



JÖNKÖPING UNIVERSITY

Project Report - Web Development Fundamentals

A Responsive Personal Portfolio, with Express, MVC and SQLite

Written by

Arash Tarafar - taar21ad@student.ju.se

Course Instructor: **Jérôme Landré**

Table of Contents

Introduction.....	3
Project Requirements.....	3
Technical Background.....	3
Method.....	4
Architecture.....	4
Database.....	4
Graphical User Interface.....	4
Web Application.....	4
Results.....	5
Discussion.....	6
Appendix 1: References.....	7

Introduction

Project Requirements

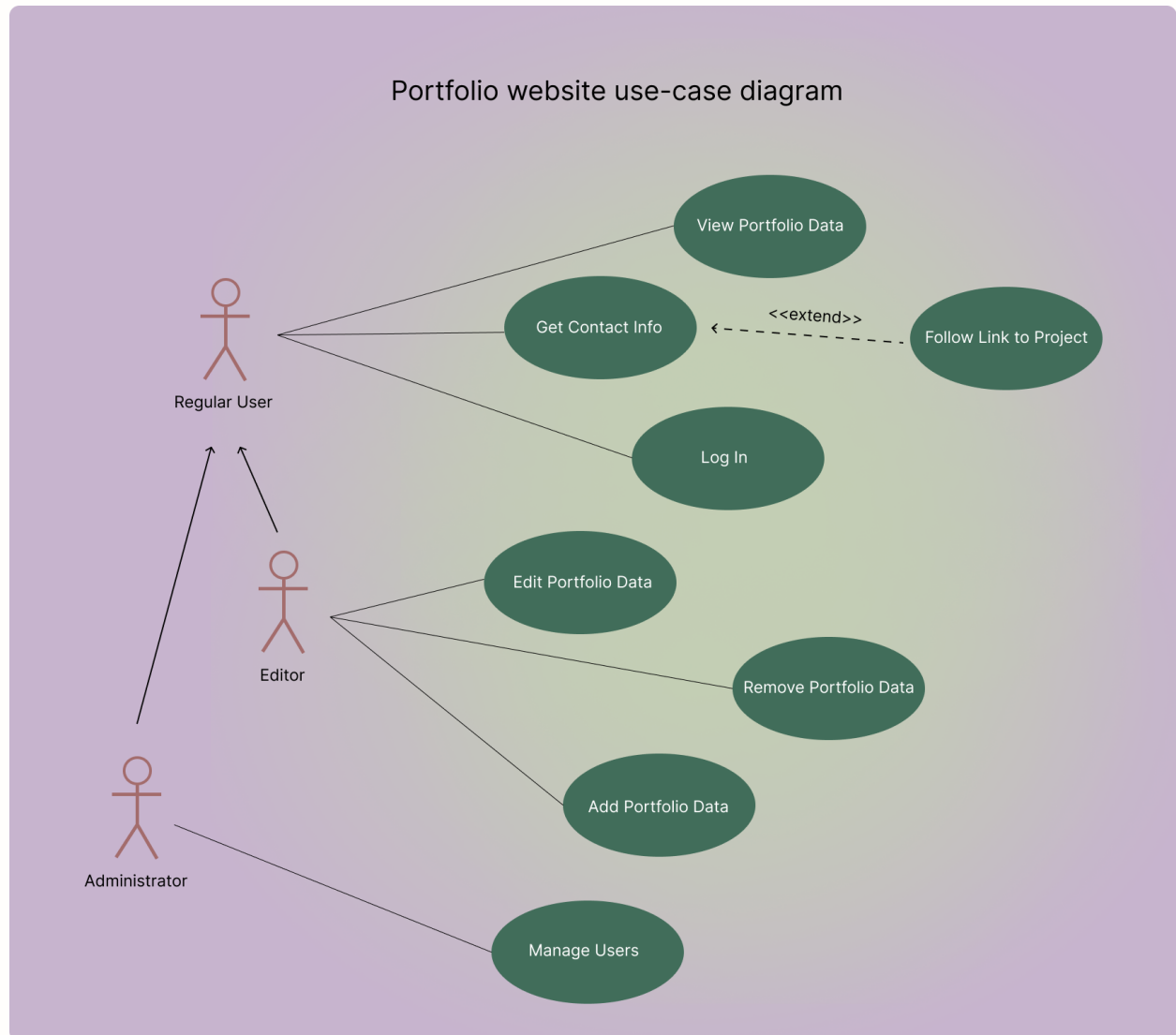
The aim of this project is to deliver a complete web experience in the form of a web app, showcasing a personal portfolio to users. The portfolio contains data about skills, educational background, past work experiences, and completed projects.

The project was developed, tested, and documented entirely by the author. The intended audience for this project are the instructors involved in the course “Web Development Fundamentals” and peers of the author in the course.

The intended outcome of the project was a complete dynamic website. This includes the webpages on the client side, the server program, and a database file. This will allow the users of the web app to gain more information about the history, and access to ways to contact the author.

This project solves the problem of lack of sufficient information about the past experiences, completed projects, and ways to contact the author. This requires the web app to have features like a contact page, basic information about the author, lists of educational and professional experiences, lists of skills, and the means to modify the content of the portfolio, add new content, or remove content.

A brief overview of the different intended users of the web app, along with the functionalities they can use, is depicted in the next page.



Use-case diagram for the portfolio web app - Users and Activities

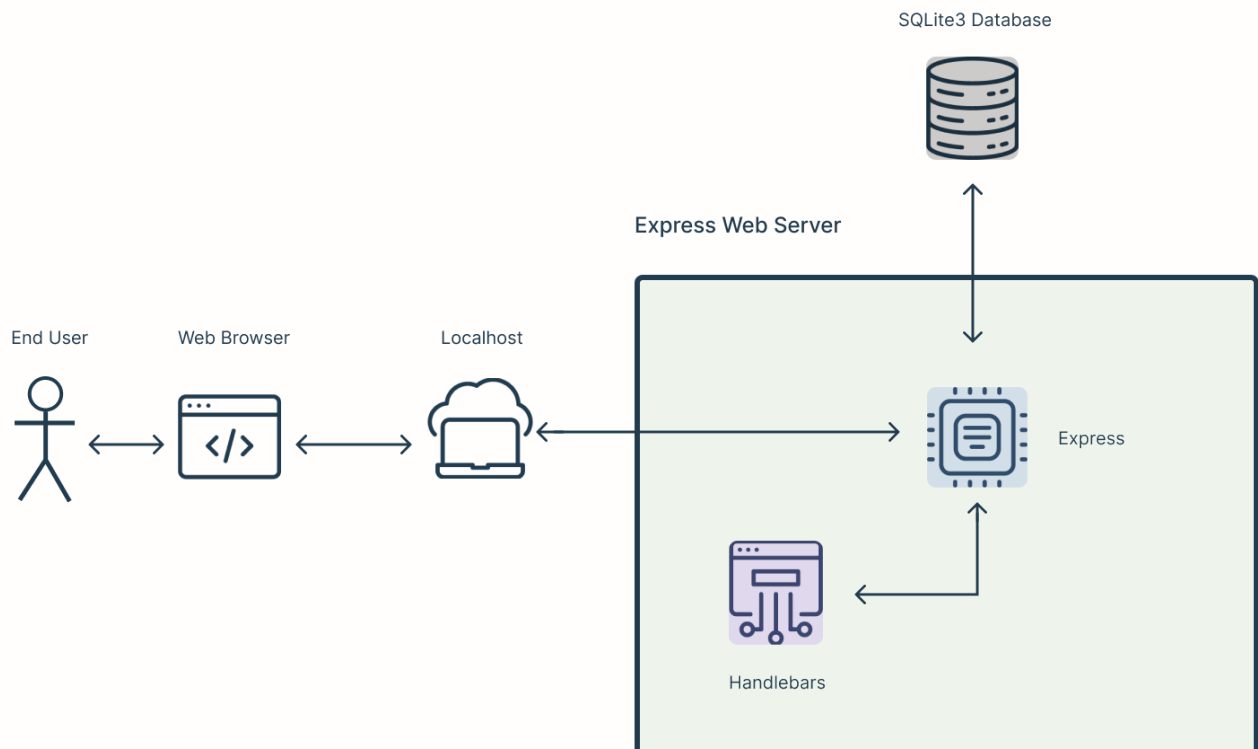
Method

Architecture

The web app has been implemented with the MVC architecture, which has three components working hand-in-hand to create a robust experience (Ramotion, 2023):

1. The portfolio data is stored inside a database file, which is called *Model* in this implementation
2. The pages of the website are created dynamically by a viewer engine, and then sent to the web browser of clients. These dynamic pages are called *Views*
3. The server application in charge of receiving requests to view pages, retrieving data from the database file, sending data to the viewer engine to create the pages, and sending them back to clients is called the *Controller* in this architecture

A visual representation of the process is shown in the figure below:



Database

For this project, the relational database management system *sqlite 3* was used, due to being a compact yet robust solution to store and retrieve data to and from a single file (SQLite, 2023).

A typical portfolio contains information about the educational and professional background, skills, and projects done by an individual, as well as their details and contact information. The general contact information of a person is not subject to short-term changes and is therefore omitted from the database i.e. it will be written directly into web pages.

For each of the remaining data groups, a separate table is created in the database. Since education and work are performed in organizations, an *organizations* table is also created, which holds basic information about different workplaces and educational institutes. Any educational or work experience is performed in an organization. Meanwhile, a single organization can be associated with more than one education or work experience. This relationship is depicted in the Entity-Relationship Diagram (ERD) on the next page. In all tables, the unique identifier of every entry is their ID, which is a primary key in the table.

The users table holds information about all non-administrator users of the website. The administrator is a special account, whose data cannot be modified.

In the entity-relationship diagram in the next page, parts of the attribute names in the tables *organizations*, *experiences*, *projects*, and *educations* are truncated, and followed by “...” which suggests that it should be substituted by the remainder of the word.

The **Administrator credentials** are as follows:

Username: admin

Password: ^r^shs0M3t1m3sL0s3s



Entity-Relationship Diagram - All Tables, their Attributes, and Relations between Them

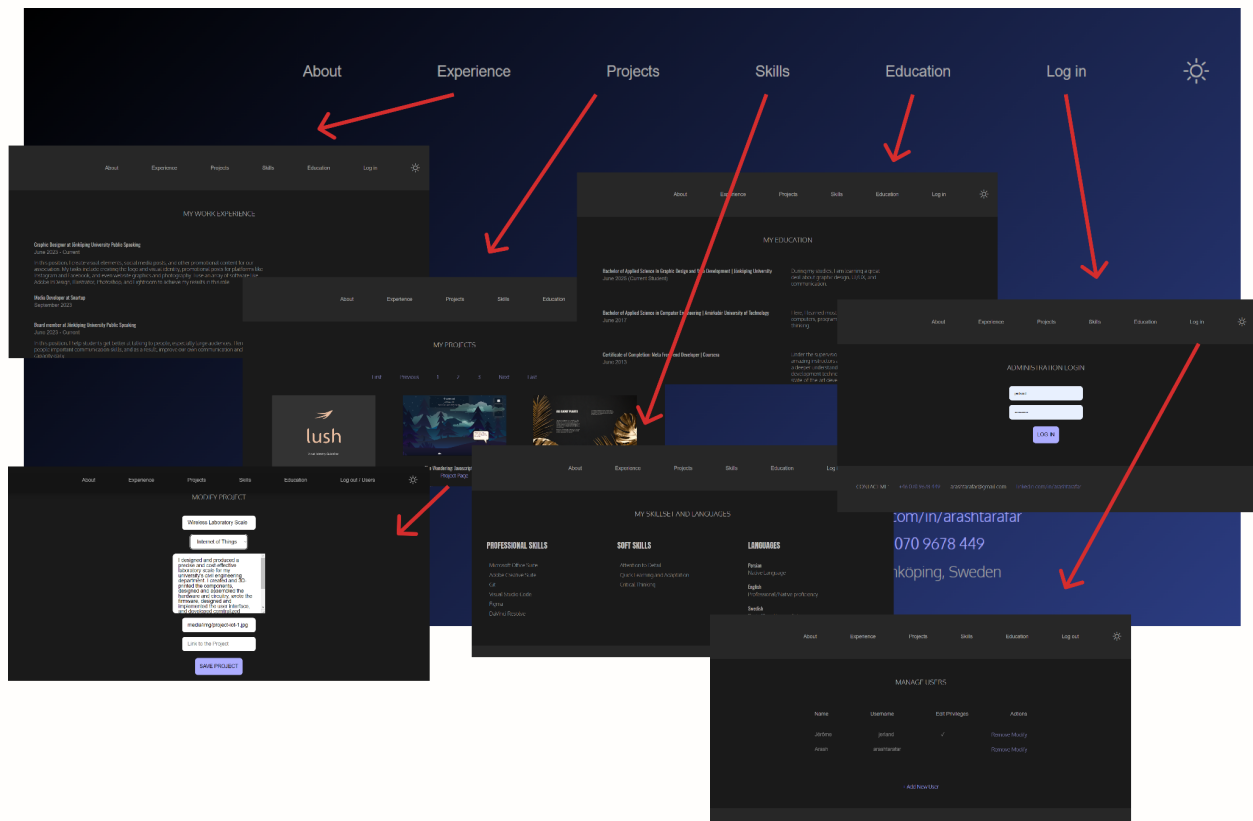
Graphical User Interface

The layout, theme, and elements of the graphical user interface for this web app have been borrowed from the original portfolio website of the author, which is a responsive portfolio page. However, in contrast to the single-page design of the original portfolio website, this web app employs a multiple-page approach, in which every page can be accessed by clicking on the respective element in the navigation menu on the top of the pages.

There are also internal pages, and a login page. For example, whenever a user selects a project to view more details, attempts to create a new data entry on a page, or modify existing data, new page layouts are rendered which are completely aligned with the overall theme of the original portfolio, keeping the design values of simplicity and function (Arash Tarafar, 2023) in mind. All functionality is intuitively accessible via properly-placed links within the layout. Client-side javascript is used for smooth scrolling, and a light-dark theme switch. CSS transitions have been used throughout all web pages to ensure a smooth browsing experience. The use of pagination in the projects page ensures that users do not have to continuously scroll to find the item they want to know more about.

Various principles and techniques have been used to provide the best user experience, such as a back-to-top button, grouped navigation items, and a hamburger menu for mobile device displays.

Below is a visual representation of navigation within the web app layout, along with most of the page views. (in this simple navigation system, users only a click away from their destination)



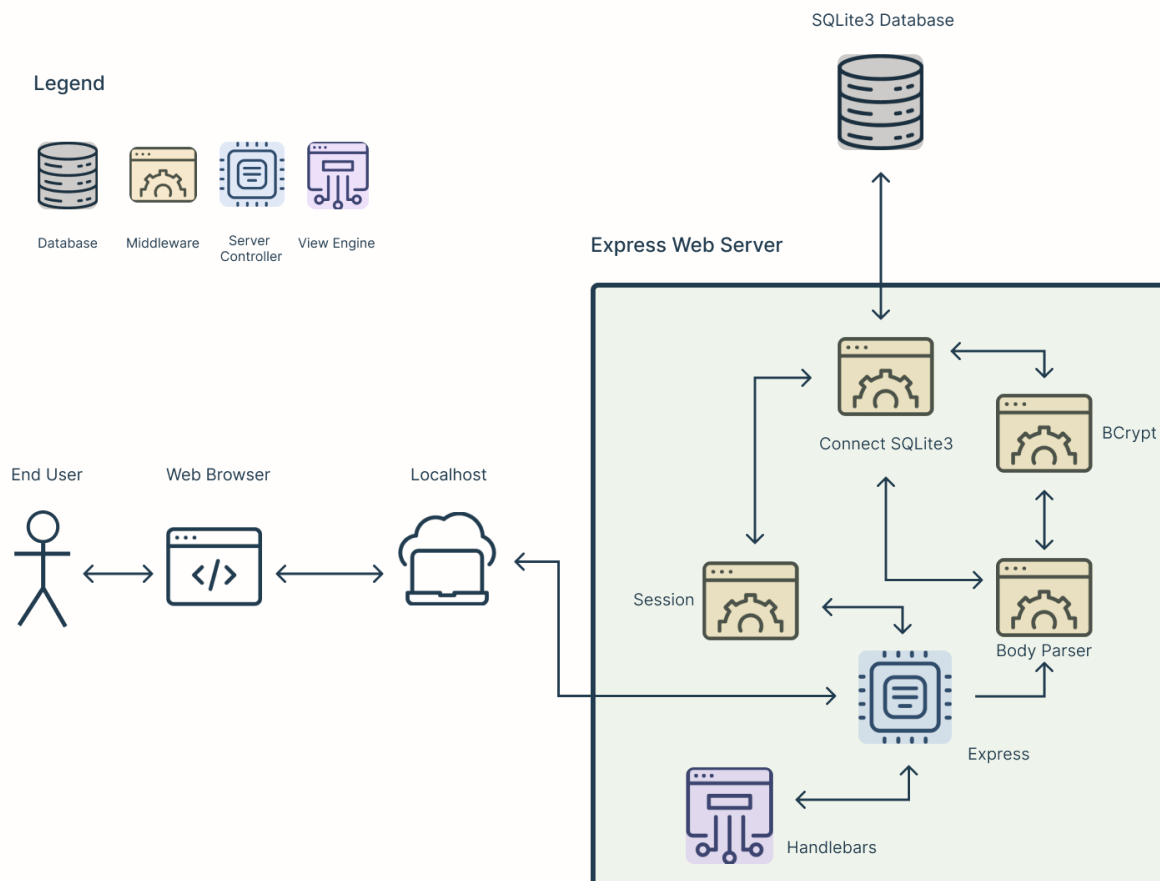
Web Application

The Javascript language has been used for the implementation of the server of this portfolio web app. The page views have been coded in the HTML language, and rely on style files containing CSS code for their visual elements and organization. Client-side Javascript has been used for the functions of the web pages that facilitate user experience.

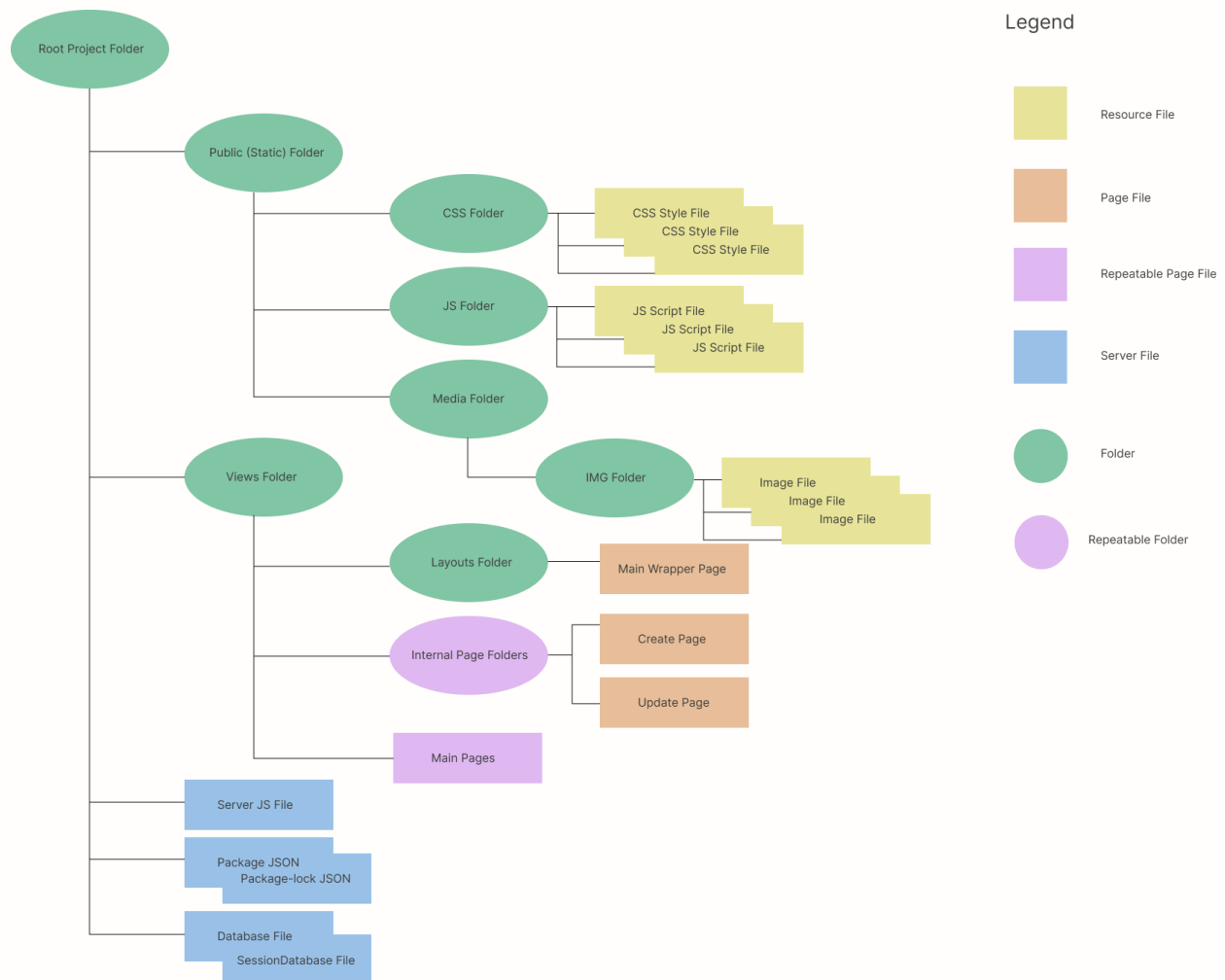
The CSS library *spectre* is used throughout the web pages to ensure a consistent grid layout, and facilitate responsiveness of the pages (Spectre Docs, n.d.). The server employs express, a server library for NodeJS. A variety of middleware function libraries have been used for different purposes (J. Landré, Web Development Fundamentals, October 3, 2023):

1. Session: enables the secure retrieval of session variables on the server, and respective session cookies on the client
2. Body Parser: enables the retrieval of variables from inside of an HTTP request body
3. BCrypt: enables hashing of strings, which is used to store password data securely on the server (J. Landré, Web Development Fundamentals, October 10, 2023)
4. Connect SQLite3: the connection between session handles on the server, and the session database file

A visual representation of all the elements of the server is as below:



The project code utilizes comments in all sections and is extremely easy to navigate for new programmers. A visual representation of the project folder structure is as below:

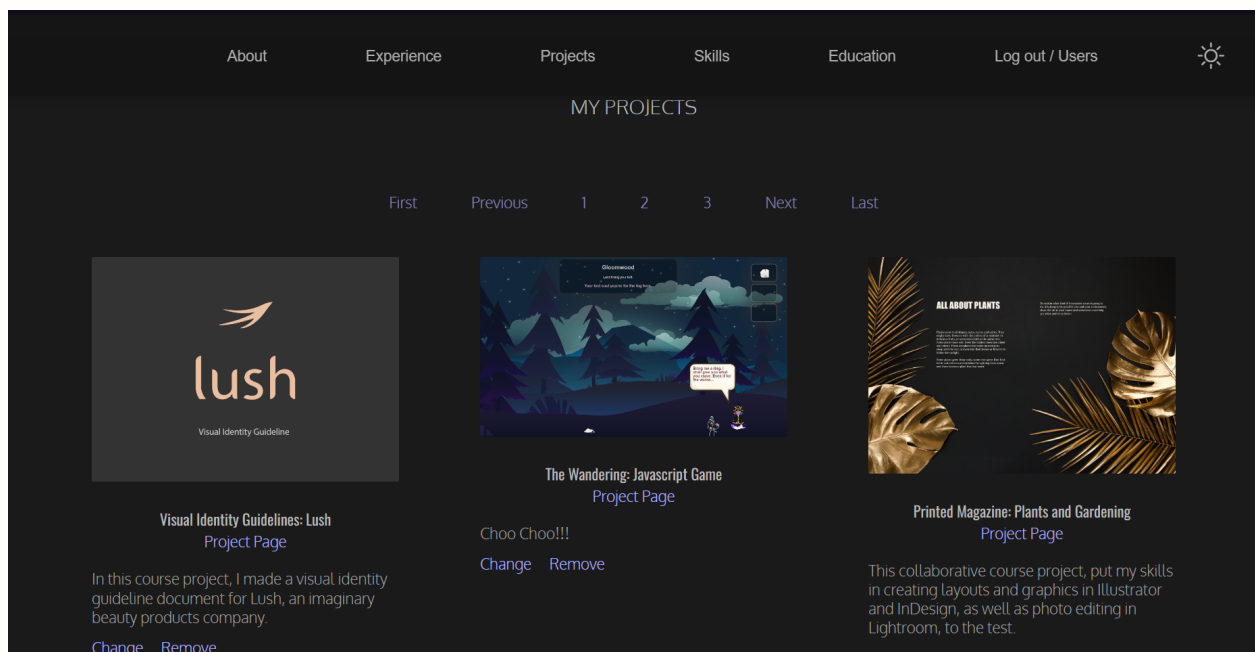
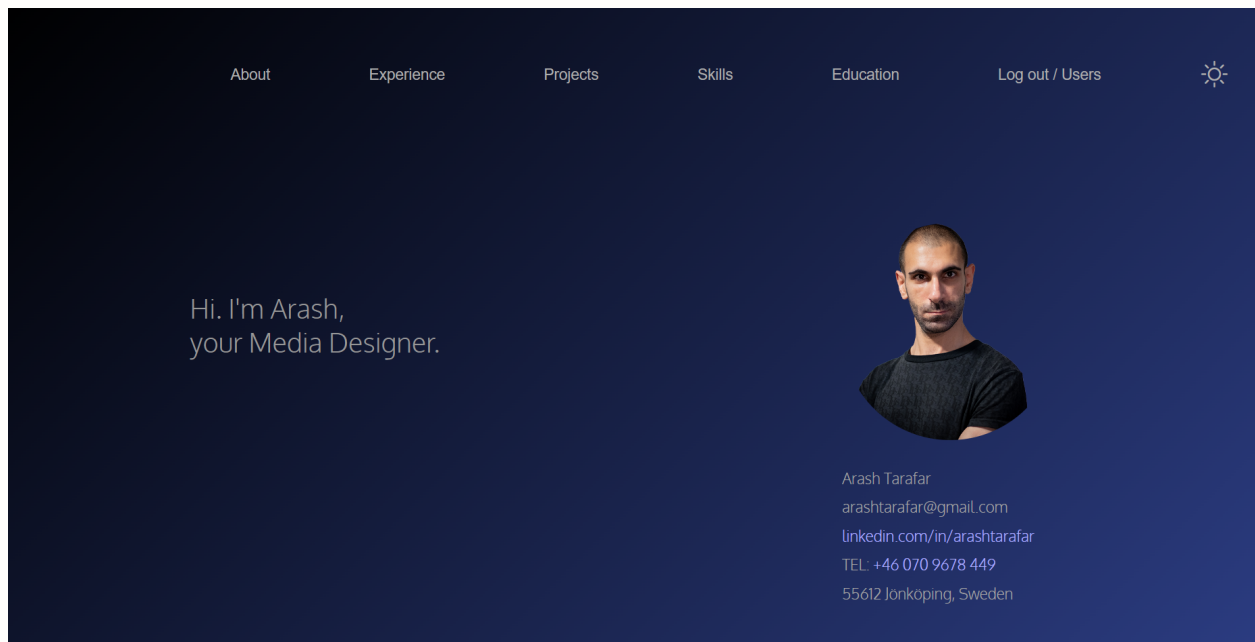



Security is a major concern in the implementation of web applications. In this project, the use of proper notation when handling SQL queries ensures that the app remains safe from attempts of SQL injection (Quick, 2022). Furthermore, the use of sessions helps protect user accounts from being abused by third parties.

The passwords of users go through the hashing process by the middleware *bcrypt* before storage in the server, which ensures that even in the case of vulnerability, sensitive user data is safe from abuse by online hackers.

Results

The result of this project is a complete web app experience, with responsive views that utilize industry-standard UX design, a complete content management system, and secure user management dashboard. Below are a few snapshots of the finished product:



[About](#)[Experience](#)[Projects](#)[Skills](#)[Education](#)[Log out / Users](#)

NEW EDUCATION

Degree

Select an Institute

Select an Institute

Amirkabir University of Technology

Jönköping University


Michigan State University

University of Michigan

Udemy

Coursera

ADD EDUCATION

[About](#)[Experience](#)[Projects](#)[Skills](#)[Education](#)[Log out](#)

MANAGE USERS

Name	Username	Edit Privileges	Actions
Jérôme	jerland	✓	Remove Modify
Arash	arashtarafar		Remove Modify

+ Add New User

[CONTACT ME](#) [+46 370 0 370 440](#) arashtarafar@gmail.com <https://arashtarafar.com>

Discussion

In the course of creating this portfolio web app, I learned many new skills, such as the ability to work with NodeJS as a back-end solution, use express to facilitate many server functions within NodeJS, and use middleware function libraries to read and manipulate data more easily.

Since my goal was to produce the best website experience as was within my capability, I paid more detailed attention to UX design concepts, UI, colors, typography, and the quality-of-life features all users expect from a well-built web solution.

I learned about the MVC architecture, and how to implement it with handlebars, which acted as a gate to understanding more about dynamic page design and led to being introduced to many other dynamic front-end frameworks such as Angular, Vue, and React.

Appendix 1: References

1. Zhu, Y. (n.d.). *Spectre Docs*. Spectre CSS Framework.
[Responsive - Layout - Spectre CSS Framework](#)
2. Arash Tarafar. (2023). *My Portfolio: Arash Tarafar*. Snartup.
[MyPortfolio: Arash Tarafar - Snartup](#)
3. Ramotion. (2023, August 22). *MVC Architecture: Simplifying Web Application Development*. Ramotion.
[MVC Architecture | Ramotion Branding Agency](#)
4. SQLite. (2023, October 10). *About SQLite*. SQLite.
[About SQLite](#)
5. Quick, J. (2022, March 3). *How to prevent SQL injection attacks in Node.js*. PlanetScale.
[How to prevent SQL injection attacks in Node.js](#)