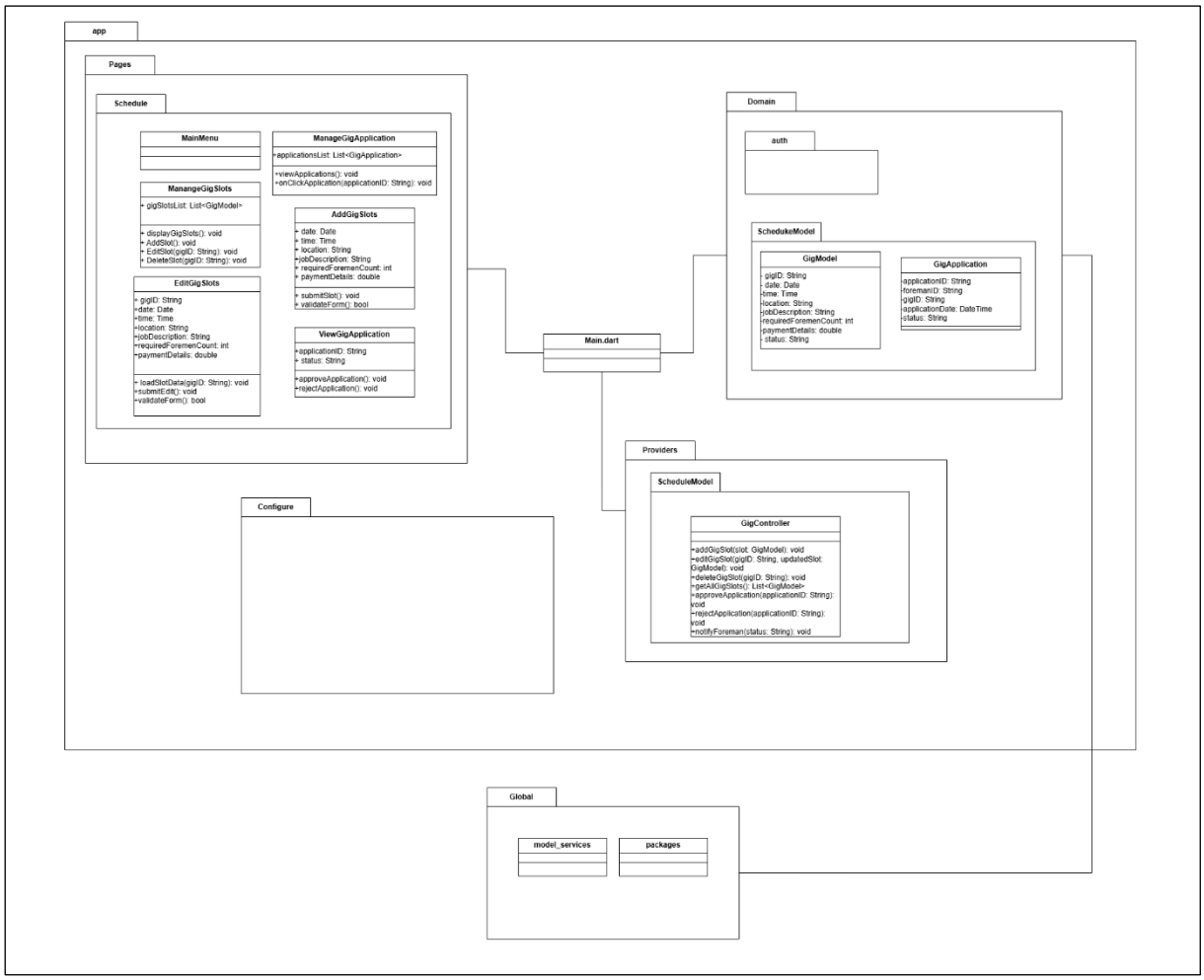
RequestStatusModel [CLS-WMS-2025-304-01]

Class Type	Entity class	
Responsibility	Keeps history of status updates.	
Attributes	Attributes Name	Attributes Type
	Request_ID Status Requested_Date	VARCHAR (10) VARCHAR (20) DATE
Methods	Method Name	Description
	None	None
Algorithm	None	

RequestInventoryController [CLS-WMS-2025-305-01]

Class Type	Control class
Responsibility	Handles item request logic.

Attributes	Attributes Name	Attributes Type
	None	None
Methods	Method Name	Description
	createRequest(Item_ID, Quantity)	-
Algorithm	createRequest(Item_ID, Quantity): request = new RequestInventoryModel(Item_ID, Quantity) model_services.save(request)	



4.4 Manage Foreman Schedule (Amala Karthigayan, CB23116)

Link:

https://drive.google.com/file/d/1fLMB7EgB9IJ6OC8IHgfWLZT6tmtvXZZb/view?usp=sharing

ManageGigSlots [CLS-WMS-2025-401-01]

Class Type	Boundary class	
Responsibility	UI to display and manage the list of created gig slots	
Attributes	Attributes Name	Attributes Type
	gigSlotsList	List<GigModel>
Methods	Method Name	Description
	displayGigSlots()	Display the list of all gig slots
	AddSlot()	Navigates to AddGigSlot form
	EditSlot (gigID: String)	Navigates to EditGigSlot with gig ID
	DeleteSlot(gigID: String)	Deletes the selected gig slot
Algorithm	**displayGigSlots()** START displayGigSlots() is called Call GigController.getAllGigSlots() Assign result to gigSlotsList Display gigSlotsList in the UI END **AddSlot()** START AddSlot() is triggered	

	Navigate to AddGigSlot interface END **EditSlot(gigID)** START EditSlot(slotID) is triggered Pass gigID to EditGigSlot page Load corresponding data from gigSlotsList or GigController Display data in editable form END **DeleteSlot(gigID)** START DeleteSlot(gigID) is triggered Prompt user to confirm deletion If confirmed: Call GigController.deleteGigSlot(gigID) Refresh gigSlotsList using displayGigSlots() END

AddGigSlot [CLS-WMS-2025-401-02]

Class Type	Boundary Class	
Responsibility	UI for adding a new gig slot via a form	
Attributes	Attributes Name	Attributes Type
	date	Date
	time	Time
	location	String
	jobDescription	String
	requiredForemenCount	int

	paymentDetails	double
Methods	Method Name	Description
	submitSlot()	Submits the new slot to the controller
	validateForm()	Checks if all required fields are filled

Algorithm	**submitSlot()**
	
	START
	submitSlot() is called
	Call validateForm()
	IF form is valid THEN
	Create GigModel object with form data
	Call GigController.addGigSlot()
	Show success message
	ELSE
	Show validation error message
	END
	
	validateForm()
	
	START
	validateForm() is called
	IF any field is null OR empty THEN
	RETURN false
	IF requiredForemenCount <= 0 THEN
	RETURN false
	IF paymentDetails < 0 THEN
	RETURN false
	RETURN true
	END	

EditGigSlot [CLS-WMS-2025-401-03]

Class Type	Boundary Class	
Responsibility	UI for editing an existing gig slot	
Attributes	Attributes Name	Attributes Type
	gigID	String
	date	Date

	time	Time
	location	String
	jobDescription	String
	requiredForemenCount	int
	paymentDetails	double
Methods	Method Name	Description
	loadSlotData()	Loads existing gig slot details
	submitEdit()	Submits edited slot to controller
	validateForm()	Validates form inputs before submitting

Algorithm	**loadSlotData()**
	START
	loadSlotData() is called
	Fetch gig slot data using gigID
	Populate UI form with retrieved data
	END
	
	submitEdit()
	START
	submitEdit() is called
	Call validateForm()
	IF form is valid THEN
	Create updated GigModel object with form data
	Call GigController.editGigSlot(gigID, updatedSlot)
	Show success message
	ELSE
	Show validation error message
	END
	
	validateForm()
	START
	validateForm() is called
	IF any field is null OR empty THEN
	RETURN false
	IF requiredForemenCount <= 0 THEN
	RETURN false
	IF paymentDetails < 0 THEN
	RETURN false	

	RETURN true END

ManageGigApplication [CLS-WMS-2025-401-04]

Class Type	Boundary Class	
Responsibility	Displays all applications submitted by foremen	
Attributes	Attributes Name	Attributes Type
	applicationsList	List<GigApplication>
Methods	Method Name	Description
	viewApplications()	Shows all received applications
	onClickApplication(applicationID)	Opens application details for action
Algorithm	**viewApplications()** START viewApplications() is called Fetch all applications from data source Populate applicationsList Display applications on the UI END **onClickApplication(applicationID)** START onClickApplication(applicationID) is called	

	Navigate to ViewGigApplication interface Pass selected applicationID to ViewGigApplication END

ViewGigApplication [CLS-WMS-2025-401-05]

Class Type	Boundary class	
Responsibility	Displays a detailed view of a selected application	
Attributes	Attributes Name	Attributes Type
	applicationID	String
	status	String
Methods	Method Name	Description
	approveApplication()	Approves the selected application
	rejectApplication()	Rejects the selected application
Algorithm	**approveApplication()** START approveApplication() is called Fetch application using applicationID Set status to "Approved" Save changes Call GigController.approveApplication(applicationID) END **rejectApplication()** START rejectApplication() is called Fetch application using applicationID Set status to "Rejected" Save changes Call GigController.rejectApplication(applicationID)	

	END

GigModel [CLS-WMS-2025-402-01]

Class Type	Entity Class	
Responsibility	Represents the data structure of a gig slot	
Attributes	Attributes Name	Attributes Type
	gigID	String
	date	Date
	time	Time
	location	String
	jobDescription	String
	requiredForemenCount	int
	paymentDetails	double
	status	String
Methods	Method Name	Description
	Not Applicable	
Algorithm	Not Applicable	

GigApplication [CLS-WMS-2025-402-02]

Class Type	Entity class	
Responsibility	Stores foreman application details for gig slots	
Attributes	Attributes Name	Attributes Type

	applicationID	String
	foremanID	String
	slotID	String
	applicationDate	DateTime
	status	String
Methods	Method Name	Description
	Not Applicable	
Algorithm	Not Applicable	

GigController [CLS-WMS-2025-403-01]

Class Type	Control class	
Responsibility	Handles all logic for managing gig slots and applications	
Attributes	Attributes Name	Attributes Type
	Not Applicable	
Methods	Method Name	Description
	addGigSlot(slot)	Adds a new gig slot to the system
	editGigSlot(gigID, updatedSlot)	Updates an existing gig slot
	deleteGigSlot(gigID)	Deletes a gig slot from the system
	getAllGigSlots()	Returns a list of all gig slots
	approveApplication(applicationID)	Changes status to "Approved" and notifies foreman
	rejectApplication(applicationID)	Changes status to "Rejected" and notifies foreman
	notifyForeman(status)	Sends notification about the status change
Algorithm	**addGigSlot(slot)** START addGigSlot(slot) is called Validate slot data Save slot to database	

END

editGigSlot(gigID, updatedSlot)

START
editGigSlot(slotID, updatedSlot) is called
Find existing slot using slotID
Update slot data with updatedSlot values
Save changes
END

deleteGigSlot(gigID)

START
deleteGigSlot(slotID) is called
Locate gig slot using slotID
Delete slot from database
END

getAllGigSlots()

START
getAllGigSlots() is called
Query and retrieve all gig slots from database
RETURN list of slots
END

approveApplication(applicationID)

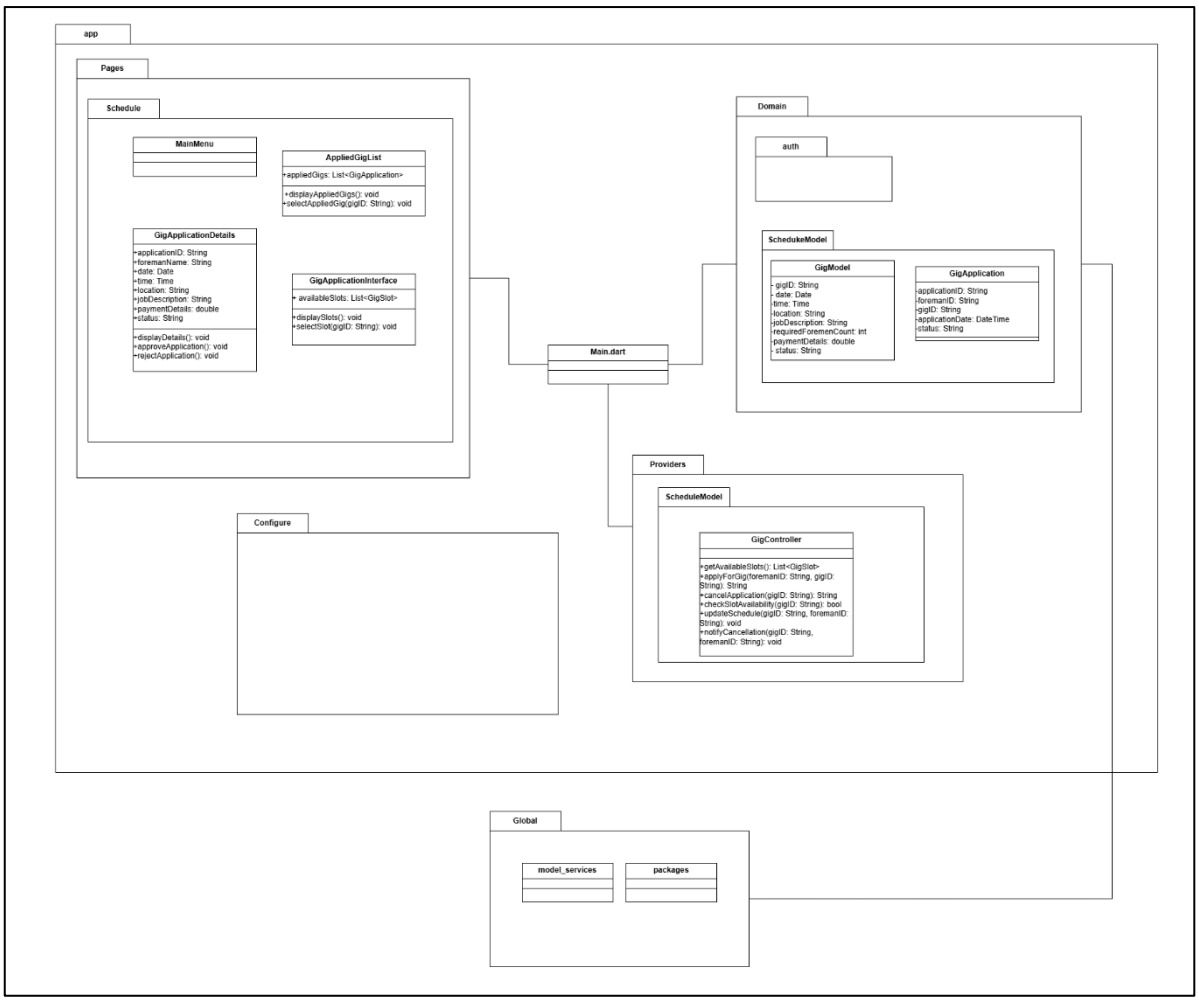
START
approveApplication(applicationID) is called
Fetch application using applicationID
Set status to "Approved"
Save application
Call notifyForeman("Approved")
END

rejectApplication(applicationID)

START
rejectApplication(applicationID) is called
Fetch application using applicationID
Set status to "Rejected"

	Save application Call notifyForeman("Rejected") END **notifyForeman(status)** START notifyForeman(status) is called Compose notification message based on status Send notification to corresponding foreman END

4.5 Select Available Schedule (Amala Karthigayan, CB23116)



GigApplicationInterface [CLS-WMS-2025-501-01]

Class Type	Boundary class
Responsibility	Displays all available gig slots and allows foremen to apply

Attributes	Attributes Name	Attributes Type
	availableSlots	List<GigSlot>
Methods	Method Name	Description
	displaySlots()	Displays all available slots
	selectSlot(gigID: String)	Selects a gig slot for application
Algorithm	**displaySlots()** START displaySlots() is called Fetch all available slots from GigController Populate availableSlots list Render slot list in UI END **selectSlot(gigID: String)** START selectSlot(gigID) is called Send selected gigID to GigController.applyForGig() Notify user of application status END	

AppliedGigList [CLS-WMS-2025-501-02]

Class Type	Boundary Class	
Responsibility	Displays all gigs the foreman has applied to	
Attributes	Attributes Name	Attributes Type
	appliedGigs	List<GigApplication>
Methods	Method Name	Description
	displayAppliedGigs()	Displays all applied gig slots
	selectAppliedGig(gigID: String)	Opens details for the selected applied gig
Algorithm	**displayAppliedGigs()** START	

	displayAppliedGigs() is called Fetch all applied gigs from data source Populate appliedGigs list Display in UI END **selectAppliedGig(gigID: String)** START selectAppliedGig(gigID) is called Navigate to GigApplicationDetails interface Pass selected gigID END

GigApplicationDetails [CLS-WMS-2025-501-03]

Class Type	Boundary Class	
Responsibility	Displays detailed information about a specific gig application	
Attributes	Attributes Name	Attributes Type
	applicationID	String
	foremanName	String
	date	Date
	time	Time
	location	String
	jobDescription	String
	paymentDetails	double
	status	String
Methods	Method Name	Description
	displayDetails()	Displays detailed view of the gig application
	approveApplication()	Approves the gig application

	rejectApplication()	Rejects the gig application
Algorithm	**displayDetails()** START displayDetails() is called Fetch application details using applicationID Display all information in UI END **approveApplication()** START approveApplication() is called Call GigController.approveApplication(applicationID) Update UI with status 'Approved' END **rejectApplication()** START rejectApplication() is called Call GigController.rejectApplication(applicationID) Update UI with status 'Rejected' END	

GigModel [CLS-WMS-2025-502-01]

Class Type	Entity Class

Responsibility	Represents the data structure of a gig slot	
Attributes	Attributes Name	Attributes Type
	gigID	String
	date	Date
	time	Time
	location	String
	jobDescription	String
	requiredForemenCount	int
	paymentDetails	double
	status	String
Methods	Method Name	Description
	Not Applicable	
Algorithm	Not Applicable	

GigApplication [CLS-WMS-2025-502-02]

Class Type	Entity class	
Responsibility	Stores foreman application details for gig slots	
Attributes	Attributes Name	Attributes Type
	applicationID	String
	foremanID	String
	slotID	String
	applicationDate	DateTime
	status	String
Methods	Method Name	Description
	Not Applicable	
Algorithm	Not Applicable	

GigController [CLS-WMS-2025-503-01]

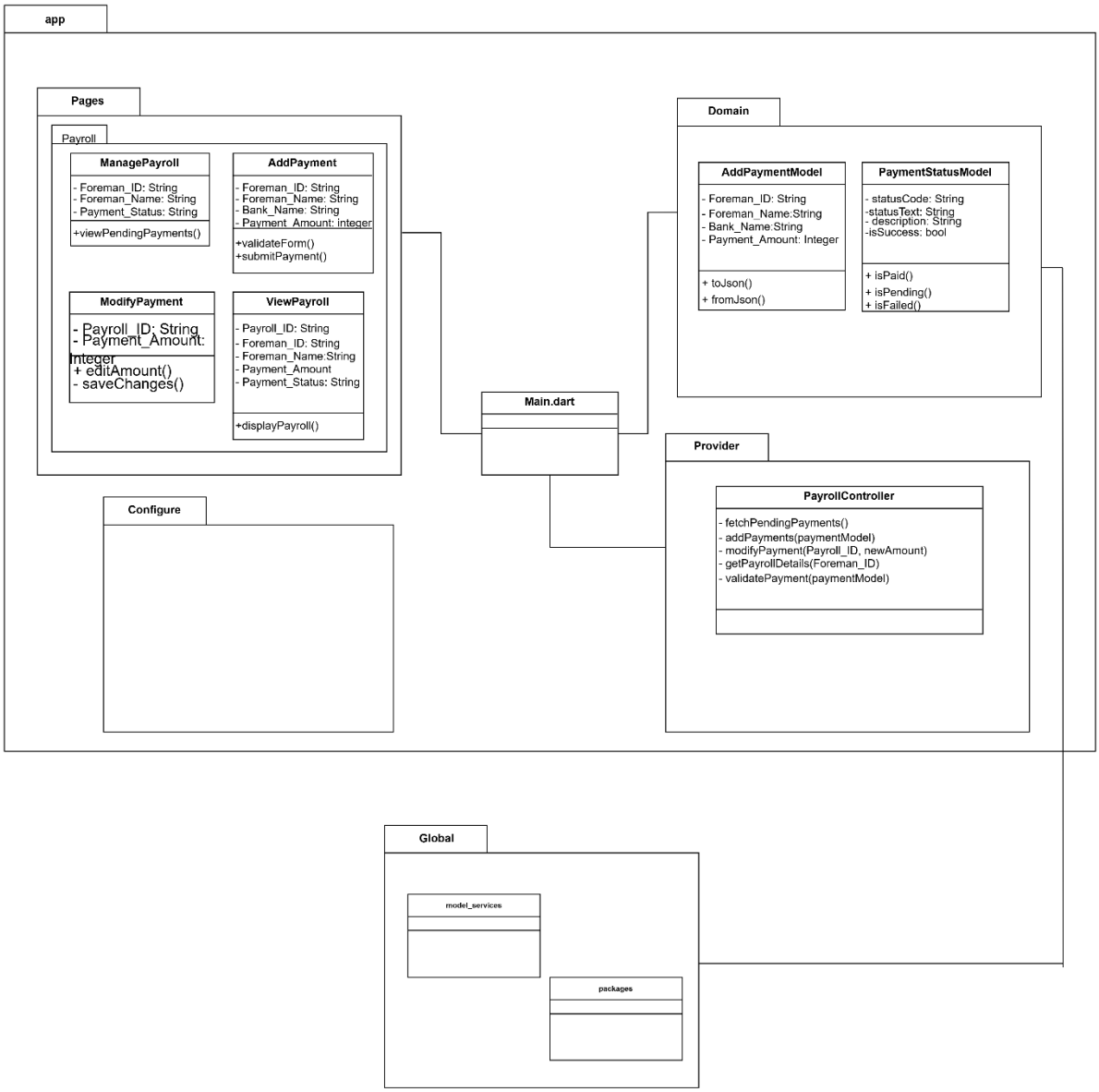
<table>
<tr><td>Class Type</td><td colspan="2">Control class</td></tr>
<tr><td>Responsibility</td><td colspan="2">Handles gig applications and slot availability logic</td></tr>
<tr><td rowspan="2">Attributes</td><td>Attributes Name</td><td>Attributes Type</td></tr>
<tr><td colspan="2">Not Applicable</td></tr>
<tr><td rowspan="7">Methods</td><td>Method Name</td><td>Description</td></tr>
<tr><td>getAvailableSlots()</td><td>Returns all available gig slots</td></tr>
<tr><td>applyForGig(foremanID, gigID)</td><td>Applies a foreman to a gig slot</td></tr>
<tr><td>cancelApplication(gigID)</td><td>Cancels a foreman’s gig application</td></tr>
<tr><td>checkSlotAvailability(gigID)</td><td>Checks if a gig slot is still available</td></tr>
<tr><td>updateSchedule(gigID, foremanID)</td><td>Updates the schedule after successful application</td></tr>
<tr><td>notifyCancellation(gigID, foremanID)</td><td>Notifies relevant parties of the cancellation</td></tr>
<tr><td>Algorithm</td><td colspan="2">getAvailableSlots()
START
getAvailableSlots() is called
Fetch gig slots with status 'Open'
RETURN list of available slots
END

applyForGig(foremanID, gigID)
START
applyForGig(foremanID, gigID) is called
Check slot availability using checkSlotAvailability()
If available, create new GigApplication record
Save to database
Call updateSchedule(gigID, foremanID)
RETURN success message
END

cancelApplication(gigID)
START
cancelApplication(gigID) is called
Find application by gigID and logged-in foreman</td></tr>
</table>

	Remove application Call notifyCancellation(gigID, foremanID) RETURN cancellation confirmation END **checkSlotAvailability(gigID)** START checkSlotAvailability(gigID) is called Retrieve gig slot by gigID If status is 'Open' and requiredForemenCount > 0, RETURN true Else RETURN false END **updateSchedule(gigID, foremanID)** START updateSchedule(gigID, foremanID) is called Decrease requiredForemenCount of the slot Log foreman to the schedule END **notifyCancellation(gigID, foremanID)** START notifyCancellation(gigID, foremanID) is called Compose cancellation message Send notification to foreman and admin END

4.6 Manage Payroll (NUR NABIHAH FATINY BINTI ISMAIL CB23101)



ManagePayroll [CLS-WMS-2025-601-01]

Class Type	Boundary class	
Responsibility	UI screen showing a list of Foremen with pending payments and navigation.	
Attributes	Attributes Name	Attributes Type
	Foreman_ID	String
	Foreman_Name	String
	Payment_Status	String

Methods	Method Name	Description
	None	None
Algorithm	None	

AddPayment[CLS-WMS-2025-601-02]

Class Type	Boundary Class	
Responsibility	Form to fill payment details with validation.	
Attributes	Attributes Name	Attributes Type
	Foreman_ID	String
	Foreman_Name	String
	Bank_Name	String
	Account_Number	String
	Payment_Amount	Integer
	Payment_Date	Date
	Payment_Reference	String
Methods	Method Name	Description
	ValidateForm()	Checks for input validity
	submitPayment()	Sends payment details to provider
Algorithm	ValidateForm() START If any field is empty OR Payment_Amount <= 0: return false return true END SubmitPayment() START If ValidateForm():	

	payment = new AddPaymentModel(...) PayrollController.addPayment(payment) END

ModifyPayment [CLS-WMS-2025-601-03]

Class Type	Boundary Class	
Responsibility	Allow modification of payment amount.	
Attributes	Attributes Name	Attributes Type
	Payroll_ID Payment_Amount	String Integer
Methods	Method Name	Description
	editAmount() saveChanges()	Allows modification of the payment amount Saves updates amount to record
Algorithm	SaveChanges() START If Payment_Amount > 0: PayrollController.modifyPayment(Payroll_ID, Payment_Amount) END	

ViewPayroll [CLS-WMS-2025-601-04]

Class Type	Boundary Class	
Responsibility	Read-only UI to view payroll records and status.	
Attributes	Attributes Name	Attributes Type
	Payroll_ID	String

	Foreman_ID Foreman_Name Bank_Name Account_Number Payment_Amount Payment_Date Payment_Reference Payment_Status	String String String String Integer Date String String
Methods	Method Name	Description
	DisplayPayroll()	Shows payroll details in read-only format
Algorithm	DisplayPayroll() START data = PayrollController.getPayrollDetails(Foreman_ID) display(data) END	

AddPaymentModel[CLS-WMS-2025-602-01]

Class Type	Entity Class	
Responsibility	Stores input data for payment before submission.	
Attributes	Attributes Name	Attributes Type
	Foreman_ID Foreman_Name Bank_Name Account_Number Payment_Amount Payment_Date Payment Reference	String String String String Int Date String
Methods	Method Name	Description

	toJson() FromJson()	Convert form to JSON Read from data map

Algorithm	To Json()
	
	START
	
	Return {
	
	"Foreman_ID": Foreman_ID,
	
	"Foreman_Name": Foreman_Name,
	
	"Bank_Name": Bank_Name,
	
	"Account_Number": Account_Number,
	
	"Payment_Amount": Payment_Amount,
	
	"Payment_Date": Payment_Date,
	
	"Payment_Reference": Payment_Reference
	
	}
	
	END
	
	FromJson()
	
	START
	
	Foreman_ID = json["Foreman_ID"]
	
	Foreman_Name = json["Foreman_Name"]
	
	Bank_Name = json["Bank_Name"]
	
	Account_Number = json["Account_Number"]
	
	Payment_Amount = json["Payment_Amount"]
	
	Payment_Date = json["Payment_Date"]	

	Payment_Reference = json["Payment_Reference"] END

PaymentStatusModel [CLS-WMS-2025-602-02]

Class Type	Entity Class	
Responsibility	Represents and describes status of a payment “Pending”, “Payment Successful”, “Rejected”.	
Attributes	Attributes Name	Attributes Type
	StatusCode StatusText Description isSuccess	String String String bool
Methods	Method Name	Description
	IsPaid() IsPending() IsFailed()	Payment paid Payment pending Payment failed
Algorithm	IsPaid() START If StatusCode == "Paid": return true return false END IsPending() START If StatusCode == "Pending": return true return false END	

	Isfailed() START If StatusCode == "Rejected": return true return false END

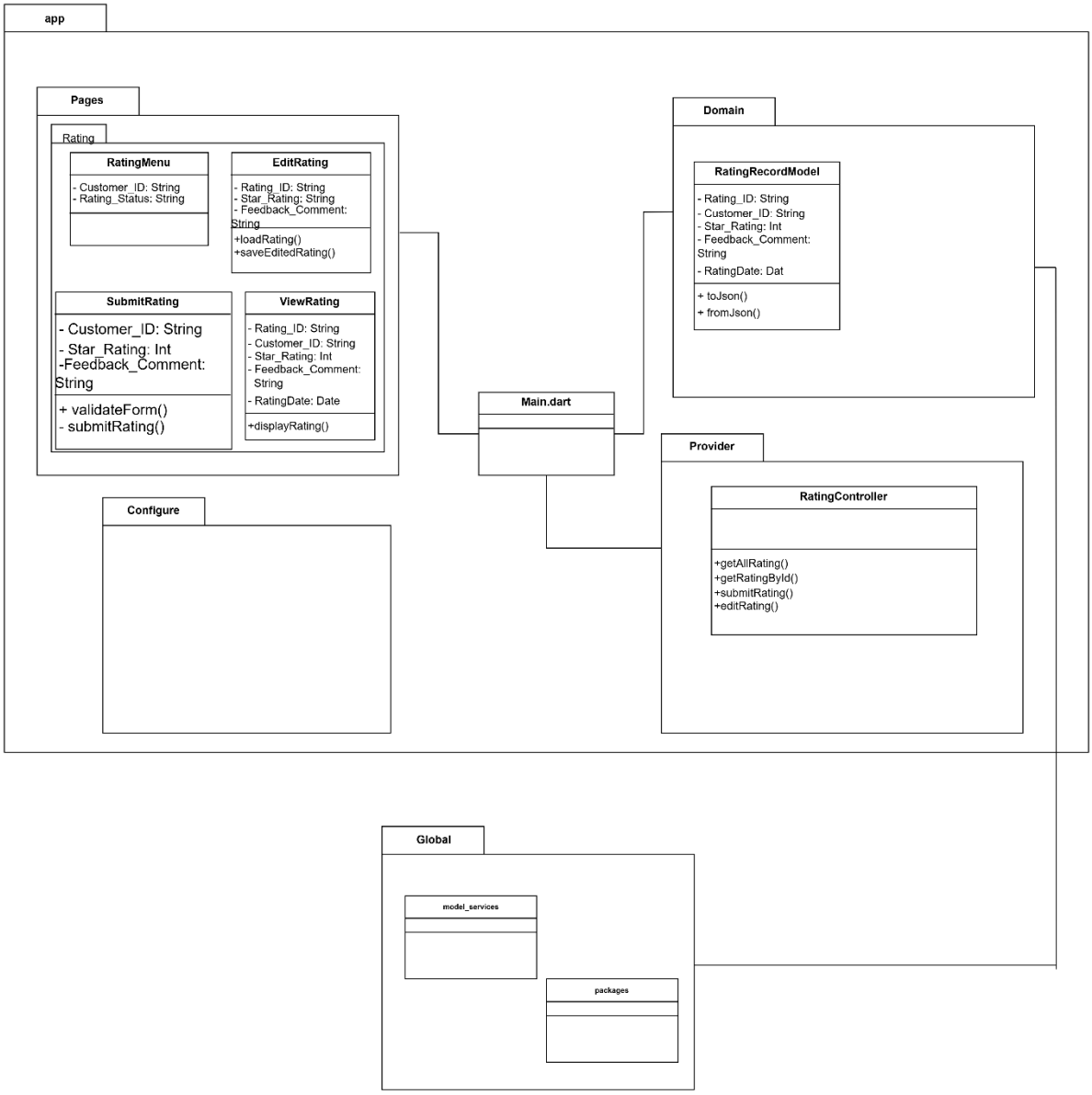
PayrollController [CLS-WMS-2025-603-01]

Class Type	Control Class	
Responsibility	Handles all logic for payroll processing, including fetching, adding, modifying, and validating payments.	
Attributes	Attributes Name	Attributes Type
	None	None
Methods	Method Name	Description
	fetchPendingPayments() addPayment() modifyPayment() getPayrollDetails() validatePayment()	Retrieves a list of Foremen Adds a new payment Edits payment amount Retrieves payroll data for foreman Validates the AddPayment form
Algorithm	FetchPending Payment() **START** Return all payrolls where Payment_Status == "Pending" **END** AddPayment() **START** If validatePayment(paymentModel): Save paymentModel to database with status "Paid" **END**	

	ModifyPayment() **START** record = find by Payroll_ID record.Payment_Amount = newAmount save(record) **END** GetPayrolDetails() **START** Return record where Foreman_ID == input **END** ValidatePayment() **START** If any required field is empty OR paymentModel.Payment_Amount <= 0: return false return true **END**

4.7 Manage Rating (NUR NABIHAH FATINY BINTI ISMAIL CB23101)

Package Diagram with classes including attributes and function



RatingMenu [CLS-WMS-2025-701-01]

Class Type	Boundary class	
Responsibility	Displays all submitted/unsubmitted ratings and provides navigation to submit/edit options	
Attributes	Attributes Name	Attributes Type
	Customer_ID Rating_Status	String String

Methods	Method Name	Description
	None	None
Algorithm	None	

SubmitRating [CLS-WMS-2025-702-01]

Class Type	**Boundary Class**	
Responsibility	UI form for the customer to submit a new rating and feedback.	
Attributes	Attributes Name	Attributes Type
	Customer_ID	String
	Star_Rating	Int
	Feedback_Comment	String
	Rating_Date	Date
Methods	Method Name	Description
	ValidateForm()	Checks input validity
	SubmitRating()	Sends rating data to controller
Algorithm	ValidateForm() START If Star_Rating < 1 or > 5 OR Feedback_Comment is empty: return false return true END SubmitRating() START submitRating() is called If validateForm(): rating = new RatingRecord(...) RatingController.submitRating(rating) END	

EditRating [CLS-WMS-2025-702-01]

Class Type	Boundary Class	
Responsibility	UI for customers to edit previously submitted ratings and comments.	
Attributes	Attributes Name	Attributes Type
	Rating_ID Star_Rating Feedback_Comment	String Int String
Methods	Method Name	Description
	loadRating() SaveEditedRating()	Checks input validity Sends updated data to controller
Algorithm	loadRating(): START Rating = RatingController.getRatingById(Rating_ID) populateForm(Rating) END SaveEditedRating(): START If Star_Rating between 1 and 5: RatingController.editRating(Rating_ID, Star_Rating, Feedback_Comment) END	

ViewRating [CLS-WMS-2025-702-01]

Class Type	Boundary Class	
Responsibility	Read-only UI for Workshop Owner and Foreman to view all submitted ratings.	
Attributes	Attributes Name	Attributes Type
	Rating_ID Customer_ID Star_Rating Feedback_Comment RatingDate	String String Int String Date
Methods	Method Name	Description
	displayRatings()	Displays all submitted ratings
Algorithm	DisplayRating() START data = RatingController.getAllRatings() Display(data) END	

RatingRecordModel [CLS-WMS-2025-702-01]

Class Type	Entity Class	
Responsibility	Represents rating data and handles validation.	
Attributes	Attributes Name	Attributes Type
	Rating_ID Customer_ID Star_Rating Feedback_Comment	String String Int String

	RatingDate	Date
Methods	Method Name	Description
	toJson() fromJson()	Convert object to JSON Convert from data to object
Algorithm	none	

RatingController [CLS-WMS-2025-702-01]

Class Type	Entity Class	
Responsibility	Handles rating submission, editing, fetching all ratings	
Attributes	Attributes Name	Attributes Type
	None	None
Methods	Method Name	Description
	getAllRatings() getRatingById() submitRating() editRating()	Fetch all ratings by customer Fetch rating by ID Submit a new rating update rating details
Algorithm	GetAllRatings START Return all ratings where Customer_ID == input END GetRatingByID START Return rating where Rating_ID == input	

	END SubmitRating START Save rating to database END EditRating START rating = find by Rating_ID rating.Star_Rating = stars rating.Feedback_Comment = comment save(rating) END

5.0 Requirement Traceability

1.1 Manage Profile (NUR ALYA SYAKIRAH BINTI NASARUDIN CB22141)

Requirement ID	Description	Design ID
REQ-WMS-2025-103	Users can log in, register an account, edit their profile and change their password with all actions managed by the controller and connected to the database through the model.	PKG-WMS-2025-101 PKG-WMS-2025-102

		PKG-WMS-2025-103
REQ-WMS-2025-104	Users can log in with email and password, register with detailed personal and professional information, edit their profile and change their password through boundary interfaces with all actions processed by the ProfileController and stored or updated in the database via the ProfileRecord entity.	CLS-WMS-2025-101-01 CLS-WMS-2025-102-01 CLS-WMS-2025-103-01 CLS-WMS-2025-104-01 CLS-WMS-2025-105-01 CLS-WMS-2025-106-01

1.2 Manage Shop Inventory (Varshini Jagarajan, CB22139)

Requirement ID	Description	Design ID
REQ-WMS-2025-201	The Workshop Owner is able to add a new inventory item by entering item name, quantity, and category.	CLS-WMS-2025-201-01 CLS-WMS-2025-202-01
REQ-WMS-2025-202	If required information is missing while adding/updating an item, the system prompts the Workshop Owner to fill in the missing fields.	CLS-WMS-2025-202-01 CLS-WMS-2025-203-01 CLS-WMS-2025-301-01
REQ-WMS-2025-203	The Workshop Owner can update existing inventory item details, such as stock quantity, name, or category.	CLS-WMS-2025-201-01 CLS-WMS-2025-203-01
REQ-WMS-2025-204	The Workshop Owner can remove an inventory item by selecting it and clicking the delete button.	CLS-WMS-2025-201-01 CLS-WMS-2025-204-01
REQ-WMS-2025-205	The system prompts the Workshop Owner for confirmation before deleting an item; cancellation aborts deletion.	CLS-WMS-2025-204-01
REQ-WMS-2025-206	The system automatically detects low stock items and displays a low stock alert.	CLS-WMS-2025-201-01 CLS-WMS-2025-301-01 CLS-WMS-2025-302-01

REQ-WMS-2025-207	When low stock is detected, the system allows the Workshop Owner to initiate the Request Inventory process.	CLS-WMS-2025-201-01 CLS-WMS-2025-301-01 CLS-WMS-2025-302-01
REQ-WMS-2025-208	If a system error occurs while updating, deleting, or saving inventory data, the system displays an error and logs the issue.	CLS-WMS-2025-202-01 CLS-WMS-2025-203-01 CLS-WMS-2025-204-01

1.3 Request Inventory (Varshini Jagarajan, CB22139)

Requirement ID		Description	Design ID
REQ-WMS-2025-301	If the Workshop Owner submits a request without specifying quantity or supplier, the system prompts to complete the required fields.	CLS-WMS-2025-301-01	
REQ-WMS-2025-302		If the chosen supplier is unavailable, the system notifies the Workshop Owner and prompts them to select an alternative.	CLS-WMS-2025-301-01
REQ-WMS-2025-303	If a system error occurs during inventory request submission, the system displays an error message and logs the error.	CLS-WMS-2025-302-01	

1.4 Manage Foreman Schedule (Amala Karthigayan, CB23116)

Requirement ID	Description	Design ID
REQ-WMS-2025-401	Workshop owner shall be able to add gig slots.	CLS-WMS-2025-401-02

		CLS-WMS-2025-402-01 CLS-WMS-2025-403-01
REQ-WMS-2025-402	Workshop owner shall be able to edit gig slots.	CLS-WMS-2025-401-03 CLS-WMS-2025-403-01 CLS-WMS-2025-402-01
REQ-WMS-2025-404	Workshop owner shall be able to approve foreman gig application.	CLS-WMS-2025-401-05 CLS-WMS-2025-403-01 CLS-WMS-2025-402-01
REQ-WMS-2025-405	Slots with same job description shall not exist at the same location, date and time.	CLS-WMS-2025-403-01 CLS-WMS-2025-402-01

1.5 Select Available Schedule (Amala Karthigayan, CB23116)

Requirement ID	Description	Design ID
REQ-WMS-2025-501	Foreman shall be able to apply for gig	CLS-WMS-2025-501-01 CLS-WMS-2025-501-03

		CLS-WMS-2025-502-01 CLS-WMS-2025-503-01
REQ-WMS-2025-502	Foreman shall be able to cancel their gig application at least 24 hours before the gig starts.	CLS-WMS-2025-501-01 CLS-WMS-2025-501-03 CLS-WMS-2025-502-01 CLS-WMS-2025-503-01
REQ-WMS-2025-503	Slots that are fully booked shall not be available for application.	CLS-WMS-2025-502-01 CLS-WMS-2025-503-01
REQ-WMS-2025-504	Foreman shall not be able to cancel an application within the 24 hours of a gig start.	CLS-WMS-2025-502-01 CLS-WMS-2025-503-01

1.6 Manage Payroll (NUR NABIHAH FATINY BINTI ISMAIL CB23101)

Requirement ID	Description	Design ID
REQ-WMS-2025-601	The Workshop Owner is able to view a list of Foremen with pending payments.	CLS-WMS-2025-601-01 CLS-WMS-2025-603-01

REQ-WMS-2025-602	The Workshop Owner can add a new payment by entering Foreman details, bank information, payment amount, date, and reference.	CLS-WMS-2025-601-02 CLS-WMS-2025-602-01 CLS-WMS-2025-603-01
REQ-WMS-2025-603	The Workshop Owner can modify the payment amount before finalizing the payment.	CLS-WMS-2025-601-03 CLS-WMS-2025-603-01
REQ-WMS-2025-605	Foremen can view their payroll records and payment status in a read-only format.	CLS-WMS-2025-601-04 CLS-WMS-2025-603-01
REQ-WMS-2025-606	The system maintains payment status information such as "Pending", "Paid", or "Rejected".	CLS-WMS-2025-602-02 CLS-WMS-2025-603-01

1.7 Manage Rating (NUR NABIHAH FATINY BINTI ISMAIL CB23101)

Requirement ID	Description	Design ID
REQ-WMS-2025-701	The system displays all submitted and unsubmitted ratings with navigation options for customers.	CLS-WMS-2025-701-01 CLS-WMS-2025-703-01
REQ-WMS-2025-702	Customers can submit new ratings by entering star rating (1-5) and feedback comments	CLS-WMS-2025-702-01 CLS-WMS-2025-702-01 CLS-WMS-2025-703-0

REQ-WMS-2025-706	Workshop Owners and Foremen can view all submitted ratings in a read-only format.	CLS-WMS-2025-702-01 CLS-WMS-2025-703-01