STAGE 4 Checkpoint 2 Project Report:

Introduction
Our project involved creating a web application designed to assist users in discovering creative ways to use their Lego Bricks. By helping users find new purposes for bricks that may not be part of any specific model, the app encourages creativity and exploration. We think that we succeeded in this purpose.

We were able to use 5 main database tables to create our web application:

Users- kept track of the Users credentials, his/her lego parts in the inventory, builds working on and favorite builds/ parts.

Parts - This table was important to identify the lego parts, and their respective colors. The lego parts are the core component of our project and almost every query is accessing this table.

Builds-This table is also one of the important tables in our databases as it holds all the builds of our database.

Theme- This can help users search for builds and provides us with a system of grouping builds.

Suppliers-This is an important table to understand the suppliers and the prices they provide for each part and build.

We have other weak entities to support the relationships between these tables and contain important functionality for our code such as build_pricing, build_pieces etc.

Our main pages are:
All parts page - We have made an all parts page which has a list of all the parts, from here we can add multiple parts into our inventory, favorite them use search operations to find some parts as well. We have an interesting feature which allows the user to take a picture of the part they are trying to find and we use APIs to get the part that we are searching for.
All builds page - We have made an all builds page which has a list of all the builds with the search features implemented such that users can search by theme and build name and get its description. We can then add the build that we are currently working on to our builder.

Inventories page- This is an important page as it allows us to favorite certain parts and remove parts from our inventory.

Builder Page- This is our main functional page which uses our most important queries. In this page we use the current build the user is working on and the parts in his/her inventory and find the missing parts, lowest possible cost of each missing part and total remaining cost for the build. This is important as it allows users to see how much he/she needs to invest more in this build and how much he/she can save using already existing parts.

Login Page-We also maintained a login page in order to improve the authentication and UI experience of our Web application.

Questions:
Please list out changes in the directions of your project if the final project is different from your original proposal (based on your stage 1 proposal submission).

The final project diverged slightly from our initial proposal to accommodate practical constraints and user feedback. Initially, we aimed to enable users to input Lego bricks via a custom search feature or by uploading images. As we progressed, we found that the image-based brick recognition feature was more intuitive and user-friendly. Consequently, we prioritized refining this functionality, while maintaining focus on the manual search feature initially planned.

Another adjustment was the omission of the Amazon Pricing API integration. While the original proposal emphasized real-time pricing for missing Lego bricks, technical and logistical challenges in obtaining reliable API data led us to use pre-aggregated cost data from existing datasets. This change ensured consistency and usability without compromising the user experience.

I think there are a few changes that we weren’t able to implement that we mentioned in stage 1, this is the Build to Brick Compatibility Algorithm that determines whether a brick fits into a specific Lego build. But on the other hand we are able to compare which suppliers have parts and what is the price of each part and choose the part with the lowest cost and display that.

Discuss what you think your application achieved or failed to achieve regarding its usefulness.

The application effectively achieved its goal of providing users with a structured, creative platform to repurpose Lego bricks. Users could efficiently manage their inventories, explore potential builds, and generate cost-effective shopping lists for missing components. Also helps users search for specific parts that they need and update their inventories accordingly.

However, we did not fully deliver on the social interaction features, such as enabling users to comment on one another's profiles. While the groundwork for this functionality was laid, we chose to focus on completing core features that directly impacted the application's primary objectives. Despite this limitation, the platform remains highly useful in fostering resourcefulness and creativity among users.

Discuss if you change the schema or source of the data for your application

Our original schema was based on Rebrickable datasets, supplemented with Kaggle data for additional attributes like ratings and difficulty levels. We preprocessed the data in order to fit it to our schema. This involved cleaning and normalizing part identifiers across datasets to ensure compatibility and reduce redundancy.

Additionally, we moved away from storing PNG images locally, as originally planned, and instead utilized URLs from the datasets. This adjustment reduced storage requirements and improved overall performance. These changes reflect a practical approach to managing and integrating data while maintaining application functionality.

We focused on only using 3 main suppliers for our dataset who are Amazon, ebay and Lego this helped maintain the list of suppliers as they had a large quantity of parts.

Discuss what you change to your ER diagram and/or your table implementations. What are some differences between the original design and the final design? Why? What do you think is a more suitable design?

The original design did not follow all normalization principles. The final design however ensured BCNF compliance through elimination of redundancies and ensuring all dependencies relied on candidate keys. The final design used index strategies that were not in the original design. These indexes improved query performance for aggregations and joins.

The final design is more suitable because of the balance of normalization and query performance. The final design is more scalable by way of normalization. It is also more efficient than the original design by way of the indexing strategies.

Discuss what functionalities you added or removed. Why?
Explain how you think your advanced database programs complement your application.

The main functionality is what we mentioned above is the same as the original proposal.

Advanced database techniques, such as stored procedures and triggers, were instrumental in automating critical functions and enhancing performance.

Stored Procedure:

For instance, We used stored procedures for the main functionality of our builder page. This stored procedure helped in showing the current build the user is working on, its required parts as seen from missing parts in the inventory and the total cost of these parts and the total build price.

We also used another stored procedure just to give information about the most used parts across all builds as a statistic. This might give the user an idea about the most frequent parts in builds.

Triggers:

We have implemented triggers in our part addition functionality. When there is an already existing part in our inventory we update just the quantity. We used a trigger for this functionality.

We also used a trigger for removing an item from our favorites page when we remove it from our inventory as well.

Transactions:

We have used a serializable(isolation level) transaction to stop multiple users from overwriting the user's table when they are creating an account at the same time.

We also had another transaction that takes place when a user adds a part from all parts to their inventory - in order to prevent a Write Read conflict. This can happen with the builder where a user might have added a new part to inventory but the builder is using the older version of the data and hence gives inaccurate results.

These database-level optimizations ensured data integrity and significantly improved the application's responsiveness, particularly in handling complex queries related to build suggestions and part recommendations.

Constraints: We have used many primary and foreign key constraints as can be seen in our DDL commands and ER diagram.

Each team member should describe one technical challenge that the team encountered. This should be sufficiently detailed such that another future team could use this as helpful advice if they were to start a similar project or where to maintain your project.

- Aryan:One of the main challenges that we faced was while hosting the databases on GCP. It took really long to create the databases and it presented a lot of issues to run the queries and the indexings on the database. Ultimately we had to switch to a local MySQL server.
- Suraj:One other challenge that we faced was integrating API calls on a local server. It was difficult to implement it and a proxy had to be used for the queries to work properly.
- Reece: Generating accurate shopping lists was challenging due to fluctuating prices. We addressed this by relying on average prices from our datasets, which provided a stable and reliable alternative.
- Ryan: Integrating the image recognition API presented difficulties due to rate limits. Implementing caching mechanisms minimized redundant API calls and improved overall efficiency.
- Aryan: Standardizing inconsistent naming conventions across datasets was a major hurdle. This was addressed by preprocessing data with regular expressions, ensuring uniformity and accuracy in searches.

Are there other things that changed comparing the final application with the original proposal?

Aside from feature adjustments, we transitioned from local storage of part images to using URLs, which improved application performance and scalability. We added a login and a user authentication page to the database and also wanted to show some statistics about some parts and some builds. We added the total build cost and cost of missing components to the builds to the builder which helps the user understand how much he/she needs to invest in completing the build or how much he/she can save using already existing parts.

Describe future work that you think, other than the interface, that the application can improve on

Future iterations could benefit from integrating machine learning into the Build to Brick Compatibility Algorithm to improve part substitution suggestions. Incorporating real-time pricing APIs and expanding community features, such as trading or collaborative builds, would also enhance the user experience. Also we would like to improve the user experience by suggesting some alternatives to existing and required lego parts which might be useful in the lego builds. Also we would like to implement the review features that we had mentioned previously in Stage 1.

Describe the final division of labor and how well you managed teamwork.
The division of labor adhered closely to our initial plan, with each team member taking responsibility for their assigned features. Weekly check-ins and clear milestones ensured steady progress. Collaboration was smooth, with all members contributing to the application’s success and assisting one another when challenges arose. Whenever we had a problem we asked the mentor and worked on improving the Web application step by step.