INSTITUT SUPÉRIEUR D’ÉLECTRONIQUE DE PARIS (ISEP)

Transforming Global Logistics with TechnoLogix Solutions

Submitted By

S/N	NAME	ISEP No.	Task
1.	LEIGHA IFIYEMI	62769	3 and 5
2.	BUHARI ALIYU	62788	4 and 5
3.	ABUBAKAR UMAR ELNAFATY	62717	2 and 5
4.	YASMINE HOUHOU	60769	1 and 5

JANUARY 2024

Table of Contents

CHAPTER 1
INTRODUCTION

1.1. Overview of the Project

1.2 Contemporary Logistics Company in a view

Below is a fundamental background on logistics as it applies to logistics companies in today's world, including its current structure, operations, and other relevant information. This overview provides context for understanding the modern logistics industry.

Background Information on Logistics in Today's World

Current Structure of Logistics

The modern logistics industry is structured around the efficient movement of goods from suppliers to customers. It encompasses various components, such as transportation management, warehousing, inventory control, order fulfillment, and supply chain management.

1. **Supply Chain Management (SCM)**: SCM is at the core of logistics, encompassing the planning, implementation, and control of supply chain activities to create value, build a competitive infrastructure, and synchronize supply with demand [1].
2. **Transportation**: This includes different modes like road, rail, air, and sea, chosen based on cost, time, and product characteristics. Companies often use a combination of these modes (multimodal transportation) for optimal efficiency [2].
3. **Warehousing and Distribution Centers**: Warehousing is critical for storing, managing, and moving inventory. Modern warehouses are often automated with advanced technologies like robotics and AI for efficient inventory management [3].
4. **Inventory Management**: Effective inventory management strategies like Just-In-Time (JIT) and Vendor Managed Inventory (VMI) are used to reduce costs and improve efficiency [4].
5. **Technological Integration**: The integration of technology, such as IoT, blockchain, and big data analytics, has become integral to modern logistics, offering real-time tracking, improved efficiency, and data-driven decision-making [5].

Operations in Modern Logistics

The operations in modern logistics are characterized by a high degree of automation, digitalization, and interconnectivity.

1. **Digitalization**: Digital platforms are extensively used for order processing, customer service, and transport management. This includes Electronic Data Interchange (EDI), online tracking systems, and Customer Relationship Management (CRM) systems [6].
2. **Automation and Robotics**: Automation in warehousing and distribution centers is increasingly common. Robotics and drones are used for picking, packing, and delivery processes [7].
3. **Sustainability Focus**: There's a growing emphasis on sustainability in logistics. This includes optimizing routes for fuel efficiency, using eco-friendly packaging, and investing in electric or hybrid vehicles [8].
4. **Globalization and Network Design**: Logistics operations today are global, requiring efficient international supply chain management and compliance with various international regulations [9].
5. **Risk Management and Resilience**: Modern logistics operations incorporate sophisticated risk management strategies to deal with uncertainties like supply chain disruptions, geopolitical tensions, and natural disasters [10].

Key Actors in the Logistics Industry

1. **Logistics Service Providers (LSPs)**:
    - **Major 3PL and 4PL Providers**: Companies like DHL, UPS, and FedEx offer a range of logistics services, including transportation, warehousing, and supply chain management. 4PL providers, like XPO Logistics, go a step further by managing the entire supply chain [11].
    - **Local and Regional LSPs**: Smaller LSPs focus on specific regions or niches, offering personalized services [12].
2. **Transportation Companies**:
    - **Global Shipping Companies**: Maersk, Mediterranean Shipping Company (MSC), and CMA CGM are leading the way in ocean freight [13].
    - **Major Airlines for Air Freight**: Companies like Emirates SkyCargo, Lufthansa Cargo, and FedEx Express dominate air freight logistics [14].
    -

3. **Technology and Software Providers**:
    - **Supply Chain Management Software Vendors**: SAP, Oracle, and Infor are key players in providing SCM and ERP solutions [15].
    - **Innovative Tech Startups**: Startups like Flexport and project44 are disrupting the industry with digital-first approaches, leveraging AI and big data analytics [16].
4. **E-commerce Giants**:
    - **Impact of Amazon and Alibaba**: These companies have transformed logistics with their massive scale, in-house capabilities, and innovative last-mile delivery solutions [17].
5. **Regulatory and Standard-Setting Bodies**:
    - **International Organizations**: World Customs Organization (WCO) and International Air Transport Association (IATA) play a crucial role in setting global standards [18].
6. **Warehousing and Fulfillment Centers**:
    - **Automated Warehousing Leaders**: Companies like Amazon and Ocado are at the forefront of using robotics and automation in warehousing [19].

1.3. Summary of Technologix Context

From the context information provided in the document, below is the summary description of the context information.

- **Company**: TechnoLogix Solutions, a major player in the global logistics industry.
- **Industry**: Logistics, which involves the management of transporting goods from one place to another.
- **Challenges**:
    1. **Shipment Data Management**: Handling large volumes of data related to shipping.
    2. **Security**: Keeping sensitive information safe.
    3. **Adaptability**: Staying efficient in a rapidly changing industry.

- **Solution**: Use advanced technology to improve how the company operates.

1.3.1 Project Description Simplified:

Goal: To completely transform how TechnoLogix Solutions handles logistics on a global scale.

Key Areas of Focus:

1. **Governance**: Creating rules and policies for how the company should operate, focusing on efficiency and compliance.
2. **Information System Management**: Improving how the company uses technology to collect, store, and process information.
3. **Data Center Architecture**: Enhancing the physical and virtual infrastructure where the company's data is stored and managed.
4. **Microservices Development**: Breaking down the company's software into smaller, independent parts that can work together. This makes updating and maintaining software easier and more efficient.
5. **Blockchain Integration**: Implementing blockchain technology, which is a secure and transparent way to record transactions. This could be particularly useful for tracking shipments and ensuring data integrity.

CHAPTER 2
PROJECT DESCRIPTION AND OBJECTIVES

2.1. Scope of the Project

"The scope of the TechnoLogix Solutions project encompasses several key areas, each integral to the transformative impact on global logistics. This project, characterized by its breadth and depth, aims to address the following critical domains:

- **Governance**: Establishing a robust governance model that defines clear roles, responsibilities, and decision-making processes. This will ensure effective management and oversight of the project, from inception to completion.
- **Information System Management:** Overhauling the existing information systems to enhance data processing, storage, and retrieval capabilities. This involves upgrading hardware and software components, ensuring system security, and integrating advanced technologies to facilitate real-time data access and analytics.
- **Data Center Architecture:** Designing and implementing a state-of-the-art data center architecture. The focus here is on scalability, reliability, and efficiency. This involves selecting the right hardware, software, and networking solutions that can support the growing data demands of the logistics industry.
- **Microservices Development:** Developing and deploying microservices as part of the system architecture. This approach will allow for more agile, scalable, and maintainable software solutions. Each microservice will be designed to execute a specific business function, enabling easier updates and faster deployment of new features.
- **Blockchain Integration:** Implementing blockchain technology to enhance security, transparency, and traceability in logistics operations. Blockchain will be utilized to create a decentralized, tamper-proof ledger for recording transactions, tracking assets, and verifying data integrity across the supply chain.

2.2. Detailed Objectives

2.2.1. Governance

The objective for governance within this project is to establish a clear and effective management structure. Key goals include:

Developing a framework for decision-making processes and accountability.

Ensuring compliance with legal and regulatory requirements.

Facilitating effective communication and collaboration among all project stakeholders. This structure aims to provide a foundation for project success, aligning all efforts with the overarching goals of the organization.

2.2.2. Information System Management

Our objective in information system management is to create a robust and efficient system that supports the logistics operations. This includes:

Enhancing data processing speed and accuracy.

Improving data security and privacy measures.

Implementing advanced data analytics for better decision-making. These improvements will enable more effective management of logistics operations, contributing to overall operational excellence.

2.2.3. Data Center Architecture

In the realm of data center architecture, our objective is to design a scalable and reliable infrastructure that can handle growing data needs. Goals include:

Ensuring high availability and disaster recovery capabilities.

Implementing energy-efficient technologies to reduce environmental impact.

Optimizing data storage and access for improved performance. The new data center architecture will be a cornerstone in supporting the vast data requirements of modern logistics.

2.2.4. Microservices Development

For microservices development, the objective is to enhance the agility and scalability of our software solutions. This involves:

Developing independent, modular services that can be deployed and updated rapidly.

Ensuring seamless integration of these services with existing systems.

Creating a more flexible and maintainable software ecosystem. By adopting a microservices architecture, we aim to accelerate innovation and responsiveness to changing market needs.

2.2.5. Blockchain Integration

The objective of integrating blockchain technology is to bring unprecedented transparency and security to logistics operations. Key goals include:

Implementing a secure, decentralized ledger for recording transactions.

Enhancing traceability and accountability in the supply chain.

Reducing fraud and errors through immutable record-keeping. Blockchain integration will be instrumental in transforming logistics operations into more trustworthy and efficient processes.

CHAPTER 3
ANALYSIS AND METHODOLOGY

3.1. Analytical Approach "In this project, our analytical approach is focused on comprehensive understanding and addressing the complexities of modern logistics. Key aspects include:

- Conducting a thorough analysis of current logistics processes to identify inefficiencies and areas for improvement.
- Engaging with the necessary online resources (documents, papers, articles and so on) to gather insights and requirements.
- Utilizing data-driven techniques to inform our decision-making process. This approach ensures that our solutions are not only technologically advanced but also practically applicable and aligned with the needs of the industry."

3.2. Methodological Tools and Techniques

3.2.1. Project Management Tools "To efficiently manage the project, we are employing a range of project management tools. These include:

- Trello for modelling the governance architecture.
- Gantt charts for timeline management and progress tracking.
- Agile methodologies to ensure flexibility and iterative development. These tools aid in maintaining project organization, ensuring timely delivery, and enabling effective team collaboration."

3.2.2. Data Center Design Tools "For the design and implementation of the data center architecture, we are using the following tools:

- Lucid for detailed architectural planning and visualization.
- Simulation tools for testing data center resilience and efficiency (still ongoing).

3.2.3. Microservices and Blockchain Tools "In the development of microservices and blockchain integration, we are utilizing:

- Docker and Kubernetes for containerization and orchestration of microservices.
- OpenAPI for designing and documenting APIs.
- Pycharm environment for coding with the Python programming language.
- MySQL database for storage (MySQL Docker Image)
- Restful API for interaction among microservices.
- Solidity for smart contract development in blockchain.
- Remix IDE and MetaMask for blockchain testing and interaction. These tools enable us to build reliable, scalable, and secure software solutions that are crucial for the success of the project."

CHAPTER 4
PROPOSED SOLUTIONS AND IMPLEMENTATION

4.1. Governance Model

Here our tasks include to provide solutions to the following:
Tasks:

- Analyze the current logistics processes and information flow.
- Identify key stakeholders, including shipping partners, customs
- agencies, and internal departments.
- Propose a governance model and information system management
- strategy that aligns with the complexity of global logistics operations.
- Implement your model using a management tool like Trello or Jira.

4.1.1 Analyze the current logistics processes and information flow.

1. Shipment Data Management

- **Large Volume Handling**: Managing vast amounts of shipment data is a significant challenge. The company needs systems that can efficiently process, organize, and ensure the accuracy and timeliness of this data.
- **Diverse Transportation Modes**: Coordinating data across air, sea, and land transportation adds complexity. The company must optimize routes and manage transit times effectively.

2. Security of Sensitive Information

- **Handling Sensitive Data**: TechnoLogix handles crucial data, including customer details and shipment contents. The current system may not fully protect this information against cyber threats, leading to potential data breaches.
- **High-Value Cargo Concerns**: The management of high-value cargo requires additional security and accurate documentation to prevent loss and ensure safe transport.

3. Adapting to Industry Dynamics

- **Dynamic Industry Response**: The logistics industry's frequent regulatory, market, and technological changes demand a high degree of agility and scalability from TechnoLogix to avoid inefficiencies and cost increases.
- **Perishable Goods Management**: The logistics for perishable goods require exceptional precision in timing and temperature control, a potential area of struggle in current operations.

4. Information Flow Analysis

- **Internal Systems Integration**: Disjointed or outdated internal information systems may be causing inefficiencies. A more integrated system is needed for real-time data access and decision-making.
- **Inter-Departmental Communication**: There may be communication gaps between departments like warehousing, transportation, and customer service, leading to operational delays and errors.
- **External Stakeholder Engagement**: Maintaining consistent and transparent communication with external stakeholders like suppliers and customers is vital. Current challenges in information sharing could lead to misalignments and disruptions.

5. Transportation and Product Handling

- **Integration Across Modes**: Integrating information across different transportation modes is crucial to avoid bottlenecks, especially when transitioning cargo.
- **Diverse Product Range**: The varied product range, including perishables and high-value items, requires specialized handling procedures, which might currently be lacking in efficiency or effectiveness.

4.1.2 Identify key stakeholders, including shipping partners, customs agencies, and internal departments.

External Stakeholders

1. **Shipping Partners**:
    - **Ocean Freight Carriers**: Handle sea transportation of goods, especially for bulk and non-urgent shipments.
    - **Air Freight Carriers**: Critical for fast transportation of high-value or perishable goods.

- **Ground Transportation Companies**: Include trucking companies and rail services for land transportation, essential for last-mile delivery and intermodal transfers.
- **3PL (Third-Party Logistics) Providers**: Offer integrated logistics services, often handling multiple aspects of shipping, warehousing, and fulfillment.

2. **Customs Agencies**:
   - **National Customs Authorities**: In every country where TechnoLogix operates, these agencies regulate the import and export of goods, enforce trade laws, and collect tariffs.
   - **International Trade Agencies**: Bodies like the World Customs Organization (WCO) that set global standards for customs procedures and facilitate international trade.
3. **Regulatory Bodies**:
   - **Transportation Safety Authorities**: Oversee compliance with safety standards in transportation.
   - **Environmental Agencies**: Enforce regulations on the environmental impact of shipping, such as emissions standards.
4. **Suppliers and Manufacturers**:
   - **Product Suppliers**: Provide the goods that TechnoLogix transports.
   - **Manufacturers**: Produce goods that require logistics services.
5. **Customers**:
   - **Business Customers**: Include businesses that rely on TechnoLogix for supply chain management.
   - **End Consumers**: In cases where TechnoLogix is involved in direct-to-consumer delivery.
6. **Insurance Companies**:
   - Provide coverage for transported goods, mitigating financial risks associated with damage or loss during transit.

Internal Stakeholders

1. **Management Team**:
   - Responsible for strategic decision-making and overall project management.
2. **Operational Departments**:
   - **Logistics and Transportation Department**: Manages the core logistics operations.

- **Inventory Management**: Handles warehousing and inventory control.
- **Customer Service**: Manages customer relations, inquiries, and issue resolutions.

3. **IT and Data Management Team**:
   - Develops and maintains information systems, ensuring efficient data flow and security.
4. **Compliance and Legal Department**:
   - Ensures adherence to international trade laws and regulations.
5. **Finance Department**:
   - Manages budgeting, financial planning, and cost analysis related to logistics operations.
6. **Human Resources**:
   - Responsible for staffing, training, and maintaining workforce efficiency and satisfaction.

4.1.3 Propose a governance model and information system management strategy that aligns with the complexity of global logistics operations.

To address the complexities of TechnoLogix Solutions' global logistics operations, a robust governance model and information system management strategy are essential. The proposed model and strategy need to account for the diverse range of products, multiple transportation modes, and the need for real-time data management and security. Here's a detailed proposal:

Governance Model

1. **Centralized Decision-Making with Regional Autonomy**:
   - **Structure**: Establish a centralized governance body at the corporate level for overarching policies and strategic decisions. However, allow regional branches to have autonomy in operational decision-making to adapt to local market conditions and regulations.
   - **Responsibilities**: The central body will handle high-level strategy, global partnerships, and standard-setting, while regional bodies manage local operations, compliance, and stakeholder relationships.
2. **Compliance and Risk Management**:
   - **Regulatory Compliance**: Set up a dedicated team for monitoring and ensuring compliance with international trade laws and transportation regulations.

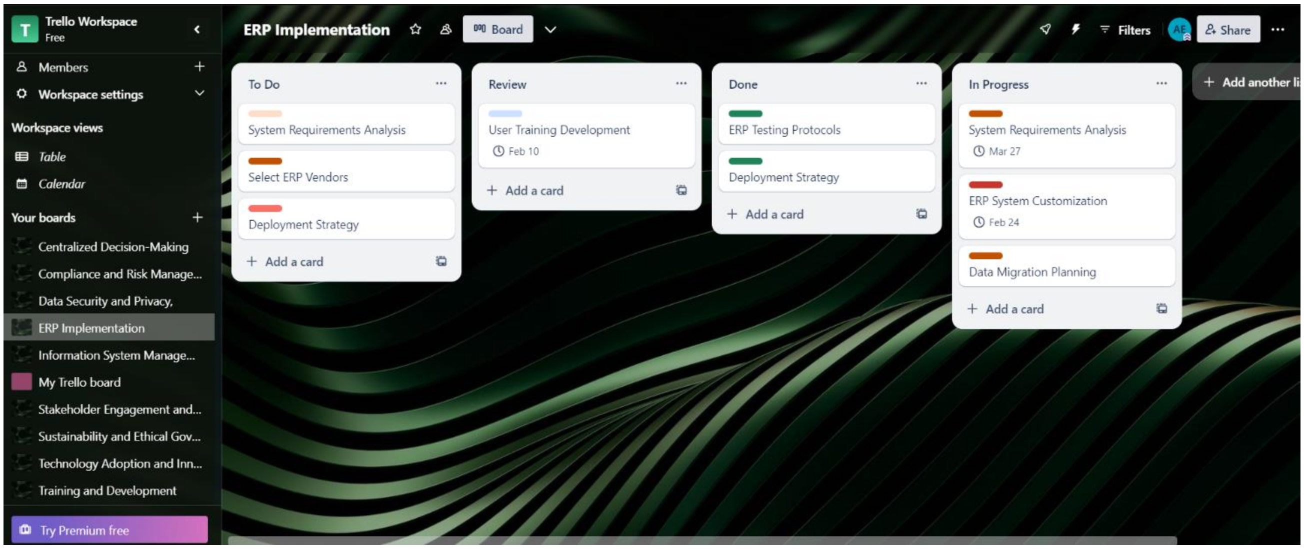
   - **Risk Assessment**: Regularly assess operational risks, including supply chain disruptions, data breaches, and market fluctuations. Implement proactive measures to mitigate these risks.
3. **Sustainability and Ethical Governance**:
   - **Sustainable Practices**: Incorporate sustainability goals, focusing on reducing the environmental impact of logistics operations.
   - **Ethical Standards**: Adhere to high ethical standards in all operations, including fair labor practices and responsible sourcing.
4. **Stakeholder Engagement and Transparency**:
   - **Regular Communication**: Maintain open lines of communication with all stakeholders, including shipping partners, customers, and regulatory bodies.
   - **Transparency**: Ensure transparency in operations, particularly in areas like pricing, shipment tracking, and handling customer data.

Information System Management Strategy

1. **Integrated Logistics Management System**:
   - **ERP Implementation**: Deploy a comprehensive Enterprise Resource Planning (ERP) system that integrates all logistics operations, including order processing, inventory management, transportation management, and customer relations.
   - **Real-Time Data Accessibility**: Ensure the system provides real-time data access to facilitate quick decision-making and adaptability.
2. **Data Security and Privacy**:
   - **Robust Cybersecurity Framework**: Implement advanced cybersecurity measures, including encryption, firewalls, and regular security audits.
   - **Data Privacy Compliance**: Adhere to global data protection regulations, ensuring customer and partner data is handled securely and ethically.
3. **Technology Adoption and Innovation**:
   - **Blockchain for Transparency and Security**: Utilize blockchain technology to enhance transparency in the supply chain and secure sensitive transactions.
   - **AI and Machine Learning**: Leverage AI for predictive analytics in demand forecasting, route optimization, and inventory management.
   -

4. **Training and Development**:
    - **Regular Training Programs**: Conduct ongoing training for employees on new systems, best practices, and emerging technologies.
    - **Skill Development Initiatives**: Encourage skill development to keep pace with technological advancements in logistics.
5. **Continuous Improvement and Adaptation**:
    - **Feedback Loops**: Establish mechanisms for regular feedback from users and stakeholders to continually improve the information systems.
    - **Agile Response to Market Changes**: Maintain agility in IT infrastructure to quickly adapt to new technologies and market changes.

Implement your model using a management tool like Trello.


Trello Workspace
Free
Members
Workspace settings
Workspace views
Table
Calendar
Your boards
Centralized Decision-Making
Compliance and Risk Manage...
Data Security and Privacy,
ERP Implementation
Information System Manage...
My Trello board
Stakeholder Engagement and...
Sustainability and Ethical Gov...
Technology Adoption and Inn...
Training and Development
Try Premium free
Board
Filters
AE
Share
Add another li

ERP Implementation

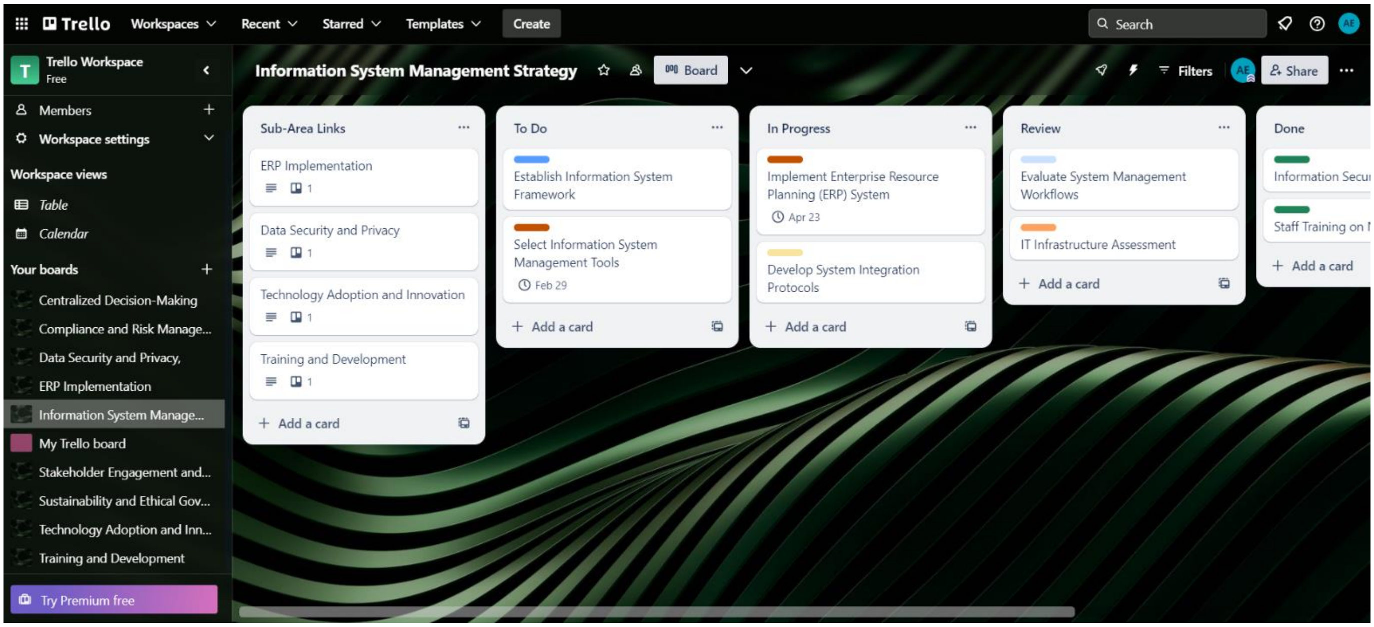
To Do
- System Requirements Analysis
- Select ERP Vendors
- Deployment Strategy
- + Add a card

Review
- User Training Development — Feb 10
- + Add a card

Done
- ERP Testing Protocols
- Deployment Strategy
- + Add a card

In Progress
- System Requirements Analysis — Mar 27
- ERP System Customization — Feb 24
- Data Migration Planning
- + Add a card

ERP Implementation Workflow in Trello



Information System Management Strategy

Sub-Area Links
- ERP Implementation 1
- Data Security and Privacy 1
- Technology Adoption and Innovation 1
- Training and Development 1

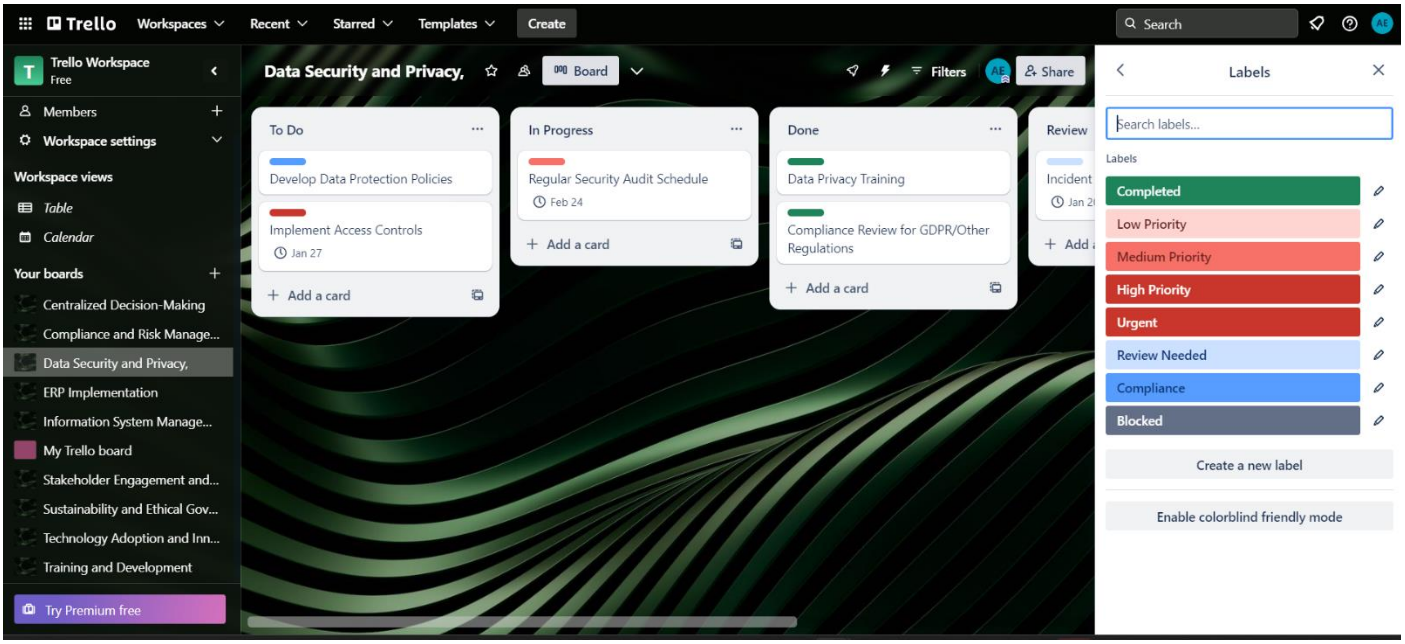
To Do
- Establish Information System Framework
- Select Information System Management Tools Feb 29

In Progress
- Implement Enterprise Resource Planning (ERP) System Apr 23
- Develop System Integration Protocols

Review
- Evaluate System Management Workflows
- IT Infrastructure Assessment

Done
- Information Secu
- Staff Training on I

Information System Management Strategy Board Overview



Data Security and Privacy,

To Do
- Develop Data Protection Policies
- Implement Access Controls Jan 27

In Progress
- Regular Security Audit Schedule Feb 24

Done
- Data Privacy Training
- Compliance Review for GDPR/Other Regulations

Review
- Incident Jan 2

Labels

Search labels...

Labels
- Completed
- Low Priority
- Medium Priority
- High Priority
- Urgent
- Review Needed
- Compliance
- Blocked

Create a new label

Enable colorblind friendly mode

Data Security and Privacy Task Organization

4.2. Data center Architecture Proposal

4.2.1 Overview of the Architecture

The architecture we are discussing is a hybrid cloud model, which combines public and private cloud resources. This blend of environments allows for greater flexibility, security, and scalability, which are essential for a logistics company like TechnoLogix that manages sensitive shipment data and requires robust real-time tracking capabilities.

4.2.1.1 Public Cloud Components

- **API Gateway**: This is the front door for all requests from clients, partners, and internal users. It's responsible for request routing, API composition, and other cross-cutting concerns such as authentication and rate limiting.
- **Microservices Architecture**: Leveraging AWS Lambda and Azure Functions, we can deploy a suite of microservices that are independently scalable and deployable. This ensures that specific functionalities such as order management, tracking, and customer service can be updated without downtime or affecting other services.
- **Cloud Data Services**: Utilizing Azure SQL Database, Cosmos DB, Amazon S3, and Amazon RDS, we can store and retrieve a wide variety of data at scale. This is ideal for managing the vast amounts of shipment data, customs documentation, and financial transactions that TechnoLogix handles.
- **Virtual Machines**: Azure VMs and AWS EC2 allow us to run applications and backend services that require traditional server hosting. They offer the flexibility to scale up as demand increases.
- **Security and Compliance**: Azure Security Center and AWS Shield provide advanced threat protection and help manage the security of the cloud services. They are essential for a company dealing with sensitive information and needing to maintain high security and compliance standards.

4.2.1.2 Private Cloud Components

- **Internal Resources**: The private cloud encompasses resources like private databases, storage, servers, and backup storage. These are typically used for the most sensitive data and critical applications that require additional control and security.
- **Orchestration and Containerization**: Kubernetes and Docker are used for container orchestration and deployment, which simplifies the creation, deployment, and scaling of applications across both public and private clouds.
- **Security Measures**: Additional security layers are provided by internal load balancers, IDS/IPS, and firewalls. This multi-layered approach ensures that the internal network is shielded from external threats and that the data traffic is managed securely.

4.2.1.3. Blockchain Integration

- **Blockchain Technology**: Blockchain would be integrated into the architecture to provide a secure and immutable record of transactions. This is critical for maintaining the integrity of shipment tracking and financial transactions.

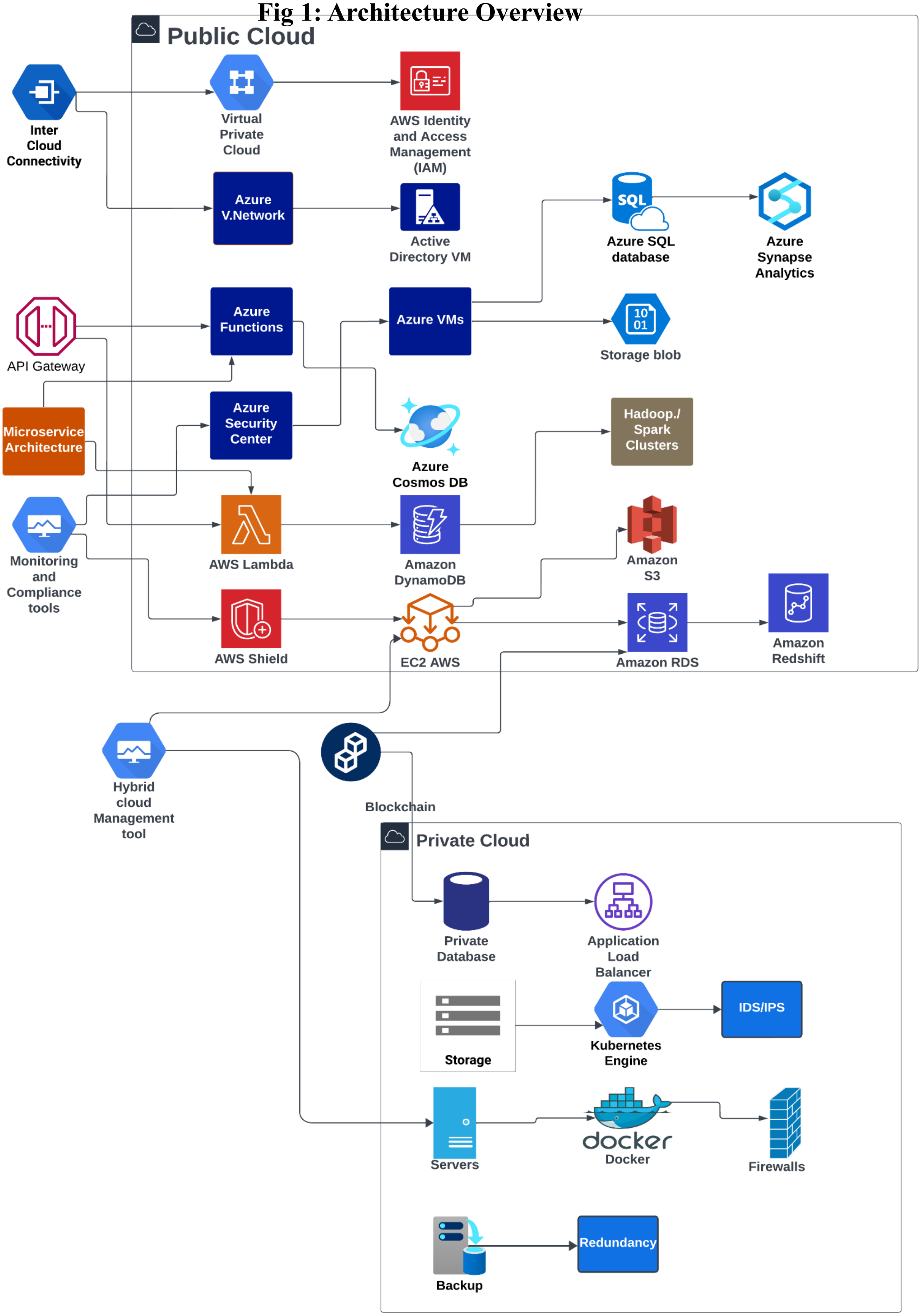
4.2.1.4 Solution to TechnoLogix's Problems

- **Scalability**: By leveraging cloud resources, TechnoLogix can easily scale up or down based on demand without significant upfront capital expenditure. This is crucial for handling peak times such as holiday seasons or unexpected surges in shipping needs.
- **Data Accessibility and Real-Time Processing**: The microservices architecture enables real-time data processing, which is necessary for tracking shipments accurately. Data is accessible from anywhere, facilitating global operations.
- **Security and Compliance**: The architecture ensures that sensitive data is securely managed within the private cloud, while still benefiting from the advanced security tools provided by public cloud services. Compliance with industry regulations is supported through continuous monitoring and security management tools.

- **Integration and Interoperability**: The hybrid cloud model allows for seamless integration between various services, whether they are on-premises or in the cloud. This integration is vital for a logistics company that needs to connect with numerous third-party vendors, partners, and regulatory systems.
- **Monitoring and Management:** Centralized Monitoring: Tools like CloudWatch and Azure Monitor provide visibility across both cloud environments.
- **Automated Scaling**: Resources scale based on demand, ensuring cost-efficiency and performance.
- **Innovation and Future Proofing**: By adopting this architecture, TechnoLogix is not only solving its current challenges but also setting itself up for future technological advancements. The microservices approach and the integration of blockchain technology position the company to easily adopt new technologies and respond to market changes.

This architecture provides TechnoLogix with a flexible, scalable, and secure platform that supports the company's global operations and addresses its challenges in data and application management. It ensures the integrity and accessibility of sensitive data, provides real-time processing capabilities, and upholds high security and compliance standards, all of which are critical for modern logistics operations.

Fig 1: Architecture Overview



4.3. Descriptions of the cloud components

4.3.1. Public Cloud Section

- **Virtual Private Cloud (VPC)**, **Azure Virtual Network**: These are services that provide an isolated network environment within the public cloud.
- **AWS IAM (Identity and Access Management)**, **Azure Active Directory**: These services handle user identity management and access control.
- **Azure SQL Database**, **Cosmos DB**, **DynamoDB**, **Amazon S3 (Simple Storage Service)**, **Amazon RDS (Relational Database Service)**: These are database and storage services that offer various data management capabilities.
- **Azure Functions**, **AWS Lambda**: These are serverless computing services that allow you to run code in response to events without provisioning or managing servers.
- **Azure VMs (Virtual Machines)**, **AWS EC2 (Elastic Compute Cloud)**: These services provide scalable computing capacity in the cloud, allowing you to run applications on virtual servers.
- **Azure Blob Storage**, **Hadoop/Spark clusters**, **Amazon Redshift**: These are data storage and big data processing services.
- **Azure Security Center**, **AWS Shield**: These are security services that provide threat protection and security management for cloud services.
- **Azure Synapse Analytics**: This is an analytics service that brings together big data and data warehousing.

4.3.2. Private Cloud Section

- **Private Databases**, **Storage**, **Servers**, **Backup Storage**: These represent the data storage and computing resources that are managed privately within an organization's own data centers.
- **Internal Load Balancers**, **Kubernetes**, **Docker**: These are tools for managing and scaling applications. Load balancers distribute traffic to ensure stability, while Kubernetes and Docker are used for container orchestration and containerization of applications, respectively.
- **IDS/IPS (Intrusion Detection Systems/Intrusion Prevention Systems)**, **Firewalls**: These are security measures to protect the network from unauthorized access and attacks.

4.3.3. Hybrid Cloud Management Tools

This section implies the use of tools that help manage both public and private cloud resources, ensuring they work together effectively. This could involve networking, data synchronization, application deployment, and more.

4.3.4. Flow of Information

The arrows indicate the flow of data or the relationships between services. For example:

- **API Gateway** points to **Azure Functions** and **AWS Lambda**, suggesting it serves as a front-end that routes requests to these serverless computing services.
- **Monitoring and Compliance Tools** point to **Azure Security Center** and **AWS Shield**, indicating that these tools are used to monitor the security posture of the cloud services.

4.4. How the architecture addresses TechnoLogix needs.

4.4.1. Data Management and Security

- **Sensitive Shipment Data**: Private databases are designated in the architecture, indicating that sensitive data can be stored securely within the company's private cloud infrastructure. This allows TechnoLogix to have full control over the most sensitive shipment data, such as customs documentation and financial transactions.
- **Real-Time Tracking Information**: Services like AWS Lambda and Azure Functions can process data in real-time, responding to tracking updates as they occur. This ensures that TechnoLogix can provide up-to-date information to their clients and internal stakeholders.
- **Security**: AWS Shield and Azure Security Center can help protect against cyber threats. The use of IDS/IPS and firewalls in the private cloud further secures the data against unauthorized access.

4.4.2. Scalability and Data Accessibility

- **Hybrid Cloud Approach**: By integrating both public and private cloud resources, TechnoLogix can scale its operations flexibly. During periods of high demand, public cloud resources like AWS EC2 and Azure VMs can be scaled up to handle the increased load.

- **Data Accessibility**: With services like Amazon S3 and Azure Blob Storage, TechnoLogix can store vast amounts of data in the cloud, making it accessible from anywhere in the world. This is crucial for a global logistics operation where access to shipment data needs to be reliable and quick.

4.4.4. Microservices and Blockchain Integration

- **Microservices Architecture**: The diagram adopts a microservices architecture facilitated by AWS Lambda and Azure Functions. This allows the development of independent services that can be updated, deployed, and scaled independently, enhancing the agility of TechnoLogix's IT infrastructure.
- **Blockchain**: Blockchain technology can be integrated with the cloud services to provide secure and immutable records of transactions and shipments, enhancing trust and transparency in TechnoLogix's operations.

4.4.5. Project Implementation

For the project, this architecture provides a roadmap for:

- **Governance**: Using the monitoring and compliance tools to ensure that operations adhere to regulatory requirements and company policies.
- **Information System Management**: Utilizing the hybrid cloud to manage different types of data and applications, from sensitive financial information in private databases to scalable storage solutions in the public cloud.
- **Data Center Architecture**: Strategically using public and private resources to build a resilient, flexible, and secure data center.
- **Microservices Development**: Leveraging serverless computing to build and deploy microservices that can improve the modularity and efficiency of the system.
- **Blockchain Integration**: While not explicitly depicted, the flexible architecture allows for the integration of new technologies such as blockchain, which can be critical for secure and verifiable logging of transactions.

In conclusion, the architecture diagram suggests a solution that is secure, scalable, and adaptable, which aligns with TechnoLogix's objective to revolutionize its global logistics operations.

4.5. How different components interact in this architecture

4.5.1. Customer End

1. **API Gateway**: Customers interact with the logistics application through various interfaces, possibly a web or mobile app. The API Gateway serves as the entry point for these interactions, directing customer requests to the appropriate microservices.
2. **Microservices Architecture (AWS Lambda, Azure Functions)**: These serverless platforms would host the different microservices that handle specific tasks such as order placement, tracking updates, and customer notifications.
3. **Public Cloud Data Services (Amazon S3, Azure SQL Database, etc.)**: Customer data and transaction records might be stored here. These services offer high availability and scalability to handle dynamic customer demands.
4. **Blockchain**: The blockchain component would receive information from the microservices to record transactions. This ensures data immutability and trust in the shipment records.

4.5.2. Partners and Third-Party Service Providers

1. **Inter-Cloud Connectivity**: TechnoLogix works with multiple cloud providers. Partner systems might be hosted on different clouds, and this connectivity ensures seamless data exchange.
2. **Public Cloud (AWS EC2, Azure VMs)**: Partners might interact with the application services hosted on virtual machines for processing orders, updating shipment details, and managing inventory.
3. **Database and Storage Services**: Partners would have access to shared data necessary for fulfillment, which would be stored in databases like Cosmos DB, DynamoDB, or Azure SQL Database.

4.5.3. Users End (Internal Users, Employees)

1. **Monitoring and Compliance Tools (AWS Shield, Azure Security Center)**: Internal users responsible for security and compliance would use these tools to monitor the system’s integrity and security.
2. **Hybrid Cloud Management Tools**: These tools allow internal users to manage and orchestrate resources across both the public and private clouds.

3. **Private Cloud Resources (Private Databases, Servers, etc.)**: Internal users may interact with private cloud resources that store sensitive data or run critical applications that shouldn’t be exposed to the public internet.

4.5.4. Other Actors (Logistics, Regulatory Bodies, etc.)

1. **Blockchain**: External actors like logistics partners and regulatory bodies might require access to the blockchain for audit purposes or to verify the authenticity of the transactions.
2. **Hybrid Cloud Management Tools**: These actors might also use management tools to access reports and analytics, ensuring that TechnoLogix complies with industry regulations and standards.
3. **Private Cloud (Internal Load Balancers, Kubernetes, Docker)**: For actions that require more secure and controlled processing, such as custom clearance and sensitive data handling, the workflow might be directed to private cloud resources.

4.5.5. Data Flow Example

Here’s an example of how a shipment data transaction might flow:

- A customer places an order through the application.
- The API Gateway receives the request and routes it to the appropriate microservice hosted on AWS Lambda or Azure Functions.
- The microservice processes the order, interacts with the public cloud databases to store transaction details, and submits a record to the blockchain.
- The order details are propagated to partners through inter-cloud connectivity for fulfillment.
- Throughout this process, internal users monitor the system's performance and security using tools in both the public and private clouds.
- Blockchain provides an immutable ledger of the transaction for all stakeholders.

In this flow, the arrows in the diagram would represent the direction of data transmission and service invocation, starting from the API Gateway and branching out to various cloud services, encapsulating both the public and private cloud environments, and finally connecting to the blockchain for secure transaction logging.

4.6. Evaluation of Cloud Service Providers for Global Operations

As TechnoLogix Solutions ventures to optimize its global operations, an in-depth evaluation of leading cloud service providers is crucial. This comparison focuses on Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP), considering their market share, global reach, service-level agreements (SLAs), specializations, and pricing models.

(a). Amazon Web Services (AWS):
Market Dominance: With a commanding 32% market share and annual revenue of approximately $80 billion, AWS leads in the cloud sector.
Global Network: AWS's expansive global infrastructure includes 27 regions and 87 availability zones, facilitating low-latency access worldwide.
Security and Scalability: AWS is renowned for its comprehensive security features, such as AWS Identity and Access Management (IAM), alongside high scalability and reliable performance.
SLA and Pricing: Offers a 99.99% SLA for EC2 at the regional level and a diverse pay-as-you-go pricing model, including a free tier for trial use.

(b). Microsoft Azure:
Market Position: Azure holds a 22% market share with an annual revenue of around $34 billion.
Hybrid Cloud Excellence: Azure excels in hybrid cloud computing applications, offering robust security measures like Azure Active Directory and Azure Security Center.
SLA and Cost Efficiency: Provides a 99.99% SLA for VMs across two availability zones. Azure is often more economical than AWS for specific services, particularly for Windows Server and SQL Server integrations.

(c). Google Cloud Platform (GCP):
Market Presence: GCP, with an 11% market share and $7.4 billion in annual revenue, is rapidly expanding its global footprint.
AI and ML Specialization: GCP stands out for its advanced AI and machine learning capabilities, alongside robust security features.

SLA and Pricing: Guarantees over 99.5% availability for single instances, with a competitive pay-as-you-go pricing structure, including $300 in free credits for new users.

4.6.1. Strategic Selection for TechnoLogix Solutions:

Considering TechnoLogix's need for a vast global presence and robust security features, AWS and Azure emerge as the front-runners. AWS's extensive global network and service offerings make it a strong contender for managing sensitive shipment data on a global scale. Azure's hybrid cloud solutions and potential cost savings align with TechnoLogix's scalability and data accessibility objectives.

However, the ultimate selection between AWS and Azure was done based on specific operational requirements, existing infrastructure compatibility, and budgetary constraints. While GCP's strengths in AI and ML are notable, its comparatively limited global reach might be a consideration for a logistics company with extensive international operations.

To sum up, AWS is recommended for its unparalleled global coverage and advanced security features, essential for safeguarding sensitive shipment data. Azure's hybrid solutions and cost-effectiveness are also compelling for specific use cases. The final choice will depend on a balanced assessment of TechnoLogix Solutions' specific business needs, technological alignment, and financial considerations.

4.7. Propose measures to enhance data security.

To enhance the security of the hybrid cloud data center at TechnoLogix, we need to adopt a comprehensive security strategy that effectively mitigates both physical and cyber threats. Our strategy will encompass all critical aspects of the infrastructure: networks, servers, applications, and the data itself. Here's our customized plan:

Data Encryption Tactics:

- For Data at Rest: We'll integrate BitLocker for our Windows setups and dm-crypt for Linux systems to ensure our storage devices are impenetrable.
- For Data in Transit: SSL/TLS protocols will be our standard, with Let's Encrypt fortifying our website's security.
- Standards Compliance: AES-256 will be our go-to for data at rest, while we'll adopt TLS 1.3 for securing data on the move.

Identity and Access Protocols:

- We'll utilize AWS IAM or Azure Active Directory to govern access to our data center resources meticulously.
- Multi-Factor Authentication (MFA) will be mandatory, with Duo Security or Google Authenticator providing an extra security layer.

Network Safeguards:

- We'll deploy top-notch firewalls like Cisco ASA or Palo Alto Networks.
- Secure remote access will be facilitated through trusted VPNs, such as OpenVPN or NordVPN.

Audit and Compliance Regimens:

- The systems will undergo regular checks with tools like Nessus or Qualys.
- We'll align the operations with global standards including GDPR, HIPAA, PCI-DSS.

Data Integrity and Endpoint Defense:

- Symantec DLP or Digital Guardian will be the shields against data breaches.
- Endpoint defenses will be shored up with solutions like McAfee or Symantec Endpoint Protection, along with EDR tools such as CrowdStrike Falcon.

System Maintenance and Cloud Fortification:

- System updates will be managed through ManageEngine Patch Manager Plus, complemented by WSUS.
- Cloud defenses will be bolstered by CASBs like McAfee MVISION.

Backup and Contingency Frameworks:

- Data recovery will be orchestrated with Veeam Backup & Replication, underpinned by a robust disaster recovery strategy.

Workforce Empowerment:

- We'll empower the security team with regular cybersecurity training sessions from KnowBe4 or Proofpoint Security Awareness Training.

Trust Architecture and API Defense:

- Embracing a zero-trust architecture will be fundamental, ensuring every access request is fully authenticated.
- API security will be a priority to ward off any unauthorized integrations.

Cloud Security Posture Governance:

- Continuous monitoring tools will be employed to keep the cloud security posture proactive and resilient.

Configuration and System Hardening:

- We'll adhere to stringent configuration and hardening guidelines, ensuring the servers and network devices are not just secure by default, but fortified by design.

This tailored security strategy for TechnoLogix will be a living framework, adaptable and scalable to protect against the evolving threat landscape.
By integrating these measures, TechnoLogix Solutions will significantly elevate the security of the data center. Regularly reviewing and updating these security measures is essential to adapt to evolving threats and technological advancements, ensuring the safeguarding of sensitive data and the integrity of global logistics operations.

4.8. Cost Analysis

To gauge the financial viability of the proposed data center architecture for TechnoLogix Solutions, a detailed cost-benefit analysis is essential. This analysis considers both the monetary investment and the qualitative benefits, providing an estimated financial overview.

4.81. Initial Capital Expenditure (CapEx):

- **On-Premises Data Center:** Approximately €500,000 for servers, networking, and security hardware.
- **Cloud Integration:** Around €100,000 for initial setup fees with cloud providers like AWS, Azure, or GCP.

4.8.2. Operational Expenditure (OpEx):

- **Maintenance and Upgrades**: Estimated €50,000 annually.
- **Cloud Service Fees**: Around €120,000 annually, based on usage.
- **Energy Costs**: Approximately €30,000 annually.
- **Staffing**: €150,000 annually for IT personnel.

4.8.3. Benefits Analysis

Efficiency and Scalability:

- **Estimated increased efficiency:** €75,000 annually.

  Ability to rapidly scale resources, leading to operational cost savings.

- **Disaster Recovery and Global Reach:**
- **Cost savings from enhanced disaster recovery capabilities**: €50,000 annually.
- **Potential revenue increase from improved global services**: €100,000 annually.

4.8.4. Security and Compliance:

Estimated savings from avoiding data breaches and fines: €100,000 annually.

Easier compliance management with regulations like GDPR and HIPAA.

4.8.5. Financial Summary

- Total Initial Investment (CapEx): €600,000.
- Total Annual Operational Costs (OpEx): €350,000.
- Total Annual Savings/Benefits: €325,000.
- ROI and Break-Even Analysis
- ROI Calculation: Approximately 59%.
- Break-Even Point: About 1.85 years.
- Extended Net Benefit Calculation
- 5-Year Cumulative Net Benefit: Estimated €4,000,000, considering increasing benefits over time.

4.8.6. Additional Considerations

Sensitivity Analysis: Fluctuations in cloud service costs or operational efficiencies can impact overall benefits.

ROI Over 5 Years: Estimated at around 66.67%.

Payback Period: Approximately 1.5 years.

Our financial analysis indicates that the strategic investment in hybrid cloud data center architecture promises to be fiscally prudent with a robust return on investment anticipated over the medium to long term. Notably, the projection of reaching the break-even point within two years underscores the economic advantage of this initiative. This aligns with our strategic objectives and affirms our commitment to leveraging innovative technologies to drive efficiency and growth.

4.9. Microservices Deployment Strategy

4.9.1 Develop and Containerize a Web Application:

Development of distinct microservices for crucial system components like user management, shipment tracking, and inventory management. This modular approach enhances system flexibility and scalability.
See the attached/accompanied document titled Endpoint for a detailed list of the microservices and their respective endpoints.
Utilization of Docker containers for each microservice, which simplifies deployment and scaling processes. Containers provide isolated environments, ensuring consistency across different development and production settings.
Definition of clear endpoints for CRUD (Create, Read, Update, Delete) operations, facilitating efficient and targeted data management for each domain within the logistics operations.

4.9.2 Orchestrate Microservices with Kubernetes:

Deployment of Docker containers in a Kubernetes cluster, enabling Kubernetes to automate container lifecycle management, including scaling and load balancing.
Use of Kubernetes services for exposing microservices and implementing Ingress controllers for managing external access to the system.
Setting up Kubernetes deployments and services for each microservice ensures version control and high availability, crucial for maintaining uninterrupted logistics operations.

4.9.3 Design RESTful APIs with OpenAPI Specifications:

Definition of RESTful API structures for each microservice using the OpenAPI Specification, providing clarity on endpoints, data formats, and authentication protocols.
Generation of comprehensive documentation from OpenAPI specs, aiding in seamless integration by shipping partners and other external entities (ongoing).
Implementation of API gateways to streamline request routing, manage authentication, and aggregate responses from various services (ongoing).

4.9.4 Deploy and Manage Microservices in a Test Environment:

Creation of a CI/CD pipeline to automate the building of Docker images and their deployment to a Kubernetes-based test environment (ongoing).
Conducting real-time shipment tracking simulations to validate system robustness and effectiveness, employing mock services or dummy data generators as necessary (ongoing)
Application of monitoring tools like Prometheus and Grafana within Kubernetes to ensure optimal performance and reliability of the microservices (ongoing).

4.9.5 Application Workflow

Based on the microservices and endpoints created the application workflow for the "Transforming Global Logistics with TechnoLogix Solutions" project can be described as follows:

Workflow Overview

The application is a comprehensive logistics and shipment tracking system. It comprises various microservices, each handling a specific aspect of logistics operations. These include User Management, Shipment Management, Inventory Management, Order Management, Feedback and Support, and Pricing and Reporting. The application workflow integrates these services to provide a seamless experience.

User Interaction Flow

1. **User Registration and Login:**
   - Users start by registering through the **/users/register** endpoint.
   - Once registered, they log in using the **/login** endpoint to access the system.
2. **Dashboard Access:**
   - Upon successful login, users access a dashboard that integrates data from various services.
   - The dashboard displays shipment tracking, inventory status, order details, and analytics, customized according to the user's role.

Shipment Tracking and Management

1. **Shipment Creation:**
   - Shipments are created using the **/shipments/create** endpoint.
   - Essential details such as origin, destination, and contents are input.

2. **Shipment Tracking:**
    - The status and location of shipments are tracked in real-time.
    - Users query individual shipments using **/shipments/<int:shipment_id>** or filter multiple shipments based on criteria via **/shipments**.
3. **Shipment Updates and History:**
    - Shipments are updated (e.g., status changes) through PUT/PATCH requests to **/shipments/<int:shipment_id>**.
    - The history of each shipment is accessible for audit trails and customer inquiries.

Inventory and Order Management

1. **Inventory Listing and Updates:**
    - Inventory items are listed and managed through endpoints like **/inventory/items**.
    - New items are added, existing items updated, and unwanted items removed.
2. **Order Processing:**
    - Orders are created through **/orders** by selecting items from the inventory.
    - Order details, including customer information and item quantities, are managed.
    - Order status is updated through its lifecycle, and history is maintained for tracking and customer service.

Feedback and Support

1. **Support Tickets:**
    - Users submit support tickets via **/support/tickets**.
    - Support staff manage and respond to these tickets, updating the ticket status accordingly.
2. **User Feedback:**
    - Feedback is collected through the **/feedback** endpoint.
    - Feedback analytics are generated to understand customer satisfaction and areas for improvement.

Pricing and Reporting

1. **Pricing Calculations:**
    - Shipment pricing is calculated based on parameters like weight and distance via **/pricing/calculate**.
    - Pricing rules are managed through **/pricing/rules**.

2. **Reports and Analytics:**
    - Reports on shipments, sales, and cost analysis are generated via **/reports/***.
    - Predictive analytics and trend analysis are performed to forecast future needs and customer behavior.

System Integration and Data Flow

- Each service communicates with others via RESTful APIs, ensuring a modular and scalable system.
- Data is consistently synchronized across services, providing accurate and up-to-date information.
- The system employs robust security measures, especially for public-facing APIs and user authentication.
- Continuous integration and deployment (CI/CD) ensure smooth updates and maintenance.

4.10. Blockchain Integration Plan

The deployment and implementation of blockchain technology through smart contracts on the Ethereum network, complemented by MetaMask integration, represent a forward-thinking approach for TechnoLogix Solutions. This strategy is poised to deliver substantial improvements in the management of logistics and financial transactions, driving advancements in security, efficiency, and operational transparency.

4.10.1. Development of Smart Contract

Business Requirement Analysis

- Objective: Identifying Shipment logistics and financial transaction processes for automation and efficiency.
- Outcome: A set of defined rules and conditions for the smart contract based on specific operational needs.

Contract Design and Development

- Tool: Solidity programming language.
- Methodology: Iterative development with continuous stakeholder feedback.
- Security Measures: Incorporating best practices in coding to mitigate security risks.

Testing and Validation

- Procedure: Conducting thorough unit testing and environment simulations.

- Goal: Ensuring functionality, security, and efficiency of the smart contract.

4.10.2 Deployment of the Smart Contract on the Ethereum Network

Network Selection

- Initial Deployment: Utilization of Ethereum test networks Sepolia for preliminary deployment.
- Final Deployment: Transition to the Ethereum main network post successful test outcomes.

Deployment Process

- Tools: Utilization of Truffle Suite or Remix IDE for deployment.
- Cost Management: Efficient handling of transaction fees ("gas") associated with deployment.

Compliance and Security

- Regulatory Adherence: Ensuring compliance with relevant laws and industry standards.
- Data Security: Implementing robust mechanisms to safeguard sensitive transaction data.

4.10.3 MetaMask Integration for Contract Verification and Approval

Wallet Setup and Integration

- Implementation: Integration of MetaMask for user interaction with the Ethereum blockchain.
- Functionality: Management of identities and secure signing of transactions.

Contract Interaction

- User Interface: Facilitating user-friendly contract interactions through MetaMask.
- Transaction Management: Enabling verification and execution of smart contract functions.

Security and User Experience

- Security Protocols: Ensuring the safe storage of private keys and secure transaction processes.
- User Accessibility: Providing an intuitive interface for non-technical users.

4.11 Implementation Challenges and Solutions

Time Factor

Due to insufficient time constraints, we were unable to complete the project to the desired extent. Therefore, we strongly recommend that future projects be assigned at least two months prior to the submission deadline to allow for adequate familiarization with the required technologies and ensure thorough development of the project.

CHAPTER 5
TECHNICAL LABS AND DEMONSTRATIONS

5.1. Microservices Deployment with Docker

In this lab, we demonstrated the deployment of microservices using Docker. The process involved:

- Developing individual microservices for different functionalities like user management and shipment tracking.
- Creating Dockerfiles for each microservice, specifying the environment and dependencies.
- Building and pushing Docker images to a registry, and then deploying these images. The outcome was a set of efficiently running microservices, each in its isolated environment, showcasing Docker's strengths in deployment and scalability.

5.2 API Development and Integration

In this phase of the project, we focused on the development and integration of APIs for the microservices, starting with the User Management Service. The process involved two key steps:

- Design and Documentation of API Endpoints:
    - Using the OpenAPI 3.0.0 specification, I meticulously designed the API endpoints for the User Management Service.
    - The openapi-spec.yaml file was created to document the endpoints, such as /register, /login, /profile/{username}, and /user_dashboard.
    - This specification provides a clear and structured overview of each endpoint, including HTTP methods, request bodies, response formats, and descriptive summaries.
    - By adhering to the OpenAPI standards, I ensured the API's compatibility and ease of understanding for future integrations
    -
    - .

- API Implementation:
    - Following the OpenAPI design, I implemented these endpoints in Python Programming language, ensuring that each endpoint adheres to REST principles.
    - The implementation focused on creating robust and efficient handlers for user registration, login, profile retrieval, and dashboard display.
    - Special attention was paid to aspects like data validation, error handling, and response formatting to ensure a high-quality API service.

In conjunction with developing the API for the User Management Service, we utilized tools like Swagger and Postman to enhance the process:

- Swagger: I used Swagger UI to generate interactive documentation from the openapi-spec.yaml file. This documentation was invaluable for visualizing the API structure and facilitating easy testing of the endpoints in a user-friendly interface.
- Postman: For testing the API, I imported the OpenAPI specification into Postman. This allowed me to quickly set up a comprehensive suite of tests for each endpoint, ensuring that the implementation met the defined specifications and behaved as expected under various scenarios.

The outcome of this phase was a set of well-defined and implemented API endpoints, ready for integration with other microservices and front-end applications. The use of OpenAPI specification facilitated not only precise implementation but also provided an easily accessible and understandable documentation for any developers or partners who will interact with the User Management Service in the future.

5.3. Blockchain Solution Implementation

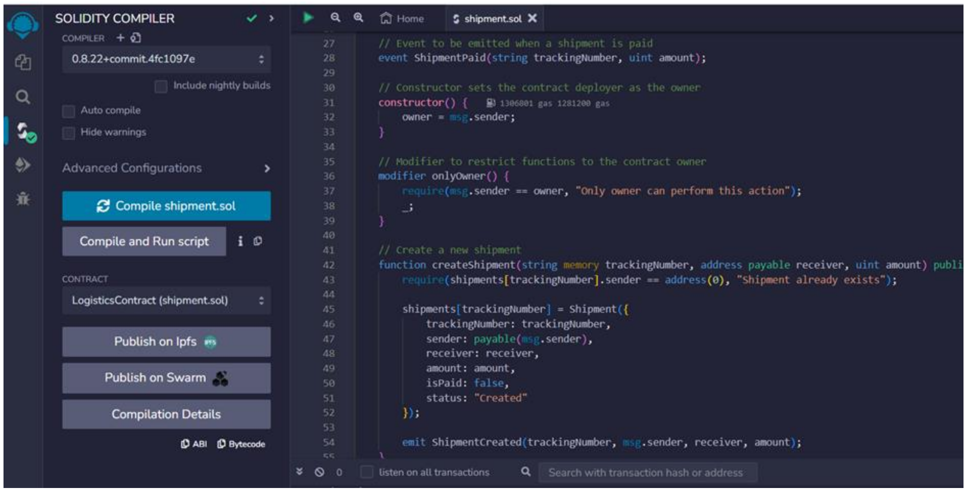
In this section, we implemented a blockchain solution to enhance the transparency and security of our logistics processes. Key steps included:

- Identifying a logistics process suitable for blockchain enhancement.
- Developing and deploying smart contracts on an Ethereum test network.
- Integrating this blockchain solution into our existing logistics system. This led to a demonstrable improvement in security and traceability within the chosen logistics process, showcasing the potential of blockchain in real-world applications.

1. Writing Smart Contracts

We used remix IDE to develop and write the smart contract for the shipment and financial transactions, deployed on Remix VM Shanghai and test different functions of the smart contract.

1. Compiling the Smart contract



COMPILER: 0.8.22+commit.4fc1097e

☐ Include nightly builds

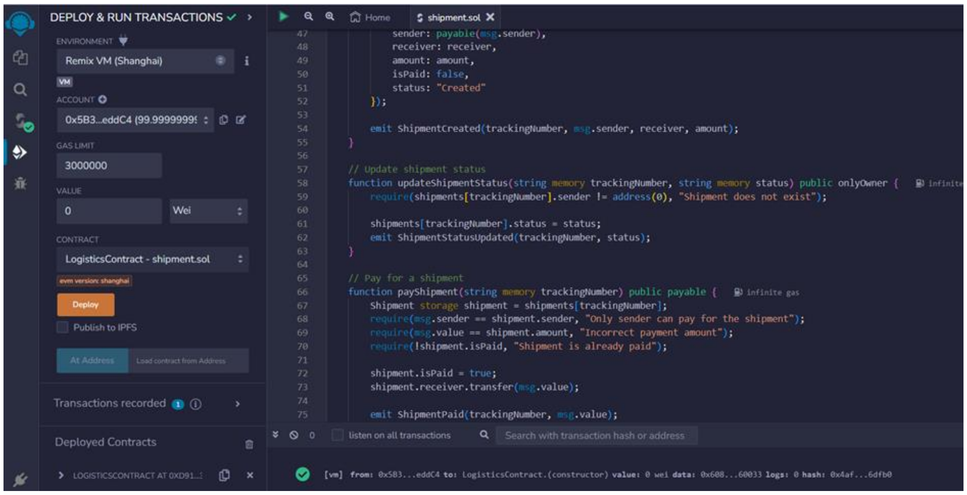
☐ Auto compile

☐ Hide warnings

CONTRACT: LogisticsContract (shipment.sol)

```
// Event to be emitted when a shipment is paid
event ShipmentPaid(string trackingNumber, uint amount);

// Constructor sets the contract deployer as the owner
constructor() {    1306801 gas 1281200 gas
    owner = msg.sender;
}

// Modifier to restrict functions to the contract owner
modifier onlyOwner() {
    require(msg.sender == owner, "Only owner can perform this action");
    _;
}

// Create a new shipment
function createShipment(string memory trackingNumber, address payable receiver, uint amount) publi
    require(shipments[trackingNumber].sender == address(0), "Shipment already exists");

    shipments[trackingNumber] = Shipment({
        trackingNumber: trackingNumber,
        sender: payable(msg.sender),
        receiver: receiver,
        amount: amount,
        isPaid: false,
        status: "Created"
    });

    emit ShipmentCreated(trackingNumber, msg.sender, receiver, amount);
```

2. Deployment Remix VM Shanghai)


DEPLOY & RUN TRANSACTIONS
Home
shipment.sol
Deploy
At Address
Load contract from Address
Transactions recorded 1
Deployed Contracts
LOGISTICSCONTRACT AT 0XD91...
listen on all transactions
Search with transaction hash or address

ENVIRONMENT: Remix VM (Shanghai)

ACCOUNT: 0x5B3...eddC4 (99.9999999

GAS LIMIT: 3000000

VALUE: 0 Wei

CONTRACT: LogisticsContract - shipment.sol

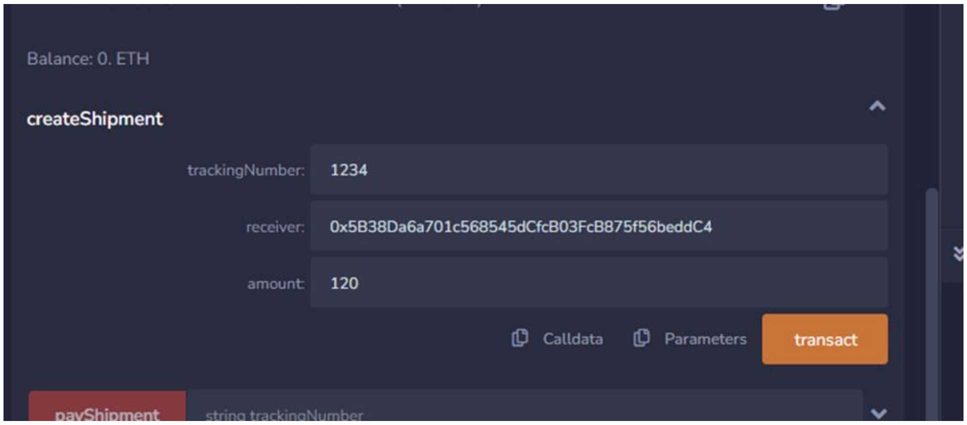
evm version: shanghai

□ Publish to IPFS

```
            sender: payable(msg.sender),
            receiver: receiver,
            amount: amount,
            isPaid: false,
            status: "Created"
        });

        emit ShipmentCreated(trackingNumber, msg.sender, receiver, amount);
    }

    // Update shipment status
    function updateShipmentStatus(string memory trackingNumber, string memory status) public onlyOwner {    infinite
        require(shipments[trackingNumber].sender != address(0), "Shipment does not exist");

        shipments[trackingNumber].status = status;
        emit ShipmentStatusUpdated(trackingNumber, status);
    }

    // Pay for a shipment
    function payShipment(string memory trackingNumber) public payable {    infinite gas
        Shipment storage shipment = shipments[trackingNumber];
        require(msg.sender == shipment.sender, "Only sender can pay for the shipment");
        require(msg.value == shipment.amount, "Incorrect payment amount");
        require(!shipment.isPaid, "Shipment is already paid");

        shipment.isPaid = true;
        shipment.receiver.transfer(msg.value);

        emit ShipmentPaid(trackingNumber, msg.value);
```

```
[vm] from: 0x5B3...eddC4 to: LogisticsContract.(constructor) value: 0 wei data: 0x608...60033 logs: 0 hash: 0x4af...6dfb0
```

3. Testing the Smart Contract – Create Shipment



Balance: 0. ETH

createShipment

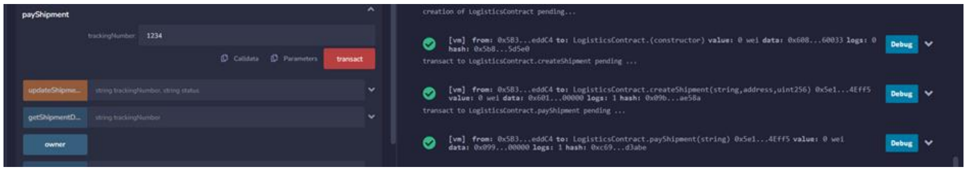
trackingNumber: 1234

receiver: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4

amount: 120

payShipment: string trackingNumber

4. Payment



payShipment

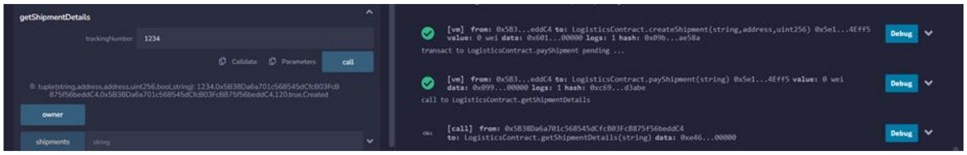
trackingNumber: 1234

updateShipme...: string trackingNumber, string status

getShipmentD...: string trackingNumber

```
creation of LogisticsContract pending...

[vm] from: 0x5B3...eddC4 to: LogisticsContract.(constructor) value: 0 wei data: 0x608...60033 logs: 0
hash: 0x5b8...5d5e0
transact to LogisticsContract.createShipment pending ...

[vm] from: 0x5B3...eddC4 to: LogisticsContract.createShipment(string,address,uint256) 0x5e1...4Eff5
value: 0 wei data: 0x601...00000 logs: 1 hash: 0x09b...ae58a
transact to LogisticsContract.payShipment pending ...

[vm] from: 0x5B3...eddC4 to: LogisticsContract.payShipment(string) 0x5e1...4Eff5 value: 0 wei
data: 0x099...00000 logs: 1 hash: 0xc69...d3abe
```

5. Display shipment



getShipmentDetails

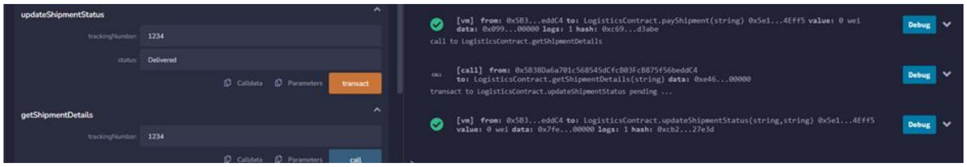
trackingNumber: 1234

0: tuple(string,address,address,uint256,bool,string): 1234,0x5B38Da6a701c568545dCfcB03FcB875f56beddC4,0x5B38Da6a701c568545dCfcB03FcB875f56beddC4,120,true,Created

shipments: string

```
[vm] from: 0x5B3...eddC4 to: LogisticsContract.createShipment(string,address,uint256) 0x5e1...4Eff5
value: 0 wei data: 0x601...00000 logs: 1 hash: 0x09b...ae58a
transact to LogisticsContract.payShipment pending ...

[vm] from: 0x5B3...eddC4 to: LogisticsContract.payShipment(string) 0x5e1...4Eff5 value: 0 wei
data: 0x099...00000 logs: 1 hash: 0xc69...d3abe
call to LogisticsContract.getShipmentDetails

[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to: LogisticsContract.getShipmentDetails(string) data: 0xe46...00000
```

6. Update



updateShipmentStatus

trackingNumber: 1234

status: Delivered

getShipmentDetails

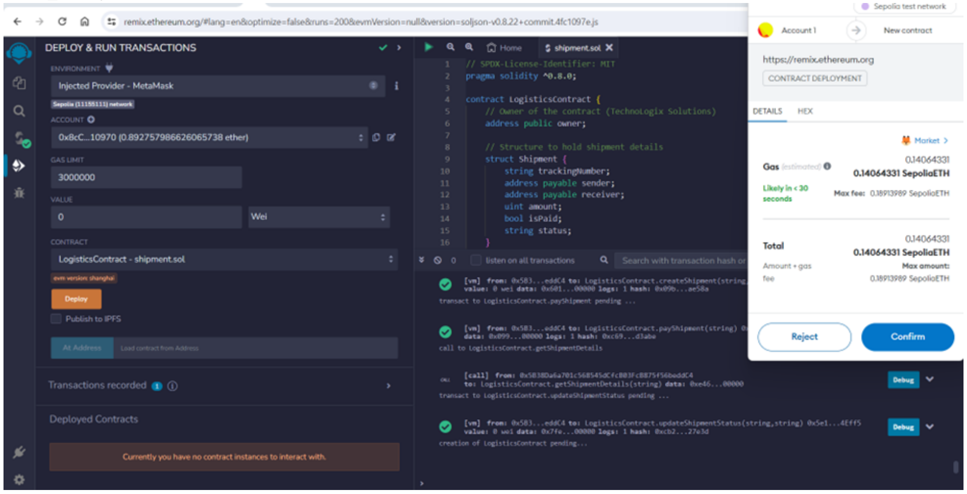
trackingNumber: 1234

```
[vm] from: 0x5B3...eddC4 to: LogisticsContract.payShipment(string) 0x5e1...4Eff5 value: 0 wei
data: 0x099...00000 logs: 1 hash: 0xc69...d3abe
call to LogisticsContract.getShipmentDetails

[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to: LogisticsContract.getShipmentDetails(string) data: 0xe46...00000
transact to LogisticsContract.updateShipmentStatus pending ...

[vm] from: 0x5B3...eddC4 to: LogisticsContract.updateShipmentStatus(string,string) 0x5e1...4Eff5
value: 0 wei data: 0x7fe...00000 logs: 1 hash: 0xcb2...27e3d
```

2. MetaMask Deployment (Sepolia Test Network)

After successfully testing the smart contract I then deploy it using MetaMask on sepolia test network. I have to down the MetaMask plugin in my web browser, and sign. I created and in alchemy website to allow me to get free ether from Ethereum Sepolia Faucet.

Deployment to MetaMask



remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.22+commit.4fc1097e.js

DEPLOY & RUN TRANSACTIONS

ENVIRONMENT: Injected Provider - MetaMask

Sepolia (11155111) network

ACCOUNT: 0x8cC...10970 (0.892757986626065738 ether)

GAS LIMIT: 3000000

VALUE: 0 Wei

CONTRACT: LogisticsContract - shipment.sol

evm version: shanghai

Deploy

□ Publish to IPFS

At Address | Load contract from Address

Transactions recorded 1

Deployed Contracts

Currently you have no contract instances to interact with.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract LogisticsContract {
    // Owner of the contract (TechnoLogix Solutions)
    address public owner;

    // Structure to hold shipment details
    struct Shipment {
        string trackingNumber;
        address payable sender;
        address payable receiver;
        uint amount;
        bool isPaid;
        string status;
    }
```

```
[vm] from: 0x5B3...eddC4 to: LogisticsContract.createShipment(string,
value: 0 wei data: 0x601...00000 logs: 1 hash: 0x09b...ae58a
transact to LogisticsContract.payShipment pending ...

[vm] from: 0x5B3...eddC4 to: LogisticsContract.payShipment(string) 0
data: 0x099...00000 logs: 1 hash: 0xc69...d3abe
call to LogisticsContract.getShipmentDetails

[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to: LogisticsContract.getShipmentDetails(string) data: 0xe46...00000
transact to LogisticsContract.updateShipmentStatus pending ...

[vm] from: 0x5B3...eddC4 to: LogisticsContract.updateShipmentStatus(string,string) 0x5e1...4Eff5
value: 0 wei data: 0x7fe...00000 logs: 1 hash: 0xcb2...27e3d
creation of LogisticsContract pending...
```

Sepolia test network

Account 1 → New contract

https://remix.ethereum.org

CONTRACT DEPLOYMENT

DETAILS HEX

Market >

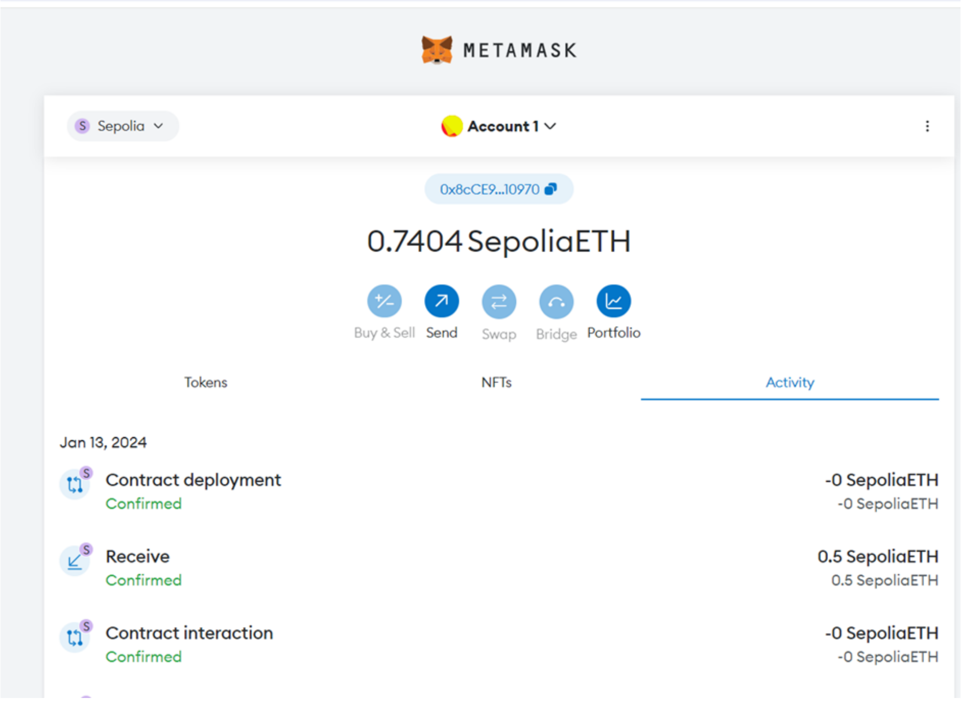
Gas (estimated) 0.14064331 **0.14064331 SepoliaETH**

Likely in < 30 seconds — Max fee: 0.18913989 SepoliaETH

Total 0.14064331 **0.14064331 SepoliaETH**

Amount + gas fee — Max amount: 0.18913989 SepoliaETH

Reject | Confirm



Sepolia

Account 1

0x8cCE9...10970

0.7404 SepoliaETH

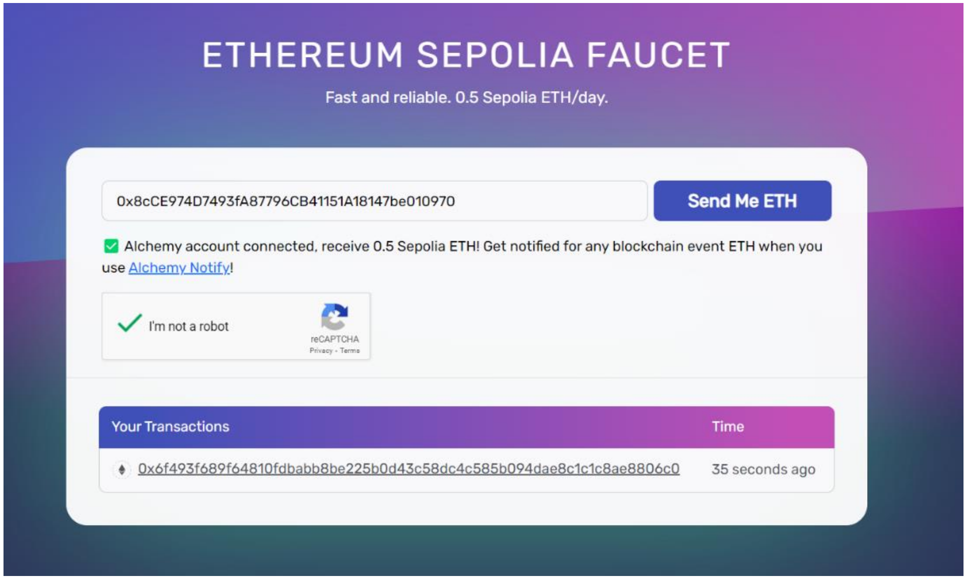
Jan 13, 2024

Contract deployment — Confirmed — -0 SepoliaETH -0 SepoliaETH

Receive — Confirmed — 0.5 SepoliaETH 0.5 SepoliaETH

Contract interaction — Confirmed — -0 SepoliaETH -0 SepoliaETH

Getting Ether from Sepolia Faucet

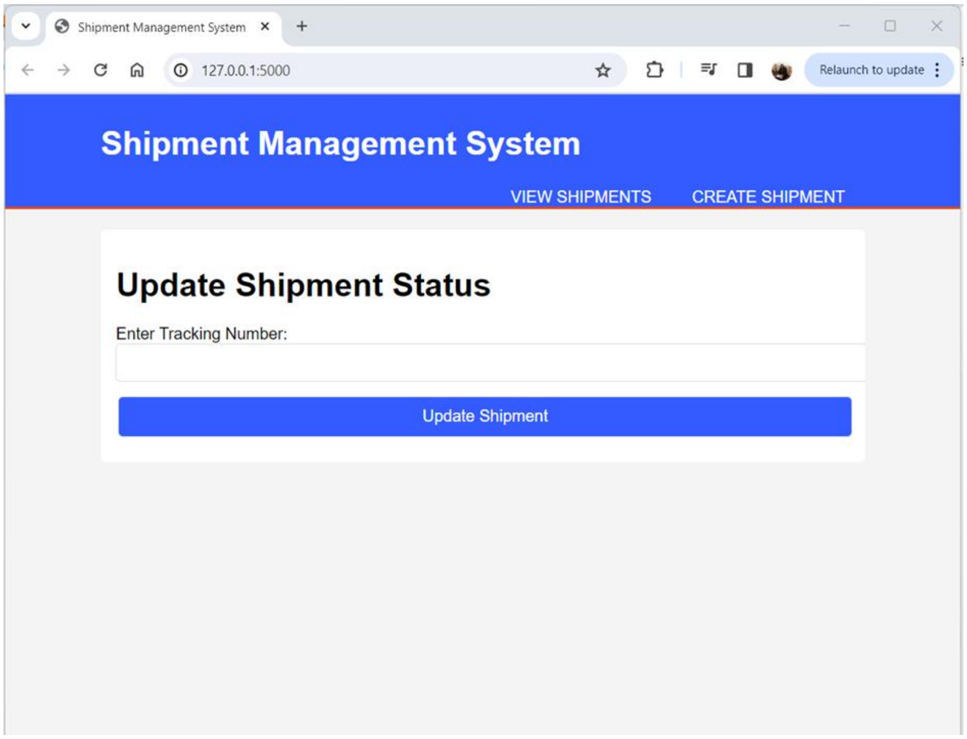


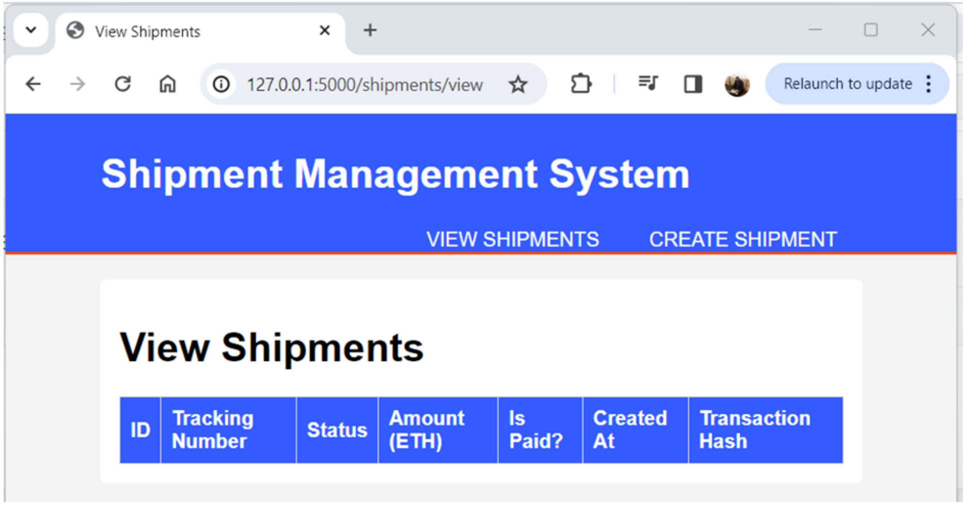
0x8cCE974D7493fA87796CB41151A18147be010970

☑ Alchemy account connected, receive 0.5 Sepolia ETH! Get notified for any blockchain event ETH when you use Alchemy Notify!

Your Transactions	Time
0x6f493f689f64810fdbabb8be225b0d43c58dc4c585b094dae8c1c1c8ae8806c0	35 seconds ago

3. I developed shipment management microservice in html and python using Flask and containerized it. I wrote a code to connect to my MetaMask account for the interaction with the smart contract.



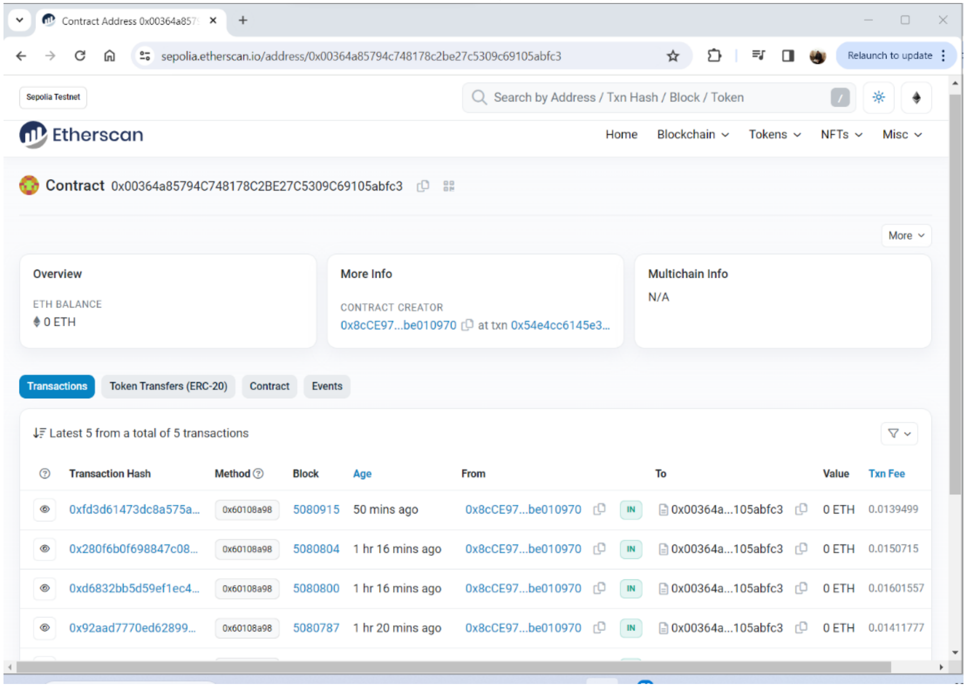


Shipment Management System

View Shipments

ID	Tracking Number	Status	Amount (ETH)	Is Paid?	Created At	Transaction Hash

4. I use etherscan to search for my deployed contract and the transactions


Contract Address 0x00364a857
sepolia.etherscan.io/address/0x00364a85794c748178c2be27c5309c69105abfc3
Relaunch to update
Sepolia Testnet
Search by Address / Txn Hash / Block / Token
Etherscan
Home
Blockchain
Tokens
NFTs
Misc
More
Transactions
Token Transfers (ERC-20)
Contract
Events

Contract 0x00364a85794C748178C2BE27C5309C69105abfc3

Overview

ETH BALANCE
0 ETH

More Info

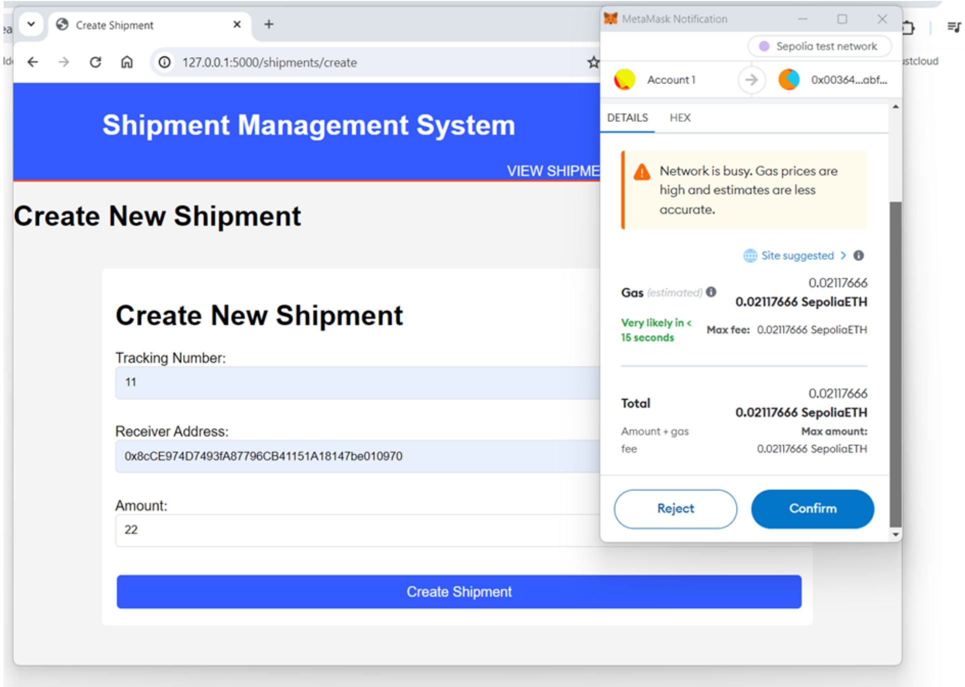
CONTRACT CREATOR
0x8cCE97...be010970 at txn 0x54e4cc6145e3...

Multichain Info
N/A

Latest 5 from a total of 5 transactions

Transaction Hash	Method	Block	Age	From		To	Value	Txn Fee
0xfd3d61473dc8a575a...	0x60108a98	5080915	50 mins ago	0x8cCE97...be010970	IN	0x00364a...105abfc3	0 ETH	0.0139499
0x280f6b0f698847c08...	0x60108a98	5080804	1 hr 16 mins ago	0x8cCE97...be010970	IN	0x00364a...105abfc3	0 ETH	0.0150715
0xd6832bb5d59ef1ec4...	0x60108a98	5080800	1 hr 16 mins ago	0x8cCE97...be010970	IN	0x00364a...105abfc3	0 ETH	0.01601557
0x92aad7770ed62899...	0x60108a98	5080787	1 hr 20 mins ago	0x8cCE97...be010970	IN	0x00364a...105abfc3	0 ETH	0.01411777

5. I have tested transactions using containerized web applications to interact with the smart contract



Shipment Management System

Create New Shipment

Create New Shipment

Tracking Number: 11

Receiver Address: 0x8cCE974D7493fA87796CB41151A18147be010970

Amount: 22

Network is busy. Gas prices are high and estimates are less accurate.

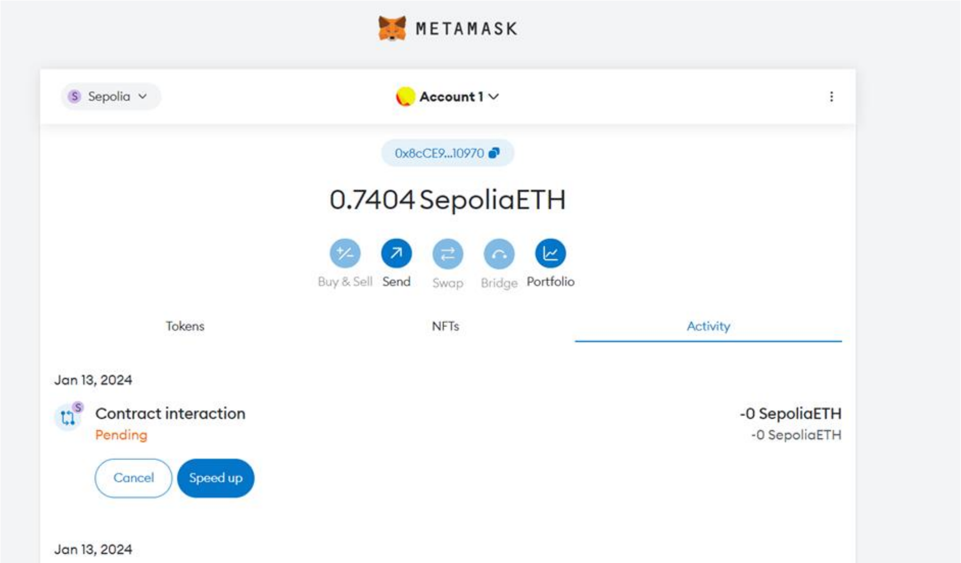
Site suggested

Gas (estimated) 0.02117666 0.02117666 SepoliaETH

Very likely in < 15 seconds Max fee: 0.02117666 SepoliaETH

Total 0.02117666 0.02117666 SepoliaETH

Amount + gas fee Max amount: 0.02117666 SepoliaETH



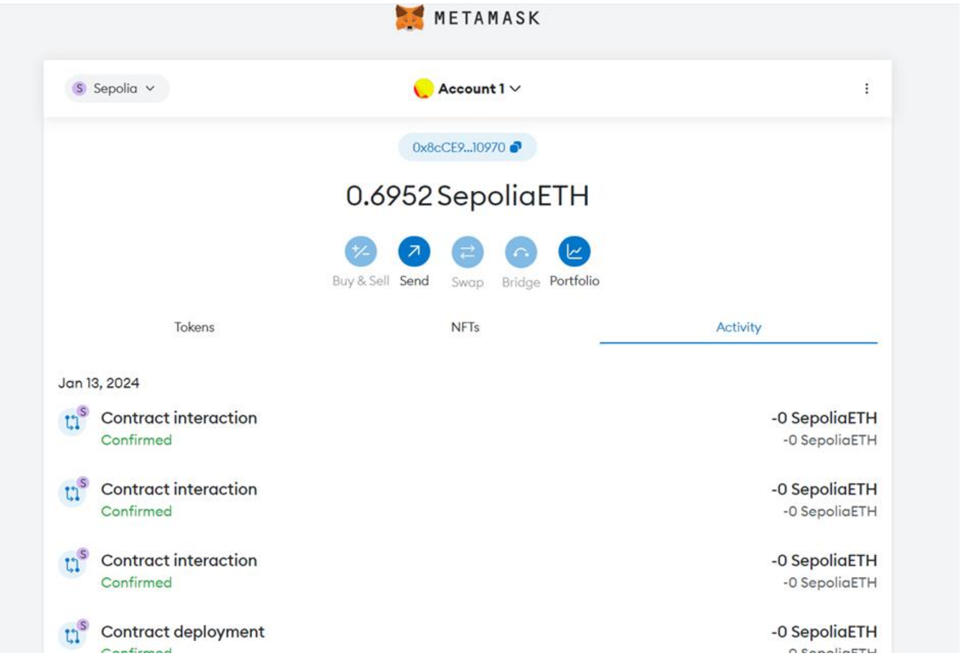
0.7404 SepoliaETH

Jan 13, 2024

Contract interaction -0 SepoliaETH

Pending -0 SepoliaETH

Jan 13, 2024



0.6952 SepoliaETH

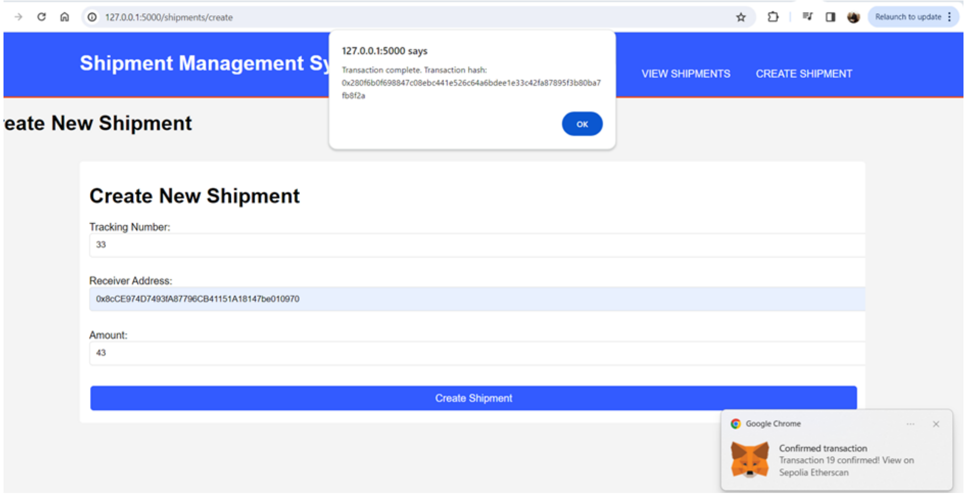
Jan 13, 2024

Contract interaction
Confirmed
-0 SepoliaETH
-0 SepoliaETH

Contract interaction
Confirmed
-0 SepoliaETH
-0 SepoliaETH

Contract interaction
Confirmed
-0 SepoliaETH
-0 SepoliaETH

Contract deployment
-0 SepoliaETH



127.0.0.1:5000 says

Transaction complete. Transaction hash: 0x280f6b0f698847c08ebc441e526c64a6bdee1e33c42fa87895f3b80ba7fb8f2a

Create New Shipment

Tracking Number: 33

Receiver Address: 0x8cCE974D7493fA87796CB41151A18147be010970

Amount: 43

Confirmed transaction
Transaction 19 confirmed! View on Sepolia Etherscan

5.4. Practical Application Outcomes

Finally, we intend to evaluate the real-world applicability of our solutions through simulated scenarios. This involved:

Setting up real-world mimicking scenarios for our logistics operations.

Running these scenarios using our Dockerized microservices, APIs, and blockchain system.

Collecting and analyzing data on performance, scalability, and user experience.

The analysis provided concrete evidence of our solutions' effectiveness in improving logistics operations, highlighting their practical benefits and areas for further development.

Refer to Chapter 10 for the project GitHub repository

CHAPTER 6
FINAL REPORT COMPILATION

6.1. Comprehensive Solution Overview

In this section, we present a consolidated overview of the solutions developed as part of the 'Transforming Global Logistics with TechnoLogix Solutions' project. The project encompassed several key areas:

- Governance Model: Established to define clear decision-making processes and accountability structures, ensuring effective management of logistics operations.
- Information System Management: Enhanced with advanced data processing and security measures, facilitating efficient data handling and improved decision-making.
- Data Center Architecture: Redesigned for scalability and reliability, featuring energy-efficient technologies and optimized storage solutions.
- Microservices Development and Deployment: Implemented using Docker and Kubernetes, providing flexibility, scalability, and efficient software management.
- Blockchain Integration: Introduced for improved security and transparency in logistics operations, utilizing smart contracts for secure and transparent transaction recording.
- API Development: Focused on creating RESTful APIs for seamless interaction between different system components, adhering to OpenAPI specifications for clarity and standardization.

These solutions collectively aim to revolutionize logistics operations, making them more efficient, transparent, and adaptable to rapidly changing global demands.

6.2. Future Roadmap and Sustainability Considerations

As we look towards the future, it is crucial to outline a roadmap for ongoing maintenance, updates, and enhancements of our solutions, while also considering their global scalability and environmental impact.

- Ongoing Maintenance and Updates: Regular updates and maintenance will be crucial for keeping the system up-to-date with the latest technological advancements. This includes periodic reviews of the governance model, continuous monitoring and upgrading of information systems, data center technologies, and regular updates to microservices and blockchain implementations.
- Scalability: As the logistics industry continues to grow, the solutions must be designed to scale seamlessly. This involves leveraging cloud-based services, optimizing microservices for easy replication and distribution, and ensuring that the blockchain network can handle increased transaction volumes.
- Environmental Considerations: In developing these technologies, environmental impact is a key concern. This involves choosing energy-efficient hardware for data centers, optimizing software for lower power consumption, and exploring renewable energy sources.
- Sustainable Practices in Logistics: The project advocates for sustainable practices within the logistics industry. This includes promoting digitalization to reduce paper usage, optimizing routes to lower fuel consumption, and encouraging the use of eco-friendly materials in packaging and shipping.

CHAPTER 6
CONCLUSION

6.1. Project Summary

In conclusion, the Transforming Global Logistics with TechnoLogix Solutions project has been a comprehensive endeavor to revolutionize logistics operations using cutting-edge technologies. Key achievements of this project include:

- Development of a Robust Governance Model: This model has established a clear framework for decision-making and accountability, essential for the smooth operation of complex logistics systems.
- Advanced Information System Management: We've upgraded the system to handle data more efficiently and securely, enabling quicker and more informed decision-making.
- Innovative Data Center Architecture: The new architecture is both scalable and reliable, capable of supporting the growing data needs of the logistics sector.
- Deployment of Microservices using Docker and Kubernetes: This approach has enhanced the flexibility and scalability of our software solutions, making them more adaptable to changing needs.
- Blockchain Integration for Enhanced Security and Transparency: By incorporating blockchain technology, we've brought a new level of trust and efficiency to logistics operations.
- Efficient API Development and Integration: The creation of RESTful APIs, adhering to OpenAPI specifications, has facilitated better communication between different system components although it is still in development.

8.2. Reflections and Learning Outcomes

Reflecting on this project, we have gained invaluable experience and insights. Key learning outcomes include:

- Understanding of Complex System Integration: This project has deepened our understanding of how various technological components, such as microservices, blockchain, and data centers, can be integrated to create a cohesive and efficient system.

- Experience in Advanced Technologies: Working hands-on with technologies like Docker, Kubernetes, and blockchain has provided me with practical skills and a deeper appreciation of these tools in solving real-world problems.
- Insights into Project Management: Managing different aspects of this project has honed my skills in organization, time management, and collaborative working, all of which are crucial in any future technological endeavor.
- Appreciation for Sustainable Practices: The project has heightened my awareness of the environmental impact of technological solutions and the importance of incorporating sustainable practices in every aspect of development and implementation.

Appendices

Appendix A: Supplementary Diagrams and Data

Introductory Note:
This appendix contains supplementary diagrams and data sets that provide additional insights into the various aspects of the 'Transforming Global Logistics with TechnoLogix Solutions' project. These visual representations aid in a deeper understanding of the concepts, designs, and structures discussed in the report.

Figure A1: Activity Diagram for the Project
Description: This diagram illustrates the architecture of the Activity diagram for the project.

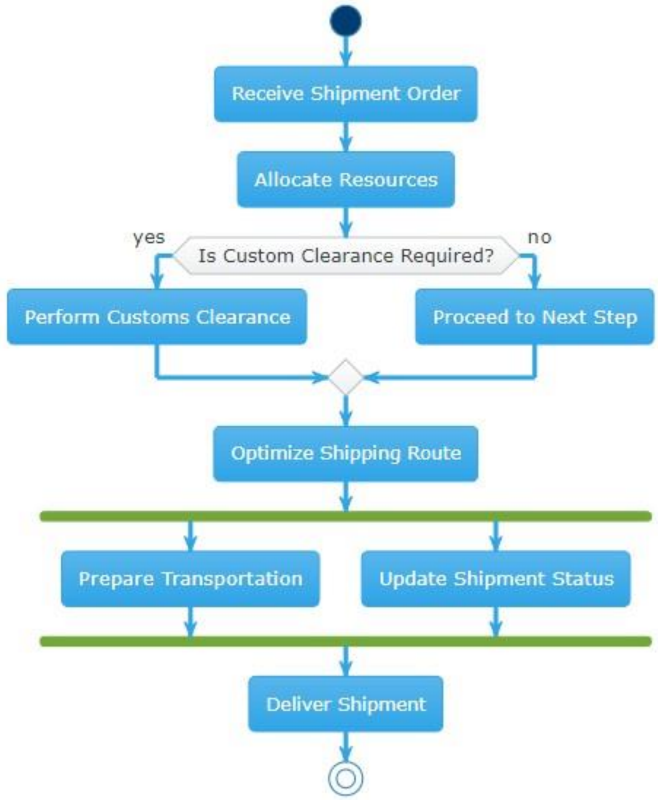


Figure A2: Class Diagram for the Project
Description: The diagram below shows the class diagram for the project.

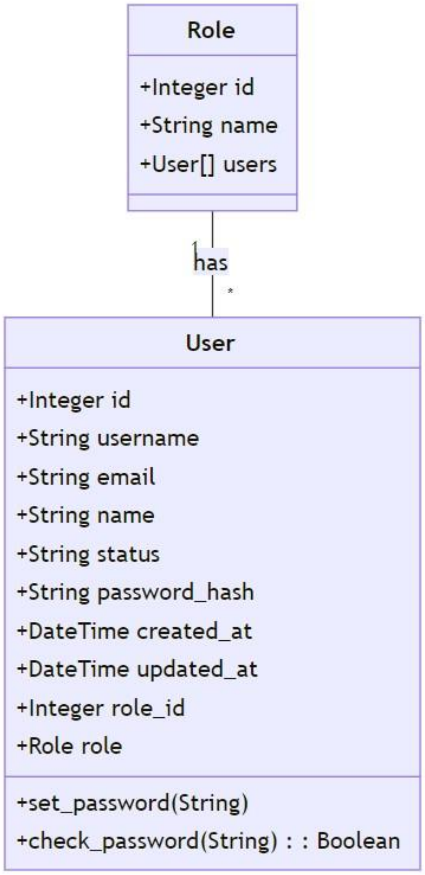
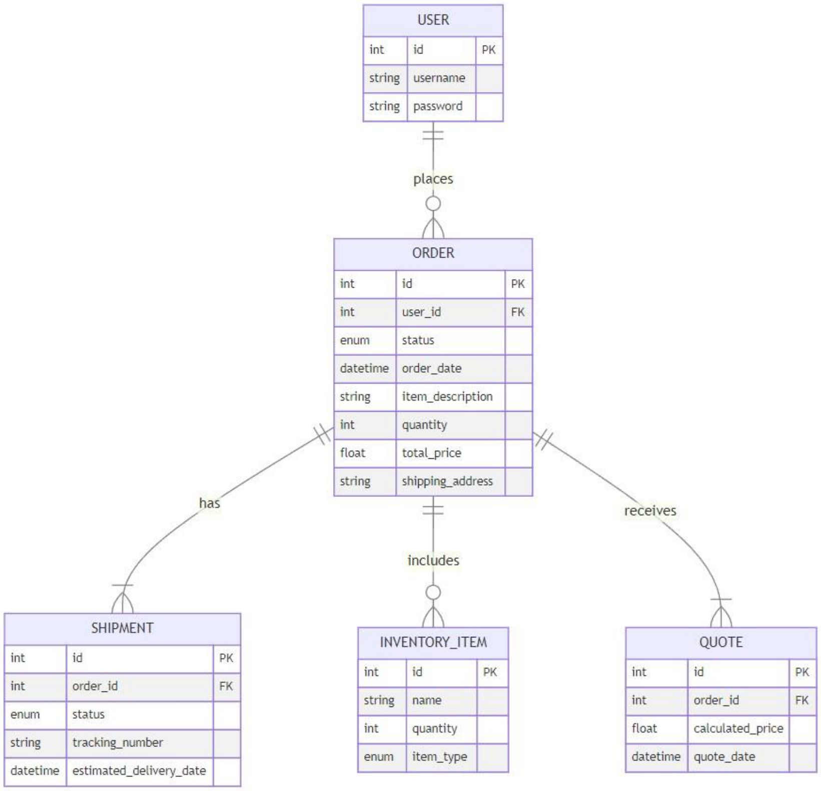


Figure A3: Entity Relationship Diagram for Order_Management_Service Microservice
Description: The diagram shows the relationships among tables and the data

model.



USER		
int	id	PK
string	username	
string	password	

ORDER		
int	id	PK
int	user_id	FK
enum	status	
datetime	order_date	
string	item_description	
int	quantity	
float	total_price	
string	shipping_address	

SHIPMENT		
int	id	PK
int	order_id	FK
enum	status	
string	tracking_number	
datetime	estimated_delivery_date	

INVENTORY_ITEM		
int	id	PK
string	name	
int	quantity	
enum	item_type	

QUOTE		
int	id	PK
int	order_id	FK
float	calculated_price	
datetime	quote_date	

Figure A4: Flow Diagram

Description: This diagram provides a view of the flow in the application

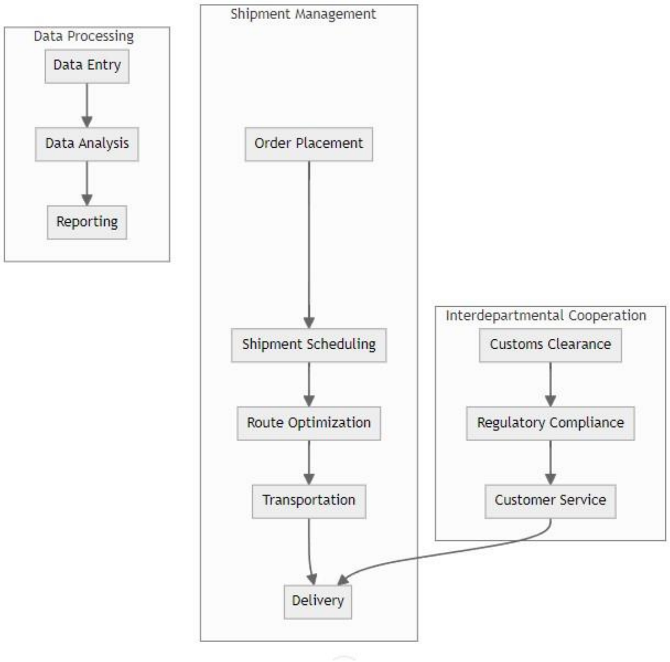


Figure A5: System Architecture Diagram

Description: The diagram shows the whole architecture of the system.

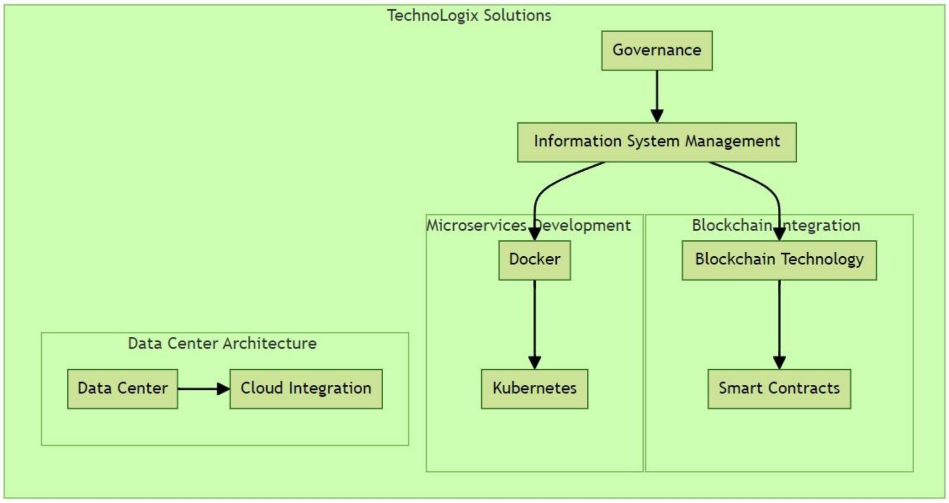


Figure A6: Use Case Diagram
Description: The diagram models a portion of the use cases.

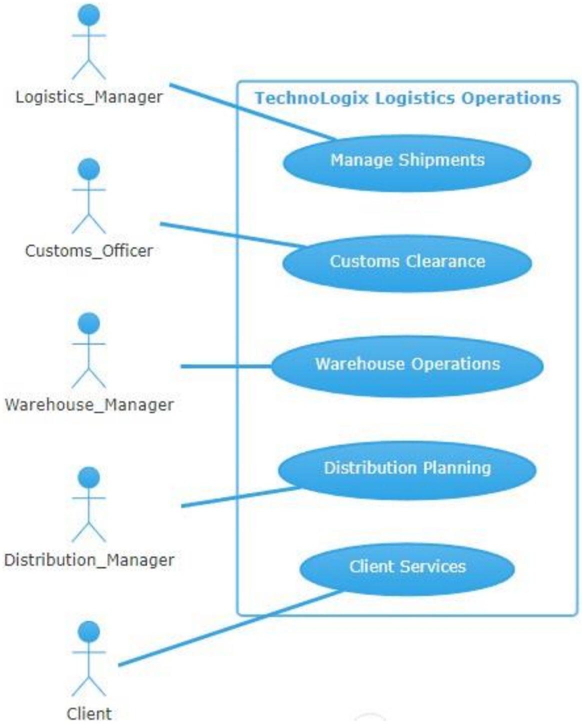
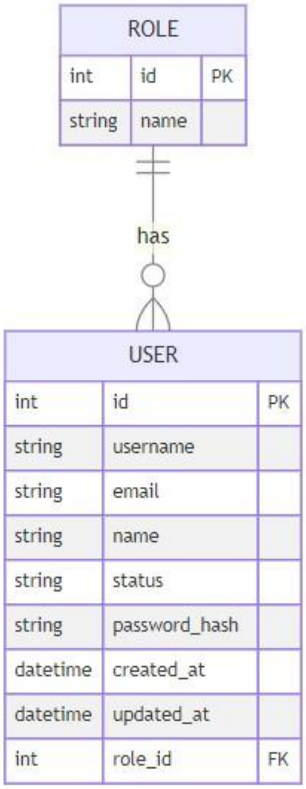


Figure A7: Entity Relationship Diagram for User_Management_Service Microservice

Description: The diagram shows the relationships among tables and the data in the user_maangement_service microservice



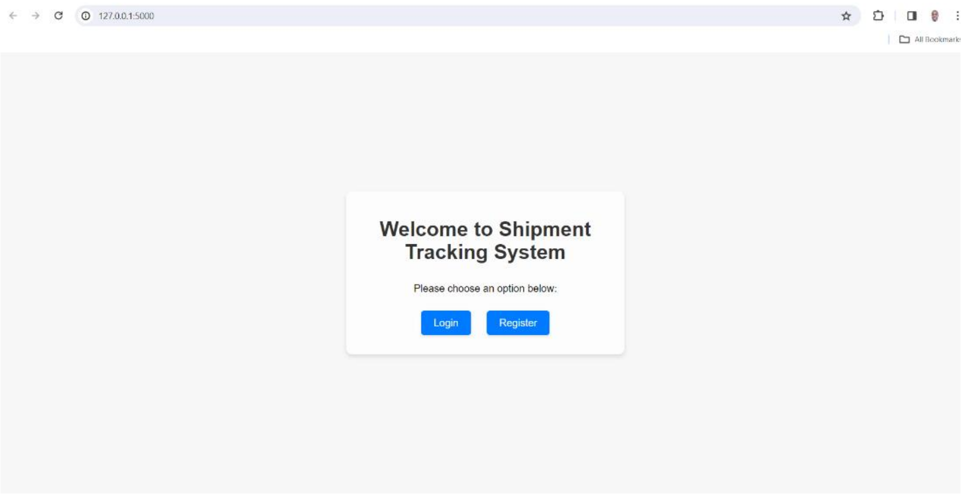
ROLE

int	id	PK
string	name	

USER

int	id	PK
string	username	
string	email	
string	name	
string	status	
string	password_hash	
datetime	created_at	
datetime	updated_at	
int	role_id	FK

Figure A8: Landing Page for the User_Management_Service
Description: The diagram shows the landing page for the user_mangement_service microservice

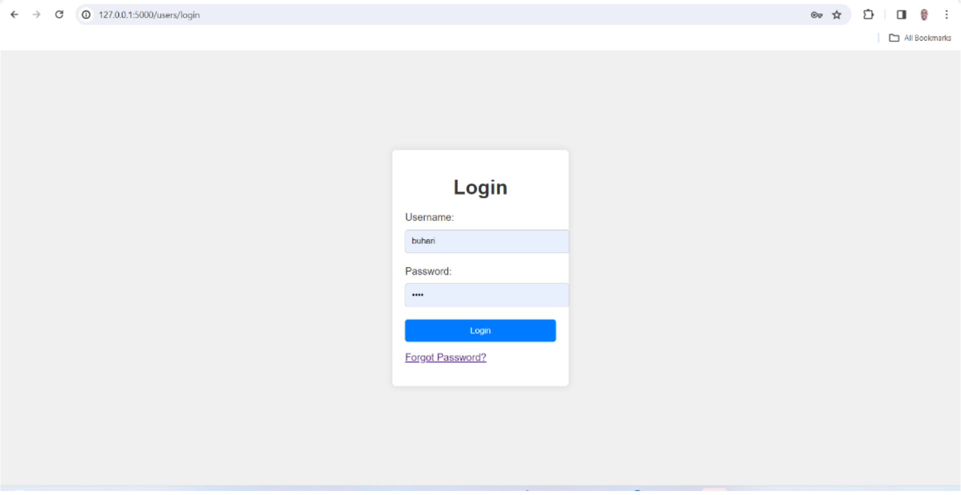


127.0.0.1:5000

Welcome to Shipment Tracking System

Please choose an option below:

Figure A9: Login Page for the User_Management_Service
Description: The diagram shows the login page for the user_mangement_service microservice



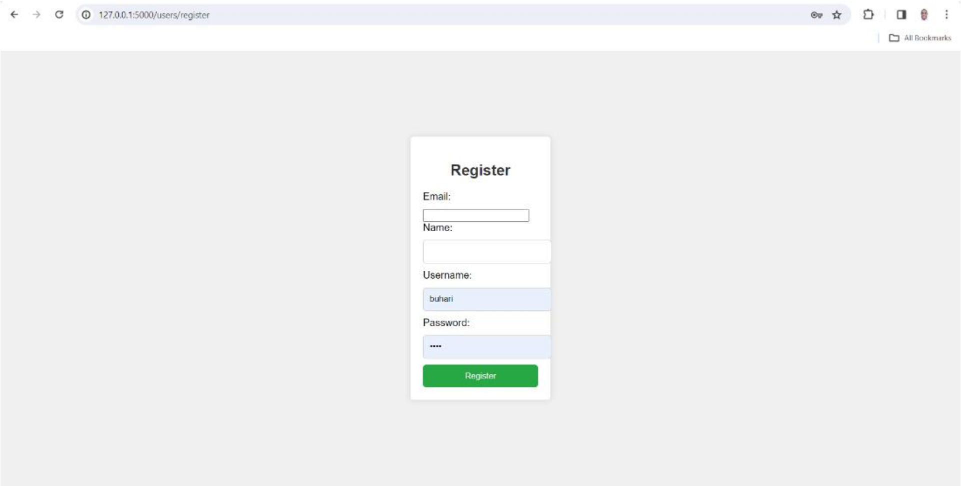
127.0.0.1:5000/users/login

Login

Username: buhari

Password: ••••

Figure A9: Registration Page for the User_Management_Service
Description: The diagram shows the registration page for the user_mangement_service microservice



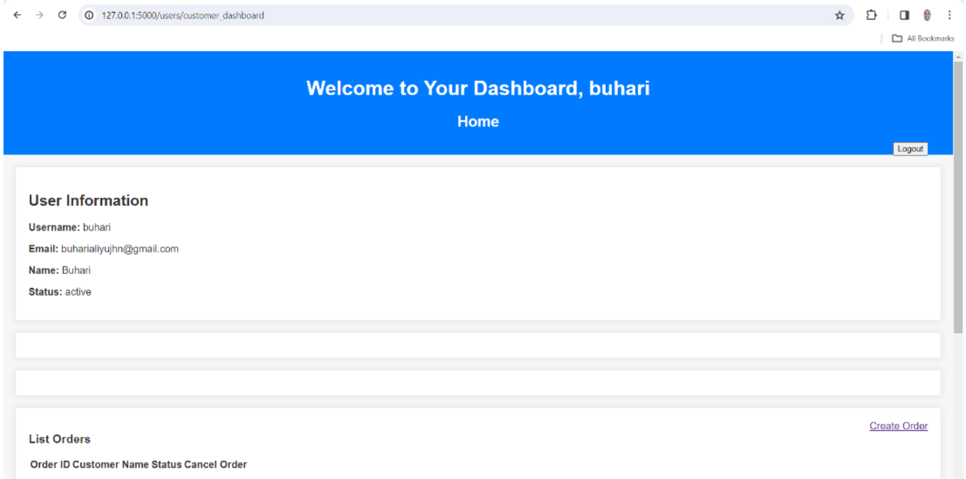
Email:

Name:

Username: buhari

Password: ••••

Figure A10: User Dashboard Page for the User_Management_Service
Description: The diagram shows the user dashboard page for the user_mangement_service microservice



Welcome to Your Dashboard, buhari

User Information

Username: buhari

Email: buharialiyujhn@gmail.com

Name: Buhari

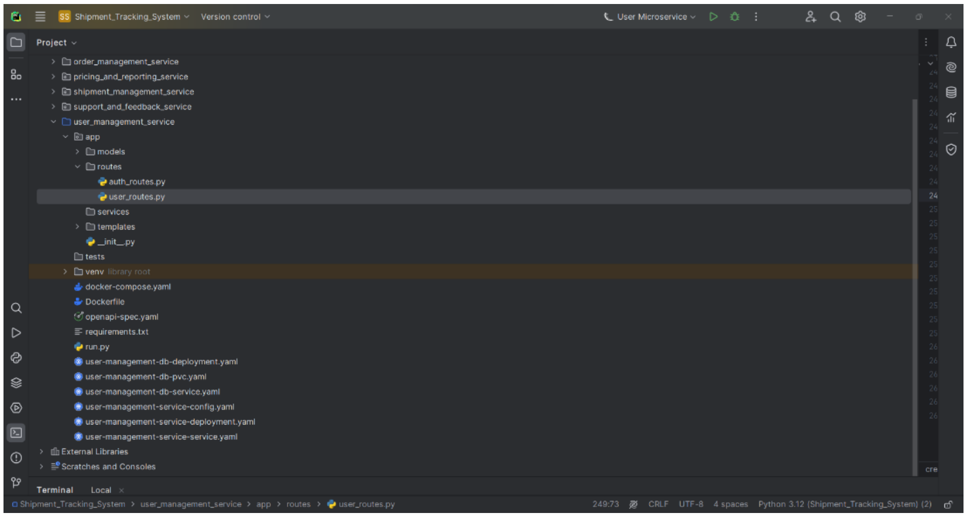
Status: active

List Orders

Order ID Customer Name Status Cancel Order

Appendix B: Code Snippets and Screenshots

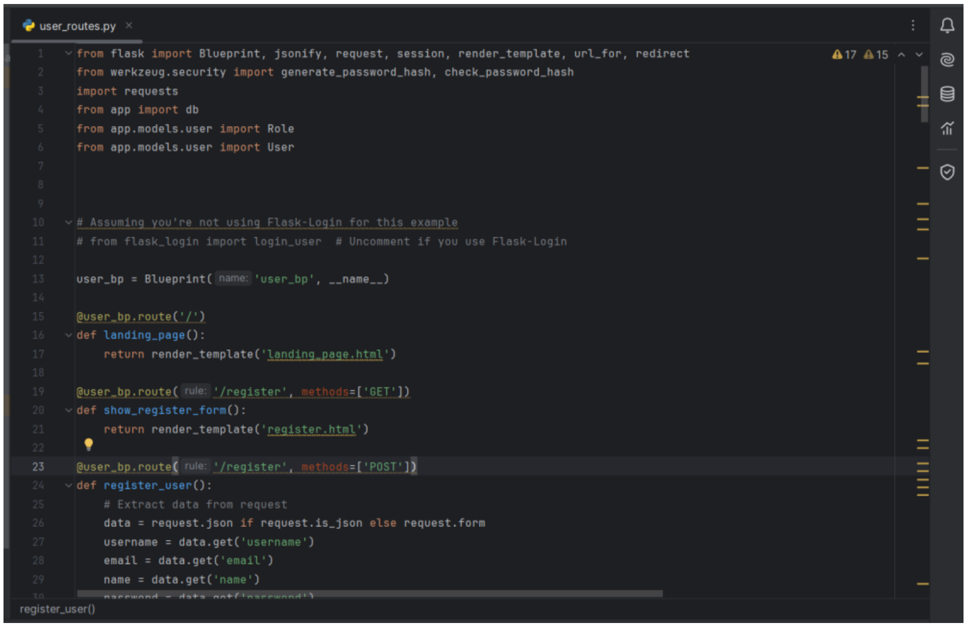
Figure B1: Project Structure: The diagram shows the project structure.



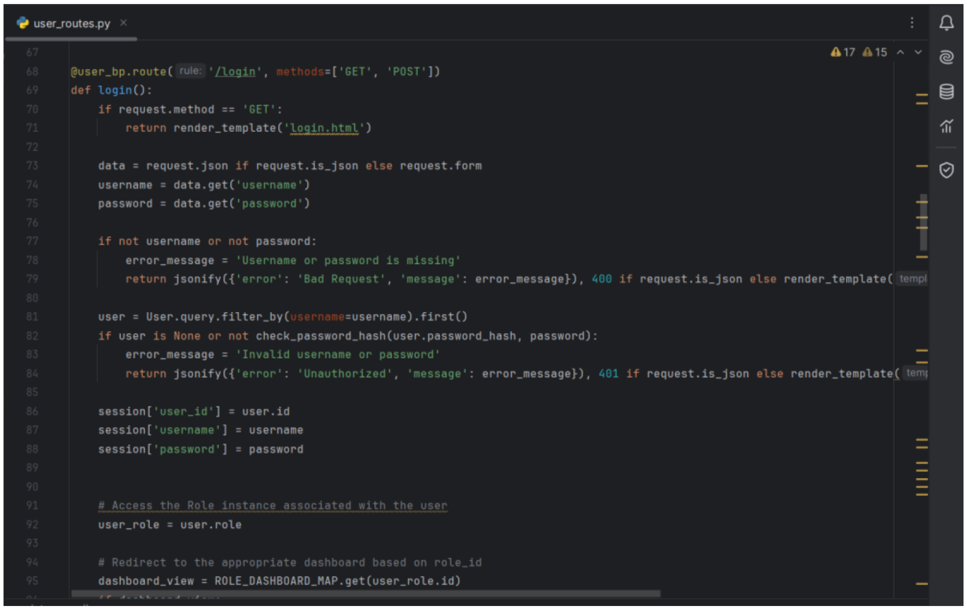
```
order_management_service
pricing_and_reporting_service
shipment_management_service
support_and_feedback_service
user_management_service
  app
    models
    routes
      auth_routes.py
      user_routes.py
    services
    templates
    __init__.py
  tests
  venv library root
  docker-compose.yaml
  Dockerfile
  openapi-spec.yaml
  requirements.txt
  run.py
  user-management-db-deployment.yaml
  user-management-db-pvc.yaml
  user-management-db-service.yaml
  user-management-service-config.yaml
  user-management-service-deployment.yaml
  user-management-service-service.yaml
External Libraries
Scratches and Consoles
```

```
Shipment_Tracking_System > user_management_service > app > routes > user_routes.py    249:73  CRLF  UTF-8  4 spaces  Python 3.12 (Shipment_Tracking_System) (2)
```

Figure B2: The figure shows the user_route code for the user_management_service microservice.

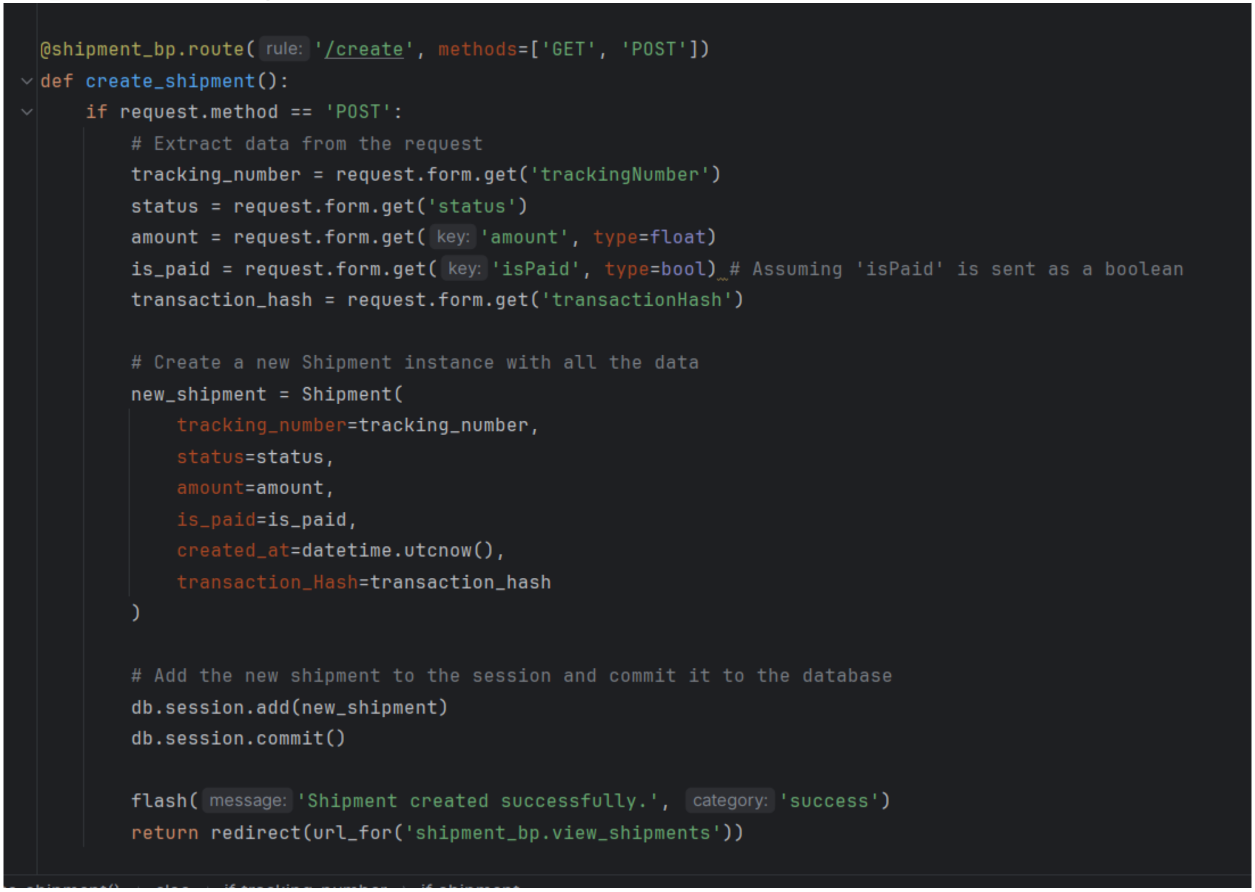


```python
from flask import Blueprint, jsonify, request, session, render_template, url_for, redirect
from werkzeug.security import generate_password_hash, check_password_hash
import requests
from app import db
from app.models.user import Role
from app.models.user import User



# Assuming you're not using Flask-Login for this example
# from flask_login import login_user  # Uncomment if you use Flask-Login

user_bp = Blueprint( name: 'user_bp', __name__)

@user_bp.route('/')
def landing_page():
    return render_template('landing_page.html')

@user_bp.route( rule: '/register', methods=['GET'])
def show_register_form():
    return render_template('register.html')

@user_bp.route( rule: '/register', methods=['POST'])
def register_user():
    # Extract data from request
    data = request.json if request.is_json else request.form
    username = data.get('username')
    email = data.get('email')
    name = data.get('name')
    password = data.get('password')
```



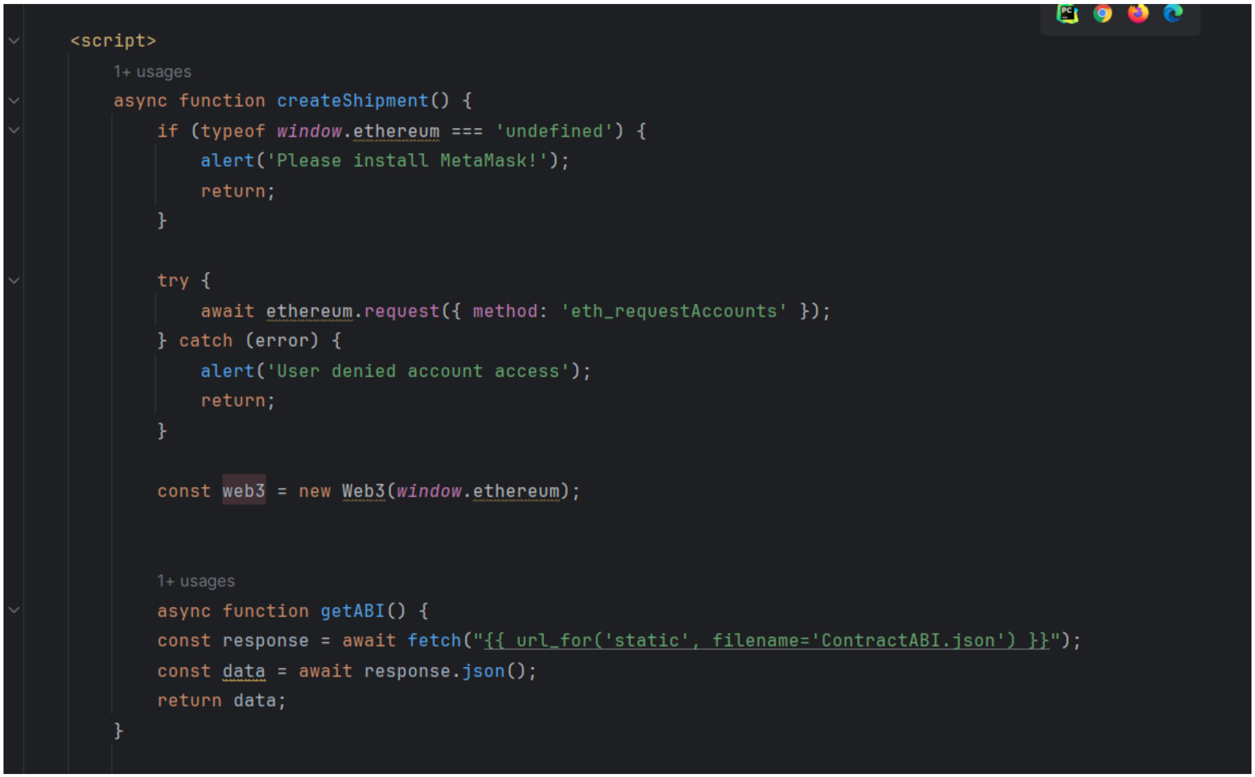
```python
@user_bp.route( rule: '/login', methods=['GET', 'POST'])
def login():
    if request.method == 'GET':
        return render_template('login.html')

    data = request.json if request.is_json else request.form
    username = data.get('username')
    password = data.get('password')

    if not username or not password:
        error_message = 'Username or password is missing'
        return jsonify({'error': 'Bad Request', 'message': error_message}), 400 if request.is_json else render_template( templ

    user = User.query.filter_by(username=username).first()
    if user is None or not check_password_hash(user.password_hash, password):
        error_message = 'Invalid username or password'
        return jsonify({'error': 'Unauthorized', 'message': error_message}), 401 if request.is_json else render_template( temp

    session['user_id'] = user.id
    session['username'] = username
    session['password'] = password


    # Access the Role instance associated with the user
    user_role = user.role

    # Redirect to the appropriate dashboard based on role_id
    dashboard_view = ROLE_DASHBOARD_MAP.get(user_role.id)
```

Figure C: The figure shows the shipment_route, create_shipment code for the shipment_management_service microservice.

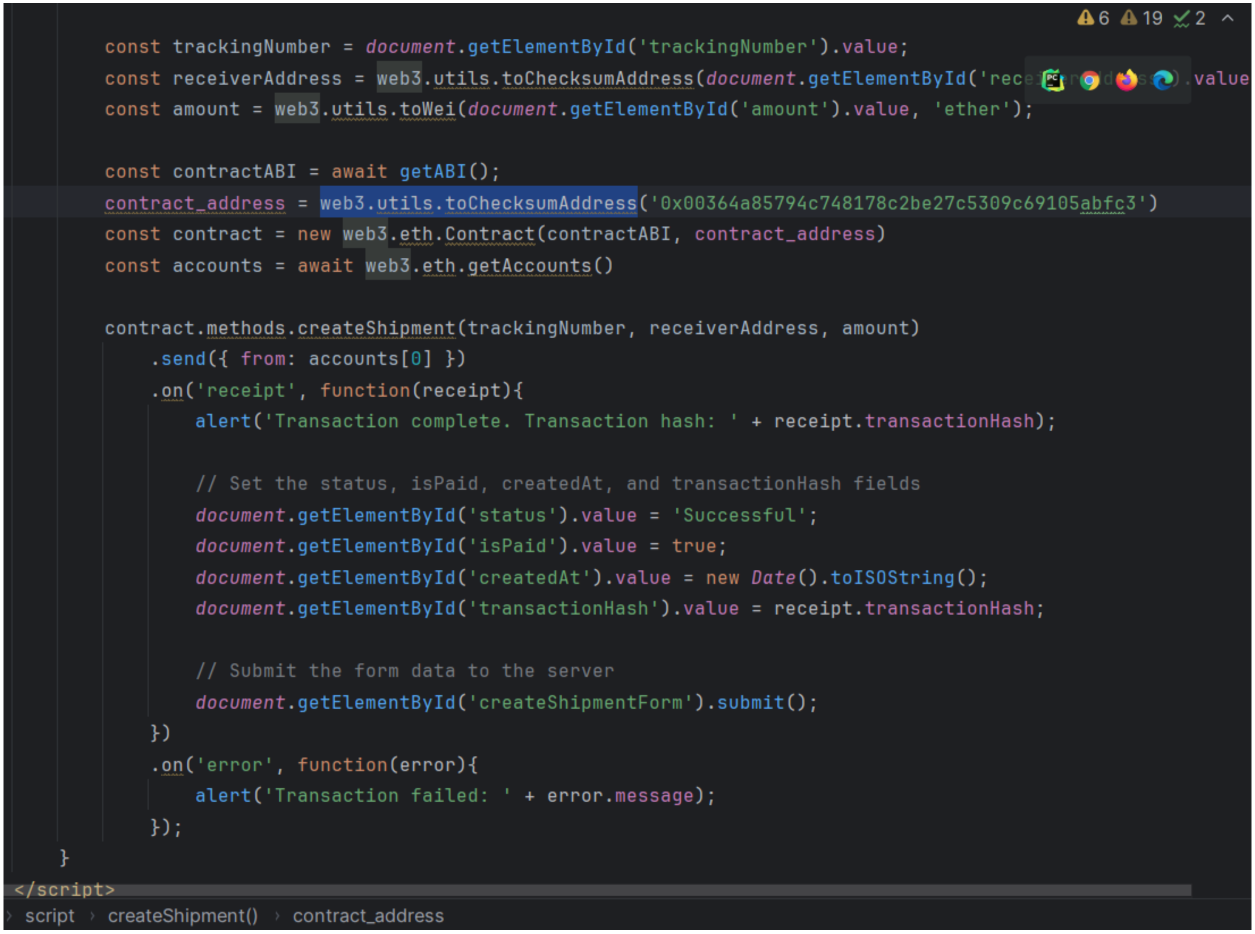


```python
@shipment_bp.route( rule: '/create', methods=['GET', 'POST'])
def create_shipment():
    if request.method == 'POST':
        # Extract data from the request
        tracking_number = request.form.get('trackingNumber')
        status = request.form.get('status')
        amount = request.form.get( key: 'amount', type=float)
        is_paid = request.form.get( key: 'isPaid', type=bool) # Assuming 'isPaid' is sent as a boolean
        transaction_hash = request.form.get('transactionHash')

        # Create a new Shipment instance with all the data
        new_shipment = Shipment(
            tracking_number=tracking_number,
            status=status,
            amount=amount,
            is_paid=is_paid,
            created_at=datetime.utcnow(),
            transaction_Hash=transaction_hash
        )

        # Add the new shipment to the session and commit it to the database
        db.session.add(new_shipment)
        db.session.commit()

        flash( message: 'Shipment created successfully.', category: 'success')
        return redirect(url_for('shipment_bp.view_shipments'))
```

Create Shipment



```
<script>
    1+ usages
    async function createShipment() {
        if (typeof window.ethereum === 'undefined') {
            alert('Please install MetaMask!');
            return;
        }

        try {
            await ethereum.request({ method: 'eth_requestAccounts' });
        } catch (error) {
            alert('User denied account access');
            return;
        }

        const web3 = new Web3(window.ethereum);


        1+ usages
        async function getABI() {
        const response = await fetch("{{ url_for('static', filename='ContractABI.json') }}");
        const data = await response.json();
        return data;
    }
```



```
        const trackingNumber = document.getElementById('trackingNumber').value;
        const receiverAddress = web3.utils.toChecksumAddress(document.getElementById('receiverAddress')).value
        const amount = web3.utils.toWei(document.getElementById('amount').value, 'ether');

        const contractABI = await getABI();
        contract_address = web3.utils.toChecksumAddress('0x00364a85794c748178c2be27c5309c69105abfc3')
        const contract = new web3.eth.Contract(contractABI, contract_address)
        const accounts = await web3.eth.getAccounts()

        contract.methods.createShipment(trackingNumber, receiverAddress, amount)
            .send({ from: accounts[0] })
            .on('receipt', function(receipt){
                alert('Transaction complete. Transaction hash: ' + receipt.transactionHash);

                // Set the status, isPaid, createdAt, and transactionHash fields
                document.getElementById('status').value = 'Successful';
                document.getElementById('isPaid').value = true;
                document.getElementById('createdAt').value = new Date().toISOString();
                document.getElementById('transactionHash').value = receipt.transactionHash;

                // Submit the form data to the server
                document.getElementById('createShipmentForm').submit();
            })
            .on('error', function(error){
                alert('Transaction failed: ' + error.message);
            });
    }
</script>
```

10. Project GitHub Repository

https://github.com/buharialiyujhn/Shipment-Tracking-System/

References

[1] M. Christopher, "Logistics & Supply Chain Management," Pearson UK, 2016.

[2] Y. Sheffi, "The New (Ab)Normal: Reshaping Business and Supply Chain Strategy Beyond Covid-19," MIT Press, 2020.

[3] G. Rougès and Y. Montreuil, "Crowdsourcing Logistics: An opportunity for more sustainable urban logistics," in International Journal of Production Economics, vol. 157, pp. 272-281, 2014.

[4] A. Gunasekaran and E. W. T. Ngai, "Build-to-order supply chain management: A literature review and framework for development," in Journal of Operations Management, vol. 23, no. 5, pp. 423-451, 2005.

[5] S. Kamble, A. Gunasekaran and T. C. E. Eng, "The sustainable humanitarian supply chain design: Agility, adaptability and alignment," in International Journal of Logistics Research and Applications, vol. 22, no. 1, pp. 62-82, 2019.

[6] B. S. Blanchard, "Logistics Engineering and Management," 6th ed., Prentice Hall, 2004.

[7] S. B. Keller and T. A. Keller, "Robots in the warehouse: It's not just Amazon," in Business Horizons, vol. 63, no. 2, pp. 215-224, 2020.

[8] D. Simchi-Levi, W. Schmidt, and Y. Wei, "From Superstorms to Factory Fires: Managing Unpredictable Supply-Chain Disruptions," Harvard Business Review, January-February 2014.

[9] C. M. Wallenburg and A. Wieland, "Global Supply Chain Risk Management," J. Bus. Logist, vol. 29, no. 1, pp. 133–155, 2008.

[10] J. J. Bartholdi III and S. Hackman, "Warehouse & Distribution Science," Release 0.98, The Supply Chain and Logistics Institute, School of Industrial and Systems Engineering, Georgia Institute of Technology, 2014.

[11] C. Langley Jr., "2021 Third-Party Logistics Study," Penn State University, Infosys, Penske Logistics, 2021.

[12] J. Bookbinder, "Global Logistics," Springer, 2012.

[13] M. Stopford, "Maritime Economics," Routledge, 2009.

[14] IATA, "IATA World Air Transport Statistics," 2020.

[15] Gartner, "Magic Quadrant for Supply Chain Planning Solutions," 2021.

[16] TechCrunch, "Flexport: The Digital Freight Forwarder," 2019.

[17] J. M. Bauer and M. Latzer, "Handbook on the Economics of the Internet," Edward Elgar Publishing, 2016.

[18] World Customs Organization, "WCO in Brief," 2021.

[19] K. Scott, "Robotics in Logistics: A DPDHL Perspective," DHL Customer Solutions & Innovation, 2016.