Large-Scale and Multi-Structured Databases

Project Design
PC Performance Evaluation on Games

Bruno Augusto Casu Pereira De Sousa

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

IN SUPREMÆ DIGNITATIS 1343

UNIVERSITÀ DI PISA

CROSSLAB
Innovation for industry 4.0

Application Highlights

The application proposed in this project is a platform for PC systems performance evaluation for Games. This type of system is useful as often when a new game is released many consumers asks questions like “Can I Run It?” or “What do I need to Run It?”.

In the proposed platform users will be able to browser trending games and check the system requirements, in terms of CPU, GPU and memory, to run those games.

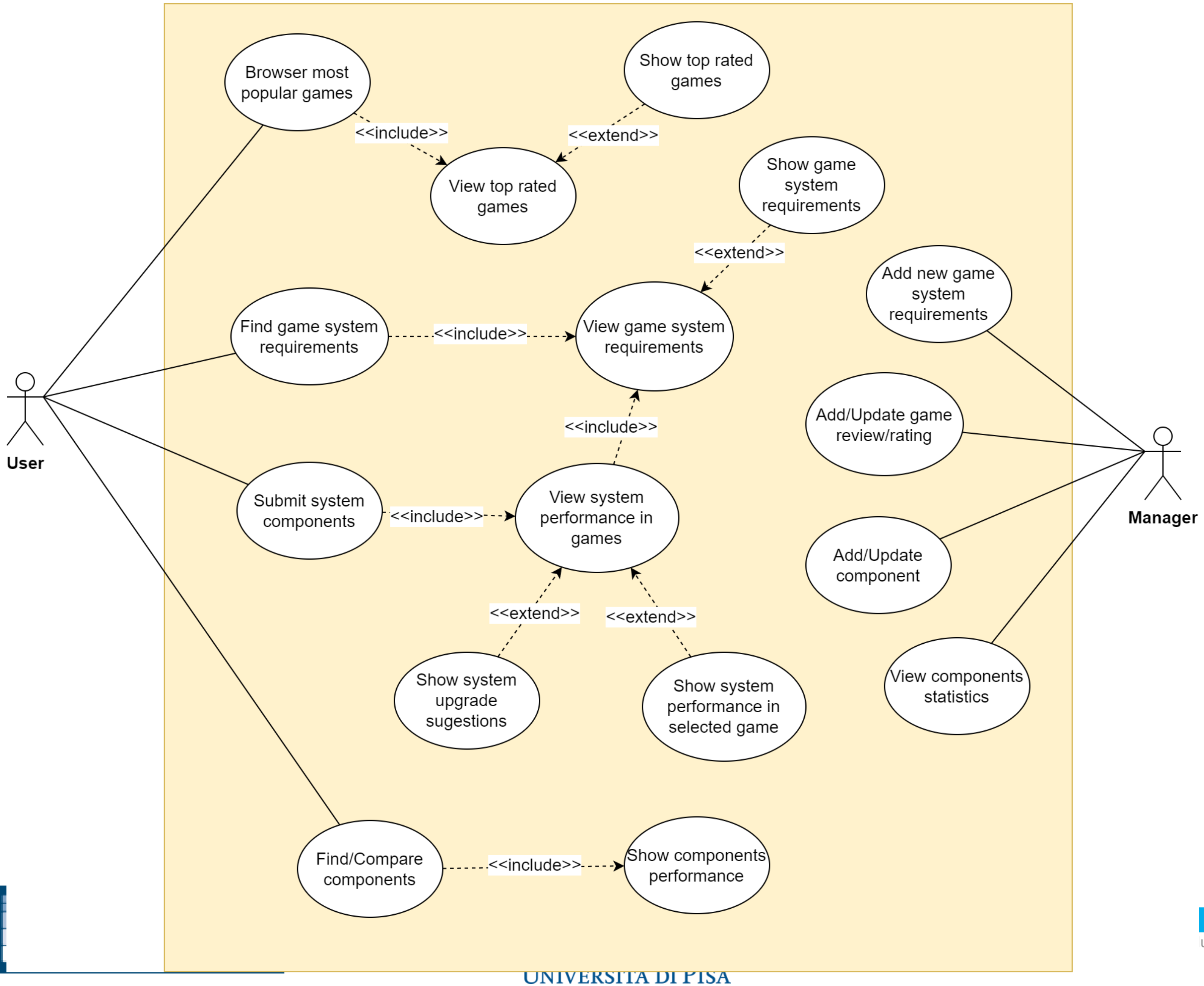
Also, users can submit their own system configuration (CPU model, etc.) and check if the system can run one or more specified games (meet the game system requirements). In the case of not meeting the requirements the application will suggest some hardware upgrades for the user. The User can also use the platform to check benchmarks and compare the PC components performance.

Application Highlights

Features:

1. *A User can browser the recent trending games*
2. *A User can check the PC system requirements for a game*
3. *A User can check the details of a PC component*
4. *A User can compare the specifications and benchmarks for 2 selected components*
5. *A User can submit his PC system configurations and see its performance evaluation in the trending games*
6. *A User can include one or more games in its submission and see his system performance on those games*
7. *The Manager can add/update games and its review and score info*
8. *The Manager can add/update a component and its benchmark score*
9. *The Manager can add a new game and its system requirements info*
10. *The Manager can check the systems statistics (popular components)*

Actors and main supported functionalities


Browser most popular games
Show top rated games
<<include>>
<<extend>>
View top rated games
Show game system requirements
<<extend>>
Add new game system requirements
Find game system requirements
<<include>>
View game system requirements
User
<<include>>
Add/Update game review/rating
Manager
Submit system components
<<include>>
View system performance in games
Add/Update component
<<extend>>
<<extend>>
Show system upgrade sugestions
Show system performance in selected game
View components statistics
Find/Compare components
<<include>>
Show components performance

Dataset Description

Source:
Game Info and Scores (metacritics)
https://www.kaggle.com/datasets/enesarslan8/metacritic-pc-games-of-all-time-2023-53k
Game Reviews (steam)
https://www.kaggle.com/datasets/whigmalwhim/steam-releases
Game System Requirements
https://www.kaggle.com/datasets/baraazaid/pc-video-game-requirements

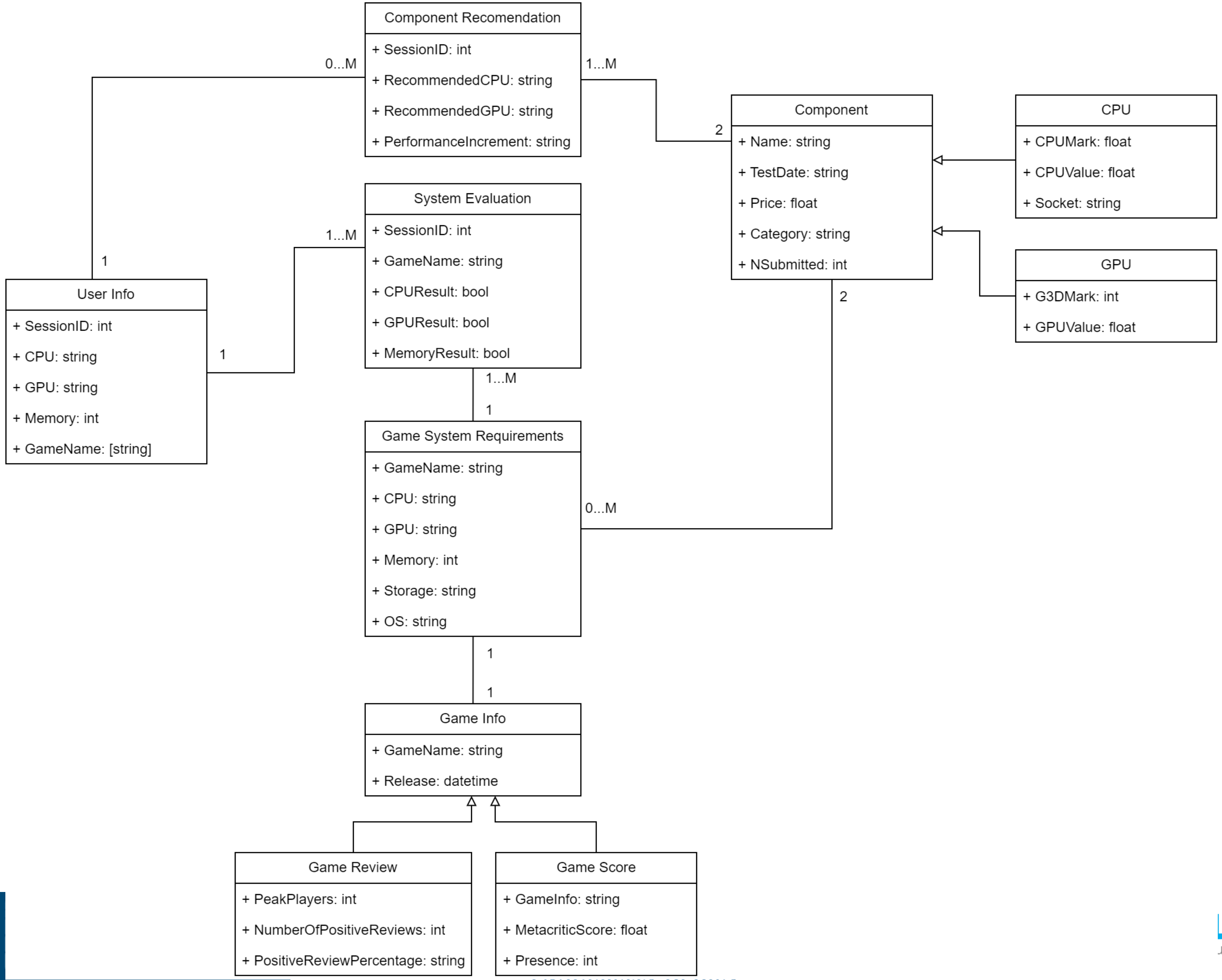
Description: *The Game Info and Scores dataset contains a brief description of the games, as well as some metrics to evaluate the game quality and relevance (Metacritic score and number of articles available). The Game Reviews dataset contains the Steam data on the max. number of concurrent players that were playing a game, as well as the number of positive reviews.*

Volume: *Game Info and Scores + Reviews (85MB); System Requirements (20MB)*

Variety: The Game Info and Scores dataset has a different format than the Steam Reviews, as they use different metrics to evaluate the games.

Velocity/Variability: Scores and the number of reviews contained in the dataset, for example, need to be frequently updated to identify top trending games.

Preliminary UML Class Diagram



Component Recomendation
+ SessionID: int
+ RecommendedCPU: string
+ RecommendedGPU: string
+ PerformanceIncrement: string

System Evaluation
+ SessionID: int
+ GameName: string
+ CPUResult: bool
+ GPUResult: bool
+ MemoryResult: bool

Component
+ Name: string
+ TestDate: string
+ Price: float
+ Category: string
+ NSubmitted: int

CPU
+ CPUMark: float
+ CPUValue: float
+ Socket: string

GPU
+ G3DMark: int
+ GPUValue: float

User Info
+ SessionID: int
+ CPU: string
+ GPU: string
+ Memory: int
+ GameName: [string]

Game System Requirements
+ GameName: string
+ CPU: string
+ GPU: string
+ Memory: int
+ Storage: string
+ OS: string

Game Info
+ GameName: string
+ Release: datetime

Game Review
+ PeakPlayers: int
+ NumberOfPositiveReviews: int
+ PositiveReviewPercentage: string

Game Score
+ GameInfo: string
+ MetacriticScore: float
+ Presence: int

Requirements and Entities handled by Document DB

Collections:

Three Collections are proposed to managed the information in the platform:

1. ***Game Info** (combines the Reviews, Scores and Description of the games)*
2. ***Game System Requirements** (reference the list of components that are the minimum required to run the game)*
3. ***Components** (maintain the information and benchmarks for the PC components)*

Operations (examples):

1. *The trending games can be present with different sort aggregations, such that a user can select the Order displayed as, for example, better reviewed or highest score, or highest number of articles etc.*
2. *When providing a suggestion for a system upgrade, the platform must execute aggregation operations to sort and select components based on recent tests results, the best value and can even include a component popularity metric, based on computed statistics.*
3. *The component statistics can be computed by analyzing the number of times certain components were submitted by users (like a Hardware Survey) or by including a review on the recommendation provided by the platform.*

Requirements and Entities handled by Key-Value DB

Entities:

Three Entities are proposed to managed the user system info and the evaluation results:

1. ***User Info*** *(contains the user system components and if added, one or more games to be evaluated, mapped in a session). <Key> {Value} examples:*

   <session:1:cpu> {string}
   <session:1:gpu> {string}
   <session:1:memory> {int}
   <session:1:games> {[string]}

2. ***System Evaluation:*** *contains the result (OK or NOK) for one game.*

   <session:1:game:1:result:cpu> {bool}
   <session:1:game:1:result:gpu> {bool}
   <session:1:game:2:result:cpu> {bool}
   <session:1:game:2:result:gpu> {bool}

3. ***Component Recommendation*** *(references the list of components suggested for the upgrade)*

   <session:1:cpu:recommended> {string}
   <session:1:gpu:recommended> {string}

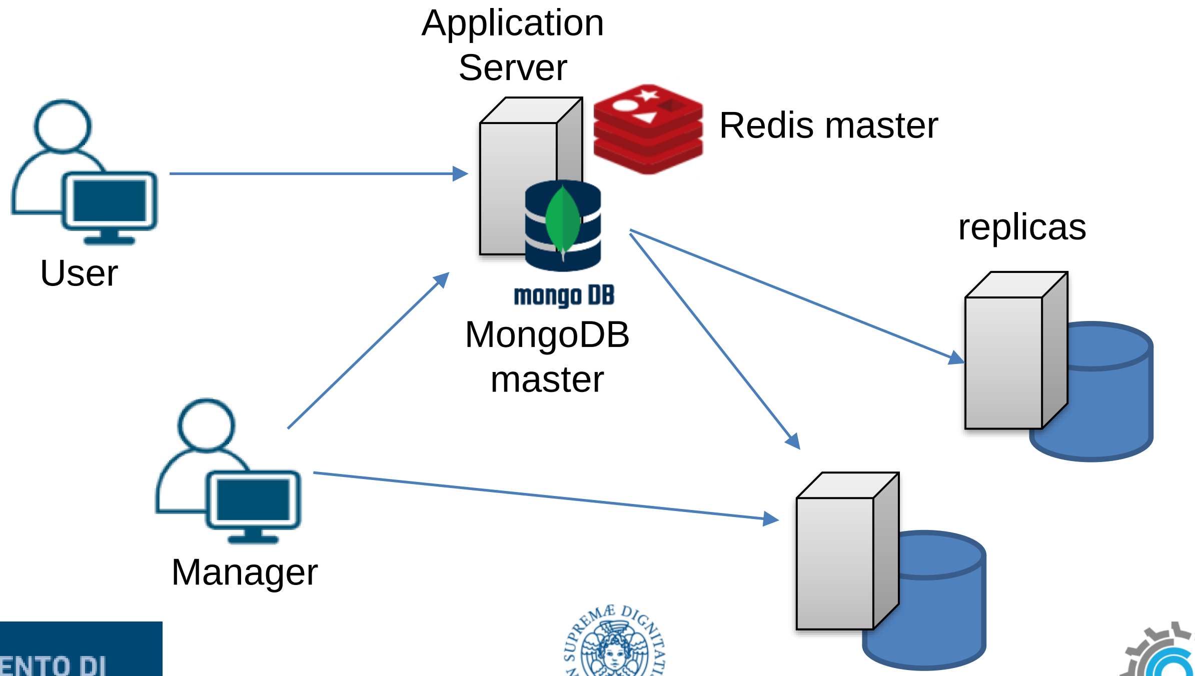
Operations (examples):

1. *After the user system evaluation is completed for all games, the platform must then quickly get a component suggestion for that Session. This can be done by parsing the evaluations and identify which is the highest requirement. With that, the platform can suggest one component that satisfies ALL the games submitted.*

Software Architecture Preliminary Idea

The selected DBMS for the Document DB proposed is **MongoDB**, and the application will be developed using Python and the available library “pymongo”.

The DBMS for the Key-Value DB proposed is **Redis**, and the application will be developed using Python and the available library ”redis-py”.


Application Server
Redis master
replicas
User
mongo DB
MongoDB master
Manager