# 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, read, 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**

My project 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).  
After that, I improved the design with style.css file where I designed all the buttons that I have in my project as well as the login and registration forms. style.css helped me to make sure that all pages share the same clean and user-friendly design.  
Afterwards, I consistently improved the project by adding: car database (adding, editing, deleting, with image upload (optional)), comments on cars, private messaging between users, and admin management.  
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 one is to delete the admin password from the code.  
The backend exposes RESTful routes while the frontend interacts with them using fetch().  
I did tests after every small update.  
Every time I changed something, I immediately restarted the server to check if the update I made actually worked well and worked as I want.

**4.** **What I learned during the project**

One of the most important things I learned during this project was how to break down a large idea into small, manageable steps. At the beginning, the project felt overwhelming, but by dividing it into stages (server, user database, car database, comments, messages, etc.) I was able to make steady progress. I also learned how valuable it is to test after every small change, because it allowed me to catch mistakes early. Another lesson was the importance of good code organization and naming conventions – keeping my models, routes, and frontend files structured made the project easier to maintain as it grew.  
If I started again, I would pay more attention to writing documentation and comments while coding. During the project, I sometimes focused so much on making features work that I didn’t always stop to clearly document what each function or route was doing. Later, when I revisited parts of the code, it took extra time to remind myself how everything worked. If I had documented more consistently throughout the process, it would have been easier to maintain the code and make changes.

**5.** **ChatGPT**

To be completely honest, ChatGPT was a great help throughout this project. I worked with it continuously, and it guided me through many challenges I encountered.   
For example, it helped me implement the image upload feature with a preview (user can see how the image will be) before adding a car, and it also helped me to connect each uploaded car to the specific user who created it.   
It was also very helpful in building the chat functionality for each car in the web-site, as well as in setting up the initial MongoDB connection and configuring GitHub integration so I could update directly from the terminal instead of manually.   
ChatGPT saved me a lot of time: whenever I wanted to add something new to the project and I didn’t know how to implement it, I simply asked ChatGPT, and it taught me the subject I needed with clear, step-by-step explanations. I feel that ChatGPT helped me truly understand how the subjects I ask about works. Even after ChatGPT assisted me in writing the code of something, if I wanted to make changes later, I was able to do so entirely on my own.

**6.**  **if I had more time**

If I had more time, I would like to add advanced search and filtering of cars, allowing users to quickly find vehicles by manufacturer, model, year and price range.  
Also I would implement password encryption using bcrypt which storing hashing password instead of storing the password as is.

**7.**  **Web Security**

In terms of web security, I ensured that when a user uploads a file, it must be an image. If a user attempts to upload any file that is not in JPG, PNG, WEBP, or GIF format, the system rejects it and displays an error message.  
At the same time, I do not yet feel comfortable running my project as a publicly accessible website. There are several important steps that would need to be taken before I could safely do so.  
One key issue is NoSQL Injection. Right now, user input is passed directly into MongoDB queries, which means an attacker could potentially try to inject MongoDB operators into the request. To prevent this, inputs should be strictly validated as strings, sanitized before being used in queries, and wrapped with $eq to ensure they are interpreted only as values. Enabling features like mongoose.set('sanitizeFilter', true) would further reduce the risk.  
In addition, there are a few other things I would do to improve web security like:  
Using JWT (JSON Web Tokens) for authentication, so that user sessions are securely managed without relying only on cookies or session storage.  
Implementing HTTPS/TLS to ensure all data (including login credentials) is encrypted during transmission.  
Adding rate limiting and brute-force protection on login routes to stop automated attacks.  
With these changes, I would feel much more confident about deploying the site in a public environment.

**Conclusion**

To be honest, I truly enjoyed working on this project. It was very exciting and rewarding to create something entirely from scratch on my own. I believe the project represents a complete and well-rounded full-stack application: it includes user registration and authentication, car listings with images, comments, private messaging, and an admin dashboard for managing users.  
This combination gave me the experience of building a real-world web application that brings together the client, server, and database layers into a functioning system. Beyond the technical skills, the project was a valuable learning experience giving me confidence, problem-solving experience, and the satisfaction of seeing my ideas come to life as a working product.