Final Project Report

Team Name: JeremyRenner
Sneha Singh, Aarul Dhawan, Abhi Ramakrishnan, Smriti Srinivasan

Major Changes from Stage 1

For our project, our team stayed close to the original vision outlined in Stage 1 while making adjustments as needed during the development process. Our original idea was to create a platform that allows users to explore and compare the impact of natural disasters on countries' economies and social metrics over time. Beyond this, we added state-level metrics for the United States, a page featuring global disaster statistics, and a feature that allows users to save and reload previous visualizations. These enhancements were made to improve the user experience and usefulness of the web application while remaining consistent with the initial concept for the project.

Achievements and Usefulness

This application achieves its intended purpose. It allows users to quickly visualize and analyze disaster impact data globally as well as in U.S. states, including historical trends beginning as early as 1960. The application offers CRUD operations for managing saved graphs, as well as keyboard functionality for filtering. Users can create, read, update, and delete custom visualizations based on their preferences and needs. The platform is unique because it blends historical disaster events with real-world economic impact through growth metrics, something that is not very common but is inherently useful for understanding the potential impact of disasters on a country. Overall, the application provides an intuitive and informative user experience.

Changes to Schema, ER Diagram, and Table Implementations

As mentioned earlier, throughout the development of this web application, we made necessary changes to the database design compared to the earlier stages of the project. We added a SavedGraphs table to support the graph-saving functionality and a Logs table to track user actions with triggers. These additions were not included in the initial proposal, but the improved schema allows for auditability and customization, making the design better aligned to provide an improved user experience.

With regard to the data sources, no significant changes were made to the original global data and extrapolations. However, for the United States, the databases we were using did not originally contain sufficient information. To address this, we decided to create a new schema specifically for U.S. data, where we measure per-state growth of industries including agriculture, manufacturing, and real estate per year, along with overarching state-level growth metrics such as GDP growth and personal income growth. We also integrated a more extensive list of natural disasters as reported by FEMA, including events such as fires, droughts, snowstorms, and tornadoes, tracked per year. This allowed us to provide detailed,

per-state information for the United States, expanding the depth and usefulness of the platform for U.S. disaster analysis.

Advanced Database Features and Their Application Utility

We implemented the required advanced database programs with SQL, not using any ORM libraries. We developed a stored procedure, CompareCountryStats, which aggregates disaster impact and economic indicators across countries with flexible filtering. It allows users to filter by country, disaster types, and year ranges, leveraging joins between the NationalEconomicImpact, SectoralEconomicImpact, NaturalDisaster, and DirectDamage tables. The stored procedure performs multi-table joins, GROUP BY to aggregate, and uses SQL to handle inputs from the users, allowing for combinations in the filtering process.

The project also includes functioning triggers on the SavedGraphs table to automatically log every insertion, update, and deletion of graphs, using a transaction to manage the updates. To ensure that the data is consistent through the process of updating the saved graphs, we used SQL transactions to ensure a safe and secure process.

Last, the primary keys, foreign keys, and attribute-level constraints were defined well. For instance:

- Users table enforces a primary key on the Username, making it unique which prevents the creation of duplicate accounts.
- To ensure data types are consistent, fields like Email are defined as text, and the Year fields on tables are integers.
- The SavedGraphs table includes a foreign key reference from Username to Users, meaning that a graph will always be associated with a valid user.
- Tables like NaturalDisaster and DirectDamage that relate to the disasters themselves are linked through foreign keys to make sure that the tracking of disasters works well and is consistent.

Technical Challenges.

1. SQL Query Construction

   As we built out the CompareCountryStats stored procedure in the req.sql file, we needed to construct the SQL query based on flexible user-selected filters. To make sure the dynamic SQL was formatted correctly and was executable, we needed to be careful about string concatenation and potential edge cases. Further, debugging the dynamic queries was challenging as mistakes might only be caught when executing, not compiling, the program.

2. Managing Triggers with Transactions

   Implementing triggers on the SavedGraphs table, also in req.sql, introduced complexity as we attempted to ensure that the values for the Logs table were accurate. At times, a CRUD operation

on a graph would succeed, but the log entry would be missing, which is inconsistent with the SavedGraphs table. To fix this, we altered the trigger logic to ensure logging of the operations happens as soon as the graph action finishes– any failures would cause the graph action to fail, which makes the data more reliable.

3. Handling GCP

We adjusted the database configurations on GCP to be more cost-effective by reducing the instance size, optimizing storage settings, and fine-tuning connection limits to better match our usage patterns. By making these changes, we significantly lowered ongoing costs without compromising performance. Both the website and the database are fully hosted on GCP, allowing us to manage everything within a unified, scalable cloud environment.

4. Maybe managing tokens for front end auth

We implemented secure user authentication by using bcrypt to hash passwords before storing them in the database, preventing plaintext credentials from being saved or exposed. For login sessions, we issue JSON Web Tokens after verifying user credentials, allowing the frontend to manage authenticated access through stateless, encrypted tokens. This approach ensures that user data remains protected and that session management is scalable and secure across our Flask backend and GCP-hosted website.

Differences from the Original Proposal

Compared to our original ideas, the final application fulfilled the main goals of the project while introducing several changes. We successfully implemented a platform for users to visualize disaster impacts on economic indicators over time, allow for searching and filtering by disaster type, country, and year, and compare country-level statistics by enabling users to gather data regarding multiple countries on the main page. However, some features, such as a machine learning model for predicting disaster impacts and exporting reports to PDF or CSV formats, were not implemented due to shifts in what we thought would be more interesting and important features.

As we expanded the scope, the web application now includes an explorer for U.S. state-level analysis, allowing users to track disaster trends for individual states. We also added saved graph functionality, which allows users to save specific visualizations to later reload or edit, enhancing the personalization aspect of the application. Additionally, we included a global statistics comparison page, allowing users to aggregate disaster and economic data over time and by disaster type. While the initial ideas emphasized more direct disaster comparisons, our final comparison page provides broader insights and is more realistic given the available data and technical feasibility.

The final platform is closely aligned with the initial proposal and its key goal: allowing users to explore the impact of disasters.

Future Work

If provided with additional time and resources, we could expand this platform in multiple ways to further enhance the functionality of the website. First, we could implement predictive analysis using machine learning models to forecast the economic impact of hypothetical future disasters based on historical trends, fulfilling one of the original goals. We could also continue developing the platform to enable event-to-event comparisons, allowing users to analyze relative impacts rather than just country-level aggregation.

Division of Labour and Teamwork

Throughout the project, we maintained a clear division of responsibilities while collaborating closely on integration, testing, and frontend development. Sneha was primarily responsible for implementing the CRUD operations and backend logic, ensuring that users could create, read, update, and delete their saved graphs and filters within the application. Aarul handled the GCP hosting and database setup, ensuring that the infrastructure was stable and accessible, and also contributed significantly to the design of the database schema. Abhi focused on helping with the overall database design and took the lead on developing the tables and structure for the United States-specific disaster and economic datasets, expanding the project to support state-level analysis. Smriti worked on implementing the advanced database features, including the development of the stored procedures, triggers, and transactions, and also contributed to database schema design improvements.

While each team member had specific areas of focus, we worked together on the frontend development of the application, collaborating on building the user interface, ensuring functionality, and improving usability. In addition, we all participated in testing and debugging across both the frontend and backend to ensure that the final application was stable, functional, and aligned with our original vision. Teamwork was overall balanced, with regular communication and collective problem-solving as we encountered technical challenges throughout the project.

Final Reflection

Overall, our team has learned a lot over the course of the semester while building this web application. Despite the changes from the original idea for the project, the final version provides a user-friendly platform that allows for a broad exploration of the economic impacts of natural disasters around the world, as well as providing further details regarding the United States. We integrated advanced database features into the frontend experience, including stored procedures, transactions, triggers, and constraints. Through building the platform, we gained valuable experience in working with real-world data, handling integration between the frontend and backend, and designing a database system that is effective. During the process, we prioritized delivering a stable and useful application that is intuitive for users. If we were to expand on the project, or if another team were to continue our work, it would involve building on the

foundation we have created by introducing more thorough analytics and further enhancing user experience. All in all, this project has not only reinforced our technical skills but also highlighted the importance of collaboration, decision-making, and the concept of iterative development with real-world applications.