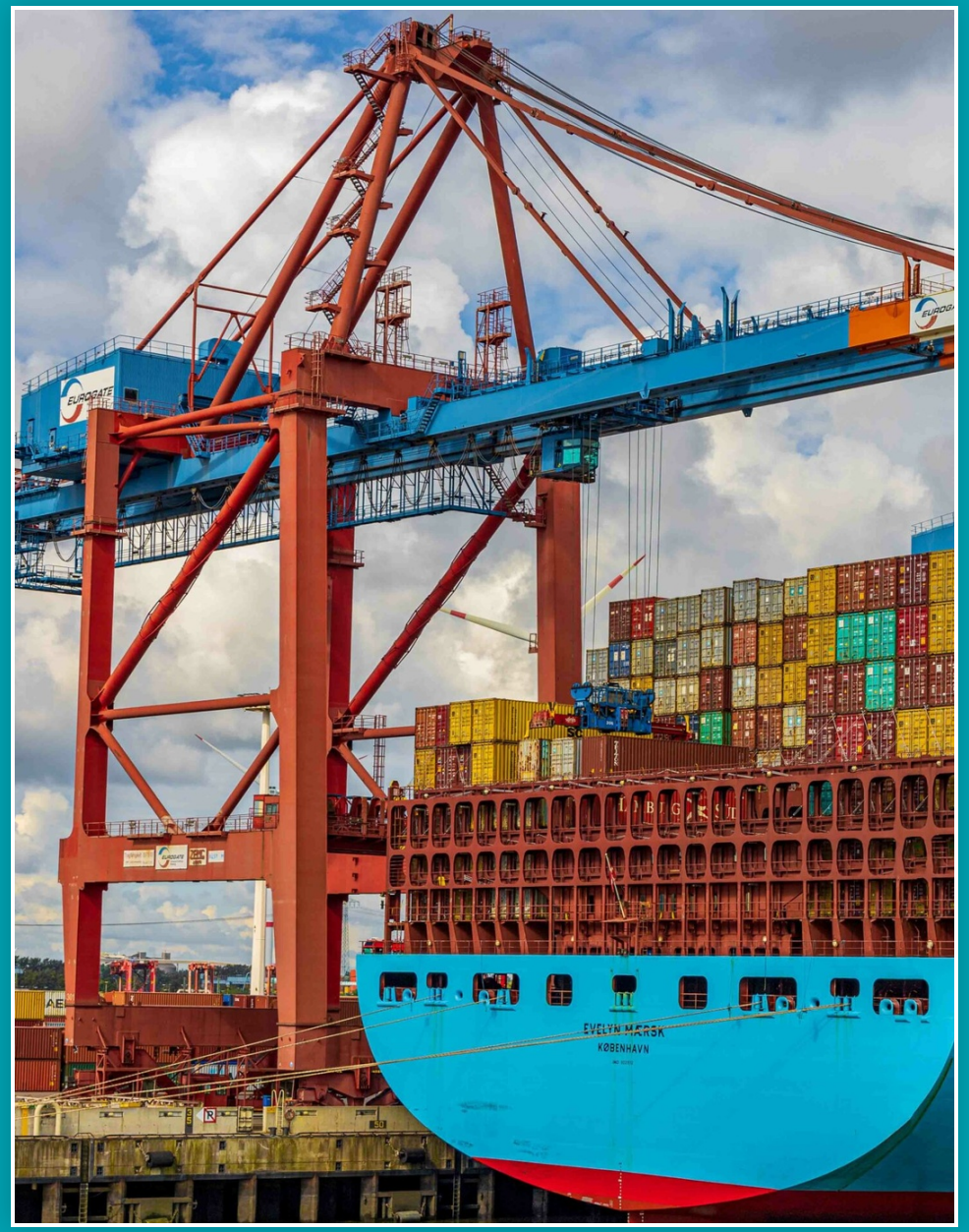
Digital Transformation in **Ports.**



Team Members.

Habiba Hossam

Front-end
Developer

Ahmed Ashraf

DevOps
Developer

Abdelrahman Mohamed

Back-end
Developer

Mohamed Gamal Abd El-Nasser

Database
Developer

Sarah Osama

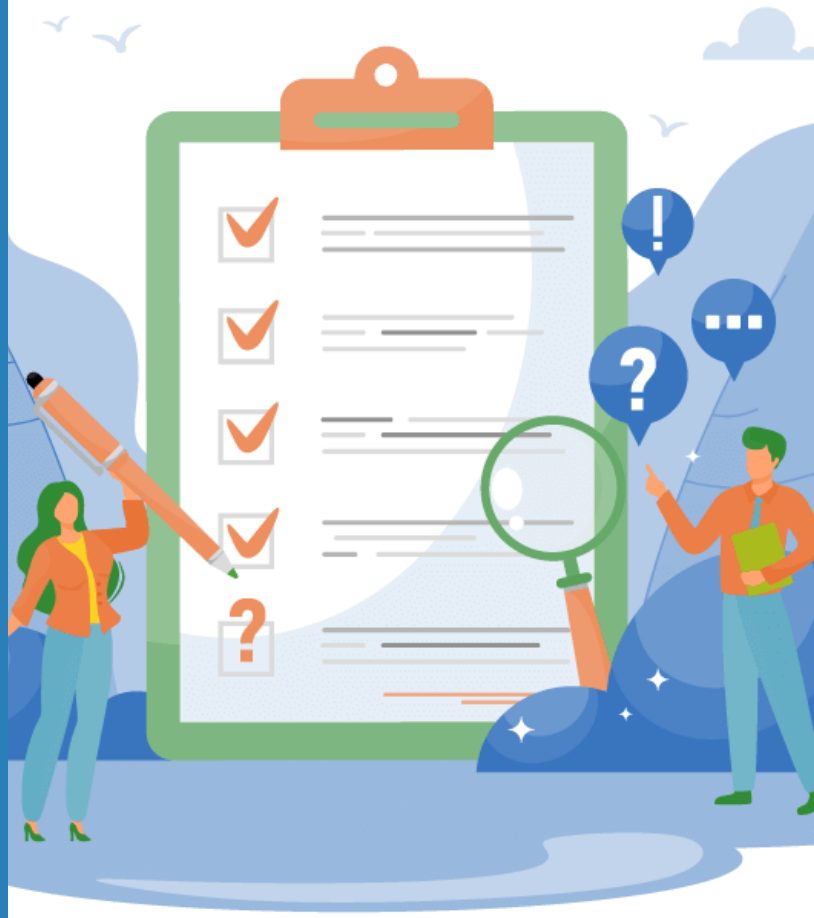
Front-end
Developer

Content

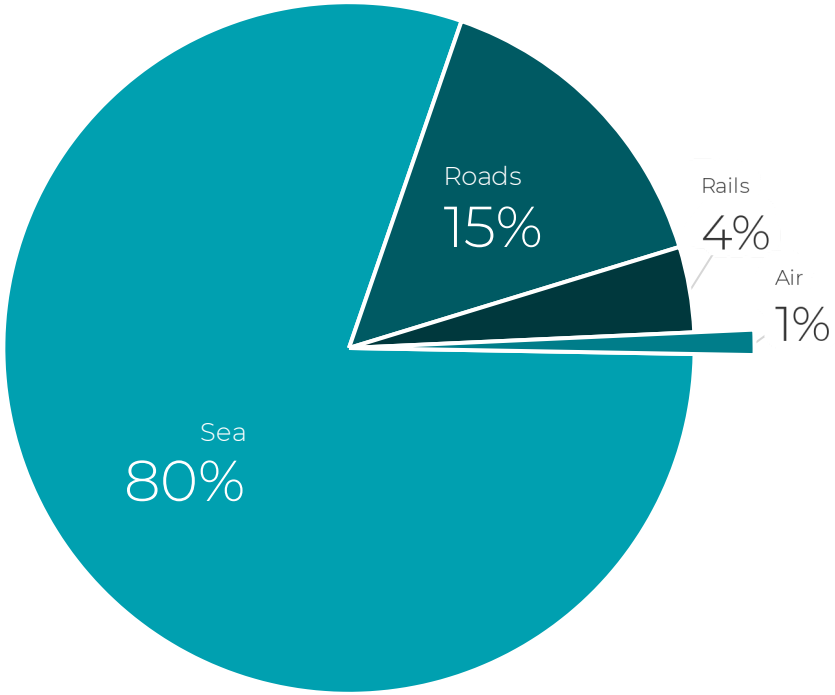
Content (cont.)

Abbreviations

- IOT: Internet Of Things
- PCS: Port Community System
- EAFMS: Egyptian Authority For Maritime Safety
- GOEIC: General Organization For Export and Import Control
- IMO: International Maritime Organization
- DB: Database
- CRUD: create, read, update and delete



Introduction

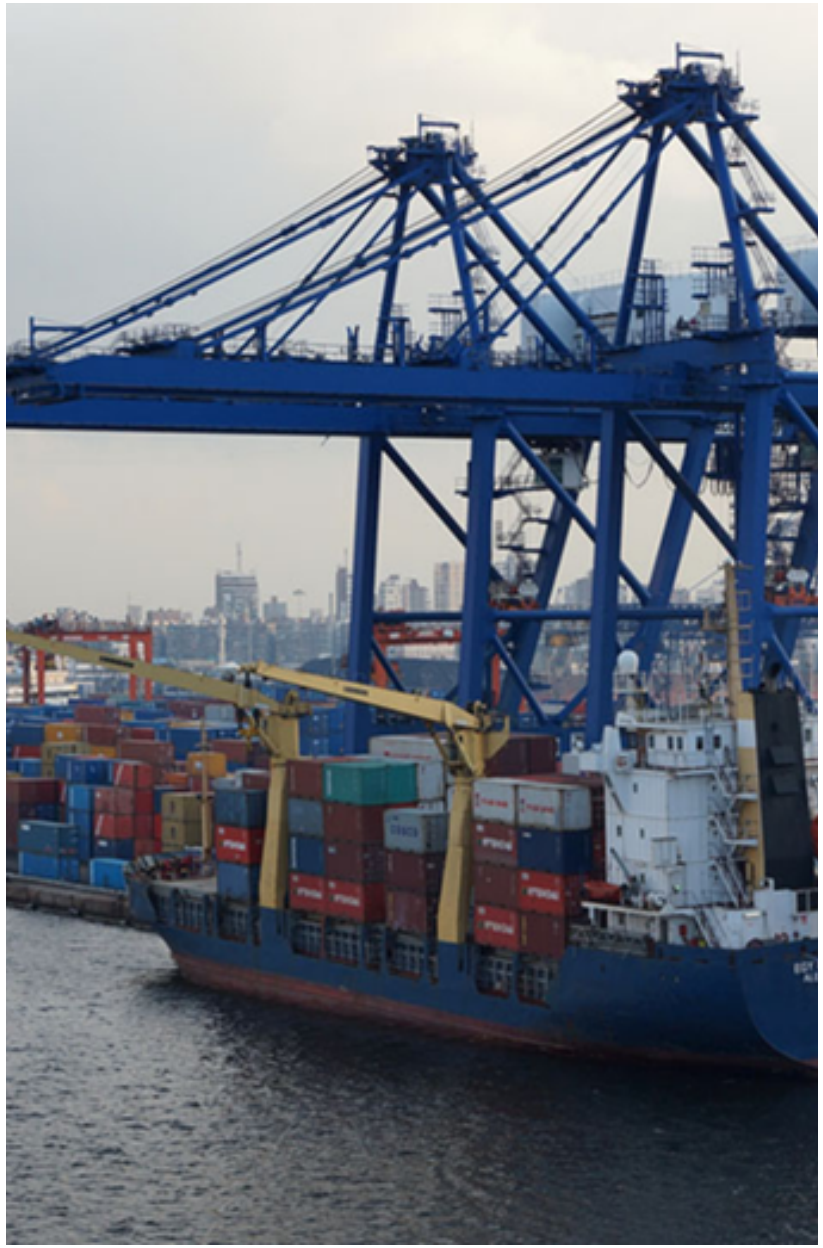


The volume of international trade in goods

Review of Maritime Transport 2021

About the Ports

- Port Definition
- Ports and global economy



About the ports

Egyptian Ports

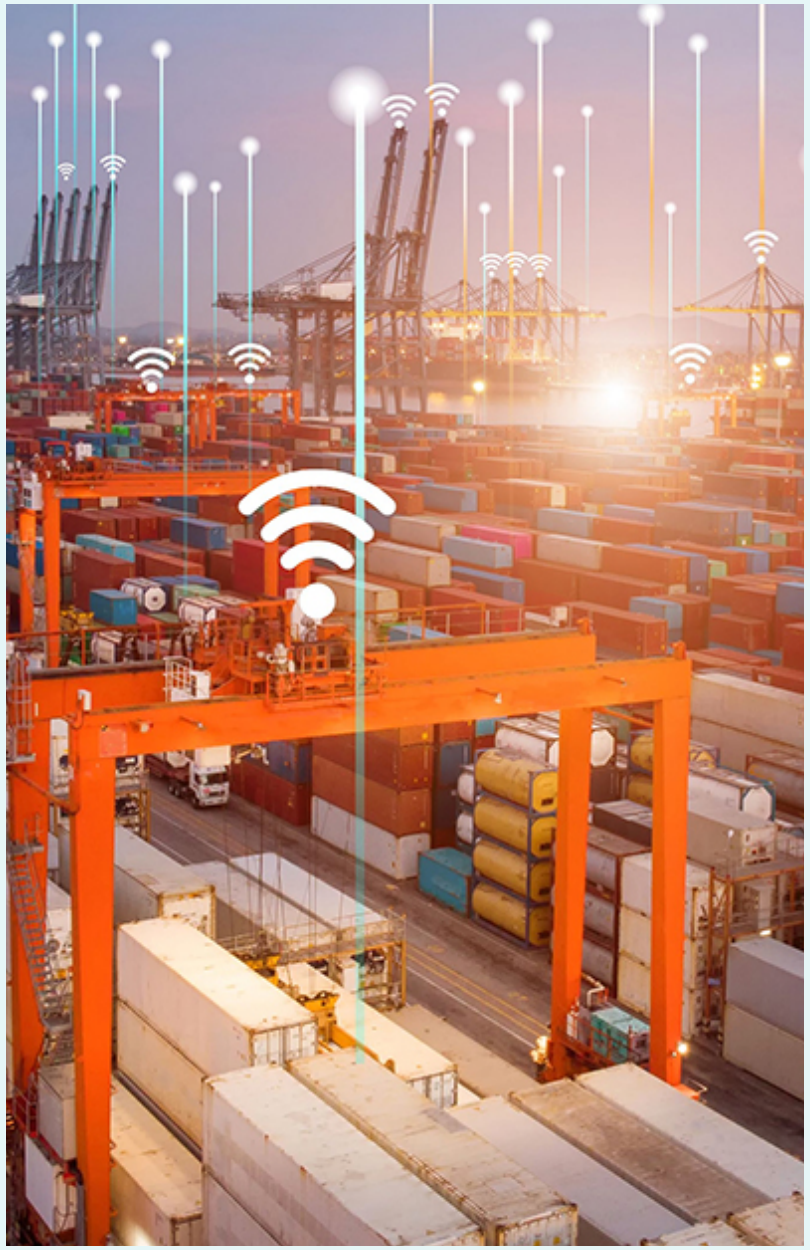
five major commercial ports in Egypt: Greater Alexandria Port (Alexandria and Dekheila port), Damietta Port, Port Said Port, Suez Port (including Adabiya Port) and Safaga Port.

Alexandria Port

- Alexandria Maritime Port occupies the leading position in the ports of Egypt.
- Alexandria Maritime Port occupies the leading position in the ports of the Arab Republic of Egypt with regard to the volume of trade movement, through which about 60% of Egypt's foreign trade is traded. (https://apa.gov.eg/en/)

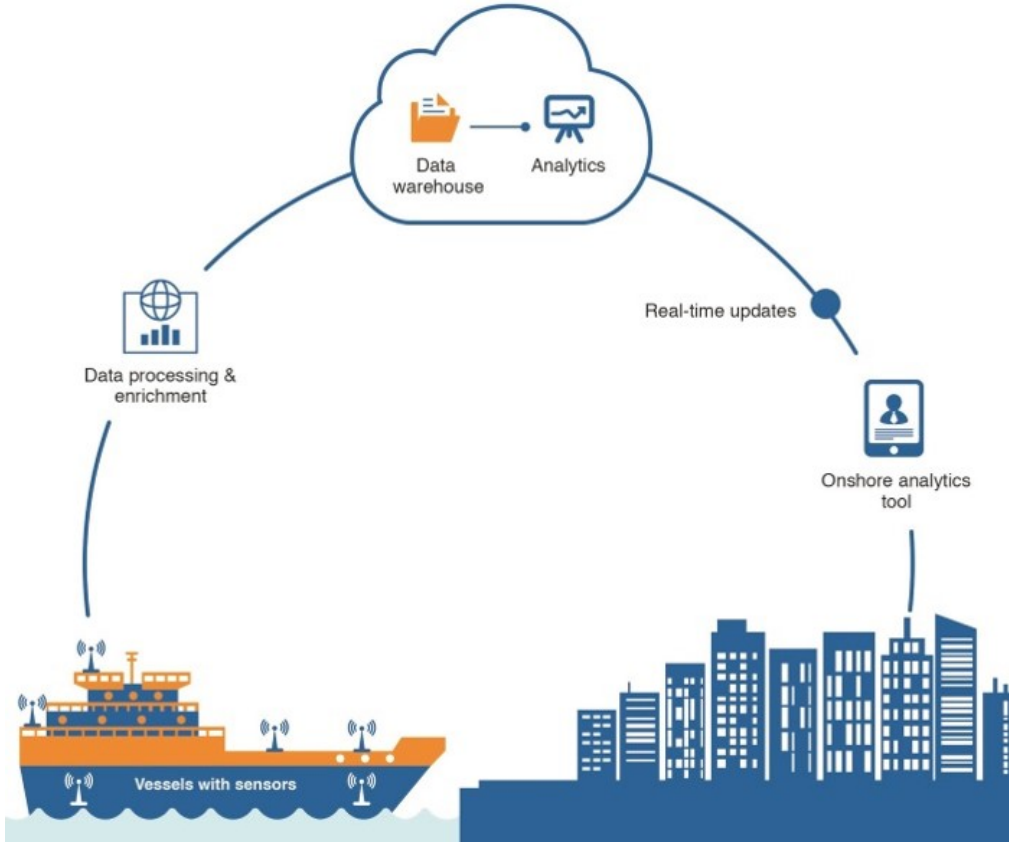
Smart ports

- Improvement of port productivity and efficiency.
- AI, IoT, and Big Data.



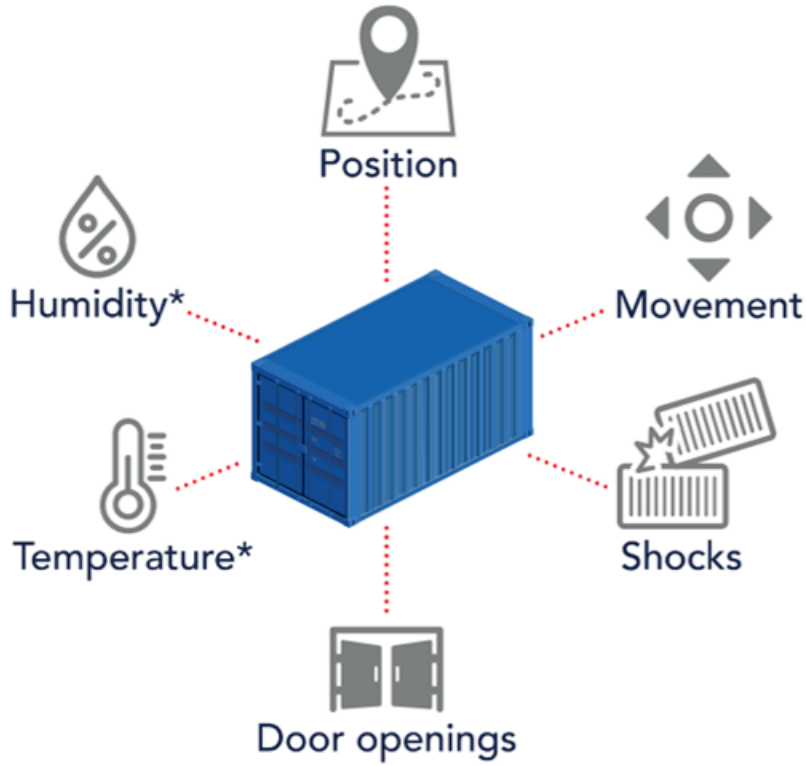
IOT and Maritime Industry

- Ships equipped with sensors.
- Provide tangible information and decision support tools.



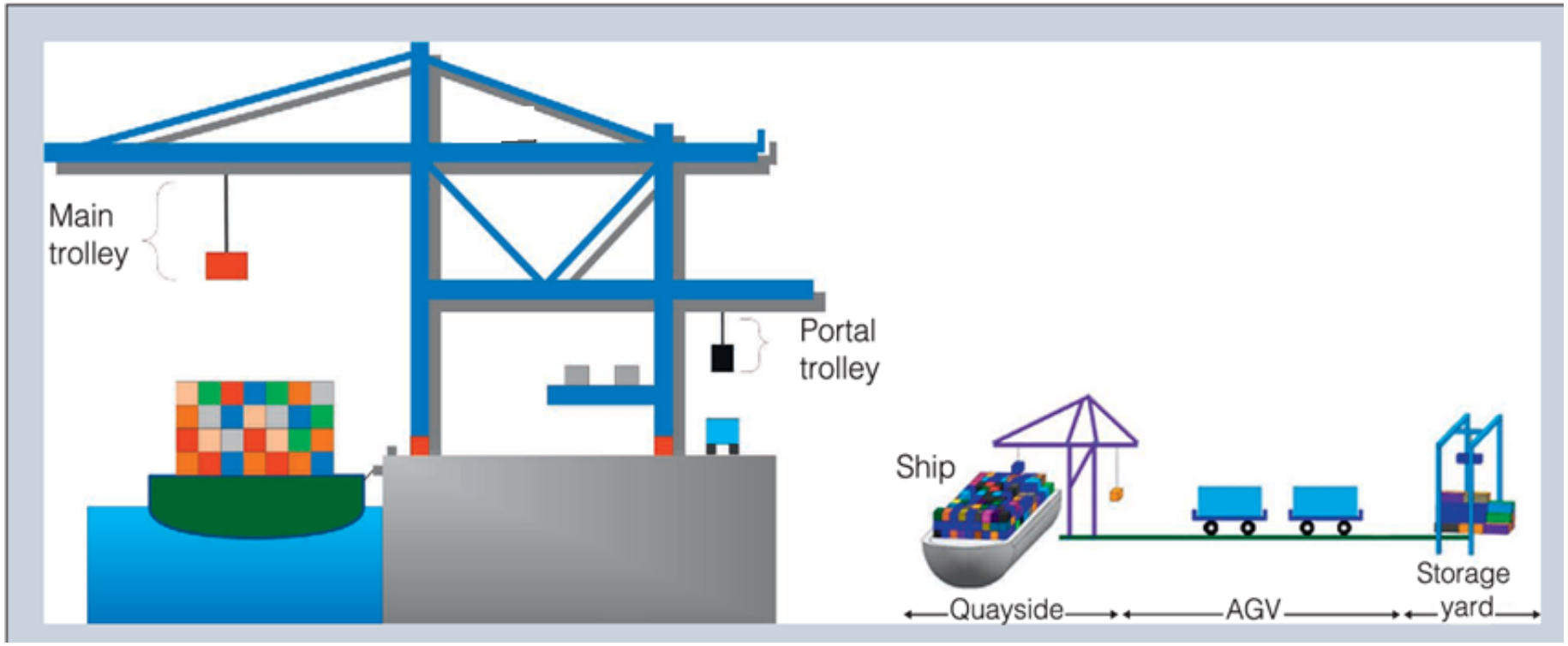
IOT Applications in Ports

Smart storage systems can detect the needs of the cargo and adjust factors to increase product quality and decrease cargo damage.



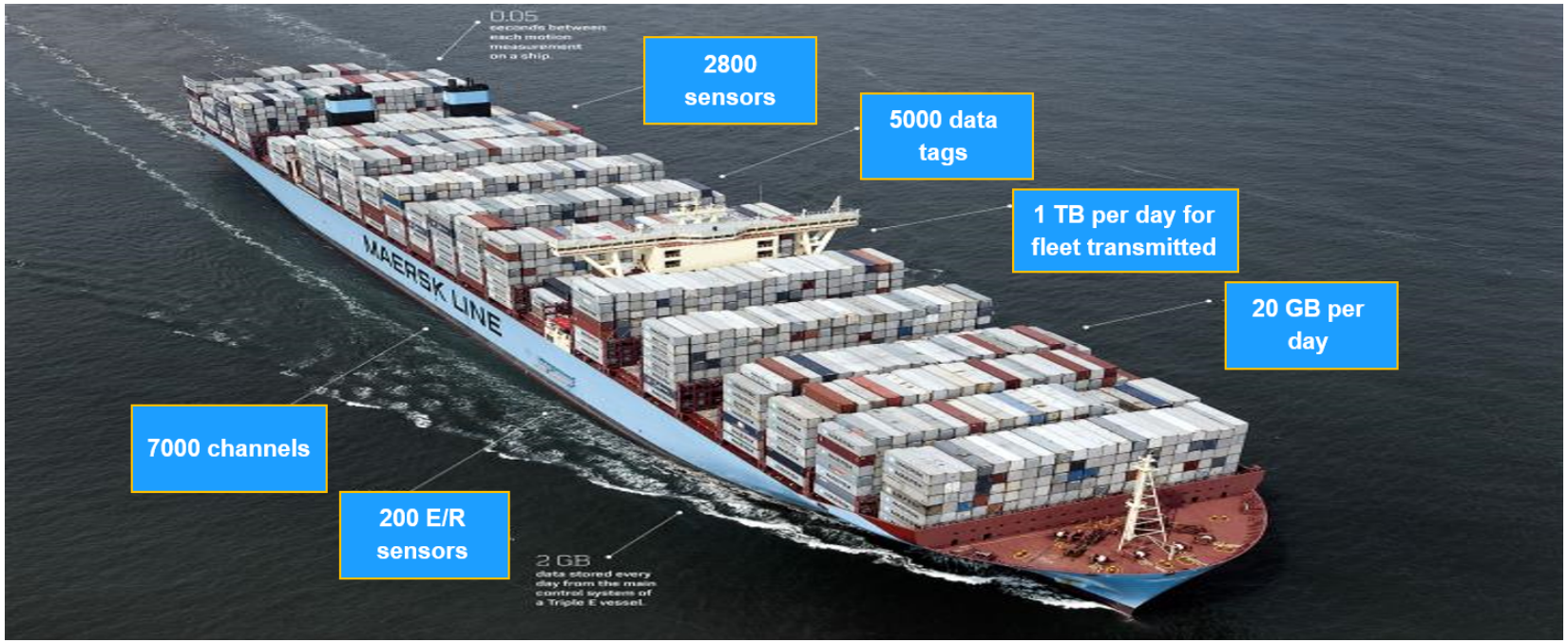
IOT Applications in Ports (cont.)

Sensors monitoring



Big Data in Maritime Industry

Forecasting & Planning

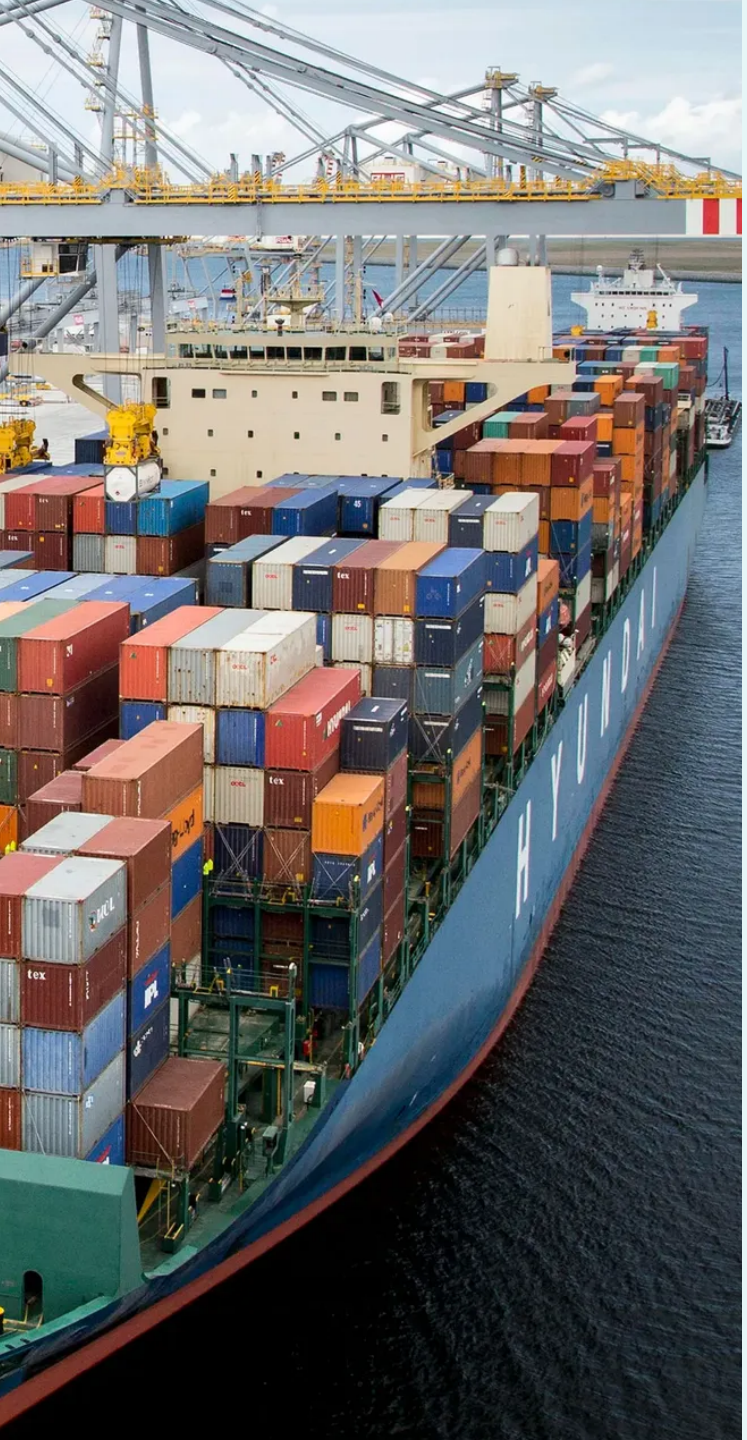


Port Community System

- Port Community Systems are a form of Single Window for Trade.
- A Port Community System optimizes, manages and automates logistics-efficient processes.

Problem Statement

- Digitalization can help port authorities comply with industry regulations and standards.
- It can also protect against data loss and tampering.
- Implementing digital technologies can be a challenge for port authorities, as it may require new infrastructure and integration with existing systems.
- The benefits of digitalization for ports include increased efficiency, reduced manual labor, and improved accuracy and resource management.



Challenges

Data Integration

The data must be collected from multiple stakeholders.

Security

system must be able to protect sensitive data from unauthorized access.

Cost

cost of the hardware and software needed to run the system must be taken into account.

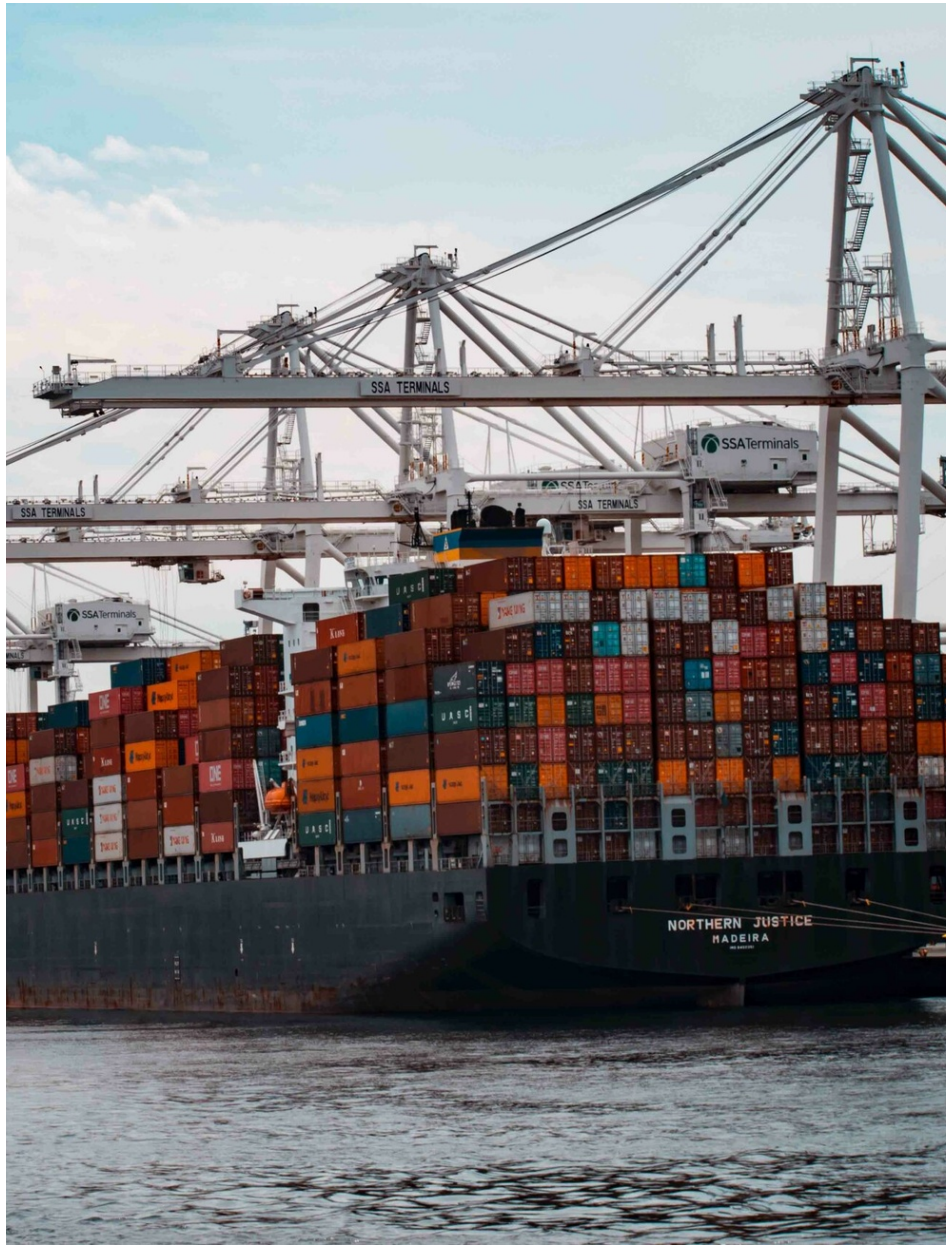
Compliance

- Port community systems must comply with local and international regulations and standards.

- the regulations and standards vary from country to country.

Scalability

The system must be able to handle increased volumes of data and transactions



Objectives.

- Create a digital system for Alexandria port Authority.
- Create a web App for Alexandria port Authority.

Services that will be introduced

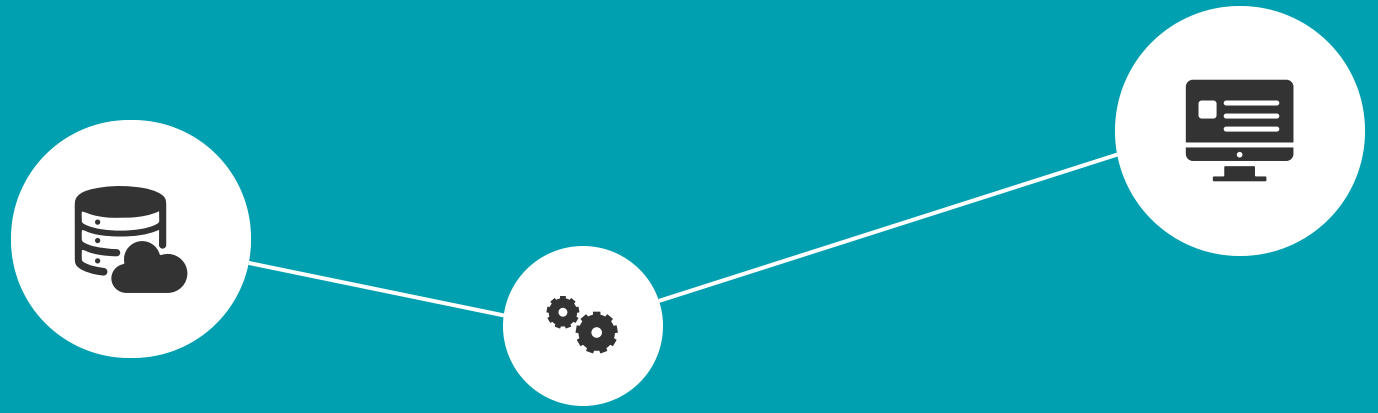
Centralized Database

Ship Movement Monitor

Information Sharing

System Platform Management

Dashboard



System Architecture Figure including phases & relations among phases

Architecture

Database

MySQL DB

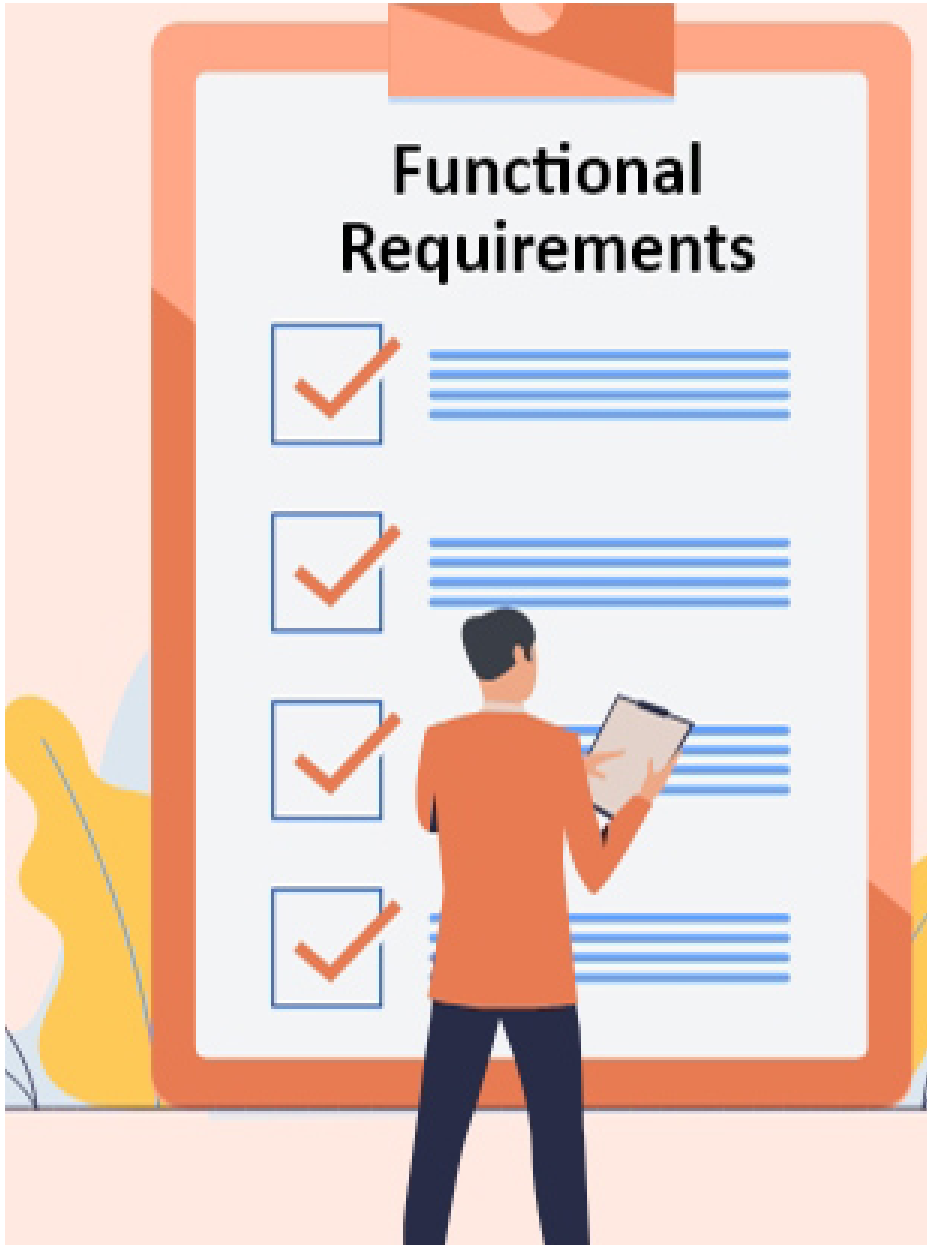
Back-end

node.js

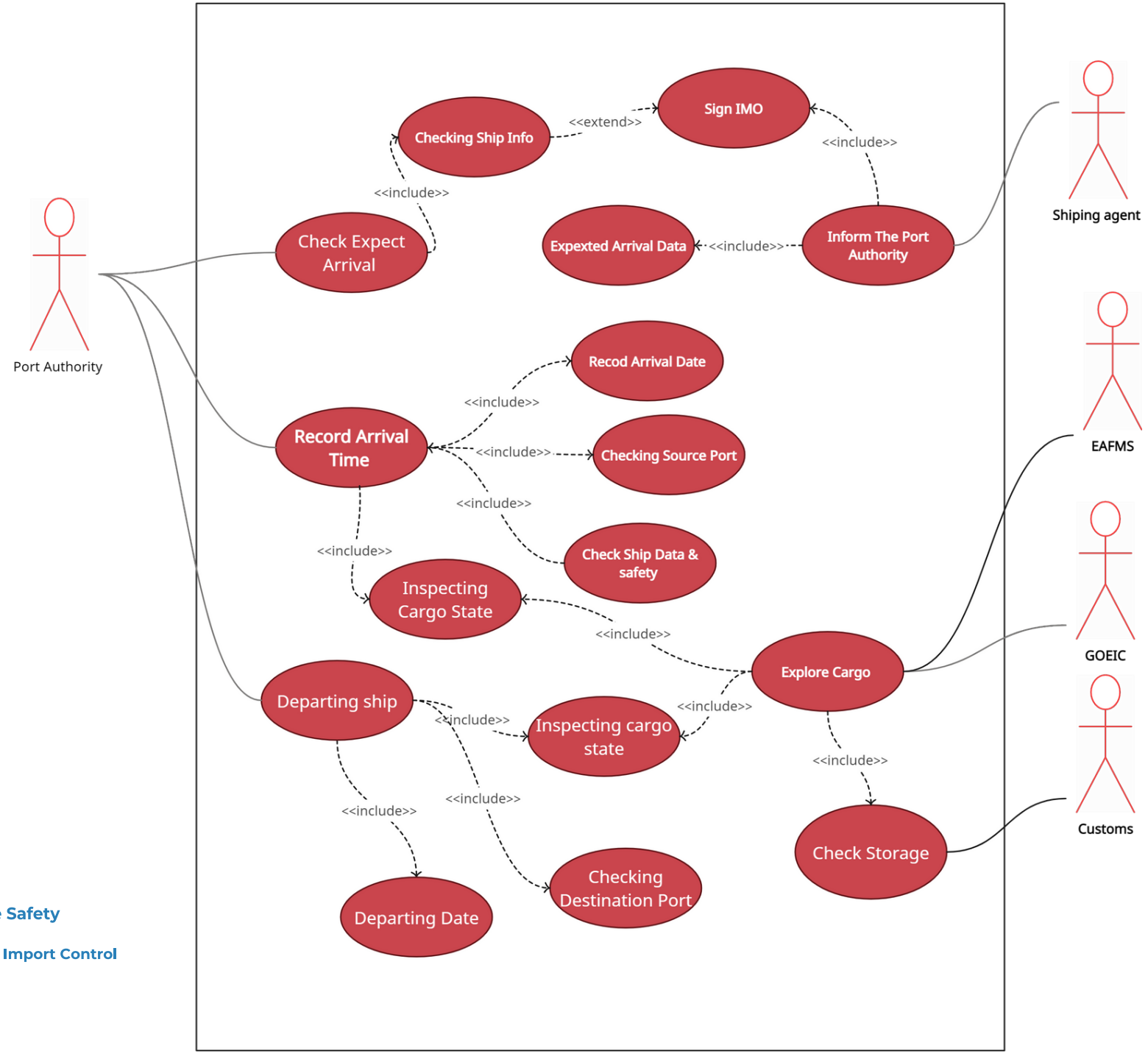
Front-end

React

- **Port Authority**
  - ✔ Ship Monitoring.
  - ✔ Arrival Information.
  - ✔ Static Data Retrieval.
  - ✔ Arrival Data Recording.
  - ✔ Departure Data Recording.
  - ✔ Administrator Privileges.
  - ✔ Database Update Rollback.
- **Customs System**
- **GOEIC**
- **EAMS**



Use Case

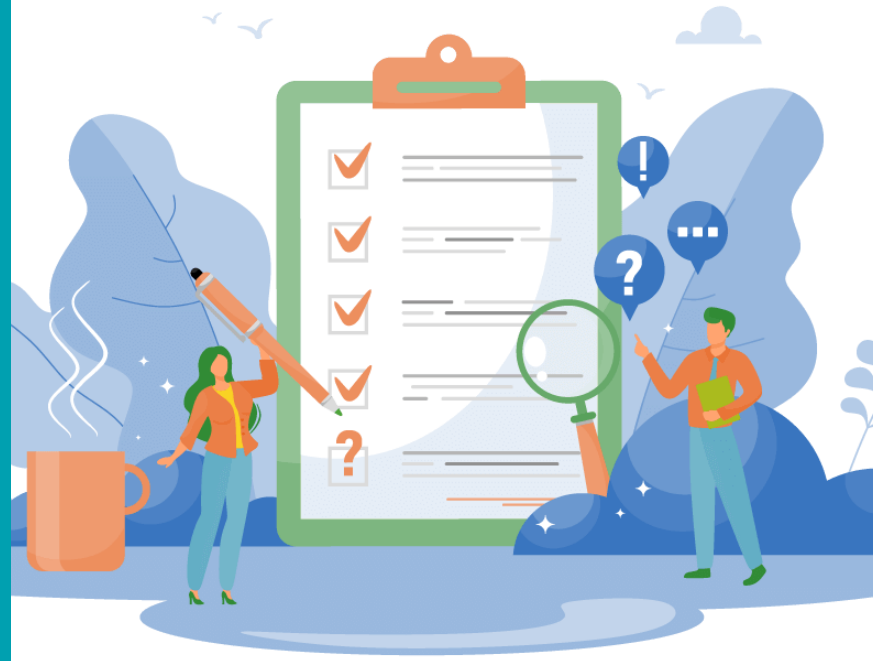


EAFMS: Egyptian Authority For Maritime Safety

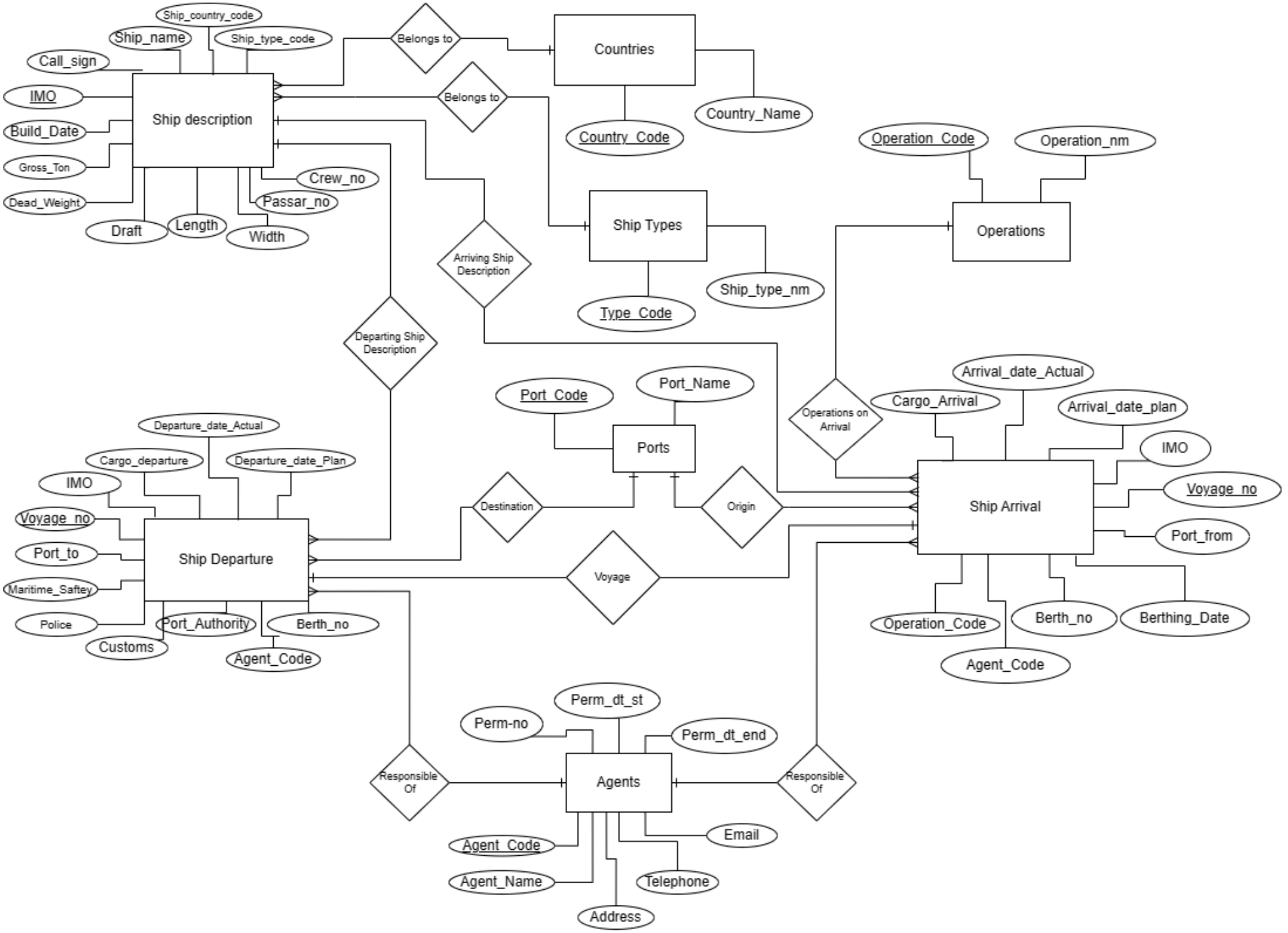
GOEIC: General Organization For Export and Import Control

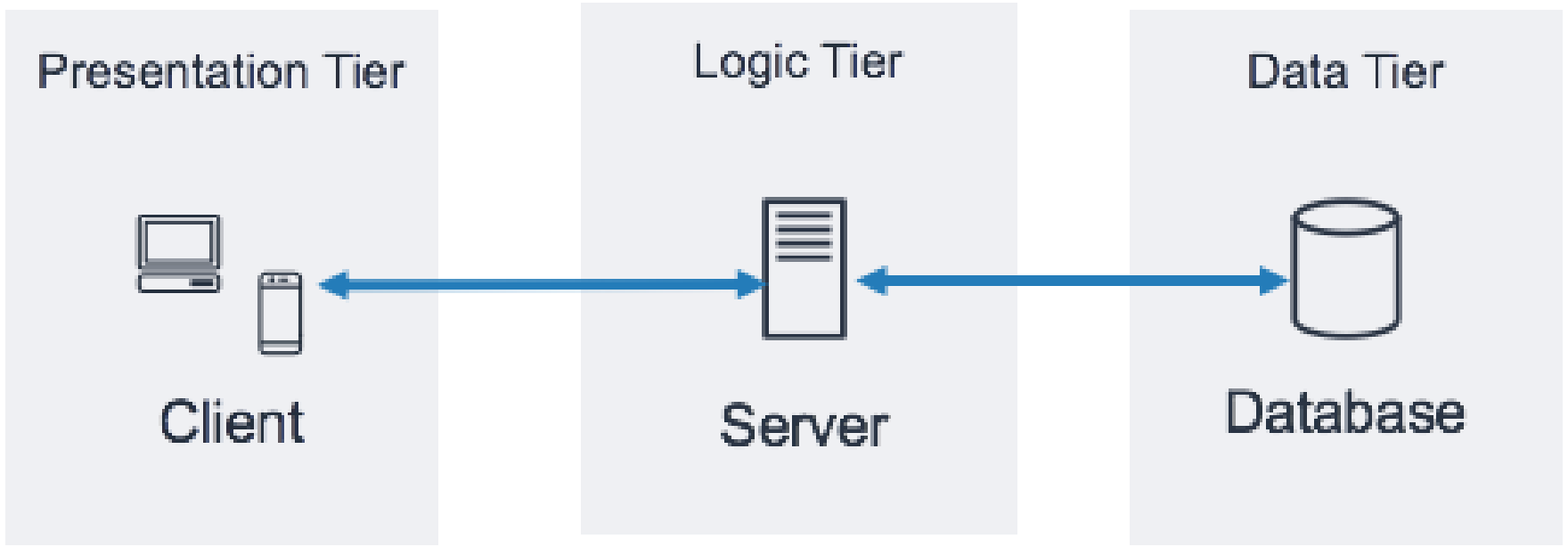
Relations between data

- Each Country can have multiple ships with different description.
- Each Ship Type can have multiple ship description.
- 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.



ERD

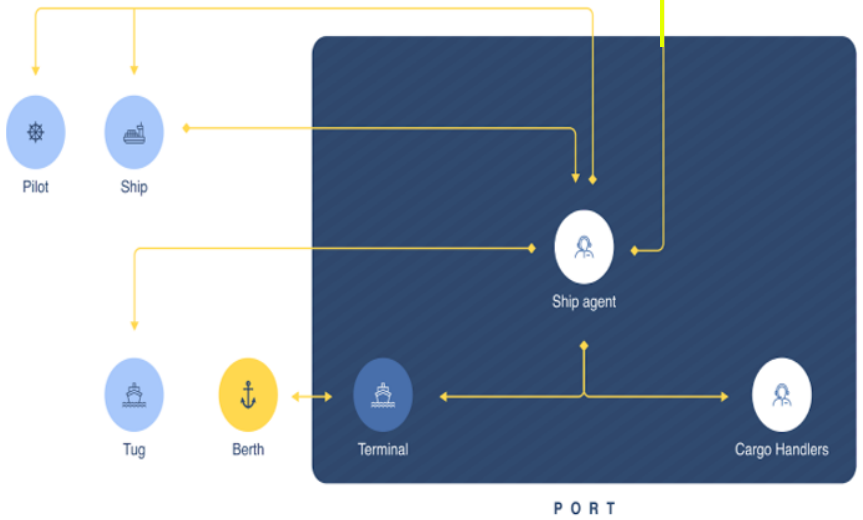

Ship_country_code
Ship_name
Ship_type_code
Call_sign
IMO
Build_Date
Gross_Ton
Dead_Weight
Draft
Length
Width
Crew_no
Passar_no
Ship description
Belongs to
Belongs to
Countries
Country_Code
Country_Name
Ship Types
Type_Code
Ship_type_nm
Arriving Ship Description
Departing Ship Description
Operation_Code
Operation_nm
Operations
Operations on Arrival
Arrival_date_Actual
Cargo_Arrival
Arrival_date_plan
IMO
Voyage_no
Port_from
Ship Arrival
Operation_Code
Berth_no
Berthing_Date
Agent_Code
Port_Code
Port_Name
Ports
Destination
Origin
Departure_date_Actual
Cargo_departure
Departure_date_Plan
IMO
Voyage_no
Port_to
Maritime_Saftey
Police
Customs
Port_Authority
Berth_no
Agent_Code
Ship Departure
Voyage
Perm_dt_st
Perm-no
Perm_dt_end
Responsible Of
Agents
Responsible Of
Email
Agent_Code
Agent_Name
Telephone
Address


Presentation Tier
Logic Tier
Data Tier
Client
Server
Database

Phase I
System Analysis

Phase II
Creating Database

Phase III
Back-end

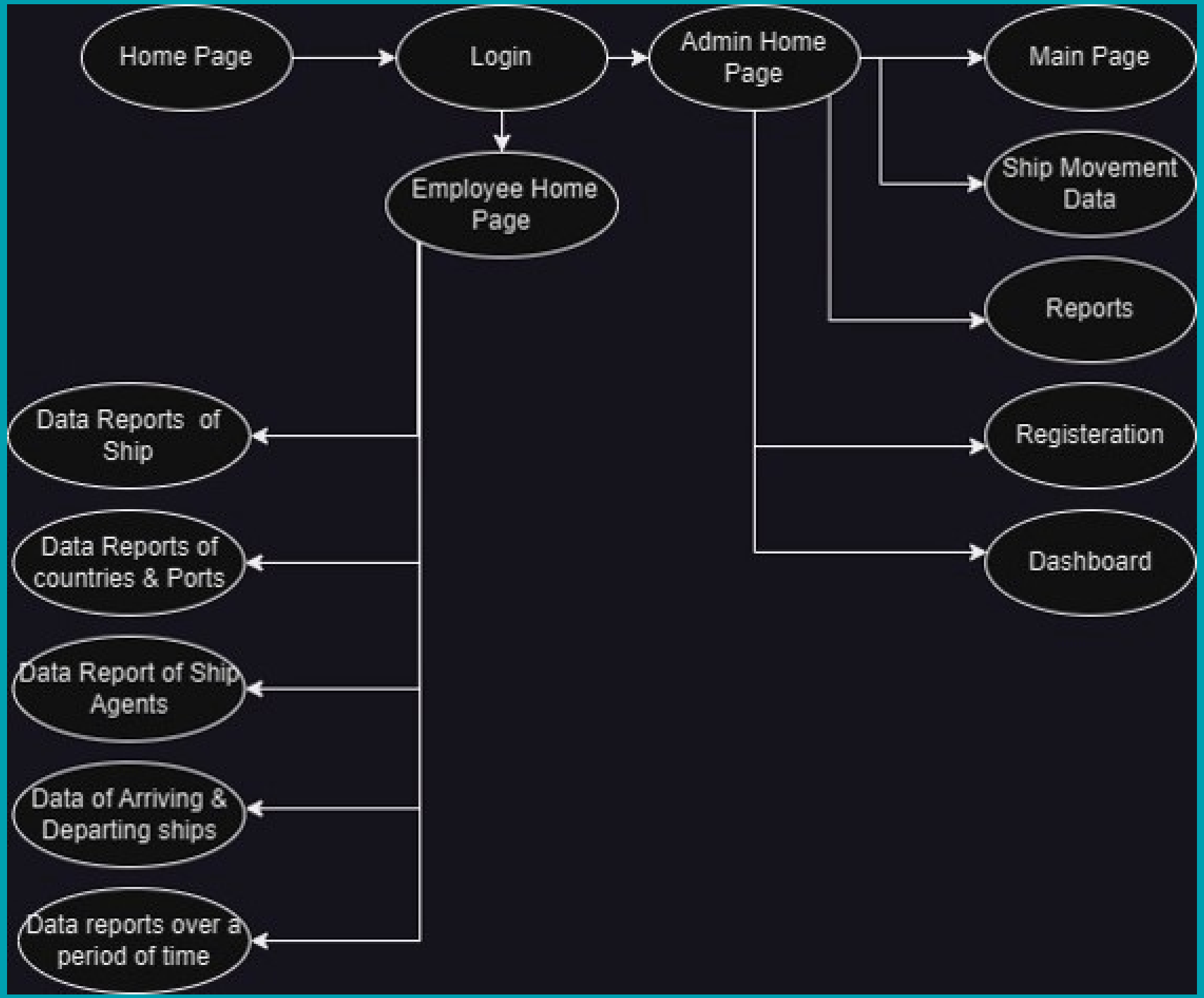


Response

Request

Phase IV
Front-end

Phase V
Testing


Home Page
Login
Admin Home Page
Main Page
Ship Movement Data
Reports
Registeration
Dashboard
Employee Home Page
Data Reports of Ship
Data Reports of countries & Ports
Data Report of Ship Agents
Data of Arriving & Departing ships
Data reports over a period of time

The technical description of each phase

The technical description of each phase

The Port management system is composed of **4 phases.**

1. Data **Gathering**
   - Data Collection
   - Data preprocessing
2. **Database** creation
   - Table Creation
   - Data Insertion
   - Running MySQL Server
3. **Back-end** Implementation
   - Technology selection
   - file structure
   - Database connection
   - Create APIs
   - Test APIs
   - Deploy server on cloud
4. **Front-end** Implementation

Phase I: Data Gathering

Data Collection

1. Manual Data Collection
2. Automated Data Download
3. Select Data Types

Data Preprocessing

- Python script hosted on a server downloaded 110 files of arriving and departing data.
- Each file contained 5 rows, resulting in a total of 550 rows of data.
- Python script merged the 110 files while removing redundancies.
- Data was filtered to remove rows without counterparts in departing or arriving data.

Data Preprocessing (cont.)

- After filtration, the data was reduced to 67 rows.
- Python scripts were created to download country names and codes.
- Country codes included 248 countries.
- Python scripts were also created to download port names and codes.
- There were a total of 44,892 ports with their respective codes.

Phase II: Database Creation

1. Select the most suitable technology for building the database.
2. Install and configure the chosen database technology on a server.
3. Enable team members to interact with the database using tools like SQL Workbench.
4. Create necessary tables within the database to organize and store data.
5. Tables serve as the structure for data organization, integrity, and efficient querying.

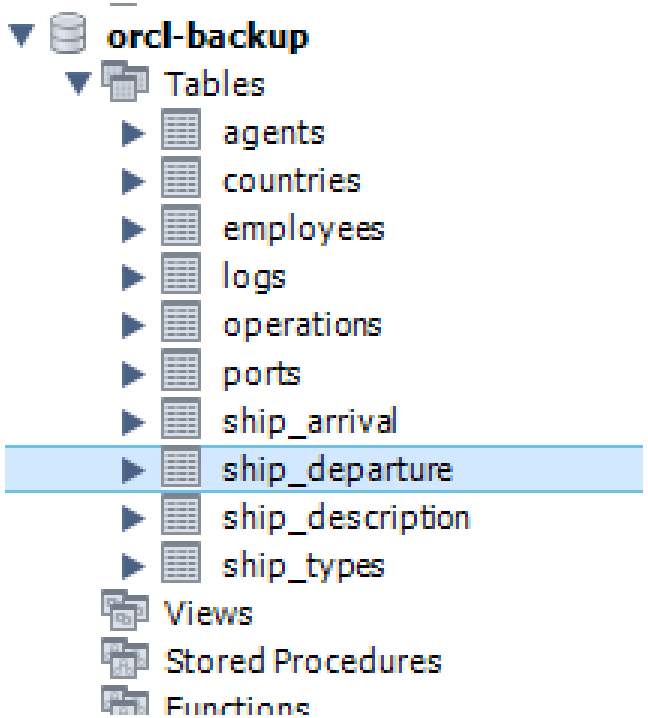
Database Creation (cont.)

Beautiful.ai makes it easy to find new ways to present your content, visually.

The first step is to identify entities in the ERD and create corresponding tables in the SQL database.

Each entity in the ERD becomes a table in SQL, and each attribute becomes a column in the table.

The primary key for each table is typically derived from the primary key attribute in the ERD.




```sql
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;
```

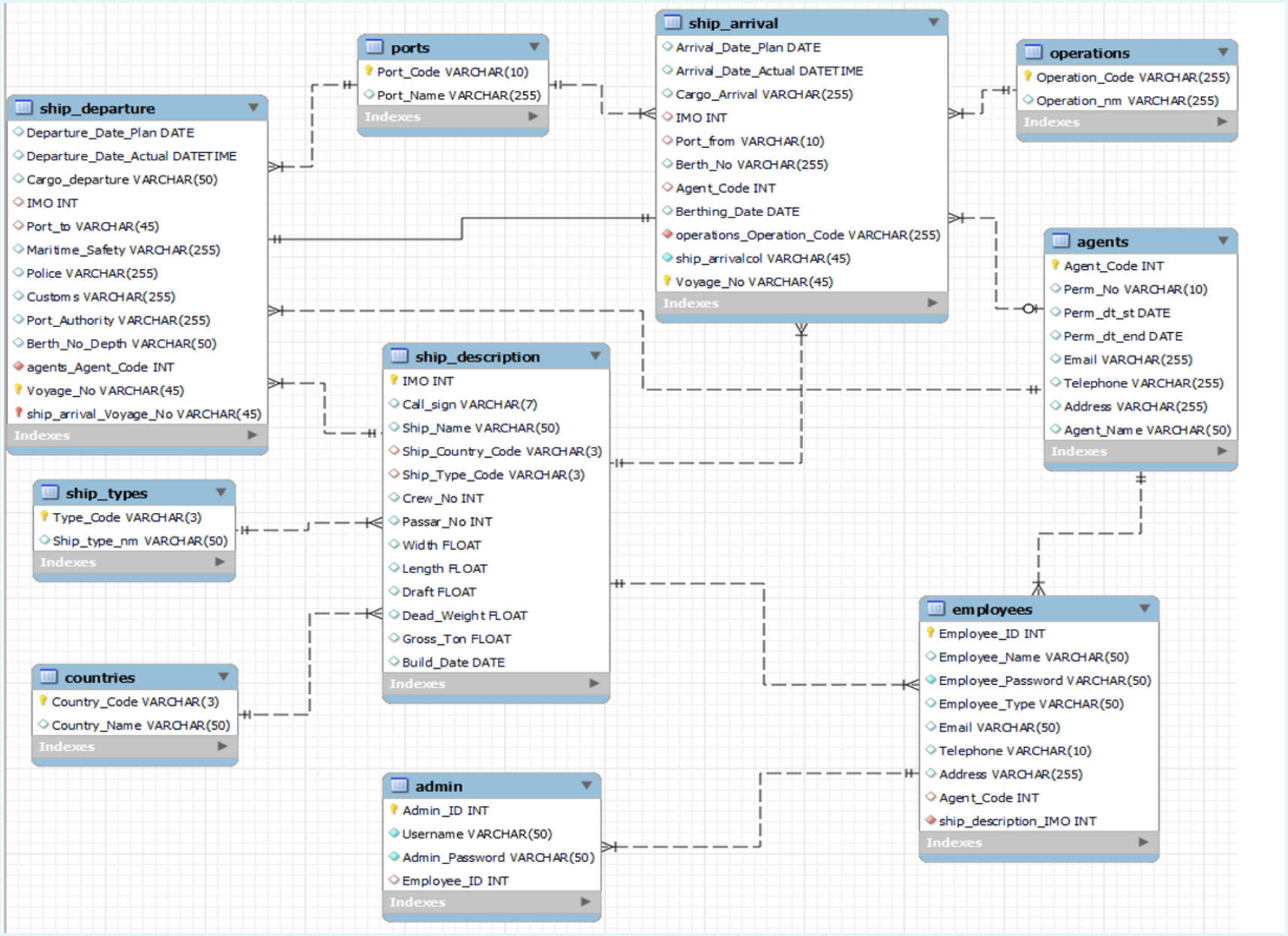
Data Insertion

- Insert the previously collected data into the newly created tables.
- Data inserted automated using python scripts.
- Map the collected data to appropriate table columns for accurate representation.
- Ensure data integrity during the insertion process.
- Establish a robust database infrastructure with required tables.
- Populated data sets the foundation for future development and optimization.

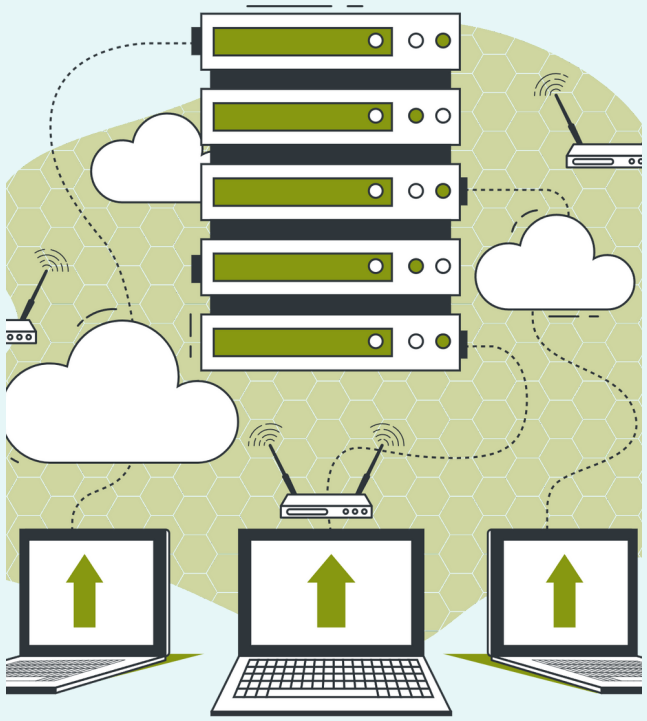
```
db = mysql.connector.connect(
    host="localhost",
    user="admin",
    passwd="admin",
    database="orcl"
)



mycr = db.cursor()
for i in range(size):
    name = str(Name_values[i]).replace("'", "''")
    mycr.execute(f"INSERT INTO countries(`Country_Code`,`Country_Name`) VALUES ('{Code_values[i]}','{name}');")


db.commit()
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Arrival_ID	Voyage_No	IMO	Arrival_Date_Plan	Arrival_Date_Actual	Port_of_Departure	Agent_Code	Berth_No	Berthing_Date	Cargo_Arrival	Op_Code
1	3752/2022	9586710	NULL	2022-12-28 11:45:00	USEMR	NULL	17	2022-12-27	NULL	NULL
2	3821/2022	8517580	NULL	2022-12-15 14:00:00	CHFLY	2	17	2022-12-16	NULL	PAX
3	3893/2022	9161845	NULL	2022-12-20 10:15:00	USRWO	3	13	2022-12-20	NULL	POL
4	3798/2022	9138393	NULL	2022-12-24 10:00:00	AUBLS	4	12	2022-12-24	NULL	POL
5	3631/2022	9718131	2023-07-01	2023-07-02 00:00:00	IDBOA	7	222	2023-06-30	2023-06-29	POL
6	3875/2022	8937352	NULL	2022-12-19 00:30:00	AUSNB	6	19	2022-12-21	NULL	ULD
7	3886/2022	9356969	NULL	2022-12-24 11:10:00	BRJUN	7	17	2022-12-24	NULL	LVU
8	3923/2022	8918344	NULL	2022-12-26 11:00:00	FRMCO	8	18	2022-12-28	NULL	LUD
9	3952/2022	9578971	NULL	2022-12-28 22:15:00	USENT	9	16	2022-12-28	NULL	BUC
10	3865/2022	9615327	NULL	2022-12-23 14:00:00	DEMMN	10	16	2022-12-30	NULL	LUD
12	3962/2022	9148738	NULL	2022-12-29 08:10:00	USQBL	7	17	2022-12-30	NULL	BUC
13	3922/2022	8908507	NULL	2022-12-29 10:35:00	SESSL	11	11	2022-12-31	NULL	LVD
14	3591/2022	9617466	NULL	2022-11-20 00:30:00	KROKP	12	12	2023-01-01	NULL	LUD
15	3914/2022	9558995	NULL	2022-12-23 12:50:00	DEBMH	5	16	2022-12-23	NULL	ULD
16	3959/2022	9338735	NULL	2022-12-25 09:30:00	NONRS	13	15	2022-12-25	NULL	LUD
17	3987/2022	9516246	NULL	2022-12-29 10:00:00	CACOY	14	14	2023-01-02	NULL	LVD
18	3987/2022	9516246	NULL	2022-12-29 10:00:00	BEKPN	3	11	2023-01-06	NULL	PAX
19	4023/2022	8806151	NULL	2023-01-04 23:45:00	USWPD	3	10	2023-01-05	NULL	SUP
20	35/2023	9555333	NULL	2023-01-08 18:10:00	JPMKK	14	15	2023-01-09	NULL	POL
21	1052/2022	9880958	NULL	2022-04-01 12:30:00	PABAG	15	11	2022-12-29	NULL	SUP
22	1052/2022	9880958	NULL	2022-04-01 12:30:00	PHLIM	15	12	2022-12-12	NULL	SUP
23	4017/2022	9139115	NULL	2023-01-01 14:50:00	DEVDR	3	11	2023-01-09	NULL	ULD


ports
Port_Code VARCHAR(10)
Port_Name VARCHAR(255)
Indexes
ship_arrival
Arrival_Date_Plan DATE
Arrival_Date_Actual DATETIME
Cargo_Arrival VARCHAR(255)
IMO INT
Port_from VARCHAR(10)
Berth_No VARCHAR(255)
Agent_Code INT
Berthing_Date DATE
operations_Operation_Code VARCHAR(255)
ship_arrivalcol VARCHAR(45)
Voyage_No VARCHAR(45)
Indexes
operations
Operation_Code VARCHAR(255)
Operation_nm VARCHAR(255)
Indexes
ship_departure
Departure_Date_Plan DATE
Departure_Date_Actual DATETIME
Cargo_departure VARCHAR(50)
IMO INT
Port_to VARCHAR(45)
Maritime_Safety VARCHAR(255)
Police VARCHAR(255)
Customs VARCHAR(255)
Port_Authority VARCHAR(255)
Berth_No_Depth VARCHAR(50)
agents_Agent_Code INT
Voyage_No VARCHAR(45)
ship_arrival_Voyage_No VARCHAR(45)
Indexes
agents
Agent_Code INT
Perm_No VARCHAR(10)
Perm_dt_st DATE
Perm_dt_end DATE
Email VARCHAR(255)
Telephone VARCHAR(255)
Address VARCHAR(255)
Agent_Name VARCHAR(50)
Indexes
ship_description
IMO INT
Call_sign VARCHAR(7)
Ship_Name VARCHAR(50)
Ship_Country_Code VARCHAR(3)
Ship_Type_Code VARCHAR(3)
Crew_No INT
Passar_No INT
Width FLOAT
Length FLOAT
Draft FLOAT
Dead_Weight FLOAT
Gross_Ton FLOAT
Build_Date DATE
Indexes
ship_types
Type_Code VARCHAR(3)
Ship_type_nm VARCHAR(50)
Indexes
countries
Country_Code VARCHAR(3)
Country_Name VARCHAR(50)
Indexes
employees
Employee_ID INT
Employee_Name VARCHAR(50)
Employee_Password VARCHAR(50)
Employee_Type VARCHAR(50)
Email VARCHAR(50)
Telephone VARCHAR(10)
Address VARCHAR(255)
Agent_Code INT
ship_description_IMO INT
Indexes
admin
Admin_ID INT
Username VARCHAR(50)
Admin_Password VARCHAR(50)
Employee_ID INT
Indexes

Running MySQL Server **Locally** vs. Running MySQL Server **Online**



Running MySQL Server Locally	Running MySQL Server Online
Initially set up a database server on a local computer using Oracle MySQL.	Transitioned to the cloud-based platform, DigitalOcean, due to initial difficulties.
Required router changes, port opening, and port forwarding.	Facilitated by team member Ahmed Ashraf, with experience in DigitalOcean.
Faced challenges with changing IP addresses and setup issues.	Improved workflow with 24/7 accessibility to the server from anywhere.
The hosting computer had to be constantly powered on, causing resource consumption.	Minimal downtime experienced, resolving IP address and resource consumption issues.
Maintaining smooth operation demanded significant effort.	Moving to the cloud provided an efficient and reliable solution.
Realized the need for a more reliable hosting strategy.	

Phase III
Back-end Implementation

TECHNOLOGY SELECTION:
NODE.JS

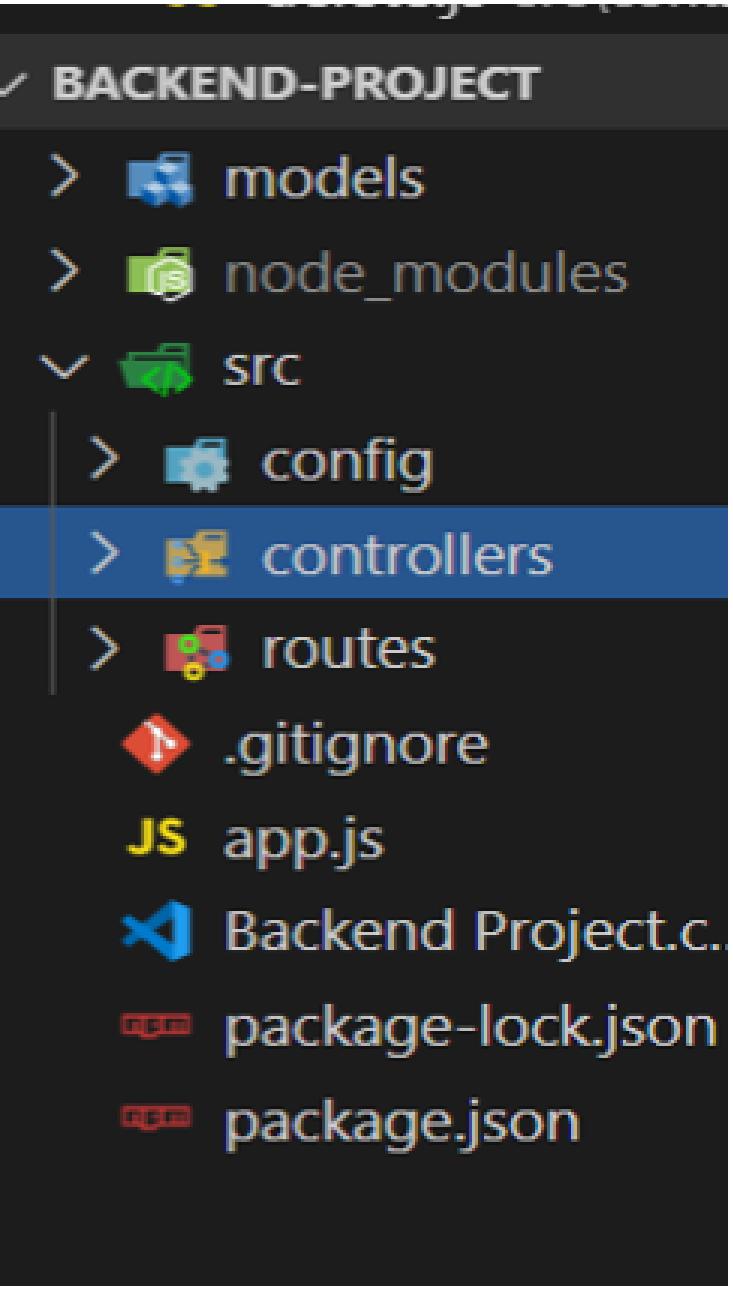
Next phase involves using Node.js to create APIs that act as the interface between the user interface and the database.

Node.js is a popular runtime environment for executing JavaScript on the server side.

Express.js, a web framework, can be used to create the APIs.

Express.js provides an intuitive way to handle HTTP requests and define routes.

Node.js is suitable for building server-side applications.

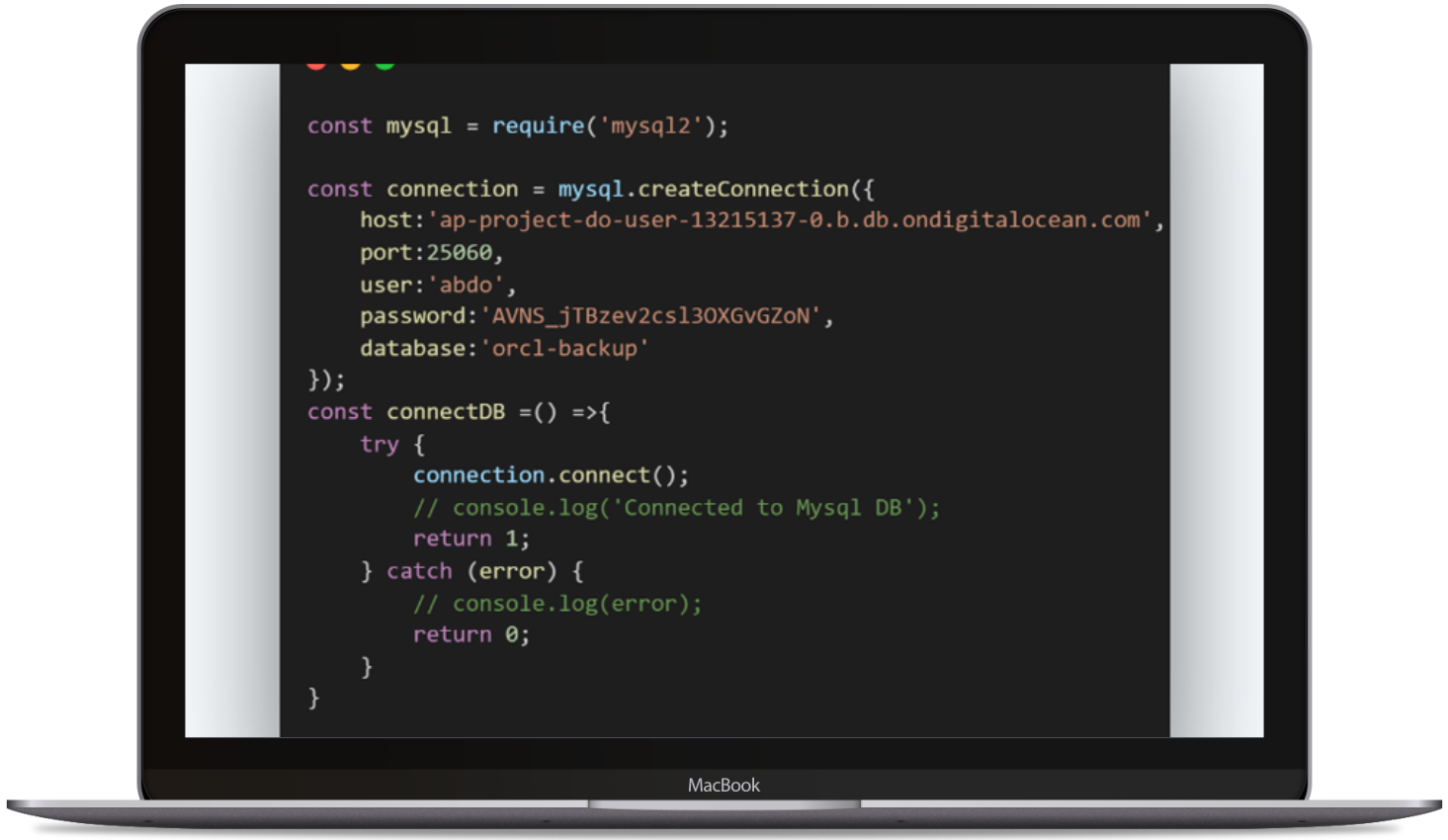


File Structure.

- The backend of the ship port system follows the **Model-View-Controller** (MVC) architectural pattern.
- **MVC** helps in organizing the codebase and separating concerns into different components.
- The **file structure** includes:
  1. App.js: Main entry point that sets up the server, middleware, and initializes components.
  2. Config: Contains configuration files, including the database connection configuration.
  3. Controllers: Holds controller files responsible for handling logic for API routes or endpoints.
  4. Routes: Contains route files that define API endpoints and map them to appropriate controller methods.
- This structure promotes **separation of concerns**, **code organization**, and **scalability.**
- Provides a **modular approach**, enabling easy addition of new routes and controllers as the application grows.
- Enhances code **reusability** and **maintainability**.

Database Connection

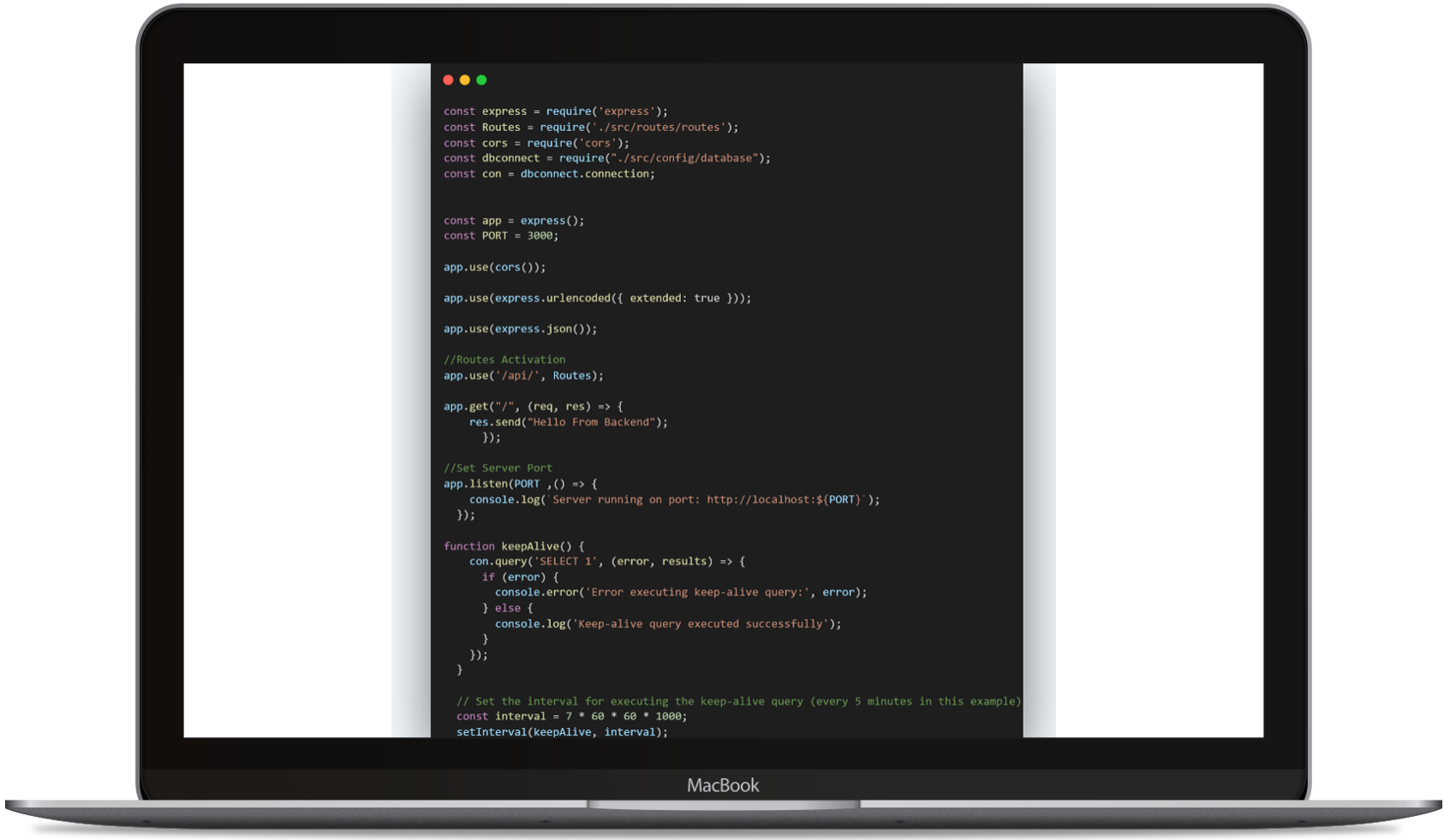
Database Connection



1. Create the database.js file inside the config folder:
2. Usage in other files to connect to the database:

App.js

- Dependencies
- Initializing the Server
- Cross-Origin Resource Sharing (CORS)
- Routing
- Handling the Root Route
- Starting the Server



```
router.post('/post/newshipdesc', (req, res) => postctrl.addshipdesc(req, res));
router.post('/post/newagent', (req, res) => postctrl.addAgent(req, res));
router.post('/post/login', (req, res) => postctrl.login(req, r
router.post('/post/logs', (req, res) => postctrl.addlogs(req,

router.put('/update/updateport', (req, res) => putctrl.updateP
router.put('/update/updatecountry', (req, res) => putctrl.upda
router.put('/update/updateshiptype', (req, res) => putctrl.upd
router.put('/update/updateop', (req, res) => putctrl.updateOpe
router.put('/update/updatedepart', (req, res) => putctrl.updat
router.put('/update/updatearrival', (req, res) => putctrl.upda
router.put('/update/updateShipDesc', (req, res) => putctrl.upd
router.put('/update/updateAgent', (req, res) => putctrl.update

router.get('/login/emp', (req, res) => logCtrl.fetchemployee(r
router.get('/login/fetchemployee', (req, res) => logCtrl.fetch
router.post('/login/emp', (req, res) => logCtrl.loginEmp(req,
router.post('/login/admin', (req, res) => logCtrl.loginAdmin(r
router.post('/login/addEmp', (req, res) => logCtrl.addEmp(req,
router.post('/login/addArrival', (req, res) => logCtrl.addArri
router.post('/login/addDepart', (req, res) => logCtrl.addDepar
router.post('/login/register', (req, res) => logCtrl.register(

router.delete('/delete/deleteEmp/:id', (req, res) => deleteCtr
router.delete('/delete/deleteport', (req, res) => deleteCtrl.deletePort(req, res));
router.delete('/delete/deletecountry', (req, res) => deleteCtrl.deleteCountry(req, res));
router.delete('/delete/deleteshiptype', (req, res) => deleteCtrl.deleteShipType(req, res));
router.delete('/delete/deleteop', (req, res) => deleteCtrl.deleteOperation(req, res));
```

You, 4 days ago • format app

Route file

- The app.js file sets up the Express server and defines the base URL as /api/.
- The routes.js file contains the following API routes and their corresponding controllers:
- GET Routes:
- `router.get('/get/example', (req, res) => ctrl.example(req, res));`
- POST Routes:
- `router.post('/post/example ', (req, res) => postctrl.example(req, res));`

PUT Routes:

- `router.put('/update/example ', (req, res) => putctrl.example(req, res));`
- DELETE Routes:

`router.delete('/delete/example ', (req, res) => deleteCtrl.example(req, res));`

Get APIs

- Create a controllers.js file (get.js)
- Insert in a routes.js file
- router.get('/get/example', (req, res)=> ctrl.example(req, res));

```
const example = async (req, res) => {
  try {
    const name = req.query.name; // Assuming the name is provided as a query parameter
    const sql = `SELECT Port_Code FROM ports WHERE Port_Name='${name}';`;
    con.query(sql, (err, result) => {
      if (err) {
        res.status(500).json({ error: 'An error occurred' });
      } else {
        if (result[0] === undefined) {
          res.json(null);
        } else {
          res.json(result[0].Port_Code);
        }
      }
    });
  } catch (error) {
    res.status(500).json({ error: 'An error occurred' });
  }
};
```

```
const addnewshiptype = async (req, res) => {
  try {
    const resobj = req.body;

    const sql = `INSERT INTO ship_types (Ship_type_nm,Type_Code)
                 VALUES ('${resobj.name}','${resobj.code}');`;
    con.query(sql, (err, result) => {
      if (err) {
        return res.status(400).json({ message: err.sqlMessage, query: false });
      } else {
        return res.status(200).json({
            message: `Ship Type  ${resobj.name} added Correctly`,
            query: true,
          });
      }
    });
  } catch (err) {
    res.status(400).send(err);
  }
```

Post APIs

- Create a controllers.js file (post.js)
- Insert in a routes.js file
- router.post('/post/example ', (req, res) => postctrl.example(req, res));

```
const deleteEmployee = async (req, res) => {
  const { id } = req.params;
  const sql = "DELETE FROM employees WHERE Employee_ID = ?";
  con.query(sql, [id], function (err, result) {
    if (err) {
      res.status(500).send({ message:`Internal server error${err}`});
      return;
    }
    res.send({ message: "Employee deleted successfully", query: true });
  });
};
```

Delete APIs

- Create a controllers.js file (delete.js)
- Insert in a routes.js file
- router.delete('/delete/example ', (req, res) => deleteCtrl.example(req, res));

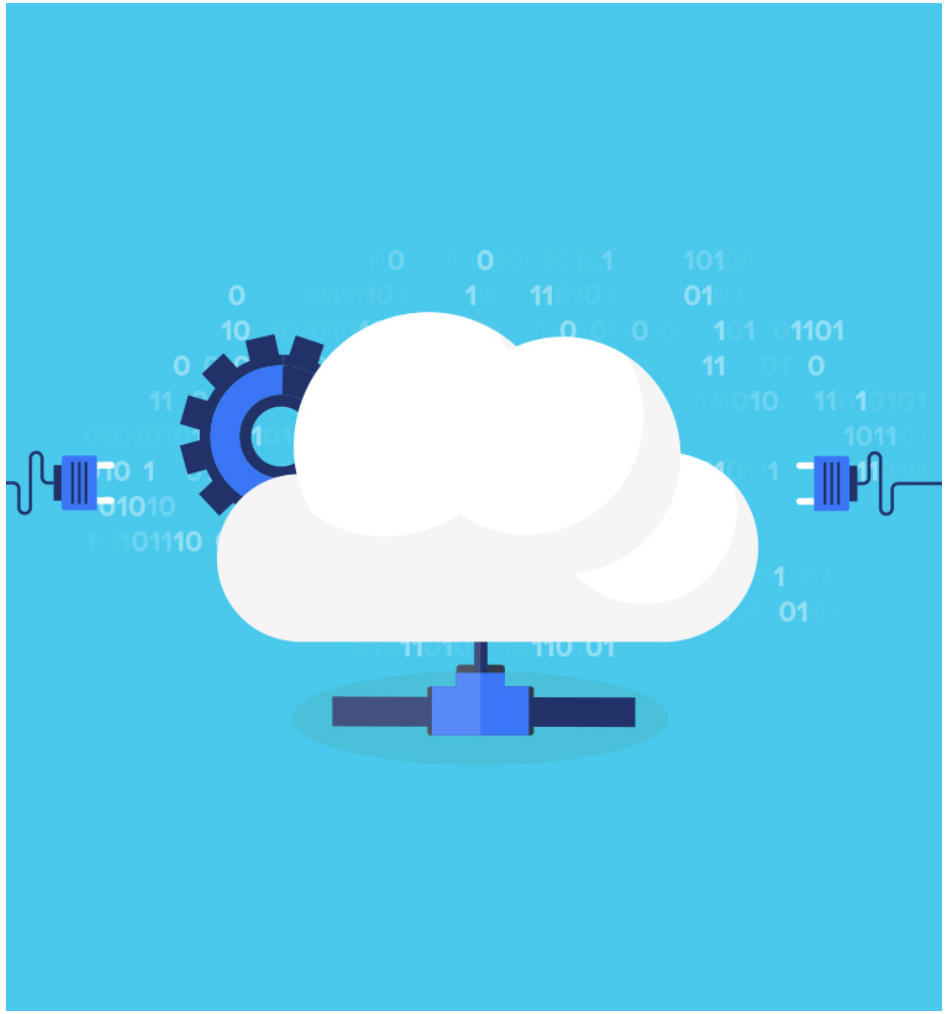
Put APIs

- Create a controllers.js file (put.js)
- Insert in a routes.js file
- router.put('/update/example ', (req, res) => putctrl.example(req, res));

```
const updateShipType = async (req, res) => {
   try {
     const resobj = req.body;

     const sql = `UPDATE ship_types SET Ship_type_nm = '${resobj.name}'
                    WHERE Type_Code = '${resobj.code}';`;
     con.query(sql, (err, result) => {
       if (err) {
         throw { message: "Error updating the entry", status: false };
       } else {
         res.status(200)
           .send({ message: `Ship Type ${resobj.name} updated`, query: true });
       }
     });
   } catch (error) {
     res.status(400).send(error);
   }
 };
```

Deploy server on cloud

- Used A Linux server hosted on Digital Ocean to run the server.
- Setup the server to accept requests only from specific IP address.
- Ran the Backend Server from that server.
- Create a script file that automatically fetches updates from the GitHub repo.



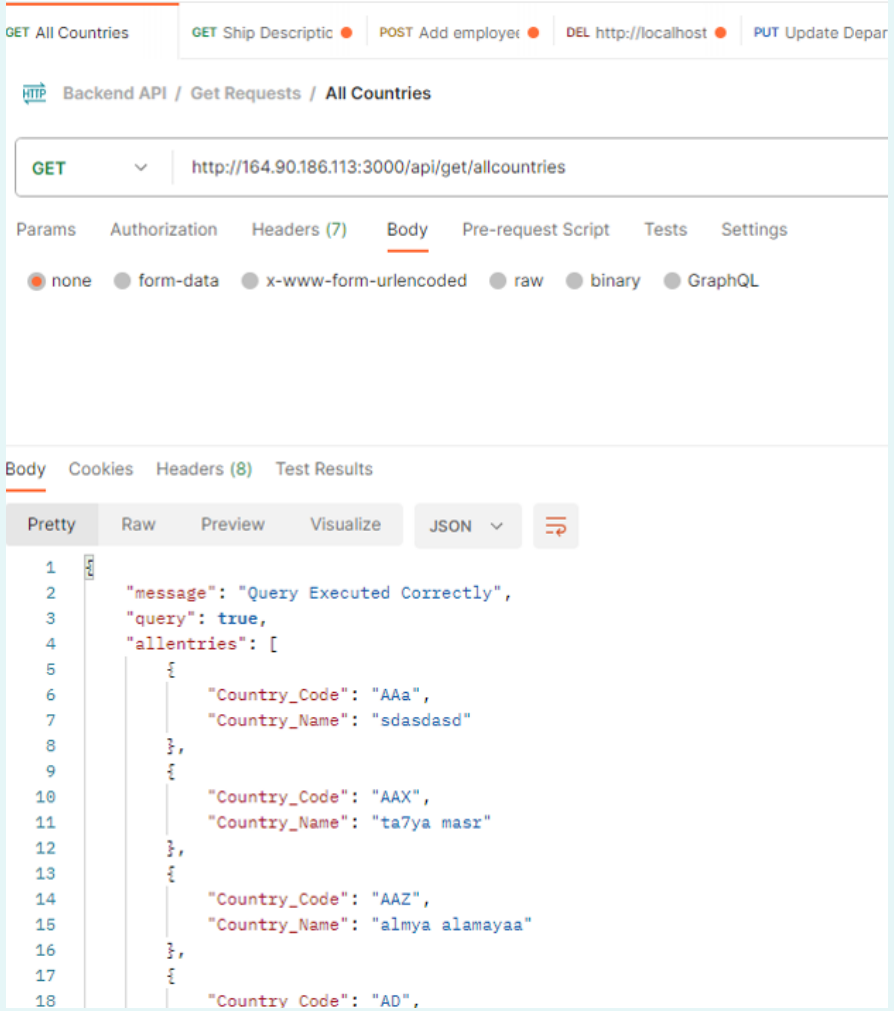
POSTMAN

1 Installation and Setup

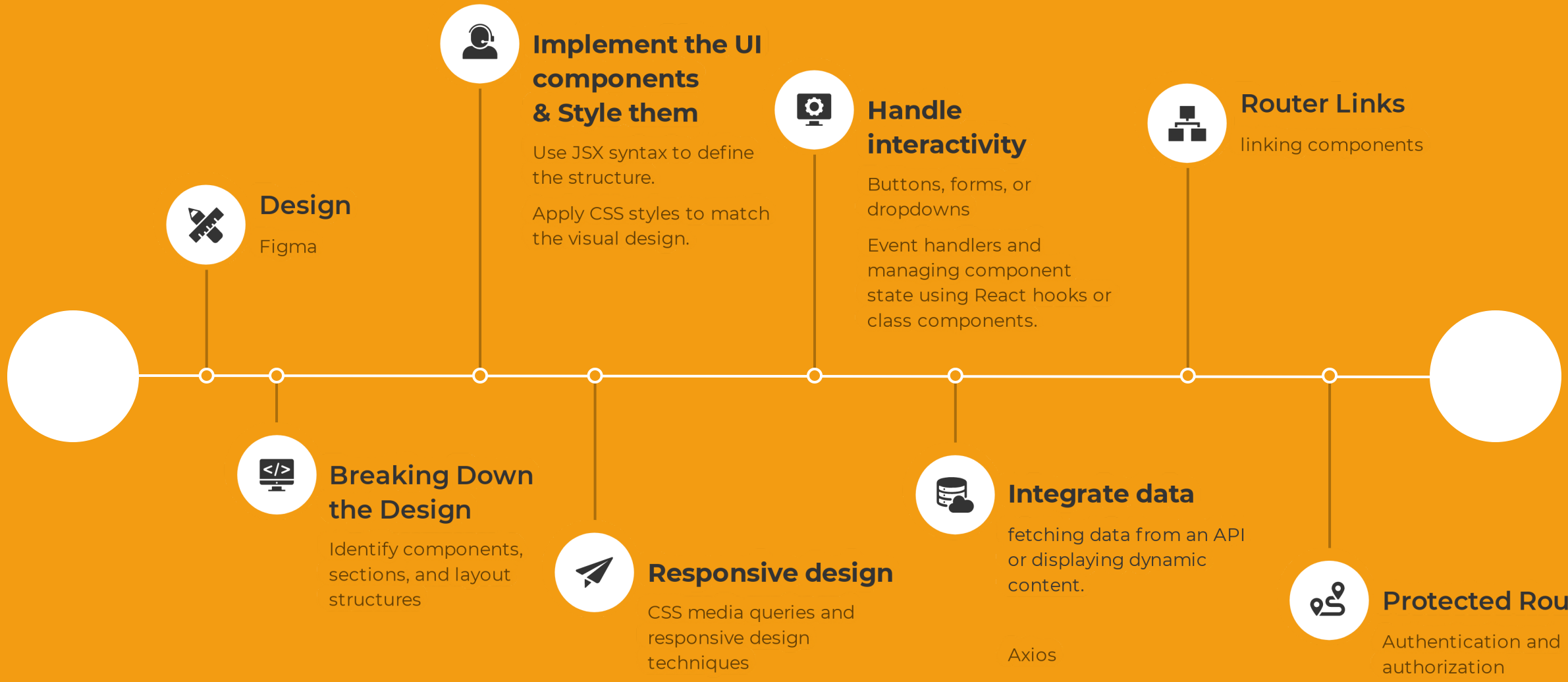
2 Organize the APIs

3 Test APIs

4 Share and Collaborate



Phase IV: **Front-end** Implementation



Design

Figma

Breaking Down the Design

Identify components, sections, and layout structures

Implement the UI components & Style them

Use JSX syntax to define the structure.

Apply CSS styles to match the visual design.

Responsive design

CSS media queries and responsive design techniques

Handle interactivity

Buttons, forms, or dropdowns

Event handlers and managing component state using React hooks or class components.

Integrate data

fetching data from an API or displaying dynamic content.

Axios

Router Links

linking components

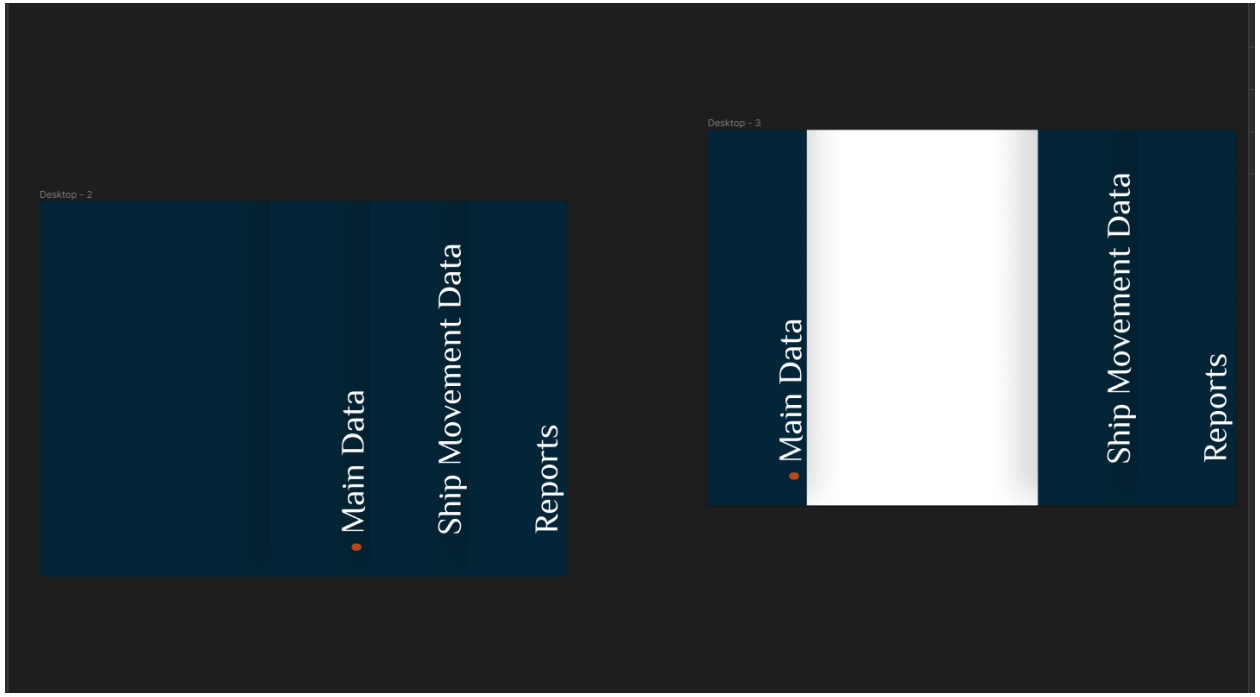
Protected Routes

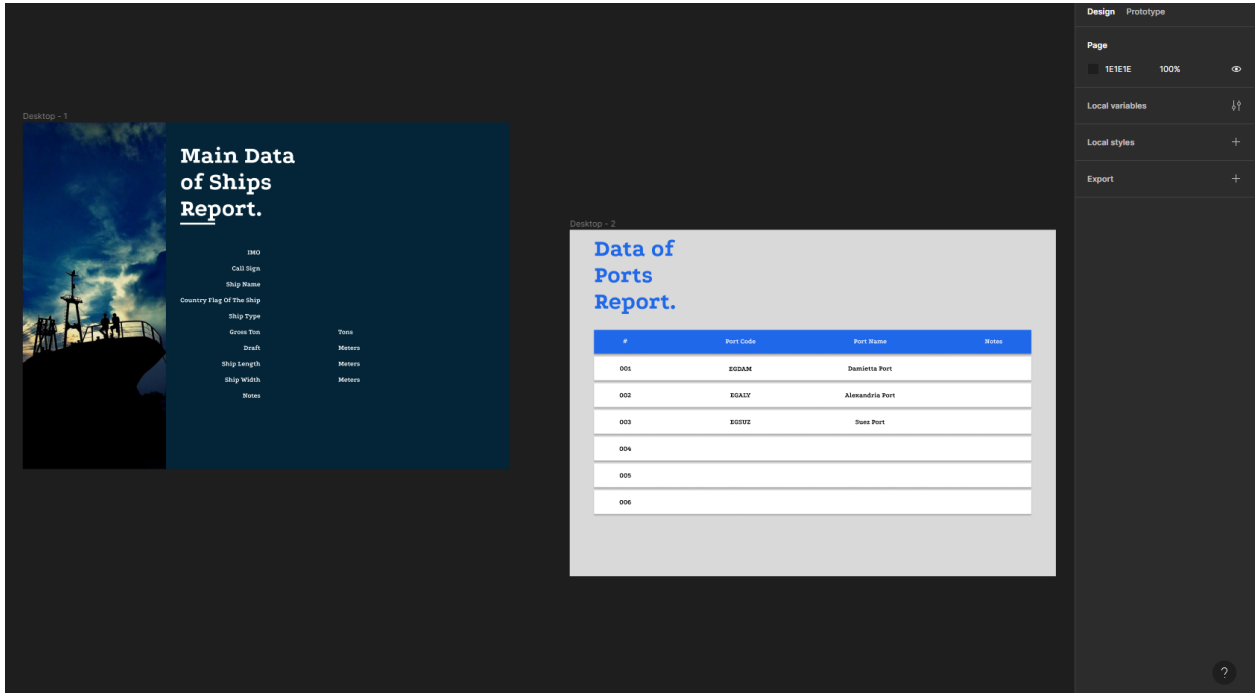
Authentication and authorization

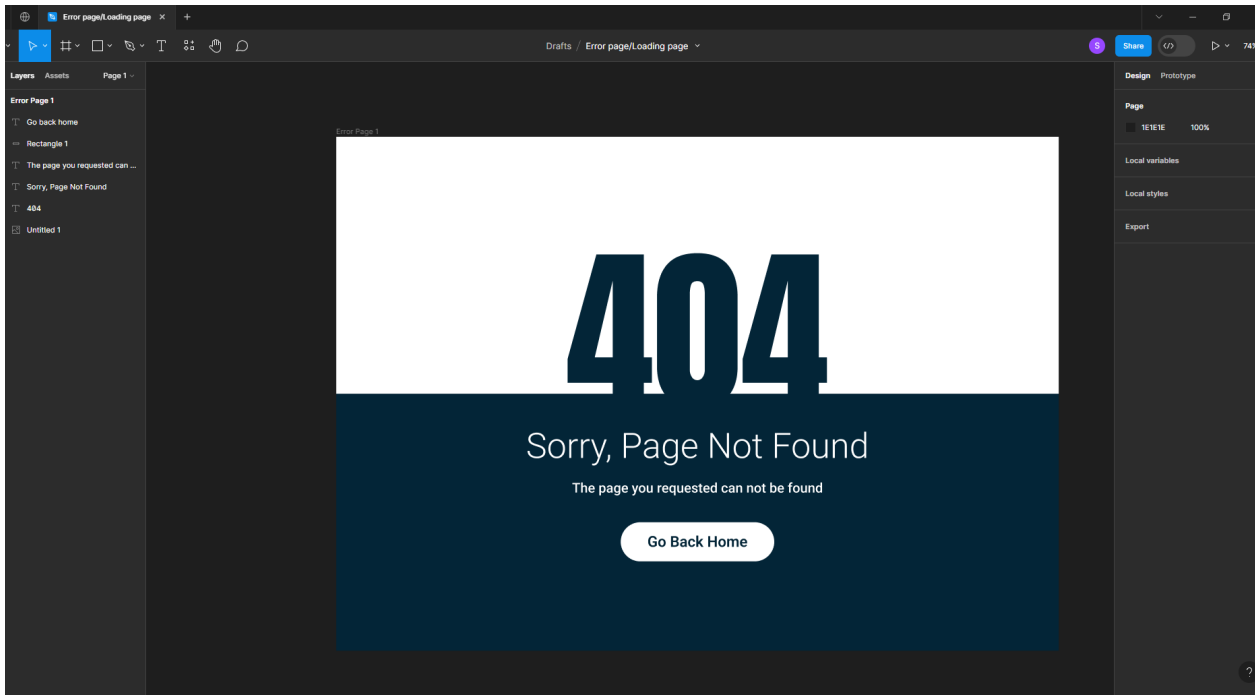
Design

Figma

- What is Figma
- What have been done

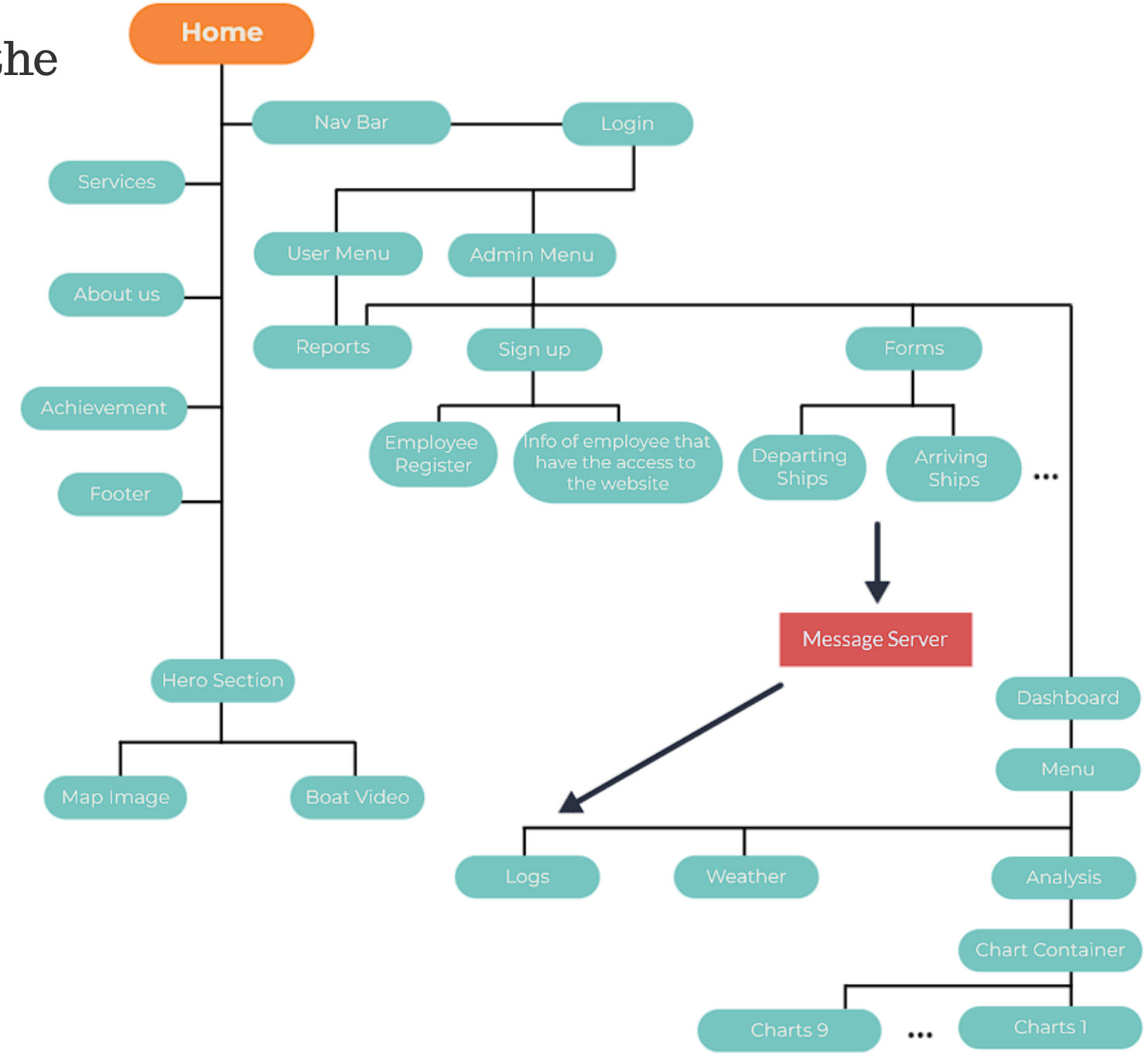

Desktop - 2
Main Data
Ship Movement Data
Reports
Desktop - 3
Main Data
Ship Movement Data
Reports


Desktop - 1
Main Data of Ships Report.
IMO
Call Sign
Ship Name
Country Flag Of The Ship
Ship Type
Gross Ton
Draft
Ship Length
Ship Width
Notes
Tons
Meters
Meters
Meters
Desktop - 2
Data of Ports Report.
| Port Code | Port Name | Notes
001 | EGDAM | Damietta Port
002 | EGALY | Alexandria Port
003 | EGSUZ | Suez Port
004
005
006
Design Prototype
Page
1E1E1E 100%
Local variables
Local styles
Export

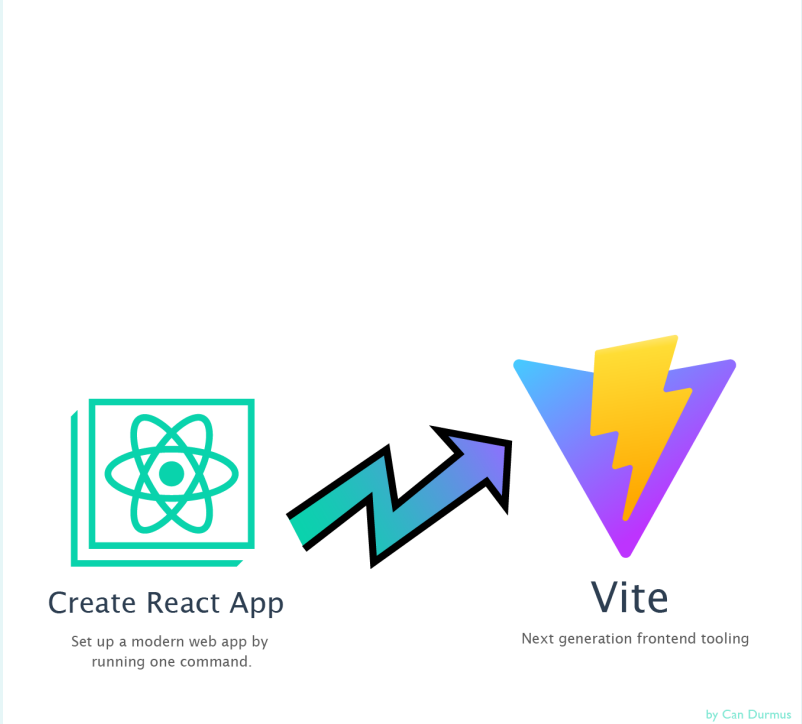

Error page/Loading page
Drafts / Error page/Loading page
Share
Layers Assets Page 1
Error Page 1
Go back home
Rectangle 1
The page you requested can ...
Sorry, Page Not Found
404
Untitled 1
Error Page 1
404
Sorry, Page Not Found
The page you requested can not be found
Go Back Home
Design Prototype
Page
1E1E1E 100%
Local variables
Local styles
Export


Data Of Ship Types
Data of Arriving Ships
Data Of Countries
Data of Departing Ships
Data Of Agents
Data Of Ports
Main Data Of Ships

Breaking Down the Design.



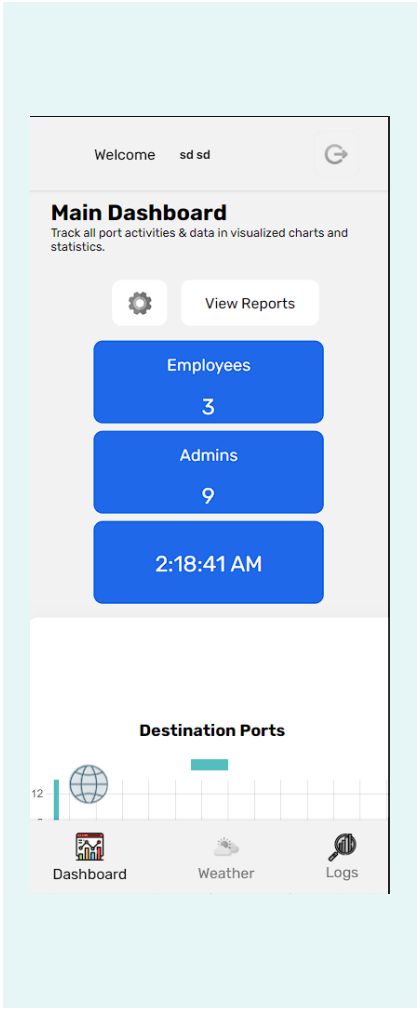
Create React App (CRA) vs. Vite

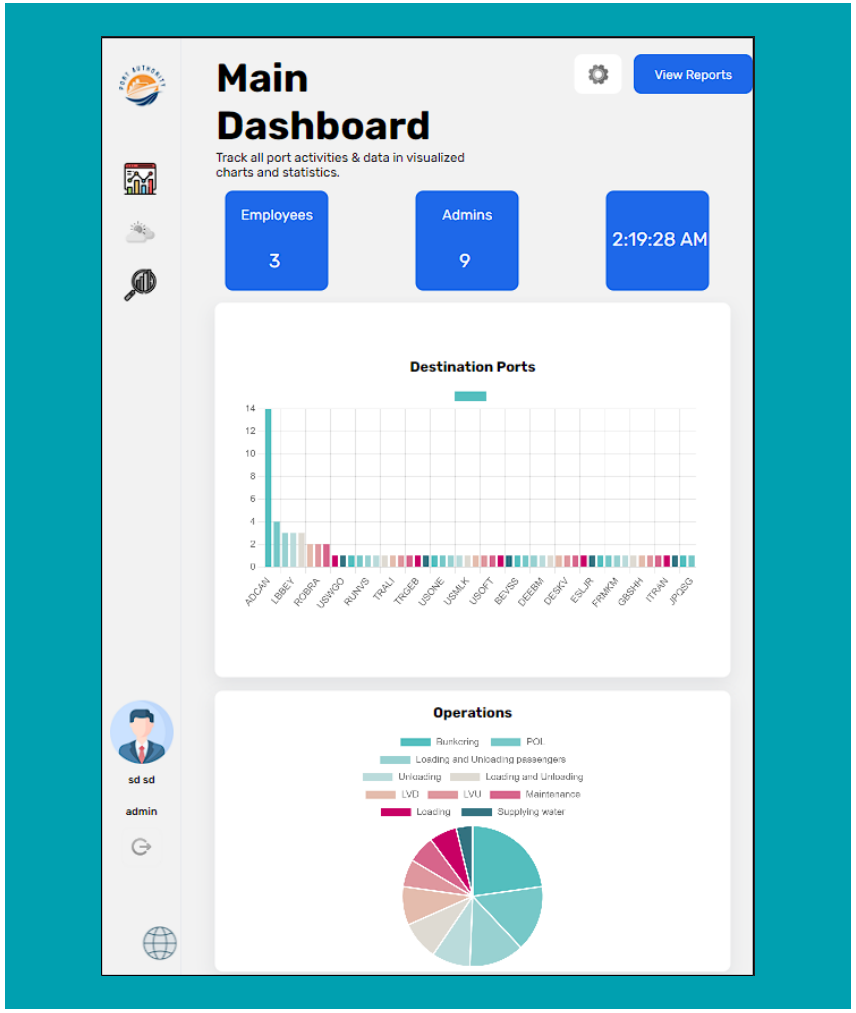


	Create React App	**Vite**
Ease of use	Easy to use	**Can be more difficult to learn**
Performance	Slower build times	**Faster build times**
Customization	Difficult to customize	**Easier to customize**
Tooling	Includes a development server	**Includes a development server**
Community support	Large community	**Smaller community**

Responsive Design Techniques.

- Media Queries
  - ✔ CSS styles based on specific conditions





Dealing With Data

Handle interactivity & Integrate data.

Dealing With Data

Handle interactivity & Integrate data.

1. React Hooks
   - ✔ useState();
   - ✔ useRef();
   - ✔ useContext();
   - ✔ useEffect();
2. Axios
   - ✔ What is Axios?
   - ✔ Features supported.
   - ✔ Reason behind choosing Axios.
3. Custom Hooks to fetch data with Axios.

Axios

- Axios is a promise-based HTTP client for JavaScript.

  It makes sending asynchronous HTTP requests to REST endpoints easier and helps front-end team to perform CRUD operations. This REST endpoint/API is backend Node.js server.

- Features of Axios

  Promise-based, Supports many request methods, Cancellation, and Progress events.

Custom Hooks to fetch data with Axios.

```
const useAxiosFunction = () => {
const [reponse , setResponse] = useState([]);
const [error , setError] = useState('');
const [loading , setLoading] = useState(false);
const [controller ,setController] = useState();

const axiosFetch = async(configObj) =>{
    const {
        axiosInstance,
        method,
        url,
        requestConfig = {}
    } = configObj;
    try{
        setLoading(true);
        const ctrl = new AbortController();
        setController(ctrl);
        const res = await axiosInstance[method.toLowerCase()](
url,{   ...requestConfig,
        signal:ctrl.signal
        });
        setResponse(res.data.allentries);
         console.log(res.data.allentries);

        }catch(err){
        setError(err.message);
        console.log(err)
        }
        finally {
        setLoading(false);
        }
}
useEffect(()=>
   {
      //useEffect clean up
   return()=> controller&& controller.abort();


   },[controller])
    //console.log("da el response",reponse);
    return [ reponse,error , loading, axiosFetch ];
}
export default useAxiosFunction;
```

```
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;
```

Custom Hooks with Axios (Cont.)

```
import axios from 'axios';
const
BASE_URL="http://164.90.186.113:3000/api/get/agentdata"
export default axios.create({
    baseURL :BASE_URL,
    headers :{
        'Content-Type' : 'application/json',
        'Accept' : 'application/json'
    }
})
```

API OF SEARCH

```
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
    }
 })
```

CALL OF HOOK

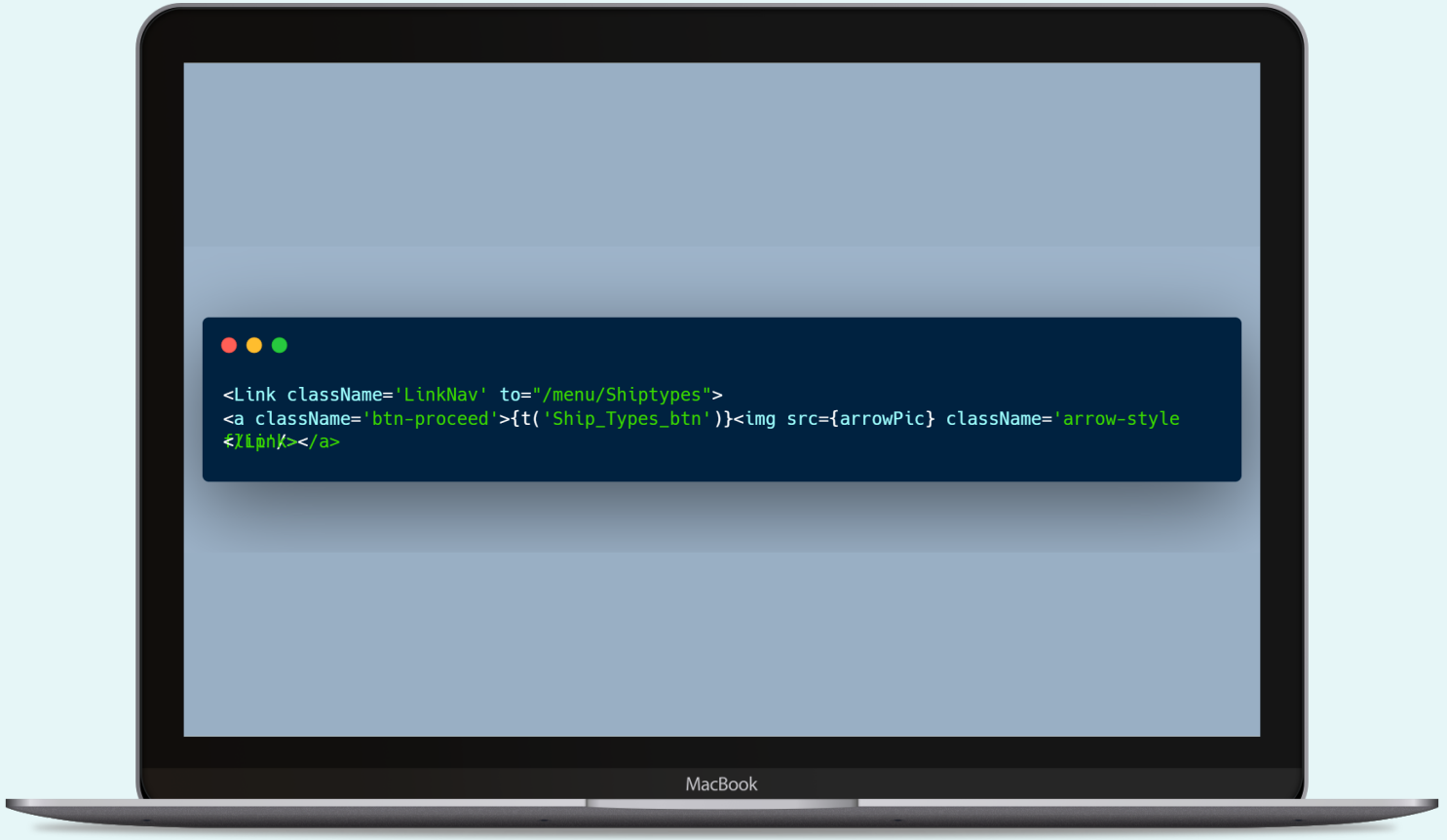
Router Links.

React Router DOM

- What is **React Router DOM**?
- Why **React Router DOM**?

Router Link.

- A component that is used to navigate to a different route in a React application.
- The **Link** component is a component provided by **React Router DOM.**



Need of **Login** Page

Authentication

Verifying the identity of a user or system. It is a critical security measure that helps to protect systems and data from unauthorized access.

Privacy

Login pages help to protect the privacy of your users by ensuring that only authorized users can access their personal information.

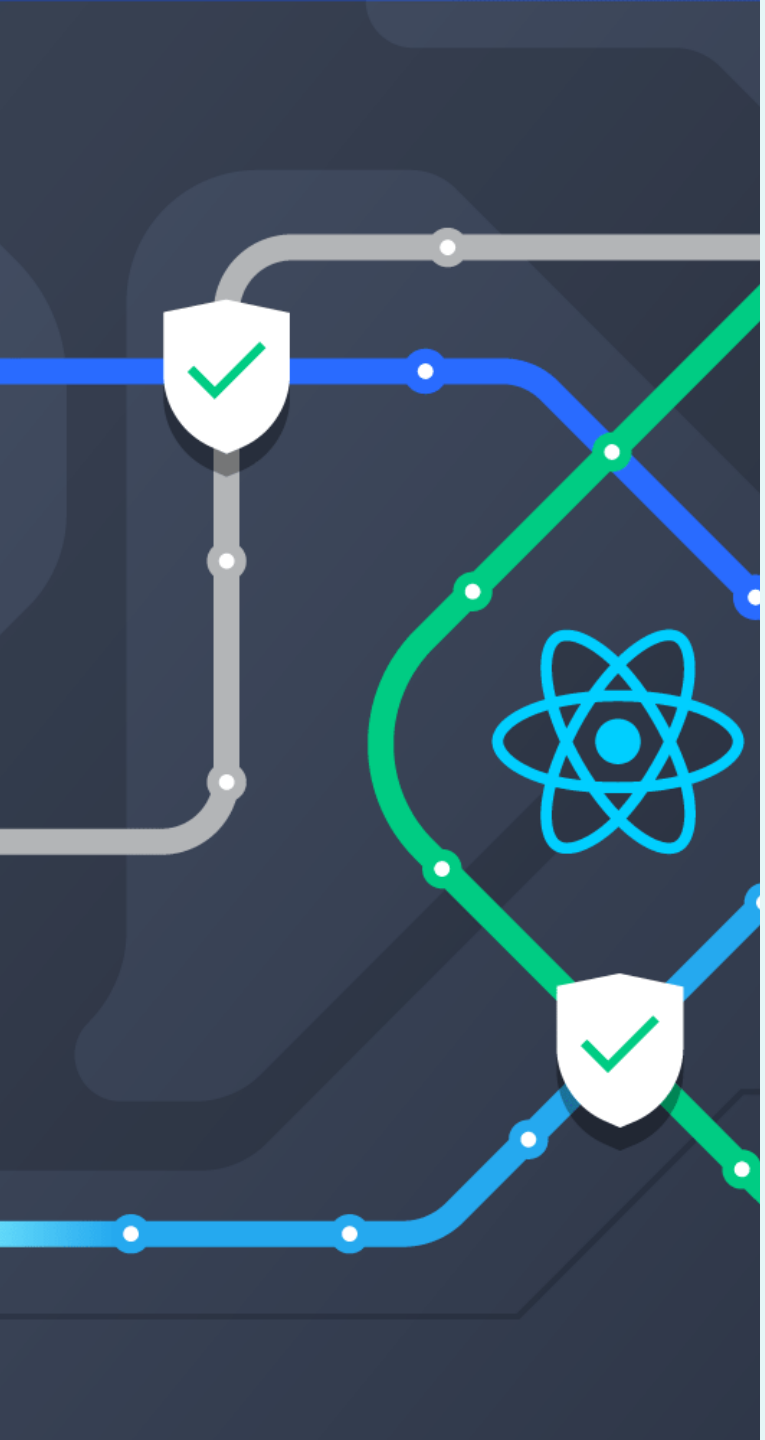
Accountability

Login pages help to ensure that users are held accountable for their actions by requiring them to log in before they can take certain actions.

Analytics

Login pages can be used to track user activity and gather data that can be used to improve the website or app.

Protected Routes.



Protected Routes.

What is protected routes?

Why Needed?

How its used in our web app in conjunction with **authentication?**

```jsx
import { useContext } from "react";
import AuthContext from "../context/AuthProvider";
const useAuth =()=>{
    return useContext(AuthContext);
}
export default useAuth;
```

USEAUTH

```jsx
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;
  return go(f, seed, [])
}
```

REQUIREAUTH

```jsx
<Route element = {<RequireAuth allowedRoles={['admin']} />}>
                <Route path="/menu" element = {<Menus/>}/>
</Route>
.
.
.
.
.
 <Route element = {<RequireAuth allowedRoles={['admin' , 'other']} />}>
...
 </Route>
```

Signup Page

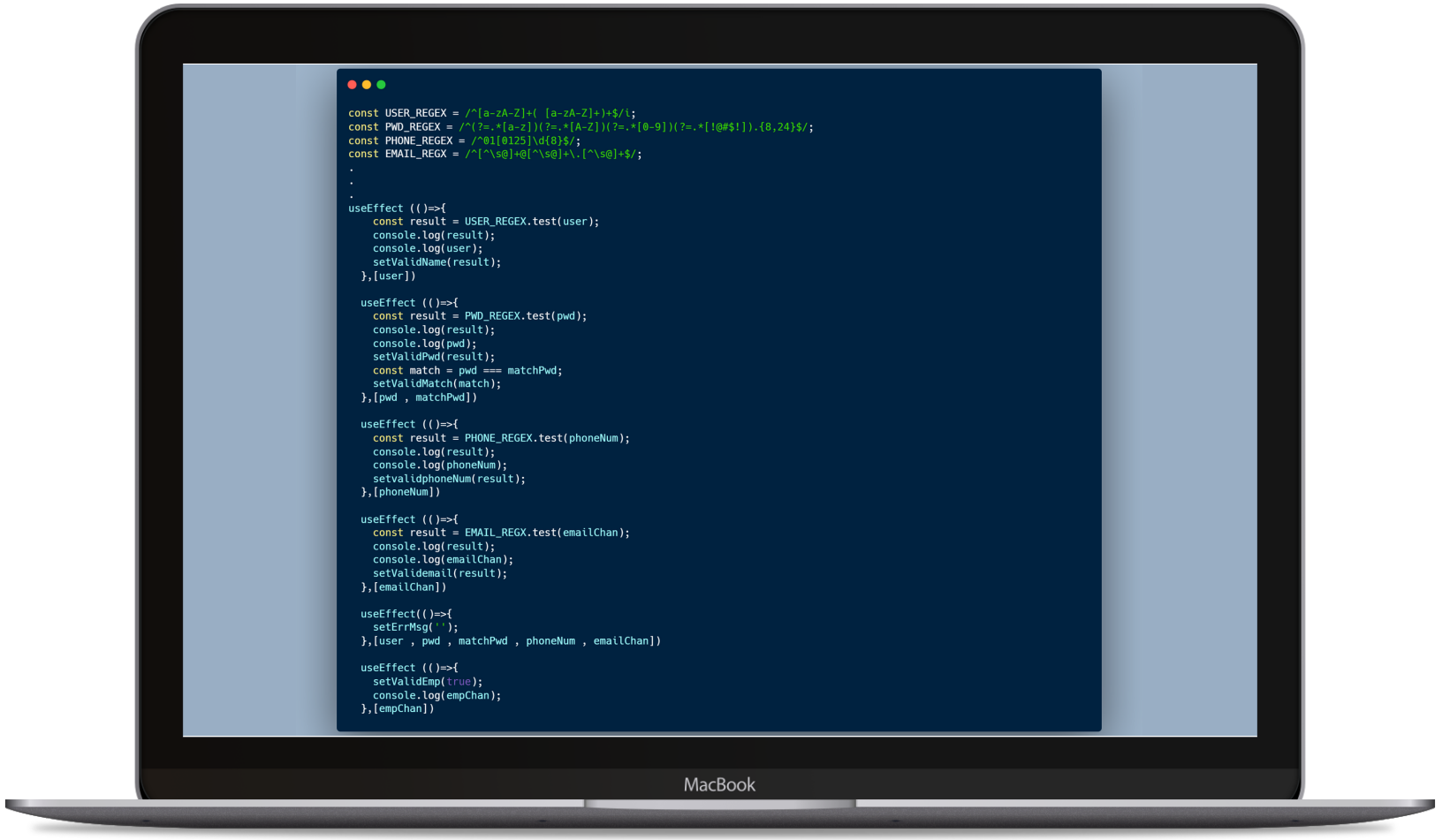
Admins

The only users who have the privilege to register new employees and access all their data as well.

Why?

Signup(cont.)

Validation



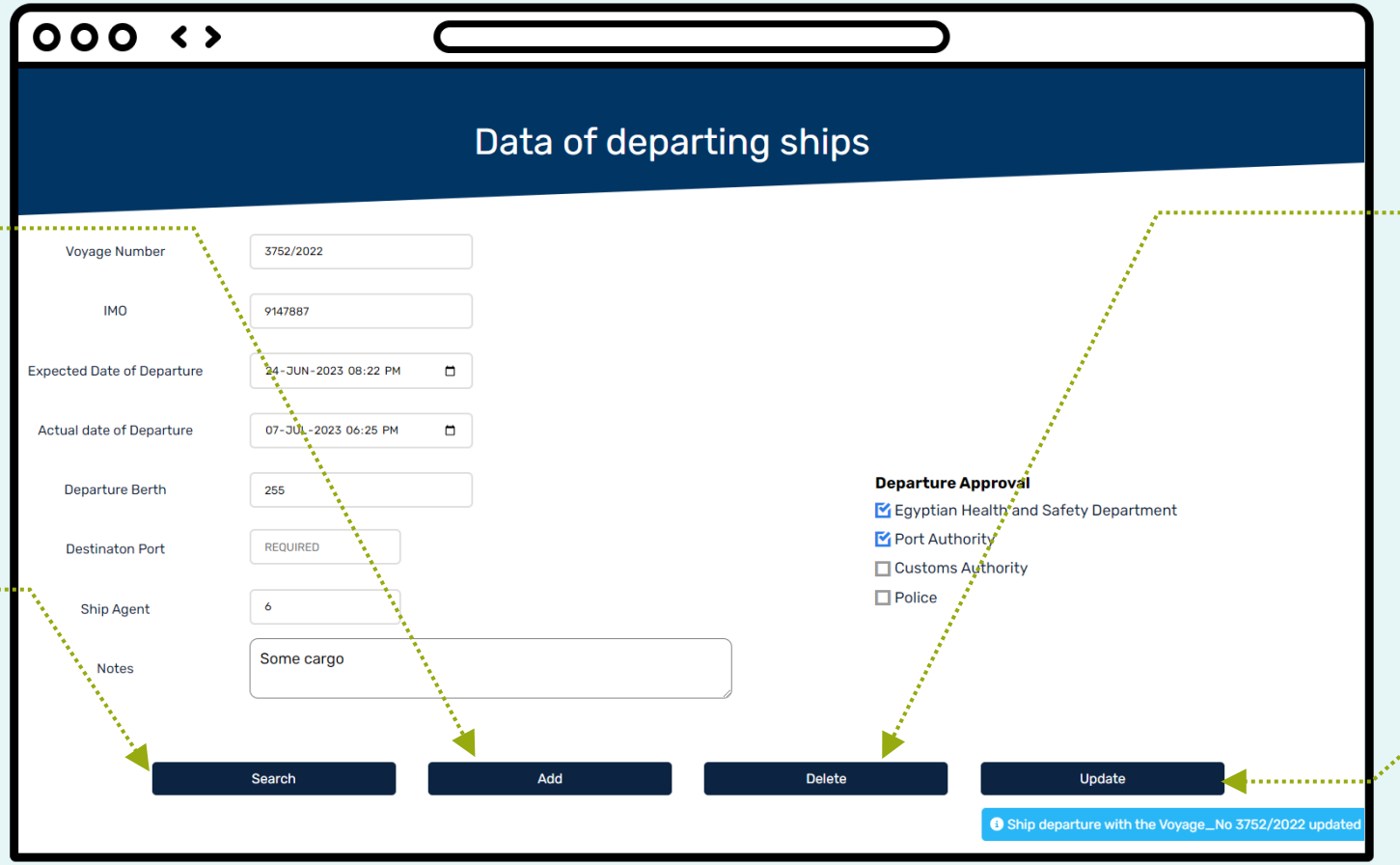
Forms, Reports & Dashboard.

Forms

CURD operations can be performed from the forms.

Create/Add

Read/Search



Delete

Update

```
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
    }
 })
 if(reponse )
 {
  setresponseText(true)
 }

 const timer = setTimeout(() => {
  setresponseText(false);
}, 5000);
 setAgentCode("");
 setAgentName("");
 setAddress("")
 setPhone("")
 setEmail("")
 setPermNo("")
 setPermDate("")
 setPermExpDate("")
 setNotes("")
}
```

ADD

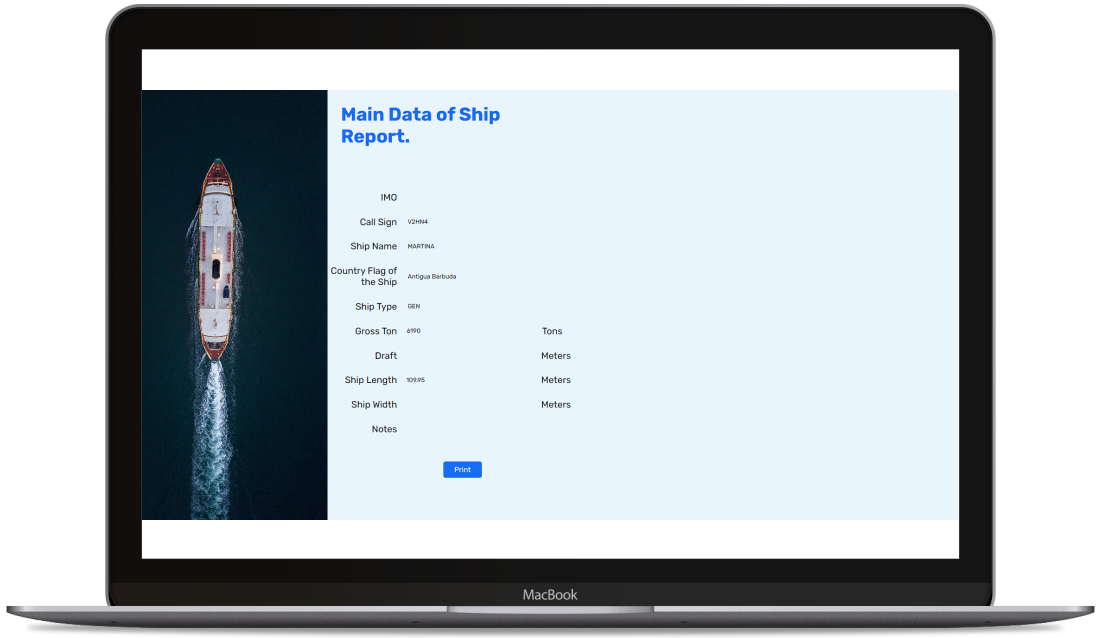
```
  const clickApi = ()=>{

    axiosFetchOne({
      axiosInstance: axios,
        method :'GET',
        url:'/',
        requestConfig :{}
    })
}

useEffect(()=>{

    const searchValue = inputRef.current.value.toUpperCase();

    reponseOne.map((datas)=>{
    if(searchValue == datas.Agent_Code)
    {
    console.log(`Search Value: ${datas.Agent_Code}`)
    setAgentCode(datas.Agent_Code);
    console.log(`Search Value: ${AgentCode}`)
    setAgentName(datas.Agent_Name);
    setAddress(datas.Address);
    setPhone(datas.Telephone);
    setEmail(datas.Email);
    setPermNo(datas.Perm_No);
    if(datas.Perm_dt_st == null)
    {
        setPermDate(null);
    }
    else{
     var date = new Date(datas.Perm_dt_st)
     setPermDate(moment(date).format("YYYY-MM-DDTkk:mm"))
    }
    if(datas.Perm_dt_end == null)
    {
        setPermExpDate(null);
    }
    else{
      date = new Date(datas.Perm_dt_end)
     setPermDate(moment(date).format("YYYY-MM-DDTkk:mm"))
    }

    setNotes(datas.notes);
    }
    if(searchValue ==""){
        setplaceValue(t('please_Target'))
      }
      })
},[reponseOne])
```

SEARCH

Reports

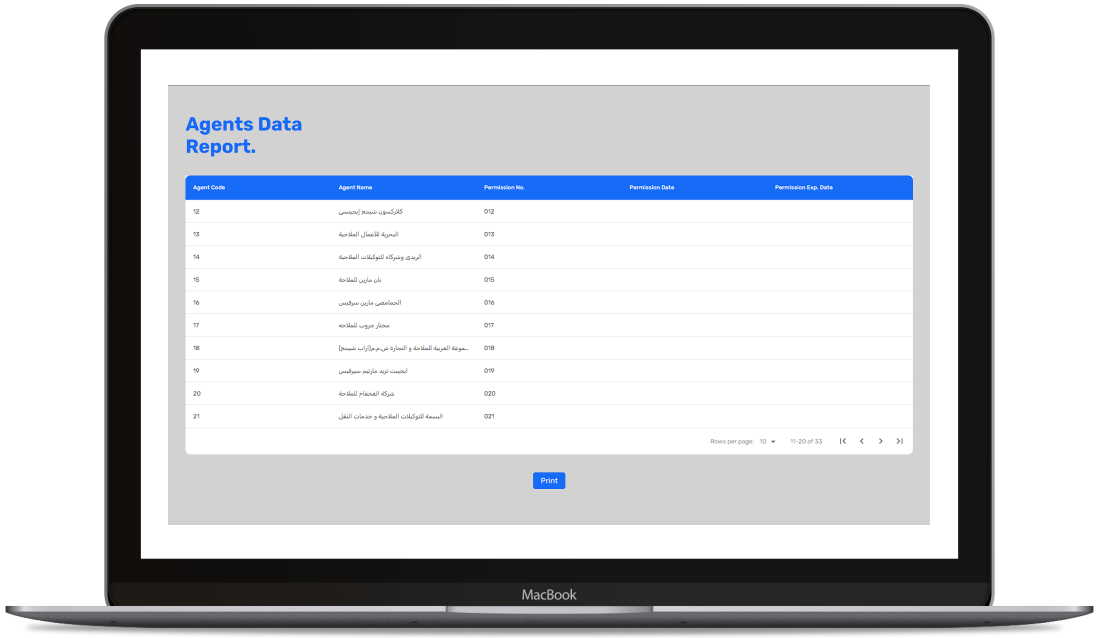
Reports was made to let the employees know the information about the ship.

* Reports can be printed



Single Record Reports.

All of them require search criteria.



Multiple Records Reports.

Most of them don't require search criteria.

```
    import DataTable from "react-data-table-component";
.
.
.
       const columns = [
        {   name: t('Country_Code'),
            id:"code",
            selector: (row) => row.Country_Code},
        {   name:t('Country_Name'),
            id:"name",
            selector: (row) => row.Country_Name }
        ]
.
.
.
   <DataTable
        columns={columns}
        data={data}
        progressPending={loading}
        pagination
    />
```

REACT-DATA-TABLE-COMPONENT

REACT COMPONENT THAT IS USED IN TABLE CREATION IN MULTIPLE-RECORDS REPORTS.

```
    const ComponentRef = useRef();

    const HandlePrint = useReactToPrint({
        content: () => ComponentRef.current,
        documentTitle:"countries-data-report",
    });

.
.
.
.
.

<button onClick={HandlePrint} className='printbtn'>{t('Print')}</button>
```

REACT-TO-PRINT

REACT COMPONENT THAT IS USED IN PRINTING ALL REPORTS.

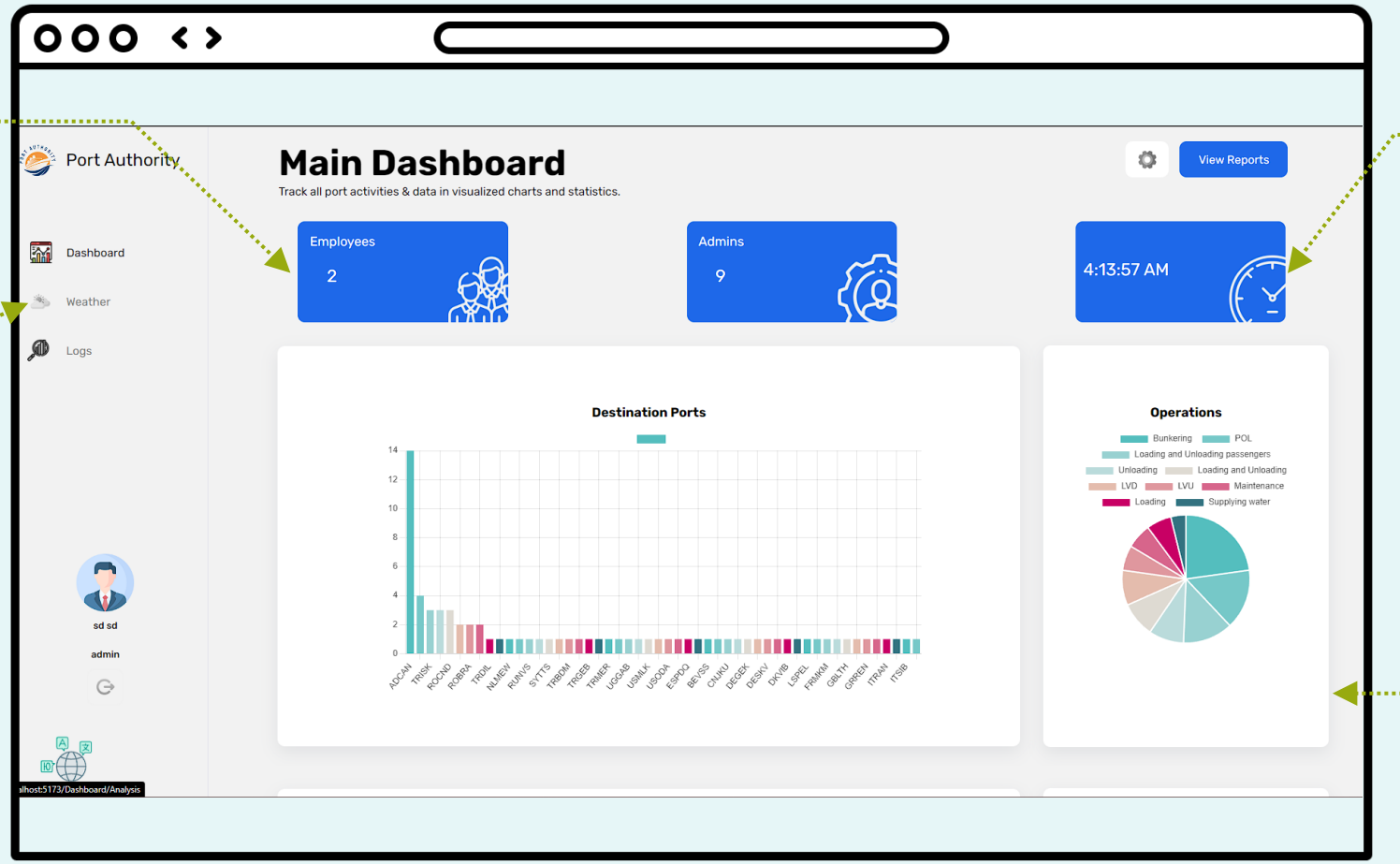
Dashboard

Count Cards

System Time

Weather

Data Visualization



OpenWeatherMap

- A free weather API that provides current weather data.
- Widely used.
- The OpenWeatherMap API is RESTful, which means that it uses HTTP requests to access the data. This makes it easy to integrate with other applications and services.

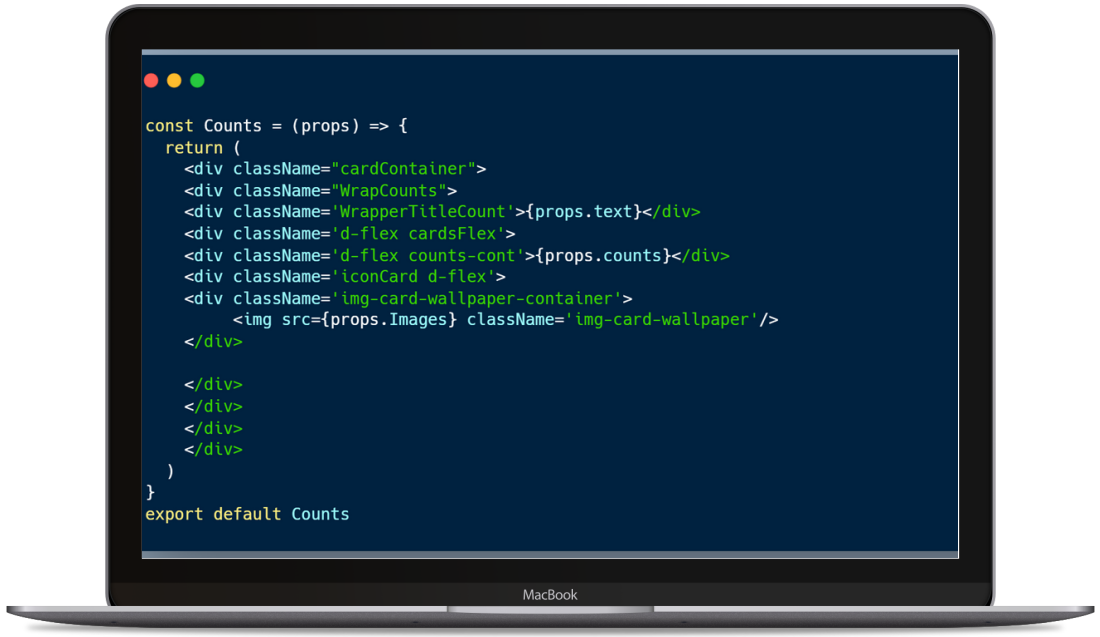
Chart.js

Chart.js

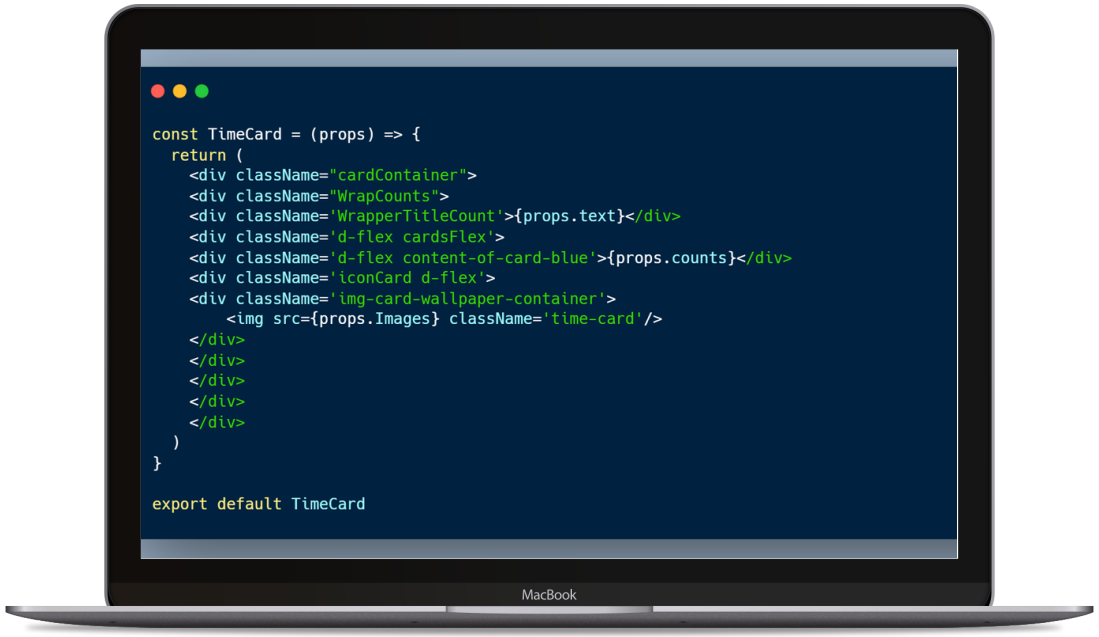
- A free, open-source JavaScript library for creating interactive charts.
- One of the most popular charting libraries in the world.
- It supports a variety of chart types.
- It is highly customizable.

Dashboard(cont.)

Count Card and Time Card Components



Count Card



Time Card

Dashboard

Logs

1. What is Logs?
2. How it's fetched?
3. How it's changes are added?

Logs(cont.)

```
{responseText && !errortwo && reponse && <MessageServer
 text={reponse.message}
 message={responseText}
 fadeDuration={30000} />}

{errortwo && <MessageServer
 text={errortwo.message}
 message={responseText}
 fadeDuration={30000} />}
```

```
const Logs = () => {
    const [data, error, loading] = useFetch({
      axiosInstance: axios,
      method: 'GET',
      url: '/',
      requestConfig: {}
    });

    const columns = [
      {name: t('Employee_Name'),
        id: 'code',
        selector: (row) => row.name },
      {name: t('date_Change'),
        id: 'Aname',
        selector: (row) => row.log_date},
      { name: t('Changes'),
        id: 'name',
        selector: (row) => row.message}];

    const sortedData = [...data].sort((a, b) => {
      // Compare the log_date fields in descending order
      return new Date(b.log_date) - new Date(a.log_date);
    });

    const ComponentRef = useRef();
```

Hook Call

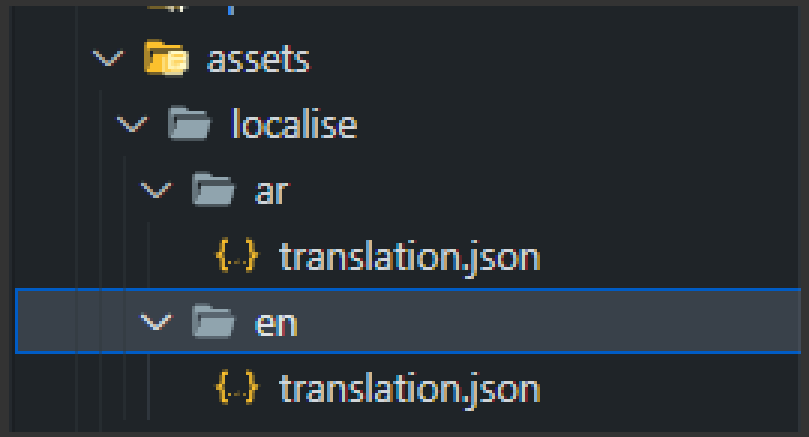
Descending order to show new changes at the top

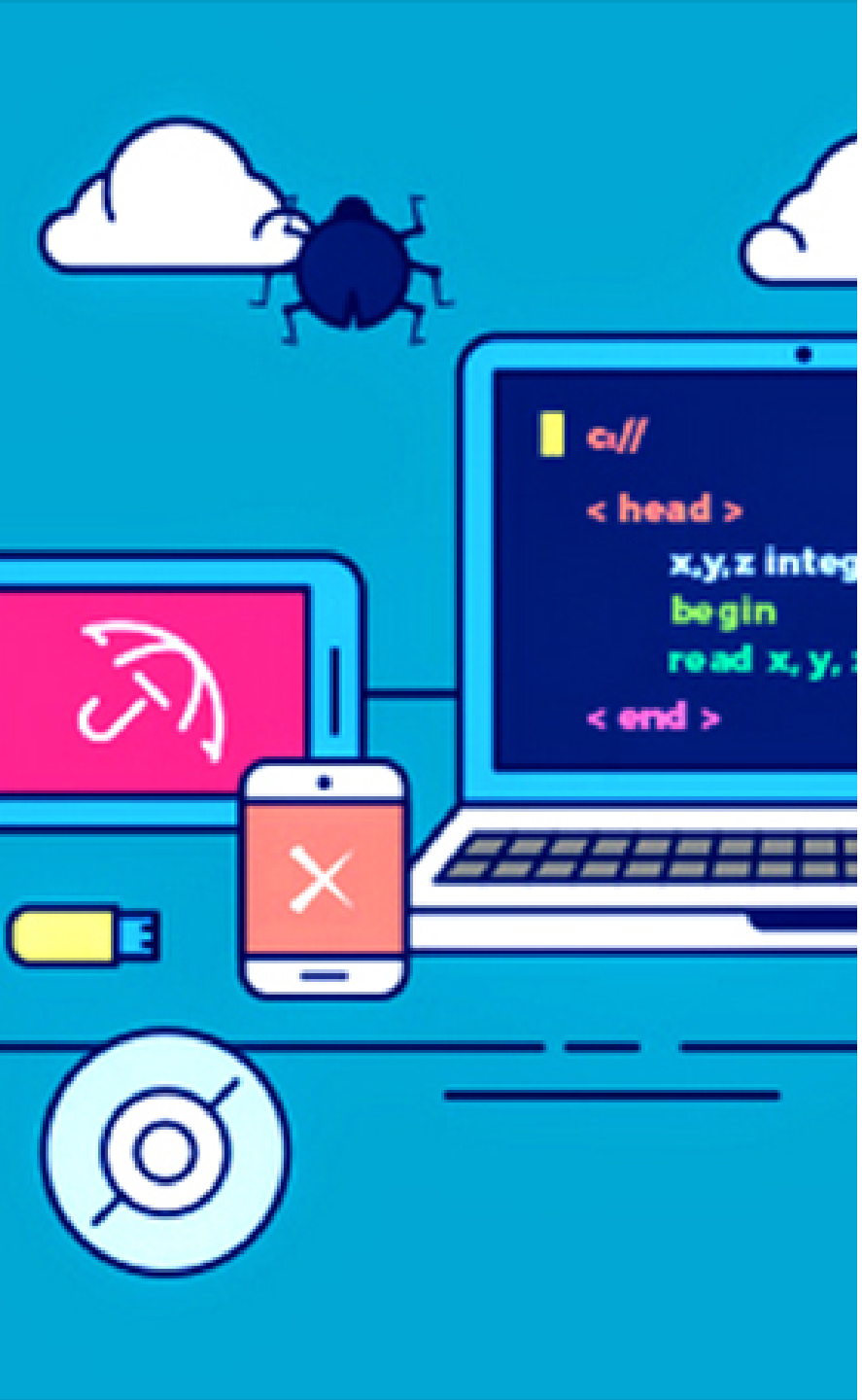
Translation

Our web app supports Arabic.

- Library used: React-i18next
- i18next is a popular library for internationalizing JavaScript applications. It is a well-documented and well-maintained library

```jsx
import { useTranslation } from 'react-i18next';
.
.
.
//inside component
const { t } = useTranslation();
.
.
.
 <h1>{t('Country_Data')} {t('Report')}</h1>
```





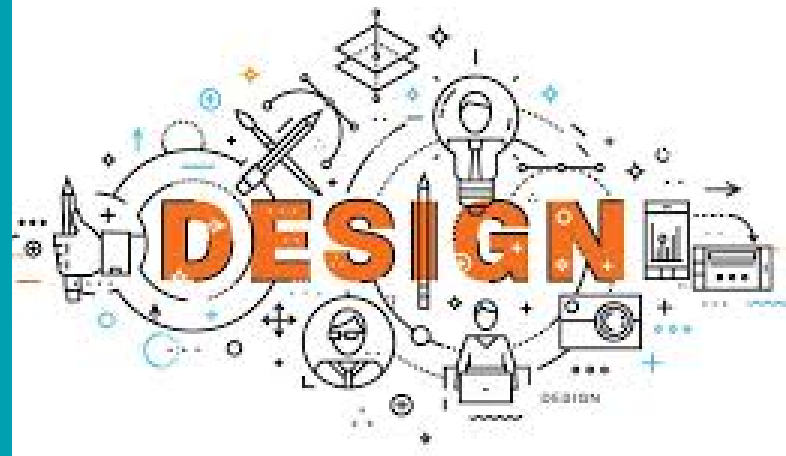
Phase V: Testing

The code is usually tested across multiple browsers on multiple devices in order to ensure cross-browser compatibility and make sure all users have the same experience while they interact with the website or application.

Design constraints

Design Constraints

1. Operation time constraint
2. Budget Constraint
3. Technical Constraint
4. Usability Constraint
5. Security Constraint
6. Legal Constraint



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.

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.

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.

Software Tools.

Software	Version	Why?
MySQL	**MySQL v8**	**Free.Huge community with a lot of libraries.**
Node.js	**node-v18.12.1-x64**	**Full-stack JavaScript.**
React	**18.2.0**	**Component-based, Declarative Syntax, Virtual DOM Utilizing**

Hardware Requirements

Database

Ram Size	HDD Size	Processors	Available Query Connection
1GB	5 GB	1 CPU	75
2GB	20 GB	2 CPU	150
4GB	40 GB	2 CPU	400
8GB	80 GB	2 CPU	800
16GB	160 GB	4 CPU	1,600

Backend Requirements

Assuming that the largest query uses 0.5% of 1 CPU

Persons Assumption	Query / sec	CPU required
2,000	1	10 CPU’s
4,000	1	20 CPU’s
8,000	1	40 CPU’s

Database Hardware Cost

Assumption	Cost	Maximum Handling Capacity
2x Xeon 4 core CPU 16 GB 1 TB HDD	$247	About 3200 queries /sec

Back-end Server Cost

Specs	Cost	Maximum Handling
2x Xeon 10 core CPU 16 GB 1TB HDD	$255	2000 request/ sec

Business Model

KEY PARTNERS

- Customs Authority
- Import and Export Monitoring Agency
- Police
- Logistics Companies
- Egyptian Health and - Safety Department
- Warehouses
- Security

KEY ACTIVITIES

- Centralized Database
- Monitor Ship Movement
- Information Sharing
- System Platform Management

KEY RESOURCES

- Port Authority Personnel
- Shipping Agent
- Database
- Technical Resources

VALUE PROPOSITIONS

- Improve port information flow
- Improved terminal planning
- Improved port call efficiency
- Improved port service planning
- Enhancing collaboration between government agencies

CUSTOMER RELATIONSHIP

- Privacy & SEC of data
- Provide annual reports
- Data Exchange

CHANNELS

- Port Website
- Database for port

CUSTOMER SEGMENTS

- Shipping Agent
- EAFMS
- GOEIC
- Customs
- Container Companies

COST STRUCTURE

- Server and DB Hardware and System Environment
- System Development

REVENUE SOURCES

- Initial Price fee
- Our optional Service
- Maintenance fee

Ethical Considerations

- Transparency
- Data Privacy and Security
- Responsible use

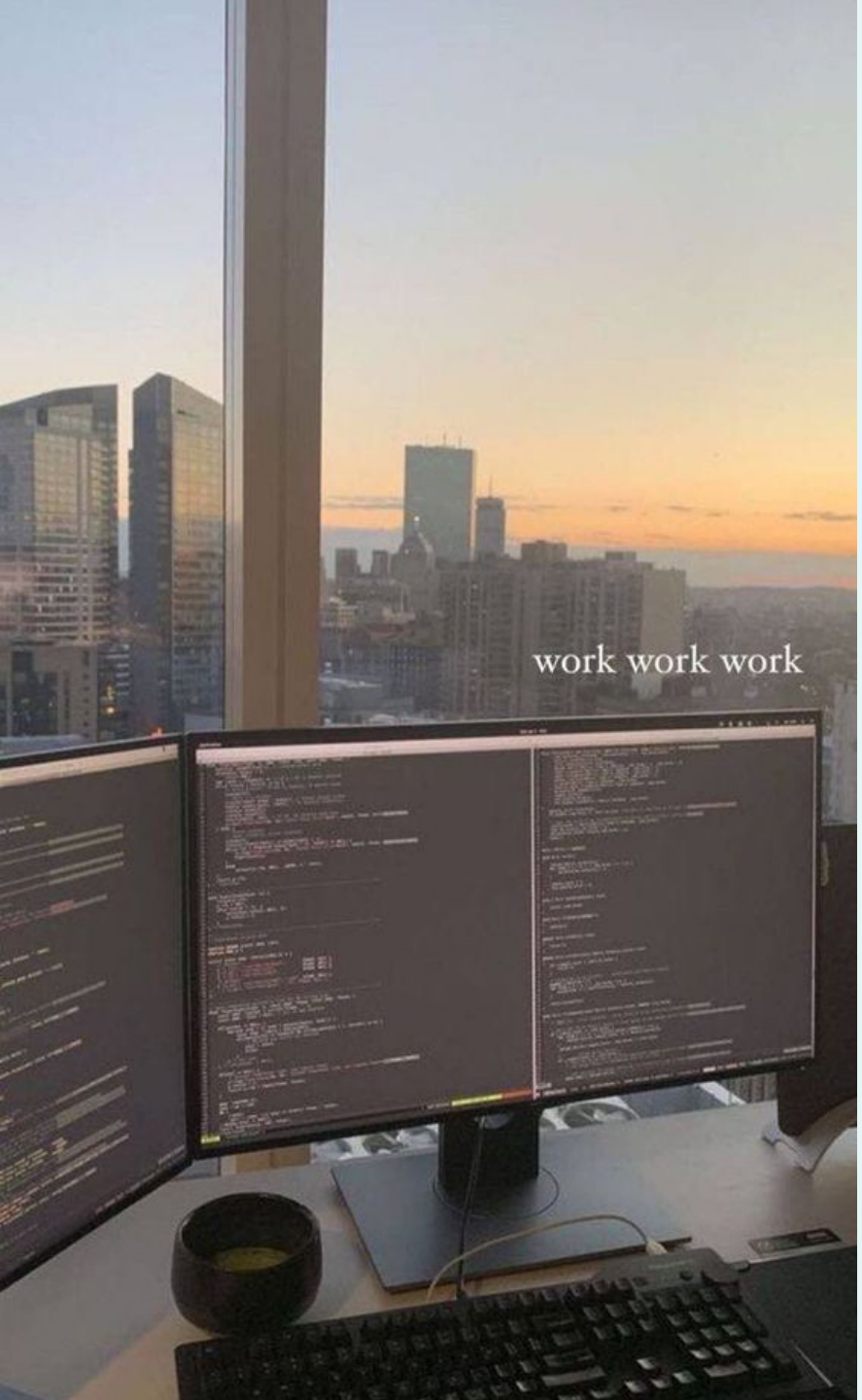
What has been done in Project I

We have paid multiple visits to Alexandra port to Know the life cycle of the port authority system.

We have already started collecting some real data.

Design the structure of the database.

progress in the front-end implementation.



What has been done In project II

- Implemented the designs we have made in project I
- Connected the Frontend and Backend and created a functioning system.
- Hosted the System on the cloud to test it easily.
- Used Real data to test the application.

Future Work

- Implement load balancing and auto-scaling feature.
- Add a new layer of Security between the Backend and DB.
- Add data validation in the Backend.
- Add more efficient routes for the frontend to use.

References

- 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. [0]

- 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.[1]

- 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.[2]

References(cont.)

- 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.[3]

- 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.[4]

Thank You

Any Questions?