DiscoverEase: Tailored App Recommendations at Your Fingertips

GROUP 13 TEAM MEMBERS

- Swarn Gaba (sgaba@iu.edu)
- Gandhar Ravindra Pansare (gpansar@iu.edu)
- Chaithra Lal Nair (cnair@iu.edu)

APPLICATION URL:

Enhance your digital experience with smart, tailored solutions at your fingertips. It's just a click away - explore to discover with DiscoverEase on Streamlit Community Cloud.

https://discoverease.streamlit.app/

FULL GITHUB URL:

The repository can be accessed via the following link:

https://github.com/swarngaba1997/DiscoverEase

PURPOSE:

The purpose of our project is to develop an interactive app recommendation system called "DiscoverEase." This system aims to redefine the app discovery experience by making it more personalized and insightful, significantly enhancing user interaction with the system. The objective is to ensure that the recommendations provided are not only relevant but also resonate with the users' preferences and needs. This involves a meticulous database design to organize data about mobile applications, developers, and pricing information, thus creating a robust foundation for executing SQL queries that will facilitate accurate app recommendations tailored to individual user profiles.

OBJECTIVES:

Our main goals are to make finding apps easier and to provide a personalized experience for users. We want to simplify the app discovery process so users don't have to go through millions of apps to find ones that genuinely interest them. Additionally, we aim to personalize app recommendations, allowing users to customize their results based on their preferences and specific parameters like genre, user rating etc. This way, users can easily discover apps that suit their tastes and needs without the hassle, making the entire process efficient and enjoyable.

Our goals include:

- **Customized User Experiences**
  The aim here is to tailor app recommendations specifically to each user's preferences. This means the system will consider individual likes, dislikes, and past interactions with apps to suggest new apps.

- **Maximizing Dataset Potential**
  This objective focuses on making the most of the extensive data available from the Apple App Store. The dataset includes various app attributes such as genre, content rating, and user ratings. By analyzing this data, the system can identify patterns and preferences, leading to more accurate and useful app suggestions.

- **Initial Experience Enhancement**
  The 'cold start' problem is a common challenge in recommendation systems, referring to the difficulty of providing personalized recommendations to new users who have not yet provided enough data about their preferences. This objective addresses this challenge by finding innovative ways to offer meaningful recommendations to new users, possibly by utilizing generalized data or trends among similar users, until the system has gathered enough individual data to personalize suggestions.

- **Broadening the Recommendation Spectrum**
  In addition to personalization, the project seeks to broaden users' horizons by providing a variety of app suggestions. This means the system will not only focus on users' established preferences but also introduce them to apps they might not have discovered on their own.

USEFULNESS:

Our app recommendation platform strives to serve a wide range of users by utilizing the power of a comprehensive database, becoming an invaluable tool for anyone seeking personalized app suggestions. The database, complemented by an interactive interface, promises significant utility by enhancing app visibility, particularly for smaller developers and niche applications struggling in a saturated market. Moreover, the database offers deep insights into market trends, user behaviors, and preferences, which are invaluable for app developers and industry stakeholders to make informed decisions on app development and marketing strategies. Our database stands out by incorporating a wealth of data from the Apple App Store, comprising around a million app entries. While our platform shares some inspiration with the Google Play Store dataset, our focus on the Apple App Store data sets us apart, offering a specialized approach to app recommendations for iOS users. This differentiation makes our database particularly relevant for individuals seeking tailored app suggestions within the Apple ecosystem, enriching the app discovery experience.

DATA:

For the "DiscoverEase" project, the dataset utilized is primarily sourced from Kaggle titled "Apple AppStore Apps".

https://www.kaggle.com/datasets/gauthamp10/apple-appstore-apps?select=appleAppData.csv

The dataset was meticulously compiled by Gautham Prakash in October 2021, drawing inspiration from a similar dataset that was designed for Google Play Store applications. The primary objective behind the creation of this dataset was to establish an extensive and detailed database of applications found in the Apple Store, thereby supporting a wide range of data-driven projects and analyses.

This dataset consists of approximately 1.2 million records, distributed across 22 distinct columns, capturing a wide array of application information including but not limited to app name, developer info, app category, size, price, rating, user ratings, and more. This data is crucial for the app recommendation system that aims to personalize and optimize app discovery for users.

Originally, the dataset was curated for the purposes of cataloging apps and facilitating in-depth analysis. As such, it stands as a significant resource for our recommendation system, offering a rich source of data regarding the variety of applications available on the Apple App Store.

With its vast collection of approximately 1.2 million entries and 22 different attributes detailing various aspects of the apps, this dataset provides a comprehensive foundation for our system. However, to align more closely with our project's scope, we had performed the data sampling based on genres to maintain a representative subset of the original dataset, which ensures that all genres are proportionately represented. This approach allows us to normalize the data effectively and organize it into separate tables as required for our analysis.

This cleaned and sampled data was used to create a relational database (**DiscoverEase.db**), which includes several tables such as **applications, developers,** and **pricing**. SQL queries were employed to populate these tables with data from the cleaned dataset.

HOW WE HAVE BUILT OUR PROJECT?

To build the DiscoverEase project, we selected a combination of tools and database technologies designed for efficiency, reliability, and ease of use. Here's how we constructed our project:

Frontend Development:

Streamlit: Chosen for its rapid development capabilities, Streamlit served as our primary tool for building the user interface. It allowed us to create an interactive and visually appealing web app without the need for complex front-end frameworks. Streamlit's ability to integrate directly with Python made it an excellent choice for incorporating our data processing scripts into a cohesive app.

Backend Development:

Python: As the core programming language for backend development, Python's vast ecosystem of libraries facilitated various aspects of our project. Libraries such as Pandas for data manipulation, SQLite3 for database interaction, and Plotly for visualizations were instrumental. Python's versatility was invaluable in handling everything from data processing to implementing business logic.

Database Management:

SQLite: For our database, we utilized SQLite due to its lightweight nature and serverless architecture, which eliminated the need for a dedicated database server. It was ideal for our use case, providing robust relational database management capabilities while being easy to integrate with our Python backend. SQLite enabled us to store, retrieve, and manage app data efficiently, offering SQL capabilities without the overhead of more complex database systems.

Data Processing and Analysis:

Pandas: This library was essential for data cleaning, transformation, and preparation tasks. We used it to read the CSV data, process and clean it, and prepare it for insertion into the SQLite database. It provided a broad range of tools for exploring and manipulating tabular data, making it perfect for our dataset which was central to the app recommendation logic.

Image Processing:

PIL (Python Imaging Library): We utilized PIL for image handling tasks, such as enhancing the brightness of background images for the Streamlit interface. It enabled us to programmatically modify images and convert them into formats suitable for embedding within our application's UI.

Integrated Development Environment and Version Control:

Visual Studio Code (VSCode): For code development, we used Visual Studio Code (VSCode), a powerful and versatile IDE that supports Python and a multitude of other languages. Its rich ecosystem of extensions, such as those for Python, SQLite, and Git, allowed us to develop efficiently with tools like intelligent code completion, linting, and syntax highlighting, directly within the IDE.

Git and GitHub: Version control was handled using Git, with GitHub serving as our remote repository and source code management platform. This combination facilitated collaboration among team members, allowing us to track changes, manage branches, and review pull requests. GitHub's integration with Streamlit Community Cloud also simplified our deployment process, enabling automatic updates to the live application with each push to the main branch.

Deployment:

Streamlit Community Cloud: For deployment, we leveraged Streamlit Community Cloud, which allowed us to deploy our web app directly from our GitHub repository. This platform made our application accessible over the web with minimal setup, providing a hassle-free deployment process. The Community Cloud supports Streamlit apps natively and offers features like continuous deployment, where any changes pushed to our GitHub repository are automatically updated in the live application. This choice offered us a streamlined deployment experience, enabling us to focus on developing features and enhancements without worrying about infrastructure management.

By integrating these tools and technologies, we created an end-to-end solution that supported data-driven app recommendations, provided an intuitive user experience, and maintained high performance with reliable data management. Each tool was chosen not only for its individual strengths but also for how well it integrated with the rest of the technology stack, creating a well-rounded and cohesive application.

FUNCTIONALITIES

1. **CRUD Operations:**

- **Create New App Entries:** Users can add new application entries into the database by providing details such as app ID, name, developer ID, genre, size, version, iOS version, release and update dates, user ratings, age group, developer name, price, and currency. This function also includes validation to ensure all inputs are correctly formatted and all required fields are filled.

- **Read Existing App Data:** This feature allows users to retrieve and view detailed information about apps stored in the database by entering the app name. It displays related data such as app details, developer information, and pricing in a structured format.

- **Update Existing App Entries:** Users can update details of existing apps. This includes fetching current app data and allowing the user to input new values for various attributes like app name, genre, and price, which will then be updated in the database.

- **Delete App Entries:** This function enables users to delete app entries from the database by providing the app ID. It ensures that all related records in pricing and developer tables are also removed.

2. **Data Visualization and Reporting:**

- **Top Developers:** Visualizes the top developers based on the number of apps they have developed using bar charts, helping users quickly understand which developers are most prolific.

- **Display Free Applications:** Filters and displays applications that are free within a selected genre, aiding users in discovering no-cost options available within specific categories.

- **Top Rated Apps by Genre:** Showcases a list of the highest-rated apps within a selected genre, making it easier for users to find quality applications.

3. **Enhanced Search and Filtering Options:**

- **App Data Filtering:** Offers filters for users to view applications based on genre, minimum user rating, and size. This feature allows users to tailor the display of app data to meet specific criteria, enhancing the user's ability to find relevant applications.

- **Search Functionality:** Includes a search bar that allows users to search for applications by name. This feature is particularly useful for quickly locating specific apps within the database.

Additionally, data integrity and validation are crucial aspects of ensuring the robustness of an application. Input validation is employed to ensure that all user inputs conform to the application's requirements. This includes verifying that numerical values, such as price and size, are non-negative, ensuring that dates are in the correct format, and validating text fields for genres and names to meet specific criteria. Furthermore, uniqueness checks are conducted to confirm the singularity of critical identifiers such as app ID and developer ID. These checks help prevent the creation of duplicate records in the database, maintaining the integrity and accuracy of the data.

ISSUES - TECHNICAL / DESIGN ISSUES FACED AND SOLUTIONS

Data Validation and Integrity

Issue: During the development of "DiscoverEase", we encountered issues related to data validation and integrity, particularly with user inputs for new app entries and updates. Input fields like app IDs and developer IDs required validation to ensure uniqueness and prevent database errors.

Solution: We implemented functions in Python to check the uniqueness of app and developer IDs before insertion into the database. These functions query the SQLite database to check for existing records and return a validation message if duplicates are found. This ensured that all data entries maintained integrity and uniqueness.

Integrating Complex SQL Queries

Issue: Complex SQL queries for data retrieval and manipulation posed challenges, especially when dealing with joins across multiple tables and ensuring optimal performance.

Solution: We refined our SQL queries and utilized pandas for some of the data manipulations, balancing load between the database and the application. This approach allowed for efficient data processing and reduced the complexity within the database.

Ensuring Data Consistency across Views

Issue: Initially, ensuring data consistency across different views (like developer details, pricing information) whenever updates occurred was problematic. Changes in one part of the application did not always immediately reflect in other related parts.

Solution: To address this, we implemented real-time data fetching mechanisms using Streamlit's session state features. Whenever data was updated in the database, the changes were propagated across all active views, ensuring that all parts of the application displayed the most current data. This was crucial for maintaining consistency and reliability in the user experience.

Deployment Configuration Challenges

Issue: During the deployment, we encountered several configuration issues on GitHub that hindered the build process of the application. These problems stemmed from missing dependencies and incorrect setup files, which led to failures in deploying the app via Streamlit Sharing.

Solution: To resolve these issues, we created a requirements.txt file that explicitly listed all the necessary dependencies for our application, ensuring that all required packages, were correctly installed during the deployment process. Additionally, we added a config.toml file to customize the theme of our Streamlit application, setting specific colors for the primary, background, and text elements, and defining the font style to enhance the user interface. This approach allowed us to overcome the deployment barriers and successfully launch the application on Streamlit Sharing.

OVERALL CONTRIBUTION SUMMARY

Data Description and Planning Phase:

Name	Tasks and Contribution	Average Time Spent (Hrs)
Swarn Gaba	Leading project conceptualization, defining objectives, contributing to dataset selection, tech stack decisions, contributing technical insights and report creation.	12 hours
Gandhar Ravindra Pansare	Preliminary data analysis, key attribute identification, contributing to dataset selection, tech stack decisions, and report creation.	12 hours
Chaithra Lal Nair	Planning technology stack, initial planning for future database design, contributing to dataset selection, and report creation.	12 hours

Database Design Phase:

Name	Tasks	Contribution	Average Time Spent (Hrs)
Swarn Gaba	Conceptual Schema Database Code	Coordinated and ideated the design process, defined and created entity relationships, validated schema accuracy, and ensured normalization aligns with project objectives. Aided in documentation. Primary Key Implementation - Identification of primary key constraints and oversaw the enforcement of not null constraints to ensure data completeness. Spearheaded the data cleaning process i.e. checking for missing values, dropping unnecessary columns, genre-based data sampling, and verifying column data types. Created the schema for the “applications” table and managed the insertion of clean data into “applications” table. Establishing “Top Developers” view for identifying top developers based on the number of applications they've developed.	20 hours

Name	Tasks	Contribution	Average Time Spent (Hrs)
Gandhar Ravindra Pansare	Conceptual Schema Database Code	Executed data analysis for schema development, created entity relationships diagram and contributed to normalization. Assisted in report creation. Foreign Key Integration - Analyzed data relationships and implemented foreign key constraints to maintain referential integrity between tables. Responsible for importing necessary modules, transforming data into categorical values. Created the schema for the “developers” table and managed the insertion of clean data into “developers” table. Establishing “Free Applications” view for listing free applications along with their developers.	20 hours

Name	Tasks	Contribution	Average Time Spent (Hrs)
Chaithra Lal Nair	Conceptual Schema Database Code	Researched tech stack implications for schema, refined entities and attributes, and contributed to schema normalization. Led report creation. Constraints Validation - Verified and validated all constraints across tables for consistency, ensuring the integrity and reliability of data relations. Focused on converting data types, renaming all columns and resetting index of the dataframe. Created the schema for the “pricing” table and managed the insertion of clean data into “pricing” table. Establishing “Top-Rated Apps per Genre” view for identifying the highest rated applications by genre.	20 hours

Implementation Phase:

Name	Tasks and Contribution	Average Time Spent (Hrs)
Swarn Gaba	**Frontend Development:** Utilized Streamlit to create an interactive and visually appealing front-end. Integrated frontend with backend functionalities. **UI Design:** Designed interface elements and ensured the application was user-friendly and responsive. **Deployment:** Managed the deployment process, configuring and launching the application on Streamlit Community Cloud for public access. **Documentation:** Report creation in order to make it easier for future developers to understand and maintain the system.	22 hours
Gandhar Ravindra Pansare	**Backend Development:** Wrote backend logic in Python, incorporating business logic and data handling capabilities. **API Integration:** Integrated external APIs to enhance the functionality of the application, ensuring smooth data integration. **Data Visualization:** Created visualization to represent data insights. **Documentation:** Report creation in order to make it easier for future developers to understand and maintain the system.	22 hours
Chaithra Lal Nair	**Database Management:** Set up and managed the SQLite database, including interactions with backend and performance optimization. **Quality Assurance:** Conducted extensive testing to ensure the application ran smoothly and efficiently under various scenarios. **Data Integrity and Validation:** Implemented checks for data consistency, uniqueness, and correctness to ensure robustness of the application data. **Documentation:** Report creation in order to make it easier for future developers to understand and maintain the system.	22 hours