jamk

A Portfolio Display of Software Development Mastery

Billy Mohajeri

Master's thesis
December 2023
Full Stack Software Development

jamk | Jyväskylän ammattikorkeakoulu
University of Applied Sciences

jamk

Description

Mohajeri, Billy

A Portfolio Display of Software Development Mastery

Jyväskylä: Jamk University of Applied Sciences, November 2023, 73 pages.

Degree Programme in Full Stack Software Development. Master's thesis.

Permission for open access publication: Yes

Language of publication: English

Abstract

The rapid evolution in software development demands a comprehensive display of mastery in various aspects of the field. The goal was to produce a portfolio that not only displayed technical expertise but also showed how this knowledge was applied in practical situations. The assignment was to choose and showcase a number of software projects that together cover a wide range of software development capabilities.

The method of implementation included two steps: first, a careful selection of software projects that represented different aspects of software development, such as managing databases, front-end, back-end, and user interface design. The development of an interactive web platform to showcase these projects came in second. With a focus on creativity, and problem-solving abilities with a variety of programming languages and tools, this method provided a comprehensive understanding of the capabilities.

The outcome was a balanced portfolio that demonstrated a wide variety of technical abilities and demonstrated how these abilities are actually used in the development of complex software solutions. Every project in the portfolio had a thorough description that included an analysis of the challenges it faced and the solutions used. This thorough demonstration successfully showed an advanced understanding of software development concepts and the ability to use them creatively in a variety of situations.

In conclusion, the portfolio served as an example of a well-rounded skill set in software development. It effectively demonstrated not only the technical proficiency but also the capacity for critical analysis and creative problem-solving. The portfolio is a crystal-clear representation of a software development journey, demonstrating a dedication to lifelong learning and flexibility in response to the ever-changing demands of the industry. This work emphasizes how crucial it is to demonstrate software development expertise through a practical, hands-on approach.

Keywords/tags (subjects)

Portfolio, Software Development, JavaScript, Ruby on Rails, PostgreSQL, React

Contents

Figures

1 Introduction

In today's world, life without software is almost impossible. From the rapid, complex decisions made by self-driving cars to the simple execution of a game on a smartphone, everything operates on hidden code. As a result, the impact of software is felt in nearly every industry. While universities offer comprehensive theoretical education, many students often find it challenging to use this knowledge in real-world situations. This highlights the importance of a personal portfolio. In the post-Corona pandemic world, having a digital presence has become more crucial than ever, especially for recent software graduates preparing to enter the job market. The establishment of a well-designed portfolio is essential for students studying software, showcases their top projects, and highlights their software development skills.

An electronic portfolio provides a platform for developers to exhibit their most prominent work and demonstrate their proficiency in software development. This platform enables industry recruiters to get insight into the qualifications and talents of developers, providing a practical vision of the developer's skills. It allows recruiters in the industry to go beyond just the qualifications listed on a resume, offering a practical look at a developer's capabilities.

In this report, an initial brief introduction is provided. Following this, the basic knowledge base section delves deeper into the importance, necessity, and benefits of this thesis.

In modern society, often referred to as the age of artificial intelligence, the importance of software development has increased a lot. Software plays a crucial role in everything from the tiny details of our daily lives to humanity's biggest and most advanced projects. With a minimum of four years of experience in software development, this portfolio presents a range of work examples, demonstrating hands-on expertise in the field. The primary objective of this thesis is to showcase a collection of accomplished projects, reflecting the practical application of skills acquired over years of experience in software development. The second aim of this study is to fill a gap between theoretical and practical application, hence facilitating the understanding of the complexities associated with software implementation within real-life contexts. aim of this thesis is not only to showcase completed projects but also to bridge the gap between theory and practice, simplifying the complexities of software implementation in real-world scenarios. The following knowledge base

section will go into detail about the thesis, covering its content, goals, results, importance, uses, tools, and methods.

This part provides detailed information on the thesis content by elaborating a thorough description of software projects. Each project exhibits different aspects of software development, from the initial idea to the final implementation.

Learning takes place in a multitude of settings and is an ongoing process. From acquiring knowledge and skills to their modification and application, these learning processes and practices have always captivated educators and researchers (Bransford, Brown, & Cocking, 2000). It is clear that learning has been extensively studied and explored given the variety of explanations for this phenomenon throughout human history (e.g., Price, 2014; Richardson, 2000). Apart from focusing only on understanding learning in and of itself, ongoing efforts are made to research and create suitable tools or approaches to facilitate learning that can improve learners' performance or learning outcomes in a variety of contexts, including adult learning (e.g., Denton, 2012; Gagné, Wager, Golas, & Keller, 2005; Maher & Gerbic, 2009; Price, 2014; Richardson, 2000). In particular, emphasis is placed on formal or informal learning, that is, learning that helps people practice a particular profession both inside and outside of the workplace and that prepares people for that career (e.g., Firssova, 2006; Herman & Kirkup, 2008). Depending on the degree obtained and the location of study, formal learning or training, particularly in a university setting, often takes a few years (e.g., 3–4 years) to fully understand theories and develop the competencies required to practice one's future career. However, depending on the field of study and curriculum, certain skills needed for a chosen career can usually be acquired in a short amount of time, such as a few months during an internship or a few years. Consequently, in order to compensate for the lack of formal training for the chosen career and to support the ongoing development of professional skills, more learning opportunities are required. Additional learning opportunities could take the shape of workshops, training, or unstructured learning activities conducted both inside and outside of the office. However, there are a number of important variables and circumstances that can either help or impede the development of workplace competencies.

As a matter of fact, advancements in technology and the broad adoption of these tools continue to be crucial factors in many facets of society, including the facilitation of knowledge management

practices and educational advancement. The technologies that are available to support our learning activities are also relatively easy to use in this digital environment, as is searching for or acquiring new information. The widespread use of electronic devices, such as laptops and cellphones, in conjunction with web-based tools or applications, such as learning management systems and knowledge management tools, in educational settings, highlights the significance of technologies in society.

The use of technological artifacts makes it clear how technology affects teaching and learning in higher education when taking into account the educational context. Price and Kirkwood (2014) have listed a few of these artifacts as follows:

- Blended learning/e-learning/hybrid courses
- Audio/podcasts
- Video resources/lectures/games
- Multimedia tools
- Virtual laboratories/fieldwork
- Blogs
- Collaborative tools/wikis
- Online discussion boards/conferences/forums
- E-portfolios
- Online course resources
- Electronic voting/personal response systems, and
- Assistive technologies

This demonstrates how technology is permeating higher education in a variety of contexts and how different technologies are being used to support students in different ways. Pervasive social technologies were also helpful for academic purposes, according to the results of some studies that asked teachers to report on their daily activities using various technologies. Harteis and Daunert (2014). For instance, in higher education settings, learners' portfolios are displayed via e-learning platforms (Daunert & Harteis, 2014). E-portfolios are thought to be a valuable tool for fostering skill development and professional learning (Denton, 2012; Ehiyazaryan-White, 2012; and

Jenson, 2011). According to empirical research, e-portfolios can help learners advance professionally by enabling them to manage and oversee their own learning by allowing them to keep track of their own progress, organize the documentation of their learning process and outcomes, and direct and oversee their own learning (Bala, Adlina, Mansor, Stapa, & Zakaria, 2012; Brown, 2011; Duncan-Pitt & Sutherland, 2006; Johnsen, 2012; and Malita, 2009). Technology gives people the chance to share their work and learning experiences with others since it makes learning artifacts from both one's own and other people's collections widely accessible.

2 Research Objectives And Methods

2.1 Research Questions

Primary Research Question

"How effective is a portfolio-based approach in assessing software development skills compared to traditional assessment methods?"

This primary research question aims to critically evaluate the effectiveness of portfolio-based approaches in assessing a developer's skills. The focus is on comparing and contrasting these modern, holistic assessment methods with more traditional, often theoretical evaluation techniques. The intention is to ascertain whether portfolios provide a more accurate and comprehensive representation of a developer's capabilities.

Secondary Research Questions

"What key skills and knowledge areas were developed through working on these projects?"

This question explores the specific skills and areas of knowledge that are developed and honed through the process of working on various software development projects. It aims to identify the breadth and depth of skills that a software development portfolio can showcase.

"How does the process of creating a software development portfolio contribute to the professional learning and growth of a developer?"

This inquiry delves into the educational and developmental aspects of creating a software development portfolio. It seeks to understand how the process of compiling and curating a portfolio facilitates a developer's ongoing professional learning, skill enhancement, and career progression.

2.2 Research Objectives

The research objectives associated with this study are to:

Evaluate Portfolio-Based Skill Assessment: To provide a comprehensive evaluation of the efficacy of portfolio-based methods in assessing software development skills, especially in comparison to traditional evaluation methods.

Identify Skills and Knowledge Development: To identify and articulate the key skills and knowledge areas that are developed through the work on various software development projects featured in the portfolio.

Understand the Role of Portfolio Creation in Professional Growth: To explore how the process of creating a software development portfolio contributes to a developer's professional learning, skill enhancement, and career development.

These objectives are designed to guide the research in uncovering the value and effectiveness of software development portfolios, both as tools for skill assessment and as mechanisms for professional development. The outcome of this research aims to provide valuable insights for developers, educators, and industry professionals regarding the benefits and potential of portfolio-based approaches in the field of software development.

2.3 Research Methodology

Introduction

This thesis section covers the development and creation methodologies used to create the software projects that together make up the "Portfolio Display of Software Development Mastery." The approaches discussed in this article are essential for demonstrating how theoretical knowledge is applied in real-world software development scenarios. They include workload distribution using a Kanban board, a strict code review process, and a mobile-first design approach, each contributing significantly to the project outcomes.

Workload Distribution

Kanban Board Implementation: The project tasks were systematically organized using a Kanban board within a GitHub project. This method facilitated the translation of project requirements into a structured set of tasks, assisting in clear visualization and tracking of progress.

Task Allocation: Tasks were fairly distributed among team members, with time estimates assigned to each to ensure balanced workload and efficient time management. Common tasks were approached collaboratively, either through pair programming or by dividing them among all team members.

Code Review Process

Peer Review: A peer review system was essential to the development process. Each task was developed on a feature branch and subjected to a separate pull request, reviewed by a team member. This process encouraged a culture of constructive feedback, with changes being committed based on the reviews.

External Review: After completing all tasks and merging them into the development branch, the projects were subject to an external code review. This phase was pivotal for receiving external feedback, guaranteeing compliance with coding standards, and improving the overall quality of the project.

Mobile-First Design Approach

The projects were developed using a mobile-first design methodology. This approach prioritized the optimization of the applications for mobile devices, recognizing the significant presence of mobile browsing in current times. It involved responsive design practices and a focus on creating user-friendly interfaces for smaller screens, before scaling up to larger devices.

Combining Methodologies

These methodologies were not isolated; instead, they worked together to improve the overall development process. The Kanban board helped in managing tasks and enhancing workflow, the code review process guaranteed code quality and facilitated team learning, and the mobile-first design approach kept the projects up-to-date with current web development trends.

Together, these methods supported the successful completion of the portfolio projects. They showcase not only a thorough application of software development skills but also a dedication to

quality, teamwork, and following modern development practices. This strategy has greatly contributed to meeting the objectives of this thesis, demonstrating expertise in software development.

This section highlights the practical and strategic methods used in developing the portfolio, emphasizing the importance of organized project management, quality control, and contemporary design principles in software development.

3 Knowledge Base for Portfolio Development

3.1 HTML

The Basics

The projects featured in this portfolio utilize a diverse range of programming languages and technologies. This section provides a concise summary of these tools and their application in the research activities undertaken for the thesis.

The standard markup language used to create web pages is called HTML, or HyperText Markup Language. HTML serves as the basis for all websites, providing the basic structure that is later improved and altered by other technologies such as JavaScript and CSS (Cascading Style Sheets).

HTML is made up of a number of elements that you can use to wrap or enclose different parts of the content to alter its appearance or behavior. The enclosing tags have the ability to italicize words, change the font size, make a word or image hyperlink to another location, and more.

Elements, Tags and Attributes

The head and body sections, HTML tags, and doctype declaration are all part of the structured format of an HTML document. The first line of an HTML document, known as the doctype declaration, informs the browser of the HTML version that the page is written in. Metadata about the document, including its title and links to stylesheets and scripts, are contained in the head section. The text, photos, and videos that will be shown on the page are all contained in the body section.

The hidden keywords that specify how your web browser should format and display content are called HTML tags. The content must be inserted between the <tag> and </tag> opening and closing tags.

Semantics and Accessibility

Elements' attributes offer more details about them. They are usually composed of a name and a value, like <a href="https://www.example.com">, and appear inside the opening tag.

Semantic HTML is a type of syntax that improves HTML readability by clearly defining web page sections and layout. It improves the readability and flexibility of web pages, making content easier for search engines and browsers to understand.

A key component of web accessibility is HTML. The use of <header>, <footer>, <article>, and <section>—appropriate HTML elements—to convey meaning aids assistive technologies in comprehending the structure and content of web pages.

HTML5: The Evolution of Web Standards

HTML5 represents a major leap forward in the evolution of web standards. Officially finalized in October 2014, HTML5 was developed to address the modern needs of web development by introducing new features, improving interoperability, and making the web more accessible and efficient. It has rapidly become the cornerstone of modern web design, setting the stage for richer, more interactive user experiences.

Enhanced Semantics and Accessibility

HTML5 introduced a set of new semantic elements that allow for more descriptive document structure. Elements like <header>, <footer>, <article>, <section>, and <nav> provide a clearer meaning to the structure of a page, which is not only beneficial for developer readability but also critical for assistive technologies used by individuals with disabilities. This semantic structure ensures that web content is more accessible and that web applications are available to a wider audience.

Multimedia and Forms

Before HTML5, embedding audio and video in web pages required third-party plugins like Flash. HTML5 changed the game with the <audio> and <video> elements, enabling native multimedia embedding that is fully integrated into the web page markup. This has allowed developers to create rich multimedia experiences that are more reliable and consistent across different browsers and devices.

HTML5 also introduced new form elements and attributes that provide greater interactivity and functionality. New input types such as email, date, range, and color offer built-in validation and controls tailored to the data they are intended to receive. Form attributes like placeholder, autofocus, autocomplete, and required improve the user experience by making forms easier to use and by providing immediate feedback.

Advanced APIs and Offline Capabilities

HTML5 included a set of robust APIs that let programmers create sophisticated apps without the need for additional plugins. These APIs include the Geolocation API for location-based services, the Drag and Drop API for simple interface interactions, and the Canvas API for 2D drawing. Web Workers enable web applications to execute scripts in background threads, and Web Storage and IndexedDB offer client-side data storage options.

The ability to create offline web applications is one of HTML5's transformative features. Developers can designate which files the browser should cache and make available to offline users using the AppCache and Service Workers features. This has made it possible to create web apps that can continue to offer seamless user experiences even in the face of unstable connectivity.

The Future with HTML5

HTML5 has set a new standard for web content that is interactive, accessible, and performant. Its forward-looking design accommodates not only current web development needs but also anticipates future advancements. With the continuous evolution of internet technologies, HTML5 remains at the forefront, providing the foundation for the next generation of web applications that are as diverse as they are powerful.

In summary, HTML5 has revolutionized web development by providing a robust framework for creating sophisticated web applications. Its emphasis on semantic markup, integrated multimedia capabilities, and powerful APIs has transformed the way developers create web content and has significantly enhanced the user experience on the web.

3.2 CSS

CSS Basics

An HTML or XML document (including XML dialects like SVG, MathML, or XHTML) can have its presentation style specified using a stylesheet language called CSS, or Cascading Style Sheets. CSS specifies how items should look on a screen, paper, in voice, or in other media.

On the web, HTML is used to structure content, whereas CSS is used for style and design. It makes it possible to style web pages. What's more, CSS lets you do this without relying on the HTML that makes up every webpage.

CSS Syntax

CSS consists of style rules that specify how content is displayed. A style rule consists of a declaration block and a selector:

The declaration block is separated by semicolons and contains one or more declarations. The selector makes reference to the HTML element that needs to be styled. In every declaration, the name of the CSS property and its value are separated by a colon (:).

The separation of presentation (CSS) and content (HTML) is a fundamental advantage of CSS. This gives you more flexibility and control over how content is presented across various browsers and devices, in addition to making HTML easier to read and maintain.

The cascade is one of the most powerful aspects of CSS. It's the process of determining which rules apply to an element when multiple rules could apply. The cascade takes into account the specificity of a selector, the location of the rule, and the order of the rules.

CSS is integral to responsive web design, which ensures that web content looks good on all devices. CSS media queries allow you to create multiple styles for different browsers and devices sizes, improving accessibility and user experience.

To speed up development and ensure consistency, many developers use CSS frameworks like Bootstrap or Foundation. Additionally, CSS preprocessors like Sass or LESS enable writing cleaner and more organized code, which then compiles into standard CSS.

CSS version 3

The most recent iteration of the Cascading Style Sheets (CSS) specification, which describes how HTML documents should be presented, is called CSS3. With the introduction of several new selectors, properties, and ideas, this CSS version has revolutionized web design and allowed programmers and designers to produce more intricate and responsive website layouts.

With its advanced selectors, which provide greater flexibility and control over document styling, CSS3 has expanded possibilities. Developers can apply styles based on attributes, an element's position within its parent, or even the element's state thanks to these selectors. Web elements can now have visual depth and realism without the need for image-based designs thanks to new properties like border-radius, box-shadow, and text-shadow. Multi-column layouts also allow text to flow naturally across multiple columns, much like in print media.

Enhanced Visual Effects

One of the most exciting aspects of CSS3 is its ability to create rich visual effects. With gradients, developers can produce smooth transitions between colors, crafting a gradient effect directly in the browser. Transitions and animations are now possible without relying on JavaScript or third-party libraries. These features allow for the creation of dynamic user interfaces with elements that change style smoothly over time and respond to user interactions.

CSS3 has been instrumental in the rise of responsive web design. Media queries enable CSS to apply different styling rules based on the device's characteristics, such as its screen size, resolution, and orientation. This responsiveness ensures that web content is accessible and legible on a wide range of devices, from desktop monitors to smartphones.

Flexbox and Grid Layouts

The Flexible Box Layout, commonly known as Flexbox, allows for a more efficient way to distribute space and align content in complex layouts. The CSS Grid Layout introduces a two-dimensional grid-based layout system, offering an even more powerful way to design grid-based user interfaces. Both Flexbox and Grid provide a level of control over web layouts that was difficult or impossible to achieve with previous CSS versions.

The adoption of CSS3 has been a gradual process, with browser vendors progressively implementing various features. Developers often need to consider browser support and may use vendor prefixes to ensure compatibility. Tools like Autoprefixer can automate this process, adding necessary prefixes to CSS rules.

The capabilities introduced with CSS3 have had a profound impact on web design, allowing for more expressive, efficient, and flexible styling options. The standard has made it easier for designers to bring their visions to life without cumbersome workarounds or extensive image use, leading to faster page load times and better user experiences.

In summary, CSS3 has pushed the boundaries of what's possible in web design, allowing for creative and responsive layouts that adapt to the ever-changing landscape of devices and screen sizes. With its advanced features and widespread support, CSS3 continues to be a cornerstone technology for anyone involved in web development.

3.3 JavaScript

The Basics

Since the beginning, JavaScript has become the most important language for building websites. It's what makes most current websites work dynamically. It is a coding language that lets developers add complicated features to web pages. These features can include showing timely content changes, interactive maps, live video streaming, and even online games.

JavaScript has significantly altered the domain of web development. In the present day, its functionality has grown beyond the creation of dynamic front-end user experiences to include server-side programming. The wide range of features of JavaScript enables programmers to use it across the entire web development stack, ensuring a smooth combination of the front and back ends of web applications.

The Evolution of JavaScript

JavaScript's journey began in 1995, a pivotal year that marked its inception as the brainchild of Brendan Eich while at Netscape. Initially designed to add interactivity to the static web pages of the time, JavaScript has since undergone a remarkable transformation. From its nascent stage as Mocha, to LiveScript, and finally JavaScript, the language became a catalyst for dynamic web development. With the World Wide Web Consortium's (W3C) endorsement, JavaScript rapidly became a standardized tool implemented in all major browsers.

Standing alongside HTML and CSS, JavaScript completes the triad of cornerstone technologies that drive the World Wide Web. HTML lays the content foundation, CSS adds stylistic flair, and JavaScript introduces the element of interactivity. This powerful trio has shaped the internet, with JavaScript playing a crucial role in enhancing user experience. It has enabled developers to write scripts that respond in real-time to user interactions, leading to dynamic updates without page reloads—ushering in a new era of responsive web design.

JavaScript transcends the traditional boundaries of web programming by allowing developers to create fully interactive experiences. It has the unique ability to respond to user events, animate content, and communicate with web servers in a seamless fashion. This has not only improved the

functionality of web pages but also the expectations of web users, who now enjoy richer and more engaging online experiences.

The advent of Node.js marked JavaScript's expansion from a browser-specific tool to a server-side technology. This development has blurred the lines between front-end and back-end web development, enabling the use of JavaScript across the entire development stack. Node.js leverages Chrome's V8 engine, allowing developers to build scalable network applications right in JavaScript. This cross-platform capability has been instrumental in popularizing JavaScript as a full-stack development language, showcasing its versatility and robustness.

Basic Syntax

The building blocks of JavaScript begin with its basic syntax. Variables are containers for storing data values, declared using var, let, or const. Data types in JavaScript are dynamic and loosely typed, which means you don't need to declare a type of value in advance. JavaScript variables can hold different data types: numbers, strings, objects, and more. Operators, from arithmetic to comparison to logical, are used to compute values, compare values, and perform logical operations, respectively. JavaScript statements, composed of one or more actions, are executed by the browser sequentially, allowing developers to write instructions for performing tasks.

Functions, Scope and Event Handling

Functions are one of the fundamental components of JavaScript, enabling developers to encapsulate code into reusable blocks. They can take parameters, perform actions, and return the results. Scope in JavaScript refers to the current context of code, which determines the visibility or accessibility of variables to other parts of the code. There are two types of scope – local and global. Understanding scope is crucial as it ensures that variables are only accessible where they’re meant to be, preventing potential issues with variable naming conflicts.

Event handling is central to JavaScript’s interactive capabilities. Events are actions or occurrences that happen in the system you’re programming, which the system tells you about so your code can

respond to them. For instance, JavaScript can listen for an event such as a button click, a form submission, or a page load, using event listeners. Once an event is detected, JavaScript can trigger a function or a series of actions, resulting in dynamic changes to the web page.

Error Handling

No matter how careful you are with writing your JavaScript code, errors are inevitable. Error handling is a critical aspect of JavaScript programming. The language provides a try...catch statement that allows you to test a block of code for errors (the try block), and then handle the error (the catch block), if one occurs. This ensures that even if an error occurs, the script doesn't just stop, but handles the error gracefully, allowing the rest of the script to continue running.

Object-Oriented Programming and Asynchronous JavaScript

JavaScript, unlike class-based object-oriented languages, utilizes prototypes for inheritance. Every JavaScript object has a prototype, and objects inherit properties and methods from their prototype. Understanding prototype-based inheritance allows developers to create an object hierarchy, facilitating code reusability and the principles of Object-Oriented Programming (OOP). Alongside prototypes, the 'this' keyword is a fundamental concept in JavaScript OOP, referring to the object it belongs to, and its value can change depending on the context in which it's used, especially when dealing with event handlers and callbacks.

JavaScript's asynchronous capabilities are vital for performing tasks like server requests without blocking the main thread. Callbacks are functions passed into other functions as arguments and are executed after the first function completes. This can lead to 'callback hell', which is where Promises come in. Promises represent the eventual completion (or failure) of an asynchronous operation and its resulting value. The async/await syntax builds on promises, providing a more straightforward way to write asynchronous code that appears synchronous or blocking.

DOM Manipulation, APIs and External Libraries

The Document Object Model (DOM) is a programming API for HTML and XML documents. It represents the page structure as a tree of objects that JavaScript can interact with. JavaScript can cre-

ate, modify, and delete HTML elements, set styles, classes, and attributes, and listen for and respond to events in the DOM, making the user experience dynamic and interactive. This interaction is crucial for tasks like validating form inputs, toggling visibility of elements, and updating content dynamically.

JavaScript extends its capabilities through Application Programming Interfaces (APIs) and external libraries. APIs provide a way for different software components to communicate with each other, and JavaScript can use Web APIs provided by browsers to perform tasks like making HTTP requests to servers, accessing device hardware, and more. Libraries like jQuery simplify DOM manipulation, event handling, and Ajax calls, providing cross-browser compatibility and allowing developers to write less and do more. These libraries have been instrumental in accelerating development and ensuring consistency across different web browsers.

Frameworks and Modern Syntax

JavaScript's ecosystem has significantly expanded with the introduction of powerful frameworks and tools that streamline the development process. Frameworks like React, developed by Facebook, allow for building user interfaces with a component-based architecture, enabling the creation of reusable UI components. Angular, by Google, offers a full-fledged MVC framework for developing robust, large-scale applications. Vue.js, known for its progressive framework, is designed from the ground up to be incrementally adoptable. These tools provide developers with rich ecosystems and the ability to write declarative, high-performance code.

With the advent of ECMAScript 6 (ES6), JavaScript introduced a new syntax that brought the language closer to other object-oriented languages, making it more accessible to developers from various programming backgrounds. ES6 features like classes provide a much-needed structure to JavaScript, making the code more organized and maintainable. Arrow functions offer a more concise syntax for writing functions and addressing common issues with the 'this' keyword. Template literals simplify complex string concatenation, significantly improving code readability.

Best Practices and Development Environments

The JavaScript community actively promotes coding standards and best practices to ensure high-quality code. Style guides from companies like Airbnb provide rules for writing consistent JavaScript code. Tools like ESLint help developers adhere to these best practices by analyzing the code for patterns that may lead to errors. Prettier, an opinionated code formatter, ensures that code conforms to an agreed-upon style, reducing the time developers spend formatting code themselves.

Integrated Development Environments (IDEs) like Visual Studio Code, WebStorm, and Atom have become central to modern web development. They offer features like syntax highlighting, intelligent code completion, and powerful navigation tools. Live reloading refreshes the application as code changes are made, providing immediate feedback. Debugging tools embedded within browsers or IDEs, such as Chrome DevTools, allow developers to step through code, examine variables, and set breakpoints to troubleshoot issues effectively.

ECMAScript 6

ECMAScript 6, also known as ES6 and ECMAScript 2015, marked a significant milestone in the evolution of JavaScript. Officially ratified in June 2015, ES6 introduced a bevy of new features, syntax, and methodologies that brought JavaScript closer to the capabilities of more traditional object-oriented programming languages, streamlining how developers could construct robust applications.

One of the most lauded introductions in ES6 is the let and const keywords for variable declarations, providing block-scope variables, unlike the function-scoped var. This addition has helped to prevent common bugs caused by the variable hoisting behavior of var. Arrow functions (=>) also made their debut, offering a more concise syntax and the lexical binding of this, which is particularly useful in callback functions and methods like map, filter, and reduce.

Classes, Modules and Objects

ES6 brought the concept of classes to JavaScript as a syntactic sugar over the existing prototype-based inheritance, making it more familiar to developers from classical OOP languages. These classes support constructors, as well as instance and static methods. Moreover, ES6 modules introduced a standardized module system, allowing developers to organize code into reusable modules which can be imported or exported, thus enhancing code manageability and maintainability.

Object literals were extended to support setting the prototype at construction, shorthand for foo: foo assignments, defining methods, making super calls, and computing property names with expressions. Template literals were another addition, enabling multi-line strings and string interpolation with embedded expressions, significantly improving the creation of strings.

Promises for Asynchronous Programming and More

Promises became part of the language standard, providing a cleaner, more powerful way to handle asynchronous operations compared to traditional callback patterns. A promise represents an operation that hasn't completed yet but is expected in the future, facilitating the management of asynchronous code sequences.

Destructuring allows for binding properties to variables using syntax that mirrors the construction of array and object literals. The spread operator (...) enables an expression to be expanded in places where multiple elements/variables/arguments are expected. Additionally, new built-in methods, data structures like Maps and Sets, and iterators were introduced to handle collections more efficiently.

Compatibility and Transpilation

While ES6 was a significant leap forward, it also presented compatibility challenges with older browsers. This led to the rise of transpilers like Babel, which allow developers to write code using ES6 features and compile it down to ES5 syntax that can be supported by older JavaScript environments.

In conclusion, ECMAScript 6 has been a game-changer for JavaScript development, offering a richer set of language features and an improved development experience. Its advancements have not only made JavaScript more powerful but have also streamlined the development process, establishing JavaScript as a language that can cater to complex and large-scale application development. ES6 set the stage for future updates to the ECMAScript specification, ensuring that JavaScript continues to evolve in step with the needs of modern web development.

JavaScript's trajectory remains steeped in innovation, with emerging trends shaping its future. The adoption of frameworks for building mobile applications, like React Native, is on the rise, blurring the lines between web and mobile development. Serverless architectures are leveraging JavaScript to reduce the complexity of backend infrastructure. Moreover, WebAssembly is opening new doors for JavaScript to perform tasks previously reserved for lower-level languages, enhancing web performance and expanding possibilities for web game development and powerful web applications.

Performance and Optimization

As web applications become more complex, performance optimization is paramount. JavaScript developers are embracing best practices such as lazy loading, which delays loading of non-critical resources at page load time, and memoization to cache function output and save on processing time for repetitive calculations. Tree shaking, a module bundling optimization technique that includes only the code you use, is becoming standard practice with modern build tools. These and other techniques are critical in writing performant JavaScript code that can scale.

The JavaScript community plays a pivotal role in the language's evolution. Open-source contributions and collaborative projects continue to refine and expand JavaScript's capabilities. Resources like GitHub, Stack Overflow, and MDN Web Docs are treasure troves of information, providing developers with documentation, discussion forums, and code repositories. Regular conferences and meetups foster a sense of community and provide platforms for sharing knowledge and networking.

Learning and Career Paths

For those looking to dive deeper into JavaScript or begin their journey, resources abound. Online platforms like freeCodeCamp, Codecademy, and Udemy offer courses ranging from beginner to advanced levels. JavaScript's prominence in the industry means that proficiency in the language is highly valued. From front-end to full-stack development, JavaScript knowledge opens doors to various career paths in tech, making it a wise investment for aspiring developers.

3.4 Webpack

The Basics

The most well-known use of Webpack in the web development toolbox is as a static module bundler for contemporary JavaScript apps. Upon processing an application, webpack creates an internal dependency graph based on one or more entry points. Subsequently, it gathers all the modules required for your project into one or more bundles. The browser can load these bundles, reshaping and packing scripts and files to maximize performance and load time.

Rich feature set that meets different needs during the development process makes Webpack stand out. It handles everything as a module, not just JavaScript, but also CSS, pictures, fonts, and HTML. The loaders and plugins that make up webpack are its fundamental components. Webpack can process various file formats to create modules that can be added to your dependency graph with the help of loaders. Plugins can be used to optimize and manage your bundled assets more thoroughly and provide a great deal of customization.

Webpack's workflow is highly configurable. Developers specify entry points, rules for how to process different modules, which plugins to use, and where to output the final bundles. The tool also offers a powerful feature known as code splitting, which allows chunks of the application to be loaded on demand, thus enhancing the user experience by reducing the initial load time.

Development and Production Environments

Webpack can be optimized for different environments. In development mode, it provides tools such as hot module replacement (HMR), which updates modules in the browser while an application is running without a full reload. This greatly improves the development experience. For production, webpack focuses on minimizing the size of bundles, optimizing load times, and improving the overall performance of the application.

The webpack ecosystem is vast and actively supported by a robust community. The tool itself is open source, and there's a plethora of loaders and plugins available, which extends its capabilities. The community also contributes to a rich set of documentation, tutorials, and third-party tools, making it easier for developers to get up to speed with using webpack.

Webpack has revolutionized the way developers bundle and serve their applications to the web. Its extensive feature set and the ability to handle a wide variety of assets make it a go-to tool for developers looking to optimize their workflow. As web applications continue to become more complex, webpack's role in the development process is likely to grow even more pivotal, making it an essential skill for modern web developers. Whether you're building a large-scale application or a small personal project, webpack provides the features and performance optimizations to ensure your project is built efficiently and runs smoothly.

3.5 PostgreSQL

Introduction to PostgreSQL

PostgreSQL stands out as an advanced, open-source object-relational database system that melds the capabilities of traditional RDBMS with the advancements of object-oriented database systems. Since its inception at the University of California, Berkeley, PostgreSQL has progressed to become a highly favored choice for developers looking for a robust, commercial-grade database solution. With its rich feature set and strong community backing, PostgreSQL has solidified its position as a go-to database for enterprises and startups alike.

Core Features

At its foundation, PostgreSQL prides itself on being ACID-compliant, ensuring that transactions are processed reliably. This compliance is paramount for businesses that require absolute precision and consistency in their operations. PostgreSQL extends its functionality beyond basic data types to support advanced data types like JSON/JSONB, arrays, and hstore, providing flexibility to developers working with diverse data structures. Its performance is further enhanced by an array of indexing techniques, such as expression indexes and partial indexes, which optimize query speeds and database performance.

Security is another cornerstone of PostgreSQL, with robust features that ensure data integrity and protection. These include strong access controls, column and row-level security, and built-in support for SSL encryption, safeguarding data both at rest and in transit.

One of PostgreSQL's most significant advantages is its extensible nature. Users can define their own data types, add custom functions, and even incorporate code written in various programming languages directly into PostgreSQL. This extensibility is a testament to its adaptability, allowing for a high degree of customization to meet specific application requirements.

An example of PostgreSQL's versatility is the PostGIS extension, which equips the database with comprehensive support for geographic objects, turning it into a spatial database for geographic information systems (GIS). This functionality opens doors to a multitude of location-based services and applications, from logistics to urban planning.

Performance

PostgreSQL is renowned for its exceptional performance, primarily due to its implementation of Multi-Version Concurrency Control (MVCC). MVCC allows for high levels of concurrent access to the database, minimizing lock contention and ensuring that readers don't block writers and vice versa. This concurrency model is essential for high-traffic databases where numerous transactions occur simultaneously.

In terms of data redundancy and availability, PostgreSQL offers robust replication features. Streaming replication allows for real-time mirroring of databases, while logical replication provides a more granular approach, replicating only specific tables or databases. These features are vital for creating high-availability environments and for load balancing across servers.

The advanced query planner and optimizer in PostgreSQL tailors the execution of queries to the specifics of the data and the hardware, resulting in efficient query processing. Its query execution engine can handle complex queries and large data sets, which is critical for enterprise-level applications and data analysis.

Advanced Capabilities

For modern applications, full-text search is a critical component, and PostgreSQL's full-text search capabilities are built directly into the database. This allows developers to implement search functionalities without the need for external search engines, streamlining the development process and reducing complexity.

As data grows, managing it efficiently becomes crucial. PostgreSQL's partitioning allows for dividing database tables into smaller, more manageable pieces, while sharding distributes data across multiple machines, allowing databases to scale horizontally. Additionally, foreign data wrappers in PostgreSQL enable the database to interact with other database systems, making it an excellent choice for integrating diverse data sources.

SQL Compliance and Procedural Languages

PostgreSQL boasts high SQL compliance, adhering closely to the SQL standard, which benefits developers with consistent, predictable behavior of SQL queries. This compliance reinforces PostgreSQL's standing as a reliable ORDBMS.

Stored procedures are fully supported in PostgreSQL, allowing for complex business logic to be processed on the database server. It also offers a variety of procedural languages, including

PL/pgSQL, which closely resembles Oracle's PL/SQL, as well as PL/Python and PL/Perl, among others. These procedural languages provide developers with the flexibility to write database logic in a language that best suits their application's needs.

PostgreSQL's robustness and advanced feature set have made it a staple in the arsenal of modern development, particularly for applications where data integrity and reliability are non-negotiable. It is widely utilized in web applications, serving as the backend where complex transactions and extensive data manipulation occur. Data warehousing solutions leverage PostgreSQL for its efficient handling of large volumes of data, while financial systems rely on its transactional integrity and security features to maintain accurate and secure records.

The database also aligns well with modern DevOps practices, supporting continuous integration and continuous deployment (CI/CD) pipelines. Its ability to integrate smoothly into automated workflows and its reliability in production environments make PostgreSQL a natural fit for agile development practices and rapid deployment cycles.

Tools and Interfaces

For database management, PostgreSQL is supported by a range of sophisticated tools and interfaces, such as pgAdmin, a popular open-source administration and development platform. Integration with various programming languages is facilitated through libraries and ORMs, allowing developers to interact with PostgreSQL using the language syntax they are most comfortable with. This seamless integration is a testament to PostgreSQL's flexibility and its community's commitment to providing comprehensive development tools.

Moreover, PostgreSQL's compatibility with containerization technologies like Docker simplifies development and deployment processes. It works hand-in-hand with orchestration systems like Kubernetes, allowing for scalable, distributed systems that are resilient and maintainable.

The PostgreSQL community is a dynamic and vital force in the database's continued success. Developers, contributors, and users actively participate in its development, documentation, and sup-

port, ensuring that PostgreSQL remains on the cutting edge. The annual pgCon conference, alongside numerous other events and meetups, fosters a collaborative environment where knowledge sharing and innovation thrive.

Conclusion

Looking forward, PostgreSQL's adaptability, reliability, and comprehensive feature set position it well within the landscape of future database technologies. Its growing market share and the trend of organizations migrating to PostgreSQL from other systems underscore its rising prominence. Whether it's the demand for high-performance web applications or the need for reliable data warehousing, PostgreSQL is poised to remain a preferred choice for developers and businesses seeking a database solution that can meet and exceed their demands.

3.6 Ruby on Rails

Ruby on Rails Defined

An open-source web application framework created in the Ruby programming language is called Ruby on Rails, or just Rails. By assuming what each developer needs to get started, it is intended to make the development process simpler. With it, developers may accomplish more with less code written than with many other languages and frameworks. Since its introduction by David Heinemeier Hansson in 2005, it has been utilized to create a wide range of online apps, from straightforward to intricate.

Don't Repeat Yourself (DRY) and Convention over Configuration (CoC) are the two guiding concepts of Rails. The DRY concept of software development tries to prevent redundancy by either utilizing data normalization or abstractions to replace program patterns that are repeated too often. CoC indicates that instead of requiring infinite configuration files to specify every little detail, Rails has predetermined conventions about how to execute various tasks in a web application.

MVC Architecture

The model-view-controller (MVC) architectural paradigm, which divides application programming into three interrelated layers, is what Rails is based on. Business logic and database data are represented by the Model layer. The controller's choice to display the data causes the View layer to deliver the data in a specific format. Serving as a bridge between models and views, the Controller layer processes requests that come in from the web browser, manipulates data that comes from the Model layer, and routes the data to the appropriate View.

Rails is known for its rich ecosystem of "gems" – libraries that extend the framework's functionality. This ecosystem is managed by the RubyGems tool, which allows developers to easily distribute and install libraries. Gems can provide anything from authentication systems to integration with payment gateways.

One of the biggest advantages of Ruby on Rails is the speed with which developers can go from concept to prototype. Rails includes a set of conventions that help speed up development. For instance, the Rails Doctrine asserts that beautiful code is better than ugly code, and good conventions can take the place of lots of configuration.

ActiveRecord and Database Integration

ActiveRecord, Rails' Object-Relational Mapping (ORM) system, simplifies data handling. Developers can write database queries in Ruby, rather than SQL, and ActiveRecord will translate these into database commands. This not only makes the code more readable but also allows it to be database-agnostic.

Rails encourages test-driven development (TDD) by including a built-in testing framework. This encourages developers to write tests for their code before writing the code itself, leading to more reliable, maintainable applications.

Rails promotes RESTful design of web applications, which means that applications are designed to respond to actions like GET, POST, DELETE, and PATCH within the stateless protocol of HTTP. This allows for cleaner design of APIs and other web services.

Community and Documentation

The Rails community is one of the framework's greatest strengths. An active community contributes to a wealth of plugins, extensive documentation, and coding tutorials. Furthermore, Rails is well-supported with resources such as screencasts, forums, and numerous conferences worldwide.

In conclusion, Ruby on Rails has established itself as a powerful and practical framework for web development. It prioritizes convention, speed, and maintainability, offering a comprehensive toolkit for developers to create web applications efficiently. With its emphasis on good coding practices and a supportive community, Rails continues to be a popular choice for startups and enterprises alike.

3.7 React

Introduction to React

React is an open-source JavaScript framework used to quickly and dynamically design user interfaces for internet and mobile applications. Facebook created it and continues to maintain it along with a network of independent developers and businesses. Because of its speed, scalability, and ease of use, React has become incredibly popular among developers.

The idea of components lies at the core of the React philosophy. Components are the fundamental building pieces of every React application and are used in its construction. These are reusable, self-contained code segments that control how UI segments look and behave. Because components can be layered inside other components, simple, reusable parts can be used to construct sophisticated programs.

JSX - JavaScript XML

JSX is a syntax extension introduced by React that resembles HTML. JSX is a preprocessing step that enhances JavaScript with XML syntax. With JSX, you can write markup right inside your JavaScript code, which improves readability and simplifies the process of creating components. JSX is a recommended technique in React development, however it's not necessary.

React makes use of the "virtual" DOM (Document Object Model) programming paradigm, in which a library like ReactDOM maintains a perfect, or "virtual," version of a user interface in memory and synchronizes it with the "real" DOM. This procedure is referred to as reconciliation. Rather than reloading the entire DOM structure, it enables React to efficiently update the DOM nodes. This is achieved by altering only those portions of the DOM that have undergone changes.

State and Lifecycle

Property values that are specific to a React component can be stored in an integrated state object. The component re-renders when the state object changes. Building apps with React requires an understanding of component lifecycles, including creation, mounting, updating, and unmounting.

Since version 16.8, React Hooks have made it possible for developers to leverage state and other React features without having to create classes. Function components can be used to "hook into" React state and lifecycle aspects through the use of hooks. They offer a more straightforward API for the React ideas—props, state, context, refs, and lifecycle—that you are already familiar with.

Context API and Redux

React's Context API allows for sharing values like user authentication data across components without having to explicitly pass props through every level of the tree. For more complex state management, especially in large applications, Redux provides a predictable state container which makes it easier to manage the state across the entire application.

React has a rich ecosystem with a wide range of tools that support and enhance its capabilities. Tools like Create React App provide a comfortable setup for developers to get started with a new React application. There are also numerous React developer tools, such as the React Developer Tools extension for Chrome and Firefox, which make debugging and performance tuning easier.

Community and Future Prospects

React has a vibrant community and its ecosystem is continuously evolving with new tools and libraries. The future of React is promising, with ongoing improvements to performance and developer experience. With the rise of server-side rendering and static site generators like Next.js, React is well-positioned to remain a dominant library for building modern web applications.

In conclusion, React's declarative component-based model provides a powerful and efficient way to build user interfaces. Its unopinionated nature gives developers the flexibility to choose the tools and libraries that best suit their project needs, ensuring that React remains a staple in web development for the foreseeable future.

3.8 Redux

Introduction to Redux

An open-source JavaScript package called Redux is used to manage the state of applications. When creating user interfaces, libraries like Angular or React are most frequently utilized with it. Redux is a predictable state container for JavaScript apps that makes state alterations transparent and trackable. It was developed by Dan Abramov and Andrew Clark, drawing inspiration from the Flux architecture and functional programming ideas.

Three key ideas form the foundation of Redux:

1. Single Source of Truth: An object tree contained in a single store contains the state of the entire program.
2. State is Read-Only: Emitting an action or object that describes what transpired is the only way to modify the state.
3. Modifications are Made using Pure Functions: You create pure reducers, which are functions that accept the previous state and an action and return the next state, to define how the state tree is changed by actions.

Actions and Reducers

Actions in Redux are just regular JavaScript objects with a type field to specify the kind of action being carried out. Reducers are simple functions that return a new state result after receiving as inputs the action and the current state. Reducers ascertain the reaction of the state by sending actions to a storage.

The item that connects reducers and actions is called the store. It offers multiple methods, including subscribe(listener), getState(), and dispatch(action). It also keeps your application's state tree up to date. Redux has just one store, however for maintainability and modularity, data handling is divided into distinct reducers.

Integration with UI Libraries

Although Redux is most frequently used in conjunction with React, it may be integrated with any UI layer. The official Redux UI binding library for React is called React-Redux. It offers a Provider component that wraps the connect() function around any nested components, enabling them to access the Redux store.

A third-party extension point is offered by Redux middleware between the time an action is dispatched and when it gets to the reducer. Logging, crash reporting, asynchronous API calls, and more sophisticated synchronous logic, such as conditional dispatching and data validation, are all done with this. Redux side effect management is made easier with libraries like Redux Thunk and Redux Saga.

Redux has a strong developer toolset, including the Redux DevTools that allow for time-travel debugging and state change observation. The community around Redux is vast, with a wealth of resources, middleware, and add-ons developed to ease common tasks and extend Redux’s capabilities.

Best Practices and Patterns

Redux emphasizes several best practices, such as keeping reducers pure, normalizing state shape, and careful usage of selectors for computing derived data. Patterns like Ducks and Redux Toolkit

suggest structuring your reducers and actions in a way that reduces boilerplate and promotes scalability.

Redux revolutionized the way developers handle state in JavaScript applications. Its simplicity and predictability made it an indispensable tool for developers striving for maintainability in large-scale applications. As the complexity of web applications grows, Redux's principles provide a roadmap to managing state in a scalable way, and its vibrant ecosystem continues to innovate, ensuring Redux will remain relevant in the landscape of state management solutions.

4 Building the Portfolio: Selection of Process and Materials

4.1 Insight into The Portfolio

In this section, the essential elements and functions of each project within the portfolio are detailed, offering an organized overview suitable for prospective colleagues or employers. Each project is accompanied by a list of requirements and a brief summary, which outlines its goals, the technologies used, and the strategies applied. This approach provides a clear and comprehensive depiction of technical and problem-solving skills. Additionally, screenshots for each project are included, delivering a visual representation of the work. These images showcase not only the user interface and design choices but also highlight the complexity and sophistication of the developed systems. By presenting these crucial project details, descriptive summaries, and visual examples, the Knowledge Base section serves as a robust reflection of technical abilities and a user-friendly guide to the tangible outcomes of expertise. This method ensures that observers can readily comprehend the range of abilities and practical applications of the work, reinforcing the portfolio as a comprehensive display of a journey in software development.

4.2 Budget app

The budget app is a mobile web app where you can manage your budget: users can add categories, show categories, add transactions, and show transactions for each category. Built with Ruby on Rails and PostgreSQL.

4.2.1 Learning objectives

- Use software package systems with ruby gems.
- Set up the Ruby on Rails platform.
- Recognize the router and RESTful design of Rails.
- To handle requests and render empty views, use controllers.
- Make safe use of the parameters from the browser request.
- Use a preprocessed HTML file that has Ruby code inserted in it.
- When creating shared material, use themes and layouts.
- To keep database schema up to date, use database migration files.
- Put model validations to use.

- Protect the app against n+1 issues.
- Recognize what ORM is.
- Utilize ActiveRecord to write SQL queries.
- Establish relationships between the models.
- Create a web application that needs the user to sign in.
- Use the device's gem to verify identity.
- Apply permission rules to restrict access to webapp resources.
- Describe in writing the reasons for your decision to choose a particular framework for coding.

4.2.2 Description

The goal of the Ruby on Rails project is to create a mobile web application for managing your finances. Figure 1 shows you a list of transactions connected to each category, allowing you to track your spending and identify what you spent and how much. Our plan is to develop a Ruby on Rails application that enables the user to do the following:

- Register and log in, ensuring their data remains private.
- Present fresh transactions connected to a particular category.
- Examine the funds allocated to each area.

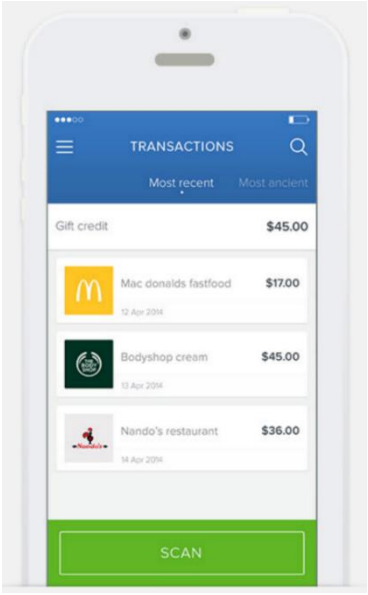

Figure 1: Budget app first page mockup design

4.2.3 Requirements

General requirements:

- Make sure that there are no linter errors.
- Make sure that you used correct gitflow.
- Make sure that you documented your work in a professional way.

Ruby requirements

- Follow our list of best practices for Ruby.

Project requirements

Design

- The following design principles should be adhered to:
- Shades and colors.
- Typography: weight, size, and font face.
- Layout: element composition and interspace.

Interactions

- Splash screen
    - A simple page with the name of your app (yes, you need to choose one), and links to the sign up and log in pages.

- Sign up and log in pages

  - The user should be able to register in the app with full name, email and password (all mandatory).
  - The user can log into the app using email and password.
  - If the user is not logged in, they can't access pages that require the user to be logged in (all the pages described below).

- Home page (categories page)
  - When the user logs in, they are presented with the categories page.
  - For each category, the user can see their name, icon and the total amount of all the transactions that belongs to that category.
  - The program opens the transactions page for the selected category when the user clicks (or taps) on one of the category items.
  - There is a button "add a new category" at the bottom that brings the user to the page to create a new category.

- Transactions page
  - For a given category, the list of transactions is presented, ordered by the most recent.
  - At the top of the page the user could see the total amount for the category (sum of all of the amounts of the transactions in that category).
  - There is a button "add a new transaction" at the bottom that brings the user to the page to create a new transaction.
  - When the user clicks on the "Back" button (<), the user navigates to the home page.

- "Add a new category" page
  - The user fills out a form to create a new category, indicating their name and icon (both mandatory).
  - The user clicks (or taps) the "Save" button to create the new category, and is taken to the home page on success.

    - When the user clicks on the "Back" button (<), the user navigates to the home page.

- "Add a new transaction" page
    - The user fills out a form to create a new transaction with:
        - name (mandatory)
        - amount (mandatory)
        - categories (mandatory at least one)
    - The user clicks (or taps) the "Save" button to create the new transaction, and is taken to the transactions page for that category.
    - When the user clicks on the "Back" button (<), the user navigates to the transactions page for that category.

Testing requirements

- Create unit and integration tests for all the most important components of your RoR application.

Technical requirements

- You should use Postgres as your database.
- You should use devise for authentication.
- You should validate all user input to make sure that anyone with bad intentions cannot compromise your app.
- You can use a view template engine of your choice (.erb, .slim, .haml).
- The project should be deployed and accessible online.
- Your database schema should reflect the following structure (Figure 2):

NOTE: do not change column names visible in the diagram. You need to change the "Entity" name according to the theme you have chosen for your project (please note: "Transaction" is a name already used by ActiveRecord, so using it as a name for your model and table will result in an error).

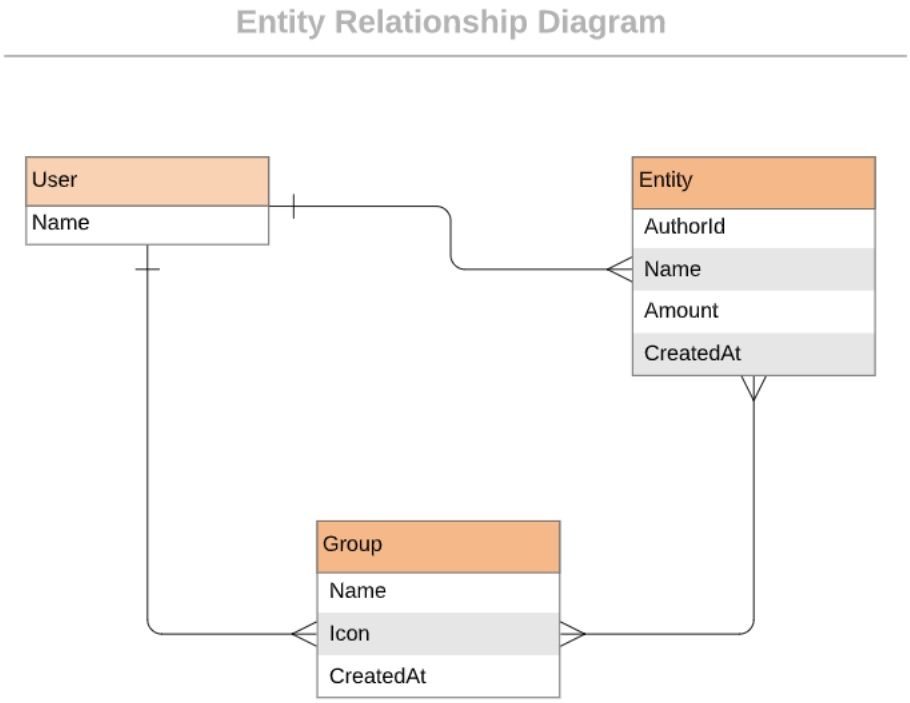

Figure 2: Budget app database schema

4.2.4 Project Insights

Billy Budget is a simple mobile web app that helps you track your money. It shows what you've spent, sorted by type, making it easy to see your spending habits. There are different pages for showing categories, transactions and new transaction (Figure 3).

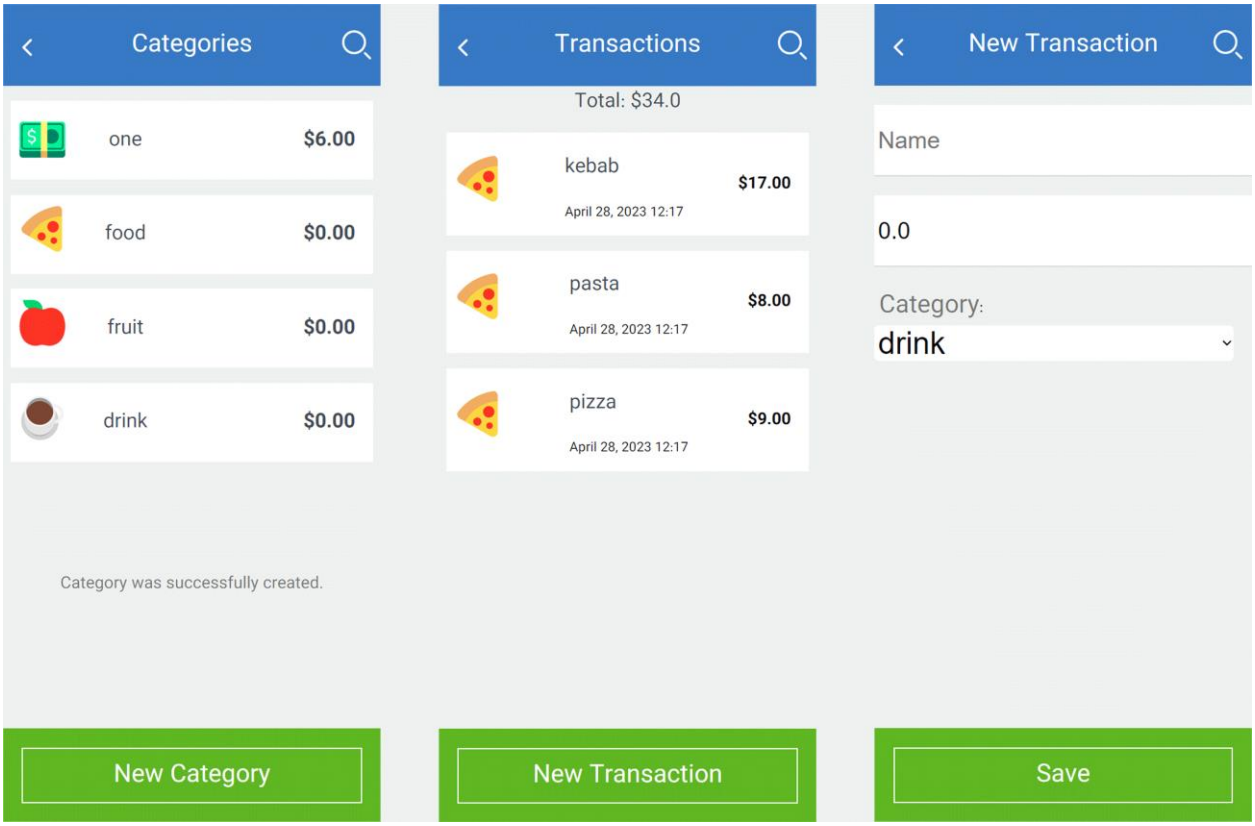

Figure 3: Budget app other pages mockup

The app is made using Ruby, Ruby on Rails, and PostgreSQL. You can add types of spending, see them, add your spendings, and see what you spent under each type.

If you want to try the app on your computer:

1. Make sure you have Ruby on Rails and PostgreSQL.
2. Get the app with: git clone git@github.com:billymohajeri/Budget-App.git.
3. Open the Budget-App folder.
4. Use bundle install to get ready, rails db:create to make a database, and rails db:migrate to update it.
5. Start the app with: rails s.

Future plans for expanding the abilities of the app are add more to the app, like removing types and spendings, making it fit computer screens, and making it look better.

4.3 The To-do list

The To-do list is an app for managing your own daily tasks: users can add a to-do, list all to-dos, mark to-dos as done, and delete to-dos. Built with HTML5, CSS3, JavaScript, and Webpack.

4.3.1 Learning objectives

Build a website based on this design (Figure 4):

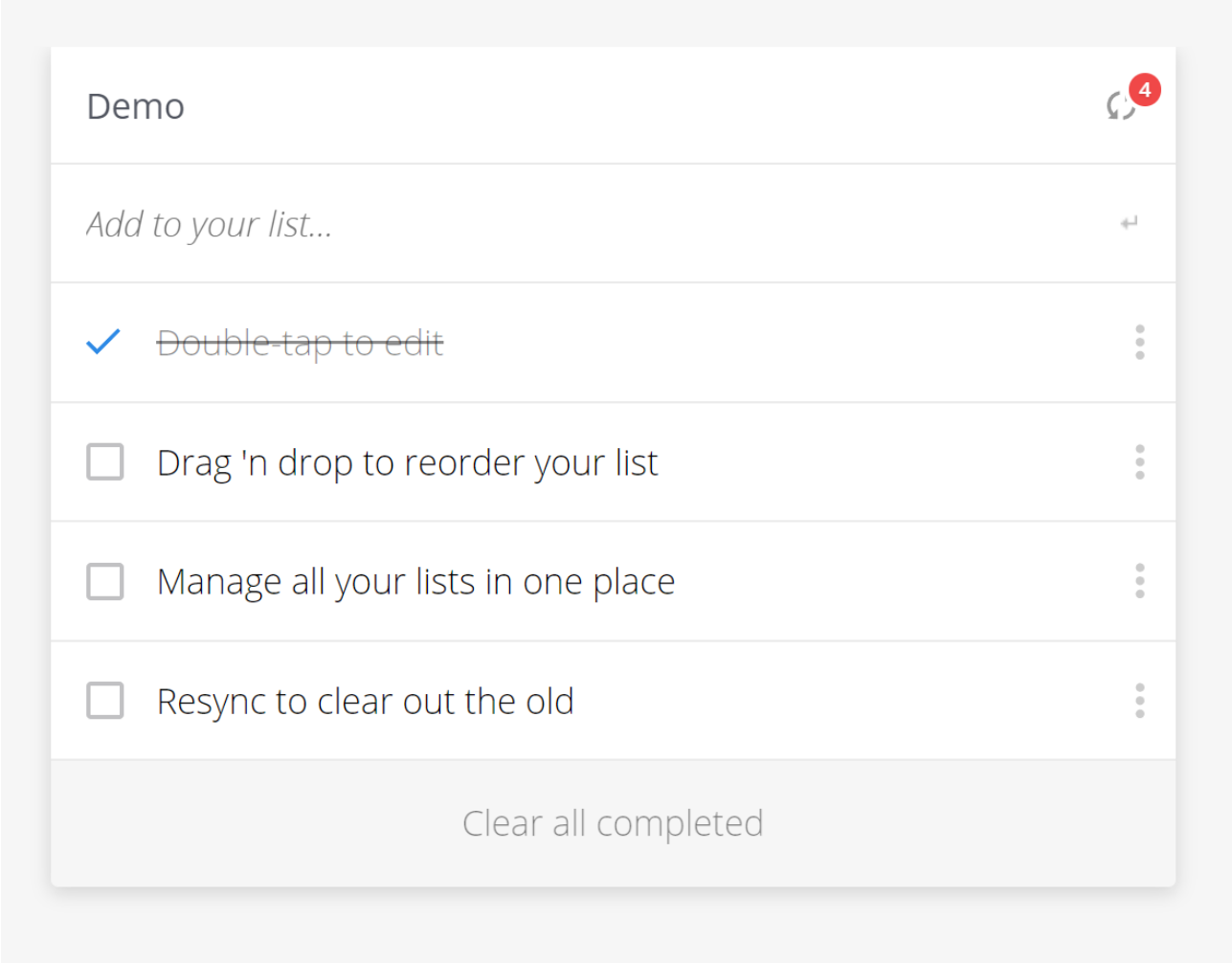

Figure 4: Todo app demo version

Also, an existing version of this app is available in this address.

4.3.2 Description

"To-do list" is an organizing tool for your day. It only makes a list of the tasks you must perform and lets you mark them as finished. You will use Webpack and ES6 to create a basic website that enables you to do that!

How to build the "To-do list" website?

You will build a very simple yet powerful to-do list. It will be inspired by the minimalist website. That website is already offline but you can still play with it thanks to web.archive.org.

The assignment is to create a webpage that exhibits a list that functions and appears like the portion of the minimalist project shown in the image below (Figure 5).

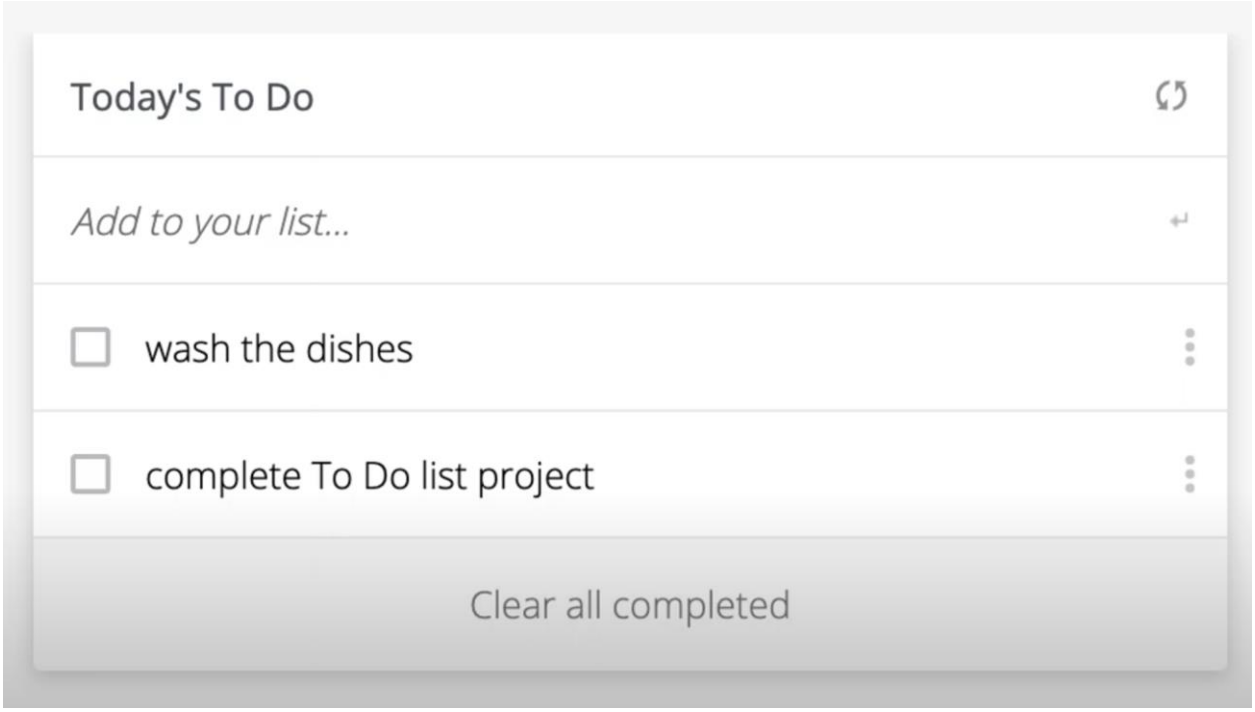

Figure 5: Todo app mockup design

You will need to implement the following functionalities (presented in the picture above):

4.3.3 Requirements

- Adding a new item.

- Removing a selected item.
- Marking a selected item as complete.
- Removing all items marked as complete at once.

4.3.4 Project Insights

In the 'To Do List' project, a straightforward HTML list was constructed to assist users in managing their tasks (Figure 6). The development of this web page utilized technologies such as HTML, JavaScript, CSS, and Webpack. With the aid of the webpack dev server, the page is seamlessly served to users.

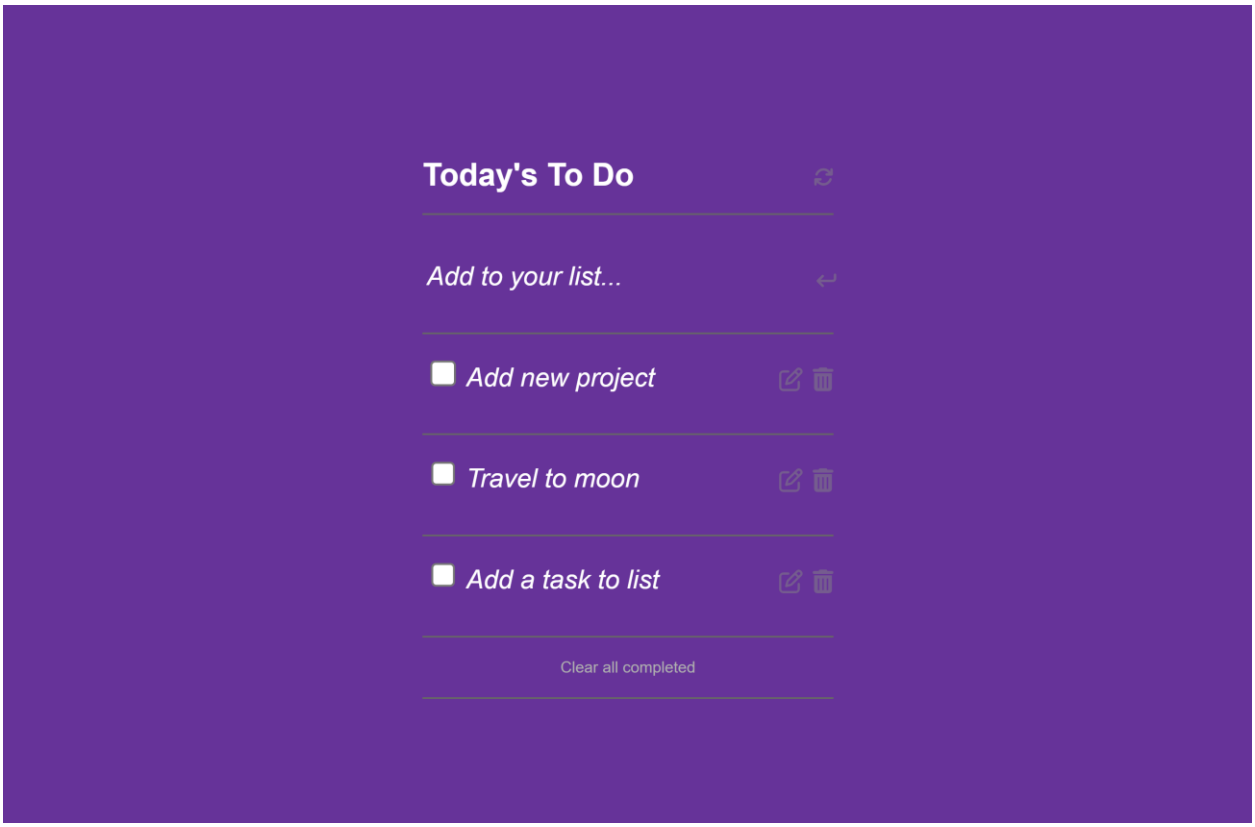

Figure 6: Todo app

If you wish to have a personal, local version of the "To Do List," there are a few steps to follow. Firstly, ensure you have a code editor, preferably VS Code, and an operational web browser. To begin, select a directory on your computer and execute the command cd my-folder, followed by git clone git@github.com:billymohajeri/To-Do-list.git to obtain the project repository. Once retrieved, initiate and install the necessary dependencies using npm init -y and npm i respectively.

After the setup, the project can be launched on your local server by running the npm run start command.

4.4 Leaderboard

The scores that various players have contributed are saved and shown on the Leaderboard. You can enter your score as well. An external API keeps all of the data safe. Constructed using Webpack, JavaScript, CSS3, and HTML5.

4.4.1 Learning objectives

- Understand how to use medium-fidelity wireframes to create a UI.
- Send and receive data from an API.
- Use API documentation.
- Understand and use JSON.
- Make JavaScript code asynchronous.

4.4.2 Description

The webpage for the leaderboard shows the scores that various players have submitted. You can enter your score as well. Because of the external Leaderboard API service, all data is kept safe.

How the leaderboard website is constructed

For this application the priority is to create a working version of the leaderboard that preserves user input. You will build a website according to the wireframe (Figure 7):

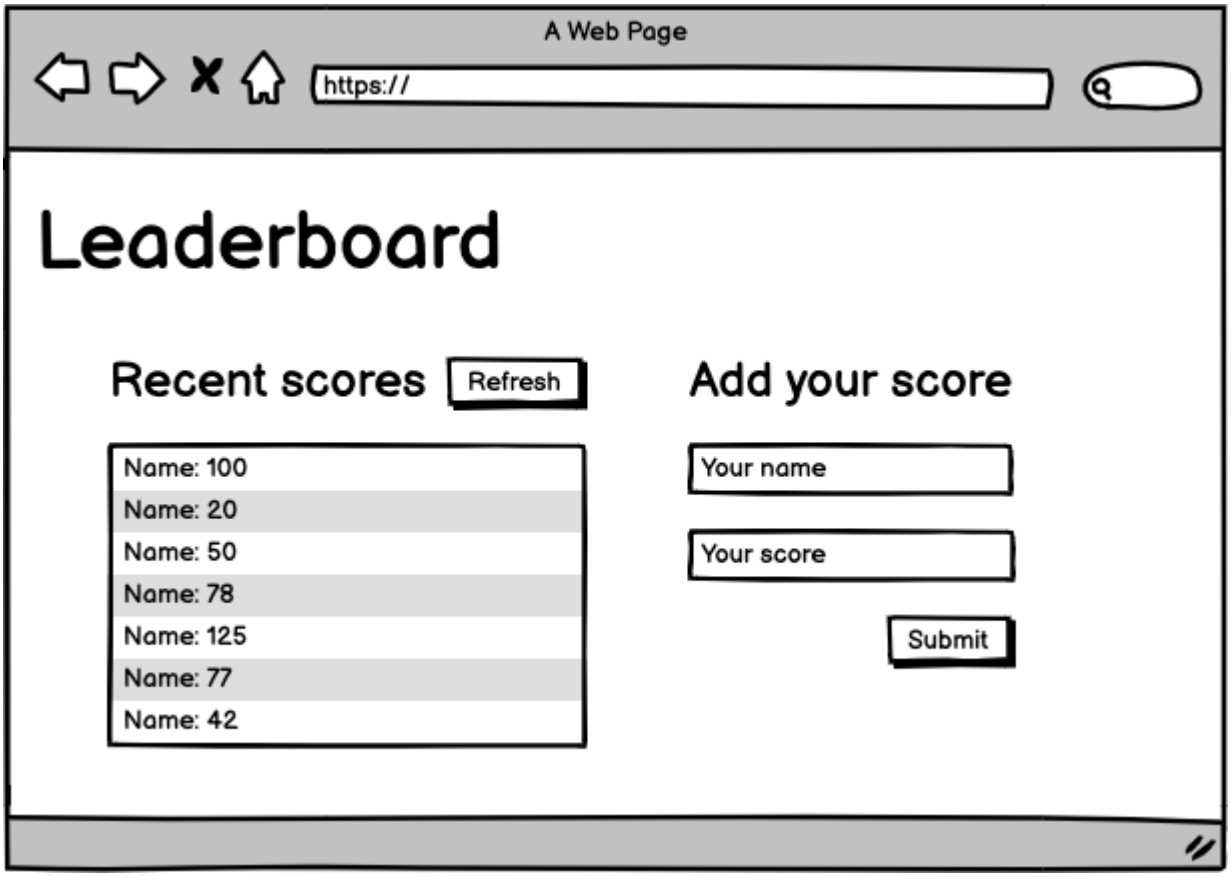

Figure 7: Leaderboard app wireframe

Then, you will connect to the Leaderboard API (as described in its documentation) in order to implement the actions behind the "Refresh" and "Submit" buttons. In the last step, you will play with async & await and add as much styling as possible in the given time.

4.4.3 Requirements

General requirements

- Make sure that there are no linter errors.
- Make sure that you used correct flow Gitflow.
- Make sure that you documented your work in a professional way.

HTML/CSS requirements

- Follow our list of best practices for HTML & CSS.

JavaScript requirements

- Follow our list of best practices for JavaScript.

Project requirements

- Read the Leaderboard API documentation to learn how to use this API.
- Create a new game with the name of your choice by using the API.
- Make sure that you saved the ID of your game that will be returned by API.
- Implement the "Refresh" button that will get all scores for a game created by you from the API (receiving data from the API and parsing the JSON).
- Implement the form "Submit" button that will save a score for a game created by you (sending data to the API).
- Use arrow functions instead of the function keyword.
- Use async and await JavaScript features to consume the API.
- At first you should have a fully working app that uses only basic styles to make the layout work, according to the wireframe.
- When functionality is done, improve the look and feel of the application, adding the styles of your choice.
- Please keep the general layout of the wireframe, this is the only mandatory requirement.
- You can use plain CSS or any CSS framework, it's up to you.

4.4.4 Project Insights

The "Leaderboard" website provides a platform where various players can view scores and also submit their own (Figure 8). This functionality ensures that players are always aware of their rankings and can compete with others. Additionally, the integration of the external Leaderboard API service ensures that all data, such as player scores, remains intact and readily available.

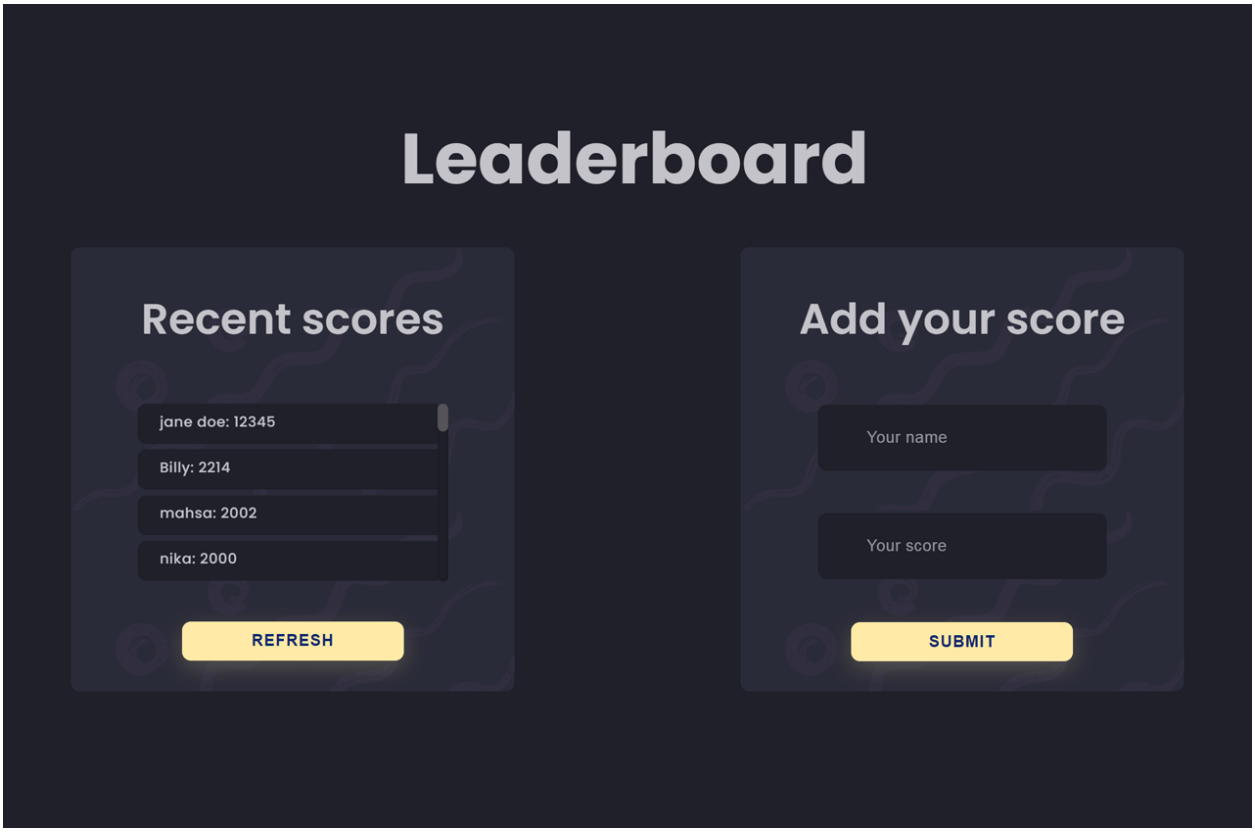

Figure 8: Leaderboard app

This web project was meticulously developed using a tech stack that encompasses HTML 5, CSS 3, JavaScript, and Webpack 5.

Should you wish to explore the project on a more personal level by setting up a local version, the process is quite straightforward. Begin by cloning the project repository using the git clone git@github.com:billymohajeri/Leaderboard.git command. Once this is achieved, build the project using npm run build and then start it with the npm start command. However, before delving into these steps, ensure that you have the necessary tools on hand, which include a code editor (preferably VS Code) and an operational web browser.

4.5 Cottage Booking

Cottage Booking is an app that helps users find cottages and reserve them. The app allows users to add new cottages, remove cottages, make reservations. Built with Ruby on Rails, React, Redux, and PostgreSQL.

4.5.1 Learning objectives

- Apply technical knowledge and skills gained in previous modules in a complex project.
- Understand pros and cons of different approaches of connecting Ruby on Rails back-end with React front-end.
- Understand principles of Ruby on Rails and React frameworks.
- Apply Ruby best practices and language style guides in code.
- Apply RoR best practices and language style guides in code (e.g. thin controllers).
- Apply JavaScript best practices and language style guides in code.
- Apply React best practices and language style guides in code.
- Learn about and practice giving constructive feedback to teammates.
- Perform a code review for a team member.
- Use the "Review change"" feature from GitHub.
- Write clear, professional, and respectful review comments for other team members.
- Explain "why" a change is requested when giving a code review.
- Plan a 2+ week project with no interim Microverse-set milestones and submit it on time.
- Apply knowledge of setting working agreements to set group project teams up for success.
- Independently implement best practices for group agreements to improve teamwork in larger group project.
- Demonstrate ability to apply best practices of communication for resolving teamwork challenges.
- Understand that respect is the foundation of strong relationship-building with teammates.
- Show up throughout group projects as a reliable and committed team member who communicates and manages your time effectively.
- Recognize the importance of investing in good working relationships with teammates.
- Understand principles of strong teamwork (being reliable, committed, and consistent) and how to apply them in group projects.
- Recognize the value of making equal contributions to group projects to produce the best outcome.
- Use empty Kanban board to manage tasks with team and own time on the project.

4.5.2 Description

This project is an app that allows you to schedule a time to test ride a motorcycle. You should follow the given design of the website, but you must personalize the content, i.e., instead of booking an appointment to try a motorcycle, you can build an app to book an appointment with a doctor, or reserve an online class with a teacher. This is highly encouraged since having unique projects in your portfolio will help you stand out while looking for jobs.

If your team is repeating the final capstone project, please use the second version of these requirements and let your reviewers know about it.

How to build the "Book an Appointment" app

The result should look exactly the same (with exception for titles and images) as in the design provided by Murat Korkmaz on Behance (Figure 9).

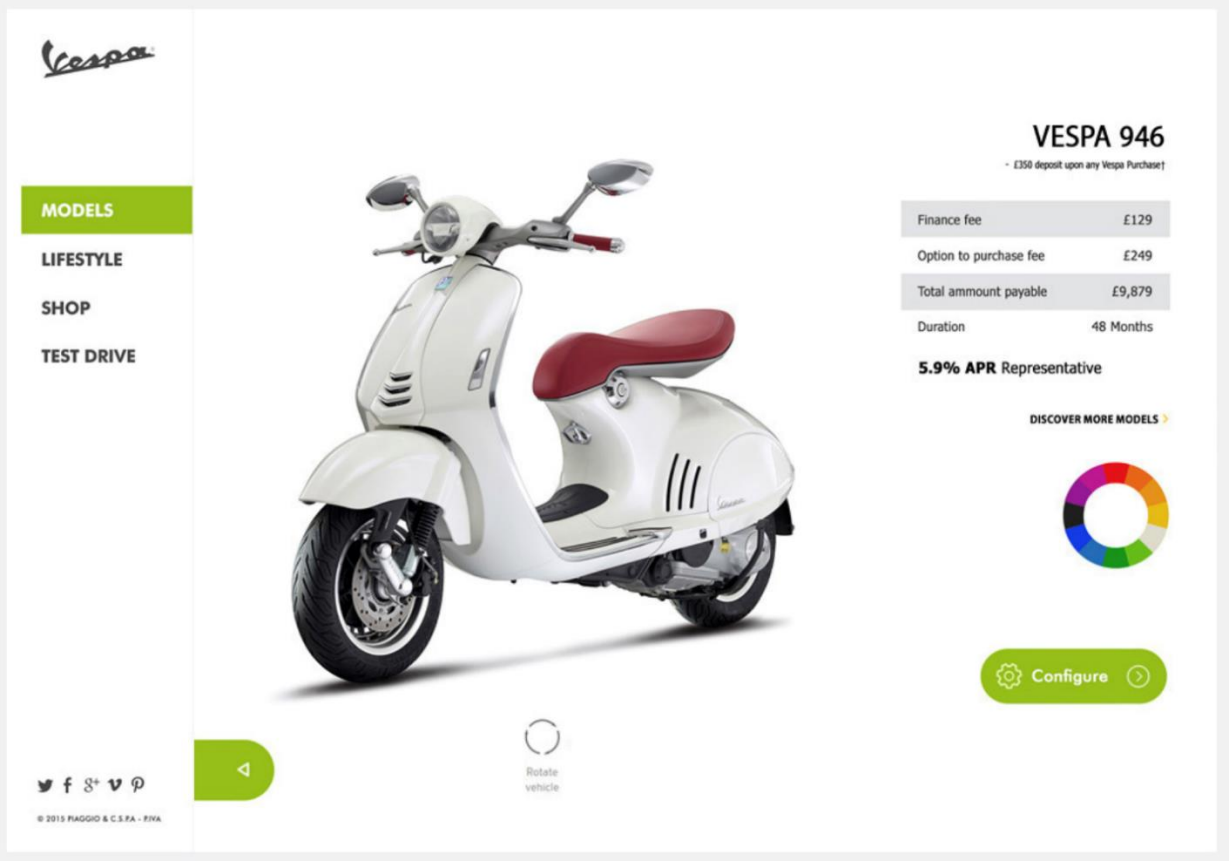

Figure 9: A demo version of Book an Appointment app

The app will have some common interfaces, but depending on your team size it will have a couple of extra ones.

4.5.3 Requirements

General requirements

- Make sure that there are no linter errors.
- Make sure that you used correct Gitflow.
- Make sure that you documented your work in a professional way.

HTML/CSS, JavaScript, and Ruby requirements

- Follow our list of best practices for HTML & CSS.
- Follow our list of best practices for JavaScript.
- Follow our list of best practices for Ruby.

Project requirements

Basics

- You should follow the layout of the provided design. You should change only the titles, descriptions, and photos - in order to create a website about something other than motorcycles.
- Select a theme for your website - is it going to be a website for booking doctor appointments, booking online classes, or something else?

Core features

- The user logs in to the website, only by typing the username (a proper authenticated login is a requirement if your group is made of 5 people).

- In the navigation panel, the user can see links to:
    - Motorcycles/doctors/classes/items that you selected as a theme.
    - "Reserve" form.
    - "My reservations".
    - "Add motorcycle/doctor/class/item that you selected as a theme" (the link is visible to everybody).
    - "Delete motorcycle/doctor/class/item that you selected as a theme" (the link is visible to everybody).
- On the main page, the user can see a list of motorcycles/doctors/classes/items that you selected as a theme (Figure 10).

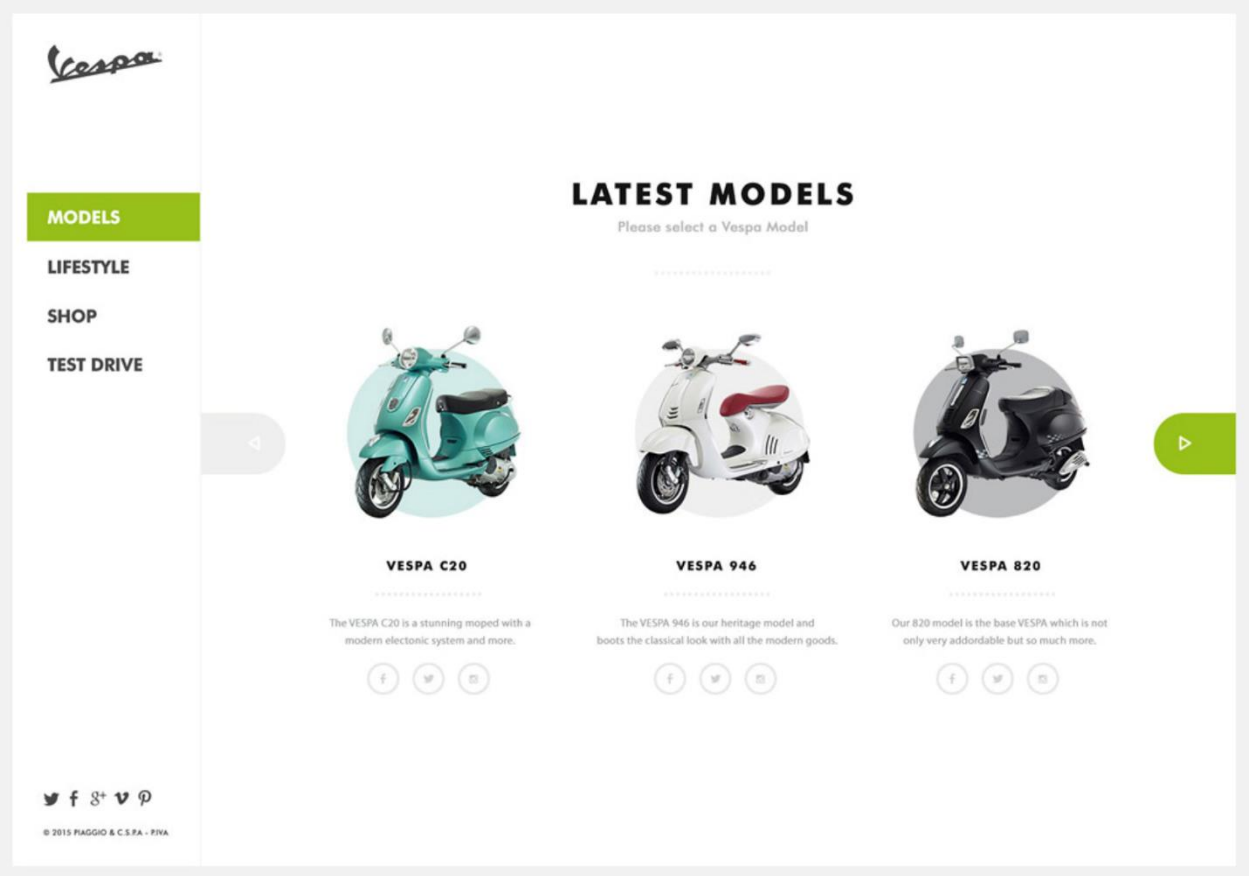

Figure 10: List of items page

- When the user selects a specific item, they can see the [details page](https://www.behance.net/gallery/26425031/Vespa-Responsive-Redesign/modules/173005579) with its full description (skip the "Rotate image" button).
    - In the details page, the user can click the "Reserve" button (in the design you can see the "Configure" button - please replace it with the "Reserve" button) (Figure 11).

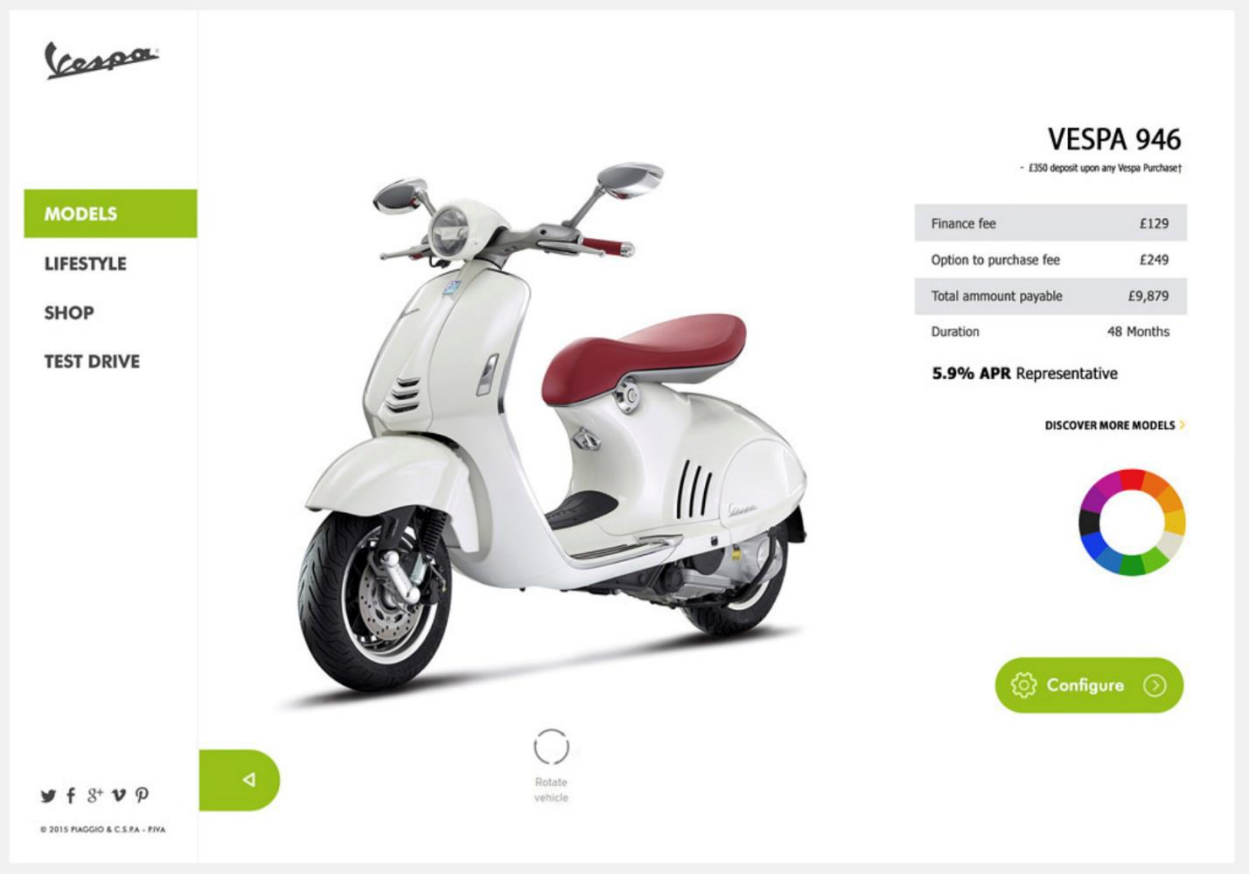

Figure 11: Details of an item page

- When the user clicks the "Add item" link in the navigation panel they can see a form for adding a new item.
- Make the app responsive, creating both mobile and desktop versions.

Additional features

- When the user clicks the "Delete item" link in the navigation panel they can see a list of all items with title and "Delete" button.
    - When the user clicks the "Delete" button, the selected item is marked as removed and does not show on the main list anymore.
- To reserve an appointment, the user has to select a date and city (username and selected item are autofilled).
    - Use the design based on the "Book a vespa test-ride" and add all necessary inputs.
    - The user can also access the "Reserve" page from the navigation panel. In that case only username is autofilled (Figure 12).

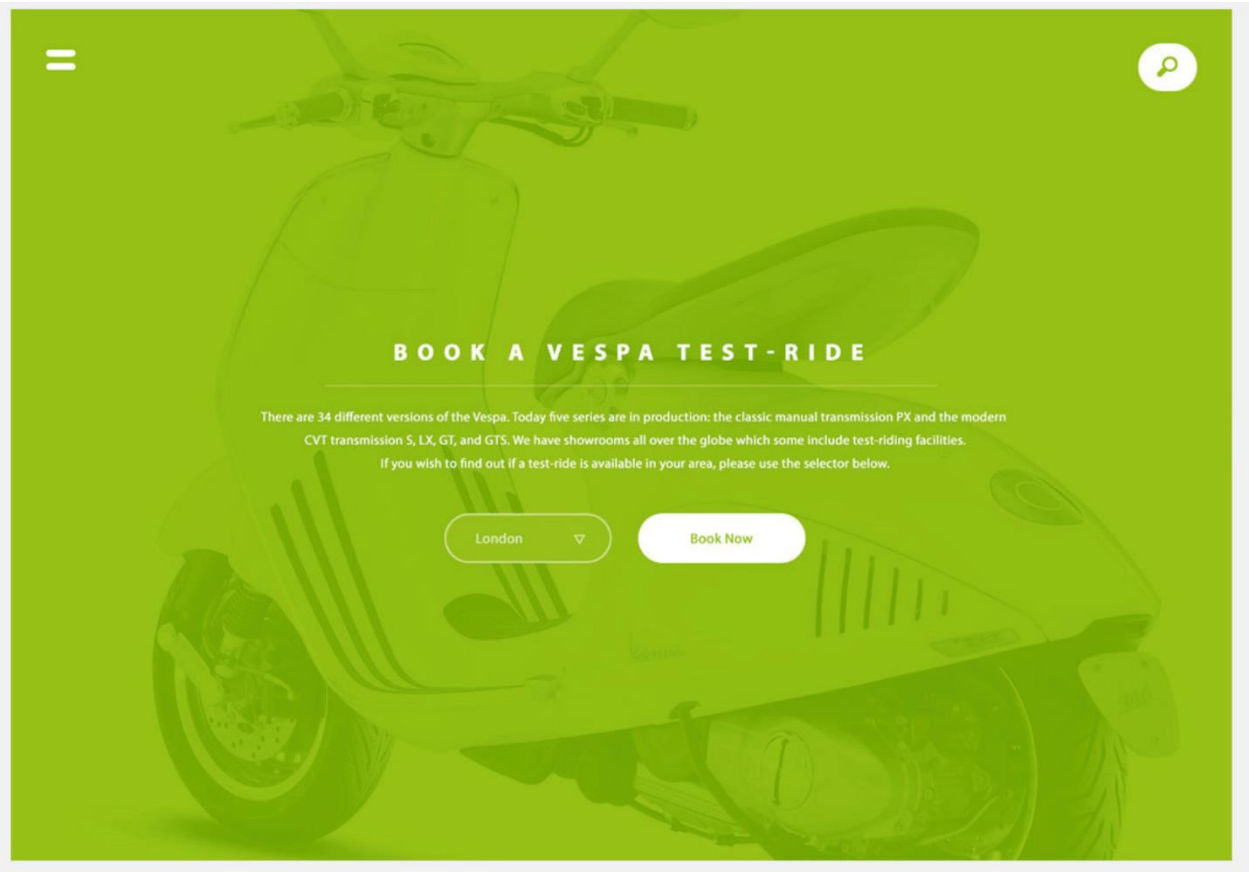

Figure 12: Reserve an item page

- When the user clicks the "My reservations" link in the navigation panel they can see a list of their reservations (with information about item name, date and city).
- Add full documentation for your API.

Additional features

- Implement proper user authentication from the front-end to the server.
- Make sure that the "Add item" and "Delete item" links are accessible only by users who are admins.

Technical set up

- Set up the repository/repositories on GitHub and use Gitflow.
- You should use Postgres as your database

- Use Rails to create backend API.
- Use React & Redux to create frontend UI.
- You can choose if you want to set up your project as two separate apps or as one.
- The Creative Commons license of the design requires that you give appropriate credit to the author. Therefore, you must do it in the README of your project.

Workload distribution

To tackle this challenge, you need to create a Kanban board with a GitHub project that translates the requirements into a set of tasks that you will be able to use to organize your work. You will create your board in a separate activity.

You will be working in this way:

- You need to translate the above requirements into Kanban cards.
- All tasks should have time estimates and your job is to distribute them between team members in a fair way.
- The common tasks will be divided among all of you or completed as a team (pair programming).
- All tasks should be based on the cards from your Kanban board.
- If you discover a new task that needs to be done, create a new card for it.

4.5.4 Project Insights

The Cottage Booking App offers users a platform to explore and reserve cottages (Figure 13). This particular repository focuses on the backend of the application, but a corresponding frontend repository is also available for those interested.

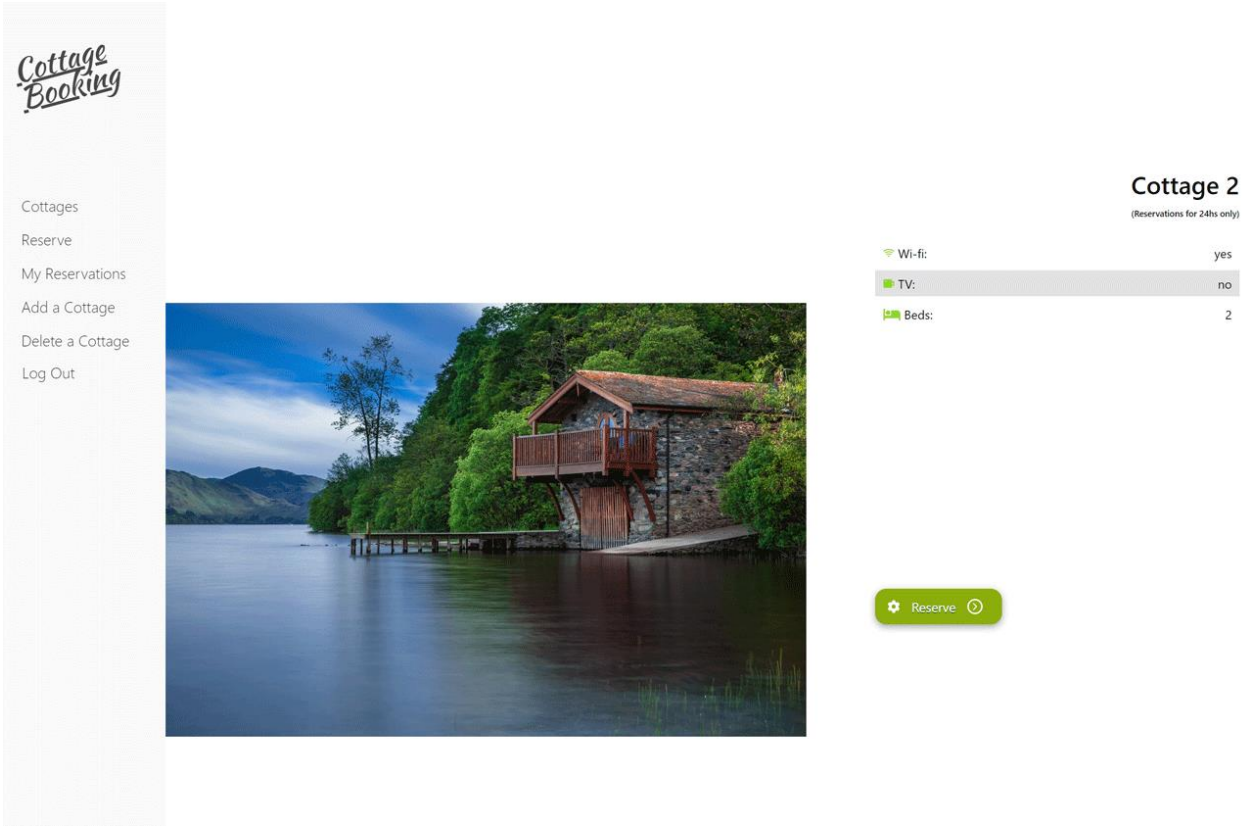

Figure 13: Cottage Booking app first page

The application is built on a robust technology stack. For the client-side, it utilizes a suite of tools including Ruby, Rails, React, Redux, Bundle, WebPack, and NPM. On the server side, the application is powered by Rails, while PostgreSQL is employed for data management. Key features of the application encompass user registration, adding or deleting cottage listings, and managing reservations.

For those tech-savvy individuals eager to get a local version of the app running, here's a brief guide. First, you'll need to have Ruby on Rails and PostgreSQL installed on your machine. Once that's sorted, you can clone the backend repository using the command git clone git@github.com:billymohajeri/Final-Group-Capstone-Backend.git and the frontend with git clone git@github.com:billymohajeri/Final-Group-Capstone-Frontend.git.

After downloading, you can set up the backend by navigating to its directory (cd Final-Group-Capstone-Backend) and running a series of commands to install the necessary gems, create the database, run migrations, seed the database, and start the development server. Similarly, the frontend can be set up in its respective directory with appropriate npm commands.

This application was a team effort. I, contributed along with Gaurav Gangwar and Botlhale Setou.

The Kanban methodology was employed for efficient project management, and access to the board is available here. The initial state of the Kanban board can be viewed at this link. For context, the project team comprises three dedicated individuals.

Looking ahead, there are ambitious plans to further refine the app. The roadmap includes enhancing the user interface, incorporating a comment feature, and refining the user authentication and authorization mechanisms.

5 Description, Analysis, and Results of the Portfolio Components

5.1 Mobile and Desktop Mockups

In this chapter, an exploration of the visual frameworks underlying the portfolio is presented through a detailed examination of the mockups. Additionally, an in-depth analysis of the portfolio's structure will be conducted, elucidating the design rationale and functionality of its distinct components.

The development of the professional portfolio commenced with a mobile-first approach, acknowledging the increasing prevalence of mobile browsing. This strategy was implemented to ensure optimal user experience across various devices, adhering to the principles of responsive design. By prioritizing mobile platforms from the outset, the portfolio was designed to adapt fluidly to a range of screen sizes, ensuring both functionality and aesthetic integrity are maintained.

The visual and interaction design of the portfolio were meticulously planned and executed using Figma, prior to the coding phase. This process was part of a collaborative project at Microverse, a well-regarded coding boot camp attended in the previous year. The Figma mockups served as a blueprint, detailing the layout and features for both mobile and desktop views (Figure 14, 15), and they were instrumental in streamlining the development phase.

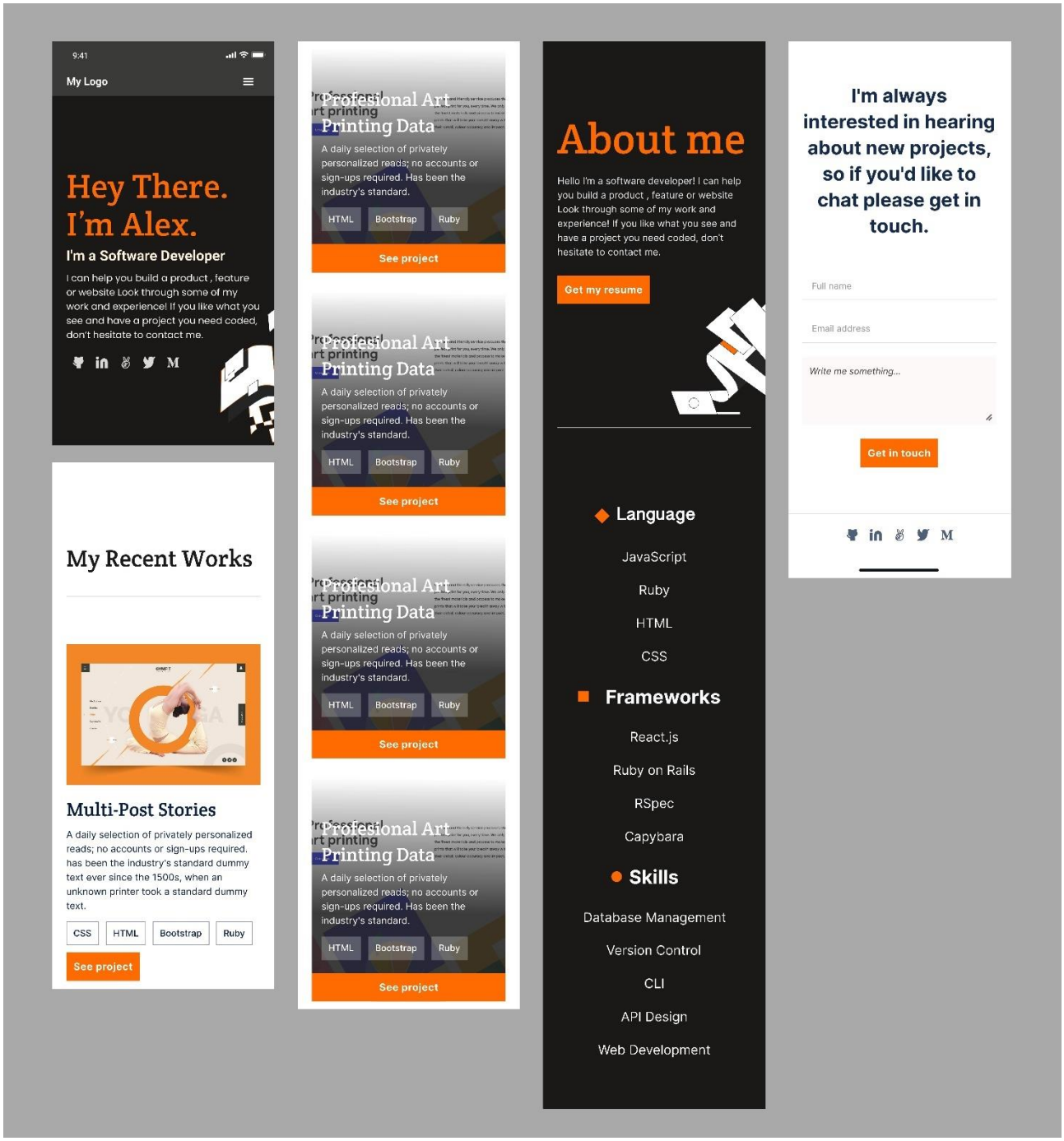

Figure 14: Mobile view mockup of the portfolio showcasing the responsive design

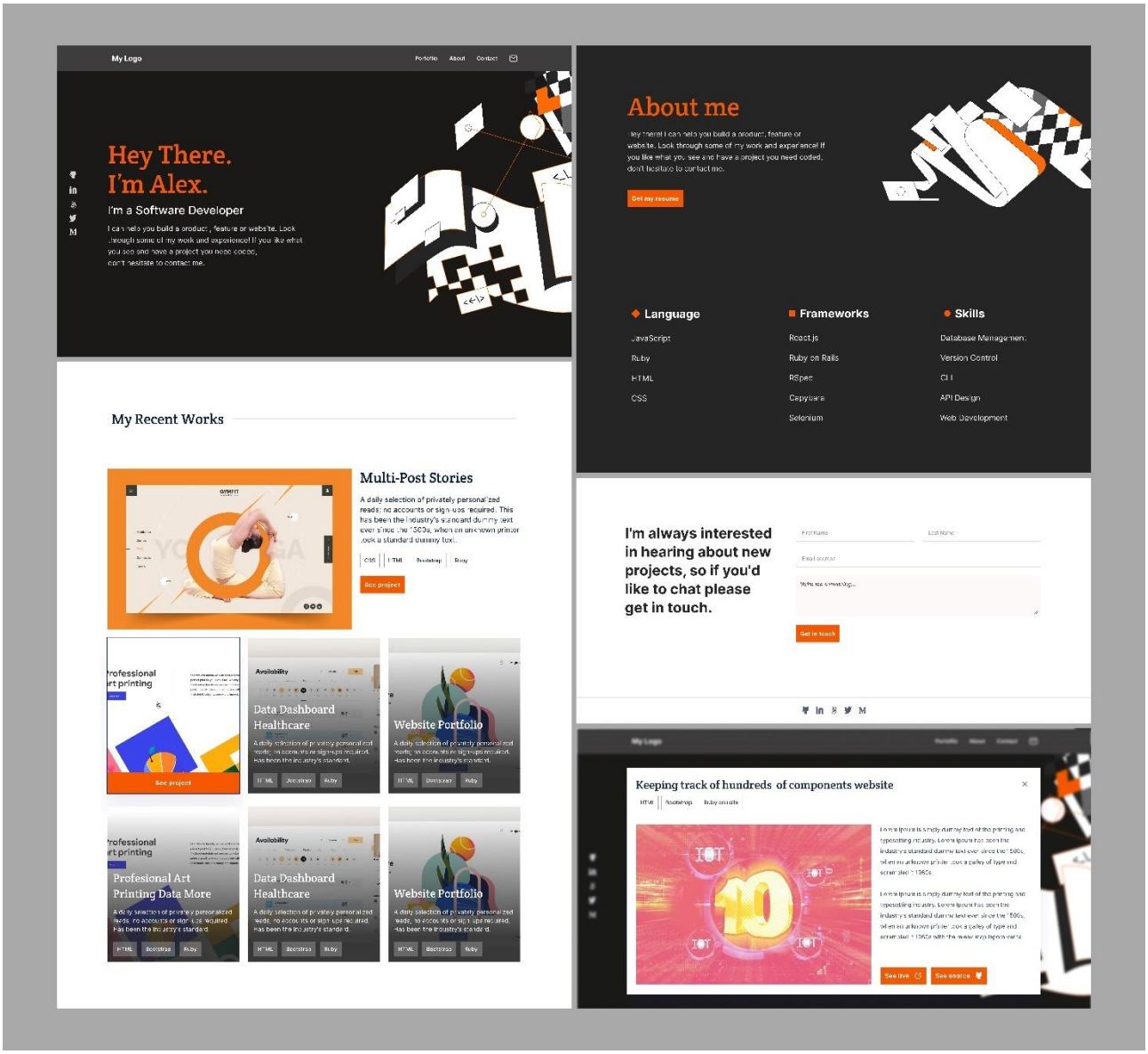

Figure 15: Desktop view mockup illustrating the adaptive layout for larger screens

These mockups not only guided the structural development of the portfolio but also ensured that the user interface was intuitive and engaging across different browsing contexts. The use of a design system and components during the boot camp facilitated a seamless transition from design to development, maintaining consistency in style and interaction patterns.

In the following sections, each component of the portfolio—ranging from the main page to the project showcase—is dissected to detail their purpose, the rationale behind their design, and the impact they have had in engaging the audience and achieving the portfolio's objectives. The analysis is supported by user engagement data and feedback, providing a comprehensive overview of the portfolio's effectiveness as a professional tool.

5.2 Detailed Exploration of Design and Structure

This professional portfolio is a curated collection showcasing expertise and accomplishments in the field of web development. It is structured to provide a comprehensive overview of capabilities and experiences, inviting viewers to engage with and connect to the displayed work. The portfolio is divided into four main components: the main page, the portfolio section, the 'About Me' page, and the 'Contact Me' section.

Main Page

The main page serves as a gateway to the professional persona showcased in the portfolio. It offers a succinct introduction, emphasizing areas of specialization and core competencies (Figure 16). This page strategically includes social media links to enhance reach and foster professional connections across various platforms. Both design and content are tailored to immediately engage visitors, establishing a distinct professional brand identity.

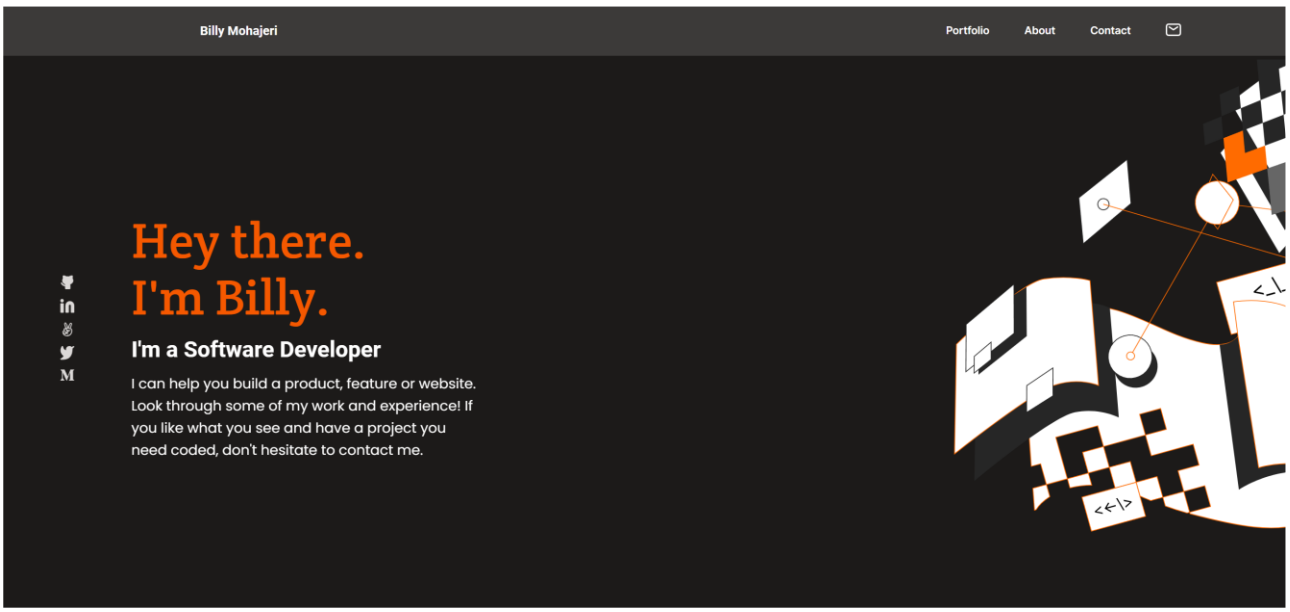

Figure 16: Main page of portfolio

Portfolio Section

This segment showcases selected projects, arranged to ensure that the most recent and significant work captures immediate attention (Figure 17). Projects are listed with the latest one featured prominently, displayed in a larger format with detailed information to emphasize its relevance and

demonstrate current skill levels. This approach serves to highlight growth and proficiency, while ensuring that the content remains fresh and aligned with current industry trends.

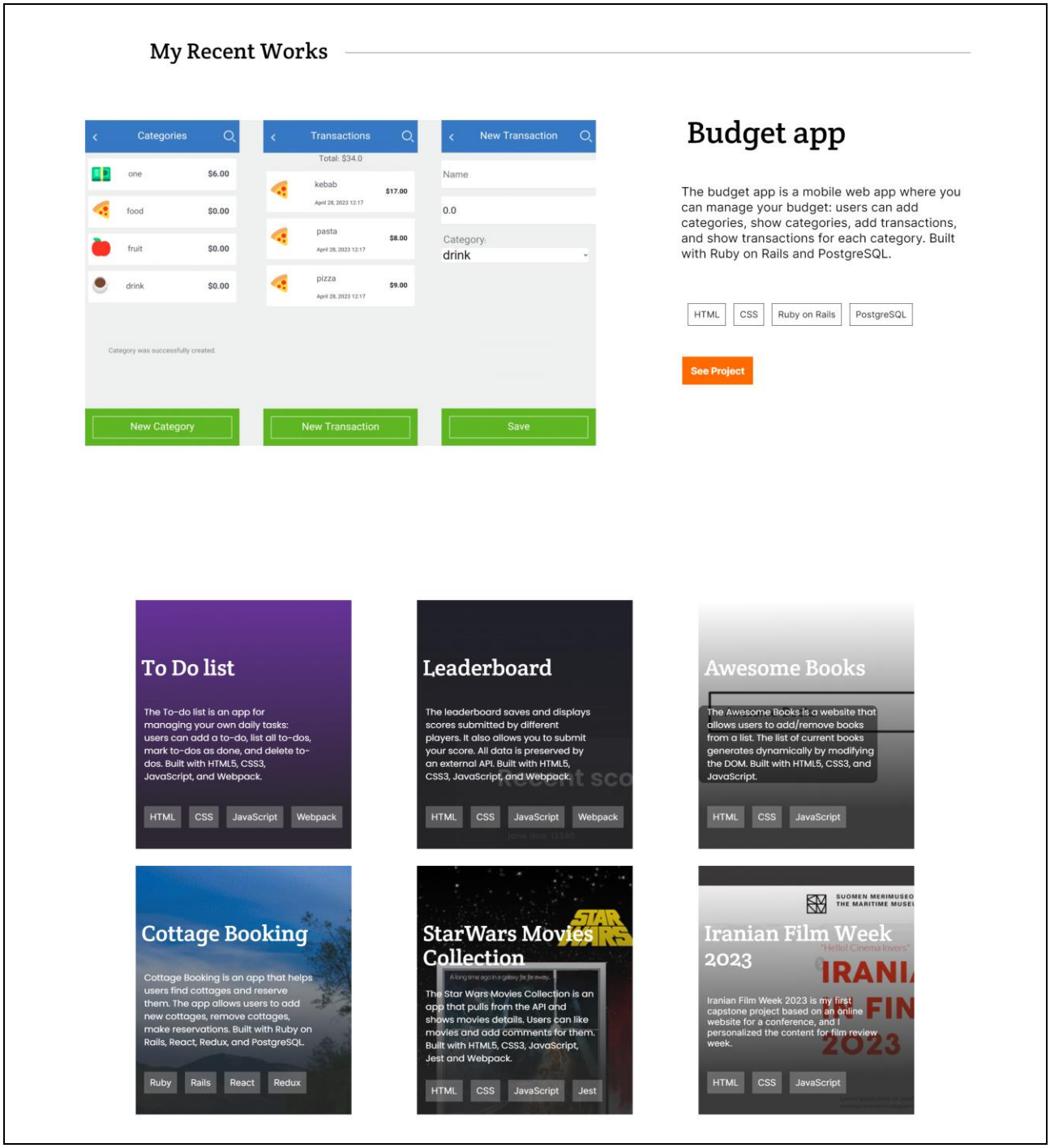

Figure 17: Portfolio section, showcasing the projects

About Me Section

The 'About Me' page offers an in-depth look into the professional journey and skill set of the individual. It presents a narrative that encapsulates experiences, technical proficiencies, and soft skills (Figure 18). The inclusion of an option to download a resume provides a tangible resource for potential employers or clients, reinforcing the portfolio's role as a professional touchpoint.

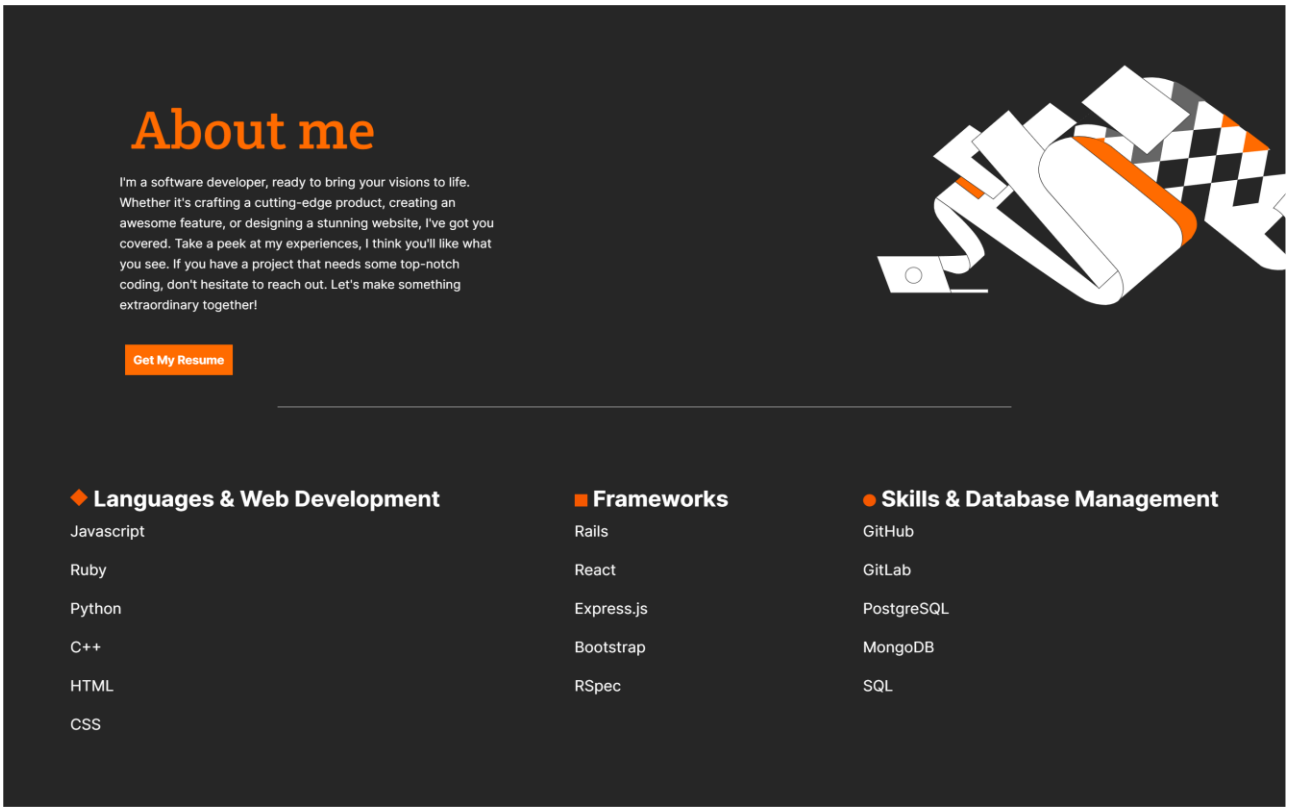

Figure 18: About me section

Contact Me Section

The portfolio is designed to convert viewers into connections, and the 'Contact Me' section is pivotal to this conversion (Figure 19). It encourages direct engagement through a contact form, making it easy for visitors to reach out with inquiries or opportunities. The inclusion of social links here acts as a reminder of alternative communication channels, ensuring accessibility and openness to dialogue.

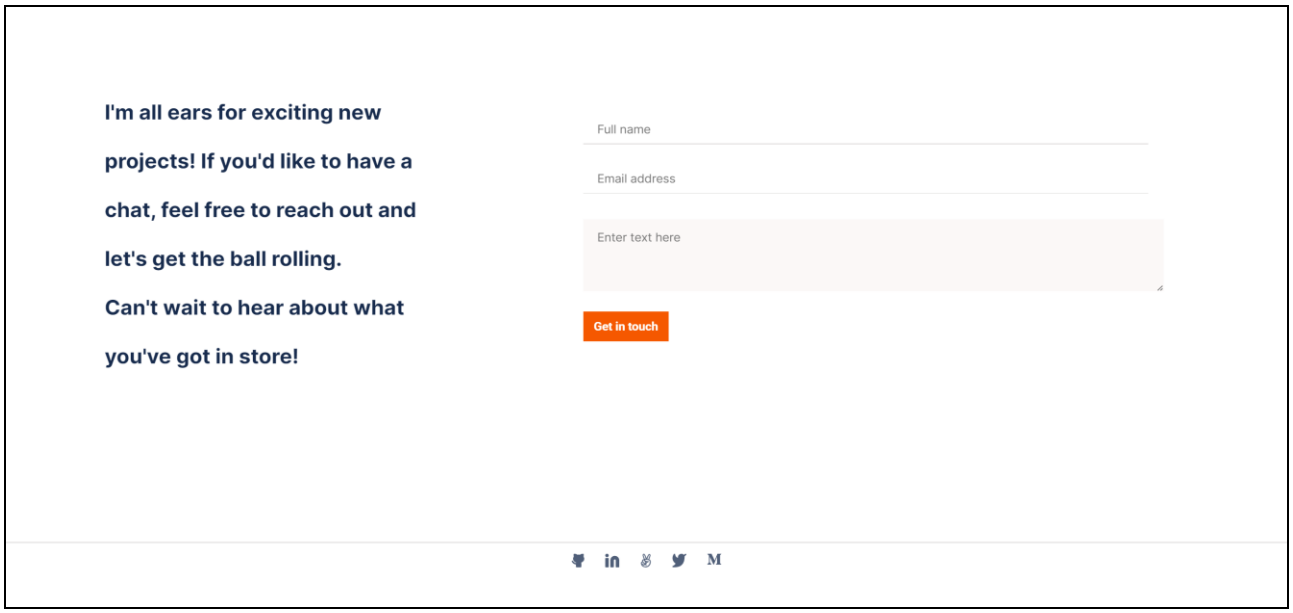

Figure 19: Contact me section

Analysis and Results

The organization of the portfolio components is strategically planned. Each section contributes significantly to a narrative that progresses from a broad introduction, through a detailed exposition of skills, to a call to action. The user-friendly layout, enhanced by responsive design, ensures accessibility across a variety of devices, thereby broadening the portfolio's reach.

The outcome of this meticulously crafted portfolio is a comprehensive digital presence that effectively communicates a professional narrative. Site analytics indicate a high user engagement, particularly in the detailed project descriptions, suggesting that the portfolio section successfully captures and retains interest. Feedback received through the contact form has been overwhelmingly positive, highlighting the portfolio's clarity and ease of navigation as key factors for user engagement and service inquiries.

In conclusion, the portfolio is designed not just to narrate a professional journey but also to engage a wider audience, ultimately serving as a conduit for potential collaborative opportunities. The portfolio has met its objectives, as evidenced by user engagement metrics and direct feedback.

6 Conclusion and Discussion

6.1 Addressing the Primary Research Question

In evaluating the effectiveness of the portfolio-based approach compared to traditional methods, this portfolio distinctly demonstrates a more dynamic and comprehensive way of showcasing software development skills. Unlike traditional resumes, the portfolio provides a holistic view of abilities, encompassing not just theoretical knowledge but also practical application and problem-solving skills in real-life scenarios. This finding aligns with the primary research question, confirming that a portfolio-based approach offers a more accurate and complete representation of a developer's capabilities.

6.2 Connecting to Secondary Research Questions:

The diverse range of technical skills developed and showcased in the portfolio directly answers the second research question. Each project within the portfolio was carefully selected to demonstrate a broad spectrum of abilities, from coding and design to project management, reflecting the depth and breadth of skills developed.

The portfolio's role in contributing to professional learning and growth, as explored in the third research question, is evident in its design and feedback mechanism. The interactive nature of the portfolio, combined with the feedback received from users and peers, fosters an environment of continuous learning and skill enhancement. This approach has led to the portfolio becoming a living document that evolves in line with professional achievements and market trends.

The inclusion of an online gallery feature and the portfolio's function as a dynamic curriculum vitae not only provide a transparent and quick way for potential employers or partners to assess qualifications but also showcase a commitment to lifelong learning and adaptation to new technologies.

The portfolio's ability to facilitate direct engagement and constructive criticism, and its role as an interactive platform for networking and professional development, further emphasizes its contribution to career advancement and personal brand building. This aspect of the portfolio reflects a

unique professional style and a creative approach to problem-solving, which distinguishes it from traditional resumes and aligns with the professional development insights sought in the research questions.

6.3 Reliability Assessment

Generalization Limitations:

The effectiveness of the e-portfolio in showcasing software development skills, as demonstrated in my thesis, must be contextualized with an understanding of its generalization limitations. The findings are based on a specific collection of projects and methodologies within my portfolio and may not universally apply across different software development scenarios. The uniqueness of these projects and their contextual development environments are crucial factors that could influence the broader applicability of these conclusions.

6.4 Research Ethics

Copyright and Intellectual Property:

Ethical considerations, particularly concerning copyright and intellectual property, were paramount in the creation of my e-portfolio. All displayed content is either my original work or used in compliance with copyright laws, ensuring that all intellectual property rights are respected and upheld.

Ethical Data Collection Practices:

For any data collection involving individuals, ethical standards were strictly followed. This included obtaining informed consent and guaranteeing the confidentiality and anonymity of the participants, thereby adhering to the highest standards of research ethics.

Connecting Theory and Practice

The practical findings from the analysis of my e-portfolio align with the broader theoretical perspectives on the effectiveness of portfolio-based assessments in skill demonstration. This alignment suggests that the insights gained from my portfolio are not only based on personal experiences but also resonate with the established understanding in the field of software development.

Conclusion

In conclusion, this thesis demonstrates the effectiveness of a portfolio-based approach in assessing software development skills, as compared to traditional methods. It effectively showcases the key skills and knowledge areas developed, while also underlining the portfolio's significant role in fostering professional learning and growth.

The analysis acknowledges the limitations in generalizing these findings across all software development scenarios, ensuring a balanced perspective. Ethical considerations have been carefully respected particularly in terms of copyright protection and data collection practices, reinforcing the research's validity.

Ultimately, the portfolio stands not just as a showcase of technical expertise, but also as a dynamic tool for career progression and personal brand enhancement. It represents a significant stride in understanding and utilizing e-portfolios in software development, contributing valuable insights and methodologies to the field.

This comprehensive study, while presenting a specific perspective, opens avenues for further research, particularly in exploring the broader applicability of portfolio-based assessments in diverse software development contexts.

References

Bala, S. S., Adlina, W. F., Mansor, W., Stapa, M., & Zakaria, M. H. (2012). Digital portfolio and professional development of language teachers. Procedia – Social and Behavioral Sciences, 66 (2012), 176–186.

Bransford, J., Brown, A., & Cocking, R. (Eds.). (2000). How people learn: Brain, mind, experience, and school. Washington, D.C.: National Academy Press.

Price, L. (2014). Modelling factors for predicting student learning outcomes in higher education. In D. Gijbels, J. Richardson, & V. Donche (Eds.), Learning patterns in higher education in the 21st century: Dimensions and research perspectives (pp. 56–77). London: Routledge.

Richardson, J. (2000). Researching student learning: Approaches to studying in campus-based and distance education. Buckingham, UK: SRHE and Open University Press.

Denton, D. W. (2012). Improving the quality of evidence-based writing entries in electronic portfolios. International Journal of ePortfolio, 2(2), 187–197.

Gagné, R. M., Wager, W. W., Golas, K. C., & Keller, J. M. (2005). Principles of instructional design (5th ed.). Belmont, CA: Wadsworth/Thomson Learning.

Maher, M., & Gerbic, P. (2009). E-portfolios as a pedagogical device in primary teacher education: The AUT University Experience. Australian Journal of Teacher Education, 34(5), Article 4. Retrieved February 27, 2013 from http://ro.ecu.edu.au/ajte/vol34/iss5/4

Firssova, O. (2006). E-portfolio as a coaching support tool for workplace learning of teachers. In ePortfolio 2006 conference papers. Oxford, UK. Retrieved February 27, 2013, from http://www.epforum.eu/sites/www.epforum.eu/files/ePortfolio%202006.pdf

Herman, C., & Kirkup, G. (2008). Learners in transition: the use of ePortfolios for women returners to science, engineering and technology. Innovations in Education and Teaching International, 45(1), 67–76.

Price, L., & Kirkwood, A. T. (2014). Using technology for teaching and learning in higher education: A critical review of the role of evidence in informing practice. Higher Education Research and Development, 33(3), 549–564.

Daunert, A. L., & Harteis, C. (2014). Pre-service teachers' perspectives and practices in utilizing ubiquitous technologies for academic-oriented learning and knowledge management. In J. E. Pelet (Ed.), E-learning 2.0 technologies and web applications in higher education (pp. 254–272). Hershey, PA: IGI Global.

Brown, J. O. (2011). Dwell in possibility: PLAR and e-portfolios in the age of information and communication technologies. International Review of Research in Open and Distance Learning, 12 (1). Retrieved April 3, 2012, from www.irrodl.org/index.php/irrodl/article/view/917

Duncan-Pitt, L., & Sutherland, S. (2006). An introduction to the use of e-portfolios in professional practice. Journal of Radiography in Practice, 3(2006), 69–75.

Johnsen, H. L. (2012). Making learning visible with ePortfolios: Coupling the right pedagogy with the right technology. International Journal of ePortfolio, 2(2), 139–148.

Malita, L. (2009). E-portfolios in an educational and occupational context. Procedia Social and Behavioral Sciences, 1(2009), 2312–2316.