### Florida International University

*School of Computing and Information Sciences*

CIS 4911 - Senior Capstone Project

Software Engineering Focus

Final Deliverable

Strategic Marketing Simulator 1.0

Team # 15

**Team Members**

Jeffrey Carman

Javier Andrial

**Product Owner**: Joseph Cilli

**Instructor**: Masoud Sadjari

# Copyright

Bootstrap v3.1.1 (http://getbootstrap.com)

Copyright 2011-2014 Twitter, Inc.

Licensed under MIT (<https://github.com/twbs/bootstrap/blob/master/LICENSE>)

Bootstrap v3.3.4 (http://getbootstrap.com)

Copyright 2011-2015 Twitter, Inc.

Licensed under MIT (https://github.com/twbs/bootstrap/blob/master/LICENSE)

Start Bootstrap - Logo Nav HTML Template (http://startbootstrap.com)

Code licensed under the Apache License v2.0.

For details, see http://www.apache.org/licenses/LICENSE-2.0.

***Abstract***

*Students of the FIU business school take marketing courses to give them a solid understanding of business activities as they relate to target markets, development of pricing models, the promotion of goods and services, the management of business relationships and consumer behavior. Much of the curriculum is theory as it pertains to marketing concepts, but there is something lacking. The subject application, Strategic Marketing Simulator, allows students and instructors to participate in a mockup of a hotel market, where the way a marketing budget is allocated and the state of the market has a direct impact on how the market trends during an interval of time, a period. Additionally, students gain an understanding on how political and social policy, force majeure, supply and demand etc., can affect one company's market share over another.*

**Table of Contents**

[Florida International University](#h.5fpwvxwxeykq)

[Copyright](#h.y4g02dpi8wms)

[Introduction](#h.30j0zll)

[Current System](#h.1fob9te)

[Purpose of New System](#h.4vlexiufozs7)

[Scope of System](#h.543xfx4tumkg)

[User Stories](#h.2et92p0)

[Implemented User Stories](#h.tyjcwt)

[Pending User Stories](#h.3dy6vkm)

[FEASIBILITY STUDY](#h.1mp9ee8isg2d)

[Problem definition](#h.uiho6r6cf82m)

[Overview of study](#h.dvgqgxqvppke)

[Description of Current System](#h.gdtol232sw5b)

[Background](#h.kprsc27qqzsm)

[Purpose of New System](#h.nnbstip2ww2u)

[High-level Definition of User Requirements](#h.ob67q74uacou)

[Alternative Solutions](#h.x59hh1m8e2fd)

[Description of Alternatives](#h.n2pmu0atxpmx)

[Selection Criteria](#h.cm74rjqft5po)

[Analysis of Alternatives (refer to Appendix C – Feasibility Matrix) – you should provide a score so that the alternatives can be compared.](#h.2iu5vg6olch6)

[Feasibility Matrix](#h.ib7wd1r17am1)

[Project Plan](#h.1t3h5sf)

[Functional Requirements](#h.712umbiqto2p)

[Analysis of System Requirements](#h.4inam92qj5q3)

[Scenarios](#h.2e1o2y3qyzcv)

[Use case model](#h.5u5wg4cgq1uc)

[Hardware and Software Resources](#h.r7b2wcrdniu7)

[Sprints Plan](#h.2s8eyo1)

[Sprint 1](#h.17dp8vu)

[Sprint 2](#h.hab2vvl9u9t8)

[Sprint 3](#h.ldzmeygku3s1)

[Sprint 4](#h.rbzq8fcwr2x3)

[Sprint 5](#h.tzf3q2bxavag)

[Sprint 6](#h.hyjbiactzn48)

[Sprint 7](#h.2mvyh01lt65g)

[System Design](#h.3rdcrjn)

[Architectural Patterns](#h.qa5pnv1usaxy)

[System and Subsystem Decomposition](#h.vbp7bm6rbwbz)

[Deployment Diagram](#h.35nkun2)

[Design Patterns](#h.1ksv4uv)

[System Validation](#h.44sinio)

[Glossary](#h.62p81erwfove)

[Appendix](#h.z337ya)

[Appendix A - UML Diagrams](#h.3j2qqm3)

[Static UML Diagrams](#h.1y810tw)

[Dynamic UML Diagrams](#h.kwwksitt8cdf)

[Appendix B - User Interface Design](#h.2xcytpi)

[Appendix C - Sprint Review Reports](#h.1ci93xb)

[Appendix D - Sprint Retrospective Reports](#h.3whwml4)

[References](#h.2bn6wsx)

# Introduction

University instructors need a means for having their students run simulations of a hotel room sales market to show the impact that different marketing budget allocations will have on the market. These allocations need to take the form of dollars spent on marketing personnel, advertising, and promotions; along with type of hotel and the location of the hotel. The way a student utilizes their budget will ultimately affect the market by either giving him or her competitors a market advantage. By the same token, these students need a means for interpreting the effects of budget allocations that other students have in the same market. Finally, instructors need a way to track these results over varying intervals of time.

## Current System

The current system of simulation is done through conversation and debate between students and instructor in a classroom setting. The instructor poses scenarios about trends in the market and asks students for their input about how a marketing budget allocation might affect the market. The system is limited by the subjective point of view of the instructor in that there is no mathematical formula being deployed to make a decision about how an allocation might affect market status and/or future trends. This system takes a lot of time to implement and is difficult to continue the same conversation from one class period to another.

## Purpose of New System

The Strategic Marketing Simulator allows students to make marketing decisions that will affect the hotel sales market in a simulation application. “Decisions” are defined by the system as a number of marketing budget items which are allocated in the form of dollars spent on marketing personnel, advertising, and promotions. Each type of decision carries a different ranking, which is quantified in a meaningful way by the system to produce an impact on the market. The market status, as displayed by the simulation, will change as a result of actors on the simulation, market trends, and current events. The actors on the simulation are other students and/or programs which simulate students by randomly selecting budget allocations (decisions) which affect the market in a similar way. At the end of each period, a user can view the current state of the market, their remaining budget, and the rankings of all the other actors. At any time, an instructor can view the results of all periods and the final results of a game. At the end of a game, all users can see the outcome of the game which can foster conversations about how a market can be affected by marketing investments.the same way. At the end of each period, a user can view the current state of the market, their remaining budget, and the rankings of all the other actors. At any time, an instructor can view the results of all periods and the final results of a game.

## Scope of System

The system consists of two main functionalities: Student functionalities and instructor functionalities. Yet there are some functionalities are user type agnostic. the following sections defines the scope of the system by these different category in order of importance and ultimately the order of delivery.

Student users:

Students have the ability to join a game and create/join a group when they first login to the system. They will be directed to the create/join group page of the system until they have made a selection. Each group must select one student to make strategic decisions on behalf of his or her group at some point during a period and then commit those decisions for the period. When a group commits decisions for a period, it must gather reasons for making decisions from its members and account for them in the commit textbox.

Instructor users:

An instructor must have the ability to create games, assign students to those games, and change group membership and he or she sees fit. An instructor must also have the ability to create instructor account with all the same privileges as he or she’s account. Also including in an instructor’s abilities is that of being able to create news articles that affects the outcome of market share for a period. Lastly, an instructor must be able to run meaningful reports on a period/game results.

User type agnostic:

All users must be able to create an account consisting of username. password, secret question, and secret answer. All users must be able to log into the system using their username and password and be able to reset their password should the misplace it.

# User Stories

In this section a list of user stories are presented. Use stories are a means for defining the required functionality and behavior of the system and how a user can manipulate the system to perform a desired task. The user stories for the Strategic Marketing Simulator come in two forms: student user stories and admin stories. Both describe how each will interact with the system to play a game or administrate over a game (respectively)

## Implemented User Stories

**User Story # 711 - deactivate Users**

* As a logged in admin I need to be able to delete users so that I can remove users from the system
* A logged in admin clicks, delete user.
* A confirmation dialog box appears. the admin clicks “yes”
* The user account is deleted from the system

**User Story # 776 - Student Create Group**

* The student decides to create a new group and clicks on the “create a new group” from the join group page.
* The student clicks on the games drop down list and selects a game.
* The student enters a group name in the group name text box
* The student clicks on the ‘create new group’ button.
* The user is redirected to his or her homepage.

**User Story # 704 - Admin Login**

* As an admin I need to be able to log into the system so that I can manage games and users
* An admin navigates to the login page.
* An admin enters his username and password into the corresponding text boxes

**User Story # 769 - Student Join Group**

* A new user is directed to Join group page. Student has the option to join a already created group or create a new group.
* If a user joins a group they first select the game from the game drop down list.
* They next select a group from the dynamically populated group drop down list.
* The user clicks join group.
* The user is directed to their homepage.

**User Story # 667- Forgot my password**

* The user clicks on the “forgot my password” link from the login page.
* The system generates a random temporary password.
* The system sends the use an email with his or her temporary password enclosed.
* The user logs onto the system using the temporary password.
* The system requires that the user reset his password by entering it in two text boxes.
* The user’s password is reset.

**User Story # 669- User login**

* As a user I need to be able to log into the system so that I can view my homepage
* A user enters his or her password and clicks “login”
* The system navigates the user’s homepage

**User Story # 666- Account Creation**

* As a user, I need to be able to create an account with the system so that I can participate in simulations
* A user enters a username and password into the corresponding text boxes on the login page.
* The user enters his or her email address
* The user click the “Create account” button.

**User Story # 718 - Manage Page**

* As a logged in admin I need to be able to navigate to the manage page so that manage users, manage games, and manage administration
* A logged in admin clicks on the manage link
* The admin can manage users, games and administration.

**User Story # 712 - User comment and commit**

* A logged in user navigates to home page after selecting strategic decisions for a period
* A user clicks the “commit” from the home Page.
* The system redirects the student to the comments page.
* The user types in reason for his or her decisions and clicks the “commit” button

**User Story # 709 - Admin News**

* As a logged in administrator I need be able to add a news story to the system so that the users can make informed decisions about current events
* A logged in admin clicks on the add news button,
* The admin enters the text for the article and checks some or all of the following check boxes:
  + Impact positive
  + Impact negative
  + location
  + hotel type – eco
  + hotel type – lux

**User Story # 686 - Admin Game Page Functionality**

* As a logged in admin I need to be able to click on a game in progress from the games page so that I can view all the groups that are playinga particular game in descending order of game success
* A logged in admin clicks on a game from a list of games on the games page.
* The system populates a listing of all users associated with that game in descending order of success

**User Story # 685 - Admin games page**

* As a logged in admin I need to be able to click on the "games" link from the homepage so that I can view all games that are in progress
* A logged in admin clicks on the “games” link from the toolbar of the home page.
* The populates a list of games that are in progress.

**User Story # 684- Bot creation**

* As a logged in admin I need to be able to create a bot account
* A logged in admin navigates to the create user page
* The admin clicks the create new user button.
* The admin enters the username and group.
* The admin ticks the “is a bot” check box
* The admin clicks the create user button

**User Story # 683- Admin Account Creation**

* As a logged in admin I need to be able to create admin accounts so that other admins can create simulations.
* A signed admin navigates the create admin page and clicks new.
* The admin enters a username and email address for the new admin.
* The system sends an account verification email to the new admin.

**User Story # 681- Admin “users” page**

* As a logged in admin I need to be able to click on the "users" link from my homepage so that I can see a list of all users
* A logged in in admin clicks on users from the homepage.
* The system navigates to the users page

**User Story # 685 - Admin games page**

* As a logged in admin I need to be able to click on the "games" link from the homepage so that I can view all games that are in progress
* A logged in admin clicks on the “games” link from the toolbar of the home page.
* The populates a list of games that are in progress.

**User Story # 680- Admin home**

* As a logged in admin I need to be able to click on the "home" button from any page of the system so that the system can navigate to the home page
* A logged in admin clicks on the home link
* The system navigates to the homepage

**User Story # 679- User View news**

* As a logged in user I need to be able to click on the "news" link to view current events so that I can make an informed decision about how to allocate my marketing budget
* user clicks on the news button.
* The system navigates to the new pages
* News articles are listed

**User Story # 678- User strategic decisions**

* As a logged in user, I need to be able to click on "Strategic Decisions" link from the toolbar so that I can assign values to the budgetary items: Marketing personnel, advertising media, room pricing, and third party vendors
* A user clicks on the strategic decisions link.
* System navigates to strategic decisions page.

**User Story # 810- Admin strategic decisions management**

* As a logged in admin , I need to be able to click on "Strategic Decisions" link from the toolbar so that I can assign values and impacts to budgetary items: Marketing personnel, advertising media, room pricing, and third party vendors

**User Story # 668- Password Recovery**

* I as a user need to be able to be able to request a "password recovery email" from the system so that I can recover my password
* A user enters his or her username or email address.
* The system sends the user an email to the user with random password.

**User Story #800 – Admin News Management**

* Admin is able to write or paste a News Article for a period, for a game.
* the Admin is able to apply News Parameters to an article. these Parameters apply an affect to specific Hotels for that Period, in a Game.
* Administrators can Choose which Game to Views the News for.
* Administrators can Choose Which Period in a Game to view/Modify the News for.
* Administrator can Save their changes to the Database or erase everything and start over.

**User Story # 708 - User Scorecard**

* The scorecard is part of the homepage view that shows a student their status versus the status of the rest of the market:
  + Hotel type
  + Location
  + Advertising
  + Room Allocated
  + Remaining Budget
  + Marketshare
  + Market Share pie chart(optional for release 1)
  + Group Name
  + Market Segment

**User Story # 676- User home**

* I as a logged in user need to be able to click on the "home" link from any page in the system so that I can navigate to the home page.
* A user clicks on the “home” link.
* The system navigates to the home page.

**User Story # 672- Market Status Graph**

* As a logged in user I need to be able to view the graphs on the dashboard so that I can view a graphical representation of the current market status for market share
* A logged in user views the graphs on the homepage.

**User Story # 671- User leaderboard**

* The leaderboard shows the total revenue for each from in descending order for the previous period.
* It also have a bar graph as a visual representation of the same

**User Story #842 - Professor Feedback**

* Professor is able to give their feedback at the end of period for a specific Group based on their strategic decisions and their end of period comment.

**User Story #841 - Professor Portfolio**

* Admin is able to download a detailed report of how Each Hotel did during the last period, as well as all the groups as a whole performed

U**ser Story #840 - Comment Repository**

* A page where an Admin user can view all Comments for a Game’s Period.
* Admin can also filter Comments by Hotel Type

**User Story #847 - End of Period Algorithm**

* Algorithm that quantifies and qualifies market share by group per period and revenue earned based on strategic decisions and news impact

**User Story # 710 - User Toolbar**

* As a logged in user I need to be able to access the toolbar from any page
* A logged in user can view the toolbar from any page of the system.
* The toolbar contains links for the following pages:
  + A Home. this button goes to the Home page
  + Metrics. this button goes to the Metrics Page
  + Strategic Decisions. this button goes to the Strategic Decisions page
  + News. this button goes to the news Page.
  + Manage. this button will only be visible for Instructor/admin users
    - this button goes to the Manage page.

**User Story** [**#**](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/strategic_marketing_simulator_/cards/842)**853 - Student Reporting Page**

* This page will display their end of period comments as well as any feedback the professor has provided.

**User Story** [**#**](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/strategic_marketing_simulator_/cards/842)**856 - Bot Play**

* End of period algorithm looks for bots in play and selects a random number of decisions and OTA allocations to simulator a group.

## Pending User Stories

# FEASIBILITY STUDY

## Problem definition

University instructors need a means for having their students run simulations of a hotel room sales market to show the impact that different marketing budget allocations will have on the market. These allocations need to take the form of dollars spent on marketing personnel, advertising, and promotions; along with type of hotel and the location of the hotel. The way a student utilizes their budget will ultimately affect the market by either giving him or her competitors a market advantage. By the same token, these students need a means for interpreting the effects of budget allocations that other students have in the same market. Finally, instructors need a way to track these results over varying intervals of time.

## Overview of study

The feasibility study and project plan analyses the proposed system of simulating a hotel market as a means for instructors to engage his or her students in a marketing simulation application. The Strategic Marketing Simulator is described in detail and is compared to other simulation systems in the market as a way of determining the feasibility of producing it over deploying an alternative systems or continuing use of the current system. Finally, a project plan is presented to show the organization of the project, what hardware and software will be used to develop the project and tasks and milestones required to complete it.

## Description of Current System

## Background

The Strategic Market Simulator (SMS) is a brain child of Joseph Cilli, MS, Director of Distance Learning at Florida International University. Mr. Cilli intended the application as a way for instructors teaching marketing class at the university to be able to engage his or her students in class by having them participate in a marketing simulator that to apply what they have learned in class in a fun and engaging way.

The current system of simulation is done through conversation and debate between students and instructor in a classroom setting. The instructor poses scenarios about trends in the market and asks students for their input about how a marketing budget allocation might affect the market. The system is limited by the subjective point of view of the instructor in that there is no mathematical formula being deployed to make a decision about how an allocation might affect market status and/or future trends. This system takes a lot of time to implement and is difficult to continue the same conversation from one class period to another.

## Purpose of New System

The *Strategic Marketing Simulator* allows students to make marketing decisions that will affect the hotel sales market in a simulation application. “Decisions” are defined by the system as a number of marketing budget items which are allocated in the form of dollars spent on marketing personnel, advertising, and promotions. Each type of decision carries a different ranking, which is quantified in a meaningful way by the system to produce an impact on the market. The market status, as displayed by the simulation, will change as a result of actors on the simulation, market trends, and current events. The actors on the simulation are other students and/or programs which simulate students by randomly selecting budget allocations (decisions) which affect the market in a similar way. At the end of each period, a user can view the current state of the market, their remaining budget, and the rankings of all the other actors. At any time, an instructor can view the results of all periods and the final results of a game. At the end of a game, all users can see the outcome of the game which can foster conversations about how a market can be affected by marketing investments.

The Strategic Marketing simulator also gives instructors a means for running reports for end of period and end of game in order to provide feedback to his or her students regarding the results of the game.

## High-level Definition of User Requirements

A Student must be able to register a new account with the system by using a unique username and password, supplying their email address, and choosing the class that they belong to. The user’s password must meet the system’s password complexity policy. A user must be able to create/join a group and be able to select strategic decisions for a period which result in an affect on the market.

Instructors must be able to log into the system, create games, users, and groups. An instructor must be able to retrieve reports from the system to aid him or her in giving appropriate feedback to students about the results of a game or a period of the game.

The system must be able to store data such as account information, news articles and their effects on the game, strategic decisions entered in my students, end of period comments and their feedbacks.

The System must also be able to compute end of period performance based on strategic decisions of all students and news parameters for a Game’s period.

## Alternative Solutions

While there are no application While there are no applications which directly address the impact of marketing budget allocations in a hotel sales market, there are some applications which address marketing in general. A couple examples of this are “Strategic Marketing” by Marketplace Live © and Markstrat by StratX.

### Description of Alternatives

Market Place Live © offers a simulation Strategic Marketing which introduces students to basic marketing concepts. Students are engaged by the simulation by receiving information about customer needs, branding, pricing, advertising and customer feedback. Simulations are done in 6 rounds where students are competing with one another to achieve a good grade which is based on a scorecard which measures profitability, market share and customer satisfaction.

Markstrat by StratX is a marketing simulation which is offered to MBA students which simulates aspects of marketing. Within the simulation, a student’s decisions affects sales, distributions and advertising. Each team commits actions with have a direct affect on the market creating a competitive environment where new product launches, sales and distribution strategies are managed by users.

### Selection Criteria

The criteria used to determine the relevant systems, which are comparable to the subject application. There are six criteria used to determine the feasibility of a system.

The first criterion is a system’s ability to simulate the status of a market given external actors making decisions which affect the market. The second criterion of a candidate system is the flexibility to group users in many teams (at least 10) for a common goal. The third criterion used is a candidate system’s ability to grade users/groups based on their decisions and how those decisions affect the market. The fourth criterion for a candidate system is its ability to generate computer users/groups which participate in the simulation in a similar manner as regular users. The fifth criterion for a system is that it must simulate the specific market for hotel sales. The final criteria is a candidate system’s ability to display the status of a market, given user/group’s decisions, over a long period of time (years).

## Analysis of Alternatives (refer to Appendix C – Feasibility Matrix) – you should provide a score so that the alternatives can be compared.

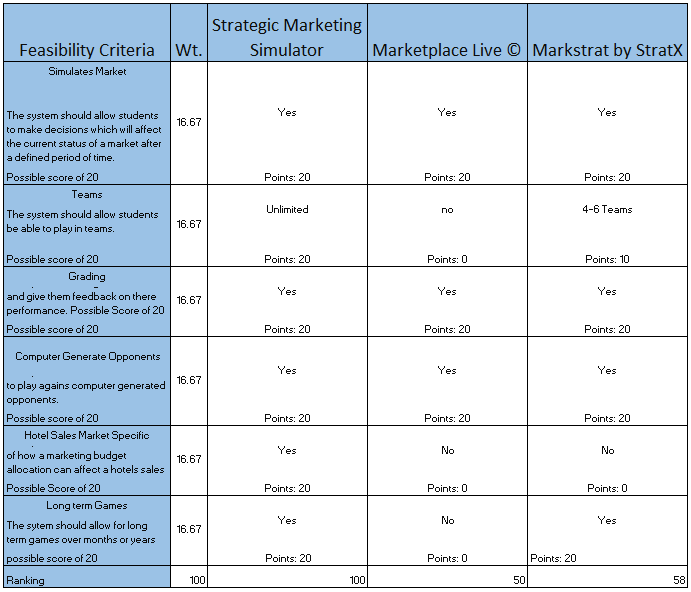
Each criterion as defined in section 2.4.2 are scored using a point system of 20 points for each criterion. There are six criteria for a total possible score of 120 points. A candidate system will get a score of 0 in the event that it doesn’t meet one of the above criterion and 10 points if it only meets the criterion partially.

For the first criterion regarding a system’s ability to simulate a market which can be affected by user decisions. The two candidates, Market Place Live © and StratXBoth scored a perfect score of 20 points. The second criterion, a system’s ability to allow users to play in many groups, rendered a score of 0 points for Market Place Live © and a partial score of 10 points for StratX since the number of possible teams doesn’t meet the threshold.

The third and fourth criterion for a system’s ability to grade user/groups based on their decisions and a system’s ability to use computer generated opponent, received a perfect score of 20 points by both candidates. However, both candidates scored 0 points for the fifth criterion for a system’s ability to simulate the specific market of hotel sales. The final criterion, a system’s ability to provide feedback on a market over an extended period of time scored a perfect score of 20 by StratX*,* but fell short of the mark by Market Place Live © which received a score of 0.

The final tally of the candidates reveals Market Place Live © with a score of 60/120 and 70/120 for StratX.

## Feasibility Matrix



# 

# Project Plan

For this project, the participants will take an an agile approach to development, wherein all of the requirements of the system will be compiled with the help of the project owner in the form of user stories. The user stories are sorted in terms of difficulty, taking into consideration what is needed to properly integrate all of the system’s parts (subsystems, objects, and interfaces). The stories are then spread out over 8 sprints between the participants to account for the feasibility of timely delivery, validation and testing, and even workload among the developers. At the end of each sprint, each story will be fully implemented and tested.

## Functional Requirements

1. The System shall allow Student users to create a Student level user account

a. This requirement should be intuitive and executes quickly.

b. Email for account needs to be a valid email

2. The System shall allow Administrative and Student users to log in

a. This function should perform efficiently

3. The System shall function and appear identical across the major web browsers

a. appearance of web pages should be appealing and well colored

4. The System shall logout the user upon user request

5. The System shall reject a user if a login email or password do not match in the database

a. If a login is rejected. it should be visible apparent that their attempt was rejected

6. The System shall allow Students to commit strategic decisions to the database once per period

7. The System shall allow Users to reset password via the Forgot password functionality

a. email response time should be quick

b. passwords should be random enough to never have duplicates

8. The System shall allow Students to join an existing game

a. this function should be easy, intuitive, and informative

9. The System shall show Students a sudo news article in the News Tab

a. this function should display the text in an appropriate font and font size

10. The System shall allow Students to view stats for the current period on the metrics page

11. The System shall allow Students to view their position amongst the other Students on the homepage

a. this function should catch the user’s attention

12. The System shall allow Users to navigate through buttons on the toolbar

a. this function should be intuitive and appealing

13. The System shall allow Administrative users to create an Administrative account

a. this function should be easy and intuitive

14. The System shall allow Administrative users to create a Game

a. this function should be easy and intuitive

15. The System shall allow Administrative users to remove entries (Users, Games, Groups) from the database

16. User must be able to change their password

17. User must be able to Change their secret answer and question

18. User must be logged into the system to be to use Account Management

19. User must enter in their current password to be able to change their password, secret question, and secret answer.

20. The system shall provide a view for group information for a given game period to include:

a.group name

b. Location

c. market segment

d. Budget

e. remaining budget

f. market share

g. revenue.

21. The system shall provide a means for a group to commit their decisions for a given period.

1. The student shall commit a period’s decisions.
2. The student shall click the commit button from the homepage
3. The student shall enter reasons for their decisions for a given period
4. The system log a user’s comments in the database.

22.. The system shall provide a view of the leaderboard for a student .

1. The system shall display the groups’ progress from the previous period by revenue in descending order
2. The system shall organize leaderboard in a table.
3. The system shall display a graph as a visual representation of the leaderboard.

23. The system shall provide a visual representation of a the market share by group for the previous period.

1. The system shall display a pie chart
2. The system shall display a message during the first period indicating that market share by group is not yet available

24. The System shall provide the admin ability to commit their Feedback to the database

25. The System shall provide the admin the ability to generate and download a detailed report for a given Game and period.

26. The System shall provide the Admin the ability to view all end of the period comments.

27. The system shall calculate the revenue and market share of each group at the end of a period.

1. The market share for each group must be a result of their respective strategic decisions.
2. The revenue is a result of strategic decisions, OTA allocations, and average rate

## Analysis of System Requirements

### Scenarios

**Scenario 1 – Student creates an account in the system**

A student in a fiu marketing course, George, is told by his professor to create an account in the system for tonight’s homework. George goes home from class and accesses the Strategic Marketing Simulator webpage. George does not have an account and selects ‘Create an Account’ on the login page. George enters in his FIU.edu email and a password for his account and enters his person information (full name and panther ID). George receives a confirmation email and was successfully able to create his Student user account.

**Scenario 2 – Admin views users that have joined his game**

Professor Joseph Cilli, wants to see how many students in his course have already created an account and joined his game. Professor Cilli, navigates to the Strategic Marketing Simulator webpage, enters his Login email and his account password. Professor is successfully logged in. He navigates to the Manage page by pressing the manage button on his homepage. Then he selects the game he created and selects ‘view users’. Professor Cilli, is then shown all the users currently registered to his game and their account details.

**Scenario 3 - Student user commits their changes**

Student George has already created an account, joined an active game, and is currently logged into the system. George has reviewed the market information and news for the current period and is ready to make his strategic decisions. George navigates to the Strategic Decision page via the “Strategic Decision” button on the toolbar. George enters his inputs and is ready to commit his decisions for the period. George presses the Commit button at the bottom of the page and is prompted to type in a short paragraph behind his reasoning. George is nervous and writes several pages about his decisions in deep detail. George, presses ‘Commit Changes’ at the bottom of the prompt. Prompt closes and he is locked from altering any other strategic decisions until next period.

**Scenario 4 - Admin removes a user from the database**

A student drops from Professor Cilli’s course. Professor Cilli does not want these inactive accounts to affect the markets for the active students. Professor Cilli navigates to the Strategic Marketing Simulator web page and logs into the system with his email and password for his account. Professor navigates to the Manage page via the toolbar and selects his game. He is shown all the users registered to that game. Professor Cilli, looks for the user accounts of the students that dropped the course. Professor Cilli, selects their account and presses the Delete Entry button. The User account will then be made inactive and will no longer be displayed or allowed to login.

**Scenario 8 - Admin creates an Administrative Account**

Professor Cilli, wants to add Stephanie, a graduate student that will be helping Professor Cilli grade the course as an Administrator. Professor Cilli, logs into the Marketsim page site with his Administrator credentials. from the Home page he presses the Manage button to navigate to the Manage Page. From there he selects 'Add Administrator User' button and then fills out all of Stephanie's information. Professor Cilli, hits the 'Create Admin Account' button and an Administrative account for Stephanie has been created.

**Scenario 9 –Admin creates an Bot and a Hotel**

This Semester Professor Cilli, will be having a smaller the usual course. To keep the class experience consistent, he will be adding Bot Student Accounts to act as real student within the MarketSim game. To do this Professor Cilli, navigates to the Manage page after Logging into the System on with his Administrator account. He then selects 'Add Bot User' and proceeds to fill out all the required information for the Bot User and their Hotel. Once completed he hits 'Create Bot Account' button and a Bot Account with a Hotel for the Bot will be created.

**Scenario 10 –Admin Views all Games**

Professor Cilli, wants to view all games in the Database. After logging into the Marketsim System with his Admin credentials navigates to the Manage page. he then presses 'View All Games' button and is shown all the Games currently listed in the Database.

**Scenario 11 –Admin Views All Users in a Game**

Professor Cilli, wants to view all the Users in his Monday class. After logging into the Marketsim System with his Admin credentials navigates to the Manage page. he then presses 'View All Games' button and is shown all the Games currently listed in the Database. Professor Cilli, finds the Game in question and selects the 'Users' button next to the entry. Professor Cilli, is then displayed all the Users registered to this Game.

**Scenario 12–Admin Creates a Game**

Professor Cilli, wants to add a Game for his Wednesday class. After logging into the Marketsim System with his Admin credentials navigates to the Manage page. he then presses 'View All Games' button and is shown all the Games currently listed in the Database. Professor Cilli, then presses the 'Create A Game' Button at the button of the page. He then proceeds to Provide all the necessary information and presses 'Create Game' button. a Game will be created with the information he provided.

**Scenario 13 –Admin Views All Users**

Professor Cilli, wants to view all the Users currently in the Database. After logging into the Marketsim System with his Admin credentials navigates to the Manage page. He then presses 'View All Users' button and is shown all the Users currently listed in the Database.

**Scenario 14 –Admin Deactivate/Activate Users**

Professor Cilli, has had Students drop from his Monday Class as well as his TA Stephanie leave the Graduates program. After logging into the Marketsim System with his Admin credentials navigates to the Manage page. He then presses 'View All Games' button and is shown all the Games currently listed in the Database. Professor Cilli, finds the Game in question and selects the 'Users' button next to the entry. Professor Cilli, is then displayed all the Users registered to this Game. Professor Cilli, looks for the Students that drop the course and check marks their entries. He then follows back selecting the De-Activate button. All the users Professor Cilli, had selected are not deactivated and no-longer able to log into the System.

Professor Cilli, now needs to De-Activate Stephanie who will no longer need access to the System. Professor Cilli, presses 'View All Users' button and is shown all the Users currently listed in the Database. He looks for Stephanie's Account, check marks it and hits the 'De-activate User' button. Stephanie will no-longer be able to log into the MarketSim System.

**Scenario 14 –Student Joins a group**

Jeremy created an account with the Strategic Marketing Simulator. Since this is the first time that he has logged into the system, the system redirects him to the join group page. Jeremy is given a list of current games in a dropdown list. Jeremy clicks on the game “Fall morning class”, Immediately upon clicking on the game, a new dropdown list with the heading “select a group” dynamically appears to Jeremy with a list of the already created groups. Jeremy selects the group “ABC Marketing” from the list and clicks the “join a group” button. Jeremy is redirected to his home page.

**Scenario 15 Create a new group**

Michael created an account with the Strategic Marketing Simulator. Since this is the first time that he has logged into the system, the system redirects him to the join group page. From the join group page, Michael clicks the “create a new group” link and is redirected to the new group page. From the new group page, Michael selects the game”fall morning class”. Jeremy types the name of his new group, “ABC Marketing” into the “new group” textbox and clicks the “create new group” button. Jeremy is then redirected to his homepage.

**Scenario 16 Strategic Decisions**

Sarah is the group leader for her group “ABC Marketing” and as such, she is responsible for selecting all the strategic decisions for her team after a discussion and analysis of their progress. Sarah ticks the City Bus Ads”, “Promotional Gifts” and “TV spots” check boxes, enters an average daily rate for rooms as “$68” and clicks the “select decisions “ button. Sarah is redirected to her home page.

**Scenario 17 Student Views the News**

Jamie, is a Student in Professor Cilli's class. Jamie, is logged into the Strategic Market Simulator System and is registered to a Game. Jamie, navigates to the News page, via the 'News' button via the Home page's toolbar. Jamie, is shown the News article for the current Period of her Game.

**Scenario 18 Admin changes a period**

Professor Cilli, wishes to set the News Articles for all the Period for a particular Game. Professor Cilli, chooses a Game and is shown that Games current news article. He inputs the desired article for that period and appropriate parameters. He then selects the 'Choose a Period' button and is shown a list of all Periods and News articles associated for those periods. Professor Cilli, selects the next Period, commits the desired inputs and repeats until all the Periods for that Game have an News Article set and appropriate parameters.

**Scenario 19 Admin removes News Parameters**

Professor Cilli, changes his mind for a news periods article and its parameters. Professor Cilli, presses the ‘Reset’ button, and then selects the ‘Clear News Parameters saved on DB’. This will remove all the saved news parameters registered to the current news article Professor Cilli, has open. Professor Cilli, also selects ‘Clear News Parameters for this session’. This clears any unsaved parameters. Now the Professor can write up a new News article and apply new appropriate parameters.

**Scenario 20 Admin commits a News Article**

Professor Cilli, is writing the News article for his course's current period. Professor Cilli, has finished the article and Presses the 'Save and Preview' button to preview the article. He is happy with how the article appears and hits the 'Commit to Database' button. A message appears that the commit was successful and Professor Cilli logs out of the system.

**Scenario 21 Admin Chooses/Changes Game News Article**

Professor Cilli, wishes to set the News Articles for all the Period and for all Games. Professor Cilli, has finished inputting all the articles for one Game and presses the 'Change Game' button. Professor Cilli, is displayed a list of all the Games and selects the desired Game and presses the 'Choose a Game' button. Professor Cilli, is now displayed the article and news parameters for that Game's current period.

**Scenario 22 Admin login**

Professor Johnson began teaching a marketing class last semester and would like to use the Strategic Marketing Simulator. He was given access by another professor that used it during the summer. Professor Johnson uses his internet browser to navigate to [http://Marteksim.com](http://marteksim.com). the website redirects him to the login page, where he enters his username and password and clicks the login button. Professor Johnson is logged into the system.

**Scenario 23 Admin Strategic Decisions Management**

Professor Johnson had a lot of success with the Strategic Marketing Simulator last semester, but found that some of the strategic decisions were too inexpensive and some didn’t have enough of an impact on the game. After logging in, he clicks on the strategic decisions link and is able to change the cost and impact of advertising, personnel, research, and OTA allocations.

**Scenario 24 – Account Management - User Changes their Password**

Student Phil, is logged into the System. Phil decided the password he choose for the System during account creation is not secure enough. Phil, from any page, selects the ‘My Account’ link. Phil then selects, “Change my password” button, enters in his current password and his new password and finishes by presses, “Change password”.

**Scenario 25 – Account Management - User Changes their Secret Question and Answer**

Student Phil, is logged into the System. Phil, decided that the secret question and secret answer he choose during account creation is not secure enough. Phil, from any page, selects the ‘My Account’ link. Phil then selects, “Change my Recovery Question/Ans” button, enters in his current password, his new secret Question, and secret answer for his secret question. Phil, finishes by presses, “Change my recovery question/ans”.

**Scenario 26 - Student commit and comment**

Robert is logged into the system and has already selected is period strategic decisions for period 6. he clicks the commit button to confirm his selections for the period. The system redirects him to the comments page where he, after a discussion with his group, types the reasons for selected decisions in the given textbox and clicks the commit button.

**Scenario 27 – User logs off**

Phil, has completed his task on the System and wishes to log his account out so his partner Tracy can log into the System and perform her accounts on the System. From any page, Phil, presses the ‘Sign Out’ link on the toolbar. A message is displayed that Phil’s email, Phil018@fiu.edu has been logged off. Tracy, now gets on the computer, presses the ‘Log in’ link on the same page.

**Scenario 28 – Change Decisions**

Student Phil, is getting ready to commit his decisions for the period when he changes his mind about the strategic decisions he choose. Phil, selects the ‘Change Decisions’ button on the home page. Phil, is navigated to the Strategic Decisions page.

Scenario 29 - Market Status Graph

Student Phil logs into the System. Phil is interested in his Hotel's market share compared to the rest of the class. Phil, looks in his Home page’s scorecard and sees that his market share is 15% of the whole market.

**Scenario 30 - User Leaderboard**

Student Phil is interested how his Hotel, Hotel Miami did last period compared to his classmates. Phil logs into the System and scrolls down to the leaderboard on the Home page. Phil, sees his Hotel is in second place.

**Scenario 31 - User scorecard**

The Game period has recently ended and Student Phil, is interested in how his Hotel’s market research and strategic decisions affected his Hotel. Phil, logs into the System and looks at his Hotel’s scorecard on the Home page. The Scorecard displays detailed information about his Hotel’s; such as the Hotels name, it’s budgets, it’s location, it’s type, results of his performed research, market share, and current status.

**Scenarios 32 - Professor provides feedback on end of period comments**

A period has ended for Professor Cilli’s morning course. Professor Cilli, logs into the System and navigates to the Reporting page. Professor Cilli, selects his game and the previous period. Professor Cilli, filters the results to only display comments for a specific type of Hotel. Professor Cilli, presses the ‘Give Feedback’ button, and enters in his feedback to a Student’s end of period comment. Professor Cilli, finishes with ‘Post Feedback’ button.

**Scenarios 33 - Professor navigates comment repository**

Professor Cilli, is logged into the System and navigates to the Reporting page. Professor Cilli, chooses Comment repository tab, and filters the comments to a specific Game, Period, and Hotel Type. Professor Cilli, is now freely able to browse through all the Students end of period comments and provide feedback as necessary.

**Scenario 34 – Professor downloads a report**

A period has ended for one of Professor Cilli’s, courses. Professor Cilli, will like to see how the Game played out as well as how each group responded to these changes and how they were affected by it. Professor Cilli, logs into the System and navigates to the Reporting page. Professor Cilli, selects the Reports tab, selects a desired Game, and Period. Professor Cilli, presses ‘Generate Report’ button and a pdf report is generated and can now be download to Professor Cilli’s computer.

**Scenario 35 – End of Period**

Dr. Roberts, a professor, has instructed in students to participate in a strategic marketing simulation. All of the groups have selected their decisions for the first period. Upon the end of the period, the system triggers the end of period algorithm to calculate the winner of the period and the respective market share.

**Scenarios 36 – Student views feedback**

Student Phil, is interested if his professor has provided feedback for his end of period comment last period. Phil, navigates to the reporting page via the toolbar. Phil, sees his comment from last period and is able to see that the professor did provide feedback and it was very positive.

## Use case model

**Use Case 1 – Student creates an account in the system**

**Actor:**

Student

**Pre-Conditions:**

Student does not currently have an account

**Normal Course:**

1. Student hits create an account button

2. Student fills out required information

3. An email is sent to students preferred email

4. Student receives the email and verifies that email

5. Student click link in email and is asked to enter a new password

**Alternative Course:**

**Email never received**

If in Step 4, the Student never receives the confirmation email then

1. Student must repeat entire process from step 1

**Syntax Error Personal Information**

If in step 2 the Student enters their personal information wrong then

1. The Student will be ask to correct their input before being able to proceed

**New Password is not valid**

If in step 5 the Student enters their password incorrectly then

1. The Student will be ask to correct their input before being able to proceed

**Post Conditions:**

**Student has successfully created an account**

The Student is viewing the homepage

**Student was unable to create an account**

Student must repeat the entire process again

**Use Case 2 – Admin views users that have joined his game**

**Actor:**

Administrator/Instructor

**Pre-Conditions:**

The Administrator has a created account

The Administrator has created a Game

**Normal Course:**

1. The Administrator logs in with their credentials

2. The Administrator navigates to the Manage page

3. Administrator selects the game in question

4. Administrator selects View Users button

5. Administrator is able to view all users registered to that game

**Alternative Course:**

**Login Credentials were incorrect**

If in Step 1, the Administrator types their credentials incorrectly

1. The administrator will be prompted email/password do not match

2. Administrator will have to repeat step 1 until able to login

3. Continue at Normal Step 2

If in Step 1, Administrator forgets their password

1. Administrator will press the Forgot Password button

2. Administrator provides his account email address

3. Received a recovery email from the System

4. Logs into the system with account email and new password

5. Is asked to set a new password

6 Continue at Normal Step 2

**Post Conditions:**

**Administrator was successful**

The View Users page will display all users registered to his game

**Administrator was unsuccessful**

The Administrator will have to repeat step 1

**Use Case 3 - Student user commits their changes**

**Actor:**

Student

**Pre-Conditions:**

1. Student has an account

2. Student is registered in a game

3. Student has logged into the System

**Normal Course:**

1. Student navigates to the Strategic Decision page via the Strategic Decision button

2. Student applies his strategic decisions for advertising, hiring personnel, room pricing, and room allocation for third party online vendors.

3. Student is happy with his changes and presses the Commit button to save his Strategic Decisions for the period

4. Student is prompted to enter his reasoning for his Strategic Decisions

5. Student types in his reasoning and presses Commit Changes

**Alternative Course:**

**Strategic Decisions incorrect parameters**

If in Step 2, the student inputs incorrect values for pricing or room allocations

1. The System will notify the Student to correct the values before being able to processed

2. Student will repeat Step 2

**Post Conditions:**

**Student was Successful**

1. Student was able to commit their decisions to the database

2. The System will lock them out from committing any more decisions to the database

Student was unsuccessful

1. Student was unable to commit changes to the database

2. Students will have to repeat Step 1 through 5

**Exceptions:**

**The Student didn’t not commit any changes for the period**

The Student’s hotel will use the previous period values or default values if no previous period exist

**Use Case 4 - Professor removes a user from the database**

**Actor:**

Administrator/Instructor

**Pre-Conditions:**

The Administrator has a created account

There are Users in the Database

The Administrator is logged into the System

**Normal Course:**

1. Administrator navigates to the Manage page

2. Administrator navigates to the View All Users page

3. Administrator selects the user they wish to deactivate in the database

4. Administrator presses the

**Alternative Course:**

**Login Credentials were incorrect**

If in Step 1, the Administrator types their credentials incorrectly

4. The administrator will me prompted email/password do not match

5. Administrator will have to repeat step 1 until able to login

6. Continue at Normal Step 2

If in Step 1, Administrator forgets their password

7. Administrator will press the Forgot Password button

8. Administrator provides his account email address

9. Received a recovery email from the System

10. Logs into the system with account email and new password

11. Is asked to set a new password

12. Continue at Normal Step 2

**Post Conditions:**

**Administrator was successful**

The View Users page will display all users registered to his game

**Administrator was unsuccessful**

The Administrator will have to repeat step 1

**Use Case 8 – Forgot my password**

**Actor:**

Student

**Pre-Conditions:**

Student forgets his or her password

**Normal Course:**

1. Student navigates to forgot my password page

2. Student enters email or PID in text box

3. Use clicks on “find my account” button

4. The user then answers the security question

5. Student enters new password in textbox

6. Student presses “change my password” button

7. Students Password has been changed

**Alternative Course:**

**Student email did not match**

Student will have to repeat step 2

**Student secret Answer did not match**

Student will have to repeat from step 1

**Student did not enter a valid password**

Student will have to repeat step 5

**Post Conditions:**

**Student was successful**

Student has a new password and can log into the system.

**Student was unsuccessful**

The Student will have to repeat from step 1

**Use Case 9 - Admin Login**

**Actor:**

Admin

**Pre-Conditions:**

Admin is not already logged into the system

Admin has an account with the system

**Normal Course:**

* 1. Admin navigates to the login page.
  2. Admin enters his or her username and password and clicks login button

**Use Case 10 - Student Strategic decisions**

**Actor:**

Student

**Pre-Conditions:**

Student has an account with the system

Student is logged into the system.

Student already belongs to a group

**Normal Course:**

1 A logged in student clicks the Strategic Decisions link

2 User selects Market segment, advertising, average room rate, personnel, OTA allocations, and/or research and clicks select decisions button.

**Use Case 11 - Admin Strategic decisions Management**

**Actor:**

Admin

**Pre-Conditions:**

Admin is logged into the system

**Normal Course:**

1 A logged in admin clicks the Strategic Decisions link

2 Admin can change the cost and/or impact of selects Market segment, advertising, average room rate, personnel, OTA allocations, and/or research and clicks select decisions button.

**Use Case 17 - Student Views News Page**

**Actor:**

Student

**Pre-Conditions:**

Is logged into the System

is Registered to a Game

**Normal Course:**

* 1. Student selects the 'News' link on their toolbar
  2. Student is Shown the news article for the current Game Period

**Use Case 18 - Admin changes a period**

**Actor:**

Admin

**Pre conditions:**

Is logged into the System

Is viewing a news article for a game

**Normal Course**

1. Admin selects the ‘Change Period’ button
2. Admin chooses a Period/News article from the list of all News articles for that game.
3. Admin is now viewing that Periods news article for that Game.

**Use Case 19 - Admin removes News Parameters**

**Actor:**

Admin

**Pre-conditions**

Is logged into the System

Is viewing a news article for a game

**Normal Course**

1. Admin selects the ‘Remove News Parameters' button
2. Admin selects the ‘Clear Database Parameters' button
3. Admin selects the ‘Clear Session Parameters' button

**Alternative Course**

**If admin does not perform Step 2**

Only Session News Parameters will be removed

**If admin does not perform Step 3**

Only Saved News Parameters will be removed

**Post-Condition**

**All parameters have been removed**

The News Article will no longer have any parameters

**All Session Parameters have been removed**

The News article will no longer have any session parameters

**All Database parameters have been removed**

The News article will no longer have any saved parameters

**Exception**

**There were not any news parameters to be removed**

A message will be shown that it was unable to remove news parameters

**Use Case 20 - Admin commits a News Article**

**Actor:**

Admin

**Pre-Conditions:**

Admin, is logged into the System

A Game is Active

**Normal Course:**

1. Admin selects the News link from the Toolbar
2. Admin, selects a Game from the list of active games.
3. Admin, checks if the period of the News Article
4. Admin, writes in or pastes an News Article in the text area
5. Admin Adds the necessary parameters to tell the System the impact of this news article
6. Admin, selects the 'Save And Preview' Button and Reviews that any html entered looks appropriate
7. Admin, is pleased and pressed the 'Commit to Database' button to save the article and its parameters to the Database

**Alternative Course:**

**Current Period is not the desired Period**

* 1. If in Step 3, the Admin see this is not the period he wishes to modify
  2. Admin will then select the 'Change Period' button
  3. Admin will choose the desired period from the list of periods registered to that Game.

**Post Conditions:**

**Admin was Successful**

The Admin committed the news article to the Database

**Admin was Unsuccessful**

1. Admin never hit the 'Commit to Database' button
2. News article was not saved to the Database

**Use Case 21 - Admin Chooses/Changes Game News Article**

**Actor:**

Admin

**Pre-Conditions:**

Is logged into the System

Is on the News Page

**Normal Course:**

1. Admin is working on a news articles for a particular Game and Wishes to switch to another Game
2. Admin selects the 'Choose Game' button and is shown a list of all Games
3. Admin selects a Game and presses the 'Choose A Game' Button
4. The chosen Games current period news article is displayed with its parameters and news article content

**Post Conditions:**

**Admin successfully changed Games**

Admin is not viewing the news article of a different game and can modify it.

**Exceptions:**

**Game does not have a News Article**

If for whatever reason, if a Game does not have a News article. an error message will be displayed stating that a news article was not found.

**Use Case 24 - Account Management - User Changes their Password**

**Actor:**

Student

Admin

**Pre-Conditions:**

Is logged into the System

**Normal Course:**

1. Actor selects the 'My Account' link on their toolbar
2. Actor selects ‘Change My Password’
3. Actor enters in their current password
4. Actor enters in their new current password
5. Actor presses ‘Change Password’ button

**Alternative Course**

**If Actor does not enter in their correct password in Step 3**

Actor will have to repeat Step 3 until the current password is entered

**If Actor does not enter in a valid password in Step 4**

Actor will have to repeat Step 3 until a valid password is entered in Step 4

**Post-Condition**

**Current password has been updated**

The Actors account has new account password

**Current password was not changed**

The Actors was unsuccessful in changing their account password

**Use Case 25 - Account Management - User Changes their Secret Question and Answer**

**Actor:**

Student

Admin

**Pre conditions:**

Is logged into the System

**Normal Course**

1. Actor selects the 'My Account' link on their toolbar
2. Actor selects ‘Change My Recovery Question’
3. Actor enters in their current password
4. Actor enters in their new secret question
5. Actor enters in their new secret answer
6. Actor presses ‘Change Password’ button

**Alternative Course**

**If Actor does not enter in their correct password in Step 3**

Actor will have to repeat Step 3 until the current password is entered

**If Actor does not enter in a valid secret question in Step 4**

Actor will have to repeat Step 3 until a valid secret question is entered in Step 4

**If Actor does not enter in a valid secret answer in Step 5**

Actor will have to repeat Step 3 until a valid secret answer is entered in Step 5

**Post-Condition**

**Current secret question and answer has been updated**

The Actors account has new account password

**Current secret question and answer was not changed**

The Actors was unsuccessful in changing their secret question and answer

**Use Case 26 - Student commit and comment**

**Actor:**

Student

**Pre-Conditions:**

Is logged into the System

Has selected their strategic decisions

**Normal Course:**

1. Actor navigates to the Home page via the nav bar
2. Actor press the ‘Commit Period’ button
3. Actor enters in their reasoning for into the textbox for their strategic decisions
4. Actor finishes up but pressing ‘Commit Comment’

**Alternative Course**

**If Actor did not write anything for their reasoning**

Actor will have to repeat Step 3 until valid input has been entered

**Post-Condition**

**Actor was successful**

The Actor’s has successfully committed their decisions for the period

**Actor was unsuccessful**

Actor’s decisions were not saved to the database

**Use Case 27 - User Logs Off**

**Actor:**

Student

Admin

**Pre conditions:**

Is logged into the System

**Normal Course**

1. Actor selects the 'Sign Out' link on their toolbar
2. Actors account has now been logged out of the System

**Post-Condition**

**Actor was successfully logged out**

The Actors session has been cleared

**Use Case 28 – Change Decisions**

**Actor:**

Student

**Pre conditions:**

Is logged into the System

Is on the Home page

**Normal Course**

1. Actor selects ‘Change Decisions’ button
2. Actor is navigated to the Strategic Decisions page

**Post-Condition**

**User is on the Strategic Decisions page**

The Actors was redirected to Strategic Decisions page

**Use Case 29 - Market Status Graph**

**Actor:**

Student

**Pre-Conditions:**

Student is registered to a Game

Student is registered to a Group

**Normal Course:**

1. The Student logs in with their credentials

2. The Student navigates to the Scorecard on the Home page

3. Student is able to see a pie chart with the market shares of all the Hotels on the market

**Alternative Course:**

**Login Credentials were incorrect**

If in Step 1, the Student types their credentials incorrectly

1. The Student will be prompted email/password do not match

2. Student will have to repeat step 1 until able to login

3. Continue at Normal Step 2

**Post Conditions:**

**Student was successful**

The Student was able to see his and every other Hotels market share on the market

**Student was unsuccessful**

The Student was unable to log in.

Student will have to repeat step 1

**Use Case 30 - Users Leaderboard**

**Actor:**

Student

**Pre-Conditions:**

Student is registered to a Game

Student is registered to a Group

**Normal Course:**

1. The Student logs in with their credentials

2. The Student navigates to the Leaderboard on the Home page

3. Student is able to see their ranking compared to all the other Hotels in their Game

**Alternative Course:**

**Login Credentials were incorrect**

If in Step 1, the Student types their credentials incorrectly

1. The Student will be prompted email/password do not match

2. Student will have to repeat step 1 until able to login

3. Continue at Normal Step 2

**Post Conditions:**

**Student was successful**

The Student was able to see their and every other Hotels ranking

**Student was unsuccessful**

The Student was unable to log in.

Student will have to repeat step 1

**Use Case 31 - User Scorecard**

**Actor:**

Student

**Pre-Conditions:**

Student is registered to a Game

Student is registered to a Group

**Normal Course:**

1. The Student logs in with their credentials

2. The Student navigates to the Scorecard on the Home page

3. Student is able to detailed information about their Hotel and Market research results

**Alternative Course:**

**Login Credentials were incorrect**

If in Step 1, the Student types their credentials incorrectly

1. The Student will be prompted email/password do not match

2. Student will have to repeat step 1 until able to login

3. Continue at Normal Step 2

**Post Conditions:**

**Student was successful**

The Student was able to see information about their Hotel

**Student was unsuccessful**

The Student was unable to log in.

Student will have to repeat step 1

**Use Case 32 - Professor provides feedback on end of period comments**

**Actor:**

Admin

**Pre-Conditions:**

1. Is logged into the System
2. A period has finished
3. A Game in the database

**Normal Course:**

1. Actor navigates to the Reporting page
2. Actor selects the ‘Comment Repository’ tab
3. Actor is Shown all the end of period comments for that Game/Period
4. Actor presses the ‘Provide Feedback’ button for one of the comments
5. Actor types in their feedback
6. Actor finishes by presses the ‘Post Feedback’ button

**Alternative Course**

**If Actor filters results after Step 3**

1. Actor can filter results by selecting a Game, Period, Hotel type
2. Actor continues unto step 4

**Post-Condition**

**Actor provided Feedback**

1. The Actors provided feedback to a Hotel's end of period comment.
2. The Student’s is now able to see the Professor feedback

**Actor did not provide feedback**

The Student will not see any entrees from their professor commenting on their performance.

**Use Case 33 - Professor navigates comment repository**

**Actor:**

Admin

**Pre-Conditions:**

1. A period has finished
2. A Game in the database

**Normal Course:**

1. Actor logs into the System with their administrative credentials
2. Actor navigates to the Reporting page
3. Actor selects the ‘Comment Repository’ tab
4. Actor is Shown all the end of period comments in the System

**Alternative Course**

**If Actor filters results after Step 4**

Actor can filter results by selecting a Game, Period, or Hotel type

**Post-Condition**

**Actor was successful**

The Actors was able to browse through all the end of period comments in the System.

**Use Case 34 - Professor downloads a report**

**Actor:**

Admin

**Pre-Conditions:**

1. Is logged into the System
2. A period has finished
3. A Game in the database

**Normal Course:**

1. Actor navigates to the Reporting page
2. Actor selects the ‘Reports’ tab
3. Actor selects a Game
4. Actor selects a period
5. Actor presses ‘Generate Report’

**Post-Condition**

**Actor was successful**

The Actors was able to download a detailed report for a Game/Period

**Exceptions:**

**If Actor did not select a Game in Step 3**

1. Actor will be unable to proceed
2. Actor will have to repeat Step 3

**If Actor did not select a Period in Step 4**

1. Actor will be unable to proceed
2. Actor will have to repeat Step 4

**Use Case 36 - Student views feedback**

**Actor:**

Student

**Pre-Conditions:**

Is logged into the System

User is registered to a Game

**Normal Course:**

* 1. Actor navigates to the Reporting page
  2. Actor selects the ‘View Feedback’ button

**Post-Condition**

**Actor viewed Feedback**

The Actor was able to view the professor’s feedback for a previous period

## Hardware and Software Resources

**Software**

**Operating systems:**

Windows 7, Windows 8, Windows 10, and Ubuntu Server 14.04.3 TLS.

**Development:**

PHP5, HTML5, CSS, Bootstrap, JSON String, JQuery, and Javascript.

**Database:**

Mysql

**Webserver:**

Apache 2

**Testing:**

PHPUnit, Qunit

**Remote access tools:**

Putty, Winscp, Logmein, and teamviewer

**Project planning tools:**

Mingle.com(Scrum tracking), Google Docs, and Microsoft Project

**Presentation:**

Microsoft Word, Microsoft PowerPoint, Microsft Excel, Microsoft Visio, StarUml, gotomeeting.com (Sprint Review), and Camtasia 8 (video production).

**Hardware:**

Lenovo Y50-80 Laptop computer: Intel Core i7 processor with 16GB RAM

HP Pro 3500 Desktop computer: Intel Core i5 processor with 8GB of RAM

## Sprints Plan

### Sprint 1

(08/31/2015 - 9/11/2015)

**User Story # 666- Account Creation**

***Tasks***

* Insert database tables for users, groups, and courses (database subsystem).
* Setup blank home page with login link.
* Setup login page (username, password text boxes; “forgot my password”, “new user” links; and login button.
* Setup account creation webpage (username, password, PID, email text boxes; and a list of courses available.
* Setup PHP class user with getter/setter methods and boolean value for isAdmin.
* Setup PHP Database class and methods for updating user and group tables of the database subsystem
* write scripts to tie it all together.

***Acceptance Criteria***

* The username must be unique.
* The PID must be unique
* Password must be alphanumeric
* System sends verification email to user
* The user clicks verification link
* The system verifies user and create entry in the database subsyste*m.*

***Modeling***

See Appendix A

Figure 7: sequence diagram

Figure 9: use case diagram

**User Story # 669- User login**

***Tasks***

* Add methods to PHP Database class for user login.
* write scripts to communicate with to Database subsystem from front end.
* write scripts to navigate user to homepage upon success.
* Appropriate error message upon value.
* (Future release should have account lockout policy)

***Acceptance Criteria***

* The user must enter a username and password which are associated with a record in the database subsystem.
* ***Modeling***

See Appendix A

Figure 6: sequence diagram.

Figure 8: use case diagram.

### Sprint 2

(09/14/2015 - 9/25/2015)

**User Story # 666- Account Creation**

***Tasks***

* Insert database tables for users, groups, and courses (database subsystem).
* Setup blank home page with login link.
* Setup login page (username, password text boxes; “forgot my password”, “new user” links; and login button.
* Setup account creation webpage (username, password, PID, email text boxes; and a list of courses available.
* Setup PHP class user with getter/setter methods and boolean value for isAdmin.
* Setup PHP Database class and methods for updating user and group tables of the database subsystem
* write scripts to tie it all together.

***Acceptance Criteria***

* The username must be unique.
* The PID must be unique
* Password must be alphanumeric
* System sends verification email to user
* The user clicks verification link
* The system verifies user and create entry in the database subsyste*m.*

***Modeling***

See Appendix A

Figure 7: sequence diagram

Figure 9: use case diagram

**User Story # 669- User login**

***Tasks***

* Add methods to PHP Database class for user login.
* write scripts to communicate with to Database subsystem from front end.
* write scripts to navigate user to homepage upon success.
* Appropriate error message upon value.
* (Future release should have account lockout policy)

***Acceptance Criteria***

* The user must enter a username and password which are associated with a record in the database subsystem.
* ***Modeling***

See Appendix A

Figure 6: sequence diagram.

Figure 8: use case diagram.

**User Story # 667- Forgot my password**

* The user clicks on the “forgot my password” link from the login page.
* The system generates a random temporary password.
* The system sends the use an email with his or her temporary password enclosed.
* The user logs onto the system using the temporary password.
* The system requires that the user reset his password by entering it in two text boxes.
* The user’s password is reset.

***Tasks***

* Forgot my password page
* Database functions

***Acceptance Criteria***

* A student has an account.

***Modeling***

Appendix A

Figure 10 Forgot my password sequence diagram

Figure 11 Forgot my password use case diagram

### Sprint 3

(09/28/2015 - 10/9/2015)

**User Story # 776 - Student Create Group**

* The student decides to create a new group and clicks on the “create a new group” from the join group page.
* The student clicks on the games drop down list and selects a game.
* The student enters a group name in the group name text box
* The student clicks on the create new group button.
* The user is redirected to his or her homepage.

***Tasks***

* Update Database class
* Create new group page
* Insert new Group helper page

***Acceptance Criteria***

* A student has an account.
* Student is logged in.
* Student does not already belong to a group.

***Modeling***

Appendix A

Figure 12 Join Group User case Diagram

Figure 13 Join Group Sequence Diagram

**User Story # 769 - Student Join Group**

* A new user is directed to Join group page. Student has the option to join a already created group or create a new group.
* If a user joins a group they first select the game from the game drop down list.
* They next select a group from the dynamically populated group drop down list.
* The user clicks join group.
* The user is directed to their homepage.

***Tasks***

* Update Database class
* Create Join group page
* Insert new Group helper page

***Acceptance Criteria***

* A student has an account.
* Student is logged in.
* Student does not already belong to a group.
* Student clicked the link “Create a group” form the join new group page.

***Modeling***

***Appendix A***

Figure 14 Create Group User case Diagram

Figure 15 Create Group Sequence Diagram

**User Story # 718 - Manage Page**

* As a logged in admin I need to be able to navigate to the manage page so that manage users, manage games, and manage administration
* A logged in admin clicks on the manage link
* The admin can manage users, games and administration.

***Tasks***

* ***Manage admin***
* ***Master admin account***
* ***Sign off***
* ***Create Page Template***
* ***Activate users***
* ***Update database class***
* ***Admin Creation***
* ***View All users in a game***
* ***View All games***
* ***View all Users***

***Acceptance Criteria***

* A admin has an account.
* Admin is logged in.

***Appendix A***

Figure 18 Admin Manage User Case Diagram

Figure 19 Activate/deactivate Student Sequence Diagram

Figure 22 view all users sequence diagram.

**User Story #681 Admin “Users” Page**

Description:

* As a logged in admin I need to be able to click on the "users" link from my homepage so that I can see a list of all users
* A logged in in admin clicks on users from the homepage.
* The system navigates to the users page

**Tasks:**

* Return an array of all Users from Database

Acceptance Criteria:

1. An admin is logged in.
2. There are user records in the database

**User Story # 684 Bot Creation**

Description:

* As a logged in admin I need to be able to create a bot account
* A logged in admin navigates to the create user page
* The admin clicks the create new user button.
* The admin enters the username and group./
* The admin ticks the “is a bot” check box
* The admin clicks the create user button

Tasks:

* Create a Bot Student Account
* Create Hotel for Bot Student Account

Acceptance Criteria:

1. The admin must be logged in to the system.
2. The bot username must not already be in use.

***Appendix A***

Figure 21 Create bot sequence diagram

**User Story # 678- Admin Account Creation**

**Description**

* A logged in admin can create an admin account.
* Admin accounts can view and change games, users, and groups.

**Acceptance Criteria**

* Admin Must be logged into the system.
* New admin must have an email address.

***Appendix A***

Figure 20 Create admin sequence diagram

**User Story # 711 Deactivate Users**

Description:

* As a logged in admin I need to be able to deactivate users so that I can remove users from logging into the system
* A logged in admin clicks, de-activate user.
* A confirmation dialog box appears. the admin clicks “yes”
* The user account is deactivated from the system

**Tasks:**

* Set Admin Entities as not Active in the Database
* Set Student Entities as not Active in the Database

Acceptance Criteria:

1. An admin is logged in.
2. There are users in the database
3. The User is not the Main Admin Account

**User Story # 678- User strategic decisions (unfinished this sprint - started on early)**

* As a logged in user, I need to be able to click on "Strategic Decisions" link from the toolbar so that I can assign values to the budgetary items: Marketing personnel, advertising media, room pricing, and third party vendors
* A user clicks on the strategic decisions link.
* System navigates to strategic decisions page.

***Tasks***

* jQuery frontend balance update
* Marketing personnel column
* Update database functions
* Advertising column
* Strategic decisions page
* Market Segment column

***Acceptance Criteria***

* A student has an account.
* Student is logged in.

***Appendix A***

Figure 16 Strategic Decisions User case Diagram

Figure 17 Strategic Decisions Sequence Diagram

### Sprint 4

(10/12/2015 - 10/23/2015)

**User Story #800 – Admin News Management**

Description:

* Admin is able to write or paste a News Article for a period, for a game.
* the Admin is able to apply News Parameters to an article. these Parameters apply an affect to specific Hotels for that Period, in a Game.
* Administrators can Choose which Game to Views the News for.
* Administrators can Choose Which Period in a Game to view/Modify the News for.
* Administrator can Save their changes to the Database or erase everything and start over.

**Tasks:**

* Add methods to Database class to create and return arrays of entries.
* Allow Admin to Choose a Game and a Period
* Admin can save an Article and affect parameters for a Games Period
* Admin can remove/change saved content
* Appropriately display News Article HTML upon preview

Acceptance Criteria:

* Admin is logged into the system
* There are Active Games on the System

**Appendix A**

Figure 25 - News Page use case diagram

Figure 26 - Admin News Choose a Game Sequence Diagram

Figure 27 - Admin News Choose a Period Sequence Diagram

Figure 28 - Admin News Save And Commit Sequence Diagram

**User Story #679 – User View News**

Description:

* As a logged in user I need to be able to click on the "news" link to view current events so that I can make an informed decision about how to allocate my marketing budget
* user clicks on the news button.
* The system navigates to the new pages
* News articles are listed

**Tasks:**

* Add methods to Database class to create a Game
* Add a method to Database class to return arrays of all games
* Display the News article for that Users Game and it’s Current Period

Acceptance Criteria:

* Student is logged into the system
* Student is registered to a Game

**Appendix A**

Figure 25 - News Page use case diagram

Figure 29 - Student News View Article Sequence Diagram

**User Story # 810- Admin strategic decisions management**

* As a logged in admin , I need to be able to click on "Strategic Decisions" link from the toolbar so that I can assign values and impacts to budgetary items: Marketing personnel, advertising media, room pricing, and third party vendors

***Tasks***

* Create Strategic Decisions Manage Page
* Update database with functions

***Acceptance Criteria***

* An admin is logged in
* Admin clicks on strategic decisions link.

***Appendix A***

Figure 24: Admin strategic Decisions management use case diagram

Figure 31: Admin strategic Decisions management sequence diagram

**User Story # 678- User strategic decisions**

* As a logged in user, I need to be able to click on "Strategic Decisions" link from the toolbar so that I can assign values to the budgetary items: Marketing personnel, advertising media, room pricing, and third party vendors
* A user clicks on the strategic decisions link.
* System navigates to strategic decisions page.

***Tasks***

* jQuery frontend balance update
* Marketing personnel column
* Update database functions
* Advertising column
* Strategic decisions page
* Market Segment column

***Acceptance Criteria***

* A student has an account.
* Student is logged in.

***Appendix A***

Figure 16 Strategic Decisions User case Diagram

Figure 17 Strategic Decisions Sequence Diagram

**User Story # 704 - Admin Login**

* As an admin I need to be able to log into the system so that I can manage games and users
* An admin navigates to the login page.
* An admin enters his username and password into the corresponding text boxes

***Tasks***

* Create admin check at login with session
* Create admin session and redirects for all pages.

***Acceptance Criteria***

* A user has an admin account with valid credentials

***Appendix A***

Figure 23 Admin login use case diagram

Figure 32 Admin login Sequence Diagram

### Sprint 5

(10/26/2015 - 11/06/2015)

**User Story #816 – Account Management**

Description:

* From this page, a User will be able to change their password, change their secret question, and change their secret answer.
* This page will also display some information about their account, such as email, name, and panther id if available.

**Tasks:**

* retrieve and display user's account information
* update user's password
* update user’s secret question and secret answer
* ask for current password before any changes

**Acceptance criteria:**

* A User is logged into the system

***Appendix A***

Figure 34 Account Management Change Password

Figure 35 Account Management Change Recovery Question/Answer

Figure 37 Account Management - User changes their password

Figure 38 Account Management - User changes secret answer and question

Figure 39 Account Management - Use Case Diagram

**User Story #822 – Sign Out**

Description:

* End the users session, logging them out the System.

**Tasks:**

* Ends the users session(s)

**Acceptance criteria:**

* User is logged into the system

***Appendix A***

Figure 36 User Logs off

Figure 40 User Logs off - Use Case Diagram

**User Story # 712 - User comment and commit**

* A logged in user navigates to home page after selecting strategic decisions for a period
* A user clicks the “commit” from the home Page.
* The system redirects the student to the comments page..
* The user types in reason for his or her decisions and clicks the “commit” button

**Tasks:**

* Homepage view
* Period Controller
* Update database functions
* Create comment page view

**Acceptance criteria:**

* A student is logged into the system
* A student already belongs to a group
* A student has already selected strategic decisions for the period

***Appendix A***

Figure 33 Student commit and comment sequence diagram

Figure 41 Home page Use Case Diagram

Figure 42 Student commit and comment use case diagram

**User Story # 708 - User Scorecard**

* The scorecard is part of the homepage view that shows a student their status versus the status of the rest of the market:
  + Hotel type
  + Location
  + Advertising
  + Room Allocated
  + Remaining Budget
  + Marketshare
  + Market Share pie chart(optional for release 1)
  + Group Name
  + Market Segment

**Tasks:**

* Update period controller
* Update homepage view
* Update database model functions

|  |  |  |
| --- | --- | --- |
| **Number** | **Name** | **Status** |
| [838](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/strategic_marketing_simulator_/cards/838) | [Update period controller](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/strategic_marketing_simulator_/cards/838) | Done |
| [837](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/strategic_marketing_simulator_/cards/837) | [update homepage view](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/strategic_marketing_simulator_/cards/837) | Done |
| [836](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/strategic_marketing_simulator_/cards/836) | [Update database model functions](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/strategic_marketing_simulator_/cards/836) | Done |

**Acceptance criteria:**

* A student is logged into the system
* A student already belongs to a group
* A student navigates to the homepage view

***Appendix A***

**User Story # 671- User leaderboard**

* The leaderboard shows the total revenue for each from in descending order for the previous period.
* It also have a bar graph as a visual representation of the same

**Tasks:**

* Update period controller
* Javascript google development chart (bar graph)
* Update database with functions

|  |  |  |
| --- | --- | --- |
| **Number** | **Name** | **Status** |
| [835](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/strategic_marketing_simulator_/cards/835) | [Update periodControler](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/strategic_marketing_simulator_/cards/835) | Done |
| [830](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/strategic_marketing_simulator_/cards/830) | [javascript google development chart (bar graph)](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/strategic_marketing_simulator_/cards/830) | Done |
| [829](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/strategic_marketing_simulator_/cards/829) | [Update database with functions](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/strategic_marketing_simulator_/cards/829) | Done |

**Acceptance criteria:**

* A student is logged in.
* A student already belongs to a group
* A student has selected their market segment
* At least 1 period has been completed.

Appendix A

**User Story # 672- Market Status Graph**

* The Market Segment graph is a visual representation of the market share for each from from the previous period.

**Tasks:**

|  |  |  |
| --- | --- | --- |
| **Number** | **Name** | **Status** |
| [**834**](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/strategic_marketing_simulator_/cards/834) | [**update homepage view**](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/strategic_marketing_simulator_/cards/834) | **Done** |
| [**833**](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/strategic_marketing_simulator_/cards/833) | [**update database wtih function**](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/strategic_marketing_simulator_/cards/833) | **Done** |
| [**832**](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/strategic_marketing_simulator_/cards/832) | [**Update periodcontroller**](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/strategic_marketing_simulator_/cards/832) | **Done** |
| [**831**](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/strategic_marketing_simulator_/cards/831) | [**Google Developer charts (pie chart)**](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/strategic_marketing_simulator_/cards/831) | **Done** |

**Acceptance Criteria:**

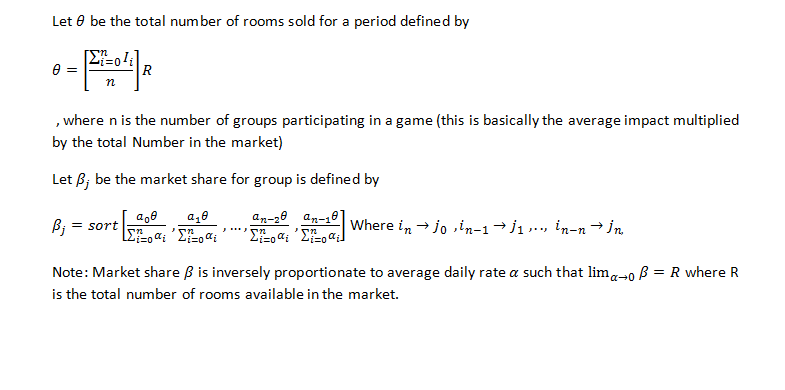
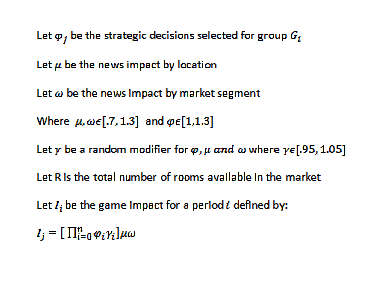
* A user is logged into the system
* A user belongs to a group
* At least one period for a game has been completed

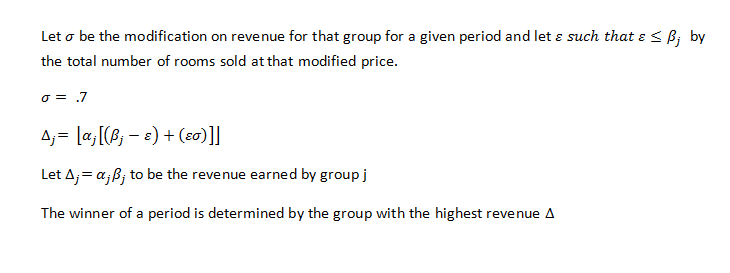
### S***print 6***

(11/09/2015 - 11/19/2015)

**User Story # 847 End of Period Algorithm**

* At the end of a each period the total market share for each group and the revenue earned is calculated using the following formulas which depend of the impact of each group’s strategic decisions, news impact on market segment and/or location, OTA, and averate rate.
* The following formulas calculate the end of period market share and revenue:





**Tasks:**

|  |  |  |
| --- | --- | --- |
| **Number** | **Name** | **Status** |
| [**851**](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/strategic_marketing_simulator_/cards/851) | [**update periodController**](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/strategic_marketing_simulator_/cards/851) | **In Dev** |
| [**850**](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/strategic_marketing_simulator_/cards/850) | [**update strategic decisions**](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/strategic_marketing_simulator_/cards/850) | **In Dev** |
| [**849**](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/strategic_marketing_simulator_/cards/849) | [**database functions**](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/strategic_marketing_simulator_/cards/849) | **In Dev** |
| [**848**](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/strategic_marketing_simulator_/cards/848) | [**gameController**](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/strategic_marketing_simulator_/cards/848) | **In Dev** |

**Acceptance Criteria:**

* A user is logged into the system
* A user belongs to a group
* The period timer must have ran out of the period

***Appendix A***

Figure 49 -

**User Story #840– Comment Repository**

Description:

* A page where an Admin user can view all Comments for a Game’s Period.
* Admin can also filter Comments by Hotel Type

**Tasks:**

* all comments for all games and periods should be displayed
* apply a filtering feature that will reduce the displayed comments to only the ones that match the filter

**Acceptance criteria:**

* A period has ended
* A game is Active or has finished
* Game has Student users
* Is an Admin user

***Appendix A***

Figure 46 Professor navigates Comment Repository

Figure 48 Use Case diagram - Reporting

**User Story #841– Professor Portfolio**

Description:

* Admin is able to download a detailed report of how Each Hotel did during the last period, as well as all the groups as a whole performed

**Tasks:**

* Gather relative information for the selected game and selected period and structure and print it to a downloadable PDF

**Acceptance criteria:**

* Period has ended
* Student users in Game
* is an Admin

***Appendix A***

Figure 47 Professor Generates a report

Figure 48 Use Case diagram - Reporting

**User Story #842– Professor Feedback**

Description:

* Professor is able to give their feedback at the end of period for a specific Group based on their strategic decisions and their end of period comment.

**Tasks:**

* This GUI should display the comment that receiving the feedback and should display any existing feedback if it exists
* professor can write their feedback and post it to the Database for a specific comment

**Acceptance criteria:**

* A period has ended
* Is Admin

***Appendix A***

Figure 45 Professor provides feedback Sequence diagram

Figure 48 Use Case diagram - Reporting

### Sprint 7

(11/23/2015 - 12/4/2015)

User Story #853 - Student Reporting Page

Description:

* This page will display their end of period comments as well as any feedback the professor has provided

**Tasks:**

* Display all of the user's end of period comments and their feedback provided by the professor

Acceptance Criteria:

1. user is a student
2. user is registered to a Game

***Appendix A***

Figure 49 Student Views end of period feedback sequence diagram

**User Story #856 - Bot Play**

Description:

* End of period date time check is made when a student visits the home page. If the current date and time is after the end of period, the game controller is called to see if there are any bots at play in the game. If there are bots at play, random strategic decisions as selected for that bot and the end of period algorithm is called.

**Tasks:**

|  |  |  |
| --- | --- | --- |
| **Number** | **Name** | **Status** |
| [858](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/strategic_marketing_simulator_/cards/858) | [Update database with functions](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/strategic_marketing_simulator_/cards/858) | Testing |
| [857](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/strategic_marketing_simulator_/cards/857) | [Update period Controller functions](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/strategic_marketing_simulator_/cards/857) | Testing |

Acceptance Criteria:

1. User is a student
2. User is registered to a Game
3. User belongs to a group
4. Current Date time is after the end of period date time

***Appendix A***

Figure 50 Bot play sequence Diagram

# System Design

The Strategic Marketing Simulator is composed of three subsystems. The database subsystem is responsible for all database transactions being made by any class in the system. The other subsystems communicate with the database to perform a variety of tasks such as adding a user record to the database or accesses a value needed for the Market subsystem. The Market subsystem is responsible for interpreting the effect of a group’s strategic decisions on the market. The Market subsystem communicates with the database subsystem to retrieve information about which groups are applying a certain decisions. Once the Market Ranking subsystem has made a determination, it can then pass the values to the Database subsystem storage for later use.

The third subsystem is the Main Controller subsystem. The Main Controller handles the game creation and keeps track of the current periods. The Main Controller subsystem communicates with both the database subsystem and the Market subsystem such that market information stored in the database subsystem can be used in conjunction with the Market subsystem to display market results on a webpage.

## Architectural Patterns

The architectural patterns used in the development of this system were three-tier and model-view-controller patterns. These patterns were adopted because during the requirements elicitation of analysis phase of the project, apparent subsystems began to emerge that have a three-tier like pattern where views are apparent, which has a controller, database, and a graphic user interface(views). Taking into consideration that emerging pattern and the team’s understanding of the system made the model-view-controller also relevant due to the apparent subsystems’ disconnect between between one another in functionality and requirements.

## System and Subsystem Decomposition

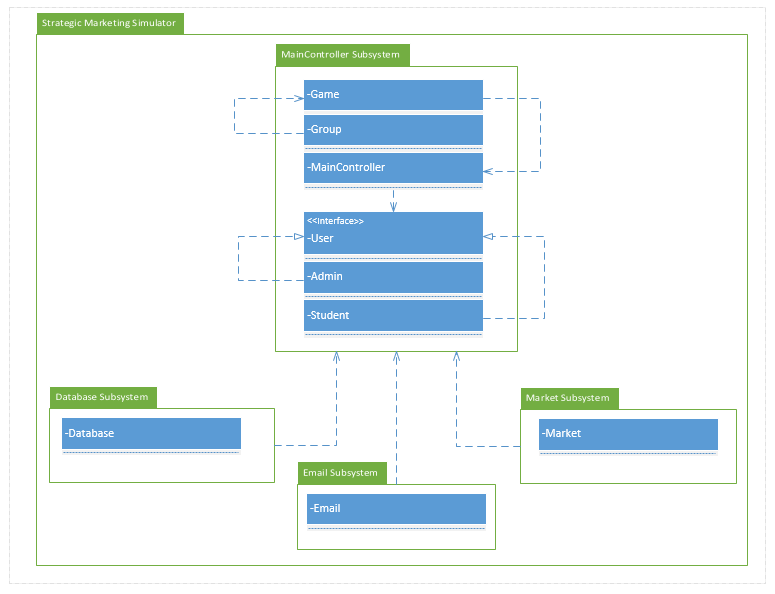


Figure 1: Package Diagram. The Strategic Marketing Simulator system can be decomposed into four subsystems. The MainController subsystem is responsible for creating games, users, and groups. All system input and output is communicated using the MainController subsystem as an intermediate between the front end and the rest of the back end. The second system, the database subsystem is responsible for all database transactions and the storing of game and user data. The third subsystem is the Market subsystem which is responsible for calculating all groups’ Strategic Decisions at the end of each period, and passes values to the MainController subsystem for display on the back end or to be handed to the Database subsystem storage for later use. The fourth subsystem is the Email subsystem, which is responsible for sending emails to users upon account creation and for password reset.

## Deployment Diagram

Using a UML deployment diagram, illustrate which subsystems will reside on each hardware component and show how the different pieces are connected.

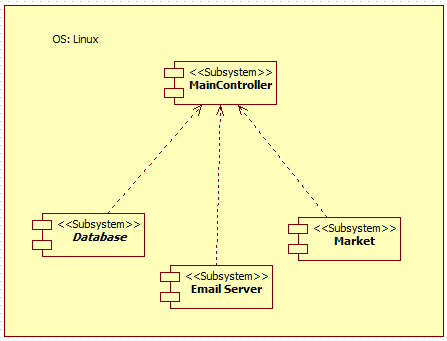


Figure #2 UML Deployment diagram for the three subsystems. All three subsystems are housed on a linux server

## Design Patterns

The Strategic Marketing Simulator is being designed using the mediator design pattern and the private class data design pattern. This was chosen because we want to follow best practices for object oriented programming to make sure that the classes do not expose their instance variables to manipulation by ensuring that all data transactions are being handled by a single intermediate (mainController subsystem) and the classes are all have private access.

# System Validation

Verification was accomplished using PHPunit to validate assertions for the model’s query functions and the system’s controller functions and objects. Unit testing was carried out to test the helper functions and general model transactions. System testing was used to test the fundamental functionality of the system to verify that all of its parts are fully integrated and tested.

Unit Testing

|  |  |
| --- | --- |
| Test ID: | databaseconstructInvalidParTest1 |
| Description: | Testing the creation of a database object with invalid parameter |
| Test Steps | Step 1: Create database object $m with 'EUR' as a parameter |
| Step 2: assertInstanceOf(database::class, $m) |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | databaseconstrucValidParTest2 |
| Description: |  |
| Test Steps | Step 1: Create database object $m with no parameters |
| Step 2: assertInstanceOf(database::class, $m) |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | databasesearchForStudentsFoundTest3 |
| Description: | Test search for students funciton of database with valid query |
| Test Steps | Step 1: Create database object $db |
| Step 2: assertEquals(count($db->searchForStudents("jcarm012@fiu.edu")), 10) |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | databasesearchForStudentsNotFoundTest4 |
| Description: | Test search for students funciton of database with invalid query |
| Test Steps | Step 1: Create database object $db |
| Step 2: assertEquals(count($db->searchForStudents("beaver")), 0) where "beaver is not in the database |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | databasegetStudentFoundTest5 |
| Description: | Test database for getStudent function with valid query to return array of student. |
| Test Steps | Step 1: Create database object $db |
| Step 2: assertEquals(count($db->getStudent("2654698")), 10) where "2654698" is found in the database. |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | databasegetStudentNotFoundTest6 |
| Description: | Test database for getStudent function with invalid query to return array of student. |
| Test Steps | Step 1: Create database object $db |
| Step 2: assertEquals(count($db->getStudent("stickle")), 0) where "stickly" is not found in the database |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | databaseaddStudentTest7 |
| Description: | Test database for AddStudent function with valid query to insert student record into the database. |
| Test Steps | Step 1: Create database object $db |
| Step 2: assertEquals($db->addStudent(1651248, "ricky", "roberts", "Ricke@roberts.com", 0, 1253481, "where am I", "everywhere", -5), "pass") |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | databasegenPassTest8 |
| Description: | Test database function genPass with valid input |
| Test Steps | Step 1: create input string $pwd = "somePass1234" |
| Step 2: create input String $email = "some@email.com"; |
| Step 3: create database object $db |
| Step 4: create database object $db2 |
| Step 5: assertEquals($db->genPass($pwd, $email),$db2->genPass($pwd, $email)) |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | databaseupdateStudentPasswordTest9 |
| Description: | Test database function updateStudentPassword with valid input |
| Test Steps | Step 1: Create input string $email = "jcarm012@fiu.edu" |
| Step 2: Create database object $db |
| Step 3: Create input $pwd = genPass($email, "Jeff\_1230") |
| Step 4: ($db->updateStudentPassword($email, $pwd), "pass") |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | databaseupdateStudentPasswordInValidTest10 |
| Description: | Test database function updateStudentPassword with invalid input |
| Test Steps | Step 1: Create input string $email = "jcarm012@fiu.edu" |
| Step 2: Create database object $db |
| Step 3: assertEquals($db->updateStudentPassword($email, NULL), "fail") |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | databaseaddGameValidTest11 |
| Description: | Test database function genPass with valid input |
| Test Steps | Step 1: Create input $db = new database(); |
| Step 2: Create input $semester = "TestSemester"; |
| Step 3: Create input $courseID = "1234567Test"; |
| Step 4: Create input $schedule = "3:45-5 Test"; |
| Step 5: Create input $section = "abcdSectionTest"; |
| Step 6: Create input $isActive = -5; |
| Step 7: Create input $courseNumber = "TestCourseNumber"; |
| Step 8: assertEquals($db->addGame($semester,$courseID,$section,$schedule,$isActive,$courseNumber), "pass") |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | databaseaddGameinValidTest12 |
| Description: | Test database function genPass with invalid input |
| Test Steps | Step 1: Create input $db = new database(); |
| Step 2: Create input $semester = "TestSemester"; |
| Step 3: Create input $courseID = "1234567Test"; |
| Step 4: Create input $schedule = "3:45-5 Test"; |
| Step 5: Create input $section = "abcdSectionTest"; |
| Step 6: Create input $isActive = -5; |
| Step 7: Create input $courseNumber = "TestCourseNumber"; |
| Step 8: assertEquals($db->addGame($semester,$courseID,$section,$schedule,$NULL,$courseNumber), "fail") |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | databaseupdateAdminPasswordValidTest13 |
| Description: | Test database function updateAdminPassword with a valid input |
| Test Steps | Step 1: Create input string $email = "test@test.com" |
| Step 2: Create input string $password = "NewTestPassword"; |
| Step 3: Create database object $db |
| Step 4: assertEquals ($db->updateAdminPassword($email,$password), 'pass') |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | databaseupdateAdminPasswordinValidTest14 |
| Description: | Test database function updateAdminPassword with a invalid input |
| Test Steps | Step 1: Create input string $email = "test@test.com" |
| Step 2: Create input string $password = "NewTestPassword"; |
| Step 3: Create database object $db |
| Step 4: assertEquals($db->updateAdminPassword($email,NULL), 'fail'); |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | databasesetStudentActiveValidTest15 |
| Description: | Test database function studentActive with a valid input |
| Test Steps | Step 1: Create input string $email = "Ricke@roberts.com" |
| Step 2: Create input $isActive = rand(2,10) |
| Step 3: Create database object $db |
| Step 4: assertEquals($db->setStudentActive($email,$isActive), 'pass') |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | databasesetStudentActiveinValidTest16 |
| Description: | Test database function studentActive with a invalid input |
| Test Steps | Step 1: Create input string $email = "Ricke@roberts.com" |
| Step 2: Create input $isActive = "not an int" |
| Step 3: Create database object $db |
| Step 4: assertEquals($db->setStudentActive($email,$isActive), 'fail') |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | databaseAdminActiveValidTest17 |
| Description: | Test database function AdminAcvite with a valid input |
| Test Steps | Step 1: Create input string $email = "test@test.com" |
| Step 2: Create input $isActive = rand(1,10) |
| Step 3: Create database object $db |
| Step 4: assertEquals($db->setStudentActive($email,$isActive), 'pass') |
| Tester: | Jeffrey Carman |
|  |  |
| Test ID: | databaseAdminActiveinValidTest18 |
| Description: | Test database function AdminAcvite with a invalid input |
| Test Steps | Step 1: Create input string $email = "test@test.com" |
| Step 2: Create input $isActive = "not an int" |
| Step 3: Create database object $db |
| Step 4: assertEquals($db->setStudentActive($email,$isActive), 'fail') |
| Tester: | Jeffrey Carman |
|  |  |
| Test ID: | databaseupdateStudentHotelValidTest19 |
| Description: | Test database function updateStudentHotel with valid input |
| Test Steps | Step 1: Create input string $email = "jcarm012@fiu.edu" |
| Step 2: Create input string $hotel = 52 |
| Step 3: Create database object $db |
| Step 4: assertEquals($db->updateStudentHotel($student, $hotel false) |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | databaseupdateStudentHotelInValidTest20 |
| Description: | Test database function updateStudentHotel with invalid input |
| Test Steps | Step 1: Create input string $email = "jcarm012@fiu.edu" |
| Step 2: Create input string $hotel = 52 |
| Step 3: Create database object $db |
| Step 4: assertEquals($db->updateStudentHotel($student, NULL), false) |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | databasegetGameValidTest21 |
| Description: | Test database function getGame with a valid input |
| Test Steps | Step 1: Create input $game = 1 |
| Step 2: Create database object $db |
| Step 3: assertEquals(count($db->getGame($game)), 7) |
| Tester: | Jeffrey Carman |
|  |  |
| Test ID: | databasegetGameinValidTest22 |
| Description: | Test database function getGame with a invalid input |
| Test Steps | Step 1: Create input $game = -1 |
| Step 2: Create database object $db |
| Step 3: assertEquals(count($db->getGame($game)), 0) |
| Tester: | Jeffrey Carman |
|  |  |
| Test ID: | databasegetGroupValidTest23 |
| Description: | Test database function getGroup with valid input |
| Test Steps | Step 1: Create input $group = 46 |
| Step 2: Create database object $db |
| Step 3: assertEquals(count($db->getGroup($group)), 10) |
| Tester: | Jeffrey Carman |
|  |  |
| Test ID: | databasegetGroupinValidTest24 |
| Description: | Test database function getGroup with invalid input |
| Test Steps | Step 1: Create input $group = -1 |
| Step 2: Create database object $db |
| Step 3: assertEquals(count($db->getGroup($group)), 0) |
| Tester: | Jeffrey Carman |
|  |  |
| Test ID: | databasegetGameByCourseValidTest25 |
| Description: | Test database function getGamebyCourse with valid input |
| Test Steps | Step 1: Create input $course = 75648 |
| Step 2: Create database object $db |
| Step 3: assertEquals(count($db->getGameByCourseNumber($course)), 7) |
| Tester: | Jeffrey Carman |
|  |  |
| Test ID: | databasegetGameByCourseinValidTest26 |
| Description: | Test database function getGamebyCourse with invalid input |
| Test Steps | Step 1: Create input $course = 75699 |
| Step 2: Create database object $db |
| Step 3: assertEquals(count($db->getGameByCourseNumber($course)), 0) |
| Tester: | Jeffrey Carman |
|  |  |
| Test ID: | databasegetLocationValidTest27 |
| Description: | Test database function getLocation with valid input |
| Test Steps | Step 1: Create input $loc = 12345 |
| Step 2: Create database object $db |
| Step 3: assertEquals(count($db->getLocation($loc)), 2) |
| Tester: | Jeffrey Carman |
|  |  |
| Test ID: | databasegetLocationinValidTest28 |
| Description: | Test database function getLocation with invalid input |
| Test Steps | Step 1: Create input $loc = 123456 |
| Step 2: Create database object $db |
| Step 3: assertEquals(count($db->getLocation($loc)), 0) |
| Tester: | Jeffrey Carman |
|  |  |
| Test ID: | databasegetAllGamesTest29 |
| Description: | Testing database function getallgames with valid input |
| Test Steps | Step 1: create database object $db |
| Step 2: assertGreaterThan(0,count($db->getadvertising()), 0) |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | databasegetAdvertisingValidTest30 |
| Description: | Testing database function newHotel with valid input |
| Test Steps | Step 1: create database object $db |
| Step 2: assertGreaterThan(0,count($db->newHotel('abc Test', 12345, "economy", 3, 50000, 35000, 50.00, -1))) |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | databasenewHotelValidTest31 |
| Description: | Test database function searchForhotels with valid input |
| Test Steps | Step 1: Create input $name = "ABC Marketing" |
| Step 2: Create database object $db |
| Step 3: assertGreaterThan(0, count($db->searchHotel($name))) |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | databaseSearchHotelsValidTest32 |
| Description: | Test database function searchForhotels with invalid input |
| Test Steps | Step 1: Create input $name = "zzz" |
| Step 2: Create database object $db |
| Step 3: assertLessrThan(0, count($db->searchHotel($name))) |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | databaseSearchHotelsInValidTest33 |
| Description: | Test database function getGroupsforGame with valid input |
| Test Steps | Step 1: Create input $id = 3 |
| Step 2: Create database object $db |
| Step 3: assertGreaterThan(0, count($db->getGroupsforGame($id))) |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | databasegetGroupsForGameValidTest34 |
| Description: | Test database function getGroupsforGame with invalid input |
| Test Steps | Step 1: Create input $id = -2 |
| Step 2: Create database object $db |
| Step 3: assertLessThan(0, count($db->getGroupsforGame($id))) |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | databasegetGroupsForGameinValidTest35 |
| Description: | Test database function getGroupsforGame with invalid input |
| Test Steps | Step 1: Create input $id = -2 |
| Step 2: Create database object $db |
| Step 3: assertLessThan(0, count($db->getGroupsforGame($id))) |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | AdminControllerconstructInvalidParTest36 |
| Description: | Test AdminController object with invalid parameter |
| Test Steps | Step 1: Create AdminController Object with 'aaa' as a parameter |
| Step 2: (AdminController::class, $m) |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | AdminControllerconstrucValidParTest37 |
| Description: | Test AdminController object with valid parameter |
| Test Steps | Step 1: Create AdminController Object with no parameter |
| Step 2: (AdminController::class, $m) |
| Tester: | Jeffrey Carman |
| Result: | Pass |

|  |  |
| --- | --- |
| Test ID: | getMarketShareTable63 |
| Description: | Test database for function updates a students secret question and returns pass if it succeeds |
| Test Steps | Step: 1$obj = new database(); |
| Step 2: $game\_period = "GAME\_3\_P\_1"; |
| Step 3:return $this->assertGreaterthan(0, count($obj->getMarketShareTable($game\_period))); |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
|  |  |
|  |  |
| Test ID: | updateStudentQuestionValid64 |
| Description: | Test database for function updates a students secret question and returns pass if it succeeds |
| Test Steps | Step 1: $obj = new database(); |
| Step 2: $email = "jcarm012@fiu.edu"; |
| Step 3: $secretQuestion = "where are you?"; |
| Step 4: $id = "265469154"; |
| Step 5: return $this->assertEquals('pass', $obj->updateStudentQuestion($email,$secretQuestion,$id)); |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
|  |  |
| Test ID: | getMarketShareTable65 |
| Description: | Test database function getMaketShare which is a function that returns the marketing share for all groups for a given game and period |
| Test Steps | Step 1 : declare $game\_period = "GAME\_3\_P\_1"; |
| Step 2: create database object $obj |
| Step 3:assertGreaterthan(0, count($obj->getMarketShareTable($game\_period))); |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | updateMarketShare66 |
| Description: | Test database function updateMaketShare which is a function that updates the marketing share table for a group for a given game and period |
| Test Steps | Step 1: $obj = new database(); |
| Step 2 : $gamePeriod = "GAME\_3\_P\_2"; |
| Step 3: $groups = $obj->getGameGroupsId(3); |
| Step 4: $groupcount = 6; |
| Step 6: $rooms = 2000; |
| Step 7: $roomsSold = 1600; |
| Step 8: $byGroup = array(); |
| Step 9: $counter = 0; |
| Step 10 : for($i = 1; $i < $groupcount; $i++)  {  $temp = rand(200,320);  array\_push($byGroup, $temp );  $counter = $counter + $temp;  } |
| Step 11: $period = 2; |
| $this->assertGreaterthan(0, count($obj->updateMarketShare($gamePeriod, $groupcount, $rooms, $roomsSold, $byGroup, $groups, $period))); |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | createMarketShare67 |
| Description: | Test database function updateMaketShare which is a function thatcreates the marketing share table given game and period |
| Test Steps | Step1: $obj = new database(); |
| Step 2 :$gamePeriod = "GAME\_3\_P\_16"; |
| Step 3: $groups = $obj->getGameGroupsId(3); |
| Step 4: $groupcount = 6; |
| Step 5: $period = 16; |
| Step 6: return $this->assertEquals("Table created", $obj->createMarketShare($gamePeriod,$groupcount, $groups, $period)); |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | getGameGroupCount68 |
| Description: | Test database function that returns the number of groups participating in a game |
| Test Steps | Step 1: $obj = new database(); |
| Step 2: $game = 3; |
| Step 3: return $this->assertGreaterThan(0, $obj->getGameGroupCount($game)); |
|  |
|  |
|  |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | getGameGroupCount68 |
| Description: | Test database function that returns the number of groups participating in a game |
| Test Steps | Step 1: $obj = new database(); |
|  | Step 2: $game = 3; |
|  | Step 3: return $this->assertGreaterThan(0, $obj->getGameGroupCount($game)); |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | getGameGroupsIdt69 |
| Description: | Test database function that returns the ids of all groups participating in a game. |
| Test Steps | Step 1: $obj = new database(); |
| Step 2: $game = 3; |
| Step 3: return $this->assertGreaterThan(0, $obj->getGameGroupsId($game)); |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
|  |  |
| Test ID: | isMarketShare70 |
| Description: | Test database function that returns true if a marketshare table exists |
| Test Steps | Step 1: $marketshare = "GAME\_3\_P\_1\_MarketShare"; |
| Step 2: $obj = new database(); |
| Step 3: return $this->assertGreaterThan(0, $obj->isMarketShare($marketshare)); |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | getRevenue71 |
| Description: | Test database function that returns the revenue of a given group |
| Test Steps | Step 1: $obj = new database(); |
| Step 2: $hotel = 110; |
| Step 3: return $this->assertGreaterThan(0, $obj->getRevenue($hotel)); |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | getLeaderboardTabl72 |
| Description: | Test database function that returns the leaderboard table for a given period |
| Test Steps | Step 1: $obj = new database(); |
| Step 2: $game = 3; |
| Step 3: assertGreaterThan(0, $obj->getLeaderboardTable($game)); |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | updatePurpos73 |
| Description: | Test database function that updates the purpose of travel for a given group |
| Test Steps | Step 1: $obj = new database(); |
| Step 2: $purpose = 'leisure'; |
| Step 3: $hotel = 110; |
| Step 4: assertEquals(true, $obj->updatePurpose($purpose,$hotel)); |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | getPeriodResearch74 |
| Description: | Test database function that gets the research assignments for a given group for a given period of a given game |
| Test Steps | Step 1: $obj = new database(); |
| Step 2: $game = 'GAME\_3\_P\_1'; |
| Step 3: $hotel = 110; |
| Step 4: assertGreaterThan(0, $obj->getPeriodResearch($game,$hotel)); |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | gethotelIdbyNam75 |
| Description: | Test database function that accepts the name of a group and returns its id. |
| Test Steps | Step 1: $obj = new database(); |
| Step 2: $name = "XYZ Marketing"; |
| Step 3: assertEquals(1, count($obj->gethotelIdbyName($name))); |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | incrementGamePeriodNum76 |
| Description: | Test database function that increments the period number in the game table |
| Test Steps | Step 1: $obj = new database(); |
| Step 2: $game = 1; |
| Step 3: $period = 16; |
| Step 4: assertEquals(true, $obj->incrementGamePeriodNum($game, $period)); |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | isMarketShareForPeriodValid77 |
| Description: | Test database function that returns true if marketshare was updated for a group for a period |
| Test Steps | Step 1: $obj = new database(); |
| Step 2: $marketshare = "GAME\_3\_P\_1\_MarketShare"; |
| Step 3: $period = 1; |
| Step 4: assertEquals(1, $obj->isMarketShareForPeriod($marketshare, $periodNum)); |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | isMarketShareForPeriodValid77 |
| Description: | Test database function that returns true if marketshare was updated for a group for a period |
| Test Steps | Step 1: $obj = new database(); |
| Step 2: $marketshare = "GAME\_3\_P\_1\_MarketShare"; |
| Step 3: $periodNum = 1; |
| Step 4: assertEquals(1, $obj->isMarketShareForPeriod($marketshare, $periodNum)); |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | isMarketShareForPeriodInValid77 |
| Description: | Test database with invalid input for function that returns true if marketshare was updated for a group for a period |
| Test Steps | Step 1: $obj = new database(); |
| Step 2: $marketshare = "GAME\_3\_P\_1\_MarketShare"; |
| Step 3: $periodNum = "carrot"; |
| Step 4: assertEquals(1, $obj->isMarketShareForPeriod($marketshare, $periodNum)); |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | addComments78 |
| Description: | Test database for function that updates the student\_comments database and when they commit their decision |
| Test Steps | Step 1: $game = -1; |
| Step 2: $period = -1; |
| Step 3: $group = 110; |
| Step 4: $comments = "these are my comments for the period"; |
| Step 5: return $this->assertEquals(true, $obj->addComments($game, $period, $group, $comments)); |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | updateStudentQuestionValid79 |
| Description: | Test database for function updates a students secret question and returns pass if it succeeds |
| Test Steps | Step 1: $obj = new database(); |
| Step 2: $email = "jcarm012@fiu.edu"; |
| Step 3: $secretQuestion = "where are you?"; |
| Step 4: $id = "2654698"; |
| Step 5: return $this->assertEquals('pass', $obj->updateStudentQuestion($email,$secretQuestion,$id)); |
| Tester: | Jeffrey Carman |

|  |  |
| --- | --- |
| Test ID: | insertdecisions80 |
| Description: | Test database for function that inserts a groups decisions. |
| Test Steps | Step 1: $obj = new database(); |
| Step 2: $game = 3; |
| Step 3: $roomRate = 85.00; |
| Step 4: $hotel = "110"; |
| Step 5: $arrDecisions = array() |
| Step6: for (int $i = 0, i<4; $i++) { array\_push($arrDecisions, $i+rand(0,2);} // advertising decisions |
| Step 6: $personArr = array(); |
| Step 7: for (int $i = 0, i<2; $i++) { array\_push($arrDecisions, $i;} // Personnel array decisions |
| Step 8: $research = "ABC Marketing"; |
| Step 9: $this->AssertEquals('pass", $obj->insertdecisions($gameNum ,$periodNum, $roomRate, $hotel, $arrDecisions, $personArr, $OTA, $research); |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | getStudentDecisions81 |
| Description: | test database function to get a groups decisions for a game/period |
| Test Steps | Step 1: $obj = new database(); |
| Step 2: $game = 3; |
| Step 3: $period = 1 |
| Step 4: $hotel = "110"; |
| Step 5: $this->AssertGreaterThan(0, count($obj->getStudentDecisions($game, $period, $hotel)); |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | getMarketShare82 |
| Description: | test database function that returns the marketshare table for a game/period |
| Test Steps | Step 1: $obj = new database(); |
| Step 2: $game = 3; |
| Step 3: $period = 1 |
| Step 4: $this->AssertGreaterThan(0, count($obj->getMarketShare($game, $period)); |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | getMarketShareforGroup83 |
| Description: | test database function that returns the market share records for a a hotel for a particular game/period |
| Test Steps | Step 1: $obj = new database(); |
| Step 2: $game = 3; |
| Step 3: $period = 1 |
| Step 4: $hotel = 110; |
| Step 5: $this->AssertGreaterThan(0, count($obj->getMarketShareforGroup($game, $period, $hotel)); |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | gethotelname84 |
| Description: | test database function that returns the name of a hotel |
| Test Steps | Step 1: $obj = new database(); |
| Step 4: $hotel = 110; |
| Step 9: $this->AssertEquals('XYZ Marketing'$obj->gethotelname($hotel); |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
|  |  |
| Test ID: | getAdvertFromStrats85 |
| Description: | test database function that returns the advertising decisions for a group for a particular game and period |
| Test Steps | Step 1: $obj = new database(); |
| Step 2: $game = 3; |
| Step 3: $period = 1 |
| Step 4: $hotel = 110; |
| Step 5: $this->AssertGreaterThan(0, count($obj->getAdvertFromStrats($game, $period, $hotel)); |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | getDecisionsImpact86 |
| Description: | test database function that returns the impact for a group for a period of a game |
| Test Steps | Step 1: $obj = new database(); |
| Step 2: $game = 3; |
| Step 3: $period = 1 |
| Step 4: $hotel = 110; |
| Step 5: $this->AssertGreaterThan(1, count($obj->getDecisionsImpact($game,$period, $group)); |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | getNewsEffects87 |
| Description: | test database function that returns the impact for a group for a period of a game |
| Test Steps | Step 1: $obj = new database(); |
| Step 2: $game = 3; |
| Step 3: $period = 1 |
| Step 4: $this->AssertGreaterThan(1, count($obj->getNewsEffects($game, $period)); |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | getGroupsIDforGame88 |
| Description: | test database function that returns the id's for groups related to a game |
| Test Steps | Step 1: $obj = new database(); |
| Step 2: $game = 3; |
| Step 3: $this->AssertEquals(6, count($obj->getGroupsIDforGame($game)); |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | getTotalRooms89 |
| Description: | test database function that returns the total number of rooms available in the game |
| Test Steps | Step 1: $obj = new database(); |
| Step 2: $this->AssertEquals(20000, count($obj->getTotalRooms()); |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
|  |  |
|  |  |
| Test ID: | getGroupComments90 |
| Description: | test database function that returns the comments for a group for a certain game and period |
| Test Steps | Step 1: $obj = new database(); |
| Step 2: $game = 3; |
| Step3: $period = 2; |
| Step 4: $group = 110; |
| Step 5: $this->AssertEquals(1, count($obj->getGroupComments($game, $period, $group)); |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | insertBeta91 |
| Description: | test database function that inserts the marketshare for a group into the market share table |
| Test Steps | Step 1: $obj = new database(); |
| Step 2: $game = 3; |
| Step3: $period = 2; |
| Step 4: $BETA\_J = Array; |
| Step 5: $temp = array("group"=> 110, "BETA\_j"=>1500) |
| Step 6: array\_push($BETA\_J, $temp); |
| Step 5: $this->AssertEquals('pass', $obj->insertBeta($game, $period, $BETA\_J); |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
|  |  |
| Test ID: | insertrevenue92 |
| Description: | test database function that inserts the marketshare for a group into the market share table |
| Test Steps | Step 1: $obj = new database(); |
| Step 2: $game = 3; |
| Step3: $period = 2; |
| Step 4: $delta = Array; |
| Step 5: $temp = array("group"=> 110, "DELTA"=>23,154) |
| Step 6: array\_push($delta, $temp); |
| Step 5: $this->AssertEquals('pass', $obj->insertrevenue($game, $period, $delta); |
| Tester: | Jeffrey Carman |
| Result: | Pass |

System Testing

|  |  |
| --- | --- |
| Test ID: | AdminControllerCreatePageTest38 |
| Description: | Test AdminController function createPage |
| Test Steps | Step 1: Create AdminController object $a |
| Step 2: Create input array $value = $a->viewUsersForGame(3); |
| Step 3: Create string $return = "<form action='' method='post'>  <div class='bs-example'>". "<div class='panel panel-default'>".  "<div class='panel-body' align='center'>". "<h2>User Accounts Management</h2>".  "<a href='http://marketsim-dev.cis.fiu.edu/admin/ManagePage.php?addAdminUser=Add+Admistrative+User' class='btn btn-primary'>Add Admistrative User</a>".  "<a href='http://marketsim-dev.cis.fiu.edu/admin/ManagePage.php?addBotUser=Add+Bot+User' class='btn btn-primary'>Add Bot User</a>".  //"<a href='http://marketsim-dev.cis.fiu.edu/admin/ManagePage.php?addBotUser=Add+Bot+User&textbox\_bot\_id=&textbox\_bot\_fname=&textbox\_bot\_lname=&textbox\_bot\_email=' class='btn btn-primary'>Add Bot User</a>".  "<a href='http://marketsim-dev.cis.fiu.edu/admin/ManagePage.php?viewAllGames=View+All+Games' class='btn btn-primary'>View All Games</a>".  "<a href='http://marketsim-dev.cis.fiu.edu/admin/ManagePage.php?viewAllUsers=View+All+Users' class='btn btn-primary'>View All Users</a>".  "</div>". "<div class='' align='center'>".  "<div class='panel-footer clearfix' style='width: 1060px' >".  $value. "</div>". "</div>". "</div>". "</div>". "</form>" |
| Step 4: assertEquals($db->updateStudentHotel($student, NULL), false) |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | AdminControlleraddAdminPageTest39 |
| Description: | Test database function addAdminPage with valid input |
| Test Steps | Step 1: Create AdminController object $a |
| Step 2: Create string $result = "<div class='' align='left'>".  "<h3>First Name</h3>". "<input type = 'Text' name = 'textbox\_fname' pattern='.{1,}' placeholder='Javier' title='Name of the administrator account'>". "<h3>Last Name</h3>". "<input type = 'Text' name = 'textbox\_lname' placeholder = 'Andrial' title='Last name of Admin account. not required' >". "<h3>Email</h3>". "<input type = 'email' name = 'textbox\_email' pattern='[a-z0-9.\_%+-]+@[a-z0-9.-]+\.[a-z]{2,4}$' placeholder='admin@marketsim.edu'>".  "<h3>Password</h3>". "<input type = 'password' name = 'textbox\_password1' pattern='.{8,}' placeholder='password' title='8 character passowrd minimum'>".  "<h3>Re-enter Password</h3>". "<input type = 'password' name = 'textbox\_password2' pattern='.{8,}' placeholder='password' title='those same 8 characters you just forgot go here'>".  "<h3>Secret Question</h3>". "<input type = 'Text' name = 'textbox\_secretQuestion' pattern='.{1,}' placeholder='Whats my favorite website?' title='Used to recover your account incase you forget those 8 characters again'>".  "<h3>Secret Answer</h3>". "<input type = 'Text' name = 'textbox\_secretAnswer' pattern='.{1,}' placeholder='marketsim-dev.cis.fiu.edu'>". "<br />". "<br />"."<br />".  //"<a href='http://marketsim-dev.cis.fiu.edu/admin%20shit/adminUsers.php?textbox\_fname=&textbox\_lname=&textbox\_email=&textbox\_password1=&textbox\_password2=&textbox\_secretQuestion=&textbox\_secretAnswer=&button\_addAdmin=Create+Admistrative+Account' class='btn btn-primary'>View All Users</a>".  "<input name='button\_addAdmin' type = 'submit' value = 'Create Admistrative Account' class='btn btn-primary'/>".  "</div>" |
| Step 3: assertEquals($a->addAdminPage(), $result) |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | AdminControlleraddBotPageTest40 |
| Description: | Test database function addBotPage with valid input |
| Test Steps | Step 1: Create AdminController object $a |
| Step 2:"<div class='' align='left'>".  "<h3>ID</h3>".  "<input type = 'Text' name = 'textbox\_bot\_id' pattern='^[+-]?\d' placeholder='-1' title='a unique number to identify this user'>".  "<h3>First Name</h3>".  "<input type = 'Text' name = 'textbox\_bot\_fname' placeholder='Javier' title='first name of the bot'>".  "<h3>Last Name</h3>".  "<input type = 'Text' name = 'textbox\_bot\_lname' placeholder='Andrial' title='lastname, hope you're not running out of creativity! email is coming next'>".  "<h3>Email</h3>".  "<input type = 'email' name = 'textbox\_bot\_email' pattern='[a-z0-9.\_%+-]+@[a-z0-9.-]+\.[a-z]{2,4}$' placeholder='bot@marketsim.edu' title='doesnt have to be real'>".  "<br />".  "<br />"."<br />".  "<input type = 'submit' value = 'Create Bot Account' class='btn btn-primary' title='only works if pressed with both eyes closed'/>".  "</div>" |
| Step 3: assertEquals($a->addAdminPage(), $result) |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | AdminControlleraddBotHotelPageTest41 |
| Description: | Test database function addBotHotelPage with valid input |
| Test Steps | Step 1: Create AdminController object $a |
|  | Step 2: create database obj $db |
|  | Step 3 Create new hotel, $db->newHotel("bot Hotel Test", 12345, "economy", 1, 50000.00, 50000, 50, -1) |
|  | Step 4: assertGreaterThan(0,count($db->newHotel("bot Hotel Test", 12345, "economy", 1, 50000.00, 50000, 50, -1))) |
|  | Step 5: Create string $result = "<div class='' align='left'>".  "<h2>Create the Bot's Hotel</h2>". "<h3>Which Game to Assign bot to?</h3>".  "<input type = 'Text' name = 'textbox\_Hotel\_game\_id' require>". "<h3>Name of Hotel</h3>". "<input type = 'Text' name = 'textbox\_Hotel\_name' pattern='^[+-]\d' placeholder='FIU inn'>". "<h3>Location</h3>". "<input type = 'Text' name = 'textbox\_Hotel\_fname' placeholder='Florida'>".  "<h3>Hotel Type</h3>".  "<label class='radio-inline' >".  "<input type='radio' name='optradio' value ='economic'>Economic".  "</label>".  "<label class='radio-inline' for 'mid'>".  "<input type='radio' name='optradio' value ='midrange'>Midrange".  "</label>".  "<label class='radio-inline' for 'lux'>".  "<input type='radio' name='optradio' value ='luxury'>Luxury".  "</label>".  "<br />".  "<br />"."<br />".  "<input type = 'submit' value = 'Create Bot Account' class='btn btn-primary'/>".  "</div>" |
|  | Step 3: assertEquals($a->addBotHotelPage(-4,"bot4", "bot4","bot4@marketsim.com"), $result) |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | AdminControllercreateGamePageTest42 |
| Description: | Test database functioncreateGamePage with valid input |
| Test Steps | Step 1: Create AdminController object $a |
| Step 2:"<div class='' align='left'>".  "<h2>Game Creation Page</h2>".    "<h3>Course Number</h3>".  "<input type = 'text' name = 'textbox\_courseNumber' pattern='[0-9]{5}' placeholder = '88529' title='5 digit number'>".    "<h3>Course ID</h3>".  "<input type = 'text' name = 'textbox\_courseID' placeholder = 'MAR2015'>".    "<h3>Section</h3>".  "<input type = 'text' name = 'textbox\_section' placeholder='U02'>".    "<h3>Semester of Game</h3>".  "<input type = 'text' name = 'textbox\_semester' placeholder='FALL 2016'>".    "<h3>Course Meeting time</h3>".  "<input type = 'text' name = 'textbox\_schedule' placeholder='mwf 12pm-1:30pm'>".  "<br />".  "<br />"."<br />".  "<input name='button\_addGame' type = 'submit' value = 'Create Game' class='btn btn-primary'/>".  "</div>" |
| Step 3: assertEquals($a->createGamePage(), $result) |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
|  |  |
|  |  |
|  |  |
| Test ID: | getadvertisingNameAndPriceValidTest43 |
| Description: | Test database function getAdvertisingName with valid input |
| Test Steps | Step 1 : Create $id = 3 |
| Step 2: create database object $db |
| Step 3: assertGreaterThan(0, $db->getAdvertisingNameAndPrice($id)); |
| Tester: | Jeffrey Carman |
| Result: | Pass |

|  |  |
| --- | --- |
| Test ID: | changeRecoveryPage60 |
| Description: | Test AccountController function that returns a template to change password page |
| Test Steps | Step 1 :$obj = new AccountController; |
| Step 2 :$result = "<h3>Change My Recovery Question/Answer</h3><br /><br />".  "<div class='' style='margin:auto; width:30%;'>".  "<div class='' align='left'>".  "<h5>Current Password</h5>".  "<input type = 'password' name = 'textbox\_CurrentPass2' required pattern='.{8,}' >".  "<h5>New Recovery Question</h5>".  "<input type = 'text' name = 'textbox\_NewQestion' required >".  "<h5>Recovery Answer</h5>".  "<input type = 'text' name = 'textbox\_NewAnswer' required >".  "</div>".  "</div>".  "<br />".  "<br />".  "<br />".  "<input type = 'submit' name='button\_ChangedRecovery' class='btn btn-primary' value = 'Change Recovery Quest/Ans' />"; |
| Step 3:assertEquals($result, $obj->changeRecoveryPage()); |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
|  |  |
|  |  |
|  |  |
| Test ID: | changePasswordPage61 |
| Description: | Test AccountController function that returns a template to change password page |
| Test Steps | Step 1 :$obj = new AccountController; |
| Step 2 :$result =  "<h3>Change My Password</h3>".  "<br /><br />".  "<div class='' style='margin:auto; width:30%;'>".  "<div class='' align='left'>".  "<h4>Current Password</h4>".  "<input type = 'password' name = 'textbox\_CurrentPass1' required pattern='.{8,}' >".  "<h4>New Password</h4>".  "<input type = 'password' name = 'textbox\_NewPass' required pattern='.{8,}' >".  "</div>".  "</div>".  "<br />".  "<br />"."<br />".  //"</div>".  "<input name='button\_ChangedPassword' type = 'submit' class='btn btn-primary' value = 'Change Password'/>"; |
| Step 3: $this->assertEquals($result, $obj->changePasswordPage()); |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
|  |  |
| Test ID: | CreatePage62 |
| Description: | Test AccountController function that returns a template to account page |
| Test Steps | Step 1 :$obj = new AccountController; |
| Step 2 :$value = "someValue"; |
| Step 3: $result = "<form action='' method='post'>".  "<div class='' align='center'>".  "<div class='panel-footer clearfix' style='margin:auto; width:30%;' >".    $value.  "</div>".  "</div>".  "</form>"; |
| Step 4:>assertEquals($result, $obj->CreatePage($value)); |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
|  |  |
|  |  |
| Test ID: | getMarketShareTable63 |
| Description: | Test database for function updates a students secret question and returns pass if it succeeds |
| Test Steps | Step: 1$obj = new database(); |
| Step 2: $game\_period = "GAME\_3\_P\_1"; |
| Step 3:return $this->assertGreaterthan(0, count($obj->getMarketShareTable($game\_period))); |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
|  |  |
|  |  |
| Test ID: | updateStudentQuestionValid64 |
| Description: | Test database for function updates a students secret question and returns pass if it succeeds |
| Test Steps | Step 1: $obj = new database(); |
| Step 2: $email = "jcarm012@fiu.edu"; |
| Step 3: $secretQuestion = "where are you?"; |
| Step 4: $id = "265469154"; |
| Step 5: return $this->assertEquals('pass', $obj->updateStudentQuestion($email,$secretQuestion,$id)); |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
|  |  |
| Test ID: | getMarketShareTable65 |
| Description: | Test database function getMaketShare which is a function that returns the marketing share for all groups for a given game and period |
| Test Steps | Step 1 : declare $game\_period = "GAME\_3\_P\_1"; |
| Step 2: create database object $obj |
| Step 3:assertGreaterthan(0, count($obj->getMarketShareTable($game\_period))); |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | updateMarketShare66 |
| Description: | Test database function updateMaketShare which is a function that updates the marketing share table for a group for a given game and period |
| Test Steps | Step 1: $obj = new database(); |
| Step 2 : $gamePeriod = "GAME\_3\_P\_2"; |
| Step 3: $groups = $obj->getGameGroupsId(3); |
| Step 4: $groupcount = 6; |
| Step 6: $rooms = 2000; |
| Step 7: $roomsSold = 1600; |
| Step 8: $byGroup = array(); |
| Step 9: $counter = 0; |
| Step 10 : for($i = 1; $i < $groupcount; $i++)  {  $temp = rand(200,320);  array\_push($byGroup, $temp );  $counter = $counter + $temp;  } |
| Step 11: $period = 2; |
| $this->assertGreaterthan(0, count($obj->updateMarketShare($gamePeriod, $groupcount, $rooms, $roomsSold, $byGroup, $groups, $period))); |
| Tester: | Jeffrey Carman |
| Result: | Pass |

|  |  |
| --- | --- |
| Test ID: | endIfGame93 |
| Description: | test GameController function that calculates the revenue and market share for a game period |
| Test Steps | Step 1: $obj = new gameController(); |
|  | Step 2 $db = new database(); |
|  | Step 2: $game = 3; |
|  | Step3: $period = 2; |
|  | Step 4: $group = $db>getGroupsForGame(3) |
|  | Step 5: $obj->endPeriod ($game, $period, $group) |
|  | Step 6: $this->AssertGreaterThan(0, count( $obj->getMaketShare($game, $period)); |
|  | Step 7: $this->AssertGreaterThan(0, count( $obj->getRevenue($game, $period)); |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | gamma94 |
| Description: | Period controller function that returns a random number between .95 and 1.05 as a modifier for impact and news |
| Test Steps | Step 1: $obj = new periodController(); |
| Step 2: $this->AssertGreaterThan(.95, $obj->gamma()) |
| Step 3: $this->AssertLessThan(1.05, $obj->gamma()) |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
|  |  |
| Test ID: | IMPACT\_SUB\_J95 |
| Description: | Period controller function that returns the impact for a group |
| Test Steps | Step 1: $obj = new periodController(); |
| Step 2: $db = new database(); |
| Step 4: $decisions = $db->getDecisions(110) |
| Step 5: $this->AssertGreaterThan(3.15, count($obj->IMPACT\_SUB\_J($decisions, gameContgroller::randomMuOmega)); |
| Step 5: $this->AssertLessThan(14.5, count($obj->IMPACT\_SUB\_J($decisions, gameContgroller::randomMuOmega)); |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | DELTA96 |
| Description: | Period controller function that returns the revenue for a group for a particular group and game |
| Test Steps | Step 1: $obj = new periodController(); |
| Step 2: $game = 3; |
| Step3: $period = 2; |
| Step 4: $BETA\_J = Array; |
| Step 5: $temp = array("group"=> 110, "BETA\_j"=>1500) |
| Step 6: array\_push($BETA\_J, $temp); |
| Step 5: $this->AssertGreateThan(0, count($obj->DELTA9($game, $period, $BETA\_J); |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | BETA\_SUB\_J96 |
| Description: | Period controller function that returns the market share for a group for a particular group and game |
| Test Steps | Step 1: $obj = new periodController(); |
| Step 2: $game = 3; |
| Step3: $period = 2; |
| Step 4: group =110 |
| Step 5: $theta = 1600 |
| Step 5: $this->AssertGreateThan(0, count($obj->DELTA9($game, $period, $BETA\_J); |
| Step 5: $this->AssertGreateThan(500, count($obj->BETA\_SUB\_J($group, $game, $period, $theta)); |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | CreatePage97 |
| Description: | Report controller function that returns the view for the website |
| Test Steps | $obj = new ReportController(); |
| Step 1: $result = "<form action='' method='post'>  <div class='bs-example'>".  "<div class='panel panel-default'>".  "<div class='panel-body' align='center'>".  "<h2>User Accounts Management</h2>".    //"<input name='button\_reports' type = 'submit' value = 'Reports' class='btn btn-primary'/>".  //"<input name='button\_commentRepository' type = 'submit' value = 'Comment Repository' class='btn btn-primary'/>".  "<a href='http://marketsim-dev.cis.fiu.edu/views/reportingPage.php?button\_reports=Reports' class='btn btn-primary'>Reports</a>".  "<a href='http://marketsim-dev.cis.fiu.edu/views/reportingPage.php?button\_commentRepository=Comment+Repository' class='btn btn-primary'>Comment Repository</a>".  "</div>".  "<div class='' align='center'>".  "<div class='panel-footer clearfix' style='width: 1075px' >".  $value.  "</div>".  "</div>".  "</div>".  "</div>".  "</form>"; |
|  |
| Step 5: $this->Assertequals (""<form action='' method='post'>  <div class='bs-example'>".  "<div class='panel panel-default'>".  "<div class='panel-body' align='center'>".  "<h2>User Accounts Management</h2>".    //"<input name='button\_reports' type = 'submit' value = 'Reports' class='btn btn-primary'/>".  //"<input name='button\_commentRepository' type = 'submit' value = 'Comment Repository' class='btn btn-primary'/>".  "<a href='http://marketsim-dev.cis.fiu.edu/views/reportingPage.php?button\_reports=Reports' class='btn btn-primary'>Reports</a>".  "<a href='http://marketsim-dev.cis.fiu.edu/views/reportingPage.php?button\_commentRepository=Comment+Repository' class='btn btn-primary'>Comment Repository</a>".  "</div>".  "<div class='' align='center'>".  "<div class='panel-footer clearfix' style='width: 1075px' >".  $value.  "</div>".  "</div>".  "</div>".  "</div>".  "</form>", count($obj->createPage()); |
|  | Step 3: $this->AssertGreateThan(500, count($obj->BETA\_SUB\_J($group, $game, $period, $theta)); |
| Tester: | Javier Andrial |
| Result: | Pass |
|  |  |
|  |  |
| Test ID: | getCommentList98 |
| Description: | Report controller function that returns a array of comments for a game |
| Test Steps | Step 1: $obj = new ReportController(); |
| Step 5: $this->AssertGreateThan(0, count($obj->getCommentList()); |
| Tester: | Javier Andrial |
| Result: | Pass |
|  |  |
| Test ID: | getCommentListFiltered99 |
| Description: | Report controller function that returns a array of comments for a game and period |
| Test Steps | Step 1: $obj = new ReportController(); |
| Step 2: $game = 3 |
| Step 3: $period = 1 |
| Step 5: $this->AssertGreateThan(0, count($obj->getCommentListFiltered()); |
| Tester: | Javier Andrial |
| Result: | Pass |
|  |  |
| Test ID: | postFeedbackValid100 |
| Description: | Report controller function that updates professor feedback with valid input |
| Test Steps | Step 1: $obj = new ReportController(); |
| Step 2: $id = 15; |
| Step 5: $this->AssertEquals("<b>Sucess:</b> Feedback has been posted", $obj->postFeedback($id, "Some feedback")); |
| Tester: | Javier Andrial |
| Result: | Pass |
|  |  |
| Test ID: | postFeedbackInValid101 |
| Description: | Report controller function that updates professor feedback with invalid input |
| Test Steps | Step 1: $obj = new ReportController(); |
| Step 2: $id = "books"; |
| Step 5: $this->AssertEquals("<b>Failed:</b> Feedback was either the same or record was not found", $obj->postFeedback($id, "Some feedback")); |
| Tester: | Javier Andrial |
| Result: | Pass |
|  |  |
| Test ID: | feedbackPage102 |
| Description: | Report controller function the feedback page |
| Test Steps | Step 1: $obj = new ReportController(); |
| $step 2: $db = new database; |
| Step 2: $comment = $db->getCommentById($id); |
| $result = "<div align='center'>"  ."<h3>Students Comment</h3>"  ."<p style='border:ridge black 3px; margin:auto; width:42.85%; padding:1em;'>"      .$comment['comments']."</p>"  ."<br />"  ."<h3>Feedback</h3>"  ."<textarea rows='10' cols='60' name=textarea\_feedback >".$comment['feedback']  ."</textarea>"  ."</div>"  ."<br />"  ."<br />"  ."<input name='button\_feedbackPost' type = 'submit' value = 'post feedback' class='btn btn-primary'/>"; |
| Step 5: $this->AssertEquals(""<div align='center'>"  ."<h3>Students Comment</h3>"  ."<p style='border:ridge black 3px; margin:auto; width:42.85%; padding:1em;'>"      .$comment['comments']."</p>"  ."<br />"  ."<h3>Feedback</h3>"  ."<textarea rows='10' cols='60' name=textarea\_feedback >".$comment['feedback']  ."</textarea>"  ."</div>"  ."<br />"  ."<br />"  ."<input name='button\_feedbackPost' type = 'submit' value = 'post feedback' class='btn btn-primary'/>";, $obj->postFeedback(15)); |
| Tester: | Javier Andrial |
| Result: | Pass |
|  |  |
|  |  |
| Test ID: | generateCommentTable103 |
| Description: | Report controller function that updates professor feedback with invalid input |
| Test Steps | Step 1: $myTable = "<table>"  ."<thead>"  ."<tr>"  ."<th>".$this->filterMenu($gamesChecked,$periodsChecked)."</th>"  //."<th> </th>"  ."<th>".$this->commentList($commentArray)."</th>"  ."</tr>"  ."</thead>" |
| Step 5: $this->AssertEquals(""<table>"  ."<thead>"  ."<tr>"  ."<th>".$this->filterMenu($gamesChecked,$periodsChecked)."</th>"  //."<th> </th>"  ."<th>".$this->commentList($commentArray)."</th>"  ."</tr>"  ."</thead>";, $obj->postFeedback($id, "Some feedback")); |
| Tester: | Javier Andrial |
| Result: | Pass |
|  |  |
| Test ID: | generateCommentTable104 |
| Description: | Report controller that generates comments table |
| Test Steps | Step: 1$mydatabase = new database(); |
| Step 2: $hotelArray = $mydatabase->getAllHotel(); |
| Step 3: $locationArray = $mydatabase->getAllLocation(); |
| Step 4: $currentHotel; |
| step 5: $result = ' <div class="table-responsive">'  .'<table class="table table-bordered">'  .'<thead>'  .'<tr>'  .'<th> Game</th>'  .'<th>Period</th>'  .'<th>Hotel</th>'  .'<th>Type</th>'  .'<th>Location</th>'  .'<th>Comment</th>'  .'<th>Feedback</th>'  .'<th></th>'  .'</tr>'  .'</thead>'; |
| Step 5: $this->AssertEquals("<div class="table-responsive">'  .'<table class="table table-bordered">'  .'<thead>'  .'<tr>'  .'<th> Game</th>'  .'<th>Period</th>'  .'<th>Hotel</th>'  .'<th>Type</th>'  .'<th>Location</th>'  .'<th>Comment</th>'  .'<th>Feedback</th>'  .'<th></th>'  .'</tr>'  .'</thead>';;, $obj->CommentList()); |
| Tester: | Javier Andrial |
| Result: | Pass |
|  |  |
| Test ID: | chooseGame105 |
| Description: | Report controller function that updates professor feedback with invalid input |
| Test Steps | Step:1: $mydatabase = new database(); |
|  | Step 2: $gamesrray =$mydatabase->getGameAllGames(); |
|  | Step 3: $locationArray = $mydatabase->getAllLocation(); |
| Steps | Step 4: $$result = '<div class="table-responsive">  <h3>Select a Game to view its details</h3><br />  <table class="table table-bordered">  <thead>  <tr>  <th></th>  <th>Course #</th>  <th>Course</th>  <th>Semester</th>  <th>Section</th>  <th>Schedule</th>  <th>is Active</th>  </tr>  </thead>'; |
|  | Step 5: $this->AssertEquals('<div class="table-responsive">  <h3>Select a Game to view its details</h3><br />  <table class="table table-bordered">  <thead>  <tr>  <th></th>  <th>Course #</th>  <th>Course</th>  <th>Semester</th>  <th>Section</th>  <th>Schedule</th>  <th>is Active</th>  </tr>  </thead>', $obj->chooseGame()); |
| Tester: | Javier Andrial |
| Result: | Pass |
|  |  |
|  |  |
| Test ID: | selectPeriod106 |
| Description: | Report controller returns the view for selected period |
| Test Steps | Step:1: $mydatabase = new database(); |
| Step 2: $game =$mydatabase->getGame(15); |
| Step 3: $result= '<h3>Select a Period</h3><br />  <div style="width: 10%">  <table class="table table-bordered">  <thead>  <tr>  <th></th>  <th>Period</th>  </tr>  </thead>' |
| Step 4: $$result = '<div class="table-responsive">  <h3>Select a Game to view its details</h3><br />  <table class="table table-bordered">  <thead>  <tr>  <th></th>  <th>Course #</th>  <th>Course</th>  <th>Semester</th>  <th>Section</th>  <th>Schedule</th>  <th>is Active</th>  </tr>  </thead>'; |
| Step 5: for($int = 1; $int <= $game['periodNum'];$int++ )  {  $result .="<tbody>"  ."<tr>"  ."<td>"."<input type='radio' name='periodRadio' value='".$int."'></td>"  ."<td>".$int."</td>"  ."</tr>"  ."</tbody>";  }  $result .= "</table>".  "<br />".  "<input name='button\_choosePeriod' type = 'submit' value = 'Choose a Period' class='btn btn-primary'/> ".  "" |
| Step 5 $this->assertEquals('<div class="table-responsive">  <h3>Select a Game to view its details</h3><br />  <table class="table table-bordered">  <thead>  <tr>  <th></th>  <th>Course #</th>  <th>Course</th>  <th>Semester</th>  <th>Section</th>  <th>Schedule</th>  <th>is Active</th>  </tr>  </thead>. <tbody>"  ."<tr>"  ."<td>"."<input type='radio' name='periodRadio' value='".$int."'></td>"  ."<td>".$int."</td>"  ."</tr>"  ."</tbody>" , $obj->selectPeriod(15)); |
| Tester: | Javier Andrial |
| Result: | Pass |
|  |  |
| Test ID: | LoadData107 |
| Description: | PDF generator function that loads the data from a file |
| Test Steps | Step:1: $lines = file($file); |
|  | Step 2: $data = array(); |
|  | Step 3:foreach($lines as $line)  $data[] = explode(';',trim($line)); |
|  | Step 5 $this->assertGreaterThan(0 , $obj->LoadData(file)); |
| Tester: | Javier Andrial |
| Result: | Pass |
|  |  |
|  |  |
| Test ID: | Alpha108 |
| Description: | test database function that returns the and array of arrays which include group name and average rate each |
| Test Steps | Step 1: $obj = new database(); |
| Step 2: $game = 3; |
| Step: $period = 2; |
| Step 5: $this->AssertEquals(6, count($obj->Alpha($game, perion)); |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | Epsilon109 |
| Description: | test database function that returns the hotel name and OTA allocations for a particular group, game, and period. |
| Test Steps | Step 1: $obj = new database(); |
| Step 2: $game = 3; |
| Step3: $period = 2; |
| Step 4: $group = 110; |
| Step 5: $this->AssertEquals(6, count($obj->Epsilon($game, $period, $group)); |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | getGroupComments110 |
| Description: | test database function that returns the comments for a group for a certain game and period |
| Test Steps | Step 1: $obj = new database(); |
| Step 2: $game = 3; |
| Step3: $period = 2; |
| Step 4: $group = 110; |
| Step 5: $this->AssertEquals(1, count($obj->getGroupComments($game, $period, $group)); |
| Tester: | Jeffrey Carman |
| Result: | Pass |
|  |  |
| Test ID: | MuOmega111 |
| Description: | Period controller function that returns the impact of news based given a string of either "Very Good", fairly Good", "None", "Fairly Bad", or "Very Bad" |
| Test Steps | Step 1: $obj = new periodController(); |
| Step 2: $this->AssertEqauls(1.3, $obj-uOmega("Very Good")); |
| Step 3: $this->AssertEqauls(1.15, $obj-uOmega("Fairly Good")); |
| Step 4: $this->AssertEqauls(.85, $obj-uOmega("Fairly Bad")); |
| Step 4: $this->AssertEqauls(.7, $obj-uOmega("Very Bad")); |
| Tester: | Jeffrey Carman |
| Result: | Pass |

|  |  |
| --- | --- |
| **Test ID:** | **feedbackPageStudent112** |
| **Description:** | **Test ReportController function that returns a page displaying a selected period’s feedback** |
| **Test Steps** | **Step 1 :$obj = new ReportController();** |
| **Step 2 :$value = 2;** |
| **Step 3: $mydatabase = new database();**  **$comment = $mydatabase->getCommentById($value);**  **$result = "<div align='center'>"**  **."<h3>Students Comment</h3>"**  **."<p style='border:ridge black 3px; margin:auto; width:42.85%; padding:1em;'>"**  **.$comment['comments']."</p>"."<br />"."<h3>Feedback</h3>"**  **."<textarea readonly rows='10' cols='60' name=textarea\_feedback >".$comment['feedback']."</textarea>"."</div>"."<br />"."<br />"**  **."<a href='http://marketsim-dev.cis.fiu.edu/views/reportingPage.php' class='btn btn-primary'>Back</a>";**  **return $result;** |
| **Step 4:>assertEquals($result, $obj->feedbackPageStudent ($value));** |
| **Tester:** | **Javier Andrial** |
| **Result:** | **Pass** |

|  |  |
| --- | --- |
| **Test ID:** | **commentListStudent113** |
| **Description:** | **Test ReportController function that returns a table of comments** |
| **Test Steps** | **Step 1 :$obj = new ReportController();** |
| **Step 2 :$value = “jandr018@fiu.edu”;** |
| **Step 3: $mydatabase = new database();**  **$student = $mydatabase->getStudent($value);**  **$comments = $mydatabase->getCommentsByHotel($student['hotel']);**    **$result = ' <div class="table-responsive">'.'<table class="table table-bordered">'.'<thead>'.'<tr>'.'<th> Period</th>'.'<th>Comment</th>'**  **.'<th>Feedback</th>'**  **.'<th></th>'**  **.'</tr>'**  **.'</thead>';**  **$int = 1;**  **foreach($comments as $key => $temp )**  **{**  **if(strlen($temp['comments'])>25)**  **$temp['comments'] = substr(preg\_replace( "/\n\s+/", "\n", rtrim(html\_entity\_decode(strip\_tags( $temp["comments"] ))) ), 0,22)."...";**  **if(strlen($temp['feedback'])>25)**  **$temp['feedback'] = substr(preg\_replace( "/\n\s+/", "\n", rtrim(html\_entity\_decode(strip\_tags( $temp["feedback"] ))) ), 0,22)."...";**    **$result .=**  **"<tbody>"."<tr>"."<td>".$temp["period"]."</td>"**  **."<td>".$temp["comments"]."</td>"."<td>".$temp["feedback"]."</td>"**  **."<td>"."<a href='http://marketsim-dev.cis.fiu.edu/views/reportingPage.php?button\_viewfeedback=View+Feedback&commentId=".$temp['id']."' class='btn btn-primary'>View Feedback</a>"."</td>"."</tr>"."</tbody>";**  **}**  **$result .= "</table></div>";**  **return $result;** |
| **Step 4:>assertEquals($result, $obj->commentListStudent ($value));** |
| **Tester:** | **Javier Andrial** |
| **Result:** | **Pass** |

|  |  |
| --- | --- |
| **Test ID:** | **getCommentsByHotel114** |
| **Description:** | **Test Database function that returns an array of comments entities** |
| **Test Steps** | **Step 1 :$obj = new Database();** |
| **Step 2 :$value = 115;** |
| **Step 3: $v = $this->conn->real\_escape\_string($value);**  **$r = $this->conn->query(sprintf("select \* from student\_comments where hotel = '%s';",$v));**  **$resArr = array();**  **while($result = $r->fetch\_assoc())**  **{**  **$resArr[] = $result ;**  **}**  **return $resArr;** |
| **Step 4:>assertEquals($result, $obj-> getCommentsByHotel($value));** |
| **Tester:** | **Javier Andrial** |
| **Result:** | **Pass** |

|  |  |
| --- | --- |
| Test ID: | CreatePage115 |
| Description: | Test ReportController function that returns a template to account page |
| Test Steps | Step 1 :$obj = new ReportController; |
| Step 2 :$value = "someValue"; |
| Step 3: $result = "<form action='' method='post'>  <div class='bs-example'>".  "<div class='panel panel-default'>".  "<div class='panel-body' align='center'>".  "<h2>User Accounts Management</h2>"."<a href='http://marketsim-dev.cis.fiu.edu/views/reportingPage.php?button\_reports=Reports' class='btn btn-primary'>Reports</a>". "<a href='http://marketsim-dev.cis.fiu.edu/views/reportingPage.php?button\_commentRepository=Comment+Repository' class='btn btn-primary'>Comment Repository</a>". "</div>". "<div class='' align='center'>". "<div class='panel-footer clearfix' style='width: 1075px' >".  $value.  "</div>". "</div>". "</div>". "</div>". "</form>"; |
| Step 4:>assertEquals($result, $obj->CreatePage($value)); |
| Tester: | Javier Andrial |
| Result: | Pass |

# Glossary

**Actors:** A user of the system or a bot.

**Budget:** An estimate, often itemized, of expected income and expense for a given period in the future.

**Bootstrap Framework:** A free and open-source collection of tools for creating websites and web applications.

**Bot**: A program which simulates a group.

**Dashboard**: a panel which displays relevant information about other users hotels from the previous period.

**CSS:** A webpage derived from multiple sources with defined order of precedence where the definitions of any style element conflict.

**Decisions (Phi)**: Budgetary allocations made by a user or bot which affect the market share.

End of Period Algorithm: The end of period algorithm, is a set of mathematical formulas which are used at the end of each period to determine the market share and revenue earned for each group. The algorithm has four main parts: Impact(Phi), Market Share(Beta) and revenue(Delta), and rooms sold(theta). The impact is calculated using the product of all decisions that each group has made. This is further modified be a random impact modifier (Omega). Total rooms sold is determined by taking the average impact by group multiplied by 1/10 the total number of rooms available in the market. The market share is determined by the Average rate of a group multiplied by the total rooms sold, divided by the summation of all average rates Finally the revenue for each group is determined by a group’s average rate multiplied by their respective market share for that period. Revenue is modified by a 30 percent penalty for OTA allocations.

**Game**: A finite number of periods encompassing the entire simulation for a hotel market where a group’s decisions are used to calculate the winner of a period.

**Group:** A group is a game object made up of at least 2 students. A group selects strategic decisions for a game period in order to affect market share.

**Home page:** The default page of the simulation from which a student/user can view the market segment information, the leaderboard, and market share.

**Impact:**  Impact is a numerical value assigned to decisions and news article that affect the outcome of a period by means of the end of period algorithm which uses impact the determine the market share for each group in a period. The impact for decisions are always positive values ranging between 1.00, 1.30. The impact for news articles for a period allow for both negative and positive values ranging between .70 and 1.30.

**Instructor:** An administrator of the system who can create actor accounts and other instructor accounts.

**Javascript**: A high-level, dynamic untyped, and interpreted programming language used for front-end web page scripting

**JSON:**  An open standard format that uses human-readable text to transmit data objects consisting of attribute- value pairs primarily used in asynchronous-browser/server communications(AJAX). .

**jQuery:** A cross-platform javascript library designed to simplify the client-side scripting of HTMl.

**Leaderboard:**  The Leaderboard is a section of the homepage which, after the first period of a game, displays the winners of the previous game in descending order of revenue earned.

**Leaderboard chart:** The leaderboard chart is a visual representation of the leaderboard in the form of a bar graph.

**Location**: A location is a modifier to game impact which allows a news article impact to affect a group which has chosen that location a the point that they either created a group or joined an already created group.

**Market:** a region in which goods and services are bought, sold, or used.

**Market Segment:** The Market segment is the selected option of a group for either economy, mid-range, or luxury hotels.

**Market Share(BETA)**: The market share is the percentage of hotels sold in the market as a whole by a group.

**Market share chart:** The market share chart is a visual representation in the form of a pie chart of the market share for the previous period. the market share chart is displayed to a student on the homepage after the first period

**News Article (parameters):** A news article is a story about what is happening within the simulated hotel market. Each story has an impact to a game period’s market share which is either negative or positive. The news impact for a period can affect either location or market segment such that any group with a selected market segment and/or location will receive the penalty or advantage given by the news article. The news article content should be written in such a way that a student can determine how it might affect the market share.

**Period:** a set length of time in a game during which a group has the opportunity to select decisions which affect the market share and revenue for their group and other participating groups in a game

**OTA(Online Travel Agencies):** OTA is an option of strategic decisions for a period where room reservations are guaranteed but a decreased rate of 70 percent of a group’s selected average rate. OTA allocations for a period directly affect a group’s revenue for that period.

**Random Impact Modifier(GAMMA):**  The Random Impact Modifier is a value between .95 and 1.05 that is used to modify the impact for both decisions and new articles (where applicable).

**Revenue (DELTA):** Revenue is defined by the total number of dollars a groups has earned during a particular period. Revenue is determined by the market share a group has received for a period multiplied by the average room rate choses by that group. Revenue can also be modified by ATO allocations which sell a guaranteed number of rooms at a 70 percent reduction off of the average rate selected.

**Report:**  A report is a PDF document generated by the system for an instructor that displays end of period and/or end of game results for a game. A report can be generated to focus on group revenue, market segment, location or market share

**Scoreboard**: a panel which displays all current information about a user's group for that period consisting of remaining budget, market share and number of rooms sold for the previous period.

**Scrum:**  An iterative and incremental agile software development methodology for managing product development.

**Strategic Marketing Simulator:** Application which simulates the effect of marketing budgets on a hotel room sales market.

**Student:** A user of the system who belongs to a group of students which collectively make decisions which affect a game period.

**Navbar**: a panel with buttons that will always be visible for both user and instructor. Navbar is used for navigating through Strategic Marketing Simulator web pages.

**Task:** A task is a fundamental operation needed to be accomplished in order to meet all of the requirements of a user story.

**User story:** A user story is a feature of the system described in the first person to communicate to a developer how the system should function and look.

# Appendix

## Appendix A - UML Diagrams

### Static UML Diagrams

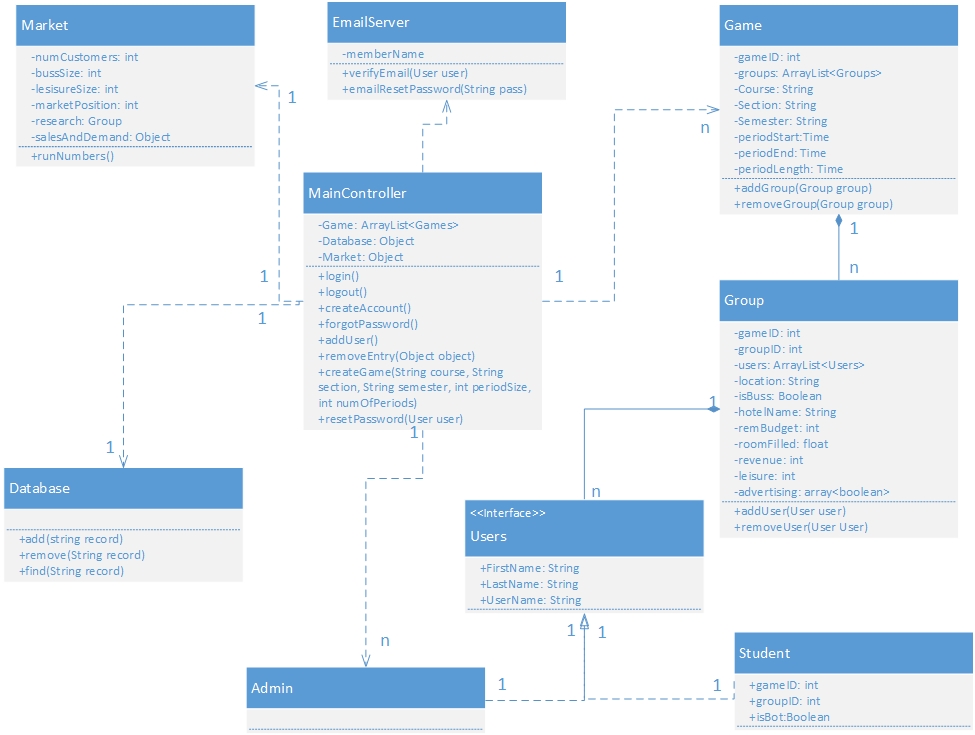


Figure 3: Class diagram

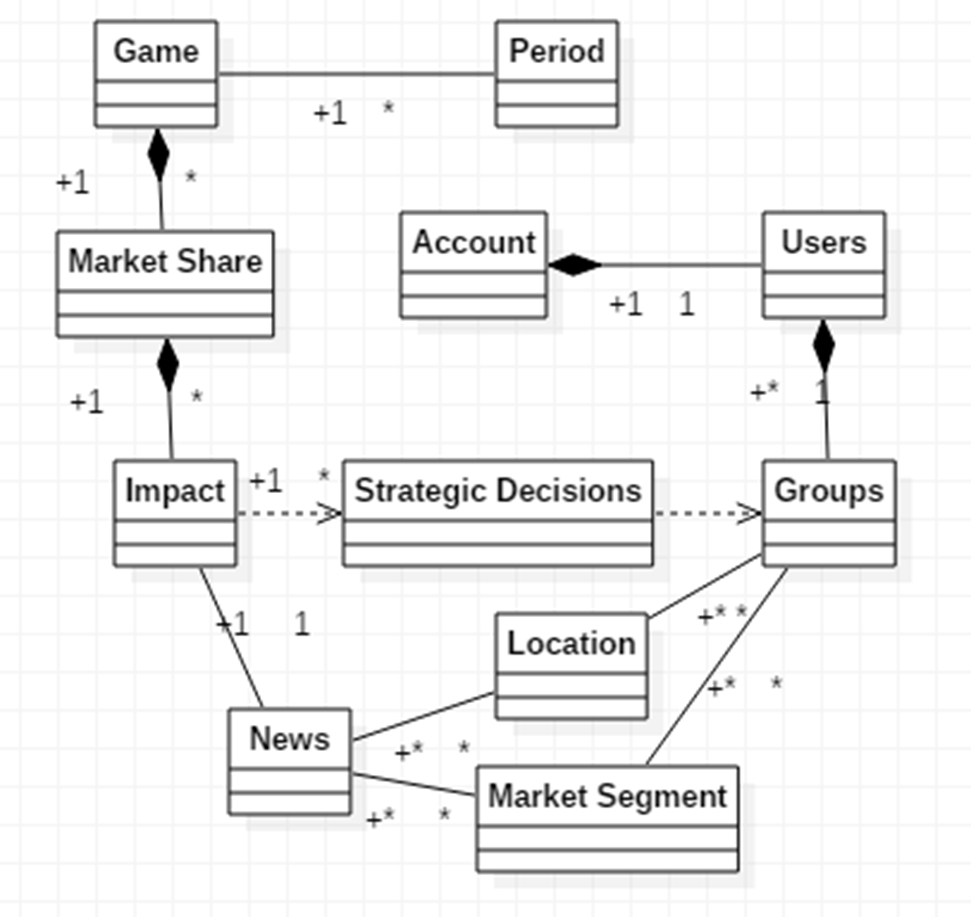


Figure 4 Object Diagram: These are the diagrams involved in a game period. There is a one to many association between a game object and period objects. A game is composed of many marketshare objects which are each composed of many group Impacts. There is a similar assocaition between news and location and market Segment.

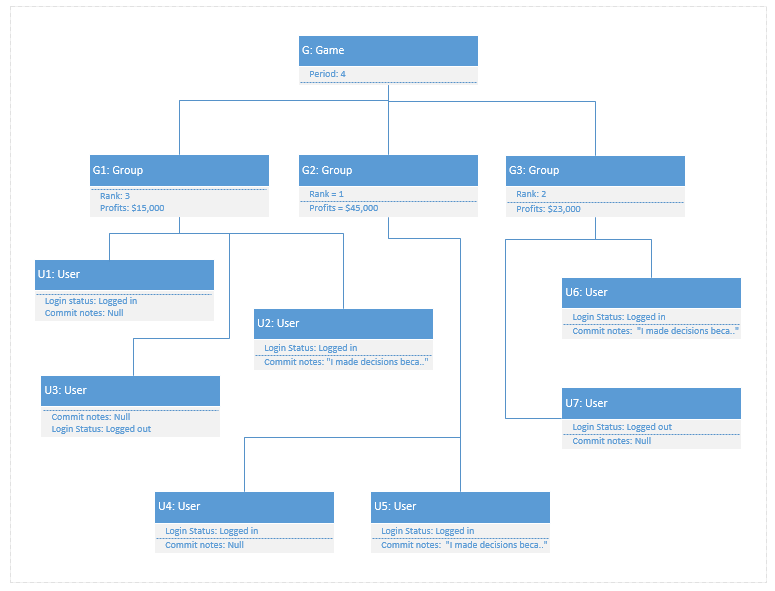


Figure 5: This is an object diagram to illustrate the instance of a Game object and at a the moment when period when period 4 ends. The object Game listed as G above, is connected to its main elements, Groups, which are listed as G1, G2, and G3. The main values associated with each group are Rank, the groups current rank compared to other groups, and profits, the total amount of profits earned based on their strategic decisions. Each group is connected to its main elements, Users, listed as U1, U2..U7 above. Each user have the two main values associated with them: the Login status binary value of either “logged in” or “logged out” and a binary value of “commit notes”, which indicates if that is the *decisive user* for the period.

…

### Dynamic UML Diagrams

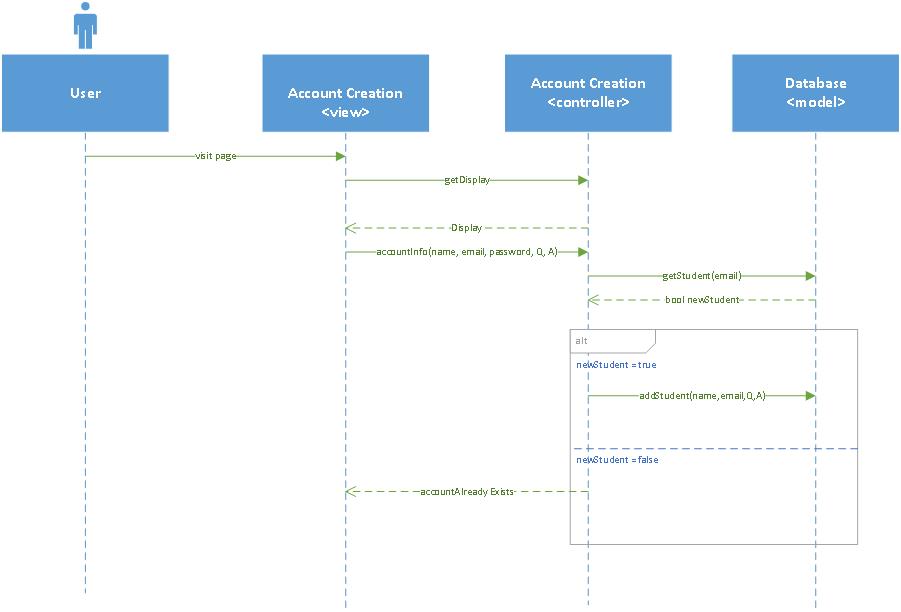


Figure 6: Account creation sequence diagram.

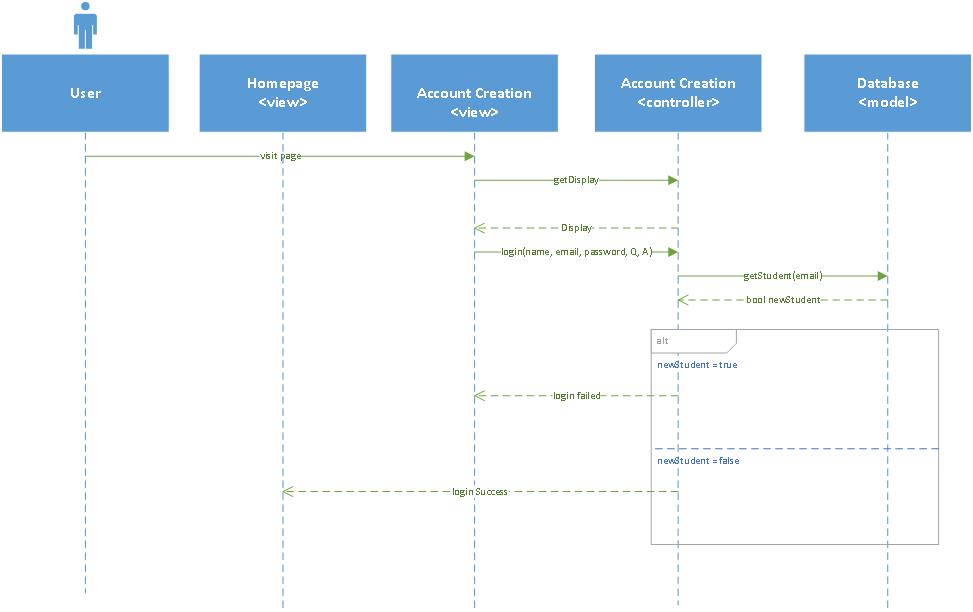


Figure 7: User login sequence diagram

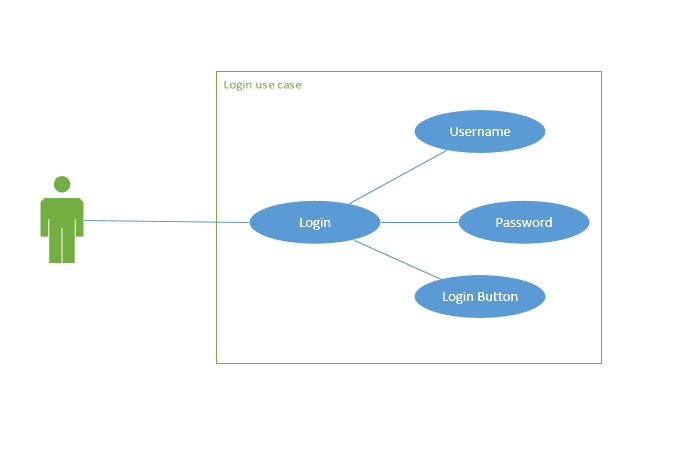


Figure 8: User login use case diagram

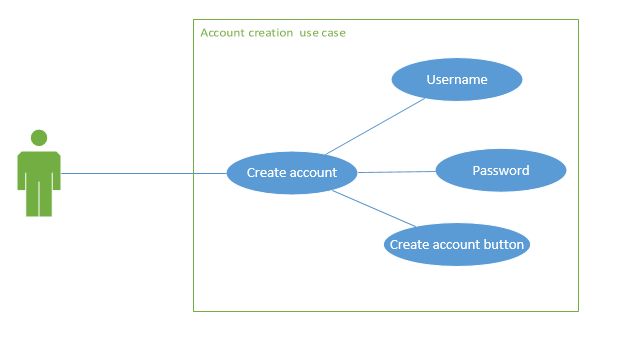


Figure 9: Account creation use case diagram

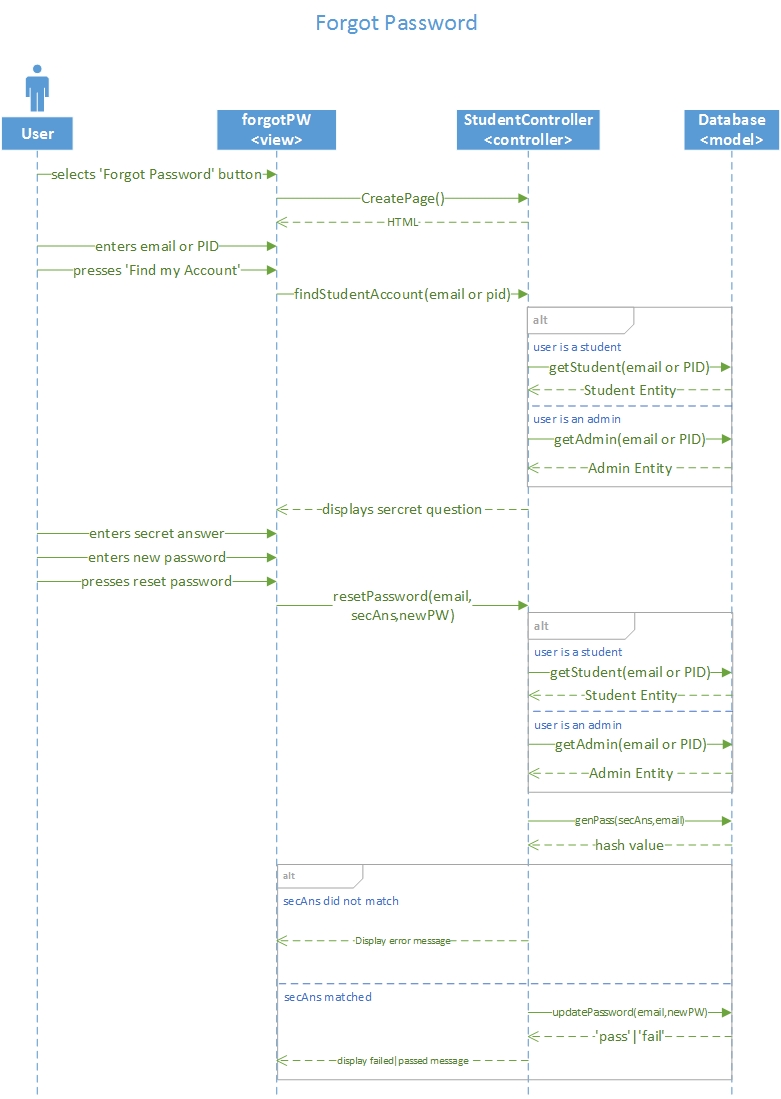


Figure 10 Forgot my password sequence diagram

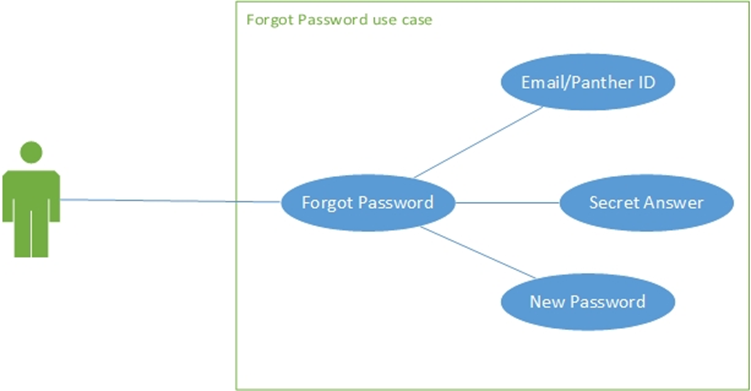


Figure 11 Forgot my password use case diagram

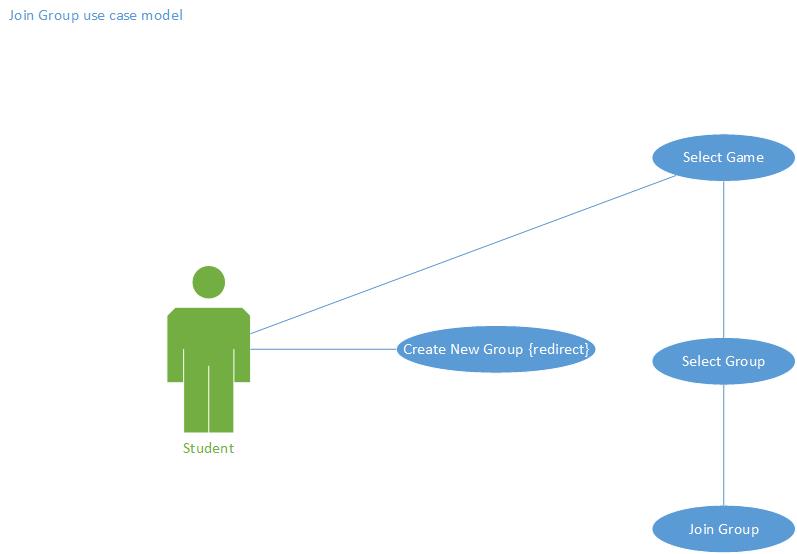


Figure 12 Join group use case diagram

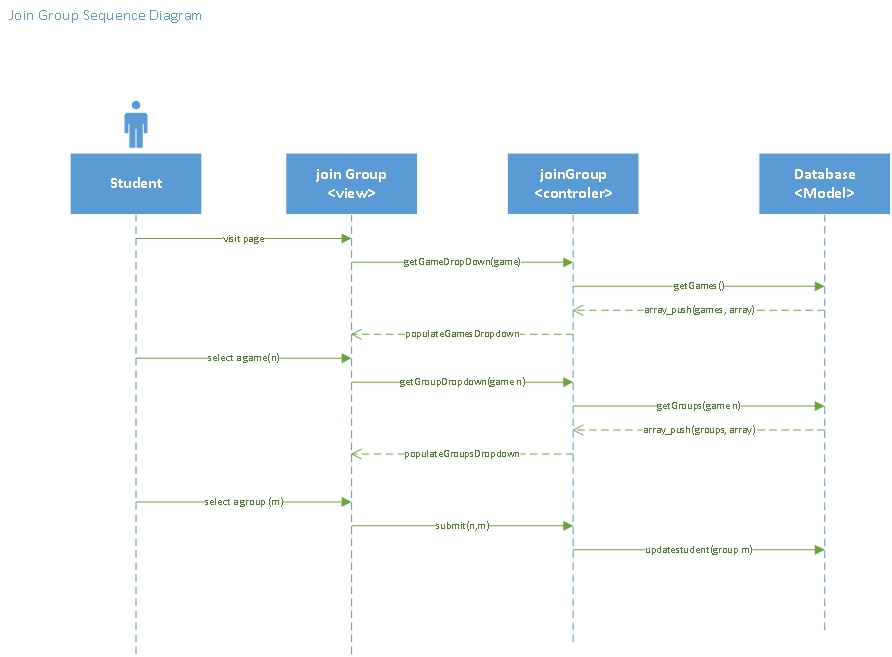


Figure 13 Join Group Sequence Diagram

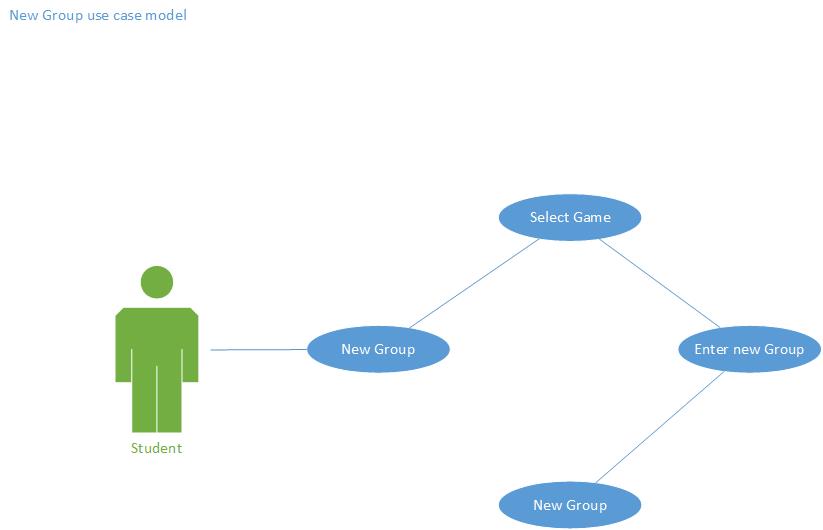


Figure 14 Create Group use case diagram

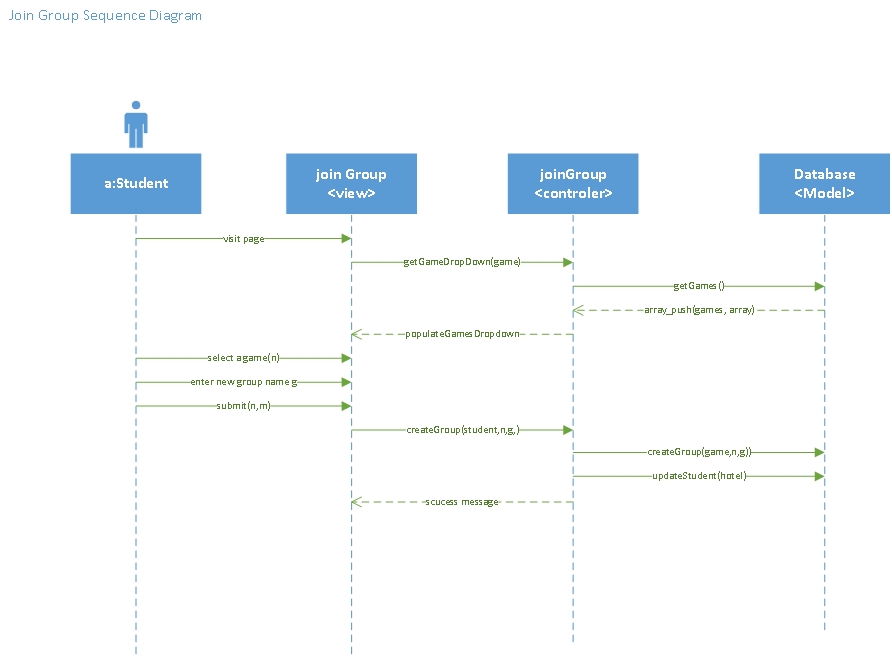


Figure 15 Create Group Sequence diagram

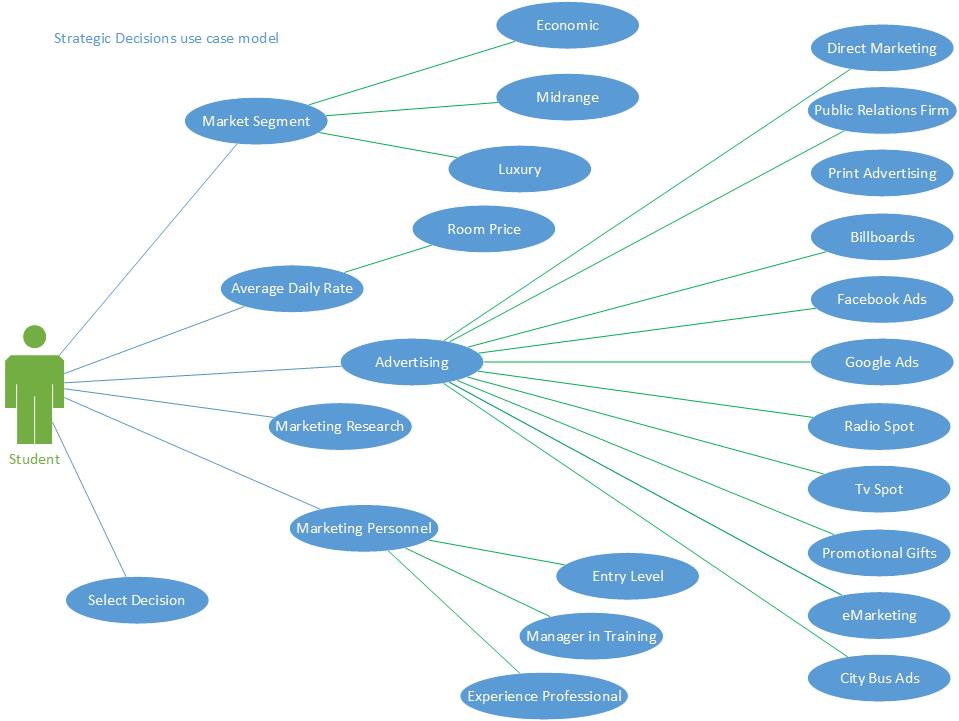


Figure 16 Strategic Marketing Use Case Diagram

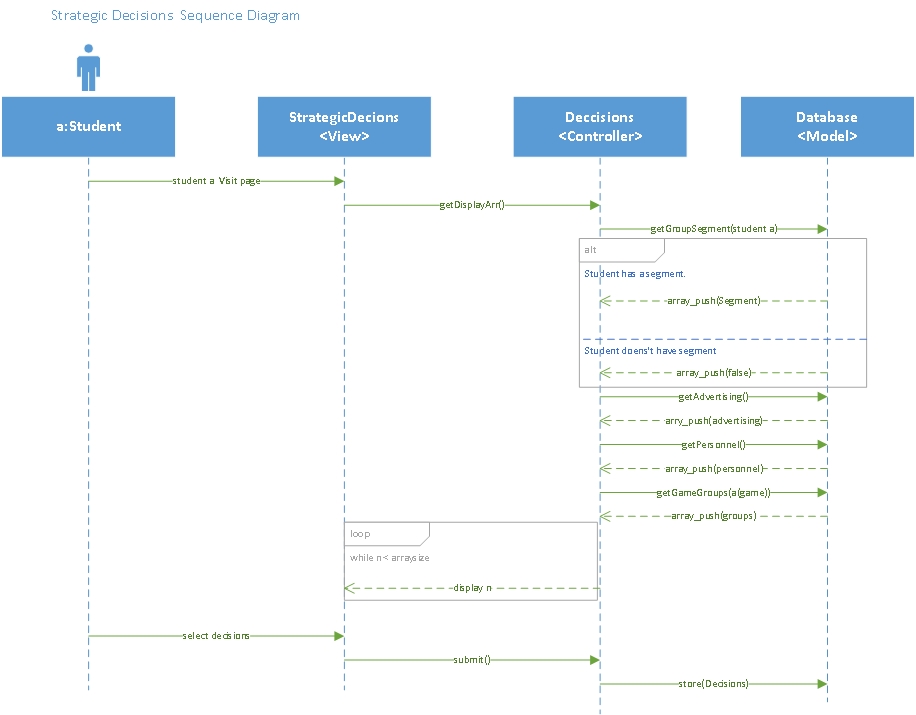


Figure 17 Strategic Decisions Sequence Diagram

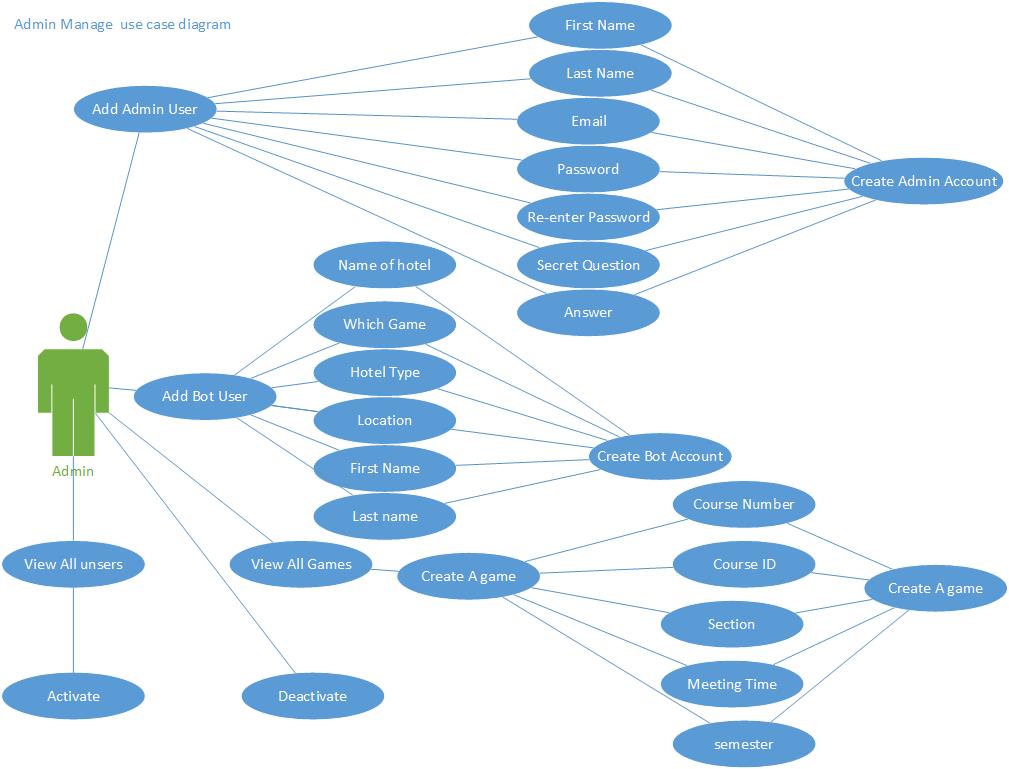


Figure 18 Admin Manage User Case Diagram

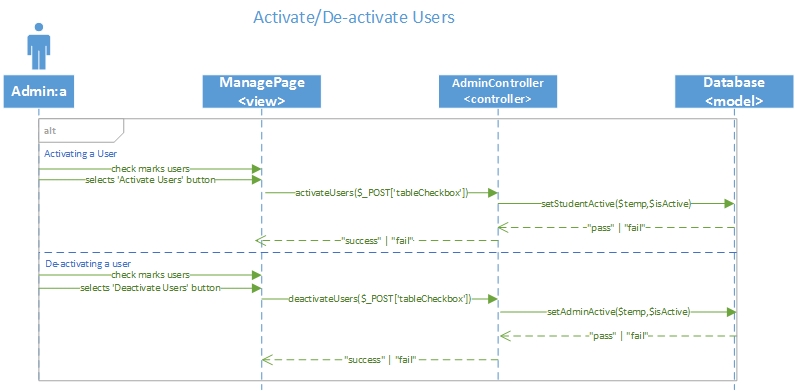


Figure 19 Activate/deactivate student sequence diagram

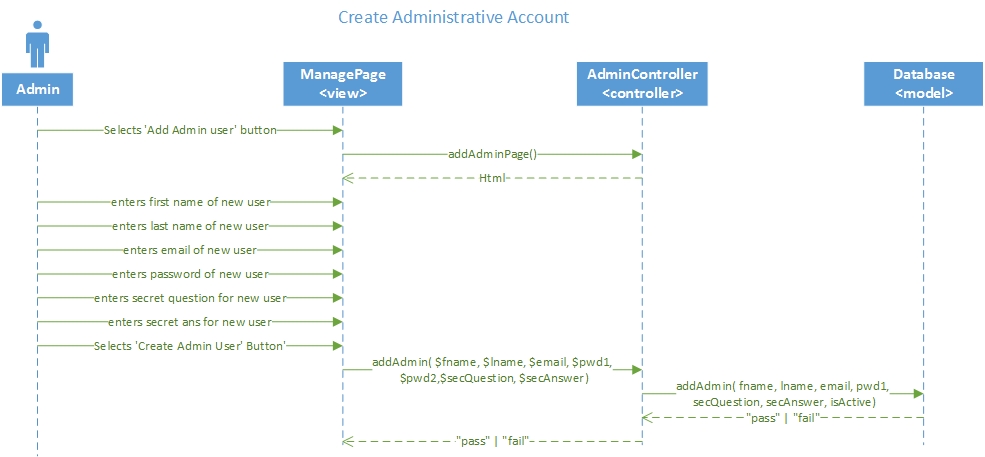


Figure 20 Create admin sequence diagram

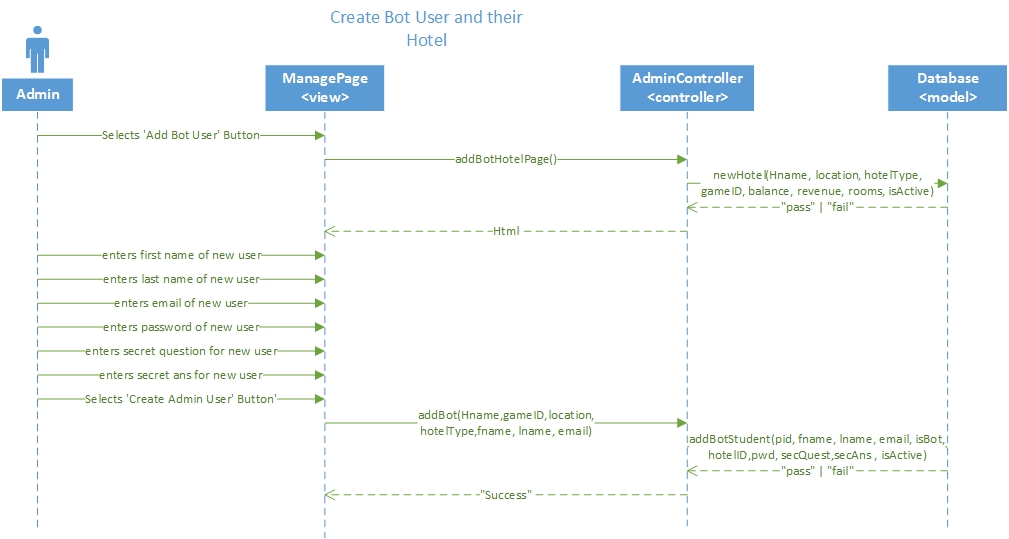


Figure 21 Create Bot sequence diagram

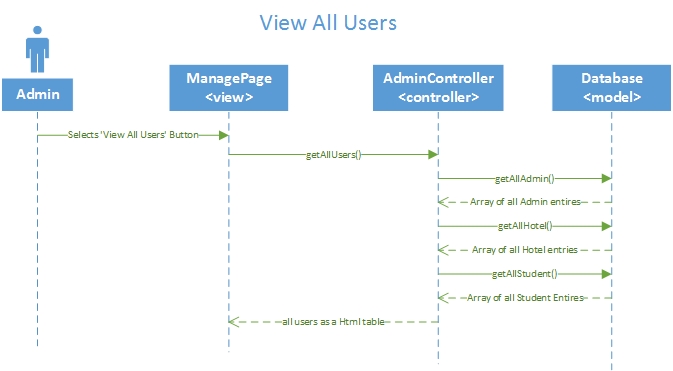


Figure 22 View All Users sequence diagram

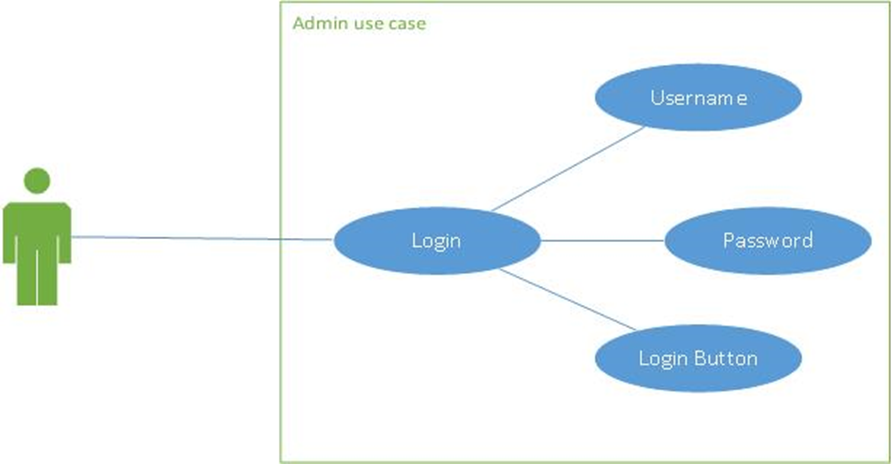


Figure 23 Admin login use case diagram

Figure 24 Admin Strategic Decisions use case diagram

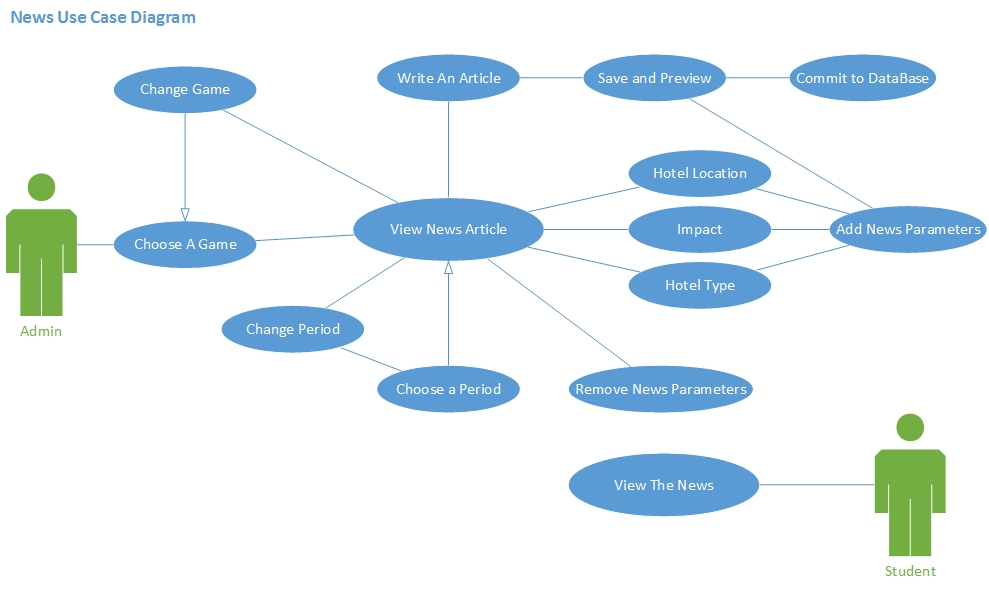


Figure 25 News Page use case diagram

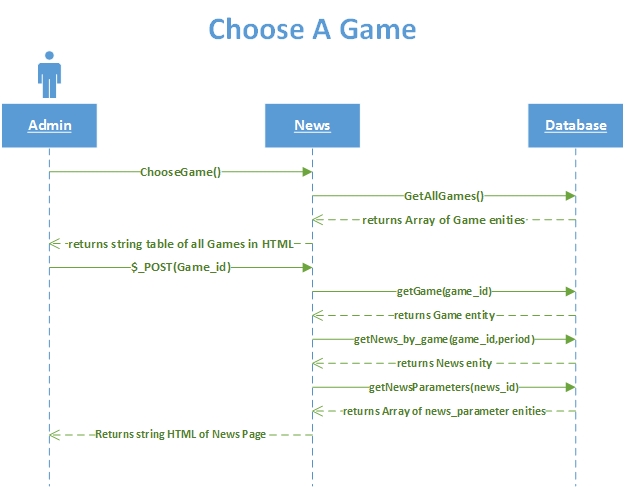


Figure 26 Admin News Choose a Game Sequence Diagram

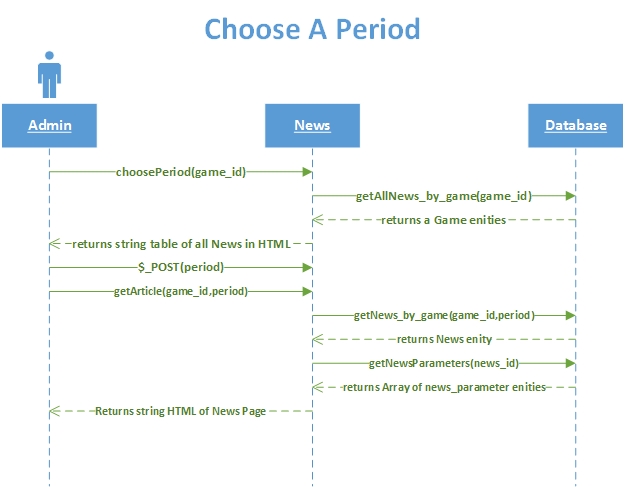


Figure 27 Admin News Choose a Period Sequence Diagram



Figure 28 Admin News Save And Commit Sequence Diagram

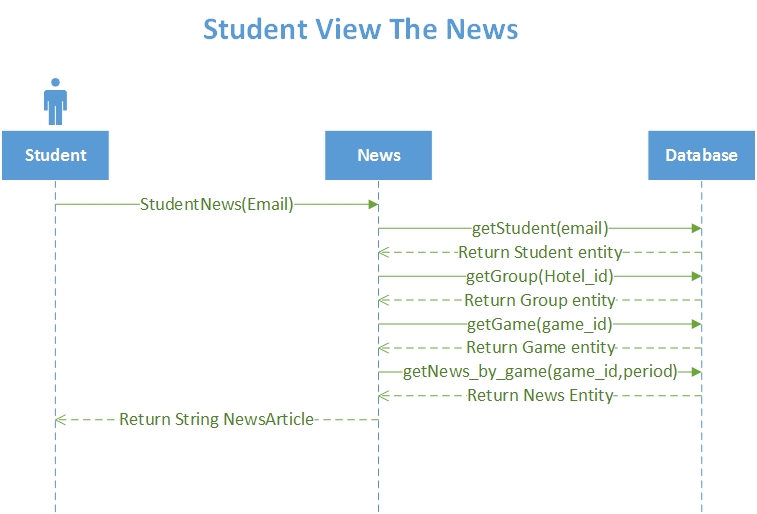


Figure 29 Student News View Article Sequence Diagram

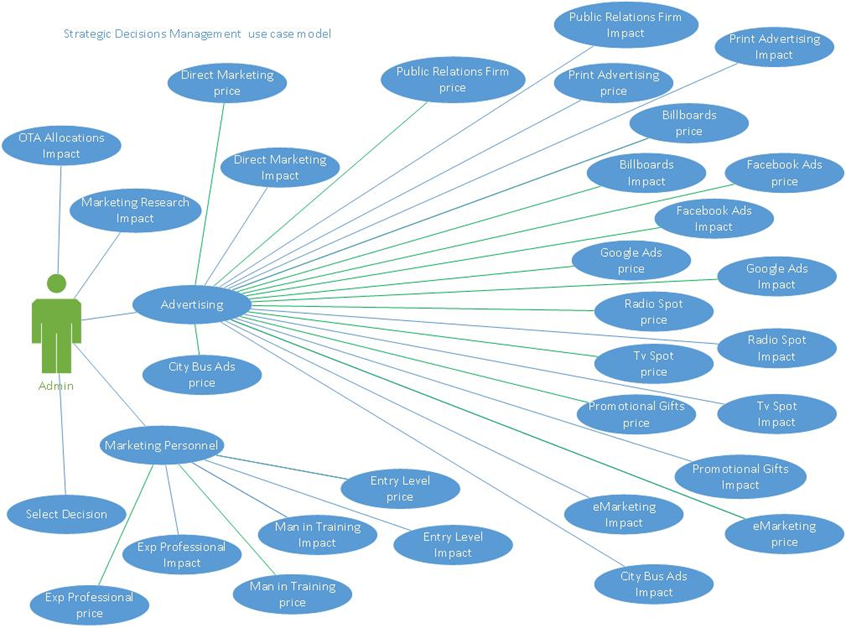


Figure 30 Strategic Decisions Management use case diagram

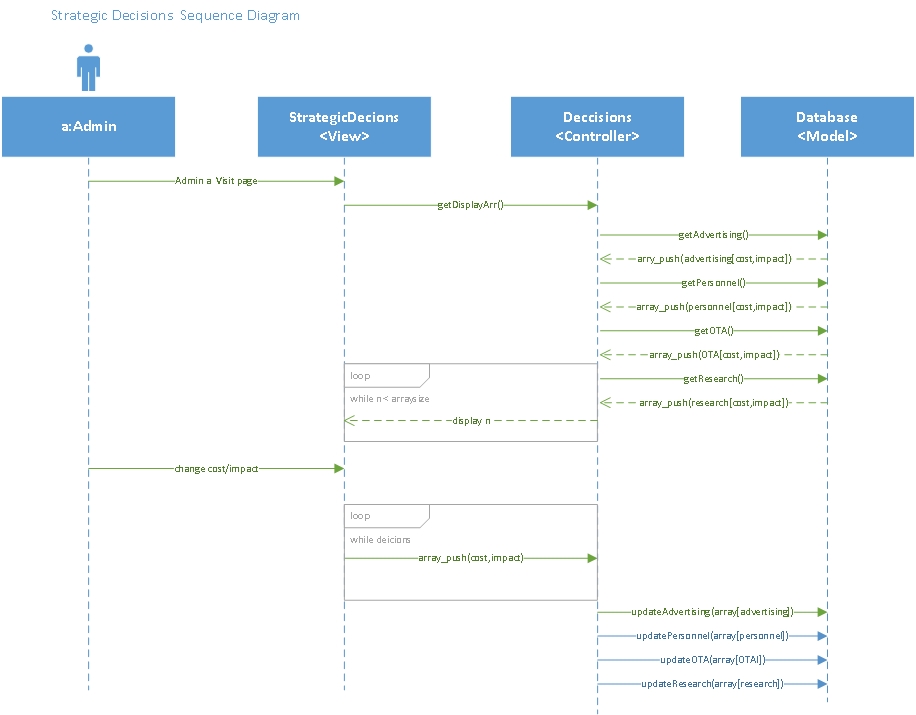


Figure 31 Strategic Decisions Management Sequence diagram

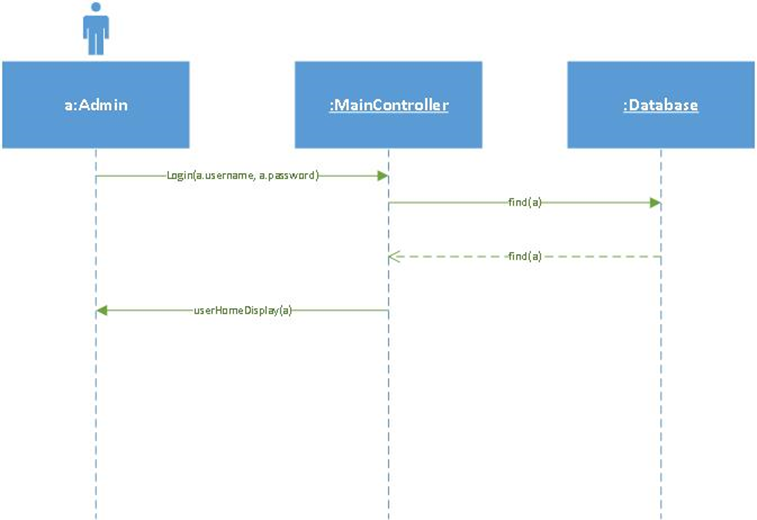


Figure 32 Admin Login Sequence Diagram

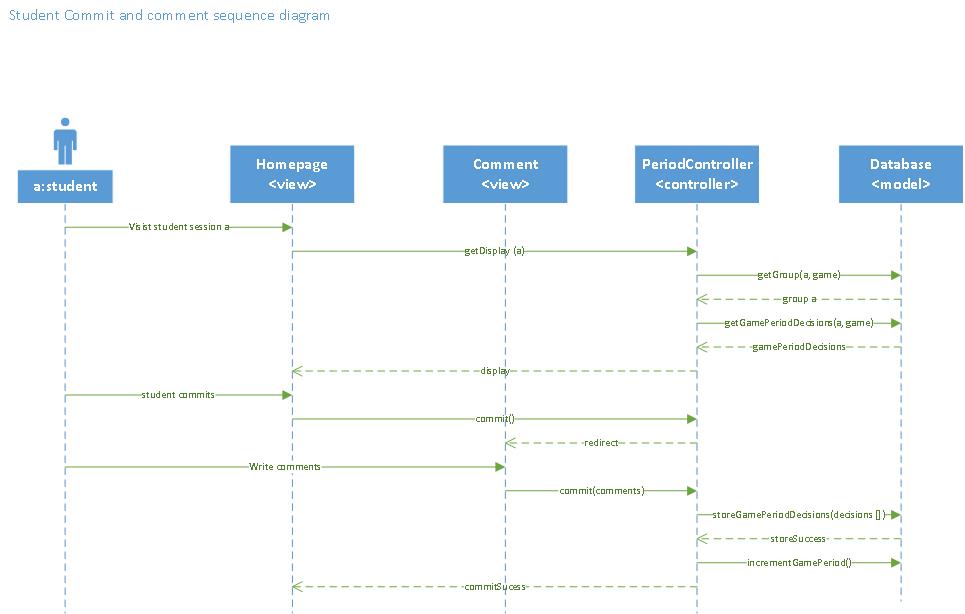


Figure 33 Student commit and comment sequence diagram

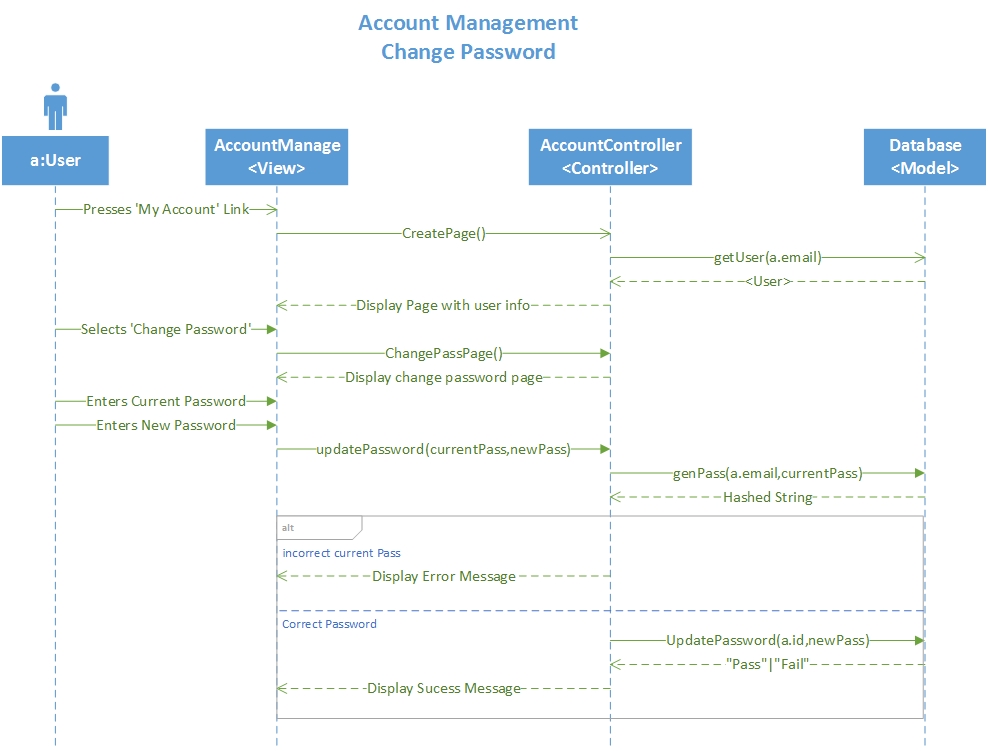


Figure 34 Account Management Change Password

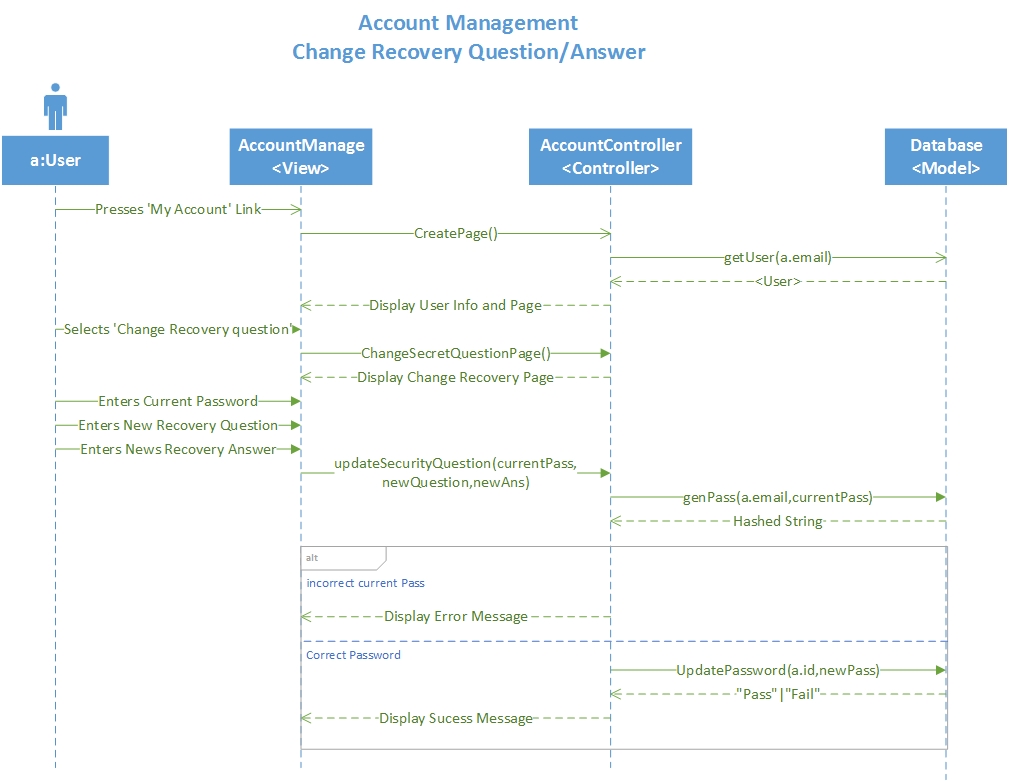


Figure 35 Account Management Change Recovery Question/Answer

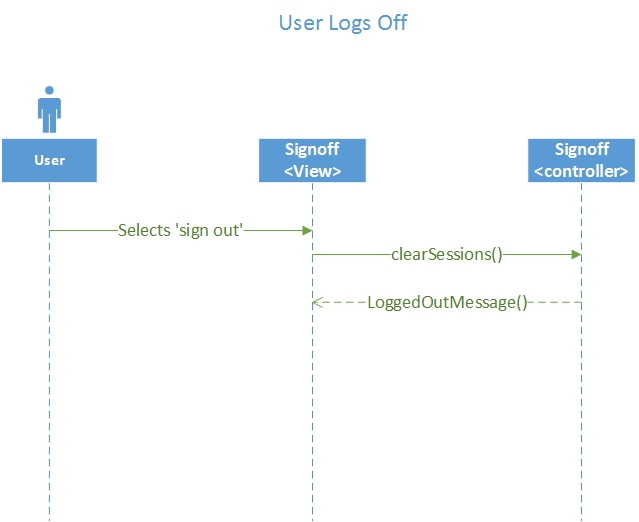


Figure 36 User Logs off

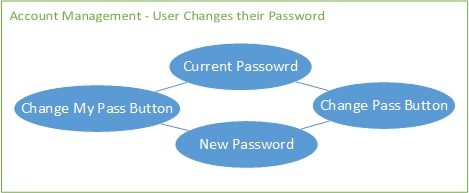


Figure 37 Account Management - User changes their password

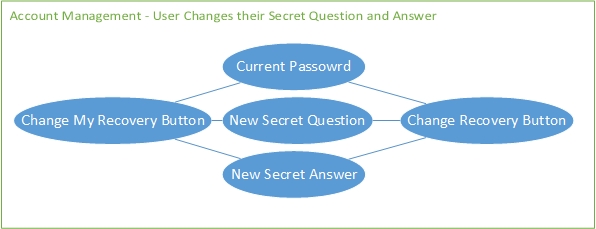


Figure 38 Account Management - User changes secret answer and question

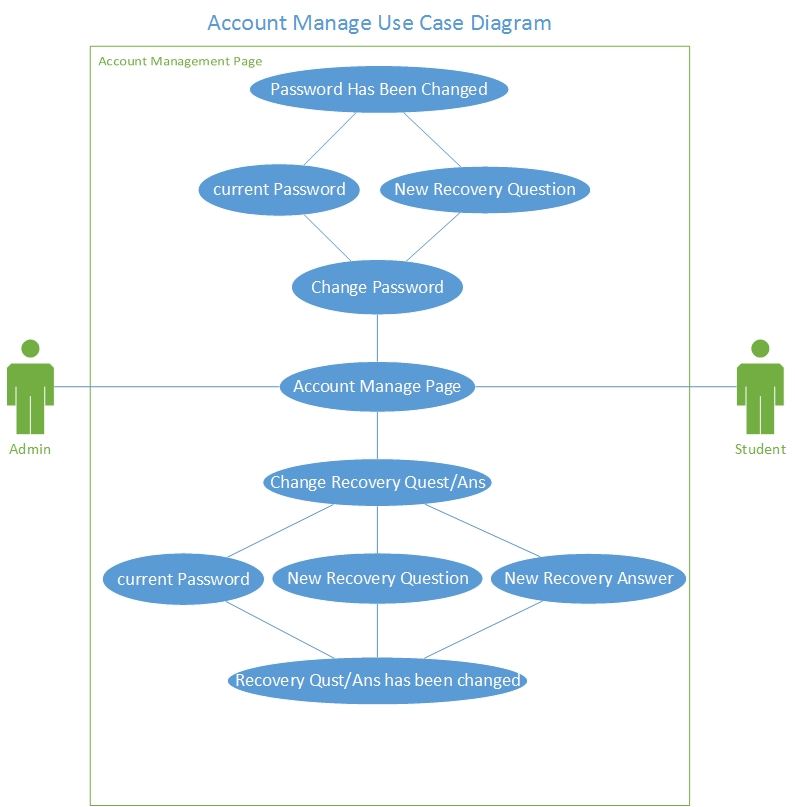


Figure 39 Account Management - Use Case Diagram

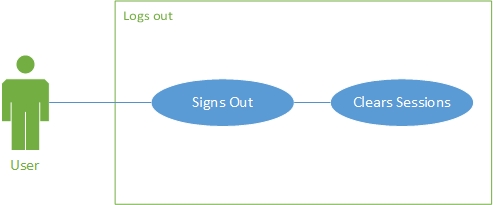


Figure 40 User Logs off - Use Case Diagram

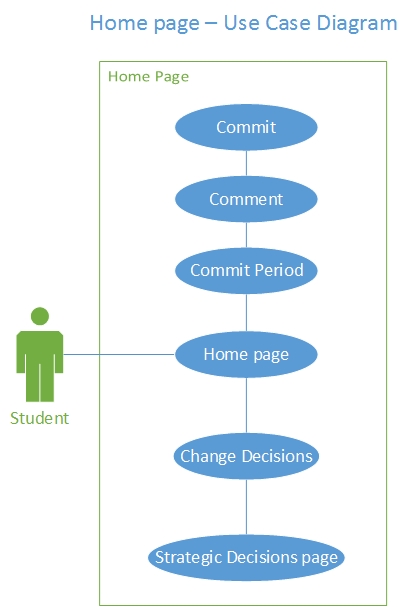


Figure 41 Student Home page - Use Case Diagram

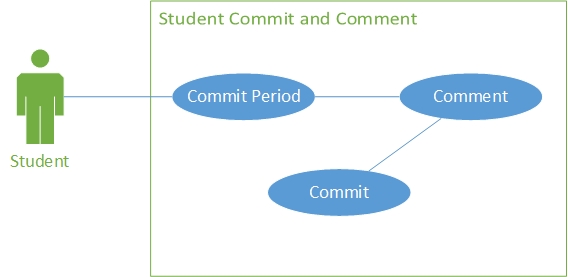


Figure 42 Student commits and Comments - Use Case Diagram

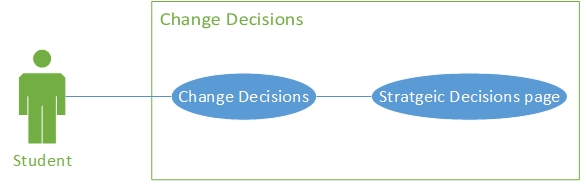


Figure 43 Student Changes their decisions - Use Case Diagram

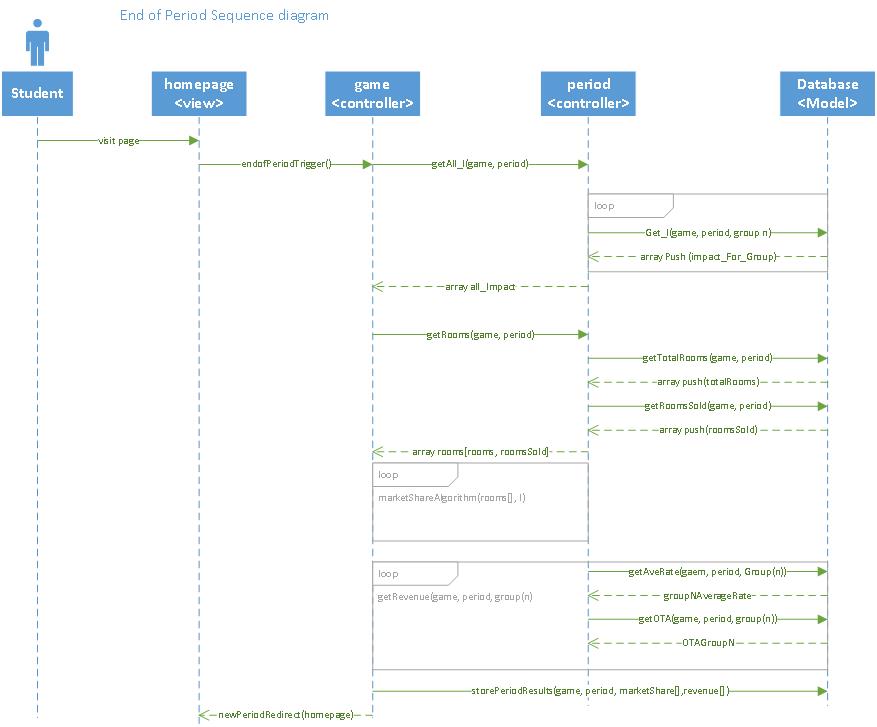


Figure 44 End of period Sequence diagram

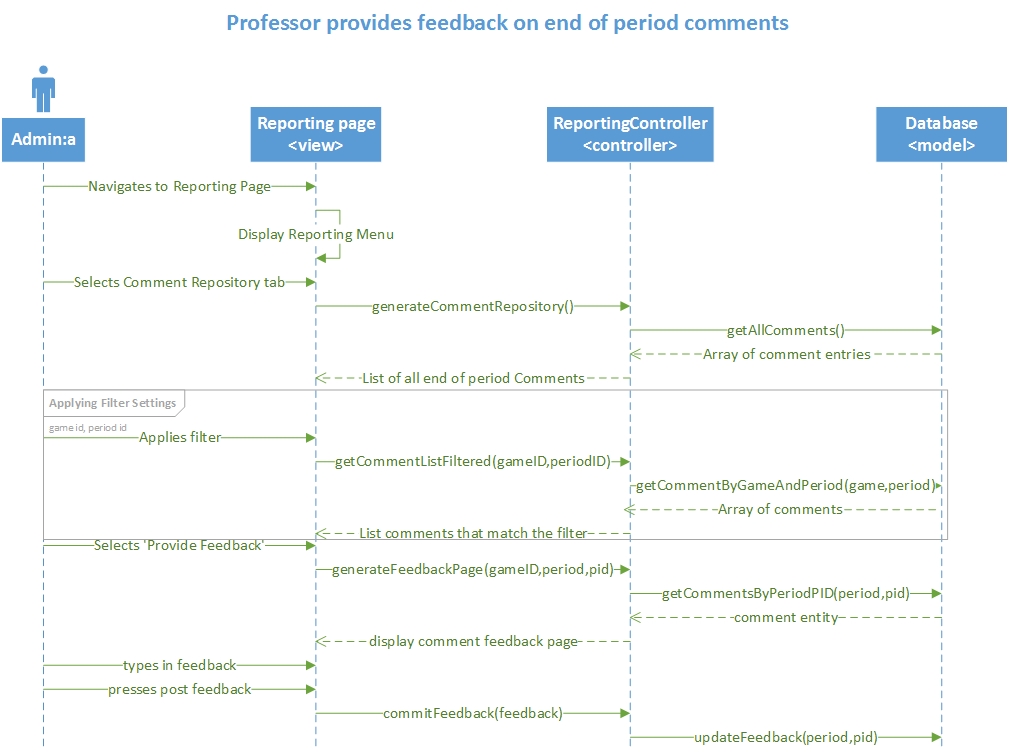


Figure 45 Professor provides feedback Sequence diagram

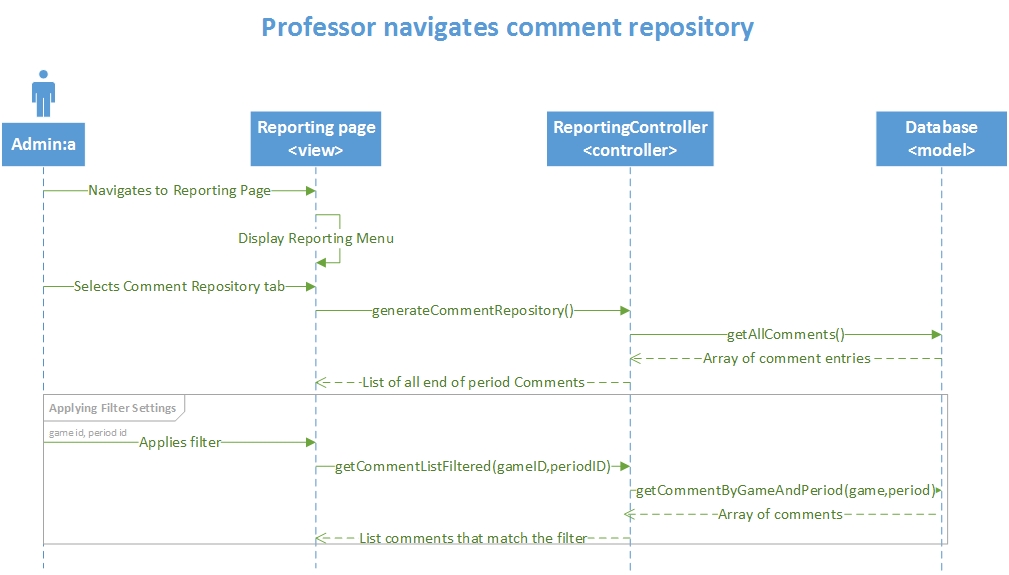


Figure 46 Professor navigates Comment Repository

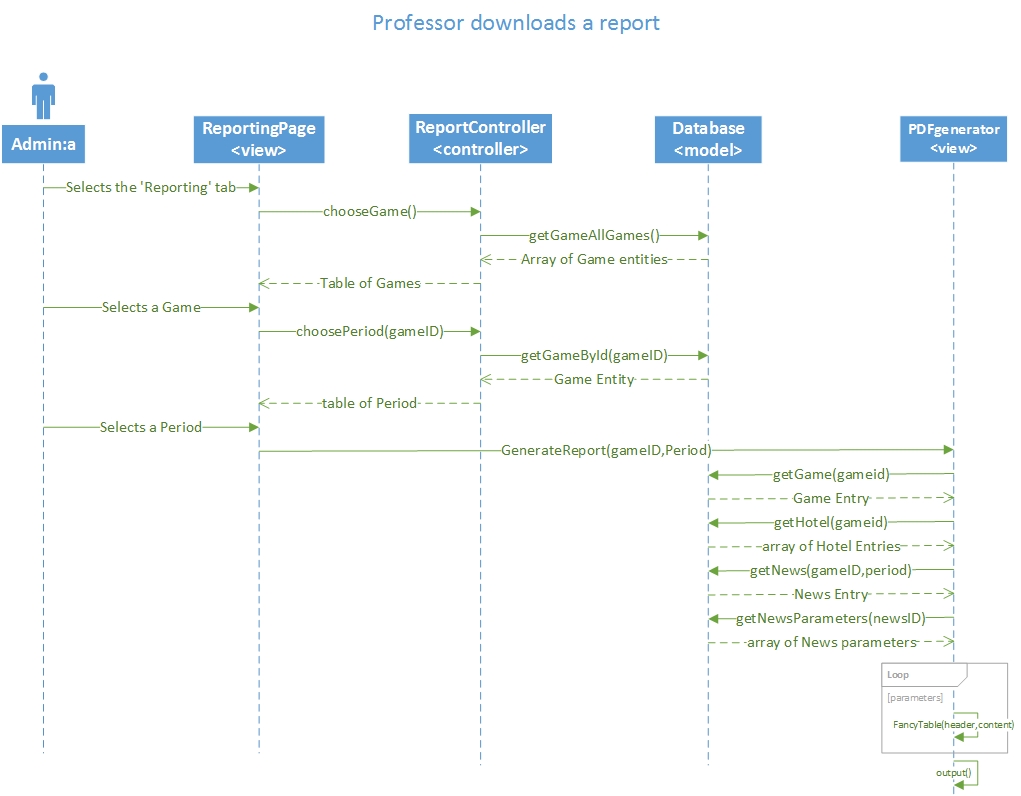


Figure 47 Professor Generates a report

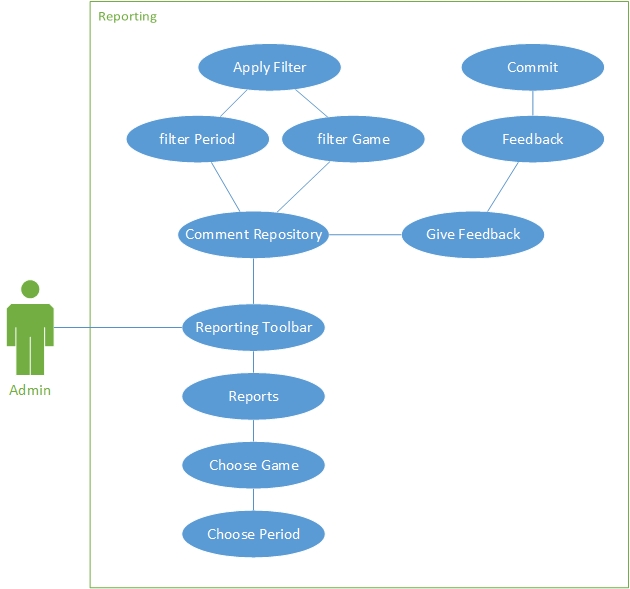


Figure 48 Use Case diagram Reporting



Figure 49 Student Views end of period feedback sequence diagram

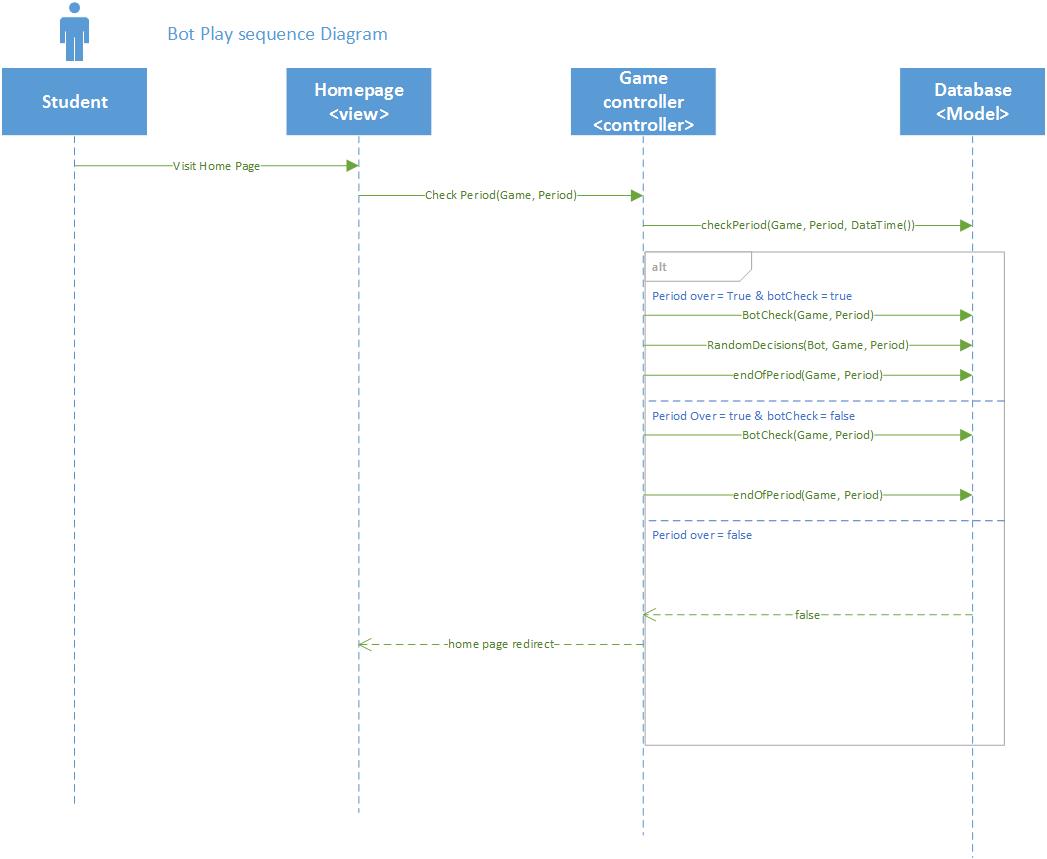


Figure 50 Bot play sequence Diagram

## Appendix B - User Interface Design

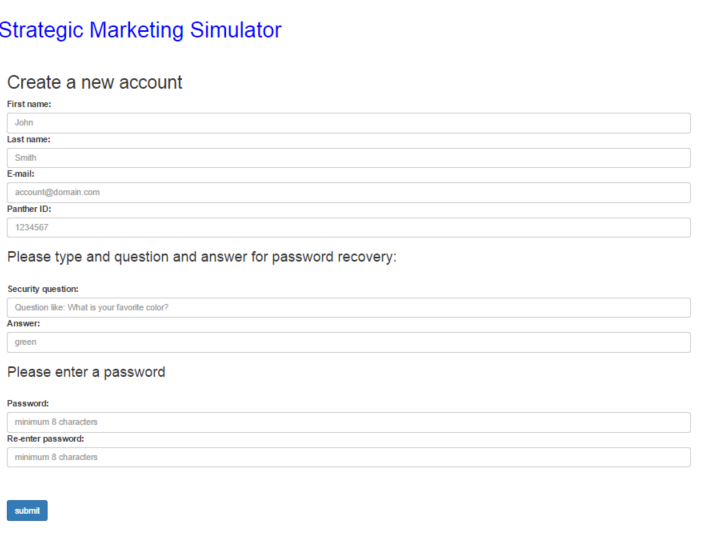
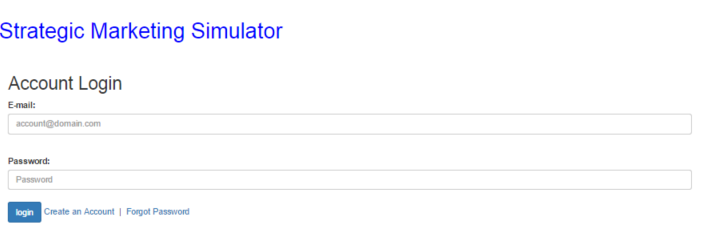
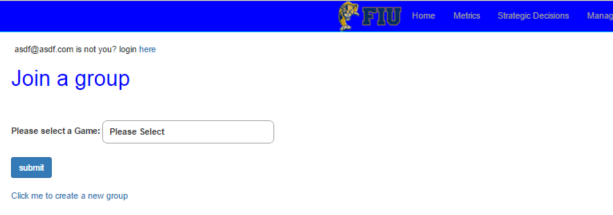
**Student account creation**

Figure 51



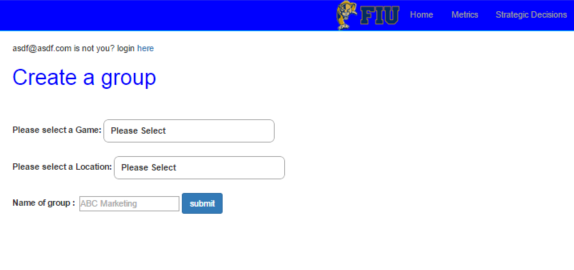
**Student Login**

Figure 52



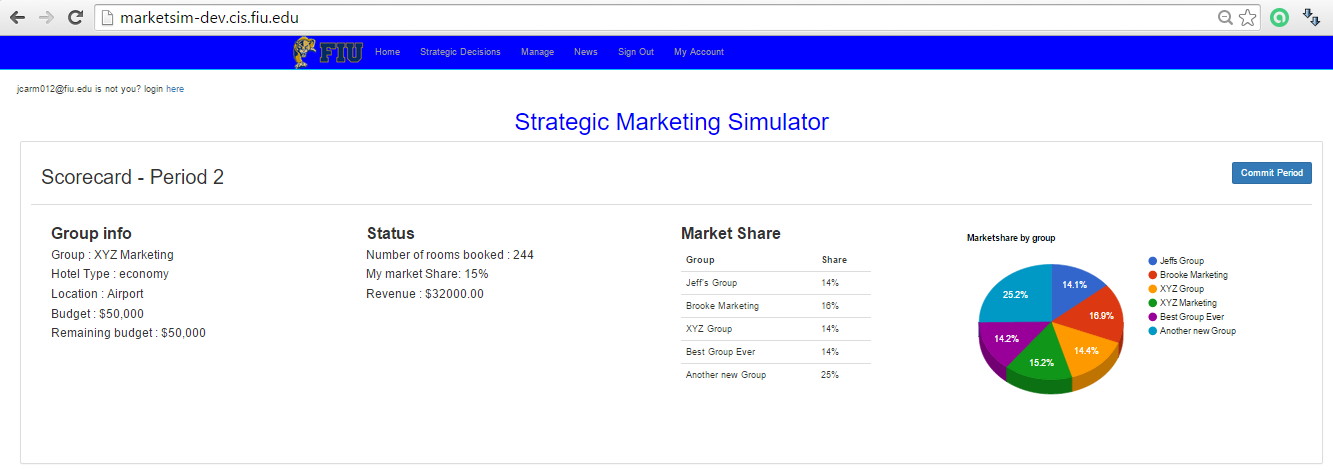
**Student Join Group**

Figure 53



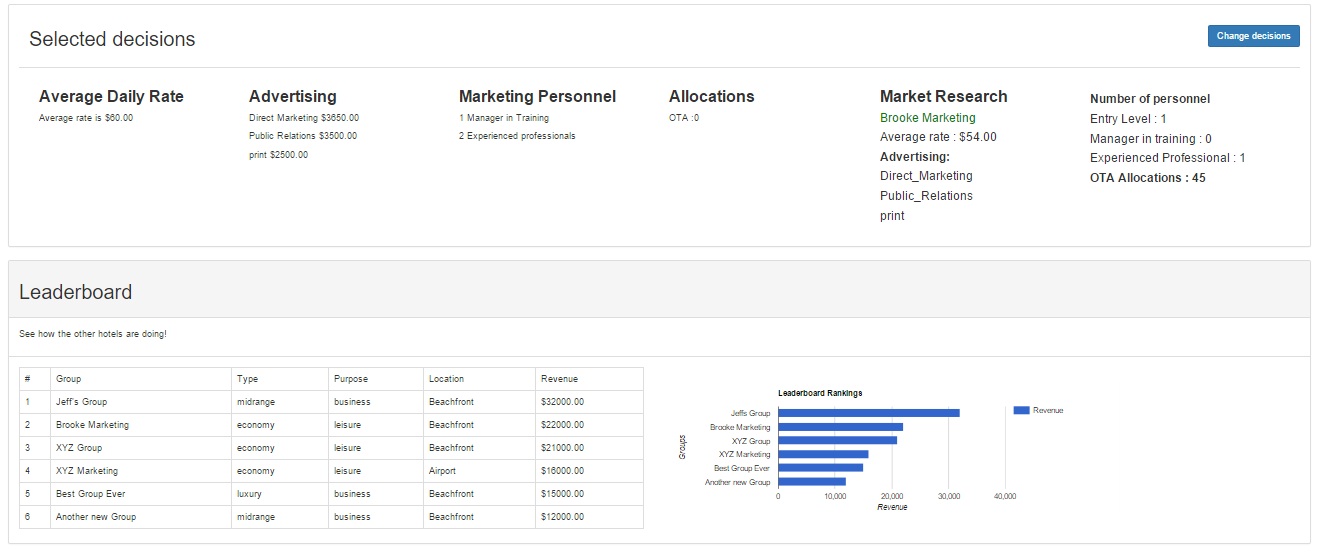
**Student create a group**

Figure 54



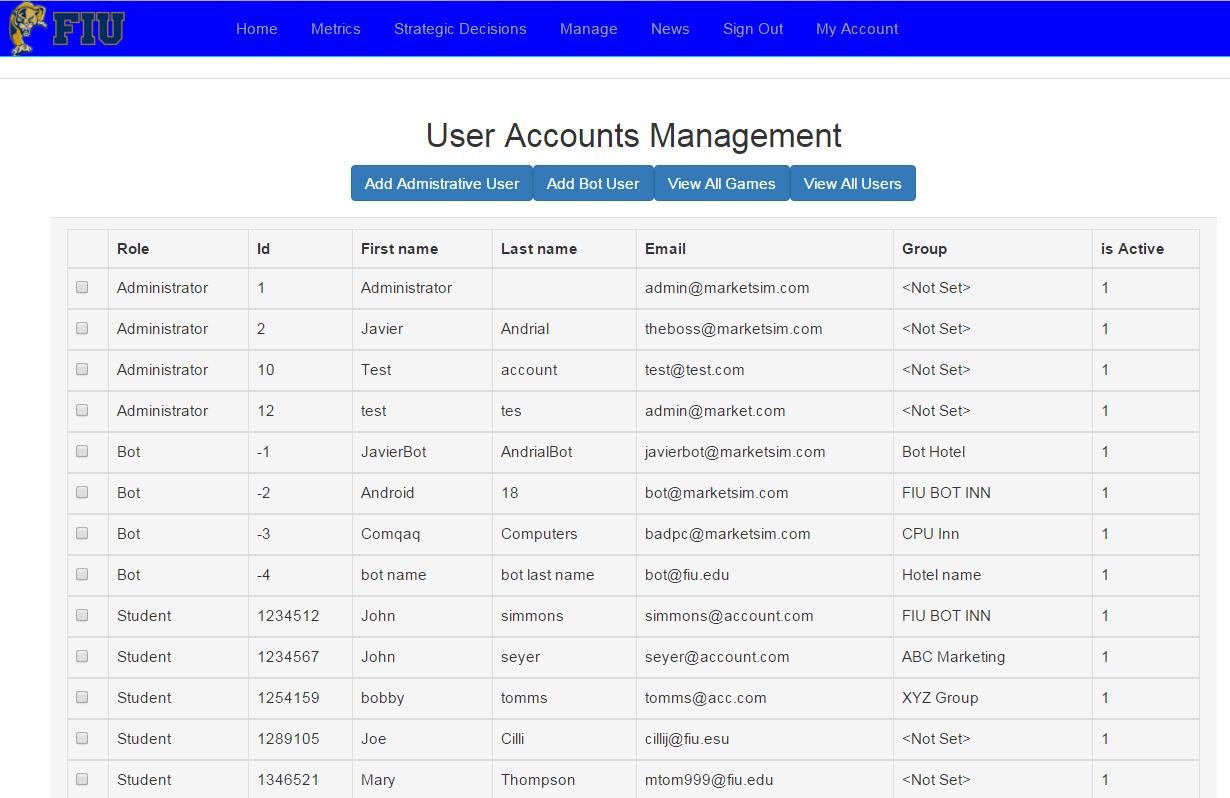
**Scorecard**

Figure 55



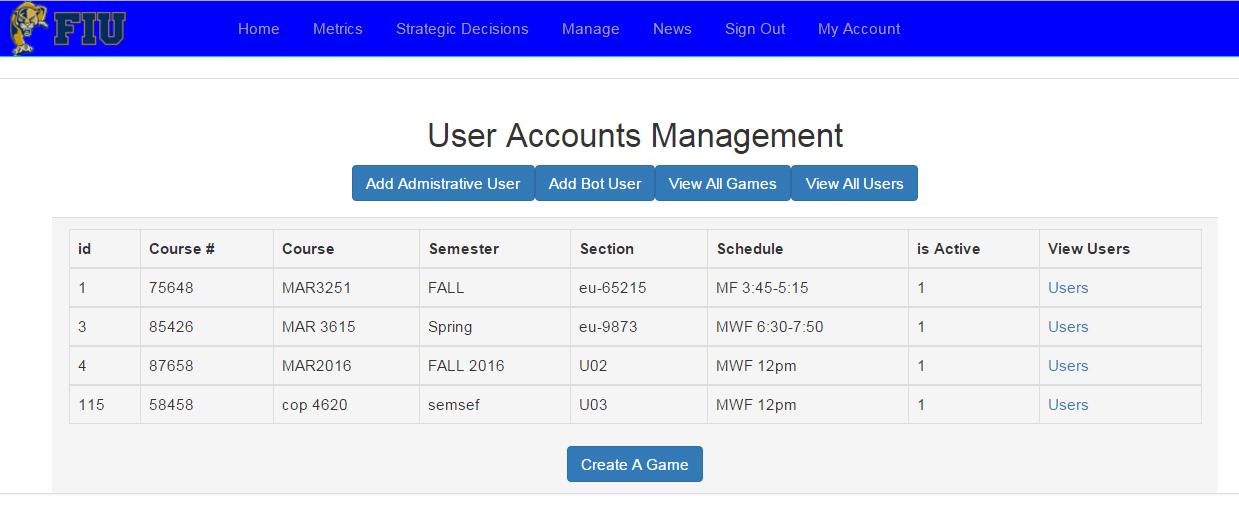
**leaderboard**

Figure 56



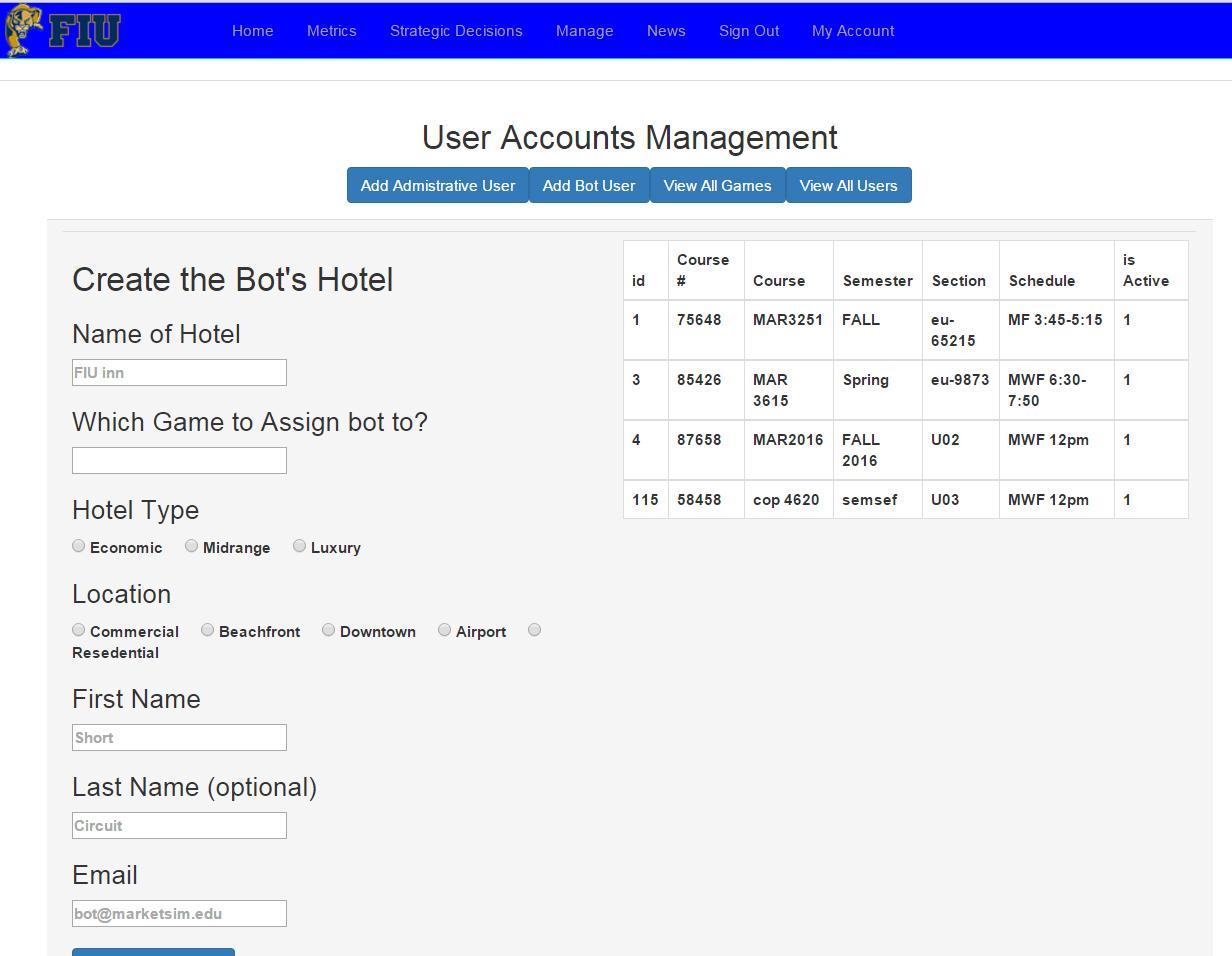
**Admin Views all users**

Figure 57 - This is the view where an Admin can view all the users in the System



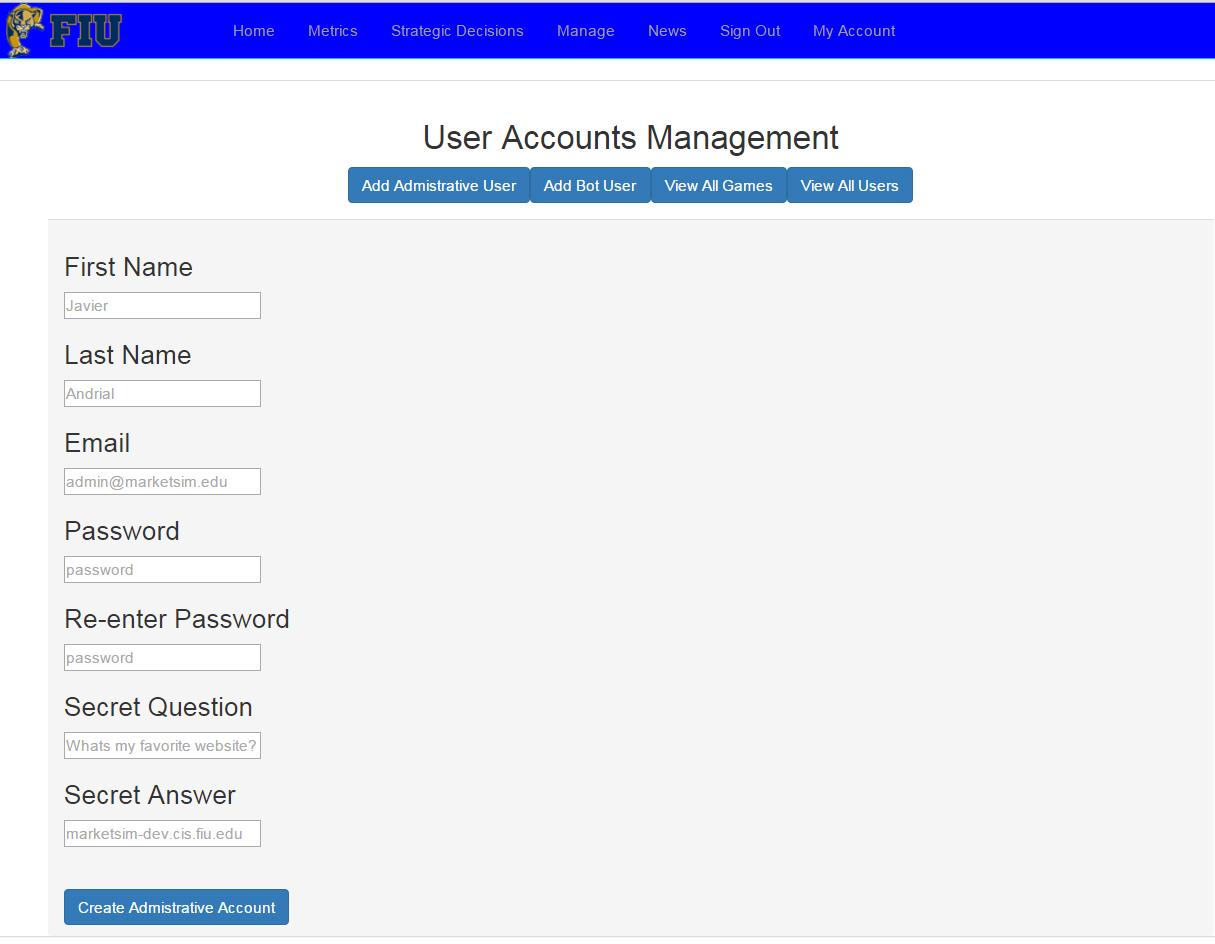
**Admin Views All Games**

Figure 58 - This is the view where an Admin can view all the games in the System



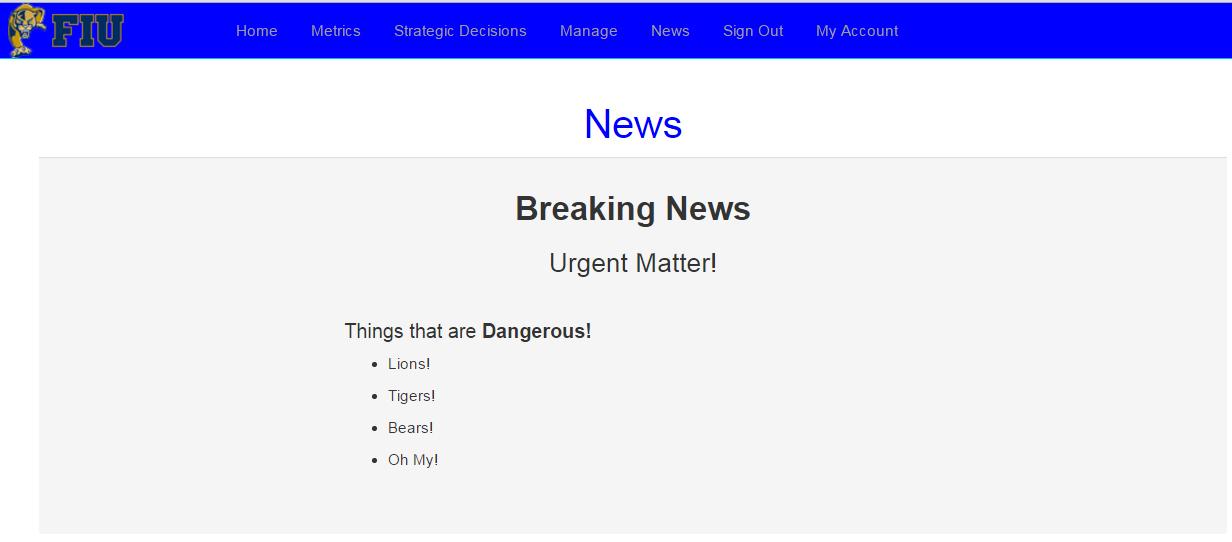
**Admin Adds a Bot**

Figure 59 - This is the view where an Admin can create a Bot Student account and its Hotel



**Admin Adds an Admin**

Figure 60 - This is the view where a logged in Admin can create a new Administrative account



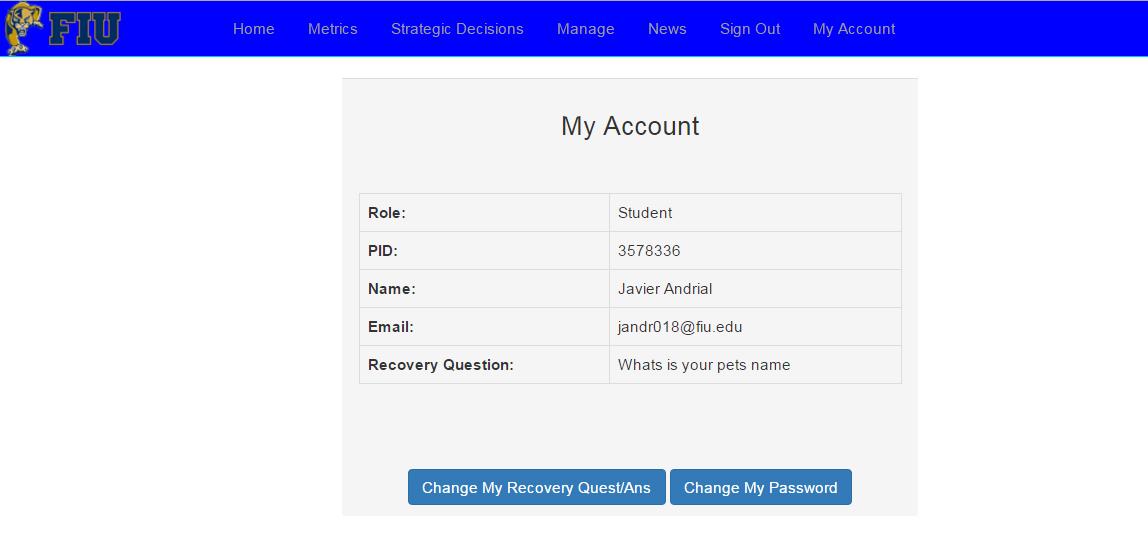
**Student View News**

Figure 61 - This is the view where a logged in Student views the News for their Games current period



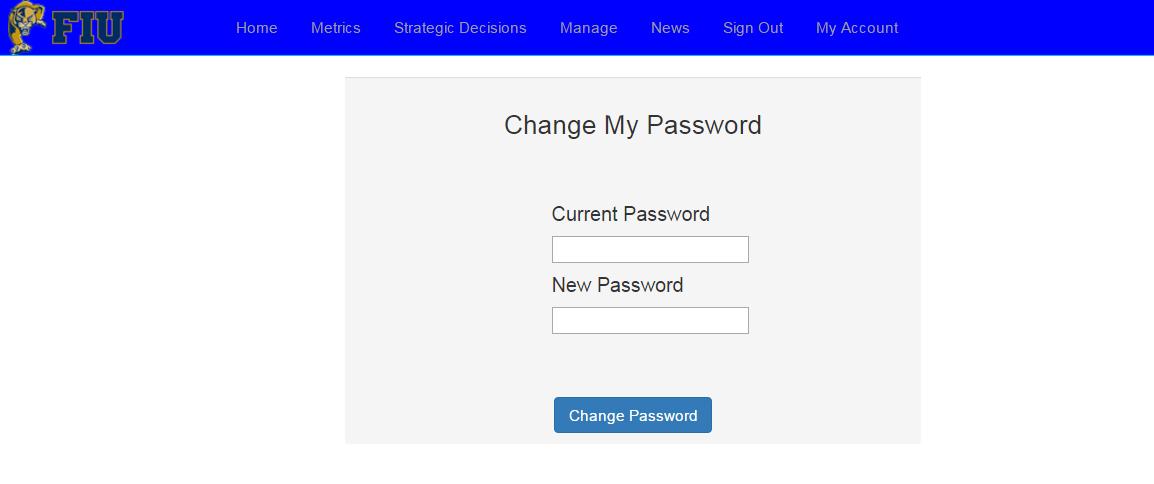
**Sign Out**

Figure 62 - This is the view where a logged in user logs out of the System



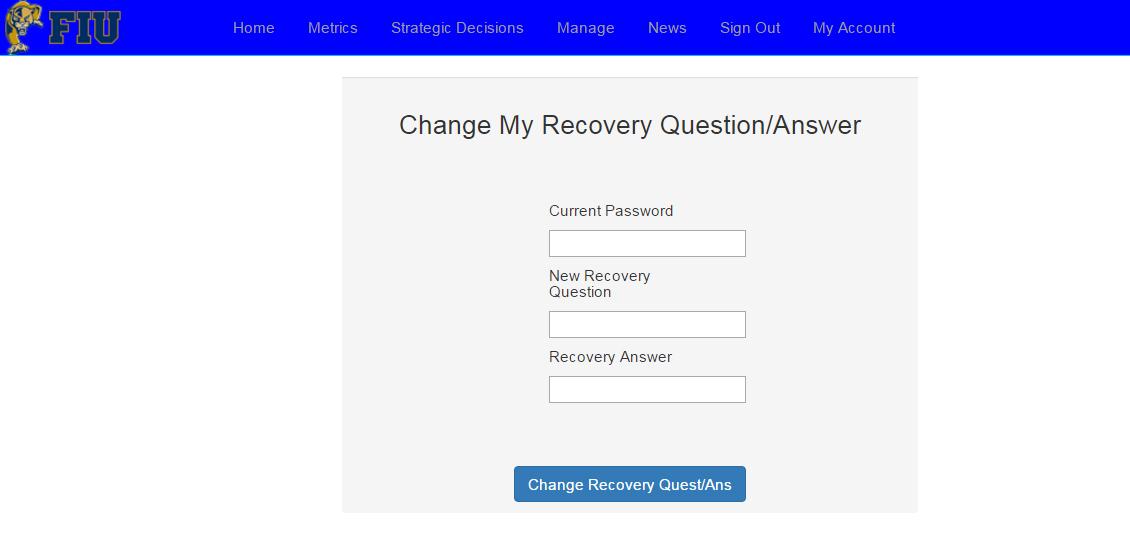
**Account Management**

Figure 63 - This is the view where a logged in user can view information about their account and change their password and recovery question and answer



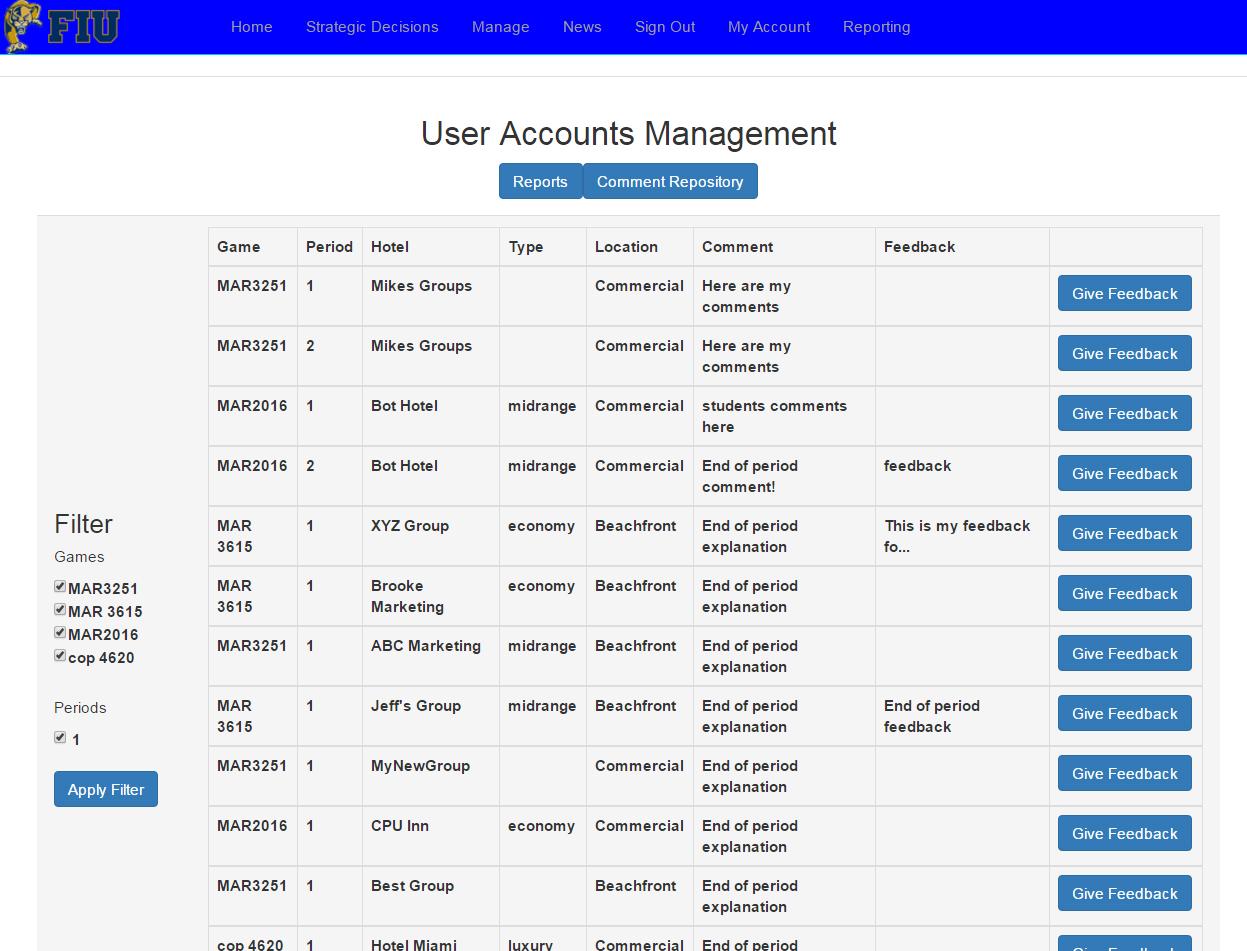
**Account Management Change Password**

Figure 64 - This is the view where a logged in user will change their account password



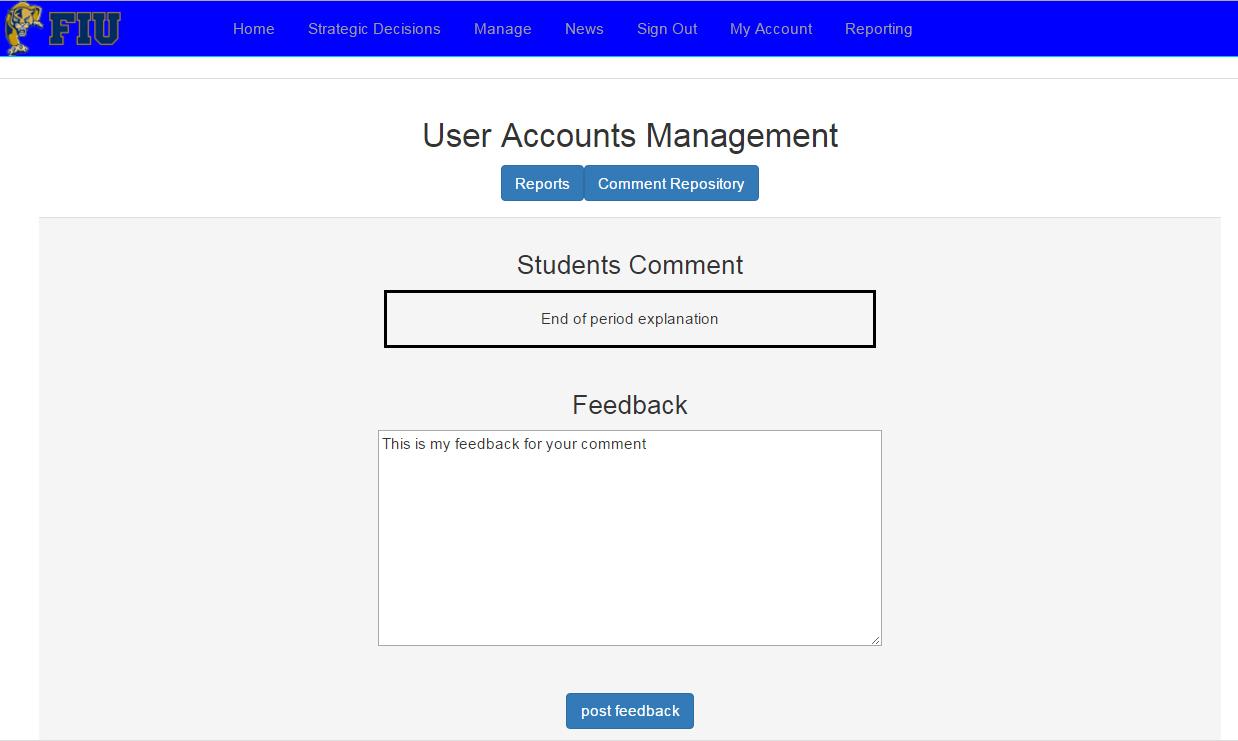
**Account Management Change Recovery Question/Answer**

Figure 65 - This is the view where a logged in user will change their account recovery question and answer



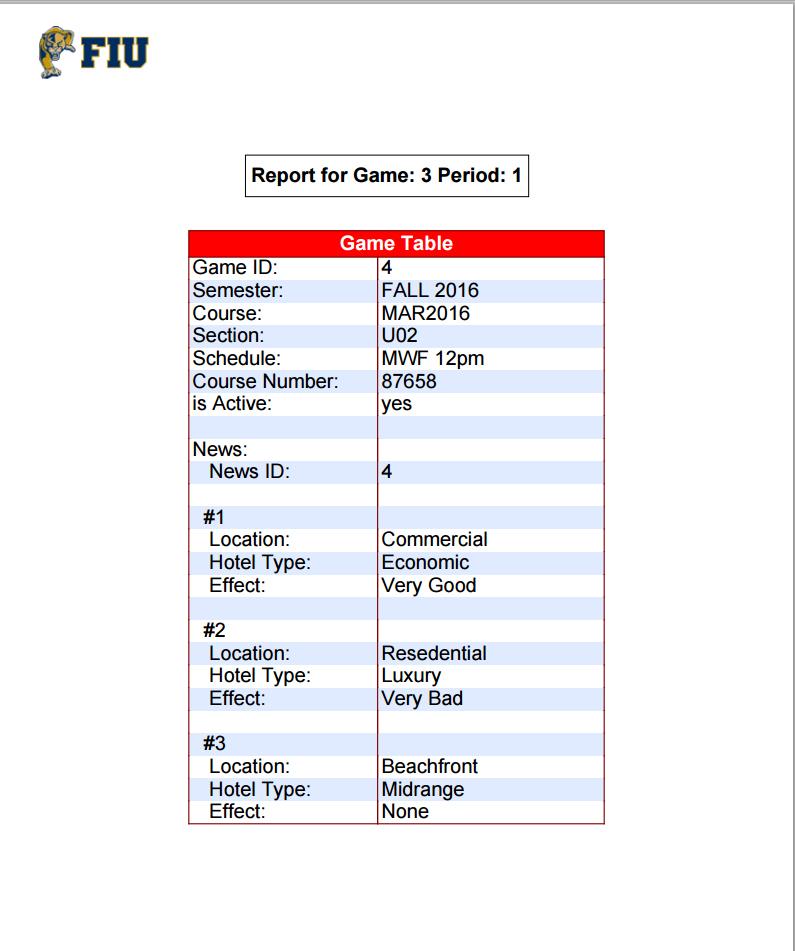
**Reporting - Comment Repository**

Figure 66 - This is the view where a logged in Admin can view all the end of period comments and provide feedback.



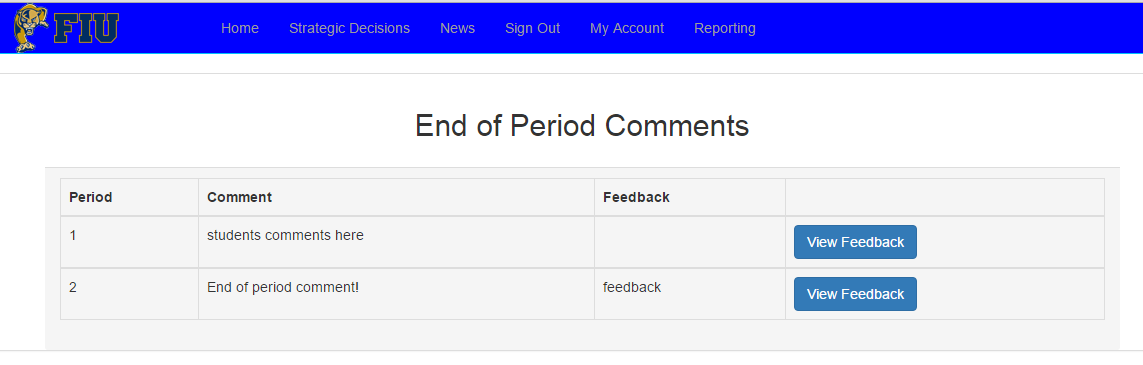
**Reporting - Comment Feedback**

Figure 67 - This is the view where a logged in Admin can enter in and commit their feedback for the student to view later.



**Reporting - PDF report**

Figure 68 - This is a generated detailed report for a game/period.



**Reporting - student Views Feedback**

Figure 69 - This is the view where a student can view their professor’s feedback on their end of period comments.

## Appendix C - Sprint Review Reports

**Sprint 1 Report**

**Date:** 9/10/2015

**Attendees:** Jeffrey Carman, Javier Andrial

**Discussed Topics:**

The goals of this sprint was to provide a foundation for the upcoming sprints.

To be able to begin this sprint, it was required that we meet up with the product owner to discuss what is expected from us and how we should go about developing it.

We were both able to achieve our goals of planning out the workload for the semester and setting up a foundation for the System.

**Discussed Topics:**

Considering the initial goals, cost estimates, and acceptance criteria, briefly explain what was achieved and what was not achieved in this sprint. Specify the reasons for not being able to finish all the work that was initially planned for this sprint. Specify if the product backlog was modified as a result of this meeting and if so, how.

**Sprint 2 Report**

**Date:** 9/24/15

**Attendees:** Javier Andrial, Jeffrey Carman

**Discussed Topics:**

The goals of this sprint was to provide initial environment a user will interact with the System.

To be able to begin this sprint, it was required that a Database be setup can store and recover user accounts.

We were both able to achieve our goals of creating student accounts, logging them into the System, as well as resetting their passwords in the effect they forget them.

**Sprint 3 Report**

**Date:** 10/8/15

**Attendees:** Javier Andrial, Jeffrey Carman

**Discussed Topics:**

The goals of this sprint was to provide a means for the administrator tools to control and monitor the entire system, as well as allowing students to join or create groups.

Some of the tools provided for the administrator are to add admin users, add bots and hotels, create games, view all users, and activate/deactivate users.

To be able to begin this sprint, it was required that the Database been completed and functional.

we were both able to achieve our goals on a timely manner.

**Sprint 4 Report**

**Date:** 10/22/15

**Attendees:** Javier Andrial, Jeffrey Carman

**Discussed Topics:**

**Sprint 5 Report**

**Date:** 11/5/15

**Attendees:** Javier Andrial, Jeffrey Carman

**Discussed Topics:**

**Sprint 6 Report**

**Date:** 11/19/15

**Attendees:** Javier Andrial, Jeffrey Carman

**Discussed Topics:**

The goals of this sprint was to provide a means for the administrator to review and provided feedback to the students as well as work on the math for transitioning the game from the current period to the next. To be able to begin this sprint, it was required that the Database has been completed and contains dummy information. we were both able to achieve our goals of pulling large amounts of information from the Database and manipulate it to acquire the desired output.

**Sprint 7 Report**

**Date:** 12/4/15

**Attendees:** Javier Andrial, Jeffrey Carman

**Discussed Topics:**

The goals of this sprint was to provide automated period cycling and bot functionality to the System. To be able to begin this sprint, it was required that the Database has been completed and contains dummy information. we were both able to achieve our goals.

## Appendix D - Sprint Retrospective Reports

**Sprint 1 Retrospective**

**Date:** 9/10/2015

**Attendees:** Jeffrey Carman, Javier Andrial

**Discussed Topics:**

We were able to set up the development environment and begin studying the web development in general because neither group member had any web development experience at the time. We set out to finish the create account and login user stories but we were unable to because we received the project designation halfway through the first week of the sprint and didn’t meet with the project owner to gather user stories until towards the end of that first week. We didn’t receive our ubuntu server designation until the second week of the sprint and it took the rest of that week to set up the development environment (LAMP server) and finish the documentation.

**Sprint 2 Retrospective**

**Date:** 9/28/2015

**Attendees:** Javier Andrial, Jeffrey Carman

**Discussed Topics:**

Sprint two we successfully set up the LAMP server and a database to be used for the System. We began developing must-have web pages for the System that no longer had any development restriction. These web pages was the Student Account Creation page and Password recovery page as well as creating the framework and implementing the design for some other pages, such as the Login page and Student Home page.

**Sprint 3 Retrospective**

**Date:** 10/10/2015

**Attendees:** Javier Andrial, Jeffrey Carman

**Discussed Topics:**

Sprint Three consisted of Admin Management page, Student Join Group page, and Student Create Group page.

Admin Management page provides access to, creation of Admin level account, ability to create Games, as well as view all users registered to a Game, Bot creation, as well as the creation of the Bot’s Hotel, View all Users, which displays all Users on the Database, and the ability to de-active/activate any User in the System with exception to the main Admin account.

**Sprint 4 Retrospective**

**Date:** 10/26/2015

**Attendees:** Javier Andrial, Jeffrey Carman

**Discussed Topics:**

Sprint Four consisted of the development of Strategic Decisions page for Students, Strategic Decisions page for Admins, News page for Students, and the News page for Admins. Some small changes were made to how Game Periods will interact with the System as well additional tables were created in that Database.

**Sprint 5 Retrospective**

**Date:**

**Attendees:** Javier Andrial, Jeffrey Carman

**Discussed Topics:**

**Sprint 6 Retrospective**

**Date:** 11/19/15

**Attendees:** Javier Andrial, Jeffrey Carman

**Discussed Topics:**

Sprint Six consisted of the development of the Reporting page for administrators, where an admin can View all end of period comments, as well as provide feedback to the comments for student to see later. Reporting page also consists of the report generator, which generates a detailed report about a selected Game and Period, which can be downloaded as a PDF. Jeffrey Carman, was implemented the End of Period Game Logic, which pulls variables from the previous period, computes a complex algorithm, and provides revenue, balance, market share, and other variables for the next game Period.

**Sprint 7 Retrospective**

**Date:** 12/4/15

**Attendees:** Javier Andrial, Jeffrey Carman

**Discussed Topics:**

Sprint seven consisted of the development of the Student Reporting page, Bot play functionality, and period end and start date cycling functionality. The Student Reporting page, implemented by Javier Andrial, allows a student to view a list of their end of period comments and few the professor's feedback of each of those comments. Bot Play and period cycling functionality was implemented by Jeffrey Carman. Bot Play, simulates a student user by selecting strategic decisions. Jeffrey’s Period Cycling functionality looks at a game’s current period's end date and time and compare it to the current time in the System. if the current time is later than the end date, the System will perform its end of period computations and cycle the game to the next period.

# 

# 

# 

# 

# 

# References

CSS <http://searchsoa.techtarget.com/definition/cascading-style-sheet-CSS>

Bootstrap 3.3.5 Released". http://blog.getbootstrap.com/2015/06/15/bootstrap-3-3-5-released/.

Zakas, Nicholas C. (2012). Professional JavaScript for Web Developers (3rd ed.). Wrox. ISBN 978-1-118-02669-4.

"jQuery: The write less, do more, JavaScript library". The jQuery Project.

http://www.w3schools.com/tags/tryit.asp?filename=tryhtml\_form\_checkbox

http://www.w3schools.com/php/php\_mysql\_create\_table.asp

<http://www.w3schools.com/php/php_mysql_create_table.asp>

<http://www.w3schools.com/bootstrap/bootstrap_button_groups.asp>

<http://www.tutorialrepublic.com/faq/populate-state-dropdown-based-on-selection-in-country-dropdown-using-jquery.php>

<http://www.w3schools.com/bootstrap/bootstrap_panels.asp>

https://developers.google.com/chart/interactive/docs/quick\_start