İSTANBUL ÜNİVERSİTESİ-CERRAHPAŞA
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

SOFTWARE DEVELOPMENT DESIGN AND PRACTICE
Project Proposal

Project Name: Bringy

Group Name: PRO-CODE

Group Members:

Student Number	First Name	Last Name	Email
1306230116	Sohail Mohamed	Elskhawy	s.elskhawy@ogr.iuc.edu.tr
1306220129	Ebrahim	Alkridi	ebrahim.alkridi@ogr.iuc.edu.tr
1306210119	Sidra	Bkdash	sdra.alkudsi@ogr.iuc.edu.tr

“Bringy” is a web-based application of an idea that looks like **(Like Getir or Cepte Şok)** There will be a first home page where the customer sees the products when the customer decides to add an item to the basket and checkout he will get sent to the login page for the customers after the customer logs in and verifies his email he will be able to make orders and checkout. The customer will be able to filter the products, sort them, add to basket and edit the basket, and after the customer confirms his purchases, the customer will be able to checkout and see his order in the purchases page. Our Team is not just copying Getir. We have an idea of integrating the app with AI where the customer will be able to talk to a chatbot that will help him add the items he needs with a click of a button.

From the admin side there will also be a login page for the admins and there will be only 2 pages one page to create, read, update, and delete the products, and suppliers.

The second page is to track the orders and change the order status to delivered and print a receipt of the order only admins will be able to see the products and order pages and the rest of staff (the delivery guy for example) will only see the undelivered orders and change their states

Software Requirements Specifications:

Functional Requirements:

- **Login Page (User Registration, User Login, Email Verification):**

    - **User Registration:**
      The user enters an email and password in the frontend. The frontend sends this data to the backend. A verification token is created using JWT and stored in MongoDB. A verification email is sent to the user via Mailjet with a unique link.The user must verify their email before logging in.

    - **Email Verification:**
      The user clicks the verification link in the email. The backend checks the token and verifies the email. The user's status in MongoDB is updated to "verified." The user can now log in.

    - **User Login:**
      The user enters their email and password. The backend checks if the email exists and is verified. The backend compares the password with the password in MongoDB. If correct, a JWT token is generated and sent to the frontend. The frontend stores the token and includes it in requests for protected pages.

- **Home Page (GetAllProducts, filterByCategory, SortbyPrice, addToBasket, AI Chatbot)**
    - **Get All Products:**
      The frontend requests all products from the backend.The backend retrieves product data from MongoDB and sends it to the frontend.The frontend displays the products on the home page.

    - **Filter by Category:**
      The user selects a category (e.g., "Drinks," "Snacks"). The frontend sends a request to the backend with the selected category. The backend retrieves products that match the category and returns them. The frontend updates the product list accordingly.

  - **Sort by Price:**
    The user chooses a sorting option (e.g., "Lowest to Highest," "Highest to Lowest"). The frontend sends a request to the backend with the sorting preference. The backend sorts the products based on price and sends the sorted list. The frontend updates the display with the sorted products.

  - **Add to Basket:**
    The user clicks "Add to Basket" on a product. The frontend updates the basket state and sends a request to the backend. The backend adds the product to the user's basket in MongoDB. The frontend updates the basket icon to reflect the added item.

  - **AI Chat Bot:**
    The user prompt the chatbot that he want to prepare for a meal for example let's say the user want to make a chicken sandwich the AI will create an array of ingredients in general then we will filter the products in db using the given array then we prompt the user if he wants to add the suggested products to the basket

- **Basket Page (Remove Product , Increase/Decrease Quantity)**

  - **Remove Product:**
    The user clicks "Remove" on a product in the basket. The frontend updates the basket state and sends a request to the backend. The backend removes the product from the user's basket in MongoDB. The frontend updates the basket display.

  - **Increase/Decrease Quantity:**
    The user clicks "+" or "−" to change the product quantity. The frontend updates the basket state and sends the new quantity to the backend. The backend updates the product quantity in the user's basket in MongoDB. The frontend updates the displayed quantity and total price.

- **Checkout Page (addAddress, choosePaymentMethod, confirmOrder)**

  The user enters their delivery address in a form.The frontend sends this address to the backend.The backend stores the address in MongoDB and associates it with the user’s order.The user selects a payment method (e.g., "Cash on Delivery," "Credit Card").The frontend sends the selected method to the backend.The user clicks "Confirm Order."The frontend sends the order details to the backend.The backend saves the order in MongoDB then generates an order confirmation email using Mailjet API to send the order details to both the customer and admins.The frontend redirects the user to the order confirmation page.

- **Customer's orders page (see customers orders with its current state , print receipt)**
  - The frontend requests the user's order history from the backend. The backend retrieves all orders associated with the logged-in user from MongoDB. Each order includes details like order items, total price, address, payment method, and current status (e.g., "Processing," , "Delivered"). The frontend displays the orders in a list with their current state.
  - The user clicks the "Print Receipt" button on an order. The frontend generates a formatted receipt with order details and total cost. The user can print the receipt using the browser’s print functionality.

- **Admin's Products Page (Create, Read, Update, Delete Products)**
  - **Create Product:**
    The admin fills out a form with product details (name, category, price, stock, image, etc.).The frontend sends this data to the backend. The backend saves the new product in MongoDB. The product appears in the product list on both the admin and customer sides.

  - **Read Products:**
    The frontend requests all products from the backend.The backend retrieves product data from MongoDB and sends it. The admin sees a list of all products with options to edit or delete them.

  - **Update Product:**
    The admin selects a product to edit and modifies its details. The frontend sends the updated data to the backend.The backend updates the product in MongoDB. The changes reflect in both the admin and customer views.

- **Delete Product:**
  The admin clicks "Delete" on a product.The frontend sends a request to the backend to remove the product.The backend deletes the product from MongoDB. The product is removed from both the admin and customer views.

- **Admin's Orders Page (getOrders, changeOrderStatus, sendEmailToCustomer, PrintRecipt)**
  - **Get Orders:**
    The frontend requests all customer orders from the backend. The backend retrieves orders from MongoDB and sends them. The admin sees a list of orders with details like customer info, products, total price, payment method, and order status.

  - **Change Order Status:**
    The admin selects an order and updates its status (e.g., "Processing" → "Delivered"). The frontend sends the updated status to the backend. The backend updates the order status in MongoDB.

  - **Send Email to Customer (When Status Gets Updated):**
    When the order status is changed, the backend generates an email notification.
    Mailjet API sends the updated order status to the customer.
    The customer receives an email confirming the new status.

  - **Print Receipt:**
    The admin clicks "Print Receipt" for an order.
    The frontend generates a formatted receipt with order details. The admin can print it using the browser’s print function.

Tech-Stacks (MERN Stack):

- **Frontend**:
    - Reactjs for building user interface
    - CSS For Component styling
- **Backend**:
    - Nodejs with Express.js to build RESTful API
    - MongoDB For Database Storage
    - OpenAI API for building the chatbot
    - Mailjet's API: For Email Sending

We Are Planning to divide the app to pages and each page with its functionality among the group members. For example The Login page will be divided into 3 parts and each group member will be working in a part. In That way we all gain experience in frontend and backend and be more productive and each week will be given a report of group member work and progress

Page	**Sohail Mohamed**	**Ebrahim Alkridi**	**Sidra Bkdash**
Login	Customer’s Register Page UI and Authentication and Email Verification Backend	Customer’s Login Page UI and Authentication Backend	Admin’s Login Page UI and Authentication Backend
Home	Products fetching backend function. Products sorting, filtering functions, Page Layout UI design.	Checkout page UI design and functionality (address , and choosing payment method, order confirmation)	basket UI design and functionality (remove product, increase/decrease quantity) backend functions
Admin’s Products	Adding Product to DB backend function, product adding pop up UI design	Products fetching and editing backend function, edit pop up UI design	Page UI design, Product deletion backend function
Admin’s Orders	Order page UI design, order authorization	Order receipt print function	Order status changing backend function
Customers’s Orders	Order receipt print function	Order card UI design	Order page UI design, orders fetching backend functions
AI Chatbot tool	OpenAI api setup, Prompt creation	Collecting Of data for better results, Adding Products To DB	Chatbot UI design

URL of the GitHub repo

https://github.com/SohailElskhawy/Bringy.git

GIT Commands and its explanation

```
// initialize the git repo
git init ()

// add readme.md file with text header
echo "# Bringy Delivery App" > README.md

// add readme.md to staging area
git add README.md

// create a commit with a comment of the steps done
git commit -m "Initial commit: Project setup and README"

// link the local repo to remote repo on github
git remote add origin https://github.com/SohailElskhawy/Bringy.git

// rename the current branch to main
git branch -M main

// pushes the main branch to the remote repository and sets it as upstream branch
git push -u origin main
```