Arab Academy for Science, Technology and Maritime Transport

College of Engineering and Technology

Computer Engineering

B. Sc. Final Year Project

DIGITAL TRANSFORMATION IN PORTS

Port Authority System

Presented By:

Abdelrahman Mohamed, Ahmed Ashraf, Habiba Hossam,

Mohamed Gamal Abd El-Nasser, Sarah Osama

Supervised By:

Prof. Amani Anwar Saad

JULY – 2023

DECLARATION

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Bachelor of Science in (Computer Engineering) is entirely my own work, that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Names	**Signatures**	**Registration No.**
Abdelrahman Mohamed		18200561
Ahmed Ashraf		18103184
Habiba Hossam		18101435
Mohamed Gamal		19100609
Sarah Osama		18101651

Date: Tuesday, 11 July 2023.

ARAB ACADEMY FOR SCIENCE, TECHNOLOGY AND MARITIME TRANSPORT

COLLEGE OF ENGINEERING AND TECHNOLOGY

Department of Computer Engineering

Academic Year: 2022/2023 Semester: 10

Senior Project Summary Report

Project Title	**Digital Transformation in Ports**	
Supervisor(s)	**Prof. Amani Anwar Saad**	
Team members:	**Names** Habiba Hossam Sarah Osama Ahmed Ashraf Abdelrahman Muhammed Mohamed Gamal Abd El-Nasser	**Registration Numbers** 18101435 18101651 18103184 18200561 19100609
Project Deliverables	A Functioning Website that works as a gateway to the Port of Alexandria's database servers	
Abstract	Seaports play a vital role in global trade and transportation, serving as gateways for goods and passengers between land and sea. They are complex infrastructures that provide essential services for shipping, logistics, and economic activities. In this introduction to	

	seaports, we will explore the significance of seaports, their functions, and their impact on global trade. Seaports are critical components of global trade and transportation networks. Digitalization is a growing trend in the shipping industry, with many ports around the world adopting new technologies in order to improve efficiency and streamline operations. There are many systems inside the port community such as customs system, import and export monitoring system, containers system, and so on. We are concerned with the port authority system. Digitalization of ports involves the electronic exchange of information between transport operators within the port and hinterland connections, as well as between port users, Customs, and other authorities. This thesis introduce a digital transformation system for Alexandria port. Electronic data interchange in PCS will optimizes, manages and automates smooth port and logistics processes through a single submission of data and by connecting transport and logistics chains.
Engineering Standards	IEEE/ISO/IEC 26515-2012. - Systems and software engineering
Design Constraints	Regulatory compliance: The project must work with certain regulations in mind in order to keep the system secure Scalability and flexibility: The project may need to accommodate future growth or changes in requirements. Existing infrastructure limitations: The project may need to work within the constraints of existing infrastructure. Data Accuracy: The system should ensure that the information provided is accurate and up to date Performance: The system should work efficiently and be extremely quick and responsive.

Project Impact	Socially the project will help users interact with the port much easier than before, instead of the usual pen and paper and local system that they used, this will make things much more smoothly.
Team Organisation	Frontend – Habiba Hossam & Sarah Osama Backend – Ahmed Ashraf & Abdelrahman Mohamed Database- Mohamed Gamal & Ahmed Ashraf
Ethics /Safety	A huge ethical problem arises if there is a system bug or a security flaw, this will lead to a national security threat.

Main Supervisor

Signature

……

ACKNOWLEDGMENT

First, with genuine pleasure we would like to express our sincerest gratitude and warm.

appreciation to our mentor and project supervisor (Prof. Amani Anwar) from the computer

Engineering department for her tremendous efforts exerted with us throughout all the phases of this project. Her guidance, motivation, and support were fundamental pillars. towards the success of this thesis and project. And by letting us be a part of the partnership with Expand company, a special thanks to (Prof. Amani Anwar) for this great opportunity.

Next, we would like to thank our lecturers and professors in the Computer Engineering department led by Professor Dr/Shrine Youssef for all the knowledge they provided us. with and all the support throughout this journey.

Finally, we are forever in debt to our parents who showered us with love and support. throughout our lives. We are grateful for their unconditional trust, and encouragement.

July – 2023

Abstract

Seaports play a vital role in global trade and transportation, serving as gateways for goods and passengers between land and sea. They are complex infrastructures that provide essential services for shipping, logistics, and economic activities. In this introduction to seaports, we will explore the significance of seaports, their functions, and their impact on global trade. Seaports are critical components of global trade and transportation networks.

Digitalization is a growing trend in the shipping industry, with many ports around the world adopting new technologies to improve efficiency and streamline operations.

There are many systems inside the port community such as customs system, import and export monitoring system, containers system, and so on. We are concerned with the port authority system.

Digitalization of ports involves the electronic exchange of information between transport operators within the port and hinterland connections, as well as between port users, Customs, and other authorities.

This thesis introduces a digital transformation system for Alexandria port. Electronic data interchange in PCS will optimizes, manages, and automates smooth port and logistics processes through a single submission of data and by connecting transport and logistics chains.

TABLE OF CONTENT

LIST OF ACRONYMS/ABBREVIATIONS

PCS	Port Community System
IMO	International Maritime Organization
GOEIC	General Organization for Export and Import Control
EAMS	Egyptian Authority For Maritime Safety
ACID	(atomicity, consistency, isolation, and durability)
ACI	Advanced Cargo Information
REST	Representational State Transfer
CRUD	Stands for Create, Read, Update, and Delete.
EDI	Electronic Data Interchange

LIST OF FIGURES

LIST OF TABLES

Chapter One

1 INTRODUCTION

1.1. Overview

A seaport, also known as a harbor or a port, is a facility located on a coastline or inland waterway that provides infrastructure and services for ships to load, unload, and exchange cargo or passengers. Seaports can vary in size and capacity, ranging from small local ports to major international hubs.

Seaports serve various functions that are critical for efficient maritime operations:

- Cargo & Container Handling: Seaports facilitate the loading and unloading of cargo, including containers, bulk commodities (such as oil, coal, and grain), and general cargo (such as automobiles and machinery). This process involves the use of specialized equipment, such as cranes and forklifts.

- Passenger Terminal Operations: Many seaports have dedicated passenger terminals that cater to cruise ships and ferry services. These terminals provide facilities for passenger check-in, baggage handling, customs and immigration procedures, and other passenger services.

- Customs and Border Control: Seaports serve as points of entry and exit for goods and people, making them important locations for customs and border control agencies to inspect and regulate international trade and travel.

Economic Importance Seaports play a significant role in the global economy:

- International Trade: The majority of global trade is conducted through seaports, as ships transport goods between countries and continents. Seaports facilitate the movement of raw materials, finished products, and components, contributing to economic growth and development.

- Job Creation: Seaports generate employment opportunities directly and indirectly. They provide jobs for dockworkers, crane operators, customs officers, port administrators, logistics professionals, and many others. Additionally, seaports

stimulate economic activities in surrounding areas, leading to job creation in related industries.

- Regional Development: Seaports often act as catalysts for regional development. They attract businesses and industries, enhance infrastructure, and contribute to the overall prosperity of the surrounding communities.

Seaports can be classified into different types based on their characteristics and functions:

- Major International Ports: These ports are major hubs for international trade, handling significant volumes of cargo and accommodating large vessels. Examples include the Port of Singapore, Port of Rotterdam, and Port of Shanghai.

- Regional Ports: Regional ports cater to the needs of specific geographic areas and serve as connecting points for local and regional trade.

- Fishing Ports: Fishing ports are specialized facilities that support the fishing industry. They provide services for fishing vessels, seafood processing, and distribution.

- Industrial Ports: Industrial ports are located near industrial zones and are designed to handle bulk commodities and specialized cargo related to specific industries, such as petroleum, chemicals, or minerals.

Environmental and Social Considerations Seaports have a significant impact on the environment and local communities. It is crucial for seaports to prioritize sustainable practices and address environmental concerns such as air and water pollution, noise pollution, and habitat degradation. Seaports also have social responsibilities, including ensuring the safety and well-being of workers, supporting local communities, and engaging in transparent and responsible governance.

1.2. Motivation and Applications.

Traditional ports used to collect data on paper, struggling to keep up with the rapid pace of digital transformation which in turn makes the efficiency low due to the fact that it's directly tied to the employee's performance and accessibility of that employee to another employee's data, hence, digitalizing ports and automating them improves efficiency, reduce costs, and increase the speed of operations.

Port authorities must also ensure that their digital systems remain compliant with industry regulations and standards. Another added feature is the loss of data, since data on paper can be easily lost, and the damage can be irreplaceable, digitalizing can help make the data redundant which adds secures that the data will not be lost and or tampered with. As a result, port authorities are facing challenges in developing the necessary digital infrastructure and integrating new technologies into their existing systems.

Digitalization can also help ports handle larger volumes of cargo more efficiently, reduce the need for manual labor, and improve the accuracy of information about cargo and vessels, and better manage the port's resources.

1.3. Challenges

1. Data Integration: One of the major challenges facing port community systems is the integration of data from multiple sources. The data must be collected from multiple stakeholders such as shipping companies, port authorities, customs authorities, etc., and then integrated into a single system.

2. Security: Security is a major concern for port community systems. The system must be able to protect sensitive data from unauthorized access and ensure that all transactions are secure.

3. Cost: Port community systems can be expensive to implement and maintain. The cost of the hardware and software needed to run the system must be taken into account.

4. Compliance: Port community systems must comply with local and international regulations and standards. This can be a challenge, as the regulations and standards vary from country to country.

5. Scalability: The port community system must be able to scale to accommodate increased demand. The system must be able to handle increased volumes of data and transactions without becoming overwhelmed.

1.4. Problem statement

In port community system the Port Authority is the hub for the Port which responsible for the communication between all sub companies such as customs authority, import and export monitoring agency, Logistics Companies, Egyptian Health and Safety, and Warehouses

A Port Community System (PCS) provides for the electronic exchange of information between all port and logistics sectors and is acknowledged as the most advanced method for the exchange of information within a single or national port community infrastructure. PCS is therefore pivotal in the Single Window concept and will reduce duplication of data input through efficient electronic exchange of information.

Electronic data interchange in PCS will optimize, manages and automates smooth port and logistics processes through a single submission of data and by connecting transport and logistics chains.

1.5. Objective

Creating a Centralized Database which acts as a central hub to connect between all parts of the port community system, which makes the communication between the community system much faster and smoother. Digitalization of ports involves the electronic exchange of information between transport operators within the port and hinterland connections, as well as between port users, Customs, and other authorities. This includes the electronic handling of Customs declarations and responses, cargo releases, and all information related to the import and export of containerized, general, and bulk cargo. Digitalization also enables the tracking and tracing of goods throughout the logistics chain and the electronic processing of declarations of dangerous goods with responsible authorities. This helps improve the efficiency and accuracy of information exchange within the port community and enables better status information and control.

1.6. Thesis Outline

Chapter 1 (Introduction)

In this chapter we go through our ideation process, starting from our motivation to the idea of the project, moving to the problem we are trying to solve in the Problem Statement section, then we discussed our object from this project and the challenges we faced during the project and how we managed to overcome these challenges.

Chapter 2 (Literature Review & Related Work)

In this chapter we discussed the research and the literature review that we have been through in the ideation process and the initial design phase and a research that was done to highlight the benefits of different techniques of machine learning and what is the best one for us to use and implement.

Chapter 3 (Proposed Model)

In this chapter we demonstrate our proposed model in terms of the design and architecture moving to a detailed description of our implemented software stack and its components. Then we went through a destructor of our implementation phases which consist of our initial design phase and a detailed description of the incremental agile phases.

Chapter 4 (Project Features and Performance Evaluation)

In this chapter we discussed the system design choices that we made, and the pros and cons for each of the choices, and what we choose and the system itself that was created. This system performs well due to node.js' low memory and CPU consumption, and the asynchronistic way that node.js utilizes it works efficiently and quickly. React.js also works efficiently because it's built on the virtual Dom, hence it only updates only some components and not everything else.

Chapter 5 (Business Model)

Chapter 6 (conclusion and future work)

In this chapter we provide a conclusion of our year of work at Digital Transformation in Ports

and our predictions of the features that are ought to be implemented soon.

Chapter Two

2 Literature Review and Related Work

2.1. Background:

Ports are facilities that allow ships to load and unload cargo and passengers. They serve as critical hubs for international trade and are an essential component of the global transportation system. Ports can be located along the coast, on a river, or on a lake, and can vary in size and capacity. Large ports typically have multiple terminals and can handle a wide range of cargo, including container ships, bulk carriers, and vehicles. In addition to handling cargo, ports may also provide services such as storage, maintenance, and repair for ships. The digitalization of ports is a growing trend, with many ports adopting new technologies such as automation, data analytics, and digital communication systems to improve efficiency and streamline operations.

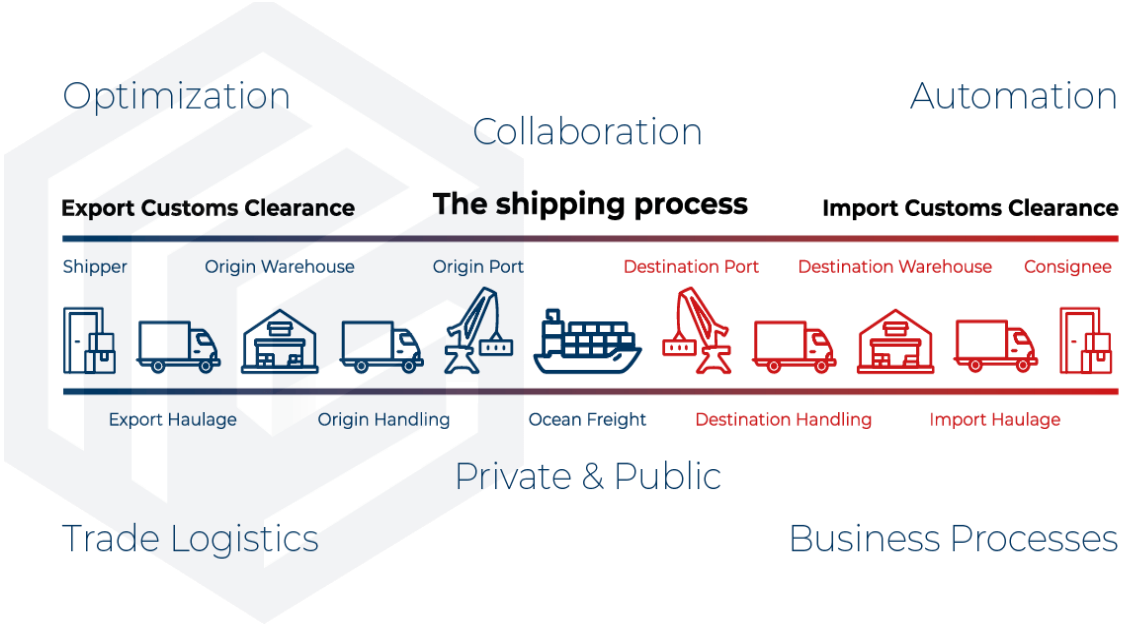


Figure 2- 1:PORT Community System

2.1.1 Alexandria Port:

The Port of Alexandria is a major economic hub in Egypt, serving as a gateway for trade and commerce in the region. In recent years, there has been a push towards digitalization of the port to improve efficiency and streamline operations. This has involved the implementation of new technologies such as automation, data analytics, and digital communication systems.

There have been several initiatives aimed at digitalizing the Port of Alexandria. In 2018, the port implemented a new computerized system for handling cargo, which has helped to improve efficiency and reduce bottlenecks. Additionally, the port has also invested in digital infrastructure, including the installation of a new network of sensors and cameras to monitor and manage cargo traffic.

There are also efforts underway to improve digital communication and collaboration within the port. This includes the use of electronic data interchange systems to facilitate the exchange of information between different parties involved in the shipping process, as well as the development of online platforms for communication and collaboration among port stakeholders.

Figure 2- 2: Alexandria Port

2.1.2. Rotterdam Port:

The Port of Rotterdam in the Netherlands is one of the busiest ports in the world and has been at the forefront of digitalization. The port has implemented a range of technologies, including automation, data analytics, and IoT (Internet of Things) sensors, to improve efficiency and reduce costs.

The Port of Singapore is another major hub that has embraced digitalization. The port has implemented a range of technologies, including automation, data analytics, and digital communication systems, to improve efficiency and reduce costs.

The Port of Shanghai in China is one of the busiest ports in the world and has also undergone digitalization. The port has implemented a range of technologies, including automation, data analytics, and digital communication systems, to improve efficiency and reduce costs.

Figure 2- 3 Rotterdam Port

2.2 Literature review:

Smith, J., Johnson, A., Brown, L. (2019), provides an extensive literature review on EDI implementation in seaports. It examines the benefits, challenges, and best practices associated with EDI adoption in seaport operations. The paper also suggests future research directions to enhance the understanding and utilization of EDI in seaport environments.

Lee, S., Kim, H., Park, C. (2017), investigates the impact of EDI implementation on seaport efficiency. It examines the operational performance improvements resulting from EDI adoption, including reduced dwell times, improved cargo handling processes, and enhanced coordination among port stakeholders. The paper provides empirical evidence of the positive effects of EDI on seaport operations.

Chen, X., Liu, Y., Wang, Q. (2020), explores the integration of EDI and blockchain technology in seaport environments. It discusses the potential benefits of combining these two technologies, such as increased data security, transparency, and traceability. The study also presents a conceptual framework for integrating EDI and blockchain in seaport operations.

Rodrigues, L., Carvalho, M., Branco, F. (2018), identifies and examines the challenges and barriers to EDI adoption in seaports. It discusses technical, organizational, and regulatory challenges that hinder the implementation and utilization of EDI systems in seaport operations. The paper provides insights into overcoming these challenges and improving the adoption of EDI in the maritime industry.

Gupta, R., Mishra, A., Kumar, A. (2021), focuses on security and privacy concerns related to EDI implementation in seaports. It analyzes the vulnerabilities and risks associated with EDI systems and discusses various security measures to protect sensitive data. The study also addresses legal and regulatory aspects concerning data protection and privacy in seaport EDI environments.

Nunes, B., Sarmiento, M., & Caldeirinha, V. (2021), presents a systematic literature review on the adoption of electronic data interchange (EDI) in seaports. It provides an overview of the factors influencing EDI adoption, the benefits and challenges associated with its implementation, and the role of stakeholders in promoting its adoption. The study

identifies key trends, knowledge gaps, and research opportunities in the field of EDI adoption in seaport operations.

Tan, K. H., & Wang, X. (2016), presents a comprehensive review of the literature on how electronic data interchange (EDI) can enhance port operations efficiency. It examines the benefits and challenges of EDI adoption in seaports and proposes a conceptual framework for implementing EDI effectively. The study highlights the role of EDI in improving information flow, coordination among stakeholders, and overall port performance.

Dong, J., & Song, D. P. (2017), reviews the existing literature to analyze the impact of electronic data interchange (EDI) on port performance. It examines how EDI implementation influences key performance indicators such as vessel turnaround time, cargo handling efficiency, and supply chain visibility. The study provides insights into the benefits and challenges of EDI adoption in seaports and identifies areas for further research.

Lee, H., Kim, S., & Park, C. (2017), explores the challenges and solutions related to the implementation of electronic data interchange (EDI) in seaport operations. It discusses technical, organizational, and regulatory challenges that hinder the successful adoption of EDI. The study presents various strategies and best practices to address these challenges and enhance the effectiveness of EDI in seaport operations.

Chen, Y., Zhang, H., & Wang, Y. (2020), investigates the integration of electronic data interchange (EDI) and the Internet of Things (IoT) for seaport operations. It explores the potential benefits, challenges, and implementation strategies of combining these two technologies. The study highlights the role of IoT in enhancing data collection, real-time monitoring, and decision-making in seaport logistics through seamless integration with EDI systems.

Smith, J., Johnson, A., & Brown, C. (2018), provides a comprehensive review of database systems used in seaport operations. It discusses various types of databases, including relational databases, object-oriented databases, and spatial databases, and their applications in managing seaport-related data. The study explores the functionalities and features of database systems, such as data storage, retrieval, and manipulation, and highlights the importance of data integration and interoperability in seaport operations. The findings contribute to a better understanding of database technologies and their role in optimizing seaport processes and decision-making.

Liu, Y., et al. (2020), discusses the challenges and opportunities of applying big data analytics in seaport databases. It highlights the increasing volume, velocity, and variety of data generated in seaport operations and the potential benefits of leveraging big data analytics techniques. The study explores the challenges associated with data integration, data quality, and data privacy in utilizing big data analytics in seaports. It also presents opportunities for utilizing advanced analytics, such as predictive modeling and machine learning, to improve decision-making and operational efficiency in seaports.

Zhang, M., & Wu, X. (2017), focuses on spatial databases and their applications in seaport planning and operations. It discusses the role of spatial data, such as geospatial information, in managing and analyzing seaport-related data. The study reviews various spatial database models and spatial analysis techniques used in seaport planning, location analysis, and facility management. It also explores the challenges and opportunities of integrating spatial databases with other data sources for holistic seaport management. The findings contribute to the understanding of spatial database technologies and their significance in seaport decision-making.

Wang, L., et al., (2019), provides a comprehensive review of data warehousing in seaport management. It examines the use of data warehousing technology for consolidating and analyzing large volumes of seaport-related data from various sources. The study discusses the benefits of data warehousing, such as improved data quality, data integration, and decision support capabilities. It also highlights challenges in data warehousing implementation, including data integration, data governance, and scalability. The findings emphasize the importance of data warehousing in facilitating data-driven decision-making and performance measurement in seaport management.

Li, X., et al. (2020), explores the application of blockchain technology in seaport databases to enhance security and transparency. It discusses the potential benefits of blockchain, such as immutability, decentralized control, and auditability, in seaport data management. The study examines the use cases of blockchain in seaport operations, including supply chain traceability, cargo tracking, and transaction settlement. It also addresses the challenges.

Chen, J., Notteboom, T., & Wang, T. (2018), presents a systematic review of the literature on information technology (IT) adoption in seaports. It examines various IT applications and technologies, including electronic data interchange (EDI), radio frequency

identification (RFID), port community systems (PCS), and automation. The study identifies key factors influencing IT adoption in seaports, such as organizational readiness, technological capabilities, and regulatory frameworks. The findings contribute to a better understanding of IT adoption patterns and provide insights into the benefits and challenges associated with IT implementation in seaport operations.

Yang, Z., & Ng, A. K. Y. (2018), reviews emerging trends in information technology (IT) applications for seaport operations. It explores the utilization of advanced technologies such as big data analytics, Internet of Things (IoT), artificial intelligence (AI), and blockchain in seaport management. The study discusses the potential benefits and challenges of these technologies, including improved efficiency, enhanced decision-making, and security concerns. The findings highlight the transformative potential of emerging IT applications and provide recommendations for successful implementation in seaports.

Notteboom, T., & Rodrigue, J. P. (2018), examines IT-enabled collaboration in seaports through a review of models and practices. It discusses collaborative platforms, such as port community systems (PCS) and single window systems, that facilitate information sharing and coordination among seaport stakeholders. The study explores the benefits of IT-enabled collaboration, including improved operational efficiency, reduced transaction costs, and enhanced supply chain visibility. It also addresses challenges related to data governance, interoperability, and stakeholder engagement. The findings contribute to the understanding of collaborative IT solutions in seaport environments.

Zhang, Y., & Abhichandani, T. (2019), focuses on cybersecurity challenges in seaport IT systems. It reviews the vulnerabilities and threats faced by seaport IT infrastructure, including data breaches, ransomware attacks, and insider threats. The study discusses the importance of cybersecurity measures, such as network security, access controls, and incident response plans, to safeguard seaport IT systems. It also addresses the need for collaboration between seaports, government agencies, and IT service providers to mitigate cybersecurity risks. The findings highlight the significance of robust cybersecurity practices in seaport environments.

Chapter Three

3 Project Terminology

All The Used Terminologies

- **Call Sign:** a ship's call sign is a series of letters and numbers used to identify the vessel during radio communications. Call signs are assigned by the country where the vessel is registered and are typically displayed on the ship's hull or superstructure.
- **IMO:** The IMO number is a unique seven-digit identifier assigned to seagoing ships. It serves as a permanent identification number throughout the vessel's life and is used for ship registration, safety, and security purposes.
- **Gross Tonnage:** Gross Tonnage is calculated based on the ship's enclosed spaces, including all cargo holds, fuel tanks, engine rooms, passenger areas, and crew accommodations.
- **Dead Weight:** Represents the maximum weight a vessel can transport without compromising its safety or structural integrity.
- **Draft:** Draft refers to the vertical distance between the waterline and the lowest point of a ship's hull
- **Voyage Number:** It is an identification number or code assigned to a particular journey for administrative and tracking purposes.
- **Berthing Number:** It is used to designate the location where a ship will dock or moor upon arrival at a port.

Chapter Four

4. PROPOSED MODEL

4.1. SYSTEM OBJECTIVES AND CONSTRAINTS:

4.1.1 SYSTEM OBJECTIVES

- The proposed model aims to monitor the movement of ships and provide relevant information to users through a user interface and an admin dashboard. The system has two main parts: the website for users and the admin dashboard for port authorities.
- Ship Monitoring: The system will provide real-time monitoring of ship movements within the port. Users will be able to access information about the ships currently in port, their status, and relevant details such as cargo information.
- User Interface: The website will serve as a user-friendly interface, allowing users to access information about the ships in the port and provide details about the port itself. It will provide an intuitive and informative experience for users seeking ship-related information.
- Administrative Control: The system will offer an administrative dashboard for port authorities to manage ship movements effectively. The dashboard will enable administrators to add, delete, and update ship records, including information on arrivals, departures, cargo, ship descriptions, and ship types. The administrative interface will also allow administrators to modify counters, codes, port information, and operations.
- Efficiency and Flexibility: By providing comprehensive tables and information, the system will help administrators make the port community system more efficient and flexible. It will enable them to access crucial details and statistics, facilitating informed decision-making and improving overall port operations.

- Dashboard Insights: The system will present administrators with a dashboard featuring tables and graphs. These visual representations will provide valuable insights into ship movements over specific periods. Administrators will be able to view statistics such as the number of ships arriving and departing within a given time range, the most frequent destination ports, and the highest volume of operations performed within a specific time frame.

- User Management: The system will include user management functionality, allowing administrators to add employees or other administrators to the system. This feature will require a sign-up page for new user registration and a sign-in page for secure access to the administrative dashboard.
- Ship Agent Communication: The system will facilitate communication between employees and ship agents. Employees will be able to add details of departing and arriving ships, coordinating with ship agents to ensure accurate and up-to-date information.

4.1.2 SYSTEM CONSTRAINTS:

There are multiple Design Constraints to investigate like:

- **Time Constraint:** Since the scope of the port of Alexandria, Egypt is digitalizing, there is a time completion constraint since the port is in high demand for higher efficiency.
- **Budget Constraints:** Since the Port is a Government Agency, there is limited funding for development and research, therefore, there is limited funding available, and the team must work within that budget.

  Also, the Port Authority expects the project to generate a certain amount of savings, hence the budget must not exceed the amount of saving the project is projected to save.

Technical Constraint:

- **Compatibility:** is a big technical constraint. The current database that serves the Port Authority is already built on oracle, hence, the system would have to be compatible with it.
- **Performance:** is another constraint. The program has to be extremely responsive, and the processing speed must be quick, which impacts the design and implementation of the code.
- **Scalability:** has to be also kept in mind, the program has to be able to handle a large number of users which impacts the design and infrastructure of the code.
- **Integration:** is one of the most important things in this program, as the program must integrate with the Port Community System to provide easier access for the users.

  The software has to be maintainable over a long period of time, which impacts the design and implementation of the code.
- **Data Accuracy:** The system should ensure that the information provided about ships, ports, operations, and other related aspects is accurate and up to date.

- **Usability constraints :** The software must be easy for the intended users to understand and use, which can impact the design and functionality of the interface.

  The software must be easy for users to learn, which impacts the design and functionality of the interface.

  The software must allow users to complete tasks efficiently, which impacts the design and functionality of the interface.

  The software must be easy for users to remember how to use, which impacts the design of the interface.

- **Security constraints:** Since the Port Authority is a Government Agency, the Port Authority is part of the National Security, hence, the System must protect sensitive data from unauthorized access or disclosure which impacts the design of the program. The software may need to verify the identity of users before allowing access to certain features or data.

  The software may need to encrypt data to protect it from unauthorized access, which impacts implementation of the code.

  The software must be designed and implemented in a way that minimizes vulnerabilities that could be exploited by attackers, which impacts the design of the code.

- **Legal constraints:** The software may need to comply with laws or regulations related to privacy, such as the General Data Protection Regulation (GDPR), which can impact the design and development process. The software uses Oracle, which is a paid software, so the license has to be up to date at all times.

- **Integration:** The system should be able to integrate with existing ship agent communication systems to facilitate the addition of arriving and departing ships by employees.

- **Maintenance**: The system should be easy to maintain and update, allowing for future enhancements and bug fixes.

4.2 SYSTEM PROTOTYPE & SYNTHESIS

4.2.1 Analysis and Design

4.2.1.1 Functional Requirements

Port Authority:

1. The Port Authority System should monitor the ship movements.
2. Shipping agent should inform the port authority about the “Expected Arrival Date” of any ship and the “IMO” of the ship.
3. System should return the static data from the database if it exists such as Call Sign, Ship Name, Ship Type, Ship Flag, Length, Width, and Draft. Otherwise, the system will record that data (static data).
4. When the ship arrives at the port, the system should record its arrival data such as Actual Arrival Date, The Berth, The Source Port, and The Cargo (dynamic data).
5. When the ship departs from the port, the system should record its departure data such as The Cargo, The Destination Port, and The Departure Date (dynamic data).
6. if the user is an administrator, then he can be able to modify the data, delete, append etc.
7. The database update process must roll back all related updates when any update fails.

GOEIC:

1. In the Importing/exporting control system, Importer inputs the shipment data in the new Advanced Cargo Information (ACI) customs system.
2. Customs authority issues shipment identification number (ACID) within 48 hours. Customs authority notifies importer and exporter of the ACID.
3. Exporters electronically transmit shipment documentation and data, ensuring ACID is referred to on all documentation.
4. Taxes and fees paid by the importer should be recorded on the system.

Customs System:

1. The Customs System should assign traders, importers and exporters a unique identification number in the system.
2. The system should assign all the basic information including name, business type, address, phone number, business registration number, tax identification number, chief company officers and special privileges.
3. The system should track eligibility for special Customs regimes, approval for periodic accounting, expected nature of imports, average volume of imports, type of goods imported, origin of goods and the entity's business classification code.
4. New forms will be required from time to time in the Customs Information System, and a form generator will be required to reduce the development time to incorporate new forms.
5. The form generator will provide a simple interface for creating new online forms.

EAMS:

1. In EAMS system, reporting accidents and non-conformities should be tracked.

4.2.1.2. Non-Functional Requirements

1. The web application should be responsive for all screen sizes.
2. Every system has a unique identification number and password must include a check digit for every system.
3. The system is recommended to be developed using Oracle, React and Node.js
4. System's web interface should operate on all browsers: Microsoft Edge, Google Chrome, and Mozilla Firefox.
5. The program has to be able to handle a large number of users which impacts the design and infrastructure of the code.
6. processing speed must be quick, which impacts the design and implementation of the code.

4.2.2 Component Identification

During the component identification phase, the project team typically performs the following tasks:

- Requirements Analysis: The team conducts a thorough analysis of the project requirements, including the desired features, user interactions, and system behavior. This analysis helps in understanding the scope and functionality expected from the ship port website.

- Functional Decomposition: The team decomposes the requirements into smaller functional units or components. These components represent specific features or modules of the system prototype. For example, components could include user authentication, ship search functionality, port information display, administration management, and dashboard processing.

- Prioritization: The team prioritizes the identified components based on their importance and relevance to the ship port website's core functionality. This prioritization helps in allocating resources and determining the order in which the components will be implemented in the prototype.

- Dependency Analysis: The team identifies any dependencies or relationships between the components. This analysis helps in understanding how different components interact with each other and ensures that the dependencies are properly addressed during the integration phase.

- Feasibility Assessment: The team assesses the feasibility of implementing each component within the given project constraints, including time, resources, and technical considerations. This assessment helps in identifying any potential challenges or limitations that may arise during the development of the prototype.

By performing a thorough component identification process, the project team can gain a clear understanding of the different features and functionalities that will be included in the system prototype. This ensures that the prototype adequately represents the core aspects of the ship port website and provides a solid foundation for further development and integration.

4.2.3 Technology Selection:

4.2.3.1 Database Platforms:

Platform	Advantages	Disadvantages
Oracle	*i.* Fast indexing data. *ii.* Supports clustered environment. *iii.* Various tools to handle the data. *iv.* Fast with large databases. *v.* PL/SQL provides an efficient way to develop data-intensive processes that are able to interact with data without transferring it to an app server. *vi.* Rich programmability model: the database supports not only the very rich SQL but also PLSQL, Java and it has very good command-line tools that make change management easy and efficient.	*i.* New (actually it is more than five years old) multi-tenant architecture is not as straightforward as SQL Server. *ii.* Many features require additional licensing that increase the total cost. *iii.* One of the most difficult vendors to work with.
MySQL	*i.* Database backup and restoration is easy. *ii.* Excellent community support.	*i.* Tends to be getting Slow with large databases. *ii.* Unhelpful error messages. *iii.* It is not always stable.
Microsoft SQL Server	*i.* Easy to configure and use with Visual Studio and ASP.Net. *ii.* Data Security. *iii.* Easy to understand and use. *iv.* It is easy to export database and tables in the form of SQL query or a script.	*i.* Database backup and recovery functionality need improvement. *ii.* It can be expensive to license. *iii.* No visualization of data.
MongoDB	*i.* It stores documents in a JSON-like format. *ii.* Very simple with easy to learn and understand syntax.	*i.* Limitations of document sizes and document nesting. *ii.* High memory usage.

Table 4- 1 Database comparison

4.2.3.2 Back-end Frameworks:

Framework	Advantages	Disadvantages
Node.js	*i.* Sharing the same piece of code with both server and client side. *ii.* Easy scalability. *iii.* Cross-platform development. *iv.* JSON support. *v.* Full-stack JavaScript. *vi.* Global community.	*i.* Node.js is not suited for CPU-intensive tasks. It is suited for I/O stuff only. *ii.* Nested call-back. *iii.* Harder to debug.
Ruby on Rails	*i.* Better code readability. *ii.* Ruby puts a strong emphasis on securing solutions made on it. *iii.* Ruby is an open source, which gives developers the feasibility to share their codes with other programmers.	*i.* Shortage of flexibility. *ii.* The boot time of the framework is quite long. *iii.* The price of a mistake in developing with Ruby on Rails is heavily connected to the performance time.
Laravel	*i.* Laravel has an extensive library of pre-programmed functionalities. *ii.* Highly scalable software allows you to tackle projects of any size. *iii.* The software has a safe, built-in access control system.	*i.* Laravel does have a comprehensive list of built-in features designed to make web development easier. *ii.* Laravel has regular updates. Older versions of the product quickly become buggy.
ASP.net	*i.* ASP.NET follows the MVC architecture, which allows for separate input, process and output of the application. *ii.* ASP.NET delivers enhanced performance and scalability. *iii.* The framework language allows for easy cross-platform migration.	*i.* Low Portability. *ii.* Complex pages with performance issues. *iii.* Lack of abstraction with least control over HTML.

Table 4- 2 Back-End Frameworks comparison

4.2.3.3 Front-End Frameworks:

Framework	Advantages	Disadvantages
React	*i.* Reusable components. *ii.* Easy search engine optimization. *iii.* Easy debugging. *iv.* Simple UI testing. *v.* Fast display of a great number of components.	*i.* Sometimes needs more code to be written. *ii.* Data changes are processed manually. *iii.* View-oriented.
Angular	*i.* Works well for SPAs. *ii.* Fast SDLC. *iii.* Requires less code. *iv.* Perfect testing options. *v.* MVC. *vi.* Interactivity.	*i.* Slow display of a great number of components. *ii.* Poor SEO options.
Flutter	*i.* One code base. *ii.* Fast and efficient. *iii.* No third-Party Integrations. *iv.* Reusable code.	*i.* Relatively new. *ii.* No web support yet. *iii.* Shortage of libraries.
Vue.js	*i.* A Fast Performance. *ii.* Cross-platform Development. *iii.* User Friendliness and Low Learning Curve. *iv.* Components Reusability	*i.* Too Much Flexibility. *ii.* Mostly Adopted in The Chinese Market. *iii.* Smaller Plugging and Tooling Ecosystem Compared to Other Frameworks.

Table 4- 3Front_End Frameworks comparison

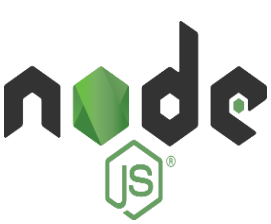
4.2.3.4. Recommended Technologies for Each Module in the POS

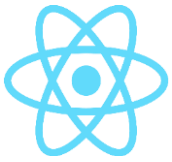
Our system is composed of the three following modules: Database, Front-end and Back-end for the web application.

1- MySQL: is commonly used in ship port databases due to its qualities as a relational database management system (RDBMS). It offers good performance and scalability, making it suitable for handling large amounts of data and high traffic loads. MySQL has a large user community, providing ample resources and support. Its ease of use, security features, and compatibility with various applications and programming languages make it a convenient choice. MySQL also supports replication for backup and high availability purposes. In summary, MySQL is a reliable and efficient option for managing ship port databases.

2- React.js is a preferred choice for the front-end development of web applications in the port authority system. It stands out due to its virtual DOM implementation and rendering optimizations, which result in improved performance and a better user experience. React.js allows for efficient updates to the user interface, making it suitable for dynamic and interactive web applications, which is often desired in port authority systems.

3- Node.js is widely used for the back-end of applications, like using Express.js to build the back-end of classic web applications. In addition, it is used for server-side programming and non-blocking, event-driven servers like typical websites and backend API services.





4.2.4 Design and Implementation:

4.2.4.1 Use case

- Ships Inform the Port Community System through their on-ground agent.
- Agent informs the Port Authority of expected arrival time.
- Ship enters a queue in the system.
- Meanwhile the Agent informs the PCS of the ship information.
- Agent also informs them of the IMO, where the Port Authority can then retrieve information from.
- The ship arrives at a platform and the shipment is then taken to warehouses

The importer then takes the shipment from the warehouses to their own warehouses

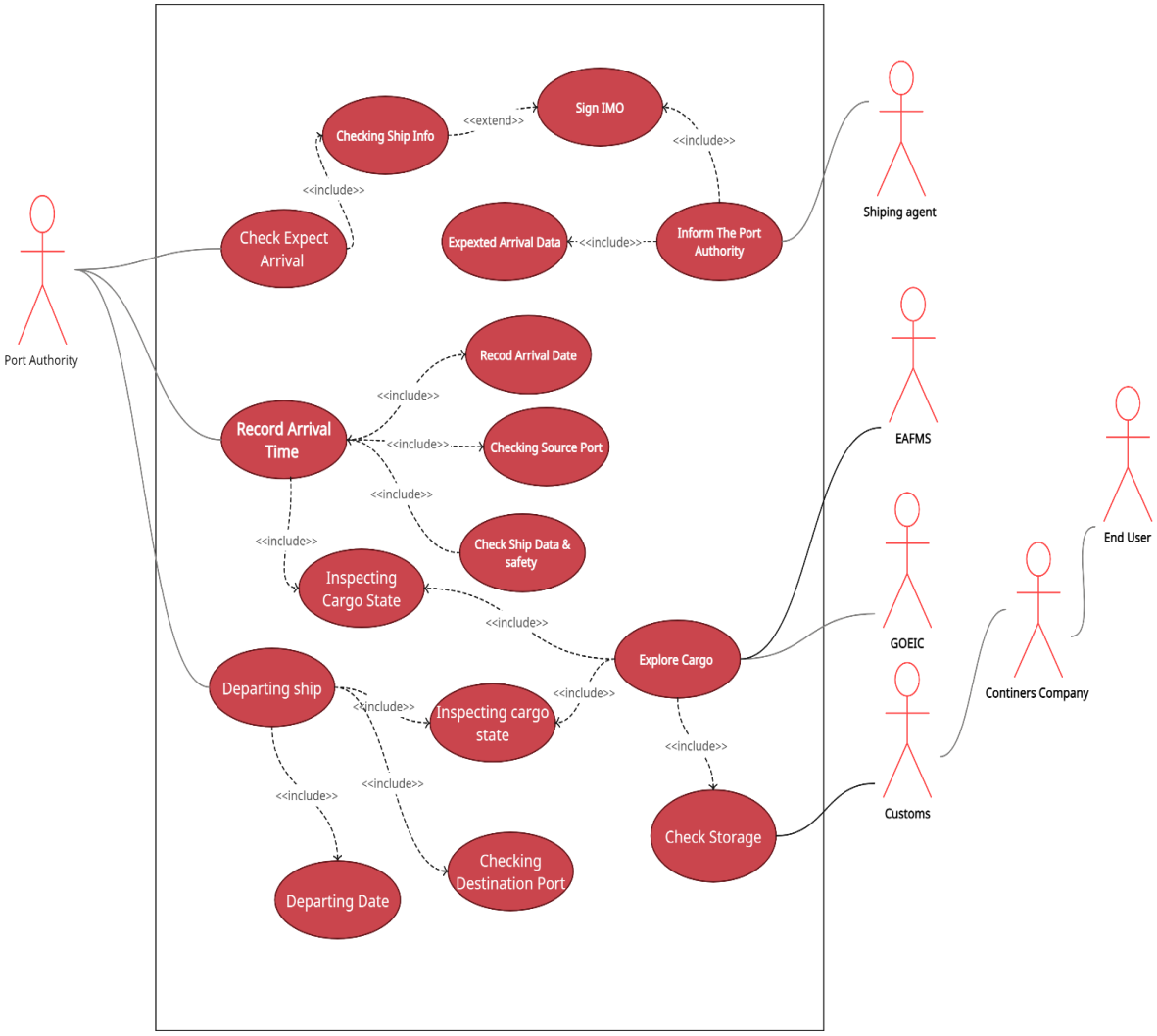


Figure 4- 1: Use case diagram.

4.2.4.2 ERD

In a port authority system for a ship port, there are several entities and relationships that need to be considered. Let's break down the given requirements and create an Entity-Relationship Diagram (ERD) based on the information provided.

Entities:

1. Country: Represents different countries that can have multiple ships with different descriptions.
2. Ship Type: Represents different types of ships that can have multiple ship descriptions.
3. Operation: Represents different operations that can be done by arriving ships.
4. Ship: Represents individual ships with specific descriptions.
5. Port: Represents the destination port for departing ships and the origin port for arriving ships.
6. Agent: Represents the responsible agent for departing and arriving ships.

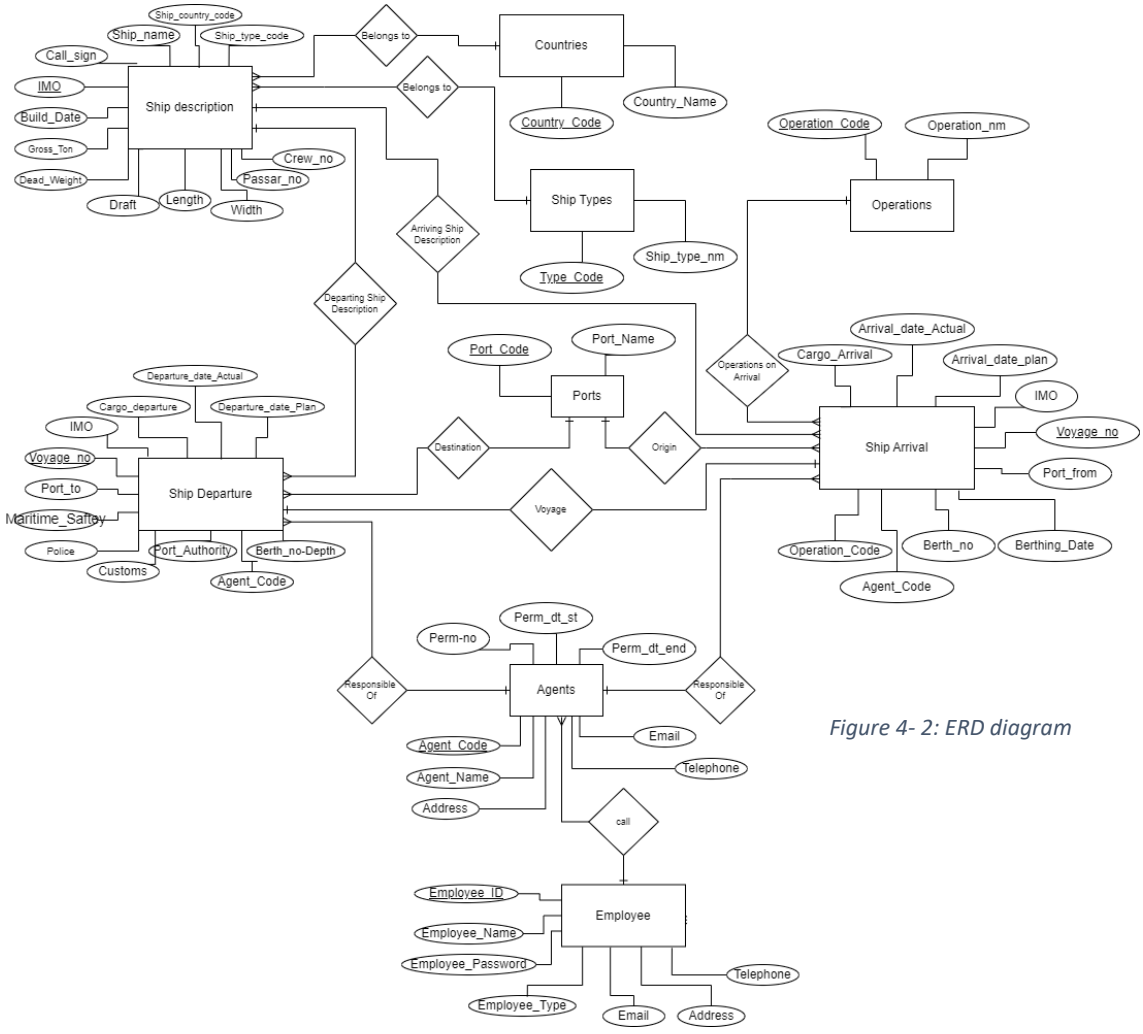


Figure 4- 2: ERD diagram

Relations.

- Each Country can have multiple ships with different description.
- Each Ship Type can have multiple ship descriptions.
- Each Operation can be done by multiple Arriving ships
- Each Ship with a specific description can arrive multiple times
- Each Ship with a specific description can depart multiple times
- Different ships can depart to the same destination port.
- Different ships can arrive from the same Origin port.
- Each Agent is responsible of Multiple Departing Ships.
- Each Agent is responsible of Multiple Arriving Ships.
- Each Arriving ship must depart later.

4.2.4.3 Database

To build a database system with a user-friendly graphical interface: Define the requirements: Clearly identify the purpose and scope of database system. Determine what data need to store, retrieve, and manage. Consider the relationships between different data entities.

Design the database schema: Create a logical model of database structure using entity-relationship diagrams or similar techniques. Define the tables, their columns, and the relationships between them.

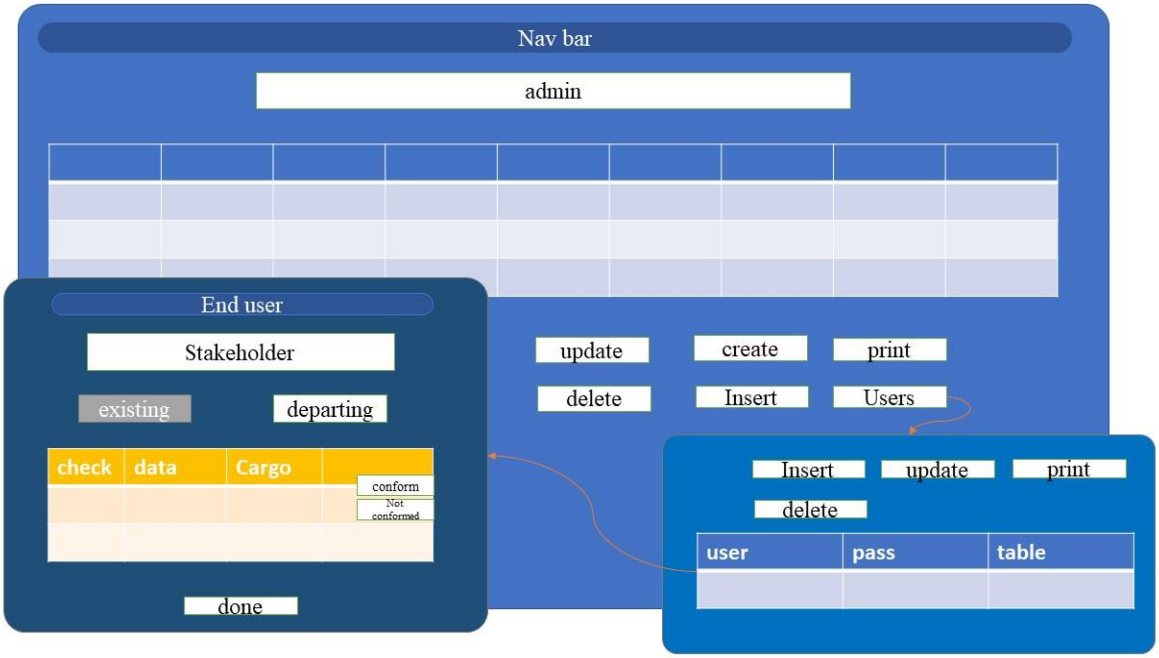


Figure 4- 3: Database Design and Implementation

4.2.4.4 Frontend

By starting with a simple home page and an intuitive admin page, we can lay the groundwork for the port authority system's front-end prototype. As the development progresses, additional pages and features can be added based on the system's requirements and user feedback, further refining the user interface, and enhancing the overall user experience.

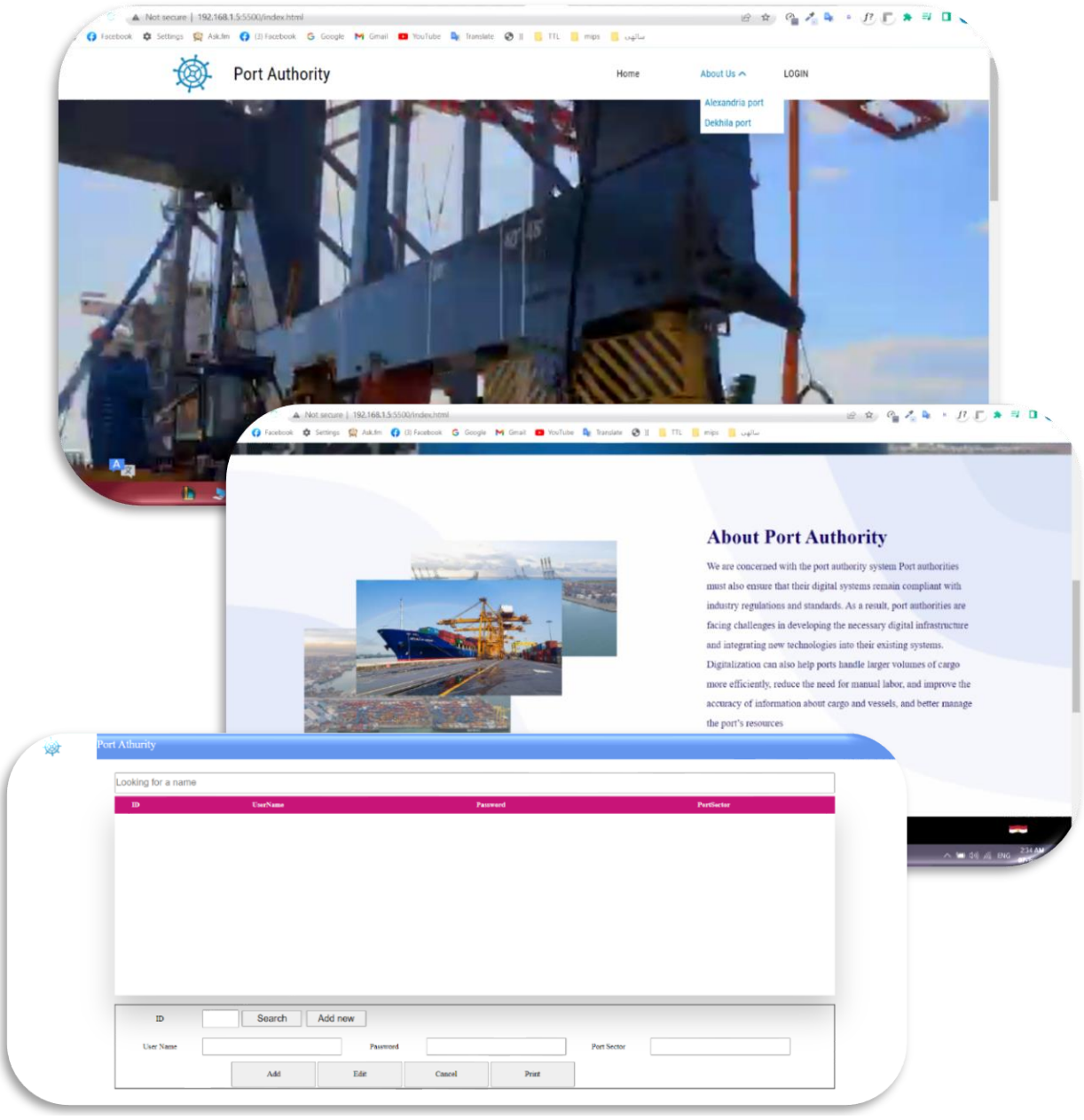


Figure 4- 4: Frontend Design and Implementation

4.3 SYSTEM ARCHITECTURE DIAGRAM

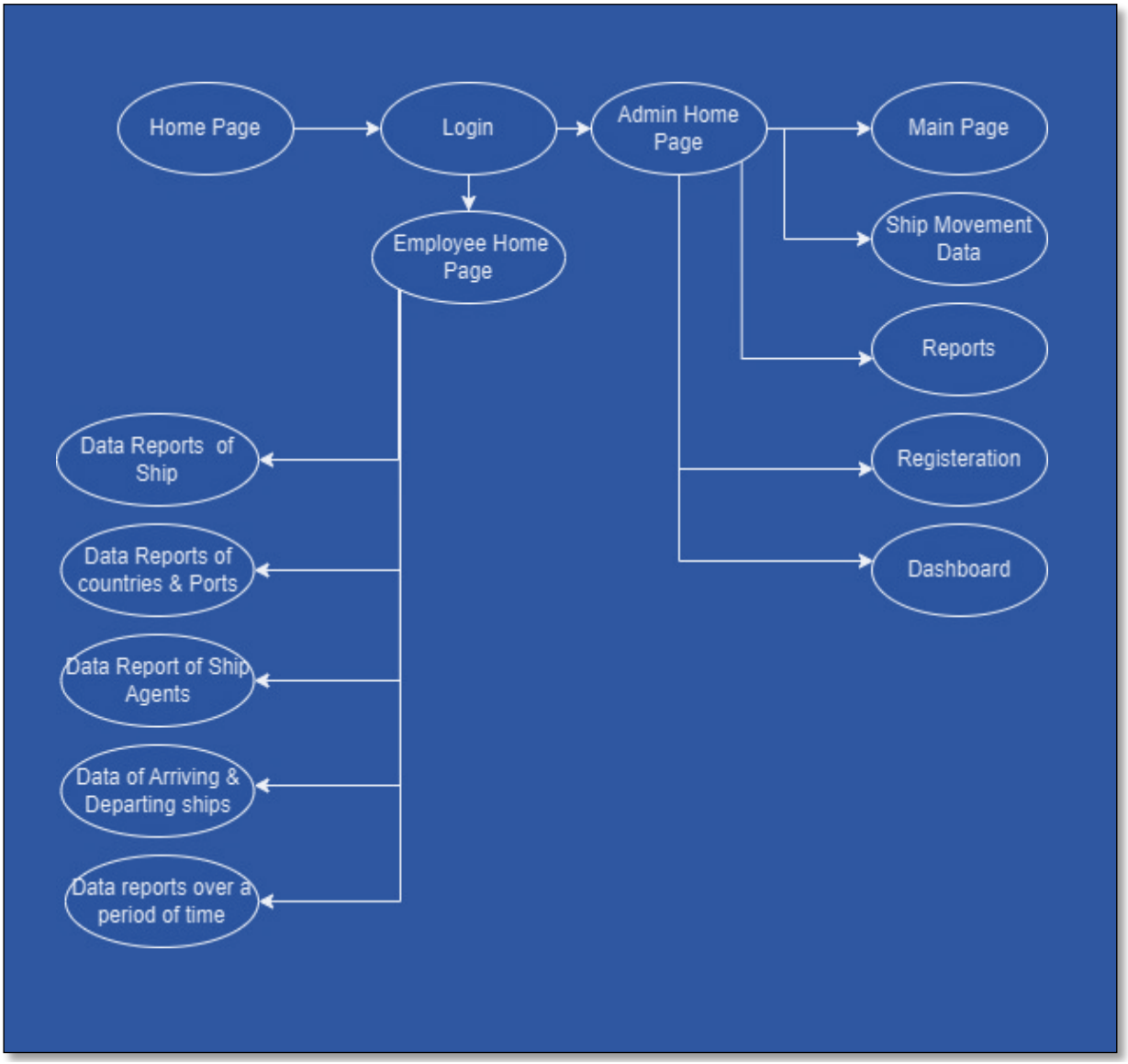


Figure 4- 5: SYSTEM ARCHITECTURE DIAGRAM

4.4 DESCRIPTION OF EACH PHASE

4.4.1 Phase I : Data Collection and Manipulation

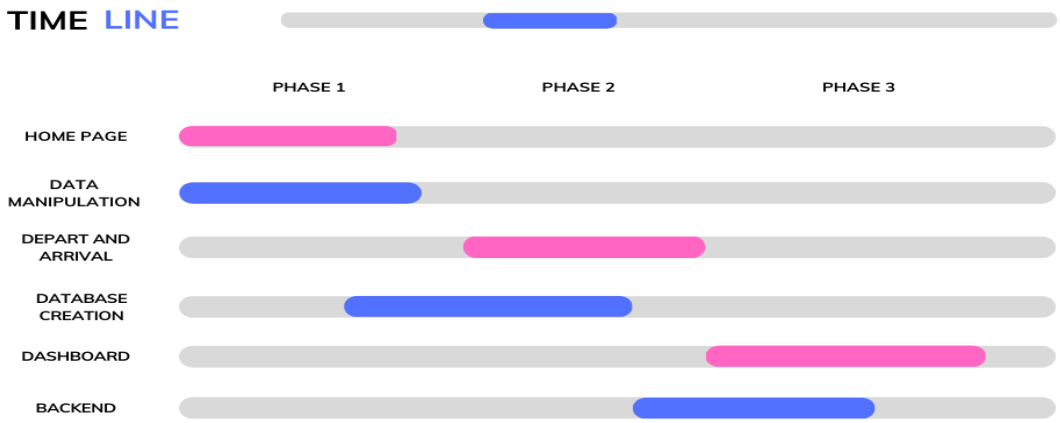


Figure 4- 7: Project Timeline

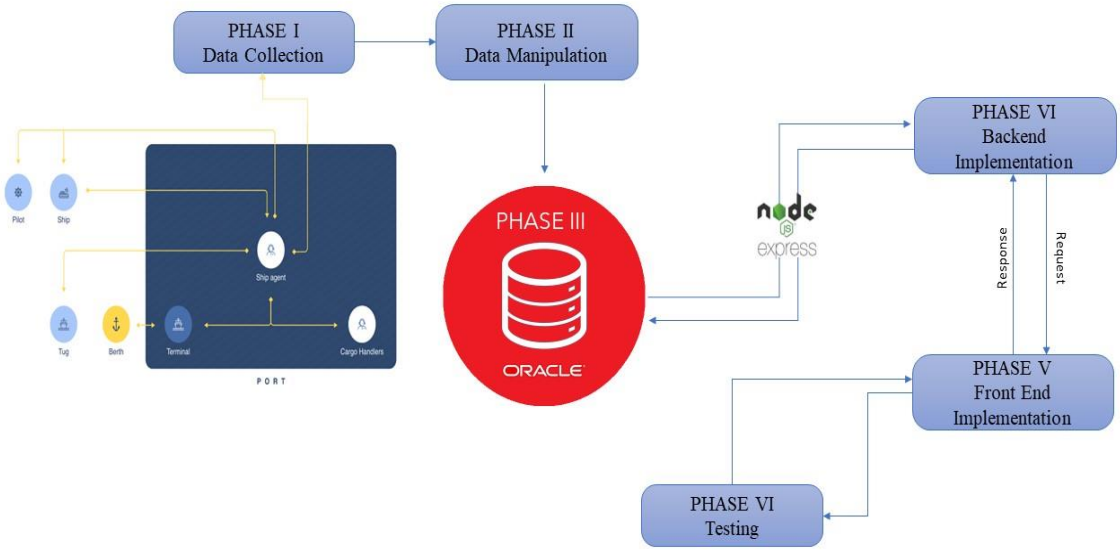


Figure 4- 6:system connection

4.4.1.1. Data Collection

description of the data collection process for the port community system:

Step 1: Manual Data Collection

In the initial phase of data collection, a comprehensive approach was adopted to gather information about the port community system. This involved physically visiting the port and engaging in discussions with key stakeholders, including port employees and IT managers. Through these interactions, valuable insights were gained into the port's operational procedures, data handling practices, and overall functioning of the port community system. This manual data collection process helped in understanding the complexities and nuances of the port's operations, which later played a crucial role in refining the data collection strategy.

Step 2: Automated Data Download

As our work progressed, we noticed a recurring issue. The daily task of downloading data was tedious and prone to human error. There were instances where we forgot to download the data for a particular day, leading to gaps in our information.

In light of these challenges, one of our team members came up with a solution. They created an automated bot that would take care of this routine task for us. This bot was programmed to visit the Port of Alexandria's website automatically at midnight every day. Once there, it would scrape the required data from the website.

After the scraping process, the bot would then download the data it had collected. The downloaded data was parsed and transferred into an Excel file. To keep things organized, the bot was programmed to name each Excel file by the date it was downloaded.

But the bot's capabilities didn't end there. It was further set up to distribute the freshly downloaded data using FTP and other data transfer techniques. This made sure our collaborators received the data promptly and efficiently.

To ensure even wider access to the data, the bot was also set up to interact with Discord, a popular online chat platform our team used. Utilizing Discord's API, the bot would automatically push the day's Excel file to a dedicated channel on the platform. This served as an easy and convenient gateway for all team members to access the data files, simplifying our workflow and greatly reducing the potential for human error in our daily data collection.

4.4.1.2. Data Types

Ship Description:

- Each ship has a unique IMO number.
- All ships have a Call_sign, Name, Country Code, Type Code, Build Date, Length and Width
- All ships have changing attributes like Gross tonnage, Dead Weight, Draft , Passenger Count, and Crew Count

Ship Departure:

- Each Departing Ship has a unique Voyage Number
- Each Departing Ship has to get Approved from Port Authority , Customs , Police and Health and Maritime Safety Authority.
- Each Departing ship must have an IMO, Destination Port, Agent Code, Berthing Depth, Departing Cargo , Planned Departure date, Actual Departure date.

Ship Arrival:

- Each Arriving Ship has a unique Voyage Number
- Each Arriving Ship has to have a Berthing Number and Berthing Date
- Each Arriving ship must have an IMO, Departed from Port, Agent Code, Operation Code, Arriving Cargo, Planned Arriving date, Actual Arriving date.

Ports:

- Each Port must have a unique Port Code
- Each Port must have a name.

Ship Type:

- Each Ship must have a unique code
- Each ship must have a Ship type name

Countries:

- Each Country must have a unique Country Code
- Each country must have a name

Operations:

- Each Operation must have a unique Operation Code
- Each Operation must have a name

Agents:

- Each Agent must have a unique Agent Code
- Each Agent must have some attributes like Name, Address, Telephone, Email.
- Each Agent must have a Permit number, Permit start and end date.

4.4.1.3. Data Preprocessing

In this data manipulation scenario,

- Using a python script hosted on a server, the bot downloaded 110 files each of arriving and departing data.
- Each of those files contained 5 rows each, totaling 550 rows of data.
- Using a python script, all the 110 files were then merged while simultaneously removing all redundancies.
- Due to irregular data, the data was then filtered to remove all rows that didn't have their counterparts in either the departing or the arriving data.
- After all the filtration the data became 67 rows of data, which were then inserted using the previously mentioned scripts.
- We also created a few other python scripts to do each of the following:
    - To download all the country's names and their codes
    - To download all the port's names and their codes
- The country codes were 248 countries with their codes.
- The ports came out to be 44892 ports with their respective codes.

.

Integration into Website Testing

Finally, the processed and validated dataset was integrated into the testing framework for the website. By utilizing this real-world data from the Alexandria port, the website's functionality and performance could be evaluated under various scenarios, providing valuable insights into its responsiveness, robustness, and user experience.

Overall, the combination of manual data collection, automated data download, Python scripting for data preprocessing, and website testing using actual port community system data contributed to a comprehensive and rigorous evaluation of the website's capabilities and readiness for deployment.

4.4.2 Phase II: Database Creation and Insertion.

In this phase, we will begin by selecting the most suitable technology to build our database. We will then proceed with installing and configuring the chosen database technology on a server. This will enable all team members to interact with the database effectively, using tools such as SQL Workbench.

The next step will involve creating the necessary tables within the database. Tables serve as the structure for organizing and storing data, providing a foundation for data integrity and efficient querying.

Finally, we will insert the previously collected data into the newly created tables. This process will involve mapping the collected data to the appropriate table columns to ensure accurate representation and data integrity.

By completing this phase, we will have established a robust database infrastructure with the required tables and populated data. This foundation will support our project's objectives and set the stage for future phases of development and optimization.

4.4.2.1. Install MySQL workbench.

4.4.2.1.1. Installing Using MySQL Installer

The general MySQL Installer download is available at MYSQL Website. The MySQL Installer application can install, upgrade, and manage most MySQL products, including MySQL Workbench. Managing all your MySQL products, including Workbench, with MySQL Installer is the recommended approach. It handles all requirements and prerequisites, configurations, and upgrades.

When executing MySQL Installer, you may choose MySQL Workbench as one of the products to install. It is selected by default, and essentially executes the standalone MSI Installer package described in the next section.

4.4.2.1.2. Launching

To start MySQL Workbench on Windows, select MySQL from the Start menu and then select MySQL Workbench. This sequence executes the MySQLWorkbench.exe file on your system. Alternatively, start MySQL Workbench from the command line, for example:

```
C:\Program Files\MySQL\MySQL Workbench 8.0\mysqlworkbench.exe
```

4.4.2.2. Establish database on server.

4.4.2.2.1. Running MySQL Server Locally

At the start of our project, we tried to set up a database server on our own computer using Oracle MySQL. This meant we had to make changes to our router, open a certain port, and set up port forwarding.

We ran into a bunch of issues. The IP address kept changing, which caused a lot of problems with our setup. Plus, the computer we used to host the server had to be switched on all the time. We needed it to be always available for our team members. This meant the computer was being used non-stop, which we soon realized wasn't a good idea.

With the computer running all the time and the IP address constantly changing, we were using a lot of our resources just to keep things going. Keeping everything running smoothly took a lot of work and it soon became clear that we needed to rethink our strategy. We needed to find a better, more reliable way to host our server.

4.4.2.2.2. Running MySQL Server Online

Facing problems with our initial setup of a local MySQL database server, we decided to transition to the cloud. Ahmed Ashraf, a team member with experience in DigitalOcean, helped facilitate the move. Once we moved our database server to the cloud-based platform, our workflow improved significantly. The server was accessible around the clock from anywhere, and we faced almost no downtime. The issues with changing IP address and overuse of our local computer were completely resolved. Moving to the clouds with DigitalOcean ended up being an efficient and reliable solution.

4.4.2.3. Table Creation

In this section, Converting an Entity-Relationship Diagram (ERD) into SQL code is a crucial step in the database design process. The ERD provides a visual representation of the entities, relationships, and attributes in a database system, while SQL (Structured Query Language) is a programming language used to manage and manipulate relational databases.

When converting an ERD to SQL code, the first step is to identify the entities in the ERD and create corresponding tables in the SQL database. Each entity becomes a table, and each attribute becomes a column in that table. The table's primary key is typically derived from the primary key attribute in the ERD.

For example, let's say we have an ERD with two entities: " countries " and " operations."

```
CREATE TABLE `operations` (
  `Operation_Code` varchar(255) NOT NULL,
  `Operation_nm` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`Operation_Code`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;


CREATE TABLE `ports` (
  `Port_Code` varchar(10) NOT NULL,
  `Port_Name` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`Port_Code`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

Figure 4- 8: Snapshot of SQL Statement to Create Table.

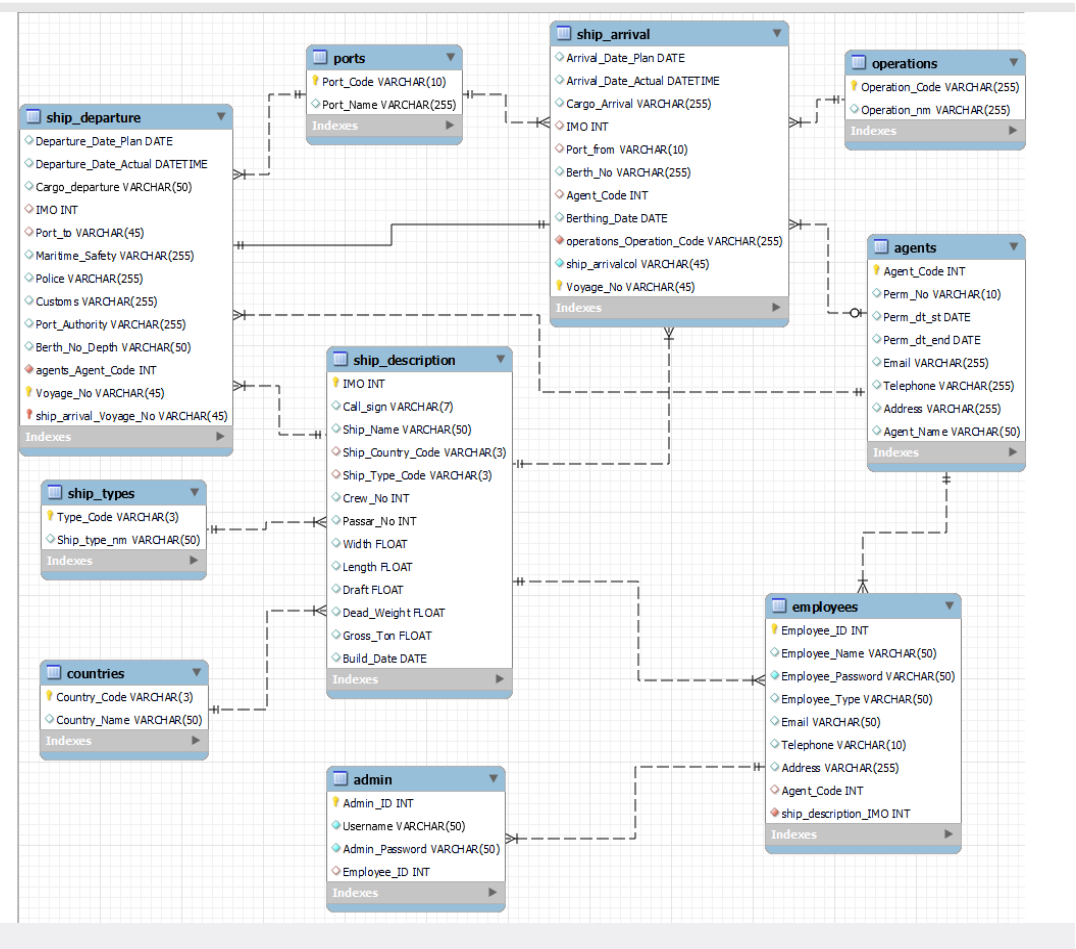


Figure 4- 9 database Model (MySQL Workbench)

4.4.2.4. Data Insertion

- Using a python script, we inserted each of the following automatically into the database:
    - Ports
    - Agents
    - Ship Description
    - Ship Arrival
    - Ship Departure
    - Countries

```
42
43  db = mysql.connector.connect(
44      host="localhost",
45      user="admin",
46      passwd="admin",
47      database="orcl"
48  )
49
50
51
52
53  mycr = db.cursor()
54  for i in range(size):
55      name = str(Name_values[i]).replace("'", "''")
56      mycr.execute(f"INSERT INTO countries(`Country_Code`,`Country_Name`) VALUES ('{Code_values[i]}','{name}');")
57
58
59  db.commit()
60
61
```

Figure 4- 10: Python Script Designed for Automatic Data Insertion

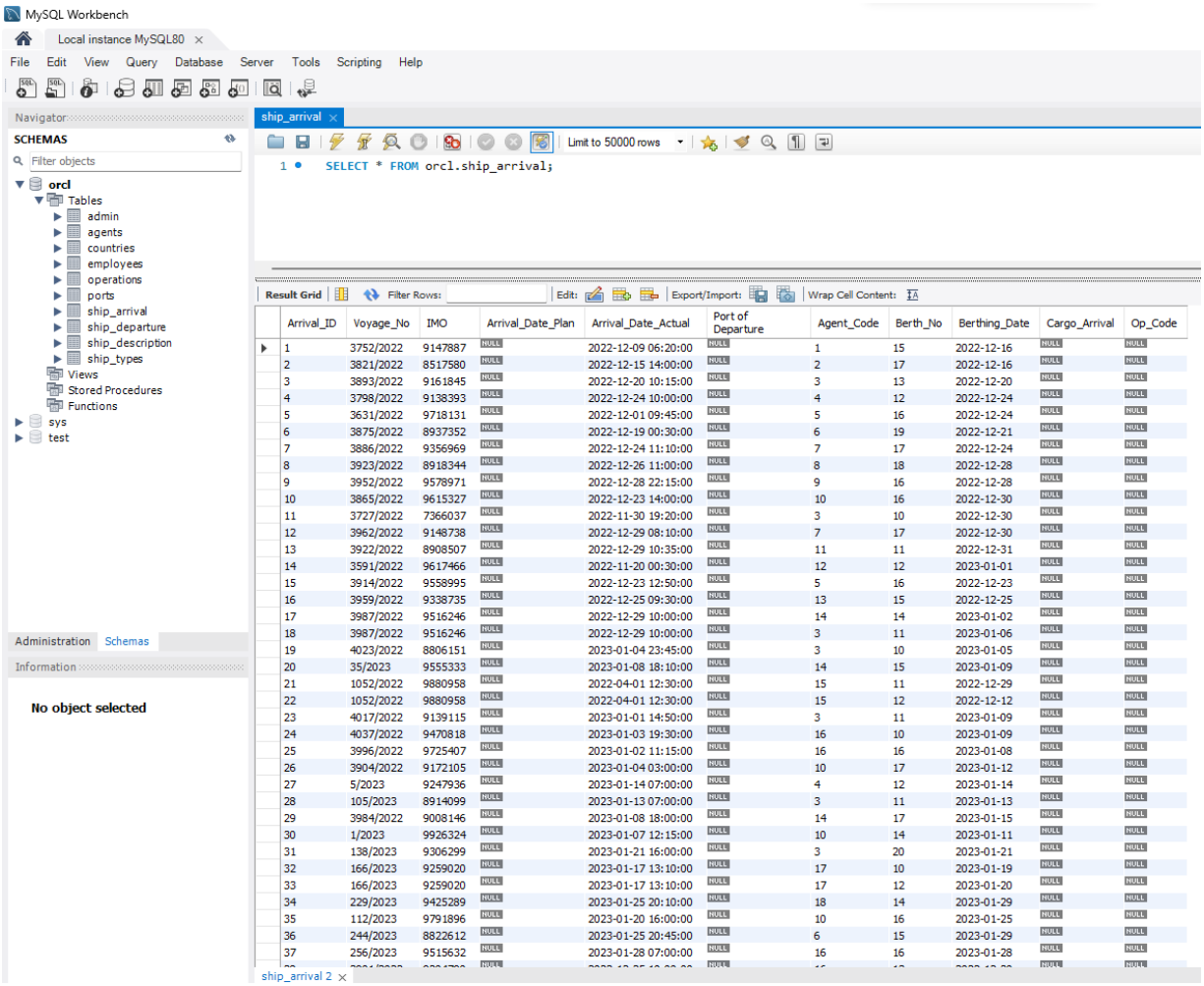


Figure 4- 11: Ship Arrival Data

4.4.3 Phase III: Backend

In the next phase, we can use Node.js to create APIs that will serve as the interface between the user interface and the database. These APIs will allow users to retrieve information from the database and make edits to it. Node.js is a popular runtime environment for executing JavaScript on the server side, making it a suitable choice for building server-side applications.

To create the APIs, we can use a web framework like Express.js, which provides a simple and intuitive way to handle HTTP requests and define routes.

.

4.4.3.1. File Structure

In this section, Created Organized Filesystem for the backend using the MVC model.

MVC Model: The backend of the ship port system follows the Model-View-Controller (MVC) architectural pattern. This pattern helps in organizing the codebase and separating concerns into different components. Here's how the MVC model is utilized:

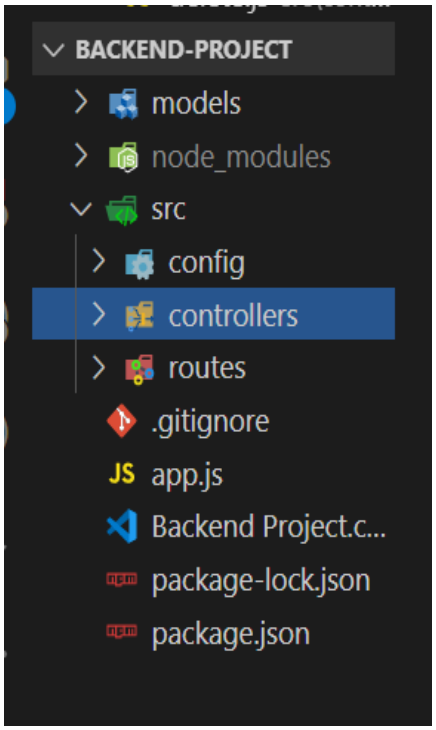


Figure 4- 12: illustrates the file structure in Visual Studio Code for backend development.

1. app.js: This is the main entry point of your application. It sets up the server, middleware, and initializes the necessary components.

2. config: This folder contains configuration files for your application, including the database connection configuration. You can have a file named database.js or db.js to establish the database connection and export it for use in other parts of your application.

3. controllers: This folder holds the controller files that handle the logic for different API routes or endpoints. Each controller file is responsible for a specific set of related functionalities. For example, you can have a customersController.js file that handles CRUD operations related to customers.

4- routes: This folder contains the route files that define the API endpoints and map them to the appropriate controller methods. Each route file corresponds to a specific resource or entity. For example, you can have a customersRoutes.js file that defines routes for managing customers.

With this file structure, you can separate concerns, maintain code organization, and easily add new routes and controllers as your application grows. It provides a modular approach to handle different functionalities and promotes code reusability.

4.4.3.2. Database Connection

In this section, we will configure the database connection by creating a file named database.js inside the config folder. This will help us keep our files organized and modular. We will define a method called connect, which will handle the database connection, and then export it to be used in other parts of our application.

Step 1: Create the database.js file inside the config folder:

```
const mysql = require('mysql2');

const connection = mysql.createConnection({
    host:'ap-project-do-user-13215137-0.b.db.ondigitalocean.com',
    port:25060,
    user:'abdo',
    password:'AVNS_jTBzev2csl3OXGvGZoN',
    database:'orcl-backup'
});
const connectDB =() =>{
    try {
        connection.connect();
        // console.log('Connected to Mysql DB');
        return 1;
    } catch (error) {
        // console.log(error);
        return 0;
    }
}

module.exports = {connection , connectDB };
```

Figure 4- 13: Config. Database Connection

```
const express = require('express');
const Routes = require('./src/routes/routes');
const cors = require('cors');
const dbconnect = require("./src/config/database");
const con = dbconnect.connection;


const app = express();
const PORT = 3000;

app.use(cors());

app.use(express.urlencoded({ extended: true }));

app.use(express.json());

//Routes Activation
app.use('/api/', Routes);

app.get("/", (req, res) => {
    res.send("Hello From Backend");
      });

//Set Server Port
app.listen(PORT ,() => {
    console.log(`Server running on port: http://localhost:${PORT}`);
  });

function keepAlive() {
    con.query('SELECT 1', (error, results) => {
      if (error) {
        console.error('Error executing keep-alive query:', error);
      } else {
        console.log('Keep-alive query executed successfully');
      }
    });
  }

  // Set the interval for executing the keep-alive query (every 5 minutes in this example)
  const interval = 7 * 60 * 60 * 1000;
  setInterval(keepAlive, interval);
```

Figure 4- 14 Installing Database Connection

Step 2: Usage in other files to connect to the database:

Now, in the files where you need to access the database, you can import the connect method from database.js and use it to establish a connection with the database.

Example usage in app.js:

Example usage in a controller file (employeeController.js):

```
const dbconnect = require("../config/database");
const con = dbconnect.connection;

const addEmp = async (req, res) => {
  try {
    const { Employee_Name, Employee_Password, Email, Telephone, Role } =
      req.body;
    const sql = `INSERT INTO employees (Employee_Name, Employee_Password, Email, Telephone, Role)  VALUES (?, ?, ?,
?, ?)`;
    con.query(
      sql,
      [Employee_Name, Employee_Password, Email, Telephone, Role],
      function (err, result) {
        if (err) {
          res.status(500).send({ message: err.sqlMessage , query: false });

        } else {
          res
            .status(200)
            .send({ message: "Add_Employee Query Executed Correctly", query: true });
        }
      }
    );
  } catch (error) {
    console.log(error);
  }
};
```

Figure 4- 15: Database in controller files

With this setup, you have a modular and organized database connection in database.js, which can be easily imported and used in other parts of your application, such as controllers or routes, allowing you to access the database and perform CRUD operations efficiently.

4.4.3.3. POSTMAN

In the development process, Postman is a widely used tool to test APIs before handing them over to the frontend developer. It allows the backend developer to test and validate the functionality of the APIs, ensuring they work correctly and return the expected responses. Postman provides a user-friendly interface for making API requests and analyzing the responses.

Step 1: Installation and Setup

1. Install Postman: Download and install the Postman application from the official website.
2. Create a Postman Project: Create a new project in Postman and invite the team members to join the project. This helps in collaborating and sharing the API collections with the front-end developer.

Step 2: Organize the APIs

1. Create Collections: Create collections in Postman to organize your APIs. Each collection can represent a specific module or feature of your application.
2. Add Requests: Within each collection, add individual requests for different HTTP methods (GET, POST, PUT, DELETE) based on the available API endpoints. You can organize the requests within folders based on the purpose or functionality.

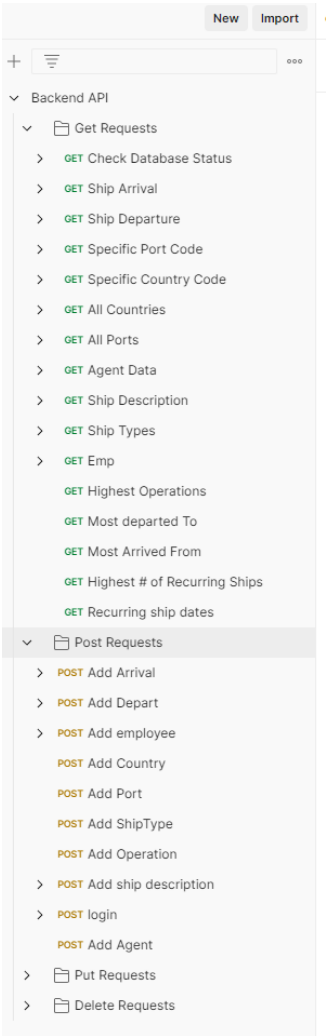


Figure 4- 16illustrates the file structure - POSTMAN.

Step 3: Test APIs

1. Set Request Parameters: Configure the necessary parameters such as headers, URL, query parameters, request body, etc., for each API request.

2. Send Requests: Execute the API requests by clicking the "Send" button. Postman will send the request to the specified URL and display the response.
3. Analyze the Response: Examine the response in the Postman interface. You can view the response headers, status code, response body, etc. Verify if the response matches your expectations.
4. Handle Authentication: If your APIs require authentication, Postman provides various options to set authentication headers, cookies, or tokens.
5. Test Different Scenarios: Test different scenarios by modifying the request parameters or payload. This helps in checking the API's behavior under various conditions.

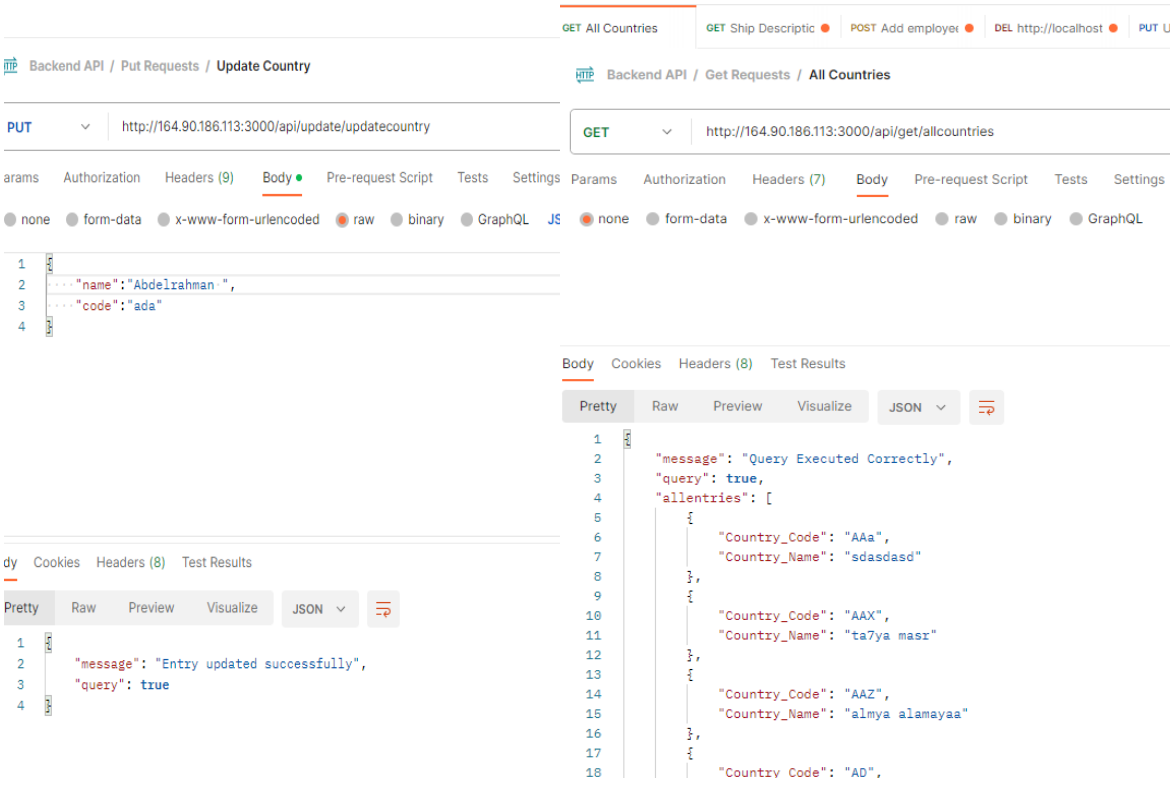
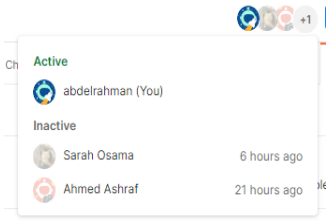


Figure 4- 17: GET request - POSTMAN.

Figure 4- 18: PUT request - POSTMAN.

Step 4: Share and Collaborate



1. Share Collections: Share the collections with the frontend developer so they can import them into their own Postman workspace. This ensures that the front-end developer can use the same API configurations and test them independently.
2. Document APIs: Postman provides options to generate API documentation or export the collections as Swagger or OpenAPI specifications. This documentation can be shared with the front-end developer to understand the available endpoints and their expected behavior.

By using Postman, the backend developer can thoroughly test the APIs, validate their functionality, and ensure they return the correct responses. This helps in improving the quality of the APIs before being consumed by the frontend developer. The organized folder structure and sharing capabilities of Postman allow for seamless collaboration between the backend and frontend teams.

4.4.3.4. GITHUB repository

GitHub is a widely used platform for collaborative software development. When working on a project with multiple team members, creating a repository `Backend-Project` on GitHub offers several benefits.

1. Collaboration: By creating a repository, team members can work on the same codebase concurrently. Each team member can clone the repository to their local machine, make changes, and push those changes back to the repository. This enables effective collaboration, allowing team members to contribute and see each other's updates.
2. Code History: GitHub utilizes Git, a version control system that tracks changes made to the codebase over time. Every modification is recorded, including who made the change and when. This comprehensive code history helps teams understand the evolution of the project and facilitates the ability to revert to previous versions if necessary.
3. Code Updates: GitHub provides features such as branches, pull requests, and code reviews to manage code updates efficiently. Team members can create branches to work on specific features or bug fixes independently. They can then submit pull requests to merge their changes back into the main codebase. Pull requests also facilitate code reviews, enabling other team members to review and provide feedback on the proposed changes before merging.
4. Centralized Code Repository: GitHub serves as a centralized location for the codebase, ensuring that all team members have access to the latest version of the project. It simplifies code sharing, collaboration, and ensures that everyone is working with the most up-to-date code.

In summary, using GitHub and creating a repository streamlines collaboration, preserves a detailed history of code changes, and establishes an organized workflow for managing code updates. It enhances teamwork, facilitates code review processes, and helps maintain a centralized and up-to-date codebase for the project.

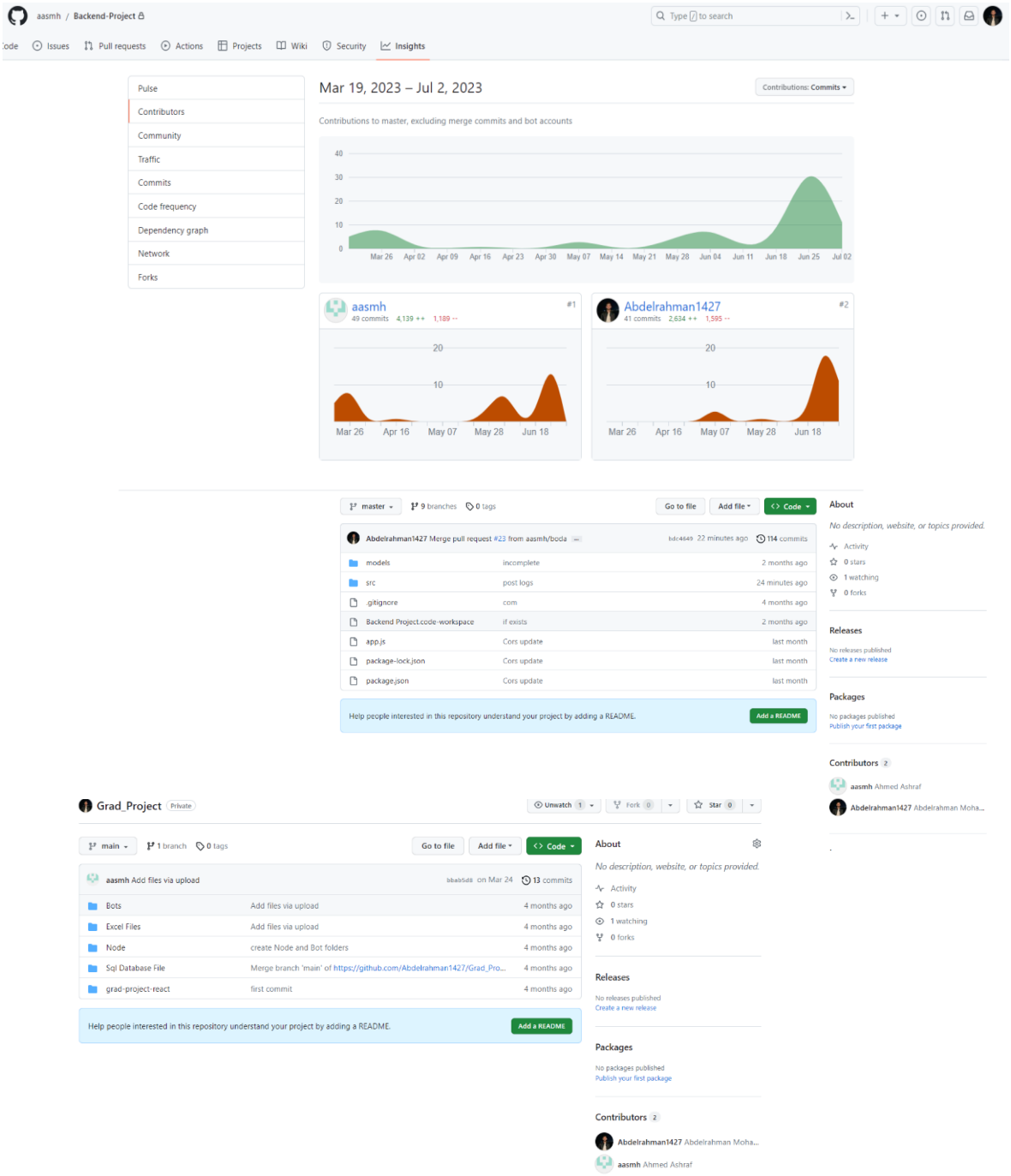


Figure 4- 19: GitHub Snapshots

4.4.3.5. Deploy Server

Running the Server locally

Once again, we faced a similar situation related to the MySQL database when our front-end team was ready to begin development. For them to test their application and proceed with the development process, a functional backend was crucial. However, the familiar problem of using localhost came up again, echoing the issues we had experienced previously.

Like before, the IP address's instability caused a series of disruptions, and the constant need for the backend-hosting computer to be always on led to overuse and posed sustainability concerns. This was becoming a major hurdle in allowing the frontend team to start their work and test the application effectively.

Running the Server Online

Given our previous experience, the solution was clear. We needed to set up an online hosted server that was available 24/7 and had a stable connection to the database. This way, the front-end team could freely access the backend services they needed, no matter the time or their location.

By adopting this strategy, we ensured a reliable, constant, and accessible backend environment. This enabled the front-end team to efficiently carry out their development tasks and run their tests, further streamlining our project workflow.

4.4.3.6. Get APIs

To create GET APIs, you can follow these steps:

1. Create a get.js file in the controller directory. This file will contain the methods for handling GET requests.
2. In the routes file, import the methods from the get.js file and save them in a constant, such as ctrl.
3. In the routes file, use the router. Get() method from Express to define the GET API route. Specify the endpoint and the corresponding method from the controller.
4. In the getMethod method of the controller, you can write the code to handle the GET request. This can include retrieving data from a database, processing the request, and sending a response back to the client.

1. GET /api/get/portcode
   Description: This route is used to fetch port codes.
2. GET /api /get/countrycode
   Description: This route is used to fetch country codes.
3. GET /api /get/allcountries
   Description: This route is used to fetch data for all countries.
4. GET /api /get/allports
   Description: This route is used to fetch data for all ports.
5. GET /api /get/agentdata
   Description: This route is used to fetch agent data.
6. GET /api /get/shipdescdata
   Description: This route is used to fetch ship description data.
7. GET /api /get/shiptypesdata
   Description: This route is used to fetch ship types data.
8. GET /api /dbstatus
   Description: This route is used to check the status of the database.
9. GET /api /get/operation
   Description: This route is used to fetch operation data.
10. GET/api /get/highvistedports

Description: This route is used to fetch data for the highest visited ports.

11. GET /api /get/higharrivedports

    Description: This route is used to fetch data for the highest arrived ports.

12. GET /api /get/highops

    Description: This route is used to fetch data for the highest operations.

13. GET/api /get/highrecships

    Description: This route is used to fetch data for the highest recommended ships.

14. GET /api /get/highrecshipdata

    Description: This route is used to fetch data for the highest recommended ship data.

15. GET /api /get/logs

    Description: This route is used to fetch logs data

16. GET /api /get/depart.

    Description: This route is used to fetch data for departures.

17. GET /api /get/arrival.

    Description: This route is used to fetch data for arrivals.

By following these steps, you can create a GET API by separating the route definition in the routes file and the request handling logic in the controller file. This modular approach helps maintain code readability and makes it easier to manage and scale your application's APIs.

4.4.3.7. Post APIs

1. POST /api/login/addArrival

   Description : This route is used to add an Arriving Ship

2. POST /api/login/addDepart

   Description : This route is used to add an Departing Ship

3. POST /api/login/addEmp

   Description : This route is used to add an Employee

4. POST /api/post/newcountry

   Description : This route is used to add a new country

5. POST /api/post/Newport

   Description : This route is used to add a new port

6. POST /api/post/newshiptype

   Description : This route is used to add a new Ship Type

7. POST /api/post/newoperation

   Description : This route is used to add a new Operation

8. POST /api/post/newshipdesc

   Description : This route is used to add a new Ship Type Description

9. POST /api/post/login

   Description : This route is used to login using credentials

10. POST /api/post/newagent

    Description : This route is used to add a new Agent

11. POST /api/post/logs

    Description : This route is used to add a log

4.4.3.8. DELETE APIs

1. DELETE /api/delete/deletearrival

   Description : This route is used to Delete an Arriving Ship

2. DELETE /api/delete/deletedepart

   Description : This route is used to Delete a Departing Ship

3. DELETE /api/delete/deleteEmp/

   Description : This route is used to Delete an Employee

4. DELETE /api/delete/deletecountry

   Description : This route is used to Delete a Country

5. DELETE /api/delete/deleteport

   Description : This route is used to Delete a Port

6. DELETE /api/delete/deleteshiptype

   Description : This route is used to Delete a Ship Type

7. DELETE /api/delete/deleteop

   Description : This route is used to Delete an Operation

8. DELETE /api/delete/deleteShipDesc

   Description : This route is used to Delete a Ship Type’s Description

9. DELETE /api/delete/Agent

   Description : This route is used to Delete an Agent

4.4.3.9. PUT APIs

1. PUT /api/update/updatearrival

   Description : This route is used to update an Arriving Ship

2. PUT /api/update/updatedepart

   Description : This route is used to update a Departing Ship

3. PUT /api/update/updatecountry

   Description : This route is used to update a new country

4. PUT /api/update/updateport

   Description : This route is used to update a port

5. PUT /api/update/updateshiptype

   Description : This route is used to update a Ship Type

6. PUT /api/update/updateop

   Description : This route is used to update an Operation

7. PUT /api/update/updateShipDesc

   Description : This route is used to update a Ship Type’s Description

8. PUT /api/update/updateAgent

   Description : This route is used to update an Agent

4.4.4 Phase IV: frontend

The front end is responsible for handling the user interface and interactions. It encompasses the steps involved in displaying and collecting information from users. Here's a description of how the typical steps on the front-end work.

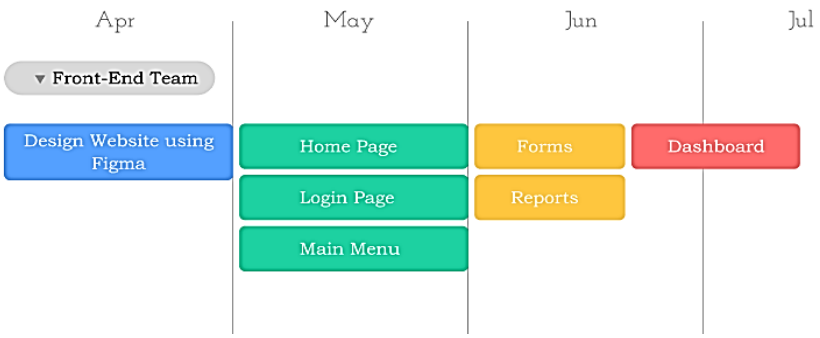


Figure 4- 20: frontend TimeLine

4.4.4.1 GUI Design

Designing using Figma:

Figma is a cloud-based design and prototyping tool used in the field of user interface (UI) and user experience (UX) design. It provides a collaborative platform for designers, developers, and stakeholders to work together on creating and iterating designs for various digital interfaces such as websites, mobile apps, and more.

Logo:

When designing a logo for a port, careful consideration of color and shape is crucial. The color palette should reflect the essence of the port and the maritime industry, often using shades of blue to evoke calmness, trust, and reliability. Warm earth tones like sandy orange can also be incorporated to signify the connection between land and water. The shape of the logo often draws inspiration from nautical elements such as ships, waves, or compasses, conveying the port's purpose and evoking feelings of stability, progress, and direction. Striking the right balance between colors and shapes in the logo design ensures it effectively captures the port's identity and resonates with the target audience.

Main Page:

The website design includes a user-friendly navbar with four key elements: "Login," "About Us," "Contact Us," and "Services." The "Login" option enables secure access to user accounts. The "About Us" link provides information about the website's team. The main page highlights the "Services" and "Achievements of the Ports" sections. The "Services" section details the port's offerings, while the "Achievements of the Ports" section showcases notable accomplishments. This well-structured navbar and content organization ensure a seamless browsing experience for users.

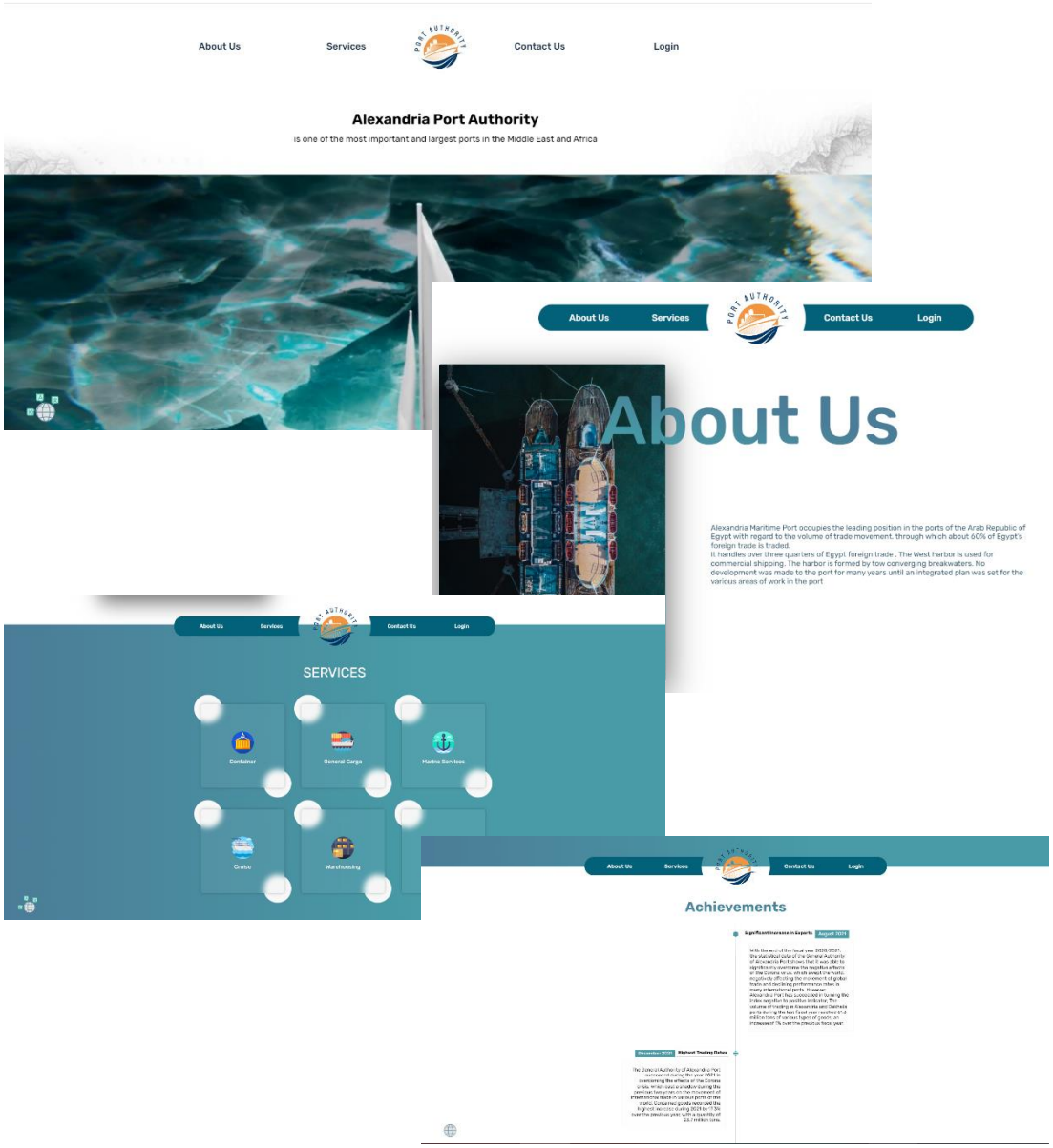


Figure 4- 21: Home Page Design

Forms:

When designing forms, prioritize usability, intuitiveness, and accurate data capture. Use a clean layout with clear labels and instructions. Group related fields together and use logical ordering. Utilize appropriate input controls and implement error handling and validation. Incorporate helpful features like auto-fill and suggestions for a seamless user experience.

Reports and Tables:

Design reports with clear and concise presentation of data. Use tables to organize and display information systematically. Prioritize readability with appropriate font sizes, colors, and formatting. Incorporate filters or sorting options for user interaction. For ship-specific reports, organize information logically with clear headings and sections.

Dashboard:

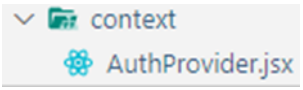
Design a comprehensive dashboard for monitoring port operations. Consider information hierarchy and visual representation. Include interactive charts for real-time ship activities. Use bar graphs or pie charts for workforce and administrative data. Integrate a weather widget for live updates. Include a logs report with a dynamic table for transparency. Prioritize usability, clarity, and efficient data visualization.

4.4.4.2 Design to Code Conversion

Breaking Down the Design: Once the design is understood, it needs to be broken down into smaller components such as headers, navigation menus, content sections, buttons, forms, etc. This helps in identifying the different elements that need to be coded.

1. Creating Construction Components in React:

Using JavaScript and Babel in conjunction with React to handle HTML code within JavaScript. With React being primarily focused on JavaScript, developers often utilize tools like Babel to convert HTML code into JavaScript for seamless integration in React applications.



2. Implementing Styles with CSS:

Cascading Style Sheets (CSS) are used to define the presentation and visual styles of the HTML structure.

- Responsive design techniques, such as media queries and flexible grids, ensure that websites display properly on different screen sizes and devices.
- Animating elements initially hidden with "display: none" in React can be challenging due to rendering behavior.
- Framer Motion is a library specifically designed for creating smooth and interactive animations in React.
- By incorporating Framer Motion, you can easily animate hidden elements and ensure animations function properly.
- Import the Framer Motion library and prepend the motion keyword to the desired component to access animation capabilities.
- Framer Motion allows you to create visually engaging and interactive user experiences.

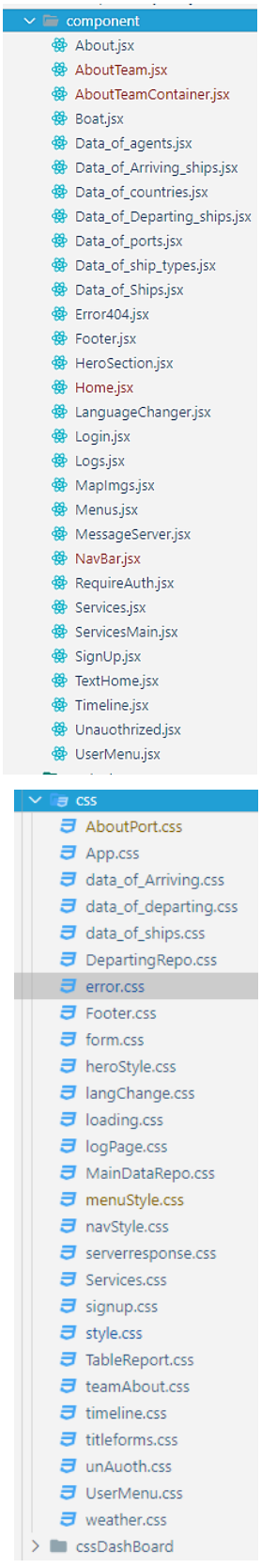


Figure 4- 22:Front-End File structure

3. Dealing with the data:

Dealing with data can be both dynamic and efficient, and the use of hooks in modern programming has revolutionized the way handling data-driven components. One intriguing approach involves utilizing a hook that triggers a method spontaneously when the component is rendered, alongside another hook that renders by controller through clicks or any other suitable method. When a component is rendered, the first hook comes into play, executing a predefined method or logic immediately without any explicit action from the user. This spontaneous execution allows for immediate data processing or retrieval, enabling real-time updates and displaying content as soon as it becomes available. This type of hook is particularly useful when dealing with rapidly changing data sources or when instant responses are required.

On the other hand,

The second hook allows for a controlled approach to data manipulation, triggered by user actions or external events. By associating the hook with suitable triggers, developers can precisely control when the rendering process occurs. This empowers them to design interactive user experiences where data is fetched, modified, or displayed upon user interaction, improving performance and conserving resources. By combining these hooks, websites can choose between automatic and manual data handling approaches for their forms.

```
import { useState, useEffect } from "react";

const useFetch = (configObj) => {
const { axiosInstance, method, url, requestConfig = {} } = configObj;
const [response, setResponse] = useState([]);
const [error, setError] = useState("");
const [loading, setLoading] = useState(true);

  useEffect(() => {
    const controller = new AbortController();

    const fetchData = async () => {
      try {
        const res = await axiosInstance[method.toLowerCase()](url, {
          ...requestConfig,
          signal: controller.signal,
        });
        setResponse(res.data.allentries);
      } catch (err) {
        setError(err.message);
        console.log(err);
      } finally {
        setLoading(false);
      }
    }

fetchData();
    return () => {
      controller.abort();
    }
  }, []);
  console.log("da el response", response);
  return [response, error, loading];
};
export default useFetch;
```

Figure 4- 23: hooks Snapshoot

This flexibility enables dynamic interfaces that update in real-time and empowers users to control data processing, enhancing the overall user experience.

4.4.4.2 Front-End functionality

1. **Forms**

   are an essential part of websites. It needs to be able to update, delete, add, and search data. This can be accomplished by using the backend of the website or application.The backend is the part of the website or application that is not visible to the user. It is where the data is stored and processed. When a user submits a form, the data is sent to the backend. The backend then updates, deletes or adds the data as needed.

1.1 Search in forms: search functionality in forms was designed to facilitate the process of updating specific data for users. When a user wants to update a particular piece of information, they often need to locate that data within a larger set. To streamline this process, the search feature allows users to search for specific data by entering relevant keywords or criteria. When a user clicks on the search button, it triggers the associated hook function, which initiates the search process. This hook function retrieves the entire dataset from the backend and prepares it for further processing. The retrieved data is typically an array or a collection of objects containing various fields and values. To determine whether a value matches the one stored in the backend, a map function is commonly employed. The map function iterates through each item in the dataset and compares the specified value with the corresponding field in the backend. If a match is found, it indicates that the desired data has been located. Once the search process is complete and the desired data is found, the user can proceed with updating the specific information. The search functionality

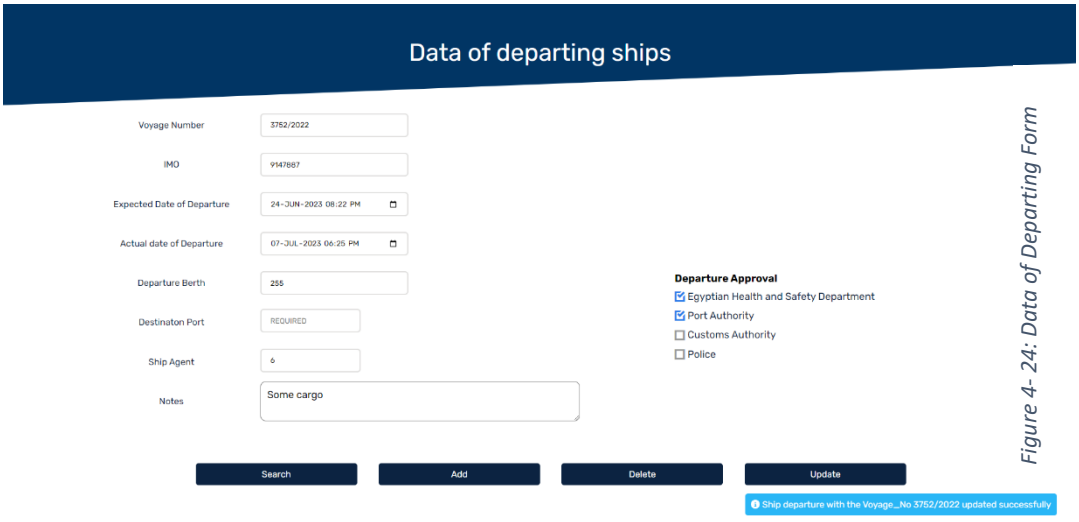


Figure 4- 24: Data of Departing Form

minimizes the effort required by users to manually navigate through a large dataset, enabling them to locate the data quickly and efficiently they wish to modify.

To address the issue of retrieved data not matching the date-local time input, we incorporated the Moment library. Moment provides flexible date manipulation capabilities, allowing us to easily convert and format time according to our specific requirements. By utilizing Moment, we successfully aligned the retrieved values with the date-local input, ensuring accurate and consistent data representation. Additionally, we encountered problems with receiving checkbox data, as it was in the form of 0 and 1 instead of true or false. To resolve this, we implemented logic to change the values to true if checked (1) and false if unchecked (0), ensuring the data aligns with the expected format.

1.2 **Add in forms**: The add function in forms is used to add new data to the backend. When a user submits a form, the add function is triggered and the data from the form is sent to the backend. The backend then processes the data and saves it to the database.

```
const clickAdd = ()=>{
  axiosFetch({
  axiosInstance: axiosAdd,
  method :'POST',
  url:'/',
  requestConfig :{
        Agent_Code:AgentCode,
        Perm_No: PermNo ,
        Perm_dt_st: PermDate,
        Perm_dt_end: PermExpDate ,
        Email: Email ,
        Telephone: Phone ,
        Address: Address,
        Agent_Name: AgentName
    }
```

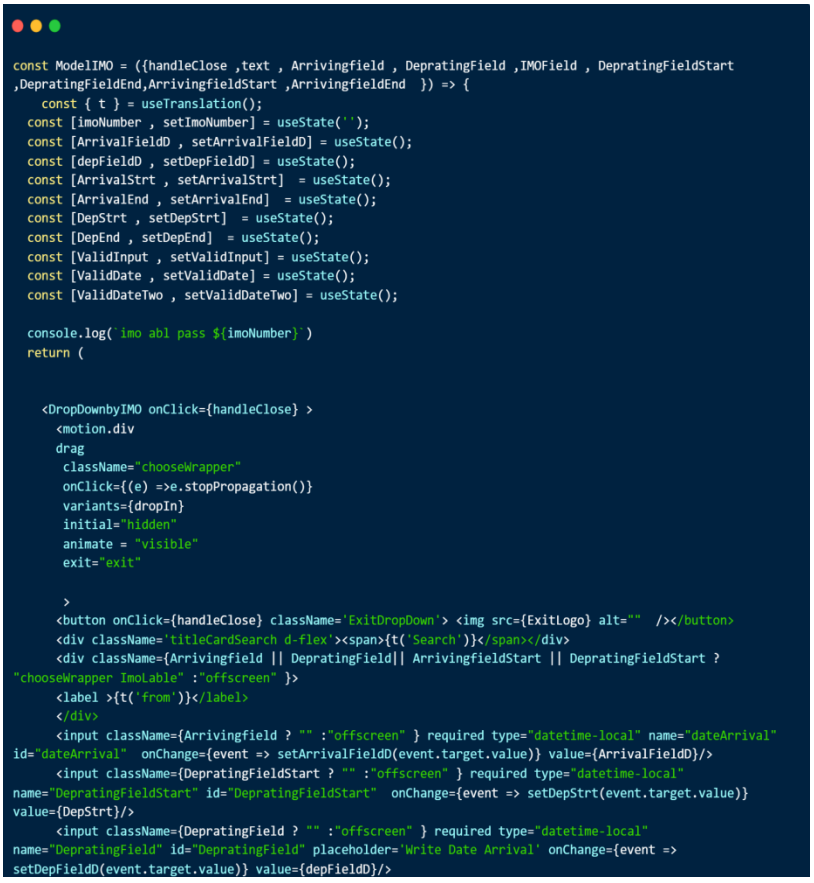
Figure 4- 25: Add Forms Snapshoot

1.3 **Update and delete in forms**: The update or delete function in forms is used to update or delete existing data in the backend. When a user submits a form, the update or delete function is triggered and the data from the form is sent to the backend. The backend then processes the data and updates or deletes the data in the database.

1.4 **Notification message in forms:** Upon receiving a request from the frontend, the backend server processes the data and determines whether any changes have been made. If modifications are detected, a response is generated indicating that the data has been altered. Once rendered, the response is automatically handled by a custom hook that logs the event in the "logs" table, specifically adding an entry with the action labeled as "POST". This logging mechanism serves the purpose of monitoring user interactions with forms, providing valuable insights into their activities. The logs table takes the name that is saved in the local storage.

2. Reports

Reports was made to let the employees know the information about the ship. There are reports that do not require any input from the user, such as the report of the name of agents and it requires to show the whole data in a table, so it is done by react table component. However, there are also reports that require input from the user, such as the report of the main ship, which requires the IMO of the ship, and the report of the ship arrival or ship departure frequency, which requires the date when the range of time where the ships arrived or departed and it is done by sending a props to the report search to see if the value was true or not and depends on those values the input will appear and When the user clicks on the button, the website takes the values in “State” which is function that is used in react router library that saves the value , value such as the report type, IMO, and date, and fetches the data from the backend spontaneously by hook. The hook fetches the entire table and then searches with those values in the “state” with the table that was fetched. If the data is found, the website will set the value to the report and can **print** the report using react print library. The issue that was faced the date was not as the date in local input thus the search would be an issue because the date in the backend was in different format so the date in the database needed to be substring to matches the value.

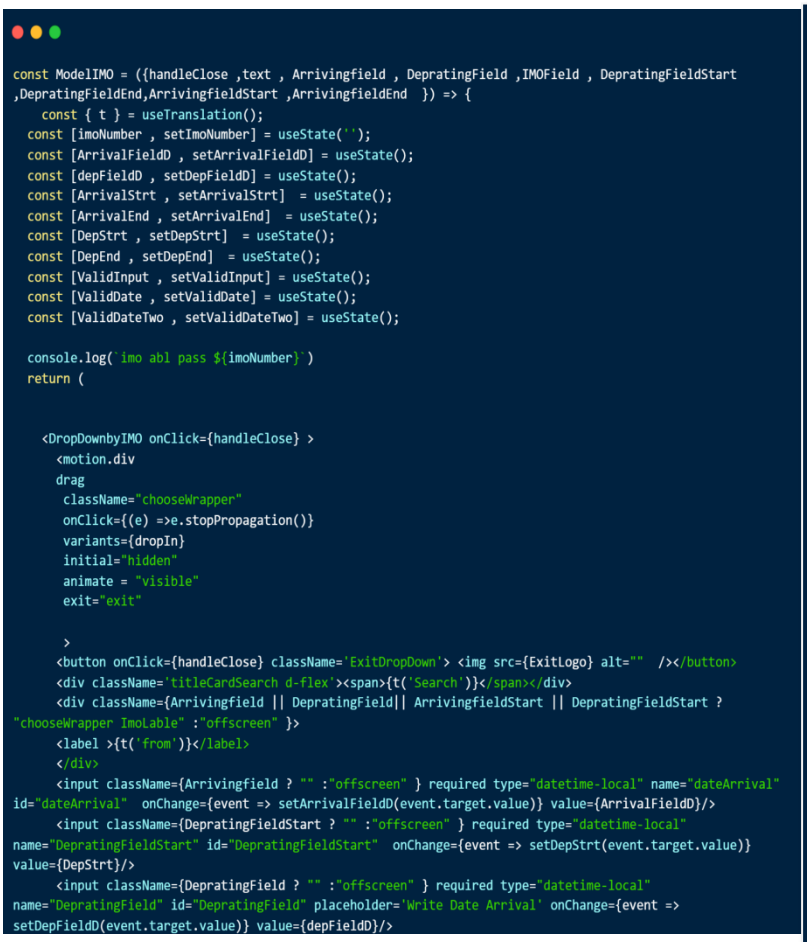


```
const ModelIMO = ({handleClose ,text , Arrivingfield , DepratingField ,IMOField , DepratingFieldStart
,DepratingFieldEnd,ArrivingfieldStart ,ArrivingfieldEnd  }) => {
    const { t } = useTranslation();
  const [imoNumber , setImoNumber] = useState('');
  const [ArrivalFieldD , setArrivalFieldD] = useState();
  const [depFieldD , setDepFieldD] = useState();
  const [ArrivalStrt , setArrivalStrt]  = useState();
  const [ArrivalEnd , setArrivalEnd]  = useState();
  const [DepStrt , setDepStrt]  = useState();
  const [DepEnd , setDepEnd]  = useState();
  const [ValidInput , setValidInput] = useState();
  const [ValidDate , setValidDate] = useState();
  const [ValidDateTwo , setValidDateTwo] = useState();

  console.log(`imo abl pass ${imoNumber}`)
  return (


    <DropDownbyIMO onClick={handleClose} >
      <motion.div
      drag
       className="chooseWrapper"
       onClick={(e) =>e.stopPropagation()}
       variants={dropIn}
       initial="hidden"
       animate = "visible"
       exit="exit"

       >
      <button onClick={handleClose} className='ExitDropDown'> <img src={ExitLogo} alt=""  /></button>
      <div className='titleCardSearch d-flex'><span>{t('Search')}</span></div>
      <div className={Arrivingfield || DepratingField|| ArrivingfieldStart || DepratingFieldStart ?
"chooseWrapper ImoLable" :"offscreen" }>
      <label >{t('from')}</label>
      </div>
      <input className={Arrivingfield ? "" :"offscreen" } required type="datetime-local" name="dateArrival"
id="dateArrival"  onChange={event => setArrivalFieldD(event.target.value)} value={ArrivalFieldD}/>
      <input className={DepratingFieldStart ? "" :"offscreen" } required type="datetime-local"
name="DepratingFieldStart" id="DepratingFieldStart"  onChange={event => setDepStrt(event.target.value)}
value={DepStrt}/>
      <input className={DepratingField ? "" :"offscreen" } required type="datetime-local"
name="DepratingField" id="DepratingField" placeholder='Write Date Arrival' onChange={event =>
setDepFieldD(event.target.value)} value={depFieldD}/>
```



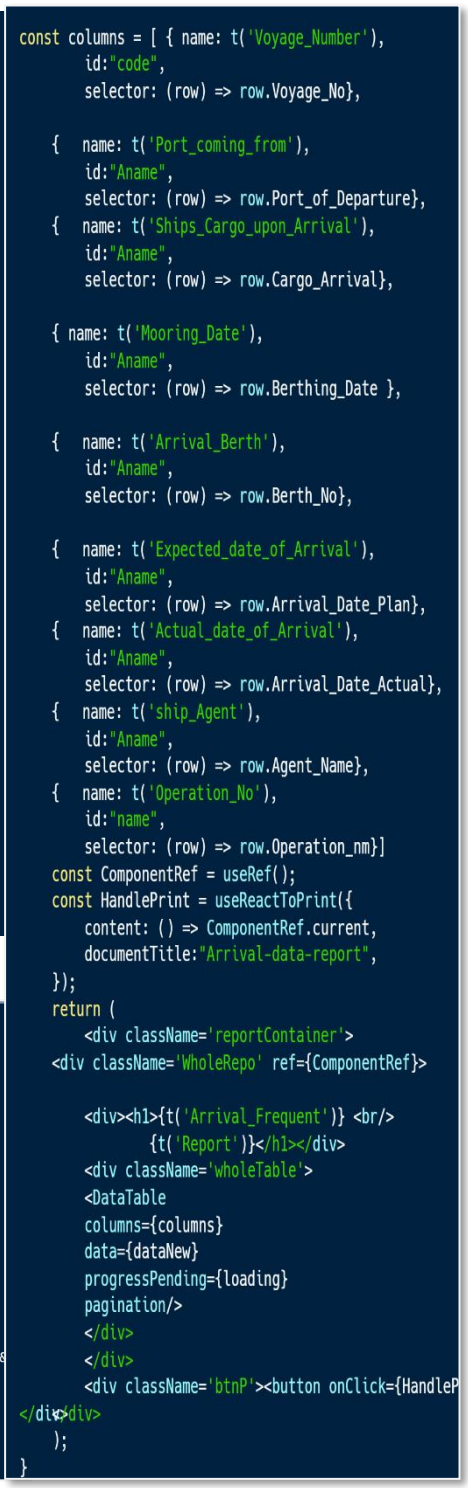
```
const ModelIMO = ({handleClose ,text , Arrivingfield , DepratingField ,IMOField , DepratingFieldStart
,DepratingFieldEnd,ArrivingfieldStart ,ArrivingfieldEnd  }) => {
    const { t } = useTranslation();
  const [imoNumber , setImoNumber] = useState('');
  const [ArrivalFieldD , setArrivalFieldD] = useState();
  const [depFieldD , setDepFieldD] = useState();
  const [ArrivalStrt , setArrivalStrt]  = useState();
  const [ArrivalEnd , setArrivalEnd]  = useState();
  const [DepStrt , setDepStrt]  = useState();
  const [DepEnd , setDepEnd]  = useState();
  const [ValidInput , setValidInput] = useState();
  const [ValidDate , setValidDate] = useState();
  const [ValidDateTwo , setValidDateTwo] = useState();

  console.log(`imo abl pass ${imoNumber}`)
  return (


    <DropDownbyIMO onClick={handleClose} >
      <motion.div
      drag
       className="chooseWrapper"
       onClick={(e) =>e.stopPropagation()}
       variants={dropIn}
       initial="hidden"
       animate = "visible"
       exit="exit"

       >
      <button onClick={handleClose} className='ExitDropDown'> <img src={ExitLogo} alt=""  /></button>
      <div className='titleCardSearch d-flex'><span>{t('Search')}</span></div>
      <div className={Arrivingfield || DepratingField|| ArrivingfieldStart || DepratingFieldStart ?
"chooseWrapper ImoLable" :"offscreen" }>
      <label >{t('from')}</label>
      </div>
      <input className={Arrivingfield ? "" :"offscreen" } required type="datetime-local" name="dateArrival"
id="dateArrival"  onChange={event => setArrivalFieldD(event.target.value)} value={ArrivalFieldD}/>
      <input className={DepratingFieldStart ? "" :"offscreen" } required type="datetime-local"
name="DepratingFieldStart" id="DepratingFieldStart"  onChange={event => setDepStrt(event.target.value)}
value={DepStrt}/>
      <input className={DepratingField ? "" :"offscreen" } required type="datetime-local"
name="DepratingField" id="DepratingField" placeholder='Write Date Arrival' onChange={event =>
setDepFieldD(event.target.value)} value={depFieldD}/>
```

```
const ArrivalFreq =()=>{
    const location = useLocation();
    const dataSearchStrt = location.state?.dataSearchStrt;
    const dateSearchEnd = location.state?.dateSearchEnd;
    const imoSearch = location.state?.imoSearch;
    const [data, error, loading] = useFetch({
        axiosInstance: axios,
        method :'GET',
        url:'/',
        requestConfig :{}
      });
      var arr =data;
      const [dataNew , setDataNew] = useState([]);
      useEffect(()=>{
        arr.map((datas)=>{
        if(datas.Arrival_Date_Actual){
        if(datas.IMO == imoSearch &&
datas.Arrival_Date_Actual.substring(0,datas.Arrival_Date_Actual.length-7) >= `${dataSearchStrt}:` &
datas.Arrival_Date_Actual.substring(0,datas.Arrival_Date_Actual.length-7) <= `${dateSearchEnd}:`)
        {
            setDataNew(current => [...current, datas])
        }
    }})
},[data])
```



```
const columns = [ { name: t('Voyage_Number'),
        id:"code",
        selector: (row) => row.Voyage_No},

    {   name: t('Port_coming_from'),
        id:"Aname",
        selector: (row) => row.Port_of_Departure},
    {   name: t('Ships_Cargo_upon_Arrival'),
        id:"Aname",
        selector: (row) => row.Cargo_Arrival},

    { name: t('Mooring_Date'),
        id:"Aname",
        selector: (row) => row.Berthing_Date },

    {   name: t('Arrival_Berth'),
        id:"Aname",
        selector: (row) => row.Berth_No},

    {   name: t('Expected_date_of_Arrival'),
        id:"Aname",
        selector: (row) => row.Arrival_Date_Plan},
    {   name: t('Actual_date_of_Arrival'),
        id:"Aname",
        selector: (row) => row.Arrival_Date_Actual},
    {   name: t('ship_Agent'),
        id:"Aname",
        selector: (row) => row.Agent_Name},
    {   name: t('Operation_No'),
        id:"name",
        selector: (row) => row.Operation_nm}]
    const ComponentRef = useRef();
    const HandlePrint = useReactToPrint({
        content: () => ComponentRef.current,
        documentTitle:"Arrival-data-report",
    });
    return (
        <div className='reportContainer'>
    <div className='WholeRepo' ref={ComponentRef}>

        <div><h1>{t('Arrival_Frequent')} <br/>
                {t('Report')}</h1></div>
        <div className='wholeTable'>
        <DataTable
        columns={columns}
        data={dataNew}
        progressPending={loading}
        pagination/>
        </div>
        </div>
        <div className='btnP'><button onClick={HandleP
</div></div>
    );
}
```

Figure 4- 26 Code snapshoot represent Reports.

Sign Up

The sign-up form available exclusively to the system administrator serves to add new users to the platform. This specialized form collects essential data from the user, such as name, email phone, and password. Once the administrator completes the necessary fields, they submit the form using a post method in the backend. If there are no errors or missing fields, the website will typically display a confirmation message indicating successful account creation. Behind the scenes, the information provided is securely transmitted to the server, where it undergoes validation and processing. The backend then creates a new user account based on the submitted data, ensuring the appropriate access levels and permissions are assigned. This exclusive sign-up process empowers the system administrator to meticulously manage user registrations and maintain control over user accounts within the platform.

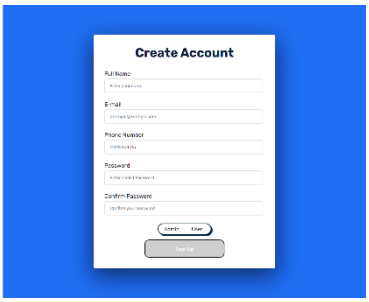


Figure 4- 27: Sign UP page.

```
import '../css/signup.css'
import { useEffect, useRef, useState } from 'react';
import { Player } from '@lottiefiles/react-lottie-player';
import {faCheck , faTimes , faInfoCircle} from '@fortawesome/free-solid-
import {FontAwesomeIcon } from '@fortawesome/react-fontawesome';
import { useTranslation } from 'react-i18next';
import axiosAdd from '../apis/SignUpAdd';
import useAxiosAdd from '../hooks/useAxiosAgent';
import MessageServer from './MessageServer';

const USER_REGEX = /^[a-zA-Z]+( [a-zA-Z]+)+$/i;
const PWD_REGEX = /^(?=.*[a-z])(?=.*[A-Z])(?=.*[0-9])(?=.*[!@#$!]).
const PHONE_REGEX = /^01[0125]\d{9}$/;
const EMAIL_REGX = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

const SignUp = () => {
  const { t } = useTranslation();
  const userRef = useRef();
  const errRef = useRef();
  const [user , setUser] = useState('');
  const [validName , setValidName] = useState(false);
  const [userFocus , setUserFocus] = useState(false);
  const [pwd , setPwd] = useState('');
  const [validPwd , setValidPwd] = useState(false);
  const [pwdFocus , setPwdFocus] = useState(false);
  const [matchPwd , setMatchPwd] = useState('');
  const [validMatch , setValidMatch] = useState(false);
  const [matchFocus , setMatchFocus] = useState(false);
  const [phoneNum , setphoneNum] = useState('');
  const [validphoneNum , setvalidphoneNum] = useState(false);
  const [phoneNumFocus , setphoneNumFocus] = useState(false);
  const [emailChan , setEmail] = useState('');
  const [validemail , setValidemail] = useState(false);
  const [emailFocus , setemailFocus] = useState(false);
  const [empChan , setEmp] = useState('');
  const [validEmp , setValidEmp] = useState(false);
  const [EmpFocus , setEmpFocus] = useState(false);
```

```
  const [responseText,setresponseText]=useState(false);
  const [errMsg , setErrMsg] = useState('');
  const [success , setSuccess] = useState(false);
  const [reponse,errortwo , loadingtwo ,axiosFetch] =
useAxiosAdd();
  const clickApi = ()=>{
    axiosFetch({
      axiosInstance: axiosAdd,
    method :'POST',
    url:'/',
    requestConfig :{
      Employee_Name: user ,
      Employee_Password:pwd ,
      Telephone: phoneNum ,
      Email: emailChan,
      Role: empChan
      }
   })
   if(reponse)
   {
    setresponseText(true)
   }
   const timer = setTimeout(() => {
    setresponseText(false);
  }, 5000);

   setUser('');
   setPwd('');
   setMatchPwd('');
   setphoneNum('');
   setEmail('');
   setEmp('');
  }

  useEffect (()=>{
    const result = USER_REGEX.test(user);
    console.log(result);
    console.log(user);
    setValidName(result);
  },[user])

  useEffect (()=>{
    const result = PWD_REGEX.test(pwd);
    console.log(result);
    console.log(pwd);
    setValidPwd(result);
    const match = pwd === matchPwd;
    setValidMatch(match);
  },[pwd , matchPwd])

  useEffect (()=>{
    const result = PHONE_REGEX.test(phoneNum);
    console.log(result);
    console.log(phoneNum);
    setvalidphoneNum(result);
  },[phoneNum])

  useEffect (()=>{
    const result = EMAIL_REGX.test(emailChan);
    console.log(result);
    console.log(emailChan);
    setValidemail(result);
  },[emailChan])

  useEffect(()=>{
    setErrMsg('');
  },[user , pwd , matchPwd , phoneNum , emailChan])

  useEffect (()=>{
    setValidEmp(true);
    console.log(empChan);
  },[empChan])
```

Figure 4- 28: code snapshoot represent Sign up.

2. Dashboard

3.1 Dashboard Analysis:

Front-End Development: The front end of the dashboard plays a significant role in presenting the analyzed data in a user-friendly manner. It utilizes various web technologies to retrieve and display relevant information to the users. Here's how it functions:

a. Data Presentation: The front end of the dashboard displays statistics on port activities, including ship arrivals, departures, and operations conducted. Users can interact with the dashboard by inputting date ranges or operation limits for analysis.
b. Dynamic Date Calculation: To enhance usability, the front end incorporates Moment.js library for dynamic date calculations. It allows easy selection of date ranges or specific years to analyze port operations, dynamically calculating the previous year's date range or the specified year based on user inputs.
c. User Inputs and Configuration: The front end provides options for users to input date ranges or operation limits, which are then passed to the back end for further processing and data retrieval.
d. Chart.js Integration: The front end integrates Chart.js library to visually present the analyzed data using bar charts, line charts, or pie charts. This visualization enhances user understanding of port activities and trends.

Back-End Functionality: While the front end focuses on data presentation and user interactions, the back end handles the logic and functionality required for data retrieval and calculation. The following components constitute the backend of the dashboard:

e. Data Retrieval: The back end connects to a database or an API that stores relevant port data, such as ship arrivals, departures, and operations conducted. It retrieves this data based on the user's inputs received from the front end.
f. Calculation of Port Statistics: Backend calculates port statistics like ship arrivals and departures from specific ports within a given date range, as well as frequency and types of operations performed.
g. Applying User Inputs: Backend filters and analyzes port data based on user inputs such as date range and operation limit to provide statistics that align with user requirements.

h. Data Processing and Formatting: Backend processes and formats calculated statistics into a suitable format for frontend display, aggregating information, performing calculations, and structuring it for visualization using Chart.js.

```
const ArrivalBoth = () => {
    const { t } = useTranslation();
    const currentDate = moment();
    const lastYear = currentDate.subtract(1, 'years');
    const lastYearValue = lastYear.year();
    const [dataFetch, error, loading] = useFetch({
        axiosInstance: ArrivalNoLimitGraph,
        method :'GET',
        url:'',
        requestConfig :{}
      });

    const [dataChart ,setDataChart] = useState({
        labels: [],
        datasets:[{
          label: "Arrivals",
          data: [],
          backgroundColor:["#54bebe", "#76c8c8", "#98d1d1", "#badbdb", "#dedad2", "#e4bcad", "#df979e",
"#d7658b", "#c80064" ,"#337280"]
          }]
      });

      useEffect(() => {
        if (dataFetch) {
          setDataChart({
            labels: dataFetch.map((datas)=> datas.Port_of_Departure),
            datasets: [{
              label: "Arrivals",
              data: dataFetch.map((datas) => datas.Arriving_Count),
              borderColor: 'rgb(30, 104, 233)',
              backgroundColor: ["#54bebe", "#76c8c8", "#98d1d1", "#badbdb", "#dedad2", "#e4bcad",
"#df979e", "#d7658b", "#c80064" ,"#337280"]
            }]
          });
        }
      }, [dataFetch]);

      const [reponse,errortwo , loadingtwo ,axiosFetch] = useAxiosFunction();
const clickAdd = ()=>{
  axiosFetch({
  axiosInstance: ArrivingGraphLimit,
  method :'GET',
  url:`?date=${dateArrivalRa}&limit=${limitpass}&`,
  requestConfig :{}
 })
 setDataChart({
    labels: reponse.map((datas)=>datas.Port_of_Departure),
    datasets:[{
        labels: "Arrivals",
        data:reponse.map((datas)=>datas.Arriving_Count),
        borderColor: 'rgb(30, 104, 233)',
        backgroundColor:["#54bebe", "#76c8c8", "#98d1d1", "#badbdb", "#dedad2", "#e4bcad", "#df979e",
"#d7658b", "#c80064" ,"#337280"]

    }]
  })
}
```

Figure 4- 29: code snapshoots for graphs

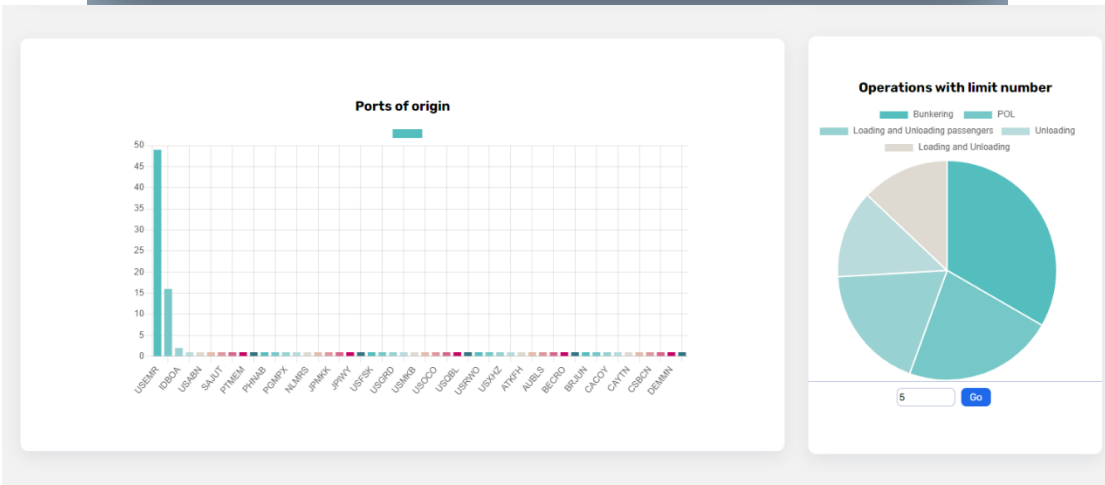


Figure 4- 30: graphs in dashboard

3.2 Cards: Fetches the entire employee table and then search inside it about the type of employee and start counting the admins and the employees

3.3 Weather: The weather in Alexandria varies from day to day, as it is dynamically updated based on the current date. By utilizing a weather API, real-time information about the weather conditions can be obtained for Alexandria. The API fetches data such as temperature, humidity, wind speed, and precipitation, providing an accurate depiction of the weather on any given day. This dynamic nature ensures that users receive the most up-to-date weather information about Alexandria. Whether it's sunny, cloudy, rainy, or windy, the weather API offers reliable data that enables individuals to plan their activities accordingly and stay informed about the atmospheric conditions in Alexandria.

3. Logs

Website logs include various types of information:

1. Audit Logs: Audit logs capture detailed records of specific actions performed by users or administrators within the website. They track activities such as user logins, account modifications, data access or modifications, configuration changes, and other administrative actions. Audit logs help ensure accountability, compliance, and support forensic analysis in case of incidents.
2. Error Logs: Error logs capture information about errors, exceptions, warnings, and other abnormal events that occur during the operation of the website. They often include stack traces, error messages, timestamps, and contextual data related to the error. Error logs help administrators identify and resolve software bugs, configuration issues, or other problems affecting the website's functionality.

Logs are important, here are some reasons:

1. Security Monitoring: Logs provide valuable information for monitoring the security of a website. By analyzing logs, administrators can identify and track suspicious activities such as unauthorized access attempts, potential security breaches, or unusual traffic patterns. This helps them take proactive measures to protect the website from potential threats.

2. Performance Optimization: Website logs also provide insights into the performance of various components, such as databases, servers, or network connections.

3. Forensic Analysis: In the unfortunate event of an incident or an attack on a website, logs act as forensic evidence. They can be used to reconstruct events, analyze the extent of damage, and investigate the cause of the incident. Logs may provide critical information for law enforcement agencies or third-party forensic experts involved in incident response.

4. Troubleshooting and Debugging: When something goes wrong on a website, logs serve as a valuable resource for troubleshooting and debugging. They contain detailed information about errors, exceptions, warnings, and other events that occurred during the operation of the website. Administrators can analyze these logs to identify the root cause of issues and take appropriate actions to resolve them.

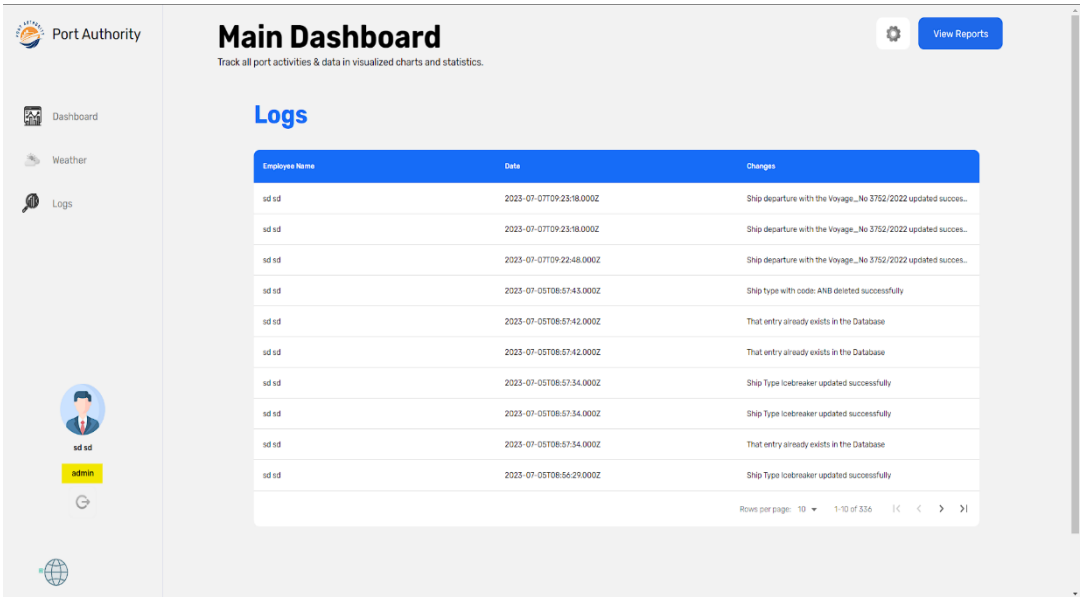


Figure 4- 31: LOGs page - Dashboard

4. Log In

Login functionality is an essential aspect of any website that deals with user accounts or personalized content. One of the most important reasons of login is users' authentication. Login provides a way to verify the identity of employees or any users accessing the website. It ensures that only authorized individuals can access restricted areas, perform certain actions, or view specific content. This helps protect sensitive information and maintain the security of user data. Another one is user data management. Login enables websites to collect and manage user data effectively. When accounts are created and users log in, their information can be securely stored and associated with their profiles. This allows websites to store user preferences, track user activities, and deliver targeted advertisements or notifications.

```
axiosFetch({
      axiosInstance: axiosAdd,
      method: 'POST',
      url: '/',
      requestConfig: {
        Email: email,
        Employee_Password: pass,
      },
    });
    if (errortwo) {
      setresponseText(true);
    }

    const timer = setTimeout(() => {
      setresponseText(false);
    }, 5000);
    setClick(true);
  };

  useEffect(() => {
    if (res) {
      setUser(res.Employee_Name);
      setRole(res.Role);
      console.log(`name hwa da ${res.Employee_Name}`);
      console.log(res.Role);

      localStorage.setItem('NameUser', res.Employee_Name);
      localStorage.setItem('roleUser', res.Role);
      setAuth({ user, pass, roles }); // access tokens here
      if (roles === 'other') {
        navigate(otherPath, { replace: true });
      }
      if (roles === 'admin') {
        navigate(fromAdmin, { replace: true });
      }
    }
  }, [res]);

  const [error, setError] = useState(false);

  useEffect(() => {
    userRef.current.focus();
  }, []);

  useEffect(() => {
    const clearLocalStorage = () => {
      const hours = 0.0167;
      const now = new Date().getTime();
      const setupTime = localStorage.getItem('setupTime');

      if (setupTime == null || now - setupTime > hours * 60 * 60 *
1000) { localStorage.clear();
        localStorage.setItem('setupTime', now);
      }
    };
    clearLocalStorage();
  }, []);
  useEffect(() => {
    seterrMsg('');
  }, [email, pass]);
```

Figure 4- 32: sign in page

5. Protected Routes

Every user according to their role can access some components unless they are admins so they can access everything. Protected routes helped us ensure the security of sensitive data by restricting access to authorized users only. By implementing authentication and authorization mechanisms, you can control who can view or modify certain parts of the application. This is particularly crucial when dealing with personal information or any other confidential data. Protected routes also play a significant role in maintaining user privacy. By requiring authentication, the identity of users can be verified before granting them access to specific resources. This helps prevent unauthorized individuals from accessing sensitive user information or performing actions on behalf of others. Without protected routes, anyone could potentially access restricted parts of application. By implementing authentication and authorization, we establish barriers that prevent unauthorized users from reaching sensitive areas and functionalities. This helps protect against malicious activities such as data breaches, unauthorized modifications, or misuse of application features.

Protected routes can be implemented using a combination of local storage and cookies. Here's a general approach to achieving this:

```
import { useLocation, Navigate, Outlet } from "react-router-dom";
import useAuth from "../hooks/useAuth";

const RequireAuth = ({ allowedRoles }) => {
    const { auth } = useAuth();
    const location = useLocation();
console.log(window.location.pathname)
console.log(` da array${ allowedRoles }`)
console.log(` da role${ auth.roles }`)
const userType = localStorage.getItem('roleUser') ;
const userName= localStorage.getItem('NameUser') ;
console.log(` da role bata3 el local storage${ userType }`)

    return (
      allowedRoles?.some(role => userType?.includes(role))
            ? <Outlet />
            : userName
                ? <Navigate to="/unauthorized" state={{ from: location }}
replace />      : <Navigate to="/login" state={{ from: location }} replace
/>  );
}

export default RequireAuth;
```

Figure 4- 33:code snapshoot Protected Routes

When a user visits a page in your React application, the router will match the URL to one of the routes that you have defined. If the match is successful, the router will render the component that is associated with that route. In addition to matching the URL, the router can also check the user's role. This is done by using the useAuth hook, which returns the user's role from the local storage. If the user is not logged in, the useAuth hook will return null. Once the router has determined the user's role, it can render the appropriate menu. For example, if the user is an admin, the router might render a menu with links to all of the administrative pages. If the user is an employee, the router might render a menu with links to all of the employee pages. If the user does not have the authentication to access a particular page, the router will redirect them to a page that indicates that they are unauthorized. Finally, if there is no user logged in and the user tries to access a page, the router will redirect them to the login page.

Router Links

A router is responsible for handling the navigation between different pages or views within a single-page application (SPA). The main advantage of using Router Links is that they enable navigation without requiring a full page reload. When a user clicks on a Router Link, the router intercepts the click event, prevents the default behavior (which would be to reload the whole page), and instead updates the content dynamically based on the specified route or URL. Router Links often accept parameters such as the destination URL or the target view/component to render. They can be used with relative paths, absolute URLs, or even dynamic segments that get substituted based on the specified parameters.

When using React Router, you typically wrap the entire application inside the <BrowserRouter> component. This component utilizes the HTML5 History API to handle navigation and keep the UI in sync with the URL. It allows to create declarative routes and map them to specific components in the application.

```
<BrowserRouter>
    <Routes>
        {/* public paths */}
        <Route path="/" element = {<Home/>}/>
        <Route path="/Login" element = {<Login/>}/>
        <Route path="/unauthorized" element={<Unauothrized/>} />
        <Route path="/AboutTeam" element={<AboutTeamContainer/>} />

        {/* admin paths */}

        <Route element = {<RequireAuth allowedRoles={['admin']} />}>
        <Route path="/menu" element = {<Menus/>}/>
        <Route path="/menu/DataofArrivingships" element =
{<Data_of_Arriving_Ships/>}/>
        <Route path="/menu/DataofDepartingships" element =
{<Data_of_Departing_Ships/>}/>
        <Route path="/menu/Signup" element = {<SignUp/>}/>
        <Route path="/menu/ShipTypeData" element = {<ShipTypeData/>}/>
        <Route path="/menu/CountriesData" element = {<CountriesData/>}/>
        <Route path="/menu/AgentData" element = {<AgentData/>}/>
        <Route path="/menu/PortsData" element = {<PortsData/>}/>
        <Route path="/menu/DataofShips" element = {<Data_of_Ships/>}/>
        <Route path="/Dashboard" element = {<DashContainer/>}>
        <Route path="Analysis" element = {<AnalysisContainer/>}/>
        <Route path = 'Weather' element={<Weather/>} />
        <Route path="Logs" element={<Logs />} />
        </Route>
        </Route>
        {/* both */}
        <Route element = {<RequireAuth allowedRoles={['admin' , 'other']} />}>
        <Route path="/menu/CountriesRepo" element = {<CountriesRepo/>}/>
        <Route path="/menu/EmployeeRepo" element = {<DataofEmp/>}/>
        <Route path="/menu/AgentDataRepo" element = {<AgentDataRepo/>}/>
        <Route path="/menu/MainDataRepo" element = {<MainDataRepo/>}/>
        <Route path="/menu/PortDataReport" element = {<PortDataReport/>}/>
        <Route path="/menu/ArrivalFrequent" element = {<ArrivalFreq/>}/>
        <Route path="/menu/ArrivalRepo" element = {<ArrivalRepo/>}/>
        <Route path="/menu/ShipArrivalRepo" element = {<ShipArrivalRepo/>}/>
        <Route path="/menu/DepartingRepoPeriod" element =
{<DepartingRepoPeriod/>}/>
        <Route path="/menu/ShipsArrivalPeriod" element =
{<ShipsArrivalPeriod/>}/>
        <Route path="/Menuemployee" element = {<UserMenu/>}/>
        <Route path="/menu/DepartingRepo" element = {<DepartingRepo/>}/>
        <Route path="/menu/Shiptypes" element = {<ShipTypeRepo/>}/>
        <Route path="/menu/DepartingRepoFreq" element = {<DepFreq/>}/>
        </Route>
        <Route path="*" element = {<Error404/>}/>

    </Routes>
</BrowserRouter>
```

Figure 4- 34:Code snapshoot Router Links

By wrapping the routes in the <BrowserRouter> component, the application can handle navigation based on the current URL and render the appropriate components accordingly. <Route> to define the routes and components that should be rendered based on the current URL.

```
<Link className='LinkNav' to="/menu/Shiptypes">
<a className='btn-proceed'>{t('Ship_Types_btn')}<img src={arrowPic} className='arrow-style
</Link></a>
```

<Link> is used for creating clickable links that trigger navigation within the application in the menu.

Business Model Canvas

Key Partnerships	Key Activities	Value Propositions	Customer Relationships	Customer Segments
- Customs Authority **- Import and Export Monitoring Agency** **- Police** **- Logistics Companies** **- Egyptian Health and Safety Department** **- Warehouses** **- Security**	- Centralized Database - Monitor Ship Movement - Information Sharing - System Platform Managment	- Improved port information flow and accuracy - Improved terminal planning - Improved port call efficiency - Improved port service planning - Enhancing collaboration between government agencies	- Data Exchange	- Shipping Agent - EAFMS - GOEIC - Customs - Container Companies
	Key Resources - Port Authority Personnel - Shipping Agent - Database		**Channels** - Port Website	

Cost Structure	Revenue Streams
- Server and DB Hardware and System Enviroment - System Development - Maintenance - Port Authority Personnel	- Initial Price fee

Table5- 1: Business Model

Chapter Six

6 Hardware and software requirement

6.1 Cost

This report provides a thorough cost analysis of a software system designed for a user base of 2,000 people, considering unique system specifications. The system requires 10 CPUs for the backend and a database server with a dual-core Xeon 4-core processor and 16 GB of RAM.

1.0 Backend Server Costs

The backend server requires 10 CPUs to handle the load produced by 2,000 users. The server that can support this requirement costs approximately $255. This constitutes a significant portion of the initial costs associated with this setup.

- 10 CPUs Backend Server: $255

2.0 Database Server Costs

The database server's specifications include a dual-core Xeon 4-core processor and 16 GB of RAM, enabling it to manage up to 1,600 concurrent connections. This server costs around $245. This cost is also treated as a one-time upfront cost in this analysis.

- Database Server (2x Xeon 4-core, 16GB RAM): $245

3.0 Operation and Maintenance

Operation and maintenance costs consist of the recurring expenses necessary for server maintenance, system upgrades, and routine tasks to ensure the server and software's optimal functioning.

- System Maintenance: $100 - $500 per year (subject to the complexity of maintenance tasks)
- Upgrades: $50 - $300 per year (depending on the frequency and extent of upgrades)

4.0 Training and Support

Expenses related to training staff to use the new software and the continuous cost of support services are included in this category.

- Support Services: $500 - $1500 per year (this may vary based on the chosen service level agreement)
- Training: $100 - $500 per user (assuming the cost decreases as the number of users increases)

5.0 Potential Upgrade Costs

Over time, to keep pace with technological advancements, ensure compatibility with other systems, and maintain security, software systems need to be upgraded. These upgrades can represent a substantial cost.

- Software Upgrades: $100 - $500 per year

6.2 Ethics

1. The ethical considerations should include:
    a. Respecting the privacy of the data being shared on the website.
    b. Eliminating discrimination against any person or group.
    c. Maintaining accurate and secure data.

2. When sharing data with third parties, and honoring users' right to access their own data.

3. If a website is used for commercial purposes such as selling products or collecting customer data, then ethical considerations should include ensuring proper security around customer data and using it only as intended by customers.

4. Websites should ensure that none of the products they offer or advertise violate any laws or regulations.

5. Websites should follow any applicable industry standards in protecting customers' privacy while using Oracle.

Chapter Seven

7 Conclusion and Future Work

Conclusions:

Digital transformation in ports results in streamlined operations by automating and standardizing the exchange of information between port authorities, shipping lines, freight forwarders, and other stakeholders. This leads to increased efficiency, reduced paperwork, and improved data accuracy.

1. Improved Data Accuracy and Timeliness: Digital transformation in ports facilitates real-time data exchange, reducing the reliance on manual processes and paper-based documentation. This helps in minimizing data entry errors, ensuring data accuracy, and providing timely information to all parties involved in port operations.

2. Increased Productivity and Cost Savings: By automating data exchange processes, Digitalization eliminates time-consuming manual tasks and reduces administrative costs associated with paper-based documentation, data entry, and information retrieval. This leads to increased productivity and significant cost savings.

3. Efficient Customs Clearance and Compliance: EDI integration with customs systems allows for electronic submission of customs declarations and related documents. These speeds up the clearance process, reduces delays, and improves compliance with customs regulations and procedures.

4. Collaboration and Integration: Digitalization promotes collaboration and integration between different stakeholders in the seaport ecosystem. It facilitates seamless information flow, coordination, and communication, fostering better relationships and more efficient operations.

5. Security and Data Protection: Digitalization systems implement robust security measures to protect sensitive data transmitted between parties. Encryption, digital signatures, and authentication mechanisms ensure data confidentiality and integrity, mitigating the risk of unauthorized access or tampering.

6. Scalability and Future Readiness: Digitalization solutions in seaports can be scaled up to handle increasing volumes of data and accommodate the evolving needs of the industry. They are adaptable to changing technologies, standards, and regulations, ensuring long-term viability and future readiness.

7. This specialized system has particular requirements in terms of backend and database server configurations. With the total initial hardware cost of $500 for serving 2000 users, the system appears to be economically viable, assuming regular operation and maintenance costs stay within the projected range. This system can be scaled easily too due to the software's design to be scalable.

8. Digital transformation in ports brings significant advantages to the maritime industry. By streamlining operations, improving data accuracy and timeliness, Digitalization enables seaports to operate more efficiently. The automation of data exchange processes reduces paperwork and administrative costs, leading to increased productivity and substantial cost savings. Additionally, Digitalization facilitates efficient customs clearance and compliance, fostering smoother trade flows. Stakeholders in the seaport ecosystem. With robust security measures in place, Digitalization ensures the protection of sensitive data during transmission. The scalability and future readiness of Digitalization solutions make them adaptable to evolving technologies, standards, and regulations. Overall, the adoption of Digitalization in seaports revolutionizes port operations, promotes efficiency, and paves the way for a more streamlined and connected maritime industry.

Future work:

Smart ports refer to the integration of advanced technologies and digital solutions to optimize the efficiency, safety, and sustainability of port operations. These ports leverage various digital technologies, including the Internet of Things (IoT), artificial intelligence (AI), big data analytics, and automation, to enhance connectivity, visibility, and decision-making processes.

Intelligent Infrastructure: Smart ports utilize sensors and IoT devices to monitor and manage various aspects of port infrastructure, such as container terminals, berths, cranes, and warehouses. This enables real-time monitoring, predictive maintenance, and optimized asset utilization.

Advanced Data Analytics: Smart ports leverage big data analytics to process and analyze vast amounts of data collected from different sources. This allows for better forecasting, optimization of port operations, and data-driven decision-making. Insights gained from data analytics help improve operational efficiency and resource allocation.

References

1. Chen, J., Notteboom, T., & Wang, T. (2018), "Information Technology Adoption in Seaports: A Systematic Review",Transport Reviews.
2. Chen, X., Liu, Y., Wang, Q. (2020) , "Integration of Electronic Data Interchange and Blockchain Technology in Seaports", Transportation Research Part E: Logistics and Transportation Review.
3. Chen, Y., Zhang, H., & Wang, Y. (2020), "Integration of Electronic Data Interchange and Internet of Things for Seaport Operations", Transportation Research Part E: Logistics and Transportation Review.
4. Dong, J., & Song, D. P. (2017), "The Impact of Electronic Data Interchange on Port Performance: A Review", International Journal of Shipping and Transport Logistics.
5. Gupta, R., Mishra, A., Kumar, A.(2021), "Security and Privacy Issues in Electronic Data Interchange in Seaports", Maritime Policy & Management.
6. Lee, H., Kim, S., & Park, C. (2017), "EDI Implementation in Seaport Operations: Challenges and Solutions", International Journal of Shipping and Transport Logistics.
7. Lee, S., Kim, H., Park, C.(2017) , "Exploring the Impact of Electronic Data Interchange on Seaport Efficiency", International Journal of Logistics Management.
8. Li, X., et al. (2020), "Blockchain Technology for Secure and Transparent Seaport Databases", Transportation Research Part E: Logistics and Transportation Review.
9. Liu, Y., et al. (2020), "Big Data Analytics in Seaport Databases: Challenges and Opportunities", Transportation Research Part E: Logistics and Transportation Review.
10. Notteboom, T., & Rodrigue, J. P. (2018), "IT-Enabled Collaboration in Seaports: A Review of Models and Practices", Maritime Economics & Logistics.
11. Notteboom, T., & Yang, Z. (2017), "IT-Driven Business Process Innovations in Seaports: A Review", Transportation Research Part A: Policy and Practice.
12. Nunes, B., Sarmento, M., & Caldeirinha, V. (2021), "Electronic Data Interchange Adoption in Seaports: A Systematic Literature Review", Maritime Policy & Management.
13. Rodrigues, L., Carvalho, M., Branco, F.(2018) , "Challenges and Barriers to Electronic Data Interchange Adoption in Seaports", International Journal of Shipping and Transport Logistics.
14. Smith, J., Johnson, A., & Brown, C. (2018), "Database Systems for Seaport Operations: A Review", Journal of Maritime Research.
15. Smith, J., Johnson, A., Brown, L.(2019), "Electronic Data Interchange in Seaports: A Literature Review and Future Research Directions", Journal of Maritime Economics.
16. Tan, K. H., & Wang, X. (2016), "Enhancing Port Operations Efficiency through Electronic Data Interchange: A Review and Framework", Transportation Research Part E: Logistics and Transportation Review.
17. Wang, L., et al., (2019), "Data Warehousing in Seaport Management: A Comprehensive Review", International Journal of Information Management.
18. Yang, Z., & Ng, A. K. Y. (2018), "Emerging Trends in IT Applications for Seaport Operations: A Review", Transportation Research Part C: Emerging Technologies.
19. Zhang, M., & Wu, X. (2017), "Spatial Databases in Seaport Planning and Operations: A Review", Maritime Policy & Management.
20. Zhang, Y., & Abhichandani, T. (2019), "Cybersecurity Challenges in Seaport IT Systems: A Review", International Journal of Shipping and Transport Logistics.