E-Logistics

By

Muhammad Abdullah

Muhammad Yasir

Muzzayan Ali

Sania Imad

Supervised by:

Asst. Prof. Mobeena Shahzad

Submitted to the faculty of Department of Computer Software Engineering,
Military College of Signals, National University of Sciences and Technology, Islamabad,
in partial fulfillment for the requirements of B.E Degree in Software Engineering.

June 2022

In the name of ALLAH, the Most benevolent, the Most Courteous

CERTIFICATE OF CORRECTNESS AND APPROVAL

This is to officially state that the thesis work contained in this report

"E-Logistics"

is carried out by

Muhammad Abdullah

Muhammad Yasir

Muzzayan Ali

Sania Imad

under my supervision and that in my judgement, it is fully ample, in scope and excellence, for the degree of Bachelor of Software Engineering in Military College of Signals, National University of Sciences and Technology (NUST), Islamabad.

Approved by

Supervisor
Asst. Prof. Mobeena Shahzad

Date: _____

DECLARATION OF ORIGINALITY

We hereby declare that no portion of work presented in this thesis has been submitted in support of another award or qualification in either this institute or anywhere else.

ACKNOWLEDGEMENTS

Allah Subhan'Wa'Tala is the sole guidance in all domains.

Our parents, colleagues, and most of all our supervisor, Asst. Prof. Mobeena Shahzad _ without your guidance this wouldn't be possible.

And all the group members, who through all adversities worked steadfastly.

Plagiarism Certificate (Turnitin Report)

This thesis has ____ similarity index. Turnitin report endorsed by Supervisor is attached.

_____

Muhammad Abdullah
00000278771

_____

Muhammad Yasir
00000278769

_____

Muzzayan Ali
00000242762

_____

Sania Imad
00000246685

_____

Signature of Supervisor

ABSTRACT

A web-based application intended to digitalize the Mass Transport Business and provide ease to small businesses to automate their management and functioning. The app is built and deployed in AWS Amplify. With React as front end and Node.js as back end, using Serverless framework for REST APIs. RDS is used for database engine. This project provides ease to its users in maintaining their transport business and goods all across the country by using our online advance resources and management support. It uses an algorithm that shows its users their required vehicle for trading that is nearest to them and lets them choose and contact hundreds of available drivers. Users get to choose from a vast network of available transporters. They can track their consignments. Get shipment insights and analytics. The system also provides secure payment gateways, allows any kind of online and bank transactions and users will be kept liable for the actions produced by any of the two parties. In short E-logistics aims to streamline, optimize, and automate the supply chain and bring innovation to the Mass Transport/ Logistics businesses of Pakistan.

Table of Contents

List of Figures

Chapter 1: Introduction

Engineering is not only limited to the smaller research areas, but it covers entire industrial setups from visual constructions on some software to on-site practical work. Engineers, who are practicing engineering principles, use their domain knowledge to invent and develop products that solve problems of the society.

With the advancement in engineering knowledge, automation has found a respectful place in the present research interests. Automation is a product of engineering research. It is the process in which machines are learned according to the existing events and are programmed in a way to make decisions in the future either by prediction or by the previous trends.

Automation has completely revolutionized research interests. Where it has been introduced, it has completely changed the on-ground realities. It is no shame to say that it has almost affected every domain of our lives.

The present-day problems demand automatic solutions that are efficient. Logistics management and administration problems such as booking, shipping, tracking, timely payments should also be dealt with solutions based on latest technology.

1.1 Overview

Today's world is a world of digitalization. With the exponential development in the supply chain industry the need for an automated logistics management solution has arisen. Unfortunately, Pakistan lags other countries in automated technologies and hence a need for a convenient, automated, and efficient systems are need of the hour. Logistics industry of Pakistan is around 23% of the service sector, 15% of public sector and gives roughly 5.4% of employment. The growth of the logistics industry rose by around 3.34% in 2019 and contributed 13.3% to the GDP.

Figure 1: Overview of Logistics Sector of Pakistan

Hence, our proposed solution is an automated logistics management system to digitalize logistics in Pakistan, streamline the supply chain management and contribute to our country's economic growth. It will bring innovation to the process of goods transportation, heavy vehicle transport booking, their payment transactions, history, record maintenance, tracking and secure logistic transport.

1.2 Problem Statement

Pakistan is a third world underdeveloped country. For logistics administration and management, traditionally manual paperwork is done, but it has many limitations that lead to ever-increasing hurdles.

Following are some highlights of the existing unautomated logistics management system:

1. There is a lot of paperwork hassle
2. Manual booking of desired vehicles
3. Clients face delayed responses
4. Lack of proper confirmation channels and communication barriers
5. Clients face tracking difficulties
6. Trust Issues related to theft and security
7. Issues of secure & timely payments

1.3 Proposed Solution

The major goal of our proposed solution is to automate and digitalize the logistics sector of Pakistan. It is an End-to-End logistics marketplace which intends to digitalize the Mass Transport Businesses in Pakistan and especially provides ease to small businesses in this industry and empowers them by automating their management and functioning. The main features that the system focuses on are as follows:

1. Online heavy vehicle Transport Booking
2. Live Consignment Tracking
3. Records and Ledger maintenance
4. Shipment Insights and Analytics
5. Intelligent Recommendation System
6. Secure Payment Integrations
7. Encrypted chat with the traders/transporters
8. Warning, danger and success alerts and notifications

1.4 Working Principle

It is a web application built and deployed in AWS Amplify. With React as Front End and Node.js as Back End, using Serverless framework for REST APIs. RDS is used for database engine.

The project is divided into different modulus and every module is inter-woven with the next module. The list of modules is as under:

- Launch Screen
- Register
- Login
- Consignment Request
- Bidding Module
- Request Forwarding
- Vehicle Adjustment
- Consignment Tracking
- Vehicle Recommendation
- Payment Gateway
- Accounting
- Chat

1.4.1 Launch Screen:

This will be our user first interaction page. This page will allow the user to login to their dashboard. This page will also be linked to registration page.

It will also show different steps required to perform a smooth working of a project for its end users to avoid any hurdle here after.

- On Login, user shall get JWT token or error message

- On Registration button, it will open registration page.

1.4.2 Registration:

The registration will be available for 5 entities: Industrialist, Transporter, Broker, Vehicle Owner, and Users.

- User will be able to register (Module Requirement)
- Each entity will be registered on separate collection with parent collection having user pool.
- It will display error, if required data is missing or schema mismatch.
- It will push events in stack if any error occurs. It will revert back.
- It will return error to user if any registration error occurs.
- It will send user-email verification link.
- It will send success response on successful registration.
- It will not allow if company or email is already registered.

1.4.3 Login:

Each user will login using email and password and will redirect to Dashboard upon successful sign in.

- It will accept and verify user login request.
- It will verify email and password.
- It will send JWT token on successful verification.
- It will show error if email is not verified.
- It will redirect to Dashboard.

- It will show error if account is not verified.

1.4.4 Consignment Request:

In consignment request, Industrialist will generate request on by putting specifications for consignment. Transporter will receive consignment and will create response on it.

- User will send consignment request.
- User will see status or response of every consignment.
- User will send consignment request to specific Transporter.
- User will perform Create, Read, Update and Delete operation.

1.4.5 Bidding:

In Bidding, every entity will create Bid on Respective consignment.

- Transporter will see recommended consignment request.
- Transporter will Bid on consignment Request.
- Transporter will see bided request.
- Industrialist will see bid on request.
- Industrialist will response on bid.
- Industrialist will negotiate on bid.

1.4.6 Request Forwarding:

In Request Forwarding, Transporter will forward request to Broker.

- Transporter will forward request to receive bidding.

- Transporter will receive response on request.
- Transporter will negotiate on Bid.

1.4.7 Vehicle Adjustment:

In Vehicle Adjustment, there is coordination between Broker and Vehicle Owner.

- Broker will send consignment request details to vehicle Owner.
- Vehicle owner will reply on consignment details.
- Broker will negotiate using built in service.
- Vehicle Owner will send proposal on successful deal.

1.4.8 Consignment Tracking:

In this module, all entities will track vehicle in their domain.

All entities can see live tracking of vehicle with coordinates.

1.4.9 Vehicle Recommendation:

For direct consignment, client will locate nearby vehicle for luggage transport.

- Client will put consignment details.
- Client will find nearby vehicles.

1.4.10 Payment Gateway:

On successful delivery, payment will be made. Payment will be To Pay or on contract basis.

- Transporter will make payment Topay.
- Payment will be pending on deal done.
- All entity will might pay complete, partial payment.

1.4.11 Accounting:

On successful delivery, payment will be made. Payment will be To Pay or on contract basis.

- Users will see revenue.
- User will see profit.
- Users will see expenses.

1.4.12 Chat:

In chat, all entities may communicate with one another.

- User will send message.
- User will receive message.
- User will send document.
- User will receive document.

1.5 Objectives

1.5.1 General Objectives:

"To build an innovative state of the art software to digitalize and automate the supply chain management and administration in the Logistics sector of Pakistan, powered by the latest technology stack, Machine Learning algorithms and techniques, providing a smart administrative solution to minimize manual operations."

1.5.2 Academic Objectives:

- Development of a smart Logistics Management System
- Node.js learning and its Architecture style
- Mode of payment methods and their integration

- JWT usage and its authentication
- Complete web automation
- User data protection and encryption of confidential data
- To implement Machine Learning techniques
- Speed and performance
- MVC implementation
- To increase productivity by working in a team
- To design a software that contributes to the welfare of society

1.6 Scope

This product will ease its users in maintaining their business and goods all across Pakistan using our End-to-End Logistics Marketplace. It optimizes the supply chain management, makes booking and shipping alot easier, ensures safe and timely payments, allows real time tracking, has an intelligent recommendation system and also provides shipment insights and analytics. It will use an algorithm that will show its users their required vehicle for trading that is nearest to them and let them choose and contact hundreds of available drivers and vehicles. Users will find what is suitable for them and will make a deal over the platform. The system will allow any kind of online and bank transactions and will be kept liable for the actions produced by any of the two parties.

1.7 Deliverables

1.7.1 Software Requirement Specification:

The purpose of this document is to present a detailed description of E-Logistics. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, its entire process, the constraints under which it must operate and how the system will react to external stimuli. This document is intended for both the stakeholders and the users.

1.7.2 Software Architecture Document:

In this document, the overall architecture of the system is discussed, including the introduction of various components and subsystems. It is mainly supported by system Architecture diagram which shows an insider's perspective of the system by describing the high-level software components that perform the major functions to make the system operational.

1.7.3 Software Design Document:

The design document captures all our functional requirements and shows how they interact with each other conceptually. The low-level design also shows as to how we have been implementing how we are going to implement all of these requirements.

1.7.4 Implementation Code Document:

The implementation code document provides details about the pseudo code for the application and project prototype.

1.7.5 Software Testing Document:

This document has testing modules in which there are certain test cases which depicts the correctness and accuracy of the project.

1.7.6 Final Project Report

This is the thesis report which compiles all the previous and current working for the project. Thesis report provides the whole summary for the project and also give details about each and every aspect of the project starting from introduction of the project, literature review, requirements leading to design discussions then testing and lastly future work and conclusion.

1.8 Relevant Sustainable Development Goals

E-Logistics aligns with the UNs' Sustainable Development Goal no. 9 which promotes developing "Innovative and Inclusive Industry Infrastructures" and extending enhanced technical and technological support for sustainable economic growth. It is an End-to-End logistics marketplace which intends to digitalize the Mass Transport Businesses in Pakistan and especially provides ease to small businesses in this industry and empowers them by automating their management and functioning and provides them an efficient technological solution.

1.9 Structure of Thesis

Chapter 2 contains the literature review and the background and analysis study this thesis is based upon.

Chapter 3 contains the design and development of the project.

Chapter 4 introduces detailed evaluation and analysis of the code.

Chapter 5 contains the conclusion of the project.

Chapter 6 highlights the future work needed to be done for the commercialization of this project.

Chapter 2: Literature Review

A new product is launched by modifying and enhancing the features of previously launched similar products. Literature review is an important step for development of an idea to a new product. Likewise, for the development of a product, and for its replacement, related to logistics system, a detailed study regarding all similar projects is compulsory. Our research is divided into the following points.

- Industrial Background
- Existing solutions and their drawbacks
- Research Papers

2.1 Industrial background

In today's era, technological advances and automation of manual systems has given rise to use of ease and optimal way of moving forward. Unfortunately, one of the major issues faced across Pakistan is the lack of automated systems as compared to the rest of the world. Manual operation of logistics in the supply chain has led to many further problems, as discussed in Problem Statement, which increases need for a smart automated system for the management and administrative support. Ultimately, results in a big marketplace for industrial development in the logistics sector contributing positively towards boosting the economy.

Initially, Pakistani Logistic Industry was operated completely manually which gave rise to many complexities. As businesses grew and the market expanded, the need for a proper logistics system became the need of the hour. Businesses now look for a hassle free solution to transport their goods across the country. And now industries are inclining towards smart industries, automation, based on latest technologies (Internet of Things (IOT), Machine Learning (ML) techniques, Artificial Intelligence (AI)). Hence, a smart logistics management system provides good market growth and impacts economy directly as it is a fully automated system.

2.2 Existing solutions and their drawbacks

No Remarkable work has been done in Pakistan to automate the logistics sector. Most companies are operating manually. Even those companies who do have a proper automated logistics system, their domain is restricted to a limited scope such as themselves or large enterprises. Due to which small businesses and startups face difficulty in managing their operations as they are unable to facilitate and accommodate an automated logistics management channel and hence their market expansion is slow.

In addition, there is a need for a smart End-to End Logistics Marketplace which will benefit not only the large enterprises but will provide ease even to the small businesses and will digitalize the logistics sector of Pakistan.

2.2.1 NLC

NLC is one of Pakistan's well known Logistics company providing services to big enterprises such as Uniliever, Ghani, Nestle, Package Limited. However small enterprises who have not yet contracted with them cannot avail their services and hence they have to struggle to find a digital logistics system suited to meet their needs. Moreover, NLC only

provides internal vehicles available to them and so their vehicle choice is limited in choice. E-Logistics is being introduced to overcome these shortcomings. E-Logistics will provide ease of logistics management and administration even to the upcoming start-ups to enhance their market growth thus empowering initiatives and enhancing progress making a positive impact towards economy. Moreover, the scope of E-Logistics is not limited to a certain vehicle provider but a customer will be able to choose any vehicle suited to meet his business needs from any vehicle provider registered with E-Logistics.

Chapter 3: Design and Development

3.1 System Overview

The system will be a web-based application. The front end will be based on React whereas the backend will be based on NodeJS. The application will require all the users to have their own profile to preserve confidentiality, authenticity, and integrity. The application can be operated on all operating systems. A mobile application of the system will also be considered.

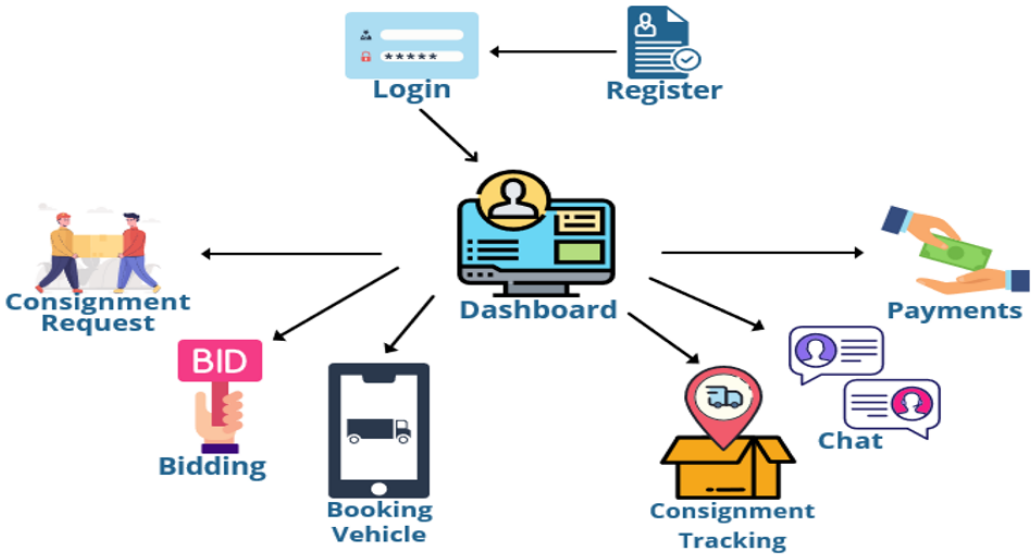

Figure 2: System Overview

3.2 Architecture

3.2.1 Architecture Design

The architectural design of E-Logistics is Event Driven Architecture. Upon the detection of an event, the system acts accordingly. Each front-end React file has a Nodejs back-end file. The back-end waits for an event to occur and changes are made to the front-end accordingly.

3.2.2 Module Decomposition

Registration allows user to register as Industrialist, Transporter, Broker, Vehicle Owner or normal user

Login Screen allows users to input their credentials to access the system.

Vehicle Adjustment allows the users to select the type of vehicle.

Consignment allows the user to **request** for their choice of consignment and also to **track** them once booked.

Bidding allows the user to **place bidding requests** against their choice of vehicle.

Chat allows the user to chat with one another and also with the technical staff in case of any problems faced.

Payments allow the user to view their pending payments and also to make them.

Tracking allows the users to track their vehicle.

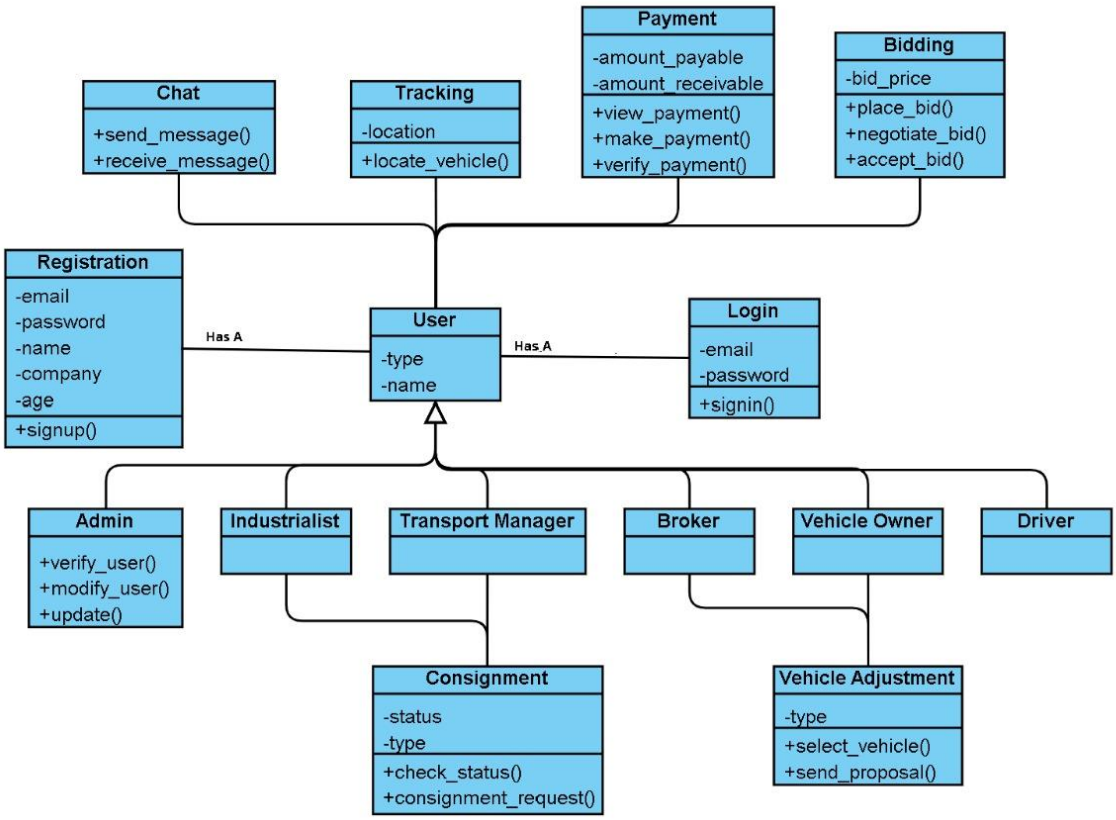

Figure 3: Class Diagram

3.2.3 Process Decomposition

The process decomposition is explained through sequence and use case diagrams which decomposes the system into well-defined and cohesive processes. The use cases explain the set of actions that a user undertakes while dealing with E-Logistics.

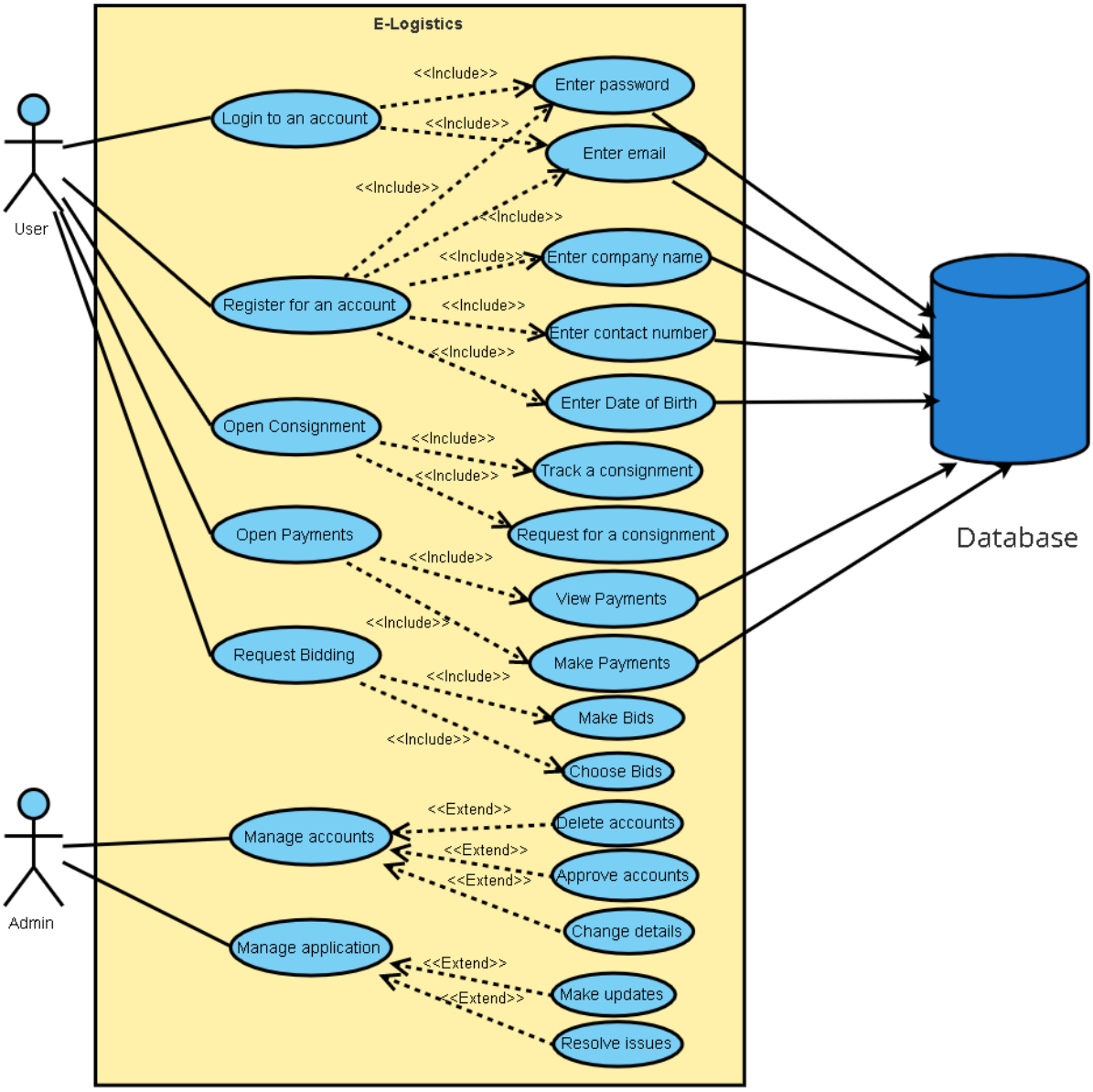

Figure 4: Use Case Diagram

There are two primary actors. User and Admin. The user can perform the desired functionality while the admin is basically concerned with the normal operation of the system and providing with updates when available.

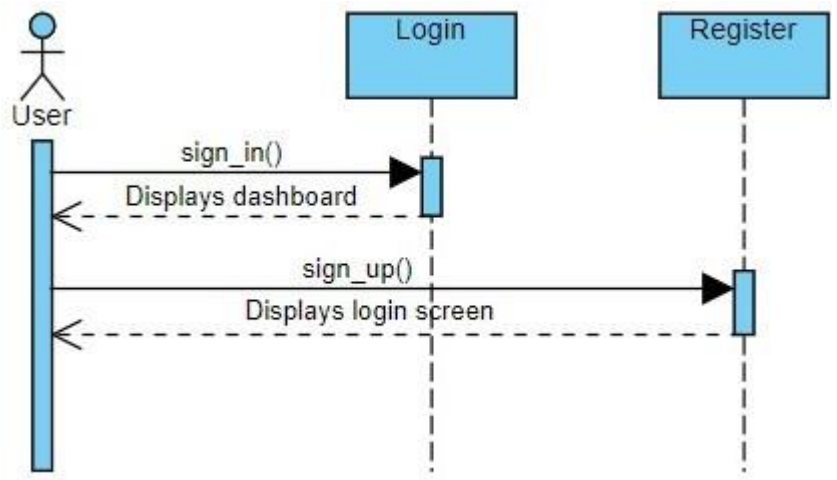

Figure 5: Sequence diagram for Login, and Register

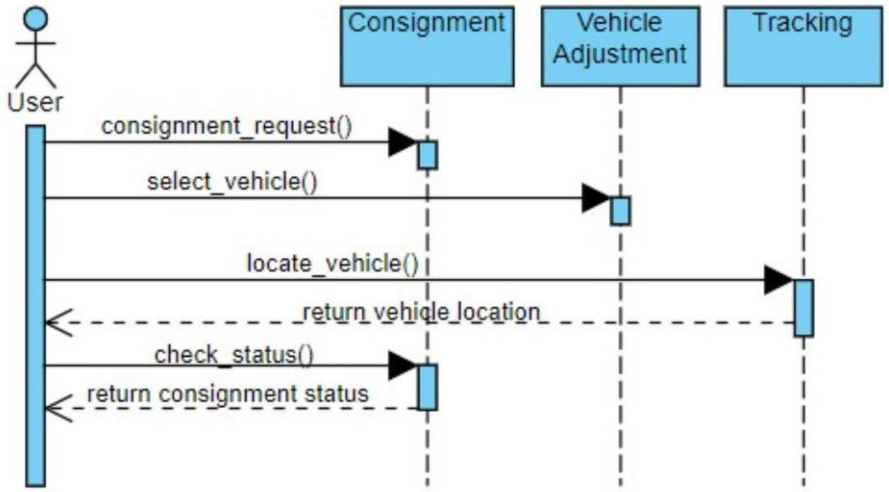

Figure 6: Sequence diagram for Consignment, Vehicle Adjustment, and Tracking

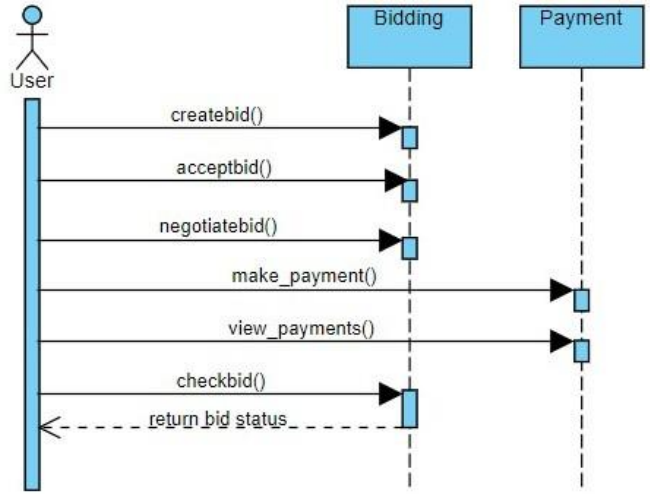

Figure 7: Sequence diagram for Bidding and Payments

3.2.4 Design Rationale

The architecture chosen for E-Logistics is **Event Driven architecture.**

Event Driven Architecture has been chosen because it provides effective solutions for designing and implementing complex systems. Event-driven architecture is a software architecture and model for application design. With an event-driven system, the capture, communication, processing, and persistence of events are the core structure of the solution. This differs from a traditional request-driven model. An event-driven architecture is loosely coupled because event producers don't know which event consumers are listening for an event, and the event doesn't know what the consequences are of its occurrence.

Event-driven architecture is made up of event producers and event consumers. An event producer detects or senses an event and represents the event as a message. It does not know the consumer of the event, or the outcome of an event.

After an event has been detected, it is transmitted from the event producer to the event consumers through event channels, where an event processing platform processes the event asynchronously. Event consumers need to be informed when an event has occurred. They might process the event or may only be impacted by it.

3.3 Component Design

In this section of component design, we will take a closer look at each component of E-Logistics in a more systematic way. Each component will have a functional description and closer detail.

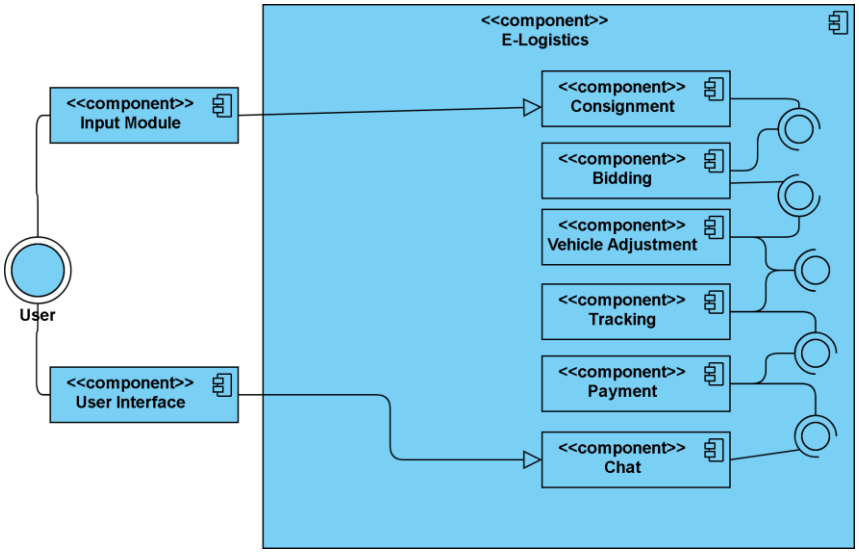

Figure 8: Component Diagram

Chapter 4: System Implementation and Testing

4.1 Pseudo Code for APIs

4.1.1 Login API

```
import { response } from "../../layers/helpers/response";

import { Validator } from "jsonschema";

import { login_schema } from "./schema";

import { validation_errors } from "../../layers/helpers/validate_errors_extraction";

import middy from "@middy/core";

// import some middlewares
import jsonBodyParser from "@middy/http-json-body-parser";
import { UserController } from "../../layers/controllers/user_controller";
import { EMPTY_BODY_ERROR, SERVER_ERROR } from "../../AppConstants";
import { is_valid_body } from "../../layers/helpers/verify_empty_body";

async function login_handler(event) {
  if (!is_valid_body(event.body)) {
    return EMPTY_BODY_ERROR;
  }
  let validate_res = new Validator().validate(event.body, login_schema);
  if (!validate_res.valid) {
    return response(400, {
      body: {
        errors: validation_errors(validate_res.errors),
      },
    });
  }

  try {
    let token = await new UserController(event.body.email).authenticate_user(
      event.body.password
    );
    return response(200, {
```

```
      token,
      user: event.body.email,
    });
  } catch (err) {
    if (err.statusCode || err)
      return response(err.statusCode, {
        body: {
          message: err.message,
        },
      });
    return SERVER_ERROR;
  }
}

export const login = middy(login_handler).use(jsonBodyParser());
```

4.1.2 Authentication

```
import * as jwt from "jsonwebtoken";
import { EXPIRE_JWT, HASH_KEY } from "../../AppConstants";
import { DB } from "../../config/db_config";
import { CustomError } from "../helpers/error";
import { USER } from "../models/users";
const crypto = require("crypto");

require("dotenv").config();

const conf = process.env;

/**
 * Middleware for user auth
 * Verifies the token in header and create user session
 * create error for unverified token
 * Scheme Bearer
 */
export const TokenVerifier = (req) => {
  var token =
    req.headers["authorization"]?.split(" ")[1] ||
    req.headers["Authorization"]?.split(" ")[1];
  if (!token) {
```

```
    throw new CustomError(403, "User token is required");
  } else {
    try {
      const decoded = jwt.verify(token, conf.JWT_SECRET);
      return decoded;
    } catch (err) {
      throw new CustomError(401, "Invalid token in header");
    }
  }
};

export const login = async (email, password, user = null) => {
  if (user) {
    delete user.password;
    const token = jwt.sign(user, conf.JWT_SECRET, {
      algorithm: "HS256",
      expiresIn: EXPIRE_JWT,
    });
    return token;
  } else {
    var sequelize = null;
    try {
      sequelize = await new DB().connect();
    } catch (error) {
      throw new CustomError(500, "DB connection Failed");
    }

    user = await USER(sequelize).findOne({ where: { email } });

    if (user) {
      user = user.dataValues;

      // console.log(user);
      if (generate_hash(password) != user.password) {
        throw new CustomError(403, "User name or password is incorrect");
      } else {
        delete user.password;
        const token = jwt.sign(user, conf.JWT_SECRET, {
          algorithm: "HS256",
          expiresIn: EXPIRE_JWT,
```

```
      });
      return token;
    }
   } else {
    throw new CustomError(403, "User name or password is incorrect");
   }
  }
};

export const generate_hash = (plain) => {
  return crypto.createHmac("sha256", HASH_KEY).update(plain).digest("hex");
};
```

4.1.3 Serverless Configuration

```
provider:
  name: aws
  runtime: nodejs14.x
env:
  functions: "src/functions"

# you can overwrite defaults here
stage: dev
region: us-east-1
custom:
  serverless-bundle:
    sourcemaps: true
  s3:
    host: localhost
    directory: ./tmp
    port: 4569

plugins:
  - serverless-offline
  - serverless-bundle
  - serverless-s3-local
functions:
  # Registeration Handlers
  register:
    handler: ${self:env.functions}/user_handler/register.register
```

```
  events:
    - httpApi:
        path: /signup
        method: post

#Complete User registeration
industrialist_registeration:
  handler: ${self:env.functions}/user_handler/complete_reg.handler
  events:
    - httpApi:
        path: /registeration/complete
        method: post
# User Authentication Handlers
login:
  handler: ${self:env.functions}/login_handler/login.login
  events:
    - httpApi:
        path: /signin
        method: post

# Order Handlers
place_order:
  handler: ${self:env.functions}/consignment_handler/place_order.place_order
  events:
    - httpApi:
        path: /order/place
        method: post
register_vehicle:
  handler: ${self:env.functions}/vehicle_handler/register.register
  events:
    - httpApi:
        path: /vehicle/register
        method: post

# File handler
get_file:
  handler: ${self:env.functions}/file_handler/get_file.handler
  events:
    - httpApi:
        path: /file/{key}
```

```
        method: get
resources:
  Resources:
    tmp:
      Type: AWS::S3::Bucket
      Properties:
        BucketName: tmp
```

4.1.4 App Constants

```
export const HASH_KEY = "sds#@csaas1s$dds#ds.$dszxc";
export const EXPIRE_JWT = 604800;

export const SERVER_ERROR = {
  statusCode: 500,
  body: JSON.stringify({
    message: "An unknown error occured",
  }),
};

export const EMPTY_BODY_ERROR = {
  statusCode: 400,
  body: JSON.stringify({
    message: "Empty body error",
  }),
};

export const UNVERIFIED_USER = {
  statusCode: 405,
  body: JSON.stringify({
    message: "Un allowed method, user is not verified",
  }),
};

export const FORBIDDEN_ACTION = {
  statusCode: 403,
  body: JSON.stringify({
    message: "User type is not allowed to perform this transaction",
  }),
};
```

```
export const FORBIDDEN_FILE_TYPE = {
  statusCode: 401,
  body: JSON.stringify({
    message: "File type not allowed",
  }),
};

export const ALLOWED_FILES = ["application/pdf"];
```

4.1.5 Consignment Handler

```
import { response } from "../../layers/helpers/response";

import { Validator } from "jsonschema";

import { order_schema } from "./schema";

import { validation_errors } from "../../layers/helpers/validate_errors_extraction";

import middy from "@middy/core";

// import some middlewares
import jsonBodyParser from "@middy/http-json-body-parser";

import {
  EMPTY_BODY_ERROR,
  SERVER_ERROR,
  UNVERIFIED_USER,
} from "../../AppConstants";
import { is_valid_body } from "../../layers/helpers/verify_empty_body";
import { TokenVerifier } from "../../layers/modules/user_auth";
import { ConsignmentController } from "../../layers/controllers/consignment_controller";

async function place_order_handler(event) {
  if (!is_valid_body(event.body)) {
    return EMPTY_BODY_ERROR;
  }
  let validate_res = new Validator().validate(event.body, order_schema);
  if (!validate_res.valid) {
    return response(400, {
```

```
      body: {
        errors: validation_errors(validate_res.errors),
      },
    });
  }

  try {
    const user = TokenVerifier(event);
    // if (!user.is_verified) {
    //   return UNVERIFIED_USER;
    // }

    let order = await new ConsignmentController().place_order(
      user.id,
      event.body
    );

    return response(200, {
      message: "Order placed",
      order_id: order.id,
    });
  } catch (err) {
    if (err.statusCode)
      return response(err.statusCode, {
        body: {
          message: err.message,
        },
      });
    return SERVER_ERROR;
  }
}

export const place_order = middy(place_order_handler).use(jsonBodyParser());
```

4.1.6 Bidding and Negotiation

```
import { CustomError } from "../helpers/error";
import { NEGOTIATION } from "../models/negotiation";
import { User } from "./user_module";

export class Bidding {
```

```
  constructor(user) {
   this.user = user
  }

  doUserNegotiation(bid) {
    var sequelize = null;
    try {
      sequelize = await new DB().connect();
    } catch (error) {
      throw new Error("DB connection Failed");
    }

    try {
      bid['fromUserId'] = this.user.id

      let toUser = await new User().getUser(bid.toUserId)
      if(!toUser){
        throw new CustomError(400, "Invalid To User")
      }else if(toUser.user_type == this.user.user_type){
        throw new CustomError(401, "Similar user type cannot exchange negotiations")
      }else if(toUser.user_type == "industrialist" && this.user.user_type == "broker"){
        throw new CustomError(401, "Direct connection not allowed")
      }else if(toUser.user_type == "transporter" && this.user.user_type ==
"vehicle_owner"){
        throw new CustomError(401, "Direct connection not allowed")
      }


      let negotiation = NEGOTIATION(sequelize).build(bid)

       await sequelize.sync()
      let prev = NEGOTIATION(sequelize).find({
        where:{
          orderId: bid.orderId,
          fromUserId: this.user.id,
          toUserId: bid.toUserId
        }
      })
```

```
    if(prev){
      throw new CustomError(403, "Action not allowed, bid already exists")
    }
    return await negotiation.save()
  } catch (error) {
    if (
      error.name == "SequelizeUniqueConstraintError" &&
      error.errors &&
      error.errors[0]
    ) {
      throw new Error(error.errors[0]["message"]);
    }
    throw new Error("Cannot place bid/ negotiation");
  }

}

getNegotiation(filter){
  var sequelize = null;
  try {
    sequelize = await new DB().connect();
  } catch (error) {
    throw new Error("DB connection Failed");
  }

  try {
    let negotiation = await NEGOTIATION(sequelize).findAll({
      where: filter
    })

    return negotiation
  } catch (error) {
    if (
      error.name == "SequelizeUniqueConstraintError" &&
      error.errors &&
      error.errors[0]
    ) {
      throw new Error(error.errors[0]["message"]);
    }
    throw new Error("Cannot find bid/ negotiation");
```

```
    }
  }
}
```

4.1.7 File Handler

```
import { v4 as uuidv4 } from "uuid";
import * as AWS from "aws-sdk";

const s3 = new AWS.S3({
  s3ForcePathStyle: true,
  accessKeyId: "S3RVER",
  secretAccessKey: "S3RVER",
  endpoint: new AWS.Endpoint("http://localhost:4569"),
});
export const saveFiles = async function (files) {
  var keys = [];
  let promises = await Promise.all(
    files.map(function (val) {
      let Key = uuidv4() + "_$name$_" + val.file.filename;
      keys.push(Key);
      let params = {
        Bucket: "tmp",
        Key,
        Body: Buffer.from(val.file.content, "ascii"),
      };

      return s3.putObject(params).promise();
    })
  );

  return promises.map(function (val, i) {
    return {
      ETag: val.ETag,
      filename: files[i].file.filename,
      key: keys[i],
      key_name: files[i].key,
    };
  });
};
```

```
export const getFile = function (key) {
  return s3
    .getObject({
      Key: key,
      Bucket: "tmp",
    })
    .promise();
};
```

4.1.8 Vehicle Handler

```
import { response } from "../../layers/helpers/response";

import { Validator } from "jsonschema";

import { vehicle_schema } from "./schema";

import { validation_errors } from "../../layers/helpers/validate_errors_extraction";

import middy from "@middy/core";

// import some middlewares
import jsonBodyParser from "@middy/http-json-body-parser";
import {
  EMPTY_BODY_ERROR,
  FORBIDDEN_ACTION,
  SERVER_ERROR,
} from "../../AppConstants";
import { is_valid_body } from "../../layers/helpers/verify_empty_body";
import { VehicleController } from "../../layers/controllers/vehicle_controller";
import { TokenVerifier } from "../../layers/modules/user_auth";

async function register_handler(event) {
  const user = TokenVerifier(event);
  // if (!user.is_verified) {
  //   return UNVERIFIED_USER;
  // } else if (user.type != vehicle_owner) {
  //   return FORBIDDEN_ACTION;
```

```
  // }

  if (!is_valid_body(event.body)) {
    return EMPTY_BODY_ERROR;
  }
  let validate_res = new Validator().validate(event.body, vehicle_schema);
  if (!validate_res.valid) {
    return response(400, {
      body: {
        errors: validation_errors(validate_res.errors),
      },
    });
  }

  try {
    let vehicle = await new VehicleController(user.id).registerNew(event.body);

    return response(200, {
      vehicle,
    });
  } catch (err) {
    console.log(err);
    if (err.statusCode)
      return response(err.statusCode, {
        body: {
          message: err.message,
        },
      });
    return SERVER_ERROR;
  }
}

export const register = middy(register_handler).use(jsonBodyParser());
```

4.2 User Interface

UI is designed according to UI design principles.

The structure principle: UI is organized in such a way that related things are combined together and unrelated things are separated.

The simplicity principle: It is easy to follow the provided interface.

The visibility principle: All system's functions are available through UI. It does not overwhelm users with too many alternatives.

The reuse principle: In design, same names were used to perform the same operations with different objects in order to reduce ambiguity.

There are a total of 4 main pages in the user interface: Main Window, Login Screen, Registration and Dashboard.

Chapter 5: Conclusion

In this thesis, we discussed an automated logistics management and administrative system that aims to digitalize the logistics sector making operations efficient and every transaction hassle free. Businesses especially the smaller ones and the startups will now be able to book their type of vehicle according to their needs from within their area of ease saving time, money and helping them expand their market reach. Our proposed system has a major advantage over the current logistics system which is manual and still includes paperwork while the whole world has moved towards automation and digitalization. In addition, our system will overshadow traditional systems due to the latest algorithms used for searching and suggestions. We are using the latest technology available for the purpose of this project such as React and Node Js., AWS Amplify and RDS; are also briefly explained included their working and importance. The purpose of increasing productivity and overcoming problems in existing solutions is being achieved by using the modern techniques. Additionally, the objectives; an automated logistics management and administrative operations, ease of use, are attained. Hence saves time as well as the prestigious goods of our customers.

Our proposed system, E-Logistics, is cost-effective as it is purely made for the core purpose of serving Pakistan, any system that could be beneficial to its citizens. Else, similar solutions provided by other countries are very costly. Moreover, E-Logistics provides an ease to adoption which can be adopted by beneficiaries, mass deployment can also be done and most importantly no product training is required.

Chapter 6: Future Work

Future milestones that need to be achieved to commercialize this project are the following.

6.1 Access to real life logistics marketplace:

The main objective of this project is to produce a product that is efficient and can be easily used by anyone in this industry. Physical access to a logistics marketplace is vital.

6.2 Incorporating IoT devices:

The Internet of Things (IoT) can be used in various ways such as, tracking vehicles, people and assets, monitoring storage and delivery conditions, and improving on-site safety and security in logistics services to help in offering solutions that improve the productivity of supply chains and, therefore, the profitability of companies and businesses. IoT technology can help businesses serve their customers better, while also saving the business money and improving overall efficiency.

References and Work Cited

1. E A Guseva *et al* 2021 *IOP Conf. Ser.: Mater. Sci. Eng.* **1064** 012014. Automation of individual operations of the transport process to create sufficient conditions for the efficient functioning of digital transport and logistics

2. Kayikci, Yasanur. "Sustainability impact of digitization in logistics." Procedia manufacturing 21 (2018): 782-789.

3. Azmi, Izwan & Abdul Hamid, Norlida & Nasarudin, Md & Ibtishamiah, Nik. (2017). Logistics and supply chain management: The importance of integration for business processes. Journal of Emerging Economies and Islamic Research. 5. 10.24191/jeeir.v5i4.8838.

4. Zijm, Henk, and Matthias Klumpp. "Logistics and supply chain management: developments and trends." In Logistics and Supply Chain Innovation, pp. 1-20. Springer, Cham, 2016.

5. Sweeney, E. (2012). The people dimension in logistics and supply chain management: its role and importance. In R. Passaro, & A. Thomas (Eds.), Supply chain management: perspectives, issues and cases (pp. 73-82). McGraw-Hill.

6. Dekhne, A., Hastings, G., Murnane, J., & Neuhaus, F. (2019). Automation in logistics: Big opportunity, bigger uncertainty. *McKinsey Q*, 1-12.

7. Dadzie, K. Q., & Johnston, W. J. (1991). Innovative automation technology in corporate warehousing logistics. *Journal of Business Logistics*, *12*(1), 63.

8. Ballou, R.H. (2007), "The evolution and future of logistics and supply chain management", European Business Review, Vol. 19 No. 4, pp. 332-348.

9. Hofmann, E., Sternberg, H., Chen, H., Pflaum, A. and Prockl, G. (2019), "Supply chain management and Industry 4.0: conducting research in the digital age", International Journal of Physical Distribution & Logistics Management, Vol. 49 No. 10, pp. 945-955.

10. Smart supply chain management: a review and implications for future research. Lifang Wu, Xiaohang Yue, Alan Jin, David C. Yen, The International Journal of Logistics Management.

Plagiarism Report