THKU Department of Software Engineering

SENG 454 Cloud Systems and Networks

Term Project

Deadlines

Forming Groups: 27.03.25 - 23.59

Code and Report Submission: 04.05.25 - 23.59

Presentations: 05.05.25 & 14.05.25 (In Class)

---

1. OBJECTIVE

This project aims to help you familiarize with Cloud Development. You will develop and deploy a simple application on a Platform as a Service (PaaS) platform that connects to an external Database as a Service (DBaaS) platform. You are free to choose your own PaaS service and the programming language. However, the database should be NoSQL which is deployed on MongoDB Atlas.

2. VIDEO GAME DISTRIBUTION SERVICE

- There are digital video game distribution services like Steam and Epic Games Store that people can buy, play, rate and comment on games. In this project, you will be implementing a simplified version of these where the games are free and no one needs to buy or pay.

- You will implement a cloud application that communicates with your MongoDB Database on Atlas. Your database will be tested through this code.

- There will be at least 3 main pages: **Home Page**, **User Page**, and **Games Page**

- **Home Page** is the start page which will be accessed through the address of your application. There should be at least 7 actions that can be applied on this page: **Add Game**, **Remove Game**, **Disable Rating and Comment**, **Enable Rating and Comment**, **Add User**, **Remove User** and **Login as a User**.
    - **Add Game** creates an entry of a game with these 6 required attributes (+2 optional ones): **Name**, **Genre(s)**, **Photo** (No need to implement a way

    to upload, photo attribute can be initialized with a link to an online image). Game initially has no **Play Time**, **Rating** and **All Comments** and it is initially enabled to be rated and commented on. For the **optional attributes**, it should be free to choose what to add. (i.e. they can be blank, the release date, PC requirements, developer name, some advertisement quotes or else. There should be no limit on what to add thanks to NoSQL)

    - **Remove Game** deletes the game and all of its **Play Time**, **Rating** and **Comments**. (The user's total play time attribute will not be affected but the other attributes such as **Most Played Game**, **Average of Ratings** or **Comments** on user pages should be updated).

    - **Disable Rating and Comment** disables rating and commenting option for a game. Old ratings and comments are **not deleted**. It can still be played but it can not be rated or commented on unless an enable operation is called afterwards.

    - **Enable Rating and Comment** enables rating and commenting options for the game. It can be played, rated and commented on.

    - **Add User** creates an entry of a user with a **Name** attribute. (User initially has no **Total Play Time**, **Average of Ratings**, **Most Played Game** and **Comments**).

    - **Remove User** deletes an entry of a user and all of its attributes. When you delete a user, all of the effects of its actions are also deleted. This means that the games that the user played, rated and commented before are also affected. (i.e. remove the user's comments from **All Comments**, subtract the user's play time from **Play Time** of games, and calculate **Rating of the Game** without the deleted user's rating)

    - **Login as a User** navigates through the homepage of the selected user as him or her. (Note that you don't need to implement passwords or any kind of protection).

    - Note that in order to execute Remove, Disable, Enable, Login actions you need to implement a way to access the relevant **Games** and **Users**. (i.e. You may use dropdown lists, or they can be written manually by looking at a dynamically updated table that is also available on **Home Page**).

- **User Page** shows 5 different elements: **User Name**, **Average of Ratings**, **Total Play Time**, **Most Played Game** and **Comments**. **Average of Ratings** is the mean of all ratings given by the user. (For this attribute play time does not have any effect). **Total Play Time** is the sum of playtimes of this user on all games. **Most Played Game** is the game that the user played the most. It is not important which one is shown in case of equality. **Comments** are the comments of the user given on different games. The order of the comments is done by the user's own play time on the games that are commented on. (Game names are also shown with the comments)

- There should be at least 4 actions that can be applied by a user: **Rate Game**, **Play Game**, **Comment on a Game**, **Look Games**.

    - **Rate Game** rates a game with an integer point between 1 and 5. When a game is rated, the **Average of Ratings** of **User Page** and **Rating of the Game** on **Games Page** are updated. For this action to be executed there is a **prerequisite** that the user played this game before (i.e. at least 1 hour of play time).

    - **Rating of the Game** is an attribute of Games Page determined by $(\Sigma(UserPlayTime * UserRating\ )/PlayTimeOfGame$, i.e. each user contributes to the average of a game with respect to their play time on the same game. If a user rates a game more than once, the old value is updated (i.e. the old rating is overwritten by the new one).

    - **Play Game** increments the **Total Play Time** of the user by the given value. It also updates the **Rating of the Game** on the **Games Page**. After this action is executed, the order of the **Comments** on this page and the order of the **All Comments** attribute on the **Games Page** may also change according to the new play time values. **Most Played Game** attributes should also be checked for update after this operation.

    - **Comment on a Game** adds a comment of a game on the **Comments** attribute. For this action to be executed there is also a **prerequisite** that the user played this game before (i.e. at least 1 hour of play time). Users can not comment on the games that they did not play. This action also updates the **All Comments** element on the **Games Page**. If a user comments on a game more than once, the old value is updated (i.e. old comment is overwritten by the new one).

    - **Look Games** navigates to the **Games Page**.

    - Note that in order to execute rate, play and comment operations you need to implement a way to access the relevant **Games**. (i.e. You may use dropdown lists, or they can be written manually by looking at a dynamically updated table that is also available on **User Page**).

- **Games Page** shows all the games with each having 6+2 different attributes: **Name**, **Genre(s)**, **Photo**, **Play Time**, **Rating**, **All Comments** and 2 optional attributes. **Name** is the name of the game. **Genre(s)** is a list of 1 to 5 elements determining the genre of the game. **Photo** is the image that represents the game. (You need to show the image on this page). **Play Time** is the sum of the play time of the game by all users. Rating is the weighted average of all ratings on a game. (Weight is determined by the play time. The formula is $\Sigma(UserPlayTime*UserRating)/PlayTimeOfGame$). **All Comments** is a list of comments on a game given by all users with user names also shown. (The order is determined by the play time. i.e. you see the comment of a user who plays the game most before the one who played less. The equality case is not important). **Optional attributes** should also be shown if they exist.

- Before submission and after project presentation, your database should have at least 10 different games, 10 different users; and at least 3 of the users with each played more than 3 games, rated 2 games and commented on 2 games. At least 3 of the games should have the optional fields filled with different kinds of elements.

3. PROJECT SPECIFICATIONS

a. Group or Individual Work:

- You may develop your application individually or as a group. Each group can consist of at most 6 students. All of the group members are responsible for the whole project. They all have to be available on presentation days and they should be able to answer all kinds of questions about the project.

b. Technical Specifications

- Your web application should be executing on a PaaS platform. It should be usable from anywhere with Internet connection. It does not need to be suitable for every device. Just state in your report that whether you developed it for PC or mobile.

- You are free to use any programming language that is supported by the PaaS platform you have chosen.

- You have to use MongoDB Atlas as your database. Your PaaS platform might also offer an internal MongoDB database, but you should not use it. You have to use Atlas and connect your application to your Atlas database **externally**.

c. **Report Specifications:**

You should submit your project with your project report. Your report should include;

- Your choice of PaaS platform
- Your choice of programming language
- The external libraries you used in your implementation
- The names of the existing users on the database
- The prompt history and the parts offered by AI, if you have gotten any help from an AI tool.

4. Useful Links

- https://www.mongodb.com/products/platform/atlas-database
- https://www.mongodb.com/resources/products/platform/mongodb-atlas-tutorial
- https://www.mongodb.com/docs/atlas/
- https://cloud.google.com/appengine/docs/an-overview-of-app-engine
- https://learn.microsoft.com/en-us/azure/app-service/overview
- https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html
- https://dokku.com/docs/getting-started/installation/

5. Submission & Presentation

- You should decide on your groups and inform me through Teams until 27.03.25 - 23.59. Otherwise, It means your work will be individual. A message by just a single member of a group is sufficient.
- You are expected to submit your codes and reports through Teams until 04.05.25 - 23:59. The submission by just a single member of a group is sufficient.
- For the in-class presentations, groups should access their applications and execute some commands to show it is working properly. They should also show their code and explain which part is responsible for which operation.
- Hint: Check the "*edge cases*" carefully before submission as the main difference of grading between groups would be according to those.