**Team Name: Dolphins** (group #: 022-1)

**Team Meeting Time:** Sundays 3:00-5:00 pm

**Team Meeting Link:** https://cuboulder.zoom.us/j/4508042408

Github Repository Link

**Team Meeting with TA Time:** Tuesday 5:05-5:20pm

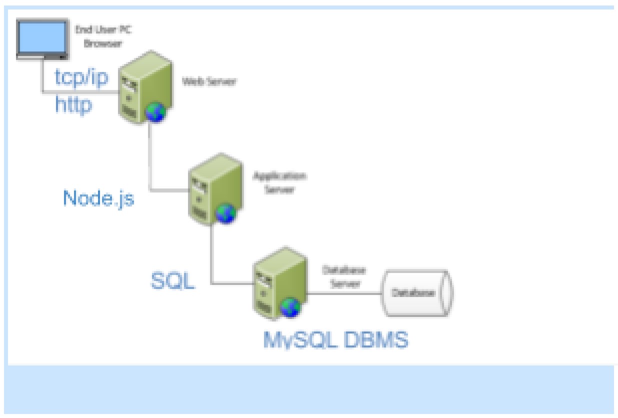
**Team Meeting with TA Link:** https://cuboulder.zoom.us/j/97473414998 **(Password : 994524)**

**Team JIRA Board Link:** https://niketh49.atlassian.net/jira/software/projects/C3S01/boards/1

**Ross Panning, Jack LeGrone, Niketh Gorla, Spencer Nikolaeff, Xizhe (Caesar) Song**

| | | |
|---|---|---|
| **Revised List of Features** | **This is an updated list of your FEATURES inventory (from Milestone 2). It is normal for feature lists to change during the course of a project. Some features may have been dropped.** | 1. Account Creation & Security<br>a. Username and Password protection<br>2. Personal Games Library<br>a. List of Games, formatted with what device you play them on<br>b. Ability to add, remove, edit game listings<br>c. What Type of Software you use (keyboard, mouse, PC specs, controllers) |

| | | |
|---|---|---|
| | **Some features may have been added. This revised features list should reflect these changes. This revised features list should identify the PRIORITY order of how the features will be developed.** | 3. In Depth Games Library<br>a. Time played<br>b. Ranking / Specialty in Game<br>c. Achievements<br>4. Further Integration<br>a. Link to discords<br>b. Link to Game Wikis<br>c. Recommended Games<br>d. Games Wish List |

| | | |
|---|---|---|
| Architecture Diagram | This deliverable is a picture or diagram that shows each architectural component of your application.<br>The diagram should identify how your application's front-end, integration layer, and backend processes will be hosted.<br>This diagram should identify the flow of data from one layer to another.<br>This diagram should identify the |  End User PC Browser<br>tcp/ip<br>http<br>Web Server<br>Node.js<br>Application Server<br>SQL<br>Database Server<br>Database<br>MySQL DBMS |

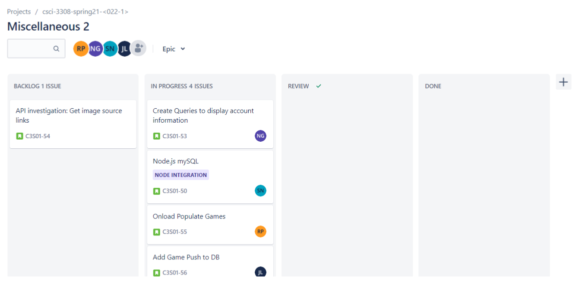
| | | |
|---|---|---|
| | **protocols being used to/from each component layer.**<br><br>**t this point your architecture is supposed to be final and you are expected to continue development for your project in the architecture specified here. You may deviate from it if required but we need a specific diagram at this point detailing work and data flow across the different** | |

| | | |
|---|---|---|
| | layers of your application. | |
| Front End design | This deliverable is a series of diagrams that show the basic design of your application's front end. Typically, this design is most easily presented in terms of a WIREFRAME. This may be hand drawn, or created using a web page wireframe drawing tool. The front end design should identify each major | Diagram down below |

|  |  |  |
|---|---|---|
|  | **feature of the application's front end** |  |
| **Web Service Design** | **If your application is using Web Services via APIs, this deliverable should list the Web Service being used along with a description of the Web Service's API including the data being passed to and received from the API.** | We plan to use IGDB.com to source video game image urls to store in the mysql database. To do this, we will pass the game name to the API, then request the game id of the game, then use this ID to fetch the url of the image of the cover. The url will then be pushed to the database. |

| | | |
|---|---|---|
| Database design | This deliverable provides a summary design of your application’s database.<br>The design document should identify each type of data being stored in your database.<br>This may be documented in terms of a schema definition, showing data entities (“files”) and attributes (“fields”)<br>This may be documented via an Entity Relationship | Database Software: MySQL<br>We have 6 relations in our database. The first one is User. This relation stores information about users who use Librario including username, password, recovery question, etc. The next table is the games relation. This table stores information about each game in our database. The user and the game relations are related by many to many, so we are using the Usergame relation to resolve this conflict.(Usergame is identifying). The usergame relation stores information specific to the user, about a specific game. For example: the individual’s personal review of the game. The Usergame table is related in a one to many identifying relationship with Achievements. This is the table that stores the specific information the users achievements in that specific game. |

| | | |
|---|---|---|
| | Diagram showing database tables and columns. The document should identify the specific DBMS technology being used to store your application data (PostgreSQL, MySQL, Firebase, etc.) | Finally there is the tags relation, this stores information describing the game. A game can have many tags, and a tag can be used to describe many games. Hence, the game tags relation was used to resolve this many to many. |
| Individual Contributions | ● This deliverable includes a couple of lines about each team member's contribution towards the project. | Ross Panning<br>- Updated, and refined database<br>- Created local host of database using node.js<br>- [https://github.com/CSCI-3308-CU-Boulder/3308SP21_section022_1/commit/6d5130d51de5baa94600bb5cd6cb6c966a943b47#diff-1e23f7](https://github.com/CSCI-3308-CU-Boulder/3308SP21_section022_1/commit/6d5130d51de5baa94600bb5cd6cb6c966a943b47#diff-1e23f7) |

| | | |
|---|---|---|
| | ● **Include a link to the latest commit made by each team member on the GitHub repository.**<br>● **Share a screenshot of the project management board being maintained for this project indicating the status of the tasks at hand.** | **f59d2516cab4636978de01286468ed0209337aee8cbb5eaa0817eccb59**<br>**Jack Legrone: Latest Commit**<br>- **JS add game methods**<br>- **Converted HTML files to ejs**<br>**Spencer Nikolaeff**<br>- **Node.js MySQL connection**<br>- **Node.js MySQL queries**<br>- **File management**<br>- **https://github.com/CSCI-3308-CU-Boulder/3308SP21_section022_1/commit/315baef3d78a415fad91a46d437b2c43d86d8954**<br>**Niketh Gorla**<br>**Xizhe Song**<br>- **AFK**<br><br>Projects / csci-3308-spring21-<022-1><br>Miscellaneous 2<br>RP NG SN JL Epic<br>BACKLOG 1 ISSUE: API investigation: Get image source links (C3S01-54)<br>IN PROGRESS 4 ISSUES: Create Queries to display account information (C3S01-53, NG); Node.js mySQL, NODE INTEGRATION (C3S01-50, SN); Onload Populate Games (C3S01-55, RP); Add Game Push to DB (C3S01-56, JL)<br>REVIEW ✓<br>DONE |

|  |  |  |
|---|---|---|
| Challenges | • Identify at least 3 challenges and/or risks to your project at this point.<br>• What is your backup/risk mitigation plan incase you are unable to resolve the challenge(s)? | 1. **Adding base games list to application**<br>  a. **We will have to manually add games**<br>2. Group member attendance<br>  a. We will lessen our specialization to work together on broad topics<br>3. File Uploads for Images, Game Data<br>  a. We will enact a placeholder or set data |

## To Do:

***Pushing and Pulling from the Server***

***Add link to images***

***Create Queries from front end***

## Revised List of Features

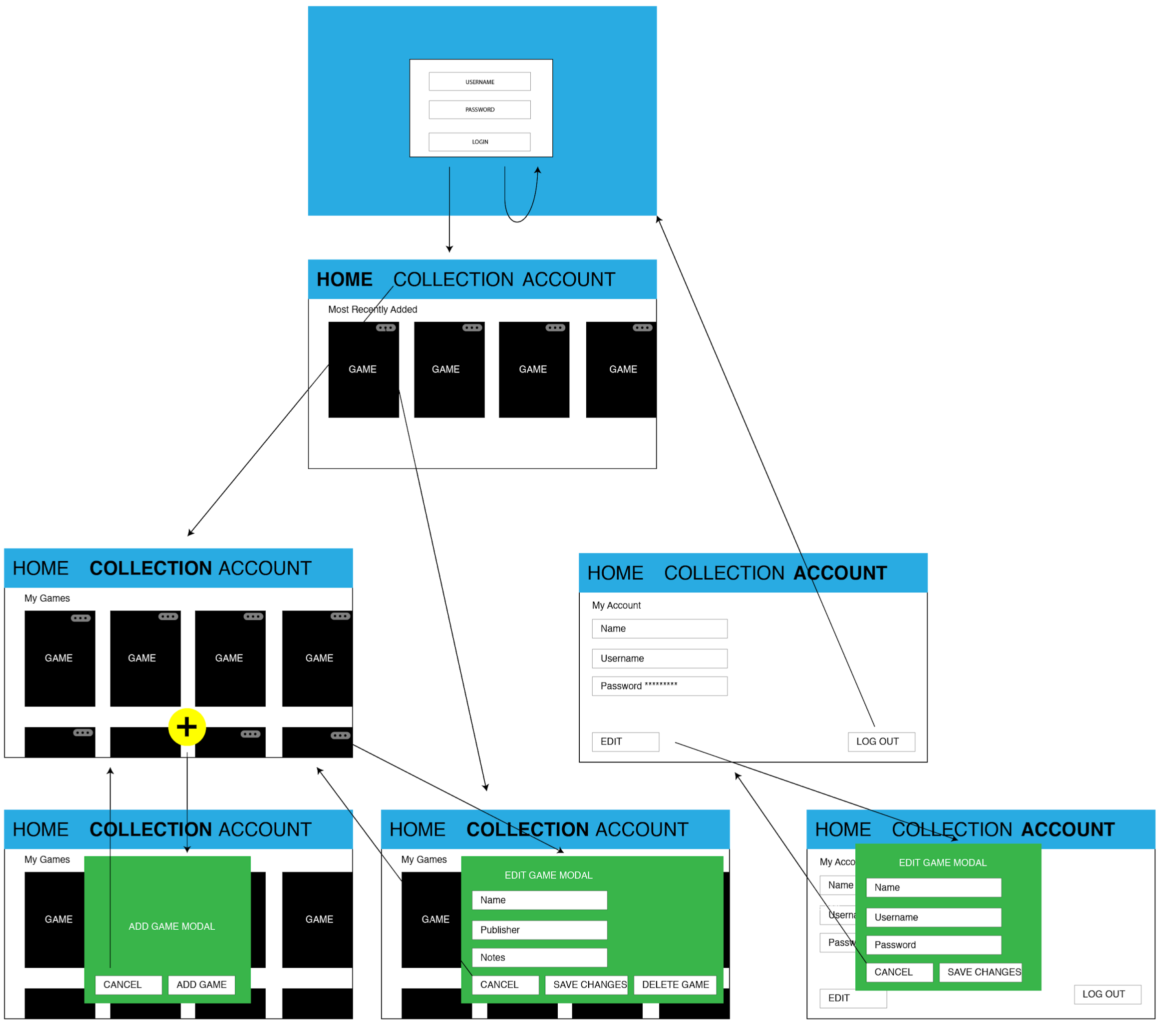
**Milestone 2 Features:**

https://docs.google.com/document/d/1ixoDDU8qErrd5WeL-dnn3oCQOUmNPfOZ4GCM1-FvGWc/edit

**Updated List of Features:**

## Architecture Design

## Front End Design



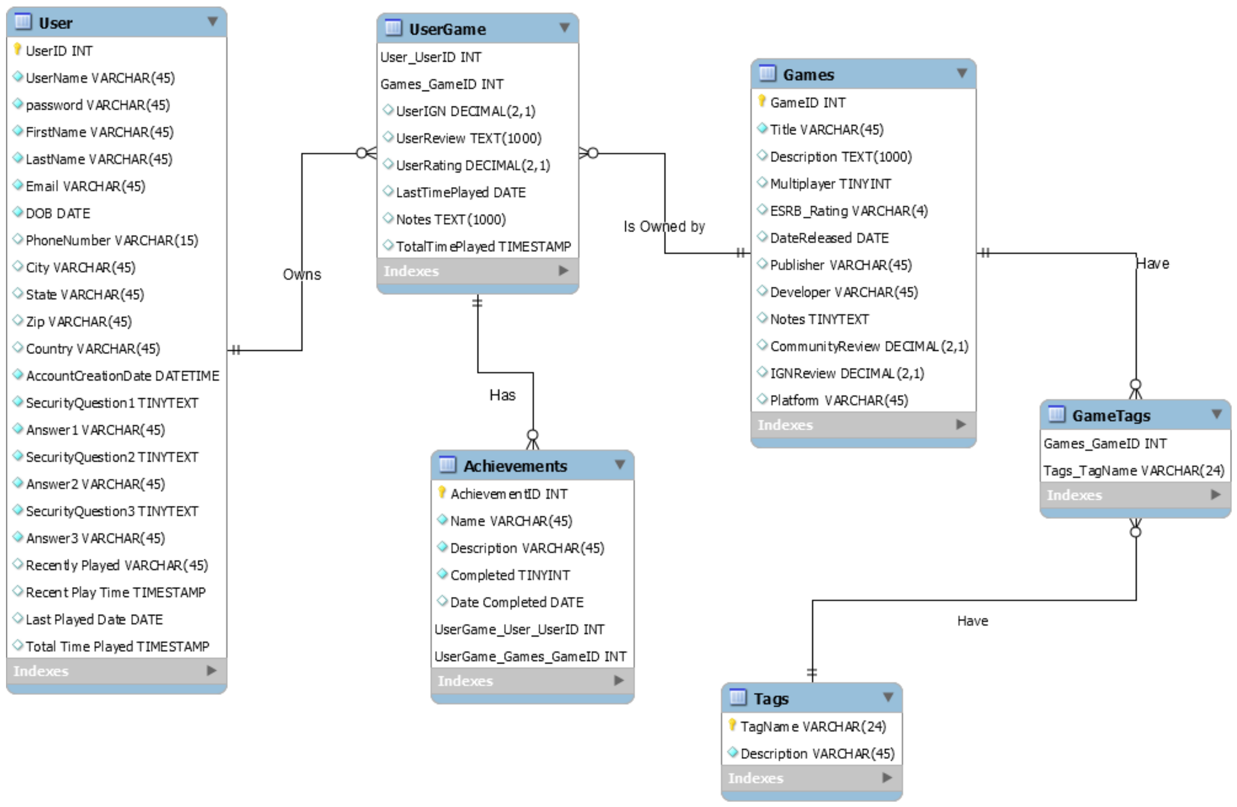
## Web Service Design

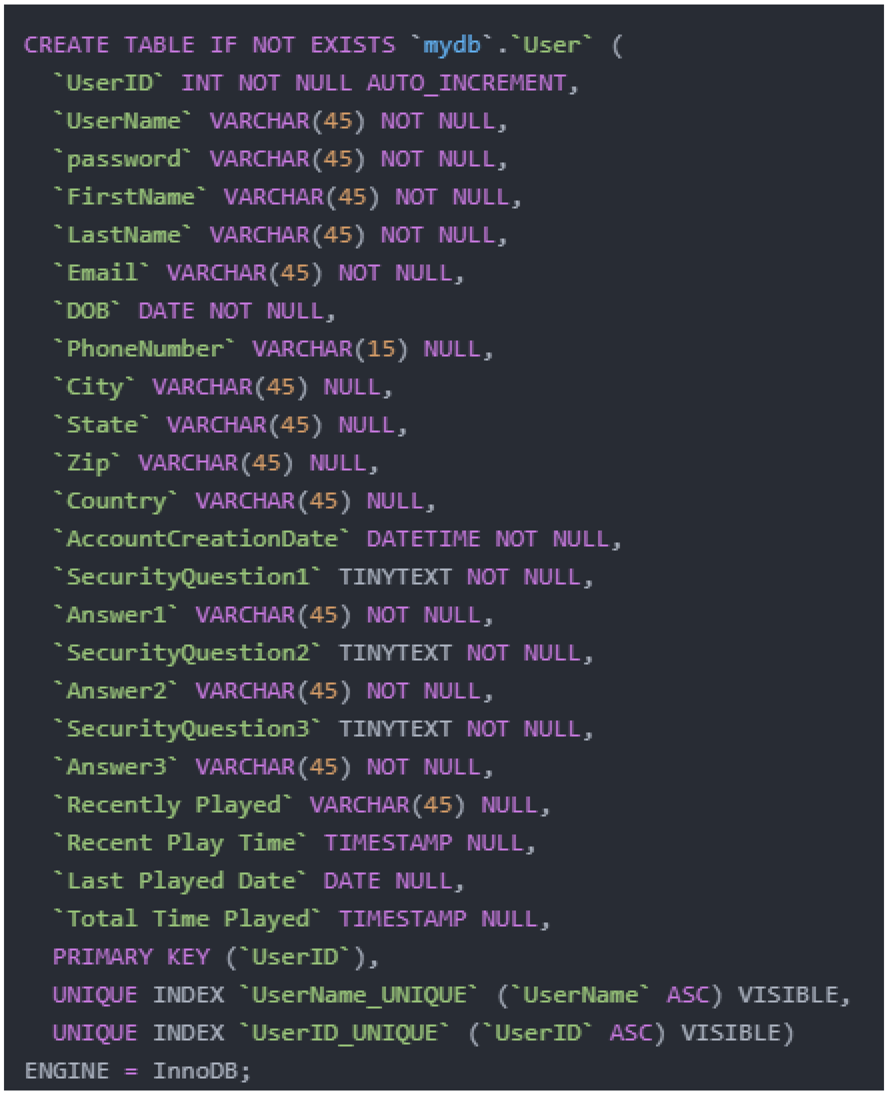
We plan to use IGDB.com to source video game image urls to store in the mysql database. To do this, we will pass the game name to the API, then request the game id of the game, then use this ID to fetch the url of the image of the cover. The url will then be pushed to the database.

## Database Design

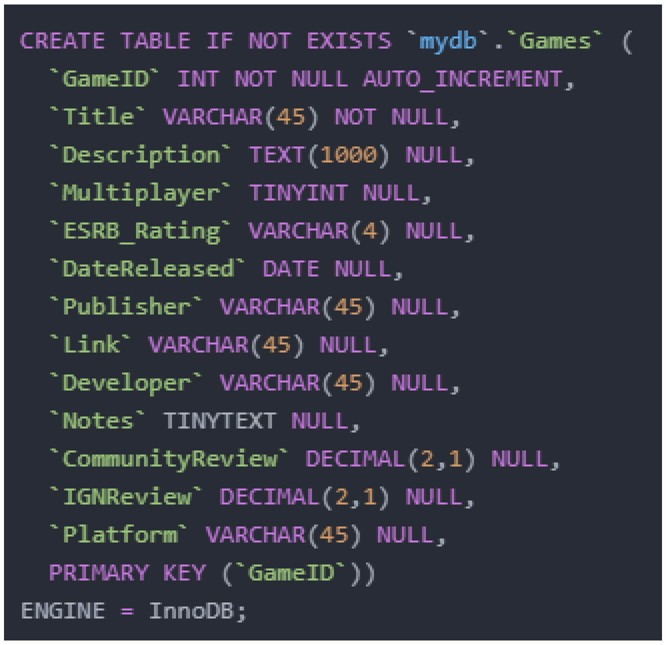
## Individual Contributions

## Challenges

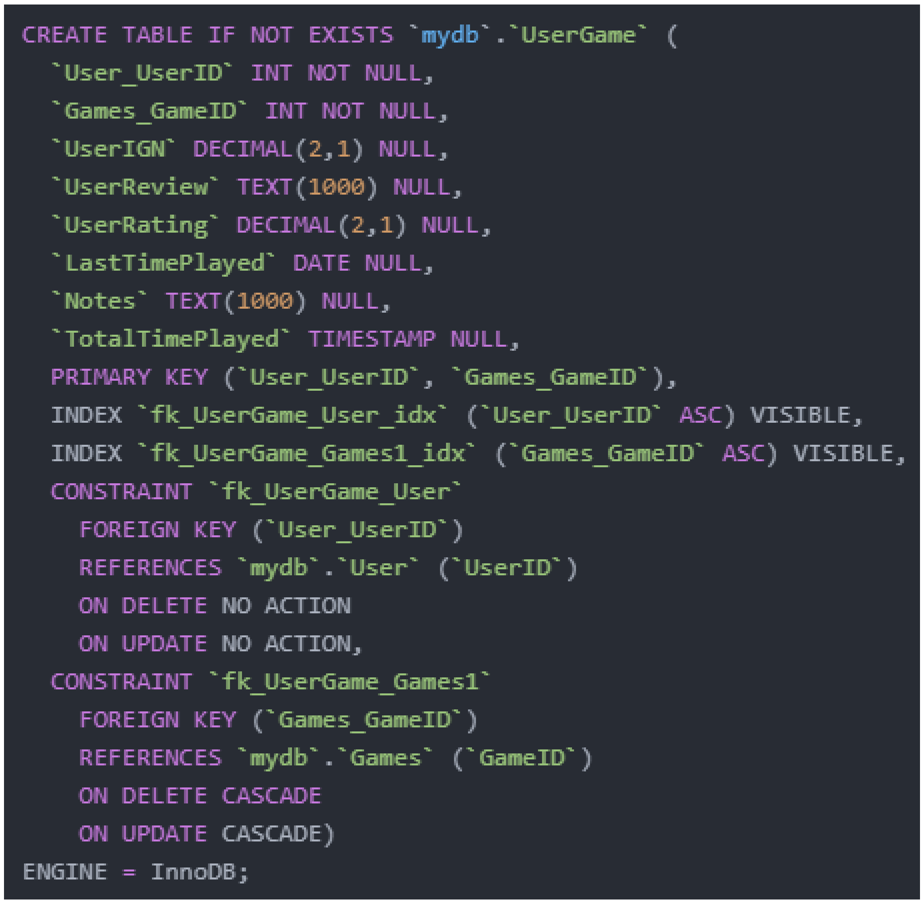

User
UserID INT
UserName VARCHAR(45)
password VARCHAR(45)
FirstName VARCHAR(45)
LastName VARCHAR(45)
Email VARCHAR(45)
DOB DATE
PhoneNumber VARCHAR(15)
City VARCHAR(45)
State VARCHAR(45)
Zip VARCHAR(45)
Country VARCHAR(45)
AccountCreationDate DATETIME
SecurityQuestion1 TINYTEXT
Answer1 VARCHAR(45)
SecurityQuestion2 TINYTEXT
Answer2 VARCHAR(45)
SecurityQuestion3 TINYTEXT
Answer3 VARCHAR(45)
Recently Played VARCHAR(45)
Recent Play Time TIMESTAMP
Last Played Date DATE
Total Time Played TIMESTAMP
Indexes
Owns
UserGame
User_UserID INT
Games_GameID INT
UserIGN DECIMAL(2,1)
UserReview TEXT(1000)
UserRating DECIMAL(2,1)
LastTimePlayed DATE
Notes TEXT(1000)
TotalTimePlayed TIMESTAMP
Indexes
Is Owned by
Games
GameID INT
Title VARCHAR(45)
Description TEXT(1000)
Multiplayer TINYINT
ESRB_Rating VARCHAR(4)
DateReleased DATE
Publisher VARCHAR(45)
Developer VARCHAR(45)
Notes TINYTEXT
CommunityReview DECIMAL(2,1)
IGNReview DECIMAL(2,1)
Platform VARCHAR(45)
Indexes
Have
Has
Achievements
AchievementID INT
Name VARCHAR(45)
Description VARCHAR(45)
Completed TINYINT
Date Completed DATE
UserGame_User_UserID INT
UserGame_Games_GameID INT
Indexes
GameTags
Games_GameID INT
Tags_TagName VARCHAR(24)
Indexes
Have
Tags
TagName VARCHAR(24)
Description VARCHAR(45)
Indexes



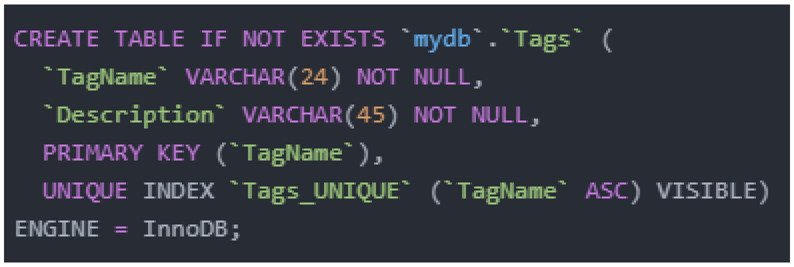
```sql
CREATE TABLE IF NOT EXISTS `mydb`.`User` (
  `UserID` INT NOT NULL AUTO_INCREMENT,
  `UserName` VARCHAR(45) NOT NULL,
  `password` VARCHAR(45) NOT NULL,
  `FirstName` VARCHAR(45) NOT NULL,
  `LastName` VARCHAR(45) NOT NULL,
  `Email` VARCHAR(45) NOT NULL,
  `DOB` DATE NOT NULL,
  `PhoneNumber` VARCHAR(15) NULL,
  `City` VARCHAR(45) NULL,
  `State` VARCHAR(45) NULL,
  `Zip` VARCHAR(45) NULL,
  `Country` VARCHAR(45) NULL,
  `AccountCreationDate` DATETIME NOT NULL,
  `SecurityQuestion1` TINYTEXT NOT NULL,
  `Answer1` VARCHAR(45) NOT NULL,
  `SecurityQuestion2` TINYTEXT NOT NULL,
  `Answer2` VARCHAR(45) NOT NULL,
  `SecurityQuestion3` TINYTEXT NOT NULL,
  `Answer3` VARCHAR(45) NOT NULL,
  `Recently Played` VARCHAR(45) NULL,
  `Recent Play Time` TIMESTAMP NULL,
  `Last Played Date` DATE NULL,
  `Total Time Played` TIMESTAMP NULL,
  PRIMARY KEY (`UserID`),
  UNIQUE INDEX `UserName_UNIQUE` (`UserName` ASC) VISIBLE,
  UNIQUE INDEX `UserID_UNIQUE` (`UserID` ASC) VISIBLE)
ENGINE = InnoDB;
```



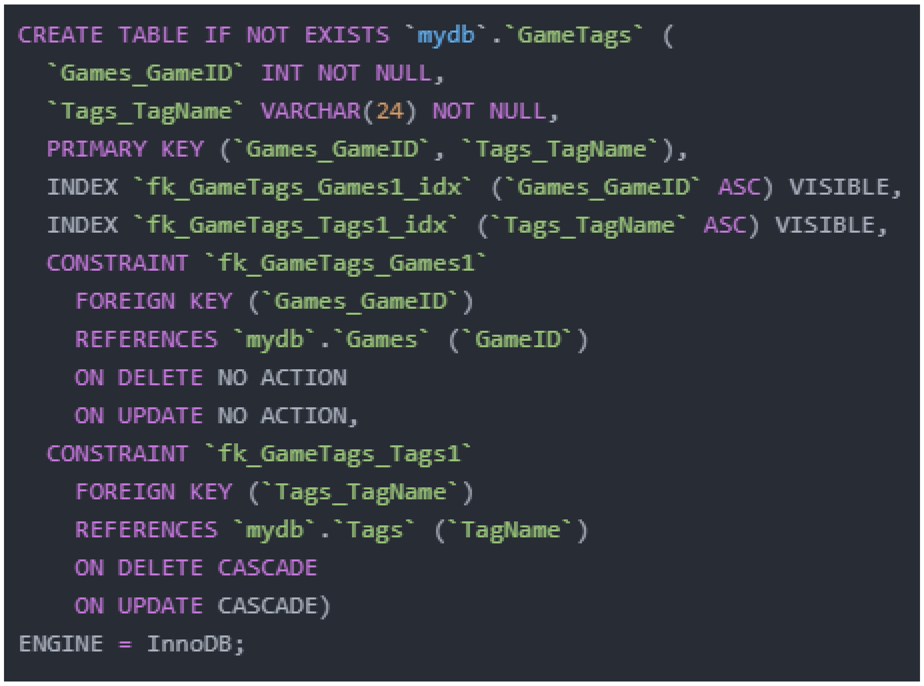
```sql
CREATE TABLE IF NOT EXISTS `mydb`.`Games` (
  `GameID` INT NOT NULL AUTO_INCREMENT,
  `Title` VARCHAR(45) NOT NULL,
  `Description` TEXT(1000) NULL,
  `Multiplayer` TINYINT NULL,
  `ESRB_Rating` VARCHAR(4) NULL,
  `DateReleased` DATE NULL,
  `Publisher` VARCHAR(45) NULL,
  `Link` VARCHAR(45) NULL,
  `Developer` VARCHAR(45) NULL,
  `Notes` TINYTEXT NULL,
  `CommunityReview` DECIMAL(2,1) NULL,
  `IGNReview` DECIMAL(2,1) NULL,
  `Platform` VARCHAR(45) NULL,
  PRIMARY KEY (`GameID`))
ENGINE = InnoDB;
```



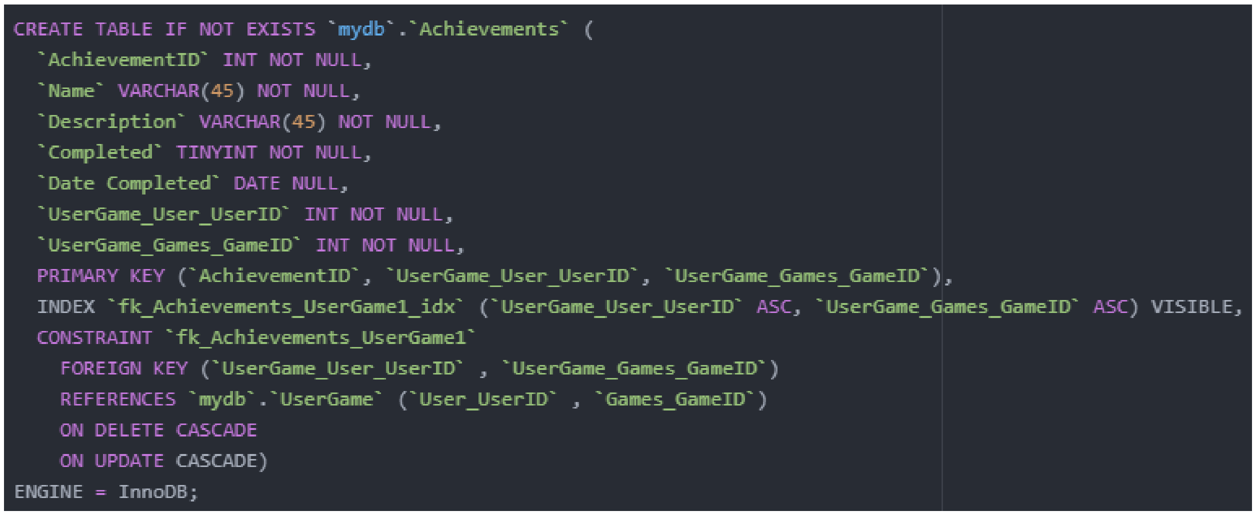
```sql
CREATE TABLE IF NOT EXISTS `mydb`.`UserGame` (
  `User_UserID` INT NOT NULL,
  `Games_GameID` INT NOT NULL,
  `UserIGN` DECIMAL(2,1) NULL,
  `UserReview` TEXT(1000) NULL,
  `UserRating` DECIMAL(2,1) NULL,
  `LastTimePlayed` DATE NULL,
  `Notes` TEXT(1000) NULL,
  `TotalTimePlayed` TIMESTAMP NULL,
  PRIMARY KEY (`User_UserID`, `Games_GameID`),
  INDEX `fk_UserGame_User_idx` (`User_UserID` ASC) VISIBLE,
  INDEX `fk_UserGame_Games1_idx` (`Games_GameID` ASC) VISIBLE,
  CONSTRAINT `fk_UserGame_User`
    FOREIGN KEY (`User_UserID`)
    REFERENCES `mydb`.`User` (`UserID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_UserGame_Games1`
    FOREIGN KEY (`Games_GameID`)
    REFERENCES `mydb`.`Games` (`GameID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;
```



```sql
CREATE TABLE IF NOT EXISTS `mydb`.`Tags` (
  `TagName` VARCHAR(24) NOT NULL,
  `Description` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`TagName`),
  UNIQUE INDEX `Tags_UNIQUE` (`TagName` ASC) VISIBLE)
ENGINE = InnoDB;
```



```sql
CREATE TABLE IF NOT EXISTS `mydb`.`GameTags` (
  `Games_GameID` INT NOT NULL,
  `Tags_TagName` VARCHAR(24) NOT NULL,
  PRIMARY KEY (`Games_GameID`, `Tags_TagName`),
  INDEX `fk_GameTags_Games1_idx` (`Games_GameID` ASC) VISIBLE,
  INDEX `fk_GameTags_Tags1_idx` (`Tags_TagName` ASC) VISIBLE,
  CONSTRAINT `fk_GameTags_Games1`
    FOREIGN KEY (`Games_GameID`)
    REFERENCES `mydb`.`Games` (`GameID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_GameTags_Tags1`
    FOREIGN KEY (`Tags_TagName`)
    REFERENCES `mydb`.`Tags` (`TagName`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;
```



```sql
CREATE TABLE IF NOT EXISTS `mydb`.`Achievements` (
  `AchievementID` INT NOT NULL,
  `Name` VARCHAR(45) NOT NULL,
  `Description` VARCHAR(45) NOT NULL,
  `Completed` TINYINT NOT NULL,
  `Date Completed` DATE NULL,
  `UserGame_User_UserID` INT NOT NULL,
  `UserGame_Games_GameID` INT NOT NULL,
  PRIMARY KEY (`AchievementID`, `UserGame_User_UserID`, `UserGame_Games_GameID`),
  INDEX `fk_Achievements_UserGame1_idx` (`UserGame_User_UserID` ASC, `UserGame_Games_GameID` ASC) VISIBLE,
  CONSTRAINT `fk_Achievements_UserGame1`
    FOREIGN KEY (`UserGame_User_UserID` , `UserGame_Games_GameID`)
    REFERENCES `mydb`.`UserGame` (`User_UserID` , `Games_GameID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;
```