



Software Engineering Department
ORT Braude College

Capstone Project Phase B – 61998

Flying Packages

23-1-D-29

Advisor: Zeev Barzily

Student 1: Mosa Srour-205798812

E-mail 1: mosa.srour.s9@gmail.com

Student 2: Wessam Ghrayeb-211756820

E-mail 2: wesamghrayeeb1@gmail.com

GIT Link: <https://github.com/MousaSrour/FlyingPackages.git>

Contents

Abstract	4
1. Introduction	4
2. Background and Related Work	5
2.1 Delivery management apps	5
2.2 NodeJS	5
2.3 React.JS	7
2.3.1 React.JS vs AnguarJS:	7
2.4 MongoDB	8
3. Expected Achievements	9
3.1 Outcomes	9
3.2 Data Security	9
3.3 Unique Features	10
3.3.1 Finding the nearest courier	10
3.3.2 GPS Tracking	10
3.3.3 Alerts and Notifications	11
3.4 Criteria for Success	12
4. Engineering Process	12
4.1 Research – Delivery Management	12
4.1.1 Constraints and Challenges – Delivery Management	13
4.2 Methodology and Development Process	14
4.2.1 Planning and preparation	14
4.2.2 Implementation	14
4.2.3 Iterative development	14
5. Product	15
5.1 MERN-STACK	15
5.2 Requirements	16
5.3 UML Diagrams	17
5.3.1 Use Case Diagram	17
5.3.2 Class Diagram	18
5.3.3 Activity Diagram	20
6. User Documentation	22
6.1 User Guide	22
6.2 Maintenance Guide	35
6.3 DataBase Guide	36

6.4 System Operating Environment.....	36
7. Verification and Evaluation	37
7.1 Unit Tests	37
7.2 Functional Tests	38
8. Results and conclusions.....	39
9. References.....	40

Abstract

Our delivery management system is called “Flying Packages”, its an online platform that connects suppliers and organizations with couriers who work for us. Our system enables suppliers to select a particular courier for deliveries, where we suggest only the K closest couriers. This approach allows us to optimize the company's performance and minimize delivery time. This ensures that orders are delivered in a timely and efficient manner. Suppliers can utilize our system by purchasing our service and accessing our website. Through the website, they can promptly update orders upon receiving them. Once an order is submitted, our system will automatically forward it to the designated courier, who will be notified and prepared to execute the delivery. Additionally, the system provides real-time tracking for the status of the orders, allowing suppliers and couriers to easily monitor the status of deliveries. The goal of the system is to improve the delivery process by automation and simplification the coordination and assignments of orders. The system is designed to be user-friendly and easy to navigate, making it accessible to businesses of all sizes.

1.Introduction

Our project aims to simplify delivery management for small businesses by providing a user-friendly platform that connects suppliers and couriers in real-time. By considering factors such as courier availability, distance, and cost, our technology helps ensure shipments are delivered quickly and cost-effectively.

The stakeholders of our project include suppliers, couriers, and the general public who rely on reliable deliveries. Our goal is to support all these groups and improve the delivery process for everyone involved.

We are excited to introduce our platform, as we believe it is the missing piece in the delivery management puzzle. Our technology enables suppliers to easily find and hire trustworthy couriers while offering couriers a selection of delivery tasks that align with their schedules and preferences. Our aim is to streamline the delivery process, eliminate unnecessary fees and delays, and make life easier for all parties involved.

2. Background and Related Work

Delivery Operation has been a crucial concern for businesses for numerous times, with colorful approaches and technologies developed to streamline the process. former work in this area includes the use of GPS shadowing to cover deliveries, as well as the development of software platforms that connect suppliers with couriers. still, numerous of these results are limited in their compass or bear homemade trouble to find the efficient courier for delivery.

2.1 Delivery management apps

There are many delivery management apps that allow businesses to plan, optimize, and track routes for their deliveries. It is essentially targeted at small and medium businesses, such as courier services, and food delivery businesses.

Some features of this apps include:

Allows users to enter multiple addresses for their deliveries and then optimizes the route for the most efficient order of stops.

Allows users to track their vehicles and drivers in real-time, including their location, speed, and the estimated time of arrival.

Users can send notifications to their customers about the status of their deliveries, including when the delivery has been dispatched, is the end route, or has been completed.

Includes features for managing and scheduling vehicles, including the ability to set maintenance reminders and track fuel usage.

In other words, the platforms that are available today help a lot of companies, but still don't have the perfect solution to interactive between the supplier and the courier, and this what we expect to achieve.

2.2 NodeJS

Node.js is a server-side runtime environment that allows JavaScript code to be executed. It is based on Chrome's V8 JavaScript engine, which offers excellent performance and scalability.

One of the primary advantages of Node.js is its event-driven architecture, which operates on a non-blocking model. By utilizing an event loop, Node.js efficiently handles multiple concurrent connections. This asynchronous approach enables Node.js to handle a large volume of requests without becoming blocked, making it ideal for real-time communication and applications with intensive input/output operations.

There are several reasons why Node.js is regarded as superior when compared to other server-side technologies:

Performance: Node.js delivers outstanding performance due to its lightweight and efficient event-driven architecture. It can effortlessly handle numerous concurrent connections while maintaining low latency.

Scalability: Node.js excels in creating scalable applications. Its event-driven nature allows it to manage a high number of concurrent requests with minimal resource consumption.

NPM (Node Package Manager): Node.js benefits from a vast ecosystem of packages and libraries accessible through NPM. This extensive collection of open-source modules empowers developers to seamlessly integrate third-party functionalities into their applications.

Cross-platform Compatibility: Node.js is compatible with major operating systems such as Windows, macOS, and Linux. This makes it a versatile choice for developing server-side applications that can run on various platforms.

JavaScript Everywhere: Node.js facilitates the development of full-stack JavaScript applications by enabling the use of JavaScript on both the front-end and back-end. This eliminates the hassle of switching between different programming languages, making development more seamless and promoting the reuse of code.

Developer Productivity: Node.js enhances developer productivity through its lightweight and modular approach. It provides a variety of tools, frameworks (like Express.js), and development environments that assist in speedy development and efficient code organization. Furthermore, the extensive community support surrounding Node.js offers valuable resources, tutorials, and libraries to aid developers in their projects.

2.3 React.JS

React.js is a JavaScript library known for its effectiveness, simplicity, and usefulness in web development. It is an ideal tool for creating user interfaces (UI) due to its component-based architecture, virtual DOM, declarative syntax, reusability, active community, and one-way data flow.

By using React.js, developers can build UIs by breaking them down into reusable components, making development more efficient and manageable. These components encapsulate their logic and UI, enabling easier maintenance and reusability.

The virtual DOM in React.js ensures faster rendering and improved performance by updating only the necessary parts of the UI. This optimization saves time and resources.

React.js adopts a declarative syntax, allowing developers to describe how the UI should look based on its current state. This simplifies code readability and reduces the likelihood of errors.

The reusability of components in React.js promotes efficiency by enabling developers to leverage existing components across different parts of an application. This streamlines development and ensures consistency.

React.js has a large and vibrant community of developers who contribute to its growth. This community provides a wealth of resources, tutorials, and libraries, supporting developers in learning and enhancing their skills.

The one-way data flow in React.js ensures a clear understanding of how data changes propagate through the application, simplifying debugging and maintenance.

Overall, React.js is an effective, simple, and valuable tool for web development, providing a streamlined approach to building UIs and delivering a smooth user experience.

2.3.1 React.JS vs AngularJS:

When comparing React.js and AngularJS, we have chosen React.js as the preferred framework for our project. Here are the reasons why React.js is the right choice:

1. **Easy to Learn:** React.js has a simpler syntax and a shallower learning curve compared to AngularJS. This makes it more accessible for developers, especially those who are new to frontend development.
2. **Flexibility:** React.js offers more flexibility in terms of architecture and integration with other tools and libraries. It focuses on the view layer, allowing us to select and combine additional technologies based on our project's specific requirements.

3. Performance: React.js is highly regarded for its performance. Its efficient virtual DOM implementation ensures fast rendering by updating only the necessary components, resulting in better overall performance.

4. Active Community: React.js has a large and active community of developers. This means we can benefit from the continuous improvements, extensive resources, and support available within the React.js ecosystem.

5. Mobile Development: React Native, which is built on React.js, enables efficient development of mobile applications for both iOS and Android platforms. This allows us to leverage code sharing and reusability between our web and mobile applications.

6. Component-Based Architecture: React.js follows a component-based approach, which promotes reusability and maintainability. By breaking down our UI into smaller, reusable components, we can manage complexity more effectively.

Considering these advantages, React.js aligns well with our project goals and requirements. It empowers us to create a user-friendly and high-performing application while benefiting from a strong community and a wealth of resources.

2.4 MongoDB

MongoDB is a popular choice for managing databases in a flexible and efficient manner. We selected MongoDB as our preferred tool for several reasons:

1. Flexible Data Storage: MongoDB stores data in a flexible document format known as BSON, allowing us to store and retrieve complex data structures easily. This flexibility is advantageous for projects with evolving data requirements.

2. Scalability and Performance: MongoDB is designed to scale horizontally, meaning it can handle large amounts of data by distributing it across multiple servers. It also provides efficient indexing and querying capabilities, resulting in improved performance.

3. Adaptability and Agility: MongoDB's schema-less nature enables easy schema changes, making it suitable for agile development environments where requirements may evolve frequently.

4. Data Consistency: MongoDB ensures data consistency through document-level atomicity, guaranteeing that operations on a single document are fully completed or not applied at all.

5. Integration with Amazon Servers: We utilized Amazon servers for hosting our MongoDB, taking advantage of the scalability, reliability, and managed services offered by Amazon Web Services (AWS). This allows us to benefit from AWS's robust infrastructure for database deployment and maintenance.

6. Developer-Friendly: MongoDB's query language, MQL, is easy to learn and resembles JavaScript syntax. It also provides a range of APIs and drivers for various programming languages, simplifying development tasks.

In summary, MongoDB offers flexibility, scalability, performance, and developer-friendliness. By using Amazon servers for MongoDB, we can leverage the benefits of AWS's infrastructure, ensuring reliable and efficient database operations for our project.

3. Expected Achievements

3.1 Outcomes

In this project, we aim to develop a comprehensive delivery management platform that connects suppliers with a network of couriers and helps them choose the best option based on location, availability, and price. Our platform will offer a range of features to make delivery management easy and efficient, including the ability for suppliers to track their orders from pickup to delivery and pick the courier from a table (K couriers) and for couriers to choose from a variety of delivery jobs and get paid for their services.

3.2 Data Security

Ensuring the security of data on our website is of paramount importance, and we have taken extensive measures to achieve this goal. By utilizing a combination of Node.js, MongoDB, and Amazon servers, we have established a robust framework that safeguards sensitive information from potential threats and unauthorized access.

Node.js, as the runtime environment for our website, provides inherent security features that contribute to a secure development process. It incorporates best practices for handling data securely and offers a wide range of security-focused libraries and modules. These resources assist us in implementing encryption, input validation, and other security measures to protect against common vulnerabilities.

MongoDB, our chosen database management system, plays a crucial role in data security. It offers various security mechanisms, such as authentication, role-based access control, and encryption at rest, ensuring that only authorized individuals can access and modify the data. Additionally, MongoDB provides auditing capabilities, allowing us to monitor and track any suspicious activities.

By hosting our MongoDB on Amazon servers, we leverage the advanced security infrastructure provided by Amazon Web Services (AWS). Amazon servers are designed with multiple layers of security, including network isolation, firewalls, and intrusion detection systems. Moreover, AWS offers robust encryption options, both in transit and at rest, providing an extra layer of protection for our data.

Through these combined efforts, we establish a comprehensive security framework that addresses data security from various angles. We strive to maintain the confidentiality, integrity, and availability of our users' data, preventing unauthorized access, data breaches, and other security risks. By continuously monitoring and updating our security practices, we ensure that our website remains a safe and trustworthy platform for our users.

3.3 Unique Features

3.3.1 Finding the nearest courier

Route optimization is a critical aspect of our operations, as we strive to deliver orders in the most efficient and cost-effective manner. To achieve this, we have implemented a route optimization algorithm that leverages the geographic coordinates of the various parties involved: the supplier, the couriers, and the customers.

Our approach focuses on saving the location data of each courier while they are online. This allows us to have real-time information about their current positions. Additionally, we also store the location of the supplier for reference.

When an order is received, our system calculates the distance between the supplier's location and the potential couriers based on their latitude and longitude coordinates. By determining the K nearest couriers in terms of distance, we can identify the most suitable candidates for handling the delivery. This ensures that we optimize the routes by selecting couriers who are geographically closest to the supplier and, consequently, minimize the overall delivery distance.

By employing this method, we aim to streamline our delivery operations, reduce transportation costs, and improve overall efficiency. Our system continually updates and tracks the location data, allowing us to dynamically adapt and optimize routes based on real-time information. This approach enables us to provide timely and efficient delivery services to our valued customers.

3.3.2 GPS Tracking

To ensure efficient monitoring and management of deliveries, we have incorporated GPS tracking into our project. This technology allows us to precisely track the location, longitude, and latitude of each courier, enabling us to gather valuable information about the status and progress of every delivery.

To implement GPS tracking in our system, we will equip each courier with GPS-enabled devices, such as smartphones. These devices will serve as the means to capture and transmit the location data to our centralized platform. By leveraging GPS technology, we can obtain real-time updates on the couriers' movements and accurately track their positions throughout the delivery process.

The integration of GPS tracking provides several advantages for our project. Firstly, it enables us to monitor the couriers' activities, ensuring they are following the designated routes and meeting delivery milestones. This real-time visibility allows us to identify any deviations or delays promptly, enabling proactive interventions and ensuring timely deliveries.

Furthermore, the GPS data collected from the couriers' devices can be seamlessly integrated into our platform. This allows us to generate comprehensive reports and analytics, providing insights into delivery performance, optimizing routes, and identifying areas for improvement. The availability of accurate location data enhances operational efficiency and enables data-driven decision-making.

By leveraging GPS tracking technology and equipping our couriers with GPS-enabled devices, we establish a robust system that facilitates real-time monitoring, efficient route management, and data-driven optimization. This approach empowers us to provide reliable and transparent delivery services to our customers while maximizing operational effectiveness.

3.3.3 Alerts and Notifications

This feature is a way to keep the supplier and couriers informed about the status and progress of each delivery. That could be used to communicate a variety of information.

3.3.3.1 Confirmation of delivery requests

When a new order request and the delivery of it received, a notification will send to the supplier and the courier to confirm the status of the order.

3.3.3.2 Updates on the status of deliveries

Notifications will send to the supplier and courier to provide updates on the status of the delivery, such as the order confirmed by the courier, order is pending and order has been delivered.

3.3.3.3 Changes to delivery schedules

If there are any changes to the delivery schedule such as a delay or a change in the delivery location, notifications will be sent to the supplier and courier to inform them of the changes and provide the suitable updates.

3.3.3.4 Alerts for missed or late deliveries

If a delivery is missed or running behind schedule, notifications (and an alerts) will be sent to the users to be informed about the situation and they must provide updates.

3.3.3.5 Alerts for emergencies or unexpected events

In the event of an emergency or unexpected event (such as an accident or the courier cannot take the order) alerts will be sent to the supplier to inform him of the situation.

3.4 Criteria for Success

The success criteria for our project will include the number of active users, the satisfaction of both suppliers and couriers with the app, and the efficiency of the delivery process. We will also measure the time and cost savings for both suppliers and couriers compared to traditional delivery management methods. In addition, we will also need to address non-trivial requirements such as ensuring the security and privacy of each user's data and providing a user-friendly interface for both suppliers and couriers.

4. Engineering Process

4.1 Research – Delivery Management

Regarding broadening our understanding of Delivery Management, we focused on answering the following questions:

How should we pick the nearest courier to the supplier to take the order?

What are the most important factors that influence a supplier's decision to use our system?

What are the most important factors that influence a courier's decision to work with our system?

How can our system be designed to ensure the safety and security of packages and couriers during the delivery process?

We consulted a range of materials, from academic papers and articles to videos, to address these issues and increase our expertise. After looking over the resources, we got together to talk about

our results and identify the key aspects we should pay attention to when developing the application.

Some of the conclusions we reached while reading about Delivery Management with regards to developing software, there exists a need to add a questionnaire after making an evaluation in the simulation that can be:

Overall, how satisfied are you with the delivery management app?

How easy or difficult was it to use the app?

What features or functionality did you find most useful in the app?

What features or functionality would you like to see added to the app in the future?

Was the app reliable and stable during use?

Were you able to easily track your deliveries using the app?

How would you rate the speed and efficiency of the app?

If we find correlation between the questionnaire and our final analysis, this will validate our delivery management application.

4.1.1 Constraints and Challenges – Delivery Management

Picking a suitable courier for an order can be a challenge in our system because there may be multiple factors to consider when determining which courier is the most suitable. Some of these factors may include:

4.1.1.1 Distance

The distance between the supplier and the courier can impact the efficiency of the delivery and the time required to complete it.

4.1.1.2 Availability

The courier's availability (such as whether they are currently on delivery or have capacity for additional orders) can impact their suitability for an order.

To address this challenge, our system considers all these factors when determining the most suitable courier for an order. This obligates us to have access to detailed information about the

couriers (such as their location, equipment, and availability) and the ability to weigh these factors against each other to decide.

4.2 Methodology and Development Process

In order to develop a system that meets our desired goals, we have created a structured work plan that outlines all the necessary steps, starting with the learning process and proceeding through the development process. Through our research, we determined that the Agile approach would be the most effective method for our project. This iterative approach emphasizes the importance of working in stages with ongoing planning and continuous learning.

To begin, it was necessary for us to thoroughly research existing solutions and technologies in the market in order to gain a comprehensive understanding of their strengths and weaknesses. Our goal was to identify ways in which we could leverage the advantages of these solutions while also seeking opportunities to improve upon their shortcomings to create a truly unique system. This required a thorough analysis of the current market landscape to identify areas where we could make a meaningful contribution.

Our work process is divided into three main parts:

4.2.1 Planning and preparation

This included meetings in which we created a detailed requirements document for the system, developed diagrams to visualize the desired outcome, and identified the tools and technologies we would need to use. Specifically, we decided to use React, JS, HTML and CSS as a front-end development, NodeJS for the back-end development, DB Microsoft SQL server for managing the data of our interactive website.

4.2.2 Implementation

This includes using the Twilio service to connect to an SMS API and send confirmation messages to the suppliers, as well as defining and completing the initial tasks for the project such as creating the main screens for the users and building the basic infrastructure.

4.2.3 Iterative development

As we learn and make progress, we will continue to develop and build upon the system in stages until we reach the final product. This will involve ongoing planning and adjustments based on lessons learned.

5. Product

5.1 MERN-STACK

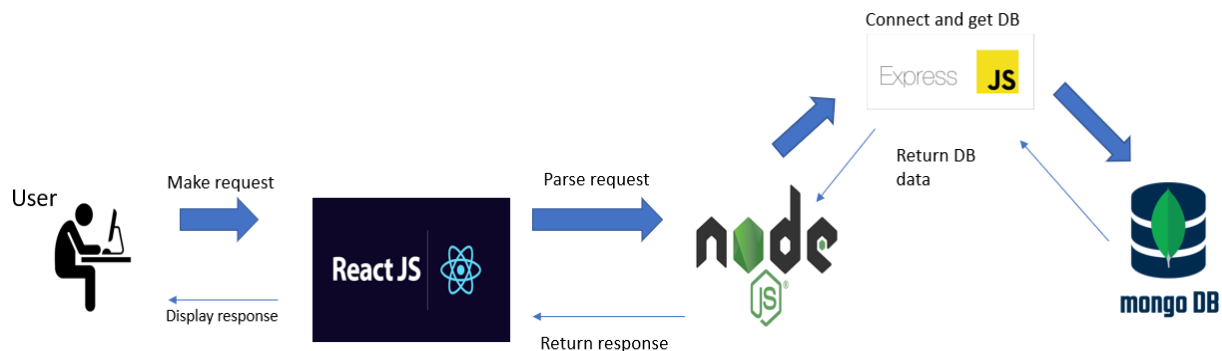


Fig1: MERN-STACK diagram

The MERN stack is a popular JavaScript-based framework used for web application development. It is an alternative to the MEAN stack, with React replacing Angular as the client-side JavaScript framework. The acronym MERN represents the four key technologies that make up the stack: MongoDB, Express.js, React, and Node.js. This stack is widely used and favored by developers for its flexibility and efficiency in building web applications.

In the MERN stack, MongoDB serves as the document database. It is an open-source NoSQL database designed for cloud applications, offering scalability and flexibility in data storage. Express.js, running on a Node.js server, acts as the web framework. It provides features for handling URL routing, HTTP requests, and responses. React, the core component of the stack, is a powerful JavaScript library used for building interactive user interfaces. It enables developers to create dynamic and responsive web experiences through its component-based architecture and efficient rendering.

Node.js, the premier JavaScript web server, remains a key component in the MERN stack. It provides the runtime environment for executing server-side JavaScript code. With Node.js, developers can handle server-side rendering, API integration, and event-driven programming.

The MERN stack allows developers to build web applications entirely in JavaScript, simplifying the handling of JSON data and enabling smooth communication between different layers of the stack. React, as the front-end framework, empowers developers to create modern and interactive user interfaces. Express.js and Node.js handle server-side logic and facilitate API integration. MongoDB, on the other hand, offers a flexible and scalable database solution.

Overall, the MERN stack provides an effective and comprehensive set of technologies for web applications development. By leveraging the power of JavaScript throughout the entire stack, developers can build robust and efficient applications.

5.2 Requirements

Req. number	Requirements description	Req.type (FR or NFR)
1	The system allows to identify users by username and password	FR
2	The username is unique for each user	NFR
3	The system allows identifying the user type immediately after logging in	FR
4	The user types are Courier, Supplier, and Manager	NFR
5	The system allows each supplier to add a new order	FR
6	Each order contains the required date and location	NFR
7	The system allows each supplier to choose the courier of his order	NFR
8	The system allows each supplier to adjust his orders	FR
9	The allowed changes to make are the date and time	NFR
10	The order can be edited/adjusted only if the order's status is "pending_confirmation"	NFR
11	The system allows each supplier to remove orders	FR
12	The supplier can remove his order only if the order's status is "pending_confirmation"	NFR
13	The system allows each supplier to pay	FR
14	The payments are monthly requested	NFR
15	The system allows the manager to add new couriers	FR
16	Each courier that will be added has to contain the personal information	NFR
17	The system allows the manager to change the courier's status	FR
18	The courier status can be: {Okay, warning, fired}	NFR
19	The system allows the manager to remove couriers	FR
20	The system allows the manager to follow the supplier's payments	FR

21	The system allows the manager to change the status of the supplier's payment	FR
22	The status of each payment can be: { Pending, Done }	NFR
23	The system allows couriers to accept/reject orders	FR
24	The system allows couriers to update the order's status to "completed"	FR

5.3 UML Diagrams

5.3.1 Use Case Diagram

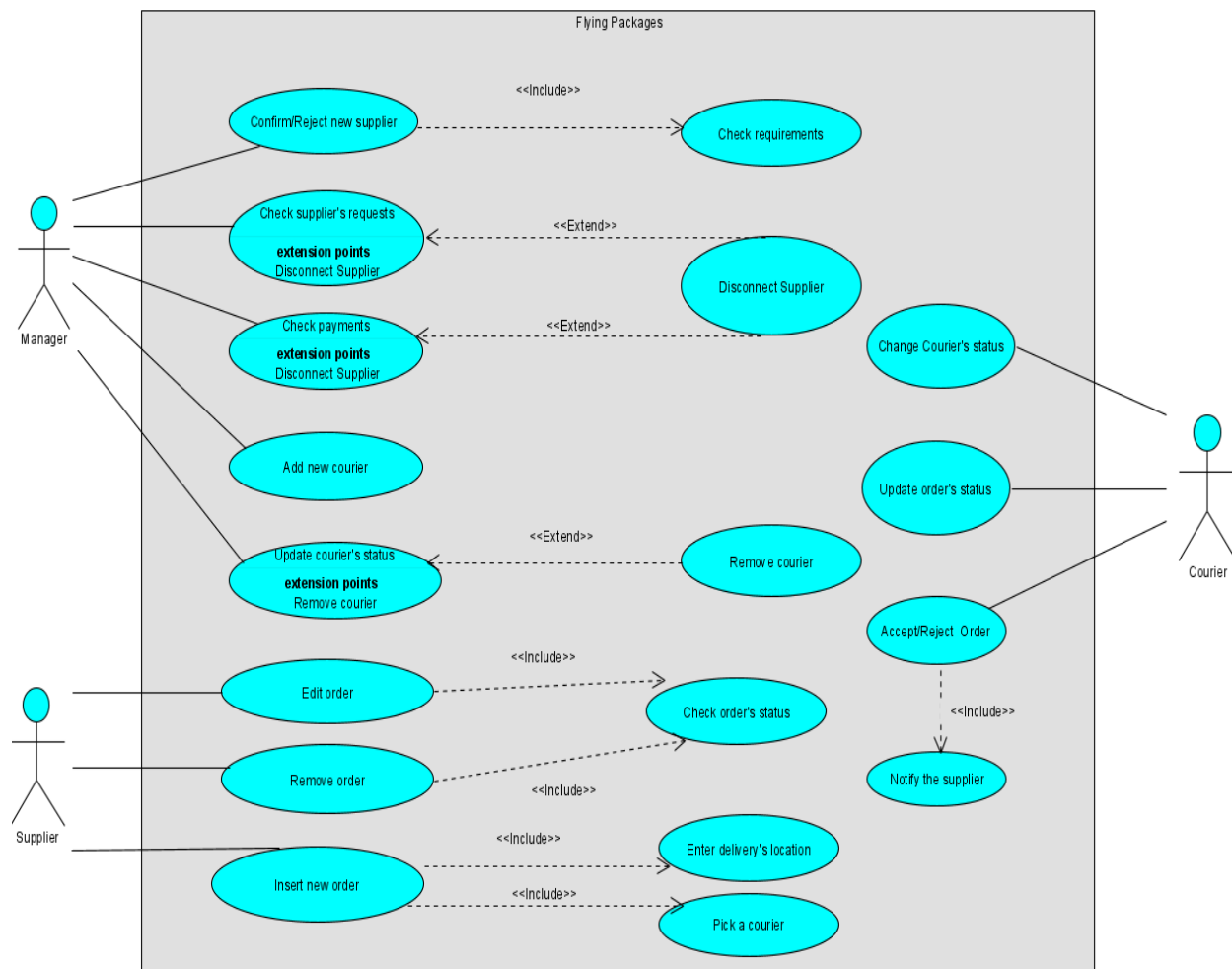


Fig2: Use Case Diagram

5.3.2 Class Diagram

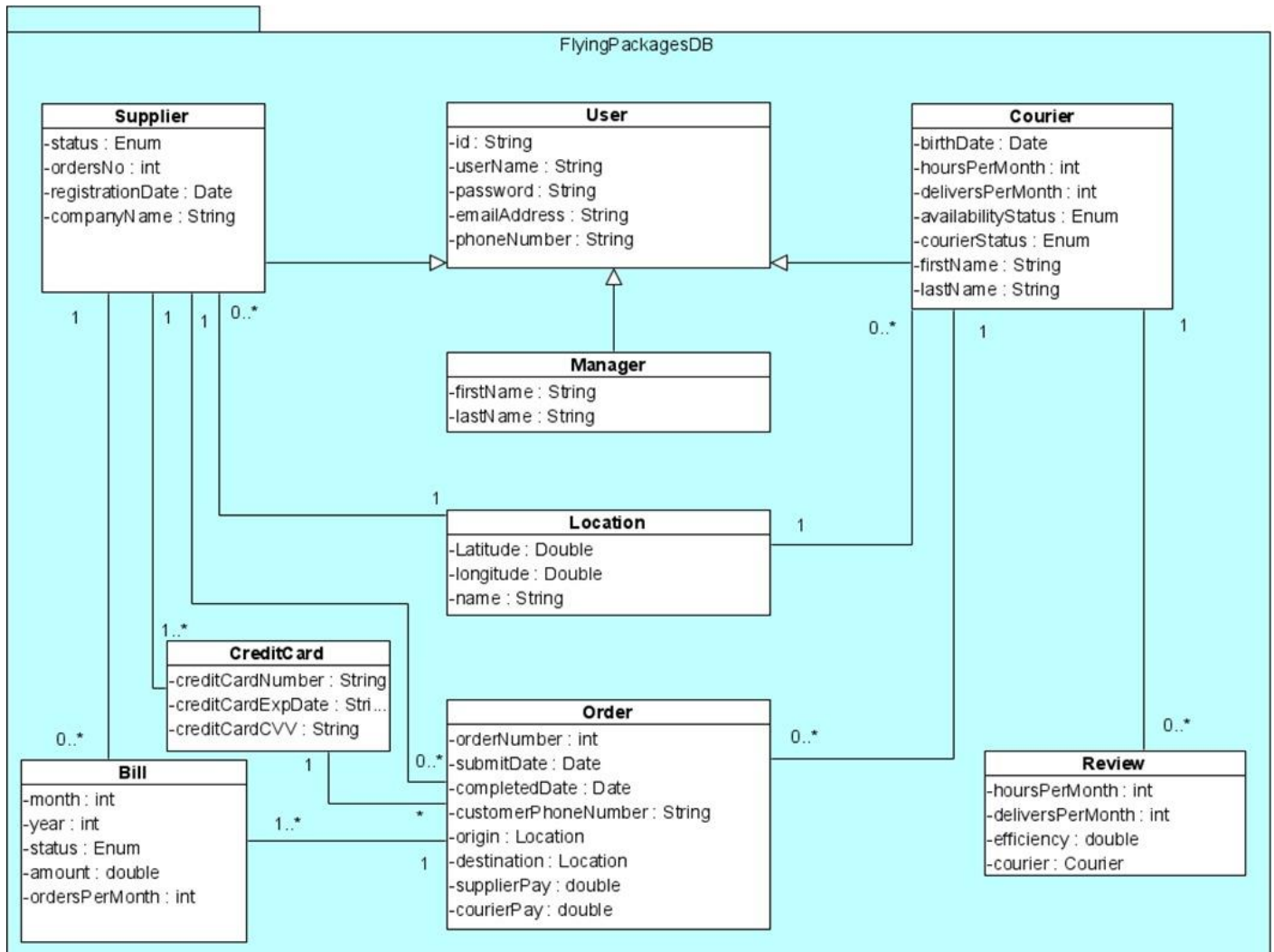


Fig3: Class Diagram - Entities

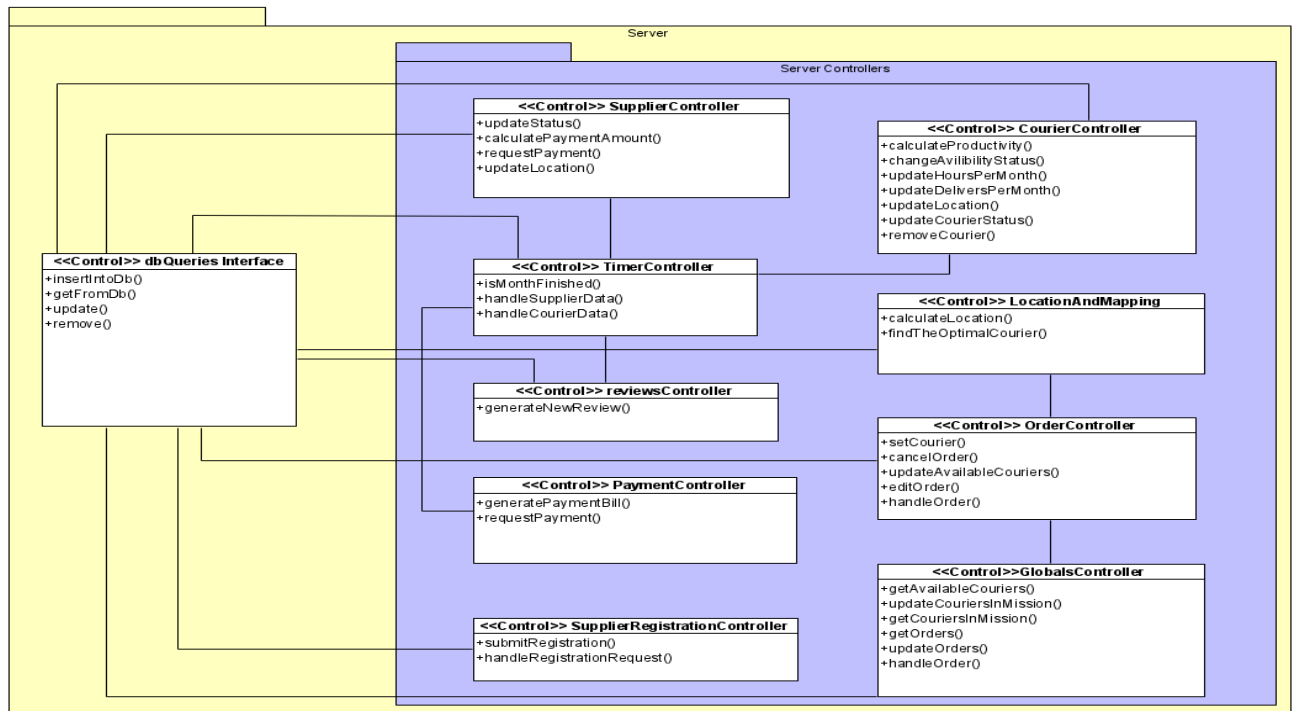


Fig4: Class Diagram - Server

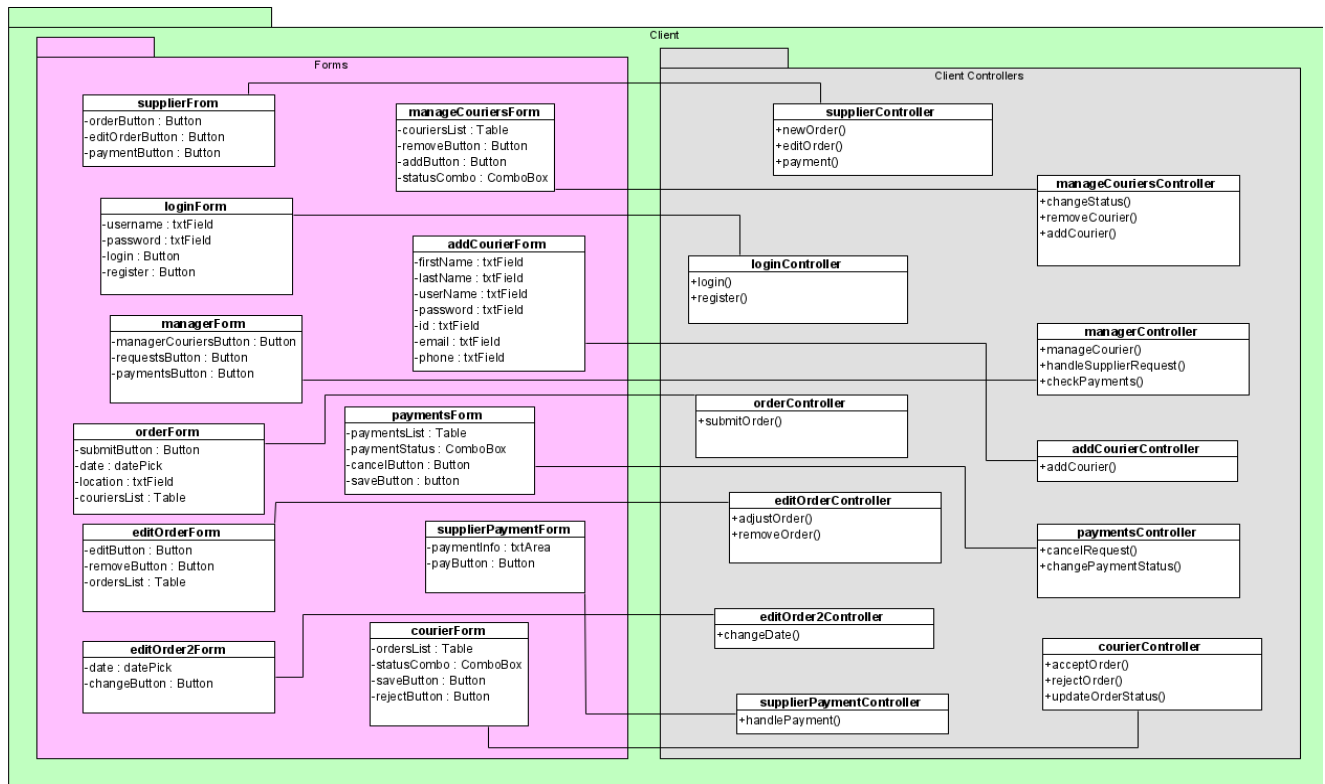


Fig5: Class Diagram - Client

5.3.3 Activity Diagram

5.3.3.1 Order

- 1- The supplier chooses the new order option.
- 2- The supplier picks a date and time for his order.
- 3- The System checks if the date is valid.
- 3.1- if the time is not valid, we return to step 2.
- 3.2- if the time is valid, the system gets a list of available couriers.
- 4- The supplier picks the courier that he wants to carry his delivery.
- 5- The courier gets a notification.
- 6- The courier will choose to accept and carry the delivery or not.
- 7- if the courier chooses to not do it, we return to step 4.
- 8- if the courier accepts it, he delivers the order.
- 9- The courier changes the status of the order after he finishes.
- 10- The system inserts the order's details into the Database.

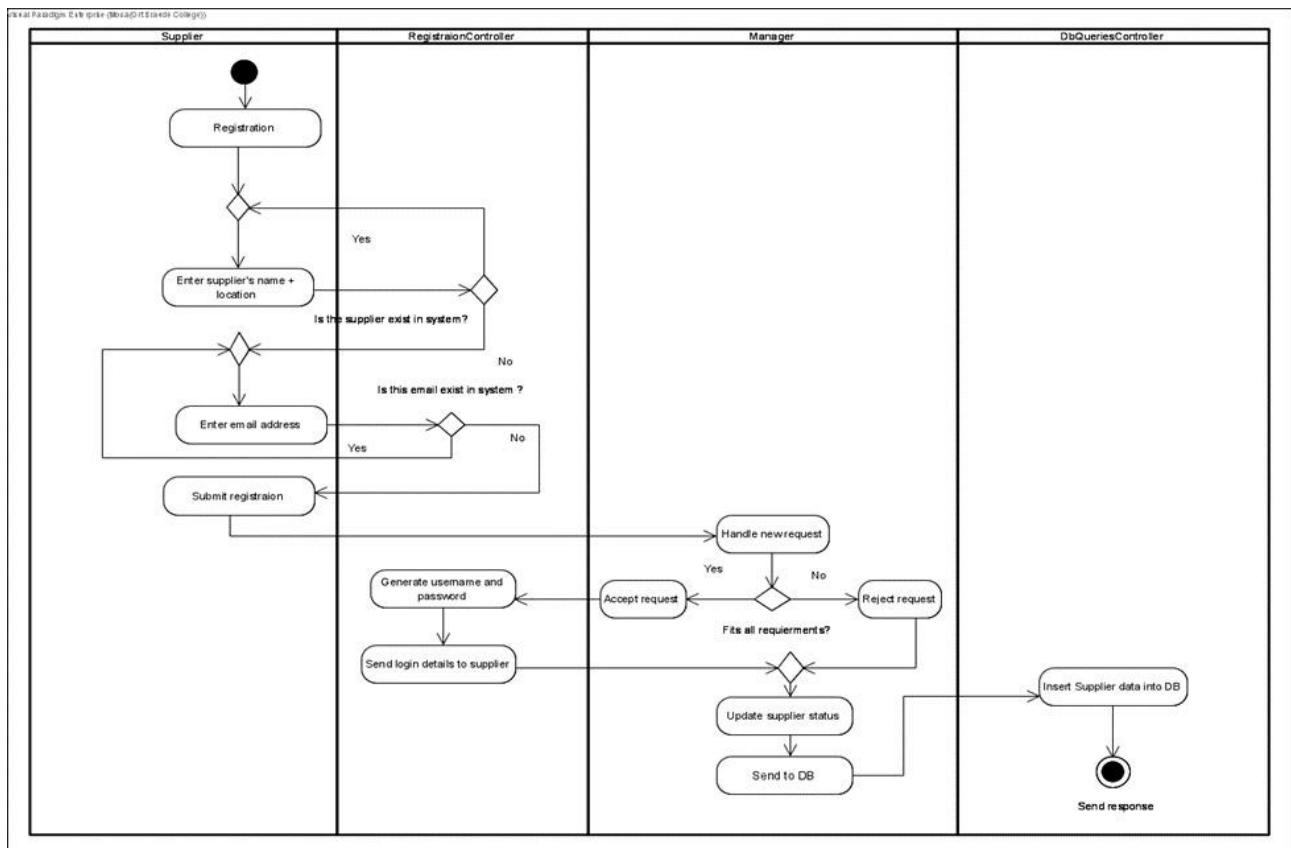


Fig6: Activity Diagram – Order

5.3.3.2 Register new supplier

- 1- The supplier chooses to register for our system.
- 2- The supplier enters the name of the company and the location.
- 3- The system checks if the name and location exist.
- 3.1- If yes, we return to step 2.
- 3.2- If no, the supplier enters the email address.
- 4- The system checks if the email is already taken.
- 4.1- If yes, we return to step 3.2.
- 4.2- If no, the supplier submits his registration.
- 5- The manager decides whether to accept the request or not.
- 6- If he accepts, the system will generate a username and password.
- 6.1- The system sends the login details to the supplier.
- 7- The manager updates the supplier's status.
- 7.1- The manager sends a message of confirmation/rejection.
- 8- The system inserts the supplier's details into the DB.

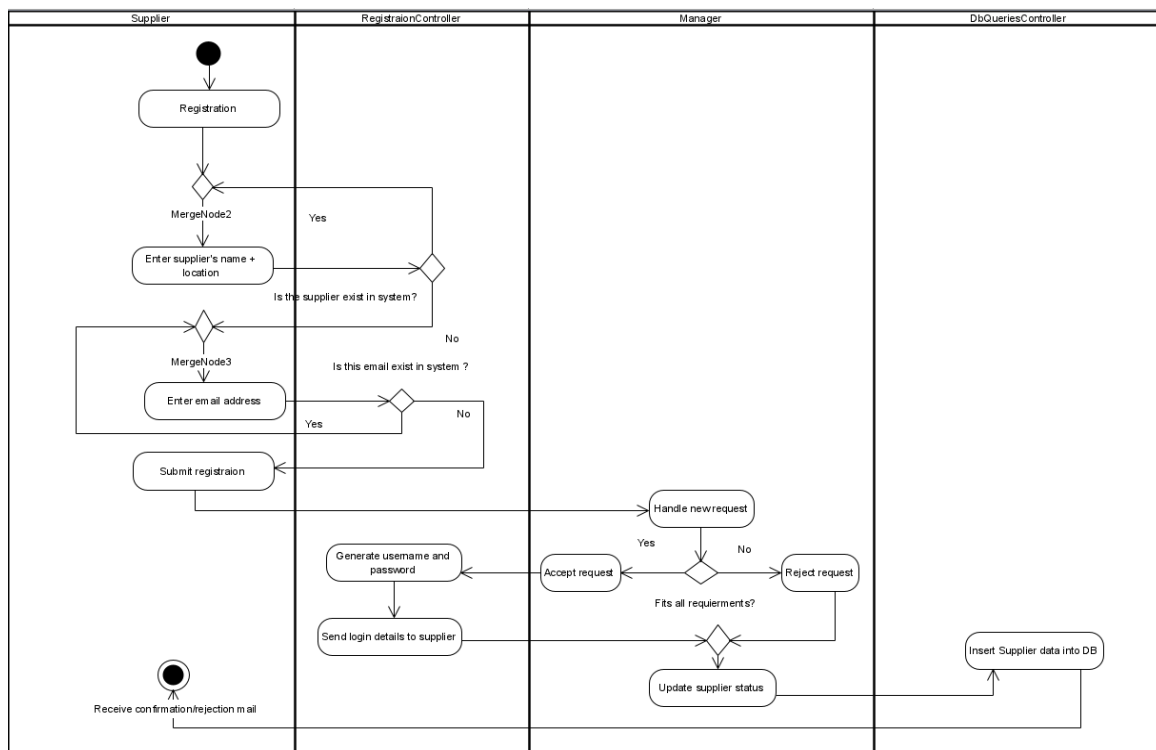


Fig7: Activity Diagram – Register new supplier.

6. User Documentation

6.1 User Guide

HomePage



Fig8: Homepage.

This is the first page that every user sees when he enters our website, the user can click login or sign up, in addition to contact us.

Sign up page.

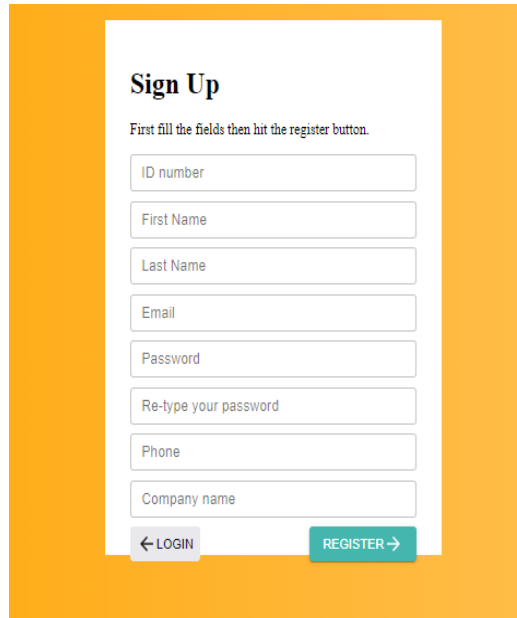
A screenshot of a 'Sign Up' form. The form is titled 'Sign Up' in bold. Below the title is a instruction: 'First fill the fields then hit the register button.' The form contains several input fields: 'ID number', 'First Name', 'Last Name', 'Email', 'Password', 'Re-type your password', 'Phone', and 'Company name'. At the bottom of the form, there are two buttons: a grey button with a left arrow and the text 'LOGIN', and a green button with the text 'REGISTER' and a right arrow. The entire form is set against a white background with a thin orange border.

Fig9: Sign up page.

We will reach this page after clicking on signup from the previous page, The supplier will enter his company's name and his location updated when he login for the first time.

Login page.

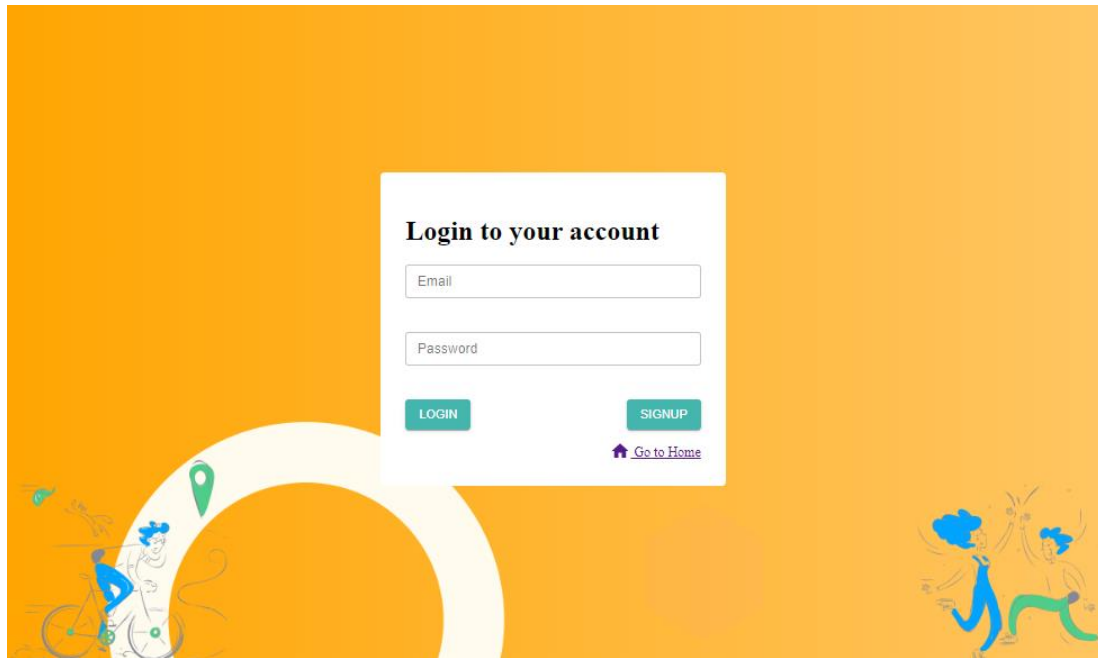


Fig10: Login page.

Each user can join with his private email and password, and the system will recognize the type of the user (courier/manager/supplier) and open the relevant page. For the suppliers who want to join our system, they can sign up by the linked text below the sign-up button.

Supplier Form

LOGOUT

Welcome, Supplier Mosa!

Last Orders

+ NEW ORDER EDIT ORDER MONTHLY BILL PAYMENTS

Order Num	Courier	Destination	Status
77915	Wesam	Nazareth 5070	PENDING
10286	Wesam	Nazareth 6031	DELIVERED
48877	Courier1	Tel Aviv Rabin Square	DELIVERED
33631	Courier1	big nazareth	DELIVERED
79802	Courier1	Nazareth 6058	DELIVERED

SHOW MORE

Fig11: Supplier page.

This is the homepage for each supplier, after clicking login from Figure 10 with the supplier's email and password. On this page the supplier can see his last orders from the table above, he can choose to insert new orders, he can choose to edit his orders (change the time if is it possible) and can see his monthly bills. The supplier can choose to click on the View more linked text and watch more orders than those who appear in the table.

New Order

New Order

*The order's cost is: 40 ILS

[← BACK](#) [SUBMIT](#)

Destination

Submit Date

Submit Hour

Phone Number

Courier ▼

Pick Courier

Name	Distance	Phone Number
Wesam	1.43 Km	0584652145
Jakob	1.43 Km	0548568752
Courier1	4.15 Km	0587451236
Jack	77.96 Km	0546874589

Fig12: New order page.

We will reach this page after clicking on the new order button from the previous one, the supplier will fill in the details including the destination of the delivery and the customer's phone number, and then he will click on continue to complete the process. The supplier can see the available couriers that can carry his delivery on a table, with the distance according to the courier's location.

We don't want to choose a courier for the supplier, so we give him advice on which courier he should choose by marking the optimal couriers according to the distance. The supplier will see the information on the table and will pick the courier he wants by the drop list, then press on submit.

Edit Order

Welcome, Supplier Mosa !

Last Orders

← BACK

Order Num	Hour	Date	Status	Edit
77915	<div>Time</div> <div>10:50</div>	<div>Date</div> <div>08/06/2023</div>	PENDING	<div>SAVE</div>
9712	<div>Time</div> <div>15:30</div>	<div>Date</div> <div>10/06/2023</div>	PENDING	<div>SAVE</div>

Fig13: Edit orders

This page will appear after clicking on the edit order button from Figure 11. The supplier will see his orders that are still in Pending status. He can change the time from the table.

Monthly payments

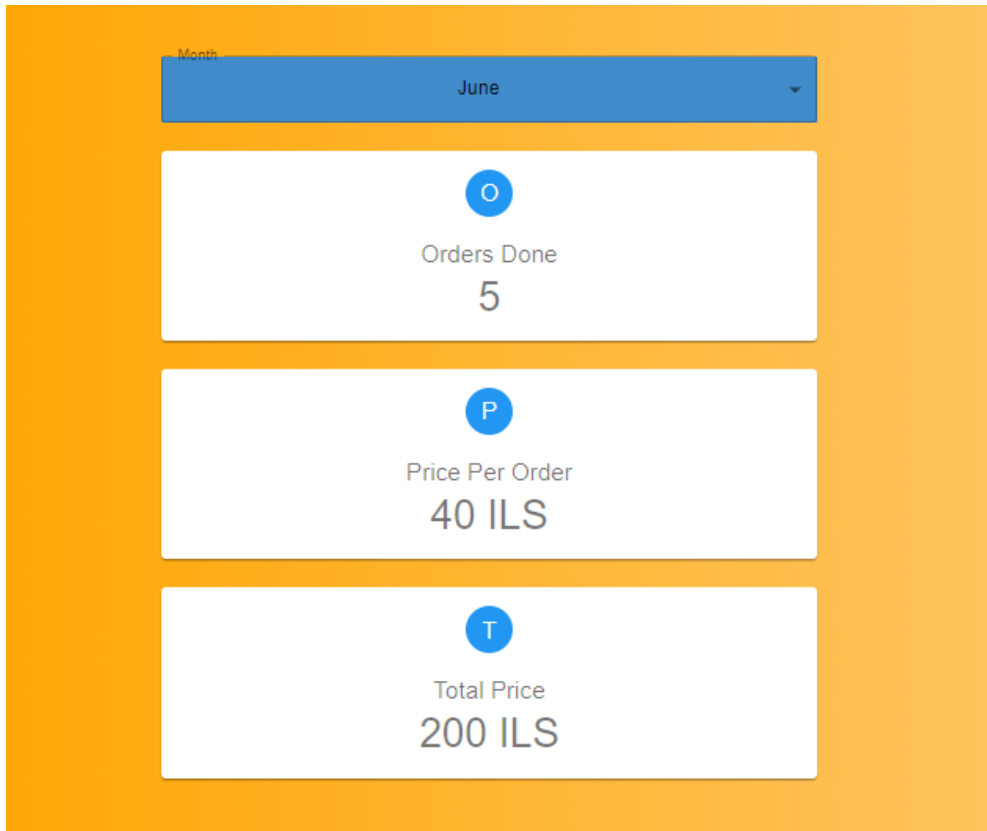


Fig13: Supplier's payments page

we will reach this page after clicking on the Monthly bill payment from Figure 11. The supplier will choose the month that he wants to see its bill, and the bill appears including the details from the specific month.

Courier page

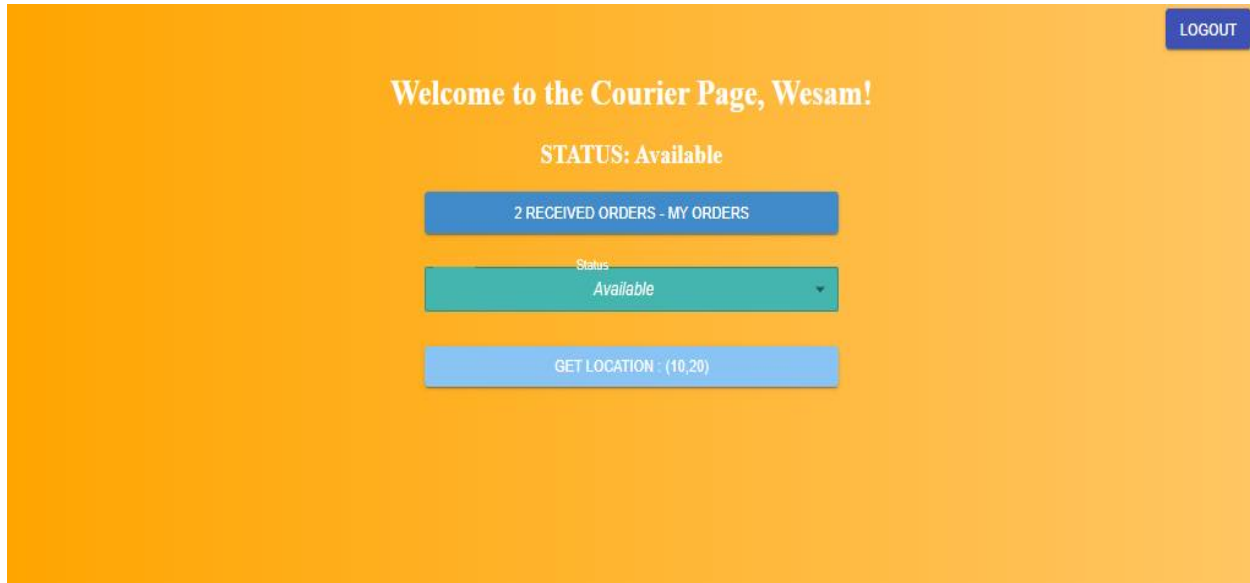


Fig13: Courier homepage

This is the homepage for each courier, after clicking login from Figure 10 with the courier's email and password. The courier can see his status below the welcome message, and he can change it from the drop table (if he wants to take a rest for example). The courier can see a notification behind the received orders label. He will see those orders by clicking on the My orders button, if he is already carrying a delivery, he can't receive orders until he finishes. And he will click on my orders to change the status of his delivery and notify the system that he completed it, in addition he can click on update location update his current location.

Received Orders



Order Num	Origin	Destination	Time	Take	Reject
77915	Nazareth	Nazareth 5070	08/06/2023 10:50	<input checked="" type="checkbox"/> TAKE	<input checked="" type="checkbox"/> REJECT
9712	Nazareth	Karmiel Ort Braude	10/06/2023 15:30	<input checked="" type="checkbox"/> TAKE	<input checked="" type="checkbox"/> REJECT

Fig14: My orders page

In this case, the courier clicked on my orders button from the previous page. He can see this page which includes the orders that pending for his delivery and he can take or reject the order.

Taking Order

Click Here'. In the top right corner, there is a green button with a left arrow and the text 'BACK'. The main content is a table with 6 columns: 'Order Num', 'Origin', 'Destination', 'Time', 'Take', and 'Reject'. There is only one row of data with '77915' as the order number, 'Nazareth' as the origin, 'Nazareth 5070' as the destination, and '08/06/2023 10:50' as the time. The 'Take' column has a blue button with a checkmark and the text 'TAKE'. The 'Reject' column has a red button with an 'X' and the text 'REJECT'." data-bbox="113 512 837 715"/>

Order Num	Origin	Destination	Time	Take	Reject
77915	Nazareth	Nazareth 5070	08/06/2023 10:50	<input checked="" type="checkbox"/> TAKE	<input checked="" type="checkbox"/> REJECT

Fig15: Taking order

while his last order's delivery has not been completed yet. He can see this page which includes the order number and a button that he can use to notify the system that the order is completed, and he can't take or reject any orders while the order in delivery.

Supplier page while delivering the order.



Order Num	Courier	Destination	Status
9712	Wesam	Karmiel Ort Braude	IN DELIVERY
77915	Wesam	Nazareth 5070	PENDING
10286	Wesam	Nazareth 6031	DELIVERED
48877	Courier1	Tel Aviv Rabin Square	DELIVERED
33631	Courier1	big nazareth	DELIVERED

Fig16: Supplier page after taking the order.

After taking the order by the courier the supplier can see that the order's status is "IN DELIVERY"

Supplier page after taking the order



Order Num	Courier	Destination	Status
9712	Wesam	Karmiel Ort Braude	DELIVERED
77915	Wesam	Nazareth 5070	PENDING
10286	Wesam	Nazareth 6031	DELIVERED
48877	Courier1	Tel Aviv Rabin Square	DELIVERED
33631	Courier1	big nazareth	DELIVERED

Fig17: Supplier page after delivering the order.

After delivering the order by the courier the supplier can see that the order's status is "DELIVERED"

Manager page



The screenshot shows the Manager homepage with a yellow background. At the top right is a 'LOGOUT' button. The main heading is 'Welcome, Manager Wesam!'. Below it is the section 'Last Orders'. There are three buttons: 'MANAGE COURIERS' (with a person icon), 'REGISTRATION REQUESTS' (with a pencil icon), and 'PAYMENTS REPORTS' (with a dollar sign icon). Below these buttons is a table of last orders. The table has four columns: 'Order Num', 'Supplier', 'Courier', and 'Status'. The data rows are: (9712, Mosa, Wesam, DELIVERED), (77915, Mosa, Wesam, PENDING), (46269, Supplier1, Jakob, DELIVERED), and (77511, Supplier1, Jack, DELIVERED). At the bottom left is a 'SHOW MORE' button.

Order Num	Supplier	Courier	Status
9712	Mosa	Wesam	DELIVERED
77915	Mosa	Wesam	PENDING
46269	Supplier1	Jakob	DELIVERED
77511	Supplier1	Jack	DELIVERED

Fig18: Manager homepage.

This is the homepage for each manager, after clicking login from Figure 10 with the manager's email and password. On this page, the manager can see the last orders from the table above, and he can click on view more to see more orders. The manager can click on each button above the table: manager couriers/registration requests and payments reports.

Manage couriers



The screenshot shows the 'Couriers Management' page with a yellow background. At the top right are two buttons: '+ ADD' and 'CHANGES' (with a save icon). Below these buttons is a table of couriers. The table has four columns: 'ID', 'Full Name', 'Status', and 'Efficiency'. The data rows are: (211756820, Wesam, OK, 6.48), (315487458, Courier1, OK, 10.00), (211588965, Jack, OK, 10.00), and (210458789, Jakob, OK, 8.57). Each 'Status' cell contains a dropdown menu with 'OK' selected.

ID	Full Name	Status	Efficiency
211756820	Wesam	OK	6.48
315487458	Courier1	OK	10.00
211588965	Jack	OK	10.00
210458789	Jakob	OK	8.57

Fig19: Manage couriers.

We will reach this page after clicking on the manage couriers' button from the previous one. The manager will see the courier's details in the table, and he can change the courier's status from the status column and then click on save changes, he can see the efficiency from the last month and decide to change the status accordingly. The manager can add a new courier by clicking on the button “add”.

Add new courier by the manager.

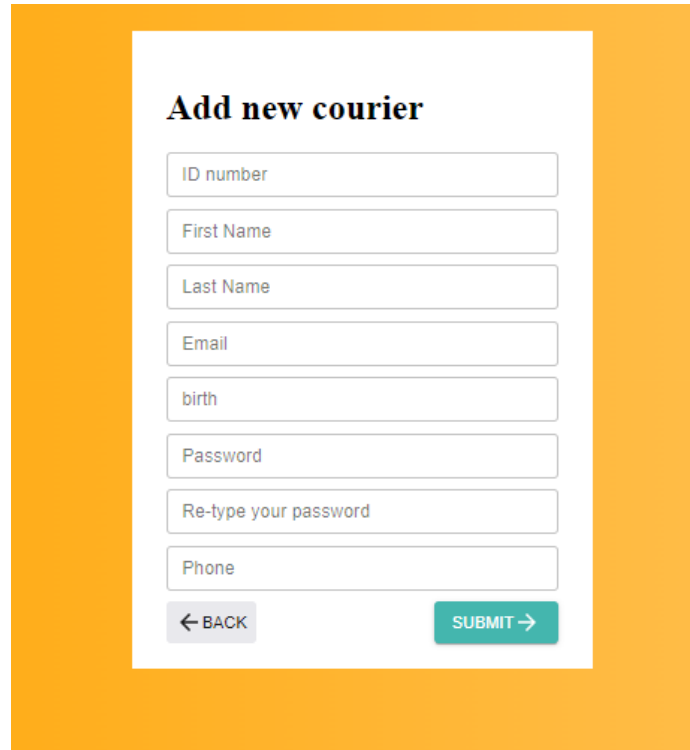
The image shows a web form titled "Add new courier" centered on a white background, which is itself set against a larger orange rectangular backdrop. The form contains several input fields stacked vertically: "ID number", "First Name", "Last Name", "Email", "birth", "Password", "Re-type your password", and "Phone". At the bottom of the form, there are two buttons: a grey button with a left arrow and the text "BACK", and a green button with the text "SUBMIT" and a right arrow.

Fig20: Add new courier.

This page is reachable by clicking on add new courier button from the previous page, the manager fills in the details and clicks on submit to add the courier to the system.

Suppliers registration requests.

Supplier Registration Requests				
Username	ID	Email	Company	Action
Edy	214574458	edy@gmail.com	More Drinks	ACCEPT REJECT

Fig21: Registration requests page.

The manager will reach this page after clicking on registration requests from Figure 18. The manager will see the table of the suppliers who requested to join our system. He can click on the accept button to accept each one and he can reject.

Monthly Payments Reports.

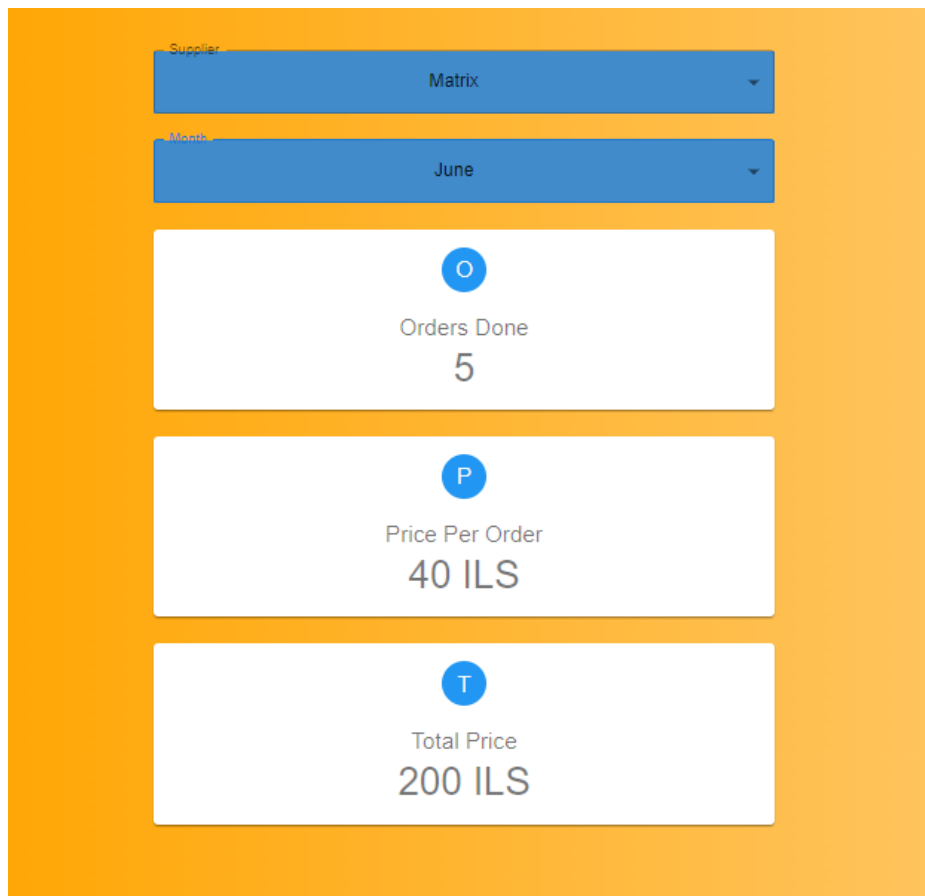


Fig22: Payments Reports.

The manager will reach this page after clicking on the payments reports button from Figure 18. On this page, he can pick a month and a supplier and see the bills from the specific month.

6.2 Maintenance Guide

If you want to start the deployment, you should initiate the process by running ‘npm start’ for both the Backend and the Frontend. This will start the respective servers and make your project accessible. Additionally, it is important to ensure that the Backend and Frontend are configured to start on the appropriate ports. For example, you might specify "localhost:3000" as the port for your project. Adjust the port settings as needed to align with your specific deployment environment.

The Maintenance Guide serves as a valuable resource for managing ongoing edits and enhancements to the project.

It focuses on the two main components: the Backend and the Frontend. In the Backend, attention is given to two pivotal files: the "Models" file, which establishes the database object structures, and the "Router" file, responsible for handling requests through Express.js routers.

To facilitate updates and improvements in the Backend, it is crucial to adhere to guidelines for modifying or introducing new models and routes. Meanwhile, within the "src" directory of the Frontend, all pages and components are housed. This location allows for seamless modifications and additions to bolster the project's functionality. Consistency in styling and UI elements is emphasized throughout the Frontend. By diligently adhering to the guidelines outlined in this Maintenance Guide, the project can be effectively maintained and continually enhanced.

The Maintenance Guide also highlights the importance of updating specific files when making changes in the Backend or Frontend. In the Backend, any modifications should prompt an update to the "backend/app.js" file to ensure proper integration of the changes. Similarly, when introducing edits in the Frontend, it is necessary to update the "frontend/App.js" file to reflect the updated functionality. These updates facilitate seamless communication between the Backend and Frontend components of the project, ensuring smooth operation and optimal performance.

6.3 DataBase Guide

MongoDB is the database management system used by the project to store and arrange its data. The database, dubbed "eshop-database," consists of several tables or collections containing various sorts of information. The tables include "bills," which keeps billing-related data, "couriers" for courier information, "creditcards" for credit card details, "flyorders" for tracking flight orders, "flyusers" for user profiles, "locations" for location data, and "suppliers" for supplier-related information. Each table is intended to gather and arrange data particular to its domain, allowing for the efficient storage and retrieval of relevant information. This well-structured layout of data tables guarantees that the project's data resources are managed and utilized optimally.

6.4 System Operating Environment

The system's operating environment for this web project is centered around the Google Chrome browser, where the application will be deployed and accessed. To run the project and begin exploring its functionality, all you need is a computer or device with Google Chrome installed.

On the software side, the project utilizes web technologies such as HTML, CSS, and JavaScript to create interactive user interfaces. These languages are supported by modern web browsers, including Google Chrome, ensuring compatibility and seamless functionality.

As for hardware requirements, the project can run on standard desktops, laptops, or mobile devices capable of running Google Chrome. No specific hardware specifications are necessary, as the project is designed to be lightweight and adaptable to various computing environments.

To start using the project, simply open the Google Chrome browser and enter the project's URL or open the project's HTML file directly. This will initiate the application and allow you to explore its features and functionality.

7. Verification and Evaluation

7.1 Unit Tests

No#	Test Subject	Expected Result	Result
Login form			
1	Enter an empty username or password	Error displays a message: "Please fill in the fields".	Passed
2	Enter the wrong username or password	Error displays a message: "Invalid details, try again" We don't want to give information about which field is invalid.	Passed
3	Log in with a 'frozen' status username and password	Display message: "User is Frozen".	Passed
4	Enter a valid username and password	Display message: "Login succeed".	Passed
Supplier's registration form			
5	Enter an existing name and location	Error displays a message: "This supplier is already existing"	Passed
6	Enter empty details (one or more)	Display message: "Please fill in the fields".	Passed
7	Enter valid details	Display a message: "Registration succeed, please wait for approvment".	Passed
Approve/Reject suppliers' form			
8	Accept the supplier's request for registration.	Display a message: "Supplier confirmed"	Passed
9	Reject supplier's request for registration.	Display a message: "Supplier's request rejected".	Passed
Add new courier			
10	Enter empty details (one or more)	Error displays a message: "Please fill in the fields".	Passed
11	Enter an existing id	Display a message: "This courier is already in the system".	Passed
12	Enter an existing phone number	Display a message: "This phone number is already in use".	Passed
13	Enter an existing email address	Display a message: "This email address is already in use".	Passed
14	Enter an existing username	Display a message: "This username is already in use".	Passed
15	Enter valid details	Display a message: "Courier registration succeeds, he can now log in with the username and password".	Passed

7.2 Functional Tests

No#	Test Subject	Expected Result	Result
Login form			
1	Login with a valid supplier username and password	Move to the supplier's form.	Passed
2	Login with a valid courier username and password	Move to the courier's form.	Passed
3	Login with a valid manager username and password	Move to the manager's form.	Passed
Supplier's registration form			
4	Register with a valid detail	The request will appear on the list on the manager form.	Passed
Approve/Reject suppliers' form			
5	Accept supplier's registration	The supplier's status will change from 'pending confirmation' to 'approved'	Passed
6	Reject supplier's registration	The supplier's status will change from 'pending confirmation' to 'Frozen'	Passed
Add new courier			
7	Enter a valid detail	The courier can log in with the given username and password	Passed
Add new order			
8	Submit a new order from the supplier's form.	The order will appear in the order list in the manager's form.	Passed
9	The optimal courier for the mission has been found.	The courier gets an SMS that includes the delivery details.	Passed
10	The courier confirms that he will handle the order	The order's status will change from 'pending' to 'in progress'.	Passed
11	The courier updates that he finished his mission	The order's status will change from 'in progress' to 'completed'	Passed
Monthly updates			
12	A courier has less efficiency than the requirements for the first time	The courier's status will be changed from 'Ok' to 'Warning'.	Passed
13	A courier has less efficiency than the requirements for the second time	The courier's status will be changed from 'Warning' to 'Fired' and his account will be frozen.	Passed
14	A supplier has '30' orders for the last month.	The supplier will receive a bill that includes the amount of payment he has to pay "30 * X".	Passed

8. Results and conclusions

We have achieved our objective with resounding success by optimizing our delivery operations through cutting-edge technologies and innovative approaches. The notable outcomes include a remarkable reduction in delivery times, enhanced customer satisfaction, and heightened operational efficiency.

Our system utilizes GPS tracking technology, providing real-time visibility into courier locations for accurate monitoring and tracking of deliveries. This transparency allows us to provide timely updates to customers and make proactive decisions in response to unforeseen circumstances.

Advanced assignment algorithms match deliveries with suitable couriers based on factors like proximity and expertise, minimizing delays and maximizing resource utilization.

The system generates invaluable insights and analytics by analyzing data related to delivery routes, courier performance, and customer feedback. This helps identify areas for improvement and make informed decisions to refine services.

Hosting our MongoDB database on Amazon servers ensures secure and reliable data storage while enabling seamless scalability to accommodate our growing user base.

Overall, our project's exceptional results demonstrate significant improvements in delivery operations, customer satisfaction, and operational efficiency. We are satisfied with our accomplishments, particularly considering that it was our first time experimenting with the tools we utilized. We believe that we have created a product with a strong foundation, which can be further improved and transformed into an effective and high-quality solution.

9. References

[1] Nearest Neighbor Algorithm:

Cover, T. M., & Hart, P. E. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1), 21-27.

Dasgupta, S., & Schulman, L. J. (2008). Analysis of a greedy algorithm: The Lazy Greedy Algorithm. In *Proceedings of the 39th annual ACM symposium on Theory of computing* (pp. 671-680).

[2] Genetic Algorithm:

Holland, J. H. (1975). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. University of Michigan Press.

Goldberger, J., & Ben-Arie, O. (2002). Genetic algorithms for optimization and machine learning. *IEEE Transactions on Evolutionary Computation*, 6(4), 369-380.

[3] Machine Learning Algorithms:

Breiman, L. (2001). Random forests. *Machine learning*, 45(1), 5-32.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3), 273-297.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436-444.

[4] Ant Colony Optimization:

Dorigo, M., & Stützle, T. (2004). *Ant colony optimization*. MIT press.

Bullnheimer, B., Hartl, R. F., & Strauss, C. (1999). An ant colony optimization approach to the generalized traveling salesman problem. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99* (Cat. No. 99TH8406) (Vol. 3, pp. 1472-1477). IEEE.

[5] Find the best delivery management software. *Best Delivery Management Software - 2023 Reviews, Pricing and Demos*. (n.d.). Retrieved January 4, 2023, from <https://www.softwareadvice.com/fleet-management/delivery-management-comparison/>