WEB-BASED PAYMENT TRACKING AND ACCOUNTING APPLICATION FOR SMALL BUSINESSES

Submitted in partial fulfillment of the
requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering

By

JAYESH BAIBHAV (Reg. No - 39110403)

JYOTISHMAN GHATAK (Reg. No - 39110421)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE

JEPPIAAR NAGAR, RAJIV GANDHI SALAI,

CHENNAI - 600119

APRIL - 2023

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Jayesh Baibhav (Reg. No - 39110403) and Jyotishman Ghatak (Reg. No - 39110421)** who carried out the Project Phase-2 entitled **"WEB-BASED PAYMENT TRACKING AND ACCOUNTING APPLICATION FOR SMALL BUSINESSES"** under my supervision from January 2023 to April 2023.

Internal Guide

[illegible]

Dr. J. ALBERT MAYAN, M.E., Ph.D.

[illegible]

Head of the Department

Dr. L. LAKSHMANAN, M.E., Ph.D.

SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
DEPARTMENT OF CSE

Submitted for Viva voce Examination held on 20.4.2023

[illegible] [illegible]

Internal Examiner **External Examiner**

DECLARATION

I, **Jayesh Baibhav (Reg. No- 39110403),** hereby declare that the Project Phase-2 Report entitled "**WEB-BASED PAYMENT TRACKING AND ACCOUNTING APPLICATION FOR SMALL BUSINESSES"** done by me under the guidance of **Dr. J. ALBERT MAYAN, M.E., Ph.D.** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering**.

DATE:

Jayesh Baibhav

PLACE: Chennai **SIGNATURE OF THE CANDIDATE**

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala, M.E., Ph.D.**, **Dean**, School of Computing, **Dr. L. Lakshmanan, M.E., Ph.D.,** Head of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. J. Albert Mayan, M.E., Ph.D.** for his valuable guidance, suggestions, and constant encouragement paved way for the successful completion of my phase-2 project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Payment Tracking System is a web-based application for tracking and managing the payments for various vendors. It provides a single point of contact that consolidates payment requests from the accounts department to the top management to deliver the supplier payments on time, using integrated best practices to manage operations and services. It offers integrated Transaction Management capabilities like Invoice Generation, Payment Reminders. It ensures visibility, insight, isolation, and faster resolution of Payment related issues for any type of organization by providing the right information at the right time as required by the user. In the business world, it is quite a tedious process to invoice clients and chase them for payments. If you are running a business or planning to start in the future, this project will provide you an application to track the payments from clients and give you an insight of earnings. This application will not only create, send, and receive the invoices but also generate automated follow-up reminders for due payments. With the help of this application client can also avoid late fee.

LIST OF FIGURES

LIST OF ABBREVIATIONS

S.NO.	ABBREVIATION	EXPANSION
1.	UI	User Interface
2.	API	Application Programming Interface
3.	AWS	Amazon Web Services
4.	CI/CD	Continuous Integration/Continuous Deployment
5.	XSS	Cross Site Scripting
6.	CSRF	Cross-site Request Forgery
7.	OWASP	Open Web Application Security Project
8.	JSON	JavaScript Object Notation
9.	CRM	Customer Relationship Management
10.	SQL	Structured Query Language
11.	SSL	Secure Socket Layer
12.	TLS	Transport Layer Security
13.	MFA	Multi Factor Authentication
14.	ERP	Enterprise resource planning
15.	DOM	Document Object Model
16.	WAF	Web Application Firewall

CHAPTER 1
INTRODUCTION

Payment Tracking System is a web-based application for tracking and managing the payments for various vendors. It provides a single point of contact that consolidates payment requests from the accounts department to the top management to deliver the supplier payments on time, using integrated best practices to manage operations and services. It offers integrated Transaction Management capabilities like Invoice Generation, Payment Reminders. It ensures visibility, insight, isolation, and faster resolution of Payment related issues for any type of organization by providing the right information at the right time as required by the user. The purpose of the project will be a Web-based system for tracking the payments made to the suppliers and allows businesses to automate the process and provides an interface to keep track of the businesses. This project uses Node.js with React as front end and it has the following features. The primary aim of this software is to provide an improved design methodology, which envisages the future expansion, and modification, which is necessary for a core sector like manufacturing industries. This necessitates the design to be expandable and modifiable and so a modular approach is used in developing the software. The software has been developed using the most powerful and secure NoSQL database MongoDB and the most widely accepted open-source server environment Node.js. The software is meant to overcome the drawbacks of the manual system.

CHAPTER 2

LITERATURE SURVEY

1] **Chandrashekhar M V** proposed a system that facilitates smart billing and invoicing for users. Users can add customers and create invoices. The system follows a 2-tier architecture. Additionally, the users are provided with sales reports and inventory reports. This system aims to provide invoicing services and financial management to retail business owners.

2] **P. Thanapal** proposed an expense tracker to prevent having to calculate income and expenses, as well as to remind someone to keep their expenses in track and also to add some details on how much money comes from other people and what expenses or payments the user have to make on a given date or month, User have categories in the expenditure tracker such as add expense, monthly expenses, add new expense, see categories of spending, export expenses in a date range, remove export files, and view expenses by category.

3] **S. Chandini** proposed an expense tracker that will maintain all the expenses record of users and manage them efficiently. The user can choose an expense category and provide additional information such as a photo, a location, and the amount of the expense, among other things. This will save the information to the local database. The user can examine and sort expenses on a weekly, monthly, or annual basis. By utilizing this, they reduced the quantity of manual calculations for their expenses and maintain track of their spending. The user can enter his income to compute his total daily expenses, and the data will be saved for each individual user. This tracker could be useful for people who frequently go on trips or to the theatre with their buddies. This tracker will make it easier for them to disburse the bill. This will show the graph in the chosen view.

4] **Karim, Md. Abdul** proposed an expense tracker to create a system for recording expenses and income that is simple, quick, and easy to use. This project also includes features that will assist the user in maintaining all financial operations, such as a digital automatic diary. So, in order to create a better expense tracking system, they created a project that will greatly benefit the users. Most people are unable to track their expenses and income, resulting in financial difficulties. In this scenario, a daily cost tracker can assist people in tracking their income and expenses on a daily basis, allowing them to live a stress-free life.

2.1. INFERENCES FROM LITERATURE SURVEY

Based on the literature review, we conclude that there are existing systems designed to ease business processes. The systems are helping business owners with smart billing facilities. There are systems that are intended for single users to track their expenses and are helping them to overcome or avoid financial difficulties. The applications track the user’s expenses on a daily basis, monthly basis, yearly basis and categorize their spending to provide them insights into their spending. Furthermore, some applications allow users to split their bills among friends, which simplifies tedious calculations and saves a lot of time. These applications are useful for personal usage but there are few existing systems which are intended for small business owners or retailers.

2.2. EXISTING SYSTEM

In order to justify and appreciate the need of the system, we have to study the existing system. There was no software available for tracking, managing and follow-up reminders for payment to the client under a single application. The company use to maintain all their transaction details manually. Even though Software like Tally has similar features like Payment Tracking which includes Company, Supplier, Employee Management with Ledger and other advanced options. But there is no software available yet for tracking of client payment with accounting tools integrated within a single application. Applications like Mobills, Prism, and TimelyBills do provide a similar interface where users can manage their expenses and get follow-up reminders for their undue payments. These applications allow user to manage their budget and gets insights of their spending. Applications like Invoice Ninja and Invoice plane helps user to create customized invoices. Note that the proposed mechanisms will provide an interface to the businesses to meet their all-accounting needs.

2.3. PROBLEM IN EXISTING SYSTEM

The purpose of this project will be a Web based system for the company to keep track of the payments made to their suppliers. This system will be designed to satisfy the accounting needs of small businesses and freelancers. The purpose is to simplify the payment processing method and to keep a hold on all the transactions happening between the client and the organization which decreases the discrepancy in the transactions also improve efficiency of the business.

Existing applications fail to integrate all the modules needed in a single application. Businesses need to use third party applications to generate invoices. Moreover, these applications are less business oriented. The applications are intended for personal use and cannot be used for business or freelancing. Users are reminded of their own undue payments through these applications instead of reminding customers or clients of businesses of their own undue payments. Although these existing apps provide the ability to send reminders, they do not provide the capability to track, manage, and generate a bill/invoice in a single application. Additionally, the existing apps do not offer cryptocurrency payment options which is an important feature of modern transaction protocols.

Other reasons for implementation include better security and integration of system, user-friendliness, and standardization of operations. The implementations are lengthy by nature and require due diligence in accurate to the documentation of requirements, process blueprinting and organizational change management.

CHAPTER 3
REQUIREMENT ANALYSIS

3.1. FEASIBILITY STUDIES/RISK ANALYSIS OF THE PROJECT

The historical background of the business is figured using a well-designed feasibility study i.e., right from description of the product or service to financial data and tax obligations, Feasibility studies helps in development and implementation of the project. During this study we concentrate on three primary area of interest.

- Economic feasibility
- Operational feasibility
- Technical feasibility

A. Economic Feasibility

The effectiveness of the system is evaluated using the Economic feasibility method. Commonly known as the cost benefit analysis, this method helps in finding out whether the system being developed is economic with respect to business application point of view. It helps us to find if the development of the software is cost effective and feasible.

B. Operational Feasibility

In order to measure how well a proposed system solves the problems, takes advantage of the opportunities identified during scope definition an operational feasibility is used. The requirements identified in the requirements analysis phase of system development is satisfied during this method. Here in our system, it regulates the flow of information inside the boundaries of the organization in addition the system manages the database which is available at the Internet level.

C. Technical Feasibility

The technical feasibility system deals with the analysis of technical consideration and also the requirements of the company for the development of the project. The analysis was done in the project and it was successful because this can enter a large amount of data in effective time. It is customized software which is developed for the company who does their monthly transaction in crore.

3.2. SOFTWARE REQUIREMENTS

- **Technologies: HTML5, CSS, Bootstrap, JavaScript, Node.js, React**
  - i) ***HTML5*** is a markup language used for structuring and presenting content on the World Wide Web. It is the fifth and final major HTML version that is a World Wide Web Consortium recommendation. The current specification is known as the HTML Living Standard.
  - ii) **CSS** (Cascading Style Sheets) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML or XML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.
  - iii) **Bootstrap** is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains HTML, CSS and JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components.
  - iv) **JavaScript**, often abbreviated as JS, is a programming language that is one of the core technologies of the World Wide Web, alongside HTML and CSS. As of 2022, 98% of websites use JavaScript on the client side for webpage behavior, often incorporating third-party libraries.
  - v) **Node.js** is an open-source, cross-platform, back-end JavaScript runtime environment that runs on a JavaScript Engine and executes JavaScript code outside a web browser, which was designed to build scalable network applications
  - vi) **React** is a free and open-source front-end JavaScript library for building user interfaces based on UI components. It is maintained by Meta and a community of individual developers and companies.

- **UI Framework: Material-UI (MUI)**
  - i) **Material-UI** is simply a library that allows us to import and use different components to create a user interface in our React applications.

- **Database: MongoDB**
  - i) **MongoDB** is a source-available cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas. MongoDB is developed by MongoDB Inc. and licensed under the Server-Side Public License which is deemed non-free by several distributions.

CHAPTER 4

DESCRIPTION OF PROPOSED SYSTEM

Taking a closer look at all the above methodologies, the recurring weakness is that most of these systems were implemented without considering all the accounting needs in a business in a single application which results in poor user experience and less utility of the applications. Moreover, these applications are less business oriented. The applications are intended for personal use and cannot be used for business and freelancing. Users are reminded of their own undue payments through these applications instead of reminding customers or clients of businesses of their own undue payments. Although these existing apps provide the ability to send reminders, they do not provide the capability to track, manage, and generate a bill/invoice in a single application. Additionally, existing apps do not support cryptocurrency payments, which is an important feature of modern transaction protocols.

To counter these shortcomings, we have proposed a system which will enable businesses and freelancers to track, manage, and generate invoices under a single application. Our system aims to help small business and freelancers to manage their business and work in an efficient way and to be profitable. We do so by providing business with all-in-one app experience and rule out discrepancy and inconsistency in managing accounts. The application can send follow-up reminders to the clients of the business so that they do not miss out payments before the due date, that helps business to save a lot of time in chasing clients for payments. The user-friendly dashboard provided within the application helps the user to gain insights of their business earnings and expenditure and can help them to make useful decisions in their business and the in-app payment option make transactions seamless. With the integration of cryptocurrency as a payment method will enable user to accept payments in cryptocurrencies such as Bitcoin, Ethereum etc. In terms of business applications, the system being developed is economically sound. It is cost effective. The development cost and operation cost are incurred by the project is feasible. The proposed system is available at internet level so that the different types of end users are involved in the system. Its purpose is to facilitate the flow of information between all functions inside the boundaries of the organization and manage the database. In this the system we look between the requirements of the company, this can enter a large amount of data in effective time. The analysis was done in the project and it was successful.

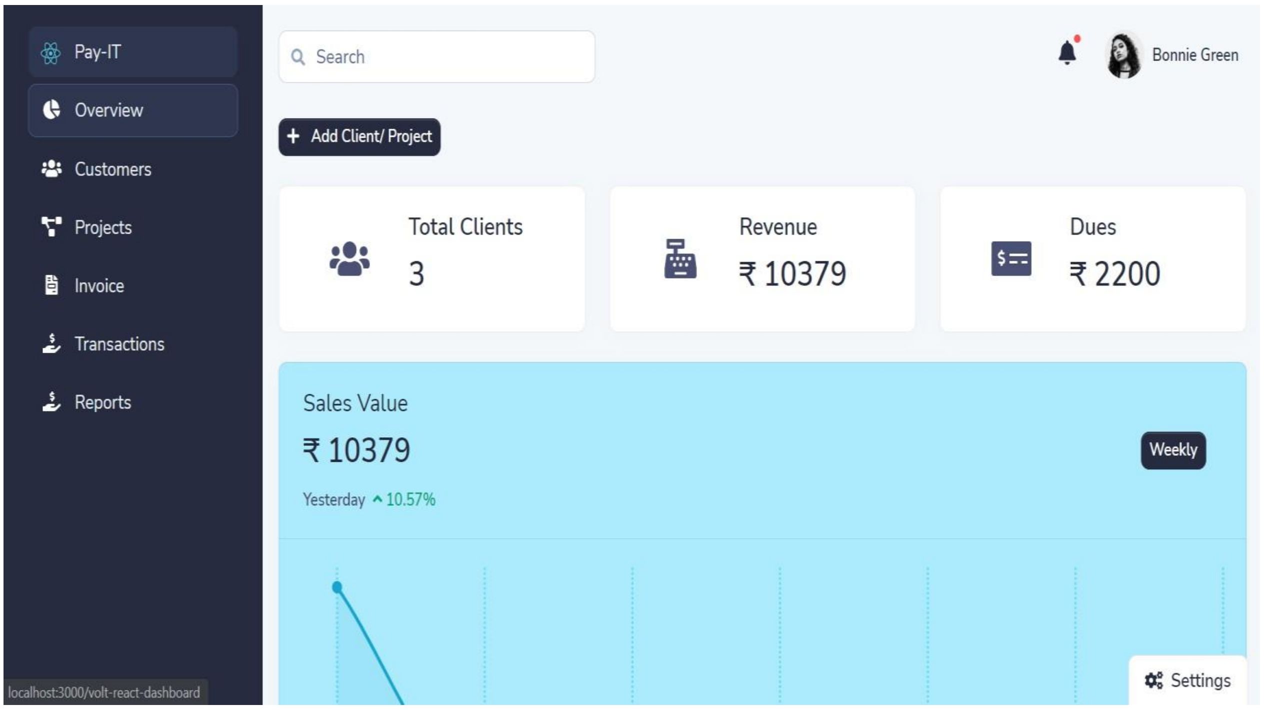


Fig 4.1: A Visual Representation of the Proposed Application

4.1. SELECTED METHODOLOGY OR PROCESS MODEL

This application will be designed to provide a hassle-free experience to the businesses. The purpose is to simplify the payment processing method and to keep a hold on all the transactions happening between the client and the organization which decreases the discrepancy in the transactions also improve efficiency of the business. The application is designed to provide a user-friendly experience to the organization for managing its accounting needs. The user is required to register their organization with their details and login credentials. The user can then use the login credentials to login to the application. An interactive dashboard is created where the organization can keep track of its business. The user can add details of their clients and generate customized invoices. An automated process will check the due date of the client payments and send follow-up reminders to them through the mail. The mail is attached with the generated invoice and a payment link. This allows clients to pay with ease. The user also has the option to track and manage upcoming and due payments. It also provides a list of clients where they can track those whose payments are due and those whose payments have been completed. Once the transaction is completed the amount gets updated in the business insights section. Using this section, users can analyze and visualize business performance and make useful business decisions.

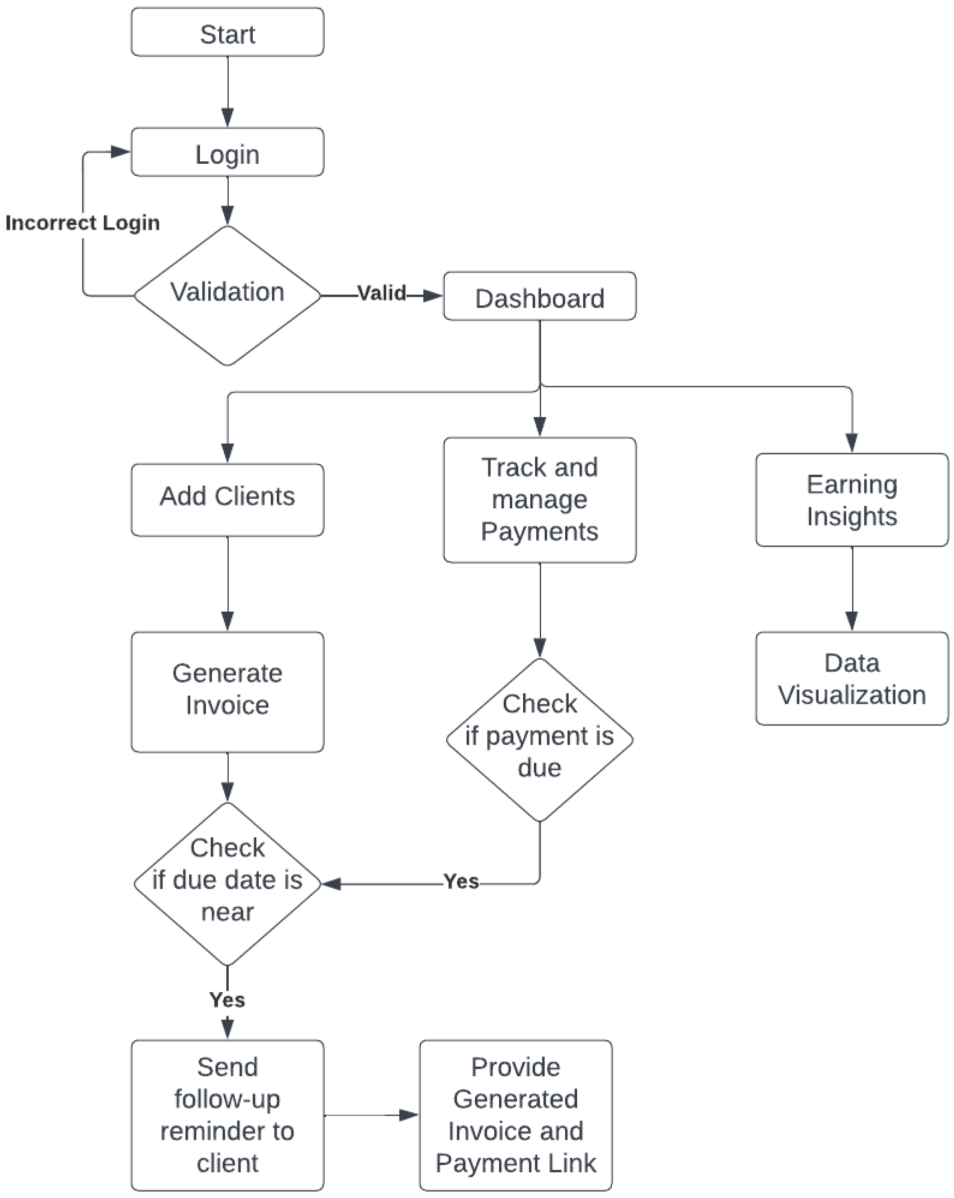


Fig 4.2: Process Flow Diagram

4.2. ARCHITECTURE / OVERALL DESIGN OF PROPOSED SYSTEM

The sytem will comprise of an web based application and will follow a 3- tier M.E.R.N stack architecture. The frontend of the application is created with HTML, CSS, and ReactJS and the backend server is created with NodeJs, and ExpressJS. The database used for this application is a NoSQL Database i.e. MongoDB. The client side of the application uses ReactJs to make the API calls using axios library. ReactJS is designed in such a manner that follows unidirectional data flow or one-way data binding. The benefits of one-way data binding give you better control throughout the application. ReactJS uses JSX file which makes the application simple and to code as well as understand. ReactJS is a component-based approach which makes the code reusable as your need. React Js virtual components turn into the DOM leading to smoother and faster performance resulting in great user experience. The server side of the application is used to create API gateways using which the frontend and the backend can communicate. The web server is created using ExpressJS which is a framework based on Nodejs. Node.js is event-driven and thus has the ability to handle thousands of client requests at the same time which is not possible with PHP. In today's world, real-time web apps and services are increasing in popularity. Node.js is designed exclusively to support real web applications. The database tier is using MongoDB, it uses the concept of sharding to scale horizontally by splitting data across multiple MongoDB instances. MongoDB can run over multiple servers, balancing the load and/or duplicating data to keep the system up and running in case of hardware failure.

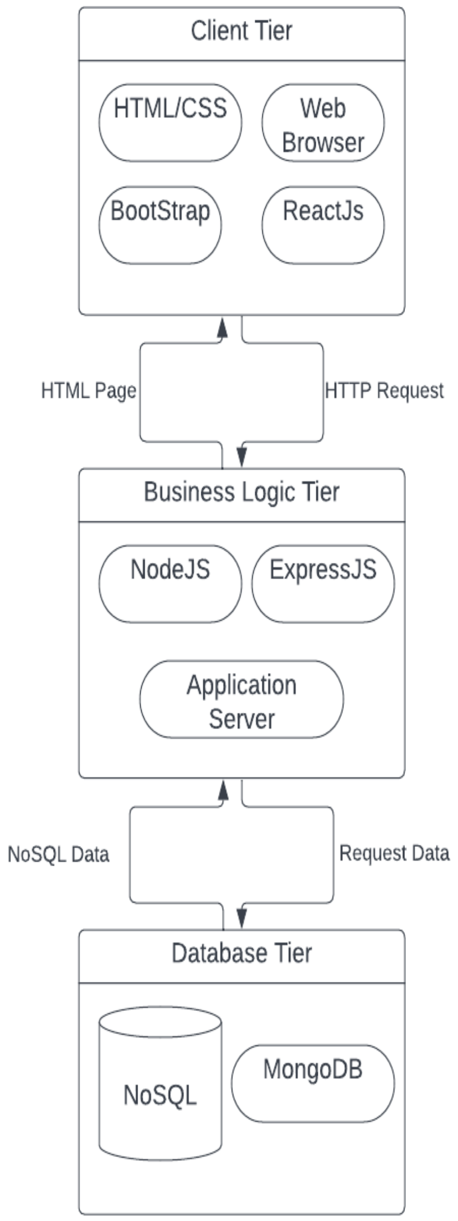


Fig 4.3: 3-Tier Architecture Diagram

4.3. DESCRIPTION OF SOFTWARE FOR IMPLEMENTATION AND TESTING PLAN

The System requires only a web-browser and it is developed as an open-source application. This system can either reside on a centralized server or be distributed across modular hardware and software units that provide "services" and communicate on a local area network. Present system is built using Node.js and React as front end and MongoDB as database. As per user's requirement whole program is designed. Additional constraints can be added to the project.

4.4. PROJECT MANAGEMENT PLAN

The System requires only a web-browser and it is developed as an open-source application built using Node.js and React as front end and MongoDB as database. Further expansion of the system also can be done in future if the company wanted to do so. The database and the information can be updated to the latest forthcoming versions. There are also possibilities for enhancing and further developing the project with developing an android app. Thus, the system can be altered in accordance with the future requirements and advancements. As per user's requirement whole program is designed. Additional constraints can be added to the project. Experimental results prove the dominance of our proposed schemes as compared to counterparts.

4.5. FINANCIAL REPORT ON ESTIMATED COSTING

The system being developed is economic with respect to business application point of view. It is cost effective. The development cost and operation cost are incurred by the project is feasible. The proposed system is available at internet level so that the different types of end users are involved in the system. Its purpose is to facilitate the flow of information between all functions inside the boundaries of the organization and manage the database.

CHAPTER 5

IMPLEMENTATION DETAILS

5.1 DEVELOPMENT AND DEPLOYMENT SETUP

5.1.1 Development Environment Setup:

- Install NodeJS, which includes npm (Node Package Manager).
- Choose a code editor such as Visual Studio Code, Atom, or Sublime Text.
- Use a version control system such as Git to keep track of changes and collaborate with other developers.

5.1.1.1 Backend Development

- Use NodeJS to build the backend of the application.
- Choose a web framework such as ExpressJS to build the API endpoints and handle requests and responses.
- Use a database such as MongoDB to store and retrieve data.
- Use a package such as Mongoose to interact with the database.

5.1.1.2 Frontend Development

- Use ReactJS to build the frontend of the application.
- Use a frontend framework such as Bootstrap to design and style the UI.
- Use a package manager such as npm to install and manage frontend dependencies.

5.1.1.3 Storage and Payment Gateway

- Use a containerization platform such as Docker to package the application and its dependencies into a portable container.
- Use AWS Elastic Beanstalk to deploy the application to a scalable and managed cloud-based platform.
- Use a continuous integration and continuous deployment (CI/CD) tool such as Jenkins or Travis CI to automate the deployment process.

5.2 MODULES

5.2.1 Dashboard:

The Dashboard module is a highly beneficial tool that allows businesses to access a comprehensive overview of various key metrics related to their operations, all within a single page. This interactive page provides an intuitive and user-friendly interface, making it easy for users to navigate and access critical information quickly.

By integrating multiple metrics such as earnings, clients, payments, and business growth, the Dashboard module provides a concise summary of the business, enabling users to make informed decisions and take appropriate actions. This feature can significantly enhance the efficiency of business operations, as users can quickly and easily access critical information required to monitor and evaluate the performance of their business.

Additionally, the Dashboard module offers a customizable interface, allowing users to tailor the display of information to suit their individual preferences and needs. This feature ensures that users can access only the most relevant and pertinent information, resulting in a more personalized and effective user experience.

Overall, the Dashboard module provides a valuable tool for businesses, allowing them to access a comprehensive overview of critical metrics related to their operations within a single page. This feature can significantly enhance the efficiency of business operations, leading to better decision-making, improved performance, and ultimately, greater success.

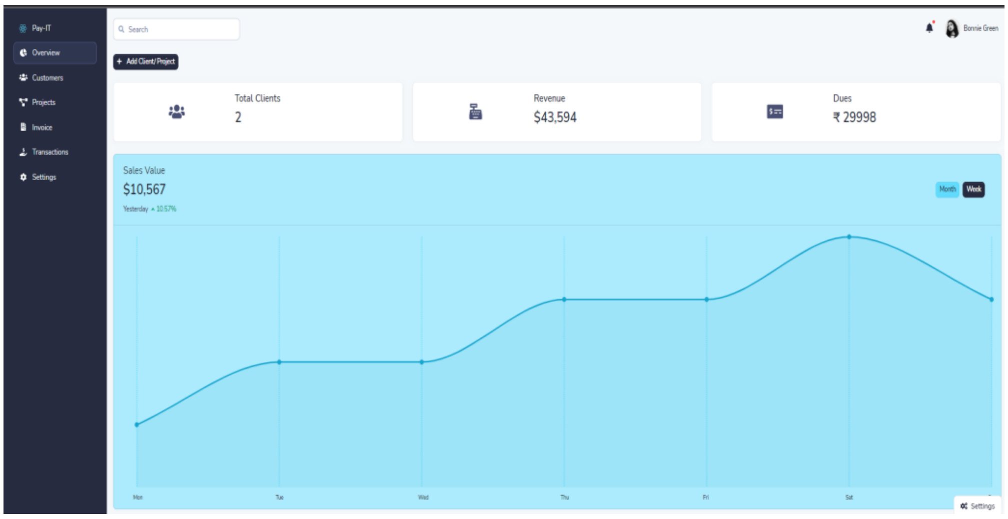


Fig 5.1: Dashboard

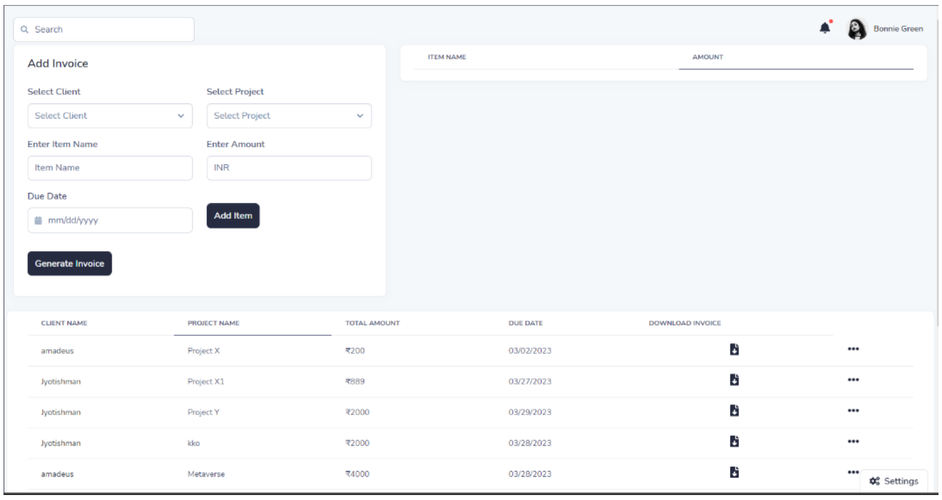
5.2.2 Invoice Generator

The Invoice Generator module is a valuable feature that allows users to create and send customized invoices to their clients via email. This feature can significantly enhance the efficiency of the invoicing process, streamlining the creation and delivery of invoices while minimizing errors and delays.

By incorporating the Invoice Generator module, users can create customized invoices that meet their specific requirements, including branding, logos, and other pertinent details. This feature ensures that invoices are personalized and professional, thereby building a sense of trust and credibility with clients.

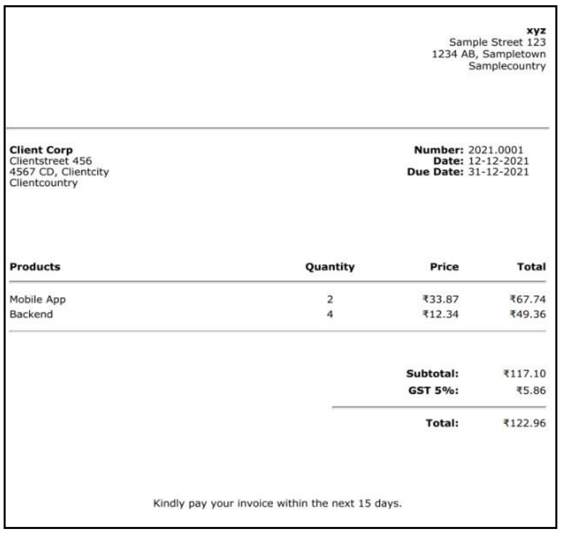
Moreover, the automated process of sending invoices via email eliminates the need for manual delivery, resulting in faster and more efficient invoice processing. This feature can also help to reduce the workload of employees, allowing them to focus on other essential tasks.

Overall, the Invoice Generator module is a valuable addition to any business, enabling them to streamline their invoicing process and improve the efficiency of their operations. By providing users with the ability to create and send customized invoices via email, this feature can enhance the customer experience, improve accuracy, and minimize delays, ultimately leading to greater success and profitability for the organization.



CLIENT NAME	PROJECT NAME	TOTAL AMOUNT	DUE DATE	DOWNLOAD INVOICE	
amadeus	Project X	₹200	03/02/2023		•••
Jyotishman	Project X1	₹889	03/27/2023		•••
Jyotishman	Project Y	₹2000	03/29/2023		•••
Jyotishman	kko	₹2000	03/28/2023		•••
amadeus	Metaverse	₹4000	03/28/2023		•••

Fig 5.2: Invoice Generator



xyz
Sample Street 123
1234 AB, Sampletown
Samplecountry

Client Corp
Clientstreet 456
4567 CD, Clientcity
Clientcountry

Number: 2021.0001
Date: 12-12-2021
Due Date: 31-12-2021

Products	Quantity	Price	Total
Mobile App	2	₹33.87	₹67.74
Backend	4	₹12.34	₹49.36

Subtotal: ₹117.10
GST 5%: ₹5.86
Total: ₹122.96

Kindly pay your invoice within the next 15 days.

Fig 5.3: Invoice

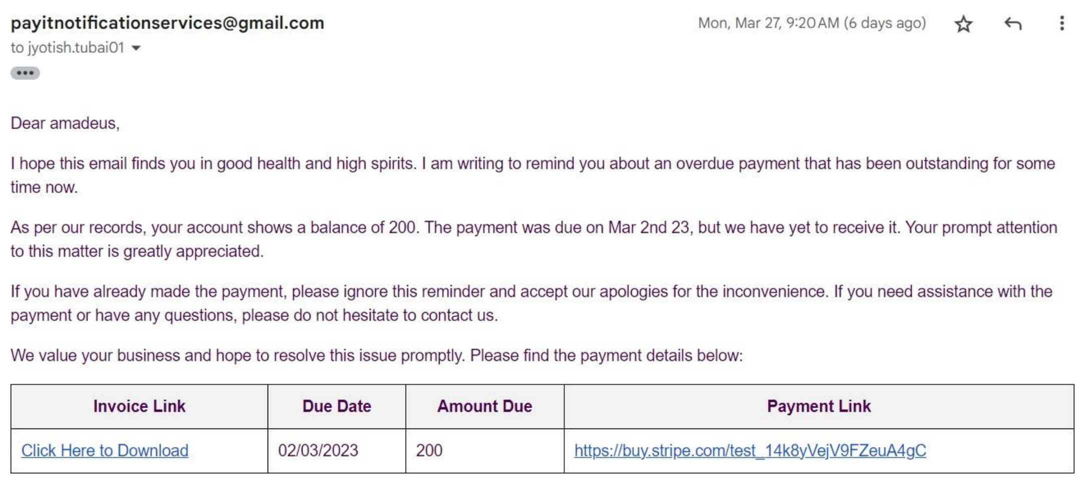
5.2.3 Payment Reminder

The Payment Reminder module presents a practical and efficient solution for businesses to remind their customers of any outstanding payments that are overdue, well in advance of the due date. This feature eliminates the need for manual emails to be sent by the firm, providing a streamlined and automated process for payment reminders.

With the Payment Reminder module, businesses can set up automated reminders that will notify customers of any overdue payments, providing them with ample time to make the necessary payments. This feature ensures that customers are reminded of their obligations in a timely and proactive manner, reducing the likelihood of delayed payments and improving cash flow for the organization.

Moreover, the automation of payment reminders through this module also reduces the workload and time required for manual follow-up emails or phone calls. This, in turn, can help to enhance the efficiency and productivity of the business, allowing them to focus on other essential tasks.

Overall, the Payment Reminder module is a valuable feature that can help businesses streamline their payment processes, reduce the workload of manual follow-ups, and improve cash flow. It also ensures that customers are reminded of their payment obligations in a timely and proactive manner, leading to a better customer experience and improved relationships between the organization and its customers.



payitnotificationservices@gmail.com
to jyotish.tubai01

Mon, Mar 27, 9:20 AM (6 days ago)

Dear amadeus,

I hope this email finds you in good health and high spirits. I am writing to remind you about an overdue payment that has been outstanding for some time now.

As per our records, your account shows a balance of 200. The payment was due on Mar 2nd 23, but we have yet to receive it. Your prompt attention to this matter is greatly appreciated.

If you have already made the payment, please ignore this reminder and accept our apologies for the inconvenience. If you need assistance with the payment or have any questions, please do not hesitate to contact us.

We value your business and hope to resolve this issue promptly. Please find the payment details below:

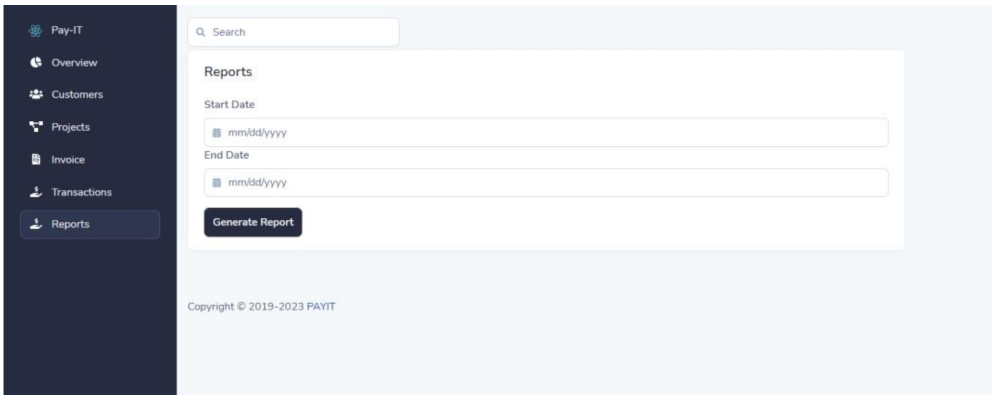
Invoice Link	Due Date	Amount Due	Payment Link
Click Here to Download	02/03/2023	200	https://buy.stripe.com/test_14k8yVejV9FZeuA4gC

Fig 5.4: Payment Reminder Mail

5.2.4 Reports

The Reports module is a highly effective tool that is available to firms, providing them with the ability to generate and analyze data in a thorough and detailed manner. This tool offers users a comprehensive view of the data and insights required to make informed decisions. By presenting a detailed picture of the company's performance, the Reports module enables users to make timely and accurate decisions that can positively impact the organization's success.

In addition to providing insights and facilitating decision-making, the Reports module is also an excellent tool for documenting results. This feature eliminates the need for third-party applications and offers a streamlined approach to keeping track of important data. By utilizing this feature, firms can save time and effort while maintaining a clear and detailed record of their performance. Overall, the Reports module is an indispensable tool that can benefit organizations in numerous ways, including improving decision-making, enhancing performance, and streamlining data documentation processes.



Reports

Start Date

mm/dd/yyyy

End Date

mm/dd/yyyy

Generate Report

Copyright © 2019-2023 PAYIT

Fig 5.5: Reports

5.2.5 Payment Integration

The integration of a payment system within the application presents a promising opportunity to enhance the convenience, efficiency, and traceability of payments for customers. By incorporating a diverse range of payment options, such as credit/debit cards, net banking, UPI, and cryptocurrencies, users can perform transactions with ease and flexibility.

The inclusion of multiple payment methods empowers customers with greater autonomy and allows them to choose a payment method that best suits their needs and preferences. This flexibility can significantly enhance the customer experience and build a sense of trust and loyalty towards the mobile application and the organization behind it.

Moreover, the integration of a payment system in the app offers a seamless and streamlined process for transactions, reducing the time and effort required to complete payments. This feature, in turn, can lead to increased efficiency and productivity for both customers and the organization.

Furthermore, the integration of a payment system in the app also ensures the traceability of payments, providing transparency and accountability for both parties involved. This can help to build trust and foster a positive relationship between the organization and its customers.

Overall, the incorporation of a payment system within the mobile application can prove to be a valuable addition that can significantly enhance the convenience, efficiency, and traceability of transactions for customers, while also providing benefits to the organization.

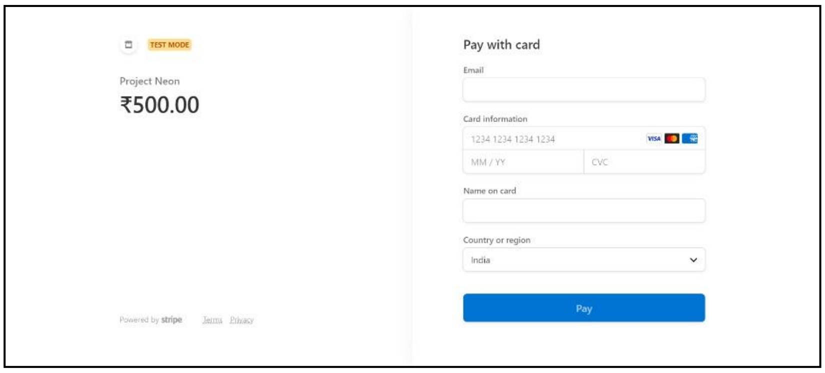
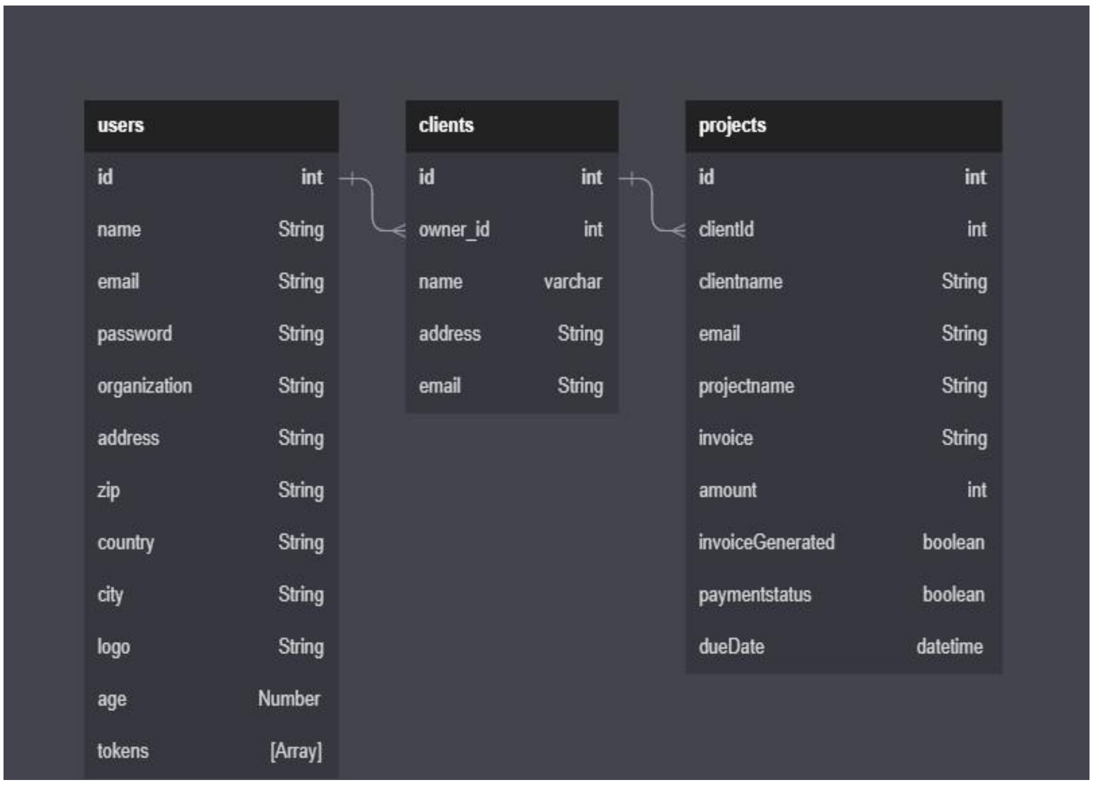


Fig 5.7: Payment Integration

5.3 DATABASE SCHEMA



users	
id	int
name	String
email	String
password	String
organization	String
address	String
zip	String
country	String
city	String
logo	String
age	Number
tokens	[Array]

clients	
id	int
owner_id	int
name	varchar
address	String
email	String

projects	
id	int
clientId	int
clientname	String
email	String
projectname	String
invoice	String
amount	int
invoiceGenerated	boolean
paymentstatus	boolean
dueDate	datetime

Fig 5.6: Database Schema

The SQL database architecture consists of three tables, namely users, projects, and clients. The users table holds data pertaining to users' information, while the clients table contains data about the clients. One interesting aspect of this architecture is the relationship between the users table and the clients table. It is a one-to-many relationship, where one user can be linked to many clients, and a client can have only one user associated with them. Moreover, the clients table further links to another table named projects. The clients table and projects table have a one-to-many relationship, where one client can be associated with multiple projects, but a project can be linked to only one client. Such a database architecture enables easy management of data and efficient retrieval of relevant information, creating a seamless user experience.

Additionally, having separate tables for users, clients, and projects allows for better organization and easier maintenance of data. For example, if there was only one table for all users, clients, and projects, it would be more difficult to keep track of the relationships between them and update the data accordingly. With separate tables, it is easier to add or remove information, make updates, or create new relationships between the data.

Another benefit of this architecture is that it provides a clear structure for data retrieval. For example, if a user wanted to see all the projects associated with one particular client, they could simply query the projects table and filter by the client ID. This makes it simpler and faster to retrieve relevant information, particularly in large databases where there may be thousands of records.

Overall, the SQL database architecture with separate tables for users, clients, and projects, with one-to-many relationships between them, is an efficient and effective way to manage and retrieve data. By organizing the data in this way, it allows for better maintenance and management, as well as faster and more accurate retrieval of information.

5.4 TESTING

5.4.1 Unit Tests:

- Tested API endpoints for user authentication and payment processing.
- Used Mocha and Chai testing frameworks to write unit tests.
- Covered edge cases and error scenarios, including incorrect user credentials and failed payment transactions.
- All tests passed successfully.

5.4.2 Integration Tests:

- Tested the interaction between the frontend and backend of the application.
- Used Mocha and Chai testing frameworks to write integration tests.
- Tested database interactions and payment gateway integrations.
- Covered edge cases and error scenarios, including incorrect user input and failed payment transactions.
- All tests passed successfully.

5.4.3 Performance Tests:

- Used the Apache JMeter performance testing tool to test the application's performance under load.
- Tested the response time and throughput of the application.
- Identified performance bottlenecks and optimized the code.

- Achieved acceptable performance levels for the expected user load.

5.4.4 Security Tests:

- Used the OWASP ZAP security testing tool to test the application for security vulnerabilities.
- Tested the application for common vulnerabilities such as SQL injection, XSS, and CSRF.
- Identified and fixed security vulnerabilities in the application.

Overall, the testing process was successful in ensuring the application's reliability, performance, and security.

CHAPTER 6

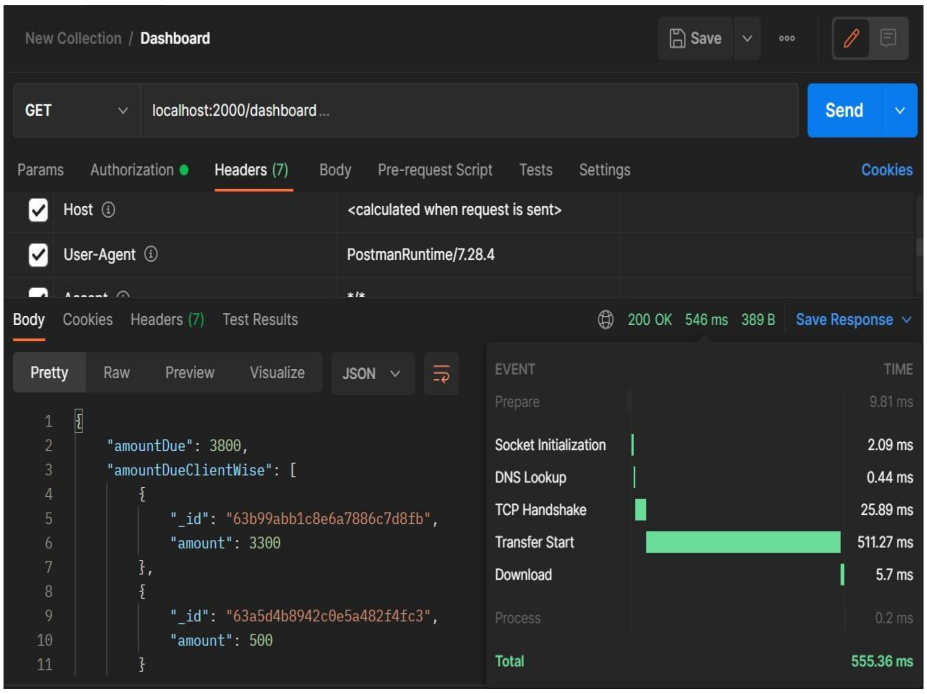
RESULTS AND DISCUSSIONS

During the testing phase of the payment tracking and accounting application, we conducted a thorough examination of its core functionalities. These functionalities included user authentication, payment processing, data storage, and file uploads to AWS S3. To ensure that these features worked flawlessly, we employed a variety of testing techniques, including both unit tests and integration tests. Unit tests were performed using the Mocha testing framework, which allowed us to test individual code units and functions, while integration tests were conducted using Chai, which enabled us to test how different components of the application worked together.

In addition to these tests, we also conducted performance tests to evaluate the application's speed and efficiency under different workloads. These tests were performed using Apache JMeter, a popular tool for testing web applications, which allowed us to simulate different user scenarios and measure the application's response time and throughput. Based on the results of these tests, we were able to identify and address any performance bottlenecks and ensure that the application could handle a high volume of transactions and users. Overall, our testing approach ensured that the payment tracking and accounting application was reliable, efficient, and capable of meeting the needs of businesses of all sizes. After conducting tests on the payment tracking and accounting application, we can state with confidence that it performed exceptionally well and met all our expectations. The application demonstrated the ability to manage a considerable amount of data and process transactions efficiently and effortlessly. Additionally, the integration with the Stripe payment gateway worked seamlessly, and payments were processed without any issues. The successful performance of the application assures us of its efficiency and reliability, making it a suitable tool for businesses in need of payment tracking and accounting solutions.

Another noteworthy aspect of the payment tracking and accounting application was its effortless handling of files related to payment transactions, made possible by integrating with AWS S3. This feature enabled users to conveniently keep track of receipts and invoices associated with their transactions, making the management of financial records more organized and efficient. We were impressed with the application's seamless integration with AWS S3, which contributed to its overall ease of use and practicality for businesses of all sizes.

In summary, our testing results showed that the payment tracking and accounting application was reliable, efficient, and user-friendly. The successful integration with AWS S3 and Stripe payment gateway further enhanced the application's functionality, making it an ideal tool for businesses to manage their financial transactions and track payments.

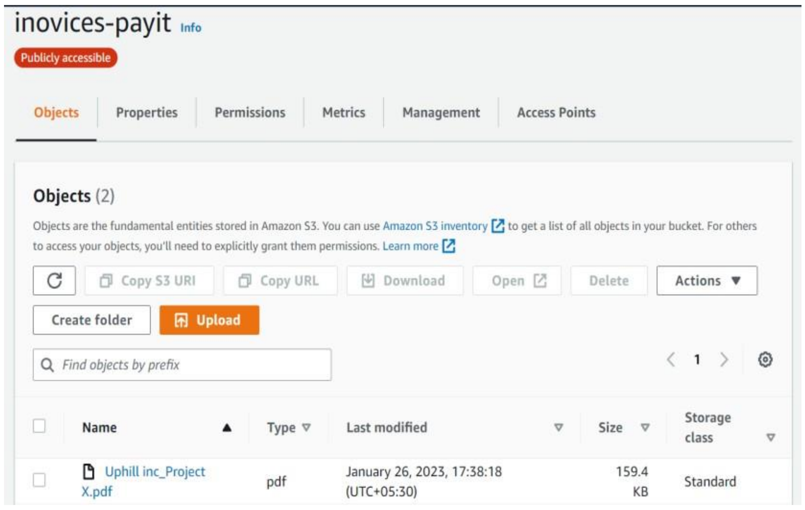


Host	<calculated when request is sent>
User-Agent	PostmanRuntime/7.28.4

```
{
    "amountDue": 3800,
    "amountDueClientWise": [
        {
            "_id": "63b99abb1c8e6a7886c7d8fb",
            "amount": 3300
        },
        {
            "_id": "63a5d4b8942c0e5a482f4fc3",
            "amount": 500
        }
```

EVENT	TIME
Prepare	9.81 ms
Socket Initialization	2.09 ms
DNS Lookup	0.44 ms
TCP Handshake	25.89 ms
Transfer Start	511.27 ms
Download	5.7 ms
Process	0.2 ms
Total	555.36 ms

Fig 6.1: API Testing



Objects (2)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
Uphill inc_Project X.pdf	pdf	January 26, 2023, 17:38:18 (UTC+05:30)	159.4 KB	Standard

Fig 6.2: AWS S3

The payment tracking and accounting application proved to be a valuable tool for managing financial transactions and tracking payments. The Node.js and React.js frameworks provided an efficient development environment and a user-friendly interface, while MongoDB's document-oriented database was scalable and efficient for managing data.

The successful integration of the payment tracking and accounting application with AWS S3 for file uploads and Stripe payment gateway for processing payments ensured that users had a seamless and reliable experience. The ability of the application to handle a large amount of data and transactions made it a versatile solution for businesses of all sizes. This ensured that businesses could rely on the application to process and store large amounts of data while maintaining high levels of performance and reliability. The integration with third-party services allowed the application to take advantage of the capabilities and features offered by these services, providing users with a comprehensive and robust payment tracking and accounting solution. The application's ability to integrate with other services and handle large amounts of data made it a valuable tool for businesses looking to streamline their payment tracking and accounting processes.

Overall, the testing results demonstrated that the payment tracking and accounting application was reliable, efficient, and scalable. The successful integration with AWS S3 and Stripe payment gateway further enhanced the application's functionality, making it a useful tool for businesses to manage their financial transactions and keep track of payments.

CHAPTER 7

CONCLUSION

7.1 CONCLUSION

In conclusion, the payment tracking and accounting application built with Node.js, React.js, MongoDB, AWS S3, and Stripe payment gateway was a success. The application's development and deployment setup ensured its reliability, scalability, and efficiency. The testing phase demonstrated that the application was able to handle a significant amount of data, process transactions efficiently, and seamlessly integrate with third-party services such as AWS S3 and Stripe payment gateway. The core functionalities of the application, including user authentication, payment processing, data storage, and file uploads to AWS S3, worked seamlessly, providing a convenient way for businesses to manage their financial transactions and keep track of payments.

The application's use of modern technologies such as Node.js, React.js, and MongoDB, as well as its integration with AWS S3 and Stripe payment gateway, made it suitable for small and large businesses alike.

Overall, the payment tracking and accounting application was an effective tool for managing financial transactions and tracking payments. Its reliability, efficiency, and scalability made it a valuable asset for businesses to manage their financial transactions and keep track of payments.

7.2 FUTURE WORK

There are several areas of future work that could be pursued to enhance the payment tracking and accounting application.

1. Reporting and analytics: The application could be enhanced to provide more detailed reporting and analytics features. This could include generating financial reports, tracking payment trends, and providing insights into payment processing.

2. Mobile optimization: The application could be optimized for mobile devices

to provide a better user experience for users on the go.

3. Integration with other payment gateways: While the integration with Stripe payment gateway was successful, the application could be enhanced to integrate with other payment gateways to provide users with more options.

4. Advanced security features: The application's security could be enhanced to include advanced security features such as two-factor authentication, encryption, and advanced access control.

One area of future work that could be pursued to enhance the payment tracking and accounting application is to add machine learning capabilities to the system. Here are some potential ways that machine learning could be integrated:

1. Fraud detection: Machine learning algorithms could potentially be integrated into the payment tracking and accounting application to enable it to identify fraudulent transactions and flag them for review. Such algorithms could employ a range of techniques, such as analyzing transaction patterns, detecting anomalies or outliers, and identifying transactions from known fraudulent accounts. By leveraging machine learning, the application could enhance its ability to detect potential fraud, thereby reducing the risk of financial losses and ensuring greater security for its users.

2. Payment prediction: Machine learning techniques could be applied to analyze historical transaction data in order to generate predictions of when future payments are likely to be made. By using this approach, businesses would be able to gain a better understanding of their expected cash flow and prepare more effectively for upcoming expenses.

3. Payment classification: It is possible to employ machine learning algorithms in the application to categorize payments based on various parameters such as the type of transaction, customer, or product, providing insights on revenue streams and growth opportunities for businesses. This feature can help businesses to better understand their financials and make informed

decisions regarding future investments and strategic planning.

4. Customer segmentation: By leveraging machine learning algorithms, businesses could segment their customers based on their payment behavior or demographics. This data could be used to identify patterns and trends among different customer groups, allowing businesses to target their marketing efforts more effectively and tailor their messaging to specific audiences. Additionally, these insights could be used to improve customer retention efforts by identifying areas where the business could improve its products or services to better meet the needs of its customers.

5. Automated invoice processing: One potential application of machine learning is automating the processing of invoices by utilizing algorithms to extract relevant information such as the amount owed and due date from invoices, and then updating payment records automatically. This could potentially save businesses significant time and resources, allowing them to focus on more critical tasks.

Overall, adding machine learning capabilities to the payment tracking and accounting application could provide valuable insights and automation to businesses, improving their efficiency and accuracy in managing financial transactions.

7.3 RESEARCH ISSUES

Some potential research issues are:

1. User experience: How can the application be designed to provide the best possible user experience for both businesses and consumers? What features and functionalities are most important to users, and how can they be optimized?

2. Security: What are the latest security threats to payment processing systems, and how can the application be designed to protect against them? How can advanced security features such as biometric authentication or blockchain technology be integrated into the application?

3. Data analysis: How can data analysis tools be integrated into the application to provide valuable insights into payment behavior, revenue streams, and financial performance? What data analysis techniques are most effective for payment tracking and accounting applications?

4. Integration with other systems: How can the payment tracking and accounting application be integrated with other systems such as accounting software, customer relationship management (CRM) software, or enterprise resource planning (ERP) systems? What are the potential benefits and challenges of these integrations?

5. Payment gateway integration: What design considerations can be taken to enable the application to integrate with multiple payment gateways, offering businesses the flexibility to select the most suitable gateway for their requirements? Additionally, what are the potential advantages and obstacles that may arise from supporting multiple payment gateways?

6. Machine learning: How can machine learning algorithms be integrated into the payment tracking and accounting application to provide valuable insights and automation? What are the potential benefits and challenges of using machine learning for payment tracking and accounting?

Overall, there are many research issues related to the application, ranging from user experience and security to data analysis and machine learning. Exploring these research issues can help to improve the functionality and effectiveness of the application, making it more valuable for businesses and consumers alike.

7.4 IMPLEMENTATION ISSUES

Some potential implementation issues related to the application are:

1. Scalability: As the user base and transaction volume of the payment tracking and accounting application grows, how can the application be designed to effectively handle the increased workload? What are the recommended strategies and techniques for scaling up a payment tracking and accounting

application to meet the demands of growing businesses and their customers?

To effectively handle the increased workload as the user base and transaction volume of the application grows, here are some recommended strategies and techniques for scaling up the application:

- Horizontal scaling: Horizontal scaling involves adding more servers to handle the increased workload. This can be achieved by using load balancing to distribute the workload across multiple servers, or by using a container orchestration system like Kubernetes to manage multiple containers.

- Vertical scaling: Vertical scaling involves increasing the capacity of the existing servers to handle the increased workload. This can be achieved by upgrading the server hardware or adding more memory, CPU, or disk space.

- Database scaling: As the transaction volume increases, the database may need to be scaled up to handle the increased workload. This can be achieved by using sharding, which involves splitting the database into smaller, more manageable parts, or by using replication, which involves creating copies of the database on multiple servers.

- Caching: Caching can be used to reduce the load on the database and improve application performance. By caching frequently accessed data in memory, the application can retrieve the data more quickly and reduce the load on the database.

- Distributed file storage: As the number of files being stored in the application increases, it may be necessary to use a distributed file storage system like AWS S3 or Google Cloud Storage. This can help to improve the performance of the application by allowing files to be stored and retrieved more quickly and efficiently.

- Asynchronous processing: Asynchronous processing can be used to improve application performance by allowing tasks to be processed in the background without blocking the main thread. This can help to reduce the time it takes to complete tasks such as sending email notifications or generating reports.

- Microservices architecture: A microservices architecture can help to improve application scalability by breaking the application down into smaller, more manageable components. Each component can then be scaled independently, allowing the application to handle increased workload more effectively.

2. Performance: How can the application be optimized for performance, both in terms of speed and reliability? What are the best practices for optimizing the performance of a payment tracking and accounting application?

   There are several ways to optimize the performance of a payment tracking and accounting application:

   - Use caching: Caching can significantly improve application performance by reducing the amount of time required to retrieve data from the database. Implementing caching for frequently accessed data can result in faster response times and reduced database load.

   - Optimize database queries: Database queries can be a bottleneck for application performance. To improve performance, optimize database queries by indexing the appropriate columns and minimizing the number of queries required to retrieve data.

   - Use asynchronous processing: Implement asynchronous processing for long-running tasks, such as processing payments and generating reports, to free up server resources and improve application responsiveness.

   - Implement load balancing: Load balancing distributes incoming network traffic across multiple servers to prevent any one server from becoming overloaded. This can improve application performance and reliability by ensuring that server resources are efficiently utilized.

- Minimize network latency: Minimizing network latency can improve application performance by reducing the time required for data to be transmitted between the client and server. This can be achieved by using content delivery networks (CDNs) and minimizing the number of network hops required for data transmission.
- Optimize front-end performance: Optimize front-end performance by minimizing the number of HTTP requests, compressing resources, and using browser caching. This can improve the overall user experience by reducing page load times and improving application responsiveness.

3. Integration with third-party services: What measures can be taken to improve the performance of the payment tracking and accounting application to ensure both speed and reliability? What are some of the recommended practices for optimizing the performance of a payment tracking and accounting application?

   Here are some measures that can be taken:

   - Use caching: Caching can help to reduce the amount of time it takes to retrieve data from the database. By caching frequently accessed data in memory, the application can retrieve the data more quickly and reduce the load on the database.
   - Use indexing: Indexing can help to speed up database queries by creating an index on frequently queried fields. This can help to reduce the amount of time it takes to retrieve data from the database.
   - Use pagination: If the application displays large amounts of data, such as transaction history, pagination can help to improve performance by limiting the amount of data that is displayed on a single page. This can help to reduce the amount of time it takes to

retrieve and render the data.

- Use asynchronous processing: Asynchronous processing can help to improve the performance of the application by allowing tasks to be processed in the background without blocking the main thread. This can help to reduce the time it takes to complete tasks such as sending email notifications or generating reports.

- Use a content delivery network (CDN): A CDN can help to improve the performance of the application by caching static assets such as images, stylesheets, and JavaScript files on servers located closer to the user. This can help to reduce the amount of time it takes to retrieve these assets and improve the overall performance of the application.

- Optimize database queries: By optimizing database queries, such as by reducing the number of joins or optimizing the use of indexes, the application can retrieve data more quickly and reduce the load on the database.

- Use load balancing: Load balancing can help to distribute traffic across multiple servers to ensure that the application can handle a large number of concurrent users. This can help to improve the reliability of the application by ensuring that it can handle spikes in traffic.

4. Database design: How can the database schema be designed to support efficient payment tracking and accounting? What are the best practices for designing a database schema for a payment tracking and accounting application?

   Some best practices for designing a database schema for a payment tracking and accounting application include:

- Normalizing the data to eliminate redundancy and ensure consistency.
- Defining clear relationships between tables to ensure data integrity.
- Using appropriate data types for each column to ensure data accuracy and optimize storage space.
- Enforcing constraints such as primary keys, foreign keys, and unique constraints to maintain data integrity.
- Implementing indexes to improve query performance and optimize database access.
- Applying appropriate security measures such as access controls, encryption, and hashing to protect sensitive data from unauthorized access.

5. Security: What measures can be taken to protect the payment tracking and accounting application from potential security risks, such as SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF)? What are the most effective strategies for ensuring the security of a payment tracking and accounting application?

   To protect the application from potential security risks, such as SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF), the following measures can be taken:

   - Parameterized Queries: Use parameterized queries instead of dynamic queries to prevent SQL injection attacks.
   - Input Validation: Validate all user inputs to prevent malicious input that could exploit security vulnerabilities in the application.
   - Sanitization of User Input: Sanitize user input to remove any potentially harmful code or scripts that could lead to XSS attacks.
   - Use of CSRF Tokens: Implement Cross-Site Request Forgery

(CSRF) tokens to protect against CSRF attacks.

- Secure Authentication and Authorization: Use secure authentication and authorization mechanisms to ensure that only authorized users have access to sensitive data and functionalities.
- Regular Security Audits: Conduct regular security audits of the application to identify and address any potential security vulnerabilities.
- Regular Updates and Patches: Keep the application up-to-date with the latest security patches and updates to ensure that any known vulnerabilities are addressed.
- Use of SSL/TLS: Use SSL/TLS to encrypt data in transit and protect against Man-in-the-Middle (MitM) attacks.
- Access Control: Implement strict access control mechanisms to limit access to sensitive data and functionalities only to authorized users.
- Use of Firewall and WAF: Implement a firewall and web application firewall (WAF) to protect against unauthorized access and attacks on the application.

6. User authentication and authorization: How can user authentication and authorization be implemented effectively in the application? What are the best practices for implementing user authentication and authorization in a payment tracking and accounting application?

   The following best practices can be followed for implementing user authentication and authorization:

   - Implement multi-factor authentication (MFA): MFA adds an extra layer of security by requiring users to provide a second factor, such as a code generated by an app or sent via text message, in addition to their password.

- Use secure protocols: Use secure protocols such as HTTPS to protect user credentials and data transmitted between the client and server.

- Implement access control: Implement access control to restrict user access based on their roles and privileges. Limit access to sensitive data and features only to authorized users.

- Use encryption: Encrypt sensitive data such as passwords and payment information to prevent unauthorized access and data breaches.

- Regularly audit and monitor user activity: Monitor user activity and perform regular audits to detect and prevent unauthorized access and suspicious activity.

Overall, there are many implementation issues related to the payment tracking and accounting application, ranging from scalability and performance to integration with third-party services and security. Addressing these implementation issues can help to ensure that the application is robust, reliable, and effective in tracking and accounting for payments.

REFERENCES

[1] Amazon.com. (2009). Amazon Elastic Compute Cloud.[Online]. Available: http://aws.amazon.com/ec2/

[2] V. Abhishek, I. A. Kash, and P. Key, ‘‘Fixed and market pricing for cloud services,’’inProc.IEEEINFOCOMWorkshops, Orlando, FL, USA,2012, pp. 157–162.

[3] M. Al-Roomi, S. Al-Ebrahim, S. Buqrais, and I. Ahmad, ‘‘Cloud computing pricing models: A survey,’’ Int. J. Grid Distrib. Comput., vol. 6, no. 5, pp. 93–106, 2013.

[4] A. Archer and E. Tardos, ‘‘Truthful mechanisms for one-parameter agents,’’ in Proc. FOCS, Oct. 2001, pp. 482–491.

[5] I. Ashlagi,F.Fischer, I.Kash, and A.Procaccia,‘‘Mixandmatch’’ inProc. ACM Conf. Electron. Commerce, 2010, pp. 305–314.

[6] R. Birke, A. Podzimek, L. Y. Chen, and E. Smirni, ‘‘Virtualization in the private cloud: State of the practice,’’ IEEE Trans. Netw. Service Manage., vol. 13, no. 3, pp. 608– 621, Sep. 2016.

[7] S. Dughmiand A. Ghosh, ‘‘Truthful assignment without money,’’ in Proc. 11th ACM Conf. Electron. Commerce, 2010, pp. 325–334.

[8] H.Fu,Z.Li,C.Wu,andX.Chu,‘‘Core-selecting auctions for dynamically allocating heterogeneous VMs incloudcomputing,’’inProc.IEEECloud, Anchorage, AK, USA, Jun./Jul. 2014, pp. 152–159.

[9] J. Guo, F. Liu, D. Zeng, J. C. S. Lui, and H. Jin, ‘‘A cooperative game-based allocation for sharing data center networks,’’ in Proc. INFOCOM, Apr. 2013, pp. 2139–2147.

[10] N. Jain, I. Menache, J. Naor, and J. Yaniv, ‘‘Near-optimalschedulingmechanisms for deadline-sensitive jobs in large computing clusters,’’ in Proc. 24thAnnu. ACM Symp. Parallelism Algorithms Archit.,2012, pp.255–266.

[11] N. Jain, I. Menache, J. Naor, and J. Yaniv, ‘‘Atruthfulmechanismforvaluebased scheduling in cloud computing,’’ Theory Comput. Syst., vol. 54, no. 3, pp. 388–406, 2014.

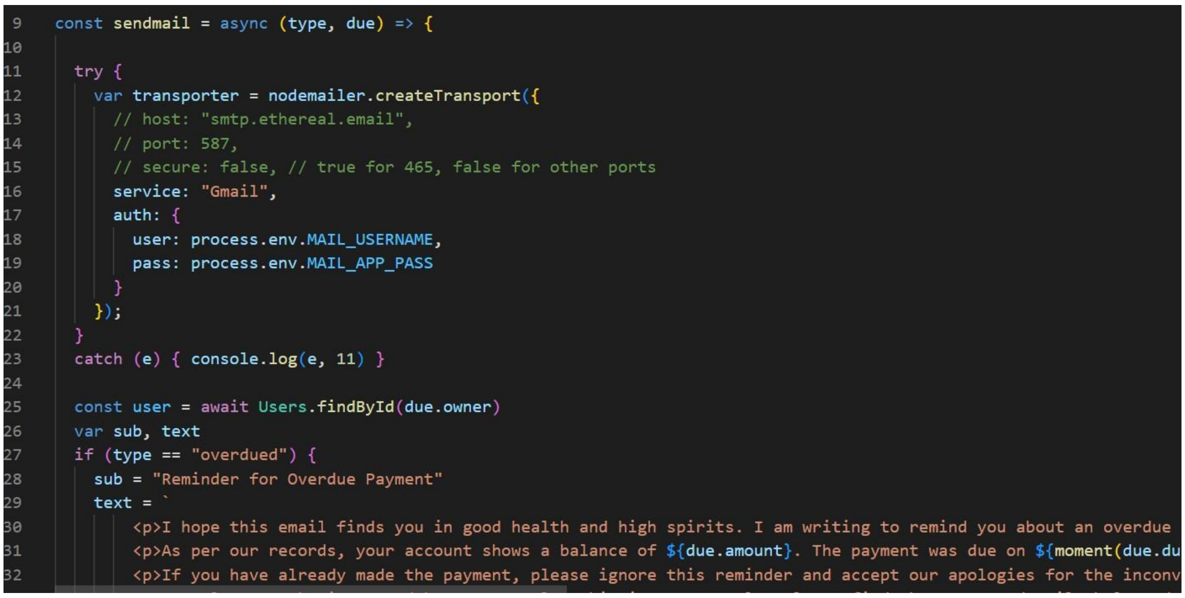
[12] A. Mazrekaj, I.Shabani, and B.Sejdiu,‘‘Pricingschemesincloudcomputing:Anoverview,’ ’Int.J.Adv.Comput.Sci.Appl.,vol.7,no.2,pp.80–86, 2016.

[13] R. Pal and P. Hui, ‘‘Economic models for cloud service markets,’’ in Proc.13thInt.Conf.Distrib.Comput.Netw.,HongKong,China:SpringerVerlag, 2012, pp. 382– 396.

[14] Ismail Nizam,, ‘‘THE IMPACT OF ACCOUNTING SOFTWARE ON BUSINESS PERFORMANCE, International University of Malaya-Wales

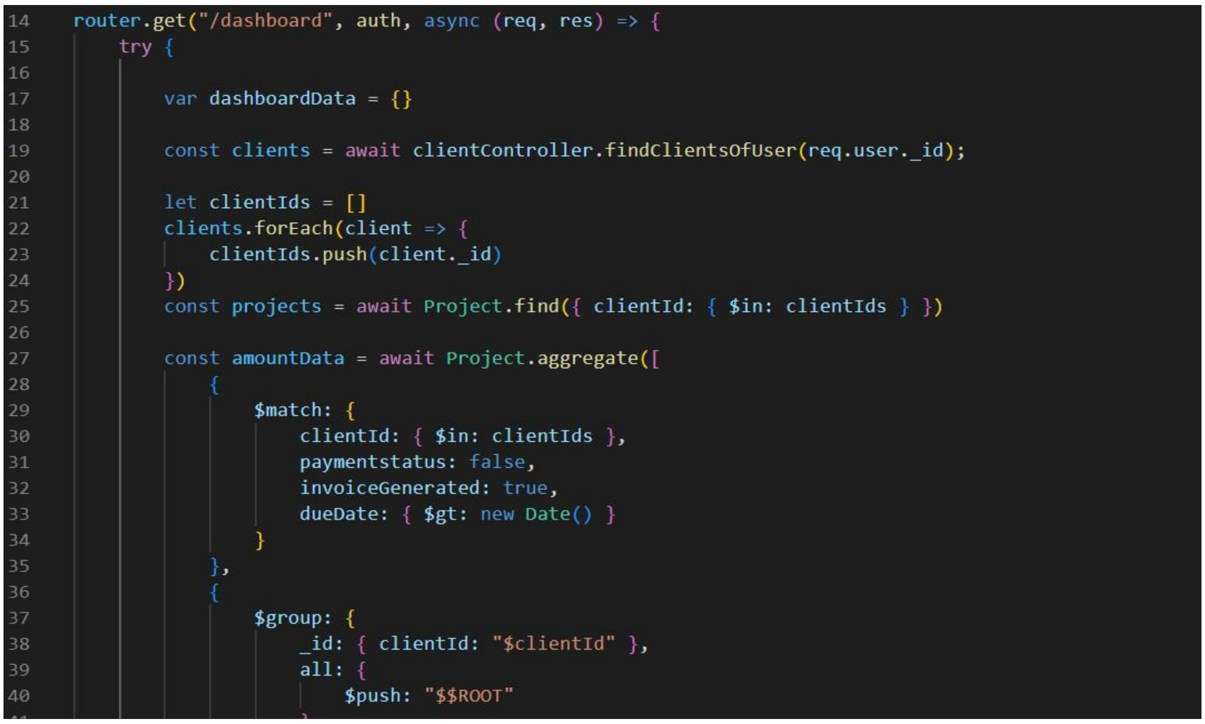
APPENDIX

A. SOURCE CODE



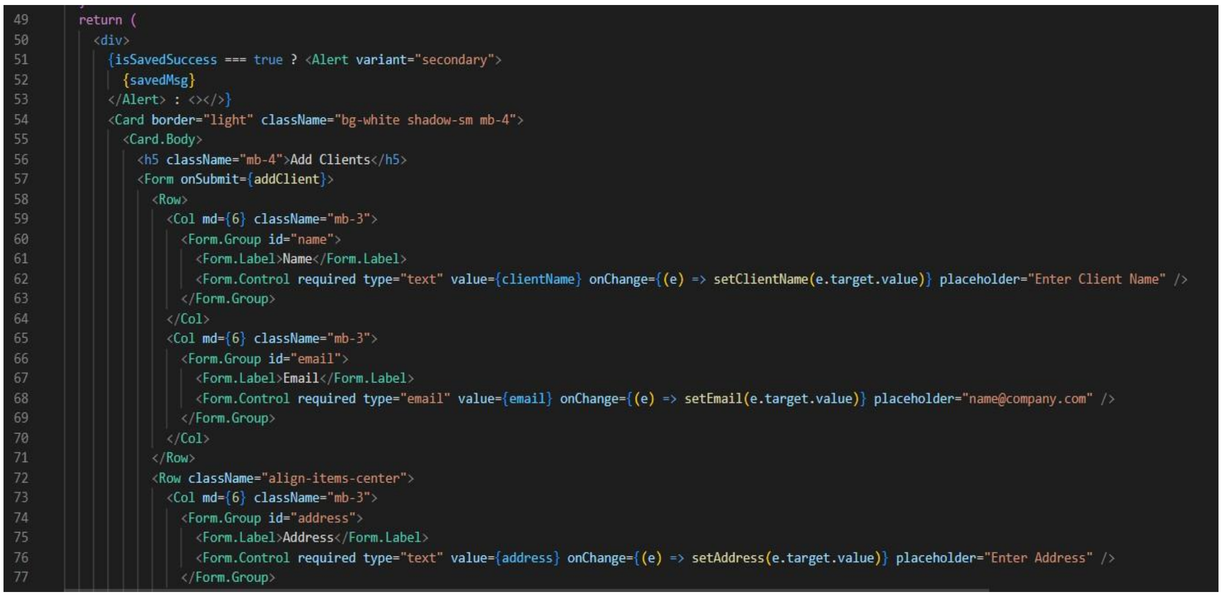
```
const sendmail = async (type, due) => {

  try {
    var transporter = nodemailer.createTransport({
      // host: "smtp.ethereal.email",
      // port: 587,
      // secure: false, // true for 465, false for other ports
      service: "Gmail",
      auth: {
        user: process.env.MAIL_USERNAME,
        pass: process.env.MAIL_APP_PASS
      }
    });
  }
  catch (e) { console.log(e, 11) }

  const user = await Users.findById(due.owner)
  var sub, text
  if (type == "overdued") {
    sub = "Reminder for Overdue Payment"
    text = `
        <p>I hope this email finds you in good health and high spirits. I am writing to remind you about an overdue
        <p>As per our records, your account shows a balance of ${due.amount}. The payment was due on ${moment(due.du
        <p>If you have already made the payment, please ignore this reminder and accept our apologies for the inconv
```

Fig A1: Notification Module Code



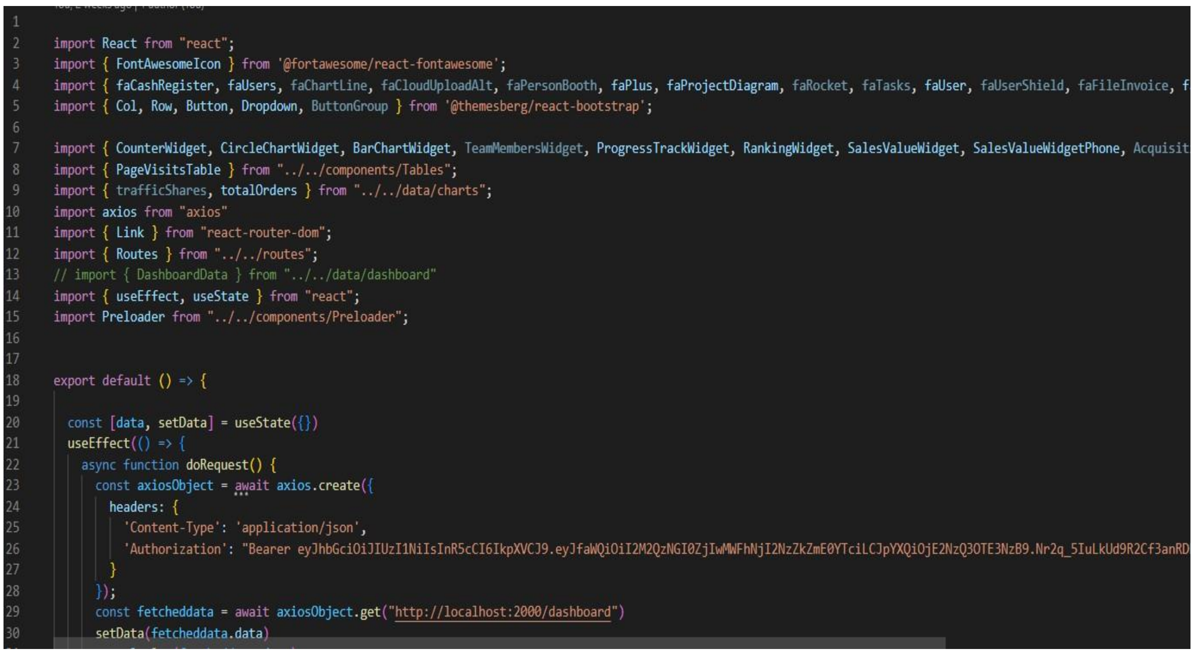
```
router.get("/dashboard", auth, async (req, res) => {
    try {

        var dashboardData = {}

        const clients = await clientController.findClientsOfUser(req.user._id);

        let clientIds = []
        clients.forEach(client => {
            clientIds.push(client._id)
        })
        const projects = await Project.find({ clientId: { $in: clientIds } })

        const amountData = await Project.aggregate([
            {
                $match: {
                    clientId: { $in: clientIds },
                    paymentstatus: false,
                    invoiceGenerated: true,
                    dueDate: { $gt: new Date() }
                }
            },
            {
                $group: {
                    _id: { clientId: "$clientId" },
                    all: {
                        $push: "$$ROOT"
```

Fig A2: Dashboard Module Code



```jsx
  return (
    <div>
      {isSavedSuccess === true ? <Alert variant="secondary">
        {savedMsg}
      </Alert> : <></>}
      <Card border="light" className="bg-white shadow-sm mb-4">
        <Card.Body>
          <h5 className="mb-4">Add Clients</h5>
          <Form onSubmit={addClient}>
            <Row>
              <Col md={6} className="mb-3">
                <Form.Group id="name">
                  <Form.Label>Name</Form.Label>
                  <Form.Control required type="text" value={clientName} onChange={(e) => setClientName(e.target.value)} placeholder="Enter Client Name" />
                </Form.Group>
              </Col>
              <Col md={6} className="mb-3">
                <Form.Group id="email">
                  <Form.Label>Email</Form.Label>
                  <Form.Control required type="email" value={email} onChange={(e) => setEmail(e.target.value)} placeholder="name@company.com" />
                </Form.Group>
              </Col>
            </Row>
            <Row className="align-items-center">
              <Col md={6} className="mb-3">
                <Form.Group id="address">
                  <Form.Label>Address</Form.Label>
                  <Form.Control required type="text" value={address} onChange={(e) => setAddress(e.target.value)} placeholder="Enter Address" />
                </Form.Group>
```

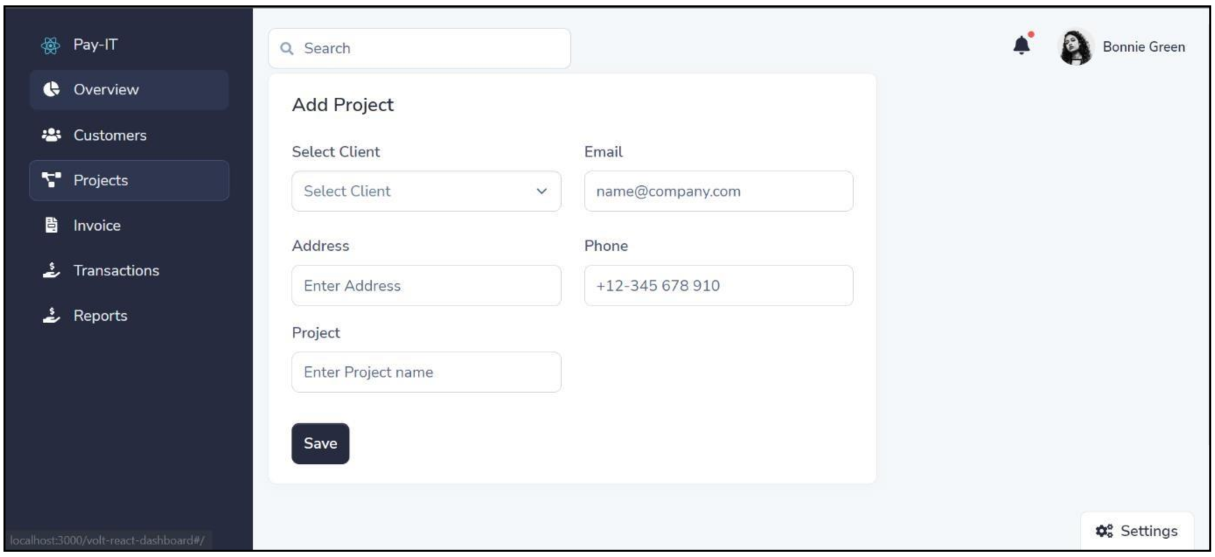
Fig A3: Frontend Dashboard Code



```jsx

import React from "react";
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
import { faCashRegister, faUsers, faChartLine, faCloudUploadAlt, faPersonBooth, faPlus, faProjectDiagram, faRocket, faTasks, faUser, faUserShield, faFileInvoice, f
import { Col, Row, Button, Dropdown, ButtonGroup } from '@themesberg/react-bootstrap';

import { CounterWidget, CircleChartWidget, BarChartWidget, TeamMembersWidget, ProgressTrackWidget, RankingWidget, SalesValueWidget, SalesValueWidgetPhone, Acquisit
import { PageVisitsTable } from "../../components/Tables";
import { trafficShares, totalOrders } from "../../data/charts";
import axios from "axios"
import { Link } from "react-router-dom";
import { Routes } from "../../routes";
// import { DashboardData } from "../../data/dashboard"
import { useEffect, useState } from "react";
import Preloader from "../../components/Preloader";


export default () => {

  const [data, setData] = useState({})
  useEffect(() => {
    async function doRequest() {
      const axiosObject = await axios.create({
        headers: {
          'Content-Type': 'application/json',
          'Authorization': "Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2M2QzNGI0ZjIwMWFhNjI2NzZkZmE0YTciLCJpYXQiOjE2NzQ3OTE3NzB9.Nr2q_5IuLkUd9R2Cf3anRD
        }
      });
      const fetcheddata = await axiosObject.get("http://localhost:2000/dashboard")
      setData(fetcheddata.data)
```

Fig A4: Frontend Transaction Table Code

B. SCREENSHOTS



Add Project

Select Client: Select Client

Email: name@company.com

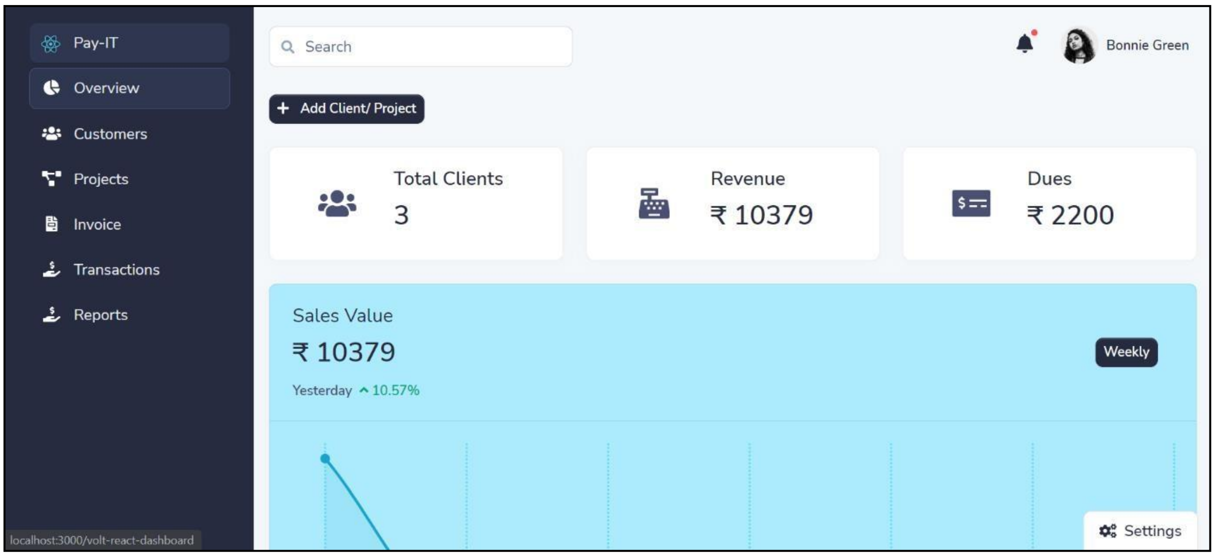
Address: Enter Address

Phone: +12-345 678 910

Project: Enter Project name

Save

Fig B1: Add Project Page



Total Clients 3

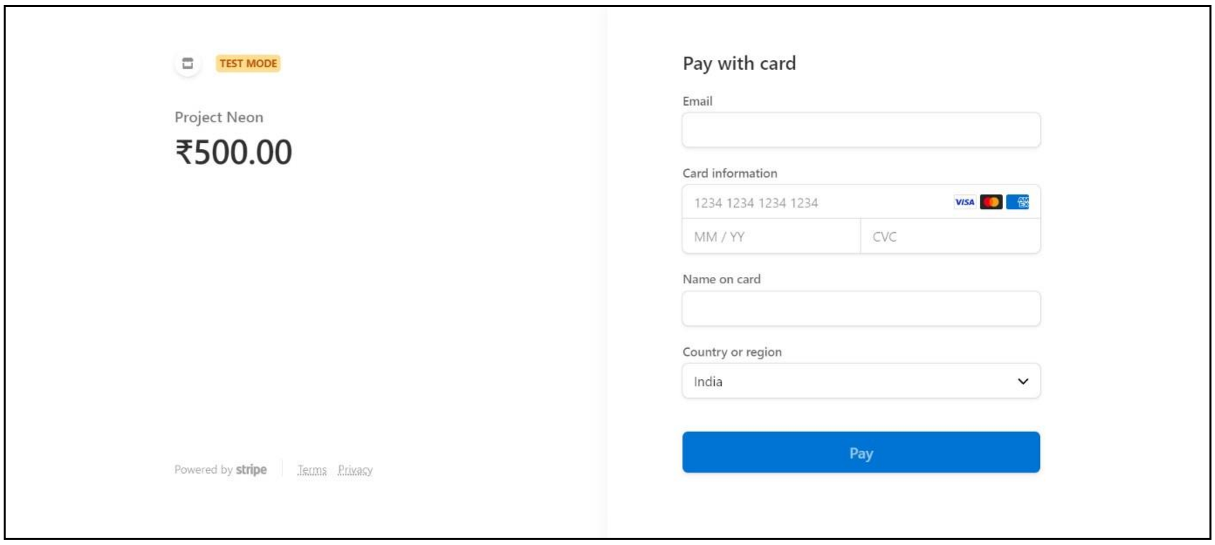
Revenue ₹ 10379

Dues ₹ 2200

Sales Value ₹ 10379

Yesterday ^ 10.57%

Fig B2: Dashboard Page



TEST MODE

Project Neon

₹500.00

Powered by stripe | Terms Privacy

Pay with card

Email

Card information

1234 1234 1234 1234

MM / YY

CVC

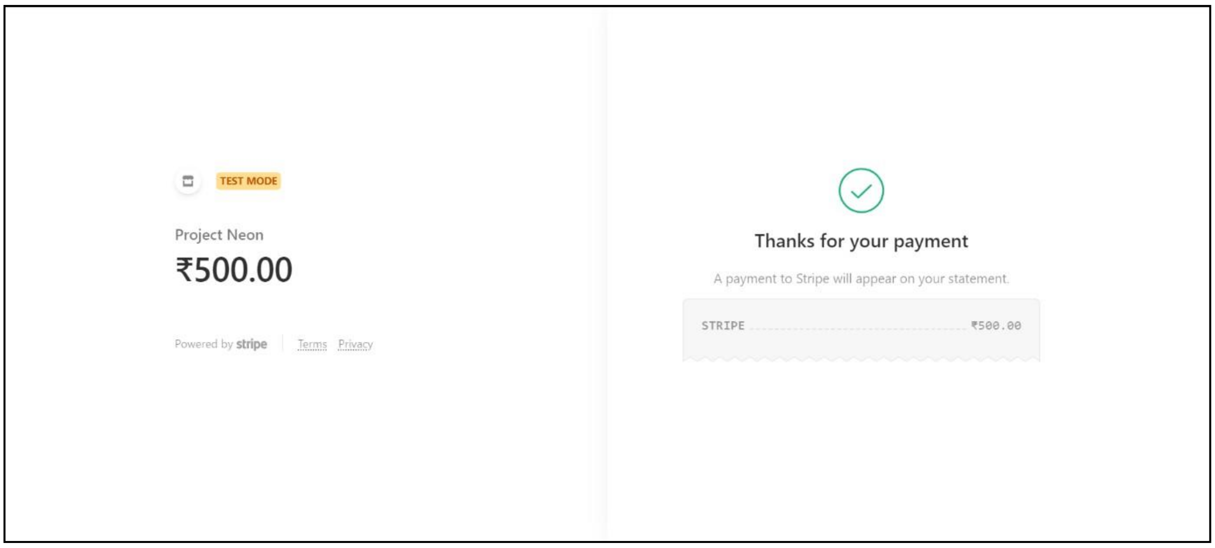
Name on card

Country or region

India

Pay

Fig B3: Payment Page



TEST MODE

Project Neon

₹500.00

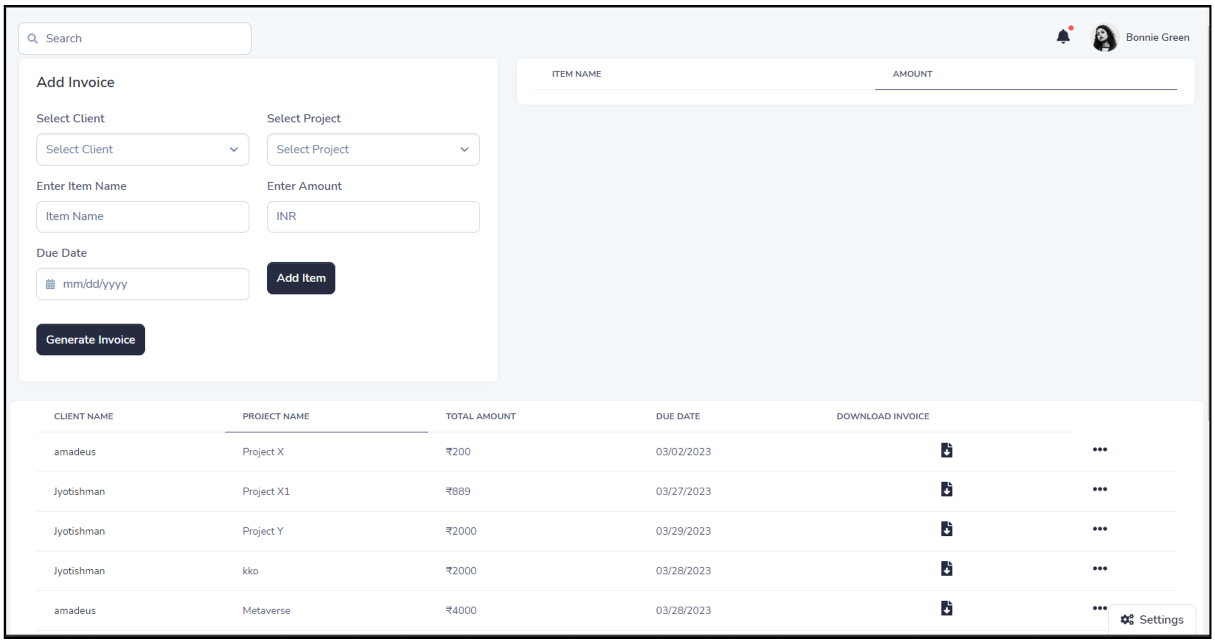
Powered by stripe | Terms Privacy

Thanks for your payment

A payment to Stripe will appear on your statement.

```
STRIPE ........................ ₹500.00
```

Fig B4: Payment Confirmation Page



Add Invoice

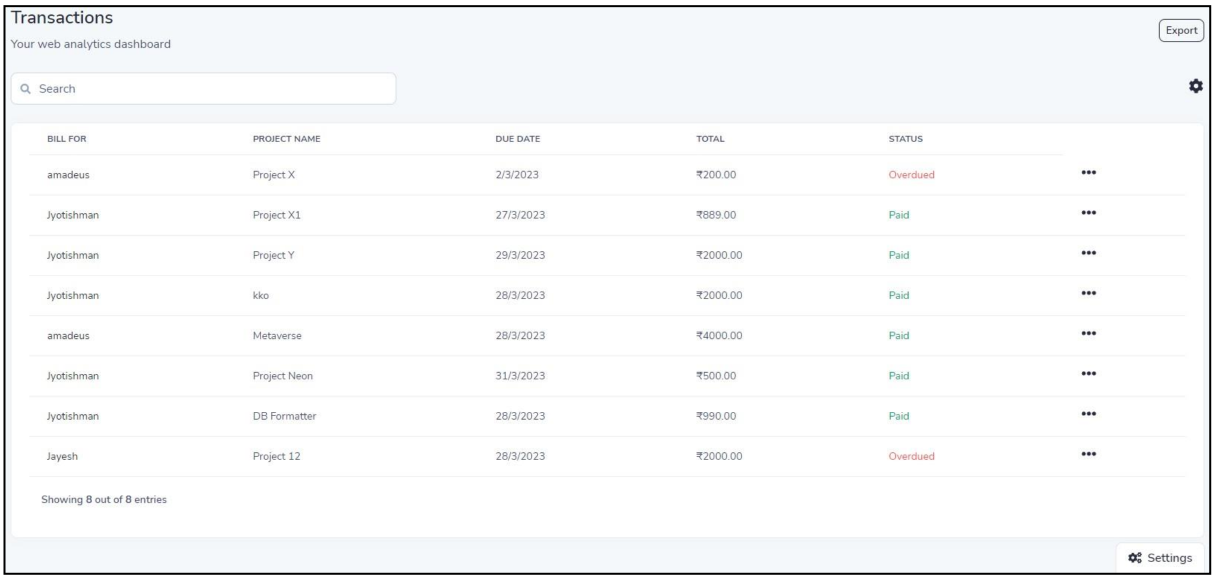
Select Client: Select Client	Select Project: Select Project
Enter Item Name: Item Name	Enter Amount: INR
Due Date: mm/dd/yyyy	Add Item

Generate Invoice

ITEM NAME	AMOUNT

CLIENT NAME	PROJECT NAME	TOTAL AMOUNT	DUE DATE	DOWNLOAD INVOICE
amadeus	Project X	₹200	03/02/2023	
Jyotishman	Project X1	₹889	03/27/2023	
Jyotishman	Project Y	₹2000	03/29/2023	
Jyotishman	kko	₹2000	03/28/2023	
amadeus	Metaverse	₹4000	03/28/2023	

Fig B5: Generate Invoice Page



Transactions

Your web analytics dashboard

BILL FOR	PROJECT NAME	DUE DATE	TOTAL	STATUS
amadeus	Project X	2/3/2023	₹200.00	Overdued
Jyotishman	Project X1	27/3/2023	₹889.00	Paid
Jyotishman	Project Y	29/3/2023	₹2000.00	Paid
Jyotishman	kko	28/3/2023	₹2000.00	Paid
amadeus	Metaverse	28/3/2023	₹4000.00	Paid
Jyotishman	Project Neon	31/3/2023	₹500.00	Paid
Jyotishman	DB Formatter	28/3/2023	₹990.00	Paid
Jayesh	Project 12	28/3/2023	₹2000.00	Overdued

Showing 8 out of 8 entries

Fig B6: Transactions Page

C. RESEARCH PAPER