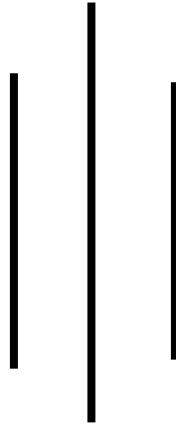


Xdezo Technologies pvt.ltd

Internship report on

Laravel Developer



By

Bipana Chapain

Intership Duration: December-April

To

Yas Shrestha

# ACKNOWLEDGEMENT

I would like to express my sincere gratitude to **XDezo Technologies** for granting me the invaluable opportunity to intern as a Laravel Developer. This experience has provided me with immense practical knowledge and exposure to real-world web development practices. I am deeply grateful to my mentor [Yas Shrestha], whose guidance, patience, and expertise have played a crucial role in enhancing my technical skills and problem-solving abilities. I also extend my heartfelt appreciation to the entire development team for their continuous support, encouragement, and willingness to share their knowledge. Their collaborative approach and constructive feedback have significantly contributed to my learning and professional growth during this internship

# TABLE OF CONTENTS

ACKNOWLEDGEMENT.....	ii
TABLE OF CONTENTS.....	iii
ABSTRACT.....	v
LIST OF FIGURES.....	vii
CHAPTER: I.....	1
INTRODUCTION.....	1
1.1 Introduction of background .....	1
1.2 Organization Profile .....	2
1.3 Introduction of system .....	3
1.3 Objectives of system.....	4
1.4 Functions of the System.....	4
CHAPTER: II.....	5
PROJECT SETUP & BASIC CRUD.....	5
2.1 Setup of laravel.....	5
2.2 database Schema.....	6
CHAPTER: III .....	9
ROLES, PERMISSIONS & FILE UPLOADS .....	9
3.1 Role Assignment.....	9
3.2 File Manages .....	9
3.3 Notifications & Status Updates.....	11
CHAPTER: IV .....	13

OUTCOMES AND DELIVERABLES .....	13
4.1 Version Control.....	14
CHAPTER: V .....	16
CONCLUSION.....	16

# ABSTRACT

The Ticket Management System is a web-based application developed to streamline the process of handling support requests, complaints, or service issues within an organization. The system enables users to raise tickets describing their issues, which are then tracked through various stages until resolution. This project focuses on creating an efficient, user-friendly, and scalable solution that improves communication between users and support teams, enhances transparency, and reduces the time taken to resolve issues. The system includes key features such as user authentication, ticket creation and assignment, real-time status updates, role-based access control, and an admin dashboard for performance monitoring. Additionally, it incorporates a notification system and report generation tools to ensure users are informed and administrators can assess service quality. A relational database is used to manage and store ticket and user data securely. Version control tools like Git were used throughout the development process to maintain a clean and organized codebase, enabling collaborative development and traceability of changes. The technologies used include HTML, CSS, JavaScript for the frontend, and a backend built using frameworks such as Laravel or Django, depending on the chosen stack. This project not only fulfills the functional and technical requirements but also demonstrates how technology can be effectively applied to real-world problems in service management. The Ticket Management System can be adapted for various domains, including IT help desks, educational institutions, and corporate environments. It sets a foundation for future enhancements such as mobile support, AI-driven ticket categorization,

and third-party integrations. In summary, this report presents the planning, design, development, and evaluation of a Ticket Management System that simplifies support processes and contributes to better service delivery and user satisfaction.

# LIST OF FIGURES

Figure 2.1: database schema .....	6
Figure 2.3 Ticket Read .....	7
Figure 2.3 Ticket Deleted .....	8
Figure 2.4 Ticket Create .....	8

# CHAPTER: I

## INTRODUCTION

### 1.1 Introduction of background

This report provides a comprehensive overview of my experiences as a Laravel Developer Intern at XDezo Technologies. Throughout the internship, I actively contributed to the development of dynamic and scalable web applications, focusing on both frontend and backend functionalities. My primary responsibilities included designing and implementing key features, integrating RESTful APIs, optimizing database queries, and debugging Laravel-based applications to enhance system efficiency and performance. Additionally, I gained hands-on experience working with MySQL, authentication systems, middleware, and third-party API integrations, all of which helped strengthen my full-stack development skills. The primary objective of this internship was to bridge the gap between theoretical knowledge and practical application, enabling me to develop a deeper understanding of Laravel's MVC architecture, security best practices, and performance optimization techniques. This report details my key projects, challenges faced, solutions implemented, and the invaluable skills I acquired during my time at XDezo Technologies.



## **1.2 Organization Profile**

XDezo Technologies is a renowned IT company headquartered in Pokhara, dedicated to delivering cutting-edge web development, software solutions, and digital transformation services. With a strong commitment to innovation and excellence, the company specializes in designing and developing high-performance, scalable, and secure applications that cater to diverse industry needs. XDezo Technologies leverages modern technologies such as Laravel for robust backend development, React.js for dynamic and responsive user interfaces, and cloud computing solutions to ensure scalability, security, and seamless accessibility of applications. The company's expertise spans across custom software development, API integrations, e-commerce solutions, enterprise applications, and cloud-based platforms, making it a trusted technology partner for businesses aiming to enhance their digital presence and operational efficiency. By embracing industry's best practices and staying updated with the latest technological advancements, XDezo Technologies continues to drive innovation and provide high-quality solutions to its clients.

### **1.3 Introduction of system**

Ticket Management System (TMS) is a specialized software solution designed to handle, track, and resolve support requests, also known as "tickets." These systems are widely used in IT help desks, customer support centers, event organizations, and project management environments to ensure seamless communication and efficient issue resolution. The primary function of a Ticket Management System is to serve as a centralized platform where users can report problems, request services, or raise queries. Once a ticket is submitted, the system assigns it to the appropriate team or personnel based on predefined criteria such as priority, department, or issue category. The team can then update the ticket status, add remarks, and track progress until the issue is resolved. One of the key benefits of a Ticket Management System is its ability to bring order and transparency to support operations. It helps prevent miscommunication, duplicate efforts, and missed requests. Every ticket is documented with important details such as submission date, issue description, assigned personnel, and resolution status. This not only helps with efficient issue tracking but also serves as a record for future analysis and reporting. Advanced ticket management systems offer features like automated ticket assignments, tracking, email notifications, performance analytics, and integration with other platforms like email. These features contribute to faster response times, better customer satisfaction, and improved team productivity.

In summary, a Ticket Management System is a valuable tool that enhances organizational efficiency by ensuring that all service requests are addressed promptly and systematically. It plays a vital role in

improving service quality, increasing accountability, and building trust among users, clients, and team members.

### **1.3 Objectives of system**

- To Role-Based Access Control
- To Assign & Prioritize Tickets
- To Status Tracking and Updates

### **1.4 Functions of the System**

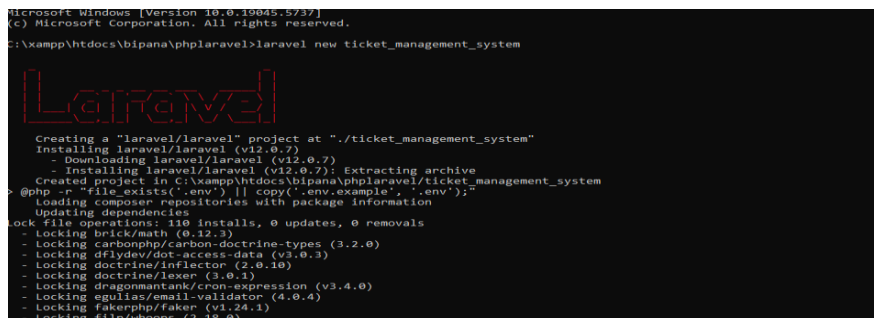
- Ticket Creation
- Ticket Assignment
- Ticket Tracking
- Ticket Prioritization
- Communication and Comments

# CHAPTER: II

## PROJECT SETUP & BASIC CRUD

### 2.1 Setup of laravel

To begin the development of the Ticketing System, Laravel 11 was installed as the core framework due to its powerful features and developer-friendly structure. The installation was done using Composer by executing the command `composer create-project laravel/laravel ticket-system`, which created a new Laravel 11 project named `ticket-system`. After setting up the project and configuring the `.env` file with appropriate database credentials, Laravel Breeze was installed to provide lightweight authentication scaffolding. This was achieved by running `composer require laravel/breeze --dev`, followed by `php artisan breeze:install` to generate the necessary authentication views, routes, and controllers. Once Breeze was installed, `npm install && npm run dev` compiled the frontend assets, and `php artisan migrate` was executed to create the default user-related tables in the database. As a result, basic user authentication features such as registration, login, and logout were fully integrated into the application, laying the foundation for user-based access to ticket functionalities.



```
Microsoft Windows [Version 10.0.19045.5377]
(c) Microsoft Corporation. All rights reserved.

C:\xampp\htdocs\bipana\phplaravel>laravel new ticket_management_system

Laravel

Creating a "laravel/laravel" project at "../ticket_management_system"
Installing laravel/laravel (v12.0.7)
- Downloading laravel/laravel (v12.0.7)
- Installing laravel/laravel (v12.0.7): Extracting archive
Created project in C:\xampp\htdocs\bipana\phplaravel\ticket_management_system
@php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Updating dependencies
Lock file operations: 110 installs, 0 updates, 0 removals
- Locking brick/math (0.12.3)
- Locking carbonphp/carbon-doctrine-types (3.2.0)
- Locking dflydev/dot-access-data (v3.0.3)
- Locking doctrine/inflector (2.0.10)
- Locking doctrine/lexer (3.0.1)
- Locking dragonmantank/cron-expression (v3.4.0)
- Locking egulias/email-validator (4.0.4)
- Locking fakerphp/faker (v1.24.1)
- Locking filp/whoops (2.18.0)
```

Figure 2.1 laravel setup

## 2.2 database Schema

The database schema for the Ticketing System was carefully designed to ensure clear relationships between entities and to support efficient data management. At the core of the system is the Users table, which stores information about registered users and is automatically provided through Laravel Breeze's authentication setup. The Tickets table is linked to the Users table via a foreign key (user\_id) to identify the creator of each ticket. It includes fields such as title, description, and status to capture essential ticket details. To facilitate discussion and updates on each ticket, a Comments table was created with foreign keys referencing both the tickets and users tables, allowing users to post comments related to specific tickets. Additionally, an Attachments table was included to support file uploads, with each attachment linked to its respective ticket through a ticket\_id foreign key. This schema ensures a well-structured relational database that supports core functionalities like ticket creation, user interaction, and file management within the system.

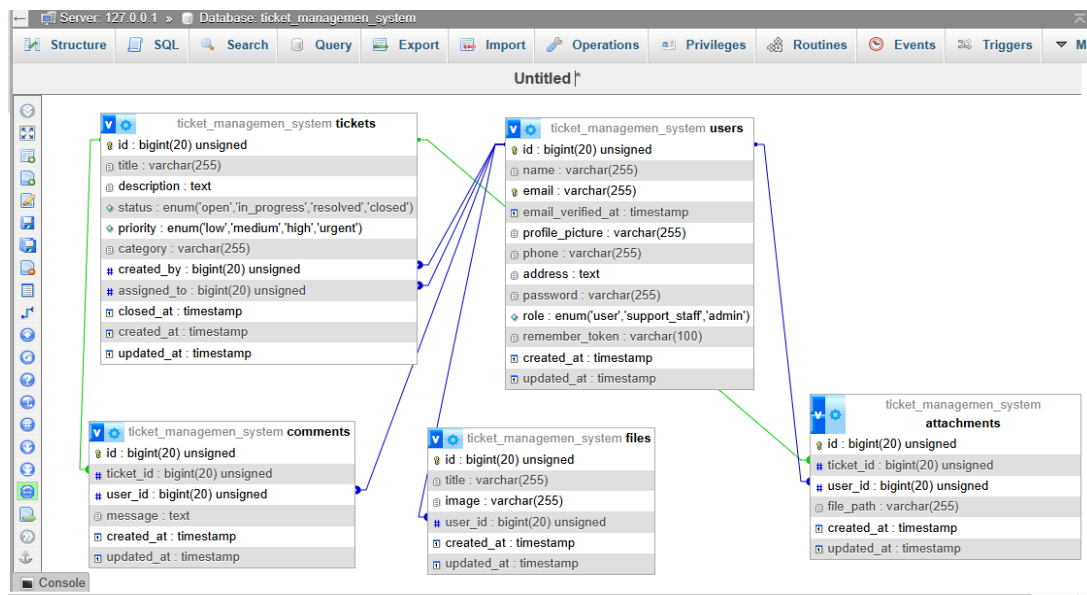
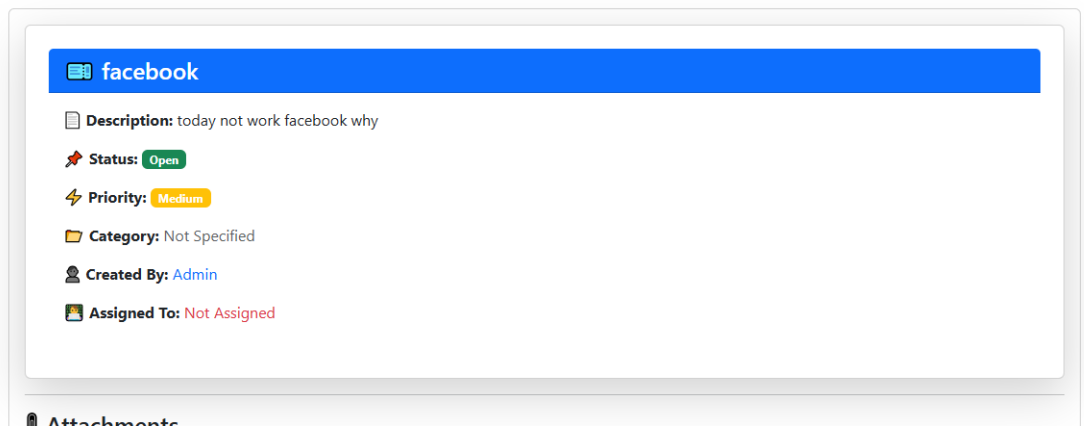


Figure 2.1: database schema

## 2.3 CRUD for Tickets

To enhance the functionality of the Ticket Management System, a report generation feature was incorporated to allow users or administrators to view a summary of ticket activity over time. This report provides an overview of key metrics such as the total number of tickets, tickets by status (open, in progress, closed), and user-specific ticket data. The implementation involved creating a dedicated route and controller method to collect and filter ticket data from the database using Laravel's Eloquent ORM. The data was then passed to a Blade view where it was presented in a structured format, optionally including charts or tables for better visualization. Laravel's built-in date filtering methods were used to generate time-based reports. This feature helps users track issue trends and administrators monitor system usage, making the application more practical and valuable for real-world use cases.

### Ticket Details



The screenshot shows a web interface for viewing a ticket. At the top, there is a blue header bar with a ticket icon and the text "facebook". Below this, the ticket details are listed:

- Description:** today not work facebook why
- Status:** Open (indicated by a green pill)
- Priority:** Medium (indicated by a yellow pill)
- Category:** Not Specified
- Created By:** Admin
- Assigned To:** Not Assigned

Below the details, there is a section labeled "Attachments" with a plus icon.

Figure 2.3 Ticket Read

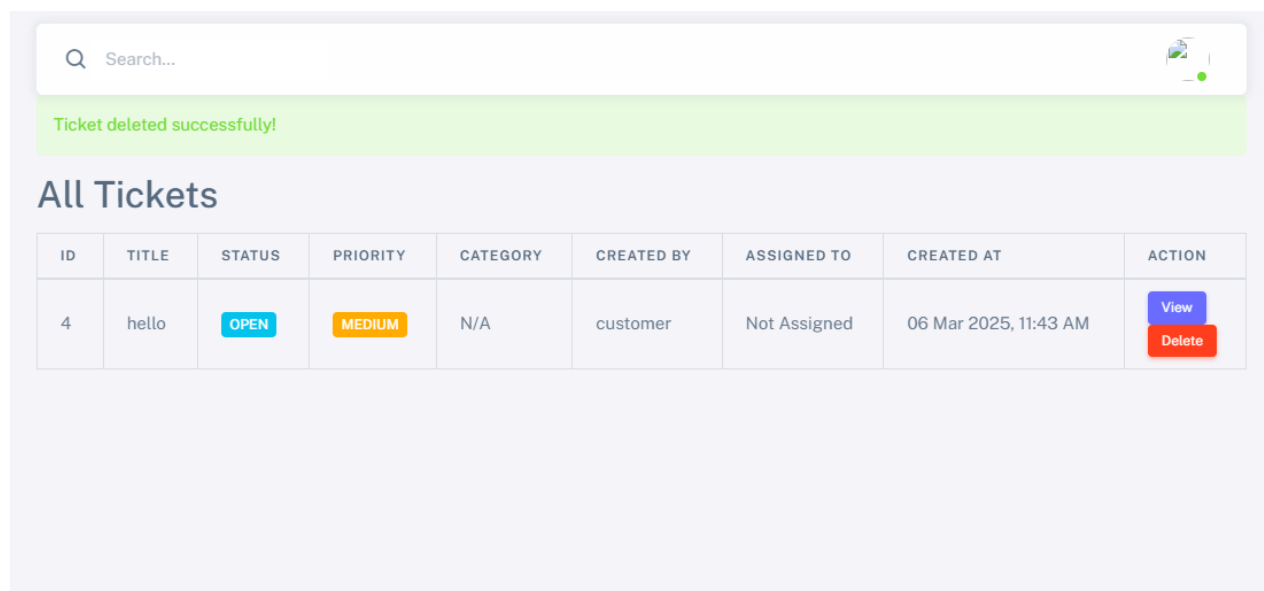


Figure 2.3 Ticket Deleted

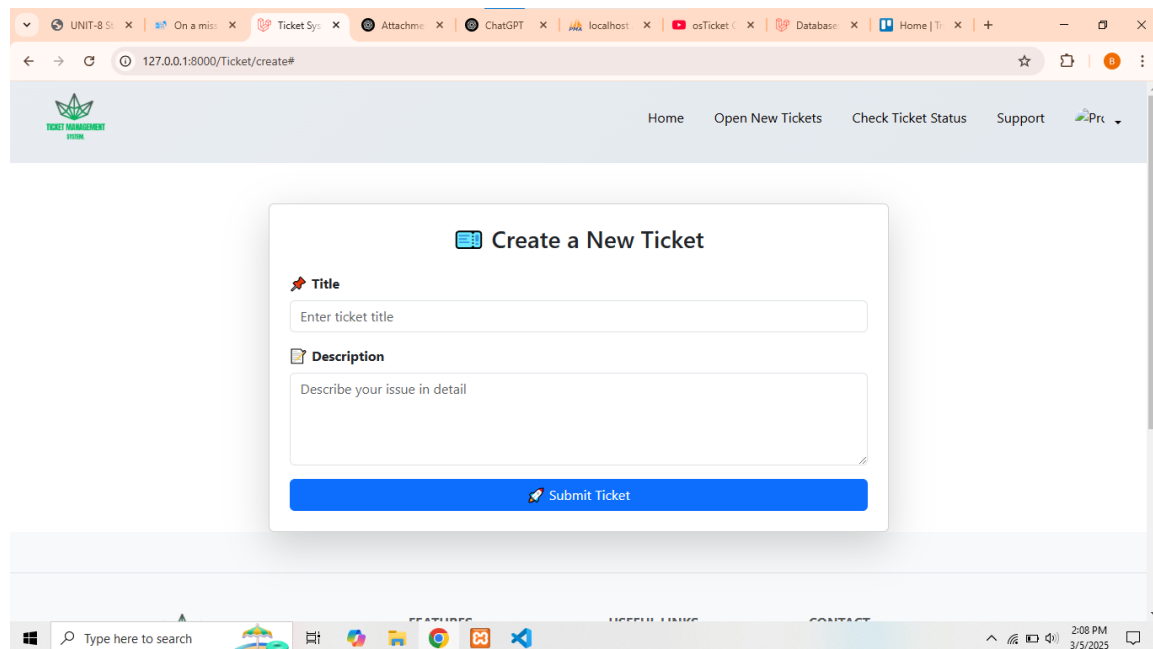


Figure 2.4 Ticket Create

## **CHAPTER: III**

### **ROLES, PERMISSIONS & FILE UPLOADS**

#### **3.1 Role Assignment**

To manage access and define responsibilities within the Ticket Management System, three distinct roles were established: User, Support Staff, and Admin. These roles were implemented using the Spatie Laravel-Permission package, which simplifies role and permission management by allowing dynamic role assignments and middleware-based access control. Users are the default role upon registration and can create, view, and comment on their own tickets. Support Staff have extended privileges to view and respond to all tickets, helping users resolve issues. Admins possess the highest level of control, including the ability to manage users, assign roles, delete any ticket, and generate reports. Roles were assigned through both database seeders and UI-based management, where necessary. Middleware such as `role:admin` or `permission:view tickets` was applied to protect routes and controller methods. This role-based system not only enhances security but also ensures that users interact with the system according to their responsibilities, providing a well-organized and scalable access structure.

#### **3.2 File Manages**

The File Management feature in the Ticket Management System was designed to efficiently handle user-uploaded files, such as screenshots or supporting documents. When users create or update a ticket, they are given the option to upload files, which are validated on the server side



to ensure they meet size and type requirements (e.g., images or PDFs). Laravel's Storage facade was used to store these files securely in the public disk, making them accessible while keeping directory structure organized. Each file's metadata, including its path and associated ticket ID, is stored in the attachments table for easy reference. Users and support staff can view or download attachments through the ticket detail view, while admins can delete unnecessary files, ensuring proper file lifecycle management. This system not only enhances communication but also ensures that storage is used efficiently and securely.

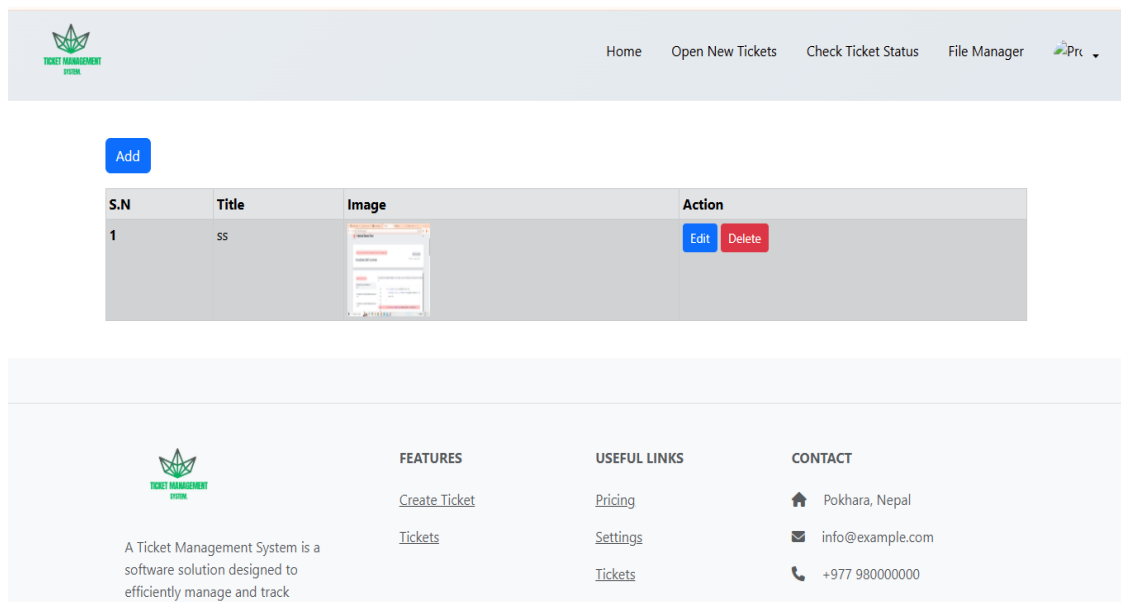


Figure 3.1 file Manage

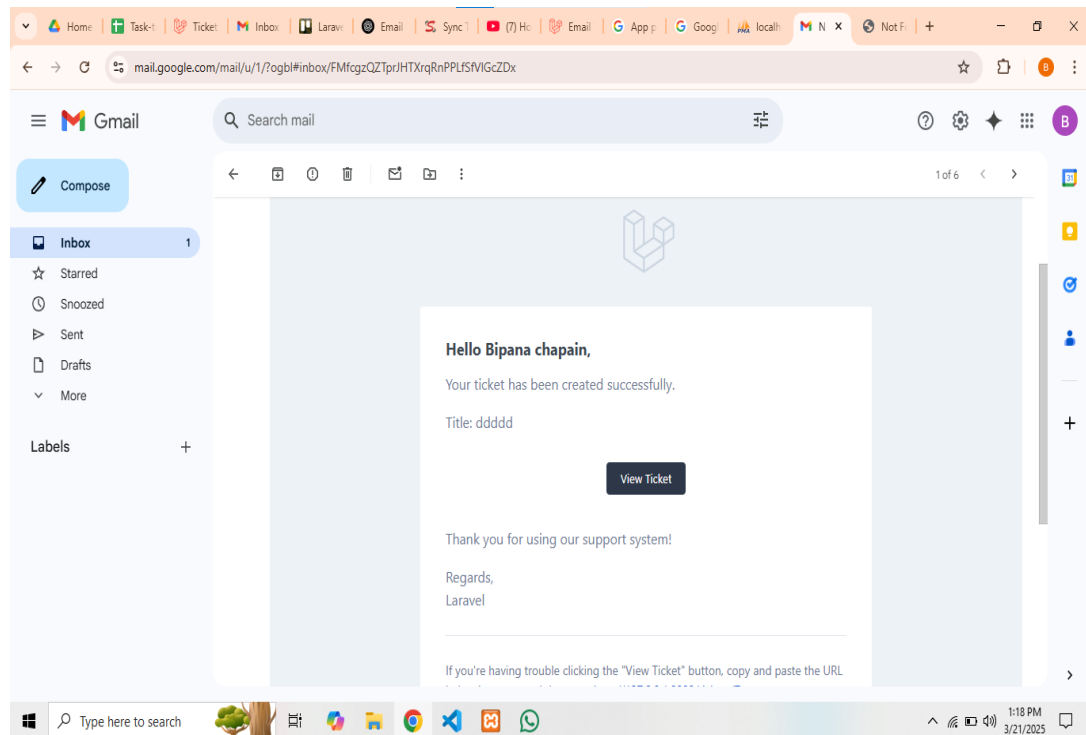
### **3.3 Notifications & Status Updates**

To ensure clear and timely communication between users and support staff, a Notifications and Status Update system was implemented in the Ticket Management System. This feature is essential for keeping users informed about the progress of their tickets and for improving the efficiency of issue resolution. Whenever a new ticket is submitted by a user, the system automatically sends an email notification to the support staff, notifying them of the new issue that needs attention. This is achieved using Laravel's built-in Notification and Mail systems, which allow the creation of custom notification classes. These classes define how the notification is structured and delivered—most commonly via email.

In addition to ticket creation, the system also handles status change notifications. For example, when a support staff member updates the status of a ticket from Open to In Progress, or from In Progress to Resolved, the ticket owner receives a real-time email alert. These updates keep users informed about the exact stage of their ticket and reduce the need for follow-up queries. The notifications are triggered by Laravel events that listen for changes in the ticket model and dispatch the appropriate notifications to the respective users.

Furthermore, a clear ticket status workflow was defined and implemented within the system. Each ticket passes through the stages of Open, In Progress, and finally Resolved. These statuses are displayed on the ticket detail view and allow both users and support staff to track progress easily. The status management functionality is embedded within the controller logic, allowing authorized users to update ticket statuses as needed.

By integrating both real-time notifications and a well-defined status workflow, the system not only improves user experience but also promotes transparency and accountability in the issue resolution process. This feature greatly contributes to the overall professionalism and usability of the application.



*Figure 3.2 Email Notification*

## CHAPTER: IV

### OUTCOMES AND DELIVERABLES

The development of a Ticket Management System results in several key outcomes and deliverables aimed at improving issue tracking and resolution within an organization. The primary outcome is the creation of an efficient and centralized platform where users can raise support tickets, track their status, and receive timely resolutions. This system enhances communication between users and support staff, improves accountability by assigning specific tickets to individuals or teams, and ultimately leads to faster and more organized problem-solving. It also provides data-driven insights through reports and analytics, helping organizations identify recurring issues and optimize their support services. The improved transparency and responsiveness contribute to greater user satisfaction and trust. Additionally, the system is designed to be scalable, ensuring it can grow alongside the organization's needs.

In terms of **deliverables**, the development process begins with a detailed requirement specification document outlining the system's goals, user roles, and functionalities. This is followed by UI/UX design deliverables such as wireframes and final interface mockups to ensure a user-friendly experience. The database schema and ER diagrams are designed to support structured and efficient data storage. The frontend of the system is developed using responsive web technologies, while the backend handles the core ticket management logic, often built with frameworks like Laravel or Django. A secure user authentication module ensures role-based access, while the ticket lifecycle module covers the creation, assignment, updating, and closure of tickets. Additional deliverables

include a notification system to alert users of status updates, an admin dashboard for managing users and viewing analytics, and comprehensive reports for performance tracking. Testing reports, deployment scripts, user manuals, and optional training materials complete the project, ensuring a smooth rollout and easy adoption of the system.

## **4.1 Version Control**

In the development of a Ticket Management System, implementing version control plays a crucial role in maintaining the integrity, organization, and traceability of the project's source code and configuration changes. Version control, typically managed through systems like Git, allows developers to track every modification made to the system over time, ensuring that all updates are documented and reversible. This is particularly important in collaborative environments, where multiple developers may be working simultaneously on different features such as ticket creation, user authentication, or notification modules. By using branches, developers can isolate their work, test new functionalities independently, and merge them into the main codebase once validated. Version control also supports rollback capabilities, enabling quick restoration to a previous stable version if a newly introduced feature causes issues. Additionally, it enhances deployment processes by maintaining clean and organized records of which version was deployed and when.

For documentation and reporting purposes, the version control system can be integrated with the ticketing module itself by referencing ticket IDs in commit. This approach ensures that every code change is directly linked to a specific requirement or bug fix, improving traceability and

accountability. The version control report, typically generated through platforms like GitHub or GitLab, includes logs of commit histories, contributor activities, merge requests, and branch management summaries. This report becomes a valuable deliverable, offering insights into the project's progress, developer contributions, and change history, which can be included as part of the final project documentation or submitted to supervisors for review.

# **CHAPTER: V**

## **CONCLUSION**

The Ticket Management System developed as part of this project has proven to be a highly effective solution for managing support and service-related tasks within an organization. It successfully automates the ticketing process by enabling users to submit their issues or requests, which are then categorized, prioritized, assigned, and tracked through to resolution. The system not only simplifies the overall support process but also ensures better transparency, accountability, and efficiency in handling tickets.

One of the key strengths of this project lies in its user-friendly interface, which allows both end-users and support staff to interact with the system effortlessly. The inclusion of features such as real-time status updates, notifications, and reporting tools enhances user experience and helps support teams stay organized and focused. Moreover, the system allows administrators to monitor overall performance, generate insightful reports, and make data-driven decisions to improve service quality.

From a development perspective, version control played a crucial role in maintaining a clean and structured codebase. Using Git and platforms like GitHub allowed the team to collaborate effectively, track all changes, and manage multiple features simultaneously without conflicts. The implementation of modules like user authentication, ticket lifecycle management, and admin dashboards further contributed to the robustness of the system.

In addition, the project has laid the groundwork for future improvements such as integrating AI-based ticket classification, mobile application support, and third-party tool integration like email or messaging platforms. These features can further enhance the system's capability and adaptability to different organizational environments.

In conclusion, the Ticket Management System is a practical and scalable solution that addresses common challenges in issue tracking and resolution. It not only meets the functional requirements but also adds value by improving communication, speeding up response times, and ensuring customer satisfaction. This project demonstrates the effective application of software development principles to solve real-world problems.