Bricked Up

Francesco Schenone Sebestyen Deak Kacper Grzyb Ignad Bozhinov
Leonardo Gianola

17-12-2024

1 Introduction

The LEGO investment market has emerged as a profitable opportunity, with an average return of 11% [1]. However, the current process for analyzing prices, trends, and data on LEGO sets needs to be more cohesive and efficient. To address this, *Bricked Up* centralizes key information, providing interactive graphs, historical price comparisons, and tools to explore investment opportunities efficiently. This report outlines the development of the software, inspired by the Bloomberg Terminal, and its role in simplifying LEGO set investments for users.

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 latter 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 Scraper** - A simple python console program utilizing *Selenium* to scrape Brickeconomy for set themes and subthemes
- **Playwright Web Scraper** - 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, this is currently 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 formatted 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 editing 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 significant 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 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 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

As said in the Introduction, in our Web Technologies Project we aimed to aid the people who want to get into investing in the world of LEGO. Achieving this required us to build and develop a system that can track and display the prices and trends of this market and show it to the user through our intuitive UI. We worked in PHP with the Laravel Framework, utilizing different libraries and Javascript for ease of use and a better experience. The heart of the application just

reads from the database and using the data it shows the bare prices of sets, some information about them and basic graphs. This itself wouldn't be special, but the individual extencions take it to a whole another level.

Individual Extention Topics:

- Graphs and Alerts: Intuitive, Trading-Like Graphs and the possibility to get notified about price changes.
- Two-Factor Authorization: Better security for the User's profile.
- Web Scraper: Allowing the application to not only work with a static database, but update it when needed.
- Dashboard Customization: Enabling the user to create the Home Page that perfectly fits them.
- User Inventory and AJAX Requests: Giving the user the ability to follow their favourite sets, and upgrading the UI to be more responsive

References

[1] Dmitry B. Krylov, "LEGO investing as an alternative asset class: Annualized returns of 11%," *ScienceDirect*, 2021. Available at: ScienceDirect Article.