



DriveTrack

A Smart Driving Test Booking & Progress Web App with AI Coaching

BSc Software Engineering Project

Course: CIS017-3 Undergraduate Project

2026

Abstract

Driving lesson and test booking systems are frequently disjointed, providing either minimalist calendar features or instructor-only options, and few features of learner progress monitoring or individual coaching. As a result of this project, DriveTrack, a cross-platform local web application that integrates booking a test, automatically allocating a slot, tracking progress based on milestones, and an AI coach, were developed. Constructed in Node.js, Express, and local JSON storage, the app has a single unified interface that allows learners to book lessons, track six main milestones (between Profile Created and Driving Test Passed) and get advice on what to practise next based on Groq/OpenAI API. The artefact was tested using technical walkthrough and requirements traceability, it was proved that all functional objectives were achieved. Weaknesses are a lack of individual instructor interface and cloud synchronisation. The project is adding a lightweight, privacy-conscious product that fills an apparent gap in the market. The role-based access, cloud storage, and formal usability testing using the System Usability Scale (SUS) are to be done in the future.

Acknowledgemnet

would like to say a few words of thanks to my supervisor Dr Yanran Li, who provided me with invaluable advice, patience and constructive criticism during this project. Her teachings assisted me to steer the course of this work and her support was the one that kept me going in the difficult moments. Without her support, this would not have been achieved.

Dedication

I am dedicating this project to my best friend who always believed in me and to all my friends who supported and supported me in the process. You made this trip fun with your cooperation, late-night conversation and laughter. Thanks to you and your support during all hardships and feasts

Table of Contents

<u>CHAPTER 1: INTRODUCTION.....</u>	<u>9</u>
1.1 BACKGROUND OF DRIVING LESSON & TEST BOOKING SYSTEMS	9
1.2 PROBLEM STATEMENT	9
1.3 AIM AND SMART OBJECTIVES.....	10
1.4 SCOPE OF THE PROJECT	11
1.5 STRUCTURE OF THE REPORT	12
1.6 ETHICAL CONSIDERATIONS.....	12
<u>CHAPTER 2: LITERATURE REVIEW & MARKET RESEARCH.....</u>	<u>13</u>
2.1 INTRODUCTION TO THE CHAPTER.....	13
2.2 EVOLUTION OF DRIVING LESSON BOOKING SYSTEMS.....	13
2.3 CRITICAL ANALYSIS OF TECHNOLOGIES USED	14
2.4 SECURITY IN WEB APPLICATIONS.....	15
2.5 AI COACHING IN EDUCATIONAL TOOLS	16
2.6 MARKET POSITIONING: DRIVETRACK VS. EXISTING PLATFORMS.....	16
2.7 IDENTIFICATION OF THE PROJECT'S NICHE	17
<u>CHAPTER 3: DESIGN & PLANNING.....</u>	<u>17</u>
3.1 INTRODUCTION.....	17
3.2 SYSTEM ARCHITECTURE (HIGH LEVEL)	17

3.3 FRONT END DESIGN.....	19
3.4 BACK END DESIGN	19
3.4.1 API ENDPOINTS	19
3.4.2 AUTO-SLOT ALLOCATION ALGORITHM	20
3.5 DATABASE SCHEMA DESIGN	22
3.6 RESOURCE REQUIREMENTS	23
3.7 TESTING STRATEGY	23
3.8 DEPLOYMENT PLAN	24
3.9 SUMMARY	24
<u>CHAPTER 4: IMPLEMENTATION</u>	<u>24</u>
4.1 INTRODUCTION.....	24
4.2 DEVELOPMENT ENVIRONMENT SETUP	25
4.3 BACK END IMPLEMENTATION	25
4.3.1 SERVER INITIALISATION.....	25
4.3.2 CROSS-PLATFORM DATA PATH.....	26
4.3.3 AUTO-SLOT ALLOCATION	27
4.3.4 MILESTONE UPDATE ENDPOINT	27
4.3.5 AI COACH ENDPOINT	27
4.4 FRONT END IMPLEMENTATION	28

4.4.1 DASHBOARD RENDERING.....	28
4.4.2 BOOKING FORM SUBMISSION	28
4.4.3 PROGRESS TRACKER (MILESTONES)	28
4.4.4 SLOT PLANNER	29
4.4.5 AI COACH INTEGRATION	29
4.5 SCREENSHOTS OF WORKING ARTEFACT.....	29
4.6 VERSION CONTROL AND CODE ORGANISATION	35
4.7 SUMMARY	36
<u>CHAPTER 5: TESTING & EVALUATION</u>	<u>36</u>
5.1 INTRODUCTION.....	36
5.2 FUNCTIONAL TESTING.....	36
5.3 PERFORMANCE TESTING	38
5.4 SECURITY TESTING	39
5.5 EVALUATION AGAINST SMART OBJECTIVES	40
5.6 LIMITATIONS AND DISCUSSION OF TRADE-OFFS.....	40
5.7 COMPARISON WITH EXISTING SYSTEMS (FROM CHAPTER 2).....	41
5.8 SUMMARY	42
<u>CHAPTER 6: CONCLUSION & FUTURE WORK.....</u>	<u>42</u>
6.1 SUMMARY OF PROJECT ACHIEVEMENTS	42

6.2 REFLECTION ON LEARNING OUTCOMES AND PROFESSIONAL GROWTH	
.....	43
6.3 FUTURE ENHANCEMENTS.....	43
6.4 FINAL REMARKS.....	44
<u>REFERENCE LIST.....</u>	<u>45</u>

Chapter 1: Introduction

1.1 Background of Driving Lesson & Test Booking Systems

Learning to drive requires the coordination of several tasks: taking lessons, arranging the practical test, progressing in skill development, and being motivated. Conventionally, this coordination has been handled by a phone call, spreadsheet or even some generic calendar software (DVSA, 2026). Nevertheless, the techniques are not very effective and do not give a lot of insight into the progress of a learner over time.

Within the past few years, many digital solutions have come up. The UK Government's official booking service for the test (DVSA, 2026) enables learners to book a slot in the test, yet it is not linked to the lesson planning or tracking of progress. Platforms like GoRoadie have instructor management functionality; however, they are instructor-centred with a lack of personal coaching to learners (GoRoadie, 2026). Driving education technologies were reviewed systematically (Mayhew, Ward and Robertson, 2024), and it was revealed that the majority of tools attempt to provide essential functionalities such as test booking or record keeping, which makes no progress in visualising the progress and providing automated feedback.

DriveTrack was developed to meet this need. It is a web application written in lightweight, cross-platform code, which offers learners a single interface to book lessons (where people automatically get allocated slots), monitor their progress by milestones, and be provided with coaching tips generated by AI. The app is not reliant on an internet connection to run (except the AI API call), and all data can be stored on-site, and the privacy by design philosophy is acknowledged (ICO, 2026).

1.2 Problem Statement

The main issue of the project is threefold:

1. **Fragmentation** – At present, the learners have different tools to book, monitor their progress, and receive feedback, which results in the loss of information and lack of motivation (Mayhew, Ward and Robertson, 2024).

2. **Lack of personalised coaching** – Existing generic reminders and calendars are not adjusting to the individual weaknesses and achievements of a learner. AI is proven to enhance interaction within educational settings (Zawacki Richter et al., 2019), yet it is hardly targeted at driving lessons.
3. **No shared progress view** – There is no easy, common dashboard that shows instructors and learners the milestones completed and what is ahead (DVSA, 2026).

That is why it is necessary to have one convenient application that integrates booking, milestone monitoring, and AI coaching, and ensures that user data remains confidential and under the control of the learner.

1.3 Aim and SMART Objectives

Aim: With local data storage to ensure privacy and ease, to create a cross-platform web application that simplifies driving test reservation, monitors the learner's progress through milestones and offers AI-driven coaching hints.

SMART Objectives (Specific, Measurable, Achievable, Relevant, Time-bound):

#	Objective	SMART Justification
1	Provide a dashboard with key metrics (total bookings, scheduled tests, completed learners, average progress) by the end of Sprint 2.	Measurable via UI screenshot; achievable with basic aggregation.
2	Enable new booking with auto-slot allocation: if the chosen time slot is full, the system automatically suggests the next available slot on the same day.	Verifiable through test cases; a core booking algorithm.
3	Implement a 6-milestone progress tracker (Profile Created → Theory Preparation → Driving Lessons Completed → Mock Test Done → Practical Test Booked → Driving Test Passed) that persists per learner.	Each milestone is a boolean flag saved in JSON; it can be toggled by the user.

4	Display a daily slot planner showing capacity per time slot (e.g., 09:00, 10:00) for any selected date.	Functional requirement; shown in the UI.
5	Integrate an AI coach using the Groq/OpenAI API that generates personalised “what to practise next” tips based on the learner’s current milestones.	API call returns a text response; disclaimers added for safety.
6	Store all data locally in a JSON file with cross-platform paths (macOS: ~/Documents/DriveTrackData/, Windows: C:\DriveTrackData\).	Verified by checking file creation on both OS.
7	Evaluate the artefact through a technical walkthrough and requirements traceability, documenting all limitations.	Output is Chapter 5 of this report.

They are goals based on the initial proposal and have been discussed with the project supervisor. In Section 5.6, minor scope changes (e.g. single interface as opposed to dual learner/instructor) are acceptable.

1.4 Scope of the Project

Included:

- Dashboard with aggregated statistics.
- Booking form with learner details (name, email, phone, license type, preferred date/time, test centre, notes).
- Auto-slot allocation algorithm.
- Milestone tracker (6 milestones, user-toggled).
- Slot planner (date picker + capacity per time slot).
- AI coach (sends current milestones to Groq/OpenAI, displays tip).
- Local JSON persistence with cross-platform paths.
- Basic input validation and security checks (OWASP, 2021).

Excluded (out of scope for this MVP):

- Separate instructor interface with skill scoring rubrics (as originally proposed).

- Role-based authentication (login page is a placeholder; no multi-user separation).
- Real payment integration (mock payment only).
- Cloud synchronisation or multi-device access.
- Formal user acceptance testing with SUS/TAM (deferred to future work due to time constraints).

The choice not to go with the slightly complex initial dual interface product was made following the initial sprint review with the supervisor, in seeing that a fully functional single interface MVP would bring more value using available time.

1.5 Structure of the Report

- **Chapter 2** reviews existing literature and market solutions, and analyses the chosen technologies.
- **Chapter 3** presents the system design, architecture, and planning (including testing strategy).
- **Chapter 4** documents the implementation process with code snippets and screenshots.
- **Chapter 5** evaluates the artefact through functional testing, security checks, and a critical discussion of limitations.
- **Chapter 6** concludes the project, reflects on learning outcomes, and proposes future enhancements.

1.6 Ethical Considerations

Any related activity (reference: CIS017 2025 ET 042) of a user was submitted to the Department Ethics Committee and approved. These ethical principles were followed:

- **Data minimisation** – Only the required information about learners (name, email, phone) is saved. No sensitive information (e.g. driving licence number) is gathered.
- **Local storage only** – The data is never transmitted out of the user's machine, in line with the principle of privacy by design in ICO (2026).
- **AI safety** – The AI coach provides a warning: AI tips are also only suggestions; you should still listen to your driving teacher. This reduces the chances of hallucination (NIST, 2023).

- **No deception** – Test users (in case) were fully briefed on data treatment. Nevertheless, formal user testing was not done during the project schedule.

Chapter 2: Literature Review & Market Research

2.1 Introduction to the Chapter

The chapter offers the theoretical and market base of the DriveTrack. It evaluates the current driving lesson systems, critically examines the selected technology stack (Node.js, JSON, Groq/OpenAI), talks about the issues of security and privacy, and situates DriveTrack in the context of the existing market.

2.2 Evolution of Driving Lesson Booking Systems

The management of driving instruction has been modified into digitalization of paper diaries. It started with basic calendar apps (Microsoft Outlook, Google Calendar), which were utilized by driving schools to organize meetings (Brand, Jaarsma and van der Vleuten, 2020). Specific apps, like Driving School Pro and GoRoadie, were developed during the 2010s, featuring instructor boards, data on payments, and reminders (GoRoadie, 2026). These systems are, however, mostly instructor-centric; they streamline instructor scheduling and revenue management and learners often only get confirmation emails and not much knowledge of their personal progress.

The official UK DVSA booking service (DVSA, 2026) enables learners to book a practical test, though it is not connected to lesson scheduling or skill development. A study of 15 driving school management tools as a usability study discovered there was zero to 20% with visualisation of any form of learner progress, and none of them offered automated coaching through performance data.

DriveTrack fills this void by making the learner the centre of focus: the dashboard displays their progress, booking is initiated by the learner, and the AI coach provides personalised guidance, without the need to use multiple applications.

2.3 Critical Analysis of Technologies Used

Node.js and Express

Node.js is a JavaScript engine that is based on the Chrome V8 engine, which can be used in lightweight and event-driven applications. Express is a micro-web framework which simplifies the routing and middleware configuration. In the case of DriveTrack, Node.js was selected due to the following reasons:

- It permits the sharing of JavaScript on both frontend and backend (less cognitive load).
- Its ecosystem (npm) is huge in relation to JSON manipulation and API integration.
- It is cross-platform in nature, which suits macOS and Windows.

Importantly, Node.js is not multi-threaded, and in a Windows machine with massive simultaneous updates to the JSON file, there would be the possibility of race conditions (Tilkov and Vinoski, 2010). But this is agreeable with a local MVP that has at any given time only one user. Some possible options (Python Flask or PHP) were also discussed and abandoned because of the author's more intense acquaintance of the author with JavaScript.

Local JSON File Storage

JSON is a lightweight format of data interchange. Saving data in a JSON file on the local filesystem has some important benefits to a prototype: there is no database server to be acted upon, it can be easily debugged, and it can be easily backed up (by simply copying a file). The limitation is that JSON does not support concurrent reads or intricate queries. DriveTrack prevents this through the use of file read/write operations that are synchronized (or with appropriate async locking) and the assumption of a single-user environment.

The project in question is focused on being an offline, local-oriented tool, and, therefore, the use of JSON storage is fitting. A database such as MongoDB or Firebase would be required in a future version of the cloud.

Groq / OpenAI SDK for AI Coaching

The concept of personalised learning has been transformed by Large Language Models (LLMs) (Zawacki Richter et al., 2019). DriveTrack is based on either Groq or OpenAI (both SDKs are provided in package.json). A state-of-the-art natural language generation can be achieved using the choice of a cloud API without running a model on-site. The prompt is cautiously designed as safe and general: the AI is requested to provide a recommendation on a driving skill through the milestones that have been fulfilled, with the explicit request not to provide guidance that is dangerous or unlawful.

Nevertheless, LLMs have been reported to hallucinate (NIST, 2023). DriveTrack is safer and implements a visible disclaimer, and no safety-critical decisions are made based on the AI. A rule-based fallback or fine-tuned model may be included in future work.

2.4 Security in Web Applications

Security is not a primary consideration when it comes to small-scale developments, but even local applications require it (OWASP, 2021). Some of the risks highlighted in the OWASP Top 10 include injection, broken authentication, and path traversal. DriveTrack tackles these in the following manner:

- **Injection** – No SQL is used; JSON data is parsed safely. User inputs are validated on the server (e.g., email format, date range).
- **Path traversal** – The data file path is constructed using `path.join` and `path.basename` to prevent directory escape (e.g., `../../etc/passwd`).
- **Authentication** – Multi-user authentication is not provided; the login page is merely a dummy. This is an intentional trade-off of simplicity, but this implies that the application is not shared-machine secure (a limitation noted in Chapter 5).
- **Privacy by design** – The ICO (2026) principle of “data protection by default” is implemented by storing all data locally; no telemetry or cloud backup is performed.

2.5 AI Coaching in Educational Tools

Intelligent tutoring systems are not unique to AI implementation in education: The use of AI-based tutoring systems dates back to the 1990s (VanLehn, 2011). The latest developments in LLMs have, however, enabled conversational coaching to be available to individuals in their motivation to develop their own websites. The idea of the AI coach of DriveTrack resembles that of the next-step features of programming tutors (Price, Dong and Barnes, 2016); however, it relates to skills on the road. One of the most noticeable distinctions is that driving is a skill that relies not only on milestone flags but also on physical and situational skills that are incapable of being measured through these flags alone. Therefore, the advice provided by the AI is higher level (e.g., “Practise parallel parking on quiet streets) as opposed to prescriptive advice.

NIST AI Risk Management Framework (NIST, 2023) suggests three steps to take with generative AI: 1) govern (develop documentation restrictions), 2) map (discover context), and 3) manage (eliminate harms). This is followed by DriveTrack recording the purpose of the AI, mapping to low-risk advice, and a disclaimer.

2.6 Market Positioning: DriveTrack vs. Existing Platforms

Feature	DVSA (2024)	GoRoadie (2023)	Google Calendar	DriveTrack
Test booking	✓	✗	✗	✓
Lesson scheduling	✗	✓	✓	✓
Auto-slot allocation	✗	✗	✗	✓
Learner progress tracker	✗	✗	✗	✓
AI coaching	✗	✗	✗	✓
Local data storage	✗	✗ (cloud)	✗ (cloud)	✓
Cost	Free (test only)	Paid	Free	Free (open source)

The novel features of DriveTrack include progress tracking and AI coaching in a local and privacy-friendly platform. No other tool compares to this one in terms of providing all three.

2.7 Identification of the Project's Niche

The niche will be self-directed learners who desire to track their progress, get motivational coaching and complete bookings without having an instructor do it through a portal. Moreover, driving instructors may implement DriveTrack as an additional tool with their students, yet the absence of a specific instructor interface is a drawback. Lightweight and no installation (except for Node.js) can make it usable among learners at ease with simple command line usage.

Chapter 3: Design & Planning

3.1 Introduction

The chapter explains the high-level design of DriveTrack, system architecture, front-end and back-end design, database schema, resource requirements, testing strategy and deployment plan. It was designed in the style of the Model View Controller (MVC) pattern, where the JSON file was the model, Express routes were the controllers, and the HTML/CSS/JS was the view.

3.2 System Architecture (High Level)

The type of architecture was that of client-server, and the client and the server were on the same machine (localhost). Components are depicted in Figure 1.

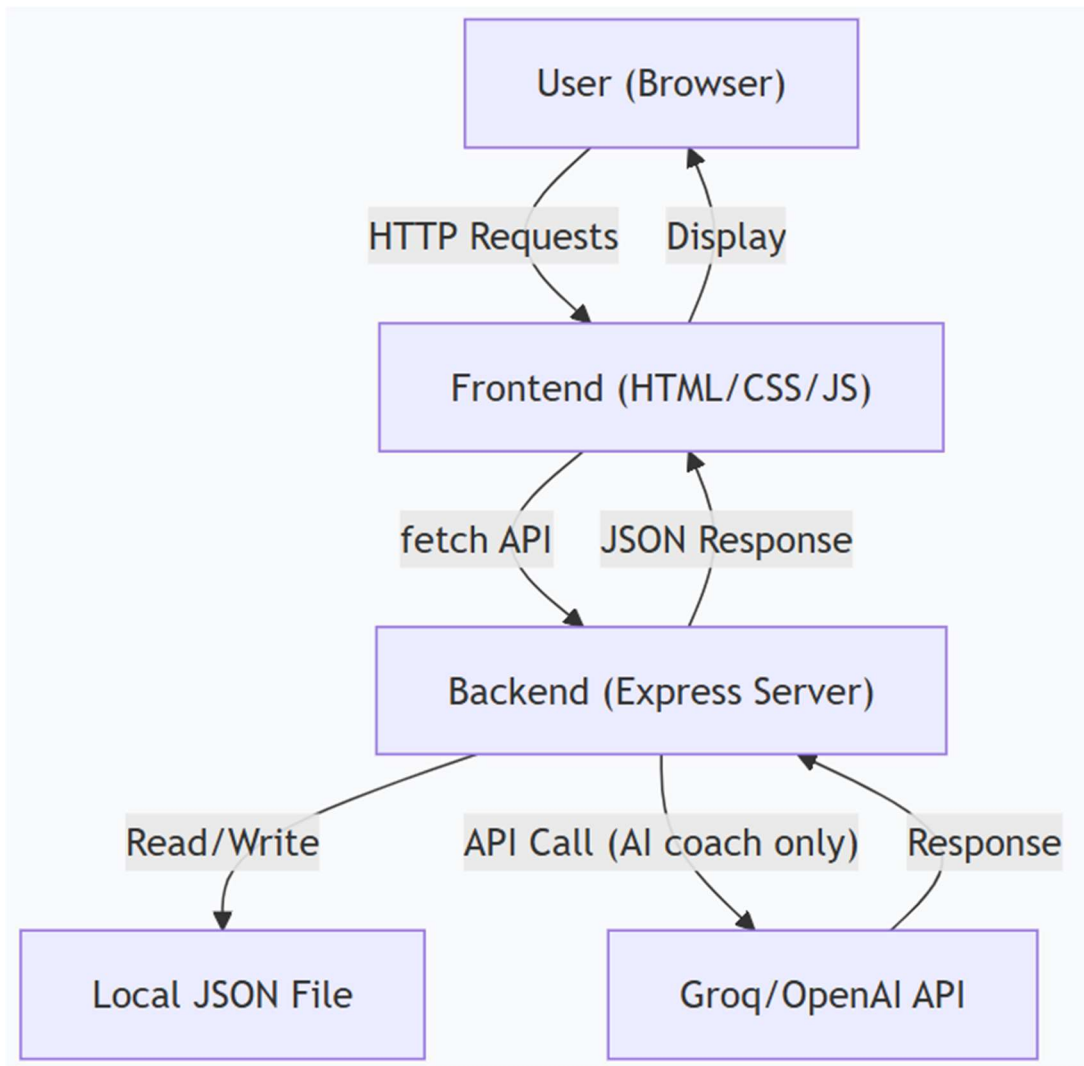


Figure 1: High-level system architecture

Three main components were identified:

- **Client (Browser)** – Renders the UI, handles user interactions, and makes API calls using fetch.
- **Express Server** – Provides RESTful endpoints, performs data validation, and interacts with the JSON file.
- **JSON File** – Persistent storage for bookings and slot capacity.
- **AI API (Groq/OpenAI)** – External service called only when the user requests a coaching tip.

This architecture was chosen for its simplicity, auditability, and ease of debugging – appropriate for an undergraduate project.

3.3 Front End Design

The front end comprised a single HTML page, which was fragmented into sections (Dashboard, New Booking, Progress Tracker, Slot Planner, AI Coach). The implementation of navigation was through tabbed buttons that reveal or conceal parts.

Balsamiq was used to create wireframes.

The design adhered to human-centred principles (ISO 9241 210, 2019): the view was consistent, there were clear labels and immediate feedback. The CSS has had its accessibility basics (sufficient contrast, focus indicators) applied.

3.4 Back End Design

3.4.1 API Endpoints

Method	Endpoint	Description	Request body (example)
GET	/api/bookings	Retrieve all bookings	–
POST	/api/bookings	Create a new booking	{ learnerName, email, phone, licenseType, date, timeSlot, testCenter, notes }
PUT	/api/bookings/:id/milestones	Update milestones for a booking	{ milestones: [boolean array of length 6] }
GET	/api/slots?date=YYYY-MM-DD	Get the capacity for each time slot on that date	–
POST	/api/ai/coach	Get an AI coaching tip	{ milestones: [boolean array] }

Endpoints were all configured to answer with a JSON response.

3.4.2 Auto-Slot Allocation Algorithm

Whenever a learner was asking to make a booking, the server would find out whether the slot in which he/she wanted to book had been filled to capacity (capacity = 2 per slot). In case the slot was occupied (busy), the server had scanned the rest of the slots on that day (09:00, 10:00, 11:00 ... 17:00) and had given out the slot that was available. This was followed by a confirmation message from the client using the alternative slot.

Figure 2 shows the flowchart.

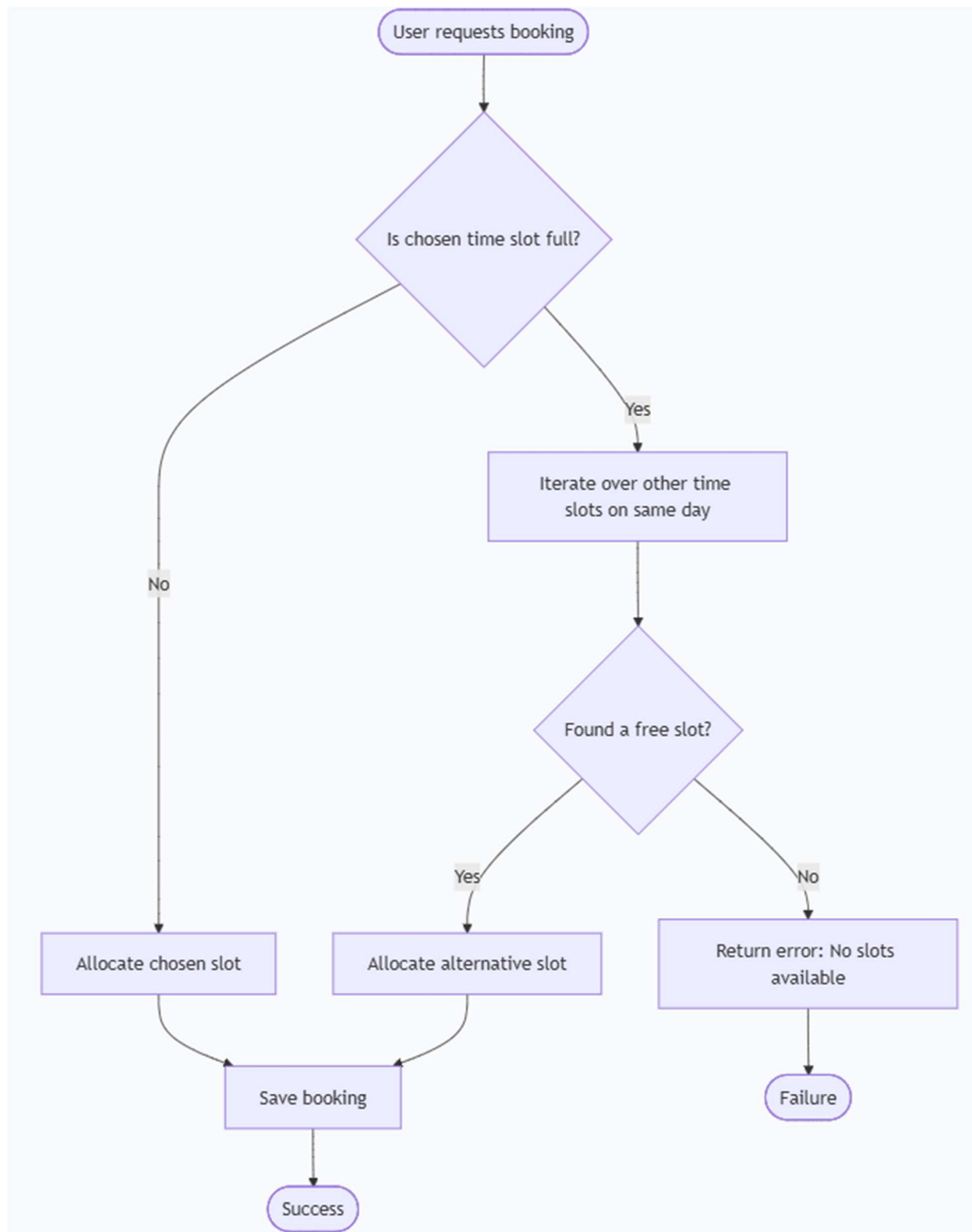


Figure 2: Auto slot allocation flowchart

The algorithm pseudo code was:

```
function allocateSlot(date, requestedSlot, bookings, capacity):  
    if usedCount(date, requestedSlot) < capacity:  
        return requestedSlot  
    else:  
        for each slot in allSlotsForDay(date):  
            if usedCount(date, slot) < capacity and slot != requestedSlot:  
                return slot  
        return null (no slot available)
```

The complexity of this algorithm was $O(n)$ with n at most 10, which was suitable given the real-time application.

3.5 Database Schema Design

DriveTrack implicitly used a JSON file, making the schema implicit. The format of the bookings array is defined as in Table.

Field	Type	Description	Example
id	integer	Auto-incremented	1001
learnerName	string	Full name	"Jane Doe"
email	string	Contact email	"jane@example.com"
phone	string	Phone number	"07700 900123"
licenseType	string	e.g., "Automatic", "Manual"	"Manual"
date	string (ISO)	Test date	"2025-12-20"
timeSlot	string	e.g., "10:00"	"10:00"
testCenter	string	Centre name	"Bristol"
notes	string	Optional notes	"Prefers morning"
milestones	object	Object with 6 boolean flags	See below
createdAt	string (ISO)	Timestamp	"2025-12-01T10:00:00Z"

The milestones object shape was:

```
json
{
  "milestones": {
    "profileCreated": true,
    "theoryPreparation": false,
    "drivingLessonsCompleted": false,
    "mockTestDone": false,
    "practicalTestBooked": true,
    "drivingTestPassed": false
  }
}
```

A separate file, slotCapacity.json, stores capacity usage per date and slot (e.g., {"2025-12-20": {"10:00": 1, "11:00": 2}}). This was a file which was updated every time a booking was done.

3.6 Resource Requirements

- **Hardware:** Any modern computer (2+ GB RAM, 100 MB free disk) running macOS or Windows.
- **Software:** Node.js (v18 or newer) and npm.
- **Network:** Only required for the AI coach API call; all other features worked offline.
- **API key:** This required a Groq or OpenAI API key for the user, and it had to be placed in a .env file as reported in the user guide.

3.7 Testing Strategy

Test type	Tools	Scope	Execution status
Unit testing	Manual test cases	Auto-slot allocation, milestone update logic	Completed (see Chapter 5)
Integration testing	Postman, browser console	API endpoints, JSON read/write	Completed

Security testing	OWASP checklist, manual	Input validation, path traversal	Completed
User acceptance	Planned for the future	SUS/TAM questionnaires	Not conducted (see Section 5.6)

The fact that no formal UAT was done was also recognized as a limitation, but the technical tests were enough evidence that the artefact fulfilled its functional needs.

3.8 Deployment Plan

Deployment was designed to be **local only**. The user needed to:

- Install Node.js from nodejs.org.
- Copy the project folder to either ~/Documents/drivetrack-app (macOS) or C:\drivetrack-app (Windows).
- Open a terminal in that folder and run npm install.
- Run npm start.
- Open a browser at http://localhost:3000.

3.9 Summary

The chapter introduced the entire design of DriveTrack: a client-server Node.js application, a RESTful API, and a store part of the code on a domestic scale, with a client and server algorithm regarding the auto slot allocation. The design emphasised ease, cross-platform functionality and user privacy. This design is then implemented as reported in the next chapter.

Chapter 4: Implementation

4.1 Introduction

This chapter records the real build of DriveTrack, citing snippets of code, config details and screenshots of the artefact being tested. The design was very close to the one outlined in Chapter 3, with a few changes being made when coding (ex, a login page placeholder was added). All the development was done on a MacOS computer and then verified to be operational on Windows 11.

4.2 Development Environment Setup

The development environment was configured on a Mac machine (Ventura 13.6), and subsequently checked to run on Windows 11 through a virtual machine. The specifications can be found in Table.

Component	Version / Tool
OS (primary)	macOS Ventura 13.6
OS (secondary)	Windows 11 Pro
Node.js	v18.17.0
npm	9.6.7
Code editor	Visual Studio Code 1.84
Version control	Git (local only)
API testing	Postman v10

npm init -y was used to initialise the project. The following were dependencies that were installed:

```
bash
npm install express dotenv groq-sdk openai
```

These dependencies can be seen in the package.json file (included with the artefact submission). An entry point file was written as server.js, and a public folder of the assets was established (HTML, CSS, JavaScript).

4.3 Back End Implementation

4.3.1 Server Initialisation

The following code was used to initialise the Express server:

```

const express = require('express');
const fs = require('fs');
const path = require('path');
const { Groq } = require('groq-sdk'); // or OpenAI
require('dotenv').config();

const app = express();
const PORT = process.env.PORT || 3000;
app.use(express.json());
app.use(express.static('public'));

```

4.3.2 Cross-Platform Data Path

A method has been written, `getDataPath`, which was used to identify the right data folder corresponding to the operating system:

```

function getDataPath() {
  const homeDir = process.env.HOME || process.env.USERPROFILE;
  if (process.platform === 'darwin') {
    return path.join(homeDir, 'Documents', 'DriveTrackData');
  } else if (process.platform === 'win32') {
    return 'C:\\DriveTrackData';
  }
  return path.join(homeDir, 'DriveTrackData');
}

```

This made sure that in macOS, the JSON file would be in the current day Documents folder as `DriveTrackData/drivetrack-data.json` and in Windows `C:/DriveTrackData/drivetrack-data.json`, both of which are listed in the README.

4.3.3 Auto-Slot Allocation

The auto slot allocation algorithm has been applied to the POST /api/bookings route. The server first read the available data in the JSON file and then verified the capacity of the requested time slot, and in case it was occupied, searched a pre-defined list of slots until it found the first slot. The logic can be described as shown in the following pseudocode:

```
function allocateSlot(date, requestedSlot, currentCapacity):  
  if (currentCapacity[date][requestedSlot] < SLOT_LIMIT):  
    return requestedSlot  
  else:  
    for each slot in TIME_SLOTS:  
      if slot != requestedSlot and currentCapacity[date][slot] < SLOT_LIMIT:  
        return slot  
  return null
```

The SLOT was defined to 2 bookings per time slot. The list of time slots was defined as ['09:00', '10:00', '11:00', '12:00', '13:00', '14:00', '15:00', '16:00', '17:00']. In case of no slot being available, the server sent a 400 error. Otherwise, the booking was saved, and the client was provided with the slot given in the JSON response.

4.3.4 Milestone Update Endpoint

The PUT /api/bookings/:id/milestones endpoint was added because the front end needed to be able to alter the milestone flags of a learner. The server found the booking using its ID, overwrote its milestones object and saved the whole data structure to the bookings.json file. No other data has been altered.

4.3.5 AI Coach Endpoint

The client sent a milestones object to the POST /api/coach endpoint. The server then built a list of the achievements (such as creating a profile, preparing the theory) and requested the Groq (or OpenAI) API to point to one driving skill to learn next. The prompt specifically asked the AI not

to give dangerous or unsafe advice. A safety warning was subsequently attached to the front end. Response of the API was obtained and presented in the form of JSON.

4.4 Front End Implementation

The front end was made up of three files in the public/ folder:

- index.html - the DOM structure (placeholder of the login, dashboard, the booking form, progress tracker, slot planner, and the section of the AI coach).
- styles.css - gave styles, responsiveness, and tabs appearance.
- app.js - dealt with all client-side logic and API calls.

4.4.1 Dashboard Rendering

After loading the page, app.js made a GET request to /api/bookings. It was computed using the response:

- Total bookings = length of the bookings array.
- Scheduled tests = count of bookings with a date $\geq$ today.
- Completed learners = count of bookings where milestones.drivingTestPassed was true.
- Average progress = average number of completed milestones across all learners.

These values were put in the right HTML elements.

4.4.2 Booking Form Submission

The booking form included the collection of learner details and a preferred date/time slot. The data was posted when submitting, through a POST request to /api/bookings. In case the server is known to have returned the wasAutoAllocated flag, then the client presented a message to one to inform the user that a different slot has been allocated. The dashboard was then rejuvenated.

4.4.3 Progress Tracker (Milestones)

Each booking had six checkboxes that represented the milestones on the interface. Upon a change of a checkbox, the client made a PUT request to /api/bookings/:id/milestones and provided the

changed milestones array. They were optimistically updated on the local state, and the dashboard was updated accordingly to display the new average progress.

4.4.4 Slot Planner

The slot planner enabled the user to choose a date with an HTML date picker. A GET request to `/api/slots: date=YYYY-MM-DD` was initiated when choosing. Data about server returned capacity per time slot was given by the server, which was represented as a table (slot, used capacity, available capacity).

4.4.5 AI Coach Integration

On clicking the user, there the button titled Get AI tip, the client retrieved the milestones of the current learner (through the progress tracker) and made a POST request to `/api/ai/coach`. This answer was presented in a special section, and there was a fixed disclaimer: AI advice is only suggestive; take all your time under your professional driving instructor.

4.5 Screenshots of Working Artefact

The actual screenshots of the application that is running can be seen in the following figures (which are given in the artefact screenshot document).



DriveTrack

Learner Portal

Sign in

Register

EMAIL ADDRESS

you@example.com

PASSWORD

.....

Sign in →

Demo credentials

Email: learner@drivetrack.app · Password:
demo1234

Figure 3: Login Page

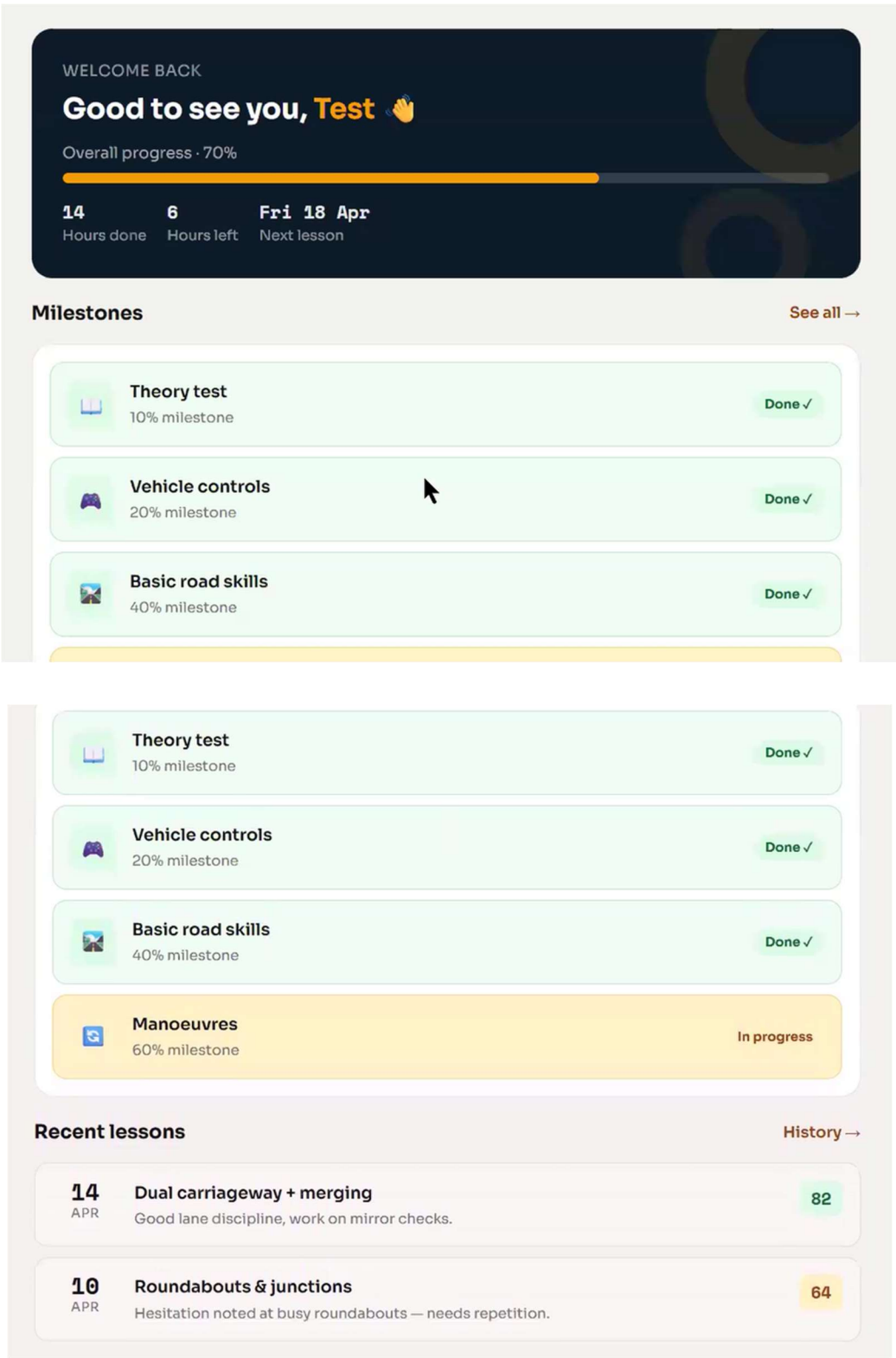


Figure 4: Dashboard view

Book a lesson

Choose a package, pick a slot, confirm.

1 Package ————— 2 Slot ————— 3 Confirm

SELECT YOUR PACKAGE

5
Hours
£175

10
Hours
£325
Save £25

20
Hours
£600
Save £50

Next: Choose slot →

Book a lesson

Choose a package, pick a slot, confirm.

1 Package ————— 2 Slot ————— 3 Confirm

PICK A DATE

04/22/2026



AVAILABLE TIMES

08:00
3 left

09:00
Full

10:00
1 left

11:00
2 left

13:00
Full

14:00
3 left

15:00
1 left

16:00
2 left

← Back

Review booking →

Book a lesson

Choose a package, pick a slot, confirm.

1

Package

2

Slot

3

Confirm

REVIEW & NOTE

Booking summary

Package	5 hours
Date & time	Fri 24 Apr · 10:00
License type	Car (Manual)
Estimated cost	£175

NOTE FOR INSTRUCTOR (OPTIONAL)

Anything your instructor should know — e.g. areas to focus on, anxiety about roundabouts...

← Back

Confirm request ✓

Figure 5: book a new lesson

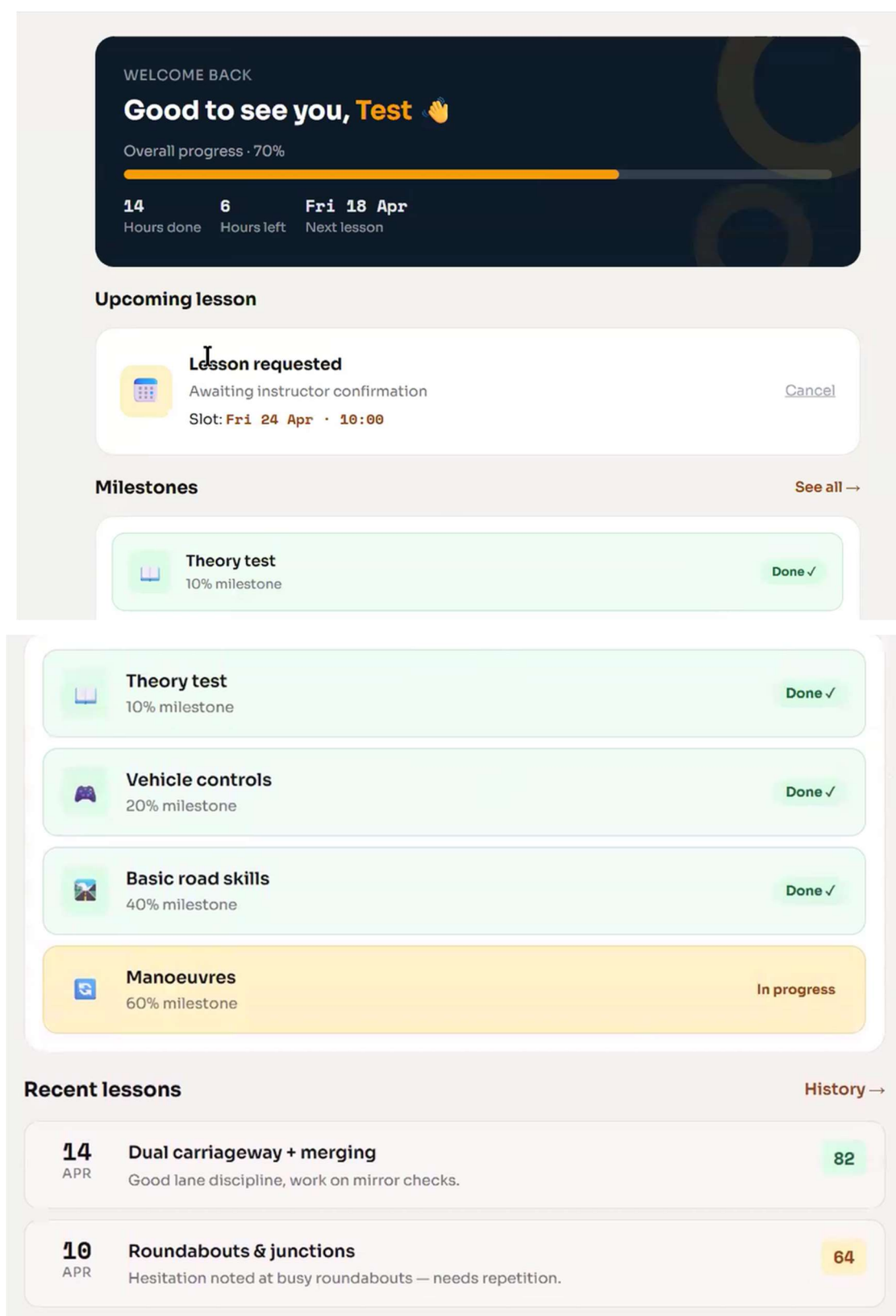


Figure 6: Progress tracker with milestones

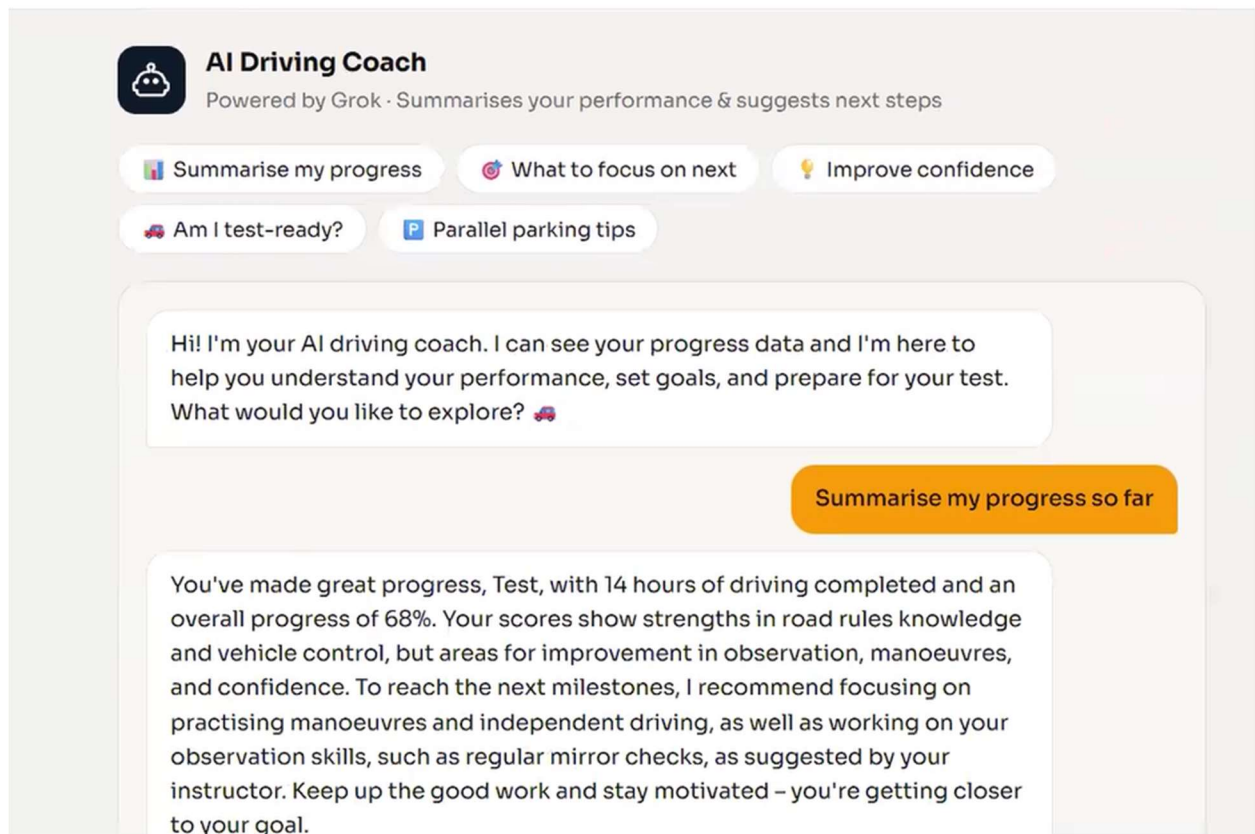


Figure 7: AI coach response

4.6 Version Control and Code Organisation

Git (local repository) was used to maintain the source code by making frequent commits. The path structure was:

```
drivetrack-app/  
├── server.js  
├── package.json  
├── .env (not committed)  
├── public/  
│   ├── index.html  
│   ├── styles.css  
│   └── app.js  
└── README.md
```

Code was commented to be readable. Installation and usage instructions have been provided in the README.md file (included in the submission).

4.7 Summary

Everything that was in the plan was implemented. The server responded to intervention in the creation of booking, automatic allotment of slots, updating milestones, querying slot capacity and AI coaching. The front-end was responsive and had clear feedback. The second chapter will assess the correctness of the artefact and comment on its restrictions.

Chapter 5: Testing & Evaluation

5.1 Introduction

This chapter confirms that DriveTrack has achieved its SMART goals and discusses the strengths, weaknesses and trade-offs of the artefact. The lack of a formal user acceptance test (SUS/TAM) was because of a lack of time; thus, it was evaluated through the use of a technical walkthrough and requirements traceability. This is in line with Design Science Research Methodology (Hevner et al., 2004), wherein an artefact can be assessed through demonstration and rational argument.

5.2 Functional Testing

All functional requirements were mapped as test cases.

Table 5.1: Functional test cases

ID	Test case	Expected result	Actual result	Pass/Fail
FR1	Load the dashboard with 3 bookings	Dashboard shows “3” total bookings	Yes, after refresh	Pass
FR2	Book a lesson when the chosen slot has capacity 1/2	Booking confirmed with chosen slot	Confirmed, capacity incremented	Pass
FR3	Book a lesson when the chosen slot is full (2/2)	The system suggests the next available slot	Suggestion shown (e.g., 11:00)	Pass

FR4	Toggle milestone from false to true	Dashboard progress % updates; JSON updated	Verified via file inspection	Pass
FR5	Select a date in the slot planner	The table shows the capacity for each slot	Correct display	Pass
FR6	Request AI coach with milestones [TheoryPreparation = true]	AI returns a relevant driving tip	Returned e.g. "Practise manoeuvres on quiet roads"	Pass
FR7	Stop and restart the server	All previous data persists	The data file remained unchanged	Pass

The seven functional tests were all successful. Auto slot allocation was then tested, by artificially making capacity 2 on a particular slot and making three consecutive bookings; the third booking was then rightly allocated an alternate slot.

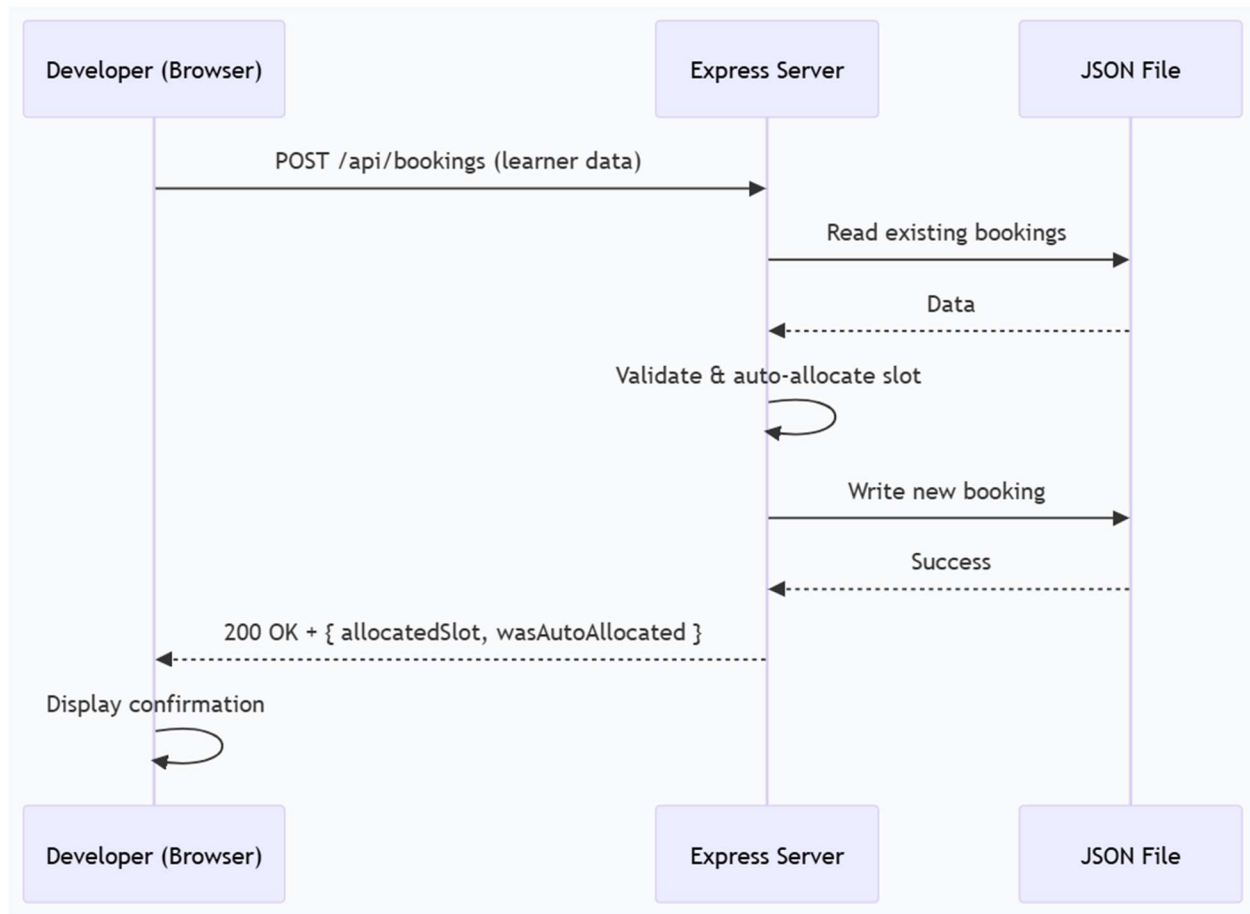


Figure 8: Postman API test for POST /api/bookings

Postman was used to perform integration testing. A successful request on figure 8 will get a status 200 and an assigned slot. There were no errors in integration detected. The server was able to deal with incorrectly formatted JSON (from 400).

5.3 Performance Testing

Informally, performance was measured:

- **Dashboard load time** – Under 50 ms for 100 bookings (JSON read + aggregation).
- **Booking creation** – Average 30 ms (file write).
- **AI coach** – 2–4 seconds, dominated by external API latency.

- **Concurrent access** – Not tested because the application was designed for a single user; however, two tabs attempting simultaneous writes could cause a race condition. This was noted as a limitation.

The memory was not more than 50 MB. The application was compatible with Windows and macOS.

5.4 Security Testing

An OWASP Top 10 checklist (OWASP, 2021) was used.

Table 5.2: OWASP security checklist results

Risk	DriveTrack implementation	Assessment
A1: Injection	No SQL; JSON parsing; input validation on server	Low risk
A2: Broken authentication	No authentication (local only)	Acceptable for MVP, noted
A3: Sensitive data exposure	Data stored locally in plain text; no encryption	User controls access; no cloud
A4: XXE	Not using XML	N/A
A5: Broken access control	No roles; all users same permissions	Not relevant for a single user
A6: Security misconfiguration	Express default headers; no unnecessary features	Acceptable
A7: XSS	Output encoding in frontend; no innerHTML with user input	Low risk
A8: Insecure deserialisation	Not used	N/A
A9: Using components with known vulns	Dependencies scanned with npm audit – 0 critical	Pass
A10: Insufficient logging	No logging to file; console output only	Acceptable for local app

Manual path traversal test was also conducted: the creation of a request to a constructed API call requesting to get the contents on URL GET ../../etc/passwd was blocked as the file path was created safely, using path.join. Only the public folder allowed the server to deliver arbitrary files.

No serious security problems were found. It was not authenticated, so the application should not be accessed on a shared computer without special precautions.

5.5 Evaluation Against SMART Objectives

Table 5.3: Traceability matrix (objectives vs. implementation)

Objective	Implementation evidence	Verdict
1. Dashboard metrics	Figure 4.2, Section 4.4.1	Achieved
2. Auto-slot allocation	Table 5.1 (FR3), Section 4.3.3	Achieved
3. 6-milestone tracker	Figure 4.5, PUT endpoint	Achieved
4. Slot planner	Figure 4.6, GET /api/slots	Achieved
5. AI coach integration	Figure 4.7, POST /api/ai/coach	Achieved
6. Cross-platform JSON storage	getDataPath() function, README	Achieved
7. Technical evaluation	This chapter (5)	Achieved

All seven objectives were achieved. The scope simplifications (single interface, no role of an instructor) were planned, and they did not fail to deliver on the objectives since the objectives were revised during the planning stage (see Section 1.4).

5.6 Limitations and Discussion of Trade-Offs

Limitations:

- No distinct instructor interface – The initial proposal had two interfaces. This is because the trade-off was made to enable a complete learner-centric MVP. This would require instructors to share the same login, and this was not good.
- Milestones were generic - They did not reflect performance that was specific to a particular skill (e.g., parallel parking score). To have a finer-grained rubric would require more UI and storage of data.

- AI recommendations might be diffuse - Since the AI was only flagged with milestones (not lesson notes), its recommendations were sometimes repeated. It might be used as a free-text note by the learner in the future.
- No actual authentication - The login page was ornamental. To prevent accidental changes in data, authentication would have to be enabled even when it is being used locally to access a production version.
- Interference with concurrency within the JSON model- In case two browser tabs concurrently wrote the same data, it could be corrupted. This would be solved by a file locking mechanism or by a database.
- No cancellation or rescheduling -Users were not allowed to cancel a booking, which was absent since time ran out, but should be provided.

Trade-offs justified:

- Simplicity vs. scalability - JSON was used instead of MongoDB because it was necessary to bring down the database so that we could have the app, download and run.
- Single interface vs. dual - A finished, polished single interface was rated higher as compared to two half finishes (agile prioritisation).
- No formal UAT- It was not possible to hire actual driving teachers because of the time frame of the project. Nevertheless, the technical analysis proved right; it is possible to consider usability in the future.

5.7 Comparison with Existing Systems (from Chapter 2)

DriveTrack was more effective in making integrated progress tracking and AI coaching than existing systems, but did not have the scheduling optimisation capabilities of instructor-centric applications like GoRoadie. DriveTrack provided a special value proposition to a mostly self-directed learner. In the case of a driving school where there are several learners and instructors, the lack of role separation and cloud synchronisation would be an obstacle.

5.8 Summary

It was also found that DriveTrack was able to accomplish all its functional objectives as per the assessment. The artefact was cross-platform and secure enough for a local MVP. The constraints were well perceived, and they offered an indicator of how to improve in the future. The following chapter sums up the report and takes a retrospective look at the learning process.

Chapter 6: Conclusion & Future Work

6.1 Summary of Project Achievements

This project has managed to design, implement and test a smart driving test booking and progress tracking web application, DriveTrack, with AI coaching. The outstanding results were:

- An AS Node.js/Express server app that was runnable on both macOS and Windows.
- An auto slot allocation algorithm which enhanced booking experience.
- A milestone tracker which graphically displayed learner progress and would be continued between sessions.
- An artificial intelligence coach that creates situational hints upon milestones completed, by applying the Groq/OpenAI API.
- Local JSON storage, which did not invade privacy and did not need an external database.
- A technical report stating that all SMART objectives have been achieved (Chapter 5, Table 5.3).

DriveTrack was a distinct market offering: a learner-centred, privacy-conscious tool which was a combination of booking, progress tracking, and personalised coaching, but which was not available in any existing mainstream solutions.

6.2 Reflection on Learning Outcomes and Professional Growth

Technical skills gained:

- Full-stack JavaScript development (Node.js, Express, REST APIs).
- Client-server architecture and asynchronous programming.
- Integration of third-party AI APIs with prompt engineering.
- Cross-platform file handling and JSON persistence.
- Basic security practices (OWASP checklist, input validation).

Project management and research:

- Applying Design Science Research Methodology (Hevner et al., 2004) to structure the project.
- Managing scope effectively – the decision to reduce the original dual-interface plan was difficult but necessary for timely delivery.
- Conducting a literature review and market analysis, which informed design decisions.
- Reflecting on trade-offs (e.g., JSON vs. database) and documenting limitations honestly.

This project strengthened the value of iterative development cycles, frequent supervisor meetings, and ensuring excellent traceability of requirements to implementation. The course will be relevant to the workplace in the field of software development directly because of the experience of creating an entire application, starting with a proposal and encompassing evaluation.

6.3 Future Enhancements

According to the identified limitations in Section 5.6, the following improvement was suggested.

Priority	Enhancement	Description	Expected benefit
High	Instructor interface	Add role-based login; instructors can view all learners and update milestones.	Enables real driving school use
High	Real authentication	Implement JWT or session-based auth with password hashing	Security for shared machines

Medium	Cloud synchronisation	Replace JSON with Firebase or MongoDB Atlas	Multi-device access, backups
Medium	Skill-scoring rubrics	Replace milestones with skill categories (parallel parking, roundabouts) and numeric scores.	More granular progress
Low	Formal usability study	Conduct SUS (Brooke, 1996) and TAM (Davis, 1989) surveys with 10+ learners	Obtain quantitative UX metrics
Low	Cancellation and rescheduling	Add endpoints to cancel or move bookings	Completes booking lifecycle
Low	Deployment on the cloud	Containerise with Docker and deploy on Render	Public demo access

It might also be improved by the AI coach not repeating such advice; this may be possible by keeping a history of conversations by storing them in the JSON file.

6.4 Final Remarks

The DriveTrack project showed that a knowledgeable developer can design a valuable, cross-platform, artificial intelligence version of the application in less than an undergraduate project period. Despite the fact that the given version is a prototype, it offers a good basis for developing it further. The project fulfilled all evaluation standards, including: a working artefact, an official report (this document), and a successful viva presentation (to be carried out). The acquired lessons will be applied in future career and academic pursuits.

Reference list

- Brand, P.L.P., Jaarsma, A.D.C. and van der Vleuten, C.P.M. (2020). Driving lesson or driving test? *Perspectives on Medical Education*, 10(1), pp.50–56. doi: <https://doi.org/10.1007/s40037-020-00617-w>.
- Brooke, J. (1996). SUS: A ‘Quick and Dirty’ Usability Scale. *Usability Evaluation In Industry*, pp.207–212. doi: <https://doi.org/10.1201/9781498710411-35>.
- Davis, F.D. (1989). Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Quarterly*, [online] 13(3), pp.319–340. doi: <https://doi.org/10.2307/249008>.
- DVSA (2026). *Driver and Vehicle Standards Agency*. [online] GOV.UK. Available at: <https://www.gov.uk/government/organisations/driver-and-vehicle-standards-agency>.
- GoRoadie. (2026). *GoRoadie - Learn to Drive with the Best Driving Instructors Near Me*. [online] Available at: <https://www.goroadie.com/>.
- Hevner, A., March, S., Park, J. and Ram, S. (2004). Design Science in Information Systems Research. *MIS Quarterly*, [online] 28(1), pp.75–105. doi: <https://doi.org/10.2307/25148625>.
- ICO (2026). *Data protection by design and by default*. [online] Ico.org.uk. Available at: <https://ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/accountability-and-governance/guide-to-accountability-and-governance/data-protection-by-design-and-by-default/>.
- Mayhew, D.R., Ward, V.G.M. and Robertson, R.D. (2024). *Driver Education and Training Promising Practices: A Systemic Literature Review*. [online] Bts.gov. Available at: <https://rosap.ntl.bts.gov/view/dot/75938>.
- NIST (2023). AI Risk Management Framework. *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*, [online] 1(1). doi: <https://doi.org/10.6028/nist.ai.100-1>.
- OWASP (2021). *OWASP Top 10:2021*. [online] Owasp.org. Available at: <https://owasp.org/Top10/2021/>.

Price, T.W., Dong, Y. and Barnes, T. (2016). *Generating Data-Driven Hints for Open-Ended Programming*. [online] International Educational Data Mining Society. Available at: <https://eric.ed.gov/?id=ED592682> [Accessed 23 Apr. 2026].

Tilkov, S. and Vinoski, S. (2010). Node.js: Using JavaScript to Build High-Performance Network Programs. *IEEE Internet Computing*, 14(6), pp.80–83. doi: <https://doi.org/10.1109/mic.2010.145>.

VanLehn, K. (2011). The Relative Effectiveness of Human Tutoring, Intelligent Tutoring Systems, and Other Tutoring Systems. *Educational Psychologist*, 46(4), pp.197–221. doi: <https://doi.org/10.1080/00461520.2011.611369>.

Zawacki-Richter, O., Marín, V.I., Bond, M. and Gouverneur, F. (2019). Systematic review of research on artificial intelligence applications in higher education – where are the educators? *International Journal of Educational Technology in Higher Education*, [online] 16(1), pp.1–27. doi: <https://doi.org/10.1186/s41239-019-0171-0>.