Sky Full of Stars by Mythic Sky Mapper

Andrew Adinolfi

Jake Brady

Jacob Swanborg

I. Introduction

We chose to focus on this project because it sits at the intersection of astronomy, creativity, and social interaction – all themes that we as students and young professionals are passionate about. This application belongs to the astronomy-themed social and creative tools domain, blending elements of scientific learning with opportunities for interactive, user-generated content. Our motivation stemmed from our belief that understanding the world around us, and beyond us, is essential for the soul. What better way to encourage curiosity and appreciation than by studying the stars and creating new connections among them?

The application itself empowers users to design their own constellations based on existing stars in the sky. By selecting and connecting stars, users can generate unique patterns that can be saved for later and edited with ease. In addition, the tool encourages collaboration in the study and enjoyment of user-generated constellations via a share option!

To complete this project, our team divided the work into three core areas: backend, frontend, and integration. Jacob concentrated primarily on frontend development and hosted the framework from which the database could be run. Andrew focused on back-end functionality, and connectivity of the database to the UI. Jake focused primarily on integration, organizing the checkpoints and report, and managing quality. Jake also created the repository to track and monitor progress of the development throughout the semester and executed mock tests to verify the program's performance and to ensure that the minimum viable product met expectations.

II. Implementation

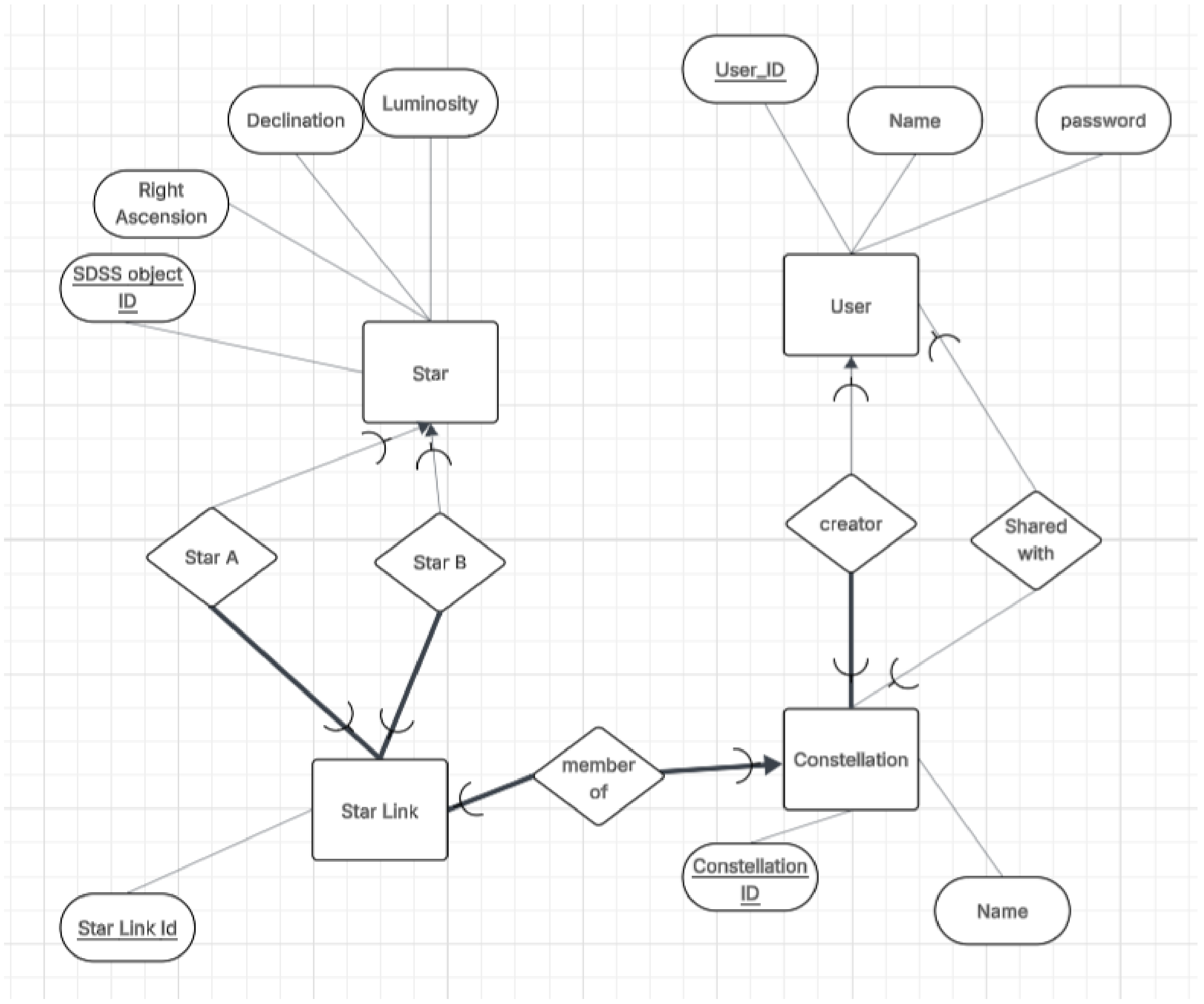
2.1 Description of the system architecture

This application used MySQL for the database backend, while the frontend was coded in Python. The front-end Python packages, to make the GUI, were NiceGUI, astropy, plotly, and numpy. To get the data to the database, we used Pandas, and MySQL connector, which allowed us to add the data from the MySQL tables, and we exclusively use stored procedures for communication back and forth.

2.2 Description of the dataset

This dataset catalogues around 100,000 observations of stellar objects. We chose it because it is the most comprehensive freely available source of stellar data. Since it is so widely known, the object IDs can be referenced by other sources of data that are available giving us some flexibility to potentially expand beyond the MVP based on user feedback.

2.3 ER diagram



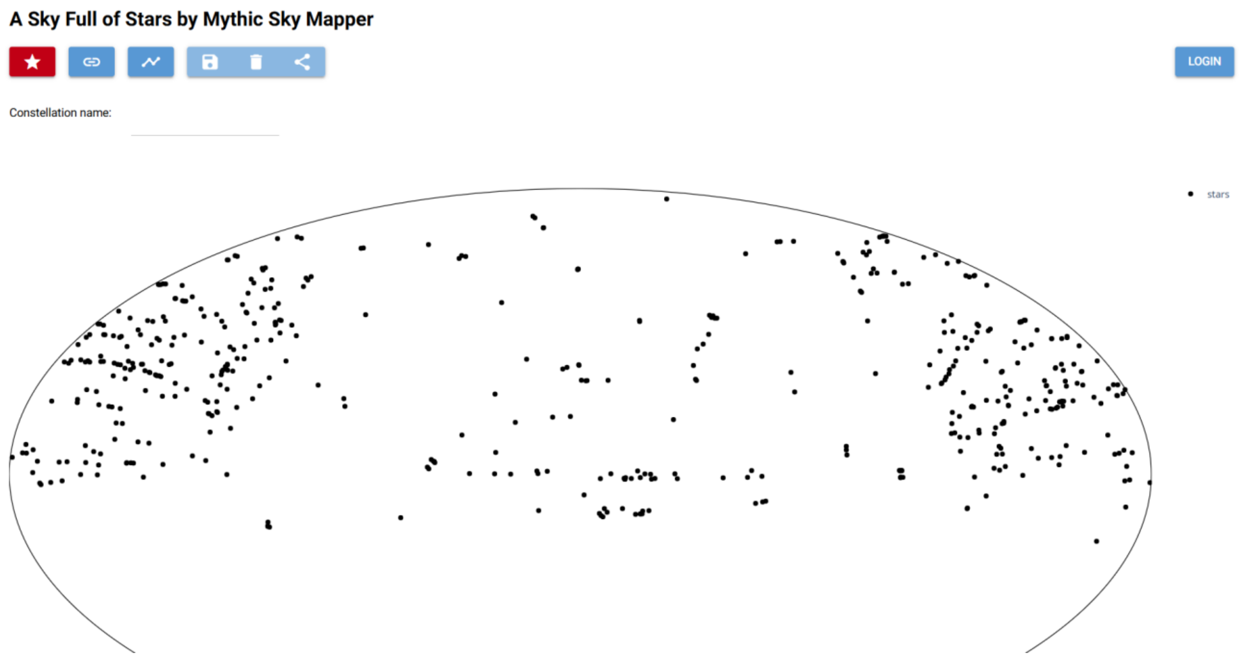
2.4 Relational model

- User table:
  - User_ID: primary key, generated when making account
  - Name: selected by the user when creating account, unique
  - Password: selected by user when making account, stored as hash
- Constellation table:
  - Constellation_ID: primary key, automatically generated when making constellation
  - Name: given by user
  - Creator: foreign key linked to User_ID on the user table
- Shared with table:
  - Constellation_ID: primary key component, foreign key linked to Constellation_ID on constellation table. Generated when a constellation is Shared with another user. Is a candidate key when combined with User_ID.
  - User_ID: primary key component, foreign key linked to User_ID on the user table. Generated when a constellation is shared with another user
- Star link table:
  - Star_Link_ID: primary key, automatically generated when a line is drawn
  - Constelation_ID: foreign key linked to Constellation_ID on constellation table. Assigned to all lines that are a part of the constellation
  - Star_A: foreign key linked to SSDS_Object_ID on the star table. One of the two stars being linked with the line
  - Star_B: foreign key linked to SSDS_Object_ID on the star table. Other of the two stars being linked with the line
- Star table:
  - SSDS_Object_ID: primary key, from data set directly.
  - Right_Ascension: directly from data set
  - Declination: directly from data set
  - Luminosity: calculated from various components of the data sets

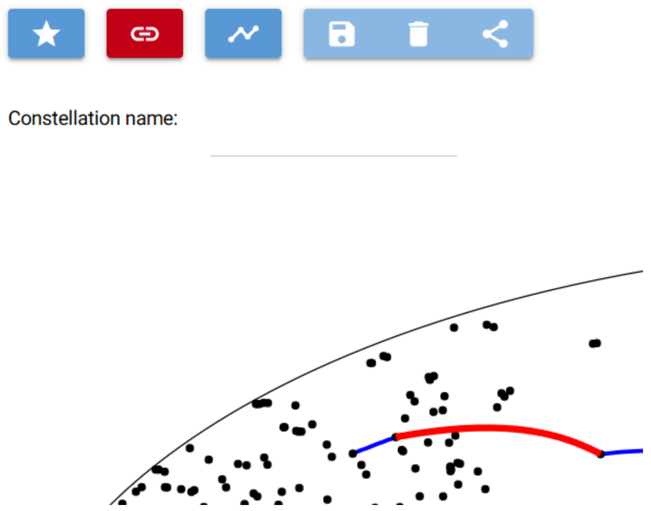
2.5 Implementation: description of the prototype

UI:

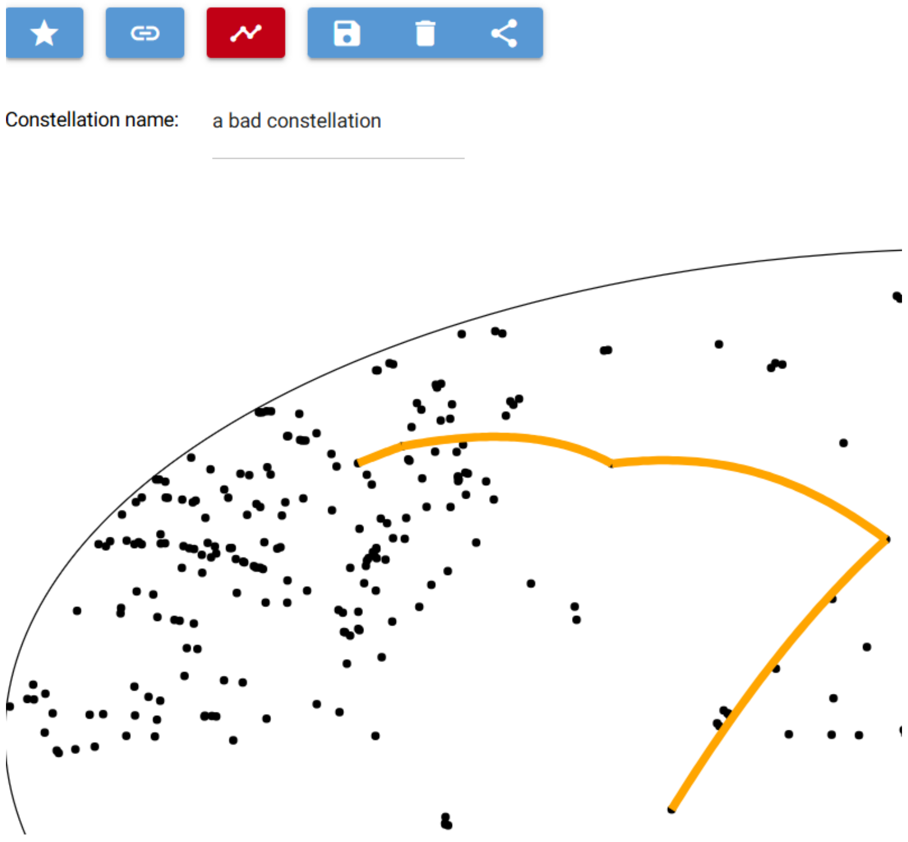
This UI is centered on a map of the night sky, which makes the end user feel right at home! All of the control buttons are above ready to be engaged with.



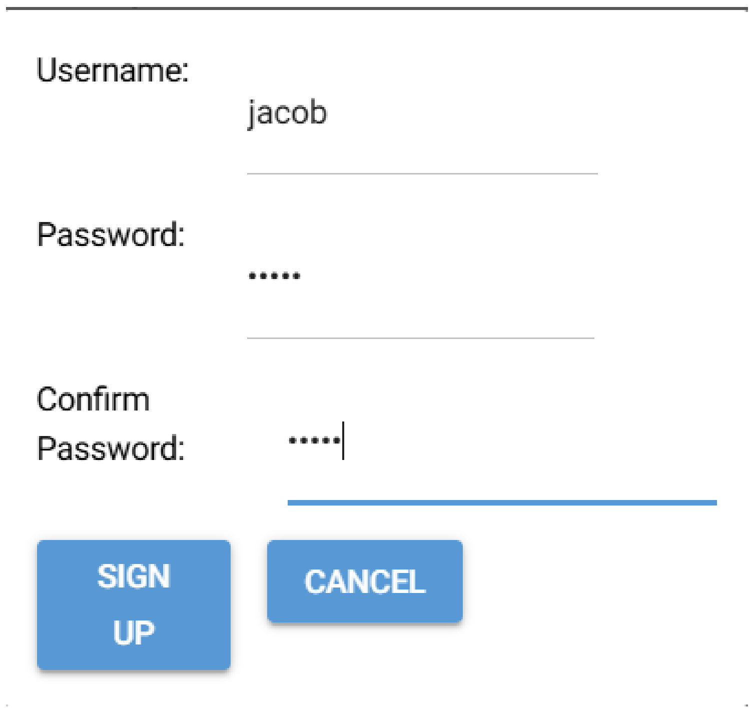
An overview of the entire interface



Selecting a link


Constellation name: a bad constellation

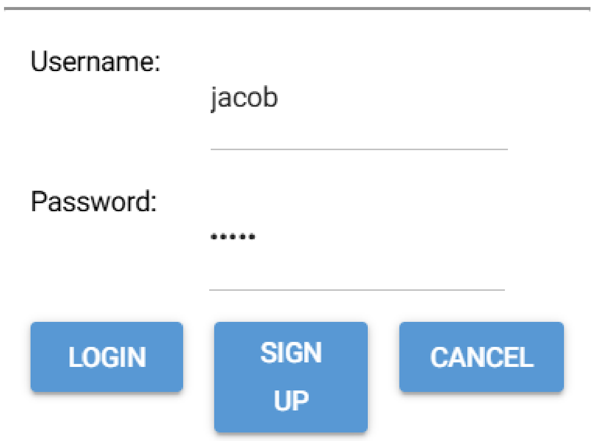
Selecting a constellation


Username: jacob

Password: •••••

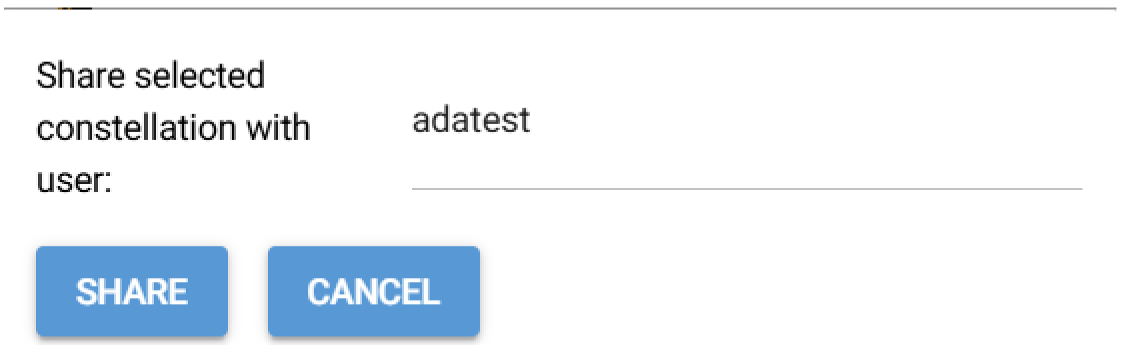
Confirm Password: •••••

Creating an account


Username: jacob

Password: •••••

Logging in


Share selected constellation with user: adatest

Sharing a constellation with another user

<u>Database</u>

The following tables were created for this application:

Star Table – Size: about 88,000 tuples

```
CREATE TABLE Star(
SSDS_OBJECT_ID BIGINT,
Right_Ascension DOUBLE,
Declination DOUBLE,
Luminosity DOUBLE,
primary key(SSDS_OBJECT_ID));
```

User Table – Size: Starts empty, size increases as user data is entered

```
CREATE TABLE User(
User_ID BIGINT,
Name varchar(255) UNIQUE NOT NULL,
Password varchar(255),
primary key(User_ID));
```

Constellation Table - Size: Starts empty, size increases as constellation data is entered

```
CREATE TABLE Constellation(
Constellation_ID BIGINT,
Name varchar(255),
Creator BIGINT NOT NULL,
primary key(Constellation_ID),
FOREIGN KEY(Creator) REFERENCES User(User_ID) ON UPDATE CASCADE ON
DELETE CASCADE);
```

Star_Link Table - Size: Starts empty, size increases as star data is entered

```
CREATE TABLE Star_link(
Star_Link_ID BIGINT,
Constellation_ID BIGINT,
Star_A BIGINT,
Star_B BIGINT,
primary key(Star_Link_ID),
FOREIGN KEY(Constellation_ID) REFERENCES Constellation(Constellation_ID) ON
UPDATE CASCADE ON DELETE CASCADE,
FOREIGN KEY(Star_A) REFERENCES Star(SSDS_Object_ID),
FOREIGN KEY(Star_B) REFERENCES Star(SSDS_Object_ID));
```

Shared Table - Size: Starts empty, size increases as user sharing data is entered

```
CREATE TABLE Shared(
Constellation_ID BIGINT,
User_ID BIGINT,
primary key(Constellation_ID, User_ID),
FOREIGN KEY(Constellation_ID) REFERENCES Constellation(Constellation_ID) ON
UPDATE CASCADE ON DELETE CASCADE,
FOREIGN KEY(User_ID) REFERENCES User(User_ID) ON UPDATE CASCADE ON
DELETE CASCADE);
```

The following 11 stored procedures are called in our application database code, taking actions as their names suggest. Please see the submitted files for the code within.

GetStars
UsernameUsed
CreateUser
CheckPassword
GetUserConst
GetConstSharedWithUser
Share
DeleteConstellation
CreateConstellation
AddLine
ConstLines

[Considered but not done] - The following options are potential areas that we could add features and would require additional stored procedures:

- Allowing users to select how many stars to show:
  - Gettings stars for all shared constellations and my constellations (since you might be missing some of the stars)
- Unsharing:
  - Unshare stored procedure
- Deleting your user record
  - Delete user function

2.6 Evaluation

To evaluate the application, we created a series of testing scenarios that reflected realistic user actions, such as creating, saving, editing, and sharing constellations. We also tested backend queries to confirm that constellations were correctly linked to users, updates were stored without duplication, and results were retrieved accurately and efficiently. Edge cases, such as saving an empty constellation, connecting the same star multiple times, or exceeding the intended number of stars, were used to assess robustness. For each test, we compared the actual behavior to expected results, logged discrepancies, and made adjustments to either the backend, frontend, or database schema as needed. Finally, we performed regression testing with mock cases to ensure that fixes did not break existing features, which helped confirm that the application met its functional goals and handled unexpected input reliably.

Since our presentation on 8/14/2025, the following tests are now passing:

- Checking a password is now implemented in the GUI
- Creating a constellation is implemented in the GUI
- Creating a star link is implemented in the GUI
- Sharing is now implemented in the GUI

- Un-sharing is supported in the backend, but not yet implemented in the GUI
- Deleting a constellation is now supported in the GUI
- Deleting a star link is now implemented in the GUI

III. Conclusion

Through this project, we gained a deeper understanding of how to design and implement a user-facing application that is both functional and engaging. One of the most interesting aspects was seeing how astronomy concepts, such as constellations, could be transformed into interactive features that encourage creativity and learning. Building something that combined education with a community-driven focus really instilled a sense of purpose in this project.

We also encountered less exciting but equally important challenges. For example, coordinating the integration of backend and frontend components sometimes involved repetitive debugging and testing, which felt more tedious but was essential to ensuring a stable final product. Similarly, structuring data consistently so that user-generated constellations could be stored, retrieved, and shared without error required careful planning that wasn't always glamorous, but proved critical.

From a database perspective, the knowledge gained throughout this course was very relevant. Concepts like schema design, normalization, and query formulation guided how we stored constellations and user information. We found that normalization helped reduce redundancy and made our database easier to maintain. Writing SQL queries also became very practical as we needed efficient ways to fetch and update user-generated constellations.

We also encountered some database-related challenges like those discussed in class. For example, managing relationships between tables, such as linking users to their saved constellations, highlighted the importance of designing with referential integrity in mind. Query performance was another issue: some queries became more complex as we added new features, which made us think carefully about indexes and optimization strategies. These lessons reinforced how classroom knowledge about relational databases translates directly into real-world application development.

IV. References

Sloan Digital Sky Survey DR17 dataset available at:
https://www.kaggle.com/datasets/fedesoriano/stellar-classification-dataset-sdss17/data

Coldplay - A Sky Full Of Stars (Official audio)