# **Final Project Part 4**

# **Hoosiers Delivery Management System**

**Section-1**

**Application URL:**

We have hosted an application on Render Cloud Application that allows to build, deploy, and operate applications in the cloud.

URL: <https://sample-slb3.onrender.com/>

**Login Credential for web application:**

Username: [hoosier@gmail.com](mailto:hoosier@gmail.com)

Password: Test@123

**GitHub URL:**

We have uploaded the application source code with github.iu.edu at the following URL.

URL: <https://github.com/banumolu2909/geb-ADT-Project>

**Section-2**

**Purpose of the Project:**

Hoosiers Delivery Management System is a web-based platform that enables users to conveniently manage and monitor their deliveries. We are utilizing a dataset from Kaggle which includes detailed information about orders, customers, and products. Using this application, the user can input the necessary delivery details such as the items to be delivered and their destination. Additionally, they can update the delivery status as delivered or in progress or not delivered. The application also allows for basic CRUD (Create, Read, Update, and Delete) operations to be performed on the delivery data points.

**How you built your project:**

**Model:** We are using MySQL Server as our database management system to store data for our web application.

**View:** For our front-end, we are using ReactJS to create a visually appealing and dynamic user interface using material UI components. There will be several operations that can be performed on data like creating, updating, deleting the orders within the user respective workflows.

**Controller (Router):** For the back-end development, we are using NodeJS as the controller in our MVC architecture. NodeJS enables us to get the desired action in a single process. We don't have to create a new thread for every new request.

**Architecture Screenshot**

Graphical user interface, diagram

Description automatically generated

**Front-end view:**

Graphical user interface, application

Description automatically generated

Text, email

Description automatically generated

**Data:**

For our web application, we have chosen MySQL as our database management system. The dataset for this project has been taken from Kaggle. It consists of data regarding the delivery of products ordered throughout United States. We loaded the csv data file into MySQL workbench and the data is organized and structured into tables to minimize data redundancy and to ensure query efficiency using Normalization. We have also performed the preprocessing steps like removing duplicates from the data and converting the data into required data types.

**MySQL connect screenshot**

Text

Description automatically generated

**Tools Used:**

Front End: ReactJs, Material-UI Library

Back End: Nodejs

Database: MySQL

Tools: Visual Studio Code, MySQL Workbench

Deployment Platform: Render

**Functionalities:**

* On the landing page, retailers are required to enter their login ID and password to access the application.
* Once the retailer is logged in, they will see a list of all placed orders and a “Create” button at the top of the page.
* Clicking on the "Create" button takes the retailer to a form where they can enter details for a new order, including product and customer information.
* Upon creating an order, the retailer will be redirected back to the main page where all orders are visible.
* The new order will be stored in the database and will be added to the end of the list in the retailer's dashboard. Every time we want to save a new order, we will be cross-verifying if the customer already exists in the database. If present, we use the same customer id. The same will be followed for the ProductId as well.
* For each order, the retailer can choose to “Read”, “Update”, or “Delete”.
* Clicking on the "Read" button displays the complete order details in read-only mode.
* Clicking on the "Update" button displays a form where retailers can update the data on an existing order.
* Clicking on the "Delete" button removes the selected order data from the list.

**Section-3**

|  |  |
| --- | --- |
| **Name:** Eswar Sai Yashwanth Eanumula |  |
| **Criteria** | Score (1-10) |
| **Task completion** | 10 |
| **Teamwork** | 10 |
| **Time Commitment** | 10 |
| **What could be done better** | Logging is absent in the code, which can make it difficult to troubleshoot issues and monitor performance. To enable efficient troubleshooting and performance monitoring, it's crucial to log events and errors in the database. |

|  |  |
| --- | --- |
| **Name:** Gowri Shankar Badugu |  |
| **Criteria** | Score (1-10) |
| **Task completion** | 10 |
| **Teamwork** | 10 |
| **Time Commitment** | 10 |
| **What could be done better** | Database security needs improvement. Hardcoding connection credentials in the URL is risky. It's better to store them securely, like in an environment variable or configuration file, and load them dynamically instead of hardcoding them. |

|  |  |
| --- | --- |
| **Name:** Bharath Chowdary Anumolu  (banumolu) |  |
| **Criteria** | Score (1-10) |
| **Task completion** | 10 |
| **Teamwork** | 10 |
| **Time Commitment** | 10 |
| **What could be done better** | The view for the web application could be created in a more artistic way so that the page will look good when opened. Should spend some more time to beautify the pages of the application. |