Name : Hency Depani

Enroll. No. : 92200133014

Implementation & Technical Documentation

Project Title : Online Grocery Delivery Web App Blink-Shop-Now

1. **Code Quality**

   I wrote clear code that's easy to read, with notes explaining each section so people can understand it.

   The project is built in parts: the part the user sees, the behind-the-scenes code, and the data setups are separate so each does one thing.

   There are checks to make sure info is correct and errors are handled (like form fields checking if things are filled in; the server gives useful error messages).

   Changes are tracked using Git, with useful notes on each change that show progress (like “Add login feature”, “Fix product listing bug”).

   2. Functionality

   These are the most important working parts of your Blinkit Clone (blinkitcp):

   User Side:

Look through and see grocery items (with pictures, names, costs).

Put items in a cart, change how many, take items out.

Make an order and pay.

Safe way to log in or sign up.

Admin / Back-end Side:

Put in or change items (name, cost, picture).

See orders that users made.

Order & Payment:

The system lets people make orders.

It should work with a way to pay (if set up).

Database:

Keeps track of users, items, orders, cart stuff, and so on.

These parts match what you wanted: a grocery web app that works, with a list of items, a cart, a way to pay, and managing users.

3. Integration Across Components

The part the user sees (React, or something like it) talks to the back-end using APIs (GET, POST, PUT, DELETE).

The Back-End takes care of requests (like “get all items”, “add to cart”, “make order”) and works with the Database to get or save info.

Login checks make sure only real requests (with tokens) can get to special actions.

Things admins do (like adding new items) show up right away for users (item list updates).

The path is: User Interface → API → Back-end Logic → Database → API → Front-End

Testing how things work together makes sure when you click “Add to Cart”, the item shows up; when you make an order, it saves; when admin adds item, users see it.

4. Technical Documentation

4.1 Code Structure

```
/frontend
  /components
    ProductCard.js
    Cart.js
    Navbar.js
  /pages
    Home.js
    Products.js
    Checkout.js
    AdminDashboard.js
  /services
```

```
    api.js   ← functions to call back-end APIs


/ backend
  /routes
    userRoutes.js
    productRoutes.js
    orderRoutes.js
  /controllers
    userController.js
    productController.js
    orderController.js
  /models
    User.js
    Product.js
    Order.js
  /middleware
    authMiddleware.js
    errorHandler.js
  server.js
  config.js  ← database connections, env variables
```

4.2 Implementation Details

Authentication: JWT tokens were the way to go; when a user signs in, the server sends back a token that is then used for future secure requests.

API Design: RESTful API points were made, like GET /products, POST /cart, POST /order.

Database (MongoDB or similar): There were collections (or tables) for Users, Products, Orders, Carts.

Image Upload / Storage: The product pictures were saved (either on the computer or online), and links were used to find them.

Security: HTTPS was used, along with token-based logins, and checking what users entered to stop SQL/NoSQL attacks.

4.3 Testing Procedures

Unit Tests: Checking little parts like product controller tasks (such as adding a product or getting a product).

Integration Tests: Checking how a request goes from the front-end to the back-end to the database; for example, testing an order or adding something to a cart.

Manual Testing / User Testing: Using the app and doing common things (like looking around, using the cart, and paying) on a web browser to see if everything is working right.

During testing, look at the logs to make sure things worked (like server logs and console logs).

4.4 Instructions for Running the System

Copy the repository: git clone

Go into the backend and frontend folders and do npm install (or yarn install)

Make a .env file in the backend with these keys: the database URI, the JWT secret, payment API keys (if used)

Start the backend server: npm run start (or node server.js)

Start the frontend: npm start

Go to http://localhost:3000 on a web browser (or whatever port your front-end uses)