

Test Project Session 4

IT Software Solution for Business

Independent Test Project Designer: Ramin Mohammaddoust

Independent Test Project Validator: Afshin Dehghani

Introduction

In this session, you will be a mobile app developer. You will create an app for Belle Croissant Lyonnais customers. This app will let customers browse products, order, and track their orders.

The focus is on making the app easy to use and nice to look at. It should also connect smoothly to the system made in Session 3.

This session tests your skills in:

- **Mobile App Development:** Making apps that work well on phones.
- **API Integration:** Connecting the app to the existing system.
- **User Interface Design:** Creating an app that is easy and enjoyable to use.
- **Problem-Solving:** Finding and fixing problems as you build the app.

Contents

This session package includes the following materials:

1. **Session Instructions (PDF):** Detailed instructions outlining the tasks to be completed and deliverables expected for this session.
2. **Common Folder:** This folder contains additional resources such as the Belle Croissant Lyonnais logo, icons, style guide, and other design assets that can be used throughout the development of the application.
3. **API Documentation (PDF):** Documentation for the product/customer/order API, providing details about endpoints, request/response formats, and authentication.

Description of Project and Tasks

In this session, you will enhance the Belle Croissant Lyonnais mobile app with new features for custom orders, user management, and order processing.

Guidelines:

1. **Easy to Use:** Make the app simple and easy for customers.
2. **Looks Good:** Follow the Belle Croissant Lyonnais Style Guide for the app's design.
3. **Works Well:** Check that all parts of the app work correctly and without errors.
4. **Secure:** Protect user information and make the app safe to use.
5. **On Time:** Finish all tasks within the time limit.

Technical Considerations:

1. **API Integration:** Follow the best practices in the "Web API Design: The Missing Link" guide when working with the backend APIs.
2. **Data Validation:** Ensure user input is correct and complete before sending it to the APIs.

3. **Error Handling:** Display clear messages to the user if there are any problems.

Additional Considerations:

- The app should work smoothly and quickly on the provided mobile device.
- Use clear labels and instructions for all UI elements.
- Organize information in a way that is easy for users to understand.
- Ensure that the app is accessible to users with disabilities.

Instructions to the Competitor

4.1 Database Design for User Management

Objective:

Design a normalized database schema to support Belle Croissant Lyonnais's user management functionality in their mobile app. The schema should effectively store and manage user data, including authentication credentials, profile information, addresses, and mailing list subscriptions.

Task:

Analyze the following sample user data and create an Entity-Relationship Diagram (ERD) that can accommodate this data and any future expansion needs:

User 1:

```
Email: marie.dupont@example.com
Password: MyPassword1
First Name: Marie
Last Name: Dupont
Phone Number: +33 6 12 34 56 78
Mailing List Subscription: Yes
Profile Picture: /images/marie_dupont.jpg
Security Question: What is your mother's maiden name?
Security Answer: (hashed)
Delivery Addresses:
  Home: 123 Rue de la République, Lyon, 69002, France
  Work: 456 Avenue Jean Jaurès, Lyon, 69003, France (Preferred)
Preferred Delivery Method: Delivery
```

User 2:

```
Email: john.smith@example.com
Password: 12John33
First Name: John
Last Name: Smith
Phone Number: +33 7 89 01 23 45
Mailing List Subscription: No
Profile Picture: /images/john_smith.jpg
Security Question: What was the name of your first pet?
Security Answer: (hashed)
Delivery Addresses:
  Home: 789 Rue Garibaldi, Lyon, 69007, France
Preferred Delivery Method: Pickup
```

Deliverables:

- **File Name:** Session4_UserDB_ERD.pdf
- **File Type:** PDF report containing:

- Screenshot of the implemented database in Microsoft SQL Server Management Studio (SSMS).

Additional Notes:

The database should be able to store multiple delivery addresses per user, with one marked as the preferred address.

- The database should track whether a user has subscribed to the mailing list.
- The database should securely store user passwords and security question answers.
- The database should store the file path or URL of the user's profile picture.
- The database should store the user's preferred delivery method (pickup or delivery).
- The database should be designed with scalability and flexibility in mind, allowing for future additions or modifications to the data model.

4.2 User Management API Development

Objective:

Implement the backend API for the Belle Croissant Lyonnais user management system, adhering to the database schema designed in Task 4.1 and integrating with the main product/order API.

Task:

Implement the following RESTful API endpoints:

- **Authentication:**
 - **POST /api/register:** Create a new user account.
 - **POST /api/login:** Authenticate a user.
 - **POST /api/forgot-password:** Initiate password reset (send security question).
 - **POST /api/reset-password:** Reset password after verifying security answer.
- **Profile Management:**
 - **GET /api/profile:** Get user profile.
 - **PUT /api/profile:** Update profile (including picture upload).
 - **GET /api/addresses:** Get user's addresses.
 - **POST /api/addresses:** Add a new address.
 - **PUT /api/addresses/{id}:** Update an existing address by ID.
 - **DELETE /api/addresses/{id}:** Delete an address by ID.
- **Subscription:**
 - **POST /api/subscribe:** Subscribe to mailing list.
 - **POST /api/unsubscribe:** Unsubscribe from mailing list.
- **Customer Matching:**

- **GET /api/customers:** Fetch customer data from the main Belle Croissant Lyonnais API based on the provided email.

Deliverables:

- **Running API:** Deployed API accessible within the competition environment (e.g., localhost)
- **Session4_UserAPI_Endpoints.txt:** A text file with all implemented API endpoints, HTTP methods, required/optional parameters, and example request/response payloads in JSON format.

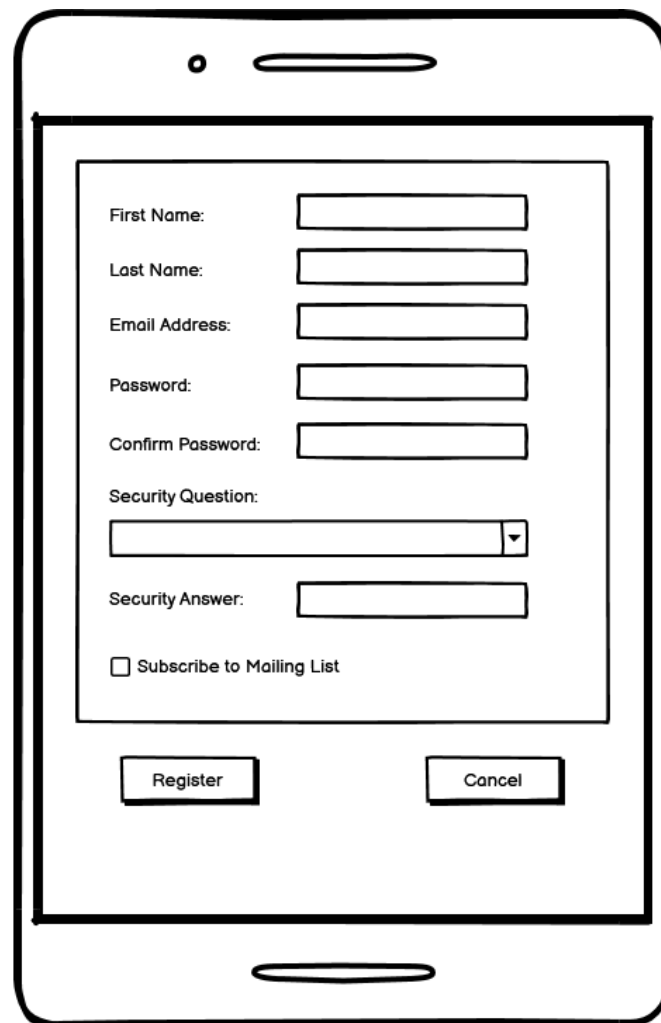
Additional Notes:

- Use the "Web API Design Best Practices.pdf" document for guidance.
- Assume the mobile app will handle image selection and display, the API only needs to store and retrieve image URLs or paths.

4.3 Registration Screen UI

Objective:

Design and implement a user-friendly registration screen for the Belle Croissant Lyonnais mobile app. This screen should allow new customers to create accounts and subscribe to the mailing list.



A wireframe of a mobile registration form displayed on a smartphone screen. The form is contained within a rectangular frame and includes the following elements: text labels for 'First Name:', 'Last Name:', 'Email Address:', 'Password:', 'Confirm Password:', 'Security Question:', and 'Security Answer:'; corresponding input fields (text boxes for names, email, passwords, and answer; a dropdown menu for the security question); a checkbox labeled 'Subscribe to Mailing List'; and two buttons at the bottom labeled 'Register' and 'Cancel'. The smartphone frame includes a camera and sensor at the top and a home button at the bottom.

Task:

1. Design and implement the UI elements as shown in the wireframe.

- **Input Fields:**

- First Name
- Last Name
- Email Address
- Password
- Confirm Password
- Security Question (Dropdown menu with pre-defined options fetched from the API)
- Security Answer

- **Checkbox:**

- "Subscribe to Mailing List"

- **Button:**

- "Register" and "Cancel"

2. Implement Functionality:

- Validate user input (ensure all required fields are filled, email format is correct, passwords match, etc.).
- Send a registration request to the API endpoint /api/register.
- Display appropriate error messages for invalid input or registration failures.
- Upon successful registration, automatically log the user in and navigate to the profile screen.

Deliverables:

- Integrate the registration screen into the Belle Croissant Lyonnais mobile app.

4.4 Login Screen UI

Objective:

Design and implement a user-friendly login screen for the Belle Croissant Lyonnais mobile app, ensuring proper interaction with the user management API and secure authentication.

Task:

1. UI Implementation:

- Design and implement the login screen as shown in the wireframe.
- Include the following elements:
 - Email address input field with validation
 - Password input field with validation
 - "Login" button
 - "Create Account" button (to navigate to registration screen)
 - "Forgot Password?" link (to navigate to password reset screen)

2. Functionality:

- Validate user input (ensure both email and password are provided).
- Send a login request to the API endpoint /api/login.
- If the login is successful:
 - Store the received JWT token securely for future API requests.
 - Navigate to the profile screen.
- If the login fails, display an appropriate error message to the user (e.g., "Invalid credentials").

Deliverables:

- Integrate the login screen into the Belle Croissant Lyonnais mobile app.

- **Create a Test User:** Create a new user account in your user management database with the following credentials:
 - **Username (Email):** testuser@belle-croissant.com
 - **Password:** BCLpass.WSL24

4.5 Forgot Password Screen UI

Objective:

Design and implement a user-friendly "Forgot Password" screen for the Belle Croissant Lyonnais mobile app.

Task:

1. **Design and Implement UI Elements:** Create the following UI elements on the forgot password screen:
 - **Email Address Input:** Text field for entering email address with validation (ensure email address is provided and correctly formatted).
 - **Security Question Display:** Display area for the security question fetched from the API displayed after Email Address Input is correctly entered (On lost focus).
 - **Security Answer Input:** Text field for entering the answer to the security question.
 - **Buttons:**
 - "Verify Answer" button to check the answer against the stored value using the `/api/reset-password` endpoint.
 - "Cancel" button to return to the login screen.
 - **New Password Input (Conditional):**
 - If the security answer is correct, display input fields for entering and confirming a new password.
 - "Reset Password" button to update the password using the `/api/profile` endpoint.
2. **Implement Functionality:**
 - **Send Reset Request:** When the "Send Reset Link" button is clicked, send a password reset request to the API endpoint `/api/forgot-password` with the provided email address.
 - **Display Security Question:** Upon successful request, display the security question fetched from the API.
 - **Verify Answer:** When the "Verify Answer" button is clicked, send the email and answer to the `/api/reset-password` endpoint for verification.
 - **Password Reset:** If the answer is correct, enable the new password input fields and allow the user to set a new password.
 - **Error Handling:** Display appropriate error messages for invalid input, incorrect security answers, or API errors.

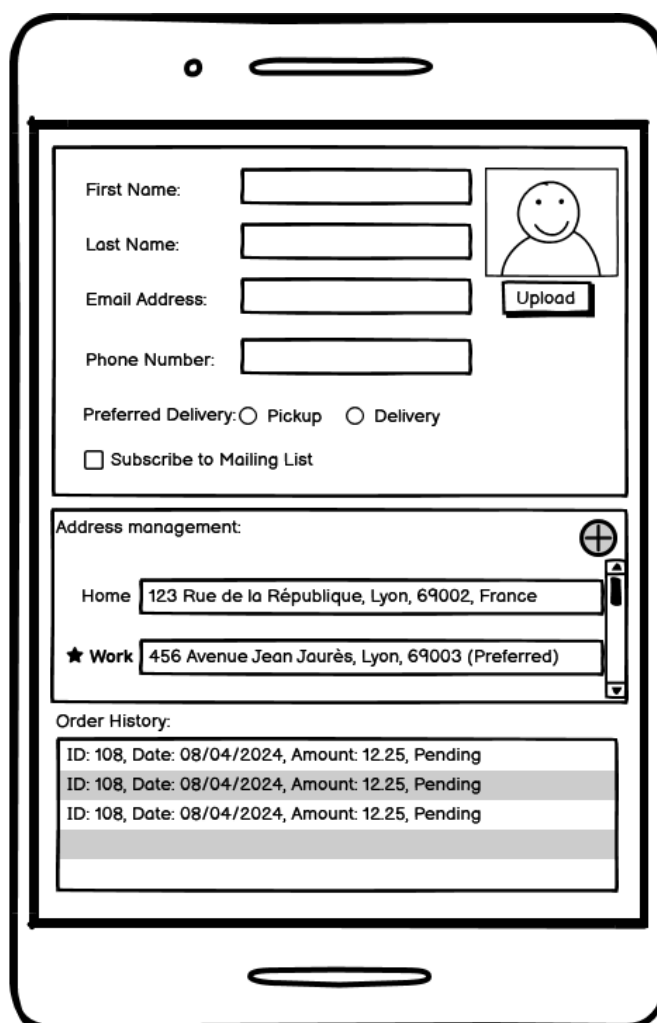
Deliverables:

- Integrate the "Forgot Password" screen into the Belle Croissant Lyonnais mobile app.
- **Create Test User:** Create a new user account in your user management database with the following credentials:
 - **Email:** testuser2@belle-croissant.com
 - **Password:** BCLpass.WSL24
 - **Security Question:** "What is your mother's maiden name?"
 - **Security Answer:** "Smith"

4.6 User Profile Screen UI

Objective:

Design and implement a user profile screen for the Belle Croissant Lyonnais mobile app, allowing users to view and edit their profile information, manage addresses, and view their order history.



The mockup shows a mobile app screen for a user profile. It features a top navigation bar with a hamburger menu icon. The main content area is divided into three sections: Profile Information, Address Management, and Order History.

Profile Information: This section contains input fields for First Name, Last Name, Email Address, and Phone Number. To the right of these fields is a circular profile picture placeholder with a smiley face icon and an 'Upload' button. Below the input fields are radio buttons for 'Preferred Delivery' (Pickup and Delivery) and a checkbox for 'Subscribe to Mailing List'.

Address Management: This section has a title 'Address management' and a plus icon for adding new addresses. It lists two addresses: 'Home' (123 Rue de la République, Lyon, 69002, France) and 'Work' (456 Avenue Jean Jaurès, Lyon, 69003 (Preferred)). The 'Work' address is marked with a star and is the selected address.

Order History: This section has a title 'Order History' and displays a list of orders. Each order entry includes the ID, Date, Amount, and Status. The first three orders are identical: ID: 108, Date: 08/04/2024, Amount: 12.25, Pending.

ID	Date	Amount	Status
108	08/04/2024	12.25	Pending
108	08/04/2024	12.25	Pending
108	08/04/2024	12.25	Pending

Task:

1. Profile Display and Inline Editing:

○ Display Fields (Editable):

- First Name
- Last Name
- Email Address
- Phone Number
- Profile Picture (with upload/change option)
- Mailing List Subscription (toggle switch)
- Preferred Delivery Method (radio buttons: Pickup or Delivery)

○ Inline Editing:

- Allow users to tap on a field to edit it directly on the profile screen.
- Upon editing, visually indicate that the field is in edit mode (e.g., change background color, show "Save" and "Cancel" buttons).
- Validate input as the user types (e.g., check email format).
- If validation fails, display an error message next to the field and prevent saving until corrected.
- Save changes to the user management API (/api/profile) only when the user explicitly clicks "Save."
- Provide visual feedback (e.g., success message) after successful save.
- Implement a "Cancel" button to discard changes.

2. Address Management:

○ Display:

- List of saved addresses with their labels and "Preferred" indicator (if applicable).
- Option to add a new address.

○ Inline Editing:

- Allow inline editing of addresses similar to profile fields.
- Validate address input (e.g., ensure postal code is valid).
- Use the user management API (/api/addresses, /api/addresses/{id}) to add, update, and delete addresses.
- Mark an address as "Preferred" using the API.

3. Order History:

○ Display:

- List of past orders, including:
 - Order ID

- Order Date
- Total Amount
- Order Status

Deliverables:

- Integrate the user profile screen, address management, and order history features into the Belle Croissant Lyonnais mobile app.