ДОДАТОК А
ГРАФІЧНИЙ МАТЕРІАЛ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Харківський національний університет радіоелектроніки
Кафедра ЕОМ

Кваліфікаційна робота
Перший (бакалаврський) рівень

Адаптивний веб-застосунок для замовлення й доставки продуктів харчування

Автор:
Даніїл Касянчук
студ. гр. КУІКУ-21-4

Керівник:
Роман Ярошевич
ст. викл. каф. ЕОМ

02

Мета і задачі роботи

Мета: Розробити адаптивний веб-застосунок для замовлення та доставки продуктів харчування, який забезпечує зручний, зрозумілий та доступний інтерфейс для користувачів.

Задачі:

- Проаналізувати сучасні сервіси доставки продуктів та визначити потреби користувачів.
- Розробити адаптивний, інтуїтивно зрозумілий дизайн інтерфейсу, що забезпечує комфортну взаємодію для усіх користувачів.
- Створити функціональний веб-застосунок, який включає особистий кабінет, систему замовлень та адміністративну панель.
- Забезпечити надійність, безпеку та коректну роботу інтерфейсу на різних пристроях (мобільних і десктопних).

03

Актуальність теми

У сучасному світі зростає попит на онлайн-сервіси.
Це зумовлено:

- пришвидшеним ритмом життя;
- зростанням частки дистанційної роботи;
- глобальними подіями, такими як пандемія;
- та в Україні – воєнними обставинами, переїздами, перебоями в інфраструктурі.

Особливої актуальносіть набувають веб-сервіси.

Fdelivery

04

Аналіз існуючих рішень

05

Технології

В роботі були використані такі технології

06

РЕАЛІЗОВАНИЙ ФУНКЦІОНАЛ

Реєстрація та Ролі

Система створена на базі Node.js: вона дозволяє реєструватися, входити й виходити з облікового запису, а також зберігає дані в локальному сховищі. Додано ролі: адміністратор, кур'єр і користувач.

Створення замовлень та відслідковування

Створена можливість створити замовлення, обравши потрібні продукти, підтвердити його та відстежувати етапи виконання.

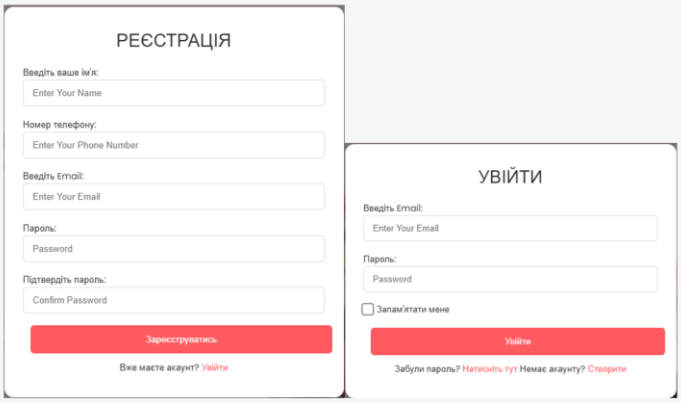
Захист та зберігання та зміна даних

Створено хешування паролів через bcrypt та збереження їх у базі даних. Додано можливість редагування й зміни даних користувачів, а також валідацію ролей.

Реєстрація та аутентифікація

07

Форма реєстрації та входу

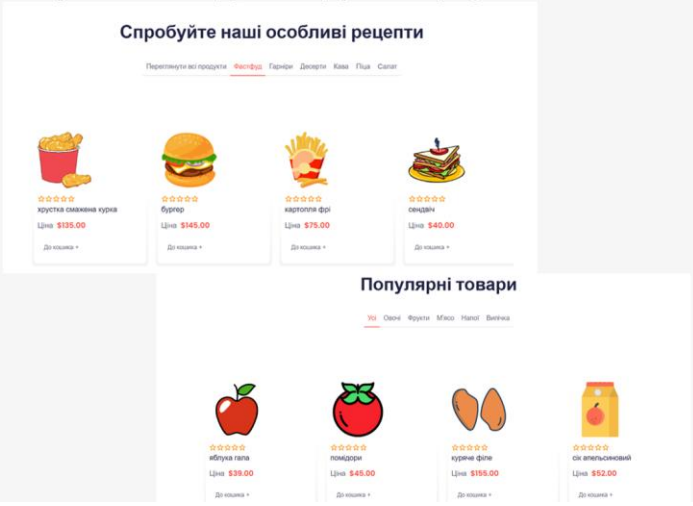


Реалізація коду

Вибір товару та сортування

08

Сторінка вибору товару та сортування

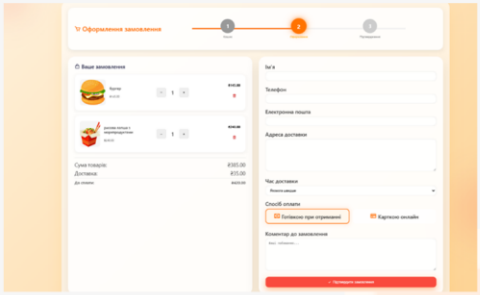


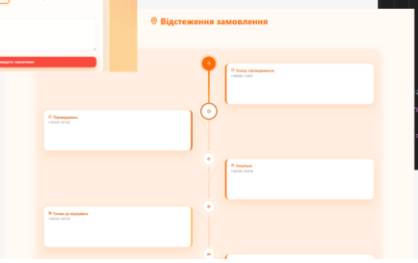
Реалізація коду

Створення замовлення та відслідковування

09

Сторінка оформлення та відстеження замовлень





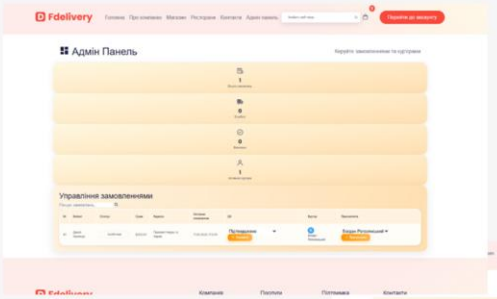
Реалізація коду

```
module.exports = {
  createOrder: (req, res) => {
    const { items, delivery_address } = req.body;
    if (!items || !items.length) return res.status(400).json({ error: 'No items in order' });

    const tx = db.transaction(() => {
      // Calculate total price
      let total = 0;
      const productPrices = new Map();

      items.forEach(item => {
        const product = db.prepare('SELECT price FROM products WHERE id = ? AND merchant_id = ?')
          .get(item.product_id, item.merchant_id);
        if (!product) throw new Error(`Product ${item.product_id} not found`);

        productPrices.set(item.product_id, product.price);
        total += product.price * item.quantity;
      });

      // Create order
      const orderStmt = db.prepare(`
        INSERT INTO orders (customer_id, total_price, delivery_address)
        VALUES (?, ?, ?)
      `);
      const order = orderStmt.run(req.user.id, total, delivery_address);

      // Add order items
      const itemStmt = db.prepare(`
        INSERT INTO order_items (order_id, product_id, quantity, price)
        VALUES (?, ?, ?, ?)
      `);

      items.forEach(item => {
        itemStmt.run(order.lastInsertRowid, item.product_id, item.quantity, productPrices.get(item.product_id));
      });

      return order.lastInsertRowid;
    }

    try {
      const orderId = tx();
      res.status(201).json({ orderId });
    catch (e) {
      res.status(400).json({ error: e.message });
```

Адмін та кур'єр панель

10

Сторінка адмін та кур'єр



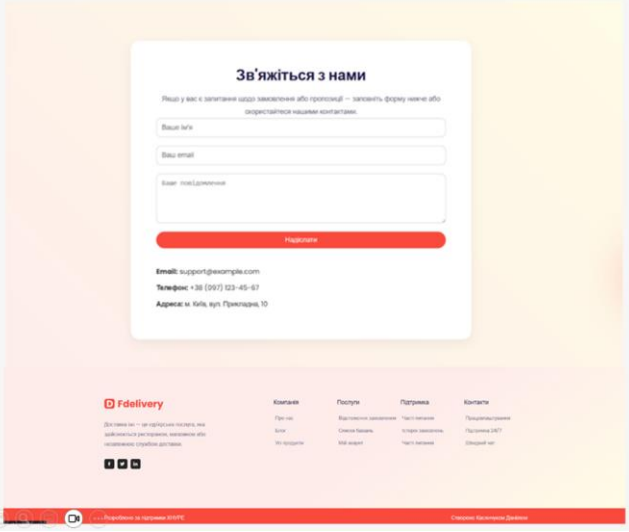
Реалізація коду

```
updateOrderStatus: (req, res) => {
    const { status } = req.body;
    const validStatuses = ['Підтверджено', 'Готується', 'Готово', 'Доставляється', 'Доставлено', 'Скасовано'];

    if (!validStatuses.includes(status)) {
      return res.status(400).json({ error: 'Invalid status' });
    }

    const result = db.prepare(`
      UPDATE orders SET status = ?
      WHERE id = ? AND (merchant_id = ? OR courier_id = ?)
    `).run(status, req.params.id, req.user.id, req.user.id);

    if (result.changes === 0) return res.status(404).json({ error: 'Order not found or not authorized' });
    res.json({ success: true });
  },

  //(admin only)
  assignCourier: (req, res) => {
    const { courier_id } = req.body;

    // Verify courier role
    const courier = db.prepare('SELECT role FROM users WHERE id = ?').get(courier_id);
    courier.role !== 'courier') {
    status(400).json({ error: 'Invalid courier' });

    db.prepare(`
    _id = ?, status = 'in_transit'
    ? AND status = 'ready'
    id, req.params.id);

    ges === 0) return res.status(400).json({ error: 'Order not ready for delivery' });
    ess: true });
```

Контактна форма та хедер

11

Контактна сторінки

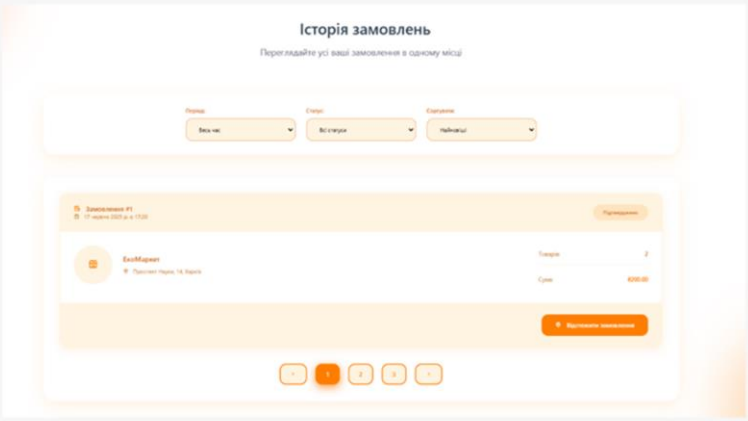


Реалізація коду

Історія замовлень

12

Сторінка історії замовлень

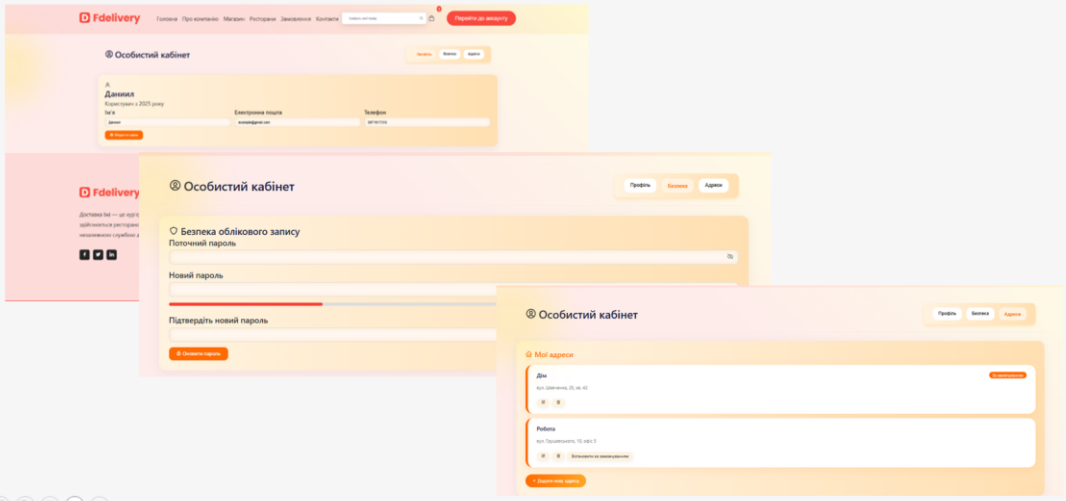


Реалізація коду

Налаштування акаунта та редагування даних

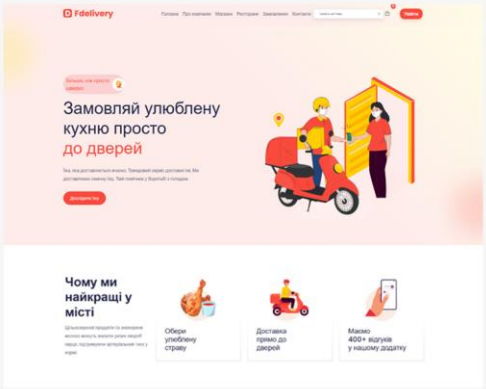
13

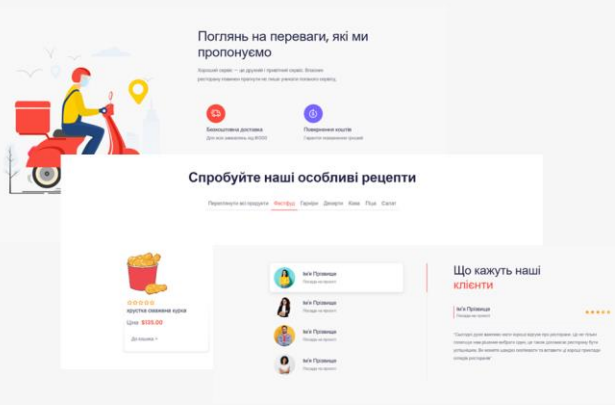
Функціональні сторінки для редагування даних



Лендінг головної сторінки

14

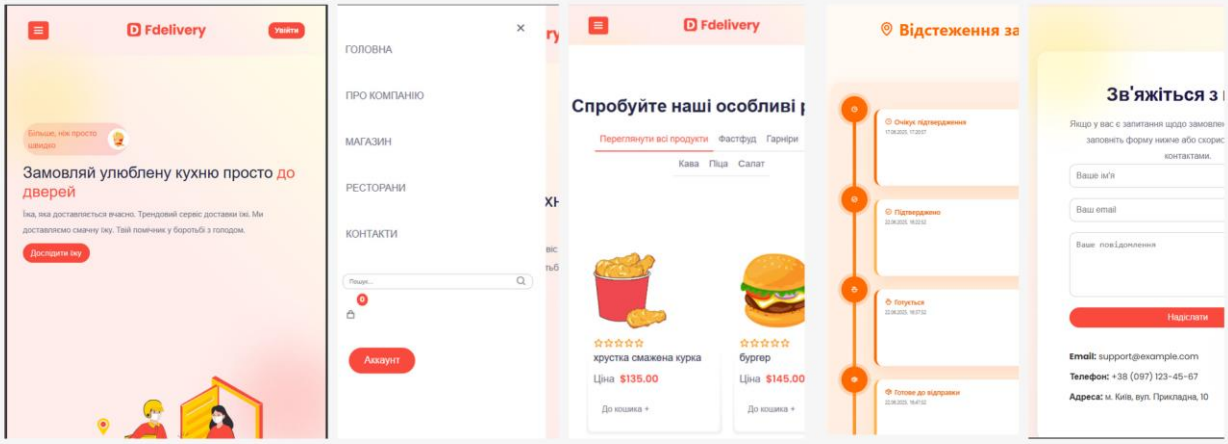




Мобільна адаптація

15

Функціональні сторінки на мобільному пристрої



16

Висновки

Результати роботи

- Реалізовано повноцінний сайт доставки
- Створенно функціонал,які закриває усі потреби
- Створено захист даних
- Реалізовано ролі для роботи з юзером

Перспективи розвитку

- Перехід з SQLite на PostgreSQL
- Інтеграція платіжної системи (Fondy, LiqPay, PayPal)
- Додавання асинхронної обробки подій (Bull.js + Redis)
- Реалізація трекінгу доставки в реальному часі

ДОДАТОК Б
ПРОГРАМНИЙ КОД

Б.1 Файли реалізації функціоналу

```
const db = require('../db');
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');
const { SECRET_KEY, TOKEN_EXPIRY } = process.env;

const tokenExpiry = TOKEN_EXPIRY || '1h';

module.exports = {
    register: async (req, res) => {
        const { email, password, name, phone, role = 'customer'
} = req.body;

        try {
            // Validate role
            const roleRow = db.prepare('SELECT id FROM roles
WHERE name = ?').get(role);
            if (!roleRow) return res.status(400).json({ error:
'Invalid role' });

            // Check email uniqueness
            if (db.prepare('SELECT id FROM users WHERE email =
?').get(email)) {
                return res.status(400).json({ error: 'Email
already exists' });
            }

            // Hash password
            const salt = await bcrypt.genSalt(10);
            const passwordHash = await bcrypt.hash(password,
salt);

            // Create user
            const stmt = db.prepare(`
                INSERT INTO users (name, email, password_hash,
phone, role_id)
                VALUES (?, ?, ?, ?, ?)
            `);
            const result = stmt.run(name, email, passwordHash,
phone, roleRow.id);

            // Generate token
            const token = jwt.sign(
                { id: result.lastInsertRowid, email, role },
                SECRET_KEY,
```

```
                { expiresIn: TOKEN_EXPIRY }
            );

            res.cookie('token', token, {
                httpOnly: true,
                secure: process.env.NODE_ENV === 'production',
                maxAge: 3600000, // 1 hour
                sameSite: 'strict'
            });

            res.status(201).json({
                id: result.lastInsertRowid,
                name,
                email,
                role,
                phone
            });
        } catch (error) {
            console.error('Registration error:', error);
            res.status(500).json({ error: 'Server error during
registration' });
        }
    },

    login: async (req, res) => {
        const { email, password } = req.body;

        try {
            const user = db.prepare(`
                SELECT u.id, u.name, u.email, u.password_hash,
u.phone, r.name AS role
                FROM users u
                JOIN roles r ON u.role_id = r.id
                WHERE u.email = ?
            `).get(email);

            if (!user) return res.status(401).json({ error:
'Invalid credentials' });

            // Verify password
            const validPassword = await bcrypt.compare(password,
user.password_hash);
            if (!validPassword) return res.status(401).json({
error: 'Invalid credentials' });

            // Generate token
            const token = jwt.sign(
                { id: user.id, email: user.email, role:
user.role },
                SECRET_KEY,
                { expiresIn: TOKEN_EXPIRY }
            );
```

```
            res.cookie('token', token, {
                httpOnly: true,
                secure: process.env.NODE_ENV === 'production',
                maxAge: 3600000, // 1 hour
                sameSite: 'strict'
            });

            res.json({
                id: user.id,
                name: user.name,
                email: user.email,
                role: user.role,
                phone: user.phone
            });
        } catch (error) {
            console.error('Login error:', error);
            res.status(500).json({ error: 'Server error during
login' });
        }
    },

    logout: (req, res) => {
        res.clearCookie('token');
        res.json({ message: 'Successfully logged out' });
    },

    getCurrentUser: (req, res) => {
        if (!req.user) return res.status(401).json({ error: 'Not
authenticated' });

        const user = db.prepare(`
            SELECT u.id, u.name, u.email, u.phone, r.name AS
role, u.created_at
            FROM users u
            JOIN roles r ON u.role_id = r.id
            WHERE u.id = ?
        `).get(req.user.id);

        if (!user) return res.status(404).json({ error: 'User
not found' });

        res.json(user);
    },

    updateCurrentUser: async (req, res) => {
        if (!req.user) return res.status(401).json({ error: 'Not
authenticated' });

        const { name, phone, currentPassword, newPassword } =
req.body;
        const updates = [];
        const params = [];
```

```
        if (name) {
            updates.push('name = ?');
            params.push(name);
        }

        if (phone) {
            updates.push('phone = ?');
            params.push(phone);
        }

        // Handle password change
        if (newPassword) {
            if (!currentPassword) {
                return res.status(400).json({ error: 'Current
password is required' });
            }

            const user = db.prepare('SELECT password_hash FROM
users WHERE id = ?')
                .get(req.user.id);

            const validPassword = await
bcrypt.compare(currentPassword, user.password_hash);
            if (!validPassword) {
                return res.status(401).json({ error: 'Invalid
current password' });
            }

            const salt = await bcrypt.genSalt(10);
            const newHash = await bcrypt.hash(newPassword,
salt);

            updates.push('password_hash = ?');
            params.push(newHash);
        }

        if (updates.length === 0) {
            return res.status(400).json({ error: 'No updates
provided' });
        }

        params.push(req.user.id);

        const stmt = db.prepare(`
            UPDATE users
            SET ${updates.join(', ')}
            WHERE id = ?
        `);

        const result = stmt.run(...params);

        if (result.changes === 0) {
            return res.status(404).json({ error: 'User not
```

```
found' });
        }

        res.json({ success: true });
    },
};
```

Б.2 Файл реалізації сторінки проекту

```
<div class="decor decor-left"></div>
<div class="decor decor-right"></div>
<div class="account-container">
    <div class="account-header">
        <h1><i class="bx bx-user-circle"></i> Особистий
кабінет</h1>
        <div class="account-tabs">
            <button class="tab active" data-
tab="profile">Профіль</button>
            <button class="tab" data-
tab="security">Безпека</button>
            <button class="tab" data-
tab="addresses">Адреси</button>
        </div>
    </div>

    <div class="tab-content active" id="profile-tab">
        <div class="profile-card">
            <div class="profile-header">
                <div class="avatar">
                    <i class="bx bx-user"></i>
                </div>
                <div class="user-info">
                    <h2 id="userName">
                        <%= user.name || 'Користувач' %>
                    </h2>
                    <p>Користувач з <%= new
Date(user.created_at).toLocaleDateString('uk-UA', { year:
'numeric' }) %>
                                року</p>
                </div>
            </div>

            <form class="profile-form" id="profileForm">
                <div class="form-grid">
                    <div class="form-group">
                        <label for="name">Ім'я</label>
                        <input type="text" id="name" name="name"
value="<%= user.name || '' %>">
                    </div>

                    <div class="form-group">
                        <label for="email">Електронна
```

```
пошта</label>
                        <input type="email" id="email"
name="email" value="<%= user.email %>" readonly>
                    </div>
                    <div class="form-group">
                        <label for="phone">Телефон</label>
                        <input type="tel" id="phone"
name="phone" value="<%= user.phone || '' %>">
                    </div>
                </div>

                <button type="submit" class="save-btn">
                    <i class="bx bx-save"></i> Зберегти зміни
                </button>
            </form>
        </div>
    </div>

    <div class="tab-content" id="security-tab">
        <div class="security-card">
            <h3><i class="bx bx-shield"></i> Безпека облікового
запису</h3>

            <form class="password-form" id="passwordForm">
                <div class="form-group">
                    <label for="current-password">Поточний
пароль</label>
                    <div class="password-input">
                        <input type="password" id="current-
password" name="current_password" required>
                        <button type="button" class="toggle-
password"><i class="bx bx-hide"></i></button>
                    </div>
                </div>

                <div class="form-group">
                    <label for="new-password">Новий
пароль</label>
                    <div class="password-input">
                        <input type="password" id="new-password"
name="new_password" required>
                        <button type="button" class="toggle-
password"><i class="bx bx-hide"></i></button>
                    </div>
                    <div class="password-strength">
                        <div class="strength-bar"
id="strengthBar"></div>
                        <span id="strengthText">Введіть
пароль</span>
                    </div>
                </div>
                <div class="form-group">
                    <label for="confirm-password">Підтвердіть
```

```
новий пароль</label>
                    <div class="password-input">
                        <input type="password" id="confirm-
password" name="confirm_password" required>
                        <button type="button" class="toggle-
password"><i class="bx bx-hide"></i></button>
                    </div>
                </div>

                <button type="submit" class="save-btn">
                    <i class="bx bx-lock"></i> Оновити пароль
                </button>
            </form>
        </div>
    </div>

    <div class="tab-content" id="addresses-tab">
        <div class="addresses-card">
            <h3><i class="bx bx-home"></i> Мої адреси</h3>

            <div class="address-list" id="addressList">
                <!-- Addresses will be loaded here -->
            </div>

            <button class="btn primary" id="addAddressBtn">
                <i class="bx bx-plus"></i> Додати нову адресу
            </button>

            <div class="address-form" id="addressForm"
style="display: none;">
                <h4>Додати нову адресу</h4>

                <div class="form-group">
                    <label for="address-title">Назва
адреси</label>
                    <input type="text" id="address-title"
placeholder="Наприклад: Дім, Робота">
                </div>

                <div class="form-group">
                    <label for="address-street">Вулиця</label>
                    <input type="text" id="address-street"
required>
                </div>

                <div class="form-grid">
                    <div class="form-group">
                        <label for="address-
building">Будинок</label>
                        <input type="text" id="address-building"
required>
                    </div>
```

```
                    <div class="form-group">
                        <label for="address-
apartment">Квартира</label>
                        <input type="text" id="address-
apartment">
                    </div>
                </div>

                <div class="form-group">
                    <label for="address-
comment">Коментар</label>
                    <textarea id="address-comment"
rows="2"></textarea>
                </div>

                <div class="form-buttons">
                    <button type="button" class="btn outline"
id="cancelAddressBtn">Скасувати</button>
                    <button type="button" class="btn primary"
id="saveAddressBtn">Зберегти адресу</button>
                </div>
            </div>
        </div>
    </div>
</div>

<style>
    :root {
        --accent: #ff6b00;
        --accent-light: #fff0e6;
        --text: #333;
        --muted: #777;
        --border: #eee;
        --bg: #fff;
    }

    body {
        font-family: 'Segoe UI', sans-serif;
        background: linear-gradient(45deg, rgba(249, 74, 61,
0.1) 60%, rgba(255, 229, 7, 0.1));
        color: var(--text);
    }

    .account-container {
        max-width: 1200px;
        margin: 2rem auto;
        padding: 0 1rem;
    }

    .account-header h1 {
        font-size: 2.5rem;
        display: flex;
        align-items: center;
```

```
        gap: 0.5rem;
    }

    .account-tabs {
        display: flex;
        gap: 1rem;
        border-bottom: 2px solid var(--orange-bright);
        background: linear-gradient(to right, #fff3e0, #ffe0b2);
        padding: 1rem 2rem;
        box-shadow: 0 4px 6px rgba(255, 165, 0, 0.2);
        border-radius: 1rem;
    }
   .tab {
    background: #fff;
    border: 16px;
    font-size: 1rem;
    padding: 0.75rem 1.25rem;
    cursor: pointer;
    border-bottom: 3px solid transparent;
    transition: 0.3s;
    border-radius: 10px; /* <--- ДОДАЙ ЦЕ */
}

    .tab.active {
        color: var(--accent);
        font-weight: bold;
        border: 16px;
        border-color: var(--accent);
        background: #fff3e0
    }

    .tab:hover:not(.active) {
        background: #fff3e0; /* мягкий светло-оранжевый */
        transition: 0.3s ease;
        border: 16px;
    }

    .tab-content {
        display: none;
        border: 16px;
        margin-top: 2rem;
    }

    .tab-content.active {
        display: block;
    }

    .profile-card,
    .security-card,
    .addresses-card {
        background: linear-gradient(to right, #fff3e0, #ffe0b2);
        padding: 2rem;
        border-radius: 15px;
```

```
        box-shadow: 0 4px 20px rgba(0, 0, 0, 0.05);
    }

    .form-group,
    .form-grid>div {
        margin-bottom: 1rem;
    }

    .form-group label {
        font-weight: 500;
        display: block;
        margin-bottom: 0.5rem;
    }

    .form-group input,
.form-group textarea {
    width: 100%;
    padding: 0.8rem 1rem;
    border-radius: 8px;
    border: 1px solid #ddd;
    font-size: 1rem;
    background: #fff8f0; /* очень светлый теплый оттенок */
    box-shadow: inset 0 0 8px 2px rgba(255, 182, 100, 0.2); /*
мягкое внутреннее свечение */
    transition: 0.3s, box-shadow 0.3s;
}

.form-group input:focus,
.form-group textarea:focus {
    background: #fff0d6; /* чуть ярче при фокусе */
    box-shadow: inset 0 0 12px 3px rgba(255, 182, 100, 0.4);
    outline: none;
}

    .form-grid {
        display: grid;
        grid-template-columns: repeat(auto-fit, minmax(250px,
1fr));
        gap: 1rem;
    }

    .save-btn,
    .btn.primary {
        background: var(--accent);
        color: white;
        padding: 0.8rem 1.5rem;
        border: none;
        border-radius: 8px;
        cursor: pointer;
        font-size: 1rem;
        transition: background 0.3s;
    }
```

```
.save-btn:hover,
.btn.primary:hover {
    background: #e55e00;
}

.btn.outline {
    background: transparent;
    border: 1px solid var(--accent);
    color: var(--accent);
    padding: 0.8rem 1.5rem;
    border-radius: 8px;
    font-size: 1rem;
    cursor: pointer;
}

.btn.outline:hover {
    background: var(--accent-light);
}

.password-input {
    position: relative;
}

.password-input input {
    padding-right: 40px;
}

.toggle-password {
    position: absolute;
    right: 10px;
    top: 50%;
    transform: translateY(-50%);
    background: none;
    border: none;
    color: var(--muted);
    cursor: pointer;
}

.password-strength {
    display: flex;
    align-items: center;
    margin-top: 0.5rem;
    gap: 0.5rem;
}

.strength-bar {
    height: 6px;
    flex: 1;
    border-radius: 3px;
    background: #ddd;
    position: relative;
}
```

```
    .strength-bar::before {
        content: '';
        display: block;
        height: 100%;
        width: 30%;
        background: #f44336;
        border-radius: 3px;
    }

    .strength-bar.medium::before {
        width: 60%;
        background: #ff9800;
    }

    .strength-bar.strong::before {
        width: 100%;
        background: #4caf50;
    }

    .address-list {
    display: flex;
    flex-direction: column;
    gap: 1rem;
    /* Можно добавить padding, если надо */
}

.address-item {
    background: var(--bg);
    padding: 2rem;
    border-radius: 15px;
    box-shadow: 0 4px 20px rgba(0, 0, 0, 0.05);
    border: none; /* убрать границу */
    display: flex;
    flex-direction: column;
    gap: 0.75rem;
}

/* Сохраняем стили для заголовков и кнопок */
/* --- Заголовок і контейнер --- */
.addresses-card h3 {
    font-size: 1.6rem;
    margin-bottom: 1.5rem;
    color: var(--accent);
    display: flex;
    align-items: center;
    gap: 0.5rem;
}

/* --- Картка адреси --- */
.address-card {
    background: #fff;
    padding: 1.5rem 2rem;
    border-radius: 16px;
```

```
    box-shadow: 0 6px 20px rgba(255, 110, 0, 0.1);
    border-left: 6px solid var(--accent);
    position: relative;
    transition: transform 0.3s ease, box-shadow 0.3s ease;
}

.address-card:hover {
    transform: translateY(-4px);
    box-shadow: 0 8px 24px rgba(255, 110, 0, 0.2);
}

.address-header {
    display: flex;
    align-items: center;
    justify-content: space-between;
}

.address-header h4 {
    margin: 0;
    font-size: 1.25rem;
    font-weight: 600;
}

.default-badge {
    background: var(--accent);
    color: #fff;
    padding: 0.25rem 0.75rem;
    font-size: 0.8rem;
    border-radius: 999px;
}

/* --- Контент адреси --- */
.address-content p {
    margin: 0.5rem 0 0;
    color: var(--muted);
    font-size: 1rem;
}

/* --- Кнопки дій --- */
.address-actions {
    display: flex;
    flex-wrap: wrap;
    gap: 0.5rem;
    margin-top: 1rem;
}

.address-actions .btn {
    border: none;
    background: #fff3e0;
    color: var(--text);
    padding: 0.5rem 1rem;
    font-size: 0.9rem;
    border-radius: 999px;
```

```
    transition: background 0.3s ease, transform 0.2s;
    display: flex;
    align-items: center;
    gap: 0.25rem;
    box-shadow: 0 2px 6px rgba(0,0,0,0.05);
}

.address-actions .btn:hover {
    background: var(--accent-light);
    transform: scale(1.05);
}

/* --- Нова адреса форма --- */
.address-form {
    margin-top: 2rem;
    background: #fff8f0;
    padding: 2rem;
    border-radius: 12px;
    box-shadow: 0 4px 12px rgba(0,0,0,0.05);
    transition: 0.3s ease;
}

/* Кнопка додати нову адресу */
#addAddressBtn {
    margin-top: 1.5rem;
    padding: 0.9rem 1.5rem;
    border-radius: 999px;
    font-weight: 500;
    font-size: 1rem;
    background: linear-gradient(90deg, var(--accent), #ffa726);
    color: #fff;
    transition: 0.3s ease;
    border: none;
}

#addAddressBtn:hover {
    background: linear-gradient(90deg, #e55e00, #ff9800);
}

</style>


<script>
    // Initialize the page
    document.addEventListener('DOMContentLoaded', () => {
        // Tab switching
        document.querySelectorAll('.tab').forEach(tab => {
            tab.addEventListener('click', function () {
                document.querySelectorAll('.tab').forEach(t =>
t.classList.remove('active'));
                document.querySelectorAll('.tab-
content').forEach(c => c.classList.remove('active'));
```

```
                this.classList.add('active');
                document.getElementById(`${this.dataset.tab}-
tab`).classList.add('active');
            });
        });

        // Password visibility toggle
        document.querySelectorAll('.toggle-
password').forEach(button => {
            button.addEventListener('click', function () {
                const input = this.previousElementSibling;
                const type = input.getAttribute('type') ===
'password' ? 'text' : 'password';
                input.setAttribute('type', type);
                this.innerHTML = type === 'password' ? '<i
class="bx bx-hide"></i>' : '<i class="bx bx-show"></i>';
            });
        });

        // Password strength indicator
        document.getElementById('new-
password').addEventListener('input', function () {
            const password = this.value;
            const strengthBar =
document.getElementById('strengthBar');
            const strengthText =
document.getElementById('strengthText');

            let strength = 0;
            let text = 'Слабкий';
            let color = '#f44336';
            let width = '30%';

            if (password.length >= 8) strength += 1;
            if (/[A-Z]/.test(password)) strength += 1;
            if (/[0-9]/.test(password)) strength += 1;
            if (/[^A-Za-z0-9]/.test(password)) strength += 1;

            if (strength === 2) {
                text = 'Середній';
                color = '#ff9800';
                width = '60%';
            } else if (strength >= 3) {
                text = 'Сильний';
                color = '#4caf50';
                width = '100%';
            }

            strengthBar.style.width = width;
            strengthBar.style.backgroundColor = color;
            strengthText.textContent = text;
            strengthText.style.color = color;
        });
```

```
        // Profile form submission
        document.getElementById('profileForm').addEventListener(
'submit', async (e) => {
            e.preventDefault();

            const formData = {
                name: document.getElementById('name').value,
                phone: document.getElementById('phone').value
            };

            try {
                const response = await fetch('/api/auth/me', {
                    method: 'PUT',
                    headers: {
                        'Content-Type': 'application/json',
                        'Authorization': `Bearer
${User.getToken()}`
                    },
                    body: JSON.stringify(formData)
                });

                if (!response.ok) throw new Error('Не вдалося
оновити профіль');

                const updatedUser = await response.json();
                document.getElementById('userName').textContent
= updatedUser.name;
                alert('Профіль успішно оновлено!');
            } catch (error) {
                alert(error.message);
            }
        });

        // Password form submission
        document.getElementById('passwordForm').addEventListener
('submit', async (e) => {
            e.preventDefault();

            const currentPassword =
document.getElementById('current-password').value;
            const newPassword = document.getElementById('new-
password').value;
            const confirmPassword =
document.getElementById('confirm-password').value;

            if (newPassword !== confirmPassword) {
                alert('Новий пароль і підтвердження не
співпадають');
                return;
            }
            try {
                const response = await fetch('/api/auth/me', {
```

```
                    method: 'PUT',
                    headers: {
                        'Content-Type': 'application/json',
                        'Authorization': `Bearer
${User.getToken()}`
                    },
                    body: JSON.stringify({
                        currentPassword,
                        newPassword
                    })
                });

                if (!response.ok) throw new Error('Не вдалося
змінити пароль');

                alert('Пароль успішно змінено!');
                document.getElementById('passwordForm').reset();
            } catch (error) {
                alert(error.message);
            }
        });

        // Address management
        document.getElementById('addAddressBtn').addEventListene
r('click', () => {
            document.getElementById('addressForm').style.display
= 'block';
        });

        document.getElementById('cancelAddressBtn').addEventList
ener('click', () => {
            document.getElementById('addressForm').style.display
= 'none';
        });

        document.getElementById('saveAddressBtn').addEventListen
er('click', async () => {
            const address = {
                title: document.getElementById('address-
title').value,
                street: document.getElementById('address-
street').value,
                building: document.getElementById('address-
building').value,
                apartment: document.getElementById('address-
apartment').value,
                comment: document.getElementById('address-
comment').value
            };

            try {
                // In a real app, you would save the address to
the backend
```

```
                // For now, we'll just add it to the UI
                addAddressToUI(address);
                document.getElementById('addressForm').style.dis
play = 'none';
                document.getElementById('addressForm').reset();
            } catch (error) {
                alert('Помилка при збереженні адреси');
            }
        });

        // Load saved addresses
        loadAddresses();
    });

    function loadAddresses() {
        // In a real app, you would fetch addresses from the
backend
        const addresses = [
            {
                id: 1,
                title: 'Дім',
                address: 'вул. Шевченка, 25, кв. 42',
                isDefault: true
            },
            {
                id: 2,
                title: 'Робота',
                address: 'вул. Грушевського, 10, офіс 5',
                isDefault: false
            }
        ];

        renderAddresses(addresses);
    }

    function renderAddresses(addresses) {
        const container =
document.getElementById('addressList');

        if (addresses.length === 0) {
            container.innerHTML = '<p class="no-addresses">У вас
немає збережених адрес</p>';
            return;
        }

        container.innerHTML = addresses.map(address => `
      <div class="address-card ${address.isDefault ? 'default' :
''}">
        <div class="address-header">
          <h4>${address.title}</h4>
          ${address.isDefault ? '<span class="default-badge">За
замовчуванням</span>' : ''}
        </div>
```

```
        <div class="address-content">
          <p>${address.address}</p>
        </div>
        <div class="address-actions">
          <button class="btn edit-btn" data-id="${address.id}">
            <i class="bx bx-edit"></i>
          </button>
          <button class="btn delete-btn" data-
id="${address.id}">
            <i class="bx bx-trash"></i>
          </button>
          ${!address.isDefault ? `
            <button class="btn set-default-btn" data-
id="${address.id}">
              Встановити за замовчуванням
            </button>
          ` : ''}
        </div>
      </div>
    `).join('');
    }
    function addAddressToUI(address) {
        const container =
document.getElementById('addressList');
        const addressText = `${address.street},
${address.building}${address.apartment ? ', кв. ' +
address.apartment : ''}`;

        const addressCard = document.createElement('div');
        addressCard.className = 'address-card';
        addressCard.innerHTML = `
      <div class="address-header">
        <h4>${address.title || 'Нова адреса'}</h4>
      </div>
      <div class="address-content">
        <p>${addressText}</p>
      </div>
      <div class="address-actions">
        <button class="btn edit-btn">
          <i class="bx bx-edit"></i>
        </button>
        <button class="btn delete-btn">
          <i class="bx bx-trash"></i>
        </button>
        <button class="btn set-default-btn">
          Встановити за замовчуванням
        </button>
      </div>
    `;
        container.prepend(addressCard);
    }
</script>
```