CAPSTONE PROJECT

MATER DEI COLLEGE PROPERTY RESERVATION MANAGEMENT SYSTEM

Group 1

CODE-DIGO

MEMBERS:

CIFRA, MC. VINCENT G.

ESTEBAN, ALDRIN U.

ESTEBAN, REINHARD U.

RESERVA, FRANCIS A.

JULY 2024

TABLE OF CONTENTS

CHAPTER I

Introduction

1.1 Project Context

Mater Dei College (MDC) is a catholic school situated in the lively town of Tubigon, Bohol, known for its dedication in delivering high-quality education rooted from Catholic principles. MDC also provides a wide range of rental services and facilities that can be used for both academic and non-academic purposes. These services include folklore performance costumes, tools and musical instruments, Activity Center (AC), Multimedia Center (MMC), transportation vehicles, chairs tables, graduation togas and more. Rentees can rent these school properties by the hour, day, week, and/or month.

Currently, MDC manages its inventory manually for these resources, which poses challenges such as labor intensiveness and potential errors. This manual approach can result in issues like double bookings, misplaced items, and a lack of accountability. Such inefficiencies provide inconvenience to both the staff handling rentals and the users relying on these resources.

To address these problems, MDC is embarking on the development of the MDC Property Rental Management System. Mr. Artemio M. Gulilat, CPA, "*Vice-President of Administration and Finance*", wonders if the system could have a calendar feature that tracks pending requests of rentees and also allow him for the approval of these requests with just a few clicks. This advanced system aims to enhance accuracy and efficiency by digitizing the inventory management process. By automating inventory tracking, the system will reduce workload, minimize errors, and provide real-time status updates on all items and facilities. Its user-friendly interface ensures ease of navigation, even for those with limited technical skills. Key features include automated booking confirmations, live inventory updates, and comprehensive usage reports, revolutionizing how the college oversees its resources. Without a proper equipment rental tracking solution in place, schools will continue to face equipment mismanagement and losses as stated by EZRentOut developed by EZO Company (2019).

Implementing this web-based software promises cost savings and potential revenue growth through improved rental services. This efficiency will allow staff to focus on more critical responsibilities, while user and rentee benefit from a more dependable and transparent rental experience. In essence, the MDC Property Rental Management System signifies a substantial advancement in utilizing technology to enhance resource management and support MDC's educational mission.

1.2 Purpose

The primary goal of this project is to transform the currently manual process of tracking transactions and inventories of property rentals at MDC into an automated, efficient, secured and accurate system.

MDC offers a school property rental which includes folklore performance tools, costumes and musical instruments, AC, MMC, transportation vehicles, chairs and tables and more. These services still adopt a manual system which is hassle and time-consuming in tracking it. However, if we develop a user-friendly system, it will help reduce those problems.

Moreover, this system will help to simplify and securely cater to rentee. Imagine, all rentees will write a letter and submit it to the assigned staff of the property. After the staff receives the letter, it will be forwarded to Mr. Artemio M. Gulilat, CPA, *“Vice-President of Administration and Finance”* for the approval. Once the request is approved, the rentee can proceed to get their rented property. Staff manually recording and processing these transactions face-to-face with the rentee. To fix this hassle and time-consuming process, users only need to browse the system using their smartphones, laptops, or PCs. With just a few clicks, they can now rent whatever they need at MDC. The rentee will be given a unique and verified digital receipt (with QR Code) and will only be required to go to the cashier to pay the amount for what they rented. The system also ensures the security of data and the safety of inventories and transactions throughout the process.

1.3 Objectives

The developers aim to build a Web Based System that will list all the properties of MDC available for rent. Rentees can add properties to a cart, proceed to checkout, system generates a unique digital receipt (with QR Code), and then go to the cashier to make the payment and finally, they can now get and use their rented property.

Specific Objectives

- Create a system that has a complete list and details of all the properties available for renting and borrowing at MDC.
- Reduce the effort required and the time-consuming process currently involved in managing rentals.
- Design a user-friendly system that is easy to use for all staff members and rentees, even for those with minimal technical knowledge.
- Develop a secure and trusted system that safely stores the sensitive information of the staff and rentee, ensuring their privacy and data protection.
- Implement a feature that tracks which rentees have rented which properties at any given time (hour, day, or month), helping to track and prevent the loss of property.
- Create distinct features for both users and admin for security purposes. This ensures that both have access only to the features and data relevant to their roles, thereby improving the overall security of the system.

1.4 Scope and Limitations

Scope

Since this system caters for rentals, individual staff and cashiers will have to register for a user account. Staff and rentee can access through their mobile phones, tablets, laptops or PCs. Different features and designs for the admin, staff, and rentee.

Rentee will simply need to browse in the browser for the application and they will be taken to the rent tab or Window, as shown below containing a list of all available properties for rent. This, in turn, allows rentees to add properties into a cart and proceed to checkout and get a digital receipt that appears on their devices after checkout and then rentees will go to the cashier for payment.

Staff and Cashiers also need to have a unique account for security purposes. Staff will sign onto the system and open a page that has an all pending inquiries list. The system will process the payment amount, and the rentee will make the payment. if the transaction is completed, the system will automatically generate a confirmation for the rentee’s account, indicating the rental was successful. The transaction history will be recorded in the database for future reference.

Limitations

For the system, it needs an internet connection to be connected and used by devices. This also implies that rentee who live in remote areas or where the Internet is not up to the mark will have trouble with rental transactions, search property online and log-in on internet signals being poor. Moreover, rentee are not given any access to the system admin panel. Simply due to the security and consistency of the system, protecting you from un-directed modifications to its settings or accessing sensitive recorded data in the database.

For added security, the solution does not offer online payment options because all the payments must be made physically at the cashier, this can help reduce the threat of fraud while ensuring that these transactions are completed in person.

Mater Dei College, in addition to having a small inventory of rental units, it classifies those available according to the type of rentee. Outsiders, or individuals who are not affiliated with MDC, are restricted from renting certain properties such as buses and vans. These specific items are reserved exclusively for rentee who are associated with the school for academic purposes only, such as teachers, students (with department dean approval), faculty, and staff.

CHAPTER II

Review of Related Literature/System

This chapter describes the related literature of the proposed MDC Property Management System. This article reviews the knowledge and step-by-step process detailed by developers and authors (required or not) to successfully develop the project.

Based on research conducted by EZO Company (2019), in order to better serve the orderly service of their propelling diverse equipment inventory throughout the educational sector around the world, EZRentOut was able to make an innovative system where academic institutions are now able to conveniently manage all asset records. Public school enrollment was 35.6 million students in 2018 and is projected to reach 52.1 million by fall 2027. Students regularly rent facilities from educational institutions for free or at no cost (with late return fees only), which can be a pain to manage as the inventory size grows. Advancements in technology are replacing paper textbooks with online learning and tablets, greatly affecting the industry. Classroom equipment falls into a few large categories, ranging from chalkboards to IT equipment such as projectors and screens; these basic technology systems require daily upkeep for educational institutions to continue operating effectively.

According to Yanxiu et al. (2015), an efficient solution is needed to manage a school's sports equipment. Traditionally, the sports equipment management center relied on manual paper records for storage, statistics, lending, and returning of equipment, which hindered timely access to statistical information and accurate record-keeping. The study set out to design a convenient school sports equipment borrowing management system. This project studied system architecture and design patterns based on J2EE development platform and B/S structure. The envisaged system consists of modules for data input, equipment online inquiry and reservation besides statistics at the end. At the same time it is to study functional requirements, database structures and modular architecture followed by an open-source coding platform.

An organization named Venyooz created a web-based software called SchoolSpace™. Venyooz is an online peer-to-peer marketplace for temporary space rentals. Schools, community centers, churches, retail stores, and other businesses and organizations list their underutilized, "rentable" spaces, founded in 2012. Parents, independent instructors, group organizers, and others can rent these spaces by the hour, day, week, and/or month. Venyooz developed a web-based software so that schools can manage their available rental facilities.

School facilities management has become a whole new ball game that can be utilized by staff members and students at different levels of the community. After signing up, each user will have one account that can be customized to fit their role and responsibilities. This account exists to keep all contact information in one place and monitor facility requests made for sponsored events. Its strong permission settings allow specified personnel to set up system preferences, standardize approval workflows, see everything they need financially at a glance, run detailed financial reports, and keep the calendar running smoothly. By enabling transparent communication and straightforward workflows, SchoolSpace™ supports operational effectiveness, which in turn helps coordinate scheduling and facility utilization that serves the broad needs of education along with those of various community uses (SchoolSpace, n.d.).

CHAPTER III

Technical Background

This chapter provides an overview of the technical background for the proposed system. It is designed to improve our understanding of the technical concepts and the system's workflow.

3.1 Architecture Design

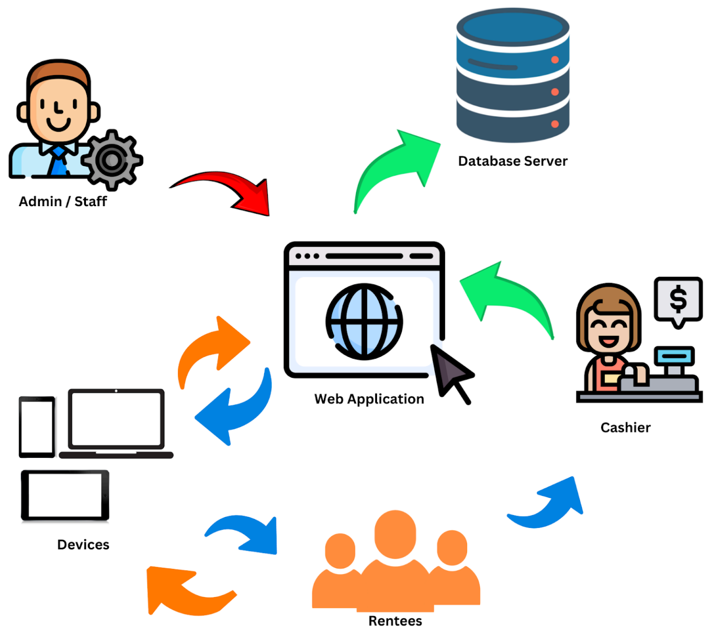


Figure 3.1.1 Architecture Design

3.2 Details of Technology Used

This section introduces the technology stack of this system that the developers will be using in the development of this project.

Microsoft Visual Studio Code - It is a Microsoft-provided freeware that allows computer programmers to create software. It is used to develop computer programs, as well as websites, web apps, web services and mobile apps. It enables programmers to create apps and websites using platforms including Visual Basic, C++, C++/CLI, C#, JavaScript, TypeScript, XML, XSLT, HTML, CSS, and others.

XAMPP - This will be used by the developers as a development tool, to allow website programmers to test their work without connection to the Internet on their own computers.

MySQL - MySQL is a database system used on the web. MySQL is a database system that runs on a server. MySQL is ideal for both small and large applications. MySQL is very fast, reliable, and easy to use. MySQL uses standard SQL.

Laravel - It's primarily used to create PHP-based custom web applications. It's a web framework that takes care of a lot of the things that are inconvenient to build yourself, like routing, HTML templating, and authentication.

Tailwind CSS - is a utility-first CSS framework, according to its creators. Tailwind is focused on how the item should be shown rather than the functionality of the item being designed. This makes it easy for the developer to experiment with different styles and layouts.

Select2 JS - is one of the most widely spread jQuery plugins for enhancing the standard <select> form controls. It offers easy to implement, highly interactive drop down list options with features that enhance the usability of their selection.

jQuery JS - is a lightweight JavaScript utility library that is used to manipulate DOM, handle DOM events, create animation and make asynchronous applications for loading HTML/HTTP. It was developed by John Resig in 2006 and is one of the most popular libraries in use on the web today because of its simplicity and ability to work on all browsers.

HTMX - is a lightweight JavaScript toolkit, which provides active HTML elements as a means of rendering HTML-based AJAX on your current HTML elements without programming or using complex front-end toolkits. It enables you to enhance the interactivity of webpage through handling of Http request, and by updating only the needed portion of the webpage upon events such as click, form submission events and many others, due to its simplicity and a feature of direct manipulation of the HTML it avails developers an easy way of building middle-heavy dynamic features without having to steep into the complexities of any front-end colossal like React or angular.

jsQR & Qrious JS - is a JavaScript library that you can use to perform QR code decoding right in the browser with simple JavaScript. It is able to read QR codes from an image or can read an image on the fly (with a webcam or anything similar to it) thus, it is a very useful library for building applications that involve scanning and reading of QR codes.

Font Awesome CSS - is the open source icon set, designed for using scalable vector icons for custom with CSS. This one is the most famous platform for web development and provides the opportunity to add icons to sites and applications, including free and paid icons.

Flatpickr JS - is a lightweight and feature-rich JavaScript library that provides a highly customizable date and time picker for web applications. It offers an easy-to-use interface for selecting dates and times, with support for a wide range of configurations and features, such as date range selection, time picking, localization, and more. Flatpickr is designed to be fast, easy to integrate, and flexible enough for use in various scenarios.

Smartphone - A smartphone is a portable computer device that combines mobile telephone functions and computing functions into one unit. They are distinguished from older-design feature phones by their more advanced hardware capabilities and extensive mobile operating systems, which facilitate wider software, access to the internet (including web browsing over mobile broadband), and multimedia functionality (including music, video, cameras, and gaming), alongside core phone functions such as voice calls and text messaging.

PC/Personal Computer - The PC is a general-purpose, cost-effective computer that stands for the personal computer. It is when Ed Roberts introduced the MITS Altair 8800, he coined the term PC. Alternatively, it is referred to as a single-user computer and a desktop that is designed to be used by a single end-user. All PCs depend on the microprocessor technology that makes it capable for PC makers to set the whole CPU (central processing unit) on a single chip.

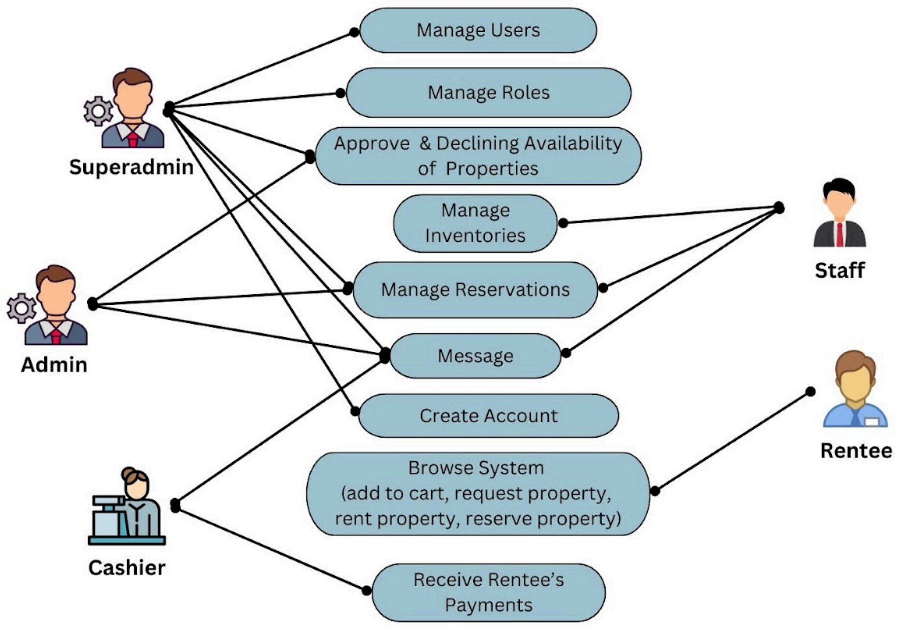
CHAPTER IV

Methodology

4.1 Requirements Analysis and Requirements Documentation

In describing how the project is designed, the developers used a use-case diagram to describe how this project operates.

Use Case Diagram


Manage Users
Manage Roles
Approve & Declining Availability of Properties
Manage Inventories
Manage Reservations
Message
Create Account
Browse System (add to cart, request property, rent property, reserve property)
Receive Rentee's Payments
Superadmin
Admin
Cashier
Staff
Rentee

4.1.1 User Diagram

Use Case Description

Use Case Name: Manage Roles	ID:1	Important Level: High
Primary Actor: Superadmin	Use Case Type: Essential, Detail	
Scenario: Registers a user then assigned to be an admin, staff or cashier role into the system.		
Brief Description: This use case describes how the superadmin registers a user then assigned to be a admin, staff or cashier role into the system.		
Trigger: The superadmin registers a user then assigned to be a staff or cashier role into the system.		
Normal Flow of Events: 1. The superadmin opens the application and requests to log in. 2. The admin enters username & password to the system. 3. The system navigates to the dashboard. 4. The admin clicks the user's sidebar function. 5. The admin clicks the Create User button. 6. The admin registers a user and assigns it into a staff or cashier role. 7. The system ends.		
Sub Flows:		
Alternative/Exceptional Flows:		

Table 4.1.1 Manage Roles

Use Case Name: Manage User's Information	ID:2	Important Level: High
Primary Actor: Superadmin	Use Case Type: Essential, Detail	
Scenario: The superadmin wants to manage accounts of users (create,update,delete).		
Brief Description: This use case describes how the superadmin manages the information of users.		

Use Case Name: Manage User’s Information	ID:2		Important Level: High
Primary Actor: Superadmin		Use Case Type: Essential, Detail	
Scenario: The superadmin wants to manage accounts of users (create,update,delete).			
Trigger: The superadmin wants to view details of the user.			
Normal Flow of Events: 1. The superadmin opens the application and requests to log in. 2. The superadmin enters username & password to the system. 3. The system navigates to the dashboard. 4. The superadmin clicks the user’s sidebar function. 5. The superadmin can now view and manage the users in the system. 6. The system ends.			
Sub Flows:			
Alternative/Exceptional Flows:			

Table 4.1.2 Manage Users

Use Case Name: Manage Inventories	ID:3		Important Level: High
Primary Actor: Administrator, Staff		Use Case Type: Essential, Detail	
Scenario: The staff and/or admin wants to create, update and/or delete properties in the database.			
Brief Description: This use case describes how the admin and/or staff manages inventory of properties.			
Trigger: The staff and/or admin wants to create, update and/or delete properties in the database.			
Normal Flow of Events: 1. The admin/staff opens the application and requests to log in. 2. The admin/staff enters username & password to the system. 3. The system navigates to the dashboard. 4. The admin/staff clicks the category's sidebar function. 5. The admin clicks the certain property			

Use Case Name: Manage Inventories	ID:3	Important Level: High
Primary Actor: Administrator, Staff	Use Case Type: Essential, Detail	
Scenario: The staff and/or admin wants to create, update and/or delete properties in the database.		
Brief Description: This use case describes how the admin and/or staff manages inventory of properties.		
Trigger: The staff and/or admin wants to create, update and/or delete properties in the database.		
5. The admin/staff can now view and manage the properties in the system. 6. The system ends.		
Sub Flows:		
Alternative/Exceptional Flows:		

Table 4.1.3 Manage Inventories

Use Case Name: Manage Transactions	ID:4	Important Level: High
Primary Actor: Administrator, Staff	Use Case Type: Essential, Detail	
Scenario: The staff/admin wants to view all transactions made by the rentees.		
Brief Description: This use case describes how the admin/staff manages all transactions made by the rentees.		
Trigger: The admin wants to view all transactions made by the rentees.		
Normal Flow of Events: 1. The admin/staff opens the application and requests to log in. 2. The admin/staff enters username & password to the system. 3. The system navigates to the dashboard. 5. The system displayed a calendar with colored days on it (‘Pending’ , ‘To Pay’, ‘Approved’, ‘Completed’, ‘Canceled’) 5. The admin/staff can now view all transactions made by the rentees. 6. The system ends.		
Sub Flows:		

Use Case Name: Manage Transactions	ID:4	Important Level: High
Primary Actor: Administrator, Staff	Use Case Type: Essential, Detail	
Scenario: The staff/admin wants to view all transactions made by the rentees.		
Brief Description: This use case describes how the admin/staff manages all transactions made by the rentees.		
Trigger: The admin wants to view all transactions made by the rentees.		
Alternative/Exceptional Flows:		

Table 4.1.4 Manage Reservations

Use Case Name: Messages	ID:5	Important Level: Medium
Primary Actor: Superadmin , Admin & Cashier	Use Case Type: Essential, Detail	
Scenario: Staff want to chat directly to admin/staff when they have a problem.		
Brief Description: This use case describes how admin, rentee and staff chats together.		
Trigger: The rentee wants to chat to the admin		
Normal Flow of Events: 1. The admin and staff open the application and requests to log in. 2. The admin and staff logged in with their username & password to the system. 3. The system navigates to the dashboard. 4. The admin and staff clicks the ‘Messages’ function in the topbar 5. The system displayed a messenger feature. 5. The admin, staff, and rentee can now chat with each other. 6. The system ends.		
Sub Flows:		
Alternative/Exceptional Flows:		

Table 4.1.5 Messages

Use Case Name: Receive Rentee's Payment	ID:6	Important Level: High
Primary Actor: Cashier	Use Case Type: Essential, Detail	
Scenario: Cashier receives payments from the rentee.		
Brief Description: This use case describes how a cashier receives payments from the rentees.		
Trigger: The cashier wants to confirm the transaction by receiving the payment from the rentee.		
Normal Flow of Events: 1. The rentee goes to the cashier and shows a digital receipt (with Qr Code). 2. The cashier logs in to the system 3. The cashier scans the digital receipt given by the rentee. 4. The system displayed a panel with rentee's all requested property and its amount to be paid. 5. The rentee goes to the cashier and hands over the payment amount of its rented property. 5. The system prompts the staff accounts requesting to prepare the rented property. 5. The rentee can now get their requested property. 6. The system ends.		
Sub Flows:		
Alternative/Exceptional Flows:		

Table 4.1.6 Receive Rentee's Payments

Use Case Name: Browse Properties	ID:7	Important Level: High
Primary Actor: Rentee	Use Case Type: Essential, Detail	
Scenario: Rentee wants to rent/borrow properties.		
Brief Description: This use case describes how rentee rents properties.		
Trigger: The rentee wants to rent a property in Mater Dei College.		
Normal Flow of Events:		

Use Case Name: Browse Properties	ID:7	Important Level: High
Primary Actor: Rentee	Use Case Type: Essential, Detail	
Scenario: Rentee wants to rent/borrow properties.		
Brief Description: This use case describes how rentee rents properties.		
Trigger: The rentee wants to rent a property in Mater Dei College.		
1. The rentee opens the system. 2. The system navigates to the welcome page. 3. The rentee clicks category from categories given by the system. 4. The system displayed all property filtered with the category selected by the rentee. 5. The rentee adds it on the cart and fills up a form with his credentials. 5. The system added his selected cart in the pending rents waiting for its approval from the administrator. 5. The rentee checks again the pending request and now its approved by the administrator 6. The rentee clicks checkout 7. The system generates a QR Code 8. The rentee goes to the cashier to scan its digital receipt (with QR Code) 9. The rentee hands over the payment 10. The rente gets his requested property 11. The system ends.		
Sub Flows:		
Alternative/Exceptional Flows:		

Table 4.1.7 Browse Properties

Use Case Name: Create Account	ID:8	Important Level: High
Primary Actor: Superadmin	Use Case Type: Essential, Detail	
Scenario: The Superadmin wants to create an account.		
Brief Description: This use case describes how the Superadmin creates an account for the users.		
Trigger: The Superadmin wants to create an account.		
Normal Flow of Events: 1. The Superadmin opens the system. 2. The system navigates to the login panel. 3. The Superadmin logs in with their credentials. 4. The system navigates to the dashboard. 5. The Superadmin clicks a button in the sidebar labeled “Users”. 6. The system displays all users with their roles.. 7. The Superadmin clicks a button labeled as “Add New User”. 8. The system displays a form with fields for the new user's information. 9. The Superadmin fills out all the fields and then submits the form. 10. The system saves the user and displays the new user in the "Manage Users" section. 11. The Superadmin clicks the approve button and confirms. 12. The system displays a success message. 13. The newly created user can now log in to the system . 14. The system ends.		
Sub Flows:		
Alternative/Exceptional Flows:		

Table 4.1.8 Create Account

Use Case Name: Approving & Declining Availability of Properties	ID: 9	Important Level: High
Primary Actor: Superadmin & Admin	Use Case Type: Essential, Detail	
Scenario: The Superadmin & Admin wants to approve and/or decline requests of the rentees.		
Brief Description: This use case describes how the Superadmin & Admin approves and/or declines requests of the rentees.		
Trigger: The Superadmin & Admin wants to approve and/or decline requests of the rentees.		
Normal Flow of Events: 1. The Superadmin & Admin opens the system. 2. The system navigates to the login panel. 3. The Superadmin & Admin logs in with their credentials. 4. The system navigates to the dashboard and displays a calendar. 5. The admin clicks a filtered day in the calendar. 6. The admin clicks a menu button from the sidebar and labeled as "Reservations". 7. The system navigates to the Reservations page. 8. The admin clicks the approve button, and the system displays a modal. 9. The system displays all of the information about the request. 10. The admin confirms the request by clicking the confirm button. 11. The system confirms the rentee's request, and it is now shown on the rentee's "Tracking" page. 12. The system ends.		
Sub Flows:		
Alternative/Exceptional Flows:		

Table 4.1.9 Approving & Declining Requests of Rentee

4.2 Design of Software, Systems, Product, and Processes

Developers show a certain view of what to be expected in the system. It is useful for the developers to guide them on how they develop the system. It serves as the blueprint of the system.

Output and User-Interface Design

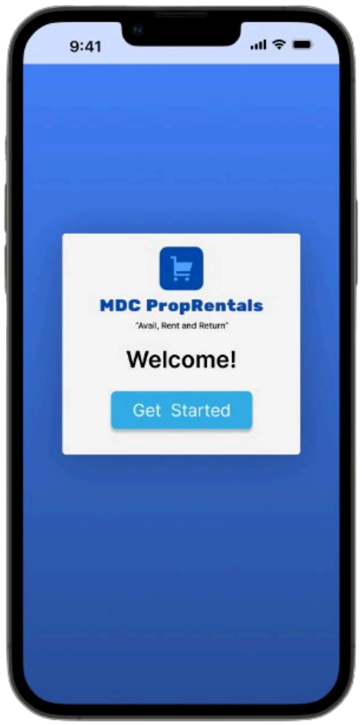


Figure 4.2.1 Welcome Page

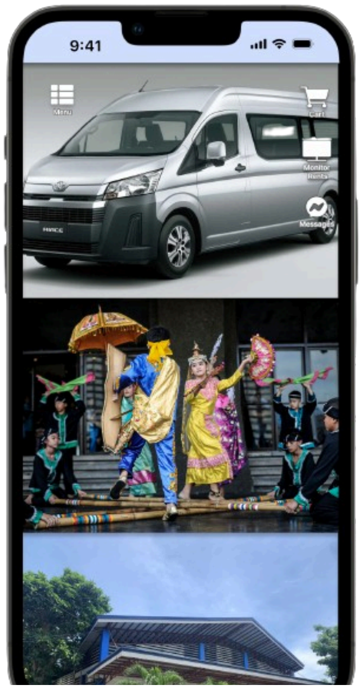


Figure 4.2.2 Home Page

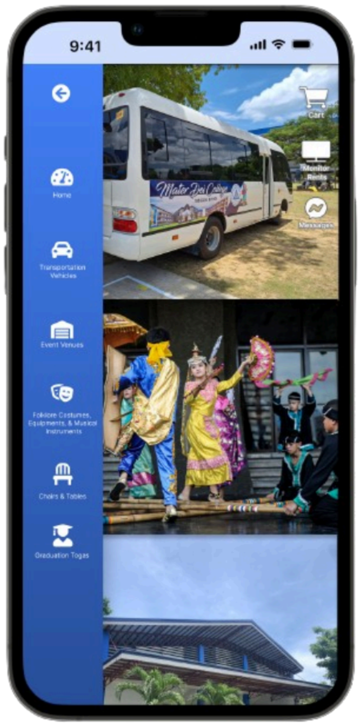


Figure 4.2.3 Main Menu

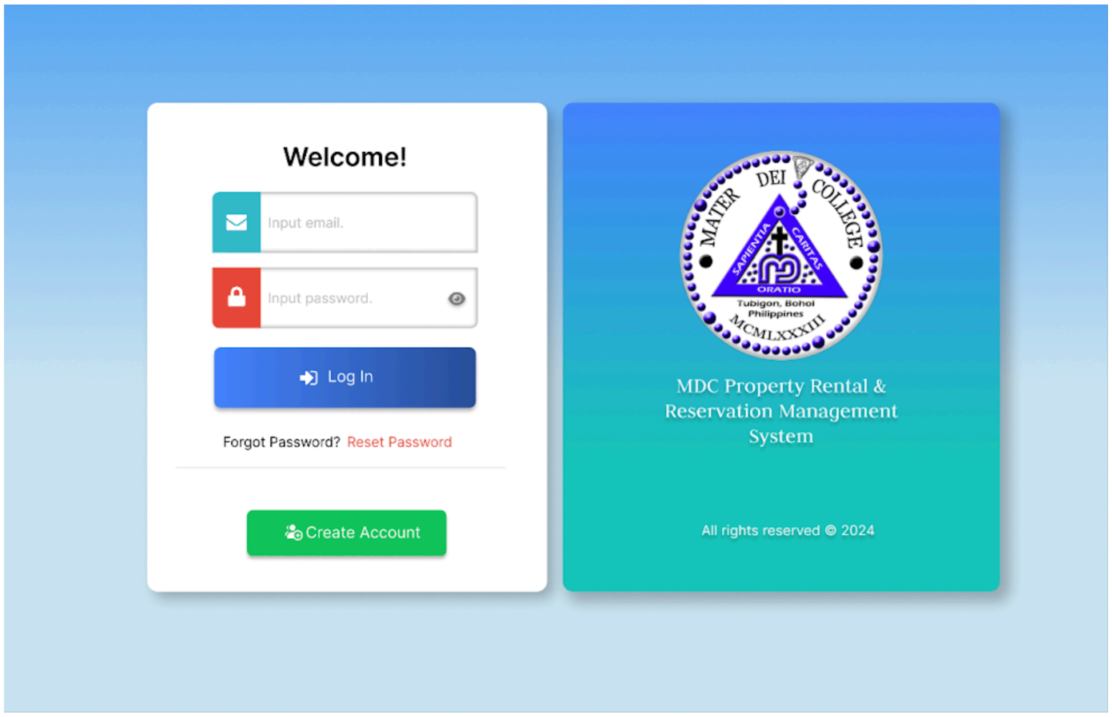

Welcome!
Input email.
Input password.
Log In
Forgot Password? Reset Password
Create Account
MATER DEI COLLEGE
SAPIENTIA CARITAS ORATIO
Tubigon, Bohol Philippines
MCMLXXXIII
MDC Property Rental & Reservation Management System
All rights reserved © 2024

Figure 4.2.4 Admin Login Panel

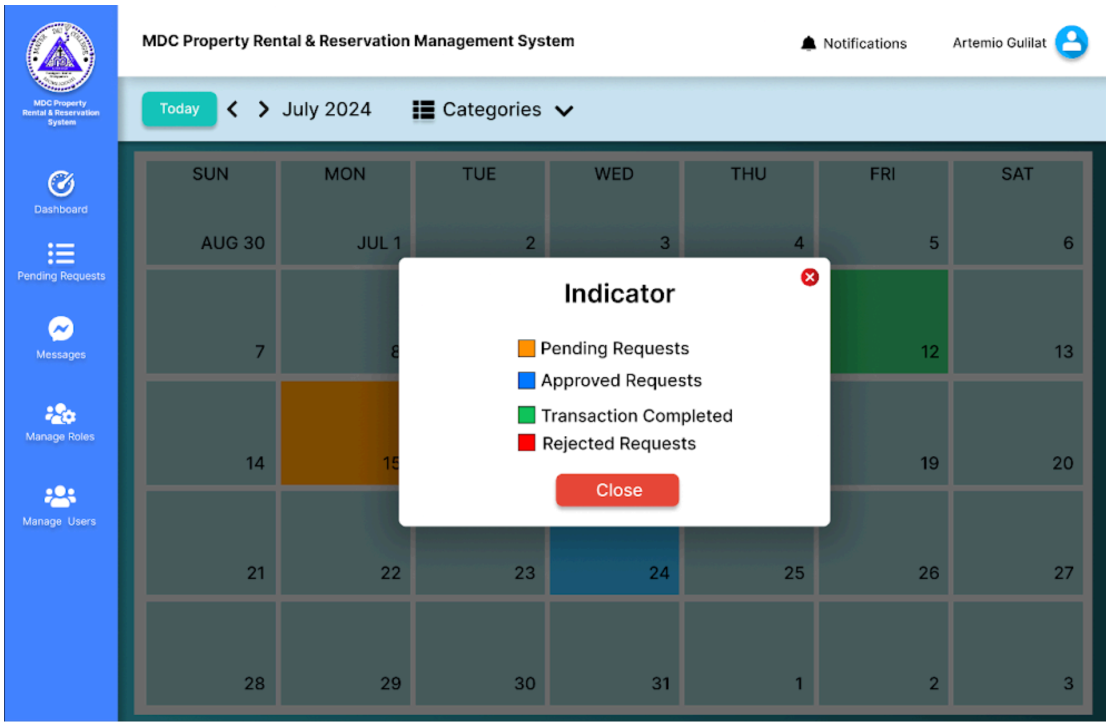


Figure 4.2.5 Admin Dashboard

4.3 Entity Relationship Diagram

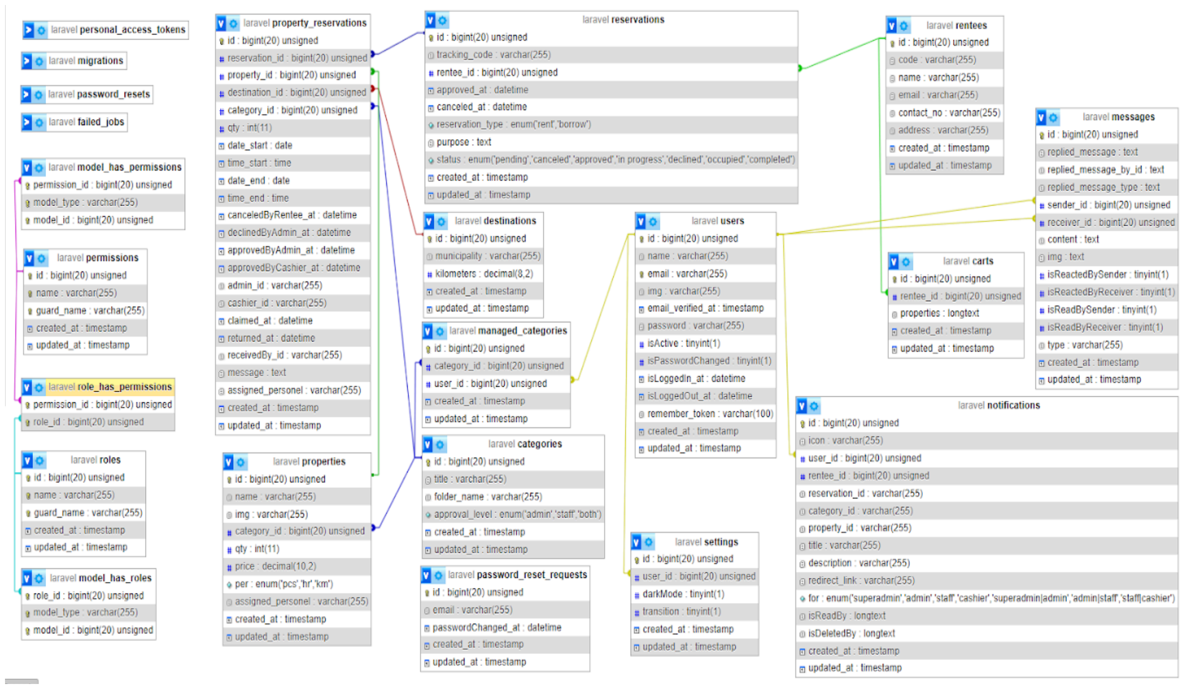


Figure 4.3.1 ER Diagram

4.4 Data Dictionary

To further understand the data that is being stored in the database, the data dictionary will help to show the fields of the tables.

Users Table			
Field Name	**Data Type**	**Length**	**Description**
id	bigint	20	ID number of users, Primary Key
name	varchar	255	Name of the user
email	varchar	255	Email of the user
img	varchar	255	Name of the image of the user's profile picture.
email_verified_at	datetime		Date of when the user's email is verified.
password	varchar	12	Password of the user.
isPasswordChanged	tinyint	1	Identifies if the user changed its password.
isLoggedIn_at	datetime		Identifies if the user is logged in.
isLoggedOut_at	datetime		Identifies if the user is logged out.
remember_token	token		
created_at	timestamp		
updated_at	timestamp		

Table 4.4.1 Database Table of Users

Notifications Table			
Field Name	**Data Type**	**Length**	**Description**
id	bigint	20	ID of notifications, Primary Key
icon	varchar	255	Name of the image
user_id	bigint	20	ID of the user, Foreign Key
rentee_id	bigint	20	ID of the rentee, Foreign Key
reservation_id	bigint	20	ID of the reservation table, Foreign Key
category_id	bigint	20	ID of the categories table, Foreign Key
proeprty_id	bigint	20	ID of the property, Foreign Key
title	text		Title of the notification.
description	text		Description of notification
redirect_link	text		Link of the notification where the receiver is being redirected when clicked.
for	enum	'superadmin','admin','staff','cashier','superadmin|admin','admin|staff','staff|cashier'	Receiver of the notification.
isReadBy	longtext		List of user_id if the notification is seen or not by the user

isDeletedBy	longtext		List of user_id if the notification is deleted or not by the user
created_at	timestamp		
updated_at	timestamp		

Table 4.4.2 Database Table of Notification

Properties Table			
Field Name	**Data Type**	**Length**	**Description**
id	bigint	20	ID of the property, Primary Key.
name	varchar	255	Name of the property, NOT NULL.
img	varchar	255	Name of the img.
category_id	bigint	20	ID of the category of which this property belongs, Foreign Key.
qty	integer	11	Quantity of properties available in the inventory.
price	decimal	10,2	Amount cost to rent this property.
per	enum		
assigned_personel	varchar	255	Name of the personnel that is assigned to this property.
created_at	timestamp		

updated_at	timestamp		

Table 4.4.3 Database Table of Properties

Categories Table			
Field Name	**Data Type**	**Length**	**Description**
id	bigint	20	ID of the Category, Primary Key.
title	varchar	255	Title of the Category
folder_name	varchar	255	Name of the folder where the images are being stored.
approval_level	enum	'admin','staff','both'	List of the user who can approve a reserved property.
created_at	timestamp		
updated_at	timestamp		

Table 4.4.4 Database Table of Categories

Carts Table			
Field Name	**Data Type**	**Length**	**Description**
id	bigint	20	ID of the cart, Primary Key.
rentee_id	bigint	20	ID of the rentee table, Foreign Key.
properties	longtext		List of properties that are stored in the cart.
created_at	timestamp		

updated_at	timestamp		

Table 4.4.5 Database Table of Carts

Reservations Table			
Field Name	**Data Type**	**Length**	**Description**
id	bigint	20	ID of the transaction made by the rentee, Primary Key.
tracking_code	varchar	8	Tracking code of the reservation.
rentee_id	bigint	20	ID of the rentees table, Foreign Key.
approved_at	datetime		Date when the reservation is approved.
canceled_at	datetime		Date when the reservation is canceled.
reservation_type	enum	'rent','borrow'	Type of the reservation.
purpose	text		Purpose of the reservation.
status	enum	'pending','canceled','approved','in progress','declined','occupied','completed'	Status of the reservation.
created_at	timestamp		
updated_at	timestamp		

Table 4.4.6 Database Table of Transactions

Messages Table			
Field Name	**Data Type**	**Length**	**Description**
id	bigint	20	ID of the messages, Primary Key.
replied_message	text		Text of the replied message from the sender.
replied_message_id	text		Id of the receiver who replied to the message.
replied_message_type	text		Type of the message, ‘text’,’like’,’img’
sender_id	bigint	20	Sender’s id, Foreign key.
receiver_id	bigint	20	Receiver’s id, Foreign key.
content	text		Message of the sender.
img	text		Name of the image that is sent by the sender.
isReactedBySender	tinyint		Boolean, if the sender reacted to the sent message.
isReactedByReceiver	tinyint		Boolean, if the receiver reacted to the sent message.
isReadBySender	tinyint	1	Boolean, if the sender saw the message or not.
isReadByReceiver	tinyint	1	Boolean, if the

			receiver saw the message or not.
type	text		Type of message,‘text’,’like’ ,’img’
created_at	timestamp		
updated_at	timestamp		

Table 4.4.7 Database Table of Messages

Destinations Table			
Field Name	**Data Type**	**Length**	**Description**
id	bigint	20	ID of the detination, Primary Key.
municipality	varchar	255	Name of the municipality.
kilometers	decimal	10,2	Kilometers from Tubigon to the destination.
created_at	timestamp		
updated_at	timestamp		

Table 4.4.8 Database Table of Destinations

Rentees Table			
Field Name	**Data Type**	**Length**	**Description**
id	bigint	20	ID of the cart, Primary Key.
code	varchar	8	Code of the rentee

name	varchar	255	Name of the rentee.
email	varchar	255	Email of the rentee.
contact_no	varchar	12	Contact number of the rentee.
address	varchar	255	Address of the rentee.
created_at	timestamp		
updated_at	timestamp		

Table 4.4.9 Database Table of Rentees

Managed Categories Table			
Field Name	**Data Type**	**Length**	**Description**
id	bigint	20	ID of the cart, Primary Key.
category_id	bigint	20	ID of the categories table, Foreign Key.
user_id	bigint	20	ID of the users table, Foreign Key.
created_at	timestamp		
updated_at	timestamp		

Table 4.4.10 Database Table of Managed Categories

Settings Table			
Field Name	**Data Type**	**Length**	**Description**
id	bigint	20	ID of the cart, Primary Key.
user_id	bigint	20	ID of the users table, Foreign Key.
darkMode	tinyint	1	Boolean, switch for turning dark mode

			feature.
transition	tinyint	1	Boolean, switch for turning animations of transition features.
created_at	timestamp		
updated_at	timestamp		

Table 4.4.11 Database Table of Settings

Password Reset Request Table			
Field Name	**Data Type**	**Length**	**Description**
id	bigint	20	ID of the cart, Primary Key.
email	varchar	255	Email of the user.
passwordChanged_at	datetime		Date of when the password is changed by the Admin or Superadmin.
created_at	timestamp		
updated_at	timestamp		

Table 4.4.12 Database Table of Password Reset Requests

Roles Table			
Field Name	**Data Type**	**Length**	**Description**
id	bigint	20	ID of the role, Primary Key.
type	varchar	255	Specific type of role such as Admin, staff, cashier, etc.
guard_name	varchar	255	Name of the guard

description	text		Description of a certain role.
created_at	timestamp		
updated_at	timestamp		

Table 4.4.13 Database Table of Roles

Permissions Table			
Field Name	**Data Type**	**Length**	**Description**
id	bigint	20	ID of the permission, Primary Key.
name	varchar	255	Names of permissions such as can view users, can view, can write, can edit, etc.
guard_name	varchar	255	Name of the guard.
description	text		Description of a certain role.
created_at	timestamp		
updated_at	timestamp		

Table 4.4.14 Database Table of Permissions

Role has Permissions Table			
Field Name	**Data Type**	**Length**	**Description**
role_id	bigint	20	ID of the role, Foreign Key.
permission_id	bigint	20	ID of the permission, Foreign Key.

Table 4.4.15 Database Table of Roles has Permissions

Model has Roles Table			
Field Name	**Data Type**	**Length**	**Description**
role_id	bigint	20	ID of role, Foreign Key
model_type	varchar	255	Type of model.
model_id	bigint	20	Id number of model Foreign Key.

Table 4.4.16 Database Table of Model has Roles

Model has Permissions Table			
Field Name	**Data Type**	**Length**	**Description**
permission_id	bigint	20	ID number of permissions Foreign Key
model_type	varchar	255	Type of model
model_id	bigint	20	ID number of model, Foreign Key.

Table 4.4.17 Database Table of Model has Permissions

4.5 Development Testing

Evaluating the system assesses the outcome of this project whether the desired objectives have been achieved. System testing is conducted to check the functionality of the system if it works properly or as expected, it is important so that the overall functionality of the system will be improved.

4.6 Implementation Plan

An implementation plan in an inventory system is meant to offer a framework through which the invention will be put in practice in our school. It highlights who, when and how what needs to be done in terms of migration of data, the employees, training and testing phases. In this strategy, it is possible to avoid the unnecessary proprietary losses of production and discontented clients' claims, even in the increased prices, and define the level of control over inventory and lower costs for it, which will also influence the level of client satisfaction.

REFERENCES

Yanxiu Wang et. al. (2015). *Sports Equipment Rental System of School Based on B/S Structure Design and Implementation*

https://www.atlantis-press.com/proceedings/etmhs-15/19265

EZO Company (2019). *Equipment Rental Management for the Education Industry*

https://ezo.io/wp-content/uploads/2019/07/EZRentOut-Education-Industry-Whitepaper-V1.0.pdf

Venyooz Org (2012) *SchoolSpace.us*

https://www.schoolspace.us/