**ABSTRACT**

The Fleet Management System is an advanced software application developed using C#.NET, designed to optimize the management of vehicle fleets. This system provides a unified platform to automate and streamline the administrative tasks associated with fleet operations, significantly enhancing overall efficiency.

The application supports two primary user roles: Admin (Owner) and Driver. The Admin role encompasses comprehensive functionalities including managing driver records, monitoring vehicle statuses, assigning vehicles to specific trips, and setting fare rates. This role ensures efficient oversight and management of fleet resources. Conversely, the Driver role enables users to view and manage trip assignments, update trip details, and close trips once completed. Additionally, drivers have access to a dashboard for receiving updates and suggestions, facilitating better communication and operational management.

Utilizing SQL Server for data storage and Visual Studio 2012 as the development environment, the system ensures robust data management and a reliable development framework. The Fleet Management System reduces manual errors, optimizes resource utilization, and improves operational productivity. Future enhancements may include real-time GPS tracking and automated reporting to further extend the system’s capabilities and value.

In essence, the Fleet Management System represents a significant advancement in fleet management technology, offering a streamlined, efficient solution for modern vehicle operations.

ABSTRACT IN TAMIL

C#.NET பயன்படுத்தி உருவாக்கப்பட்ட வாகன மேலாண்மை அமைப்பு, வாகனக் கூட்டத்தின் செயல்திறனை மேம்படுத்த ஒரு வலுவான மென்பொருள் தீர்வு ஆகும். இந்த பயன்பாடு, வாகனங்கள், ஓட்டுநர்கள் மற்றும் பயணங்களை மையமாகக் கொண்டு நிர்வகிக்க, முக்கிய நிர்வாக பணிகளை தானாக செய்யும் மற்றும் செயல்திறனை மேம்படுத்தும் வழியாக வடிவமைக்கப்பட்டுள்ளது.

இந்த அமைப்பு இரண்டு முக்கிய பயனர் வகைகளை ஆதரிக்கிறது: நிர்வாகி (உரிமையாளர்) மற்றும் ஓட்டுநர். நிர்வாகி, ஓட்டுநர் பதிவு விவரங்களை நிர்வகிக்க, வாகன நிலைகளை கண்காணிக்க, வாகனங்களை பயணங்களுக்கு நியமிக்க மற்றும் பயண செலவுகள் மாறுதலைச் செய்ய முடியும், இது வாகன குழுவின் செயல்பாடுகளின் முழுமையான கண்காணிப்பை உறுதிப்படுத்துகிறது. அதேவேளை, ஓட்டுநர் பயண விவரங்களைப் பார்க்க மற்றும் நிர்வகிக்க, பயண நிலைகளைப் புதுப்பிக்க மற்றும் பயணங்களை முடிக்கப்பட்டதாகக் குறிக்க முடியும், மேலும் அறிக்கைகள் மற்றும் சிந்தனைகள் பெற ஒரு டாஷ்போர்டை அணுக முடியும்.

தரவுகளை நிர்வகிக்க SQL Server மற்றும் மேம்பாட்டு சூழலாக Visual Studio 2012 பயன்படுத்தப்படுவதால், இந்த அமைப்பு நம்பகமான தரவுப் பராமரிப்பை மற்றும் நிலையான மேம்பாட்டு தளத்தை உறுதிப்படுத்துகிறது. இது கையேடு பிழைகளை குறைப்பதுடன், வளங்களின் பயன்திறனை மேம்படுத்தி, வாகன மேலாண்மை செயல்பாடுகளை எளிதாக்க முயற்சிக்கிறது. எதிர்கால மேம்பாடுகள், நேரடி GPS கண்காணிப்பு மற்றும் தானாகவே அறிக்கைகள் போன்ற அம்சங்களை உள்ளடக்கக்கூடும்.

வாகன மேலாண்மை அமைப்பு, நவீன வாகனக் கூட்டங்களுக்கு ஒருங்கிணைந்த, செயல்திறன் அதிகரிக்கும் தீர்வாக அமைகின்றது.

**CHAPTER 1**

**INTRODUCTION**

**1.1 OVERVIEW OF THE PROJECT**

The Fleet Management System is a sophisticated software solution designed to optimize and automate the management of vehicle fleets. Developed using C#.NET, this application offers a centralized platform for efficiently handling various aspects of fleet operations. The primary objective of the system is to enhance operational efficiency and effectiveness by streamlining vehicle management, driver coordination, and trip scheduling.

The system is structured to support two main user roles: Admin (Owner) and Driver, each with specific functionalities tailored to their responsibilities. For Admins, the system provides a comprehensive set of tools to manage fleet operations. Admins can log in to the system to access and manage driver details. This includes the ability to add new driver records, view existing ones, edit information as needed, and delete records when necessary. The system ensures that driver information is current and accurately reflects the status of each driver.

In addition to managing driver details, Admins have the capability to monitor the status of each vehicle within the fleet. This includes tracking vehicle availability, scheduling maintenance, and overseeing current assignments. By having real-time visibility into the status of each vehicle, Admins can make informed decisions regarding resource allocation and ensure that the fleet operates smoothly.

One of the critical features of the system is its ability to facilitate the assignment of vehicles and drivers to specific trips. Admins can assign vehicles to drivers based on availability and requirements, optimizing the use of fleet resources. Additionally, Admins can set and adjust fare rates for trips, taking into account factors such as distance traveled, vehicle type, and demand. This functionality helps in managing the financial aspects of fleet operations and ensures transparency in fare calculations.

Drivers benefit from a range of features designed to support their day-to-day activities. Upon logging into the system, drivers can view their assigned trips and access detailed information about each trip. They have the ability to add details, such as start and end times, locations, and other relevant information. Drivers can also update the status of their trips, marking them as in progress or completed as appropriate.

The system includes a dashboard feature for drivers, which provides them with notifications, suggestions, and updates related to their assignments. This dashboard serves as a central point of communication, helping drivers stay informed about any changes or requirements associated with their trips. By offering real-time updates and recommendations, the dashboard enhances drivers' ability to manage their tasks efficiently.

The technological foundation of the Fleet Management System includes C#.NET for application development, SQL Server for database management, and Visual Studio 2012 as the development environment. C#.NET provides a robust framework for building the application’s core functionality, while SQL Server ensures reliable data storage and management. Visual Studio 2012 offers a stable and comprehensive environment for coding, debugging, and testing the application.

The development process of the Fleet Management System involves several phases, each crucial to the successful implementation of the project. The initial phase is requirement analysis, during which the needs and objectives of the system are gathered and documented. This phase sets the foundation for the subsequent stages by defining what the system must accomplish.

Following requirement analysis, the system design phase involves creating the architecture of the application, including the database schema and user interface layouts. This design phase ensures that the system’s structure supports its intended functionality and provides a user-friendly experience.

The development phase encompasses the actual coding of the application, using C#.NET to implement the features and functionalities defined during the design phase. Concurrently, the SQL Server database is set up to handle data storage and management. This phase also includes integrating various components of the system to ensure they work together seamlessly.

Testing is a critical phase of the project, involving rigorous checks to ensure that the system operates correctly and meets all user requirements. This includes unit testing, integration testing, and system testing to identify and resolve any issues before deployment.

The final phase of the project is deployment, during which the system is launched and made available to users. Training is provided to ensure that users can effectively utilize the system’s features and functionalities. This training helps users become familiar with the system and ensures a smooth transition to its use in daily operations.

Looking to the future, the Fleet Management System is designed with scalability and enhancement in mind. Potential future improvements include integrating real-time GPS tracking to monitor vehicle locations and incorporating automated reporting features to generate insights and track performance metrics. These enhancements aim to further increase the system’s capabilities and provide additional value to fleet management operations.

In summary, the Fleet Management System is a sophisticated and integrated solution that addresses the complexities of modern vehicle fleet management. By automating key processes and providing a centralized platform for administration, the system enhances operational efficiency, reduces manual errors, and optimizes resource utilization. The combination of C#.NET, SQL Server, and Visual Studio 2012 ensures a robust and reliable application that meets the needs of today’s fleet management requirements and supports future growth and enhancements.

**1.2 EXISTING SYSTEM**

Traditional fleet management systems often rely on manual processes and outdated technologies that present several challenges. Typically, these systems involve manual record-keeping, paper-based documentation, and basic spreadsheet tools, each contributing to various inefficiencies and limitations.

In traditional fleet management, data is usually entered manually into spreadsheets or paper records. This method is not only time-consuming but also prone to human error, which can result in inaccuracies in managing vehicle information, driver details, and trip data. The reliance on paper forms for recording and tracking information—such as vehicle maintenance, driver logs, and trip reports—can lead to the loss or misplacement of crucial documents, complicating data retrieval and management.

Furthermore, many organizations use multiple, separate systems for different fleet management aspects, such as maintenance tracking, driver management, and trip scheduling. These disparate systems often lack integration, leading to fragmented data and inefficiencies in coordinating fleet operations. The absence of a unified platform makes it challenging to obtain a comprehensive view of fleet activities, thus impacting decision-making and operational efficiency.

Traditional systems also typically lack real-time tracking capabilities. Fleet managers may not have up-to-date information regarding vehicle locations, maintenance status, or driver performance, which impedes their ability to make prompt, informed decisions. The reliance on inefficient communication methods—such as phone calls or messages—between drivers and fleet managers can lead to delays in receiving updates and instructions, further affecting operational coordination.

These existing methods are associated with several disadvantages. The high error rate is a significant concern, as manual data entry and paper documentation are susceptible to inaccuracies. This can lead to unreliable records and hinder effective decision-making. Additionally, the time-consuming nature of manual processes diverts attention from strategic planning and operational efficiency.

Limited integration among disparate systems creates fragmented data, complicating the coordination of tasks and resources. This fragmentation can also delay the retrieval of comprehensive information, affecting the fleet manager’s ability to respond quickly to issues and make timely decisions. Ineffective communication channels contribute to delays in updating and instructing drivers, impacting overall fleet performance and coordination.

Data security is another area of concern with traditional systems. Paper-based and manually managed systems often lack robust security measures, making sensitive data vulnerable to unauthorized access, loss, or damage. Finally, scalability issues arise as fleet size grows. Traditional systems may struggle to manage increasing volumes of data and operations, leading to inefficiencies and difficulties in maintaining effective fleet management.

Overall, the traditional methods of fleet management highlight a clear need for modern solutions that address these challenges through automation, integration, and real-time capabilities. Transitioning to an advanced Fleet Management System can significantly improve operational efficiency and effectiveness, overcoming the limitations of existing systems.

**1.3PROPOSED SYSTEM**

The proposed Fleet Management System aims to address the shortcomings of traditional fleet management methods by providing a comprehensive, automated solution that enhances operational efficiency and effectiveness. Developed using C#.NET, the system integrates modern technologies to streamline and optimize various aspects of fleet management.

This advanced system introduces a centralized platform that supports two primary user roles: Admin (Owner) and Driver. For Admins, the system offers robust tools to manage and oversee fleet operations seamlessly. Admins can access a unified interface to handle all driver-related tasks, including adding new drivers, updating records, and managing existing information. This centralization ensures that driver details are always current and accurately maintained.

In addition to driver management, Admins can monitor the status of each vehicle within the fleet. The system provides real-time visibility into vehicle availability, maintenance schedules, and current assignments, enabling Admins to make informed decisions about resource allocation and fleet management. Admins have the capability to assign vehicles and drivers to specific trips efficiently. This functionality optimizes resource utilization by matching the right vehicle and driver to each trip based on availability and requirements.

The system also allows Admins to manage fare rates for trips, adjusting rates based on factors such as distance traveled, vehicle type, and demand. This feature helps in managing the financial aspects of fleet operations transparently and effectively.

Drivers benefit from an array of features designed to simplify their tasks and improve their efficiency. Upon logging into the system, drivers can view their assigned trips and access detailed information about each trip. They can add details such as start and end times, locations, and other relevant information, as well as update the status of their trips as they progress. This functionality ensures that trip records are accurate and up-to-date.

The system includes a dashboard feature for drivers, providing a central point for accessing notifications, suggestions, and updates related to their assignments. This real-time communication tool enhances coordination between drivers and fleet managers, ensuring that drivers receive timely updates and instructions.

To support the proposed system, modern technologies are employed. C#.NET is used for application development, offering a robust framework for building and maintaining the system’s core functionalities. SQL Server is utilized for data management, providing a reliable and secure platform for storing and retrieving data related to vehicles, drivers, and trips. Visual Studio 2012 serves as the development environment, facilitating efficient coding, debugging, and testing of the application.

The development of the proposed system involves several key stages. The first stage is requirement analysis, where the specific needs and objectives of the fleet management process are gathered and documented. This stage establishes a clear understanding of what the system must achieve. Following this, the system design phase focuses on creating the architecture, database schema, and user interfaces necessary for the application. This design ensures that the system’s structure supports its intended functionality and provides a user-friendly experience.

The development phase encompasses the actual coding of the system using C#.NET, with SQL Server set up to manage data storage and retrieval. Integration of various components is carried out to ensure seamless operation. Rigorous testing is conducted to validate that the system functions as expected and meets user requirements. Finally, the system is deployed, and training is provided to ensure that users can effectively utilize the new platform.

Looking ahead, the proposed system is designed with scalability and future enhancements in mind. Potential upgrades may include integrating real-time GPS tracking to monitor vehicle locations and implementing automated reporting features to provide insights and track performance metrics. These enhancements aim to further improve the system’s functionality and value, supporting the evolving needs of fleet management.

In summary, the proposed Fleet Management System offers a modern solution to the challenges faced by traditional methods. By automating key processes, integrating functionalities, and providing real-time capabilities, the system enhances operational efficiency and effectiveness, making it a valuable tool for contemporary fleet management

**CHAPTER 2**

**SYSTEM ANALYSIS**

System analysis is a critical phase in the development of an information system that involves examining and understanding the current system's functionality and identifying areas for improvement. This process includes gathering detailed requirements from stakeholders, evaluating existing systems and processes, and determining the needs and objectives of the new system. The goal of system analysis is to define the specific functionalities, constraints, and expectations for the proposed system, ensuring it effectively addresses the problems or limitations of the existing system. By analyzing the current setup and envisioning the desired improvements, system analysis lays the foundation for designing and implementing a solution that enhances operational efficiency and meets organizational goals.

**2.1 SYSTEM REQUIREMENTS**

System requirements are a set of specifications that define the necessary hardware, software, and operational capabilities needed for a system to function effectively. These requirements detail the technical and functional aspects of the system, including the hardware configurations (such as processors, memory, and storage), software components (such as operating systems, development tools, and database management systems), and user interfaces required to support the intended operations. They also outline performance criteria, security measures, and integration needs to ensure compatibility with existing systems and meet user needs. By clearly defining these requirements, organizations can ensure that the system is capable of supporting its intended functions, delivering reliable performance, and achieving the desired outcomes.

**2.1.1 HARDWARE REQUIREMENTS**

Hardware requirements define the physical components needed to support the operation of a software system effectively. They include specifications for the central processing unit (CPU), memory (RAM), storage, network connectivity, and peripheral devices, all of which are essential for running the system smoothly and efficiently. These requirements ensure that the hardware can handle the system's processing demands, store necessary data, provide reliable network access, and support user interactions. By establishing clear hardware requirements, organizations can ensure that their infrastructure is capable of supporting the software’s functionality, delivering optimal performance, and meeting the operational needs of the users

|  |  |  |
| --- | --- | --- |
| **Component** | **Minimum Requirements** | **Recommended Requirements** |
| Processor | Intel Core i3 or equivalent | Intel Core i5 or higher |
| RAM | 4 GB | 8 GB or higher |
| Storage | 250 GB HDD | 500 GB SSD or higher |
| Network | Reliable internet connection | High-speed internet connection |
| Display | 1024 x 768 resolution | 1920 x 1080 resolution or higher |

**2.1.2 SOFTWARE REQUIREMENTS**

Software requirements specify the essential software components and configurations needed to support the effective operation of an information system. These include the operating system, development tools, database management systems, and any additional software necessary for running and maintaining the system. Software requirements define the specific versions and capabilities of these components to ensure compatibility, performance, and security. They also encompass user interface design requirements, reporting tools, and integration capabilities with other systems. By outlining these requirements, organizations ensure that the software environment is conducive to the successful implementation and functioning of the system, providing the necessary tools and infrastructure to meet user needs and system objectives.

|  |  |
| --- | --- |
| **Component** | **Details** |
| Operating System | Windows 7 or later (Windows 10/11 recommended) |
| Development Tools | Visual Studio 2012 |
| Programming Language | C#.NET |
| Database Management System | SQL Server 2012 or later |
| Database Management Tool | Microsoft SQL Server Management Studio (SSMS) |

**CHAPTER 3**

**SYSTEM SPECIFICATION**

**3.1 OPERATING SYSTEM**

**WINDOWS 10**

Windows 10 is a major release of the Microsoft Windows operating system, unveiled in July 2015. It represents a significant evolution from its predecessor, Windows 8.1, incorporating numerous enhancements and features designed to improve usability, security, and performance. This version of Windows marked a strategic shift for Microsoft, as it sought to unify the user experience across a range of devices, including PCs, tablets, and smartphones, under a single operating system.

One of the hallmark features of Windows 10 is its return to the traditional desktop interface, which was a response to widespread feedback from users who were dissatisfied with the radical changes introduced in Windows 8. The Start Menu, a familiar element in previous versions, was reintroduced and redesigned to blend the classic list-based format with the live tiles introduced in Windows 8. This hybrid approach allows users to access their most-used apps and programs quickly while benefiting from dynamic, real-time updates from live tiles.

Windows 10 also introduced the concept of a unified operating system that spans across multiple device types. This concept is embodied in the Windows Universal Platform (UWP), which allows developers to create applications that can run seamlessly on a wide range of devices, from desktops and laptops to tablets and smartphones. The UWP framework ensures that apps provide a consistent experience regardless of the device being used, which enhances usability and streamlines application development.

Another significant innovation in Windows 10 is the integration of Cortana, Microsoft's virtual assistant. Cortana offers voice-activated assistance, helping users with tasks such as setting reminders, performing web searches, and managing calendars. Cortana's integration into the operating system reflects a broader trend toward incorporating artificial intelligence into everyday computing, aiming to provide users with more intuitive and responsive interactions.

Windows 10 also emphasizes security with the introduction of several new features and improvements. One of the key security advancements is Windows Hello, which offers biometric authentication options, including facial recognition and fingerprint scanning, to enhance user security and convenience. This feature represents a shift away from traditional password-based authentication, providing a more secure and user-friendly alternative. Additionally, Windows 10 includes improved encryption capabilities and advanced threat protection to safeguard against emerging cyber threats.

The operating system incorporates the concept of continuous updates, with the introduction of Windows as a Service (WaaS). Unlike previous versions of Windows, which received periodic major updates, Windows 10 receives regular updates that introduce new features, security patches, and performance improvements. This approach ensures that the operating system remains up-to-date with the latest advancements and security measures, providing users with a more stable and secure computing environment.

Windows 10 also enhances the multitasking experience with features such as Task View and Snap Assist. Task View allows users to view and switch between open applications and virtual desktops, making it easier to manage multiple tasks simultaneously. Snap Assist improves window management by offering suggestions for snapping open windows into predefined areas of the screen, enhancing productivity and organization.

Gaming on Windows 10 has also been improved with the integration of DirectX 12, which provides better performance and graphics capabilities for gaming applications. The operating system supports a range of gaming features, including the Xbox app, which allows users to connect with Xbox Live, stream games, and manage their gaming library. These enhancements reflect Microsoft's commitment to providing a robust gaming experience on its platform.

In terms of compatibility, Windows 10 is designed to support a wide range of hardware and software. It offers backward compatibility with many older applications and devices, ensuring that users can transition smoothly from previous versions of Windows without encountering significant compatibility issues. The operating system also includes built-in tools and features to assist with the upgrade process, making it easier for users to migrate to Windows 10 from older versions.

Overall, Windows 10 represents a comprehensive and forward-looking operating system that blends familiarity with innovation. It addresses user feedback from previous versions, introduces new features to enhance productivity and security, and embraces the evolving landscape of computing with support for a diverse range of devices and applications. Through continuous updates and a focus on user experience, Windows 10 aims to provide a versatile and reliable platform for both personal and professional use.

**3.2 SOFTWARE DESCRIPTION**

**VISUAL STUDIO 2012**

Visual Studio 2012, released by Microsoft, is an integrated development environment (IDE) designed to support the development of a wide range of applications across various platforms. As part of the Visual Studio family, it represents a significant evolution from its predecessors, offering a suite of tools and features aimed at enhancing productivity and streamlining the development process. This version is well-suited for developing applications for Windows, web, and mobile environments, and it continues to be used by many developers despite newer versions being available.

At its core, Visual Studio 2012 is built around the concept of providing a comprehensive development environment where developers can write, test, and debug code all within a single interface. The IDE supports multiple programming languages, including C#, VB.NET, C++, and F#, among others, allowing developers to work on diverse projects without needing to switch between different tools. This multi-language support is complemented by advanced code editing features, such as syntax highlighting, code completion, and refactoring tools, which help streamline the coding process and improve code quality.

One of the notable features introduced in Visual Studio 2012 is the revamped user interface, which includes a more modern and streamlined design compared to earlier versions. The IDE incorporates a new color scheme and layout that aims to enhance usability and reduce visual clutter. The improved interface includes features such as the new Start Page, which provides quick access to recent projects, useful resources, and tutorials. Additionally, the IDE's enhanced navigation and search capabilities make it easier for developers to locate files, classes, methods, and other elements within large codebases.

Visual Studio 2012 also introduces enhancements to the debugging experience, a critical aspect of software development. The IDE offers advanced debugging tools such as the ability to debug code across multiple processes, attach to remote processes, and inspect real-time values of variables and objects. The "IntelliTrace" feature is particularly noteworthy, as it provides a historical view of code execution, allowing developers to step back through previous states of the application to diagnose issues more effectively. This historical debugging capability can significantly accelerate the process of identifying and fixing bugs.

The IDE includes comprehensive support for version control systems, including integration with Microsoft Team Foundation Server (TFS) and Git. This integration facilitates collaborative development by allowing developers to manage source code, track changes, and coordinate with team members seamlessly. Visual Studio 2012's built-in tools for version control help streamline the workflow, reduce conflicts, and ensure that changes are properly documented and managed throughout the development cycle.

For web development, Visual Studio 2012 provides robust support for building and deploying web applications. It includes features for developing ASP.NET applications, designing user interfaces with HTML5 and CSS3, and leveraging JavaScript for interactive functionality. The IDE's integrated web development tools, such as the new web designer and improved support for client-side scripting, enhance the development process and make it easier to create responsive and feature-rich web applications.

In the realm of mobile development, Visual Studio 2012 offers tools for building Windows Phone applications, although it does not support the broader range of mobile platforms seen in later versions. The IDE provides an emulator for testing Windows Phone apps and tools for designing user interfaces specific to mobile devices. While support for mobile development has expanded in subsequent versions, Visual Studio 2012 laid the groundwork for creating applications in this space.

Testing is another critical area where Visual Studio 2012 shines. The IDE includes testing tools that support various types of testing, including unit tests, integration tests, and performance tests. The testing framework allows developers to write and execute tests directly within the IDE, view test results, and identify issues quickly. Integration with TFS also enables automated builds and test runs, supporting continuous integration and continuous deployment practices.

Visual Studio 2012's extensibility is another key feature, allowing developers to customize and extend the IDE through a rich ecosystem of extensions and plugins. The Visual Studio Marketplace offers a wide range of extensions that add new functionalities, integrate with third-party tools, and enhance the development experience. This extensibility enables developers to tailor the IDE to their specific needs and workflows, further improving productivity.

Overall, Visual Studio 2012 represents a powerful and versatile development environment that caters to a wide range of development needs. Its modern interface, advanced debugging and testing tools, support for multiple programming languages, and integration with version control systems make it a valuable asset for developers working on various types of applications. Despite the release of newer versions, Visual Studio 2012 remains a relevant and capable IDE for many development projects, providing a robust platform for building high-quality software.

**SQL SERVER 2012**

SQL Server 2012, developed by Microsoft, is a powerful relational database management system (RDBMS) that provides robust support for data storage, retrieval, and management. Released as part of the Microsoft SQL Server family, this version offers a range of advanced features and improvements over its predecessors, aimed at enhancing performance, scalability, and security for enterprise applications.

One of the standout features of SQL Server 2012 is its improved performance and scalability capabilities. The introduction of the Columnstore Index feature is particularly noteworthy. This technology significantly improves query performance by storing data in a columnar format rather than the traditional row-based format. This approach reduces I/O operations and accelerates data retrieval, which is especially beneficial for analytical workloads and large-scale data warehouses. The Columnstore Index enables faster querying and reporting, making it a valuable tool for businesses that rely on extensive data analysis and reporting.

SQL Server 2012 also enhances high availability and disaster recovery with several key improvements. The AlwaysOn Availability Groups feature offers a comprehensive solution for maintaining data availability and disaster recovery. It provides automatic failover and redundancy for databases, ensuring that applications remain operational even in the event of server failures. AlwaysOn Availability Groups support multiple secondary replicas, allowing for offloading of read-only queries and backups, thereby improving overall system performance and reliability. This feature is particularly important for mission-critical applications that require continuous data access and minimal downtime.

Security has been a major focus in SQL Server 2012, with several enhancements designed to protect data and ensure compliance with regulatory requirements. One of the significant improvements is the introduction of the new security model, which includes enhancements to data encryption and auditing. Transparent Data Encryption (TDE) provides encryption for the entire database, protecting data at rest without requiring changes to the application. This feature is essential for safeguarding sensitive information and meeting compliance standards. Additionally, SQL Server 2012 introduces enhanced auditing capabilities that allow for detailed tracking of database activities and changes, helping organizations maintain regulatory compliance and monitor data access.

The integration of SQL Server Data Tools (SSDT) represents another major advancement in SQL Server 2012. SSDT provides a unified development environment for database developers, allowing them to design, develop, and deploy database solutions within the same tool. The integration of database design and development with Visual Studio simplifies the workflow, making it easier to manage database projects and integrate them with application development. SSDT includes features such as schema comparison, data comparison, and database unit testing, which help improve the quality and reliability of database solutions.

SQL Server 2012 also introduces enhancements to business intelligence (BI) capabilities, making it a more powerful tool for data analysis and reporting. The SQL Server Analysis Services (SSAS) and SQL Server Reporting Services (SSRS) have been improved to provide more advanced features and better performance. SSAS now supports the creation of multidimensional and tabular models, allowing for more flexible and efficient data analysis. SSRS offers enhanced report design capabilities, including improved charting and visualization options, as well as support for interactive and drill-through reports. These enhancements enable organizations to gain deeper insights into their data and make more informed business decisions.

Another notable feature of SQL Server 2012 is its improved support for cloud integration. The introduction of SQL Server Integration Services (SSIS) enhancements allows for better data integration and migration between on-premises databases and cloud-based services. SQL Server 2012 provides tools and features that facilitate the seamless integration of data with Microsoft Azure, enabling organizations to leverage cloud resources for scalability, backup, and disaster recovery. This integration helps organizations extend their on-premises databases to the cloud, providing greater flexibility and resilience.

Performance monitoring and tuning have also been enhanced in SQL Server 2012. The introduction of the new Data Compression feature helps reduce the storage footprint of databases, improving I/O performance and reducing costs. The SQL Server Management Studio (SSMS) includes improved tools for performance monitoring, such as the new Activity Monitor and Data Collection, which provide valuable insights into database performance and help identify potential issues before they impact operations. These tools assist database administrators in maintaining optimal performance and ensuring efficient use of resources.

Overall, SQL Server 2012 represents a significant advancement in database management technology, offering a range of features designed to enhance performance, security, and manageability. Its improvements in high availability, disaster recovery, business intelligence, and cloud integration make it a robust and versatile platform for managing complex and large-scale data environments. Whether used for transactional processing, data warehousing, or business intelligence, SQL Server 2012 provides the tools and capabilities needed to support a wide range of data management needs, making it a valuable asset for organizations seeking to leverage their data for strategic advantage.

**MICROSOFT WORD**

Microsoft Word is a powerful word processing application developed by Microsoft. It is part of the Microsoft Office suite, which includes other productivity applications such as Excel, PowerPoint, and Outlook. Since its initial release in 1983, Microsoft Word has evolved significantly, becoming a ubiquitous tool for creating, editing, and formatting text documents.

One of the key features of Microsoft Word is its rich text formatting capabilities. Users can create documents with various font styles, sizes, and colors, and apply different text effects such as bold, italics, and underlining. The application supports complex document structures, allowing users to insert headings, subheadings, and sections to organize content effectively. Additionally, Word offers tools for managing document layout, including margin adjustments, text alignment, and column formatting, which are essential for producing professional-looking documents.

Microsoft Word provides extensive support for inserting and formatting multimedia elements within documents. Users can embed images, charts, tables, and graphs to enhance the visual appeal of their documents. The application also allows for the integration of other media types, such as audio and video files, making it possible to create multimedia-rich documents. Word’s advanced editing features enable users to crop, resize, and adjust the positioning of these elements, ensuring that they fit seamlessly within the document’s layout.

Another notable feature of Microsoft Word is its support for collaborative work. The application allows multiple users to work on a document simultaneously through its collaboration tools. Users can share documents via OneDrive or SharePoint, enabling real-time editing and feedback. The Track Changes feature facilitates the review process by highlighting edits and allowing users to accept or reject changes. Comments can be added to provide additional context or feedback, which is particularly useful for collaborative projects and document revisions.

Microsoft Word also includes powerful referencing and citation tools that are essential for academic and professional writing. The application supports various citation styles, including APA, MLA, and Chicago, and can automatically generate bibliographies and reference lists based on the citations included in the document. This feature streamlines the process of creating properly formatted citations and references, ensuring accuracy and consistency.

The application offers a range of templates and document styles to help users get started quickly. Whether creating a resume, business letter, report, or brochure, users can choose from a variety of pre-designed templates that suit their needs. These templates provide a structured starting point, which can be customized with specific content and formatting to meet individual requirements.

Word’s integration with other Microsoft Office applications enhances its functionality and versatility. For instance, users can import data from Excel spreadsheets directly into Word documents, or embed PowerPoint presentations and Outlook email content. This seamless integration allows for efficient cross-application workflows and helps maintain consistency across different types of documents.

For users concerned about document accessibility, Microsoft Word includes features designed to make content accessible to individuals with disabilities. The Accessibility Checker tool helps identify and address potential accessibility issues, ensuring that documents are usable by all individuals, including those using screen readers or other assistive technologies.

Security features in Microsoft Word include password protection and encryption options, which help safeguard sensitive information. Users can restrict access to documents by setting passwords, and sensitive content can be encrypted to prevent unauthorized access. These features are particularly important for handling confidential or proprietary information.

Additionally, Microsoft Word supports a wide range of file formats, making it versatile for sharing documents with users who may not have the latest version of the application. Users can save documents in formats such as PDF, RTF, and HTML, among others, ensuring compatibility with various platforms and devices.

Overall, Microsoft Word remains one of the most widely used word processing tools due to its comprehensive feature set and ease of use. Its robust formatting, collaboration, and multimedia capabilities make it an essential application for personal, academic, and professional document creation. As technology continues to evolve, Microsoft Word adapts by incorporating new features and improvements, ensuring that it remains a valuable tool for a diverse range of users.

**MICROSOFT POWERPOINT**

Microsoft PowerPoint is a widely used presentation software developed by Microsoft, integral to the Microsoft Office suite. Since its initial release in 1987, PowerPoint has become a standard tool for creating dynamic and visually engaging presentations. It enables users to design, organize, and display content in a slide-based format, making it an essential application for various professional, educational, and personal uses.

PowerPoint’s core functionality revolves around the creation of slides, which serve as individual pages in a presentation. Each slide can contain a combination of text, images, charts, videos, and other multimedia elements, allowing users to present information in a structured and visually appealing manner. The software offers a range of pre-designed templates and themes that provide a cohesive and professional look, simplifying the design process and ensuring consistency throughout the presentation.

One of PowerPoint’s notable features is its robust set of design and formatting tools. Users can customize slides with a variety of text styles, fonts, colors, and effects to enhance the visual appeal of their content. The application includes tools for adding transitions and animations, which can be applied to individual slides or specific elements within a slide. These effects help to capture the audience's attention and emphasize key points, making the presentation more engaging and memorable.

PowerPoint also supports the integration of multimedia elements. Users can embed images, audio clips, and video files directly into their presentations, creating a rich multimedia experience. The software includes tools for editing and formatting these elements, such as cropping images, adjusting video playback settings, and synchronizing audio with slide transitions. This multimedia capability allows for a more interactive and immersive presentation, catering to various content types and audience preferences.

For those requiring advanced data visualization, PowerPoint provides tools to insert and customize charts and graphs. Users can create charts based on data from Excel or manually input data into PowerPoint, enabling the presentation of complex information in a clear and understandable format. The application supports various chart types, including bar, line, pie, and scatter plots, and allows for customization to match the presentation’s design and style.

Collaboration and sharing features are integral to PowerPoint’s functionality. Users can collaborate on presentations in real-time through cloud services like OneDrive or SharePoint, allowing multiple people to work on the same file simultaneously. Comments and feedback can be added directly to the slides, facilitating the review and editing process. Additionally, PowerPoint offers options for sharing presentations in various formats, such as PDF or video, and provides tools for creating online presentations or exporting slides as image files.

PowerPoint includes a presenter view feature that enhances the delivery of presentations. This view allows presenters to see their speaker notes, a timer, and a preview of upcoming slides on their screen while the audience views only the current slide. This feature helps presenters stay organized and maintain their flow during a presentation, ensuring they cover all key points effectively.

For educational and training purposes, PowerPoint offers features for creating interactive presentations. Users can incorporate hyperlinks, action buttons, and interactive elements to create quizzes, tutorials, and other interactive content. This interactivity makes PowerPoint a versatile tool for educational settings, enabling instructors to create engaging and interactive learning materials.

Security and protection features in PowerPoint allow users to safeguard their presentations. Users can set passwords to restrict access to files, control editing permissions, and prevent unauthorized modifications. This level of security is essential for protecting sensitive or proprietary content and ensuring that only authorized individuals can make changes to the presentation.

PowerPoint’s integration with other Microsoft Office applications enhances its functionality. For example, users can import data from Excel to create charts, embed content from Word documents, or incorporate information from Outlook emails. This seamless integration supports efficient workflows and ensures consistency across various types of documents and presentations.

Overall, Microsoft PowerPoint remains a powerful and versatile tool for creating presentations that effectively communicate information and ideas. Its comprehensive design features, multimedia support, collaboration capabilities, and integration with other Microsoft Office applications make it an invaluable resource for a wide range of users, from business professionals and educators to students and individuals creating personal presentations. As technology advances, PowerPoint continues to evolve, incorporating new features and improvements to meet the changing needs of its users.

**CHAPTER 4**

**SYSTEM DESIGN**

Design is the first step into the development phase for any engineered product or system. Design is a creative process. A good design is the key to effective system. The term “design” is defined as “the process of applying various techniques and principles for the purpose of defining a process or a system in sufficient detail to permit its physical realization”. It may be defined as a process of applying various techniques and principles for the purpose of defining a device, a process or a system in sufficient detail to permit its physical realization. Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm that is used. The system design develops the architectural detail required to build a system or product. As in the case of any systematic approach, this software too has undergone the best possible design phase fine tuning all efficiency, performance and accuracy levels. The design phase is a transition from a user oriented document to a document to the programmers or database personnel. System design goes through two phases of development: Logical and Physical Design.

**LOGICAL DESIGN**

The logical flow of the system and define the boundaries of the system. In the Logical Designing we had done the following activities

* Reviews the current physical system – its data flows, file content, volumes, frequencies etc.
* Prepares output specifications – that is, determines the format, content and frequency of reports.
* Prepares input specifications – format, content and most of the input functions.
* Prepares edit, security and control specifications.
* Specifies the implementation plan.
* Prepares a logical design walk through of the information flow, output, input, controls and implementation plan.
* Reviews benefits, costs, target dates and system constraints.

**PHYSICAL DESIGN**

Physical system produces the working systems by define the design specifications that tell the programmers exactly what the candidate system must do. In the Physical Designing Phase we had the following activities

* Design the physical system.
* Specify input and output media.
* Design the database and specify backup procedures.
* Design physical information flow through the system and a physical design

Walk through.

* Plan system implementation.
* Prepare a conversion schedule and target date.
* Determine training procedures, courses and timetable.
* Devise a test and implementation plan and specify any new hardware/software.
* Update benefits , costs , conversion date and system constraints

**4.1 INPUT DESIGN**

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data into a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. In the Input designing we had considered the following things

* What data should be given as input?
* How the data should be arranged or coded?
* The dialog to guide the operating personnel in providing input.
* Methods for preparing input validations and steps to follow when errors occur.
* It is achieved by creating user friendly screens for the data entry to handle large volume of data.
* The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities
* Input design is the process of converting a user oriented description of the input into a computer based system .This design is important to avoid errors in the data Input process and show the correct direction to the management for getting correct information from the computerized system.

When the data is entered it will check for its validity. Data can be entered for its Validity. Mostly input data can be selected from a list of data items . Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input is to create an input lay out that is easy to follow. For example in user registration form all the fields except some optional fields must be enter, otherwise display appropriate error message.

**4.2 OUTPUT DESIGN**

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the user and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the systems relationships to help user decision making.

Designing computer output should proceed in an organized well thought manner, the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. In this phase we had done the followings

* Identify the specific output that is needed to meet the requirements.
* Select methods for presenting information.
* Create document, report or other format that contain information produced by the system.

**4.3 DATA FLOW DIAGRAM**

A Data Flow Diagram (DFD) is a powerful tool used in the field of systems analysis and design to visually represent the flow of data within a system. It is used to map out the processes, data stores, data flows, and external entities involved in the system, providing a clear and concise overview of how information moves and is transformed within the system. DFDs help stakeholders understand the system's functionality, identify potential inefficiencies, and design effective solutions.

At the core of a DFD are processes, which represent the activities or functions that transform incoming data into outgoing data. These processes are depicted as circles or rounded rectangles and are essential in showing how data is manipulated within the system. Each process must have at least one input and one output, ensuring that the flow of data is well-defined and logical. For example, a process could represent the calculation of an invoice total or the validation of a user login.

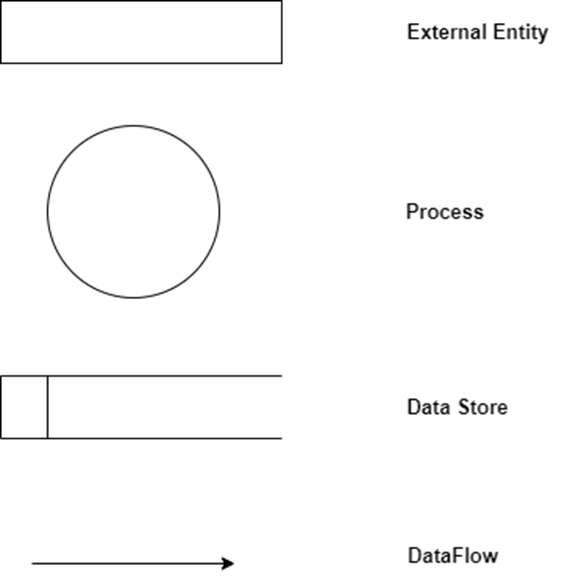
Data stores are another critical component of a DFD. These are represented by open-ended rectangles or two parallel lines and signify where data is stored within the system, such as databases, files, or any other form of data storage. Data stores are crucial for understanding where information resides and how it is accessed or modified by different processes. For instance, a customer database might store user details that are retrieved and updated by various processes within the system.

Data flows are depicted as arrows and represent the movement of data between processes, data stores, and external entities. These flows are typically labeled to indicate the specific data being transferred, making it easier to trace how information travels through the system. Understanding data flows is vital for identifying how data inputs are transformed into outputs, helping to ensure that all necessary data paths are accounted for.

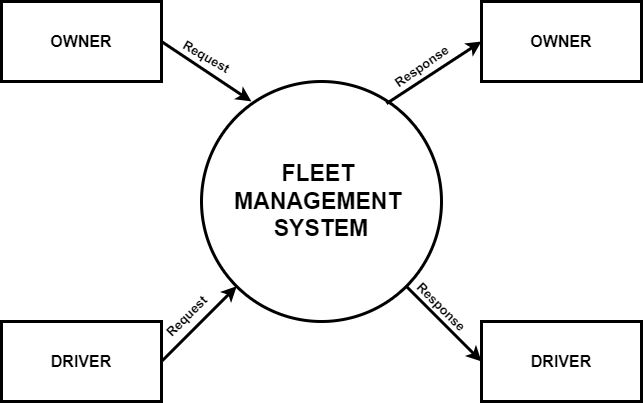
Finally, external entities are represented by rectangles and denote outside actors that interact with the system, such as users, other systems, or external organizations. These entities provide input to the system or receive output from it, making them a crucial part of understanding the system’s boundaries and interfaces.

DFDs can be developed at different levels of detail. The context level (Level 0) DFD offers a broad overview of the entire system, showing it as a single process with its interactions with external entities. This high-level view is useful for understanding the system's overall scope and its external relationships. As the DFD is further decomposed into Level 1 and beyond, more detailed processes and data flows are revealed, providing a deeper understanding of the system's internal workings. This progressive detailing helps in refining the system's design and ensuring that all functional requirements are met.

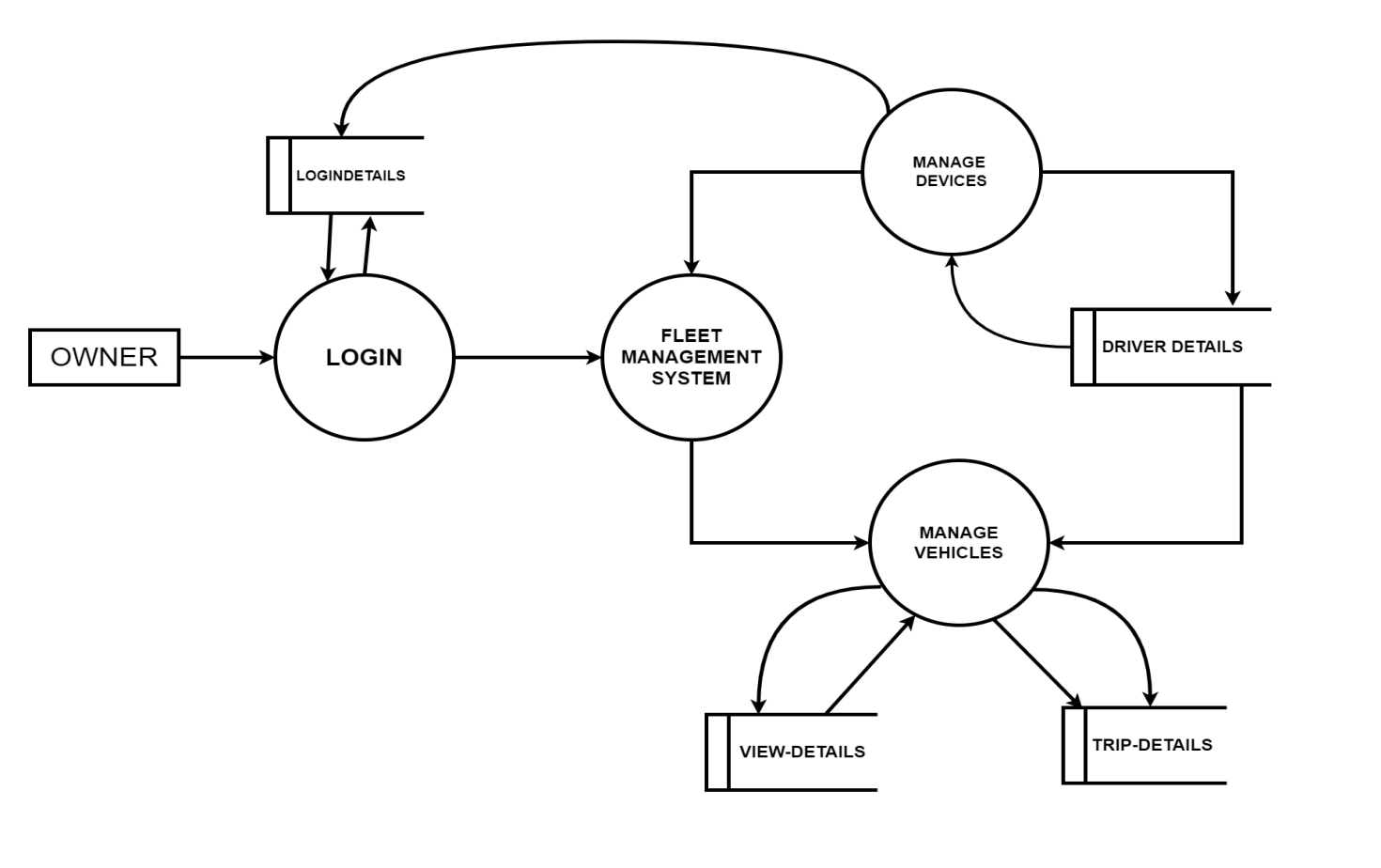
In summary, a DFD is an invaluable tool for visualizing and analyzing the flow of data within a system. By breaking down the system into its fundamental components—processes, data stores, data flows, and external entities—a DFD provides a clear and organized representation of how information is handled. This makes it easier for stakeholders to understand the system's functionality, identify potential issues, and design effective solutions that meet the system’s requirements.

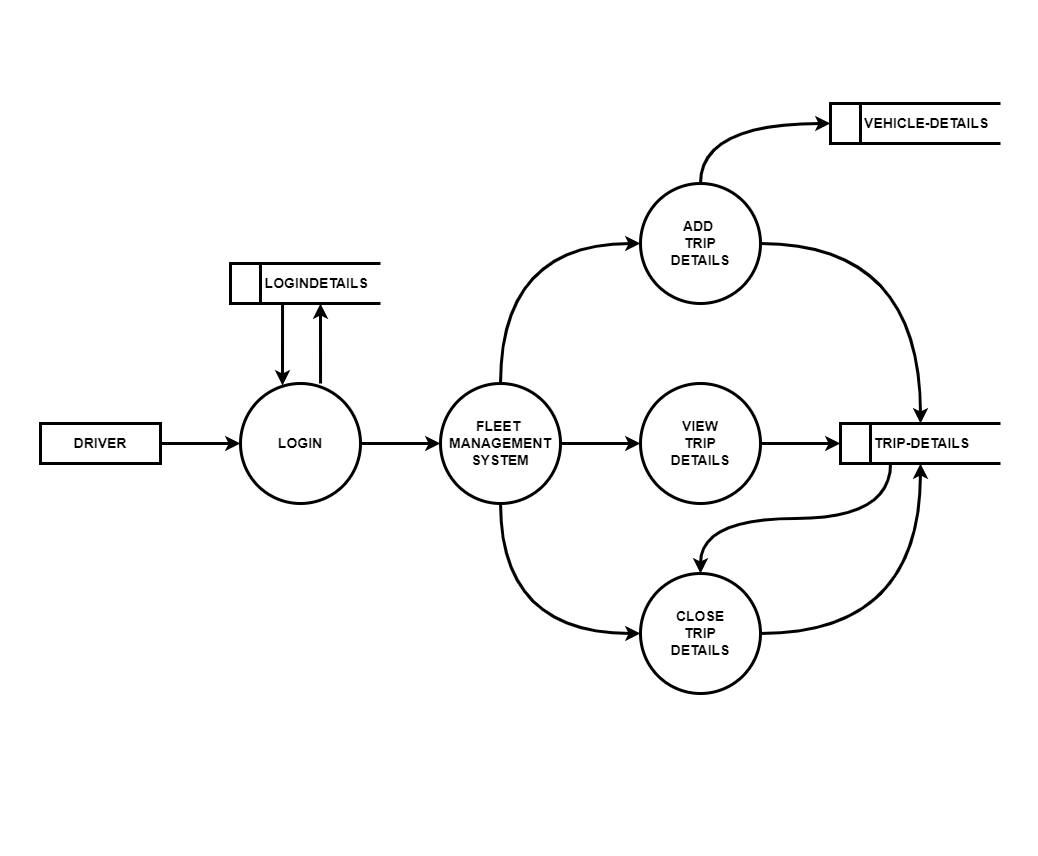
****

**Level 0 (Context Diagram )**

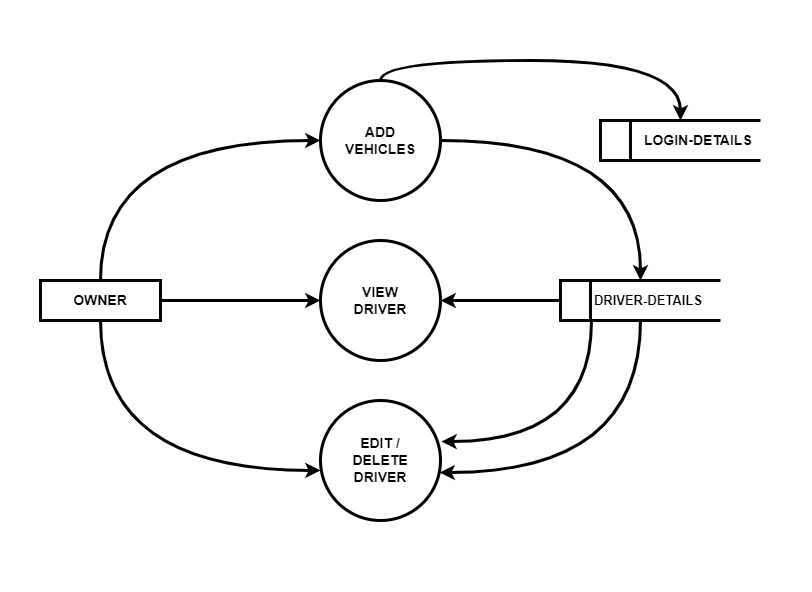
****

**Level 1 (Owner)**

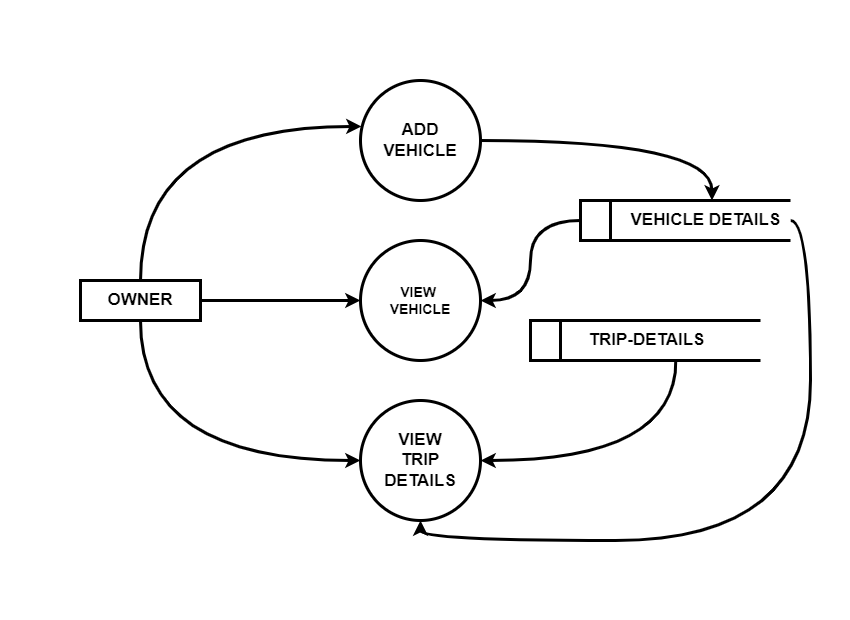
**Level 1 (Driver)**

****

**Level 2 (Owner manage driver )**

****

**Level 2 (Driver manage vehicles)**

****

**4.4 DATABASE DESIGN**

Database Design is an important activity in design. The efficiency of the system lies in the efficiency of the database. The database design consists of predetermining number of tables that are to be used and fields that are to be used in each table. The efficiency of the system to retrieve the appropriate data depends on how the fields are specified and the coding mostly depends on format of the database.

**Table Design**

The tables in the system are

**1)Login Details**

|  |  |  |
| --- | --- | --- |
| Column Name | Datatype | Constraint |
| uid | int | Primary key |
| username | Varchar(30) | Not null |
| password | Varchar(30) | Not null |
| usertype | Varchar(30) | Not null |

**2)** **Driver Details**

|  |  |  |
| --- | --- | --- |
| Column Name | Datatype | Constraint |
| Did | int | Primary key |
| Name | Varchar(30) | Not null |
| Age | Varchar(30) | Not null |
| Mobile | int | Not null |
| Gender | Varchar(30) | Not null |
| Image | Varchar(30) | Not null |

**3)** **Vechicle Details**

|  |  |  |
| --- | --- | --- |
| Column Name | Datatype | Constraint |
| Vid | int | Primary key |
| Model | Varchar(30) | Not null |
| Regno | int | Not null |
| Image | Varchar(30) | Not null |

**4)** **Trip Details**

|  |  |  |
| --- | --- | --- |
| Column Name | Datatype | Constraint |
| Tid | int | Primary key |
| Vid | Varchar(30) | Not null |
| Did | int | Not null |
| Starting\_Reading | Varchar(30) | Not null |
| End\_Reading | Varchar(30) | Not null |
| Kilometers | int | Not null |
| Destination | Varchar(Max) | Not null |
| Start date | Int | Not null |
| Toll | Int | Not null |
| Fare | Int | Not null |
| Status | Varchar(30) | Not null |
| Total | Int | Not null |

**5) Assign Vehicles Details**

|  |  |  |
| --- | --- | --- |
| Column Name | Datatype | Constraint |
| did | int | Primary key |
| name | Varchar(30) | Not null |
| vid | int | Not null |
| regno | Varchar(30) | Not null |

**4.5 USE CASE DIAGRAM**

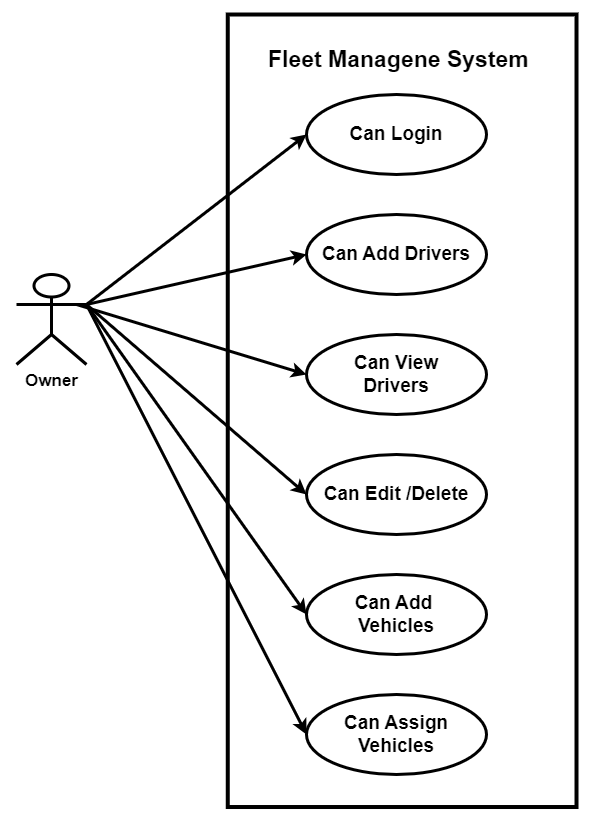
A use case diagram is a type of behavioral diagram in the Unified Modeling Language (UML) that visually represents the interactions between external entities (known as actors) and the system under consideration. It captures the functional requirements of a system by illustrating how different users will interact with it to achieve specific goals. The diagram helps to identify and define the various functions, or "use cases," that the system must perform in response to these interactions.

In a use case diagram, actors represent the different roles played by external entities, which could be human users, external systems, or devices that interact with the system. These actors are depicted as simple stick figures. The diagram also includes use cases, which are depicted as ovals or ellipses within the system boundary, representing the specific actions or services the system performs in response to an actor's interaction. The system boundary, usually represented as a rectangle, delineates the scope of the system, indicating which use cases fall within the system's responsibilities.

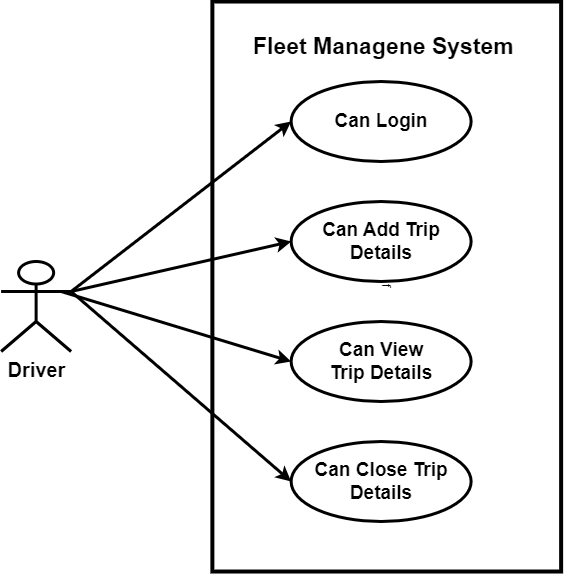
The relationships in a use case diagram show how actors are connected to the use cases they participate in. These relationships can take several forms, such as associations (which connect actors to use cases), inclusions (where one use case always involves another), and extensions (where one use case optionally adds behavior to another). There is also the possibility of generalization, where actors or use cases are shown in a hierarchical relationship, indicating that one is a specialized version of the other.

Overall, a use case diagram serves as a high-level representation of the system's functional requirements, offering a clear and concise way to understand the interactions between the system and its users or other external entities. This makes it an essential tool in both the design and communication phases of software development.

**Owner Use Case Diagram**



**Driver Use Case Diagram**



**CHAPTER 5**

**SYSTEM TESTING**

System testing is actually a series of different tests whose primary purpose is to fully exercise the computer based system. The common view of testing held by users is that it is performed to prove that there are no errors in the system. All though each test has a different purpose, all work should verify that all system elements have properly integrated and perform allocated functions. System tests are test designed to verify that the finished system meets its requirements. Once the application has undergone system testing, it may be put through actual use within the development organization. The purpose of this is to test the system under realistic condition, but with understanding and forgiving users.

Software testing is a critical element of software quality assurance and represents the ultimate review of specifications, design and coding. Testing includes verification that the entire system works properly. The programmers facilitate testing by coding as clearly as possible. Test case design focuses on a set of techniques for the creation of test cases that meet overall testing objectives.

Testing objectives are:

* Testing is a process of executing a program with the intent of finding an error.
* A good test case is one that has a high probability of finding an unconsidered error.
* A successful test is one that uncovers an undiscovered error.

Testing demonstrates that software functions appear to be working according to specifications that performs requirements has been met. In addition data collected as testing is conducted provide a good indication of software reliability and some indication of software quality as a while.

But, testing cannot show the absence of defects, it can only show that software defects are present. The debugging process is the most unpredictable part of the testing process.

**5.1 TEST PLAN**

A test plan implies a series of desired course of action to be followed in accomplishing various testing methods. The Test Plan acts as a blue print for the action that is to be followed. The software engineers create a computer program, its documentation and related data structures. The software developers is always responsible for testing the individual units of the programs, ensuring that each performs the function for which it was designed. There is an independent test group (ITG) which is to remove the inherent problems associated with letting the builder to test the thing that has been built. The specific objectives of testing should be stated in measurable terms. So that the mean time to failure, the cost to find and fix the defects, remaining defect density or frequency of occurrence and test work-hours per regression test all should be stated within the test plan. The levels of testing include:

* Unit Testing
* Integration Testing
* System Testing
* Output Testing

**5.2 UNIT TESTING**

The unit testing forces verification of errors on the smallest unit of the software design, the module. The unit testing tests the interface, local data, structure, boundary conditions, independent paths, error handling paths. In this testing each module is tested individually. This testing is carried out during the programming stage itself. This is essential for verification of the code produced during the coding phase, and hence the goal is to test the internal logic of the modules.

In this website the unit testing carried out by individual web pages. The following test procedures are used.

* All the links are available for navigation.
* For login process first inputting invalid username and password and check that showing error messages when submit the form.
* In registration each fields are check with valid and invalid data.
* Check the output data that must be produced according to the input data.
* Check the security of user’s private data.

**5.3 INTEGRATION TESTING**

Integration testing is a systematic technique for constructing the program structure while at the same time conducting test to uncover errors associated within the interface. The objective is to take the unit tested modules and then testing them as a whole. The goal here is to see if the modules can be integrated properly, emphasize being on testing interfaces between modules.

* All the modules are integrated for this testing.
* Test the system by navigating from all available links to another.
* All the functionality should maintain at a single run.

**5.4 SYSTEM TESTING**

This is the final step in testing. In this the entire system was tested as a whole with all forms, code, modules and class modules. This form of testing is popularly known as Black Box testing or System Testing.

Black Box testing method focuses on the functional requirements of the software. That is, Black Box testing enables the software engineer to derive sets of input conditions that will fully exercise all functional requirements for a program.

Black Box testing attempts to find errors in the following categories; incorrect or missing functions, interface errors, errors in data structures or external data access, performance errors and initialization errors and termination errors.

**5.5 OUTPUT TESTING**

User acceptance of a System is a key factor for the Success of a system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system users at the time of development and making changes whenever required. This testing is done to find out whether the user gets the real output for the input entered.

The system considered is tested for user acceptance; here it should satisfy the firm’s need. The software should keep in touch with perspective system; user at the time of developing and making changes whenever required. This done with respect to the following points

* Input Screen Designs
* Output Screen Designs

The above testing is done taking various kinds of test data. Preparation of test data plays a vital role in the system testing. After preparing the test data, the system under study is tested using that test data. While testing the system by which test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

In my project “Fully Automated Services of Transport Department” testing was conducted at every step. Initially each module was tested separately to check whether they gave the desired output for the given input. The forms used to enter data by user were validated and appropriate error messages were displayed if incorrect data was entered. Once the data was entered correctly, the processing was done and testing was done to check whether the correct output was obtained. Once the test cases were conducted successfully for each module, the modules were integrated together as a single system. After integration, the test cases were again applied to check whether the entire system as a whole produced the desired output. At times, the test cases failed and the shortcomings were noted down and appropriate corrections were done. Once the integration testing was performed correctly, output testing was done and it did not result in any change or correction in the system. Black box testing and white box testing was also conducted successfully. All the loops, decisions, relations were executed at least once before giving it to the users for testing. In black box testing, it was checked whether the data in the proper format was stored in the database or not. Also, it was checked whether the interfaces were working properly or not. On successful completion of these tests, the system was then given to undergo user acceptance testing where the users entered test data to check whether the correct output was obtained. The users were satisfied with the output and thus the testing phase was completed successfully.

**CHAPTER 6**

**IMPLEMENTATION AND RESULT**

**6.1 Implementation**

**1. System Architecture**

* **Frontend:** Utilize C#.NET with either Windows Forms or WPF to create a user-friendly, responsive graphical interface.
* **Backend:** Use C#.NET for business logic to separate concerns between the UI and data handling.
* **Database:** Implement SQL Server to ensure robust data storage, efficient queries, and management.

**2. Database Design**

Design a relational database structure to handle key fleet operations, including driver details, vehicle assignments, and trip records.

**Key Tables:**

* **Users**: Store user credentials and role information (Admin/Driver).
* **Drivers**: Maintain driver profiles and status.
* **Vehicles**: Track the status and details of each vehicle.
* **Trips**: Log assigned trips with timestamps, fares, and completion status.

**3. User Authentication**

* Implement a login system with username and password fields.
* Use **s**ecure password hashing techniques, such as SHA-256, to protect credentials.
* Verify user credentials against the database and grant role-based access accordingly.

**4.Role-Based Access Control**

Define user roles with specific access privileges:

* **Admin**:
  + Manage driver and vehicle records.
  + Assign and monitor trips.
  + View fare summaries and operational reports.
* **Driver**:
  + View assigned trips.
  + Update trip status (e.g., In Progress, Completed).
  + Receive trip notifications and updates via the dashboard.

**5. Trip and Vehicle Management**

Design functionalities to handle trip creation, vehicle assignment, and fare management:

* **Trip Assignment**: Admins assign available vehicles to drivers.
* **Vehicle Status**: Mark vehicles as "In Service" or "Available" based on assignments.
* **Trip Closure**: Drivers update trip status upon completion.

**6. Data Security**

* Encrypt sensitive information, such as user passwords, using AES encryption.
* Use SHA-256 hashing for login passwords to prevent tampering or unauthorized access.
* Secure database access by using parameterized queries to prevent SQL injection attacks.

**7. User Interface Design**

* **Admin Dashboard**:
  + Manage drivers, vehicles, and trips.
  + View performance metrics and reports.
* **Driver Dashboard**:
  + Display assigned trips and allow trip status updates.
  + Notifications and trip reminders are accessible from the dashboard.
* **Forms and Navigation**:
  + Use clear labels and easy-to-navigate forms for all user interactions (e.g., login, trip assignment, vehicle updates).

**8. Testing and Quality Assurance**

* **Unit Testing**: Verify individual functions, such as trip assignment and login authentication.
* **Integration Testing**: Ensure different modules work together seamlessly (e.g., Admin assigning trips, Drivers updating status).
* **User Acceptance Testing**: Test the entire flow to ensure it meets user expectations.
* Ensure compliance with security standards (e.g., encryption and secure authentication).

**9. Deployment**

* Prepare the production environment by mirroring the development setup to minimize deployment issues.
* Configure SQL Server to ensure proper databasesecurity (e.g., access control, backups).
* Install the Fleet Management System on designated Admin and Driver systems.

**10. Documentation**

* Develop user manuals to outline system functionalities, including:
  + Login and account management.
  + Trip assignment and closure procedures.
* Provide technical documentation for developers detailing:
  + System architecture and database schema.
  + Code structure and module descriptions to facilitate future maintenance.

**6.2 RESULT**

**6.2.1 SOURCE CODE**

**Login Page**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SqlClient;

namespace fleetmanagementsystem

{

public partial class login : Form

{

SqlConnection con = new SqlConnection("server=DESKTOP-PBIP1KS; database=fleet; Integrated Security=true; ");

public login()

{

InitializeComponent();

}

protected override CreateParams CreateParams

{

get

{

CreateParams handleParam = base.CreateParams;

handleParam.ExStyle |= 0x02000000; // WS\_EX\_COMPOSITED

return handleParam;

}

}

private void login\_Load(object sender, EventArgs e)

{

}

private void button1\_Click(object sender, EventArgs e)

{

string query = "Select \* from logindetails where username='" + textBox1.Text + "' and password='" + textBox2.Text + "'";

con.Open();

SqlCommand cmd = new SqlCommand(query, con);

SqlDataReader dr = cmd.ExecuteReader();

if (dr.Read())

{

string utype = dr[2].ToString();

if (utype == "owner")

{

Program.username = dr[0].ToString();

ownerlogin obj = new ownerlogin(dr[0].ToString());

dr.Close();

ActiveForm.Hide();

obj.Show();

}

else if (utype == "driver")

{

Program.username = dr[0].ToString();

dr.Close();

string query1 = "Select \* from driverdetails where name='" + textBox1.Text + "'";

SqlCommand cmd1 = new SqlCommand(query1, con);

SqlDataReader dr1 = cmd1.ExecuteReader();

if (dr1.Read())

{

Program.did = dr1[0].ToString();

Program.dname = dr1[1].ToString();

driverlogin obj1 = new driverlogin(dr1[1].ToString());

ActiveForm.Hide();

obj1.Show();

}

}

else

{

MessageBox.Show("Incorrect username or password ");

textBox1.Text = "";

textBox2.Text = "";

}

con.Close();

}

else

{

con.Close();

}

}

private void button2\_Click(object sender, EventArgs e)

{

Application.Exit();

}

private void pictureBox1\_Click(object sender, EventArgs e)

{

}

}

}

**Owner login Page:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SqlClient;

namespace fleetmanagementsystem

{

public partial class ownerlogin : Form

{

public ownerlogin(string username\_from\_login)

{

InitializeComponent();

greeting();

}

public void greeting()

{

int hour = DateTime.Now.Hour;

// Determine greeting based on the hour

if (hour >= 0 && hour < 12)

{

label1.Text="Good Morning "+Program.username ;

}

else if (hour >= 12 && hour < 17)

{

label1.Text = "Good Afternoon " + Program.username;

}

else

{

label1.Text = "Good Evening " + Program.username;

}

}

private void ownerlogin\_Load(object sender, EventArgs e)

{

}

private void toolStripButton1\_Click(object sender, EventArgs e)

{

driverregistration obj = new driverregistration();

obj.Show();

}

private void toolStripButton2\_Click(object sender, EventArgs e)

{

driverdetailsview obj = new driverdetailsview();

obj.Show();

}

private void toolStripButton6\_Click(object sender, EventArgs e)

{

addvehicles obj = new addvehicles();

obj.Show();

}

private void toolStripButton3\_Click(object sender, EventArgs e)

{

updatedetails obj = new updatedetails();

obj.Show();

}

private void toolStripButton5\_Click(object sender, EventArgs e)

{

assign\_vechicles obj = new assign\_vechicles();

obj.Show();

}

private void label1\_Click(object sender, EventArgs e)

{

}

private void panel1\_Paint(object sender, PaintEventArgs e)

{

}

private void toolStrip1\_ItemClicked(object sender, ToolStripItemClickedEventArgs e)

{

}

private void toolStripButton4\_Click(object sender, EventArgs e)

{

}

private void toolStripButton4\_Click\_1(object sender, EventArgs e)

{

Application.Exit();

}

private void toolStripButton7\_Click(object sender, EventArgs e)

{

ownerpasswordchange obj = new ownerpasswordchange();

obj.Show();

}

private void toolStripButton8\_Click(object sender, EventArgs e)

{

}

private void toolStripButton8\_Click\_1(object sender, EventArgs e)

{

OwnerViewTrips obj = new OwnerViewTrips();

obj.Show();

}

}

}

**Driver Registration Form:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SqlClient;

using System.IO;

namespace fleetmanagementsystem

{

public partial class driverregistration : Form

{

SqlConnection con = new SqlConnection("server=DESKTOP-PBIP1KS; database=fleet; Integrated Security=true; ");

public driverregistration()

{

InitializeComponent();

gettid();

}

public void gettid()

{

con.Open();

string query = "select isnull(MAX(did),10100)+1 from driverdetails";

SqlCommand cmd = new SqlCommand(query, con);

SqlDataReader dr = cmd.ExecuteReader();

if (dr.Read())

{

textBox1.Text = dr[0].ToString();

}

con.Close();

}

private void driverregistration\_Load(object sender, EventArgs e)

{

}

private void button1\_Click(object sender, EventArgs e)

{

if (textBox1.Text == "" || textBox2.Text == "" || textBox3.Text == "" || textBox4.Text == "" || textBox5.Text == "" || textBox7.Text == "" || comboBox1.Text == "")

{

MessageBox.Show("Fill");

return;

}

string shin1 = "select \* from logindetails where username='" + textBox2.Text + "'";

SqlCommand cmd2 = new SqlCommand(shin1, con);

con.Open();

SqlDataReader shin = cmd2.ExecuteReader();

if (shin.Read())

{

MessageBox.Show("username already exist");

return;

}

con.Close();

string extension = Path.GetExtension(textBox6.Text);

string filename = textBox1.Text + extension;

string actualpath = Application.StartupPath + "\\driverpics\\" + filename;

string virtualpath = "\\driverpics\\" + filename;

File.Copy(textBox6.Text,actualpath, overwrite: true);

string q1 = "insert into driverdetails values('" + textBox1.Text + "','" + textBox2.Text + "','" + textBox3.Text + "','" + textBox4.Text + "','" + textBox5.Text + "','" + comboBox1.Text + "','"+virtualpath+"')";

SqlCommand cmd = new SqlCommand(q1, con);

con.Open();

cmd.ExecuteNonQuery();

con.Close();

string usertype = "driver";

string q2 = "insert into logindetails values('" + textBox2.Text + "','" + textBox7.Text + "','" + usertype + "')";

SqlCommand cmd1 = new SqlCommand(q2, con);

con.Open();

if (cmd1.ExecuteNonQuery() > 0)

{

MessageBox.Show("Driver Created successfully");

textBox1.Text = (Convert.ToInt32(textBox1.Text) + 1).ToString();

textBox5.Text = "";

textBox4.Text = "";

textBox2.Text = "";

textBox3.Text = "";

textBox7.Text = "";

comboBox1.Text = "";

}

con.Close();

this.Close();

}

private void label1\_Click(object sender, EventArgs e)

{

}

private void button2\_Click(object sender, EventArgs e)

{

using (OpenFileDialog openFileDialog = new OpenFileDialog())

{

openFileDialog.Title = "Select an Image File";

openFileDialog.Filter = "Image Files|\*.jpg;\*.jpeg;\*.png;\*.bmp;\*.gif|All Files|\*.\*";

// Show the dialog and check if the user clicked OK

if (openFileDialog.ShowDialog() == DialogResult.OK)

{

textBox6.Text = openFileDialog.FileName;

pictureBox1.ImageLocation = openFileDialog.FileName;

}

}

}

private void pictureBox1\_Click(object sender, EventArgs e)

{

}

private void pictureBox2\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

**Vechicle Registration Form:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SqlClient;

using System.IO;

namespace fleetmanagementsystem

{

public partial class addvehicles : Form

{

private Dictionary<string, List<string>> vehicleModels = new Dictionary<string, List<string>>();

SqlConnection con = new SqlConnection("server=DESKTOP-PBIP1KS; database=fleet; Integrated Security=true; ");

public addvehicles()

{

InitializeComponent();

comboBox1.Items.Add("Light-Duty");

comboBox1.Items.Add("Medium-Duty");

comboBox1.Items.Add("Medium-Duty");

comboBox1.Items.Add("Heavy-Duty");

comboBox1.Items.Add("Service Vehicles");

// Add sub-items for each vehicle type to the dictionary

vehicleModels.Add("Light-Duty", new List<string> { "Ford Transit", "Mercedes-Benz Sprinter", "Ram ProMaster" });

vehicleModels.Add("Medium-Duty", new List<string> { "Isuzu N-Series", "Hino 195", "Freightliner M2 106" });

vehicleModels.Add("Heavy-Duty", new List<string> { "Freightliner Cascadia", "Volvo VNL", "Kenworth T680" });

vehicleModels.Add("Service Vehicles", new List<string> { "Ford F-550", "Chevrolet Silverado 3500HD", "Ram 5500" });

gettid();

}

public void gettid()

{

con.Open();

string query = "select isnull(MAX(vid),30100)+1 from vehiclesdetails";

SqlCommand cmd = new SqlCommand(query, con);

SqlDataReader dr = cmd.ExecuteReader();

if (dr.Read())

{

textBox1.Text = dr[0].ToString();

}

con.Close();

}

private void button1\_Click(object sender, EventArgs e)

{

string extension = Path.GetExtension(textBox6.Text);

string filename = textBox1.Text + extension;

string actualpath = Application.StartupPath + "\\vehiclepicss\\" + filename;

string virtualpath = "\\vehiclepicss\\" + filename;

File.Copy(textBox6.Text, actualpath, overwrite: true);

if (textBox1.Text == "" || textBox2.Text == "" || comboBox1.Text == "" || comboBox2.Text == "")

{

MessageBox.Show("Fill");

return;

}

string q1 = "insert into vehiclesdetails values('" + textBox1.Text + "','" + comboBox1.Text + "','" + comboBox2.Text + "','" + textBox2.Text + "','"+virtualpath+"')";

SqlCommand cmd = new SqlCommand(q1, con);

con.Open();

if (cmd.ExecuteNonQuery() > 0)

{

MessageBox.Show("Vechicle Added successfully");

textBox1.Text = (Convert.ToInt32(textBox1.Text) + 1).ToString();

comboBox2.Text = "";

textBox2.Text = "";

comboBox1.Text = "";

}

con.Close();

this.Close();

}

private void comboBox2\_SelectedIndexChanged(object sender, EventArgs e)

{

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

comboBox2.Items.Clear();

// Get selected item from ComboBox1

string selectedVehicleType = comboBox1.SelectedItem.ToString();

// Populate ComboBox2 with the corresponding sub-items

if (vehicleModels.ContainsKey(selectedVehicleType))

{

comboBox2.Items.AddRange(vehicleModels[selectedVehicleType].ToArray());

}

// Optionally, select the first item in ComboBox2 automatically

if (comboBox2.Items.Count > 0)

{

comboBox2.SelectedIndex = 0;

}

}

private void label5\_Click(object sender, EventArgs e)

{

}

private void label1\_Click(object sender, EventArgs e)

{

}

private void label2\_Click(object sender, EventArgs e)

{

}

private void label4\_Click(object sender, EventArgs e)

{

}

private void label3\_Click(object sender, EventArgs e)

{

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

}

private void textBox2\_TextChanged(object sender, EventArgs e)

{

}

private void comboBox3\_SelectedIndexChanged(object sender, EventArgs e)

{

}

private void button2\_Click(object sender, EventArgs e)

{

using (OpenFileDialog openFileDialog = new OpenFileDialog())

{

openFileDialog.Title = "Select an Image File";

openFileDialog.Filter = "Image Files|\*.jpg;\*.jpeg;\*.png;\*.bmp;\*.gif|All Files|\*.\*";

// Show the dialog and check if the user clicked OK

if (openFileDialog.ShowDialog() == DialogResult.OK)

{

textBox6.Text = openFileDialog.FileName;

pictureBox1.ImageLocation = openFileDialog.FileName;

}

}

}

private void addvehicles\_Load(object sender, EventArgs e)

{

}

}

}

**Assign Vechicles Form:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SqlClient;

using System.IO;

namespace fleetmanagementsystem

{

public partial class assign\_vechicles : Form

{

SqlConnection con = new SqlConnection("server=DESKTOP-PBIP1KS; database=fleet; Integrated Security=true; ");

public assign\_vechicles()

{

InitializeComponent();

fillcombo1();

fillcombo();

}

public void fillcombo()

{

string query = "Select did from driverdetails";

con.Open();

SqlCommand cmd = new SqlCommand(query, con);

SqlDataReader dr = cmd.ExecuteReader();

while (dr.Read())

{

comboBox1.Items.Add(dr[0].ToString());

}

con.Close();

}

private void assign\_vechicles\_Load(object sender, EventArgs e)

{

}

public void fillcombo1()

{

string query = "Select vid from vehiclesdetails";

con.Open();

SqlCommand cmd = new SqlCommand(query, con);

SqlDataReader dr = cmd.ExecuteReader();

while (dr.Read())

{

comboBox2.Items.Add(dr[0].ToString());

}

con.Close();

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

string query = "Select \* from driverdetails where did='" + comboBox1.Text + "'";

con.Open();

SqlCommand cmd = new SqlCommand(query, con);

SqlDataReader dr = cmd.ExecuteReader();

if (dr.Read())

{

textBox1.Text = dr[1].ToString();

pictureBox1.Image = Image.FromFile("E:/fleet project/fleetmanagementsystem/bin/Debug" + dr[6].ToString());

}

con.Close();

}

private void comboBox2\_SelectedIndexChanged(object sender, EventArgs e)

{

string query = "Select \* from vehiclesdetails where vid='" + comboBox2.Text + "'";

con.Open();

SqlCommand cmd = new SqlCommand(query, con);

SqlDataReader dr = cmd.ExecuteReader();

if (dr.Read())

{

textBox3.Text = dr[3].ToString();

pictureBox2.Image = Image.FromFile("E:/fleet project/fleetmanagementsystem/bin/Debug" + dr[4].ToString());

}

con.Close();

}

private void textBox3\_TextChanged(object sender, EventArgs e)

{

}

private void button1\_Click(object sender, EventArgs e)

{

if (textBox1.Text == "" || textBox3.Text == "" || comboBox1.Text == "" || comboBox2.Text == "" )

{

MessageBox.Show("Fill");

return;

}

string q1 = "insert into assignvechicle values('" + comboBox1.Text + "','" + textBox1.Text + "','" + comboBox2.Text + "','" + textBox3.Text + "')";

SqlCommand cmd = new SqlCommand(q1, con);

con.Open();

if (cmd.ExecuteNonQuery() > 0)

{

MessageBox.Show(" Vehicle Assigned Successfully");

textBox1.Text = "";

textBox3.Text = "";

comboBox1.Text = "";

comboBox2.Text = "";

}

con.Close();

}

private void label4\_Click(object sender, EventArgs e)

{

}

private void pictureBox1\_Click(object sender, EventArgs e)

{

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

}

}

}

**View Details Form:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SqlClient;

using System.IO;

namespace fleetmanagementsystem

{

public partial class driverdetailsview : Form

{

SqlConnection con = new SqlConnection("server=DESKTOP-PBIP1KS; database=fleet; Integrated Security=true; ");

public driverdetailsview()

{

InitializeComponent();

fillgrid();

fillgrid1();

fillgrid2();

fillgrid3();

LoadImagesFromFolder(Application.StartupPath + "\\vehiclepicss");

LoadImagesFromFolder2(Application.StartupPath + "\\driverpics");

}

public void fillgrid()

{

con.Open();

string str = "select \* from driverdetails";

DataSet ds = new DataSet();

SqlDataAdapter sqlda = new SqlDataAdapter(str, con);

sqlda.Fill(ds);

dataGridView1.DataSource = ds.Tables[0].DefaultView;

con.Close();

}

public void fillgrid3()

{

}

private void LoadImagesFromFolder(string folderPath)

{

// Clear existing images

imageList1.Images.Clear();

listView1.Items.Clear();

// Get all image files in the specified folder

string[] imageFiles = Directory.GetFiles(folderPath, "\*.\*")

.Where(f => f.EndsWith(".jpg", StringComparison.OrdinalIgnoreCase) ||

f.EndsWith(".jpeg", StringComparison.OrdinalIgnoreCase) ||

f.EndsWith(".png", StringComparison.OrdinalIgnoreCase) ||

f.EndsWith(".bmp", StringComparison.OrdinalIgnoreCase) ||

f.EndsWith(".jfif", StringComparison.OrdinalIgnoreCase) ||

f.EndsWith(".gif", StringComparison.OrdinalIgnoreCase))

.ToArray();

foreach (string file in imageFiles)

{

try

{

// Load the image and add it to the ImageList

Image img = Image.FromFile(file);

imageList1.Images.Add(img);

// Add an item to the ListView

ListViewItem item = new ListViewItem

{

ImageIndex = imageList1.Images.Count - 1, // Set

Text = Path.GetFileNameWithoutExtension(file) // Display the file name

};

listView1.Items.Add(item);

}

catch (Exception )

{

}

}

// Set the ImageList for the ListView

listView1.LargeImageList = imageList1;

}

private void LoadImagesFromFolder2(string folderPath)

{

// Clear existing images

imageList2.Images.Clear();

listView2.Items.Clear();

// Get all image files in the specified folder

string[] imageFiles = Directory.GetFiles(folderPath, "\*.\*")

.Where(f => f.EndsWith(".jpg", StringComparison.OrdinalIgnoreCase) ||

f.EndsWith(".jpeg", StringComparison.OrdinalIgnoreCase) ||

f.EndsWith(".png", StringComparison.OrdinalIgnoreCase) ||

f.EndsWith(".bmp", StringComparison.OrdinalIgnoreCase) ||

f.EndsWith(".jfif", StringComparison.OrdinalIgnoreCase) ||

f.EndsWith(".gif", StringComparison.OrdinalIgnoreCase))

.ToArray();

foreach (string file in imageFiles)

{

try

{

// Load the image and add it to the ImageList

Image img = Image.FromFile(file);

imageList2.Images.Add(img);

// Add an item to the ListView

ListViewItem item = new ListViewItem

{

ImageIndex = imageList2.Images.Count - 1, // Set the image index

Text = Path.GetFileNameWithoutExtension(file) // Display the file name

};

listView2.Items.Add(item);

}

catch (Exception )

{

}

}

// Set the ImageList for the ListView

listView2.LargeImageList = imageList2;

}

public void fillgrid1()

{

con.Open();

string str = "select \* from vehiclesdetails";

DataSet ds = new DataSet();

SqlDataAdapter sqlda = new SqlDataAdapter(str, con);

sqlda.Fill(ds);

dataGridView2.DataSource = ds.Tables[0].DefaultView;

con.Close();

}

public void fillgrid2()

{

con.Open();

string str = "select \* from assignvechicle";

DataSet ds = new DataSet();

SqlDataAdapter sqlda = new SqlDataAdapter(str, con);

sqlda.Fill(ds);

dataGridView3.DataSource = ds.Tables[0].DefaultView;

con.Close();

}

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

private void dataGridView1\_CellContentClick\_1(object sender, DataGridViewCellEventArgs e)

{

}

private void tabPage2\_Click(object sender, EventArgs e)

{

}

private void dataGridView2\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

private void tabPage3\_Click(object sender, EventArgs e)

{

}

private void pictureBox1\_Click(object sender, EventArgs e)

{

this.Close();

}

private void dataGridView4\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

private void tabControl1\_SelectedIndexChanged(object sender, EventArgs e)

{

}

private void dataGridView1\_CellContentClick\_2(object sender, DataGridViewCellEventArgs e)

{

}

private void listView2\_SelectedIndexChanged(object sender, EventArgs e)

{

if (listView2.SelectedItems.Count > 0)

{

// Get the selected item

ListViewItem selectedItem = listView2.SelectedItems[0];

// Get the text of the selected item

string selectedText = selectedItem.Text;

persondeatilsdriver obj = new persondeatilsdriver(selectedText);

ActiveForm.Hide();

obj.Show();

}

}

private void driverdetailsview\_Load(object sender, EventArgs e)

{

}

}

}

**View Trip Form:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SqlClient;

using System.Windows.Forms.DataVisualization.Charting;

namespace fleetmanagementsystem

{

public partial class OwnerViewTrips : Form

{

SqlConnection con = new SqlConnection("server=DESKTOP-PBIP1KS; database=fleet; Integrated Security=true; ");

public OwnerViewTrips()

{

InitializeComponent();

filltotal();

fillactive();

fillclosed();

fillgrid1();

fillgrid2();

fillrevenue();

fillgraph1();

}

public void filltotal()

{

string query = "Select count(tid) from tripdetails";

con.Open();

SqlCommand cmd = new SqlCommand(query, con);

SqlDataReader dr = cmd.ExecuteReader();

if (dr.Read())

{

label4.Text=dr[0].ToString();

}

con.Close();

}

public void fillgraph1()

{

string query = "SELECT did, SUM(Fare) AS TotalFare FROM tripdetails GROUP BY did";

con.Open();

DataTable dataTable = new DataTable();

SqlDataAdapter adapter = new SqlDataAdapter(query, con);

adapter.Fill(dataTable);

// Plot the data

PlotBarGraph1(dataTable);

con.Close();

}

private void PlotBarGraph1(DataTable dataTable)

{

// Create a new chart

// Create a series and add data points

Series series = new Series

{

Name = "Fare",

};

chart1.Series.Add(series);

// Add data points from DataTable

foreach (DataRow row in dataTable.Rows)

{

string driverId = row["did"].ToString();

decimal totalFare = Convert.ToDecimal(row["TotalFare"]);

series.Points.AddXY(driverId, totalFare);

}

// Add the chart to the form

}

public void fillactive()

{

string query = "Select count(tid) from tripdetails where status='active'";

con.Open();

SqlCommand cmd = new SqlCommand(query, con);

SqlDataReader dr = cmd.ExecuteReader();

if (dr.Read())

{

label5.Text = dr[0].ToString();

}

con.Close();

}

public void fillrevenue()

{

string query = "Select sum(fare) from tripdetails where status='close'";

con.Open();

SqlCommand cmd = new SqlCommand(query, con);

SqlDataReader dr = cmd.ExecuteReader();

if (dr.Read())

{

label8.Text = dr[0].ToString();

}

con.Close();

}

public void fillclosed()

{

string query = "Select count(tid) from tripdetails where status='close'";

con.Open();

SqlCommand cmd = new SqlCommand(query, con);

SqlDataReader dr = cmd.ExecuteReader();

if (dr.Read())

{

label6.Text = dr[0].ToString();

}

con.Close();

}

public void fillgrid1()

{

con.Open();

string query = "Select \* from tripdetails where status='active'";

DataSet ds = new DataSet();

SqlDataAdapter sqlda = new SqlDataAdapter(query, con);

sqlda.Fill(ds);

dataGridView1.DataSource = ds.Tables[0].DefaultView;

con.Close();

}

public void fillgrid2()

{

con.Open();

string query = "Select \* from tripdetails where status='close'";

DataSet ds = new DataSet();

SqlDataAdapter sqlda = new SqlDataAdapter(query, con);

sqlda.Fill(ds);

dataGridView2.DataSource = ds.Tables[0].DefaultView;

con.Close();

}

private void OwnerViewTrips\_Load(object sender, EventArgs e)

{

}

}

}

**Owner Password Change Form:**

using System;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace fleetmanagementsystem

{

public partial class ownerpasswordchange : Form

{

// Establish a connection to the database

SqlConnection con = new SqlConnection("server=DESKTOP-PBIP1KS; database=fleet; Integrated Security=true;");

// Assuming this is the driver ID (for example, you may retrieve this from the session)

private string driverId = Program.username; // Replace with actual logic to get the logged-in driver ID

public ownerpasswordchange()

{

InitializeComponent();

}

private void driverpasswordchange\_Load(object sender, EventArgs e)

{

}

private void button1\_Click(object sender, EventArgs e)

{

// Step 1: Validate user input

if (textBox1.Text == "" || textBox2.Text == "" || textBox3.Text == "")

{

MessageBox.Show("Please fill in all fields.");

return;

}

string oldPassword = textBox1.Text; // Old password

string newPassword = textBox2.Text; // New password

string confirmPassword = textBox3.Text; // Confirm new password

// Step 2: Check if new password and confirm password match

if (newPassword != confirmPassword)

{

MessageBox.Show("New password and confirm password do not match!");

return;

}

try

{

// Step 3: Open the database connection

con.Open();

// Step 4: Retrieve the current (old) password from the logindetails table for the logged-in driver

string selectQuery = "SELECT Password FROM logindetails WHERE username = @DriverId";

SqlCommand selectCmd = new SqlCommand(selectQuery, con);

selectCmd.Parameters.AddWithValue("@DriverId", driverId);

object result = selectCmd.ExecuteScalar();

if (result != null)

{

string storedPassword = result.ToString();

// Step 5: Check if the old password entered by the user matches the stored password

if (storedPassword == oldPassword)

{

// Step 6: If old password is correct, update the password in the logindetails table

string updateQuery = "UPDATE logindetails SET Password = @NewPassword WHERE username= @DriverId";

SqlCommand updateCmd = new SqlCommand(updateQuery, con);

updateCmd.Parameters.AddWithValue("@NewPassword", newPassword);

updateCmd.Parameters.AddWithValue("@DriverId", driverId);

int rowsAffected = updateCmd.ExecuteNonQuery();

if (rowsAffected > 0)

{

MessageBox.Show("Password changed successfully!");

}

else

{

MessageBox.Show("Password Change Failed.");

}

}

else

{

MessageBox.Show("Old Password is Incorrect.");

}

}

else

{

MessageBox.Show("Driver Not Found.");

}

}

catch (Exception ex)

{

MessageBox.Show("Error:" + ex.InnerException.ToString());

}

finally

{

// Step 7: Close the connection

if (con.State == System.Data.ConnectionState.Open)

{

con.Close();

}

}

}

// Other event handlers (if needed)

private void label2\_Click(object sender, EventArgs e) { }

private void textBox1\_TextChanged(object sender, EventArgs e) { }

private void textBox2\_TextChanged(object sender, EventArgs e) { }

private void label1\_Click(object sender, EventArgs e) { }

private void label3\_Click(object sender, EventArgs e) { }

private void textBox3\_TextChanged(object sender, EventArgs e) { }

private void driverpasswordchange\_Load\_1(object sender, EventArgs e)

{

}

}

}

**Driver Login Form:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SqlClient;

namespace fleetmanagementsystem

{

public partial class driverlogin : Form

{

SqlConnection con = new SqlConnection("server=DESKTOP-PBIP1KS; database=fleet; Integrated Security=true; ");

public driverlogin(string username\_from\_login)

{

InitializeComponent();

greeting();

fillcontent( Program.did );

}

public void fillcontent(string id)

{

string query = "Select \* from driverdetails where did='" + id + "'";

con.Open();

SqlCommand cmd = new SqlCommand(query, con);

SqlDataReader dr = cmd.ExecuteReader();

while (dr.Read())

{

pictureBox1.ImageLocation = Application.StartupPath + dr[6].ToString();

}

con.Close();

}

public void greeting()

{

int hour = DateTime.Now.Hour;

// Determine greeting based on the hour

if (hour >= 0 && hour < 12)

{

label3.Text = "Good Morning " + Program.dname;

}

else if (hour >= 12 && hour < 17)

{

label3.Text = "Good Afternoon " + Program.dname;

}

else

{

label3.Text = "Good Evening " + Program.dname;

}

}

private void label1\_Click(object sender, EventArgs e)

{

driverpasswordchange obj = new driverpasswordchange();

obj.Show();

}

private void label2\_Click(object sender, EventArgs e)

{

tripdetails obj = new tripdetails();

obj.Show();

}

private void panel1\_Paint(object sender, PaintEventArgs e)

{

}

private void label3\_Click(object sender, EventArgs e)

{

}

private void panel2\_Paint(object sender, PaintEventArgs e)

{

}

private void label4\_Click(object sender, EventArgs e)

{

closetrip obj = new closetrip();

obj.Show();

}

private void pictureBox1\_Click(object sender, EventArgs e)

{

}

}}

**Trip Details Form:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SqlClient;

namespace fleetmanagementsystem

{

public partial class tripdetails : Form

{

public tripdetails()

{

InitializeComponent();

textBox3.Text = Program.did;

gettid();

gettvid();

textBox8.Text = System.DateTime.Now.ToShortDateString();

}

SqlConnection con = new SqlConnection("server=DESKTOP-PBIP1KS; database=fleet; Integrated Security=true; ");

public void gettid()

{

con.Open();

string query = "select isnull(MAX(tid),5000)+1 from tripdetails";

SqlCommand cmd = new SqlCommand(query, con);

SqlDataReader dr = cmd.ExecuteReader();

if (dr.Read())

{

textBox1.Text = dr[0].ToString();

}

con.Close();

}

public void gettvid()

{

comboBox1.Items.Clear();

con.Open();

string query = "select distinct(vid) from assignvechicle where did='"+Program.did+"'";

SqlCommand cmd = new SqlCommand(query, con);

SqlDataReader dr = cmd.ExecuteReader();

while (dr.Read())

{

comboBox1.Items.Add(dr[0].ToString());

}

con.Close();

}

private void label1\_Click(object sender, EventArgs e)

{

}

private void textBox2\_TextChanged(object sender, EventArgs e)

{

}

private void textBox3\_TextChanged(object sender, EventArgs e)

{

}

private void textBox8\_TextChanged(object sender, EventArgs e)

{

}

private void textBox4\_TextChanged(object sender, EventArgs e)

{

}

private void textBox5\_TextChanged(object sender, EventArgs e)

{

}

private void tripdetails\_Load(object sender, EventArgs e)

{

}

private void button1\_Click(object sender, EventArgs e)

{

if (textBox1.Text == "" || comboBox1.Text==""|| textBox3.Text == "" || textBox4.Text == "" || textBox5.Text == "" || textBox8.Text == "" )

{

MessageBox.Show("Fill");

return;

}

string q1 = "insert into tripdetails values('" + textBox1.Text + "','" + comboBox1.Text + "','" + textBox3.Text + "','" + textBox5.Text + "','','','" + textBox4.Text + "','" + textBox8.Text + "','','','','active','')";

SqlCommand cmd = new SqlCommand(q1, con);

con.Open();

if (cmd.ExecuteNonQuery() > 0)

{

MessageBox.Show("Trip Added Successfully");

textBox1.Text = "";

textBox5.Text = "";

textBox4.Text = "";

textBox3.Text = "";

textBox8.Text = "";

}

con.Close();

this.Close();

}

}

}

**Close Trip Form:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SqlClient;

namespace fleetmanagementsystem

{

public partial class closetrip : Form

{

SqlConnection con = new SqlConnection("server=DESKTOP-PBIP1KS; database=fleet; Integrated Security=true; ");

public closetrip()

{

InitializeComponent();

gettid();

getvid();

}

public void gettid()

{

con.Open();

string query = "select distinct(tid) from tripdetails where did='"+Program.did+"' and status='active'";

SqlCommand cmd = new SqlCommand(query, con);

SqlDataReader dr = cmd.ExecuteReader();

while (dr.Read())

{

comboBox1.Items.Add(dr[0].ToString());

}

con.Close();

}

public void getvid()

{

con.Open();

string query = "select \* from tripdetails where starting\_reading = '" + textBox3.Text + "'";

SqlCommand cmd = new SqlCommand(query, con);

SqlDataReader dr = cmd.ExecuteReader();

if (dr.Read())

{

textBox1.Text = dr[3].ToString();

}

con.Close();

}

private void closetrip\_Load(object sender, EventArgs e)

{

}

private void button1\_Click(object sender, EventArgs e)

{

string query = @"update tripdetails

set end\_reading=@endreading,kilometers=@kilo,enddate=@enddate,toll=@toll,Fare=@fare,status='close',total=@total

where tid=@tid";

con.Open();

SqlCommand cmd = new SqlCommand(query, con);

cmd.Parameters.AddWithValue("@endreading", textBox4.Text);

cmd.Parameters.AddWithValue("@kilo", textBox5.Text);

cmd.Parameters.AddWithValue("@enddate", dateTimePicker1.Value.ToString("yyyy-mm-dd"));

cmd.Parameters.AddWithValue("@toll", textBox6.Text);

cmd.Parameters.AddWithValue("@fare", textBox2.Text);

cmd.Parameters.AddWithValue("@total", label13.Text);

cmd.Parameters.AddWithValue("@tid", comboBox1.Text);

cmd.ExecuteReader();

con.Close();

MessageBox.Show("Trip Closed Successfully");

this.Close();

// comboBox1.Text = "";

// textBox4.Text="";

//textBox1.Text = "";

//textBox8.Text = "";

// textBox5.Text = "";

//textBox9.Text = "";

//textBox3.Text = "";

//textBox6.Text = "";

// t/extBox2.Text = "";

//label13.Text = "";

//string query = "UPDATE tripdetails SET end\_reading='" + textBox4.Text + "', kilometers='" + textBox5.Text + "',enddate='" + dateTimePicker1.Value.ToString("yyyy-mm-dd") + "',Toll='" + textBox6.Text + "',total='" + label13.Text + "',status='close' WHERE did='" + Program.did + "'";

//con.Open();

//SqlCommand cmd = new SqlCommand(query, con);

//con.Close();

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

string query = "Select \* from tripdetails where tid='" + comboBox1.Text + "'";

con.Open();

SqlCommand cmd = new SqlCommand(query, con);

SqlDataReader dr = cmd.ExecuteReader();

if (dr.Read())

{

textBox1.Text = dr[3].ToString();

textBox8.Text = dr[7].ToString();

}

con.Close();

}

private void textBox4\_TextChanged(object sender, EventArgs e)

{

// Display the difference in textBox5

// Check if the value in textBox4 is less than the value in textBox1

}

private void dateTimePicker1\_ValueChanged(object sender, EventArgs e)

{

}

private void textBox5\_TextChanged(object sender, EventArgs e)

{

}

private void button2\_Click(object sender, EventArgs e)

{

panel1.Visible = true;

DateTime secondate = dateTimePicker1.Value;

DateTime firstdate = Convert.ToDateTime(textBox8.Text);

TimeSpan dt = secondate - firstdate;

textBox9.Text = dt.Days.ToString();

if (string.IsNullOrEmpty(textBox1.Text) || string.IsNullOrEmpty(textBox4.Text))

{

textBox5.Text = ""; // Clear textBox5 if either is empty

return; // Exit if either textBox is empty

}

// Try to parse the values of textBox1 and textBox4

decimal value1 = Convert.ToDecimal(textBox4.Text);

decimal value4 = Convert.ToDecimal(textBox1.Text);

// Calculate the difference

decimal difference = value1 - value4;

textBox5.Text = difference.ToString();

if (dt.Days == 0)

{

textBox9.Text = "1";

textBox3.Text = Convert.ToString(Convert.ToDecimal(textBox9.Text) \* Convert.ToDecimal(Program.bata));

}

else

{

textBox3.Text = Convert.ToString(Convert.ToDecimal(textBox9.Text) \* Convert.ToDecimal(Program.bata));

}

if (Convert.ToDecimal(textBox5.Text) > 0)

{

textBox2.Text = Convert.ToString(Convert.ToDecimal(textBox5.Text) \* Convert.ToDecimal(Program.fare));

}

label13.Text = Convert.ToString(Convert.ToDecimal(textBox6.Text) + Convert.ToDecimal(textBox2.Text) + Convert.ToDecimal(textBox3.Text));

}

}

}

**Update Details Form:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SqlClient;

namespace fleetmanagementsystem

{

public partial class updatedetails : Form

{

SqlConnection con = new SqlConnection("server=DESKTOP-PBIP1KS; database=fleet; Integrated Security=true; ");

public updatedetails()

{

InitializeComponent();

fillcombo();

fillgrid();

fillcombo1();

}

public updatedetails(string id)

{

InitializeComponent();

comboBox1.Text = id;

string query = "Select \* from driverdetails where did='" + comboBox1.Text + "'";

con.Open();

SqlCommand cmd = new SqlCommand(query, con);

SqlDataReader dr = cmd.ExecuteReader();

if (dr.Read())

{

textBox1.Text = dr[1].ToString();

textBox2.Text = dr[2].ToString();

textBox3.Text = dr[3].ToString();

textBox4.Text = dr[4].ToString();

comboBox3.Text = dr[5].ToString();

}

dataGridView1.Visible = false;

con.Close();

}

public void fillcombo()

{

string query = "Select did from driverdetails";

con.Open();

SqlCommand cmd = new SqlCommand(query, con);

SqlDataReader dr = cmd.ExecuteReader();

while (dr.Read())

{

comboBox1.Items.Add(dr[0].ToString());

}

con.Close();

}

public void fillcombo1()

{

string query = "Select did from driverdetails";

con.Open();

SqlCommand cmd = new SqlCommand(query, con);

SqlDataReader dr = cmd.ExecuteReader();

while (dr.Read())

{

comboBox2.Items.Add(dr[0].ToString());

}

con.Close();

}

private void tabPage1\_Click(object sender, EventArgs e)

{

}

private void updatedetails\_Load(object sender, EventArgs e)

{

}

public void fillgrid()

{

con.Open();

string str = "select \* from driverdetails";

DataSet ds = new DataSet();

SqlDataAdapter sqlda = new SqlDataAdapter(str, con);

sqlda.Fill(ds);

dataGridView1.DataSource = ds.Tables[0].DefaultView;

con.Close();

}

private void button1\_Click(object sender, EventArgs e)

{

string query = "update driverdetails set name='" + textBox1.Text + "',age='" + textBox2.Text + "',mobile='" + textBox3.Text + "',gender='" + comboBox3.Text + "'where did='"+comboBox1.Text+"'";

con.Open();

SqlCommand cmd = new SqlCommand(query, con);

if (cmd.ExecuteNonQuery() > 0)

{

MessageBox.Show("Value updated successfully");

textBox1.Text = "";

textBox2.Text = "";

textBox3.Text = "";

textBox4.Text = "";

comboBox3.Text = "";

comboBox1.Text = "";

}

con.Close();

fillgrid();

}

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

string query = "Select \* from driverdetails where did='" + comboBox1.Text + "'";

con.Open();

SqlCommand cmd = new SqlCommand(query, con);

SqlDataReader dr = cmd.ExecuteReader();

if (dr.Read())

{

textBox1.Text = dr[1].ToString();

textBox2.Text = dr[2].ToString();

textBox3.Text = dr[3].ToString();

textBox4.Text = dr[4].ToString();

comboBox3.Text = dr[5].ToString();

}

con.Close();

}

private void button2\_Click(object sender, EventArgs e)

{

string query = "delete from driverdetails where did='" + comboBox2.Text + "'";

con.Open();

SqlCommand cmd = new SqlCommand(query, con);

if (cmd.ExecuteNonQuery() > 0)

{

MessageBox.Show("driver details deleted");

}

con.Close();

}

}

}

**Owner View Trip Form:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SqlClient;

using System.Windows.Forms.DataVisualization.Charting;

namespace fleetmanagementsystem

{

public partial class OwnerViewTrips : Form

{

SqlConnection con = new SqlConnection("server=DESKTOP-PBIP1KS; database=fleet; Integrated Security=true; ");

public OwnerViewTrips()

{

InitializeComponent();

filltotal();

fillactive();

fillclosed();

fillgrid1();

fillgrid2();

fillrevenue();

fillgraph1();

}

public void filltotal()

{

string query = "Select count(tid) from tripdetails";

con.Open();

SqlCommand cmd = new SqlCommand(query, con);

SqlDataReader dr = cmd.ExecuteReader();

if (dr.Read())

{

label4.Text=dr[0].ToString();

}

con.Close();

}

public void fillgraph1()

{

string query = "SELECT did, SUM(Fare) AS TotalFare FROM tripdetails GROUP BY did";

con.Open();

DataTable dataTable = new DataTable();

SqlDataAdapter adapter = new SqlDataAdapter(query, con);

adapter.Fill(dataTable);

// Plot the data

PlotBarGraph1(dataTable);

con.Close();

}

private void PlotBarGraph1(DataTable dataTable)

{

// Create a new chart

// Create a series and add data points

Series series = new Series

{

Name = "Fare",

};

chart1.Series.Add(series);

// Add data points from DataTable

foreach (DataRow row in dataTable.Rows)

{

string driverId = row["did"].ToString();

decimal totalFare = Convert.ToDecimal(row["TotalFare"]);

series.Points.AddXY(driverId, totalFare);

}

// Add the chart to the form

}

public void fillactive()

{

string query = "Select count(tid) from tripdetails where status='active'";

con.Open();

SqlCommand cmd = new SqlCommand(query, con);

SqlDataReader dr = cmd.ExecuteReader();

if (dr.Read())

{

label5.Text = dr[0].ToString();

}

con.Close();

}

public void fillrevenue()

{

string query = "Select sum(fare) from tripdetails where status='close'";

con.Open();

SqlCommand cmd = new SqlCommand(query, con);

SqlDataReader dr = cmd.ExecuteReader();

if (dr.Read())

{

label8.Text = dr[0].ToString();

}

con.Close();

}

public void fillclosed()

{

string query = "Select count(tid) from tripdetails where status='close'";

con.Open();

SqlCommand cmd = new SqlCommand(query, con);

SqlDataReader dr = cmd.ExecuteReader();

if (dr.Read())

{

label6.Text = dr[0].ToString();

}

con.Close();

}

public void fillgrid1()

{

con.Open();

string query = "Select \* from tripdetails where status='active'";

DataSet ds = new DataSet();

SqlDataAdapter sqlda = new SqlDataAdapter(query, con);

sqlda.Fill(ds);

dataGridView1.DataSource = ds.Tables[0].DefaultView;

con.Close();

}

public void fillgrid2()

{

con.Open();

string query = "Select \* from tripdetails where status='close'";

DataSet ds = new DataSet();

SqlDataAdapter sqlda = new SqlDataAdapter(query, con);

sqlda.Fill(ds);

dataGridView2.DataSource = ds.Tables[0].DefaultView;

con.Close();

}

private void OwnerViewTrips\_Load(object sender, EventArgs e)

{

}

}

}

**Personal Details View Form:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.IO;

using System.Data.SqlClient;

namespace fleetmanagementsystem

{

public partial class persondeatilsdriver : Form

{

SqlConnection con = new SqlConnection("server=DESKTOP-PBIP1KS; database=fleet; Integrated Security=true; ");

public persondeatilsdriver()

{

InitializeComponent();

}

public persondeatilsdriver(string id)

{

InitializeComponent();

fillcontent(id);

filltrips(id);

fillkilometers(id);

fillrevenue(id);

}

public void fillrevenue(string id)

{

con.Open();

string query = "SELECT SUM(CAST(total AS INT)) AS Revenue FROM tripdetails WHERE ISNUMERIC(total) = 1 and did='" + id + "'";

SqlCommand cmd = new SqlCommand(query, con);

SqlDataReader dr = cmd.ExecuteReader();

while (dr.Read())

{

label16.Text = dr[0].ToString();

}

con.Close();

}

public void filltrips(string id)

{

con.Open();

string query="select count(tid) from tripdetails where did='" + id + "'";

SqlCommand cmd = new SqlCommand(query, con);

SqlDataReader dr = cmd.ExecuteReader();

while (dr.Read())

{

label7.Text = dr[0].ToString();

}

con.Close();

}

public void fillkilometers(string id)

{

con.Open();

string query = "SELECT SUM(CAST(kilometers AS INT)) AS TotalKilometers FROM tripdetails WHERE ISNUMERIC(kilometers) = 1 and did='" + id + "'";

SqlCommand cmd = new SqlCommand(query, con);

SqlDataReader dr = cmd.ExecuteReader();

while (dr.Read())

{

label14.Text = dr[0].ToString();

}

con.Close();

}

public void fillcontent(string id)

{

textBox1.Text = id;

string query = "Select \* from driverdetails where did='"+id+"'";

con.Open();

SqlCommand cmd = new SqlCommand(query, con);

SqlDataReader dr = cmd.ExecuteReader();

while (dr.Read())

{

textBox2.Text = dr[1].ToString();

textBox3.Text = dr[2].ToString();

textBox6.Text = dr[3].ToString();

textBox5.Text = dr[4].ToString();

textBox4.Text = dr[5].ToString();

pictureBox1.ImageLocation = Application.StartupPath + dr[6].ToString();

}

con.Close();

}

private void persondeatilsdriver\_Load(object sender, EventArgs e)

{

}

private void button1\_Click(object sender, EventArgs e)

{

updatedetails obj = new updatedetails(textBox1.Text);

ActiveForm.Hide();

obj.Show();

}

private void pictureBox1\_Click(object sender, EventArgs e)

{

}

}

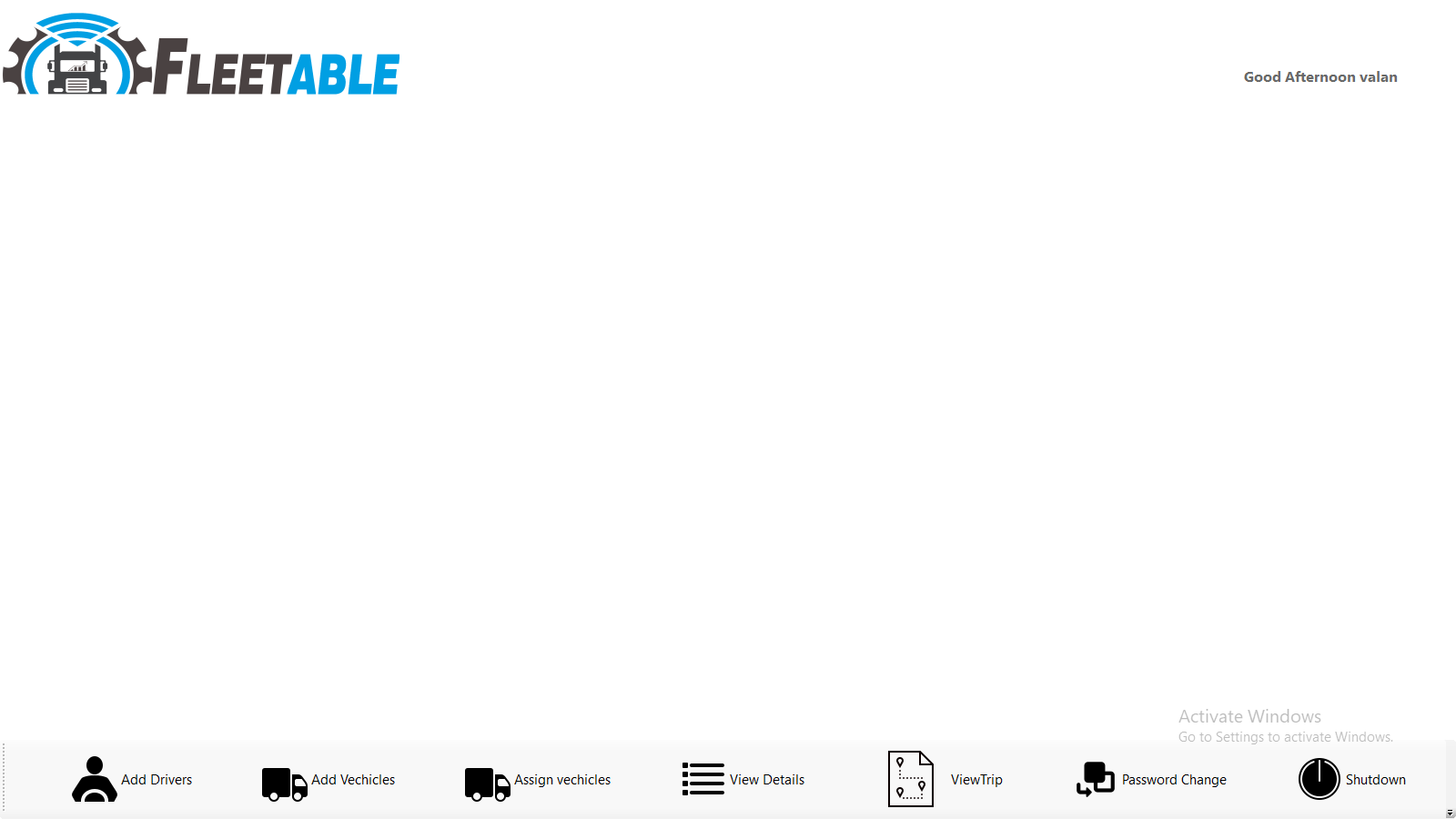
}

**6.2.2 SCREENSHOTS**

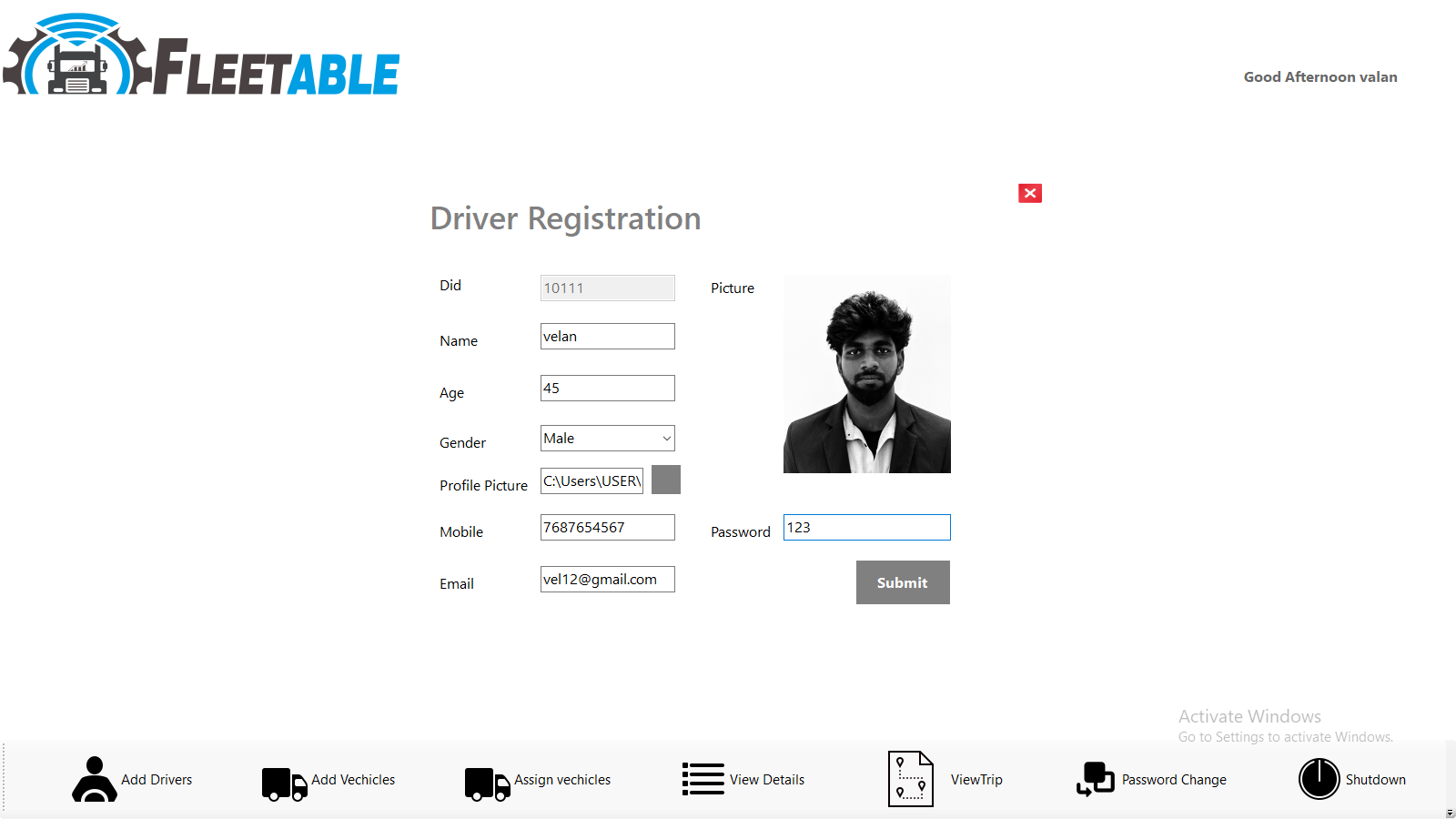
**Login**

****

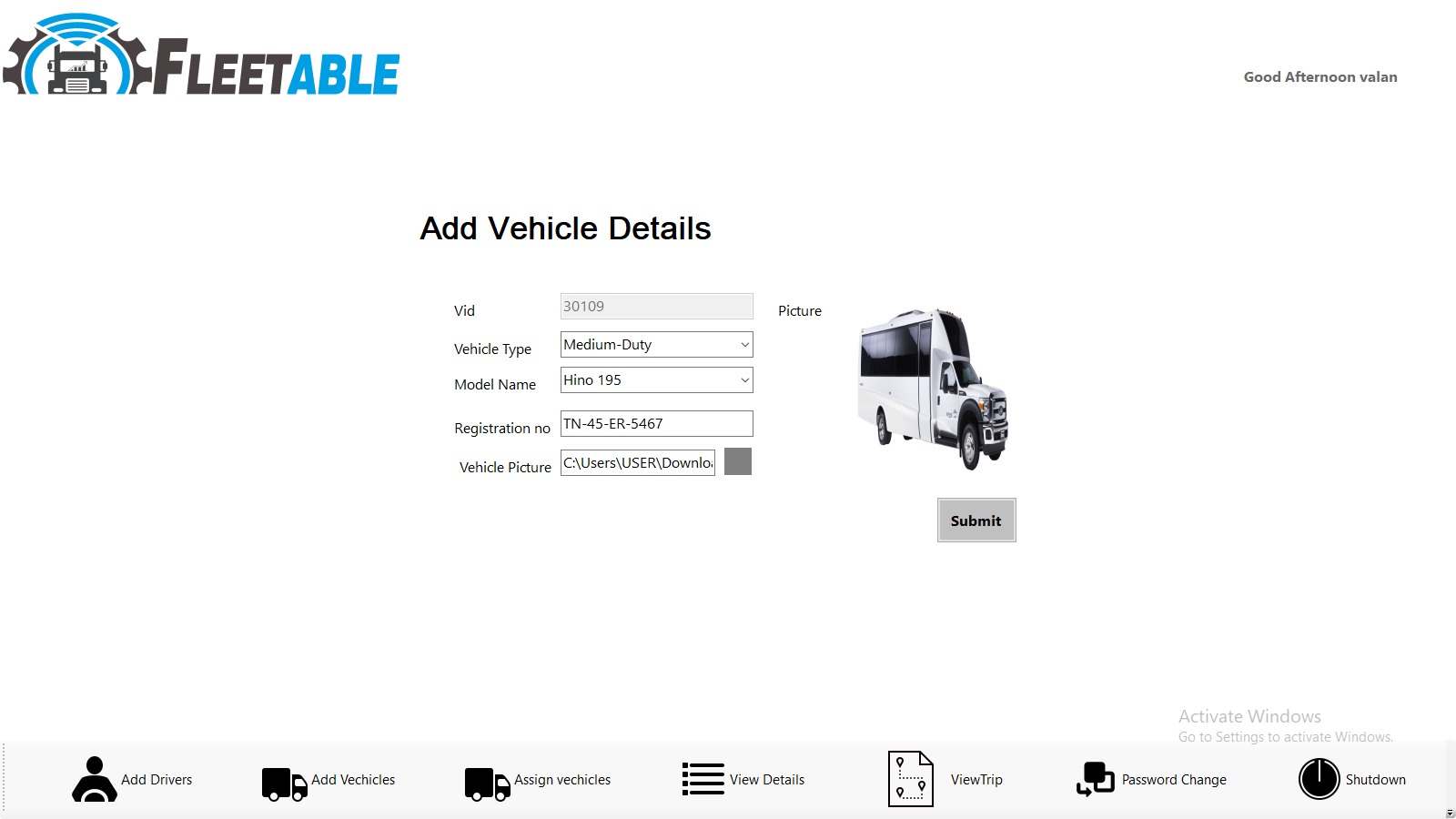
**Owner Login**

****

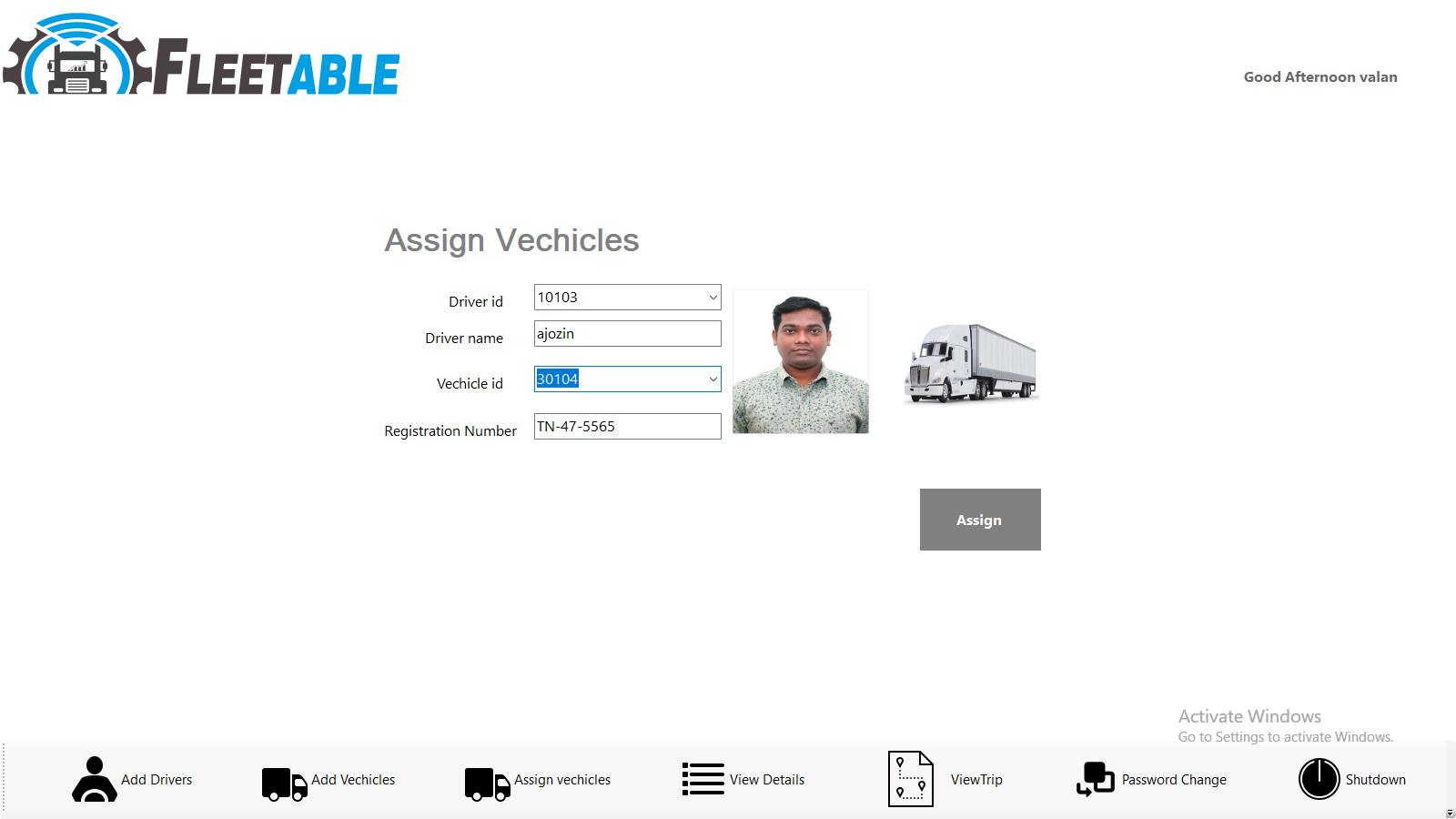
**Add Driver**

****

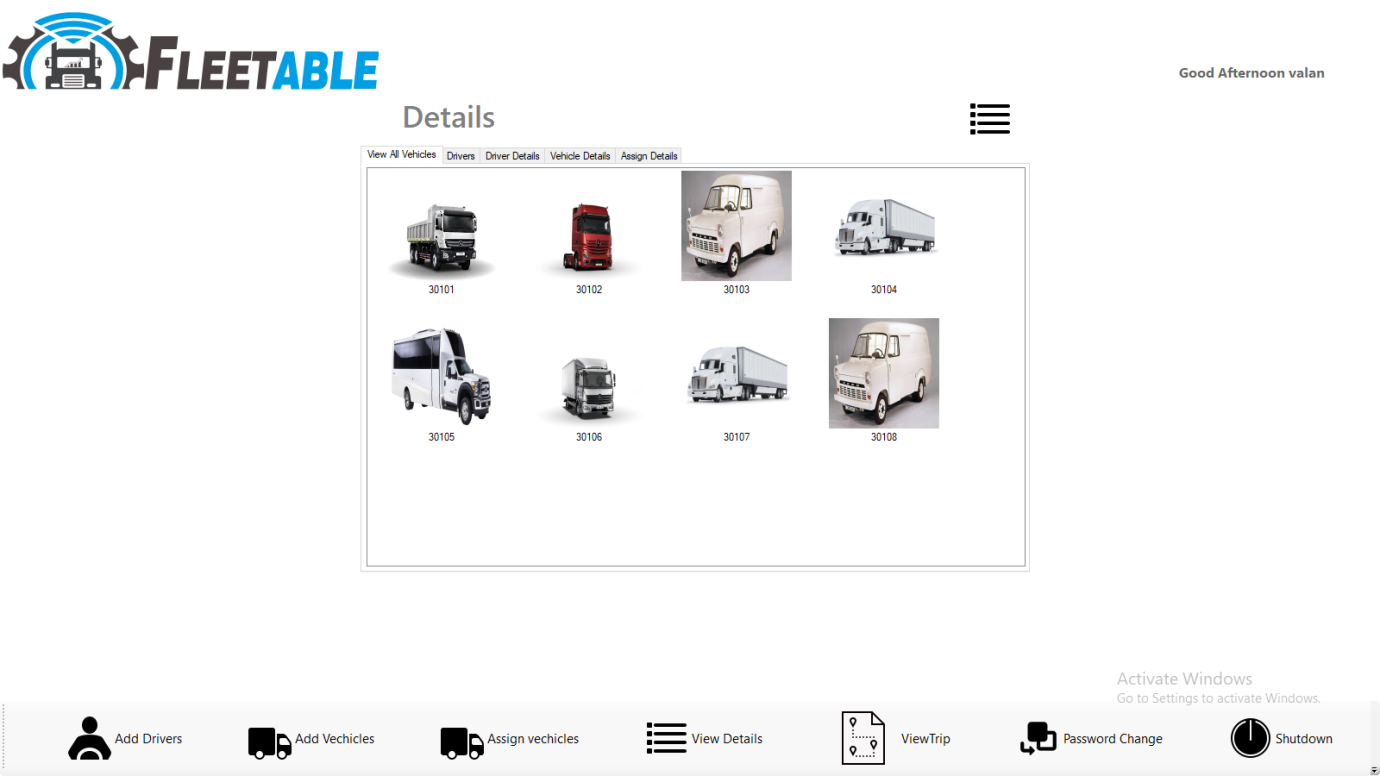
**Add Vechicles**

****

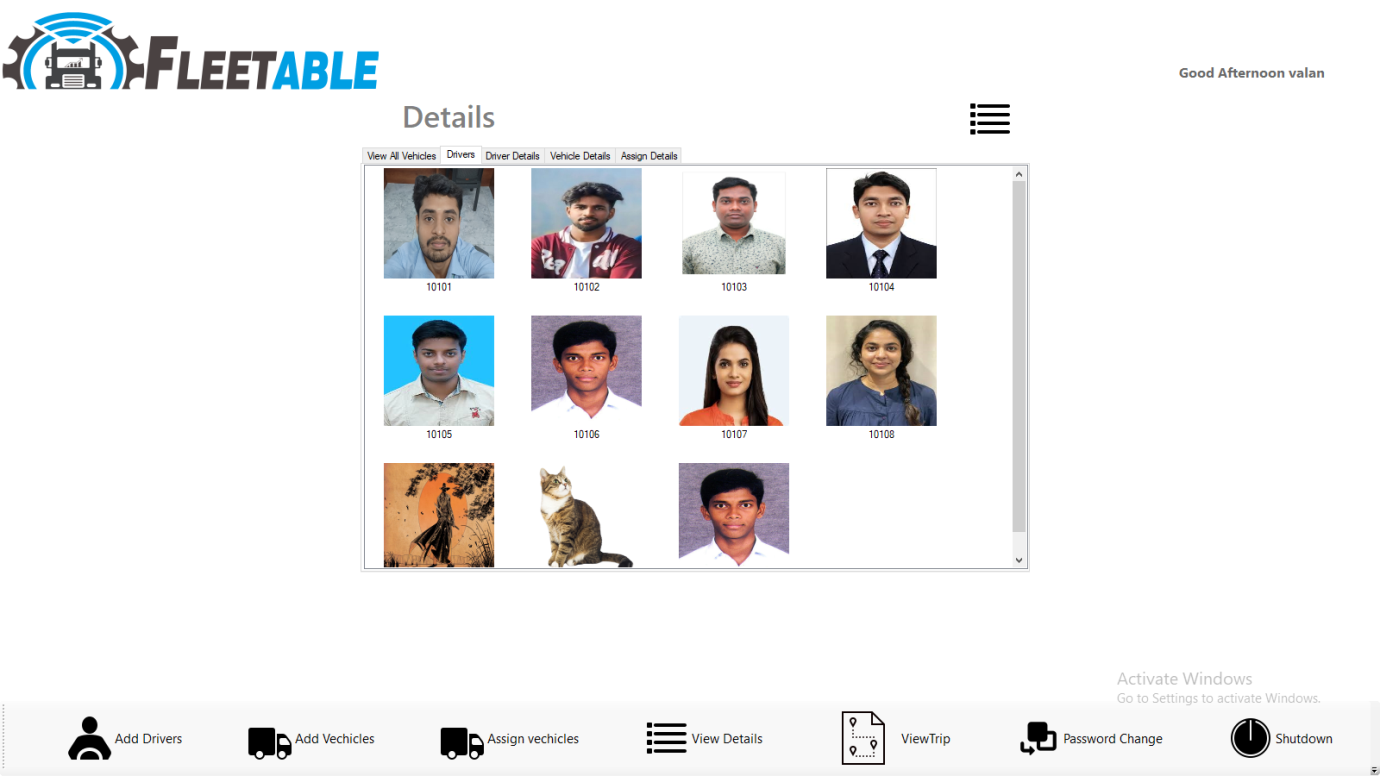
**Assign Vehicles**

****

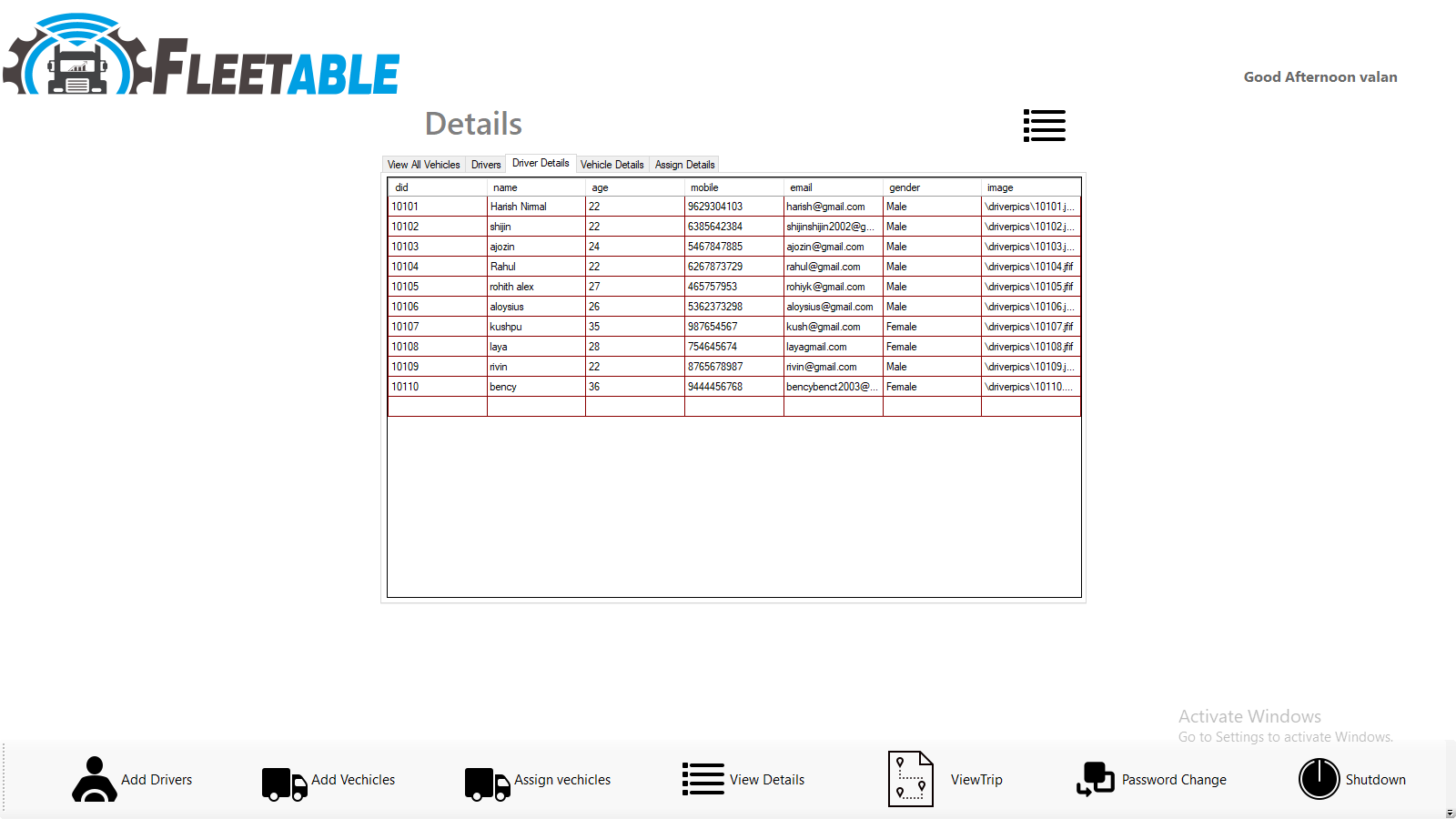
**View Details**

****

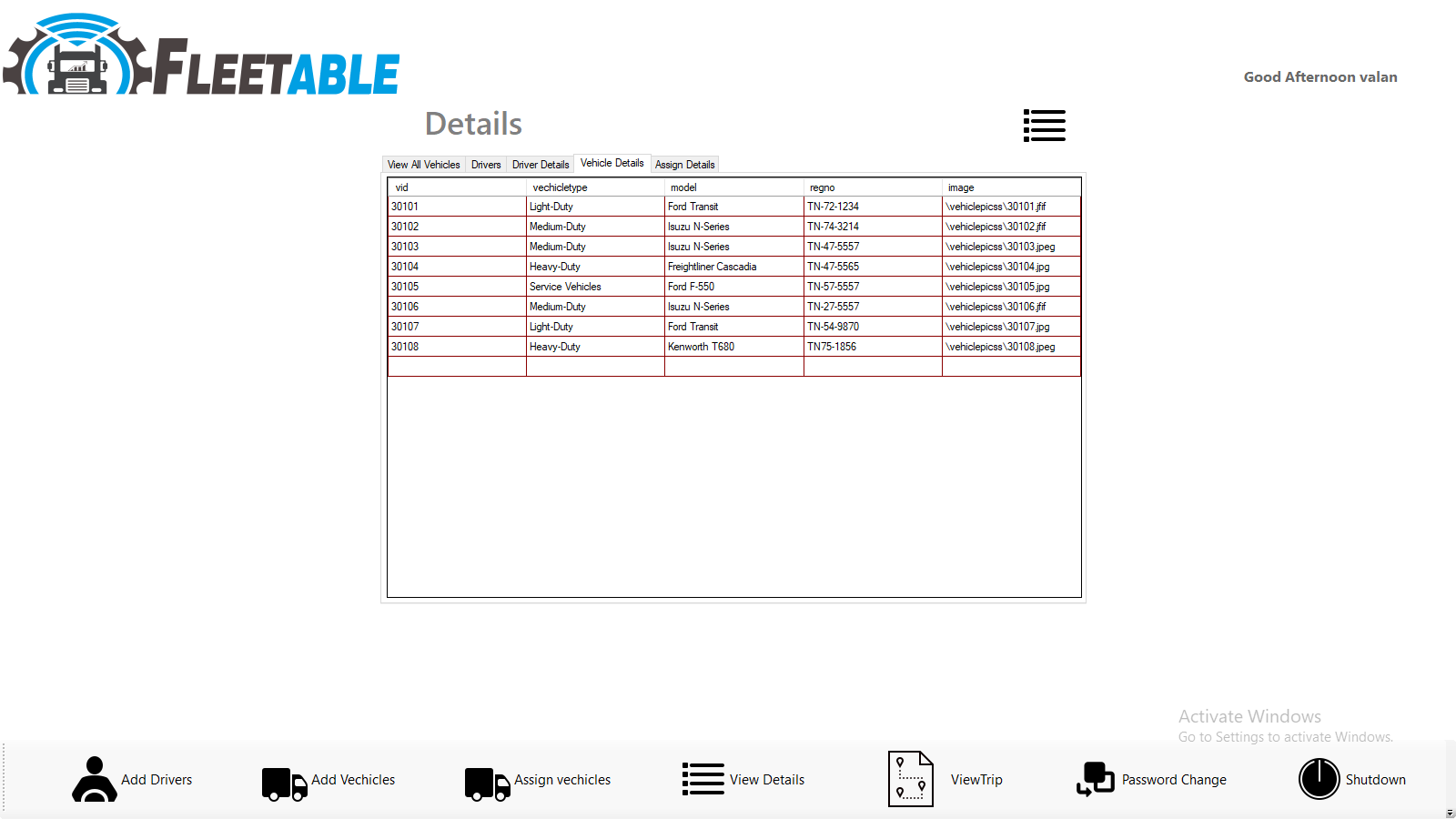
**View Driver**

****

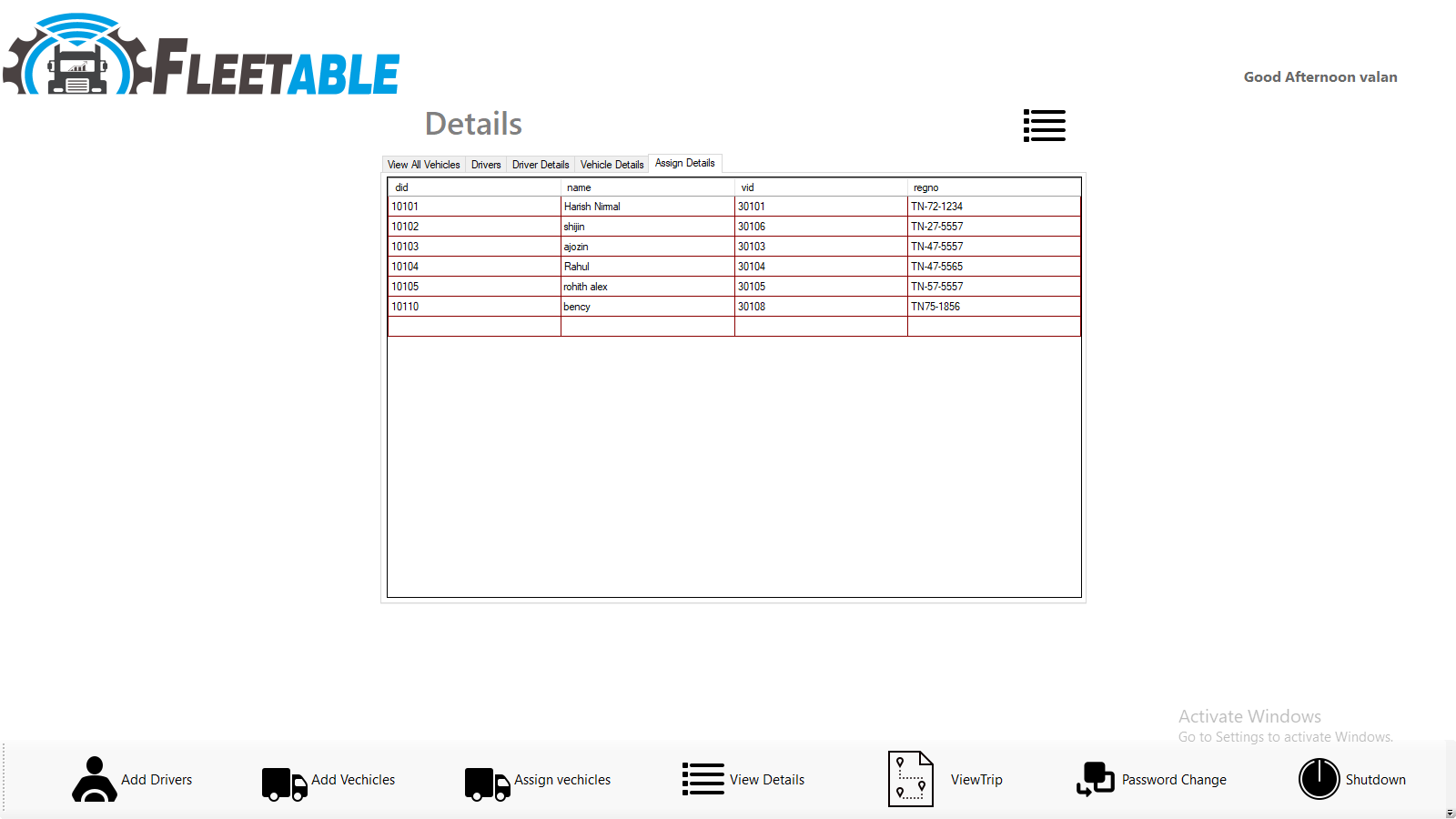
**View Driver Details**

****

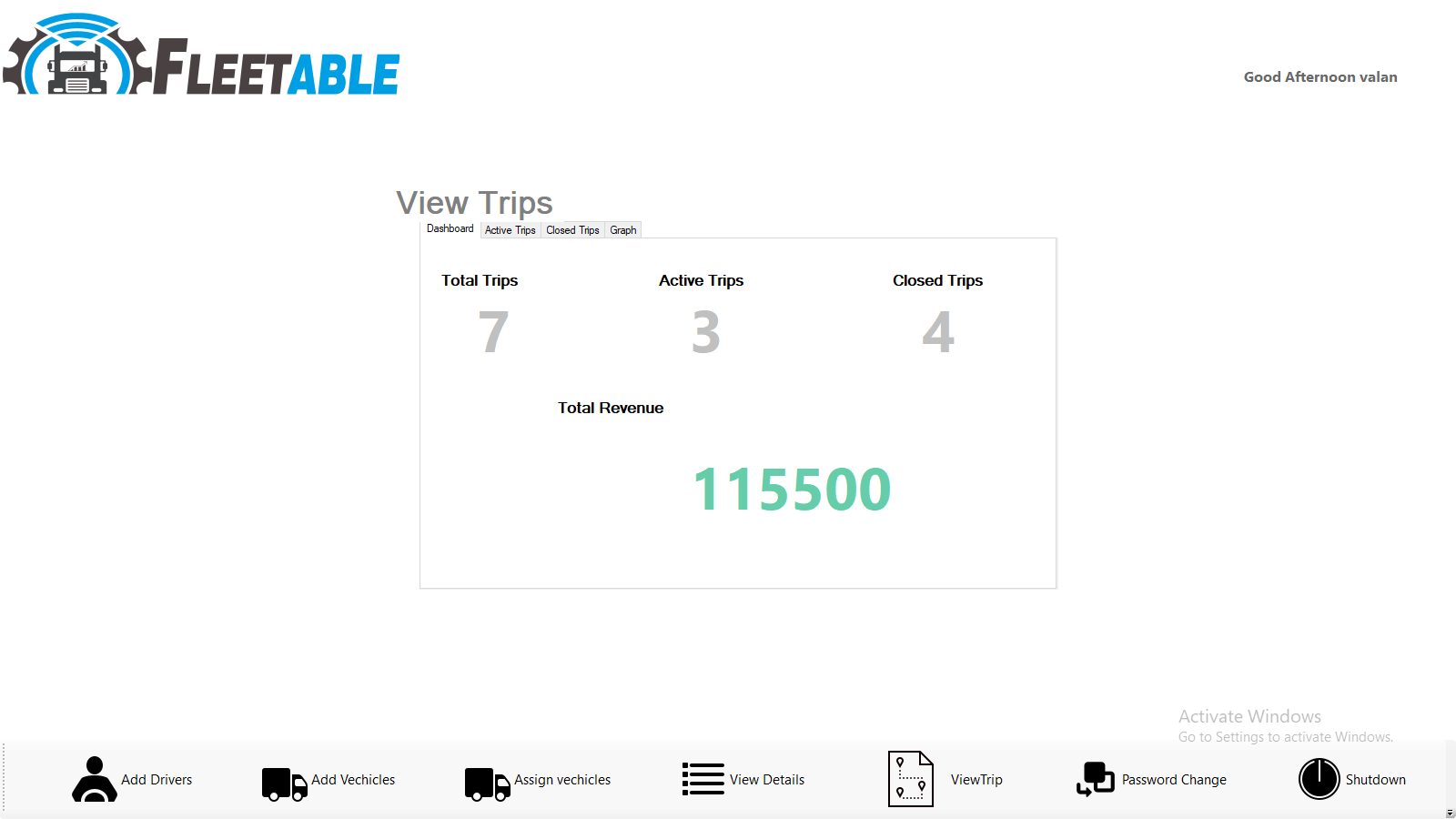
**View Vehicles Details**

****

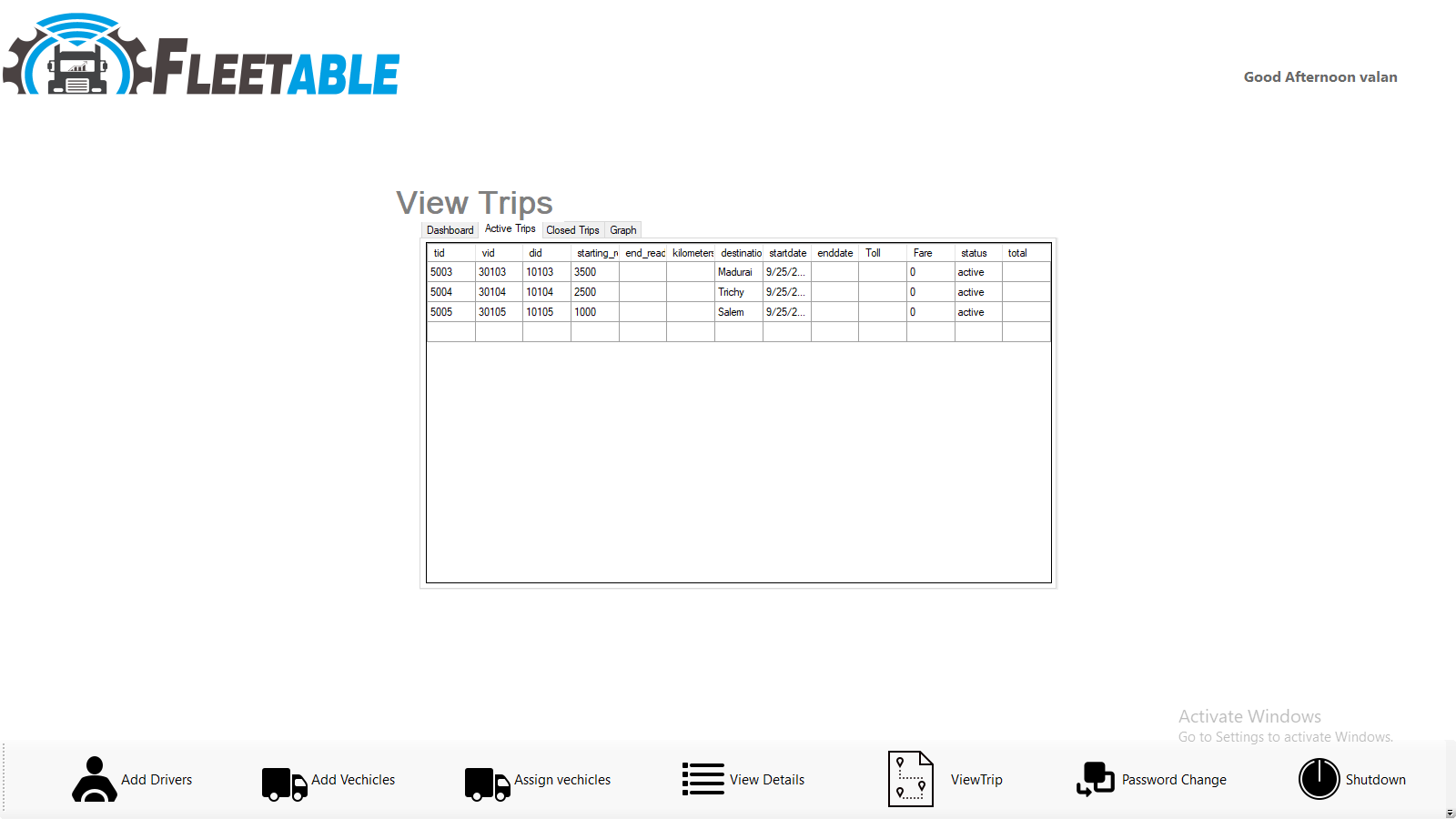
**View Assign Details**

****

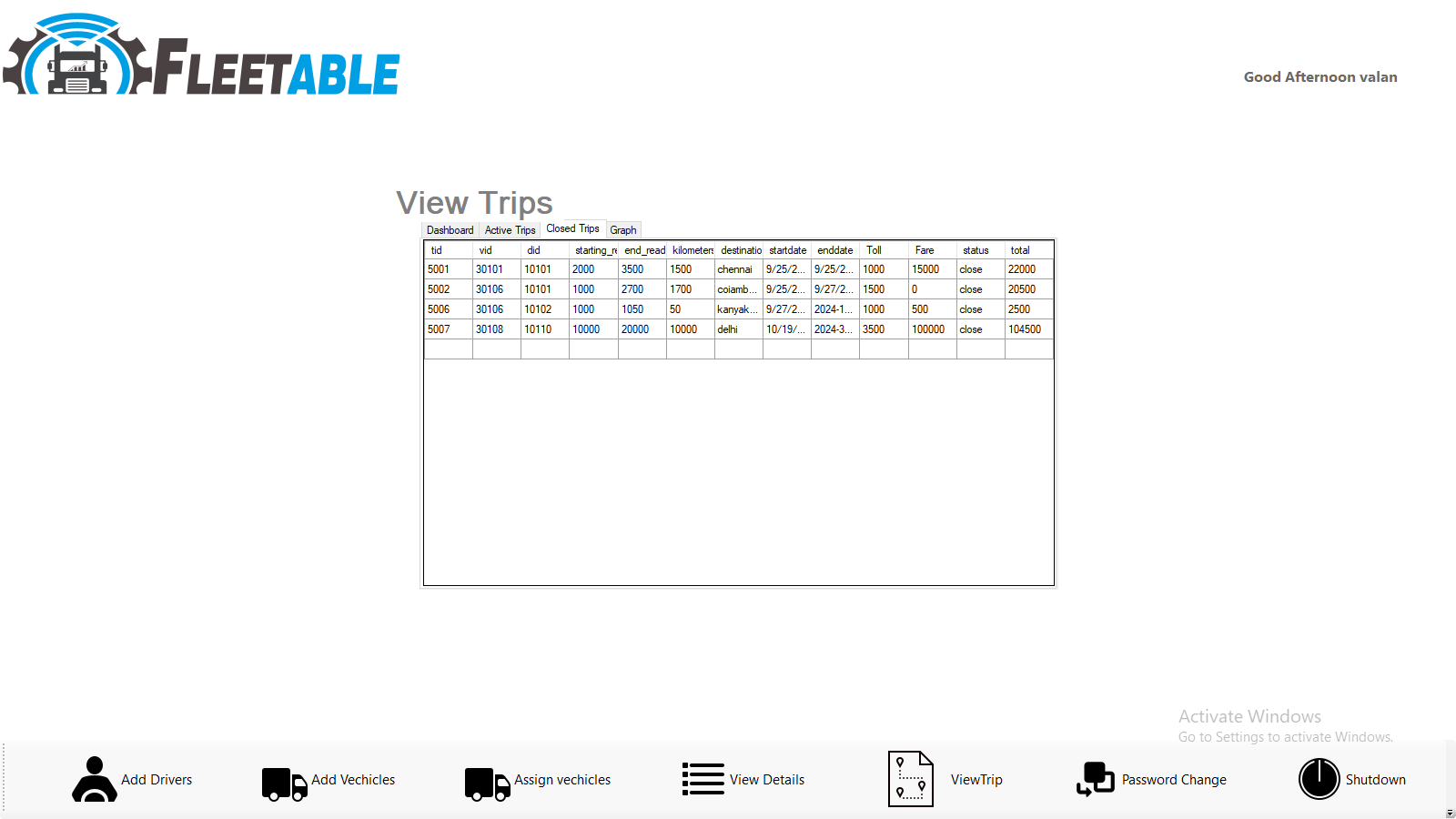
**View Trip**

****

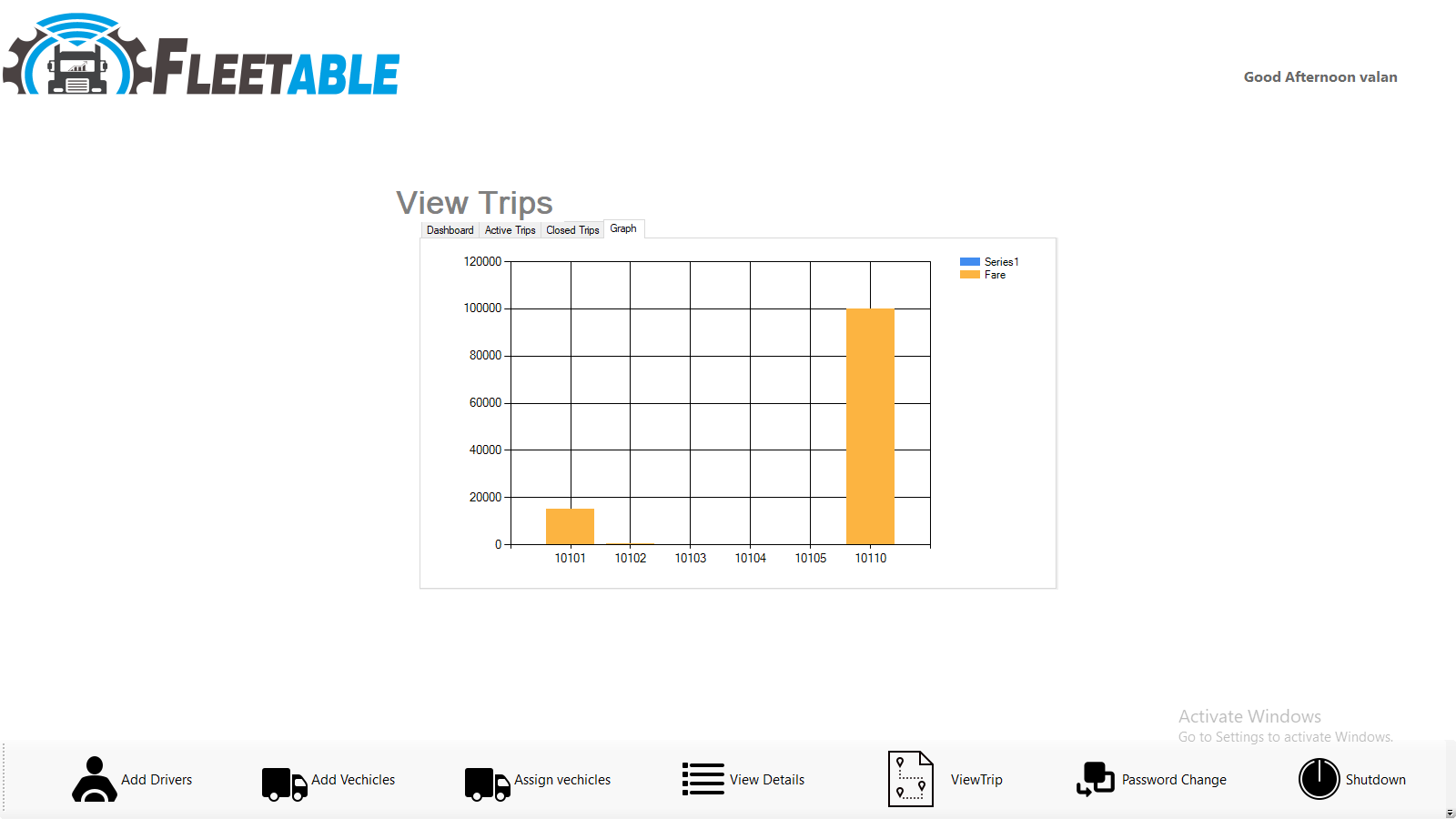
**View Active Trip Details**

****

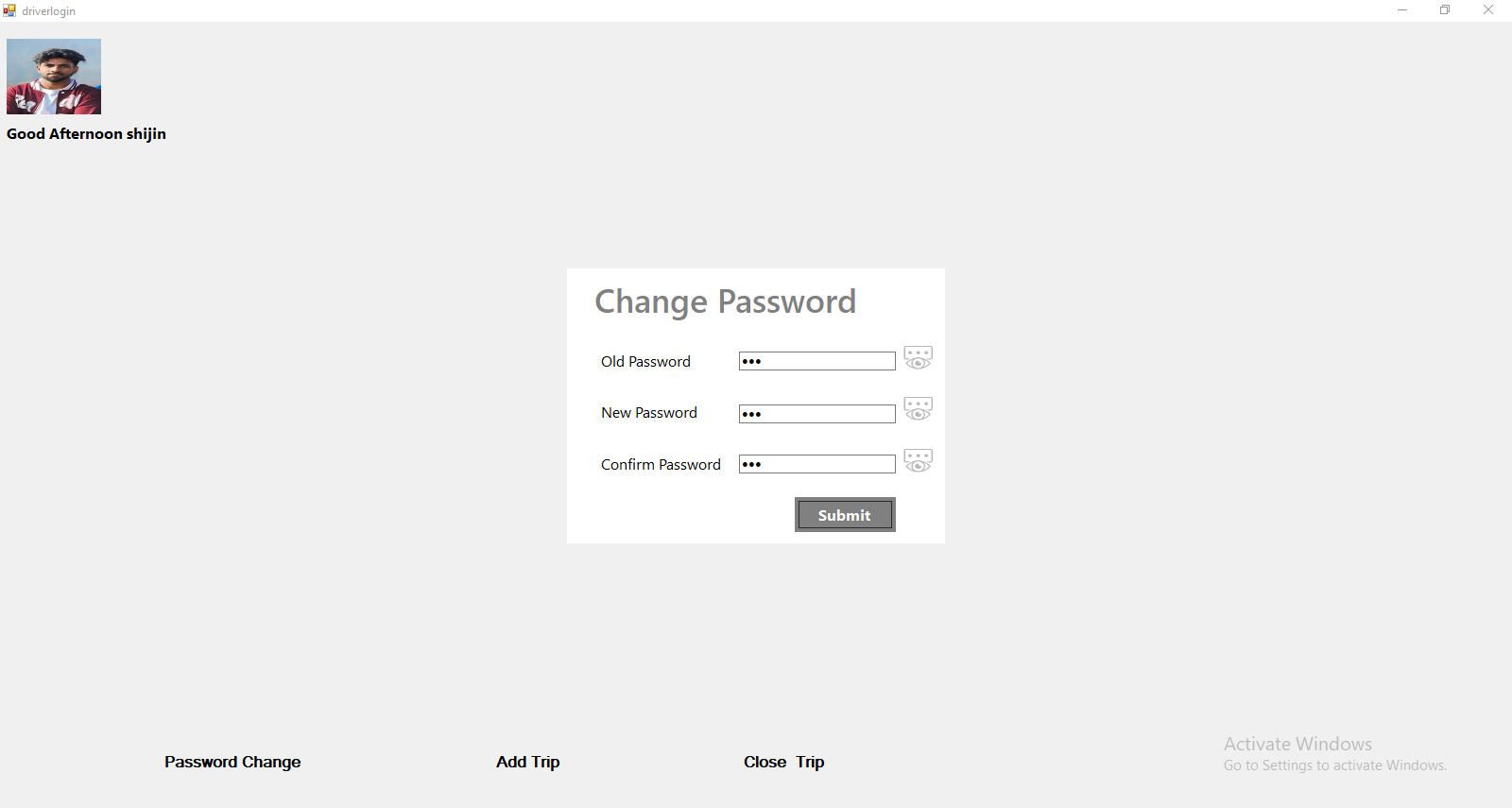
**View Close Trip Details**

****

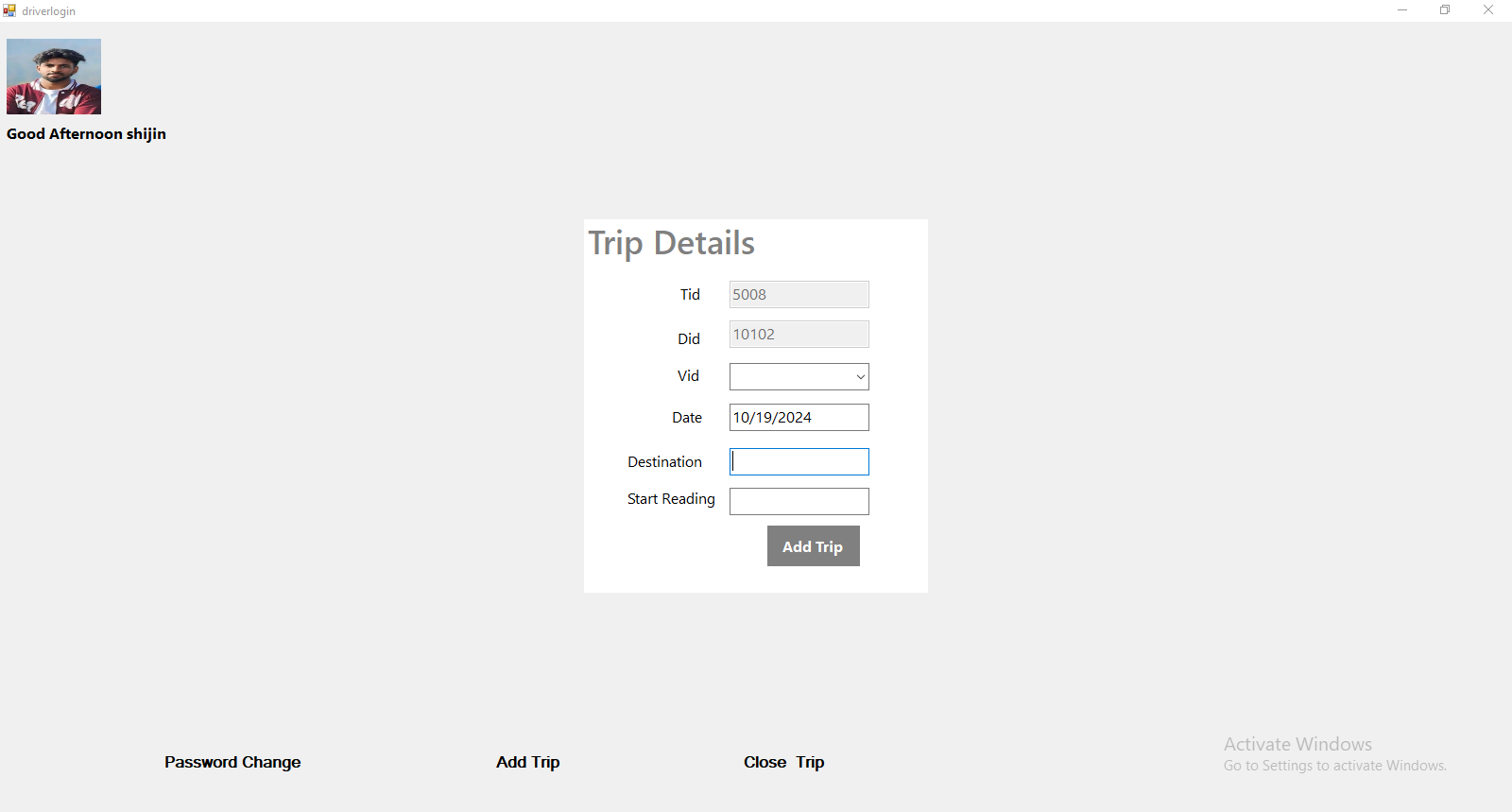
**View Graph Details**

****

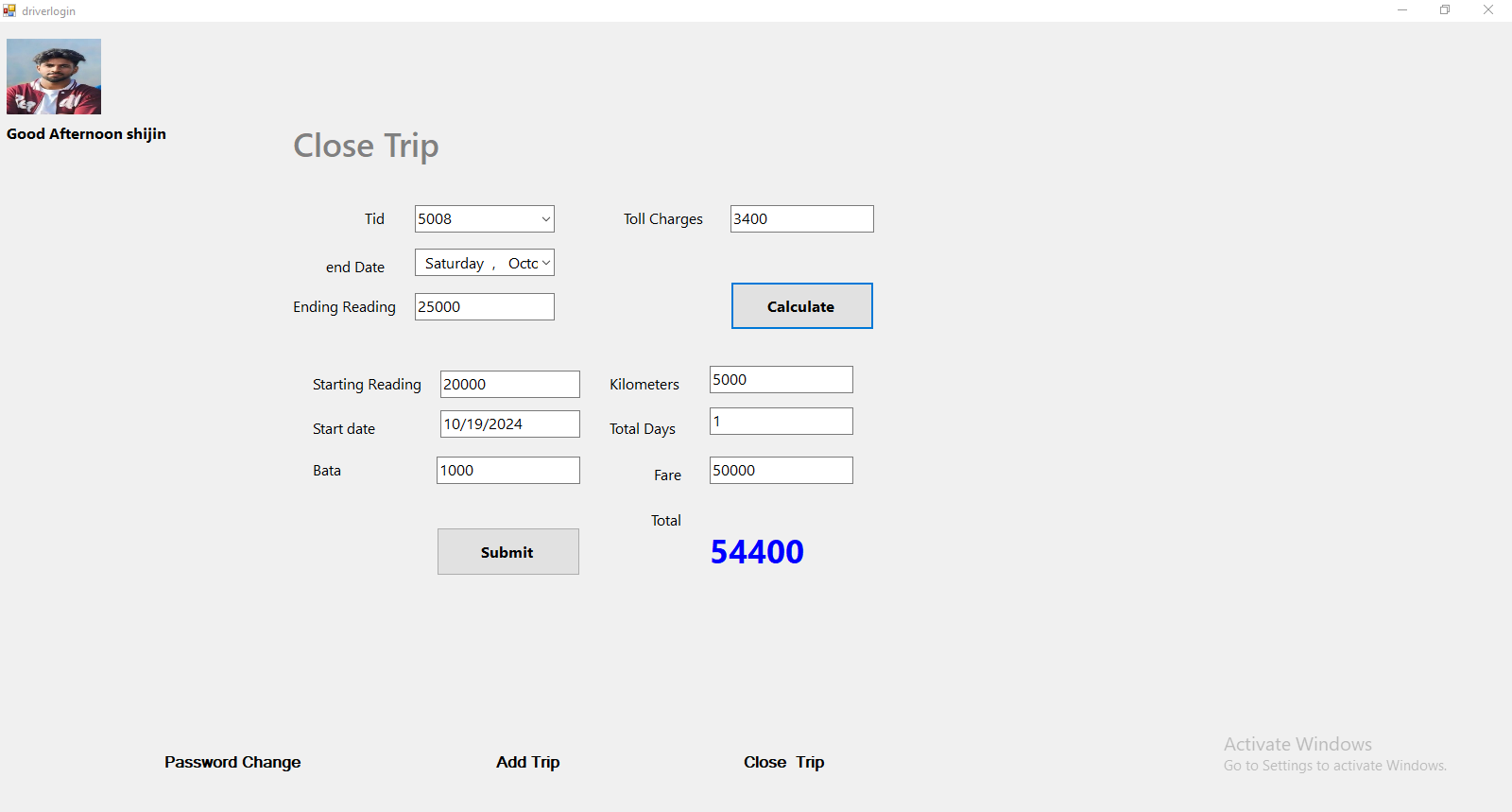
**Driver Change Password**

****

**Driver View Trip**

****

**Driver Close Trip**

****

**CHAPTER 7**

**CONCLUSION**

The proposed enhancements will significantly elevate the Fleet Management System, transforming it into a more efficient, scalable, and future-ready platform tailored to meet the demands of modern fleet operations. By integrating real-time GPS tracking, the system will provide unparalleled visibility into vehicle locations, enabling optimized route planning that adapts to traffic conditions and minimizes delays. The incorporation of IoT sensors will facilitate advanced fleet monitoring and data collection, allowing for proactive maintenance and better management of vehicle health, thereby reducing downtime and enhancing overall performance. Additionally, AI-powered demand forecasting will empower administrators to predict service demands accurately, leading to more effective trip scheduling and resource allocation. The development of a mobile app for drivers will enhance accessibility and streamline communication, allowing them to receive updates and make informed decisions on the go. Furthermore, secure payment gateway integration will simplify financial transactions for seamless fare collection. A driver performance evaluation system will foster accountability and excellence, encouraging safe driving practices through performance-based incentives. Collectively, these upgrades will improve daily operational efficiency while establishing a robust framework for long-term sustainability and profitability, empowering fleet operators to navigate the challenges of a dynamic transportation landscape and ensuring a competitive edge in the industry.

**CHAPTER 8**

**FUTURE ENHANCEMENT**

The proposed enhancements to the Fleet Management System will significantly improve its efficiency and functionality, creating a robust platform that addresses the diverse needs of modern fleet operations. Integrating real-time GPS tracking will allow for complete visibility into vehicle locations and facilitate optimized route planning based on live traffic data, reducing delays and improving punctuality while minimizing fuel consumption. A preventive maintenance module will track vehicle health and performance, issuing timely service alerts to minimize downtime and extend vehicle life. The development of a dedicated mobile app will enhance driver accessibility, enabling them to manage assignments and receive push notifications while providing in-app navigation for more efficient journeys. Additionally, a fuel management system will monitor consumption and identify inefficiencies, helping administrators control operational costs effectively. Integrating IoT sensors will enhance fleet monitoring by collecting real-time data on vehicle performance and driver behavior, allowing for early detection of anomalies to maintain safety and reliability. A driver performance and incentive management system will evaluate key metrics, promoting safe driving practices through rewards for top performers. AI-powered demand forecasting will optimize trip scheduling and resource allocation, while the integration of secure payment gateways will streamline financial transactions, allowing for online payments and eliminating the need for manual record-keeping. Together, these enhancements will create a comprehensive and scalable solution that drives operational efficiency, improves productivity, and positions the fleet for long-term success in a competitive landscape.

**REFERENCES**

1. John Sharp, Microsoft Visual C# Step by Step, Microsoft Press, 2012
2. Andrew Troelsen and Philip Japikse, Pro C# 7: With .NET and .NET Core, Apress, 2017
3. William Assaf, Randolph West, Sven Aelterman, and Mindy Curnutt, SQL Server 2017 Administration Inside Out, Microsoft Press, 2018
4. Martin Kleppmann, Designing Data-Intensive Applications, O'Reilly Media, 2017
5. Andrew Hunt and David Thomas, The Pragmatic Programmer: Your Journey to Mastery, Addison-Wesley, 2019
6. Joseph Albahari and Ben Albahari, C# 7.0 in a Nutshell: The Definitive Reference, O'Reilly Media, 2017
7. Heather Adkins, Betsy Beyer, Paul Blankinship, et al., Building Secure and Reliable Systems: Best Practices for Designing, Implementing, and Maintaining Systems, O'Reilly Media, 2020
8. Jon Skeet, C# in Depth, Manning Publications, 2019
9. Lisa Bodell, Kill the Company: End the Status Quo, Start an Innovation Revolution, Per Capita Publishing, 2016
10. James O. Coplien and Neil B. Harrison, Organizational Patterns of Agile Software Development, Cambridge University Press, 2004