# פרויקט יישומי בפיתוח אפליקציות final report This document summarizes the work done in my Car-Market project – a web platform for listing, browsing, and managing cars for sale, with user authentication, comments, private messages, and an admin dashboard. 1. Why I Chose This Topic

I imagined a clean platform where users could register, list their cars with images, and communicate with potential buyers directly. The project turned out close to my vision, with additional features such as comments on cars and an admin panel for managing users.

**2. Choice of Technologies**

I used VS code as my work environment, I found VS code very convenient for my project, connected it really easy to GitHub with Git and update through the terminal.  
I used Node.js for the backend because it is lightweight, fast, and worked great for my needs.   
MongoDB with Mongoose was chosen as the database to handle flexible car and user data.  
I considered to use SQL instead of the MongoDB but I found MongoDB better choice for my project for it's scalability and flexibility.  
Most of the functions in this project are create, eead, update, delete, and MongoDB is particularly convenient for such operations thanks to Mongoose, which provided a clean abstraction layer.  
Multer was integrated for file uploads, allowing users to add images of the cars they want to sale.  
The frontend was built with HTML, CSS, and JavaScript for simplicity and direct integration with the backend API.

**3. design and implementation**

The system was designed around three main models: User, Car, and Message.   
I planned the implementation in stages:   
First I wrote a shallow server.js file to connect to Node.js and MongoDB,  
then I wrote the files in the public folder which are simple frontend files to see that something is running.   
After that I started to improved the server file and created a user database with user authentication (register, login).  
car listings (adding, editing, deleting, with optional images), comments on cars, private messaging between users, and admin management.  
And my last 2 updates are related to passwords,  
One update is to require a strong password upon registration (At least 8 characters where at least one of the characters is a number, a capital letter and a symbol)  
and the other is to delete the admin password from the code.  
The backend exposes RESTful routes while the frontend interacts with them using fetch().