Vanilla Node.js Website with Map — Full Project Files

This document contains a full, working project (no frontend framework) that matches your requirements: homepage, login/register/logout, form page with left-side inputs and right-side map (Leaflet), phone validation, orders saved to SQLite and shown on homepage.

Project structure

```
my-website/
├── server.js
├── package.json
├── README.md
├── db.sqlite            # created at runtime
└── public/
    ├── index.html
    ├── login.html
    ├── register.html
    ├── form.html
    ├── order.html
    ├── style.css
    ├── script.js        # homepage + auth helpers
    └── map.js           # form/map logic
```

Instructions (quick)

1. Create the folder and files from this document.
2. Run `npm install` in the folder.
3. Run `node server.js`.
4. Open `http://localhost:3000`.

---

package.json

```
{
  "name": "my-website",
  "version": "1.0.0",
  "main": "server.js",
```

```
  "scripts": {
    "start": "node server.js"
  },
  "dependencies": {
    "bcrypt": "^5.1.0",
    "body-parser": "^1.20.2",
    "express": "^4.18.2",
    "express-session": "^1.17.3",
    "sqlite3": "^5.1.6"
  }
}
```

---

server.js

```
const express = require('express');
const session = require('express-session');
const bodyParser = require('body-parser');
const sqlite3 = require('sqlite3').verbose();
const path = require('path');
const bcrypt = require('bcrypt');

const app = express();
const PORT = process.env.PORT || 3000;

// Middleware
app.use(bodyParser.urlencoded({ extended: true }));
app.use(bodyParser.json());
app.use(express.static(path.join(__dirname, 'public')));
app.use(session({
  secret: 'replace_this_with_a_real_secret',
  resave: false,
  saveUninitialized: false,
  cookie: { maxAge: 1000 * 60 * 60 * 24 } // 1 day
}));

// DB
const db = new sqlite3.Database(path.join(__dirname, 'db.sqlite'));

// Create tables
db.serialize(() => {
  db.run(`CREATE TABLE IF NOT EXISTS users (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    username TEXT UNIQUE,
    password TEXT
```

```
  )`);

  db.run(`CREATE TABLE IF NOT EXISTS orders (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    user_id INTEGER,
    text_field TEXT,
    checkbox INTEGER,
    select_box TEXT,
    phone TEXT,
    latitude REAL,
    longitude REAL,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY(user_id) REFERENCES users(id)
  )`);
});

// Utility: auth middleware
function requireAuth(req, res, next) {
  if (req.session && req.session.userId) return next();
  return res.status(401).json({ error: 'not_authenticated' });
}

// Routes
app.get('/', (req, res) => {
  res.sendFile(path.join(__dirname, 'public', 'index.html'));
});

// API: register
app.post('/api/register', async (req, res) => {
  const { username, password } = req.body;
  if (!username || !password) return res.status(400).json({ error:
'missing_fields' });

  try {
    const hash = await bcrypt.hash(password, 10);
    const stmt = db.prepare('INSERT INTO users (username, password) VALUES
(?, ?)');
    stmt.run(username, hash, function (err) {
      if (err) {
        if (err.message && err.message.includes('UNIQUE')) return
res.status(409).json({ error: 'user_exists' });
        return res.status(500).json({ error: 'db_error' });
      }
      req.session.userId = this.lastID;
      req.session.username = username;
      return res.json({ ok: true });
    });
    stmt.finalize();
```

```
  } catch (e) {
    return res.status(500).json({ error: 'server_error' });
  }
});

// API: login
app.post('/api/login', (req, res) => {
  const { username, password } = req.body;
  if (!username || !password) return res.status(400).json({ error:
'missing_fields' });

  db.get('SELECT id, password FROM users WHERE username = ?', [username], async
(err, row) => {
    if (err) return res.status(500).json({ error: 'db_error' });
    if (!row) return res.status(401).json({ error: 'invalid_credentials' });

    const match = await bcrypt.compare(password, row.password);
    if (!match) return res.status(401).json({ error: 'invalid_credentials' });

    req.session.userId = row.id;
    req.session.username = username;
    return res.json({ ok: true });
  });
});

// API: logout
app.post('/api/logout', (req, res) => {
  req.session.destroy(err => {
    if (err) return res.status(500).json({ error: 'logout_failed' });
    res.clearCookie('connect.sid');
    return res.json({ ok: true });
  });
});

// API: get current user
app.get('/api/me', (req, res) => {
  if (req.session && req.session.userId) {
    return res.json({ id: req.session.userId, username: req.session.username });
  }
  return res.json({ id: null });
});

// API: submit order
app.post('/api/orders', requireAuth, (req, res) => {
  const { text_field, checkbox, select_box, phone, latitude, longitude } =
req.body;
  if (!text_field || typeof checkbox === 'undefined' || !select_box || !phone
|| !latitude || !longitude) {
```

```
    return res.status(400).json({ error: 'missing_fields' });
  }

  const stmt = db.prepare(`INSERT INTO orders (user_id, text_field, checkbox,
select_box, phone, latitude, longitude)
                           VALUES (?, ?, ?, ?, ?, ?, ?)`);

  stmt.run(req.session.userId, text_field, checkbox ? 1 : 0, select_box, phone,
latitude, longitude, function (err) {
    if (err) return res.status(500).json({ error: 'db_error' });
    const orderId = this.lastID;
    db.get('SELECT * FROM orders WHERE id = ?', [orderId], (err2, row) => {
      if (err2) return res.status(500).json({ error: 'db_error' });
      return res.json({ ok: true, order: row });
    });
  });
  stmt.finalize();
});

// API: get all orders for user
app.get('/api/orders', requireAuth, (req, res) => {
  db.all('SELECT * FROM orders WHERE user_id = ? ORDER BY created_at DESC',
[req.session.userId], (err, rows) => {
    if (err) return res.status(500).json({ error: 'db_error' });
    return res.json({ orders: rows });
  });
});

// Serve form and other static html directly from /public
app.get('/form', (req, res) => res.sendFile(path.join(__dirname, 'public',
'form.html')));
app.get('/login', (req, res) => res.sendFile(path.join(__dirname, 'public',
'login.html')));
app.get('/register', (req, res) => res.sendFile(path.join(__dirname, 'public',
'register.html')));
app.get('/order', (req, res) => res.sendFile(path.join(__dirname, 'public',
'order.html')));

// Start
app.listen(PORT, () => console.log(`Server running at http://localhost:${PORT}
`));
```

---

public/index.html

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width,initial-scale=1" />
  <title>My Website — Home</title>
  <link rel="stylesheet" href="/style.css" />
</head>
<body>
  <header>
    <h1>My Website</h1>
    <div id="auth-area"></div>
  </header>

  <main>
    <section id="info">
      <h2>What this site does</h2>
      <p>This is a simple site where logged-in users can fill a form that
includes a map location. Orders are saved in the database and shown on the
homepage.</p>
      <button id="to-form">Fill the form</button>
    </section>

    <section id="orders-section" style="display:none;">
      <h2>Your Orders</h2>
      <div id="orders-list"></div>
    </section>
  </main>

  <script src="/script.js"></script>
</body>
</html>
```

---

public/login.html

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width,initial-scale=1" />
  <title>Login</title>
```

```
  <link rel="stylesheet" href="/style.css" />
</head>
<body>
  <main class="auth-page">
    <h2>Login</h2>
    <form id="login-form">
      <label>Username<input name="username" required /></label>
      <label>Password<input name="password" type="password" required /></label>
      <button type="submit">Login</button>
    </form>
    <p>Don't have an account? <a href="/register">Register</a></p>
  </main>
  <script>
    document.getElementById('login-form').addEventListener('submit', async (e)
=> {
      e.preventDefault();
      const fd = new FormData(e.target);
      const body = { username: fd.get('username'), password:
fd.get('password') };
      const res = await fetch('/api/login', { method: 'POST', headers: {
'Content-Type': 'application/json' }, body: JSON.stringify(body) });
      if (res.ok) location.href = '/';
      else alert('Login failed');
    });
  </script>
</body>
</html>
```

---

public/register.html

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width,initial-scale=1" />
  <title>Register</title>
  <link rel="stylesheet" href="/style.css" />
</head>
<body>
  <main class="auth-page">
    <h2>Register</h2>
    <form id="register-form">
      <label>Username<input name="username" required /></label>
      <label>Password<input name="password" type="password" required
```

```
minlength="6" /></label>
      <button type="submit">Register</button>
    </form>
    <p>Already have an account? <a href="/login">Login</a></p>
  </main>
  <script>
    document.getElementById('register-form').addEventListener('submit', async
(e) => {
      e.preventDefault();
      const fd = new FormData(e.target);
      const body = { username: fd.get('username'), password:
fd.get('password') };
      const res = await fetch('/api/register', { method: 'POST', headers: {
'Content-Type': 'application/json' }, body: JSON.stringify(body) });
      if (res.ok) location.href = '/';
      else {
        const json = await res.json();
        alert('Register failed: ' + (json && json.error ? json.error :
'unknown'));
      }
    });
  </script>
</body>
</html>
```

---

public/form.html

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width,initial-scale=1" />
  <title>Fill Form</title>
  <link rel="stylesheet" href="/style.css" />
  <link rel="stylesheet" href="https://unpkg.com/leaflet@1.9.4/dist/
leaflet.css" />
  <style>
    /* a few layout tweaks for the form page */
    .two-col { display: flex; gap: 20px; }
    .left { flex: 1; }
    .right { flex: 1; min-height: 400px; }
    #map { height: 100%; min-height: 400px; }
  </style>
</head>
```

```
<body>
  <header>
    <h1>Fill the Form</h1>
    <nav><a href="/">Home</a></nav>
  </header>

  <main>
    <div class="two-col">
      <div class="left">
        <form id="main-form">
          <label>Some text field<input name="text_field" required /></label>
          <label><input type="checkbox" name="checkbox" /> Tick me</label>
          <label>Choose an option
            <select name="select_box" required>
              <option value="">-- choose --</option>
              <option value="option1">Option 1</option>
              <option value="option2">Option 2</option>
              <option value="option3">Option 3</option>
            </select>
          </label>

          <label>Phone (format +123456789)<input name="phone" required
placeholder="+420123456789" /></label>

          <p id="loc-info">No location chosen yet.</p>

          <button id="submit-btn" type="submit" disabled>Submit</button>
        </form>
      </div>

      <div class="right">
        <div id="map"></div>
      </div>
    </div>
  </main>

  <script src="https://unpkg.com/leaflet@1.9.4/dist/leaflet.js"></script>
  <script src="/map.js"></script>
</body>
</html>
```

---

public/order.html

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width,initial-scale=1" />
  <title>Order Details</title>
  <link rel="stylesheet" href="/style.css" />
</head>
<body>
  <header><h1>Order Details</h1></header>
  <main>
    <div id="details"></div>
    <p><a href="/">Back to homepage</a></p>
  </main>
  <script>
    // details can be passed by query string e.g. /order?id=123, but we'll just
let the server response after submission redirect here
    // If the user lands directly, just show a message
    function readQuery() {
      const q = new URLSearchParams(location.search);
      return q.get('id');
    }
    const id = readQuery();
    if (!id) document.getElementById('details').innerText = 'No order to show.';
  </script>
</body>
</html>
```

---

public/style.css

```
body { font-family: system-ui, -apple-system, Segoe UI, Roboto, Arial; margin:
0; padding: 0; }
header { background: #0f172a; color: white; padding: 1rem 1.5rem; display:flex;
justify-content:space-between; align-items:center; }
main { padding: 1.5rem; }
button { padding: .5rem 1rem; border-radius: 6px; border: none; background:
#2563eb; color: white; cursor: pointer; }
button[disabled] { opacity: 0.5; cursor: not-allowed; }
.auth-page { max-width: 420px; margin: 3rem auto; padding: 1rem; border: 1px
solid #ddd; border-radius:8px; }
label { display:block; margin: .6rem 0; }
```

```
input, select, textarea { width:100%; padding: .4rem; border:1px solid #ccc;
border-radius:6px; }
#orders-list { display:flex; flex-direction:column; gap:10px; }
.order-card { padding:10px; border:1px solid #ddd; border-radius:8px; }
```

---

public/script.js

```
// Homepage logic: auth area + show orders + navigation
async function api(path, opts = {}) {
  const res = await fetch(path, opts);
  if (res.headers.get('content-type') && res.headers.get('content-
type').includes('application/json')) return res.json();
  return res;
}

async function refreshAuthArea() {
  const me = await api('/api/me');
  const auth = document.getElementById('auth-area');
  auth.innerHTML = '';
  if (me && me.id) {
    const span = document.createElement('span');
    span.textContent = 'Hello, ' + me.username;
    const outBtn = document.createElement('button');
    outBtn.textContent = 'Logout';
    outBtn.style.marginLeft = '10px';
    outBtn.onclick = async () => { await fetch('/api/logout', { method:
'POST' }); location.reload(); };
    auth.appendChild(span);
    auth.appendChild(outBtn);

    // show orders
    const ordersSec = document.getElementById('orders-section');
    ordersSec.style.display = 'block';
    const list = document.getElementById('orders-list');
    list.innerHTML = 'Loading...';
    const json = await api('/api/orders');
    if (json && json.orders) {
      list.innerHTML = '';
      if (json.orders.length === 0) list.innerText = 'No orders yet.';
      json.orders.forEach(o => {
        const div = document.createElement('div');
        div.className = 'order-card';
        div.innerHTML = `<strong>Order #${o.id}</strong> — ${new
Date(o.created_at).toLocaleString()}<br>
```

```
                        Text: ${escapeHtml(o.text_field)}<br>
                        Phone: ${escapeHtml(o.phone)}<br>
                        Location: ${o.latitude}, ${o.longitude}`;
        list.appendChild(div);
      });
    }
  } else {
    auth.innerHTML = `<a href="/login">Login</a> | <a href="/
register">Register</a>`;
    document.getElementById('orders-section').style.display = 'none';
  }
}

function escapeHtml(s='') { return s.replace(/&/g,'&amp;').replace(/</
g,'&lt;').replace(/>/g,'&gt;'); }

window.addEventListener('DOMContentLoaded', () => {
  refreshAuthArea();
  const btn = document.getElementById('to-form');
  if (btn) btn.addEventListener('click', () => { location.href = '/form'; });
});
```

---

public/map.js

```
// Map + form logic for form.html
let chosen = null; // {lat, lng}
let marker = null;

function phoneValid(val) {
  // simple international phone: starts with + optional, digits, length 7-15
  return /^\+?\d{7,15}$/.test(val.trim());
}

function updateSubmitState() {
  const form = document.getElementById('main-form');
  const data = new FormData(form);
  const text = data.get('text_field') || '';
  const select = data.get('select_box') || '';
  const phone = data.get('phone') || '';
  const checkbox = form.querySelector('[name="checkbox"]') ?
form.querySelector('[name="checkbox"]').checked : false;
  const ok = text.trim() !== '' && select.trim() !== '' && phoneValid(phone) &&
chosen;
  document.getElementById('submit-btn').disabled = !ok;
```

```
}

window.addEventListener('DOMContentLoaded', () => {
  // init map
  const map = L.map('map').setView([50.0755, 14.4378], 13); // Prague by default
  L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
    maxZoom: 19,
    attribution: '© OpenStreetMap'
  }).addTo(map);

  map.on('click', function (e) {
    chosen = { lat: e.latlng.lat, lng: e.latlng.lng };
    if (marker) marker.setLatLng(e.latlng);
    else marker = L.marker(e.latlng).addTo(map);
    document.getElementById('loc-info').innerText = `Selected: $
{chosen.lat.toFixed(6)}, ${chosen.lng.toFixed(6)}`;
    updateSubmitState();
  });

  const form = document.getElementById('main-form');
  form.addEventListener('input', updateSubmitState);

  form.addEventListener('submit', async (e) => {
    e.preventDefault();
    const data = new FormData(form);
    const body = {
      text_field: data.get('text_field'),
      checkbox: form.querySelector('[name="checkbox"]').checked,
      select_box: data.get('select_box'),
      phone: data.get('phone'),
      latitude: chosen.lat,
      longitude: chosen.lng
    };

    // send to server
    const res = await fetch('/api/orders', { method: 'POST', headers: {
'Content-Type': 'application/json' }, body: JSON.stringify(body) });
    const json = await res.json();
    if (res.ok && json.ok) {
      // show confirmation (simple alert) and redirect to homepage
      alert('Order submitted!');
      location.href = '/';
    } else {
      alert('Submit failed: ' + (json && json.error ? json.error : 'unknown'));
    }
  });
});
```

README.md

```
# My Website (Vanilla JS + Node/Express + SQLite)

## Run

1. `npm install`
2. `npm start` or `node server.js`
3. Open http://localhost:3000

## Notes
- Passwords are hashed with bcrypt.
- Map uses Leaflet + OpenStreetMap tiles.
- Phone validation is basic; adjust regex if you need stricter rules.
- Replace the session secret in `server.js` for production.
```

If you want, I can: - produce these as an actual ZIP file you can download, or - create small improvements (email confirmation, edit/delete orders, nicer UI), or - convert to PostgreSQL or another DB.

Tell me which of those you'd like next.