# Internship MEP CONSULTANT Report

**A PROJECT REPORT**

***Submitted by***


**VARASADA SHIVAM R.**

**2203031457022**


***In partial fulfilment for the award of the degree of***

**BACHELOR OF ENGINEERING**

***in***

**COMPUTER ENGINEERING**


**Parul Institute of Engineering and Technology, Limda**




**Parul University, Limda**

**March-2024**

**Parul Institute of Engineering & Technology, Limbda**


# CERTIFICATE

This is to certify that the project report submitted along with the project entitled **Internship MEP CONSULTANT Report** has been carried out by **VARASADA SHIVAM R.** under my guidance in partial fulfilment for the degree of Bachelor of Engineering in **COMPUTER ENGINEERING**, 8th Semester of Parul University, Vadodara during the academic year 2024-2025.

| | |
|---|---|
| Dr. Warish Patel | Dr. Amit Barve |
| Internal Guide | Head of the Department |

Established & Incorporated Under Gujarat Private Universities
(Second Amendment) Act, 2015 (Guj. Act No. 7 of 2015)

Parul®
University

Date: 11/30/2024

**To,**
**W3nuts**
**Rajkot**

Subject: NOC for immediate joining of selected student

Dear Sir / Madam,

This is to inform that **Enrollment No** 2203031457022,**Varasada Shivam Rajeshbhai** 8B20 from our institute is allowed to join from date **02/12/2024** up to **31/05/2025**. This student can join your organisation on full time basis but at the same time, he/she will be required to appear for all Weekly Tests, Mid-Sem Exams, External Semester Exams, vivas, submission and practical exams and must perform satisfactorily in order to become eligible to get degree certificate.

We would request you to kindly consider the same and approve leaves accordingly as per the exam schedule as & when gets finalised.

**Yours Faithfully,**

**Dr. Amit Barve**
Head-Computer Science Engineering Dept.,
Parul Institute of Engineering & Technology,
Parul University, Vadodara.

# Internship Offer Letter

Dear Varasada Shivam Rajeshbhai,

We are pleased to offer you an internship position at W3nuts for the MERN stack development role. We believe your skills and enthusiasm will be a valuable addition to our team.

**Internship Details:**

- **Position:** MERN Stack Intern
- **Start Date:** December 02, 2024
- **Location:** Office-based (Rajkot)

During your internship, you will work closely with our development team on various projects, gaining hands-on experience in the MERN stack. You will have the opportunity to enhance your skills while contributing to real-world applications.

Please confirm your acceptance of this offer by signing and returning this letter by 25/11/2024. If you have any questions or need further clarification, feel free to reach out.

Please be noted that this will be non paid internship.

We are excited to have you on board and look forward to your contributions at W3nuts!

Best regards,

Samir Kaila

For W3nuts

Partner


▪ Web Design & Development
▪ Internet Marketing & SEO
▪ Mobile Application

hello@w3nuts.co.uk
www.w3nuts.co.uk

**Parul Institute of Engineering & Technology, Limbda**


# DECLARATION

We hereby declare that the Internship / Project report submitted along with the Internship / Project entitled **Internship MEP CONSULTANT Project** submitted in partial fulfilment for the degree of Bachelor of Engineering in COMPUTER ENGINEERING to Parul University, Vadodara, is a Bonafide record of original project work carried out by me at W3NUTS under the supervision of Dr. Warish Patel and that no part of this report has been directly copied from any students' reports or taken from any other source, without providing due reference.

| | Name of the Student | Sign of Student |
|---|---|---|
| 1 | Varasada Shivam R. | _____ |

# Acknowledgements

I would like to express my heartfelt gratitude to all those who supported and guided me throughout the development of the **MEP CONSULTANT Website** project. My sincere thanks go to my guide, **Dr. Warish Patel**, for his invaluable insights, continuous encouragement, and constructive feedback, which were pivotal in shaping the final outcome of this project. I am also grateful to my project supervisor and mentors for their unwavering support and technical assistance during challenging phases of development. Special thanks are due to my peers and colleagues for their collaborative spirit and to my educational institution for providing the necessary resources and an inspiring environment for innovation. Lastly, I extend my deepest appreciation to my family and friends for their constant support and motivation, which kept me focused and driven throughout this journey.

# Abstract

The MEP CONSULTANT Website is a comprehensive web-based application developed as an internship project to streamline management processes for MEP consultants. This project centralizes client data management, project tracking, expense recording, and quotation generation into a single, intuitive platform. The system features a dedicated Client Management Module where users can add and update client details—including GST and PAN numbers, contact information, point of contact, and address. A robust Project Management Module allows for the creation and monitoring of projects with detailed information such as project name, status, budget, advance payment, balance, location, and team members. Additionally, the Expense Management Module enables the entry of multiple expenses linked to specific projects and clients, while the Quotation Module facilitates the generation of detailed quotations based on selected services and client requirements.

An integrated dashboard consolidates key performance indicators and data insights, enabling users to make informed decisions quickly. Developed using modern web technologies and frameworks, the application adheres to industry best practices in terms of scalability, usability, and security. The project not only demonstrates full-stack web development capabilities but also provides a tangible solution for real-world business challenges, significantly improving data accuracy and operational efficiency in managing MEP consultancy projects.

# List of Figures

# List of Table

# Abbreviations

**API:** Application Programming Interface

**DB:** Database

**SQL:** Structured Query Language

**CRUD:** Create, Read, Update, Delete

**MVC:** Model-View-Controller

**ORM:** Object-Relational Mapping

**Git:** Distributed Version Control System

# Table of Contents

# Chapter 1

## Overview of the company

### 1.1 History

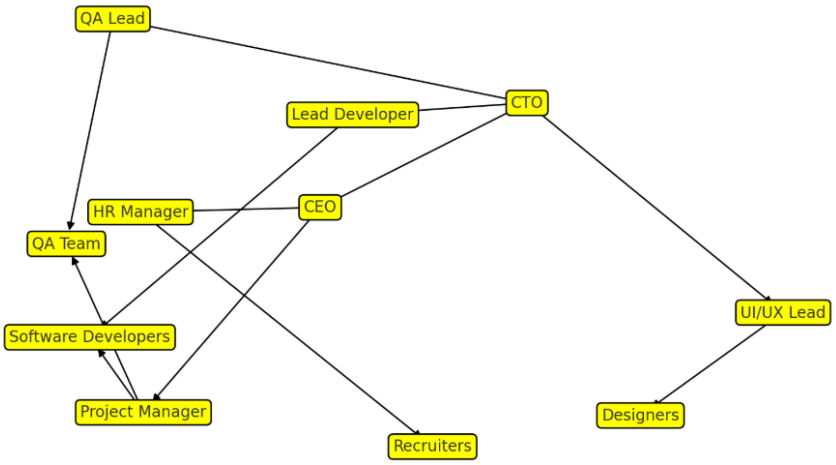
- W3NUTS is a software development company founded with the vision of delivering cutting-edge technology solutions. Established in 2010, the company has rapidly grown to a team of 50 skilled professionals specializing in software engineering, application development, and IT consulting. W3NUTS focuses on innovation, quality, and customer-centric solutions, helping businesses streamline operations through modern software applications.

### 1.2 Scope of the work

W3NUTS specializes in the following areas:

- Custom Software Development – Tailored solutions for various industries.
- Web Development – Scalable and interactive websites.
- Mobile App Development – Android and iOS applications.

### 1.3 Organization chart




### 1.4 Capacity of plant

- With a team of 50 employees, W3NUTS operates efficiently in a collaborative environment. The company has the capacity to handle multiple projects simultaneously, utilizing modern infrastructure, cloud-based workflows, and agile methodologies.

# Chapter 2

## Overview of different process being carried out in company

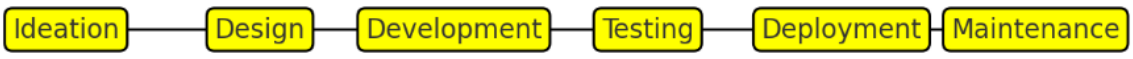
### 2.1 It includes the details about the work being carried out in each department

- Software Development Team: Writes and tests code for various software projects.
- Quality Assurance (QA): Ensures software functionality, performance, and security.
- UI/UX Design Team: Designs user-friendly interfaces and experiences.
- Project Management Team: Oversees project timelines, budgets, and client requirements.
- IT Support & Maintenance: Handles bug fixes, software updates, and customer support.
- Human Resources (HR): Manages employee recruitment, training, and well-being.

### 2.2 List the technical specification of major equipment used in each department

- Servers: Cloud-based AWS/Azure/GCP infrastructure for hosting applications.
- Networking: Secure VPNs, firewalls, and high-speed internet connections.
- Programming: Python, Java, JavaScript, PHP, Html, CSS.
- Frameworks: React, Angular, Bootstrap, Django, NextJS, ExpressJS
- DevOps: Docker, Kubernetes, GitHub, Jenkins

### 2.3 Prepare schematic layout which shows the sequence of operation for




### 2.4 Explain in detail about each stage of production

- Requirement Analysis: Understanding client needs and defining project goals.
- Design & Prototyping: Creating wireframes, UI/UX designs, and architecture planning.
- Development: Writing, testing, and refining the codebase.
- Testing & QA: Running manual and automated tests to ensure quality.
- Deployment & Maintenance: Hosting, monitoring, and updating software solutions.

# Chapter 3

## Introduction to Internship project

### 3.1 Summary

- This internship project involved the development of the MEP CONSULTANT Website, a comprehensive web-based application designed for managing client data, projects, expenses, and quotations. The system streamlines administrative workflows by allowing users to enter and track detailed information related to clients, projects, and financial records—all accessible through an intuitive dashboard.

### 3.2 Purpose

- The primary psurpose of this project was to gain practical experience in full-stack web development and database management while creating a functional business management tool. The website was designed to simplify the processes of managing client information, tracking project progress, recording expenses, and generating quotations.

### 3.3 Objective

- Client Management: Implement a module to add and manage client details—including client name, GST number, PAN number, contact information, point of contact, and address.
- Project Management: Create a project management feature where users can add new projects by selecting a client from a dropdown list and entering details such as project name, status, budget, advance payment, balance, location/address, and team member(s).
- Expense Tracking: Develop an expense module that allows users to associate expenses with specific clients and projects. Users can select an expense category and enter the corresponding amount, with support for adding multiple expense entries.
- Quotation Generation: Design a quotation module that integrates client and project information with service selection and additional details required for generating professional quotations.

### 3.4 Scope

- **What it can do**:
    - Client Module: Add and manage client details including GST and PAN numbers, contact information, and address.
    - Project Module: Enable project creation with comprehensive.
    - Expense Module: Allow multiple expense entries linked to clients and projects for accurate financial tracking.
    - Quotation Module: Facilitate the generation of quotations by selecting appropriate client, project, and service details.
    - Dashboard: Present a summary view of all data inputs for quick insights and reporting.
- **What it can't do:**
    - The system is focused on administrative and tracking functionalities and does not support complex customer relationship management (CRM) features.

### 3.5 Technology and Literature Review

- **Technology**:
    - Backend: Developed using a modern web framework Laravel to handle server-side logic and database interactions.
    - Frontend: Utilized a contemporary frontend framework Bootstrap along with HTML5, CSS3, and JavaScript to build a responsive and interactive user interface.
    - Database: Implemented using a relational database system MySQL to securely store client, project, expense, and quotation data.
    - Additional Tools: Employed version control Git and development tools to streamline coding, testing, and deployment processes.
- Literature Review: Extensive research was conducted on full-stack web development best practices, including MVC architecture, RESTful API design, and responsive UI/UX principles. Reviewed documentation and online tutorials specific to the chosen technology stack to address security, scalability, and performance considerations.

## 3.6 Project / Internship Planning

### 3.6.1 Project / Internship Development Approach and Justification

- Phase 1: Developed the core module for managing client information.
- Phase 2: Added the project management module with dropdown selection and detailed input forms.
- Phase 3: Integrated the expense tracking system to handle multiple entries linked to specific projects.
- Phase 4: Implemented the quotation module with client and project.
- Phase 5: Created a comprehensive dashboard for overall data visualization.

### 3.6.2 Project / Internship Effort and Time, Cost Estimation

- Effort and Time: The project was completed over a period of approximately 4 weeks, with each module being developed and integrated sequentially.
- Cost Estimation: As an internship project, direct costs were kept to a minimum. In a commercial context, potential costs would include developer hours, software licenses, hosting fees, and any necessary third-party integration expenses.

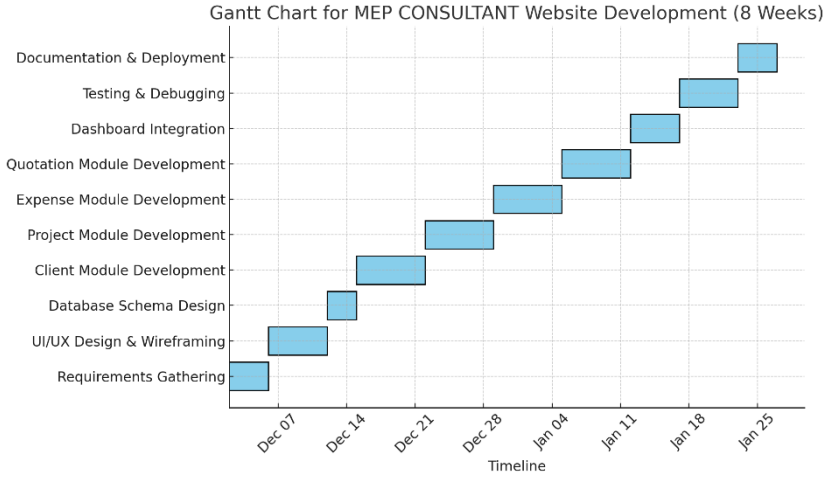
### 3.6.3 Roles and Responsibilities

- As this was an individual internship project, Responsible for both backend and frontend development, ensuring a cohesive user experience Varasada Shivam R.

### 3.6.4 Group Dependencies

- N/A (Solo Project)

## 3.7 Project / Internship Scheduling (Gantt Chart) : Gant Chart Figure (3.7.1)



# Chapter 4

## System Analysis

### 4.1 Study of Current System

- Lack of centralized data storage.
- Difficulty in tracking project finances and status.
- No streamlined method for managing expenses and quotations.
- Limited visibility into overall project performance.

### 4.2 Problems and Weaknesses of Current System

- Inefficient Data Management: Client and project details are not stored in a unified system, leading to data inconsistencies.
- Limited Financial Tracking: No real-time tracking of budgets, payments, and expenses.
- No Automated Reporting: Users must manually compile reports, increasing workload and chances of errors.
- Lack of Integration: Different tools are used for different tasks (e.g., Excel for finances, Word for quotations), making it hard to maintain consistency.

### 4.3 Requirements of New System (MEP CONSULTANT Website)

- Centralized Client Management: Store and retrieve client details efficiently, including GST and PAN numbers, contact details, and addresses.
- Project Tracking: Manage multiple projects with real-time updates on budget, payments, and status.
- Expense Management: Record and track project expenses with ease.
- Quotation Generation: Create detailed quotations based on selected clients, projects, and services.
- Dashboard for Overview: Provide a summary of all key metrics for better decision-making.

## 4.4 System Feasibility

### 4.4.1 Contribution to Objectives

- Enhancing efficiency: Automating client, project, expense, and quotation management.
- Improving accuracy: Reducing manual errors in financial tracking.
- Providing a tangible portfolio project: Demonstrating expertise in web application development.

### 4.4.2 Implementation Constraints

- Use of modern web development frameworks that speed up development.
- Availability of relational databases for structured data storage.
- A clear, well-defined scope that avoids unnecessary complexity.

### 4.4.3 Integration with Other Systems

As a standalone web-based application, no direct integration with third-party systems is required at this stage. However, the system is designed to be scalable, allowing for future integration with external tools like accounting software or payment gateways.

## 4.5 Activity / Process in New System

### 4.5.1 MEP CONSULTANT Website Process.

- User Authentication
- Client Management (Adding/editing client details)
- Project Creation & Tracking
- Expense Entry & Management
- Quotation Generation
- Dashboard Reporting

## 4.6 Features of New System

### 4.6.1 MEP CONSULTANT Website Features

- Client Management Module: Store client details including GST, PAN, contact, and address.
- Project Management Module: Create and track project details such as budget, payments, and status.
- Expense Management Module: Record multiple expenses per project.
- Quotation Module: Generate and store quotations for different projects and clients.
- Dashboard: View all key project, financial, and client information in one place.

## 4.7 Main Modules / Components / Processes

### 4.7.1 System Components

- Client Management System (CRUD operations for client data)
- Project Management System (Project creation, budget tracking, status updates)
- Expense Tracking System (Expense input, linking expenses to projects and clients)
- Quotation Management System (Quotation creation and storage)
- Dashboard System (Summarized reports for decision-making)

## 4.8 Selection of Hardware / Software

### 4.8.1 Development Environment

- Hardware: Standard PC with web development capabilities. If deployed online, a cloud-based server or local server setup will be required.
- Software
    - Frontend: HTML, CSS, Bootstrap, JS for a modern user experience.
    - Backend: Laravel for handling business logic.
    - Database: MySQL for structured data storage.
    - Additional Tools: Git for version control, and UI/UX design tools like Figma

# Chapter 5

## System Design

### 5.1 System Design & Methodology

- The application is developed as a web-based project management tool intended for real-client interaction and incorporates features such as dashboards, client management, project tracking, quotation creation, expense control, recurring payments, and analytical reporting. The overall methodology follows an iterative development approach with rapid prototyping and continuous testing. This ensures that the core functionalities—client creation, project management, expense tracking, and analytics—are built incrementally and refined based on user feedback and testing.
- Key points include:
    - Development Framework: The project is built using Laravel (PHP framework) following the MVC architecture. This structure separates concerns and makes maintenance and scaling easier.
    - Front-end Technologies: HTML, CSS, JavaScript, Bootstrap, and Chart.js are used for creating responsive and interactive interfaces.
    - Tools & Version Control: Git and GitHub are used for source control, enabling collaborative development and version management.
    - Testing: Continuous integration practices are employed to ensure that each module (e.g., client, project, quotation) functions as intended before integration into the main system.

### 5.2 Data Structure Design

- Data management is achieved through Laravel's Eloquent ORM, which maps relational database tables to model classes
- Client Module:
    - Attributes: Client name, GST number, address, phone numbers, and associated points of contact.
    - Relationships: One-to-many with projects.
- Project Module:
    - Attributes: Project name, location, selected expense percentages, team members (with names and contact numbers).

- Relationships: Belongs to a client; one-to-many with quotations and expenses.

## 5.3 Data Structure Design

- The user interface is designed to be intuitive and responsive, making use of modern web technologies to deliver a seamless experience:
- Forms & Views**:**
  - Client Management: Forms to create, update, and delete client information.
  - Project Creation: Interfaces that allow users to assign projects to clients, input project details, and set expense limits.
  - Quotation & Expense Entry: Dynamic forms that auto-generate quotation numbers and provide real-time feedback on expense limits.
  - Recurring Payments: Interfaces to configure payment types and frequencies.

### 5.3.1 State Transition Diagram

• A simple state diagram:

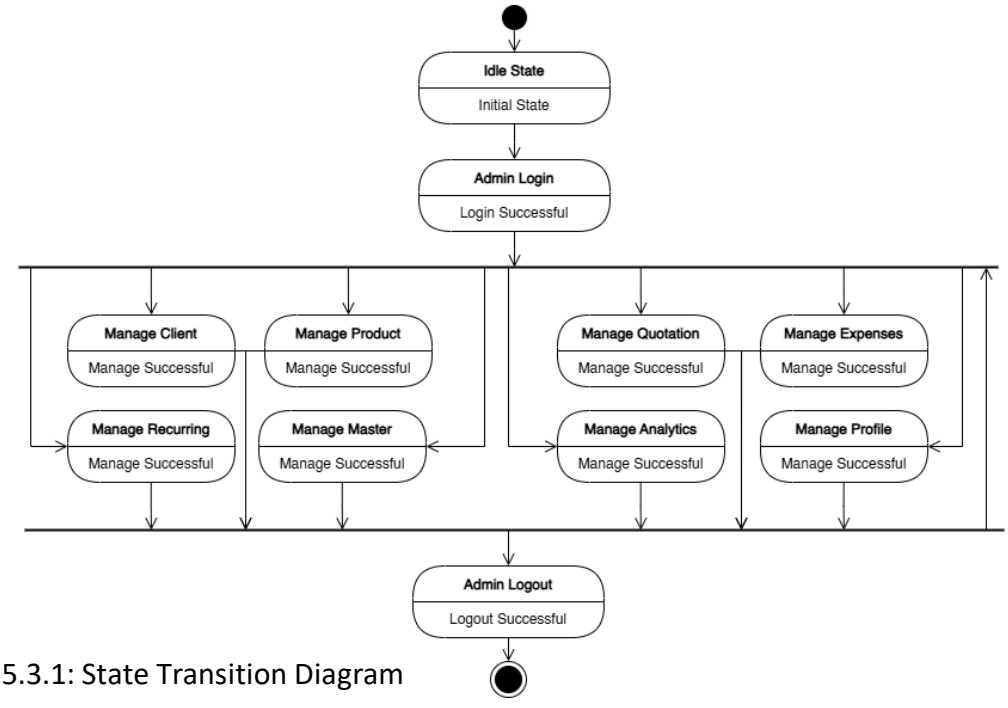



Fig 5.3.1: State Transition Diagram

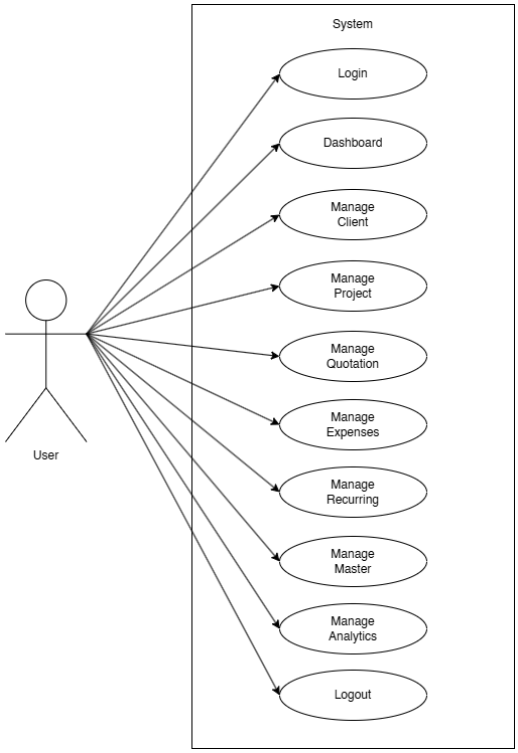



Fig 5.3.2: Use State Diagram

### 5.3.2 Sample of Interface

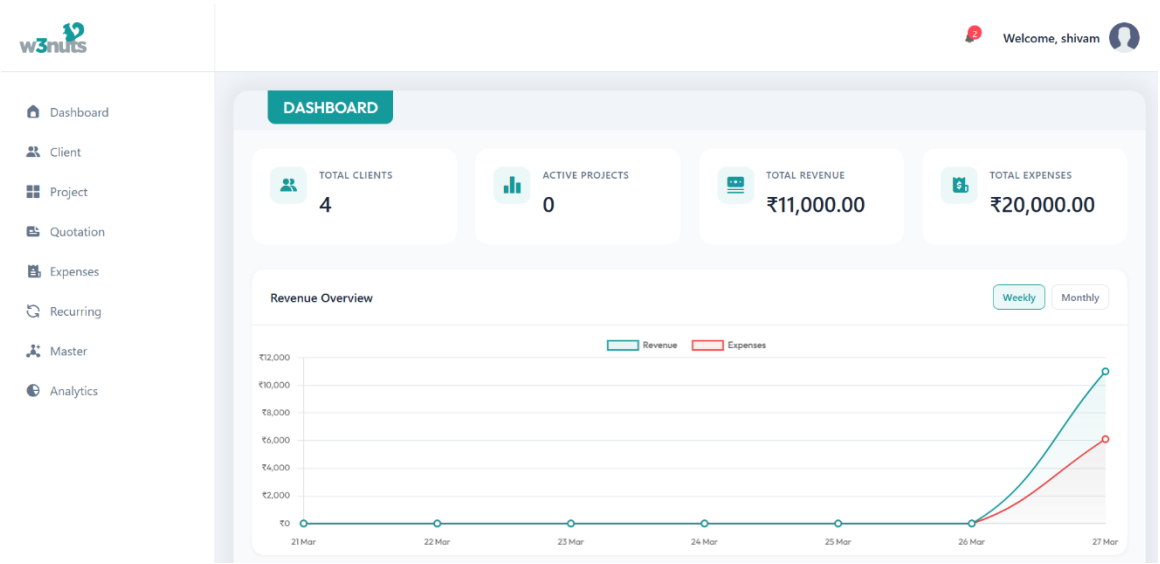



Fig 5.3.2.1: Dashboard

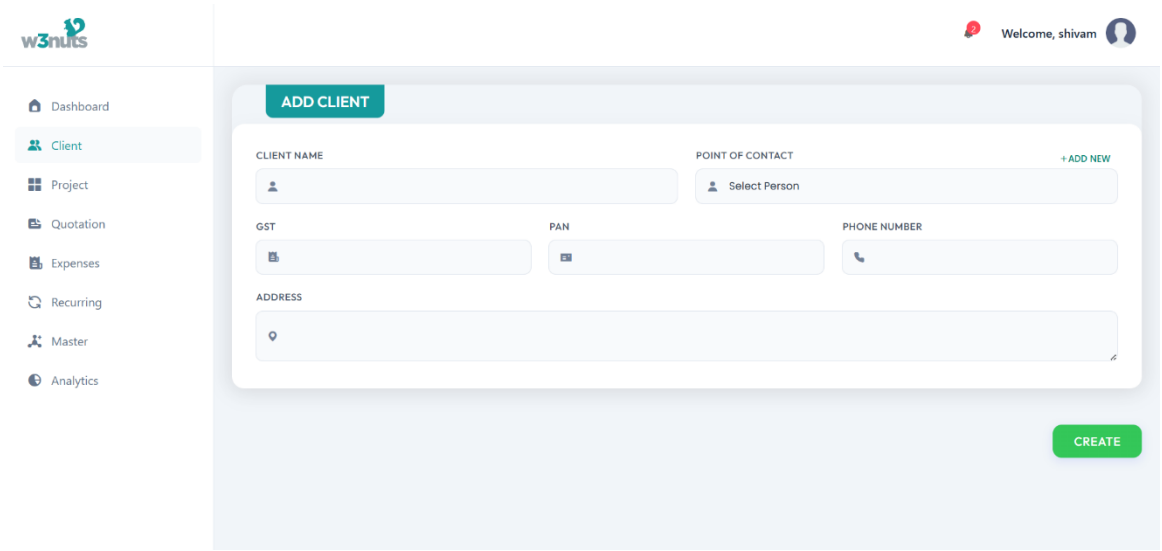



Fig 5.3.2.2: Add Client

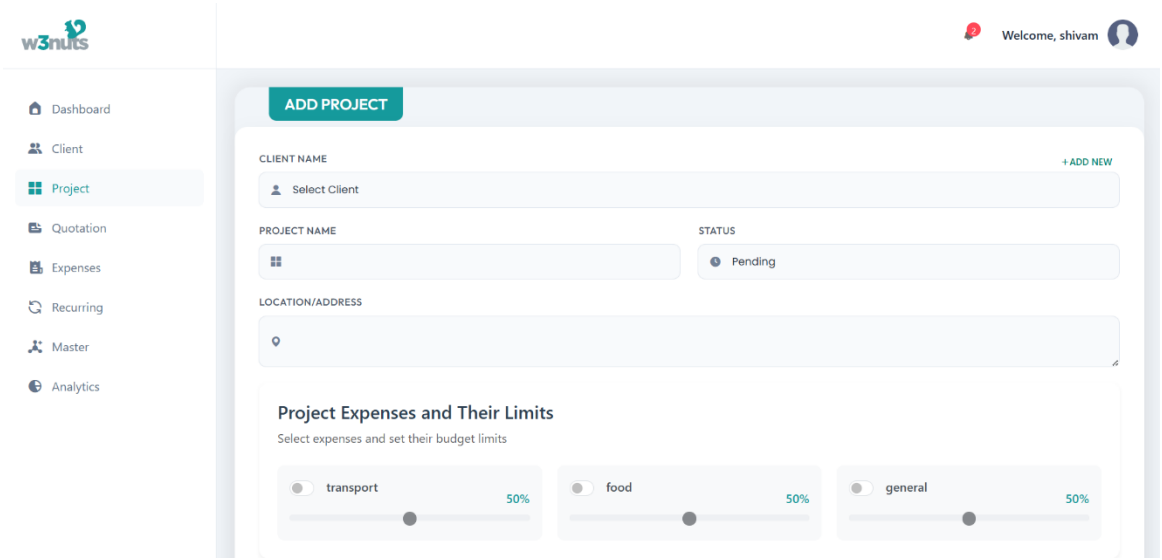



Fig 5.3.2.3: Add Product

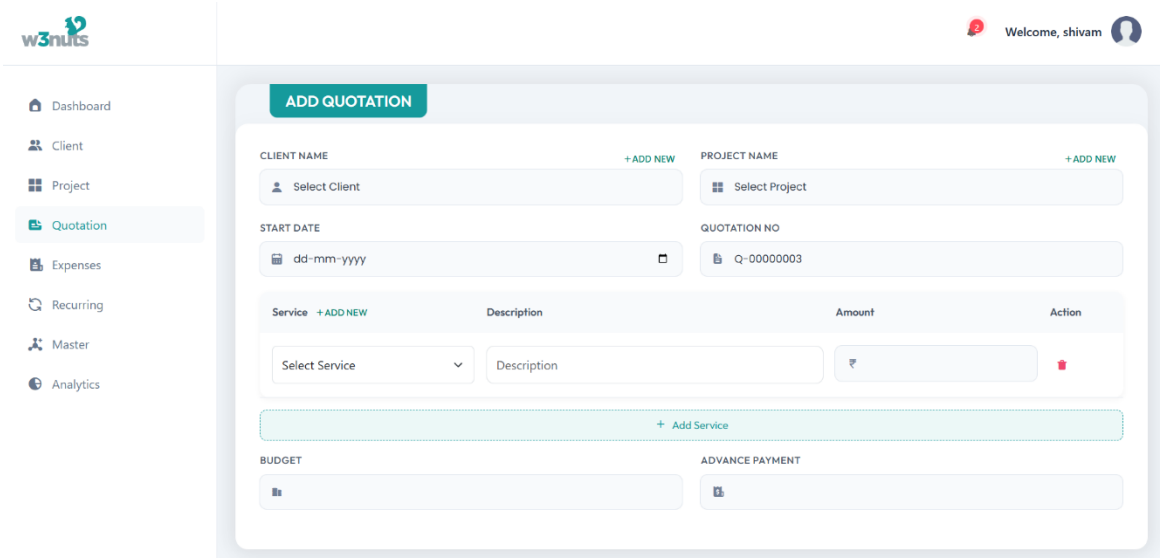



Fig 5.3.2.4: Add Quotation

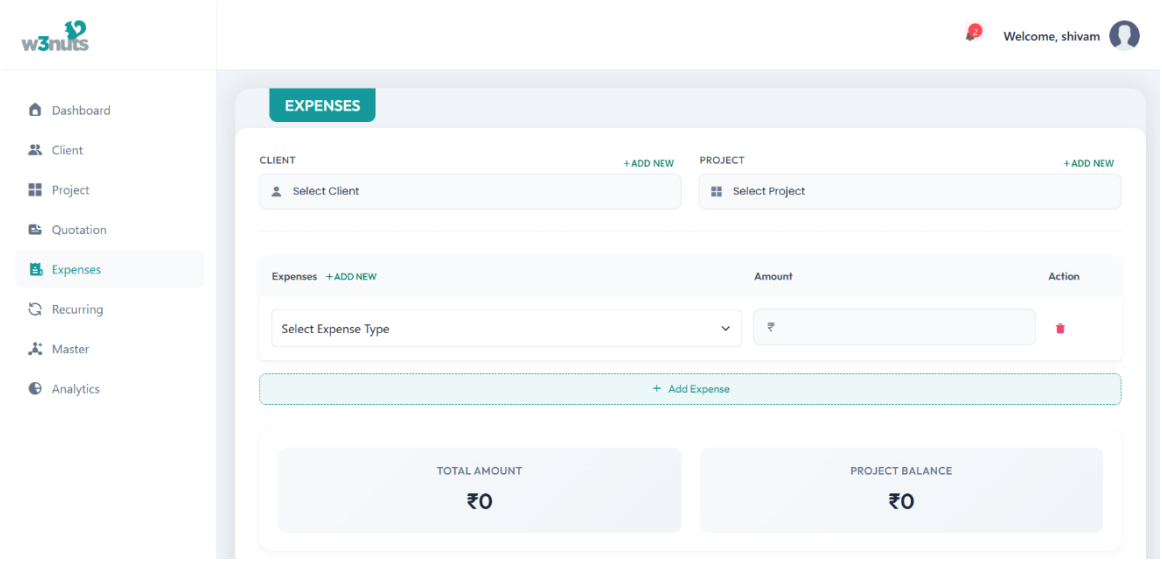



Fig 5.3.2.5: Expenses

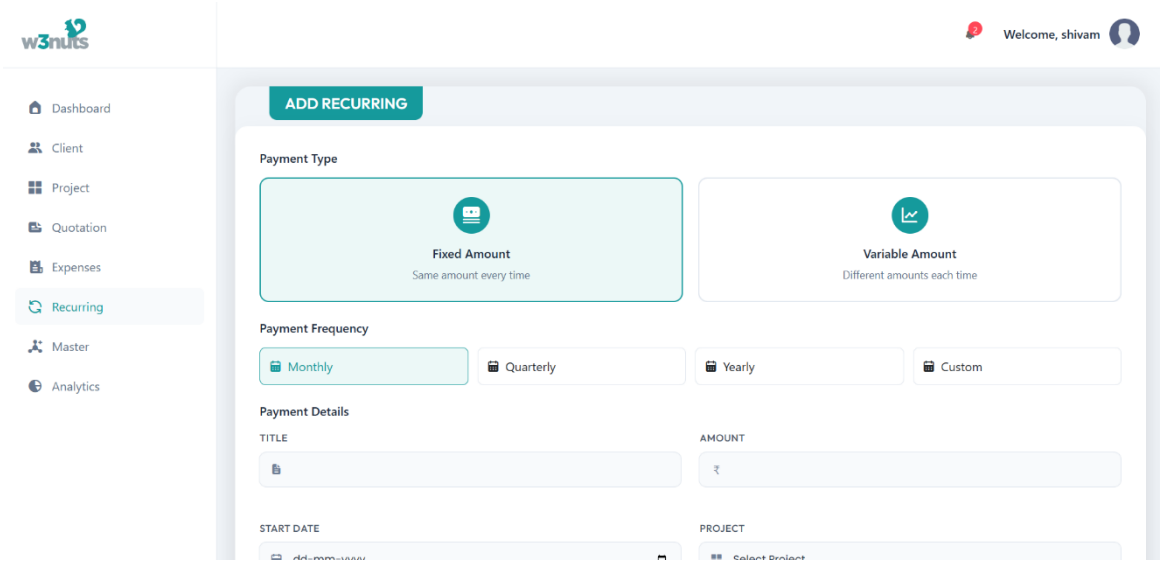



Fig 5.3.2.6: Add Recurring

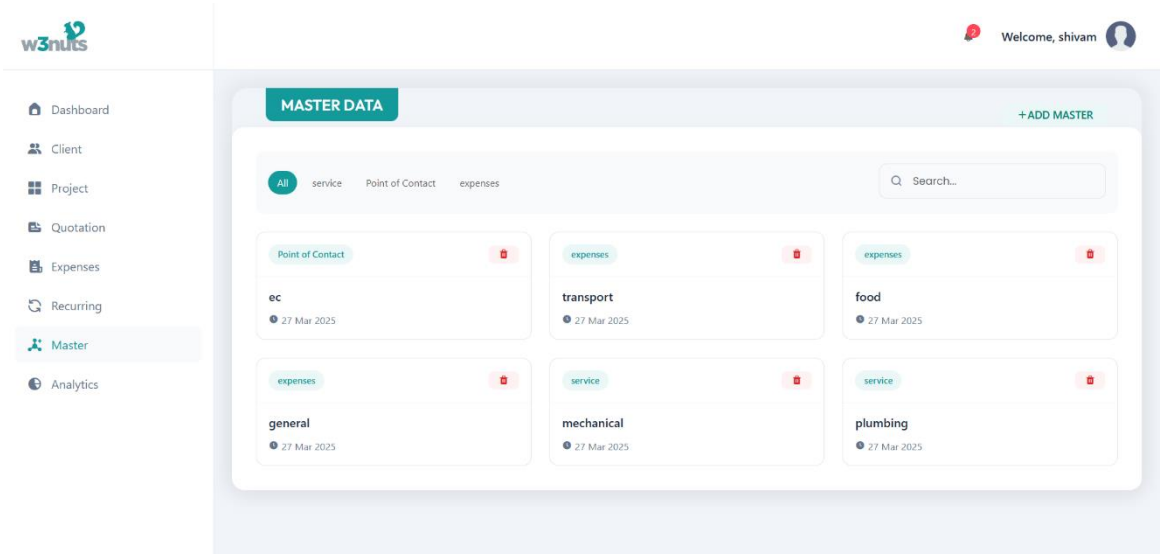



Fig 5.3.2.7: Master

# Chapter 6

## Implementation

### 6.1 Implementation Platform / Environment

- Development Framework: The project management system was developed using Laravel, which follows the MVC architectural pattern.
- Primary Languages & Libraries:
    - Back-End: PHP (Laravel Framework)
    - Front-End: HTML, CSS, JavaScript, Bootstrap
    - Charting: Chart.js for data visualization
- Database: MySQL (or an equivalent relational database) for managing client, project, quotation, and expense data.
- Development Environment:
    - Local development was conducted on a Windows 10 machine.
    - Visual Studio Code was used as the primary code editor, integrated with debugging tools for Laravel and JavaScript.
- Version Control: Git was used for source code management, and GitHub served as the remote repository to facilitate collaboration and version tracking.

### 6.2 Process / Program / Technology / Modules Specification(s)

- Dashboard Module:
    - Functionality: Visualizes overall system metrics, including total revenue, total expenses, and the count of active projects using interactive charts.
    - Implementation:
        - Data is fetched from Laravel controllers, processed, and rendered via Chart.js on the dashboard page.

- Example Code Snippet: Fig 6.2.1

```php
class DashboardController extends Controller
{
    public function index()
    {
        $totalClients = Client::count();
        $activeProjects = Project::where('project_status', 1)->count();

        // Calculate total revenue from quotations
        $totalRevenue = Quotation::sum('budget');

        // Calculate total expenses
        $totalExpenses = Expense::sum('amount');

        // Get recent projects
        $recentProjects = Project::with('client')
            ->orderBy('created_at', 'desc')
            ->take(5)
            ->get();

        // Get recent clients directly from Client model
        $recentClients = Client::orderBy('created_at', 'desc')
            ->take(5)
            ->get();
```

- Client Module:
  - Functionality: Enables users to create, update, and delete client details including names, GST numbers, addresses, phone numbers, and designated points of contact.
  - Implementation:
    - CRUD operations are managed through Laravel controllers and Eloquent models.
    - Example Code Snippet: Fig 6.2.2

```php
class ClientController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index()
    {
        $clients = Client::all();
        return view('client', compact('clients'));
    }
    /**
     * Show the form for creating a new resource.
     */
    public function create()
    {
        $pocMasterType = MasterType::where('name', 'Point of Contact')->first();
        $serviceMasterType = MasterType::where('name', 'Service')->first();

        $pointOfContacts = $pocMasterType ? Master::where('master_type_id', $pocMasterType->id)->get() : collect();
        $pocMasterTypeId = $pocMasterType ? $pocMasterType->id : null;

        return view('add_client', compact('pointOfContacts', 'pocMasterTypeId', 'serviceMasterType'));
    }
```

- Project Module:
    - Functionality: Allows creation of projects by linking them to existing clients, entering project details, and setting expense percentages for budget control.
    - Implementation:
        - Example Code Snippet: Fig 6.2.3

```php
class ProfileController extends Controller
{
    public function edit()
    {
        return view('profile.edit', ['user' => auth()->user()]);
    }

    public function update(Request $request)
    {
        $user = auth()->user();

        $validated = $request->validate([
            'name' => 'required|string|max:255',
            'email' => ['required', 'email', Rule::unique('users')->ignore($user->id)],
            'phone' => 'nullable|string|max:20',
            'address' => 'nullable|string',
            'gst_number' => 'nullable|string|max:15',
            'pan_number' => 'nullable|string|max:10',
            'company_name' => 'nullable|string|max:255',
            'current_password' => 'nullable|required_with:new_password',
            'new_password' => 'nullable|min:8|confirmed',
        ]);

        if ($request->filled('current_password')) {
            if (!Hash::check($request->current_password, $user->password)) {
                return back()->withErrors(['current_password' => 'The current password is incorrect.']);
            }
            $validated['password'] = Hash::make($request->new_password);
        }
```

- Quotation Module:
    - Functionality: Provides the ability to generate quotations tied to specific projects and clients, including auto-generation of quotation numbers and selection of service types (e.g., mechanical, plumbing).
    - Implementation:
        - Quotation numbers are auto-generated using Laravel's built-in functions.
        - Example Code Snippet: Fig 6.2.4

```php
namespace App\Http\Controllers;

use App\Models\Quotation;
use App\Models\Client;
use App\Models\Project;
use App\Models\Master;
use App\Models\MasterType;
use App\Models\QuotationItem;
use Illuminate\Http\Request;
use Barryvdh\DomPDF\Facade\Pdf;

class QuotationController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index()
    {
        $quotations = Quotation::with(['client', 'project', 'service'])->get();
        return view('quotation', compact('quotations'));
    }
```

- Expenses Module:
    - Functionality: Permits entry of expenses against projects with built-in checks for percentage limits. Alerts are shown if an entered expense exceeds the predefined limit, and the amount is deducted from the project balance.
    - Implementation:
        - JavaScript validation is implemented on the client side with additional server-side checks.
        - Example Code Snippet: Fig 6.2.5

```php
class ExpenseController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index()
    {
        $clients = Client::all();
        $projects = Project::where('project_status', 'active')->get();
        $expenses = Expense::with(['client', 'project'])->get();

        $serviceMasterType = MasterType::where('name', 'expenses')->first();
        $services = $serviceMasterType ? Master::where('master_type_id', $serviceMasterType->id)->get() : collect();
        $pointOfContacts = MasterType::where('name', 'Point of Contact')
            ->first()
            ?->masters()
            ->get() ?? collect();

        return view('expenses', compact('clients', 'projects', 'expenses', 'services', 'serviceMasterType', 'pointOfC
    }
```

- Master Module:
    - Functionality: Manages global dropdown values for services, expense types, and contact points.
    - Implementation:
        - Admin interfaces are built to facilitate easy updates to these master lists.
        - Example Code Snippet: Fig 6.2.7

```php
class MasterController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index()
    {
        $masters = Master::all();
        $masterTypes = MasterType::all();
        return view('master', compact('masters', 'masterTypes'));
    }
```

- Recurring Payments Module:
    - Functionality: Supports scheduling recurring payments (fixed or variable) based on configurable frequencies.
    - Implementation:
        - Recurring payment logic is handled via Laravel scheduler (cron jobs) that update project balances and send notifications.
        - Example Code Snippet: Fig 6.2.6

```php
class RecurringPaymentController extends Controller
{
    public function index()
    {
        $recurringPayments = RecurringPayment::with(['project.client'])->get();
        return view('recurring', compact('recurringPayments'));  // Updated view path
    }

    public function create()
    {
        // Get only active projects with their clients and format for dropdown
        $projects = Project::with('client')
            ->where('project_status', 'active')
            ->whereHas('client') // Only get projects that have clients
            ->orderBy('project_name')
            ->get();
```

- Analytics Module:
    - Functionality: Provides a comparative analysis between two selected periods to assess profit and loss, allowing deeper insights into project performance.
    - Implementation:
        - Data is aggregated via Laravel queries and visualized using Chart.js for comparative analysis.
        - Example Code Snippet: Fig 6.2.8

```blade
@section('content')
    <div class="main-page">
        <div class="main-page-content">
            <div class="section-heading mb-4">
                <h3>Analytics</h3>
            </div>
            <div class="form-wrapper">
                <!-- Date Range Comparison Form -->
                <div class="card mb-4">
                    <div class="card-body">
                        <form action="{{ route('reports.yearComparison') }}" method="POST">
                            @csrf
                            <div class="row g-3">
                                <!-- First Period -->
                                <div class="col">
                                    <label class="form-label">First Period</label>
                                    <div class="period-inputs">
                                        <div class="input-group">
                                            <span class="input-group-text">From</span>
```

### 6.3 Finding / Results / Outcomes

- Successful Module Integration:
    - All modules (Dashboard, Client, Project, Quotation, Expenses, Recurring, Master, and Analytics) have been integrated seamlessly to provide end-to-end functionality.
- User Experience:
    - The UI elements, including forms and charts, deliver clear visual feedback and support intuitive navigation.
- Real-Time Notifications:
    - Both email and in-app notifications work reliably to alert users regarding upcoming payment schedules and other critical events.
- Testing & Performance:
    - The system was successfully tested on multiple browsers and devices, confirming that performance meets the project requirements.
- Challenges Overcome:
    - Fine-tuning the expense validation alerts and recurring payment schedules required iterative testing and adjustments, ensuring robust handling of real-client data.

### 6.4 Result Analysis / Comparison / Deliberations

- Performance and Scalability:
    - The application demonstrates efficient data processing and display, even with increasing numbers of projects and transactions. The use of Laravel's Eloquent ORM and MySQL ensures scalability.
- Security and Data Integrity:
    - Role-based access control and thorough input validation have contributed to maintaining data integrity and security.
- Future Enhancements:
    - While initial tests on select devices were successful, further testing across a broader range of devices and browsers is recommended to ensure consistent performance.

# Chapter 7

## Testing

### 7.1 Testing Plan / Strategy

- The testing strategy focused on both functional testing and integration testing of each module, ensuring that individual components (such as Dashboard, Client, Project, Quotation, Expenses, Recurring, Master, and Analytics) work correctly and seamlessly together.
- UI/Responsive Testing: Given that the application is web-based, tests were conducted across multiple browsers (Chrome, Firefox, and Edge) and various screen sizes to ensure responsiveness.
- User Acceptance Testing (UAT): Initial tests were performed by the developer using simulated real-client data. Future tests with end-users are planned to further validate usability and functionality.

### 7.2 Test Results and Analysis

- Module-Specific Observations:
    - Dashboard Module:
        - Chart data reflecting total revenue, expenses, and active projects was displayed accurately.
        - Minor issues with chart scaling on smaller screens were identified and adjusted via responsive CSS settings.
    - Client Module:
        - Creating, updating, and deleting client records functioned as expected.
        - Data validation prevented incomplete or malformed entries.
    - Project Module:
        - Projects linked to clients were successfully created and displayed.
        - Expense percentage limits were correctly enforced, with alerts triggering when inputs exceeded predefined values.
    - Quotation & Expense Modules:
        - Auto-generation of quotation numbers worked reliably.

- Expenses added beyond the allowed limits correctly triggered warning alerts, and the deduction from project balances was accurate.

- Recurring Payments Module:
  - Scheduled recurring payments (both fixed and variable) were properly added to project balances, and notifications were dispatched as per the defined schedule.
- Analytics Module:
  - Comparative reports for selected time periods produced correct /profit and loss insights.

### 7.2.1 Test Cases

| Test ID | Test Condition | Expected Output | Actual Output | Remark |
|---|---|---|---|---|
| TC001 | Create a new client with valid details | New client record appears in the client list | Client created and listed as expected | Pass |
| TC002 | Create a new project for a client | Project is linked to the client and displayed in listings | Project created and associated correctly | Pass |
| TC003 | Generate a quotation for a project | Auto-generated quotation number and quotation appears | Quotation generated with auto-number, correct details | Pass |
| TC004 | Add an expense within allowed percentage limit | Expense is added and deducted from project balance | Expense validated and processed as expected | Pass |
| TC005 | Add an expense exceeding the allowed percentage limit | Alert is triggered; expense amount not deducted improperly | Warning alert shown; expense flagged for review | Pass |
| TC006 | Set up a recurring payment (fixed or variable) | Recurring payment scheduled and project balance updated | Payment scheduled and balance updated correctly | Pass |
| TC007 | Run analytics for two different time periods | Comparative profit and loss report is generated | Analytics module produced correct results | Pass |
| TC018 | Load dashboard on various screen sizes | Dashboard and charts adjust responsively | Charts reflow correctly after CSS adjustments | Pass |

# Chapter 8

## Conclusion and Discussion

### 8.1 Overall Analysis of Internship / Project Viabilities

- The project management system was successfully developed using Laravel, HTML, CSS, JavaScript, Bootstrap, and Chart.js, demonstrating the feasibility of creating an integrated platform for managing clients, projects, quotations, expenses, and recurring payments.
- The project showcased effective modular design and data management, emphasizing real-client interactions and practical financial tracking.
- The iterative development approach allowed for rapid prototyping, continuous testing, and incremental improvements, ensuring that the core functionalities were robust and user-friendly.
- The internship/project provided invaluable hands-on experience in full-stack web development, project management, and system integration, reinforcing both technical and problem-solving skills.

### 8.2 Problem Encountered and Possible Solutions

- Expense Validation Complexity:
  - Problem: Implementing percentage-based expense limits and ensuring accurate balance deductions proved challenging, especially when handling edge cases and alert conditions.
  - Solution: Enhanced validation logic was implemented using both client-side JavaScript and server-side Laravel checks, with iterative testing to refine the alert thresholds and balance calculations.
- UI Responsiveness Across Devices:
  - Problem: Some UI elements and charts did not initially adjust well on different screen sizes, affecting the user experience on mobile and smaller devices.
  - Solution: Responsive design principles were further applied using Bootstrap's grid system and media queries, and additional testing was conducted across various devices to ensure consistent display.
- Data Consistency and Relationship Management:

- Problem: Ensuring the integrity of relational data between clients, projects, and quotations required careful design, especially during CRUD operations.
- Solution: Laravel's Eloquent ORM was leveraged to enforce model relationships and cascade operations, and comprehensive testing was performed to validate data consistency.

### 8.3 Summary of Internship / Project Work

- The project involved developing a comprehensive project management system that streamlines client management, project tracking, quotation generation, expense monitoring, and recurring payment scheduling.
- It was developed using an iterative approach, which allowed for continuous improvements based on developer feedback and early testing results.
- The game was developed using an iterative approach, with rapid prototyping and continuous testing.
- The implementation included a range of modules—Dashboard, Client, Project, Quotation, Expenses, Recurring, Master, and Analytics—each contributing to the overall functionality and user experience.
- Testing, both functional and integration-based, confirmed that all key features were operational and met the project requirements.
- Overall, the project work provided a practical environment to apply and enhance skills in full-stack development, system design, and user interface design.

### 8.4 Limitation and Future Enhancement

- Limitations:
  - The system was primarily tested on a limited number of devices and browsers; broader cross-platform testing is needed.
  - Some advanced features, such as dynamic reporting and real-time analytics, have been implemented in a basic form and could benefit from further refinement.
  - Integration with external systems (e.g., payment gateways, CRM tools) was not included in the initial release.

- Future Enhancements:
  - Expand testing across a wider array of devices and browsers to further ensure responsiveness and compatibility.
  - Enhance analytics capabilities by incorporating real-time data processing and more detailed reporting features.
  - Integrate additional features such as automated client reminders, enhanced security protocols, and comprehensive audit logs.
  - Develop a mobile-responsive or native mobile application version to increase accessibility.
  - Consider integration with third-party services (e.g., payment processors, CRM systems) to extend the platform’s functionality and usability.

# Reference


1. Laravel Documentation**.** (2024). *Laravel: The PHP Framework for Web Artisans.* Retrieved March 28, 2024, from https://laravel.com/docsMDN Web Docs. (2024).
2. MDN Web Docs**.** (2024). *HTML: HyperText Markup Language.* Retrieved March 28, 2024, from https://developer.mozilla.org/en-US/docs/Web/HTML
3. MDN Web Docs. (2024). *CSS: Cascading Style Sheets.* Retrieved March 28, 2024, from https://developer.mozilla.org/en-US/docs/Web/CSS
4. MDN Web Docs. (2024). *JavaScript.* Retrieved March 28, 2024, from https://developer.mozilla.org/en-US/docs/Web/JavaScript
5. Bootstrap. (2024). *Bootstrap Documentation.* Retrieved March 28, 2024, from https://getbootstrap.com/docs
6. Chart.js. (2024). *Chart.js Documentation.* Retrieved March 28, 2024, from https://www.chartjs.org/docs/latest/
7. Git SCM. (2024). *Git Documentation.* Retrieved March 28, 2024, from https://git-scm.com/doc
8. GitHub. (2024). *GitHub Documentation.* Retrieved March 28, 2024, from https://docs.github.com/en
9. W3C. (2024). *Web Accessibility Initiative (WAI).* Retrieved March 28, 2024, from https://www.w3.org/WAI/
10. OWASP. (2024). *OWASP Top 10 Web Application Security Risks.* Retrieved March 28, 2024, from https://owasp.org/www-project-top-ten/
11. Nielsen Norman Group. (2024). *User Experience (UX) Design.* Retrieved March 28, 2024, from https://www.nngroup.com/