Test Project

Session 5 – CityCycle

IT Software Solutions for Business

Submitted by:
Bence Lakos - Ingenimind Kft, Independent Test Project Designer
Benjamin Frost, Independent Test Project Validator

WSC2026_TP09_en

Contents

Introduction **3**
Description of project and tasks **3**
The domain and requirements (what the design must support) 3
Your deliverables 4
The standard expected (what "good" looks like) 4

© WorldSkills International

Introduction

CityCycle, a non-profit, is launching a city shared-mobility service - pedal bikes and e-scooters parked across the city. An automated design assistant produced a first cut of the system design overnight, and **nobody has reviewed it**. Before a line of code is written, the city needs that design **reviewed and put right**: the data model, the system boundaries, the API and the security decisions are expensive to change once built - and the draft has problems. You are the developer brought in to fix it.

Description of project and tasks

Review the supplied **DraftDesign v0.1** against the requirements below, find what is wrong or missing, then deliver a **corrected and improved design** together with a short critique explaining your changes. You write **no application code -** you produce the design artefacts and the reasoning behind them. The draft is plausible but contains real defects across the data model, architecture, API, and security/privacy, and it leaves some requirements unmet. Everything you need is in this document and the draft pack — **no prior mobility knowledge is required**.

The domain and requirements (what the design must support)

CityCycle runs a fleet of **vehicles** (pedal bikes and e-scooters; e-scooters report a **battery level**). Vehicles sit in **zones** — designated parking areas across the city; a trip must start and end in a zone. **Members** register with contact details and a payment method and may hold a **pricing plan** (e.g. Pay-as-you-go or monthly).

- **Trip:** begins when a member unlocks a vehicle in a zone and ends when they lock it in a zone; it records start/end time, start/end zone, and distance.
- **Pricing:** an unlock fee plus a per-minute rate, modified by the member's plan (e.g. a monthly plan waives the unlock fee and includes some free minutes), with a **daily cap**. Payment is taken per trip or settled monthly.
- **Vehicles** report a **status** (available, in-use, low-battery, out-of-service) and accumulate **main-tenance tickets**; members can file **incident reports** (damage, hazard).
- **Member roles:** most members are riders; some are **operators**, who rebalance vehicles between zones and handle maintenance — the model must distinguish the two.
- **Non-functional:** vehicle availability and location must be queryable in near-real-time; the system must resolve **two members trying to unlock the same vehicle** (only one succeeds); personal data, trip/location history and payment details are **sensitive and regulated**.

The target environment is the team's standard **.NET + relational-database** stack.

© WorldSkills International

Your deliverables

1. **Design review** (Review.md) — the defects you found in DraftDesign v0.1: each with **what** is wrong, **why** it matters, and **your fix**; plus any requirements the draft fails to meet.
2. **Data model** - corrected in two files:

- `schema.sql` - the relational data model and the single place you fix it: entities, primary keys, foreign keys and referential integrity, normalisation to 3NF (no duplicated or misplaced data), the missing entities, plus column types, NOT NULL , uniqueness, and money/timestamp precision.
- `DataDictionary.md` - a short **data dictionary** describing what the data **means** and how it must be governed, not its SQL types: one row per attribute giving its meaning, units (where applicable), whether it is personal data (PII), and any retention rule. A draft version ships in the pack — correct and complete it.

3. **Architecture** - corrected C4 diagrams: context, container, and component.
4. **API contract** (API.md) - a **REST** design for **these flows only**: start a ride, end a ride, the rider's own ride history, available vehicles for the map, the rider's own profile, and filing an incident. For each, give the method, path, request and response bodies, and status codes — under a `/v1` prefix,
5. token auth, a single error model, and pagination on lists. You do not need registration / login / password endpoints (assume a bearer token is issued) or general CRUD for other entities — that boilerplate is out of scope. `200` `{ ok, … }` Replace the draft's always - envelope with proper HTTP status codes. A start-/end-ride sequence diagram is encouraged.
6. **Decision log** (Decisions.md) — your key trade-offs, the non-functional decisions (concur-rency, real-time availability), and the security/privacy choices, each justified.

Given the 2.5 hours, model what you can and **outline** what you cannot — note any gap and sketch your approach (in Decisions.md) rather than leaving it silent. The operator role and vehicle rebalancing in particular may be outlined rather than fully modelled.

The standard expected (what "good" looks like)

- **Data model:** no update/insert/delete anomalies; relationships at the right cardinality (history preserved, not overwritten); sensible keys and constraints.
- **Architecture:** clear C4 boundaries and separation of concerns; the client does not reach the database directly; cross-cutting concerns (auth, billing, real-time location) are placed deliberately; the design addresses the stated NFRs.
- **API:** resource-oriented, correct verbs and status codes, a consistent error shape, authentica-tion on protected operations, idempotency where it matters (unlock/lock), pagination on collections, and versioning.
- **Security and privacy:** payment data and credentials handled correctly (never stored in the clear); access control between members (no member can act on another's trips/data); a stated retention stance for location history.
- **Communication:** diagrams and documents an engineer could build from, and decisions a reviewer could defend.

© WorldSkills International