Bricked Up

Francesco Schenone Sebestyen Deak Kacper Grzyb Ignad Bozhinov
Leonardo Gianola

17-12-2024

1 Introduction

The goal of the introduction is to let the readers (the professor and TAs) know the topic of your work and the main takeaways of it. The introduction should be broad enough to understand the document without reading it and specific enough to let the reader know: *If you are interested in this topic, you should read this work.*

In the context of the Web Technologies course, the introduction should clearly describe:

- Motivation: what problem does this work try to solve (and why is it important)
- Project: clearly describes the topic the group chose to work with
- Contributions: main takeaways that readers will get from reading this work

Remember to keep this and all other sections within the page and column limits. It is your responsibility to describe first the most important and interesting aspect of each section. That way, you can leave behind non-interesting and repeated information more easily.

Length. Half a column.

2 Front-End

This project's first task is to build your application's front-end side. This section should clearly describe the technical implementation of the work put into building the front-end:

- Technically describe the use of HTML 5: which HTML tags do you use, where, and why
- Technically describe the use of CSS: why and how you use CSS (including interesting selectors/declarations and how it is incorporated in the application)
- Technically describe the use of JavaScript: why and how you use it in your application (including interesting behaviors and how they are incorporated into the application)

Resources. Lectures 1 to 3.

Length. 2 columns.

3 Resource Management

Since we chose Laravel as our MVC framework, before we could define a resource management system in our project, we first needed a database. Our database solution of choice ended up being Supabase with PostgreSQL. We settled on it because it is very easy to set up, provides a generous free plan, has an intuitive UI, and can be easily scaled up by upgrading the database plan.
With the database set up, we created a schema for our application using draw.io (the ERD diagram can be found in the Appendix) and started writing Laravel migrations to implement the database. The central two models of our database are User and Set (stored in *users* and *sets* tables respectively) , the ladder storing all of the most important data about a LEGO set. The rest of the models within our database center

around adding additional typechecking or information to the sets, such as the set's price records.
As of the current state of the project, we populate the database from multiple different sources, and using multiple tools, but this could be streamlined if this application were to evolve.

- **Seeders** - We defined some basic database seeders for including static data such as the testing admin account (to be removed for production) or set availability types
- **Python Web Scrapper** - A simple python console program utilizing *Selenium* to scrape Brickeconomy for set themes and subthemes
- **Playwright Web Scrapper** - The main tool for obtaining the current prices of the sets within our database, an implementation of *Playwright* that scrapes eBay for price records
- **Admin Upload Data Page** - Albeit a temporary solution, for now the main tool for adding new sets into the database

Adhering to the MVC framework, we hydrate our views with data by the use of Laravel Controllers. Almost every single page has its own controller, so that the data can be custom formated and optimized to the needs of that specific page. Our controllers perform different CRUD operations, and some of them can only be accessed by the admin user, as to comply with the project's requirements of user roles and application functionality. The operations include:

- **Account CRUD** - Before a user is logged in (checked by Laravel Breeze), they are only able to see our landing page with the ability to create an account. Most of the account management functionality was already pre-provided for us by Laravel Breeze, which made the development process a lot smoother, as we had to either user the pre-existing controllers or recycle their functions. Most note-worthy, the *RegisteredUserController* manages user account creation, and the *PasswordController* manages user authentication. The rest of the controllers within the Auth directory are used within the Settings page for editting the account details and deleting the account itself.
- **Set Details CRUD** - The admin-only Upload Data page allows the admin to upload CSV files that contain information about the set they want to add into the database. We created a small, initial dataset for our application, since we knew adding sets would require a lot of moderation. The data sanitization and creation is handled by the *FileUploadController*.
- **User Favourites CRUD** - In the settings page, a user can select their favourite sets, themes and subthemes from all of the available ones in the database. The controller responsible for this functionality is *SettingsController*.
- **User Inventory CRUD** - From the settings page, a user is able to add a set to their own set Inventory, where they can see a summary of all of the sets they own. This is handled by the *InventoryController*.
- **User Dashboard Layout CRUD** - The *DashboardController* is responsible for both providing the data for the dashboard view, as well as saving the user custom created dashboard within the Edit Dashboard Layout page.

4 Authentication and Authorization

This project's third and final task is to incorporate authentication and authorization capabilities. This section should clearly describe:

- Authentication: The different users of the system and how it is implemented
- Authorization: Summarize the access of the different users in the system and how it is implemented
- Role table: Include a role table associating actions over the system (you can think of them as use cases) and users that can perform these actions.

Resources. Lecture 7.

Length. 2 columns.

5 Conclusions

The goal of the conclusion is similar to the introduction: it summarizes the work itself and the takeaways a reader should take when reading this work. However, it can use the information presented in the work to be more specific than the introduction.

In the context of the Web Technologies course, the conclusion should clearly describe:

- Summary: summary of the work and main takeaways. Also include a class diagram of the system (the models).
- Future Work: interesting directions on how the presented work can evolve in the future (it may be the starting point to choose individual extension topics)

Length. Half a column.