Middle East Technical University
Department of Computer Engineering

CENG 495
Cloud Computing
Spring 2019-2020 Homework 2

Due Date: 02.06.2021, 23:55

This homework aims to get you familiar with the NoSQL databases and Database as a Service (DBaaS) platforms. MongoDB is classified as a NoSQL database program, which is open-source. As a document store database, it uses JSON like queries to handle the database operations. Atlas is the cloud platform of MongoDB. You are going to develop and deploy a simple digital video game distribution service using MongoDB database on Atlas.

Keywords: PaaS, DBaaS, BaaS, MongoDB, JSON, Atlas, Realm, GAE, Cloud Computing

1. MongoDB Atlas

- Create an Atlas account as a free user. (mongodb.com/cloud/Atlas)
- Create a database with MongoDB and deploy it into Atlas.
- Use the MongoDB Realm, which includes Backend as a Service Platform (BaaS) of Atlas (formerly Stitch) to communicate with your database through an html code. Alternatively, you can use GAE to reach your MongoDB database.

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 homework, 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 an html code that communicates with your MongoDB on Atlas through Realm Web SDK(see https://docs.mongodb.com/realm/web/quickstart/). Your database will be tested through this code. Or you can just deploy an application on GAE that communicates with MongoDB.

- 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 index.html or via your GAE address. 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 it's **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 option 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 it's attributes. When you delete a user, all of the affects of it's 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 play times 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. (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, **Average of Ratings** of **User Page** and **Rating of the Game** on **Games Page** are updated. (Rating of the Game is an attribute of Games Page determined by $(\sum(UserPlayTime * UserRating)/PlayTimeOfGame$, 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. 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 **Rating of the Game** on **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** attribute 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 **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 $\sum(UserPlayTime * UserRating)/PlayTimeOfGame$). **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, 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. Useful Links

- mongodb.com/cloud/atlas
- mongodb.com/cloud/realm
- docs.mongodb.com/realm
- docs.mongodb.com/realm/web
- docs.mongodb.com/
- tutorialspoint.com/mongodb/

4. Submission

- In this assignment, you are expected to submit your html code(s) through ODTÜClass. The main file should be named as “index.html”. For submission on ODTÜClass, a tar.gz archive file (named hw2.tar.gz) that contains all your source code files and a

README file that includes the design choices you made and a user's guide. If you want to use GAE instead of html code, just include your GAE source codes and give your app link on the README file.

- Do not forget to change the privacy settings on Atlas since your database will be reached from an IP address other than yours for grading. (To be sure, just execute a final test for your webapp from a different internet connection or using VPN before submitting your work).
- The work you submit should be implemented by only you and genuine. However, you can use external libraries for graphical user interface if any. If you do so, you need to state your references for these codes in your README file.
- We have zero tolerance policy for cheating. There is no teaming up! People involved in cheating will be punished according to the university regulations and will get 0. You can discuss design choices or language preferences, but sharing code between each other or submitting third party code as a whole is strictly forbidden. In case a match is found, this will be considered as cheating.