Development/ Reflection Phase Eco Route planner

Manoj Marakala

Matriculation: 4251374

DLMCSPSE01 - project: Software Engineering

The basic MVP of EcoRoute was planned and developed in the course of the Development/Reflection (Agile Sprints 1 and 2) phase. It was built into the full-stack:

- Frontend React + Vite React developer with map and visuals with Leaflet.js and Chart.js, respectively.
- Backend: Node.js + career paths Express CSREST API.
- Database MongoDB Atlanta (free)

Key features realized:

- Full user authentication (register/ login) based on bcrypt + JWT.
- Geolocation browser-based, responsive route input form.
- Original copies of the carbon calculator (static EEA/ EPA emission factors).
- Basic route look-up page, with CO2 savings and eco-badges displayed.

Every piece of work was done under the Phase 1 backlog and project plan. The estimated time was approximately 32 hours, and only free tools were utilized (VS Code, Node.js 20, GitHub, Render.com free tier, and MongoDB Atlas).

Problems and solutions pointed out:

- CORS flaws between frontend (5173) and backend (5000)- resolved with proxy and CORS mid-resolution.
- OpenStreetMap-Nominatim rate limiting - simple caching and fallback to mock data. Only visible here after mining.
- Connection failures on MongoDB free tier - added reconnection code.

Git commits, GitHub issues resolved, and features implemented of planned features reached 18%, 9%, and 72%, respectively. When on the way to testing using real-life routes (e.g., Amsterdam Centraal - Schiphol), manual overnight testing has been used, which demonstrates a 68-87 percent reduction of CO2 when traveling by bike + train rather than the car.

The Agile method, which is used cyclically, was very effective- issues were resolved on the same day rather than on completion. The prototype already available is intuitive and can be used by customers; it also shows that the prototype has customer value for the target group.

Sprint progress table

Sprint Background and Present Position.

Sprint	Goal	Tasks accomplished	Status	Commits
1	Project setup + Auth	Vite+React, express server, JWT auth, MongoDB models	100%	9
2	Core UI + Calculator	Route form, geolocation, carbon service, result page	72%	9
3	Polish + Deployment	Nature of tests, hosting, and final report planned	-	-

This is a view of the running prototype

```
Live prototype screenshots (localhost)

1. Login / Register page        → clean form, responsive on mobile
2. Route input screen           → "From" + "To" fields + "Use my
   location" button
3. Results page                 → Leaflet map + bar chart comparing car
   vs. bike+train
                                  → shows "You save 4.1 kg CO₂ → Bronze
                                    Eco-Badge earned!"
```

The code in Example 1 (carbon calculator)

```
// utils/carbonCalculator.js
const EMISSION_FACTORS = {
  car: 0.192,       // kg CO₂ per km (average EU petrol car)
  bicycle: 0.016,
  walking: 0.0,
  train: 0.041,
  metro: 0.068
};

export const calculateRouteEmissions = (distanceKm, mode) => {
  const emissions = distanceKm * (EMISSION_FACTORS[mode] || 0.192);
  return { distance: distanceKm, co2: emissions.toFixed(2), mode };
};
```

Command 2 (JWT auth route by test)

```
// server/routes/auth.js   (full file – copy exactly like this)
const express = require('express');
const router = express.Router();
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');
const User = require('../models/User');

// POST /api/auth/register
router.post('/register', async (req, res) => {
  const { name, email, password } = req.body;
  try {
    let user = await User.findOne({ email });
    if (user) return res.status(400).json({ msg: 'User already
        exists' });

    user = new User({ name, email, password });
    const salt = await bcrypt.genSalt(10);
    user.password = await bcrypt.hash(password, salt);
    await user.save();

    const token = jwt.sign({ id: user._id }, process.env.JWT_SECRET || 'fallbacksecret', {
        expiresIn: '7d' });
    res.json({ token });
  } catch (err) {
    res.status(500).json({ msg: 'Server error' });
  }
});

// POST /api/auth/login   ← this is the one you tried to run
router.post('/login', async (req, res) => {
  const { email, password } = req.body;
  try {
    const user = await User.findOne({ email });
    if (!user) return res.status(400).json({ msg: 'Invalid credentials' });

    const isMatch = await bcrypt.compare(password, user.password);
    if (!isMatch) return res.status(400).json({ msg: 'Invalid credentials' });

    const token = jwt.sign({ id: user._id }, process.env.JWT_SECRET || 'fallbacksecret', {
        expiresIn: '7d' });
    res.json({ token, user: { id: user._id, name: user.name } });
  } catch (err) {
    res.status(500).json({ msg: 'Server error' });
  }
});

module.exports = router;
```

Risk & decision log

Risk & Decision Log (selected entries)

Date	Risk/ Problems	Decision/Fix
2025-11-18	CORS blocking frontend-backend	Added CORS middleware + Vite proxy
2025-11-20	Nominatim 1req/sec limit	simple in-memory caching and fallback with mock data
2025-11-21	MongoDB disconnection repository on Render	mongoose reconnect logic + local fallback