PORTFOLIO

Andreea-Bianca Enache – enan22pt@student.ju.se

Web Development Fundamentals - Jönköping University - 2023

Contents

The project port is 2911.

Logging in as admin:
Username: Admin
Password: TrueAdmin

Logging in as a normal user:
Username: OtherUser
Password: OtherUser1234

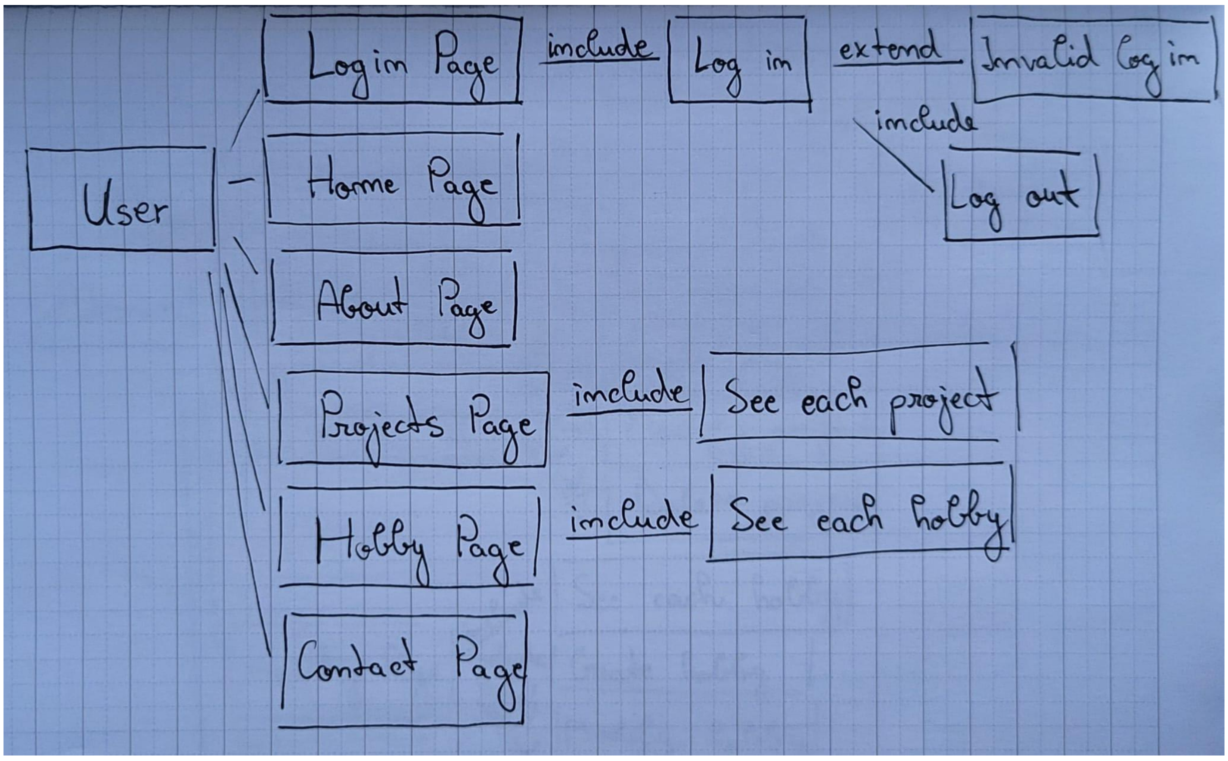
1) Introduction

Personal portfolio websites have become a popular way for professionals across many fields to showcase their work and skills online. For creative professions like designers, developers, and artists, an online portfolio is an essential tool to demonstrate talents and attract potential clients or employers (Owen & Watson, 2015).

This project implements a portfolio website focused on highlighting programming projects and creative hobbies. The goal is to provide a clean, engaging platform to display these works to the public The intended audience is of two kinds:

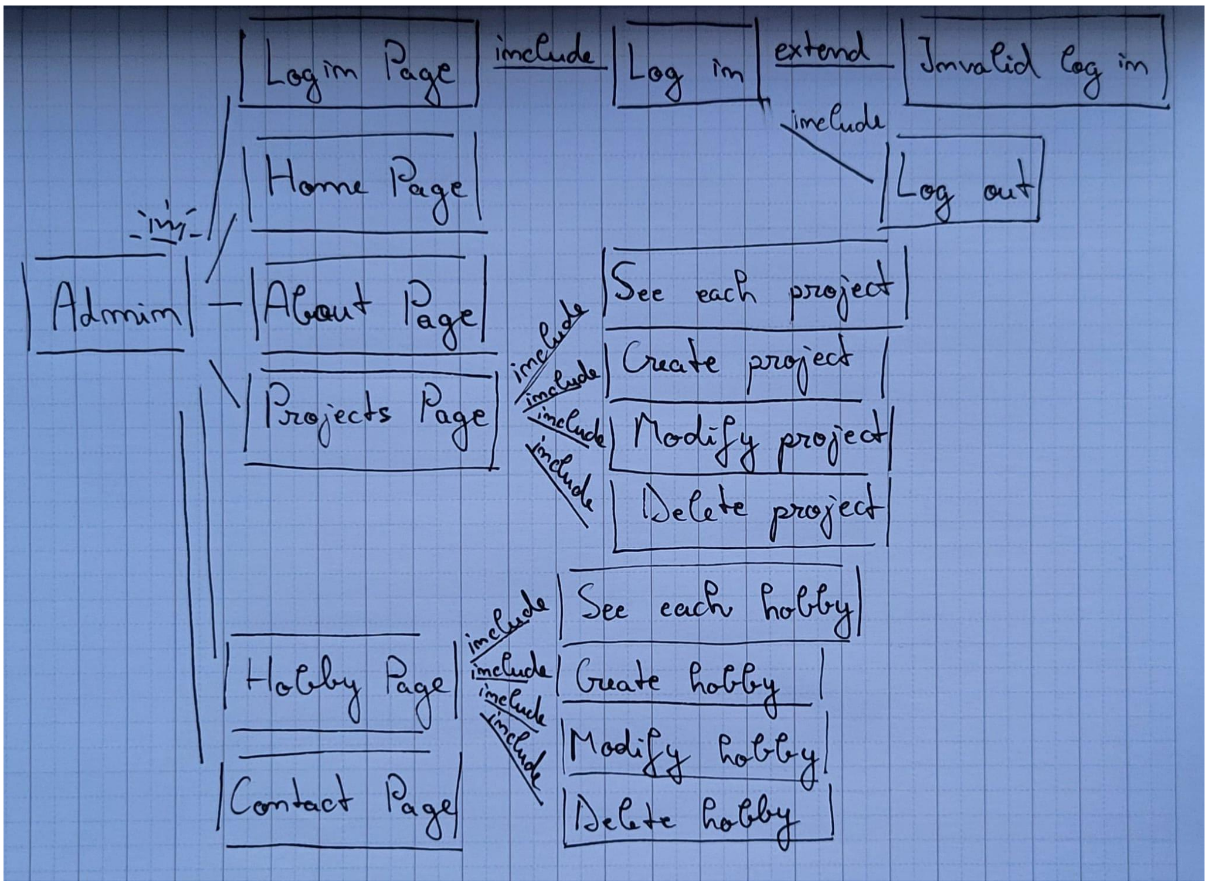
First, potential employers or clients can visit the site to evaluate the skills and talents of the portfolio owner. Well-presented projects and hobbies will convey competence and ability.

Second, fellow programmers can view the site as a sample implementation of a portfolio site. They can reference the code and setup as an example if they are building their own similar site.



UML Use-Case Diagram (for users)

The portfolio site allows for projects and hobbies to be easily added, modified, and removed. This gives admin users the ability to actively manage the content.



UML Use-Case Diagram (for admin)

The implementation uses Node.js for the server environment, Express as the web framework, Handlebars for the view templates, and SQLite for the database. This combination of technologies offers a robust, flexible foundation for a dynamic web application.

For end users, the site provides an aesthetic and usable interface to browse the portfolio content. Visitors can view profiles of each project and hobby, providing details through images, descriptions, and other metadata.

2) Method

2.1) Architecture

The portfolio website is built using a traditional web application architecture, with the key components being the frontend user interface, backend application logic, and database.

The major components are:

Client - This refers to the web browsers used by end users to access the site. The most common browsers, like Chrome, Firefox, Safari, etc., are supported.

Web Server - The Node.js Express application runs on the web server. It handles all request routing and response generation.

Database - A SQLite database stores all project and hobby content, along with users and skills. It contains tables for the users, skills, projects, and hobbies and their attributes.

Static Assets - CSS stylesheets, JavaScript files, images, and other static assets are served from the public folder.

Templates - Handlebars template files generate the HTML markup displayed in the browser.

The client's web browser sends requests to the web server. The server handles each route, querying data from the database if needed. It populates the template files with this data and returns rendered HTML.

Common web technologies like HTTP underpin the communication between client and server. The web server runs continuously, listening on a defined port number for any incoming requests.

The frontend and backend can also scale independently. If necessary, the database and server could run on separate machines or clusters to handle increased traffic.

Overall, this architecture ensures the portfolio site is performant, secure, maintainable, and extensible. Each component focuses on a single responsibility, following best practises.

2.2) Database

SQLite is a lightweight, self-contained database engine that simplifies local data storage for web applications (Bi, 2009). The portfolio site uses a SQLite database to store and manage all four tables - users, skills, projects, and hobby content.

Users

Column	Data Type	Description
id	INTEGER (primary key)	ID for each user
username	TEXT	User's username
password	TEXT	Hashed password for the user
hash	TEXT	Password hash
is_admin	INTEGER	1 if admin user, 0 if regular user

Skills

Column	Data Type	Description
sid	INTEGER (primary key)	ID for each skill
sname	TEXT	Skill name
sdesc	TEXT	Skill description

Projects

Column	Data Type	Description
pid	INTEGER (primary key)	ID for each project
pname	TEXT	Project name
pdesc	TEXT	Project description
ptype	TEXT	Project type (Other or University assignment)
pimgURL	TEXT	URL of the project's image

Hobbies

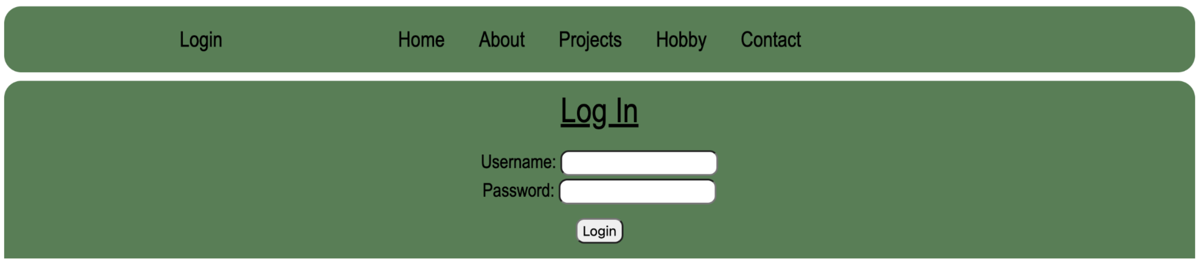
Column	Data Type	Description
hid	INTEGER (primary key)	ID for each hobby
hname	TEXT	Hobby name
hdesc	TEXT	Hobby description

htype	TEXT	Hobby type (Other or Drawing made by me)
himgURL	TEXT	URL of the hobby's image

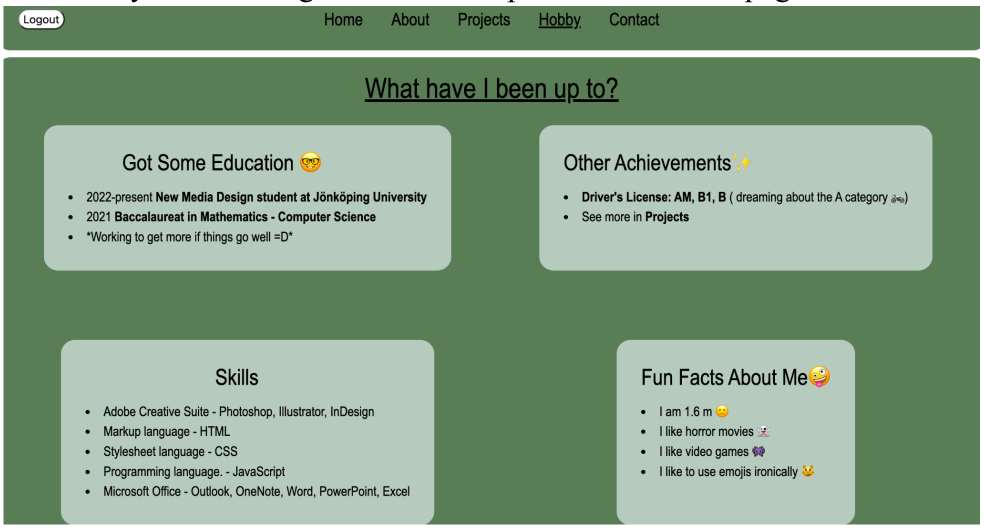
This structure makes it possible to organise all the important project and hobby details because these two tables form the main content of the portfolio. The TEXT columns allow storage of long form descriptions and other text content.
The users and skills tables store supplemental data for authentication, as does the about page.

2.3) Graphical User Interface

The portfolio website features a clean, modern GUI designed for aesthetics and usability.
A menu bar at the top provides navigation on every page. It contains links to the main sections: Home, About, Projects, Hobbies, and Contact. The Login link appears when you are not already logged in.



A muted, low-saturation palette sets the foundation for a refined style, with accent colours carefully sparingly highlighting key elements such as buttons and links. The use of open, friendly fonts in legible sizes ensures that body text remains approachable. Meanwhile, more compact spacing and varying font weights are employed to clearly denote headers and labels. Strategic implementation of white space and meticulous alignment work harmoniously to guide the eye seamlessly through each page, making sure that related elements are visually grouped (Watzman & Re, 2009). Icons are thoughtfully used to signal interactions, maintaining an uncluttered and airy design throughout. This user-centered approach ensures that users can easily scan and digest the content presented on each page.



What have I been up to?

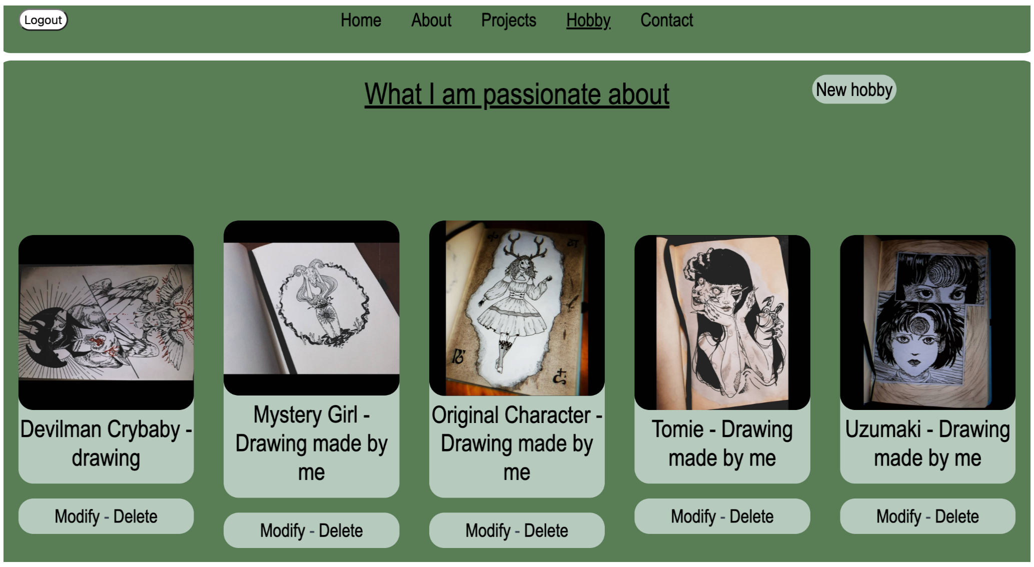
Got Some Education 🤓
- 2022-present **New Media Design student at Jönköping University**
- 2021 **Baccalaureat in Mathematics - Computer Science**
- *Working to get more if things go well =D*

Other Achievements ✨
- **Driver's License: AM, B1, B** (dreaming about the A category 🏍️)
- See more in **Projects**

Skills
- Adobe Creative Suite - Photoshop, Illustrator, InDesign
- Markup language - HTML
- Stylesheet language - CSS
- Programming language. - JavaScript
- Microsoft Office - Outlook, OneNote, Word, PowerPoint, Excel

Fun Facts About Me🤪
- I am 1.6 m 🙁
- I like horror movies 👻
- I like video games 👾
- I like to use emojis ironically 😼

Project and hobby gallery pages are designed with large preview images that serve as enticing click-through points, encouraging users to explore more details. To reinforce the sleek aesthetics, filtered image effects are applied, creating an immersive visual experience. Detail pages maintain this aesthetic consistency by featuring prominent media that further engages users with the content. Throughout the entire website, common elements, such as the navbar, colour scheme, and typography, remain constant. This deliberate uniformity enhances branding while allowing flexibility for page-specific layouts, resulting in a seamless and cohesive identity that connects the overall user experience. In terms of accessibility, alt text descriptions are provided for all imagery, ensuring inclusivity for all users. A strong semantic structure and legible fonts are also employed to maximise the website's accessibility. As a result, users can comfortably navigate and consume content without any barriers.


Logout
Home
About
Projects
Hobby
Contact
What I am passionate about
New hobby
Devilman Crybaby - drawing
Modify - Delete
Mystery Girl - Drawing made by me
Modify - Delete
Original Character - Drawing made by me
Modify - Delete
Tomie - Drawing made by me
Modify - Delete
Uzumaki - Drawing made by me
Modify - Delete

Through these strategic design choices, the GUI establishes a refined yet inviting aesthetic. More importantly, purposeful organisation and UX decisions minimise cognitive load. Users can easily access, absorb, and enjoy the portfolio contents.

2.4) Web Application

The portfolio site is implemented as a Node.js web application using the Express framework. Some key aspects are:

app.js - This central script starts the app, configures Express middleware, sets up routing, and defines database interactions.

```
// Required constants
const express = require("express"); // Loads the express package
const session = require("express-session"); // Loads the express-session
package for managing sessions
const app = express(); // Creates an instance of the Express application
// Add the express-session middleware
app.use(
session({
```

```
store: new SQLiteStore({ db: "session-db.db" }),
secret: "mamaligacucarnatisicucastravetimurati", // Secret key
resave: false,
saveUninitialized: true,}));
```

MVC Pattern - The code follows the Model-View-Controller pattern to separate concerns.

- Models define the SQLite database schemas. Separate controller files contain handler functions for each route. They get or modify data from the models and populate the views.

```
// Handles the GET request to send the form for modifying a project.
app.get("/projects/update/:id", (req, res) => {
const id = req.params.id;
db.get(
"SELECT * FROM projects WHERE pid=?",
[id],
function (error, theProject) {
(I am showing only the more important part of the code.)
});
```

- Views are Handlebars templates that render HTML. Handlebars templates generate the HTML sent to the client based on the data provided by controllers.

```
{{#each projects}}
<div>
<div class="projects-container">
<div><a href="/projects/{{pid}}"><img
src="{{pimgURL}}"
alt="{{pimgURL}}"
/></a></div>
<h2><a href="/projects/{{pid}}">{{pname}} - {{ptype}}</a></h2>
</div>
{{/each}}
```

- Controllers handle routing logic. Express routing handlers defined in app.js respond to HTTP requests. This includes middleware to parse request bodies and handle sessions.

```
// Route to handle login form submission
app.post("/login", (req, res) => {
// Lookup user in database
// Compare password
// If valid:
// Set session data
req.session.user = user;
```

Security - User passwords are hashed with bcrypt before storage. HTTPS and input validation provide further protection.

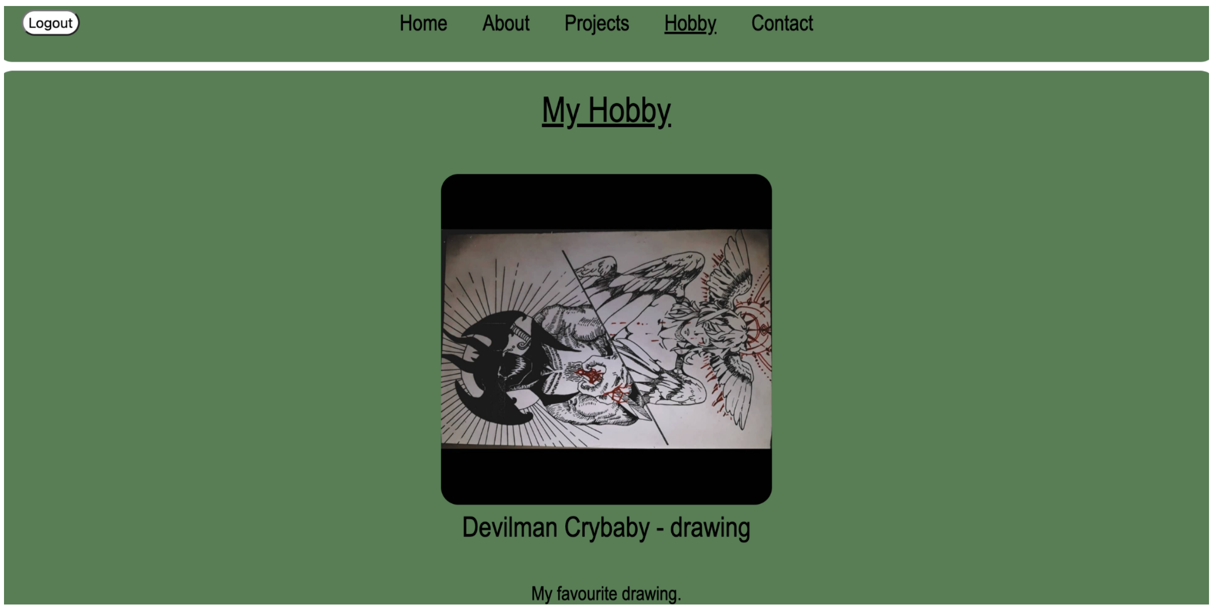
Sessions - Express sessions track logged in state. Session data is stored securely using connect-sqlite3.

Error Handling - Custom middleware catches errors and renders friendly public pages. Detailed errors are shown in development.

Public Assets - CSS, JavaScript, and images are served statically from the public folder.

3) Results

The completed portfolio website allows users to browse featured projects and hobbies, view details on each one, and contact the site owner.

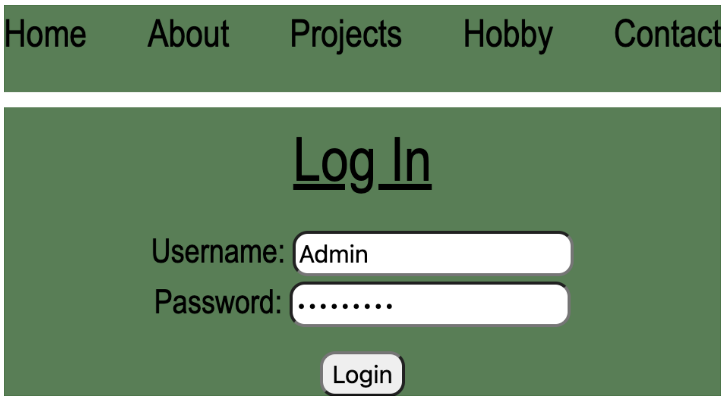


My Hobby

Devilman Crybaby - drawing

My favourite drawing.

User passwords are hashed with bcrypt before storage for protection. The site also uses HTTPS encryption and sanitises all inputs to prevent common attacks.

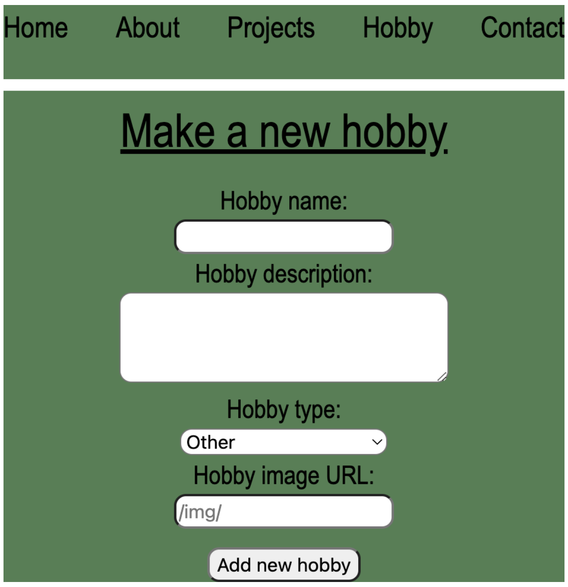


Log In

Username: Admin

Password: •••••••••

The login system provides the administrator access to special management pages. For administrators, it enables managing the content by adding, editing, and deleting projects and hobbies.



Make a new hobby

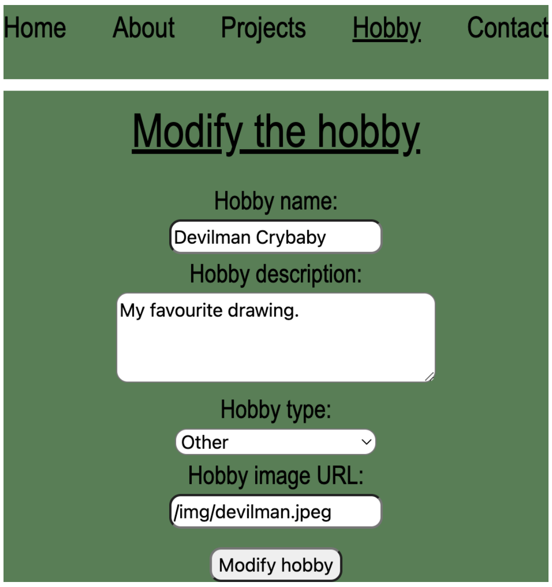
Hobby name:

Hobby description:

Hobby type: Other

Hobby image URL: /img/

Add new hobby



Modify the hobby

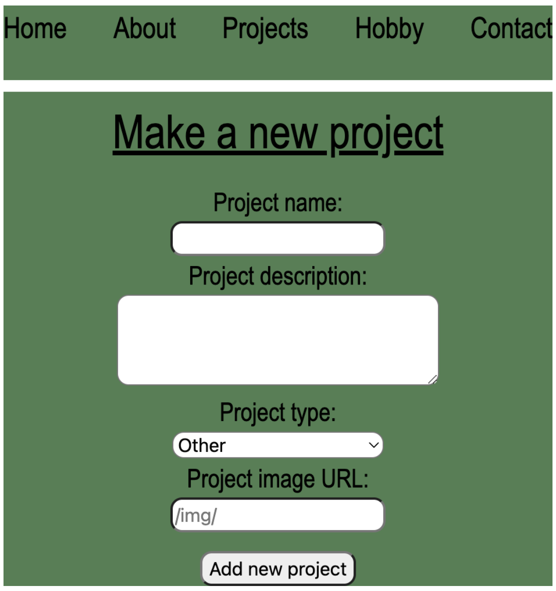
Hobby name: Devilman Crybaby

Hobby description: My favourite drawing.

Hobby type: Other

Hobby image URL: /img/devilman.jpeg

Modify hobby



Make a new project

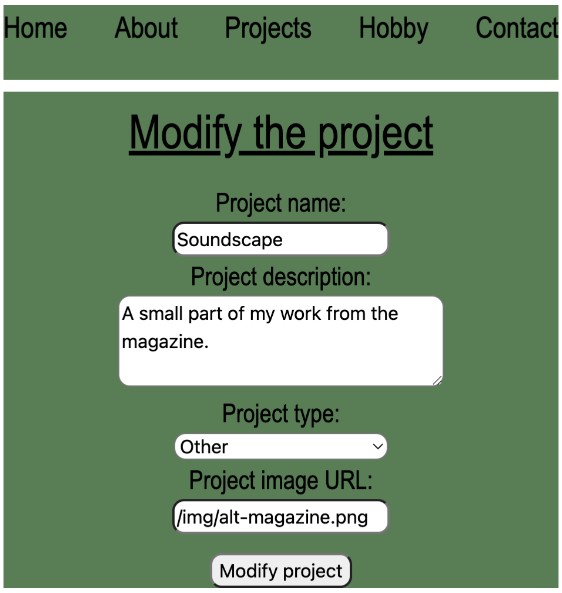
Project name:

Project description:

Project type: Other

Project image URL: /img/

Add new project



Modify the project

Project name: Soundscape

Project description: A small part of my work from the magazine.

Project type: Other

Project image URL: /img/alt-magazine.png

Modify project

4) Discussion

Implementing this portfolio website was an invaluable learning experience. It allowed me to go hands-on with core technologies like Node.js, Express, Handlebars, and SQLite. While challenging at first, grasping these tools opened up whole new possibilities.

For example, Node.js provides a robust environment for building performant web applications with JavaScript end-to-end. Likewise, Express accelerated the process of routing and server logic by a lot compared to Node.js. Features like intuitive middleware and routing parameters took away the repetitive code.

Using Handlebars templates to separate presentation from business logic resulted in clean, maintainable code. Keeping visuals in their own files prevented the code from being hard to follow. Finally, SQLite allowed local data persistence without needing to configure a separate database server. It offers a very straightforward query API that allows me to easily retrieve and update content as needed.

Piecing together these technologies to build a dynamic web app was incredibly rewarding. The project improved my skills in designing modular architectures and reusable components. I gained valuable experience with implementation, security, optimisation, and other aspects of software that is well prepared.
While CSS and JavaScript were familiar, diving deeper into core server-side development widened my perspective. I look forward to working with these modern web development technologies on even more projects in the future. The portfolio website pushed me to grow as a programmer in invaluable ways.

5) References

Watzman, S., & Re, M. (2009). Visual design principles for usable interfaces: Everything is designed: Why we should think before doing. In *Human-Computer Interaction* (pp. 19-44). CRC Press.
Owen, K., & Watson, M. (2015). *Building Your Portfolio: The Cilip Guide* (3rd ed.). Facet Publishing.
Bi, C. (2009). Research and application of SQLite embedded database technology. *WSEAS Trans. Comput*, *8*(1), 83-92.