**Team Name:** The Gang
**Team Members Name:** Pallav Jain, Landon Swartz
**Project Title:** Diamondhead Inventory Management

1. **Application Url**: https://whispering-headland-85349.herokuapp.com/
2. **Application Summary:**

   The Inventory Management System that is used currently by Diamondhead Dental Lab is outdated desktop-based legacy software that requires a lot of manual labor and paper trail for inventory management and record keeping. The reports that are required by the upper management are generated manually with the help of excel sheets, which are printed out and kept as physical records. This process is tedious and requires a lot of time and effort to generate sub-optimal reports that are not easy to read. With the inventory software, we hope to create a user-friendly application for Diamondhead to use daily.

   This application is only for employees within Diamondhead to easily track inventory and orders. With this inventory application, admin users are able to see and control all product shipments and orders from customers. Admins are able to create new users for employees that will be handling inventory and orders as well. Purchases are in-house orders that diamond is purchasing to fill their inventory. Orders are outgoing orders to purchase products from Diamondhead. The order comes from an external source, the admin or employee will then fill out the order form to keep track of the inventory going out. The product tab is a master list of the inventory. It tells the employees the product name and quantity that is in stock.

3. **Routing Guide:**

**"Home Route"**
GET "/": This is the home screen of the application.
GET "/aboutus": Give you information about mission, values, and team.

**"Employees Route" -** All employees or Admin should log in before adding employees, product, purchase, and order.
GET "/employees": Give details of all the employee list working in Diamondhead.
GET "/employees/create": Render to create new employees form in Diamondhead.
POST "/employees/create": Submit form data of new employees from the create view to the database.
GET "/employee/:id/edit": Open form to edit employee data.
PUT "/employee/:id/update": Update employee data to the database.

GET "/employee/login": Open login view to login in Diamondhead.
POST "/employee/login": Authenticate the user to login into Diamondhead.
GET "/employee/logout": Employee logout from Diamondhead.

**"Product Route"** - Only Admin can view, add, and edit Product in Diamondhead database.
GET "/product": Details of all products list available in Diamondhead.
GET "/product/create": Render view to add a new product to the Diamondhead database.
POST "/product/create": Admin submits new product form to the Diamondhead database.
GET "/product/:id/edit": Open form to edit product data.

PUT “/product/:id/update”: Update product data to the database.
DELETE “/product/:id/delete”: Delete product data to the database.

**“Purchase Route”** - All Employees and Admin can add the purchase of products in the Diamondhead database.
GET “/purchase”: Details of all purchased products list in Diamondhead.
GET “/purchase/create”: Render view to add a new purchase of a product to the Diamondhead database.
POST “/purchase/create”: Submit new purchase of product form to the Diamondhead database.
GET “/purchase/:pName”: Details of all purchased list by product name.
GET “/purchase/:id/edit”: Open form to edit purchased product data.
PUT “/purchase/:id/update”: Update purchased product data to the database.
DELETE “/purchase/:id/delete”: Delete purchased product data to the database.

**“Order Route”** - All Employees and Admin can add the purchase of products in the Diamondhead database.
GET “/order”: Details of all order list selling products to the customer.
GET “/order/create”: Render view to create a new order for a customer coming to Diamondhead to purchase products.
GET “/order/search”: This is an auto-populated product list used in creating orders and also when adding the purchase of new products route.
GET “/order/get/:id”: Get the details of a product when the employee clicks on a product from the auto-populated list.

4. **User Credentials and Roles:**
   Admin can add new employees or make a new admin in Diamondhead while creating employees.
   Application Roles:
   1. Admin:
      1. Exclusive Route Access: All Employee routes, Product routes, Purchase routes, Order routes.
      2. Sample User Account: email:mark.hoffman@example.com, password: mark

   2. Employee:
      1. Exclusive Route Access: Only see Product list and All Purchase routes, Order routes.
      2. Sample User Account: email:marvin.riley@example.com, password: marvin

5. **Bug Hunt Response**

| **Comment** | **Response** |
|---|---|
| Able to manually enter /employees link logged in as employee to return page and see sensitive information on other employees. | Fixed: Modify the employee route. |
| The Create Purchase says to select from auto-populate list, but it doesn’t pop up until you type in a full word. | You have to type at least 3 words then only it will populate. |

| | |
|---|---|
| Home page styling could have the picture centered, a bit more information regarding the company and the purpose of the application. | Ejs file modified and added more information on the home screen. |

6. **Project Requirements**

| | |
|---|---|
| 5 interlinked dynamic pages (views) that display dynamic, database-driven content per team member. | 1. GET “/employees”: Give details of all the employee list working in Diamondhead<br>2. GET “/product”: Details of all products list available in Diamondhead.<br>3. GET “/purchase”: Details of all purchased products list in Diamondhead.<br>4. GET “/order”: Details of all order list selling products to the customer.<br>5. GET “/purchase/:pName”: Details of all purchased list by product name.<br>6. GET “/employee/:id/edit”: Open form to edit employee data.<br>7. GET “/product/:id/edit”: Open form to edit product data.<br>8. GET “/purchase/:id/edit”: Open form to edit purchased product data.<br>9. GET “/order/search”: This is an auto-populated product list used in creating orders and also when adding the purchase of new products route.<br>10. GET “/order/get/:id”: Get the details of a product when the employee clicks on a product from the auto-populated list. |
| At least 5 database transactions (a transaction includes selecting, inserting, updating, or deleting data from the database) per team member | 1. GET “/employees”: Give details of all the employee list working in Diamondhead.<br>2. PUT “/employee/:id/update”: Update employee data to the database.<br>3. POST “/employees/create”: Submit form data of new employees from the create view to the database.<br>4. DELETE “/employee/:id/delete”: Delete employee data to the database<br>5. GET “/product”: Details of all products list available in Diamondhead.<br>6. POST “/product/create”: Admin submits new product form to the Diamondhead database. |

| | |
|---|---|
| | 7. GET "/purchase": Details of all purchased products list in Diamondhead.<br>8. POST "/purchase/create": Submit new purchase of product form to the Diamondhead database.<br>9. GET "/order": Details of all order list selling products to the customer.<br>10. GET "/employee/login": Open login view to login in Diamondhead. |
| Support for user authentication and secure storage of user credentials | Employees' credentials are managed through the Passport.js package, and users authenticate through a local authentication strategy that verifies the user credentials through the user mongoose model. |
| Preservation of state (may include cookies, session state, url parameters, query string, etc.) | The url parameter and query string is functioning through dotenv library. With the help of express session library, we are maintaining cookies and session state. |
| Development of at least one REST API and consumption of another third-party API | 1. GET "api/product": Details of all products list available in Diamondhead.(Rest API)<br>2. POST "airQuality/search": Shows temperature, pressure, humidity, air quality of search city.(3rd party API) |
| Inclusion and use of at least two npm libraries that are not covered elsewhere in the course | 1. We are using date-format library to format date.<br>2. We are using date picker to select date in create employee, purchase and product and selectmenu to select employee type in create employee page(jQuery) |
| Input validation and error handling for all free-form user entries that could potentially result in errors. | 1. We are using the required key word in html forms.<br>2. For error handling we are using try catch. |