Zid-Task documentation

Safwan Saigh [safwan9f@gmail.com](mailto:safwan9f@gmail.com), 0534501056

Table of Contents

[II. Introduction 2](#_Toc66400054)

[III. Technologies Used 2](#_Toc66400055)

[1) Databases 2](#_Toc66400056)

[2) Backend language 2](#_Toc66400057)

[3) Backend framework 2](#_Toc66400058)

[4) Backend Libraries 3](#_Toc66400059)

[5) Other services 3](#_Toc66400060)

[6) Hosting services 3](#_Toc66400061)

[IV. Resources 3](#_Toc66400062)

[V. Implementation Summery 4](#_Toc66400063)

[VI. Assumptions and Constraints 5](#_Toc66400064)

[VII. Data used 6](#_Toc66400065)

[VIII. API Usage 7](#_Toc66400066)

[IX. Conclusion 11](#_Toc66400067)

# **Introduction**

This document is used to demonstrate for you what are the technologies and recourses used to develop a Shipment API System. In addition, it illustrates the steps followed to develop the system. Further, the document shows how to use the system. However, this document is used as a task requirement requested by Zid. All the information here is written by the developer (Safwan Saigh). If there is anything not clear, please contact me through my email.

# **Technologies Used**

## Databases

1. MySQL

I have decided to go with SQL database because it is the first thing came to my mind.

1. MongoDB

After a while from implementing the SQL database, I decided to follow the convention of the MEAN stack. In addition, working with NoSQL is much easier that working with SQL database.

## Backend language

1. JavaScript (Node js)

I have chosen Node js because I can work with it quickly. Also, it is more fun than Python(Flask) and PHP.

## Backend framework

1. Express

## Backend Libraries

1. Bcryptjs: Encrypting passwords saved in the database
2. Jsonwebtoken: Authenticate API’s users
3. Mongoose: Work with MongoDB
4. Joi: Validate body requests coming from the frontend
5. Pdfkit: Create labels as pdf documents
6. Aws-sdk: Work with S3 service for storing attachments
7. Multer: File uploads
8. Multer-s3: File uploads to S3

## Other services

1. Postman

## Hosting services

1. Heroku for the backend server
2. FreeDB for the SQL database server
3. MongoDB Atlas for the NoSQL database server
4. Amazon S3 for the files’ storage

# **Resources**

* Stack overflow
* Heroku documentation
* Amazon S3 documentation
* Multer library documentation
* Pdfkit library documentation
* Jwt library documentation
* Youtube
* ShipEngine

# **Implementation Summery**

1. Setup the node js project
2. Implement simple endpoints for listing and creating shipments with proper validation
3. Implement an endpoint for tracking shipments
4. Implement an endpoint to download a label
5. Implement generating shipments labels and save them as pdf docs
6. Update shipment’s schema
7. Setup Heroku server and deploy the app
8. Developing SQL schema for shipments data
9. Setup MySQL database
10. Use the database to save and retrieve shipments
11. Setup the S3 service, this includes creating the buckets that will hold the files
12. Implement uploading files and attachments to the S3
13. Refactoring and bugs fixes
14. Deciding to replace MySQL with MongoDB, this decision was clarified in the Database section
15. Change the database schemas to schemeless, which means tables to collections and tuples to documents
16. Setup MongoDB database in the MongoDB Atlas
17. Create the required models and implement the new database
18. Re-implement the previous functionalities with the new database
19. Refactoring and bug fixes
20. Setup authentication schemas and models
21. Implement register and login a user with decrypting the sensitive information
22. Implement token authentication
23. Refactoring
24. Change the shipment’s model with adding more attributes and enhancing the population from another collections
25. Change the user input validation when creating a new shipment
26. Trying to secure the API more
27. Refactoring and bug fixes

# **Assumptions and Constraints**

1. Assuming there are two actors for this system which are a regular user and an admin
2. Assuming in the frontend side there should be two main views, one for tracking the shipments and the second is an Admin panel. The admin can create and list shipments. Also, print shipments labels.
3. Assuming an admin is a generalized actor for the seller, the warehouse worker, carrier, or the system owner. All of them can enter the system and update a shipment status.
4. **Schemas and Models**
5. MySQL Schema

Diagram

Description automatically generated

1. MongoDB collections and Models
2. Collections: Carriers, Shipments, Users, and TrackingStatuses
3. Models: Carrier, Shipment, User, and TrakingStatus

# **Data used**

I have used the tracking statuses and carrier data used in *ShipEngine* API with little adjustments.

Graphical user interface, application

Description automatically generated

Table

Description automatically generated

# **API Usage**

This section will show you how to use the API based on the functionality.

1. Create a shipment

Example Request

Text

Description automatically generated

Graphical user interface, text

Description automatically generated

Example Response

Text

Description automatically generated

1. Listing previous shipments

Example Request



Example Response

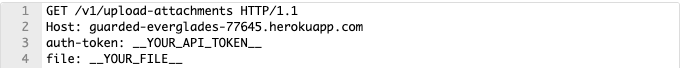
Graphical user interface, text, application

Description automatically generated



1. Attach documents to a shipment

Example Request



Example Response

The attachment will be added to the specified shipment

1. Tracking a shipment

Example Request



Example Response

Graphical user interface, text, application, email

Description automatically generated

1. Get shipments’ labels

Example Request

Text

Description automatically generated

Example Response



Text

Description automatically generated

1. Simple Register

Example Request

Graphical user interface, text

Description automatically generated with medium confidence

Example Response



1. Simple Login

Example Request

Graphical user interface, text, application

Description automatically generated

Example Response Header

auth-token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2MDQ2NWM0YTEyYzk0ZGE2MDIwY2U5YjEiLCJpYXQiOjE2MTU0OTMzNTN9.usVBcNkgs8omdMLOtzaOWEFYEO0VF8MUQRRS-5ZnAJA

# **Conclusion**

In conclusion, I really enjoyed working on such a system and I have learned many things. Unfortunately, it was a very busy 2 weeks due to the midterm exams at KFUPM, so I could not implement the webhooks and the optional task. However, I have more ideas and I believe I can enhance the code as well as some functionality. I hope I did what it requires to have an internship opportunity within Zid. If there is anything not clear, do not hesitate to contact me on safwan9f@gmail.com.