German International University
Faculty of Informatics and Computer Science
Dr. Iman Awaad
Eng. Ahmed Sherif
Amir Haythem
Mohamed Essam
Yassein Eldamasy

Software Engineering, Winter 2025

Project Milestone 4

Submission: Thursday December 18th 2025 at 23:59

Objective

The objective of milestone 4 is to build the **frontend** for the GIU Food-truck System. You will create user interface pages that connect to the backend API endpoints implemented in Milestone 3. The frontend should provide a seamless user experience for both customers and truck owners.

Important: This milestone builds upon Milestone 3. Ensure your backend API endpoints are fully functional before starting the frontend implementation.

Note: Screenshots of each page are provided in this document to help you understand the expected UI design. You may use the provided designs as reference or create your own custom designs that meet the functional requirements.

Technology Stack

You are required to use the following technologies:

- **HTML** - For page structure (.hjs template files)
- **CSS** - For styling (custom styles in style.css)
- **Bootstrap** - CSS framework for responsive UI components
- **JavaScript** - For frontend logic
- **jQuery** - For DOM manipulation and event handling
- **AJAX** - Using jQuery `$.ajax()` for API requests

Libraries included in the project:

- `/styles/bootstrap.min.css` - Bootstrap CSS
- `/styles/style.css` - Custom styles
- `/js/jquery-2.2.0.min.js` - jQuery library
- `/js/bootstrap.min.js` - Bootstrap JavaScript

Project Structure

Your frontend files are organized as follows:

```
Frontend-Extended/
|-- connectors/
|   |-- db.js                # Database connection
|   |-- scripts.sql          # SQL schema and seed data
|-- middleware/
|   |-- auth.js              # Authentication middleware
|-- public/
|   |-- src/                 # JavaScript files
|   |   |-- login.js
|   |   |-- register.js
|   |   |-- trucks.js
|   |   |-- truckMenu.js
|   |   |-- cart.js
|   |   |-- myOrders.js
|   |   |-- ownerDashboard.js
|   |   |-- menuItems.js
|   |   |-- addMenuItem.js
|   |   |-- truckOrders.js
|-- routes/
|   |-- private/
|   |   |-- api.js           # Protected API endpoints
|   |   |-- view.js          # Protected view routes
|   |-- public/
|       |-- api.js           # Public API (register/login)
|       |-- view.js          # Public view routes
|-- utils/
|   |-- session.js           # Session helper functions
|-- views/                   # Hogan.js templates
|   |-- login.hjs
|   |-- register.hjs
|   |-- customerHomepage.hjs
|   |-- trucks.hjs
|   |-- truckMenu.hjs
|   |-- cart.hjs
|   |-- myOrders.hjs
|   |-- ownerDashboard.hjs
|   |-- menuItems.hjs
|   |-- addMenuItem.hjs
|   |-- truckOrders.hjs
|-- screenshots/             # Page screenshots
|-- server.js                # Application entry point
|-- package.json
|-- README.md                # Required documentation
```

Frontend Pages Overview

The following pages must be implemented:

Public Pages (No Authentication Required)

Page	Route	View File	JS File
Login	/	login.hjs	login.js
Register	/register	register.hjs	register.js

Customer Pages (Authentication Required)

Page	Route	View File	JS File
Dashboard	/dashboard	customerHomepage.hjs	-
Browse Trucks	/trucks	trucks.hjs	trucks.js
Truck Menu	/truckMenu/:truckId	truckMenu.hjs	truckMenu.js
Shopping Cart	/cart	cart.hjs	cart.js
My Orders	/myOrders	myOrders.hjs	myOrders.js

Truck Owner Pages (Authentication Required)

Page	Route	View File	JS File
Owner Dashboard	/ownerDashboard	ownerDashboard.hjs	ownerDashboard.js
Menu Items	/menuItems	menuItems.hjs	menuItems.js
Add Menu Item	/addMenuItem	addMenuItem.hjs	addMenuItem.js
Truck Orders	/truckOrders	truckOrders.hjs	truckOrders.js

Page Requirements

1. Login Page

- **Route & Files:** `/` → views/login.hjs, public/src/login.js
- **API Endpoint:** POST /api/v1/user/login

Requirements:

- Display email and password input fields
- Show validation errors for empty fields
- On successful login, redirect to /dashboard
- Display error message for invalid credentials
- Include link to registration page

2. Register Page

- **Route & Files:** `/register` → views/register.hjs, public/src/register.js
- **API Endpoint:** POST /api/v1/user

Requirements:

- Display input fields for: name, email, password, birth date
- Validate all required fields before submission
- On successful registration, redirect to login page
- Display appropriate error messages
- Include link to login page

3. Customer Dashboard

- **Route & Files:** `/dashboard` → views/customerHomepage.hjs

Requirements:

- Display welcome message with user's name
- Show quick action buttons: Browse Trucks, View Cart, My Orders
- Include navigation bar with links to all customer pages
- Display "How it works" section explaining the ordering process
- Include logout button

4. Browse Trucks Page

- **Route & Files:** `/trucks` → views/trucks.hjs, public/src/trucks.js
- **API Endpoint:** GET /api/v1/trucks/view

Requirements:

- Fetch and display all available trucks from API
- Show truck name, logo (or placeholder icon), and status
- Each truck card should have a "View Menu" button
- Clicking "View Menu" navigates to /truckMenu/:truckId
- Display message if no trucks are available

5. Truck Menu Page

- **Route & Files:** `/truckMenu/:truckId` → views/truckMenu.hjs, public/src/truckMenu.js
- **API Endpoints:**
  - GET /api/v1/menuItem/truck/:truckId
  - GET /api/v1/menuItem/truck/:truckId/category/:category
  - POST /api/v1/cart/new

Requirements:

- Extract truckId from URL path
- Fetch and display menu items for the specific truck
- Allow filtering menu items by category
- Show item name, description, price, and category
- Include quantity selector for each item
- "Add to Cart" button for each item
- Display success message when item is added to cart

6. Shopping Cart Page

- **Route & Files:** `/cart` → views/cart.hjs, public/src/cart.js
- **API Endpoints:**
  - GET /api/v1/cart/view
  - PUT /api/v1/cart/edit/:cartId
  - DELETE /api/v1/cart/delete/:cartId
  - POST /api/v1/order/new

Requirements:

- Fetch and display all items in user's cart
- Show item name, price, quantity, and subtotal
- Allow quantity modification with +/- buttons
- Remove item button for each cart item
- Display cart total (sum of all items)
- Scheduled pickup time selector
- "Place Order" button to submit the order
- Display empty cart message with link to browse trucks

7. My Orders Page

- **Route & Files:** `/myOrders` → views/myOrders.hjs, public/src/myOrders.js
- **API Endpoints:**
  - GET /api/v1/order/myOrders
  - GET /api/v1/order/details/:orderId

Requirements:

- Fetch and display all user's orders (sorted by most recent)
- Show order ID, truck name, status, total price, and date
- Color-code order status (pending=yellow, preparing=blue, ready=green)
- "View Details" button to show order items
- Display message if no orders exist

8. Owner Dashboard

- **Route & Files:** `/ownerDashboard` → views/ownerDashboard.hjs, public/src/ownerDashboard.js
- **API Endpoints:**
  - GET /api/v1/trucks/myTruck
  - GET /api/v1/order/truckOrders
  - PUT /api/v1/trucks/updateOrderStatus

Requirements:

- Display truck name and current status
- Show statistics: total menu items, pending orders, completed orders
- Toggle switch or dropdown to change order availability
- Quick action buttons: Add Menu Item, Manage Menu, View Orders
- Display recent orders summary
- Logout button

9. Menu Items Management Page

- **Route & Files:** `/menuItems` → views/menuItems.hjs, public/src/menuItems.js
- **API Endpoints:**
  - GET /api/v1/menuItem/view
  - GET /api/v1/menuItem/view/:itemId
  - PUT /api/v1/menuItem/edit/:itemId
  - DELETE /api/v1/menuItem/delete/:itemId

Requirements:

- Display table/list of all menu items
- Show columns: ID, Name, Category, Description, Price, Status, Actions
- View button to see full item details (using view/:itemId endpoint)
- Edit button to modify item
- Delete button with confirmation dialog
- "Add New Item" button linking to /addMenuItem

10. Add Menu Item Page

- **Route & Files:** `/addMenuItem` → views/addMenuItem.hjs, public/src/addMenuItem.js
- **API Endpoint:** POST /api/v1/menuItem/new

Requirements:

- Form with fields: Name, Category, Description, Price
- Validate required fields (name, category, price)
- Price input should accept decimal values
- "Add Menu Item" submit button
- Cancel button to return to menu items list
- Display success message and redirect on successful creation

11. Truck Orders Page

- **Route & Files:** `/truckOrders` → views/truckOrders.hjs, public/src/truckOrders.js
- **API Endpoints:**
  - GET /api/v1/order/truckOrders
  - GET /api/v1/order/truckOwner/:orderId
  - PUT /api/v1/order/updateStatus/:orderId

Requirements:

- Display all orders for the truck owner's truck
- Allow truck owner to view all order details for a specific order
- Filter tabs: All, Pending, Preparing, Ready, Completed, Cancelled
- Show order ID, customer name, status, total, pickup time
- Status update dropdown for each order
- "Update Status" button to change order status
- Color-coded status badges

API Endpoints Reference

Public Endpoints

Method	Endpoint	Description
POST	/api/v1/user	Register new user
POST	/api/v1/user/login	User login

Customer Endpoints

Method	Endpoint	Description
GET	/api/v1/trucks/view	View available trucks
GET	/api/v1/menuItem/truck/:truckId	View truck menu
GET	/api/v1/menuItem/truck/:truckId/category/:category	Filter by category
POST	/api/v1/cart/new	Add item to cart
GET	/api/v1/cart/view	View cart
PUT	/api/v1/cart/edit/:cartId	Update cart quantity
DELETE	/api/v1/cart/delete/:cartId	Remove from cart
POST	/api/v1/order/new	Place order
GET	/api/v1/order/myOrders	View my orders
GET	/api/v1/order/details/:orderId	View order details

Truck Owner Endpoints

Method	Endpoint	Description
POST	/api/v1/menuItem/new	Create menu item
GET	/api/v1/menuItem/view	View my menu items
GET	/api/v1/menuItem/view/:itemId	View specific item
PUT	/api/v1/menuItem/edit/:itemId	Edit menu item
DELETE	/api/v1/menuItem/delete/:itemId	Delete menu item
GET	/api/v1/trucks/myTruck	View my truck info
PUT	/api/v1/trucks/updateOrderStatus	Update availability
GET	/api/v1/order/truckOwner/:orderId	View order details
GET	/api/v1/order/truckOrders	View truck orders
PUT	/api/v1/order/updateStatus/:orderId	Update order status

README Documentation Requirements

You **MUST** include a comprehensive README.md file in your repository with:

1. **Project Title and Description** - Team members (Name, ID, Tutorial)
2. **Features** - List of implemented features for customers and truck owners
3. **Technology Stack** - Frontend, Backend, Database technologies
4. **ERD** - along with your suggested tables
5. **Installation and Setup** - Step-by-step instructions
6. **Test Credentials** - Email and password for testing
7. **Screenshots** - Screenshots of all implemented pages
8. **API Endpoints Summary** - Table of endpoints
9. **Contributors** - Team member contributions

Screenshots Reference

The following screenshots show the expected UI design for each page.

1. Login Page

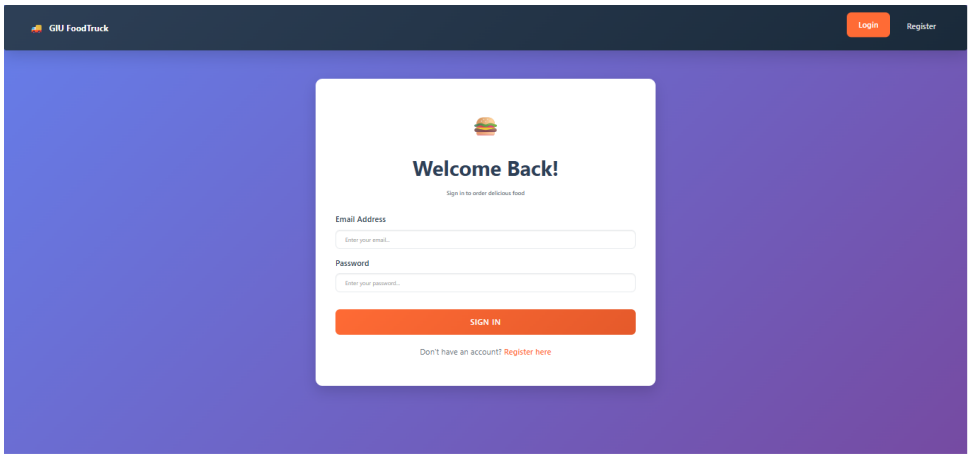

Figure 1: Login page with email and password input fields, sign in button, and link to register

2. Register Page

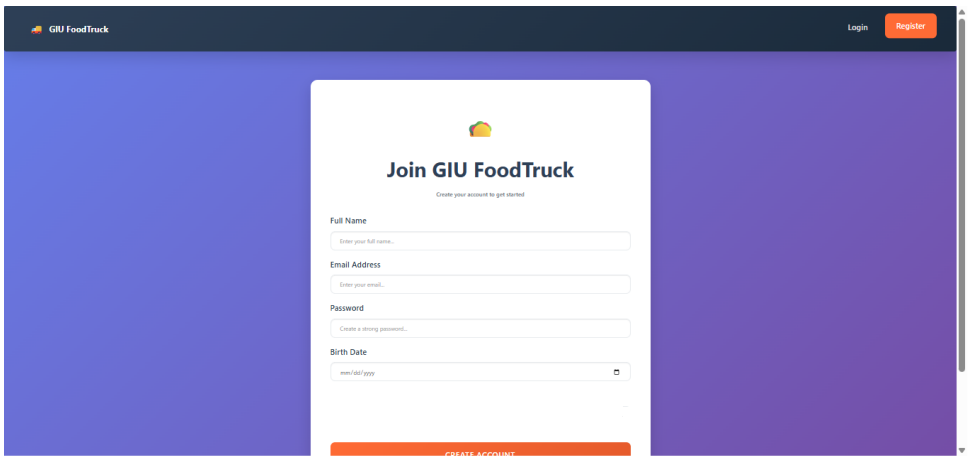

Figure 2: Registration page with name, email, password, and birth date

3. Customer Dashboard

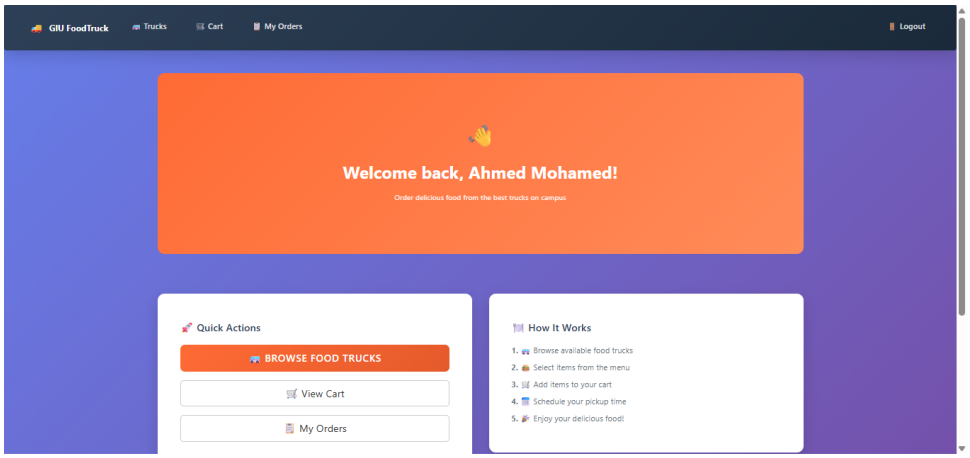

Figure 3: Customer welcome page showing navigation bar, quick actions, and how it works section

4. Browse Food Trucks

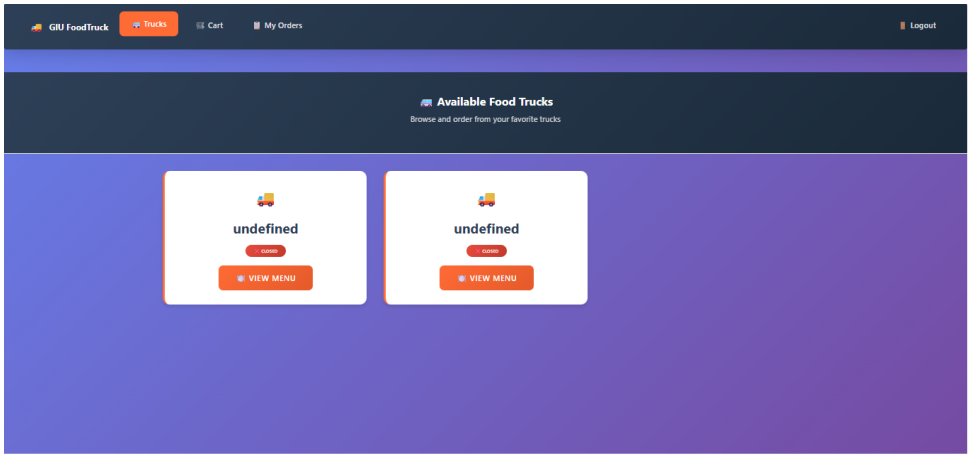

Figure 4: Available food trucks listing with truck cards showing name, status, and View Menu button

5. Truck Menu Page

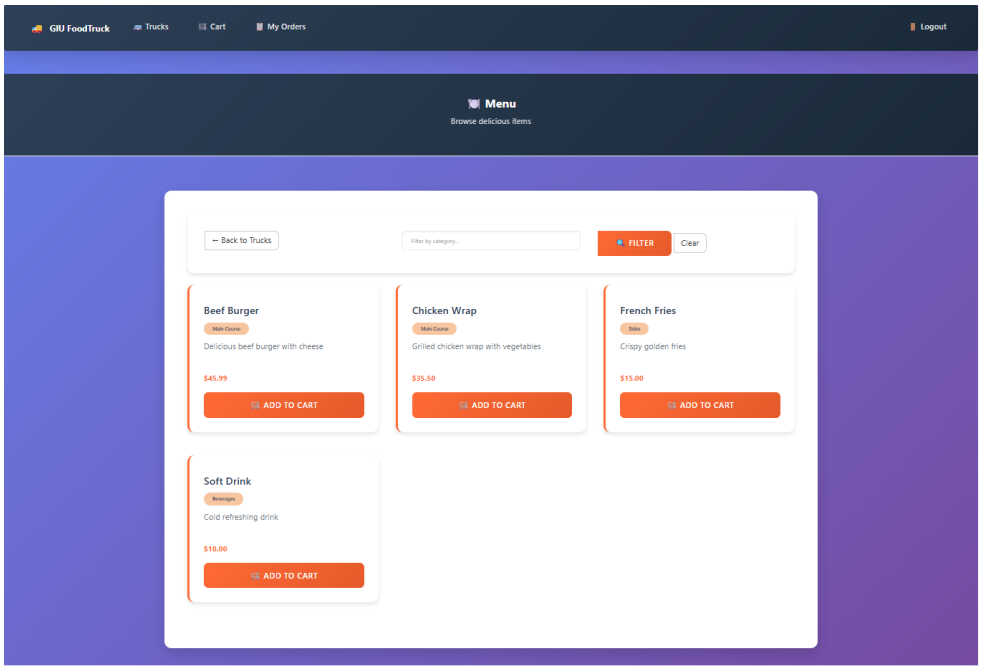

Figure 5: Truck menu page showing menu items with name, category, description, price, and Add to Cart button

6. Shopping Cart

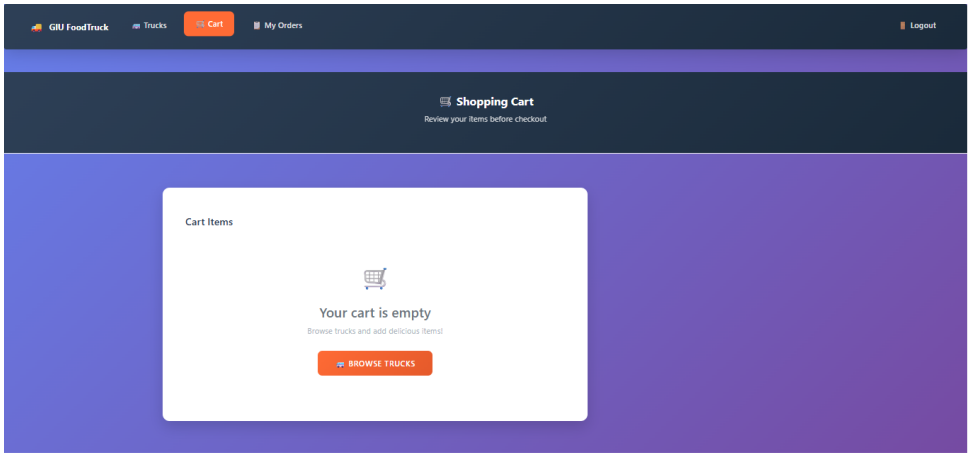

Figure 6: Shopping cart page showing cart items, quantities, prices, and Browse Trucks button when empty

7. My Orders (Customer)

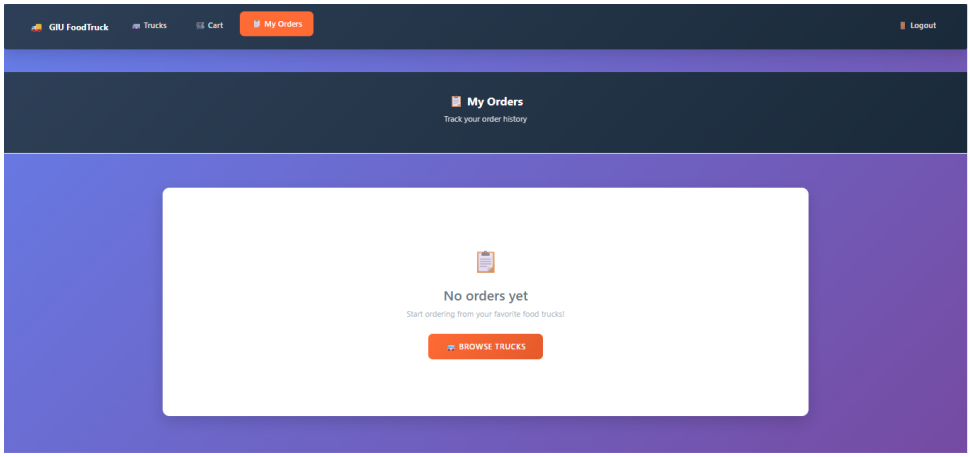

Figure 7: Customer order history page showing orders with status tracking

8. Owner Dashboard

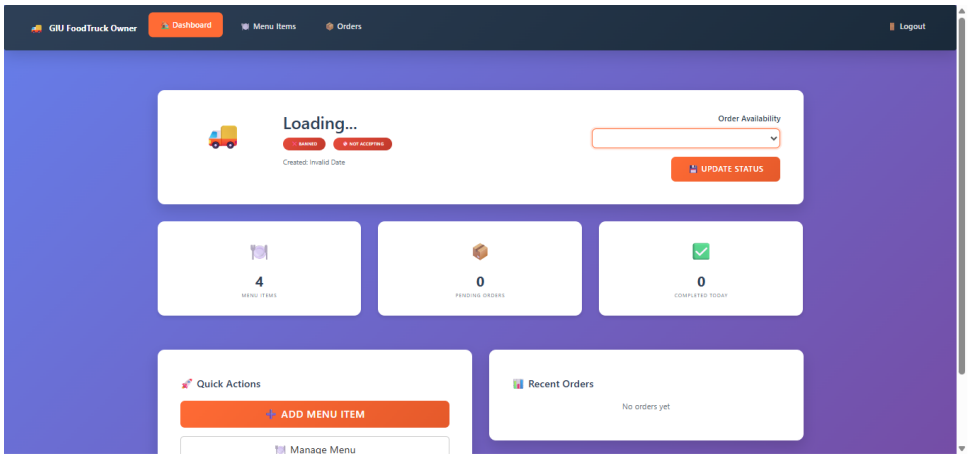

Figure 8: Truck owner dashboard showing truck info, statistics cards, quick actions, and recent orders

9. Menu Items Management

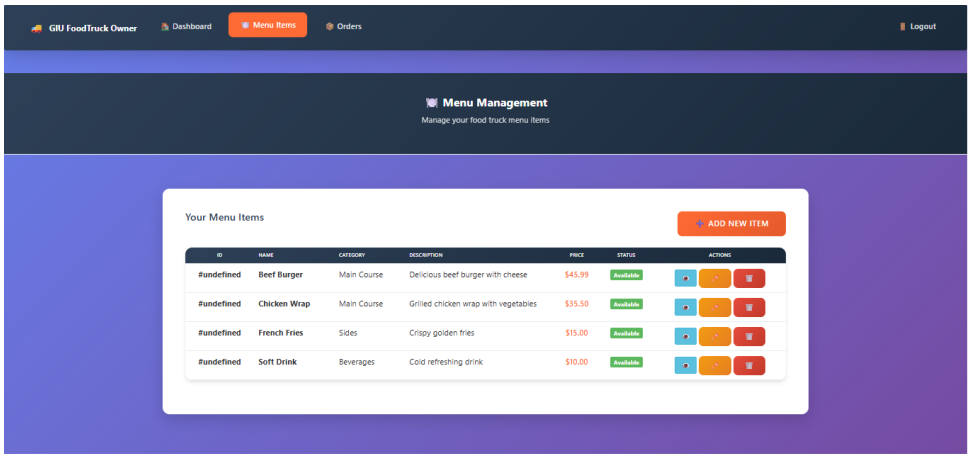

Figure 9: Menu items table showing ID, Name, Category, Description, Price, Status, and action buttons

10. Add Menu Item

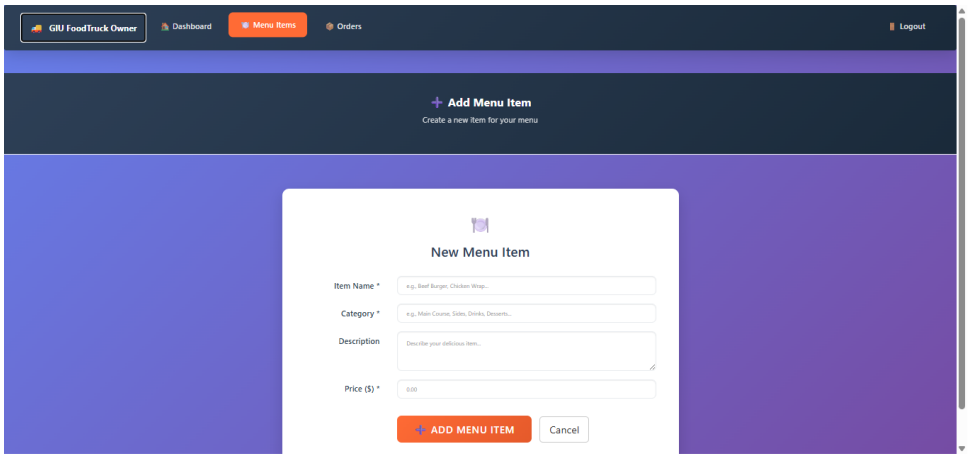

Figure 10: Form to create new menu item with fields for name, category, description, and price

11. Truck Orders Management

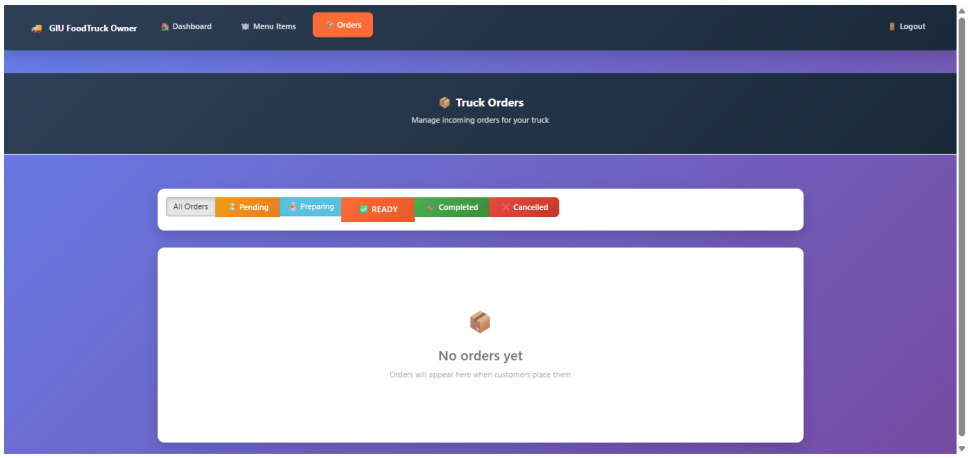

Figure 11: Truck orders page with filter tabs (All, Pending, Preparing, Ready, Completed, Cancelled)

Submission Guidelines

- Submit all your code in **ONE** .zip file
- Rename the .zip folder in the following format: ID-Tutorial#ID-Tutorial (e.g., 1000546-T8#1000678-T9)
- Include all team members' IDs and tutorials in the same format
- Ensure the following files are included:
  - All view files (.hjs) in the views/ folder
  - All jQuery JavaScript files (.js) in the public/src/ folder
  - README.md in the root folder
  - scripts.sql in connectors/ folder
- You must invite the following emails as collaborators to your GitHub repository:
  - amir.haytham@giu-uni.de
  - ahmed.sherif@giu-uni.de
  - mohamed.essam@giu-uni.de
  - yassein.eldamasy@giu-uni.de
- Push all your code to the GitHub repository before submission
- It is **YOUR** responsibility to ensure that you have submitted the correct .zip file before the deadline
- Make sure to submit before deadline
- **Submission Link:** `https://forms.gle/ssFszrAjEAEwgn6Z7`
- Good luck! :)

Important Notes

- The backend API must be fully functional for the frontend to work
- Test all pages thoroughly before submission
- Use browser developer tools to debug JavaScript errors
- Check the Network tab to verify API calls are successful
- Ensure all screenshots in the README match your actual implementation
- The README.md file is **mandatory**