Movie Library

DSCI 551 Final Project - Team #79

Team Details

Student #1

- Name: Pooridon Rattanapairote
- Email: prattana@usc.edu
- USC ID: 1469709999

Student #2

- Name: Pannawat Chauychoo
- Email: pchauych@usc.edu
- USC ID: 7282127237

Implementation Questions

Which database are you using?

- The database being used is MongoDB, a NoSQL database system. Our database system is hosted on four separate EC2 instances including 2 sharding servers, 1 configuration server, 1 mongos (for client connection). Both sharding servers and configuration server will have replication factors of 3 replicas.

What is the approach for Distributed scaling of data?

- The approach for distributed scaling involves MongoDB's sharding across EC2 instances using hash function, with each shard consisting of a replica set for high availability and data redundancy, thus ensuring both horizontal scalability and fault tolerance. Additionally, techniques such as indexing, query optimization, and efficient schema design are employed in the backend.

Which application did you choose to implement?

- Two web applications are implemented: an admin application for CRUD operations on movie data, and a user application for searching and interacting with movie content. The implementation includes an admin application for database management and a user application for interactions like search and feedback. User actions are collected in the database to refine and enhance movie recommendations.

Planned Implementation

Project Scope

Project:
- Develop a scalable movie database with user interaction capabilities

Objectives:
- Design and implement a MongoDB database to store movie, user, and user interaction
- Utilize MongoDB sharding on EC2 Ubuntu instances for horizontal scalability and fault tolerance.
- Build two web applications:
    - Admin application: Provides CRUD functionality for managing movie data.
    - User application: Enables users to search and explore movie information, with functionalities such as liking, disliking, marking as watched, and clicking to view content. These interactions will be stored in the database for further analysis.

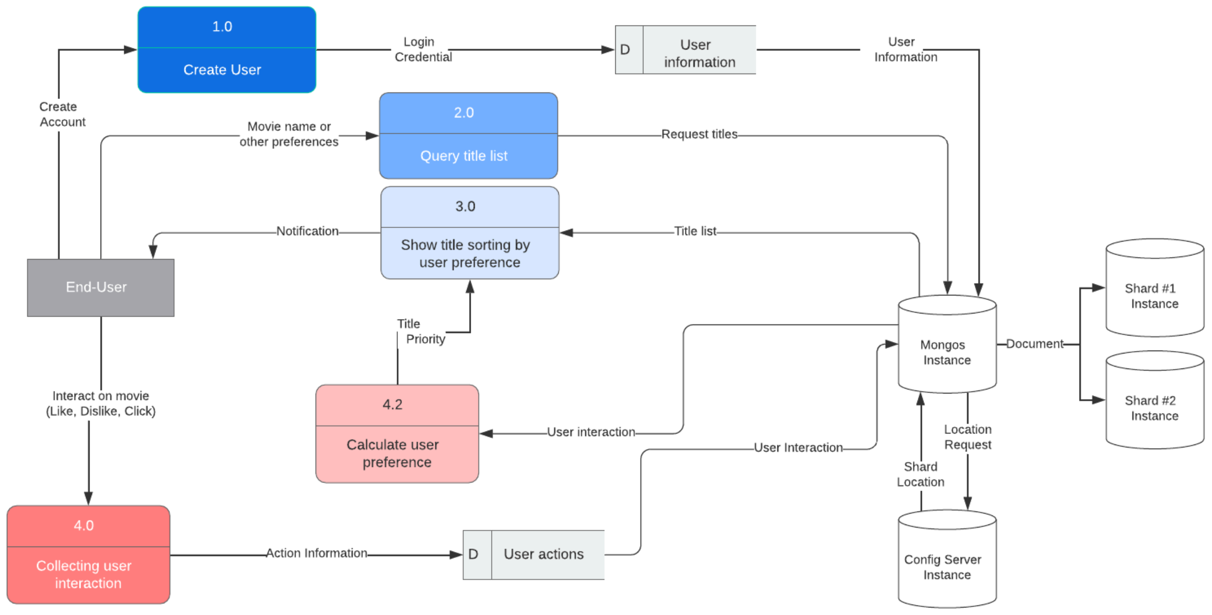
Database Specifications

- Database: MongoDB
- Hosting platform: EC2 Ubuntu instances
- MongoDB sharding:
    - 2 shard servers (3 replicas each)
    - 1 config server (3 replicas)
    - 1 mongos server for application access
- Database collections:
    - movies
    - users
    - user_interactions

Frontend Specification

- Libraries: HTML, CSS, Javascript
- Host: Github
- 3 pages: Home (search), recommendation, catalog

Data Flow Diagram



Status of the project

Accomplishment

Database

- Successfully set up the MongoDB database on an EC2 instance, complete with an imported dataset of Netflix titles.
- Configured the entire sharded cluster, including the configuration server, two shard servers, and the mongos routing service.
- Established sharding with two chunks on the configuration server, enabling data distribution across shards according to the configured shard key.

Admin Interface

- Developed and tested the backend's ability to connect to mongos and perform basic CRUD operations on the database.

End-user interface

- Created the layout for 3 pages in Figma
- Created the HTML layout and CSS for the first page

Next Step

Database

- Integrate advanced database techniques into the backend, including indexing and optimized schema design, to improve query performance and ensure efficient data access

Admin Interface

- Extend the backend to support and test the full range of CRUD operations required by the project.
- Develop the first version of the web application for admin users, featuring straightforward interfaces for single record insertions, updates, and deletions.

End-user interface

- Finished the HTML and CSS for the remaining pages
- Use Javascript to add animation and interactivity
- Create the website layout on Figma (3 hours)
    - Link:https://www.figma.com/file/vstGU0OXPSHORKdEdPy5bF/Movie-rec-website-layout?type=design&node-id=0-1&mode=design&t=z218OCInJPoqASvq-0
    - 1 hour (Feb 19)
    - 1 hour (Feb 29)
    - 1 hour
- Build a prototype website with HTML and CSS (10 hours)
    - 2 hours
    - 2 hours
    - 2 hours
    - 2 hours
    - 2 hours
- Add animation with Javascript & other libraries (10 hours)
- Connect the front end to back end

Challenge

- Database Switching
    - Initially, the project was set up using MongoDB Atlas, but due to new requirements from our instructor, we had to rebuild the database infrastructure from scratch. This task included reconfiguring servers, re-establishing sharding, and re-establishing client connections, all of which extended our timeline and added complexity to the project setup.
- Hosting on EC2:

- ○ Our team, not having access to high-specification local machines, opted to host our database on an EC2 instance. This decision allowed for better accessibility across various devices and user accounts.
- ○ However, the access management features of EC2 presented a steep learning curve. We had to navigate through a complex system of permissions and security settings, which was particularly challenging for team members unfamiliar with AWS's intricate access management system.

Updated Timeline

Milestone	**Wk1** (2 Feb)	**Wk2** (9 Feb)	**Wk3** (16 Feb)	**Wk4** (23 Feb)	**Wk5** (1 Mar)	**Wk6** (8 Mar)	**Wk7** (15 Mar)	**Wk8** (22 Mar)	**Wk9** (29 Mar)	**Wk10** (5 Apr)	**Wk11** (12 Apr)	**Wk12** (19 Apr)	**Wk13** (26 Apr)
Project Proposal	🟩												
Design													
Database		🟩											
Backend Interface			🟩	🟩									
Frontend Interface			🟩	🟩									
Database													
Implementation			🟩	🟩	🟨	🟨	🟨	🟨					
Testing										🟨	🟨		
DB Sharding				🟩	🟨	🟨	🟨	🟨	🟨				
Backend (Data Managers)													
Implementation					🟨	🟨	🟨	🟨					
Testing								🟨					
Frontend (End-Users)													
Implementation				🟩	🟨	🟨	🟨	🟨	🟨	🟨			
Testing									🟨	🟨	🟨		
Recommendation engine									🟨	🟨	🟨		
Final Delivery													
Full Demo											🟨	🟨	
Final Report													🟨