Summary

# Introduction

This Gaming Loyalty System is designed and developed for showing the top ten patrons based on their turnover and the games they like playing. Two functions are implemented, including importing data from a CSV file to database and displaying the top ten patrons.

The techniques used in this system including HTML, Javascript, JQuery, CSS, Bootstrap, PHP, Postgres. This website is mobile friendly with responsive design.

# Development Process

The system is developed with three components, including View, Controller and Models. The general logic behind these components is as follow:

Controllers

Database

CSV File

Convert CSV to Models

Read Data into Models

Data Import

Data Extraction & Visualisation

More specifically, I developed the system in the following process:

* A DatabaseAdapter class is created to store parameters and basic functions relating to database operation.
* Two models are defined for data input and output.
  + LoyaltyModel: This is a model representing each row in the CSV file. Each property also corresponding to each field in database
  + LoyaltyViewModel: The is a model representing each row shown in the View.
* Two Controllers are then added to handle data input and output.
  + FileUploadController: This controller will be used when a file is uploaded from the view. In this controller, each line in the CSV file will be read and converted to a LoyaltyModel object. Then, each object is inserted to the Loyalty table in database.
  + TopTenPatronController: This controller is used to extract data from database and convert each record into a LoyaltyViewModel object. Eventually, an array of LoyaltyViewModel objects is established for View.
* A View is designed as the user interface of the system. This view uses Bootstrap so that it is responsive to the size of the screen.

# Tests

Due to the time limitation, all the tests are done manually. The main approach is that I created a testing webpage with a list of test cases:

* Test insert function in DatabaseAdapter
  + Test steps:
    - Initialise a DatabaseAdapter object
    - Write an Insert SQL statement
    - Invoke insert method
    - Check Testing table in database
  + Expected outcome:
    - A record with correct information is appended in Testing table
* Test select function in DatabaseAdatper
  + Test steps:
    - Initialise a DatabaseAdapter object
    - Write a Select SQL statement
    - Invoke select method
    - Print the result
  + Expected outcome:
    - The records in the database are shown in the webpage
* Test insertItem function in LoyaltyModel
  + Test steps:
    - Initialise a DatabaseAdapter object
    - Initialise a LoyaltyModel object
    - Populate data to the LoyaltyModel object
    - Invoke insertItem function
  + Expected outcome:
    - Data in LoyaltyModel object is inserted to the Testing database
* Test getTopTenPatrons function in TopTenPatronController
  + Test steps:
    - Initialise a DatabaseAdapter object
    - Invoke getTopTenPatrons function with correct parameters
    - Print result
  + Expected outcome:
    - Top ten patrons with highest turnover are printed
* Test getGamesForOnePatron function in TopTenPatronController
  + Test steps:
    - Initialise a DatabaseAdapter object
    - Invoke getTopTenPatrons function with correct parameters
    - Invoke getGamesForOnePatron with the first patron\_id, DatabaseAdapter object and table name
    - Print result
  + Expected outcome:
    - The games liked by the patron is shown correctly.