Steamalytics: A Database Management System for Games

Group: SongXWenC

Group Member: Xiuyuan Song, Chongjie Wen

Introduction

In this report, we delve into the intricate design and development of a state-of-the-art application that stands at the forefront of gaming database management. At the heart of this project lies a meticulously crafted database schema, thoughtfully structured to encompass a diverse range of entities and complex relationships, tailored to comprehensively address the multifaceted aspects of gaming data. This foundation is bolstered by a suite of well-conceived procedures, functions, and triggers, specifically designed to optimize and facilitate the core CRUD functionalities - a testament to our commitment to precision and efficiency. Complementing the robust backend, the application boasts a sophisticated front-end interface, developed with the cutting-edge javascript framework, ensuring a seamless and user-centric experience. Together, these components coalesce to form a harmonious synergy of technology and usability, marking a significant stride in the realm of gaming database applications. The ensuing sections of this report will provide a detailed exposition of the conceptual and logical designs, the technical specifications, and the seamless integration of backend and frontend technologies that define the essence of our project.

Discussion

The architecture of our CRUD application is artfully depicted through a UML diagram, offering an overarching view of the complex interplay between various key entities—Game, Publisher, Genre, Images, Platform, Users, Review, and Rating. Each of these entities interlocks within the schema, mirroring real-world associations, such as games linked to their publishers and genres, while users are poised to provide feedback via reviews and ratings. This UML framework navigates the development trajectory, illuminating every critical element of the gaming database, from the intricacies of game organization to the dynamics of user engagement, thereby painting a comprehensive picture of the application's design philosophy.

As we transition from conceptual mapping to logical structuring, the MySQL Workbench tool becomes instrumental, transforming our established database schema into an Entity-Relationship

(ER) diagram. This meticulous representation lays bare the intricacies of our database—tables bloom with attributes while relationships weave a tight network, ensuring data consistency, as highlighted by the foreign key constraints connecting games to their publishers. This logical architecture not only dictates the secure storage and retrieval of data but also becomes synonymous with the application's functional aspirations—be it in the realm of game entries, user management, or the critique ecosystem of reviews and ratings. The ER diagram, thus, serves a dual purpose: it is a beacon for our database implementation strategy and also a conduit for clear communication between the abstract and the tangible realms of database design.

Beneath the surface of our application lies a sophisticated database schema, a testament to the intricate planning that caters to the diverse and interrelated facets of a rich gaming database. This schema underpins the entire spectrum of our data management—from cataloging game particulars to curating user-generated content. To amplify the precision and agility of these operations, a suite of bespoke procedures, functions, and triggers have been engineered, each acting in concert to refine data transactions, safeguard consistency, and expedite automatic processes, thereby bolstering the database’s resilience and performance.

In harmony with the backend prowess, the project is complemented by a fluid front-end, crafted with the agility of javascript, ensuring that users navigate through a responsive and adaptive interface. This front-end gateway facilitates a myriad of interactions with the database, such as inserting new games, personalizing user profiles, and articulating feedback through reviews and ratings. It is a simple to use application, all the operation users need to do is to click on the function they like. Javascript not only injects vitality and user-centric design into the application but also promises scalability, paving the way for ongoing enhancements. This symbiotic fusion of a resilient database structure and an agile front-end ecosystem ensures that our application stands ready to deliver a user experience that is as immersive as it is intuitive, setting a new benchmark in the digital curation of gaming data.

Lesson Learned

In developing our application, we've gained significant technical expertise, particularly in Python and JavaScript libraries, which were vital for integrating the backend and frontend. The project sharpened our understanding of data structures and operations, revealing the importance of systematic design and the nuances of creating a user-friendly interface from the ground up.

Balancing this project with academic commitments and job searches taught us critical time management skills. We learned to prioritize effectively, an essential skill in software development. Reflecting on the project, we recognize the value of considering alternative approaches and the impact of design choices on usability and performance. A noteworthy challenge was troubleshooting code that failed to connect the database with the frontend, providing us with a practical lesson in the importance of meticulous testing and the opportunity to learn from our missteps. These insights are invaluable for future endeavors in both the technical and operational domains of software development.

Future Direction

Looking ahead, the database is poised for several exciting applications and enhancements. A primary avenue for expansion is the implementation of an API, which would open the door for users to add games at their discretion. This feature not only democratizes the database, allowing for a more personal and dynamic collection of gaming data but also transforms the platform into a tool for individuals to catalog and track their personal game libraries. The envisioned API would facilitate integration with external services and marketplaces, offering users the ability to update their collections in real-time as they acquire new games. Another potential area for added functionality is the adaptation of the database's tracking capabilities to other domains. The modular nature of the design allows for the tracking system to be repurposed for a variety of collections, such as clothing, books, or any other items of interest. By abstracting the core components of the database, we can create a versatile application that can serve as a personal inventory management system across different contexts.

Appendix 1. Figures

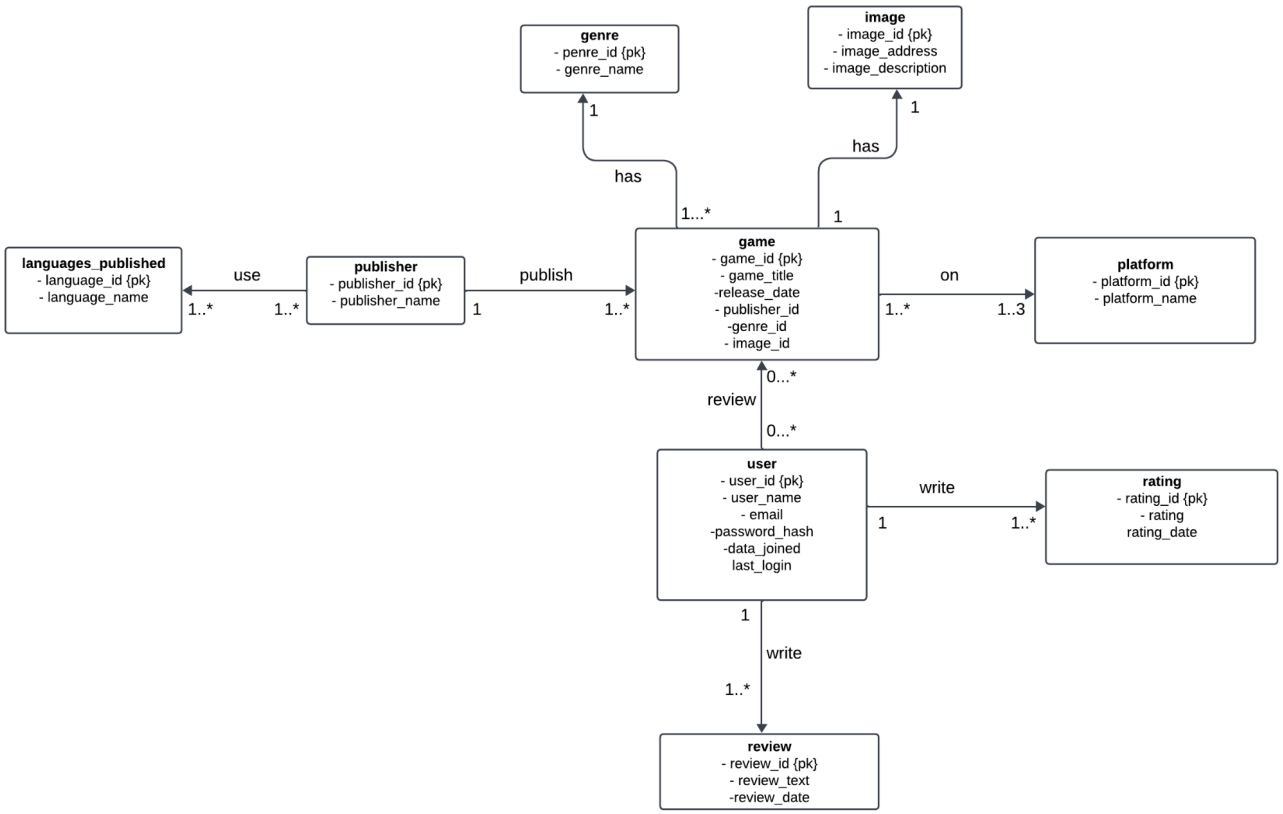

Figure 1. UML of the schema

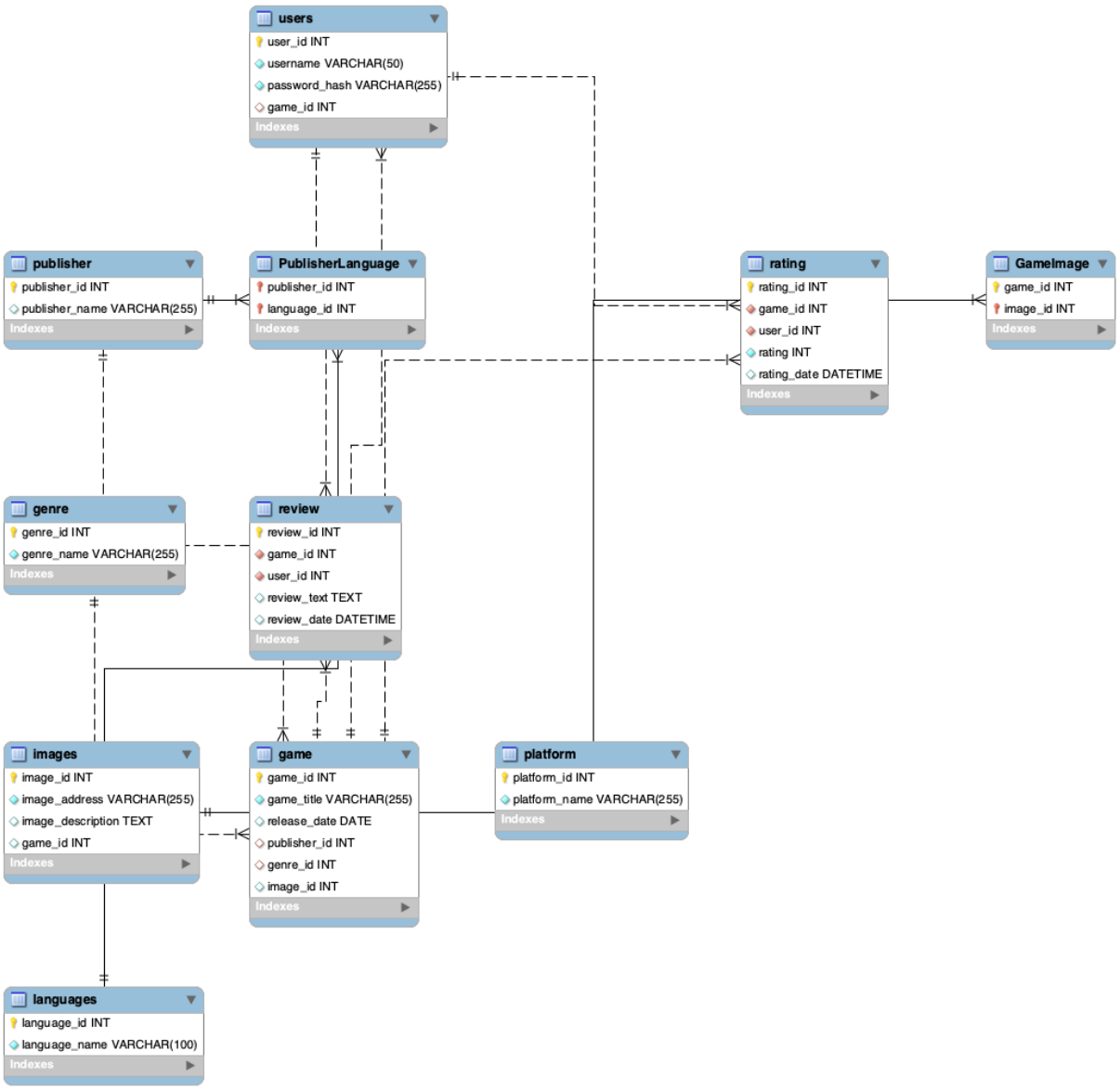

Figure 2. ER of gamedatabase

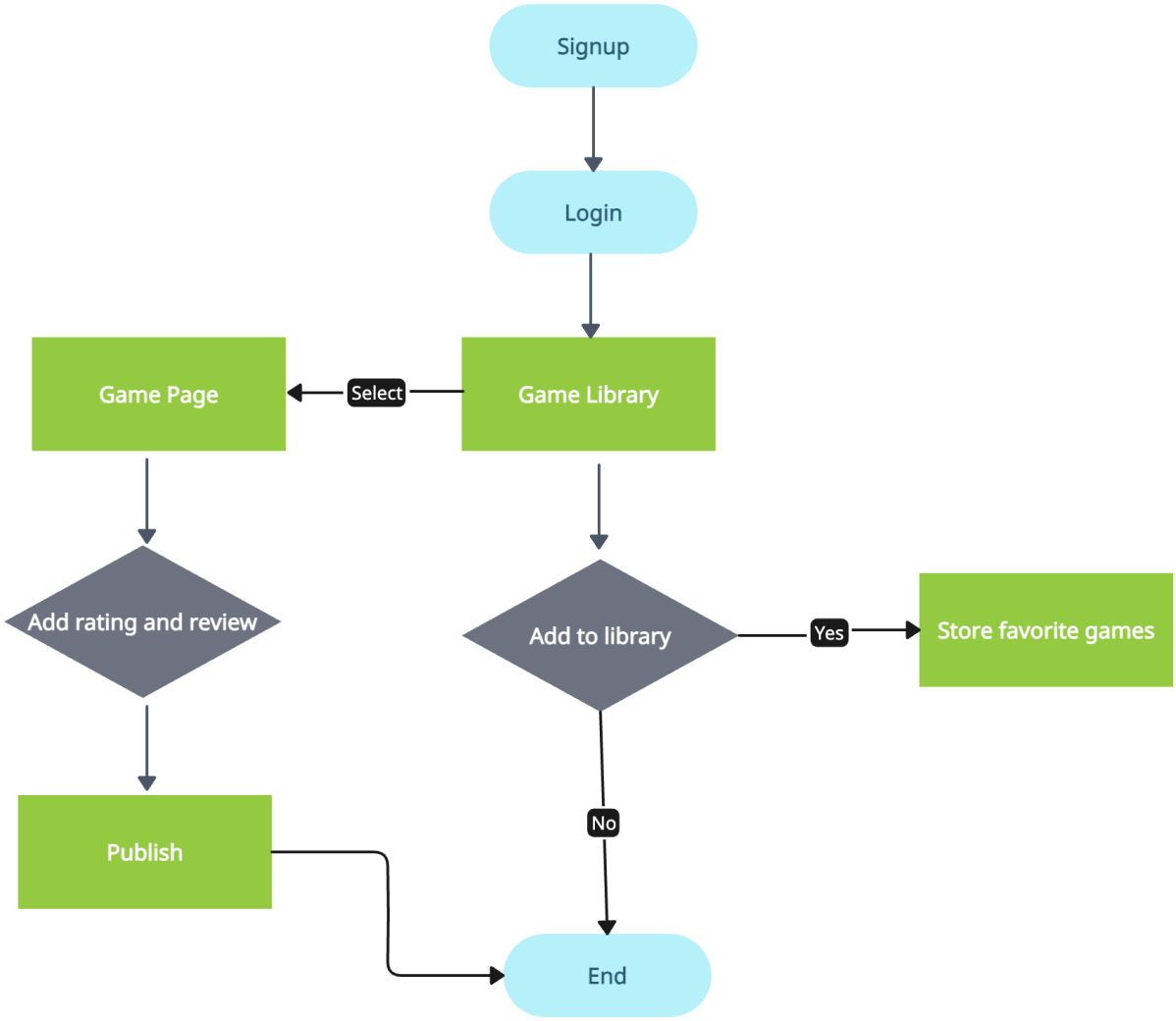

Figure 3. User Flow

Appendix 2. README

This section is included in the github repository with the complete code of project: https://github.com/lucklyjosh/5200-database-final-project

After unzipping the zip file of this project downloaded from github, we highly suggest opening it with an IDE for a better experience. And your equipment needs to have Python, MySQL, node npm installed in order to run this application successfully.

Installing necessary package to you equipment:

1. Have MySQL and MySQL workbench installed, tutorial can be found in the links below:
    a. https://dev.mysql.com/downloads/installer/
    b. https://dev.mysql.com/doc/workbench/en/wb-installing.html
2. Have python installed:
    a. https://www.python.org/downloads/
3. Have node.js and node npm installed:
    a. Installation tutorial: https://docs.npmjs.com/downloading-and-installing-node-js-and-npm

Before running the application:

1. Make sure you download and run the data dump "gamedatabase.sql" on workbench first and check if all the tables are shown

Opening the application in local environment:

1. Open the project folder in you selected IDE, I'll use VS Code as an example here
2. Navigate through the folders, go to Backend/app.py, at line 9, change the information of SQL server:
    a. app.config['SQLALCHEMY_DATABASE_URI'] = 'mysql+mysqlconnector://username:password@localhost:3306/gamedatabase"
    b. Change the field of server name(in red) if yours is not root
    c. Replace sql server password (in blue)to yours

3. Open an terminal window at root directory, then install the package to run the back end first:
    a. pip install Flask
    b. pip install Flask-SQLAlchemy
    c. pip install mysql-connector-python
4. Make sure all package are installed properly, type these two command in terminal window to start the back end:
    a. cd Backend
    b. python -m flask run
5. While the back end is running properly, now we can start running the front end. Open a new terminal window at the root directory named: “game-system” by right clicking the folder name and select “Open in integrated terminal”
6. If you have node npm installed in your equipment successfully, please type the following commands in the terminal window:
    a. npm install
    b. npm start
7. A new website should pop up in the browser, now it’s time to organize your game collection!