Web Based System for Photography Management System

for Pathum L Weerasinghe

By

Samoda Wijesooriya IM/2021/059

A report submitted in partial fulfillment of the requirements for the degree of Bachelor of Science Honors in Management and Information Technology (B.Sc. MIT)

Name of the Supervisor:

Dr. Dinesh Asanka

Department of Industrial Management

Faculty of Science

University of Kelaniya

Sri Lanka

Declaration

I hereby certify that this project and the all the artifacts associated with it is my own work and it has not been submitted before nor is currently being submitted for any other degree program.

Full name of the student: Manikge Samoda Leshan Wijesooriya

Student No: IM/2021/059

Signature of the student:  Date: 14/11/2024

Name of the supervisor: Dr. Dinesh Asanka

Signature of the supervisor: ……. Date …….

Acknowledgement

I would like to express my sincere gratitude to **Dr. Dinesh Asanka** for his invaluable support and guidance throughout this project. His supervision and insights were instrumental in shaping the Photography Management System and pushing me to achieve my best.

I am also deeply grateful to the department and staff who provided me with this opportunity to work on a real-world problem and apply my technological knowledge to create meaningful solutions.

My heartfelt thanks go to the judges from the poster presentations and interim presentations for their constructive feedback, which helped refine and enhance the system.

Lastly, I am truly thankful to my friends for their continuous support, shared knowledge, and encouragement throughout this project.

Abstract

This report details the design and implementation of a comprehensive Web-Based Photography Management System for Pathum L. Weerasinghe's photography studio. The project addresses the significant inefficiencies and limitations inherent in the studio's existing manual processes, which rely on fragmented communication channels and manual record-keeping for bookings, payments, and client interactions. Through an in-depth analysis of current workflows, including the development of Business Activity Models (BAM), Context Diagrams, and Data Flow Diagrams (DFDs), the report identifies key pain points such as double bookings, poor payment tracking, and lack of real-time progress updates for clients.

Based on these findings, the system requirements were defined to support a seamless digital transformation. The proposed solution features a user-friendly web interface that enables clients to browse categorized portfolios, register accounts, schedule appointments via an integrated calendar, track the progress of their photo albums in real time, and make secure online payments. An administrative panel empowers the photographer to efficiently manage bookings, communications, and financial transactions. Several business system options were evaluated, with a custom web-based platform selected as the optimal approach to balance functionality, scalability, and user experience.

The developed system streamlines studio operations, enhances customer satisfaction, and supports business growth by leveraging modern web technologies. This report demonstrates how tailored digital solutions can resolve real-world business challenges, laying a foundation for future enhancements and innovation in the photography industry.

Table of Contents

List of Figures

List of Tables

Chapter 1 – Introduction

Outline of the Chapter

1.1 Introduction about the photography management system

1.2 Business Process

1.2 Problem Definition

1.4 Overview

1.5 Aims and Objectives

1.6 Scope of the Project

1.7 Project Feasibility

- 1.7.1 Technical Feasibility
- 1.7.2 Legal Feasibility
- 1.7.3 Economical Feasibility

1.8 Organization of the Report

- 1.8.1 Chapter 1 - introduction
- 1.8.2 Chapter 2 – System Analysis
- 1.8.3 Chapter 3 – System Design
- 1.8.4 Chapter 4 – System Development
- 1.8.5 Chapter 5 - Implementation
- 1.8.6 Chapter 6 - Conclusion
- 1.8.7 Chapter 7 – References

1.9 Summary

1.1 Introduction to the photography management system

Figure 1-Logo

Pathum L. Weerasinghe, a professional employed photographer working with around nearly three years, has problems in coordinating bookings and payments, and communicating with customers primarily via Facebook and WhatsApp. The photoshoot business owner has experienced exponential growth, thus requiring an efficient system to enhance the clients' experience. Using this system client expected to achieve these features: virtual gallery, simplified booking/scheduling services, user account, progress tracking in an album, and payments – to improve the business process and customers' experience.

Contact Details:

Name of the Client: Pathum L Weerasinghe

Mobile: 076 451 8697

Address: Kegalle, Sri Lanka

Email: pethumlakmal6@gmail.com

1.2 Business Process

Currently, Pathum L. Weerasinghe's photography business operates through a manual process. Clients browse his recent images to find the type of work they need, then contact him through Facebook or WhatsApp. Following initial contact, Pathum sends package details as a PDF for clients to review and choose a suitable option. To secure a booking, clients make an initial payment of 20,000 rupees and select an available date on his calendar. The remaining balance is due before the event. After the event, Pathum begins processing the album, but clients must check in via WhatsApp to inquire about progress. Once the album is ready, clients receive a drive link to view and select images for printing. The proposed Photography Service Management System addresses these limitations, streamlining communication, scheduling, payment, and progress tracking for a smoother, more efficient experience.

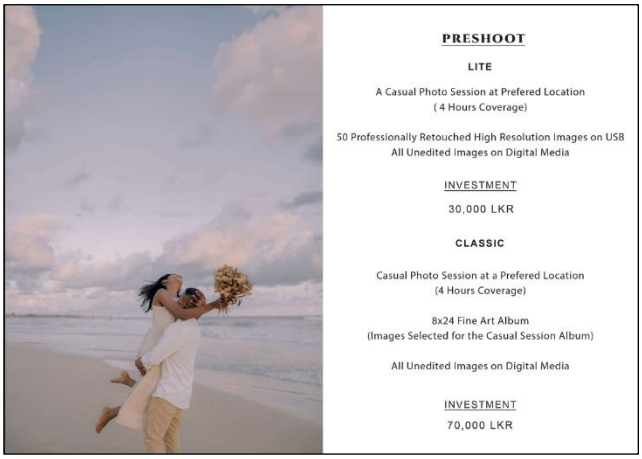


Figure 2-Package 2

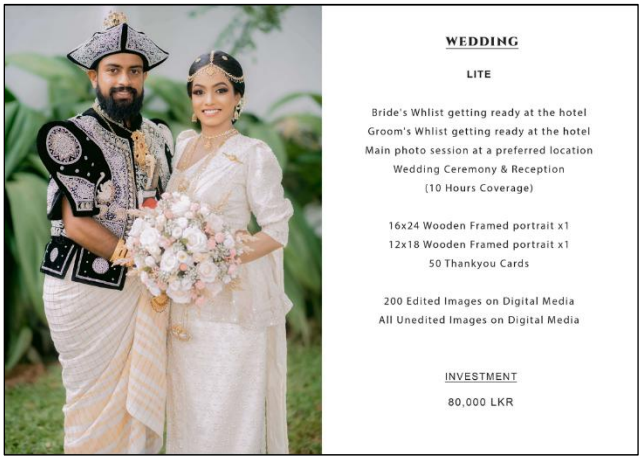


Figure 3-Package 1

1.2 Problem Definition

Currently the client relies on a manual system for bookings and all other processes. Communication across multiple platforms (Facebook, Instagram, WhatsApp), resulting in fragmented threads, missed details, and double bookings.

- Lack of proper payment tracking, leading to errors with deposits and balances.
- Inefficient organization of customer information, slowing retrieval processes and reducing productivity.
- No centralized gallery, requiring customers to scroll through various posts to view the photographer’s past work.
- No independent booking tracking or album progress updates for clients, causing uncertainty.

- Manual scheduling, which delays photo processing and product delivery.

These issues highlight the need for a more efficient, integrated system to optimize operations and enhance client satisfaction.

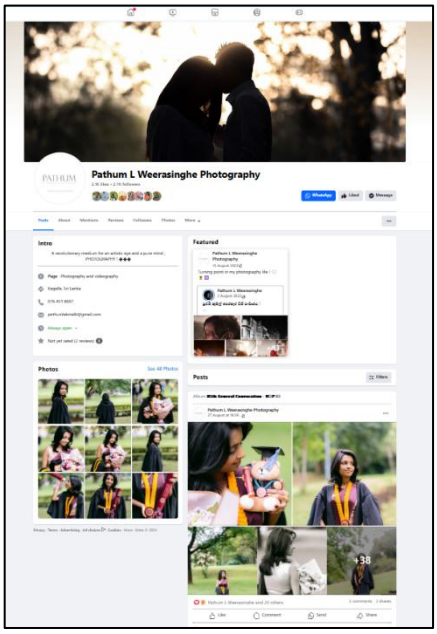


Figure 4-FB Page 1

Figure 5-FB Page 2

Figure 6-FB Page 3

1.4 Overview

The system digitizes the traditional manner in which the photography studio operates through the use of paper with regard to client portfolio, package/service offering, appointment making, and payment. It makes appointment and payment easier for the photographer while on the other hand it makes access, communication, and satisfaction easier for the client through technology.

1.5 Aims and Objectives

To modernize the photography studio management process by transitioning from a manual system to an integrated web platform that enhances customer satisfaction and engagement.

- Develop a user-friendly web interface showcasing the photographer's portfolio.
- Implement secure registration and login functionalities for customers.
- Enable real-time appointment booking to prevent double bookings.

- Facilitate secure online payments with automated reminders for fees.
- Provide an admin panel for photographers to manage bookings and communications effectively.

1.6 Scope of the Project

The project will involve developing a web application for managing studio and customers where, categorized gallery of past work, User account creation and log in system, Calendar to enable scheduling, Payment mechanism where customers can securely pay for services online, Online admin panel where general managers can monitor communicating with the customers and financial transactions taking place.

1.7 Project Feasibility

1.7.1 Technical Feasibility

The current computer setup in Pathum L. Weerasinghe's photography business supports the implementation of the proposed Photography Management System. To achieve functionality and adaptability the system will employ mainly Web 2.0 technologies including: HTML, CSS, JavaScript, PHP and MySQL. Milestones in the integration with secure payment gateways will be checked for availability.

1.7.2 Legal Feasibility

Automated tools to detect potential AH referring to individuals to avoid GDPR-related issues will be used, and the copyright of photographs and other customer data will be considered.

1.7.3 Economic Feasibility

A cost-benefit analysis will assess the reality of the costs of development against the benefits of expected gains in efficiency and customer satisfaction, in addition to potential revenues from additional services.

1.8 Organization of the Report

This section provides a brief explanation about the content of the report.

1.8.1 Chapter 1 – Introduction

Chapter 1 provides a brief description about the client, their current process, the problem identified with the current process to come with the below designed system, aims and the objective of the project, the scope of the project to which it is spread, and the project feasibility is discussed.

1.8.2 Chapter 2 – System Analysis

Chapter 2 presents a clear analysis of the existing system and the requirement analysis of the system. It further analyses the available business system options (BSOs) and defines the best option to proceed with.

1.8.3 Chapter 3 – System Design

After identifying the project requirements and functionality, the next step is to visually represent it through diagrams that illustrate how various system components interact and function together. This section presents the database design, detailing the tables that the system will use, and provides an in-depth view of the system's operation through Graphical User Interfaces (GUIs).

1.8.4 Chapter 4 – System Development

This chapter presents the most critical phase of the project: system development. It outlines the elementary processes involved and discusses the technologies and tools used throughout the development. Each technology and application utilized in the project is clearly explained, along with their respective implementations. Additionally, the chapter showcases the user interface components, emphasizing their organization and usability. It also includes a detailed explanation of the reports and main dashboards integrated into the system.

1.8.5 Chapter 5 – Implementation

This chapter outlines the implementation process of the Photography Management System. It begins with an introduction that explains how the implementation was carried out. An installation guide is provided to assist users in setting up the system for their specific needs. Additionally, a user guide is included to help users navigate the system and understand its features and functionalities.

1.8.6 Chapter 6 – Conclusion

The conclusion summarizes the final aspects of the project, highlighting the objectives that were successfully achieved during development. It also evaluates key quality factors such as usability,

accessibility, reliability, and user-friendliness. Additionally, the chapter addresses any limitations encountered and outlines potential future improvements and extensions for the system.

1.8.7 Chapter 7 – References

This chapter provides the references used in creating the report.

1.9 Summary

The features and nature of the current process were analyzed, identifying key issues and objectives. This led to a clear definition of the system's scope, boundaries, and project feasibility, ensuring that all requirements align with the overall goals.

Chapter 2 – System Analysis

Outline of the Chapter

2.1 User Stories

2.2 BAM for current system

2.3 Context Diagram for current system

2.4 Level 01 DFD for current system

2.5 Level 02 DFD for current system

2.6 Software Requirement Specification (SRS)

2.7 Business System Options (BSOs)

2.8 Functional Requirements Vs. BSO

2.9 Non-Functional Requirements Vs. BSO

2.10 Selected BSO with justification

2.11 Summary

2.1 User Stories

ID	User Stories
US01	**As a customer,** I want to inquire about photography services via WhatsApp so that I can quickly get information about packages and availability.
US02	**As a photographer,** I want to manually send available time slots and price packages to customers via Facebook or WhatsApp so that they can choose a package that suits their needs.
US03	**As a customer,** I want to receive a PDF with pricing and package details from the photographer so that I can review my options before make a decision.
US04	**As a customer,** I want to inform the photographer of my chosen package and desired date so that they can check availability and confirm my booking.
US05	**As a photographer,** I want to mark the confirmed bookings on my calendar manually so that I can keep track of my scheduled appointments.
US06	**As a customer,** I want to pay 20,000 Rs. deposit to confirm my booking so that I can secure the photographer for my event.
US07	**As a photographer,** I want to update the customers on the editing progress of their photos so that they are informed about the status of their album.
US08	**As a customer,** I want to ask the photographer about the progress of my photo album so that I can have updates on when I can expect to receive it.
US09	**As a photographer,** I want to deliver soft copies of photos via Google Drive so that customers can easily access their images after the event.
US10	**As a customer,** I want to receive hard copies of my photos in a physical album after selecting my preferred images from the soft copies so that I can have a tangible keepsake of my event.
US11	**As a photographer,** I want to manually track payments and client details to ensure I have all necessary information for my bookings, despite the potential for errors and confusion.

Table 1-User Stories for Current System

2.2 BAM for current system

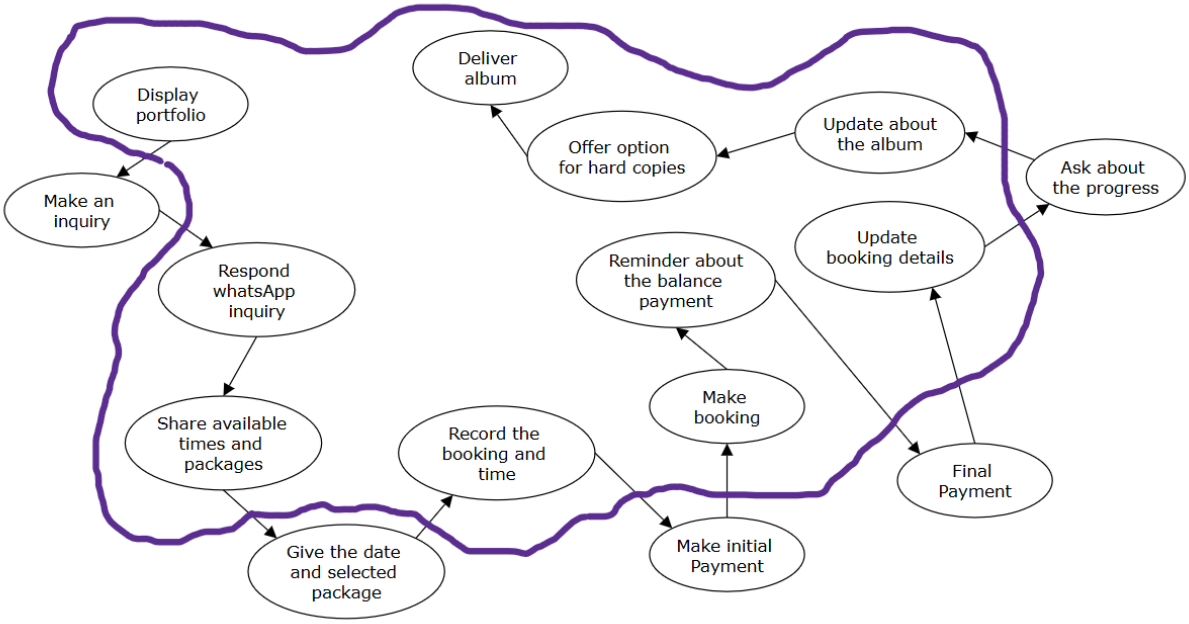


Figure 7-BAM for Current System

2.3 Context diagram

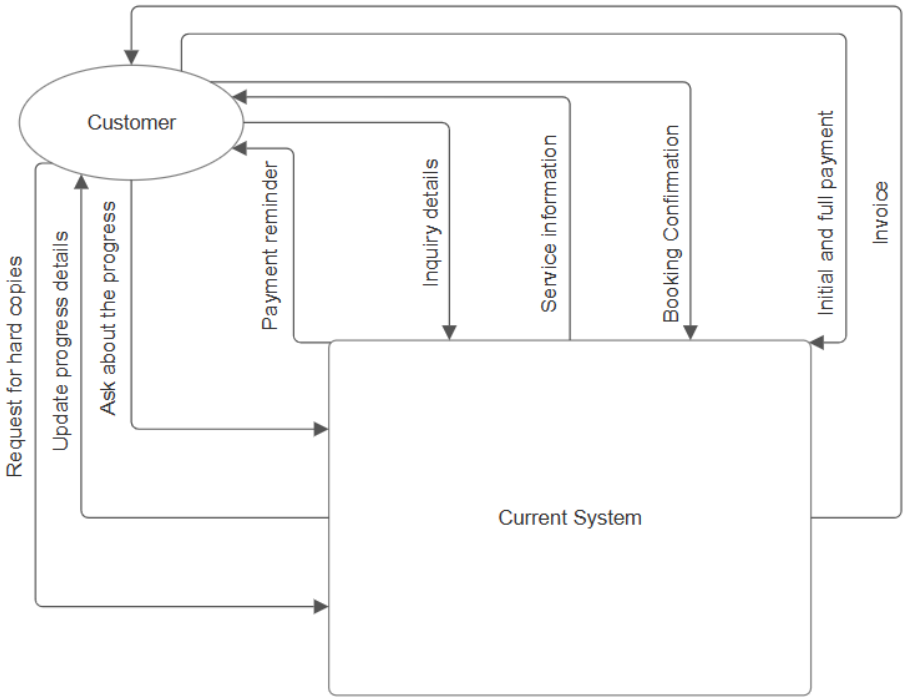


Figure 8-Context Diagram for Current System

2.4 Level 01 DFD

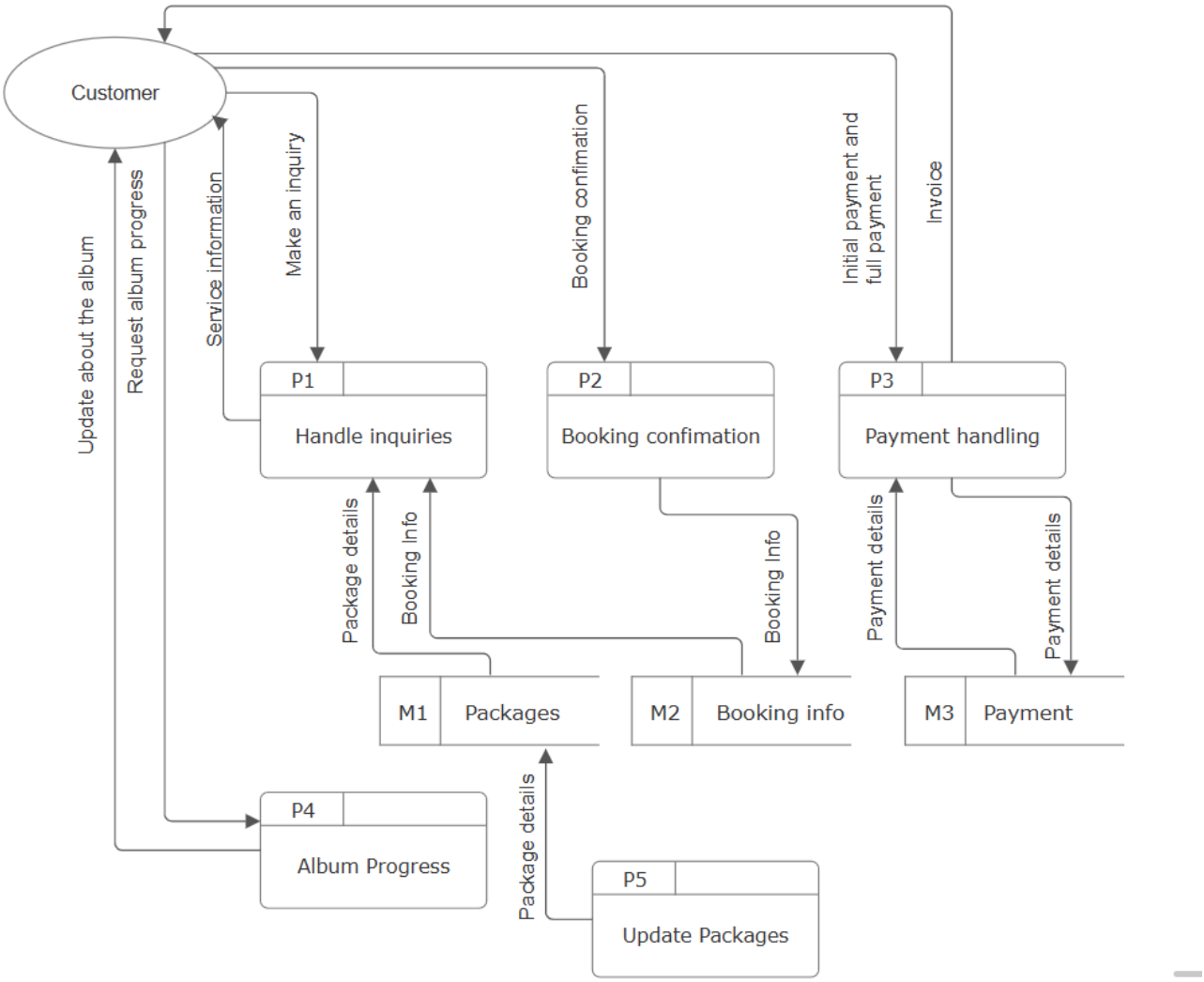


Figure 9-DFD L1 for Current System

2.5 Level 02 DFD

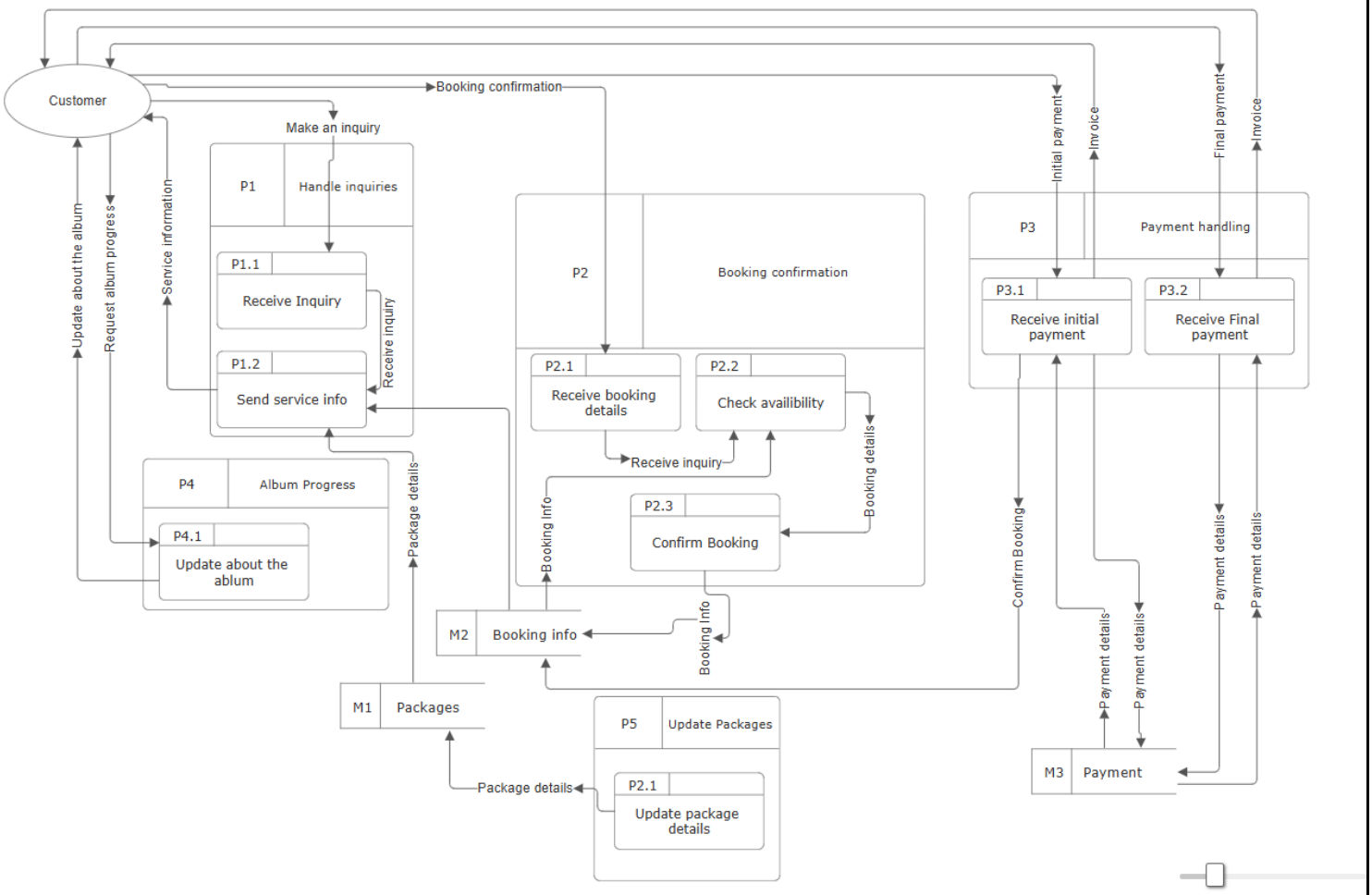


Figure 10-DFD L2 for Current System

2.6 Software Requirements Specification

Functional Requirements

ID	Requirement Description	Quantification
1	User Registration and Authentication: The system must enable new users to register and log in.	Support up to 500 new users per day; authentication response time < 5 seconds.
2	Gallery Display and Filtering: Users can view a gallery of the photographer's past work and filter by categories.	Display at least 50 images at a time; filtering results in under 1 second.
3	Event Package Information: Provide detailed information about photography packages	Display at least 5 packages; loading time for package info < 1 second.
4	Package Customization: Logged-in users can customize photography packages.	Real-time updates reflected in under 2 seconds.
5	Appointment Booking: Users can book appointments by selecting a date and time.	Accommodate up to 500 simultaneous bookings; return available time slots in under 2 seconds.
6	Payment Management: Handle payment transactions with multiple methods.	99% success rate; confirmation within 2 seconds for credit cards; bank transfers confirmed within 24 hours.
7	Booking Management: Photographers can manage bookings and update statuses.	Handle up to 200 bookings per hour; updates reflected within 2 seconds.
8	Notification and Messaging: Send automated notifications regarding bookings and payments.	Notifications sent within 1 minute; support up to 1000 notifications per hour.
9	Admin Dashboard: Provide photographers access to manage bookings, view stats, and access reports.	Generate reports on demand within 5 seconds.
10	Add Additional Payments: Photographers can add amounts to the final payment.	N/A (no specific quantification provided).

Figure 11-Functional requirements

Non-Functional Requirements

ID	Requirement Description	Quantification
1	Performance: The system must ensure a smooth user experience with minimal delays.	All user interactions should have a response time of less than 2 seconds under normal load conditions.
2	Usability: The interface should be easy to navigate and understand, making it simple for everyone to use the system.	User satisfaction rating should be at least 80% in usability surveys conducted post-implementation.
3	Security: Implement security measures to protect user data and financial transactions; verify user permissions for image access.	Achieve compliance with security standards for information security management.
4	Scalability: The system should be able to handle an increasing number of users and bookings without degradation in performance.	Support growth to handle up to 10,000 users concurrently without performance issues.
5	Reliability: The system must be dependable and available for users without frequent downtimes.	Ensure system uptime of 99.9% per month.
6	Maintainability: The system should be easy to maintain and update as needed.	The code should allow for updates or fixes to be implemented within 24 hours of identifying an issue.

Figure 12-Non-Functional Requirements

2.7 Business System Options (BSO)

The Business Option Specifications (BSOs) are sources that identify possible ways to accomplish the objectives of the photography studio management system. They make up several strategies through which the business could implement solutions that will solve the challenges of the current system, and they compare different important factors including the time, size and costs of implementing the solutions. When evaluating these approaches, the business can determine the most effective means through which the management system can be upgraded, how trade satisfaction can be improved besides efficiency in its operations.

The considered BSOs can be listed as below.

- **BSO 1: Mobile Application Solution**
- **BSO 2: Use of Existing Scheduling Platforms (e.g., Calendly or Square)**
- **BSO 3: Comprehensive Web-Based Platform**

2.7.1 BSO 1

Functional Description:

Create an application that will allow them to schedule appointments, to have a chance to see the portfolio of the photographer, to make payments and to receive information of booking through the mobile application. This application could entail push notifications such as booking notifications, payment notifications, and album notifications.

High-Level Technical Description:

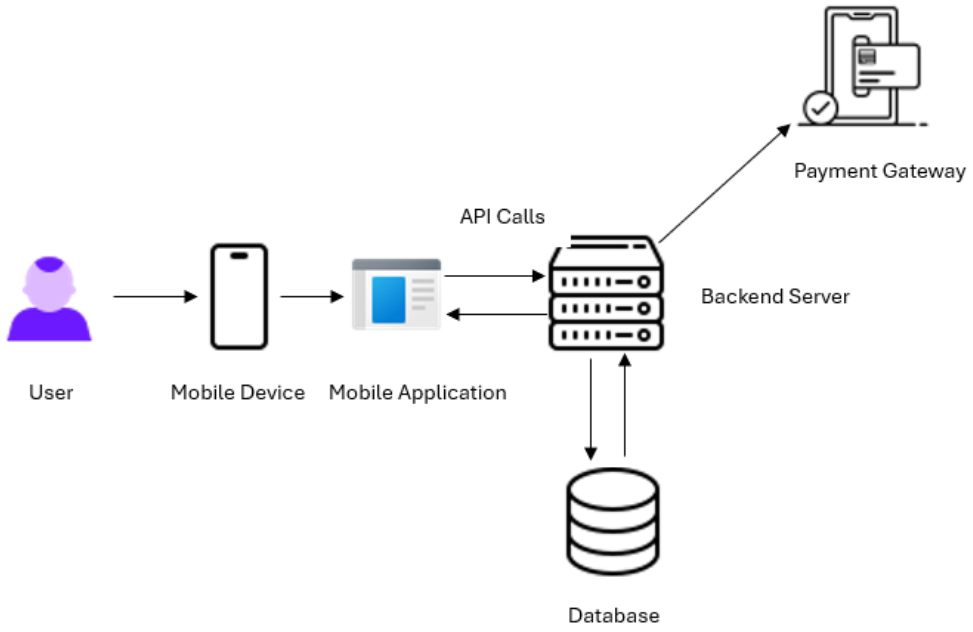


Figure 41 – BSO1 mobile application

Benefits:

- Allows clients to book appointments, view portfolios, and make payments directly from their mobile devices.
- Real-time updates on bookings, payments, and album progress.
- Enables easy interaction between clients and the photographer.

Cons:

- Higher upfront investment required for mobile app development and maintenance.
- Not all customers may have compatible devices or prefer mobile apps.
- Mobile applications may require regular updates, adding to the long-term costs and management overhead.

2.7.2 BSO 2

Functional Description:

Employ Calendly or Square, an existing online platform to schedule appointments, transact and communicate with the clients. This solution could be integrated on the photographer's website or social media pages, as an efficient measure without having to build completely new software.

High-Level Technical Description:

Figure 42 – BSO2 Third Party Application

Benefits:

- Less initial development cost compared to building a custom system.
- Minimal setting time, like Calendly or Square provide ready-to-use solutions for booking and payments.
- Maintenance and updates are handled by the platform providers.
- Accessible to a wider audience.

Cons:

- Reduced flexibility to add customized features like full portfolio management or album tracking.
- Some platforms may require ongoing subscription fees, which can increase over time.
- Relying on external services may introduce potential issues if the service provider changes its terms or discontinues services.

2.7.3 BSO 3

Functional Description:

Design a complex and intuitive real estate web presence that would effectively encompass the needs of the photography studio. Within this structure, customers could come in, register, view a collection of previous projects, schedule actual appointments at their own convenience, make secure forms of payment, and follow booking and album status on their own accord. Clients would have an admin portal for booking, payments, contact details and modification of the listed photograph Portfolios.

High-Level Technical Description:

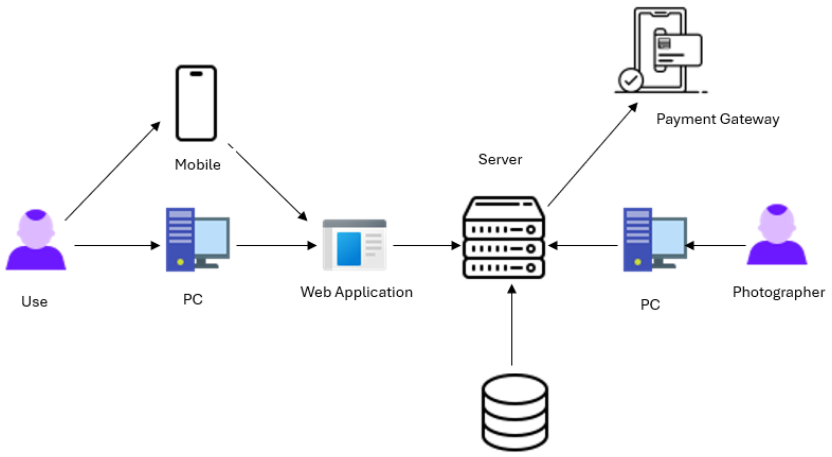


Figure 43 – BSO3 Web Application

Benefits:

- Combines booking, payment, portfolio display, and album progress tracking into a single platform.
- Allows for future customization and expansion as business needs evolve.
- Accessible from any device with an internet connection.
- Supports custom design and branding.

Cons:

- Involves substantial time and financial resources.
- Require a longer timeframe for development and testing.
- Responsibility for updates, security, and maintenance lies with the photographer or their team, which may require technical resources.

2.8 Functional Requirements Vs BSO

ID	Requirements	BSO1	BSO2	BSO3
1	The system **shall** enable new users to register by providing essential information. Registered users shall log in to the system.	✔	✔	✔
2	The registration process shall support up to 500 new users per day.	✔	✘	✔
3	The system shall allow users to view a gallery showcasing the photographer's past work.	✔	✔	✔
4	The system should enable users to filter images by categories.	✔	✘	✔
5	The gallery should display at least 50 images at a time, with filtering options.	✔	✔	✔
6	The system shall provide detailed information about various photography packages.	✔	✘	✔
7	Users shall log in to view this information and customize packages according to their specific requirements.	✔	✔	✔
8	The system should display at least 5 different packages with comprehensive details, and the loading time for package information should be less than 1 second.	✔	✘	✔
9	Logged-in users shall customize photography packages based on their specific needs.	✔	✔	✔
10	The customization process should allow users to make changes in real-time, with updates reflected in under 2 seconds.	✔	✘	✔
11	The system shall allow users to book appointments for photography services by selecting a date and time from a calendar.	✔	✔	✔
12	The system shall check for availability and display time slots.	✔	✔	✔
13	The booking feature should accommodate up to 500 simultaneous bookings, returning available time slots in under 2 seconds.	✔	✘	✘

14	The system shall handle payment transactions, including initial and final payments.	✔	✔	✔
15	The system shall support multiple payment methods, including credit cards and bank deposits.	✔	✔	✔
16	Photographers shall check the payment receipt before confirming the booking.	✔	✔	✔
17	Payments should be processed in real time with a success rate of 99%, providing confirmation within 2 seconds.	✔	✔	✔
18	For bank transfers, confirmation of booking shall occur once the photographer verifies the payment receipt, which may take up to 24 hours.	✔	✘	✔
19	Photographers shall manage bookings; view booked dates and update the status of each booking. The system shall allow cancellations and modifications.	✔	✘	✘
20	The system should send automated notifications to users and photographers regarding booking confirmations, payment reminders, and event reminders.	✔	✔	✘
21	Photographers shall have access to an admin dashboard for managing bookings,	✔	✘	✔
22	The dashboard should provide real-time insights into bookings, generating reports on demand within 5 seconds.	✔	✘	✔

Table 2-BSO Justification for Functional Requirements

2.9 Non-functional Requirements

ID	Requirements	BSO1	BSO2	BSO3
1	The system shall ensure a smooth user experience with minimal delays.	✔	✔	✔
2	All user interactions should have a response time of less than 2 seconds under normal load conditions.	✔	✔	✔
3	The interface shall be easy to navigate and understand	✔	✔	✔
4	User satisfaction rating should be at least 80% in usability surveys	✔	✔	✔
5	The system shall implement security measures to protect user data and financial transactions.	✔	✔	✔
6	The gallery display shall ensure that images are secured to prevent unauthorized access.	✔	✔	✔
7	The system should verify user permissions before allowing access to specific images in the gallery.	✔	✔	✔
8	The system shall achieve security standards for information security management.	✔	✔	✔
9	The system should be able to handle an increasing number of users and bookings without degradation in performance.	✔	✔	✔
10	The system shall support growth to handle up to 10,000 users concurrently without performance issues.	✔	✔	✔
11	The system should be dependable and available for users without frequent downtimes.	✔	✔	✔
12	The system shall ensure an uptime of 99.9% per month.	✔	✔	✔
13	The booking feature should accommodate up to 500 simultaneous bookings, returning available time slots in under 2 seconds.	✔	✔	✔
14	The system should be easy to maintain and update as needed.	✔	✔	✔

15	The code shall allow for updates or fixes to be implemented within 24 hours of identifying an issue.	✔	✔	✔

Table 3-BSO Justification for Non-Functional Requirements

2.10 Selected BSO with Justification

The specific choice of **the Web-Based Solution (BSO1)** for the overall photography management system is justified by its capability to satisfy functional and non-functional requirements regarding improving the entire user and organizational experience. This solution is featured by intuitive navigation through which clients can easily and quickly register, view the portfolio of the selected photographer, adjust services to the required price and schedule an appointment online. Furthermore, it also authenticates online payments and affirms data security that can meet and enhance the security and reliability standards. The former feature accommodates increasing customer needs, and the latter – increases satisfaction due to the availability of a device. Altogether, the web-based solution gives the best and integrated option for this project to manage the photography services.

2.11 Summary

This section outlines the requirements identification of the current system and it helps to understand the current system for better development in the system design.

Chapter 03 – System Design

Outline of the Chapter

3.1 Functional and Nonfunctional Requirements satisfied by the selected BSO

3.2 User Stories for the Proposed System

3.3 Business Activity Model (BAM) for Proposed System

3.4 Context Diagram for Proposed System

3.5 Level 01 DFD for Proposed System

3.6 Level 02 DFD for Proposed System

3.7 Entity Matrix for Proposed System

3.8 Logical Data Structure

3.9 ER Diagram

3.10 Normalized ER diagram

3.11 Sample User Interfaces for the Proposed System

3.12 Summary

3.1 Functional and Nonfunctional Requirements satisfied by the selected BSO

Functional Requirements

ID	Requirements
1	The system shall enable new users to register by providing essential information. Registered users shall log in to the system.
2	The system shall allow users to view a gallery showcasing the photographer's past work.
3	The system should enable users to filter images by categories.
4	The system shall provide detailed information about various photography packages.
5	The system shall provide a chance to make a booking to the customer.
6	The system should have add-ons which can be included into the package.
7	Users shall log in to view this information and customize packages according to their specific requirements.
8	The system shall calculate the price of the package, if a user customized a package without choosing a base package.
9	Logged-in users shall customize photography packages based on their specific needs.
10	The customization process should allow users to make changes in real-time, with updates reflected in under 2 seconds.
11	The system shall allow users to book appointments for photography services by selecting a date and time from a calendar.
12	The system shall check for availability and display time slots.
13	The booking feature should accommodate up to 500 simultaneous bookings, returning available time slots in under 2 seconds.
14	The system shall handle payment transactions, including initial and final payments.
15	The system shall support multiple payment methods, including credit cards and bank deposits.
16	Photographers shall check the payment receipt before confirming the booking.
17	Payments should be processed in real time with a success rate of 99%, providing confirmation within 2 seconds.
18	For bank transfers, confirmation of booking shall occur once the photographer verifies the payment receipt, which may take up to 24 hours.
19	Photographers shall manage bookings; view booked dates and update the status of each booking. The system shall allow cancellations and modifications.

20	The system should send automated notifications to users and photographers regarding booking confirmations, payment reminders, and event reminders.
21	Photographers shall have access to an admin dashboard for managing bookings,
22	The dashboard should provide real-time insights into bookings, generating reports on demand within 5 seconds.

Table 4-Selected Functional Requirements

Non-Functional Requirements

ID	Requirements
1	The system shall ensure a smooth user experience with minimal delays.
2	All user interactions should have a response time of less than 2 seconds under normal load conditions.
3	The interface shall be easy to navigate and understand
4	User satisfaction rating should be at least 80% in usability surveys
5	The system shall implement security measures to protect user data and financial transactions.
6	The gallery display shall ensure that images are secured to prevent unauthorized access.
7	The system should verify user permissions before allowing access to specific images in the gallery.
8	The system shall achieve security standards for information security management.
9	The system should be able to handle an increasing number of users and bookings without degradation in performance.
10	The system shall support growth to handle up to 10,000 users concurrently without performance issues.
11	The system should be dependable and available for users without frequent downtimes.
12	The system shall ensure an uptime of 99.9% per month.
13	The booking feature should accommodate up to 500 simultaneous bookings, returning available time slots in under 2 seconds.
14	The system should be easy to maintain and update as needed.
15	The code shall allow for updates or fixes to be implemented within 24 hours of identifying an issue.

Table 5-Selected Non-Functional Requirements

3.2 User Stories for the Proposed System

ID	User Story
US01	As a **new user**, I want to create an account, so that I can access the services of the system.
US02	As a **registered user**, I want to log in to my account, so that I can access features like booking a photographer or viewing packages.
US03	As a **user**, I want to view the photographer's portfolio or gallery, so that I can see past work before making a booking decision.
US04	As a **user**, I want to browse the available photography packages, so that I can choose the one that best fits my event needs or customize a package.
US05	As a **registered user**, I want to select a package and book a date for my event using a calendar, so that I can confirm the photographer's availability.
US06	As a **registered user**, I want the system to check the availability of dates and time slots automatically, so that I can book a feasible date for my event.
US07	As a **registered user**, I want to make an initial payment when booking, so that I can secure the photographer’s services for my event.
US08	As a **registered user**, I want to receive notifications of the when my booking is confirmed, so that I know my event is fully scheduled through WhatsApp.
US09	As a **registered user**, I want to receive a reminder for the final payment before the event, so that I can ensure the booking is confirmed.
US10	As a **photographer**, I want to receive a notification when the user makes the final payment, so that I can confirm the booking.
US11	As a **photographer**, I want to receive a reminder on the event day, so that I can be prepared to attend and provide my services.
US12	As a **photographer**, I want to update the status of the album editing process in the system, so that the user can track progress.
US13	As a **registered user**, I want to check the progress of my event album online, so that I can stay informed about its status.
US14	As a **registered user**, I want to download the soft copies of my photos from the system, so that I can review and select images for the hard copy album using third party album storage.

US15	As a **photographer**, I want to receive a notification when the customer has finalized their photo selections for the album, so that I can proceed with printing or delivering hard copies.
US16	As a **photographer**, I want to track my financial transactions and bookings within the system, so that I can manage my business finances effectively.
US17	As a **photographer**, I want to see a summary of customer bookings, payments, and upcoming events, so that I can stay organized and plan my schedule.

Table 6-User Stories for Proposed System

3.3 Business Activity Model (BAM) for Proposed System

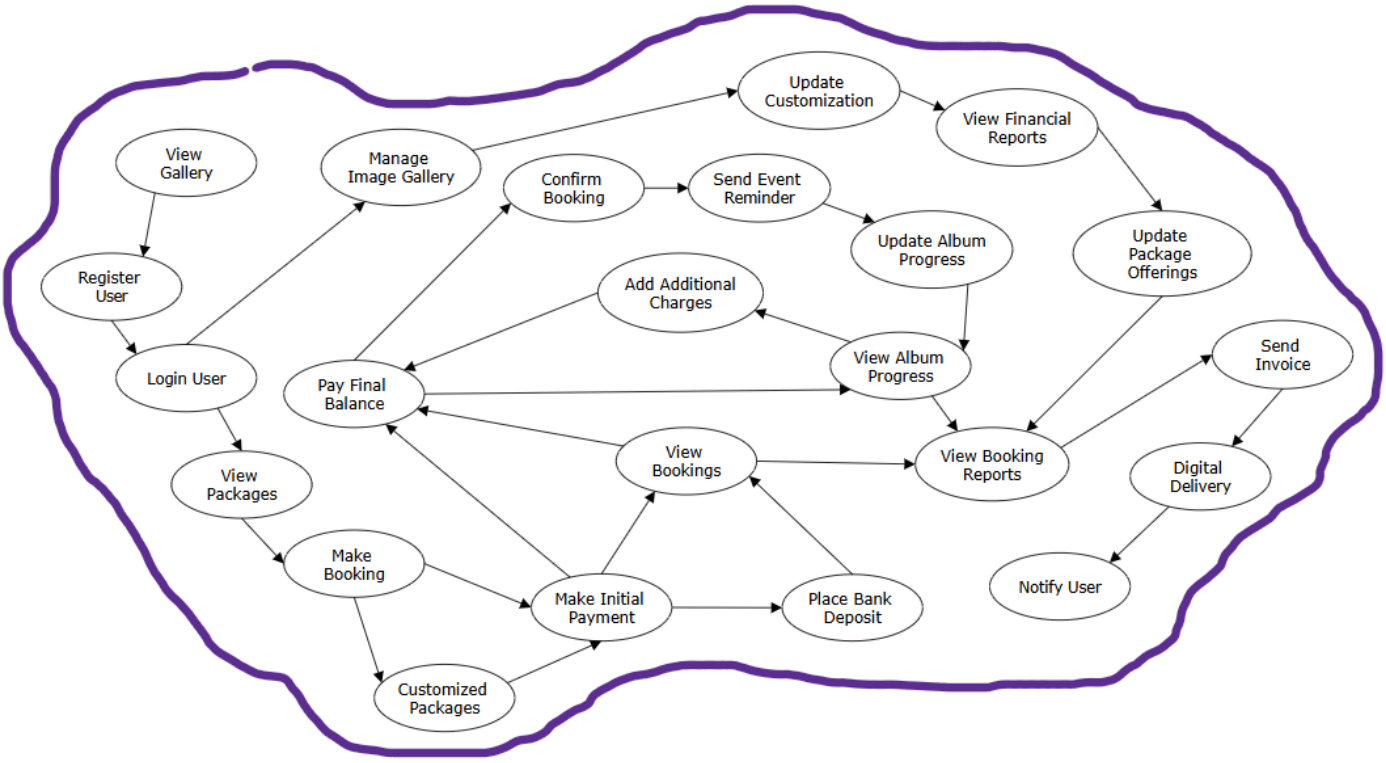


Figure 13-BAM for Proposed System

3.4 Context Diagram for Proposed System

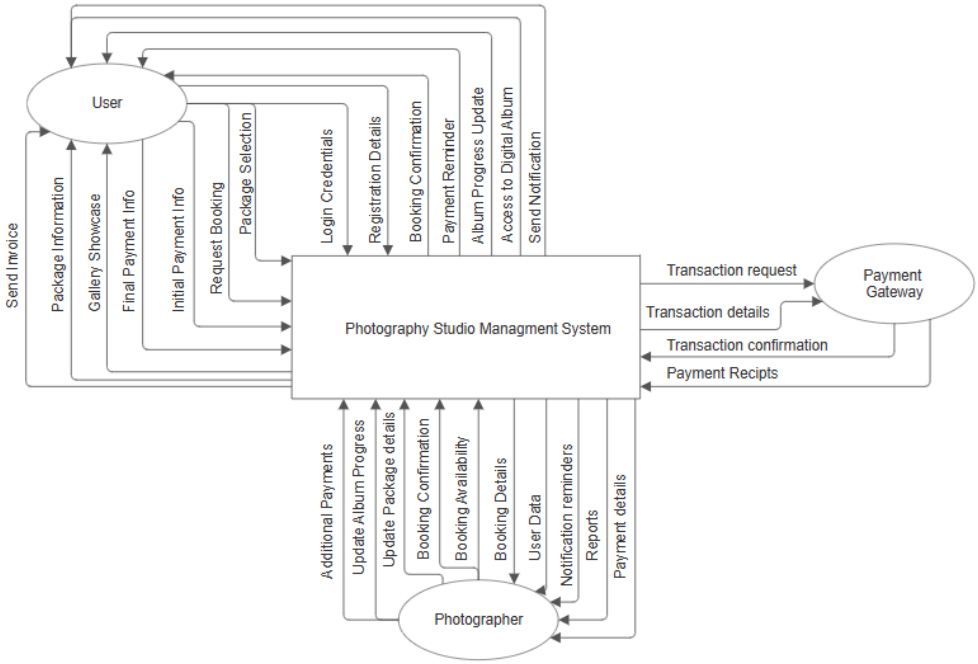


Figure 14-Context Diagram for Proposed System

3.5 Level 01 DFD for Proposed System

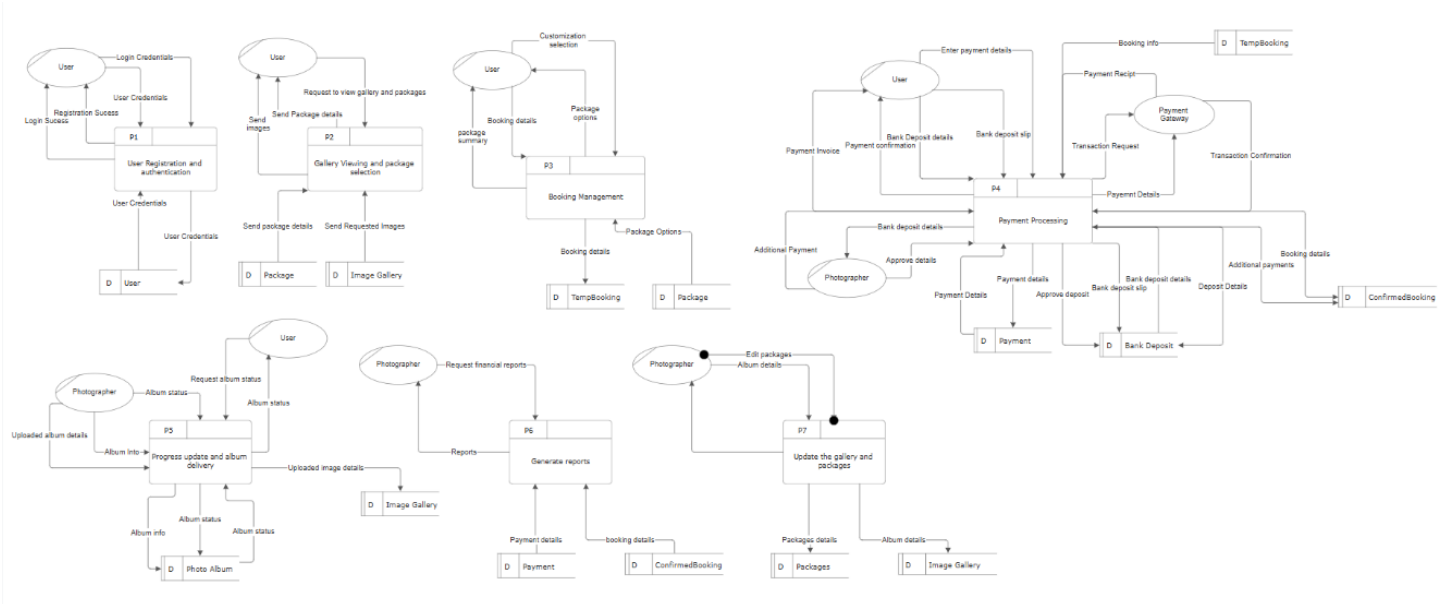


Figure 15-DFD L1 Diagram for Proposed System

User registration and authentication

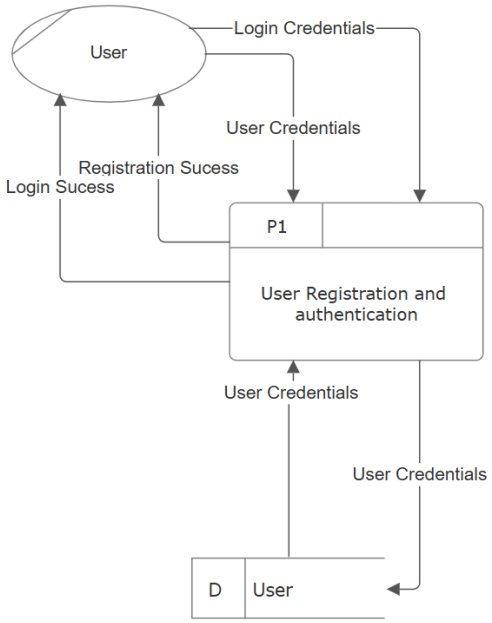


Figure 16-DFD L1 Process 1

Gallery view and package selection

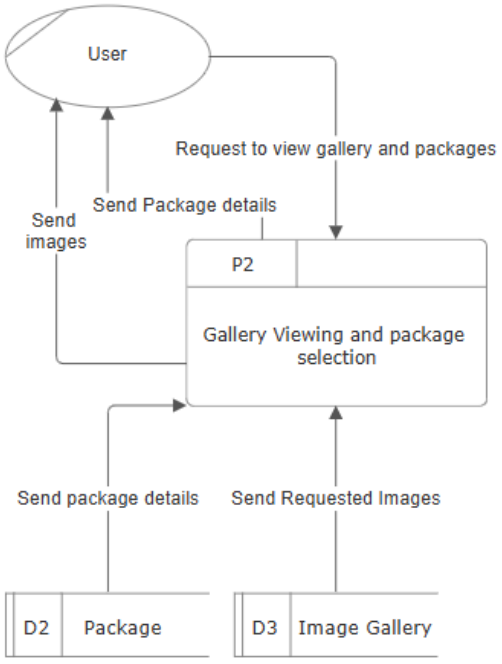


Figure 17-DFD L1 Process 2

Booking management

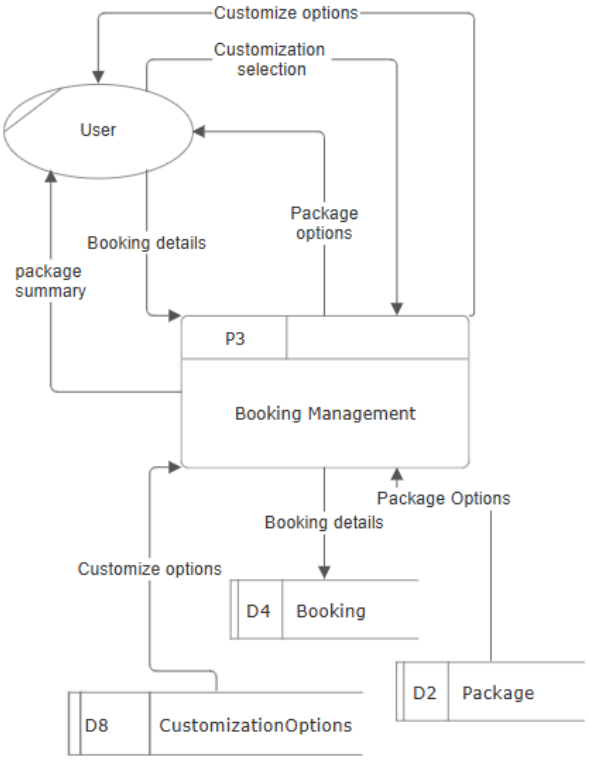


Figure 18-DFD L1 Process 3

Payment Processing

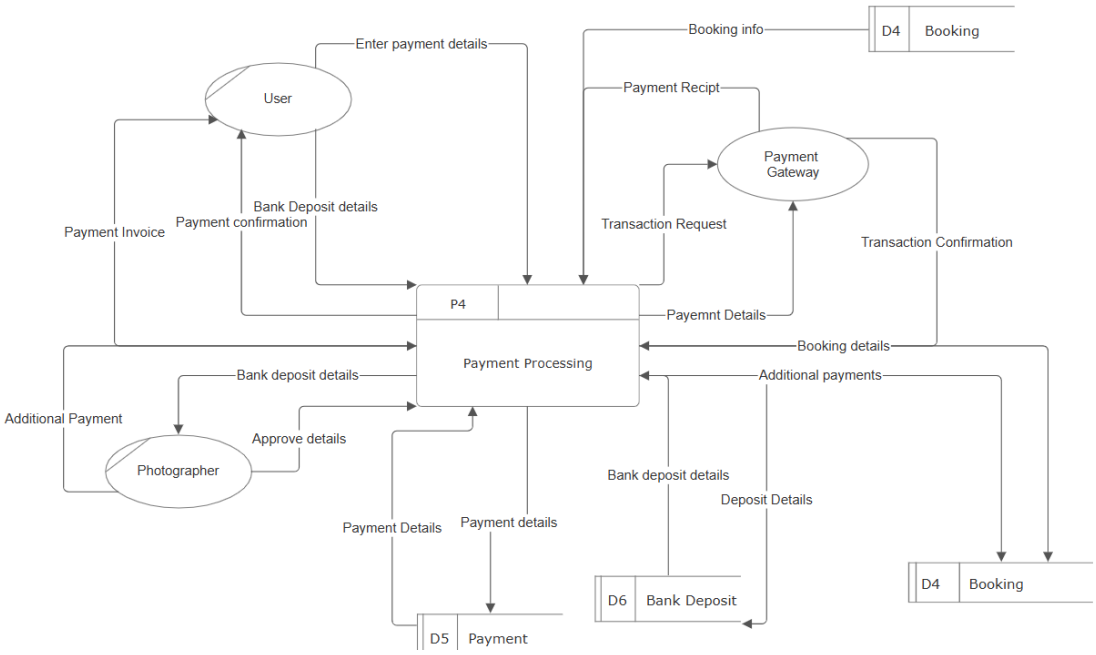


Figure 19-DFD L1 Process 4

Progress update and album delivery

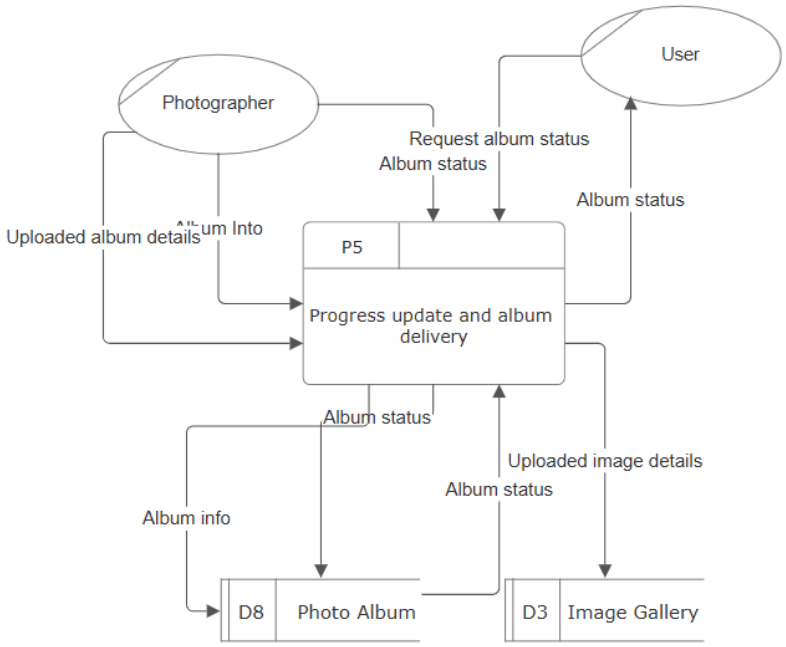


Figure 20-DFD L1 Process 5

Generating reports

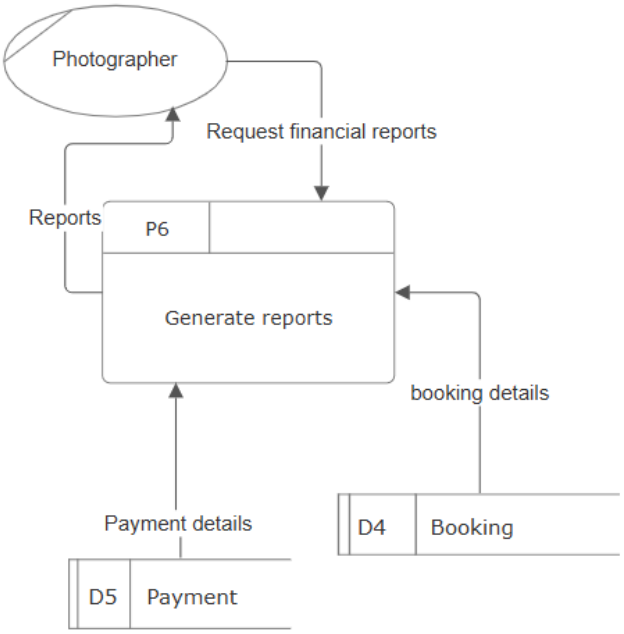


Figure 21-DFD L1 Process 6

Update package and gallery

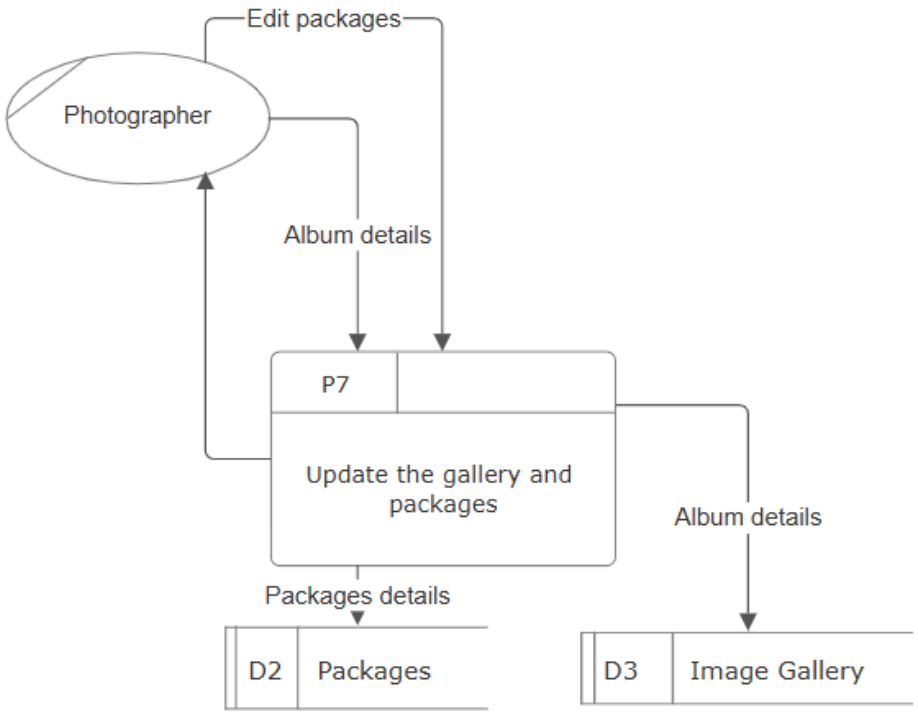


Figure 22-DFD L1 Process 7

3.6 Level 02 DFD for Proposed System

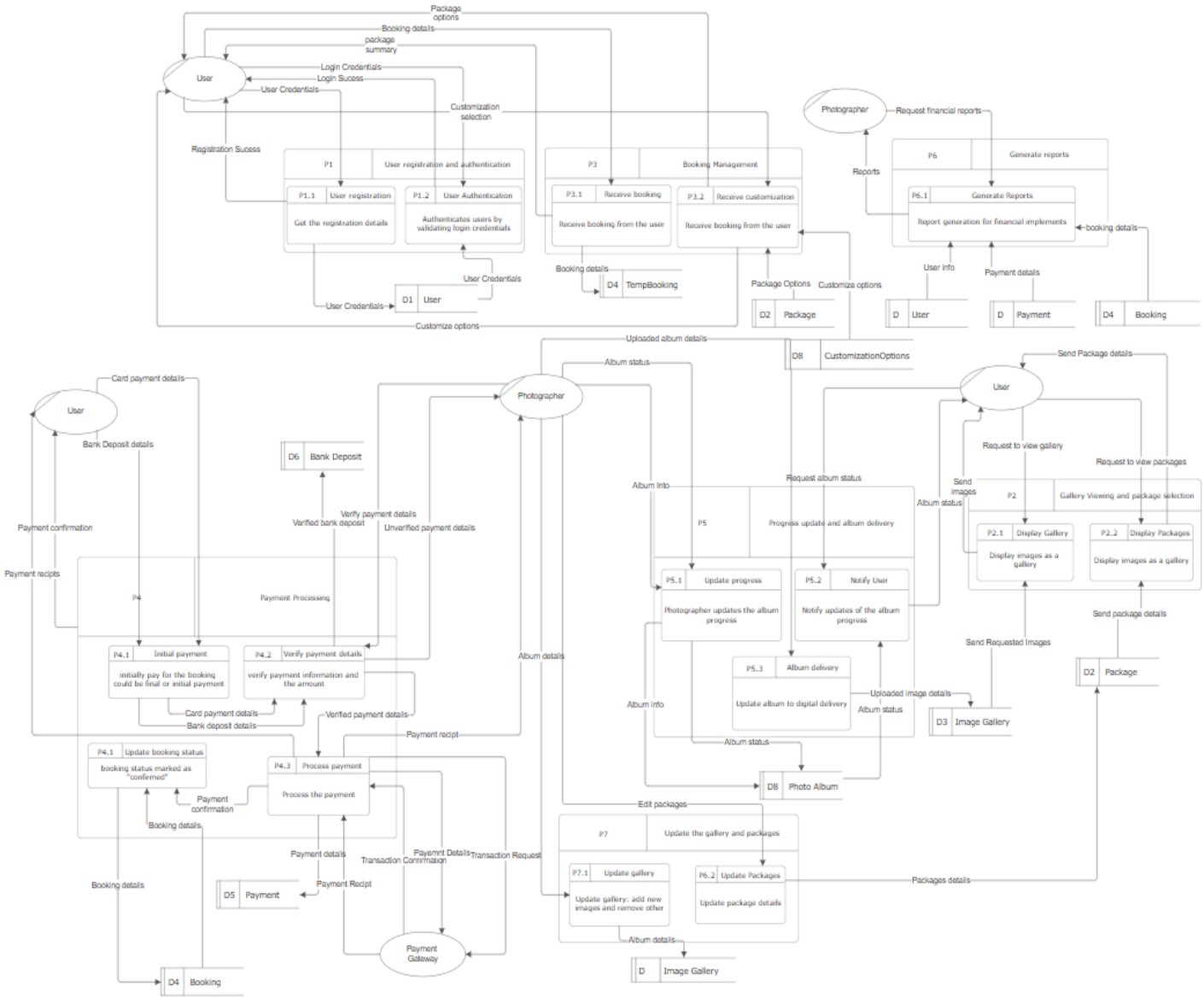


Figure 23-DFD L2 Diagram for proposed System

User registration and authentication

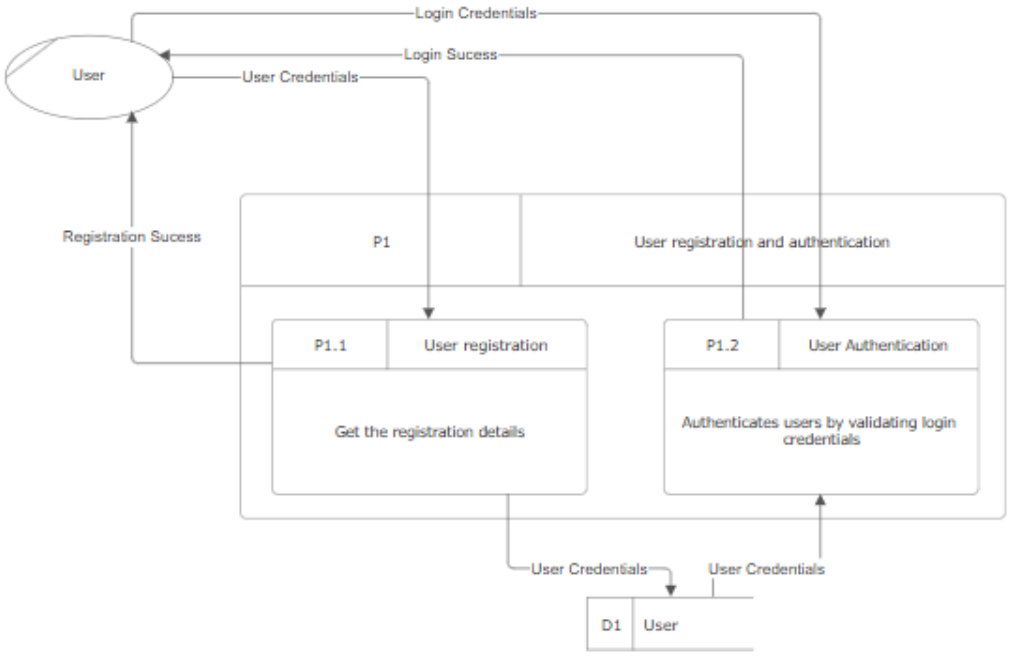


Figure 24-DFD L2 Process 1

Gallery viewing and package selection

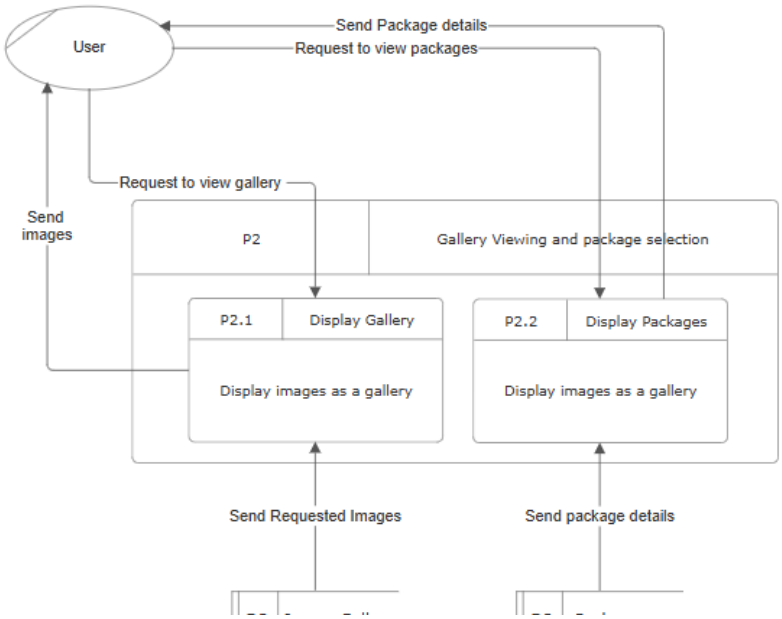


Figure 25-DFD L2 Process 2

Booking Management

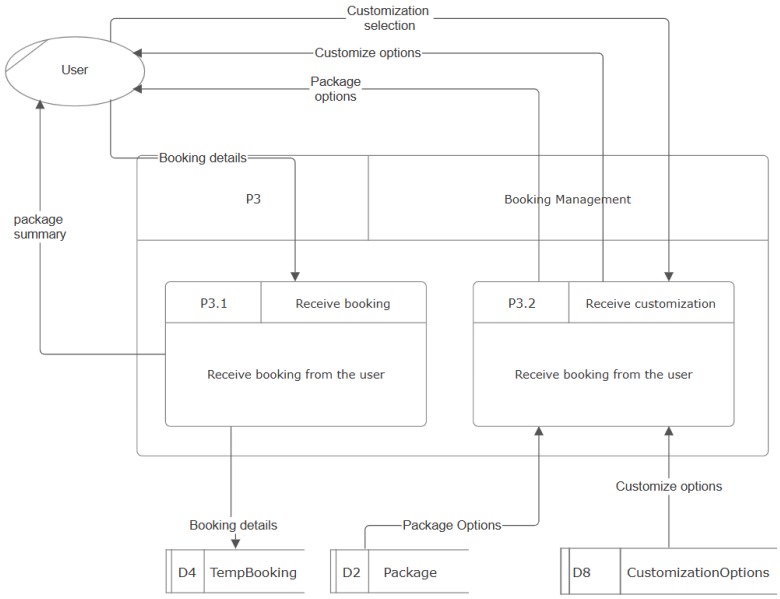


Figure 26-DFD L2 Process 3

Payment processing

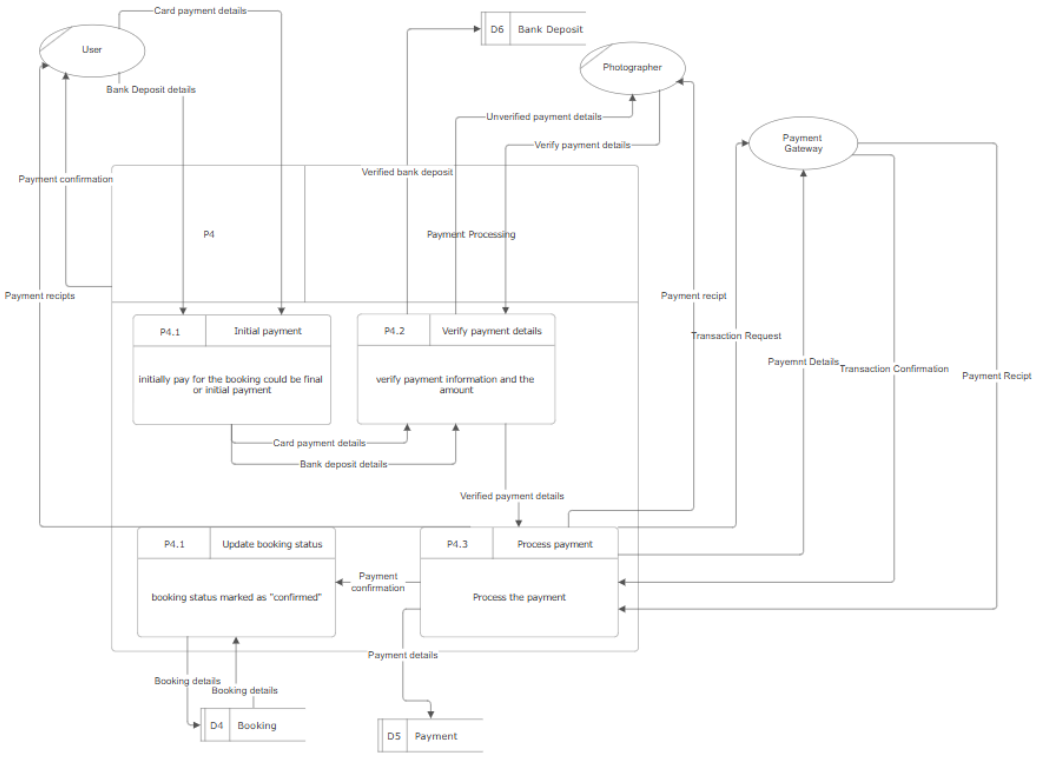


Figure 27-DFD L2 Process 4

Progress update and album delivery

Figure 28-DFD L2 Process 5

Report generation

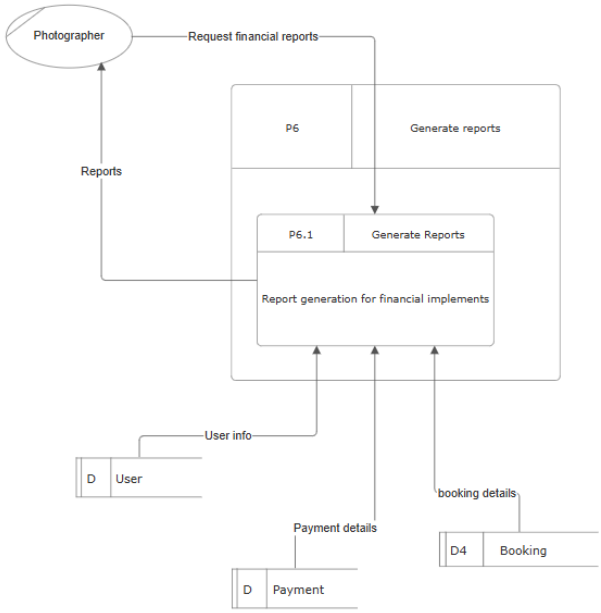


Figure 29-DFD L2 Process 6

Update gallery and packages

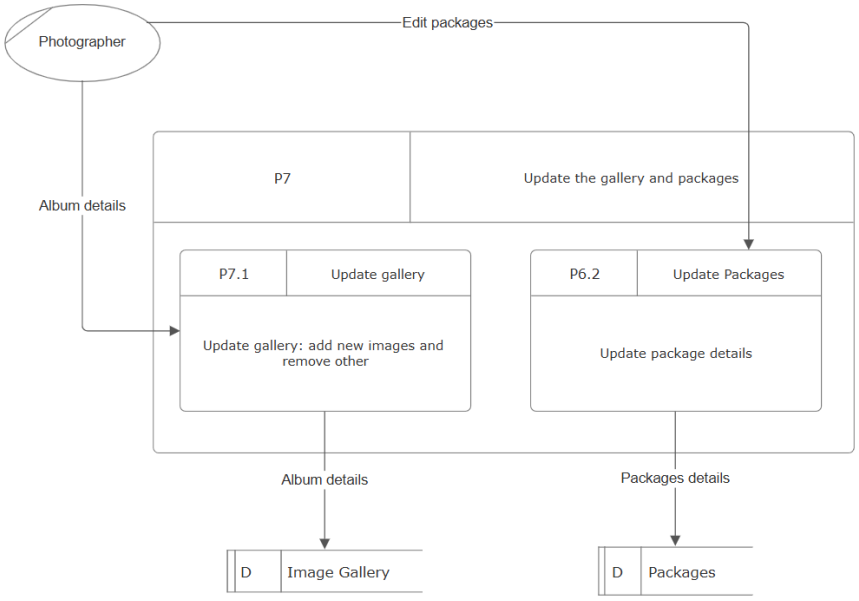


Figure 30-DFD L2 Process 7

3.7 Entity Matrix for Proposed System

	User	Payment	Booking	Package	Bank Deposit	Custamization Option	Photo Album	Image Gallery
User		X	X					
Booking						X	X	
Payment					X			
Package								X
Bank Deposit								
Custamization Option								
Photo Album								
Image Gallery								

Figure 31-Entity Matrix

3.8 Logical Data Structure

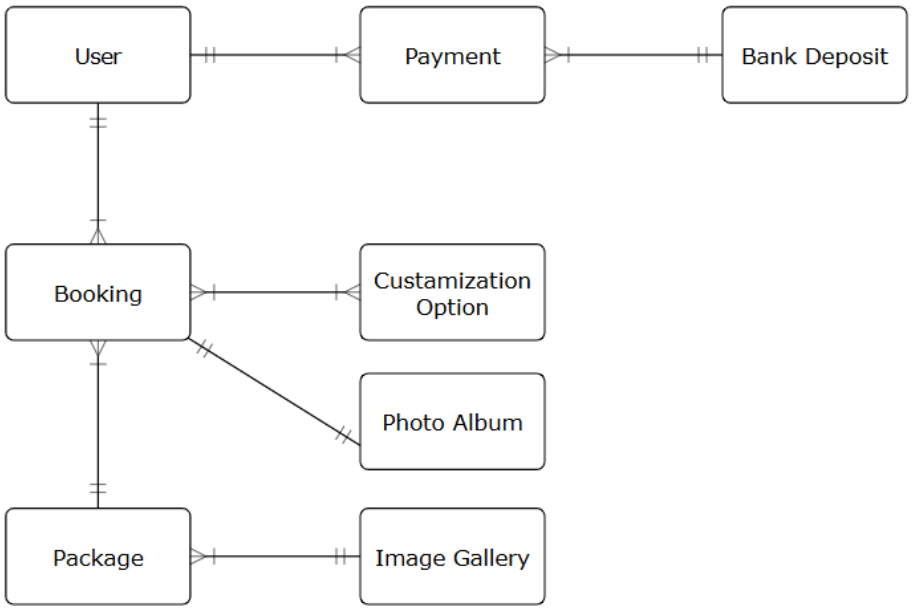


Figure 32-LDS

3.9 ER Diagram

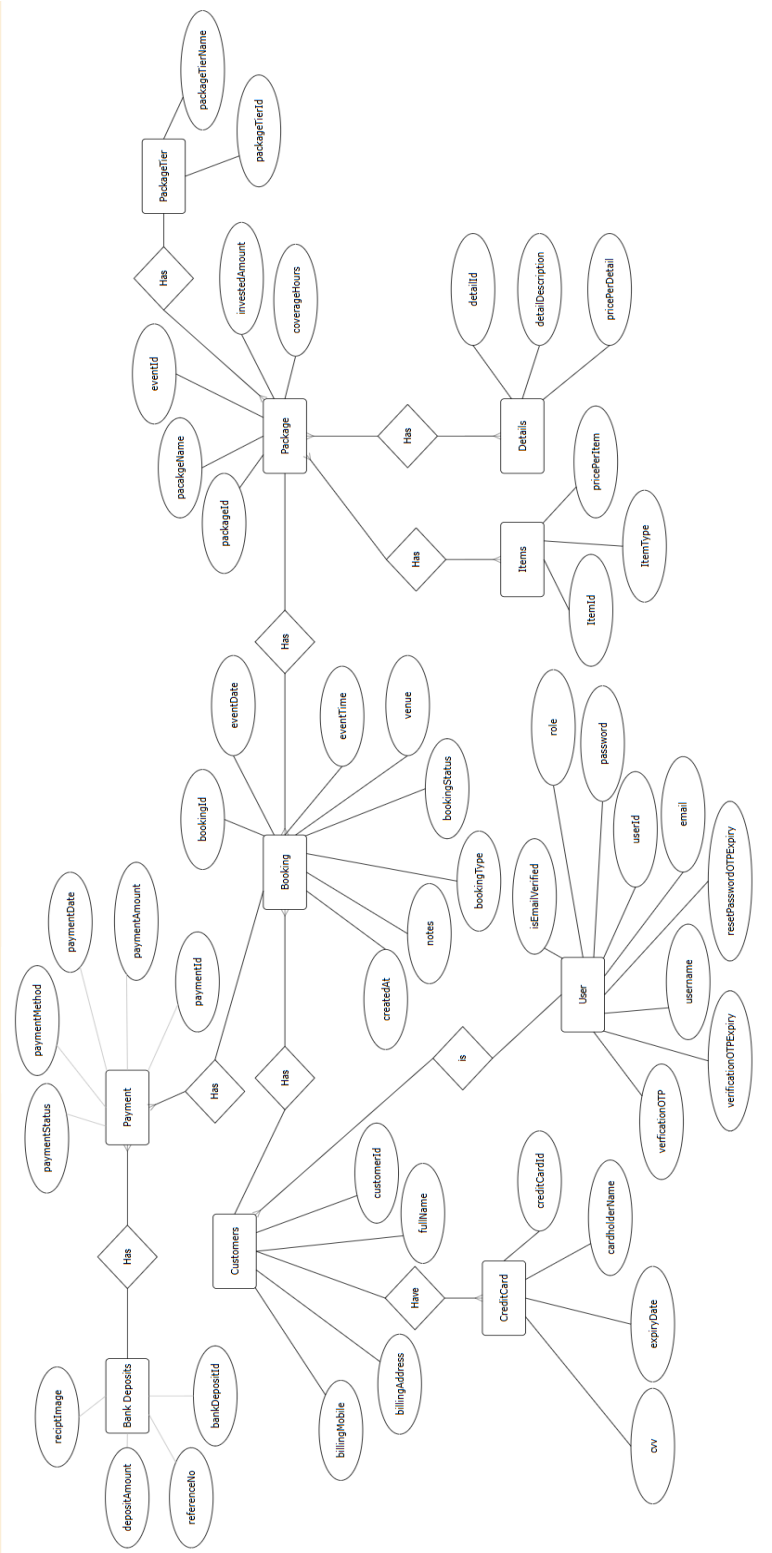


Figure 33-ER diagram

3.10 Normalized ER Diagram

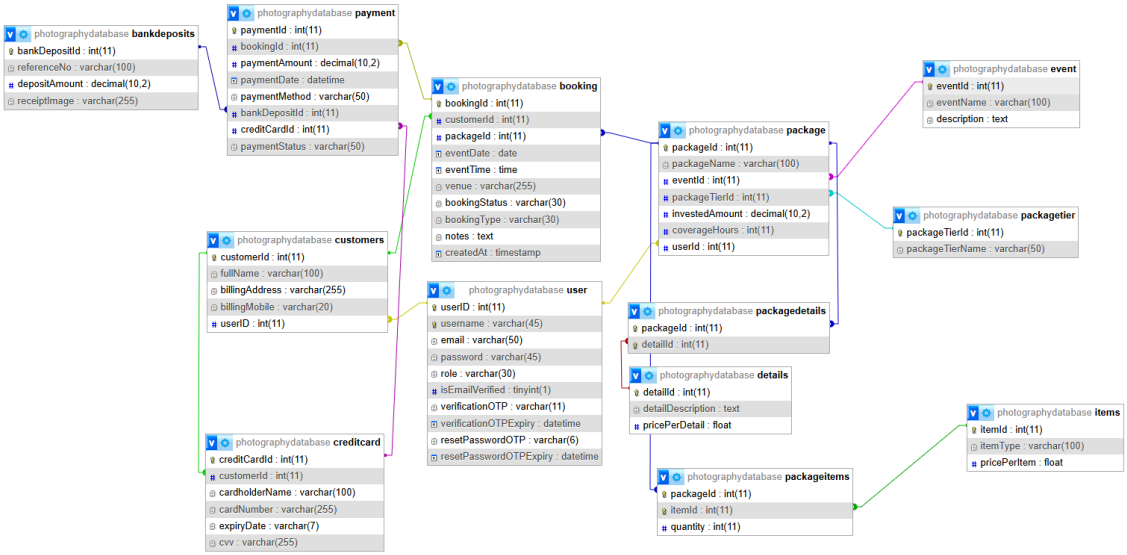


Figure 34-Normalized ER Diagram

3.11 Sample User Interfaces for the Proposed System

These are some of the UIs that designed for the proposed system.

Figma Link: https://www.figma.com/design/H9OcjaZzV9ZnOjiwWb2P4N/Photography-Managment-System?node-id=0-1&t=lgEajk9XejMUr11o-1

Welcome Page

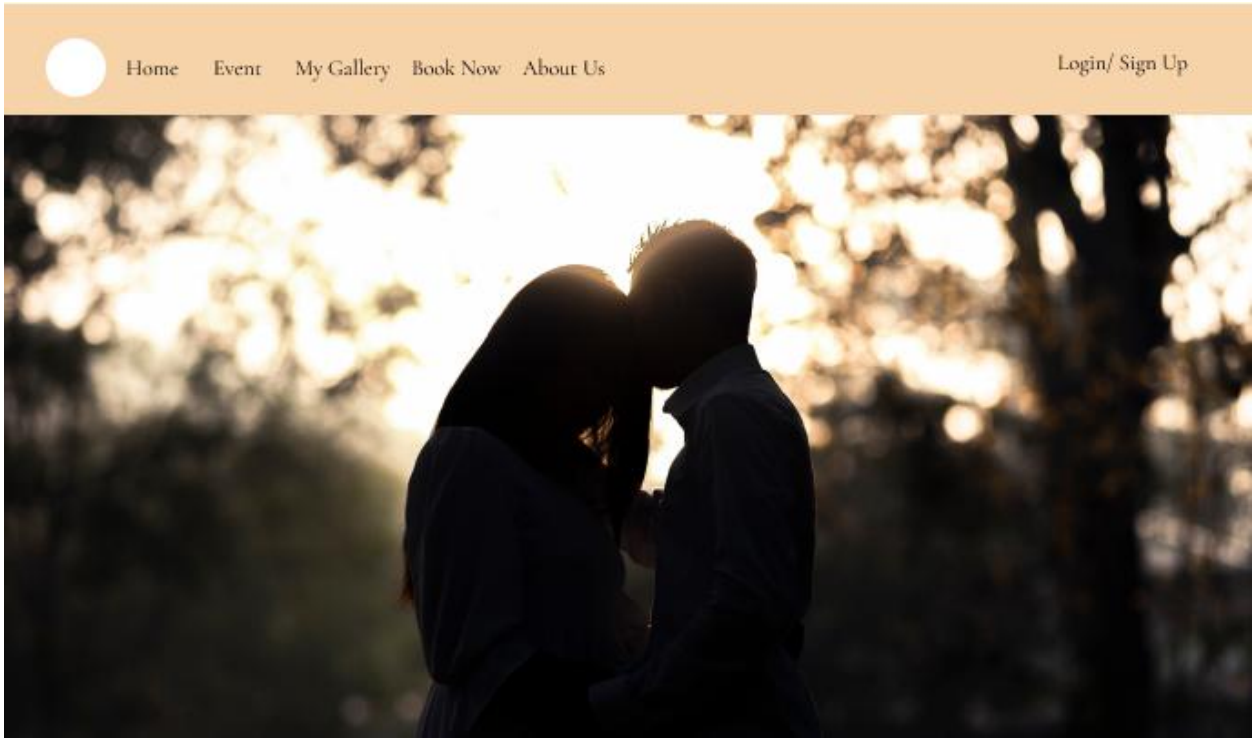


Figure 35-UI Welcome Page

Gallery

Figure 36-UI Gallery

Booking

Figure 37-UI Booking

Payment Invoice

Figure 38-UI Payment Invoice

Album Tracking

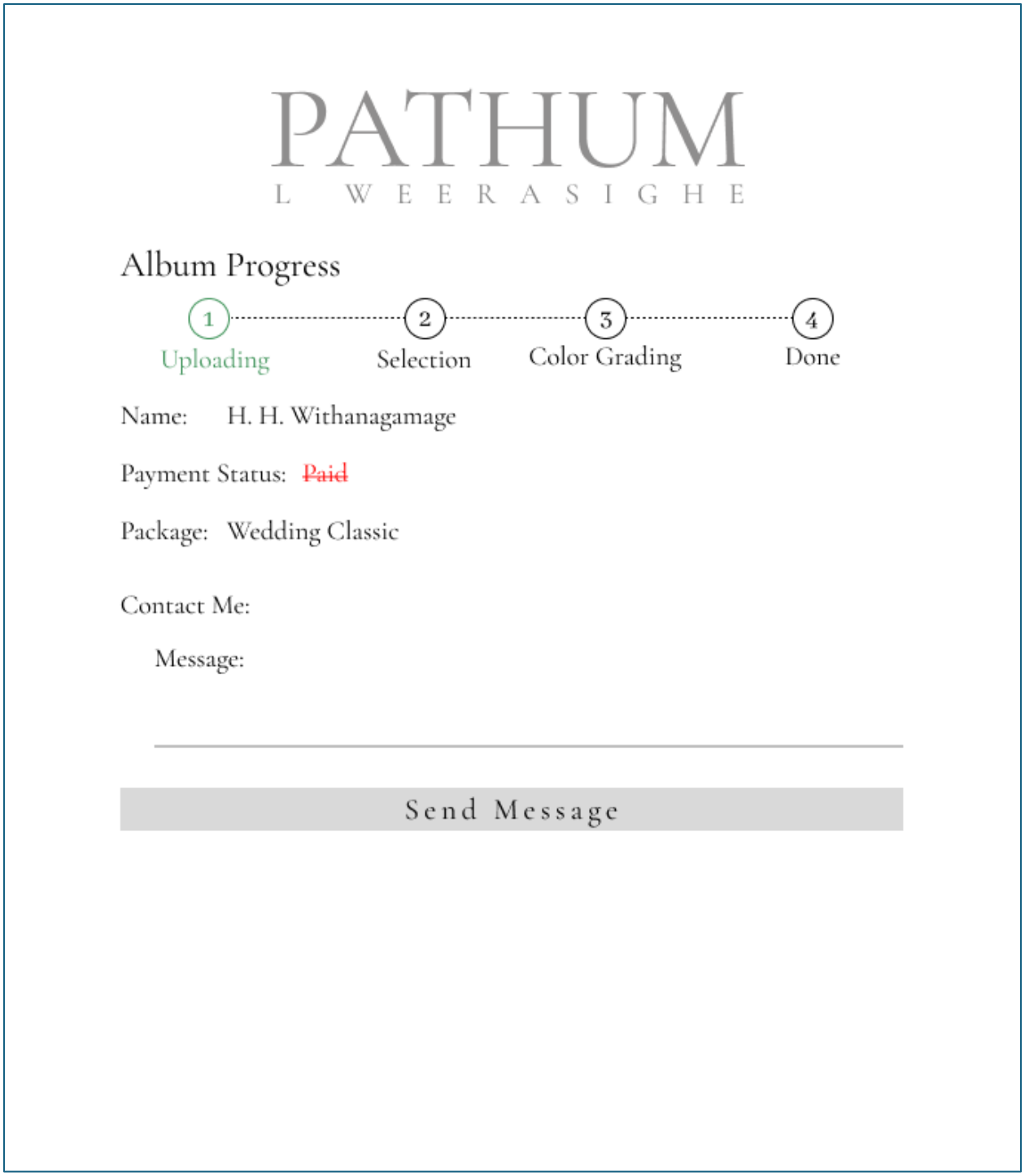


Figure 39-UI Album Tracking

Invoice

Figure 40-UI Invoice

Chapter 4 – System Development

Outline of the Chapter

4.1 Elementary Process Descriptions using pseudo code

This section outlines the elementary processes that define the core functionalities of the Photography Management System. By employing pseudo code for each process, we provide a clear, language-agnostic representation of system logic and workflow sequences. These elementary process descriptions break down complex operations—such as user registration, appointment booking, payment processing, and gallery management—into structured logic steps. Presenting these tasks in pseudo code helps bridge the gap between technical requirements and practical implementation, making the system's design more transparent and easier to communicate to both technical and non-technical stakeholders.

4.1.1 User Registration and Authentication

```
Process: User Registration
BEGIN
    Prompt user for registration details (name, email, password,
etc.)
    IF email is not already registered THEN
        Save user details to database
        Send confirmation email
        RETURN "Registration successful"
    ELSE
        RETURN "Email already exists"
    END IF
END

Process: User Login
BEGIN
    Prompt user for email and password
    IF credentials are valid THEN
        Initiate user session
        Redirect to dashboard
    ELSE
        RETURN "Invalid Login"
    END IF
END
```

Figure 41 - Pseudo code User Reg

4.1.2 Gallery Viewing and Package Selection

```
Process: View Gallery and Packages
BEGIN
    Retrieve image albums from Google Drive (via backend API)
    Display thumbnails and categories
    Allow user to filter/view packages
    IF user selects a package THEN
        Show package details
    END IF
END
```

Figure 42 - Pseudo Code Gallery

4.1.3 Appointment Booking

```
Process: Book Appointment
BEGIN
    User selects preferred date and time
    Check available slots in the system
    IF slot available THEN
        Prompt for booking details and initial payment
        Save booking to database
        Update calendar and notify photographer
        RETURN "Booking Confirmed"
    ELSE
        RETURN "Slot Unavailable"
    END IF
END
```

Figure 43 - Pseudo Code for Appointment

4.1.4 Progress Update and Album Delivery

```
Process: Book Appointment
BEGIN
    User selects preferred date and time
    Check available slots in the system
    IF slot available THEN
        Prompt for booking details and initial payment
        Save booking to database
        Update calendar and notify photographer
        RETURN "Booking Confirmed"
    ELSE
        RETURN "Slot Unavailable"
    END IF
END
```

Figure 44 - Pseudo Code For Progress Update

4.1.5 Notification Handling

```
Process: Send Notifications
BEGIN
    ON booking, payment, or status updates
        Use Nodemailer to send email to relevant users (user or
photographer)
    Notifications sent for:
        - Booking placed, confirmed, or completed
        - Payment reminders and confirmation
        - Album upload and ready-for-selection alerts
END
```

Figure 45 - Pseudo Code for Notification

4.1.6 Reporting & Admin Dashboard

```
Process: Send Notifications
BEGIN
    ON booking, payment, or status updates
        Use Nodemailer to send email to relevant users (user or
photographer)
    Notifications sent for:
        - Booking placed, confirmed, or completed
        - Payment reminders and confirmation
        - Album upload and ready-for-selection alerts
END
```

Figure 46 - Pseudo Code for Reporting

4.1.7 Manage Gallery and Packages (Admin)

```
Process: Update Gallery or Packages
BEGIN
    Admin adds/updates packages and images through dashboard
    System updates information in real time
    Gallery contents are updated from Google Drive folders
END
```

Figure 47 - Pseudo Code for Manage Gallery

Above are Elementary Process Descriptions using high-level pseudo code. Each process represents a fundamental user or admin operation and maps to a functional requirement in the system.

4.2 Programming Languages and Development Tools

4.2.1 MVC Architecture

The **MVC (Model-View-Controller) architecture** is a widely used design pattern that separates an application into three interconnected components:

Model: Handles data and business logic, including database queries and data management.

View: Manages the user interface and presentation of data to the user.

Controller: Acts as an intermediary, processing user input, calling the appropriate model functions, and determining which view to display.

In this photography management system, the MVC architecture organizes the codebase for clarity and maintainability. **Node.js** is used for the backend, where routes direct requests to controllers, controllers process logic and interact with models, and models handle SQL queries and data operations. This separation of concerns allows for easier development, testing, and scalability.

4.2.2 React js

React.js is used to build the frontend, which is responsible for displaying the user interface and providing a smooth and interactive experience for users. Each main page of the application is created as a separate component with its own JSX file, and the styling for each page is managed with individual CSS files that are imported into these components. The App.js file acts as the central hub, running the overall UI and rendering other components as needed. Navigation between different pages is handled using a navigation function, and important information such as the main frontend URL is stored in a context file, making it easy to manage and update shared data across the application. This structure helps keep the code organized, maintainable, and scalable.

4.2.3 Node js and Express

In this project, Node.js and Express serve as the backend, forming the core of the system by handling all server-side operations. The backend is organized according to the MVC architecture, ensuring a clear separation of concerns and maintainable code structure. The main server file, **server.js**, is responsible for initializing the server and importing all route definitions necessary for the application's functionality.

Each major feature of the application, such as package management or booking management, has its own dedicated route file (e.g., **package_routes** and **booking_routes**). These route files define specific endpoints, each associated with a URL path, an HTTP method (such as GET, POST, or DELETE), and a corresponding controller function.

```
25   // Get all bookings
26   bookingRouter.get('/', getAllBookings);
27
28   bookingRouter.get('/user/:userId', getBookingByUserId);
29
30   // create a booking
31   bookingRouter.post('/create', upload.single('bankReceiptImage'), createBooking);
32
```

Figure 48- booking router

When the frontend makes an API request to a defined URL, the backend processes the request by matching it to the appropriate route. The relevant controller function is then executed, often involving database operations through SQL queries. These queries may be defined directly in the controllers or within separate model files for better organization.

```
// Update the fetch call to not set Content-Type header (browser will set it with boundary)
const response = await fetch(`${url}/api/bookings/create`, {
  method: 'POST',
  headers: {
    'Authorization': `Bearer ${localStorage.getItem('token')}` // Keep only the auth header
  },
  body: formDataToSubmit, // Send the FormData object directly
});
```

Figure 49- Backend URL and response

After processing the request, the backend sends a response back to the frontend, which then displays the results to the user. This architecture ensures efficient communication between the frontend and backend and supports the scalability and maintainability of the overall system.

4.2.4 CSS

Each JSX component is paired with its own dedicated CSS file. These basic CSS files are directly imported into their respective JSX components, providing localized styling. This approach ensures that the visual presentation of each component is well-defined and contributes to a more appealing and engaging user experience.

4.2.5 MySQL

The database for this project is managed using MySQL, with XAMPP serving as the local server environment. XAMPP provides an easy-to-use interface to start the MySQL server and manage the database through phpMyAdmin, where tables are created and maintained. Visual Studio Code is connected to the MySQL database via a configured database connection, which is utilized whenever SQL queries need to be

executed. The primary role of MySQL in this system is to provide reliable and structured data storage for all project-related information.

	Table	Action	Rows	Type	Collation	Size	Overhead
☐	bankdeposits	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	32.0 KiB	-
☐	booking	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	48.0 KiB	-
☐	creditcard	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	32.0 KiB	-
☐	customers	Browse Structure Search Insert Empty Drop	2	InnoDB	utf8mb4_general_ci	32.0 KiB	-
☐	details	Browse Structure Search Insert Empty Drop	13	InnoDB	utf8mb4_general_ci	32.0 KiB	-
☐	event	Browse Structure Search Insert Empty Drop	6	InnoDB	utf8mb4_general_ci	16.0 KiB	-
☐	items	Browse Structure Search Insert Empty Drop	9	InnoDB	utf8mb4_general_ci	32.0 KiB	-
☐	package	Browse Structure Search Insert Empty Drop	5	InnoDB	utf8mb4_general_ci	64.0 KiB	-
☐	packagedetails	Browse Structure Search Insert Empty Drop	6	InnoDB	utf8mb4_general_ci	32.0 KiB	-
☐	packageitems	Browse Structure Search Insert Empty Drop	7	InnoDB	utf8mb4_general_ci	32.0 KiB	-
☐	packagetier	Browse Structure Search Insert Empty Drop	7	InnoDB	utf8mb4_general_ci	16.0 KiB	-
☐	payment	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	80.0 KiB	-
☐	user	Browse Structure Search insert Empty Drop	3	InnoDB	utf8mb4_general_ci	32.0 KiB	-
	13 tables	Sum	58	InnoDB	utf8mb4_general_ci	480.0 KiB	0 B

Figure 50- Database Relations

4.2.6 Figma

Before the frontend development began, the UI was designed using the Figma platform. Figma was used to create initial layouts and prototypes for the primary user interfaces, ensuring that the designs aligned with client requirements. The main purpose of using Figma was to develop clear and effective UI designs that would guide the subsequent development process.

4.2.7 Visual Studio Code

Visual Studio Code (VS Code) was the main IDE used for this project, enhanced with several extensions to improve development. Extensions like Prettier ensured consistent code quality, and GitLens facilitated version control. These tools streamlined the workflow and helped maintain high coding standards.

4.2.8 GitHub

The primary version control tool used in this project was GitHub. Daily commits were made to update the repository with the latest work, ensuring that all changes were tracked. This allowed us to easily share code and access previous versions whenever needed, facilitating effective collaboration and project management.

4.2.9 Thunder Client

Thunder Client is a lightweight REST API client extension integrated directly within Visual Studio Code, used in this project to test backend API responses efficiently. It allows developers to send HTTP requests (GET, POST, PUT, DELETE, etc.) and view responses without leaving the VS Code environment,

eliminating the need for external tools like Postman. Thunder Client features a user-friendly graphical interface for configuring request parameters, headers, and bodies, and supports organizing requests into collections and managing environment variables. It also provides scripless testing capabilities, enabling automatic validation of API responses based on criteria such as status codes and response times. Additionally, Thunder Client supports Git integration for version control and CI/CD workflows, making it a powerful yet lightweight tool for seamless API development and testing within the IDE. This integration streamlines the development process by reducing context switching and improving productivity.

4.2.10 Oracle Cloud

Oracle Cloud Virtual Machines (VMs) were provided for developing the system to ensure project reliability and fail-safe operation. These VMs offer a secure, scalable, and flexible virtualized environment that mimics physical servers, allowing the development environment to be isolated and easily managed. Using Oracle Cloud VMs enables seamless backup and restoration, reducing downtime and protecting hardware during development. Additionally, the ability to configure VM resources such as CPU, memory, and storage to match application needs ensures optimal performance. This virtualized setup supports secure and efficient development workflows, safeguarding the project against failures and facilitating smooth deployment and testing processes.

4.3 Database Structure and Relationships

The database for this system was designed and implemented using MySQL Server, with XAMPP serving as the local development environment. Tables were created using standard SQL queries, ensuring proper data structure and maintaining referential integrity using primary and foreign keys. Intermediate tables, such as those linking packages to their details and items, were also included to accurately represent many-to-many relationships within the system.

The provided ER diagram visually represents the relationships between the main entities, such as users, customers, bookings, payments, packages, and events. It clearly illustrates how tables are interconnected, ensuring data consistency and supporting the core functionalities of the photography management system. This relational structure enables efficient data retrieval and management, which is essential for the smooth operation of the application.

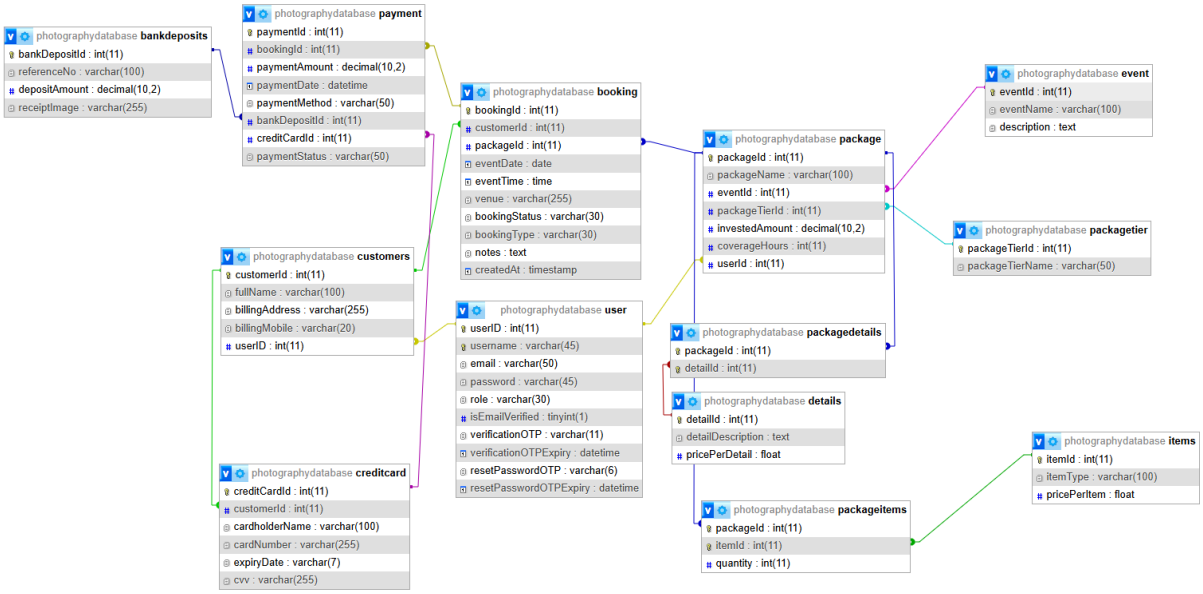


Figure 51 - Database Schema

4.4 User Interface Demonstration and Data Entry Screens

4.4.1 Home Page

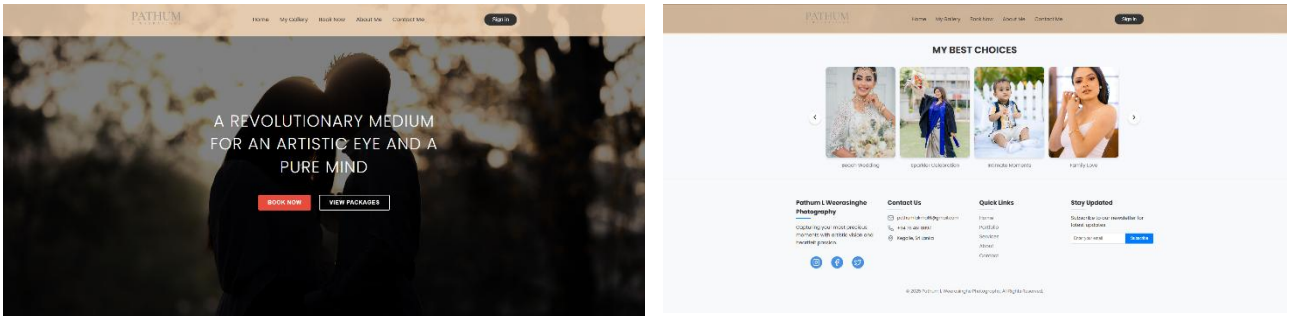
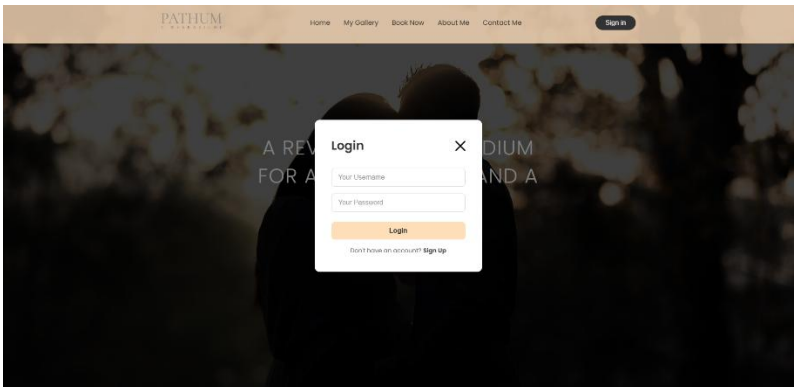


Figure 52 - Home Page UI

Home page or the landing page is shown in the above images. Once the user is coming to the system the first page that the user will see is the home page. There is a navigation bar and a footer with a portfolio in the middle of the page. If the user is not logged in the user cannot book or view any packages. Once the user logged in the user can book or view any packages in the system.

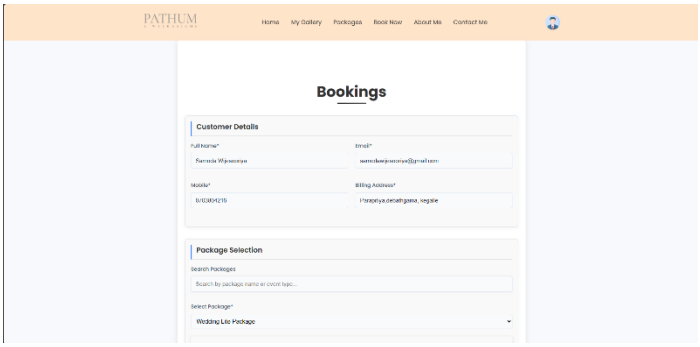
4.4.2 Login and Sign in



After a user clicks on the Sign in button the login popup message will displayed and If the user does have an account then the user have to register to this system by clicking Sign in link. If the user forgot the password then the user can rest the password by clicking the forget password.

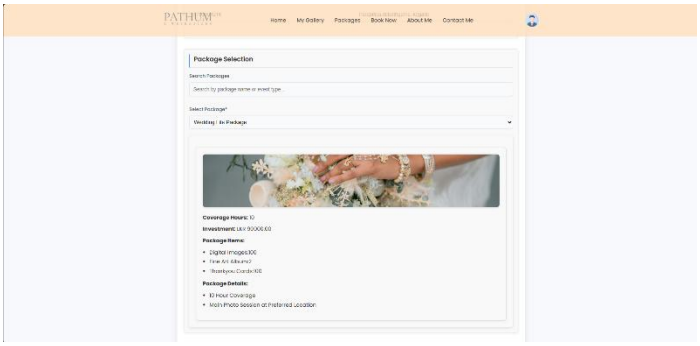
Figure 53 - login Page UI

4.4.3 Bookings



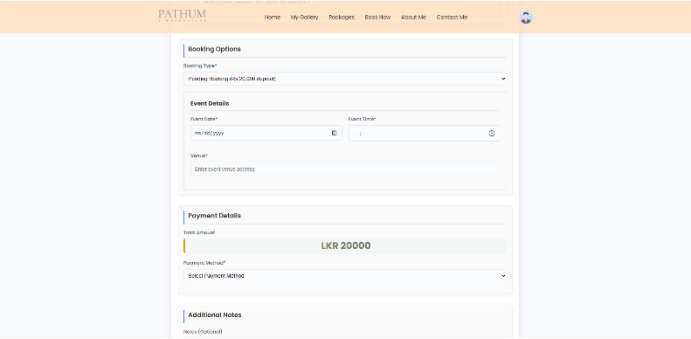
The main booking UI is given here the main customer details are automatically filled from the database by getting the current user logging details.

Figure 54- Booking Page UI



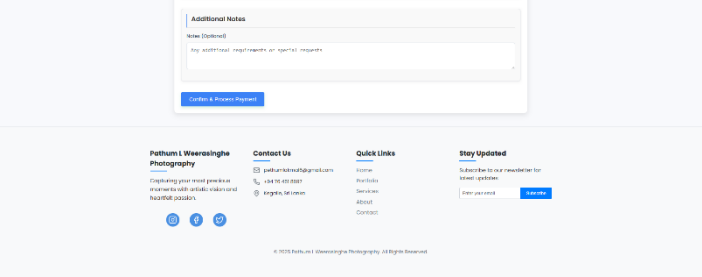
The package should be selected from the user by himself from a dropdown menu. Then the package details are automatically filled.

Figure 55 - Booking Page UI 02



The booking options are available as the pending or confirm bookings.

- Pending bookings – LKR 20,000
- Confirmed Bookings – Total Package Amount



Any additional details can be added to the booking and users can submit a booking.

Figure 56 - Booking Page UI 03

4.4.4 Packages

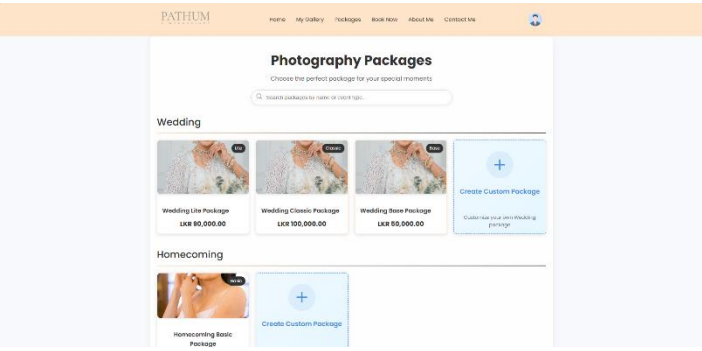


Figure 57 - Package Page UI

Users can view available packages on the package page and customize them according to their preferences. Starting from a basic package, users have the option to add or remove items as needed. Additionally, users are allowed to create entirely new packages, providing flexibility to tailor services to their specific requirements.

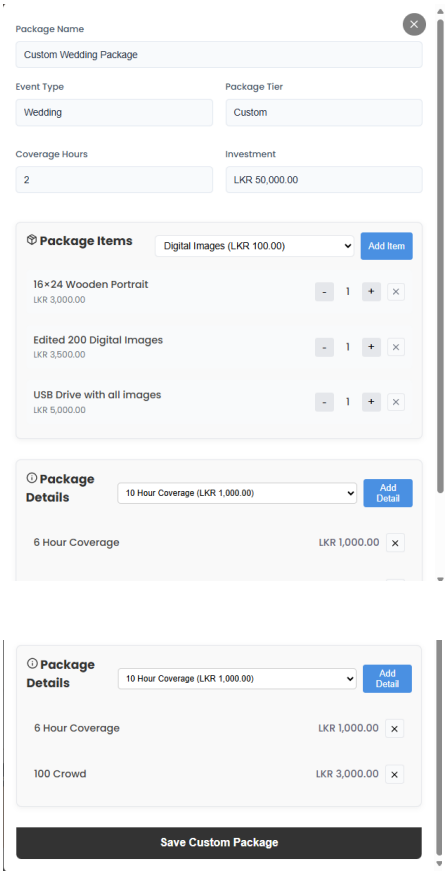


Figure 58 - Customize Package

This image shows the customizing package that give the access to the user to customize and add any options that the photographer offers.

4.4.5 Your Booking

In here user can view the bookings and the status of the booking. And there is a option for canceling the booking from here.

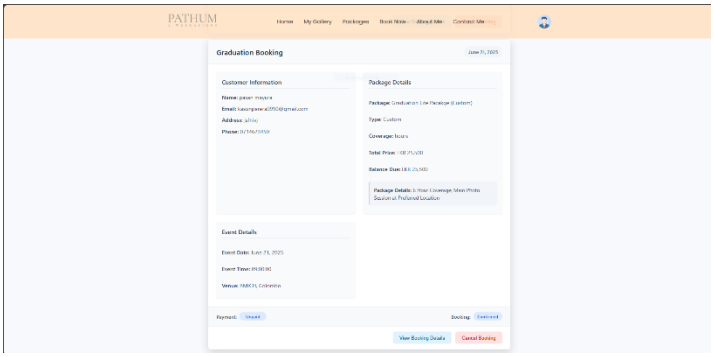
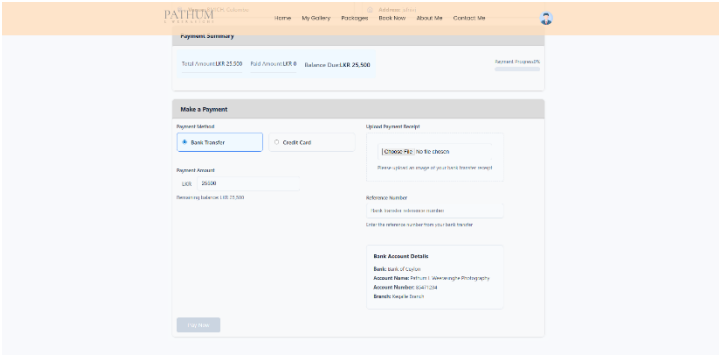


Figure 59 - Your Booking Page



After clicking the view booking details the user can view all the details of the booking. If the user have to pay the remaining amount of the payment, then there is a option to pay the remaining amount. After completing the event user can see the

progress bar and the status of the booking in the booking details tab.

Figure 60- Your Booking Details Page

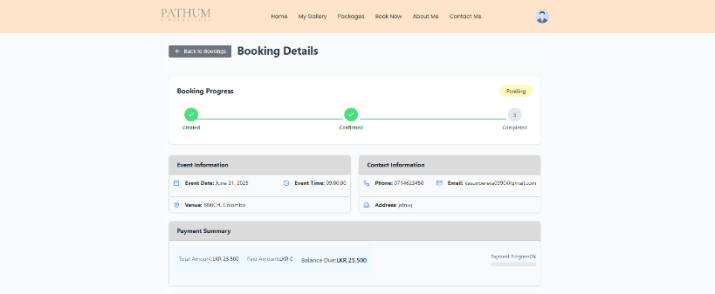
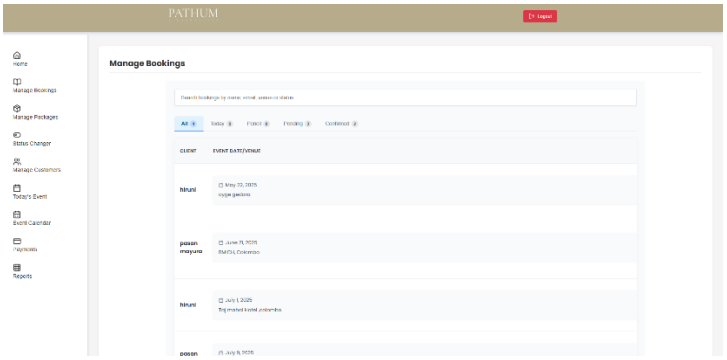


Figure 61 - Your Booking Details 02

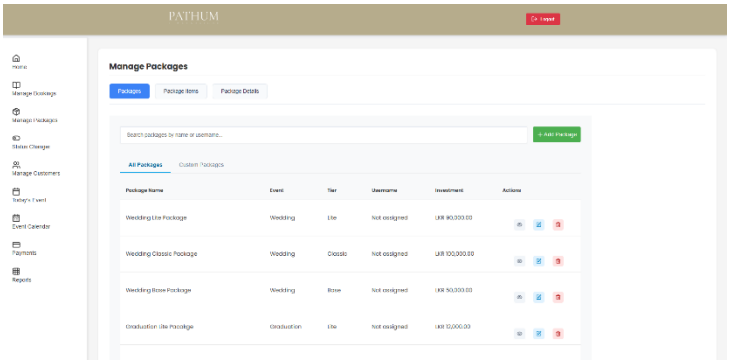
4.4.6 Manage Booking



In here the photographer can view the bookings from the admin view. All bookings can be viewed and also the photographer can view all the categories like Today's bookings, Pending bookings, Confirmed bookings with the count of the bookings.

Figure 62 - Manage Booking Page

4.4.7 Manage Packages



The manage packages also helps to the photographer to see the all packages that is in the system. And all the packages that have been created by the users as customized packages.

Event Calendar

The event calendar is where the photographer can view an overview of the events that is happening in the future and the events in the past.

Figure 63 - Manage Packages Page

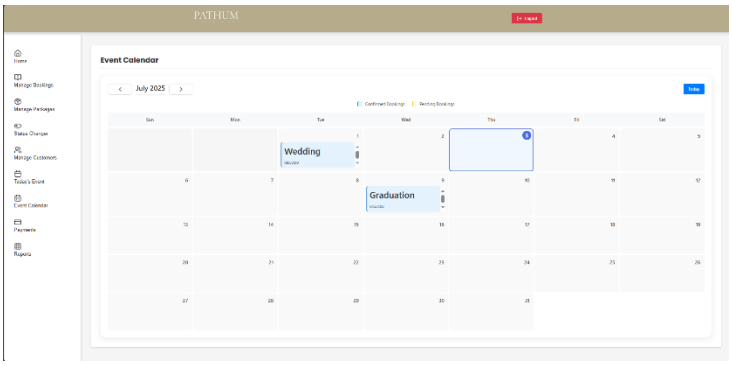


Figure 64 - Calendar

Those UIs are some of the data capturing events and main UIs that is focused for the customers and photographers.

4.5 Special Implementations

4.5.1 Image uploads

In this project, image albums in the gallery are managed by integrating Google Drive with the system. Photographers can create albums simply by creating folders in a Google Drive account linked to an admin email. Each folder, named after the album, contains the relevant images. The system fetches and displays these images as albums for users, allowing photographers to easily showcase their recent work while customers can conveniently browse the albums.

```
const auth = new google.auth.GoogleAuth({
    keyFile: process.env.GOOGLE_APPLICATION_CREDENTIALS,
    scopes: ['https://www.googleapis.com/auth/drive.readonly'],
});

const drive = google.drive({
    version: 'v3',
    auth,
});
```

Figure 65 - Google Drive Implementation

This integration is achieved by using Google Drive API with an API key from Google Cloud. The backend accesses Google Drive to retrieve folder contents dynamically, simplifying album management without requiring complex uploads through the application itself.

In the frontend React application, albums can be displayed by fetching the folder contents via backend API endpoints that wrap the above logic. The UI can render album thumbnails linking to the images stored on Google Drive, providing a seamless and user-friendly gallery experience.

This approach allows photographers to manage albums simply by creating and organizing folders in Google Drive, while the system dynamically reflects these changes to users

```
driveRouter.get('/', async (req, res) => {

    try {
        // Query for folders in the specified parent folder
        const rootFolderId = process.env.ROOT_FOLDER_ID;

        const response = await drive.files.list({
        q: `'${rootFolderId}' in parents and mimeType='application/vnd.google-apps.folder' and trashed=false`,
        fields: 'files(id, name, description)',
        });

        res.json(response.data.files);
    } catch (error) {
        console.error('Error fetching folders:', error);
        res.status(500).json({ error: 'Failed to fetch albums' });
    }
});
```

Figure 66 - Google Drive Routes

without additional manual uploads or configuration.

4.5.2 Notification systems

The notification system in this project is implemented using email communication to keep customers and photographers informed about important events. The project utilizes **Nodemailer**, a popular Node.js module for sending emails, along with predefined email templates to ensure consistent and professional messaging. Notifications are automatically sent for key actions such as when a booking is placed, cancelled, completed, or finalized. The backend is responsible for handling the email process: it retrieves the recipient's email address from the database and then uses Nodemailer to send the appropriate notification based on the event. This automated email system enhances user engagement and ensures timely communication throughout the booking lifecycle.

```
// Create email transporter
const transporter = nodemailer.createTransport({
  service: process.env.EMAIL_SERVICE,
  auth: {
    user: process.env.EMAIL_USER,
    pass: process.env.EMAIL_PASSWORD,
  }
});
```

This is the image of the transporter that make the initial template and make the process to send the email.

Figure 67 - Nodemailer

4.5.3 Calendar integrations

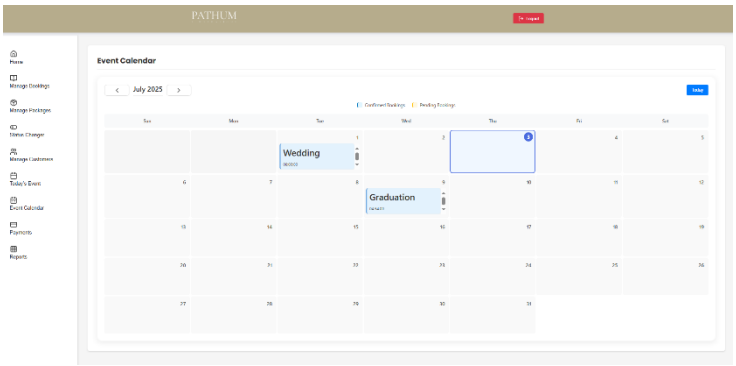


Figure 69 - Calender integration

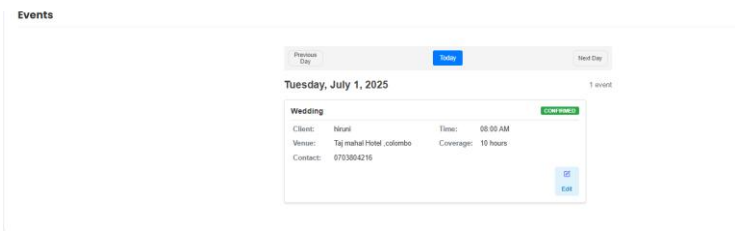


Figure 68 - Today Event MGT

The calendar feature is a key component of the system, designed to help photographers manage their events efficiently and avoid missing any bookings. Each day, the system automatically sends a summary email to the photographer in the morning, reminding them of the events scheduled for that day. This proactive notification ensures the photographer stays informed and prepared. After completing an event, the photographer updates the event status in the system to "done," which in turn updates the booking status visible to the

customer as "completed." This workflow streamlines event management, improves communication, and enhances customer satisfaction by providing real-time updates on booking statuses.

4.5.4 Real-time status updates

The system includes a real-time status update feature that allows photographers to update the booking status immediately after completing an event. Once the photographer marks the booking as completed in the system, customers can view the status of their bookings in the "Your Booking" section. Additionally, detailed progress of each booking is visually represented using a progress bar, providing customers with clear and up-to-date information on the booking's advancement. This real-time update mechanism enhances transparency and improves user experience by keeping both photographers and customers informed throughout the booking life cycle.

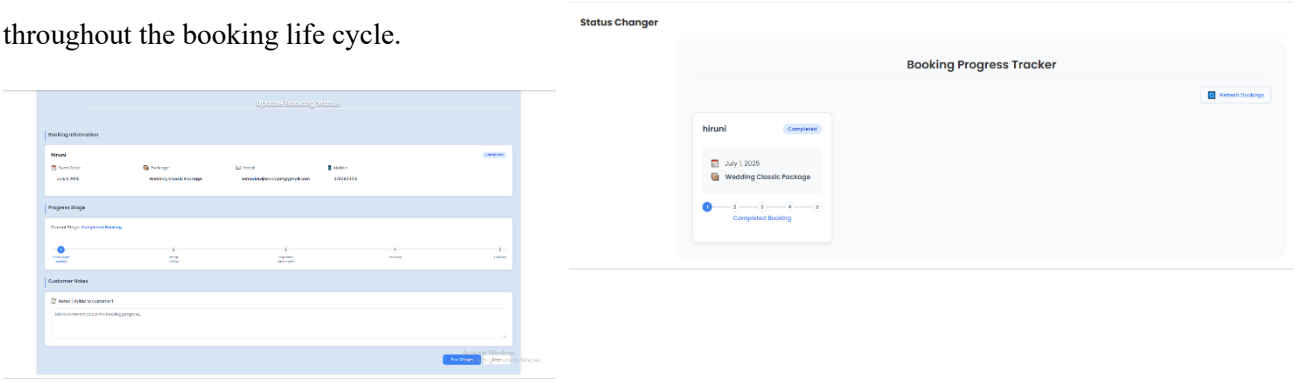


Figure 70 - Status Changer

The system features key implementations such as Google Drive integration for easy album management, automated email notifications using Nodemailer, a calendar with daily event reminders for photographers, and real-time booking status updates with progress tracking. These enhancements improve usability, communication, and overall management efficiency.

4.6 Reports and Dashboard

4.6.1 Reporting Process of the System

The purpose of the reporting process within the photography management system is to provide key analytical insights and monitor overall system performance. Reports are generated dynamically based on user requests, enabling tailored views of relevant data. Specifically, within the photographer's dashboard, a dedicated reporting section offers comprehensive visibility into critical business metrics such as sales performance, current booking rates, payment summaries, and other financial indicators. This empowers photographers to track their individual productivity, revenue generation, and client engagement efficiently.

Here are some of the reports in the photographer's dashboard,

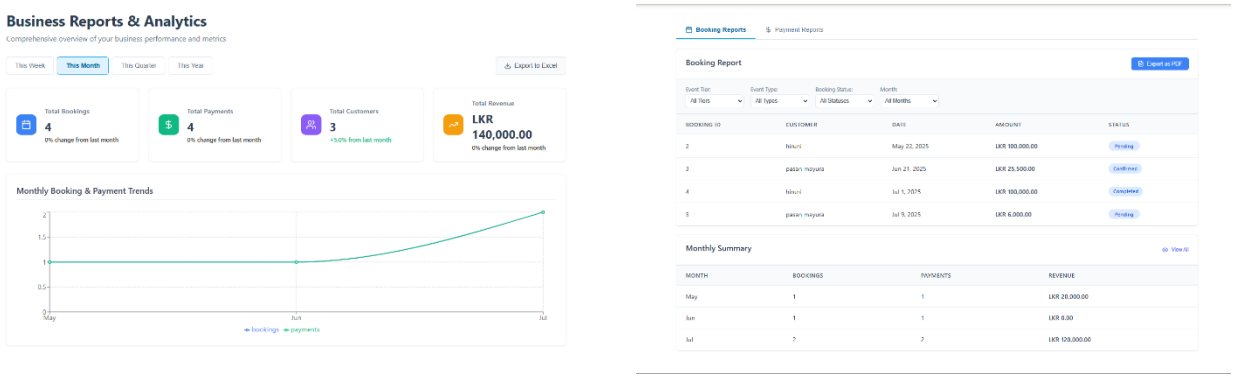


Figure 71 - Reports

4.6.2 Invoice Generation Module

The invoices are also generated when a booking is placed. The user can download the invoice and after booking is submitted to the system. And a copy of the invoice will be sent to the user's email. There is a screenshot of the invoice that was sent to the user.

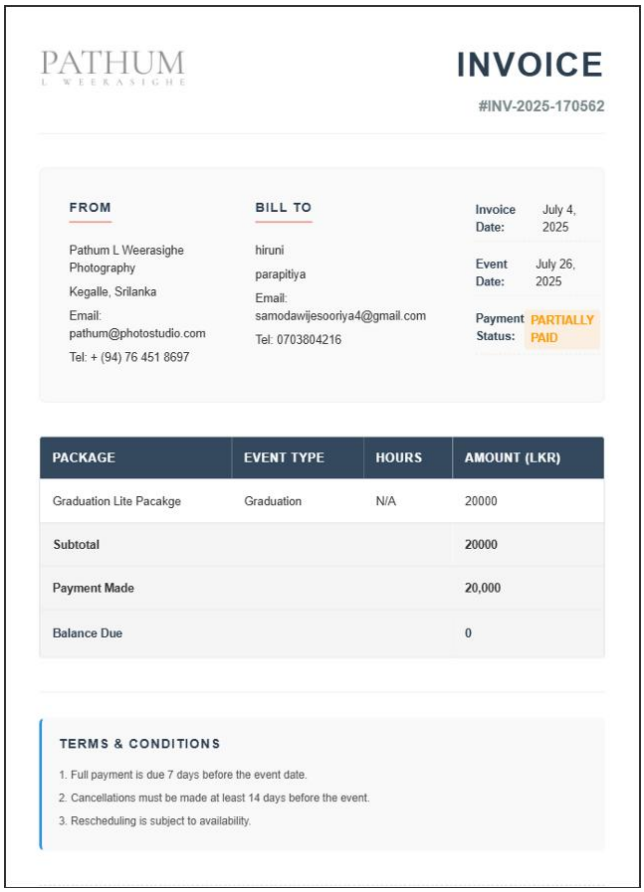


Figure 72 - Booking Invoice

4.6.3 Admin Dashboard and the Photographer Dashboard

During the design phase of the photography management system, there were initially two primary user roles. However, during development, it became necessary to introduce an admin role to effectively manage both photographers and customers. Consequently, the system now supports three distinct actors.

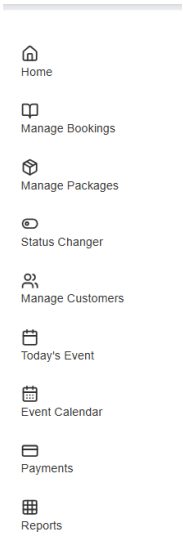


Figure 73 - Admin Dashboard Menu

The admin role is primarily responsible for managing system users, including photographers and customers. This includes administrative tasks such as managing user credentials, specifically passwords and email addresses for photographers. While the admin oversees user management and system settings, the photographers act as the primary operational users, having access to the full range of system functionalities.

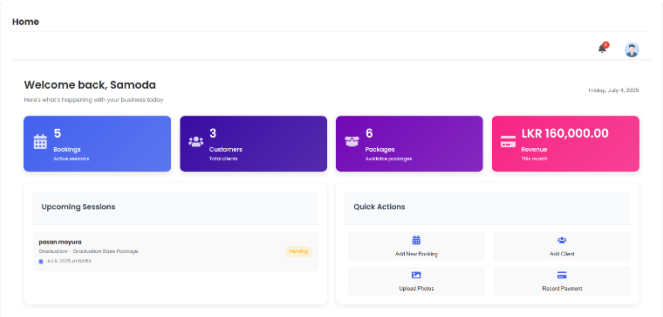


Figure 74 - Admin Dashboard Summary

Photographers, in their role, can view and manage all aspects related to bookings, packages, reports, event calendars, and status updates. This separation of roles ensures that administrative control is maintained for security and user management, while photographers have comprehensive access to manage their workflow and client interactions within the system.

4.7 Testing and Validations

During the development of the photography management system, unit testing was performed continuously to ensure that individual components and modules functioned correctly and met the specified requirements. This approach allowed for early detection and resolution of defects, improving the overall quality and reliability of the system.

In addition to unit testing, validation mechanisms were incorporated throughout the system to enforce data integrity and prevent errors during user input. These validations include checks for required fields, correct data formats (such as email addresses and dates), and logical constraints (such as booking availability and payment status). Implementing these validations helped to enhance user experience by providing immediate feedback and reducing the likelihood of invalid or inconsistent data entering the system.

Together, the combination of rigorous unit testing and comprehensive input validations contributed to building a robust and dependable photography management system that meets both functional and usability standards

Testing document conducted:

Excel Unit Testing Document Structure								
Test case ID	**Module/Feature**	**Test Description**	**Test Steps**	**Input Data**	**Expected Results**	**Actual Result**	**Status (Pass/ Fail)**	**Remarks**
TC-001	User Registration	Verify user can register with valid data	Go to registration, fill valid details, Submit	Name, Email, Username, Password	User account created, redirected to login	User registered successfully and redirected to	pass	-
TC-002	Login	Verify login with correct credentials	Go to login, enter valid username and password and submit	Registered Username, Password	User logged in, dashboard displayed	User logged in and dashboard displayed correctly	pass	-
TC-003	Booking Module	Verify booking creation	login, go to booking, enter booking details, submit	Date, Venue, Package, Payment	Booking created, confirmation message	Booking created; confirmation message	pass	-
TC-004	Invoice Generation	Verify invoice is generated after booking	complete a booking and submit the booking, check the invoice and download	Booking ID	Invoice generated and viewable	Invoice generated and visible in the invoice section	pass	-
TC-005	Reporting	Verify sales report generation	login as a photographer, naviagate to Reports, Generate any report and print them using an Excel	Date Range	Sales report displayed with correct data	Sales report generated with correct totals for selected date range	pass	-
TC-006	Admin Dashboard	Verify admin can reset photographer password	login as admin, go to user management, reset password	Photographer Email	Password reset email sent	Password reset email sent to photographer; login with new password worked	pass	-
TC-007	Validation	Verify email format validation on registration	go ot registration enter an invalid email and submit or try to submit	Invalid Email	Error message: "Invalid email format"	Error message "Invalid email format" displayed	pass	-
TC-008	Booking Module	Prevent booking on unavailable date	try to book on a already booked date	Booked Date	Error: "Date not available"	Error "Date not available" shown; booking not created	pass	-
TC-009	Payment Module	Verify payment summary is updated after payment	make a payemnt check payment summary	payment details	Payment summary updated	Payment summary updated immediately after payment processed	pass	-

Figure 75 - Testing Document

Chaper 5 – Implementation

Outline of the Chapter

5.1 Introduction

5.2 Installation Guide

5.3 User Guide

5.4 Conculsion

5.1 Introduction

This chapter focuses on the implementation of the **Photography Management System**, detailing the fundamental components required for successful deployment in both development and live environments. The guide seeks to assist users whether they are non-technical individuals, photographers, or administrators in understanding and utilizing the system effectively.

5.2 Installation Guide

5.2.1 Localhost (Development/Test Environment)

Requirements:

- XAMPP: For managing local web and database servers (Apache, MySQL)
- Node.js & npm: Backend and package management
- React.js: For the frontend, enabling modular UI development
- MySQL: Relational database management
- Git: Version control, facilitating collaboration and tracking

Step-by-Step Installation:

Setup XAMPP:

- Download and install XAMPP for your operating system.
- Launch XAMPP Control Panel and start Apache and MySQL services.
- Rationale: Provides an isolated, reproducible environment for safe local testing and database management.

Clone Project Repository:

Use Git to clone the project:'

```
git clone <repository-url>
```

Keeps your local copy up-to-date with source control.

Database Initialization:

- Use provided schema/scripts to create the required database and tables.
- Access phpMyAdmin at localhost/phpmyadmin, import SQL files as necessary.
- Ensure credentials in your app match the local database setup.

Backend Setup:

Navigate to the backend folder:

```
cd backend
```

Install dependencies:

```
npm install
```

- Set up environmental variables for:
    - Database connection (.env file)
    - API keys for Google Drive & Email
- Security tips: Never commit sensitive keys to the repository.

Frontend Setup:

Navigate to the frontend folder1:

```
cd frontend
```

Install dependencies and build the app:

```
npm install
npm run build
```

- Ensures React code is transpired and optimized.

Start the Backend Server:

Launch the backend:

```
node server.js
```

Log output should indicate successful database connection and API readiness.

Access Application:

- Open your browser and go to http://localhost or the indicated port.

Testing Tools:

- Figma: For UI prototyping and feedback during iterative design.
- Thunder Client/Postman: Test backend APIs directly, validate endpoints before integrating with frontend.

5.2.2 Hosting on a Live Server (Production Environment)

Requirements:

- Cloud Hosting Provider: (e.g., AWS, Oracle, Digital Ocean)

- Node.js & npm: Server runtime
- Managed MySQL Database: For security, backup, and scalability
- Domain & SSL: Custom domain for branding, SSL for security
- CI/CD Tools: (e.g., GitHub Actions, Jenkins) for streamlined deployments

Production Deployment Steps:

1. **Provision Cloud Server:**
    - Select and launch a VM or web instance with your chosen cloud provider.
    - Ensure OS and instance specs match application needs (CPU, RAM, storage).
2. **Install Required Software:**
    - Install Node.js, npm, and Git.
3. **Database Setup:**
    - Set up a managed (preferably cloud-hosted) MySQL database.
    - Import your local schema and data.
    - Secure database: Use strong root/admin passwords, minimum required permissions for app users.
4. **Code Deployment:**
    - Clone your repository on the server.
    - Configure environment variables securely (never hardcode secrets).
    - In backend and frontend folders, run

```
npm install
npm run build  # Frontend only
```

    -
    - Deploy frontend build artifacts to a web root (can use nginx/Apache for serving).
5. **Server Configuration:**
    - Point your domain to the server's IP address via DNS provider.
    - Set up HTTPS using SSL/TLS (with Certbot, Let's Encrypt, or provider tools).
    - Configure reverse proxy rules to forward user requests to correct backend/frontend services (ex: nginx config).
6. **Continuous Deployment:**
    - Set up CI/CD pipelines to automate builds, tests, and deployments.
    - Benefits: Reduces manual errors, ensures consistent updates, supports rollback on failure.
7. **Security Hardening:**
    - Restrict open ports (minimum 80/443/any custom app ports).

- Enable firewalls and DDoS protection.
- Enforce HTTPS everywhere.
- Regularly update dependencies to patch vulnerabilities.

8. **Post-Deployment Testing:**
    - Validate system end-to-end across browsers/devices.
    - Perform security and stress tests.
    - Monitor logs for errors and performance bottlenecks.

5.3 User Guide

Overview

This guide provides step-by-step instructions for using the **Photography Management System**, allowing users to seamlessly navigate through its features. The system is designed to simplify interactions between clients and photographers, making bookings and payments more efficient.

Accessing the System

Login/ Signup

- Open your web browser and navigate to the system's URL.
- New Users: Click on the Sign Up button to create a new user account. Fill in your details (name, email, password) and submit.
- Existing Users: Click on Login, enter your email and password, and press Submit to access your account.

Main Features

1. **Gallery Browsing:**
    - After logging in, navigate to the Gallery section.
    - Browse various photography portfolios. You can view albums categorized by different themes or events.
    - Expect images to load quickly and display in an organized format.
2. **Booking Appointments:**
    - After logging in, navigate to the Gallery section.
    - Browse various photography portfolios. You can view albums categorized by different themes or events.
    - Expect images to load quickly and display in an organized format.
3. **Payment Processing:**

- After logging in, navigate to the Gallery section.
- Browse various photography portfolios. You can view albums categorized by different themes or events.
- Expect images to load quickly and display in an organized format.

4. **Notifications:**
   - You will receive email notifications for:
     - Booking confirmations
     - Payment reminders
   - Event updates
   - Ensure your contact information is accurate to receive timely alerts.
5. **Album Tracking:**
   - To monitor the progress of your album, visit the Your Booking section.
   - A status update will indicate when your album is being edited and when it's ready for viewing.
   - You can download soft copies of photos directly from this section.

Need Help? If you encounter issues, refer to the Help section or contact customer support directly.

5.4 Summary

The Photography Management System is designed to streamline the workflow between clients and photographers, from portfolio browsing to booking, payment, and album delivery. The installation guide covers both local development (using XAMPP, Node.js, React.js, MySQL, and Git) and production deployment (on cloud servers with CI/CD, managed databases, and enhanced security). The user guide details how to access the system, manage bookings, receive notifications, and track albums, ensuring users can fully leverage all functionalities with ease. This document serves as a one-stop reference for both technical setup and everyday operation.

Chapter 6 – Conclusion

Outline of the Chapter

6.1 Degree of Objectives met

6.2 Usability, accessibility, Reliability and friendliness

6.3 Limitations and Drawbacks

6.4 Future modifications, Improvements and Extensions Possible

6.5 Summary

6.1 Degree of Objectives met

The primary objectives of the Photography Management System were to streamline studio operations, improve client communication, and automate booking and payment processes. Through the development and deployment of this web-based solution, all major objectives have been successfully achieved. The system now provides a centralized platform for managing bookings, payments, and client portfolios, significantly reducing manual work and the risk of errors such as double bookings. Real-time notifications, progress tracking, and a user-friendly interface have enhanced both the photographer's workflow and the client experience. The project has thus met its intended goals, as outlined in the initial requirements.

6.2 Usability, accessibility, Reliability and friendliness

The system was designed with a strong focus on usability and accessibility. The intuitive user interface ensures that both photographers and clients can easily navigate the platform, perform bookings, and access relevant information without requiring technical expertise. Accessibility features, such as responsive design and clear navigation, make the system usable across various devices and for users with different needs. Reliability has been ensured through rigorous testing and validation, resulting in stable performance and minimal downtime. The friendly design and helpful feedback messages contribute to a positive user experience, fostering trust and satisfaction among users.

6.3 Limitations and Drawbacks

Despite its strengths, the system has some limitations. Currently, the platform supports only basic reporting and analytics features; more advanced data visualization and export options could further benefit users. Integration with third-party calendar and payment services is limited, which may restrict flexibility for some clients. Additionally, the system relies on internet connectivity, which could be a drawback in areas with unstable connections. As with any software, ongoing maintenance is required to address potential security vulnerabilities and to keep up with evolving user needs.

6.4 Future modifications, Improvements and Extensions Possible

There are several opportunities for future enhancements. Integrating advanced analytics and reporting tools would provide deeper business insights. Expanding payment gateway options and enabling integration with popular calendar applications could improve convenience for both photographers and clients. Introducing a mobile application would further increase accessibility and engagement. Additional features, such as automated marketing tools, customer feedback modules, and multi-language support, could also be considered to broaden the system's appeal and utility.

6.5 Summary

In summary, the Photography Management System has successfully transformed the traditional manual processes of a photography studio into a modern, efficient, and user-friendly digital solution. While some limitations remain, the system has demonstrably improved operational efficiency, client satisfaction, and business management. With ongoing development and the incorporation of future enhancements, the platform is well-positioned to support the continued growth and success of the photography business.

References

Figma: the collaborative interface design tool. (2024, November 8). Retrieved from Figma: https://www.figma.com/

SmartDraw - Create Flowcharts, Floor Plans, and Other Diagrams. (2024, November 8). Retrieved from SmartDraw: https://www.smartdraw.com/

Appendices

Appendix 1: Github_Link

Appendix 2: Figma_Link