**Big Data**

**Gamification**

**Final Report**

Ali Tarraf

Ali Al-mugoter

Steve Karim

August 5th, 2019

### Acknowledgment(s)

This document contains all information on the development of the project “Gamification of Big Data” for Sargon Partners Inc.

The entirety of the Development Life Cycle is provided.

Instructions for Installation of the web application are provided.

User Interface description is provided.

Special thanks to Sargon Partners Inc for their continued communication and help with the project to insure all requirements were met.

[Acknowledgment(s)](#_heading=h.gjdgxs) **1**

[**Introduction**](#_heading=h.30j0zll) **[5](#_heading=h.30j0zll)**

[**Overview**](#_heading=h.1fob9te) **[6](#_heading=h.1fob9te)**

[**Requirements**](#_heading=h.3znysh7) **[6](#_heading=h.3znysh7)**

[**Functional Model and Description**](#_heading=h.2et92p0) **[6](#_heading=h.2et92p0)**

[Function 1](#_heading=h.tyjcwt) [6](#_heading=h.tyjcwt)

[Function 2](#_heading=h.3dy6vkm) [6](#_heading=h.3dy6vkm)

[Function 3](#_heading=h.1t3h5sf) [7](#_heading=h.1t3h5sf)

[Function 4](#_heading=h.4d34og8) [7](#_heading=h.4d34og8)

[Function 5](#_heading=h.2s8eyo1) [8](#_heading=h.2s8eyo1)

[Function 6](#_heading=h.17dp8vu) [8](#_heading=h.17dp8vu)

[Function 7](#_heading=h.3rdcrjn) [8](#_heading=h.3rdcrjn)

[Function 8](#_heading=h.26in1rg) [8](#_heading=h.26in1rg)

[Function 9](#_heading=h.lnxbz9) [9](#_heading=h.lnxbz9)

[Function 10](#_heading=h.35nkun2) [9](#_heading=h.35nkun2)

[**Use Case Diagrams**](#_heading=h.1ksv4uv) **[10](#_heading=h.1ksv4uv)**

[System Use Case](#_heading=h.44sinio) [10](#_heading=h.44sinio)

[Use Case 1, 2, 3](#_heading=h.2jxsxqh) [10](#_heading=h.2jxsxqh)

[Use Case 8](#_heading=h.z337ya) [11](#_heading=h.z337ya)

[Use Case 9](#_heading=h.3j2qqm3) [12](#_heading=h.3j2qqm3)

[Use Case 10](#_heading=h.1y810tw) [12](#_heading=h.1y810tw)

[**Sequence Diagrams**](#_heading=h.4i7ojhp) **[13](#_heading=h.4i7ojhp)**

[View Team History](#_heading=h.2xcytpi) [13](#_heading=h.2xcytpi)

[View Weekly Ranking](#_heading=h.1ci93xb) [13](#_heading=h.1ci93xb)

[View All Time Ranking](#_heading=h.3whwml4) [14](#_heading=h.3whwml4)

[Create Team](#_heading=h.2bn6wsx) [14](#_heading=h.2bn6wsx)

[Edit Team](#_heading=h.qsh70q) [15](#_heading=h.qsh70q)

[Delete Team](#_heading=h.3as4poj) [15](#_heading=h.3as4poj)

[Create Employee Account](#_heading=h.1pxezwc) [16](#_heading=h.1pxezwc)

[Deactivate Employee Account](#_heading=h.49x2ik5) [16](#_heading=h.49x2ik5)

[Edit Metrics](#_heading=h.2p2csry) [17](#_heading=h.2p2csry)

[Change Password](#_heading=h.147n2zr) [17](#_heading=h.147n2zr)

[**Communication Diagrams**](#_heading=h.3o7alnk) **[18](#_heading=h.3o7alnk)**

[View Team History](#_heading=h.23ckvvd) [18](#_heading=h.23ckvvd)

[View Weekly Ranking](#_heading=h.ihv636) [18](#_heading=h.ihv636)

[View All Time Ranking](#_heading=h.32hioqz) [18](#_heading=h.32hioqz)

[Create Team](#_heading=h.1hmsyys) [18](#_heading=h.1hmsyys)

[Delete Team](#_heading=h.41mghml) [19](#_heading=h.41mghml)

[Edit Team](#_heading=h.2grqrue) [19](#_heading=h.2grqrue)

[Create Employee Account](#_heading=h.vx1227) [20](#_heading=h.vx1227)

[Deactivate Employee Account](#_heading=h.3fwokq0) [20](#_heading=h.3fwokq0)

[Edit Metrics](#_heading=h.1v1yuxt) [20](#_heading=h.1v1yuxt)

[Change Password](#_heading=h.4f1mdlm) [21](#_heading=h.4f1mdlm)

[**Description for Software Behavior**](#_heading=h.2u6wntf) **[21](#_heading=h.2u6wntf)**

[Events](#_heading=h.19c6y18) [22](#_heading=h.19c6y18)

[State](#_heading=h.3tbugp1) [23](#_heading=h.3tbugp1)

[**State Transition Diagrams**](#_heading=h.28h4qwu) **[24](#_heading=h.28h4qwu)**

[Manage Metrics](#_heading=h.37m2jsg) [24](#_heading=h.37m2jsg)

[Login](#_heading=h.46r0co2) [24](#_heading=h.46r0co2)

[Change Password](#_heading=h.2lwamvv) [25](#_heading=h.2lwamvv)

[Manage Teams](#_heading=h.111kx3o) [26](#_heading=h.111kx3o)

[Managing Employees](#_heading=h.3l18frh) [27](#_heading=h.3l18frh)

[**Activity Diagrams**](#_heading=h.206ipza) **[28](#_heading=h.206ipza)**

[Update Asset](#_heading=h.4k668n3) [28](#_heading=h.4k668n3)

[Update Ranking](#_heading=h.1egqt2p) [29](#_heading=h.1egqt2p)

[Manage Employee](#_heading=h.2dlolyb) [30](#_heading=h.2dlolyb)

[Manage Team](#_heading=h.sqyw64) [31](#_heading=h.sqyw64)

[**Restrictions, Limitations, and Constraints**](#_heading=h.3cqmetx) **[31](#_heading=h.3cqmetx)**

[Restrictions:](#_heading=h.1rvwp1q) [31](#_heading=h.1rvwp1q)

[Limitations:](#_heading=h.4bvk7pj) [31](#_heading=h.4bvk7pj)

[Constraints:](#_heading=h.2r0uhxc) [31](#_heading=h.2r0uhxc)

[**Software Design**](#_heading=h.1664s55) **[32](#_heading=h.1664s55)**

[**Data Model**](#_heading=h.3q5sasy) **[32](#_heading=h.3q5sasy)**

[ERD](#_heading=h.25b2l0r) [33](#_heading=h.25b2l0r)

[Class Diagram](#_heading=h.kgcv8k) [34](#_heading=h.kgcv8k)

[**Data Dictionary**](#_heading=h.34g0dwd) **[34](#_heading=h.34g0dwd)**

[**Data Design**](#_heading=h.43ky6rz) **[37](#_heading=h.43ky6rz)**

[Internal Software Data Structure](#_heading=h.2iq8gzs) [37](#_heading=h.2iq8gzs)

[Global Data Structure](#_heading=h.xvir7l) [38](#_heading=h.xvir7l)

[Temporary Data Structure](#_heading=h.3hv69ve) [39](#_heading=h.3hv69ve)

[Database Description](#_heading=h.1x0gk37) [39](#_heading=h.1x0gk37)

[**Program Structure**](#_heading=h.4h042r0) **[39](#_heading=h.4h042r0)**

[Description for Component Manage Team](#_heading=h.2w5ecyt) [39](#_heading=h.2w5ecyt)

[Description for Component team detail](#_heading=h.1baon6m) [41](#_heading=h.1baon6m)

[Description for Component manage employees](#_heading=h.3vac5uf) [42](#_heading=h.3vac5uf)

[Processing narrative (PSPEC) for component main-page](#_heading=h.2afmg28) [44](#_heading=h.2afmg28)

[Processing narrative (PSPEC) for component team-rank-history](#_heading=h.pkwqa1) [45](#_heading=h.pkwqa1)

[Processing narrative (PSPEC) for component n](#_heading=h.39kk8xu) [46](#_heading=h.39kk8xu)

[Software Interface Description](#_heading=h.1opuj5n) [46](#_heading=h.1opuj5n)

[**User Interface Design**](#_heading=h.48pi1tg) **[47](#_heading=h.48pi1tg)**

[Login page](#_heading=h.2nusc19) [47](#_heading=h.2nusc19)

[Main page](#_heading=h.1302m92) [48](#_heading=h.1302m92)

[Profile page](#_heading=h.3mzq4wv) [48](#_heading=h.3mzq4wv)

[Manage Teams](#_heading=h.2250f4o) [49](#_heading=h.2250f4o)

[Manage assets](#_heading=h.haapch) [50](#_heading=h.haapch)

[Manage employees](#_heading=h.319y80a) [50](#_heading=h.319y80a)

[Team detail](#_heading=h.1gf8i83) [51](#_heading=h.1gf8i83)

[Interface design rules](#_heading=h.40ew0vw) [52](#_heading=h.40ew0vw)

[Components available](#_heading=h.2fk6b3p) [52](#_heading=h.2fk6b3p)

[Performance bounds](#_heading=h.upglbi) [52](#_heading=h.upglbi)

[Identification of critical components](#_heading=h.3ep43zb) [52](#_heading=h.3ep43zb)

[**Implementation Details**](#_heading=h.1tuee74) **[53](#_heading=h.1tuee74)**

[Gamification Service](#_heading=h.4du1wux) [53](#_heading=h.4du1wux)

[Gamification WebApi](#_heading=h.2szc72q) [54](#_heading=h.2szc72q)

[Database](#_heading=h.184mhaj) [54](#_heading=h.184mhaj)

[ShardManager](#_heading=h.3s49zyc) [54](#_heading=h.3s49zyc)

[Shard](#_heading=h.279ka65) [58](#_heading=h.279ka65)

[**Testing/SQA**](#_heading=h.meukdy) **[59](#_heading=h.meukdy)**

[Statement of Scope](#_heading=h.36ei31r) [59](#_heading=h.36ei31r)

[Major Constraints](#_heading=h.1ljsd9k) [59](#_heading=h.1ljsd9k)

[Test Plan](#_heading=h.45jfvxd) [59](#_heading=h.45jfvxd)

[Software (SCI’s) to be tested](#_heading=h.2koq656) [60](#_heading=h.2koq656)

[Testing Strategy](#_heading=h.zu0gcz) [61](#_heading=h.zu0gcz)

[Unit Testing](#_heading=h.3jtnz0s) [61](#_heading=h.3jtnz0s)

[Integration Testing](#_heading=h.1yyy98l) [61](#_heading=h.1yyy98l)

[Validation testing](#_heading=h.4iylrwe) [61](#_heading=h.4iylrwe)

[High-order testing](#_heading=h.2y3w247) [61](#_heading=h.2y3w247)

[Testing resources and staffing](#_heading=h.1d96cc0) [62](#_heading=h.1d96cc0)

[Test work products](#_heading=h.3x8tuzt) [62](#_heading=h.3x8tuzt)

[Test record keeping](#_heading=h.2ce457m) [62](#_heading=h.2ce457m)

[Test metrics](#_heading=h.rjefff) [62](#_heading=h.rjefff)

[Testing tools and environment](#_heading=h.3bj1y38) [63](#_heading=h.3bj1y38)

[Test schedule](#_heading=h.1qoc8b1) [63](#_heading=h.1qoc8b1)

[Testing procedure](#_heading=h.4anzqyu) [64](#_heading=h.4anzqyu)

[Testing resources and staffing](#_heading=h.14ykbeg) [70](#_heading=h.14ykbeg)

[Test Work Products](#_heading=h.3oy7u29) [70](#_heading=h.3oy7u29)

[Test record keeping and test log](#_heading=h.243i4a2) [73](#_heading=h.243i4a2)

[SQA Task Overview](#_heading=h.j8sehv) [76](#_heading=h.j8sehv)

[SQA Resources](#_heading=h.338fx5o) [77](#_heading=h.338fx5o)

[Reviews and Audits](#_heading=h.1idq7dh) [78](#_heading=h.1idq7dh)

[SQA Audits](#_heading=h.42ddq1a) [83](#_heading=h.42ddq1a)

[Problem Reporting mechanisms](#_heading=h.2hio093) [83](#_heading=h.2hio093)

[Responsibilities](#_heading=h.wnyagw) [83](#_heading=h.wnyagw)

[Data collection and evaluation](#_heading=h.3gnlt4p) [83](#_heading=h.3gnlt4p)

[Statistical SQA](#_heading=h.1vsw3ci) [83](#_heading=h.1vsw3ci)

[Goal and objectives of Software Process Improvement (SPI)](#_heading=h.4fsjm0b) [85](#_heading=h.4fsjm0b)

[SPI tasks and responsibilities](#_heading=h.2uxtw84) [85](#_heading=h.2uxtw84)

[Software Configuration Management Overview](#_heading=h.1a346fx) [85](#_heading=h.1a346fx)

[SQA Tools, Techniques, Methods](#_heading=h.3u2rp3q) [86](#_heading=h.3u2rp3q)

[**Future Maintenance Suggestions**](#_heading=h.2981zbj) **[86](#_heading=h.2981zbj)**

[**Delivery/Acceptance Statement**](#_heading=h.odc9jc) **[86](#_heading=h.odc9jc)**

[**References & Bibliography**](#_heading=h.38czs75) **[86](#_heading=h.38czs75)**

[Client Api:](#_heading=h.1nia2ey) [86](#_heading=h.1nia2ey)

[Client Software demonstration:](#_heading=h.47hxl2r) [86](#_heading=h.47hxl2r)

[Gamification Definition:](#_heading=h.2mn7vak) [86](#_heading=h.2mn7vak)

[Gamification Software:](#_heading=h.11si5id) [86](#_heading=h.11si5id)

[Kpi: https://www.google.com/search?q=key+performance+indicators&oq=key+performance+indicators&aqs=chrome..69i57j0l5.456j0j7&sourceid=chrome&ie=UTF-8](#_heading=h.3ls5o66) [86](#_heading=h.3ls5o66)

[Data Dictionary:](#_heading=h.20xfydz) [86](#_heading=h.20xfydz)

[Database:](#_heading=h.4kx3h1s) [87](#_heading=h.4kx3h1s)

[Backend:](#_heading=h.302dr9l) [87](#_heading=h.302dr9l)

[Design patterns used for web application:](#_heading=h.1f7o1he) [88](#_heading=h.1f7o1he)

[CQRS:](#_heading=h.3z7bk57) [88](#_heading=h.3z7bk57)

[SQA:](#_heading=h.2eclud0) [88](#_heading=h.2eclud0)

[SQL optimization: https://docs.microsoft.com/en-us/previous-versions/sql/sql-server-2008-r2/ms191227(v=sql.105)](#_heading=h.thw4kt) [89](#_heading=h.thw4kt)

[**Appendix A - User Manual**](#_heading=h.3dhjn8m) **[89](#_heading=h.3dhjn8m)**

[Setting up the environment:](#_heading=h.1smtxgf) [89](#_heading=h.1smtxgf)

[How to clone the project](#_heading=h.4cmhg48) [89](#_heading=h.4cmhg48)

[What tools you need](#_heading=h.2rrrqc1) [89](#_heading=h.2rrrqc1)

[**Appendix B - Program Listing**](#_heading=h.16x20ju) **[93](#_heading=h.16x20ju)**

[**Appendix C - Team Member Resumes**](#_heading=h.3qwpj7n) **[94](#_heading=h.3qwpj7n)**

[Ali Al-mugoter](#_heading=h.261ztfg) [94](#_heading=h.261ztfg)

[Steve Karim](#_heading=h.l7a3n9) [94](#_heading=h.l7a3n9)

[Ali Tarraf](#_heading=h.356xmb2) [95](#_heading=h.356xmb2)

[**Appendix D - Project Plan & Log Book**](#_heading=h.1kc7wiv) **[99](#_heading=h.1kc7wiv)**

[Project task set](#_heading=h.44bvf6o) [99](#_heading=h.44bvf6o)

[**Appendix E - Project Demo Notes**](#_heading=h.2jh5peh) **100**

[**Appendix F - Final Presentation Slides**](#_heading=h.ymfzma) **101**

### Introduction

The purpose of this software is the gamification of big data. The data that will be analyzed is data received from an inventory software, which keeps inventory of equipment that is used by rescue teams. These teams should utilize the inventory software whenever they have jobs. The data will be monitored to check if the software is being utilized, and if inventory is being returned. The data will be analyzed, and a scoring system will rank the teams, based on how much they utilized the software and if their equipment was returned. This will increase productivity.

### Overview

The software is a web application. This software is going to be used by employees that are divided into teams. The objective is to boost the teams’ productivity by making them compete against each other to rise to the top of the ranking. The software is able to track and measure productivity accurately, Incentives could be put in place as an additional motivation to become more productive. The goal is to ultimately be able to utilize the software to solve a large variety of similar problems by implementing the gamification method.

Users (Employees) will be able to login and view their teams weekly or all time ranking amongst other teams. They can view analytics of their teams history and see what needs to be improved to move up in the ranks.

Administrators will be able to view all teams and their rank in the organization. The admin has the ability to create/update/delete employee accounts. Admins can add/update/delete teams. They also have the ability to add employees to multiple teams and remove employees from teams. Admins can also change the metrics of the ranking system by editing the weight of assets.

### Requirements

#### Functional Model and Description

##### Function 1

* + **Description**: User views teams weekly ranking.
  + **Use Case Name**: Viewing Weekly Ranking
  + **Actors**: User, Time
  + **Preconditions**: User must log in and be verified.
  + **Triggers**: Time triggers the weekly ranking to be updated daily. User selects the weekly viewing option.
  + **Scenario Description**: User logs in and selects view weekly ranking. The ranking for all teams for the week is displayed to the user. the scores are updated daily and then reset every week.
  + **Post Conditions**: User is able to see the teams ranking for the week.
  + **Exceptions**: If two groups have the same ranking it will be indicated that there is a tie. If there is no data in the backend nothing will be displayed.

##### Function 2

* + **Description**: Displaying over all rankings for company teams.
  + **Use Case Name**: Viewing all time ranking
  + **Actors**: User, Time
  + **Preconditions**: User must log onto the website and be authenticated. There must be teams already formed prior to first deployment.
  + **Triggers**: Time triggers the leaderboard to update weekly. The user must open up the web page and view over all rankings.
  + **Scenario Description**: The user logs onto the website and clicks all time ranking. The data is collected from the back end and displayed as information to the user. The user selects which week he would like to view.
  + **Post Conditions**: User is able to see their all-time rankings that is updated every week.
  + **Exceptions**: Ties apply like weekly ranking. If data is not retrieved from the back end no leader board will be displayed and an error will be shown.

##### Function 3

* + **Description**: As a user, I would like to view my team history and other teams’ history, seeing how they lost or gained points by viewing their metrics in a graph or pie chart format, including previous position.
  + **Use Case Name**: View team history
  + **Actors**: Employee
  + **Preconditions**: Employee must be logged in. Employee must be on the view my weekly/overall ranking.
  + **Triggers**: Employee must click on a team’s name.
  + **Scenario Description**: Employee logs in views ranking, clicks on a team’s name. team’s history will be displayed to the user.
  + **Post Conditions**: Employee has seen team information.
  + **Exceptions**: N/A

##### Function 4

* + **Description**: Get authenticated as a user to Login and be given permission to view the application’s content with a user access level or an admin depending on the credentials.
  + **Use Case Name**: User Authentication
  + **Actors**: User
  + **Preconditions**: User opens a browser. User must type in the application’s URL, click enter. User enters credentials, username and password in the appropriate boxes.
  + **Triggers**: User must click on the Login Button.
  + **Scenario Description**: User opens browser, types Websites URL, Enter credentials and clicks the login button, credentials are compared and verified, access approved as a user.
  + **Post Conditions**: User gets redirected to the main page of the website.
  + **Exceptions**: If the verification result is false, deny access and prompt user telling them to check their information and try again.

##### Function 5

* + **Description**: User Should be able to log out.
  + **Use Case Name**: Log out
  + **Actors**: User
  + **Preconditions**: User must log onto the website and be authenticated.
  + **Triggers**: User clicks on the log out button.
  + **Scenario Description**: User enters credentials, gets verified and redirected to the main page. User Clicks on the Log out button.
  + **Post Conditions**: User is redirected to the login page.
  + **Exceptions**: N/A

##### Function 6

* + **Description**: User should be able to see their profile and which team they belong to.
  + **Use Case Name**: View My Profile.
  + **Actors**: Employee
  + **Preconditions**: Employee Must Log On.
  + **Triggers**: Employee Clicks on the view my profile button.
  + **Scenario Description**: Employee logs in view main page, clicks on view my information, user information gets fetched from the database and displayed to the user.
  + **Post Conditions**: N/A
  + **Exceptions**: Employee can belong to multiple teams.

##### Function 7

* + **Description**: As a user, I would like to be able to change my password.
  + **Use Case Name**: Change password
  + **Actors**: Employee
  + **Preconditions**: User must log onto the website and be authenticated. User must have previous password.
  + **Triggers**: User clicks the option on the menu to change password.
  + **Scenario Description**: User enters credentials, gets verified and redirected to the main page. User then has options on the menu. user clicks the option to change their password. Password must satisfy conditions to create a new password.
  + **Post Conditions**: User password is changed.
  + **Exceptions**: If user has forgotten password he will be sent a link to his email that will allow him to change his password.

##### Function 8

* + **Description**: Being able to create, edit and remove teams.
  + **Use Case Name**: Team management
  + **Actors**: Admin
  + **Preconditions**: N/A
  + **Triggers**: Admin chooses the option to create a team.
  + **Scenario Description**: Admin logs in, after his account is authenticated. they are on the main page, they click the option to manage teams, they click add team, create team members and add members on initial creation by searching their name. going back to team management they click remove team, they search the specific team and remove it.
  + **Post Conditions**: Managed teams by either creating or deleting a team. Also adding or removing employees from teams.
  + **Exceptions**: Creating a team with no members assigned, allowing them to be assigned later.

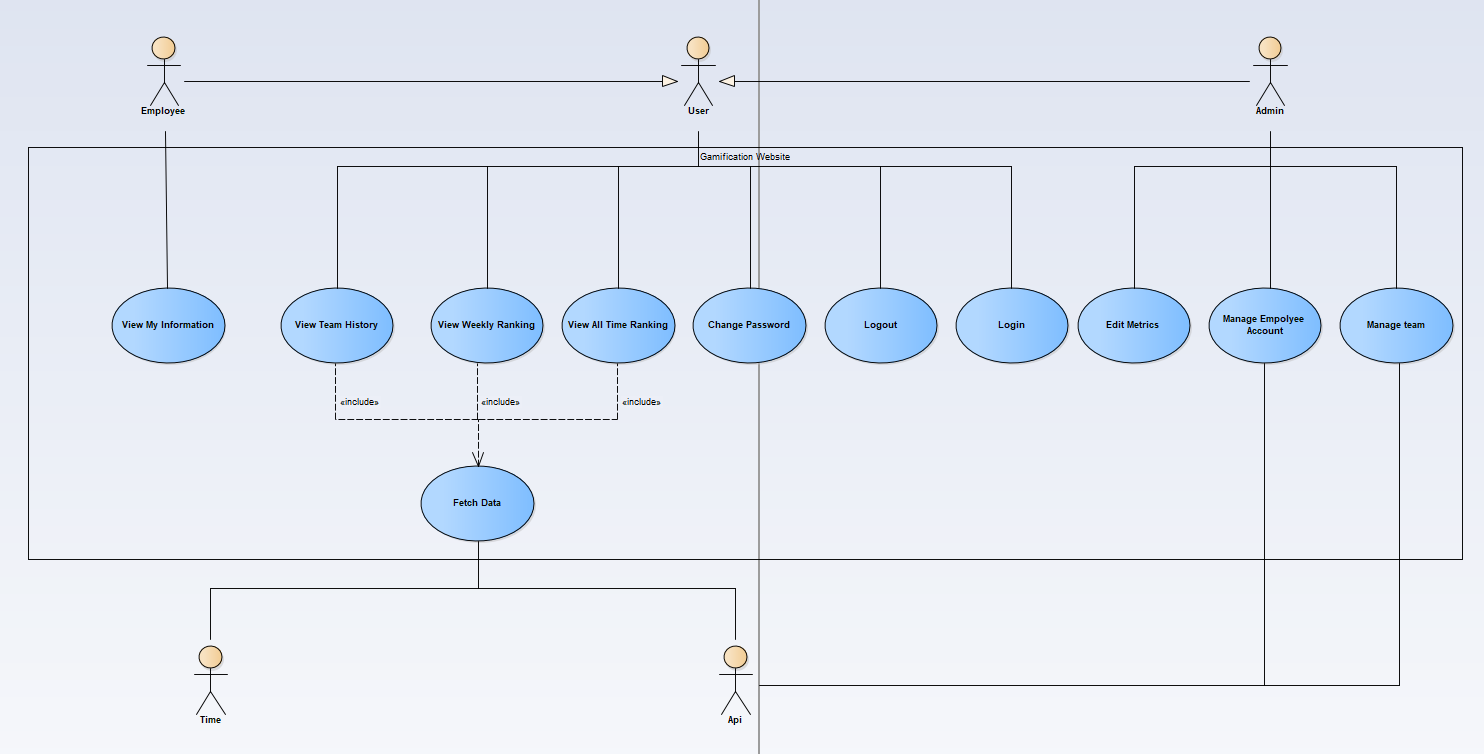
##### Function 9

* + **Description**: Allow admin the ability to create and deactivate employee accounts.
  + **Use Case Name**: Manage Employee Account
  + **Actors**: Admin
  + **Preconditions**: Employee Must be in the client’s database.
  + **Triggers**: Admin selects the option to Manage Employee.
  + **Scenario Description**: Admin selects the option to manage employee, then they select create employee account, they select employee from a list of employees that come from the clients api. Once the employee is selected they click the option to generate a username and password which creates the account. the admin can also select the option to deactivate and account, when deactivating an account, it disables access for the employee on the website and deregisters them from their team.
  + **Post Conditions**: Employee account is created. Employee account is deactivated.
  + **Exceptions**: N/A

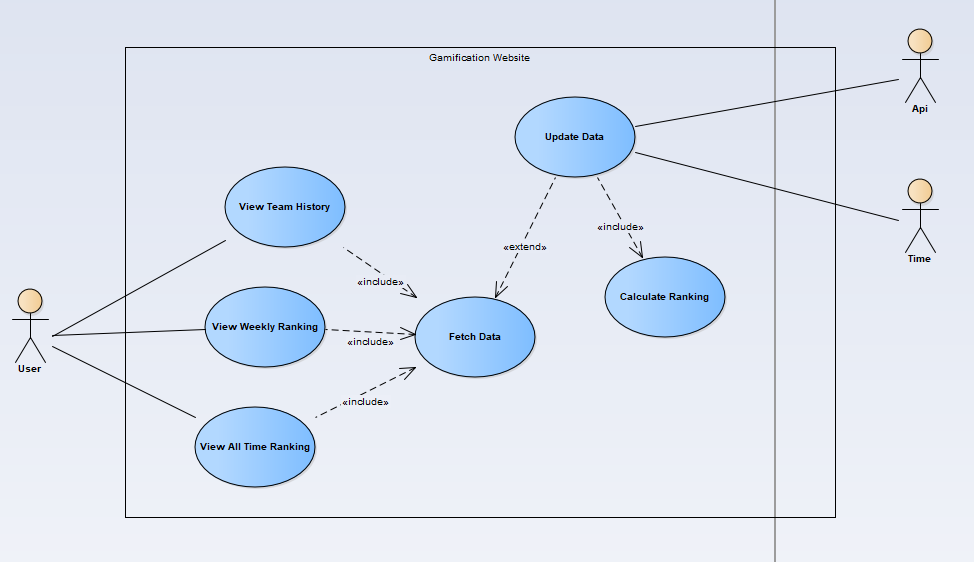
##### Function 10

* + **Description**: Admin edits weights of metrics.
  + **Use Case Name**: Edit Metrics
  + **Actors**: Admin
  + **Preconditions**: Admin must log in and be verified.
  + **Triggers**: Admin must select edit metrics.
  + **Scenario Description**: Admin selects edit metrics. The admin can choose the weight of each individual asset which will then update fields in the database.
  + **Post Conditions**: The weight metrics will affect only future rankings.
  + **Exceptions**: N/A

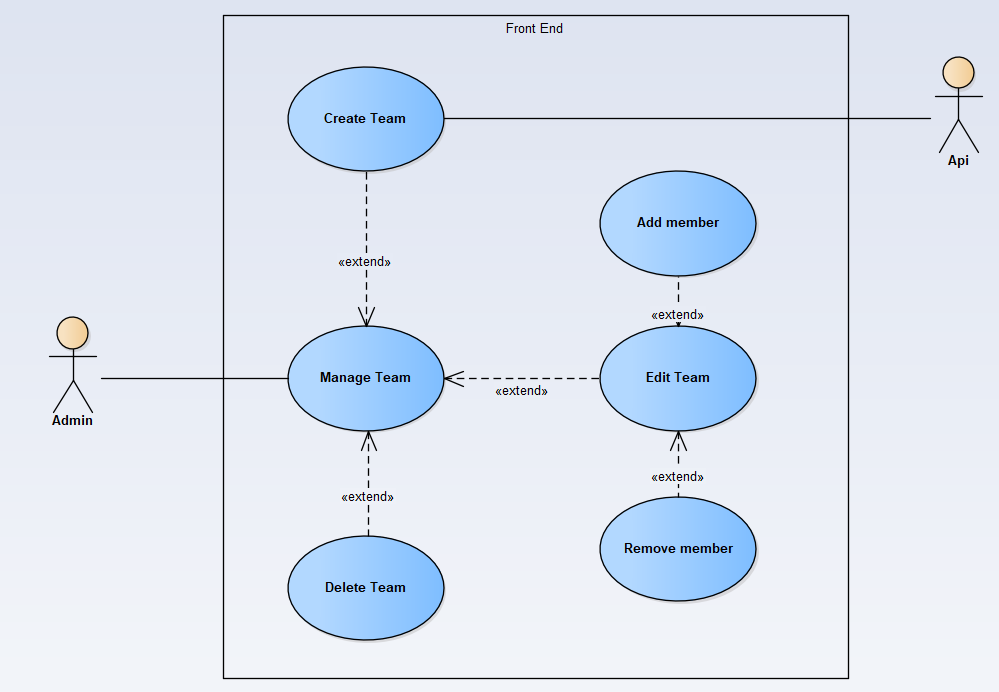
### Use Case Diagrams



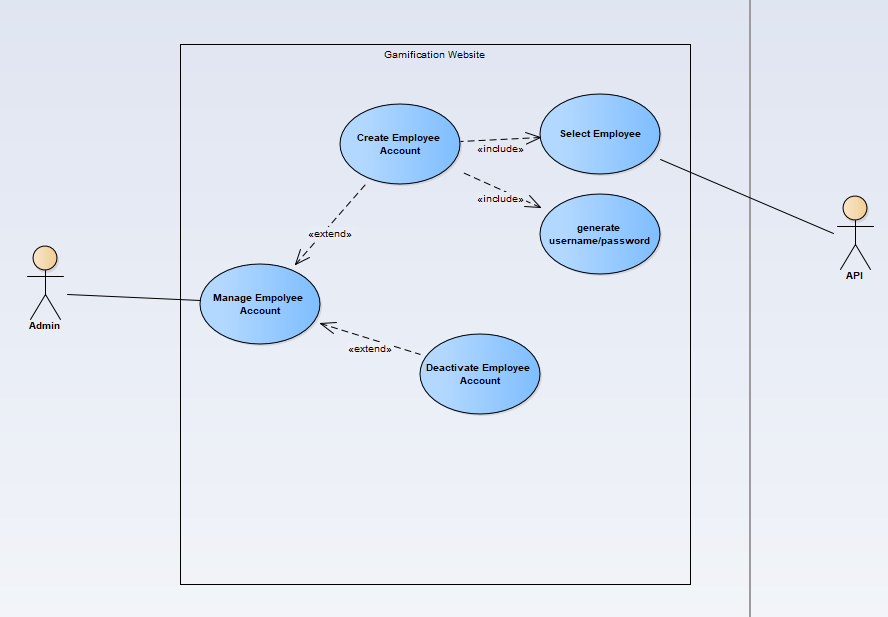
#### System Use Case



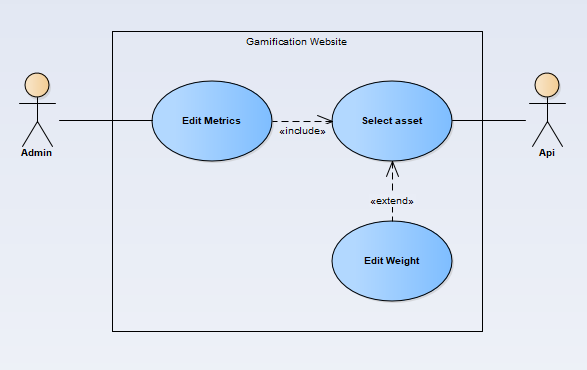
#### Use Case 1, 2, 3



#### Use Case 8

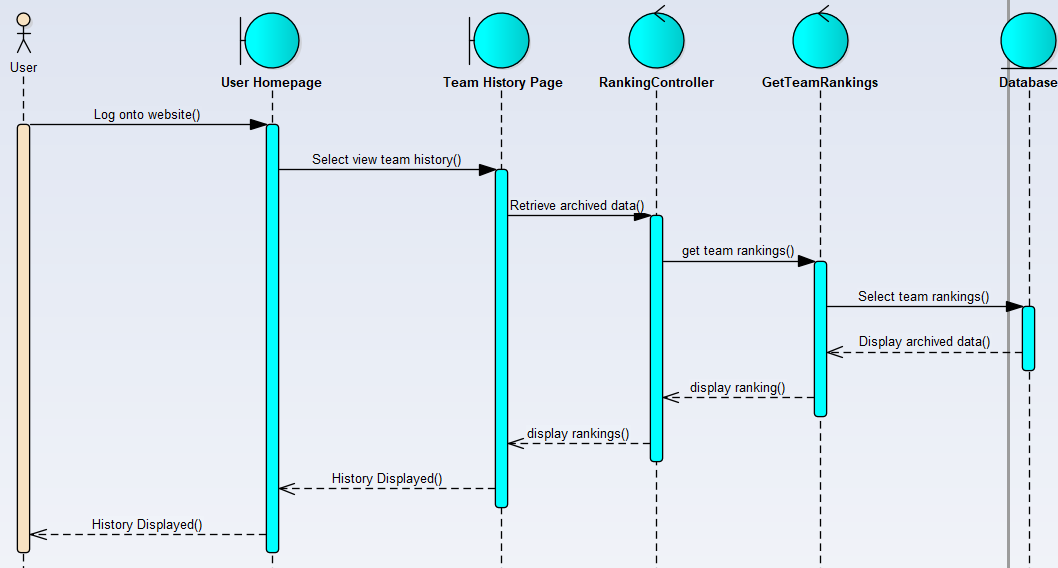


#### Use Case 9

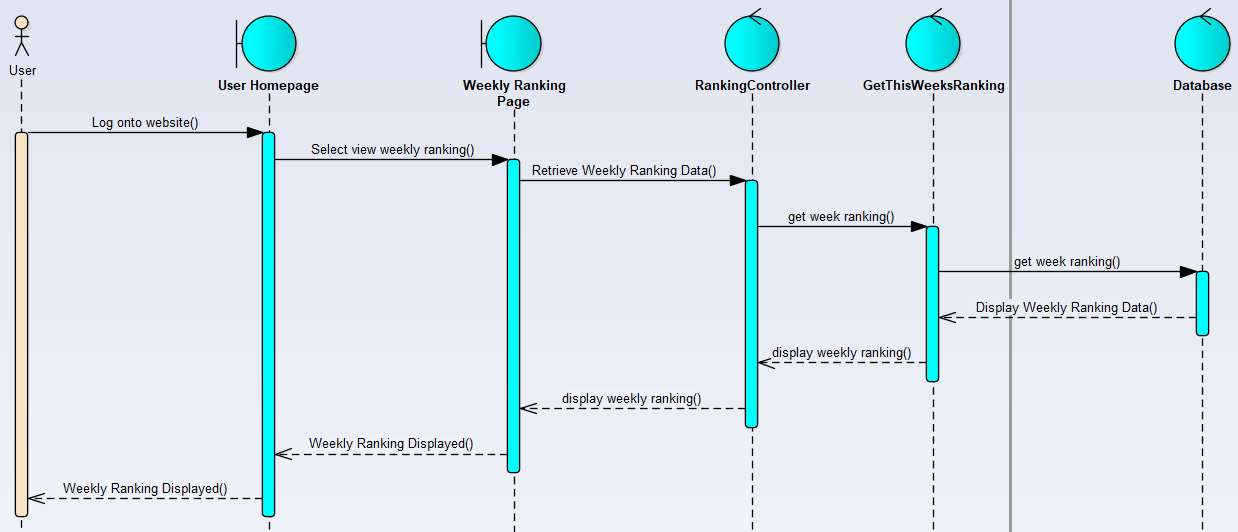


#### Use Case 10

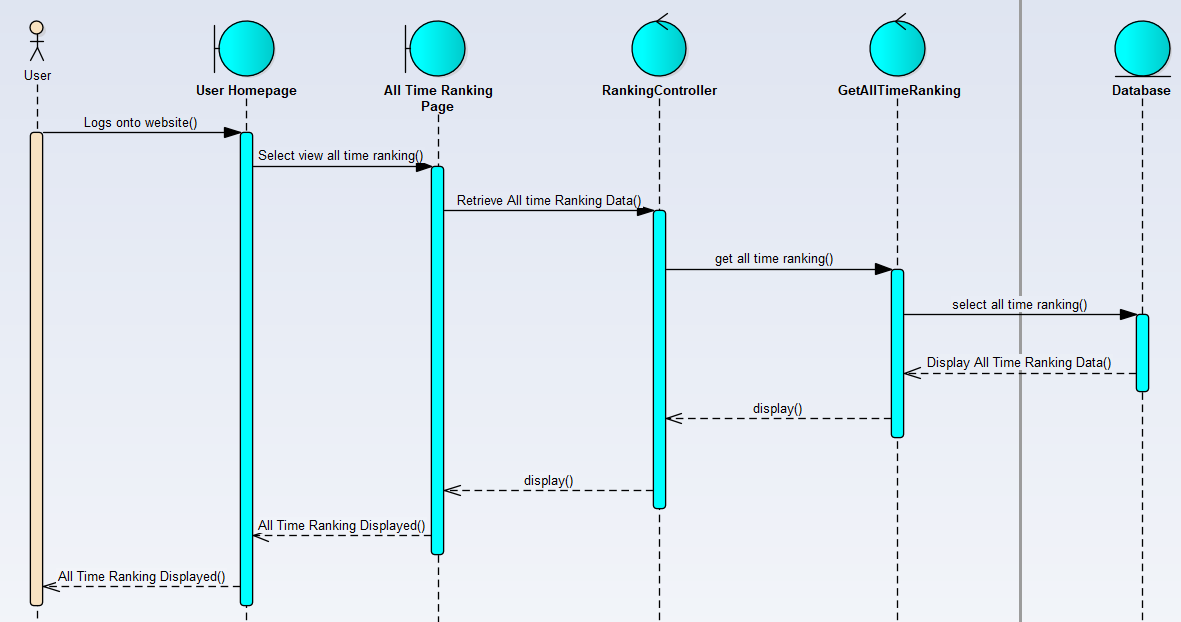
### Sequence Diagrams



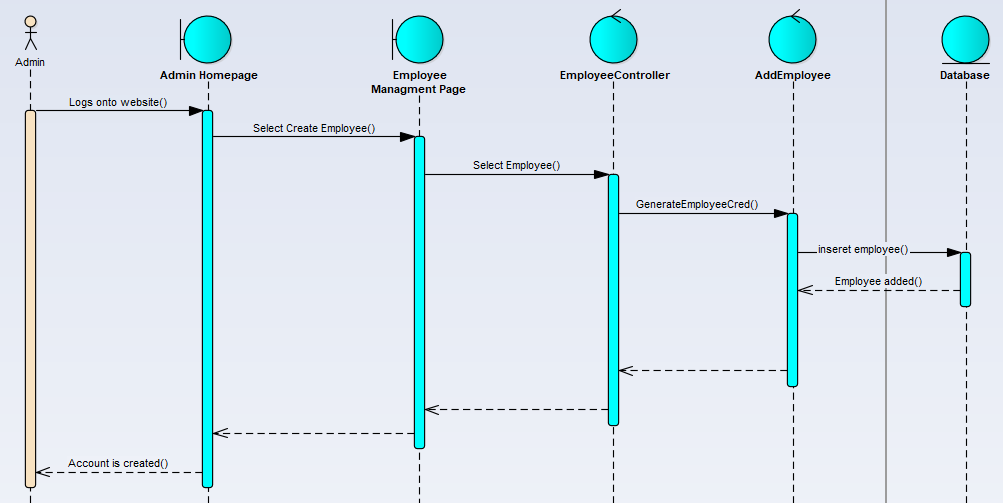
#### View Team History



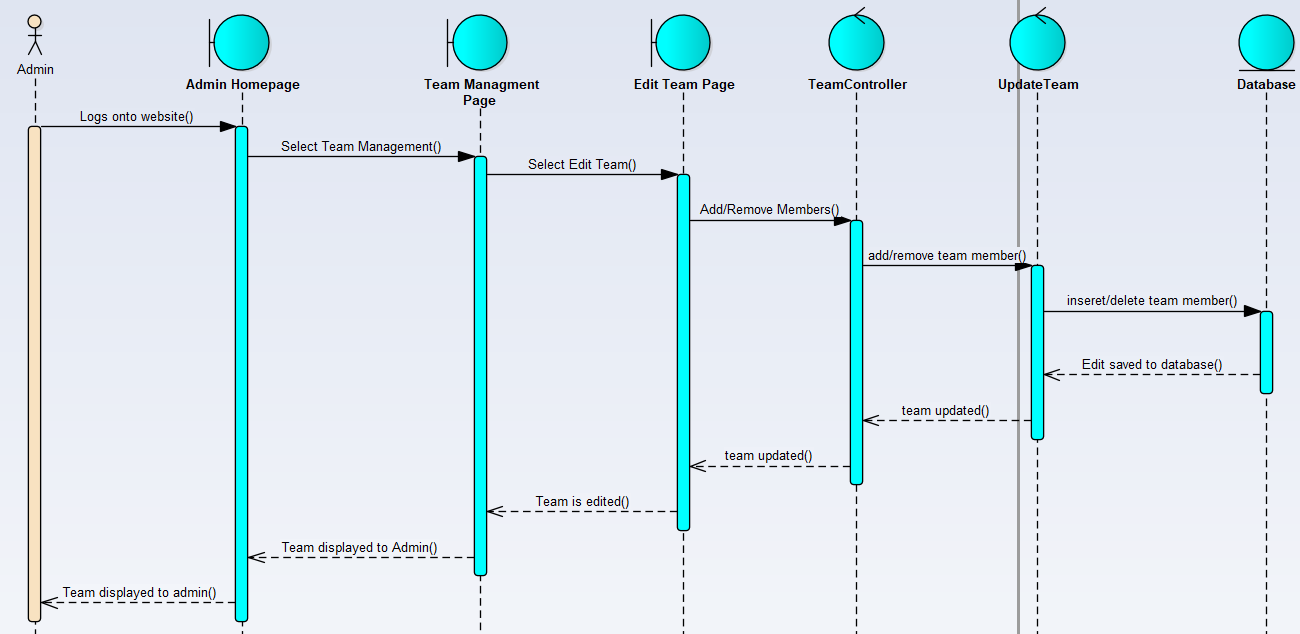
#### View Weekly Ranking



#### View All Time Ranking



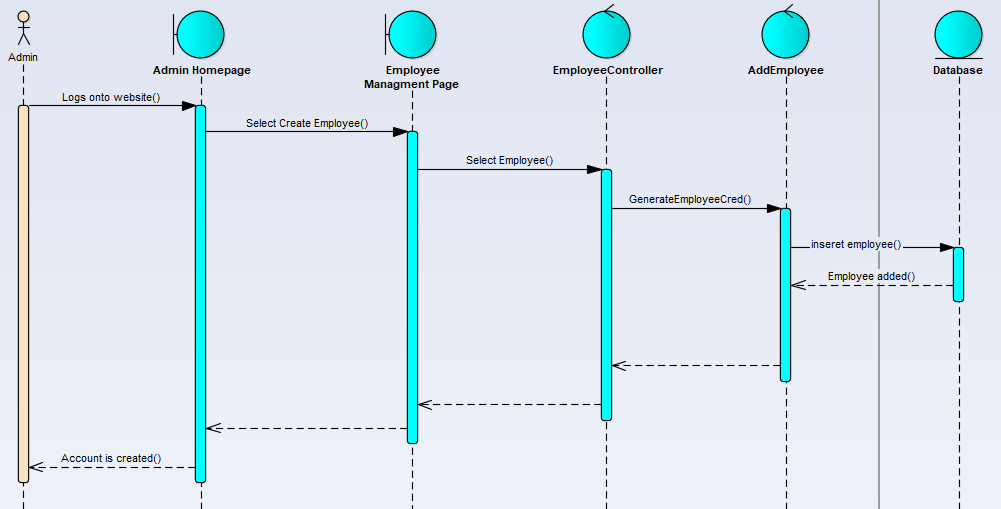
#### Create Team



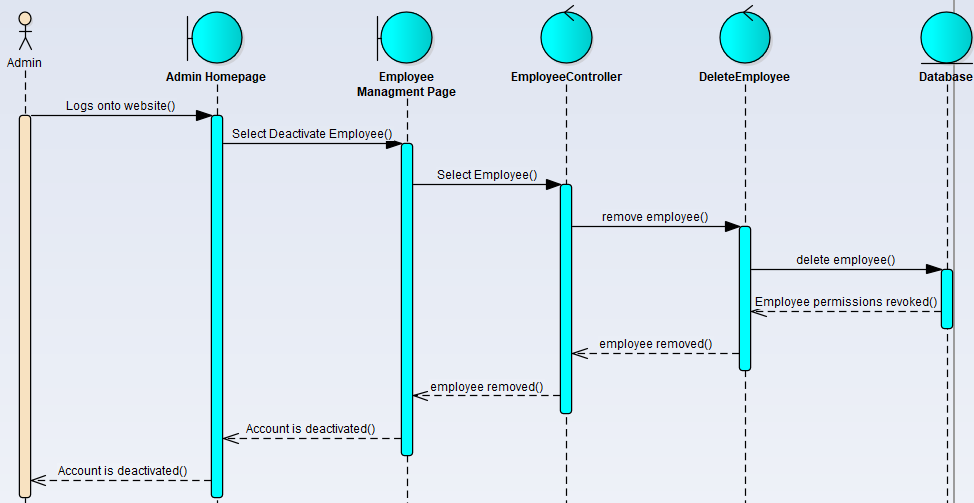
#### Edit Team



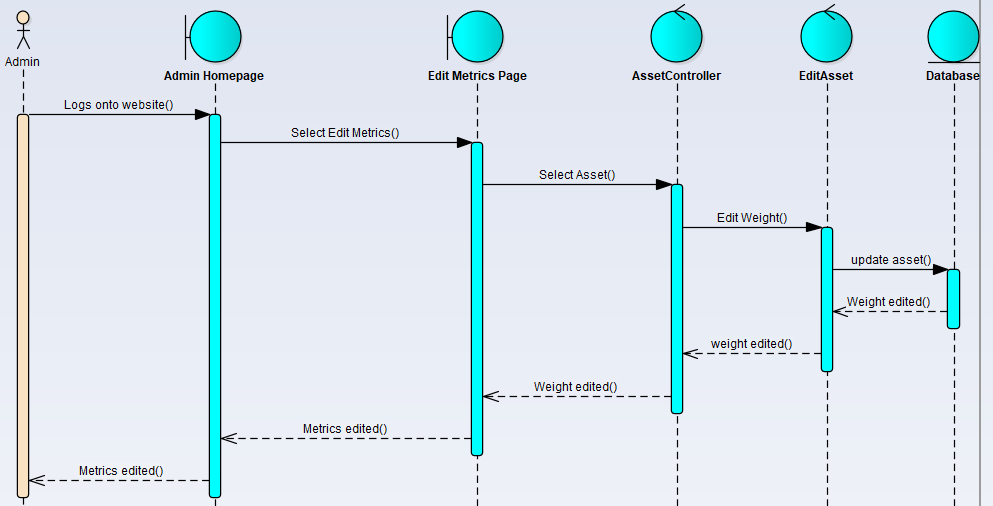
#### Delete Team



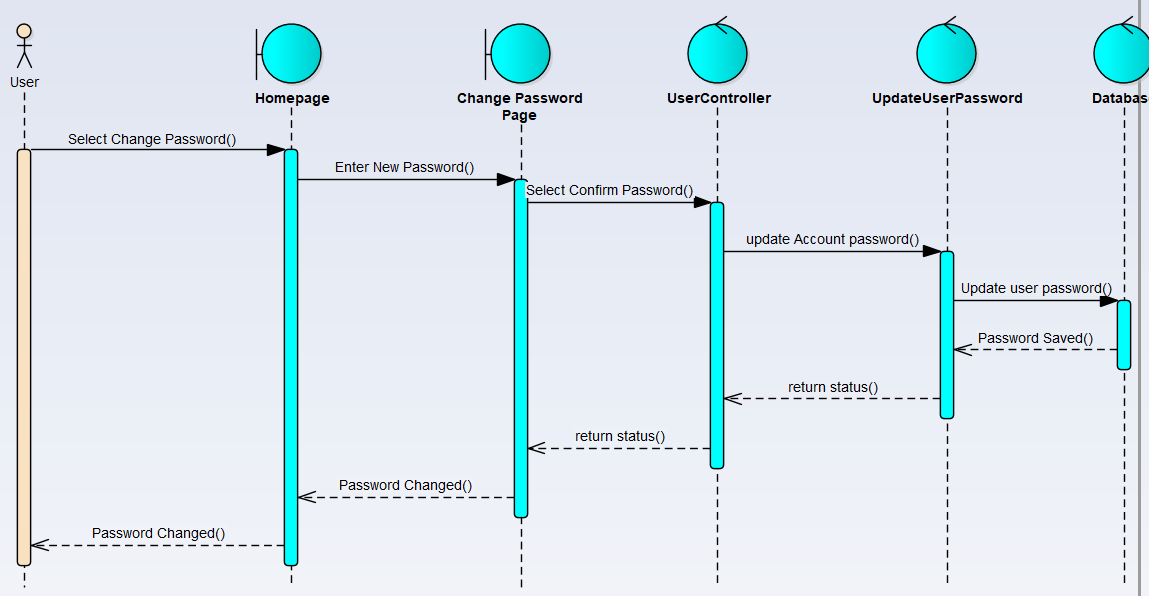
#### Create Employee Account



#### Deactivate Employee Account

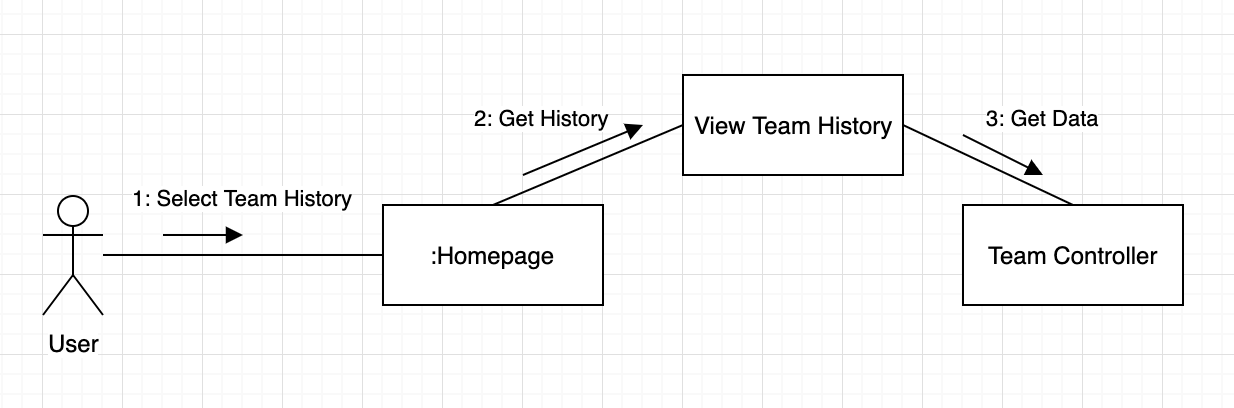


#### Edit Metrics



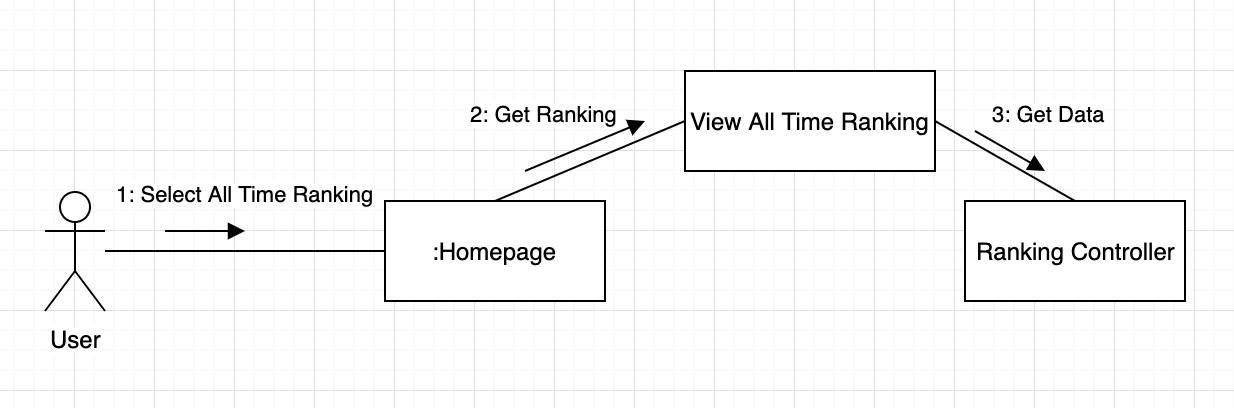
#### Change Password

### Communication Diagrams

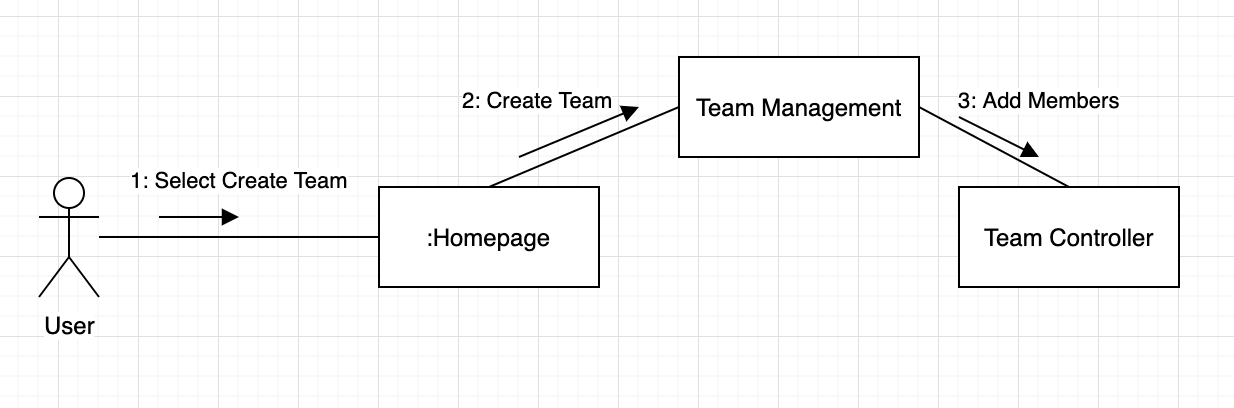


#### View Team History

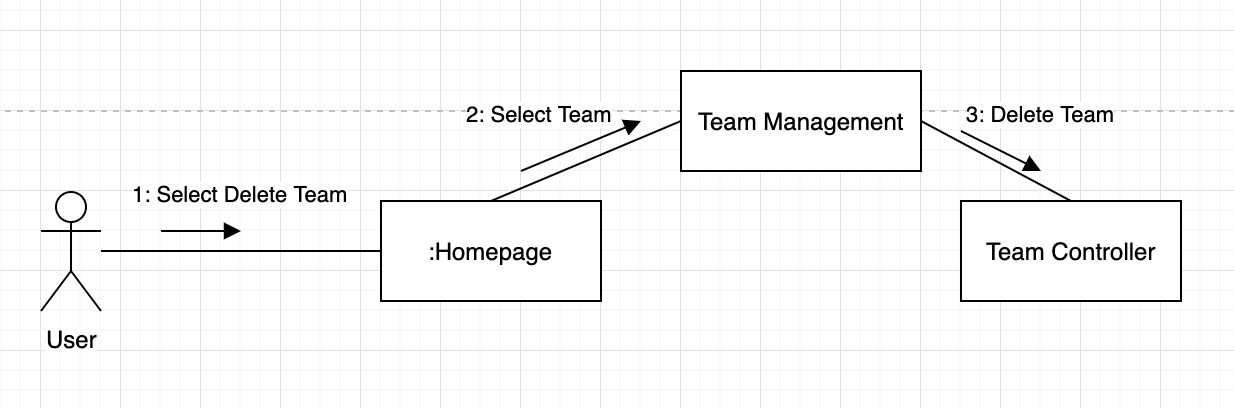
#### View Weekly Ranking



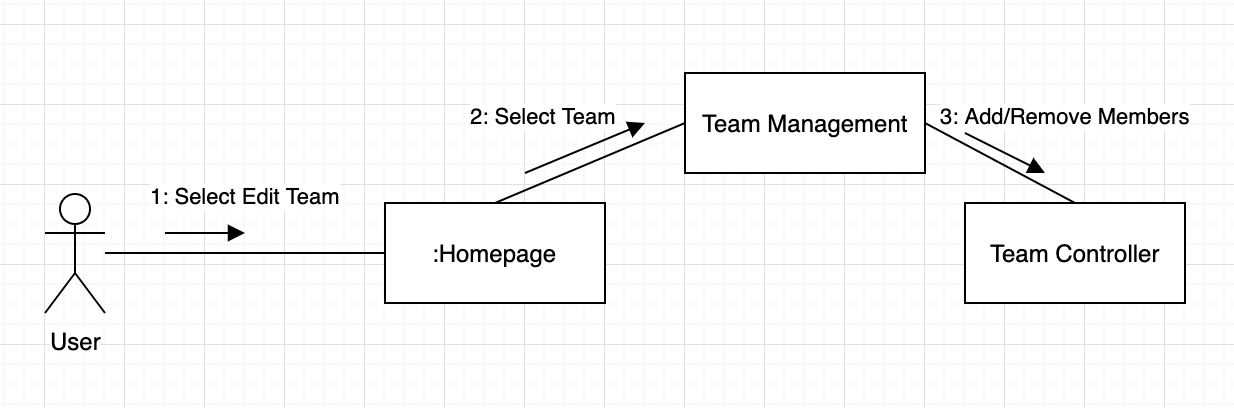
#### View All Time Ranking



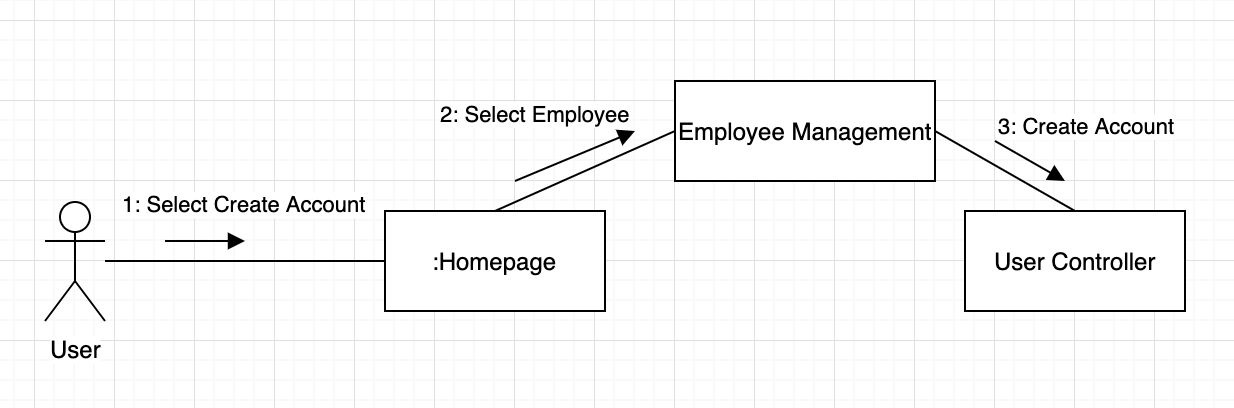
#### Create Team



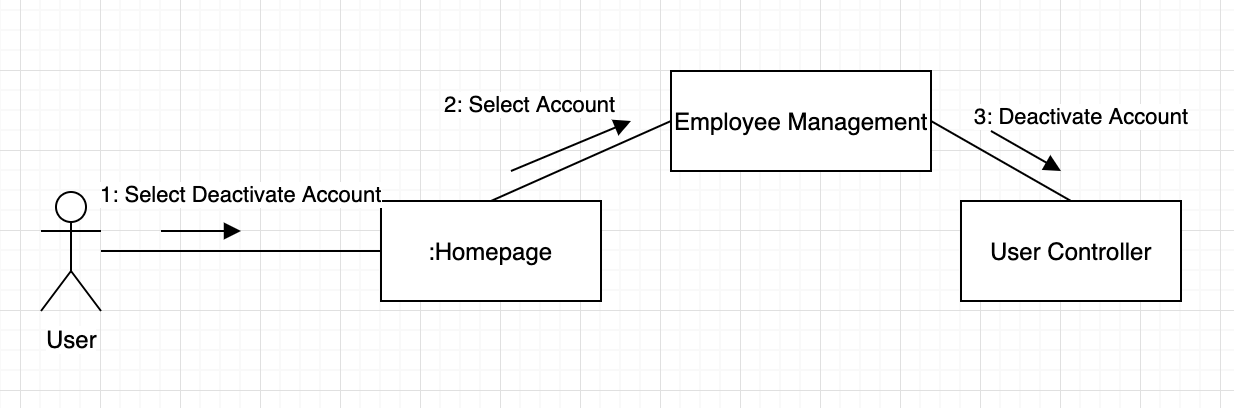
#### Delete Team



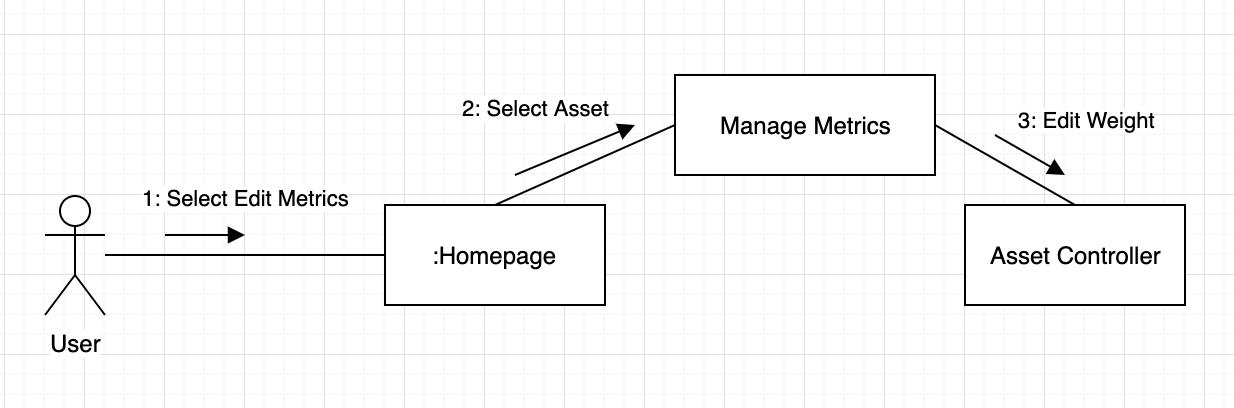
#### Edit Team



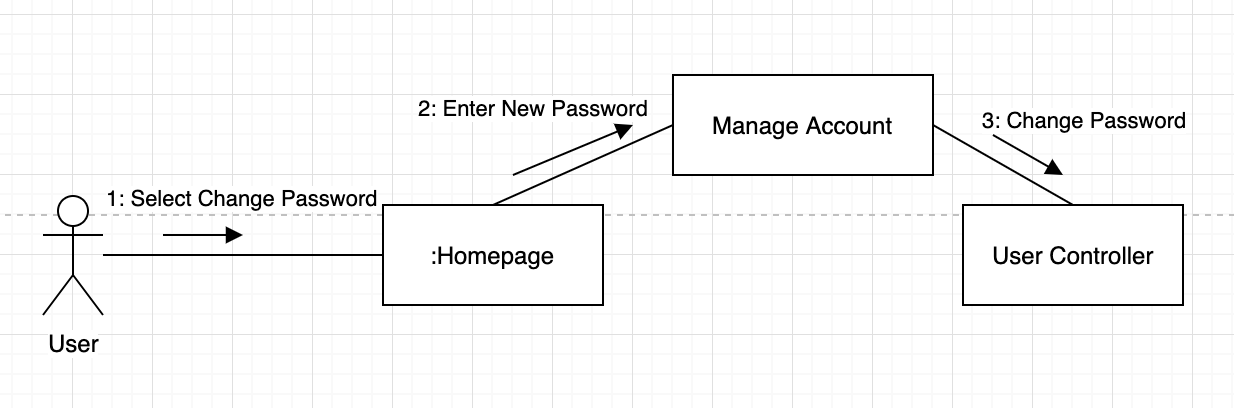
#### Create Employee Account



#### Deactivate Employee Account



#### Edit Metrics



#### Change Password

### Description for Software Behavior

Admin log onto the website, the admin navigates to the menu for managing employees. Admin Creates an employee account, the creation of an employee account will require an employee email address, name, and password (the employees’ company will be automatically saved depending which company the admin belongs to).

Admin selects the option to deactivate an employee. that employee will no longer be able to access the website until reactivated.

Admin navigates to the menu for managing teams, admin selects the option to create a team. All the crews are retrieved from the Invisi-tag Api, the crews that do not currently exist as teams in the database will be projected to the admin (or teams that are deactivated). the admin then selects the crew, the admin is then presented a list of active employees, the admin can select all the employees that belong to that team (the admin does not need to select any members), the admin then creates the team.

Admin can select the option to deactivate a team. The team is deactivated and will not be shown on any further leaderboard reporting.

Admin selects the option edit teams, the admin is presented of all the teams registered to his company, the admin selects the team they wish to edit. a list of members who belong to the team will be displayed. The admin can select an employee and remove them, the admin can also add an employee using an add employee option.

Admin navigates to the edit metrics, a list of all the assets with their assigned weights will displayed. the admin then selects the asset he would like to edit, the admin changes asset weight and saves (the assets new weight will on effect the previous days that week and not previously archived ones). Upon initial population of database by fetching asset information from Invisi-tag Api the asset weight will be proportionate to the asset price as default.

Once daily the website will update the current week leaderboard by updating by inserting new points to the teams based on new reports (reports are what the employees are being ranked on, report shows what assets need to be scanned out, the assets that are scanned backing, for each job). There are 3 states for each asset green, red, and yellow, green indicates the correct asset was scanned out or in, red indicates the item was not scanned out or in, yellow indicates an incorrect item (belonging to another team) is scanned in or out. There are times when teams’ items are faulty, and they are assigned temporary items, this should not affect ranking because the asset is swapped on the report.

At the end of the week, the leaderboard is archived into the database and the current week ranking is reset (once archived into the database the weekly ranking cannot be changed). Essentially creating a new week record for information to be saved to.

Users are able to view weekly leaderboards, users have the option of selecting a team, the team’s information will be shown, including employees assigned to and the team and archived weekly ranking positions.

Users have the option of viewing all time rankings, the member selects the week they would like to view, the leaderboard ranking for that week will be displayed.

Employees select the option to view their profile, the employees profile information will be displayed, the information includes the employee’s teams they belong to, the employee then can select a team and teams’ information will be displayed.

Users can change their passwords either by forgotten password which sends link to their email or through login onto the website. after password is changed it is saved onto the website.

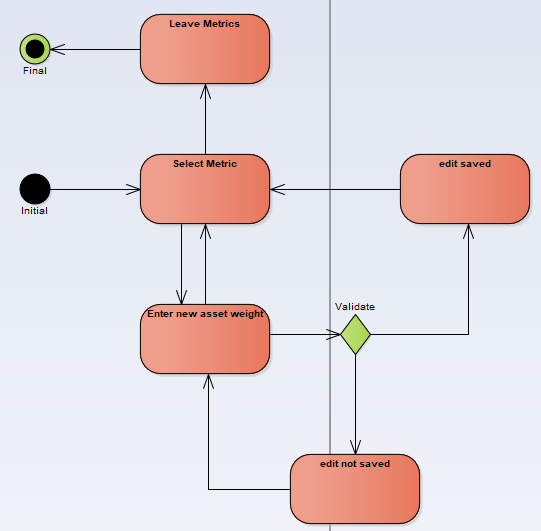
#### Events

1. create team
2. select employees
3. deactivate team
4. edit team
5. select employee to add
6. select employee to remove from team
7. select employee account to create
8. select employee account to deactivate
9. select edit metrics
10. select asset
11. select view leaderboard
12. select team
13. select view team information
14. select view profile information
15. select change password
16. attempt to login

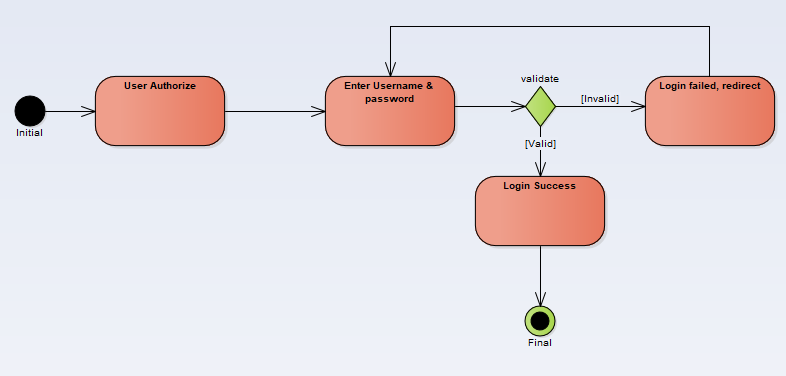
#### State

1. team is created
2. employees are selected
3. team is deactivated
4. employee is selected
5. employee is added to team
6. employee is removed from team
7. team is edited
8. employee account is created
9. employee account is deactivated
10. select asset
11. asset weight is changed
12. metrics are edited
13. view leaderboard
14. team is selected
15. daily ranking update
16. weekly ranking archive

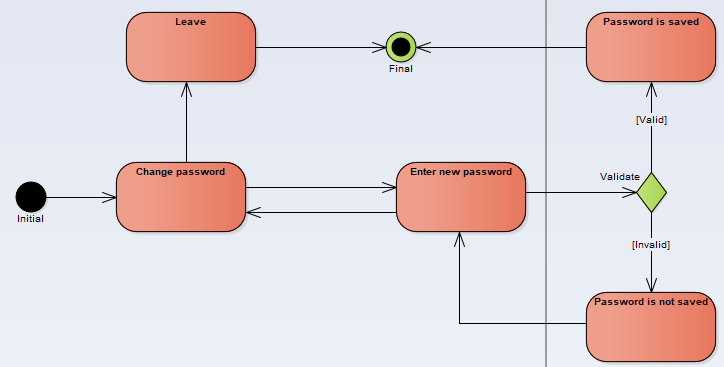
### State Transition Diagrams



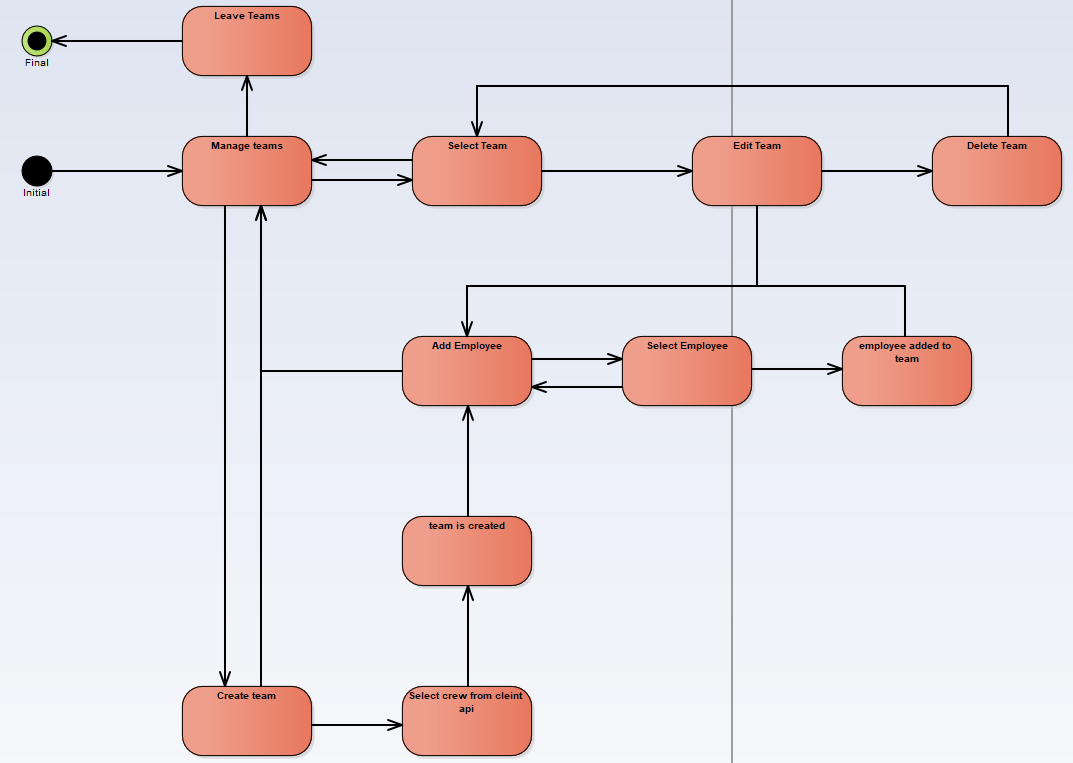
#### Manage Metrics



#### Login



#### Change Password

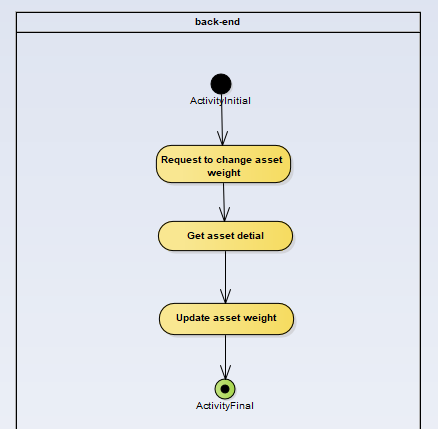


#### Manage Teams

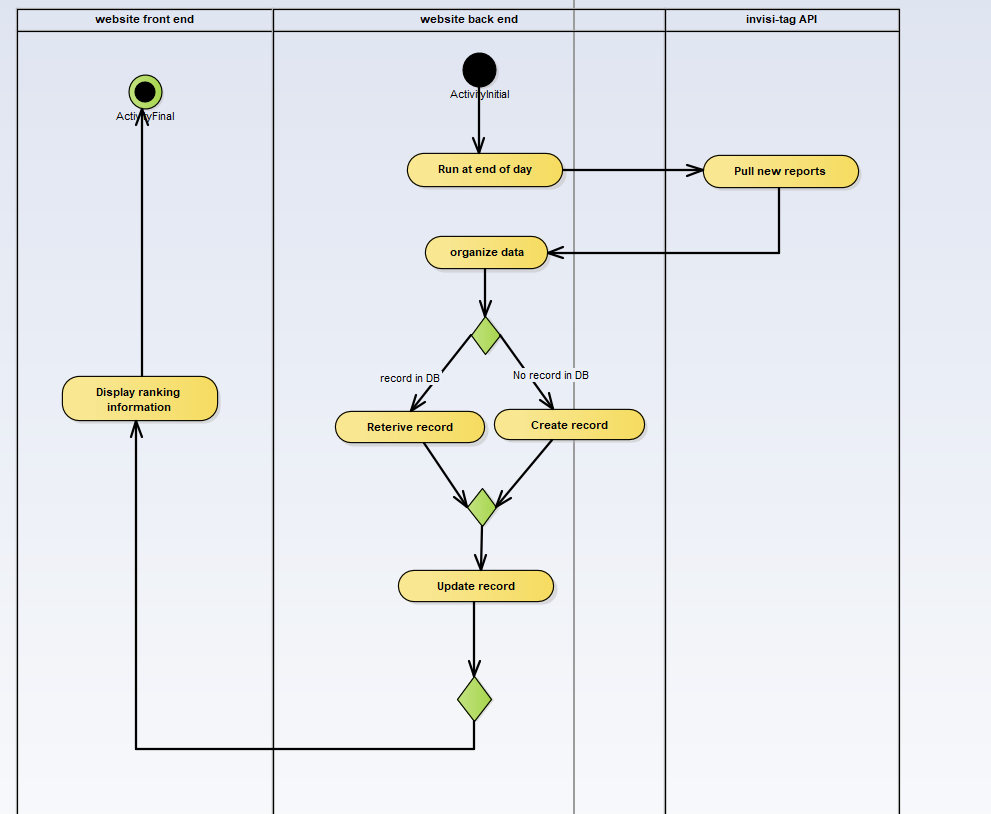


#### Managing Employees

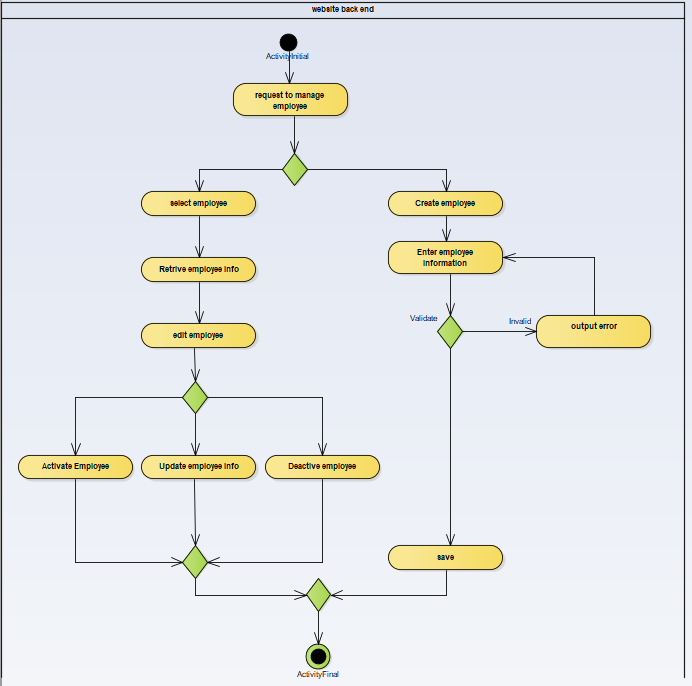
### Activity Diagrams



#### Update Asset



#### Update Ranking



#### Manage Employee

#### Manage Team

### Restrictions, Limitations, and Constraints

#### Restrictions:

The software will work with future browser updates meaning that we did not use 3rd party feature like angular modals and sticking strictly to angular vanilla. That restricts the front-end design to a more static web pages and with few dynamic features.

#### Limitations:

We created the gamification web-application to fit the client's current software. Using the clients Api we pull information, but we are limited to how the client’s software is designed. Their software was not initially created with this feature in mind. Ex tags in the clients Api don't belong to any one company, everyone has the same set of tags. but in the gamification website the client can change the weight for the asset which means each company can modify their assets(tag). There are no team IDs or team tables, just reports that hold the team names. We had to find clever solutions to overcome this initial design flaw.

#### Constraints:

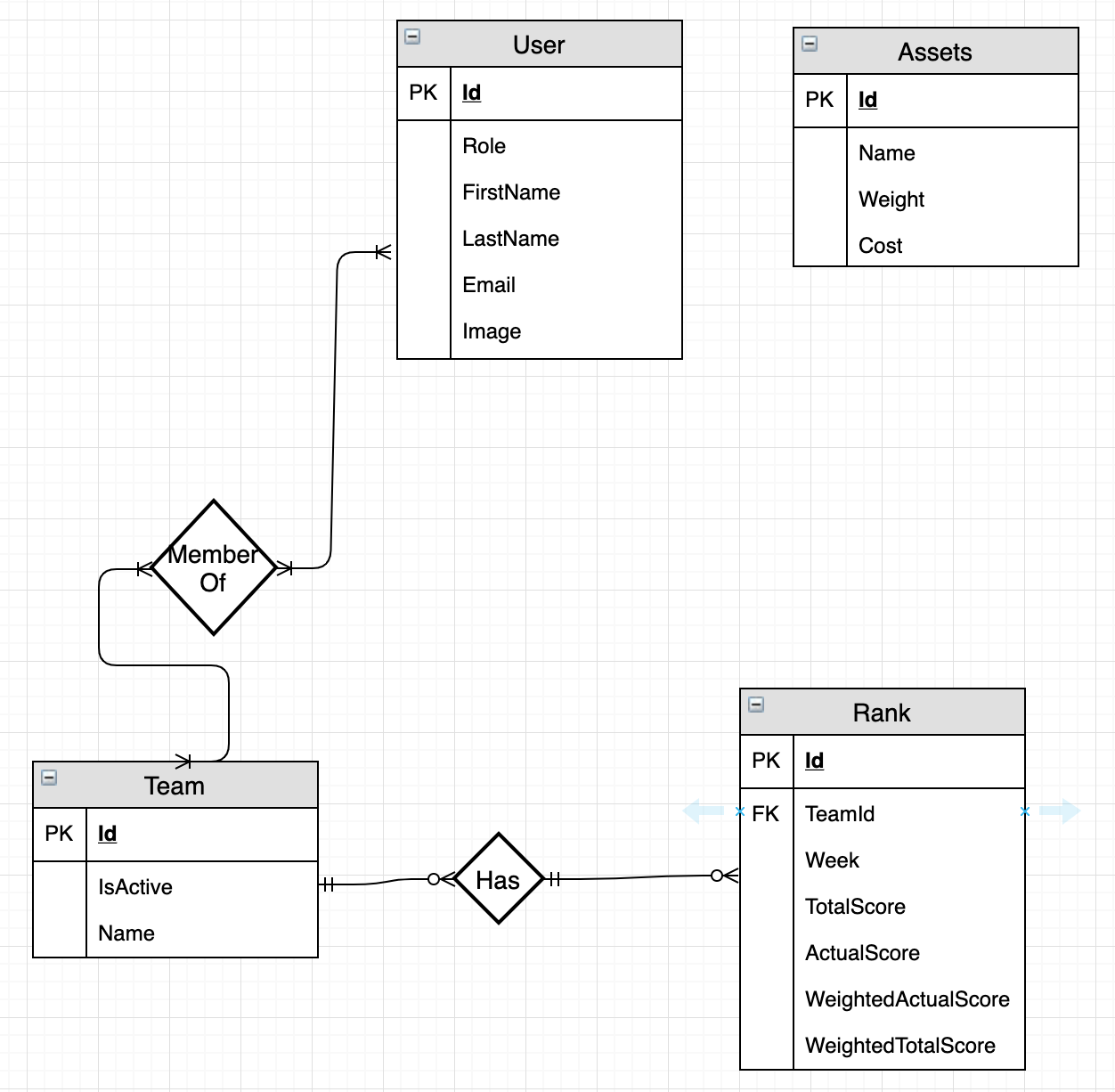
Software/database is used for multiple companies meaning there may be cross over in names. teams’ names and asset names will have to be partially unique, some attribute values must be unique for a company, but other companies may use that name: ex team1 can only be used by ford once but Chevy could also use the name team1 once.

Software will pull data from the client Api once a day.

### Software Design

#### Data Model

##### ERD

****

##### Class Diagram

Class Diagram explained:

We are using the MVC architecture design pattern. The website will act as the view, the view will connect to the controller when retrieving or sending information to the database. The controllers are also acting as a facade they will connect to the classes that will interact with the database, the database operations are broken up into 2 categories: Commands and queries this is following the CQRS(command and query responsibility segregation) the commands will update the database and the queries will retrieve information. Both commands and queries will use the models

The service application will run on a time-based trigger. Rank service will pull new reports from the client Api and populate the ranking records in the database.

UpdateDataService will focus on pulling new company, employee, team and asset data from the client Api.

### Data Dictionary

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Table Name | Column Name | DB Data Type | Size | DataType | Constraint | Description | Example |
| Company | Id | UniqueIdentifier | 36 | Guid | Unique, Not Null | Primary key | 0f8fad5b-d9cb-469f-a165-70867728950e |
| Company | Name | VarChar | 100 | string | Not Null | Company name | Ford |
| Company | ShardName | VarChar | 100 | string | Unique, Not NUll | Shard company will direct to | Shard1 |
| Company | CreatedDate | DateTime | n/a | datetime | Not Null,  1753-01-01 through 9999-12-31 | Indicates when company was inserted into the database | 2019-10-10 00:00:00 |
| Team | Id | UniqueIdentifier | 36 | Guid | Unique, Not Null | Primary key | 0f8fad5b-d9cb-469f-a165-70867728950e |
| Team | Name | VarChar | 100 | string | Not Null, Only unique to the company | Team name, partial key, each company can only have one team name, but different companies can have the same team name. | Alchemist |
| Team | IsActive | Bit | 1 bit | boolean | Not Null, Default 1 | Indicates if the team is currently active | 1 |
| User | Id | UniqueIdentifier | 36 | Guid | Unique, Not Null | Primary key | 0f8fad5b-d9cb-469f-a165-70867728950e |
| User | Role | varChar | 100 | string | Not null | User role (Admin, Employee) | Admin |
| User | FirstName | VarChar | 100 | string | Not Null | User first name | John |
| User | LastName | VarChar | 100 | string | Not Null | User last name | Smith |
| User | Email | VarChar | 100 | string | Unique, Not Null, must be valid Email | User email address needed to identify user and help him recover password in the scenario of a lost password | JohnSmight@Example.com |
| user | Image | VarBinary | max | binary | Nullable | Holds employees profile pic | 010101 |
| Asset | Name | VarChar | 100 | string | Not Null | Asset name | Shovel |
| Asset | Weight | Integer | 1-2 | int | Not Null, 1 <= length <= 10 | Asset weight | 1 |
| Asset | Cost | Decimal | (18,2) | float | Not Null, 0.00 <= length <= msx | Asset cost | 15.99 |
| Asset | Id | UniqueIdentifier | 36 | Guid | Unique, Not Null | Primary key | 0f8fad5b-d9cb-469f-a165-70867728950e |
| Asset | Rfid | VarChar | 30 | string | Unique, Not Null | key assigned by client for scanning | 800-00-00-00 |
| Rank | Id | UniqueIdentifier | 36 | Guid | Unique, Not Null | Primary key | 0f8fad5b-d9cb-469f-a165-70867728950e |
| Rank | TeamId | UniqueIdentifier | 36 | Guid | Unique, Not Null | Foreign key from team, shows the team rank belongs to | 0f8fad5b-d9cb-469f-a165-70867728950e |
| Rank | Week | DateTime | 17 | DateTime | Not Null,  1753-01-01 through 9999-12-31 | The week that this rank belongs to, between every Sunday at midnight | 2019-03-13 00:00:40.000 |
| Rank | ActualScore | big Integer | 18 | long | Not Null, Default 0, 0 <= length <= 2^63-1 | The score a team has, based on items detected on a report that is pulled from the client api | 50 |
| Rank | TotalScore | big Integer | 18 | long | Not Null, Default 0, 0 <= length <= 2^63-1 | The maximum points the team could get if all assets are scanned | 100 |
| Rank | ActualWeightedScore | big Integer | 18 | long | Not Null, Default 0, 0 <= length <= 2^63-1 | The score a team has, based on items detected on a report that is pulled from the client api | 50 |
| Rank | TotalWeightedScore | big Integer | 18 | long | Not Null, Default 0, 0 <= length <= 2^63-1 | The maximum points the team could get if all assets are scanned | 100 |
| MemberOf | UserId | UniqueIdentifier | 36 | Guid | Unique, Not Null | Foreign key from User | 0f8fad5b-d9cb-469f-a165-70867728950e |
| MemberOf | TeamId | UniqueIdentifier | 36 | Guid | Unique, Not Null | Foreign key from Team | 0f8fad5b-d9cb-469f-a165-70867728950e |

### 

### Data Design

#### Internal Software Data Structure

* User
* The user table consists of the following attributes.
* There will be two levels of security – User (Employee) and Admin (Owner/Manager).
  + Id (Primary Key) – The Id representing the user.
  + CompanyId (Foreign Key) – The Id representing the company the user works for.
  + FirstName – First name of the user.
  + LastName – Last name of the user.
  + Email – Work email for the user. This will be used as the username at login.
  + Password – Password for the user to login. This will be encrypted.
* Team
* The team table consists of the following attributes.
  + Id (Primary Id) – The Id representing the Team working on a task.
  + CompanyId (Foreign Key) – The Id representing the company the team works for.
  + Name – The name of the team.
* Ranking
* The ranking table consists of the following attributes.
  + Id (Primary Key) – The Id representing the rank of the team.
  + CompanyId (Foreign Key) – The Id representing the company the team works for.
  + TeamId (Foreign Key) – The Id representing the team the rank belongs to.
  + Week – The week of the rank to be viewed.
  + ActualScore – The score the team earns in a specific week.
  + TotalScore – The highest score the team can earn for the week.
* Assets
* The assets table consists of the following attributes.
  + Id (Primary Key) – The Id representing the asset.
  + CompanyId (Foreign Key) – The Id representing the company the asset belongs to.
  + Name – The name of the asset.
  + Weight – The weight of the asset. This is what is used when calculating the actual score for a teams rank.
  + Cost – The cost of the asset. If no weight is specified by the admin, the cost by default will be used when calculating the actual score for a teams rank.

#### Global Data Structure

* Shard Manager
* The shard manager will be the only global data structure used.
  + Company
  + The company table consists of the following attributes.
    - Id (Primary Key) – The Id representing the company that uses the Invisi-tag software.
    - ShardId (Foreign Key) – The shard name that belongs to the company the user works for.
    - Name – Name of the company that uses the Invisi-tag software.
  + Shard
  + The shard consists of the following attributes.
    - Id (Primary Key) – The Id representing the shard for a specific company.
    - Name – The name of the shard.
* UserLogin
* The UserLogin table consists of the following attributes.
* This table consists of the same attributes as User table, however this table will consist of all users for all companies.
  + Id (Primary Key)
  + CompanyId (Foreign Key)
  + FirstName – First name of the user.
  + LastName – Last name of the user.
  + Email – Work email for the user. This will be used as the username at login.
  + Password – Password for the user to login. This will be encrypted.

#### Temporary Data Structure

* Files created for interim use are described.
  + No temporary data structures are used.

#### Database Description

* Database(s) created as part of the application is(are) described.
  + The database we are using is a Microsoft SQL Database. Microsoft SQL Server Management Studio is used to create the database. It will contain all data inputted during development. It will also contain any information added from the user end.

### Program Structure

#### Description for Component Manage Team

* 1. Processing narrative (PSPEC) for component team list
     1. Users clicks manage teams, a get request will be sent to the team controller, for the get teams api. The mediator will invoke the query GetTeams and retrieve the team data from the database.
  2. Component interface description
     1. A table displays all the team name and the teams active status shows beside it.
     2. A search bar will be on top of the table that filters the team based on what the user searches.
     3. A button with the label Add will take the user to the add team sub component.
  3. Sub-Component add team processing detail
     1. Interface description
        1. A list of teams from the clients api are displayed. The teams are not in the gamification in the database at this point. There is an add button near.
     2. Algorithmic model (e.g., PDL)
        1. Get teams from api
        2. Check teams retrieved
        3. If team exist in database
        4. remove from list
        5. Return teams to the front end
        6. If admin clicks add
        7. Add team to list
        8. If admin clicks save
        9. Send teams to the add team api
        10. For each team in list
        11. Insert team into database
     3. Restrictions/limitations
        1. The team list page must work and allow the user to click the add button to open this sub component
        2. The front end service that makes api calls
        3. The team controller must be working properly
        4. The client api must be allow request to be made and results to be returned
     4. Local data structures
        1. Team
           1. Id: guid
           2. Name: string
           3. Is active
        2. User
           1. FirstName: string
           2. LastName: string
           3. Email: string
           4. Image: binary
           5. Front end
           6. Array of team objects
     5. Performance issues
        1. Retrieving teams from the clients api - making and waiting for a response may cause a delay.
     6. Design constraints
        1. When requesting teams from the clients api a list of teams are retrieved, then the teams that are in the database are removed. This is a constraint because all the teams are sent with each request, even if there is only one team in the database.

#### Description for Component team detail

* 1. Processing narrative (PSPEC) for component team detail
     1. Users selects a team from team list, a get request will be sent to the team controller, for the get team api with the team id. The mediator will the invoke the query GetTeam and retrieve the team data from the database.
  2. Component interface description
     1. A list of team members with a remove button to the right
     2. A search bar that allows the admin to filter the team
     3. The team name and the active status are on top
  3. Sub-Component add team processing detail
     1. Interface description
        1. A list of employees in the database are displayed. The list does not contain any members from the team.
     2. Algorithmic model (e.g., PDL)
        1. admin selects employees to add to team
        2. Admin saves
        3. List is added to team
        4. Team is sent to the update team api
        5. The mediator invokes the command update team
     3. Restrictions/limitations
        1. The team detail page must work and allow the user to click the add button to open this sub component
        2. The front end service that makes api calls
        3. The team controller must be working properly
     4. Local data structures
        1. Team
           1. Id: guid
           2. Name: string
           3. Is active
        2. User
           1. FirstName: string
           2. LastName: string
           3. Email: string
           4. Image: binary
        3. Front end
           1. Array of User
     5. Performance issues
        1. N/A
     6. Design constraints
        1. When user removes members he must save before opening this dialog
        2. If the request fails or an error is encountered that user is skipped and the rest are saved and an error is then thrown after processing all the users.

#### Description for Component manage employees

* 1. Processing narrative (PSPEC) for component manage employees
     1. User selects manage employees. A request is sent to the employee controller to get all employees listed in the user table. The mediator then invokes the get employees query and retrieves the employee data from the database.
  2. Component manage employees interface description
     1. When a admin logs in they have a menu selection. The admin will then select Manage Employees. A list of employees will be displayed. A menu will be displayed with the options to add remove or edit employees.
  3. Sub-Component n.m processing detail

A detailed algorithmic description for each sub-component within the component n is presented. Section 4.4 3C is repeated for each of the m sub-components of component n.

* + 1. **Interface description for Add Employee**
       1. Add employee is selected. Admin inputs employee information and submits data. The employee controller will send a request to command. Command will then insert into the user table in the database.
    2. Algorithmic model (e.g., PDL)
       1. Admin inputs employee information.
       2. Admin submits employee information.
       3. Employee password is generated and emailed to them.
       4. The mediator invokes the command insert employee.
    3. Restrictions/limitations
       1. There must be an internet connection.
       2. A connection to the database must be established.
       3. The manage employee page must load for the admin to be able to select add employee.
    4. Local data structures
       1. User
          1. FirstName: string
          2. LastName: string
          3. Email: string
          4. Image: binary
    5. Performance issues
       1. N/A
    6. Design constraints
       1. When and admin enters in the employee information he must submit for the employee to be inserted into the database.
       2. If invalid information is filled when the admin submits, an error message is displayed and the employee is not inserted.
    7. **Interface description for Remove Employee**
       1. Remove employee is selected. Admin confirms removal of employee. The employee controller will send a request to command. Command will then delete the user from the user table in the database.
    8. Algorithmic model (e.g., PDL)
       1. Admin selects remove employee.
       2. Admin confirms removal of employee.
       3. The mediator invokes the command delete employee.
    9. Restrictions/limitations
       1. There must be an internet connection.
       2. A connection to the database must be established.
       3. The manage employee page must load for the admin to be able to select remove employee.
    10. Local data structures
        1. User
           1. FirstName: string
           2. LastName: string
           3. Email: string
           4. Image: binary
    11. Performance issues
        1. N/A
    12. Design constraints
        1. When and admin removes an employee they must confirm the removal of that employee in order for them to be deleted from the database.
        2. If admin does not confirm removal of the employee, the employee is not deleted from the database.
    13. **Interface description for Edit Employee**
        1. Edit employee is selected. Admin updates employees information and submits data. The employee controller will send a request to command. Command will then update the user table in the database.
    14. Algorithmic model (e.g., PDL)
        1. Admin updates employees information.
        2. Admin submits changes.
        3. The mediator invokes the command update employee.
    15. Restrictions/limitations
        1. There must be an internet connection.
        2. A connection to the database must be established.
        3. The manage employee page must load for the admin to be able to select update employee.
    16. Local data structures
        1. User
        2. FirstName: string
        3. LastName: string
        4. Email: string
        5. Image: binary
    17. Performance issues
        1. N/A
    18. Design constraints
        1. When and admin updates the employees information he must submit for the employees information to be updated in the database.
        2. If invalid information is filled when the admin submits, an error message is displayed and the employees information is not updated.

#### Processing narrative (PSPEC) for component main-page

User is redirected from the login page to the main-page. The main-page is responsible for displaying all the teams and their ranking for the current week.

* 1. Component main-page interface description
     1. At the very top a main page header is displayed indicating to the user that they are indeed on the main page. Underneath a ranking header is displayed and a field titled filter is shown allowing the user to enter the team name they desire and the filter would filter out the results displayed in the team list. A table is used to display all the teams and their ranking. The table pulls the information from the backend and displays the data to the user.
  2. Sub-Component main-page.table processing detail
     1. Interface description
        1. Input for table is company id and the output is the team names and ranks for all the teams in that company.
     2. Algorithmic model (e.g., PDL)
        1. Pull company id from token, search team and ranking table,
        2. Calculate total ranking of the week for each team, sort the table so the team with the most ranking would rise to the top
     3. Restrictions/limitations
        1. Connection to the database must be established for the information to be displayed.
     4. Local data structures
        1. Rank
           1. id: string
           2. teamName: string
           3. currentRank: number
           4. previousRank: number
     5. Performance issues
        1. N/A
     6. Design constraints
        1. Attributes of the overall software design (including data structures, OS features, I/O, and interoperable systems) that constrain the design of this sub-component are presented.

#### Processing narrative (PSPEC) for component team-rank-history

From the main-page.tableRanking if the user clicks on a team’s name they would get redirected to the team-rank-history component. Where they would see a chart displaying the rank history of a team.

* 1. Component team-rank-history interface description
     1. The component takes in a team id as input and returns a list of all rankings related to that specific team.
  2. Sub-Component Line-chart processing detail

A line chart component is used to display the data to the user. The component contains all the ranks displayed on the y axis vs all the weeks listed in the x axis.

* + 1. Interface description
       1. Charts is made of an x and y axis displaying ranks and the corresponding weeks.
    2. Algorithmic model (e.g., PDL)
       1. Send team id to backend,
       2. Search rank table and return all ranks,
       3. Display ranks to user in the line-chart format
    3. Restrictions/limitations
       1. Connection to the database must be established for the information to be displayed.
    4. Local data structures
       1. TeamRanks
          1. id: string
          2. week: Date
          3. rankScore: number
    5. Performance issues
       1. Information on topics that may affect the run-time performance, security, or computational accuracy of this sub-component are presented.
    6. Design constraints
       1. Attributes of the overall software design (including data structures, OS features, I/O, and interoperable systems) that constrain the design of this sub-component are presented.

#### Processing narrative (PSPEC) for component n

* 1. Component n interface description

A detailed description of the input and output interfaces for the component is presented.

* + 1. Sub-Component n.m processing detail

A detailed algorithmic description for each sub-component within the component n is presented. Section 3.2.3 is repeated for each of the m sub-components of component n

* + - 1. Interface description
         1. A description of sub-component m inputs and outputs is presented.
      2. Algorithmic model (e.g., PDL)
         1. The pseudocode listing for sub-component m is presented.
      3. Restrictions/limitations
         1. The external environment and/or infrastructure that must exist for sub-component m to operate correctly is provided.
      4. Local data structures
         1. The data structures used within sub-component m are presented.
      5. Performance issues
         1. Information on topics that may affect the run-time performance, security, or computational accuracy of this sub-component are presented.
      6. Design constraints

#### Software Interface Description

The software's interface(s) to the outside world are described.

* 1. External Machine Interfaces
     1. The interface for the system is going to be a web application that can be accessed on any device that can browse the web.
  2. External system interfaces
     1. The system interacts with an api to fetch the metrics which would be used to calculate the rankings of the teams. The link for the api’s interface is provided in the Supplementary information section labeled Client’s Api.
  3. Human interface
     1. An overview of any human interfaces to be designed for the software is presented. See Section 4.0 for additional detail.
     2. The interface that we chose will be designed in Html, CSS, and Typescript. The interface will have many controls found in basic web applications. The interface will provide users a way to quickly and easily check their rankings amongst other teams. For a more detailed description of the interface, view section 4.0.

### User Interface Design

The interface for the system is going to be a web application developed using angular framework. Angular is a TypeScript-based open-source web application framework that is used by many enterprises in the industry for its professional look and code reusability. Upon calling the website’s link, the user would be redirected to the login page where he/she would have to enter their credentials. If approved they would be redirected to the main page. There will be 3 main pages as of now. in addition to few sub-pages. the main page would hold information regarding the ranking of the team. and it will hold options that would enable the user to be redirected to the other pages. there will be a separate page to view the user’s account information, and a page that would have detailed data and graphs explaining each team’s ranking.

#### Login page

The login page is the first page the user sees. There they enter their credentials and if valid they automatically get transferred to the main page.

#### Main page

at the top right the logout button is located when pressed it would end the user session and log the user out. On the left of the page you will find the navigation bar. The main page is responsible for displaying the ranking for teams of the most recent week. You can use the arrows located at the right to move back and forth between weeks to view previous week rankings.

#### Profile page

Here is where the user updates their information and upload an image.

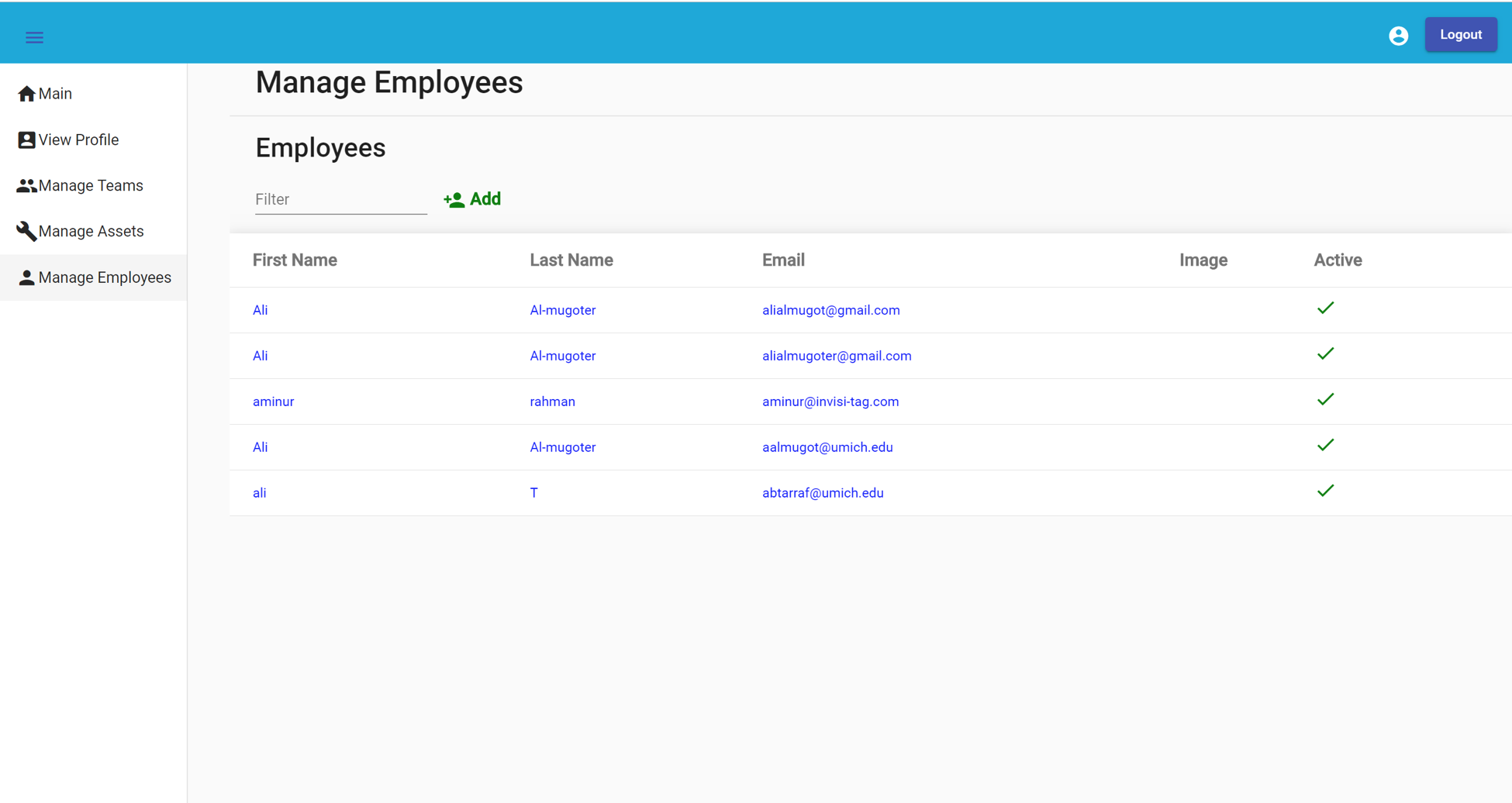
#### Manage Teams

Manage teams is only for admin users. Here you are able to add teams from the clients api to the database

#### Manage assets

Manage assets is where you add assets from the client’s api and set weight and cost for them.

#### Manage employees



Manage employees allows the admin to create accounts for employees. Also activate and deactivate their accounts.

#### Team detail

Team detail page shows a graph plotting ranking of each week of that specific team as points in the graph. This will allow the users to have an overview of the performance of the team.

#### Interface design rules

* + - * Design rules were mainly focused towards the user. Giving the user a friendly intuitive interface
      * Kept everything consistent through the design, if the header of a row is clickable in one table it must behave the same way for all the other tables in the website. That way the user will not be confused or left guessing on the use of a functionality.

#### Components available

* + - * We are using Angular as a framework and importing many libraries to use the parts of which to complete the design of the website, we will be listing some of them below:-
        + Angular material: Angular material provides a variety of components used as parts injected inside the website’s interface.
        + Mat-table: mat-table is used for all the tables that are listed on the website
        + Mat-dialog: all the dialog boxes that popup when user clicks on a feature that require them to select users to be added to a team
        + Mat-drawer and mat-navbar: used to set up the overall structure of the website placing different page navigation on the left side and having a toolbar at the top displaying important information for the user like logout and the name of the application.
        + Mat-input
      * Mat-dropdown
      * Bootstrap: bootstrap is used to provide graphs and charts for the website as well as some additional front-end design components.
      * Charts.js: is a library containing many different variety of charts that are minpulatibale and can dynamically changed. We are using the line chart to display ranking for teams.

#### Performance bounds

* + 1. The login step with the sharded database should work seemingly.
    2. Creating a user account will require the system to connect to 2 databases and insert the rows, and if one fails it should not save in either.

#### Identification of critical components

* + 1. Service should be tested thoroughly without using the client API.
    2. Team-management should be tested to see if everything works properly with the client API.
    3. Assets-management should be tested if everything works properly with the client API.
    4. The database has been sharded, this will require to check if creating users works properly with the partitioned databases. Shard manager will need to be tested for security reasons as well.

### Implementation Details

#### Gamification Service

RankUpdater – A time triggered function, it is triggered at 12 am daily

* Creates Rank data records for all active teams, records are created each new week Sunday - Saturday
* Retrieves the jobOrders for a company the previous day
* It saves all jobOrder scores (using tags and weighted calculations) to their correct active teams.

ShardCreator – A queue triggered function, it is triggered when an item is enqueued. It can parallel execute multiple items simultaneously, this is used to minimize the bottle neck. To overcome errors of items shard names matching from the nature of parallel executions. company row being inserted with matching shard name will throw an error due to a unique constraint applied to it. Items that cause an error to be thrown 5 times will be sent to a poison container. The poison container will have the same name as the queue with poison appended to the end e.g: queue name: create-shard. Poisoned container: create- shard-poison. The error can be seen on azure.

* The shardCreator will get the last shard name created and increment it, e.g shard1 => shard2
* It will then add the company to the company table reserving the shard name and setting the created date for other items to increment
* It will then create the database shard using the shard name
* And then create tables in that shard
* It will then send the admin account to the web api for a user row to be created for this shard. (it is recommended to remove this call from the from queue trigger and the functionality to be removed from the web api. Instead a queue should be created that is tasked with inserting the admin accounts when shards are created the items will be enqueued via the shardCreator and also any other area(programs) the client finds necessary. This method will be a more secure approach than the current process. It will also Allow multiple requests to be made if an admin account creation fails, in comparison with the current mechanism only being able to create the admin account during shard creation. It will overcome both shortcomings of the current process and still preserve the core functionality with added benefits)
* Any changes made to the company table or the shard database schema must be reflected here. If the schema is not maintained it is probable that version issues will occur. The shard database schema must be constantly updated and maintained any time that a change is made to the database.

#### Gamification WebApi

##### Database

###### ShardManager

Begin

CREATE TABLE [dbo].[Company](

[Id] [uniqueidentifier] NOT NULL,

[Name] [varchar](100) NOT NULL,

[ShardName] [varchar](100) NOT NULL UNIQUE,

[CreatedDate] [datetime] NOT NULL DEFAULT (getdate()),

primary key([Id]));

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[AspNetUsers](

[Id] [nvarchar](450) NOT NULL,

[UserName] [nvarchar](256) NULL,

[NormalizedUserName] [nvarchar](256) NULL,

[Email] [nvarchar](256) NULL,

[NormalizedEmail] [nvarchar](256) NULL,

[EmailConfirmed] [bit] NOT NULL,

[PasswordHash] [nvarchar](max) NULL,

[SecurityStamp] [nvarchar](max) NULL,

[ConcurrencyStamp] [nvarchar](max) NULL,

[PhoneNumber] [nvarchar](max) NULL,

[PhoneNumberConfirmed] [bit] NOT NULL,

[TwoFactorEnabled] [bit] NOT NULL,

[LockoutEnd] [datetimeoffset](7) NULL,

[LockoutEnabled] [bit] NOT NULL,

[AccessFailedCount] [int] NOT NULL,

[CompanyId] [nvarchar](450) NULL,

CONSTRAINT [PK\_AspNetUsers] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[AspNetUserTokens](

[UserId] [nvarchar](450) NOT NULL,

[LoginProvider] [nvarchar](450) NOT NULL,

[Name] [nvarchar](450) NOT NULL,

[Value] [nvarchar](max) NULL,

CONSTRAINT [PK\_AspNetUserTokens] PRIMARY KEY CLUSTERED

(

[UserId] ASC,

[LoginProvider] ASC,

[Name] ASC

)WITH (STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

ALTER TABLE [dbo].[AspNetUserTokens] WITH CHECK ADD CONSTRAINT [FK\_AspNetUserTokens\_AspNetUsers\_UserId] FOREIGN KEY([UserId])

REFERENCES [dbo].[AspNetUsers] ([Id])

ON DELETE CASCADE

GO

ALTER TABLE [dbo].[AspNetUserTokens] CHECK CONSTRAINT [FK\_AspNetUserTokens\_AspNetUsers\_UserId]

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[AspNetUserRoles](

[UserId] [nvarchar](450) NOT NULL,

[RoleId] [nvarchar](450) NOT NULL,

CONSTRAINT [PK\_AspNetUserRoles] PRIMARY KEY CLUSTERED

(

[UserId] ASC,

[RoleId] ASC

)WITH (STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[AspNetUserRoles] WITH CHECK ADD CONSTRAINT [FK\_AspNetUserRoles\_AspNetRoles\_RoleId] FOREIGN KEY([RoleId])

REFERENCES [dbo].[AspNetRoles] ([Id])

ON DELETE CASCADE

GO

ALTER TABLE [dbo].[AspNetUserRoles] CHECK CONSTRAINT [FK\_AspNetUserRoles\_AspNetRoles\_RoleId]

GO

ALTER TABLE [dbo].[AspNetUserRoles] WITH CHECK ADD CONSTRAINT [FK\_AspNetUserRoles\_AspNetUsers\_UserId] FOREIGN KEY([UserId])

REFERENCES [dbo].[AspNetUsers] ([Id])

ON DELETE CASCADE

GO

ALTER TABLE [dbo].[AspNetUserRoles] CHECK CONSTRAINT [FK\_AspNetUserRoles\_AspNetUsers\_UserId]

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[AspNetUserLogins](

[LoginProvider] [nvarchar](450) NOT NULL,

[ProviderKey] [nvarchar](450) NOT NULL,

[ProviderDisplayName] [nvarchar](max) NULL,

[UserId] [nvarchar](450) NOT NULL,

CONSTRAINT [PK\_AspNetUserLogins] PRIMARY KEY CLUSTERED

(

[LoginProvider] ASC,

[ProviderKey] ASC

)WITH (STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

ALTER TABLE [dbo].[AspNetUserLogins] WITH CHECK ADD CONSTRAINT [FK\_AspNetUserLogins\_AspNetUsers\_UserId] FOREIGN KEY([UserId])

REFERENCES [dbo].[AspNetUsers] ([Id])

ON DELETE CASCADE

GO

ALTER TABLE [dbo].[AspNetUserLogins] CHECK CONSTRAINT [FK\_AspNetUserLogins\_AspNetUsers\_UserId]

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[AspNetUserClaims](

[Id] [int] IDENTITY(1,1) NOT NULL,

[UserId] [nvarchar](450) NOT NULL,

[ClaimType] [nvarchar](max) NULL,

[ClaimValue] [nvarchar](max) NULL,

CONSTRAINT [PK\_AspNetUserClaims] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

ALTER TABLE [dbo].[AspNetUserClaims] WITH CHECK ADD CONSTRAINT [FK\_AspNetUserClaims\_AspNetUsers\_UserId] FOREIGN KEY([UserId])

REFERENCES [dbo].[AspNetUsers] ([Id])

ON DELETE CASCADE

GO

ALTER TABLE [dbo].[AspNetUserClaims] CHECK CONSTRAINT [FK\_AspNetUserClaims\_AspNetUsers\_UserId]

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[AspNetRoles](

[Id] [nvarchar](450) NOT NULL,

[Name] [nvarchar](256) NULL,

[NormalizedName] [nvarchar](256) NULL,

[ConcurrencyStamp] [nvarchar](max) NULL,

CONSTRAINT [PK\_AspNetRoles] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[\_\_EFMigrationsHistory](

[MigrationId] [nvarchar](150) NOT NULL,

[ProductVersion] [nvarchar](32) NOT NULL,

CONSTRAINT [PK\_\_\_EFMigrationsHistory] PRIMARY KEY CLUSTERED

(

[MigrationId] ASC

)WITH (STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

end

###### Shard

Begin

Create Table [Team]

([Id] uniqueidentifier not null,

[Name] varchar(100) not null,

[IsActive] [bit] not null default 1,

primary key([Id]));

Create Table [User]

([Id] uniqueidentifier not null,

[FirstName] varchar(100) not null,

[LastName] varchar(100) not null,

[Email] [varchar](100) NOT NULL,

[Image] [varbinary](max) NULL,

[Role] [varchar](100) NOT NULL,

[IsActive] [bit] NOT NULL default 1,

primary key([Id]));

Create Table [Asset]

([Id] [uniqueidentifier] not null,

[Name] varchar(100) not null,

[Weight] int check ([Weight] between 1 and 10) not null default 1,

[Cost] decimal(18,2) check ([Cost] between 0.00 and 9999999999999999.99) not null default 0.00,

[Rfid] [varchar](30) Not NULL UNIQUE,

primary key([Id]));

Create Table [Rank]

([Id] [uniqueidentifier] NOT NULL,

[TeamId] uniqueidentifier not null,

[week] datetime not null,

[ActualScore] bigint check ([ActualScore] between 0 and 9223372036854775807) not null default 0 ,

[TotalScore] bigint check ([TotalScore] between 0 and 9223372036854775807) not null default 0,

[ActualWeightedScore] bigint check ([ActualWeightedScore] between 0 and 9223372036854775807) not null default 0 ,

[TotalWeightedScore] bigint check ([TotalWeightedScore] between 0 and 9223372036854775807) not null default 0,

primary key([Id]),

Foreign key([TeamId]) references[Team]);

Create Table [MemberOf]

([TeamId] uniqueidentifier not null,

[UserId] [uniqueidentifier] NOT NULL,

primary key([TeamId], [UserId]),

Foreign key([UserId]) references[User],

Foreign key([TeamId]) references[Team]);

End

### Testing/SQA

#### Statement of Scope

The scope of the testing will focus on providing a software that functions properly, the essential functionalities have been implemented and are as follows: user login, team management, user management, asset management, rank history, and rankUpdater Service. The main focus of testing is to provide a solid working foundation with these functionalities allowing the client to be confident in the product they are deploying into a production environment.

* Unit Testing - Using Black Box Testing
  + Web Application
  + Database
* Integration Testing
  + Web Application
  + Database
* Validation Testing - Test software as whole
  + Web Application
  + Database
* High-order Testing
  + Web Application
  + Database

#### Major Constraints

Any business, product line or technical constraints that will impact the manner in which the software is to be tested are noted here.

* Development team does not have enough members to perform an accurate stress test related testing on the software.
* Software will not be connected to a host of the clients choosing until the project is complete and signed off on, so no accurate testing of page load times can be made.

#### Test Plan

The end goal is for the software to be bug free. To ensure there are no defects we will spend a large amount of time testing the software. Below is the description of the testing procedure and strategy.

#### Software (SCI’s) to be tested

The software to be tested is identified by name. Exclusions are noted explicitly.

* + - * Login
        + Login will be tested using multiple user accounts. Both Admin and Employee security levels will be tested.
      * Employee User Interface

The employee interface will be tested.

* + - * + User profile page will be tested to ensure the user information is displayed.
        + All data fields will be edited and we will test the submit button to make sure changes are saved to the database.
        + Users Team ranking page will be tested to see if all teams are ranked in the correct order.
        + We will also test if the user has the ability to click on their team to view their teams analytics.
        + We will test to see if team history is displayed when user selects page.
      * Manager Admin Interface

The Managers admin interface will be tested.

* + - * + Manage team page will be tested to see if add team button creates a new team.
        + Manage team page will be tested to see if clicking add on an employee will add them to a team.
        + Manage team page will be tested to see if clicking remove on an employee will remove them from a team.
        + Manage employee page will be tested to see if on load a list of all employees are displayed.
        + Manage employee page will be tested to see if clicking add routes to a blank employee information page to create a new employee.
        + Add employee page will be tested to see if clicking submit after entering an employee's information inserts the data in the database.
        + Edit employee page will be tested to see if clicking submit after updating an employee's information updates the data in the database.
        + Manage employee page will be tested to see if clicking remove on an employee removes the user from the database.
      * RankUpdater
        + Test to see if the function creates a record for active teams weekly
        + Test if teams that are added also have ranks generated for them
        + Test to see if the ranks updated with the correct score information
* totalScore, actualScore, totalWeightedScore, and actualWeightedScore
  + - * + Test to see if each company rank record are updated

#### Testing Strategy

#### Unit Testing

For unit testing we will be testing each component individually. We will be using black box testing method. Each component will be tested by inputting data and checking the output. We will do this to find any possible bugs/defects in the software.

#### Integration Testing

For integration testing we will be testing each component before committing the remote branch to the master branch. When a functionality is complete and has been tested on the remote branch, it will be reviewed by two other team members. Once both members approve the code, it will be pushed to the master branch and tested to see if the functionality integrates properly with the software.

#### Validation testing

For validation testing we will work with the client to ensure the software is up to the clients requirements. We will go through every feature with them and have them test out the software the way they will use it.

Black box testing will be used for our validation testing, since the client does not have the ability to see what's happening in the backend. Data will be inputted and the output will be analyzed for accuracy.

#### High-order testing

For high-order testing we will be using different test methods to test different conditions.

* Stress Testing
  + For stress testing we will be testing at varying frequencies with abnormal data inputs in large sums to validate the braking point of the software.
* Performance Testing
  + For performance testing we will be testing the range of performance to know the software compatibility and functionality.
* Security Testing
  + For security testing we will be testing the two-level user authentication at login as well as SQL injection testing.

#### Testing resources and staffing

Information on testing resources and staffing can be found in section 3.3 of this document.

#### Test work products

The work products produced as a consequence of the testing strategy are identified in section 3.4 of this document. Please refer to section 3.4 for further information on test work products.

#### Test record keeping

Mechanisms for storing and evaluating test results are specified in section 3.5 of this document. Please refer to section 3.5 for further information regarding test record keeping.

#### Test metrics

Our testing metrics will be dependant on the quantity of test cases and the quality.

We will use a simple percentages to determine how successful testing is.

We will divide passed test by all tests.

This will indicate the percentage of tests we have accomplished.

#### Testing tools and environment

A description of the test environment, including tools, simulators, specialized hardware, test files, and other resources is presented here.

Testing environment will be using visual studio for back end. The front end testing will be using the developer tools built into modern browsers e.g chrome F12 tool. The front end testing environment will be done using chrome, edge and firefox.

We will be simulating the workload to the website and database using sample data provided by the client, the sample data is based on real data from companies that use their software.

The sample data includes 5 companies with many FixedJobs (teams), Tags (assets) and postedJobs (ranking data).

The ranking data will be simulated by either asking the client to update posted jobs daily or adding data in manually to the database. This will show the ranking system with different input.

#### Test schedule

A detailed schedule for unit, integration, and validation testing as well as high order tests is described.

* Unit Testing
  + 07/11/2019 - 07/14/2019
* Integration Testing
  + 07/15/2019 - 07/17/2019
* Validation Testing
  + 07/28/2019 - 07/29/2019
* High-order Testing
  + Stress Testing
    - 07/15/2019
    - 07/28/2019
  + Performance Testing
    - 07/28/2019
  + Security Testing
    - 07/11/2019 - 07/18/2019
    - 07/28/2019

#### Testing procedure

The overall procedure for software testing is described.

* + 1. Unit Test Cases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Type** | **Description** | **Input** | **Expected Output** |
| 1 | Black Box Testing | Testing log in. | Username: admin  Password: password | User is logged in. |
| 2 | Black Box Testing | Testing if admin can create a team. | Admin creates team. | Team is created. Team is saved into database. Admin is now able to edit team. |
| 3 | Black Box Testing | Testing if admin can deactivate a team. | Admin deactivated team. | Team is deactivated. Team is not shown on ranking page, and does not show up as part of active teams. |
| 4 | Black Box Testing | Testing if admin can add employee to team. Through team management page | Admin adds employee to team. | Employee is added to the team. Changes made to team are saved in database. |
| 5 | Black Box Testing | Testing if admin can remove employee from team. Through team management pag. | Admin removes employee from team through team management page. | Employee is removed from team. Changes made to team are saved in database. |
| 6 | Black Box Testing | Testing if admin can create employee account. | Admin creates employee account. | Employee account is created. Credentials are generated for employee. Credentials are saved in database and sent to user email. |
| 7 | Black Box Testing | Testing if admin can deactivate employee account. | Admin deactivates employee account. | Employee account is deactivated. Credentials are revoked. |
| 8 | Black Box Testing | Testing if admin can create a team with the same name. | Admin creates team with same name. | Account is not created. |
| 9 | Black Box Testing | Testing if admin can add the same employee to a team twice. Via team management page | Admin selects employees to add. | Team members do not appear |
| 10 | Black Box Testing | Testing if admin can change metrics. | Admin Changes weight of asset. | Weight of asset is changed. Metrics are updated. |
| 11 | Black Box Testing | Testing if user can view team history. | User selects view team history. | Team history is displayed to user. User can view all their ranking history. |
| 12 | Black Box Testing | Testing if user can view weekly ranking. | User selects view weekly ranking. | Weekly ranking is displayed to user. User can see what ranking position they are in and the ranking of all the other teams. |
| 13 | Black Box Testing | Testing if user can view all time ranking. | User selects view all time ranking. | All time ranking is displayed to the user. user can view their all-time ranking and the all-time ranking of the other teams. |
| 14 | Black Box Testing | Testing if user can change their password. | User selects change password and confirms new password. | Users password is changed and saved in database. |
| 15 | Black Box Testing | Testing if user can use the same password when changing password. | User attempts to change password to existing password. | User password is not changed. |
| 16 | Black Box Testing | Testing if user can change their account information. | User makes changes to account information. | User account information is changed and saved in database. |
| 17 | Black Box Testing | Testing if user can save account information if nothing was changed. | User selects change account information and saves without making any changes. | User account information is changed and saved in database. |
| 18 | Black Box Testing | Test to see if invalid credentials are allowed. | Username:invalid  password:invalid | Error is displayed saying the credentials are invalid |
| 19 | Black Box Testing | Test to see the result of running the service function when no assets are in the table. | N/A | The service should not create a rank record unless the total score is greater than 0 |
| 20 | Black Box Testing | Test to see if deactivated teams show on the leaderboard. | Deactivate team | Deactivated teams should show on the archived leaderboard, but not new records |
| 21 | Black Box Testing | Test to see if a deleted employee can still log in | Delete employee and then attempt to log in with the credentials | The “deleted” employee should get an error that is the same as invalid credentials |
| 22 | Black Box Testing | Test to see if job reports are duplicated/recounted into ranking for the week when past 12 and the service runs | Have the service run at 12, then have a jobreport be entered while its running. It should have a timestamp past 12 | The job reports that come after 12 should not be added into the score of rank but saved for the next day. |
| 23 | Black Box Testing | Test to see if employee account can be created twice. | Create duplicate employee account. | An error is thrown. “Employee already exists”. |
| 24 | Black Box Testing | Test to see if deactivated employee shows in a team list. | Deactivate an employee and view the team list. | Employee should not be present in the team list. |
| 25 | Black Box Testing | Testing if user receives email containing username and password once their account is created. | Create new employee account. | Employee should receive an email containing their username and password. |
| 26 | Black Box Testing | Testing log out. | Click logout button | User is logged out |
| 27 | Black Box Testing | Testing if admin can add employee to team. Employee management page | Admin adds employee to teams. | Employee is added to teams. Changes made to team are saved in database. |
| 28 | Black Box Testing | Testing if admin can remove employee from team. Through employee management paeg. | Admin removes employee from team through team management page. | Employee is removed from teams. Changes made to team are saved in database. |
| 29 | Black Box Testing | Testing if admin can add the same employee to a team twice. Via employee management page | Admin selects teams to add employee to. | Teams the employee is a member of do not appear |
| 30 | Black Box testing | Test if user can import tags (assets) from api | Admin select asset to import through asset management | Assets are imported |
| 31 | Black box testing | Test if tags can be duplicated | Admin selects to import assets | Only assets that are not in database are shown |
| 32 | Black box testing | Test if users created by admin are inserted to correct shard and uses correct shard when accessing database | Create user account | User has correct companyId, and is directed to the correct shard |
| 33 | Black box testing | Test if users can access different shards | Login as user | User is only shown data from shard he is associated with |
| 34 | Black box testing | Employees should only be able to see ranks and user profile, test if he can see other menus | employee logs in and opens navigation bar and | Rank menu and user profile buttons are displayed and are functional |
| 35 | Black box testing | Admin should be able to see all the pages for admin role including: team management, user management, asset management, ranking menu | Admin logs in and opens navigation bar | Navigation bar displays  team management, user management, asset management, ranking menu buttons and are functional |

## 

* + 1. Integration Testing
       - Testing procedure for integration
         * For integration testing we will be testing the software components individually and then testing the software to make sure it has been integrated properly.
       - Stubs and drivers required
         * Each component will be tested with a test function (stubs) which will be used to test the component without having all the functions of the software available.
       - Test cases and their purpose
         * Each test case will be tested once the software has been integrated into the master branch. We will do this to ensure the software still functions the same once a new functionality has been integrated. We will also test to make sure the software retrieves data from Invisi-tag once the software is up and running.
       - Expected results
         * A
    2. Validation Testing
       - Testing procedure for validation
         * For validation testing we will be testing the software with our client to make sure the software is up to the clients requirements. All test cases will be tested once the software is fully complete.
       - Expected results
         * We expect all tests to have positive results. All functionalities should satisfy the clients requirements. All buttons should work and function as specified. All data should be stored correctly in the database.
       - Pass/fail criterion for all validation tests
         * All test cases should pass to show the client the validity of the software being presented to them. Any issues presented will be fixed and tested again. Any requirements not met will be added and tested in front of the client.
    3. High-order testing (a.k.a. System Testing)
       - Security testing
         * For security testing we will be testing the two-level user authentication at login as well as SQL injection testing.
       - Stress testing
         * For stress testing we will be testing at varying frequencies with abnormal data inputs in large sums to validate the braking point of the software.
       - Performance testing
         * For performance testing we will be testing the range of performance to know the software compatibility and functionality.
       - Pass/fail criterion for all validation tests
         * All high order tests should pass without any issues. Performance of the software will be dependent on the host's server response time.

#### Testing resources and staffing

Since this is a web application and requires an internet connection and browser to work, we will be using our own laptops and mobile devices to test the software. All members of the development team will be testing the software.

#### Test Work Products

The work products produced as a consequence of the testing procedure are identified.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test #** | **Test Description/Basis** | **Test Conditions** | **Test Coverage** |
| 1 | Testing log in. | Pass | Tested |
| 2 | Testing if admin can create a team. | Pass | Tested |
| 3 | Testing if admin can delete team. | Pass | Tested |
| 4 | Testing if admin can add employee to team. | Pass | Tested |
| 5 | Testing if admin can remove employee from team. | Pass | Tested |
| 6 | Testing if admin can create employee account. | Skip | Not Tested |
| 7 | Testing if admin can deactivate employee account. | Skip | Not tested |
| 8 | Testing if admin can create a team with the same name. | Skip | Not Tested |
| 9 | Testing if admin can add the same employee to a team twice. | Skip | Not Tested |
| 10 | Testing if admin can change metrics. | Skip | Not Tested |
| 11 | Testing if user can view team history. | Pass | Tested |
| 12 | Testing if user can view weekly ranking. | Pass | Tested |
| 13 | Testing if user can view all time ranking. | Pass | Tested |
| 14 | Testing if user can change their password. | Skip | Not Tested |
| 15 | Testing if user can use the same password when changing password. | Skip | Not Tested |
| 16 | Testing if user can change their account information. | Skip | Not Tested |
| 17 | Testing if user can save account information if nothing was changed. | Skip | Not Tested |
| 18 | Test to see if invalid credentials are allowed. | Pass | Tested |
| 19 | Test to see the result of running the service function when no assets are in the table. | Skip | Not Tested |
| 20 | Test to see if deactivated teams show on the leaderboard. | Pass | Tested |
| 21 | Test to see if a deleted employee can still log in | Pass | Tested |
| 22 | Test to see if job reports are duplicated/recounted into ranking for the week when past 12 and the service runs | pass | Tested |
| 23 | Test to see if employee account can be created twice. | pass | Tested |
| 24 | Test to see if deactivated employee shows in a team list. | pass | Tested |
| 25 | Testing if user receives email containing username and password once their account is created. | pass | Tested |
| 26 | Testing log out. | pass | tested |
| 27 | Testing if admin can add employee to team. Employee management page | pass | tested |
| 28 | Testing if admin can remove employee from team. Through employee management paeg. | pass | tested |
| 29 | Testing if admin can add the same employee to a team twice. Via employee management page | pass | tested |
| 30 | Test if user can import tags (assets) from api | pass | tested |
| 31 | Test if tags can be duplicated | pass | tested |
| 32 | Test if users created by admin are inserted to correct shard and uses correct shard when accessing database | pass | tested |
| 33 | Test if users can access different shards | pass | tested |
| 34 | Employees should only be able to see ranks and user profile, test if he can see other menus | pass | tested |
| 35 | Admin should be able to see all the pages for admin role including: team management, user management, asset management, ranking menu | pass | tested |

#### Test record keeping and test log

We will be using a table for storing and evaluating test results. The test log is used to maintain a chronological record of all tests and their results. Below is a table used to log all tests.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test #** | **Test Description** | **Test Results** | **Date Tested** |
| 1 | Testing login. | User login successful. | 07/11/2019 |
| 2 | Testing if admin can create a team. | Admin successfully created a team. | 07/11/2019 |
| 3 | Testing if admin can delete team. | Admin can successfully delete a team. | 07/11/2019 |
| 4 | Testing if admin can add employee to team. | Admin can add an employee to a team. | 07/11/2019 |
| 5 | Testing if admin can remove employee from team. | Admin can remove an employee from a team. | 07/11/2019 |
| 6 | Testing if admin can create employee account. | Admin can create employee account. | 07/11/2019 |
| 7 | Testing if admin can deactivate employee account. | Admin can deactivate employee account. | 07/11/2019 |
| 8 | Testing if admin can create a team with the same name. | Admin cannot create team with same name. | 07/12/2019 |
| 9 | Testing if admin can add the same employee to a team twice. | Admin cannot add the same employee to a team twice. | 07/12/2019 |
| 10 | Testing if admin can change metrics. | Admin can change metrics. | 07/12/2019 |
| 11 | Testing if user can view team history. | User can view team history. | 07/12/2019 |
| 12 | Testing if user can view weekly ranking. | User can view weekly ranking. | 07/12/2019 |
| 13 | Testing if user can view all time ranking. | User can view all time ranking. | 07/12/2019 |
| 14 | Testing if user can change their password. | User can change their password. | 07/12/2019 |
| 15 | Testing if user can use the same password when changing password. | User cannot change password. | 07/12/2019 |
| 16 | Testing if user can change their account information. | User can change account information. | 07/12/2019 |
| 17 | Testing if user can save account information if nothing was changed. | N/A | 07/13/2019 |
| 18 | Test to see if invalid credentials are allowed. | Invalid credentials are not allowed. | 07/13/2019 |
| 19 | Test to see the result of running the service function when no assets are in the table. | Default is accumulated | 07/13/2019 |
| 20 | Test to see if deactivated teams show on the leaderboard. | Deactivated members did not show | 07/13/2019 |
| 21 | Test to see if a deleted employee can still log in | A deleted employee cannot log in. | 07/13/2019 |
| 22 | Test to see if job reports are duplicated/recounted into ranking for the week when past 12 and the service runs | Points where not duplicated | 07/13/2019 |
| 23 | Test to see if employee account can be created twice. | Error is thrown | 07/14/2019 |
| 24 | Test to see if deactivated employee shows in a team list. | Deactivated employee did not show | 07/14/2019 |
| 25 | Testing if user receives email containing username and password once their account is created. | Email was received with correct credentials | 07/14/2019 |
| 26 | Testing log out. | User logged out | 07/14/2019 |
| 27 | Testing if admin can add employee to team. Employee management page | Employee was addded | 07/14/2019 |
| 28 | Testing if admin can remove employee from team. Through employee management paeg. | Employee was removed | 07/14/2019 |
| 29 | Testing if admin can add the same employee to a team twice. Via employee management page | Error thrown | 07/14/2019 |
| 30 | Test if user can import tags (assets) from api | Tags were imported | 07/14/2019 |
| 31 | Test if tags can be duplicated | Tags where not added, error thrown | 07/14/2019 |
| 32 | Test if users created by admin are inserted to correct shard and uses correct shard when accessing database | Users were added to correct database with correct company | 07/14/2019 |
| 33 | Test if users can access different shards | User cannot access different shards | 07/14/2019 |
| 34 | Employees should only be able to see ranks and user profile, test if he can see other menus | Employee only saw pages that his role allowed | 07/14/2019 |
| 35 | Admin should be able to see all the pages for admin role including: team management, user management, asset management, ranking menu | Admin saw pages that his role allowed | 07/14/2019 |

#### SQA Task Overview

* + 1. Description of SQA task 1

The SQA team will have the responsibility of ensuring the programming quality does not suffer the larger the program becomes. The SQA team will have to understand the desired quality from the client and ensure that it is met.

* + 1. Work products and documentation

The SQA team must measure the quality of each functionality, metrics will be based on SQA attributes like usability ect. They will then communicate that information to the client and the software development team. The SQA team will have to research how to measure those metrics and the best method of testing them. The advantage will be when reviewing code that was written by another programmer attributes like readability and understandability would be measured.

* + 1. Description of SQA task 2

The SQA Team will have the responsibility of reviewing each pull request (merge to master) and ensure that the code follows the coding convention for the code base.

* + 1. Work products and documentation

The SQA will comment on each branch. They will also have to do research to understand the best programming conventions and the reasoning.

* + 1. Description of SQA task 3

The SQA Team will have to ensure program security and encryption standards are met.

* + 1. Work products and documentation

The SQA must research correct security practices for certain functionality. For example best method to encrypt passwords, standard to prevent SQL injection attacks, XSS(cross site scripting) attacks, and ect.

* + 1. Description of SQA task 4

The SQA team will ensure that the correct data is rolled back during a program crash. They will also have to focus on the error handling aspect of the program.

* + 1. Work products and documentation

The SQA team will have to analyses each task and ensure that the error handling measures are included in the implementation.

* + 1. Description of SQA task 5

As a result of the previous tasks the SQA team will have the responsibility of ensuring the correct refactoring is done after each review. And resubmit.

* + 1. Work products and documentation

Reviewing, documentation of events, assign refactoring task.

#### SQA Resources

People, hardware, software, tools, and other resources required to perform SQA tasks are noted here.

An SQA member will rotate through out the project to ensure that each member has the opportunity to program and review. There will only be one SQA member at a time there main focus being the tasks above. When there period is over the next member will assume the SQA role. The rotation cycles are going to last weekly/bi weekly.

The team will use azure devops (visual studio team service) to keep track of updates, test if they work on the intended machines, approve pull request, and merage. Azure devops will help organize the project and automate some features like virtual machine testing. This will allow the SQA team to have a view over all commits being made.

The software the SQA team will use are quality metric software, refactoring software. Quality metric software will analyze the project and produce statical graphs identifying the quality of the program. Based on this software can suggestion some type of refactoring to improve program quality.

Ndepend is potteintal .net package that would be used.

For SQL optimization SQL Server Management Studio has features that allow you to see execution plans for queries/commands. The SQA team will test the queries using these features and optimize the queries if need be.

The SQA team will still have to do extra research added to what the development team has done.

#### Reviews and Audits

* + 1. Generic Review Guidelines

A set of guidelines for all formal technical reviews (FTR's) is presented in this section.

* + - * Conducting a Review

General guidelines for conducting a review.

* + - * All members of the development team will review test cases. We will also review test cases with the client. If any changes need to be made all team members must approve the changes. If there are any performance issues when the client uses the software, those issues will need to be fixed.
    1. Roles and Responsibilities

The roles people play during a FTR and the responsibilities of each player.

* + - * Perform all SQA tasks: Ali Tarraf
      * Perform all SQA tasks: Ali Al-Mugoter
      * Perform all SQA tasks: Steve Karim
    1. Review work products

Documents, forms, lists produced as a consequence of the FTR.

* + - * Bi-weekly all team members will generate a work report. Each report will list the work done by that team member. It will list any problems encountered, and if they were able to solve that problem or not. We will meet with the client and answer any questions they have. We will ask any questions we have and schedule the next meeting.
    1. Formal Technical Reviews

A description of the specific character and intent of each major FTR conducted during the software process.

* + 1. Description of review n

The sections that follow are included for each Section 6.17.5.n

The sections that follow represent typical reviews conducted as part of a software engineering project and are included as part of Section 3.2.n

* + - * Walkthroughs Review

For this part we will be focusing on the integration of parts that each member worked on. These parts include user interfaces, backend, and database. Each member will do a walk through of the parts they worked on so all members have a clear understanding of what was worked on.

* + - * Inspections Review

For this part we will focus on the quality and accuracy of the parts each member designed. Since this is team of three, two team members will review the others work and create test cases without the input of the original designer.

* + 1. System specification review
       - Description and focus of the review

The system specification is part of the overall system engineering process of allocating and formally defining requirements and occurs after the system/software level decisions have been made. It will provide a forum to analyze the proposed design of the software. The SSR is usually changed after every meeting with the client.

* + - * Timing of the review

The system specification review will be held after the completion of the system specification. This is to ensure the design of the software is sound and will not cause any issues for the software engineers in the future.

* + - * Work products produced

A summary report of the system specification review will be created by the SQA leader. The report will include all changes if any made to the softwares design. If any defects in the design are identified the team will try to find a solution.When all defects have been fixed the system specification will be updated with all design changes made.

* + - * Review checklist
        + Is the design of the software the best possible solution.
        + Are there any obvious flaws in the design.
        + Do any enhancements to the current software needed.
    1. Software project plan review

Description and focus of the review

The purpose of this review is to review the SPMP and overlook the entire project. More information can be found in the SPMP document.

* + - * Timing of the review

The software project plan review will be held when the the software project plan is completed. This is necessary to do the review within a few weeks of completing the software project plan so that scheduling and cost estimate are within reason.

* + - * Work products produced

A summary report of the software project plan review will be created. Any over or under estimations that have been made will be put in the report. The project plan will be updated with any problems found.

* + - * Review checklist
        + Is this enough to complete the project?
        + Do we have enough resources to complete the project?
        + Has enough time been allocated to develop a subsystem.
    1. RMMM review

Description and focus of the review

* + 1. The Risk Mitigation, Monitoring, and Management (RMMM) document is used to monitor, manage, and prevent any possible risks. More information can be found in the RMMM document.
       - Timing of the review

The RMMM review will be held after completing the RMMM document.

* + - * Work products produced

A summary report of the RMMM review will be created. This report will include any possible risks that have not been covered, and any risks that have been accounted for but not managed properly.

* + - * Review checklist
        + Have all risks been accounted for?
        + Have all accounted risks been managed correctly?
    1. Requirements reviews (models, specification)

Description and focus of the review

For this part we will review any changes made to Software Requirements Specifications (SRS) document, since requirements are subject to change. If any requirements are changes the document will be updated with those changes. More information can be found in the SRS document.

* + - * Timing of the review

The requirements specification review will be held when the requirement specification has been completed.

* + - * Work products produced

A summary report of the requirements specification review will be created. It will include any defects or enhancements that have been found.

* + - * Review checklist
        + Is design the best possible solution?
        + Are there any flaws in the design?
        + Do any enhancements need to be made to the software?
    1. Data design review

Description and focus of the review

The data design document provides an architectural overview of the data flow between each interface, and forms to the database. More information can be found in the architectural design document.

* + - * Timing of the review

The data design review will be held when the system specification and requirements specification have been completed.

* + - * Work products produced

A summary report of the data design review will be created. This report will include data design flaws have been found.

* + - * Review checklist
        + Is the data design a possible solution?
        + Do all data objects make sense?
        + Do all data objects enhance the software?
    1. Architectural design review

Description and focus of the review

The architectural design review covers the entire projects design, laout, and data flow. More information can be found in the architectural design document.

* + - * Timing of the review

The architectural design review will be held when the system specification and requirements specification have been held.

* + - * Work products produced

A summary report of the architectural design review has been completed. This will include any defects that have been found.

* + - * Review checklist
        + Is the architectural design the best solution?
        + Will any changes further enhance the softwares performance?
    1. Interface (GUI) design review

Description and focus of the review

The interface design review is a review of the user interface. It typically deals with identifying usability problems related to the layou, logic flow, and structure of the interface. Upon request of the client, we will redesign the interface of the previous version of the software.

* + - * Timing of the review

The Interface design review will be held when the system specification and requirements specification have been completed.

* + - * Work products produced

A summary report of the Interface design review will be created. This will include any changes that have been mad or need to be made to the GUI.

* + - * Review checklist
        + Is the GUI pleasant to look at?
        + Does the GUI make sense?
    1. Component design review(s)

Description and focus of the review

The component design review provides a forum where questions can be answered. It is used to optimize the design of the product through a systematic review.

* + - * Timing of the review

The component design review will be held upon completion of the interface design review.

* + - * Work products produced

A summary report of the component design review will be created.

* + - * Review checklist
        + Does the component design the best solution?
    1. Code Reviews

Description and focus of the review

Code reviews are conducted by the development team and occur whenever a piece of code is written and ready for review. Once one developer is ready to submit their code two developers will review their code for mistakes before it is committed.

* + - * Timing of the review

The code review will be held when the product has neared completion.

* + - * Work products produced

A summary report of the code review will be created. This is to ensure that the software engineers have produced reliable code.

* + - * Review checklist
        + Is the source code easy to maintain?
        + Is the code easy to read?
        + Is the code reliable?
        + Is the code efficient?
    1. Test specification review

Description and focus of the review

The test specification document describes both the test plan and the test procedure.

* + - * Timing of the review

The test specification review will be held upon completion of the code review.

* + - * Work products produced

A summary report of the test specification review will be created.

* + - * Review checklist
        + Do the tests cover all possible issues?
    1. Change control reviews and audits

Description and focus of the review

These audits consist of evaluations of companywide procedures and standards that safeguard electronic information from loss, damage, and unintended disclosure. Change control is the process used to request, review, specify, plan, approve, and implement changes to a system. This is a review of an organization’s change control environment.

* + - * Timing of the review

The change control review will be held when the change control document has been completed.

* + - * Work products produced

A summary report of the change control review will be created. This includes any changes that can be added to the change control document.

* + - * Review checklist
        + Are change requests thoroughly evaluated?
        + Are they being thoroughly documented?

#### SQA Audits

Software Quality Assurance Audits inspects the software development process by comparing it to the established process. Monthly audits of the Software Quality Assurance activities will be held. These audits will help the team know which activities are working to discourage product defects and determine how well each activity is being implemented. These audits will keep the team on track and eliminate any useless sqa activities. The quality of the project handling can be analyzed only through the results of the review.

#### Problem Reporting mechanisms

Describes how and to whom problems are reported All defects or enhancements made to the software are referred the SQA team leader. This will be completed in every SQA review. If problems/defects occur between meetings they will be analyzed by the SQA team leader. These problems will be reported by the members of the SQA team.

#### Responsibilities

The SQA team leader is responsible for all SQA activities. All defects and enhancements reported by the SQA team will be priority and be analyzed. The SQA team leader will assign each member of the software engineering team a problem to fix or enhancement to add. Any solutions or enhancements made that do not meet the standards will be reassigned by the SQA team leader until they meet the standards.

#### Data collection and evaluation

This is a systematic way to collect data on the activities and results of a software in order to make decisions about the software. The SQA team leader will be responsible for collecting and evaluating the data. All product defects and enhancements are reported to the SQA team leader, who will then record the problems and evaluate their priority. The data is collected during all SQA team meetings.

#### Statistical SQA

This technique measures the quality of the product. Any defects that are found and reported will be analyzed by the SQA team leader. All information about defects are collected and organized. Each defect will be analyzed to trace the underlying cause. When the cause of the defect is found the team will discuss the problem to determine the best possible solution. The software will then be further analyzed to ensure the solution to the defect won't cause anymore defects. If the solution won't cause defects the changes to the software will be made to fix the defect. Underlying causes can be traced back to incomplete specifications, misunderstanding the clients requirements, not following program standards, error in the data design, not thoroughly testing, or inaccurate documentation.

To quantify our quality assurance we will use the following steps:

· Information about software defects is collected and categorized.

· An attempt is made to trace each defect to its underlying cause (e.g., nonconformance to specification, design error, violation of standards, poor communication with customer).

· Using the Pareto principle (80% of the defects can be traced to 20% of all possible causes), isolate the 20% (the “vital few”).

· Once the vital few causes have been identified, move to correct the problems that have caused the defects. WMITS Software Quality Assurance (SQA) Plan (06/06/00) Page 12 Although hundreds of different errors can be uncovered, even for a “small scale” project like this, all can be tracked to one (or more) of the following causes:

· Incomplete or erroneous specification (IES)

· Misinterpretation of customer communication (MCC)

· Intentional deviation from specification (IDS)

· Violation of programming standards (VPS)

· Error in data representation (EDR)

· Inconsistent module interface (IMI)

· Error in design logic (EDL)

· Incomplete or erroneous testing (IET)

· Inaccurate or incomplete documentation (IID)

· Error in programming language translation of design (PLT)

· Ambiguous or inconsistent human-computer interface (HCI)

· Miscellaneous (MIS)

Error Index (EI)

As for the overall indication of improvement is software quality, the Error Index technique would be used.

We will compute the error index as follows

Let:

* Ei = the total number of errors uncovered during the ith step in the software engineering process
* Si = the number of serious errors
* Mi = the number of moderate errors
* Ti = the number of minor errors
* PS = size of the product (LOC, design statements, pages of documentation)

At each step in the software engineering process, a phase index, PIi, is computed.

PIi = Ws (Si/Ei) + Wm(Mi/Ei) + Wt(Ti/Ei)

EI is the cumulative effect on peach PIi = sum(I x PIi)/ PS

#### Goal and objectives of Software Process Improvement (SPI)

The goal and objectives of SPI are to ensure the decrease in software defects. They will also determine the underlying causes of all defects that occur. When the underlying causes are found, the SQA team will determine the best possible solution to eliminate the problem.

#### SPI tasks and responsibilities

The SQA team leader will be responsible for keeping track of any errors that have occurred frequently. If a defect occurs frequently from the same underlying causes, then that area will be considered more problematic. The SQA team will investigate the area from the defect logs and try to find if the problematic area is being caused by one engineer or multiple. If it is found that one engineer is the cause of those underlying issues, they will be notified. The engineer will be better suited to find out why those issues are happening so frequently. The SQA leader will also try to give the engineer suggestions on how to fix the problem. If the problems have been occurring from multiple engineers, then their development practices will be analyzed to determine the cause of the problems. The SQA team will then hold a meeting with the engineers where possible solutions will be discussed.

#### Software Configuration Management Overview

The software configuration management is the ability to control and manage change in a software project. When any changes are made to the design of the software, they will have to be submitted to the software configuration manager. Change is continuously ongoing in any software project. Software configuration management defines a process for change control. A change request will be submitted and reviewed to determine the possible impacts it will have on the software. If the change will have an impact it will be re evaluated to see if the change is worth the problems it may cause. All change requests are kept on record whether it is implemented or not.

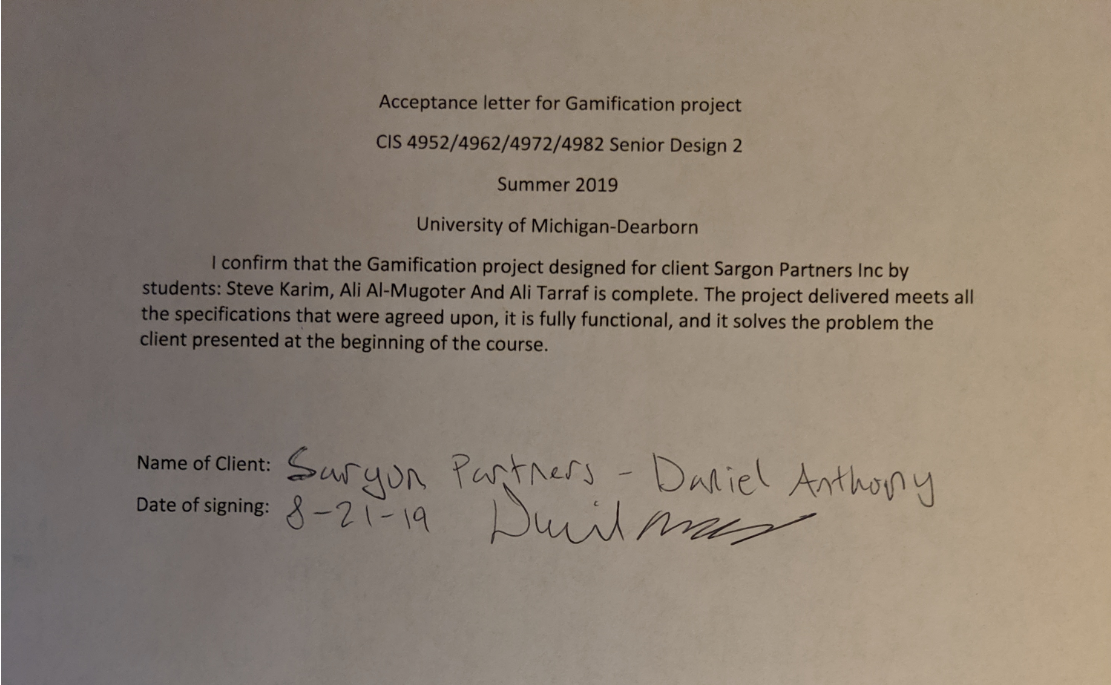
#### SQA Tools, Techniques, Methods

All SQA activities will follow the same guidelines and methods. All members of the SQA team will be present at all meetings. All members are expected to participate. Close contact will be kept with the client to ensure all requirements are met.

### Future Maintenance Suggestions

Any changes made to the company table or the shard database schema must be reflected ShardCreator. If the schema is not maintained it is probable that version issues will occur. The shard database schema must be constantly updated and maintained any time that a change is made to the database.

### Delivery/Acceptance Statement



### References & Bibliography

##### Client Api:

<http://api.invisi-tag.com/docs/#/>

##### Client Software demonstration:

<https://www.youtube.com/watch?v=e9aQ1T808Aw>

##### Gamification Definition:

<https://en.wikipedia.org/wiki/Gamification>

##### Gamification Software:

<https://appsource.microsoft.com/en-us/product/dynamics-365/mscrm.f6d23ec7-255c-4bd8-8c99-dc041d5cb8b3?tab=Overview>

##### Kpi: <https://www.google.com/search?q=key+performance+indicators&oq=key+performance+indicators&aqs=chrome..69i57j0l5.456j0j7&sourceid=chrome&ie=UTF-8>

##### Data Dictionary:

<https://docs.microsoft.com/en-us/sql/t-sql/data-types/decimal-and-numeric-transact-sql?view=sql-server-2017>

<https://docs.microsoft.com/en-us/sql/t-sql/functions/date-and-time-data-types-and-functions-transact-sql?view=sql-server-2017>

<https://docs.microsoft.com/en-us/sql/t-sql/functions/date-and-time-data-types-and-functions-transact-sql?view=sql>)-server-2017

##### Database:

<https://docs.microsoft.com/en-us/azure/sql-database/sql-database-single-database-get-started>

serialize image to binary & save to database:

<http://www.nullskull.com/articles/20020929.asp>

<https://www.youtube.com/watch?v=09ftaulKUVM>

##### Backend:

Web Application:

<https://docs.microsoft.com/en-us/visualstudio/ide/quickstart-aspnet-core?view=vs-2017>

Program scheduler (time trigger):

<https://docs.microsoft.com/en-us/azure/azure-functions/functions-bindings-timer>

<https://www.quartz-scheduler.net/>

<https://www.thewindowsclub.com/how-to-schedule-batch-file-run-automatically-windows-7>

Login Authentication:

<http://www.primaryobjects.com/2015/05/08/token-based-authentication-for-web-service-apis-in-c-mvc-net/>

REST Guidelines: <https://tools.ietf.org/html/rfc7231#section-4.3>

Frontend: <http://sudiptachaudhari.com/angular-7-asp-net-core/>

Picture upload: <https://www.academind.com/learn/angular/snippets/angular-image-upload-made-easy/>

Graph:

<https://docs.microsoft.com/en-us/power-bi/visuals/power-bi-visualization-types-for-reports-and-q-and-a>

##### Design patterns used for web application:

MVC:

<https://docs.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-2.2>:

Mediator:

<https://www.dofactory.com/net/mediator-design-pattern>

Facade:

<https://www.dofactory.com/net/facade-design-pattern>

##### CQRS:

<https://docs.microsoft.com/en-us/azure/architecture/patterns/cqrs>

##### SQA:

<http://www.helpingtesters.com/software-quality-assurance-activities/>

<http://people.ucalgary.ca/~design/engg251/First%20Year%20Files/design_reviews.pdf>

<https://www.uxmatters.com/mt/archives/2009/02/reviewing-user-interfaces.php>

<https://iaonline.theiia.org/change-control-audits-a-must-for-critical-system-functionality>

<http://www.mhhe.com/engcs/compsci/pressman/graphics/Pressman5sepa/common/cs1/sqa.pdf>

<http://www.mhhe.com/engcs/compsci/pressman/graphics/Pressman5sepa/common/cs2/sqa.pdf> (Links to an external site.)

<http://ibiblio.org/gferg/ldp/SCM-OpenSource/scm-overview.html>

##### SQL optimization: <https://docs.microsoft.com/en-us/previous-versions/sql/sql-server-2008-r2/ms191227(v=sql.105)>

### Appendix A - User Manual

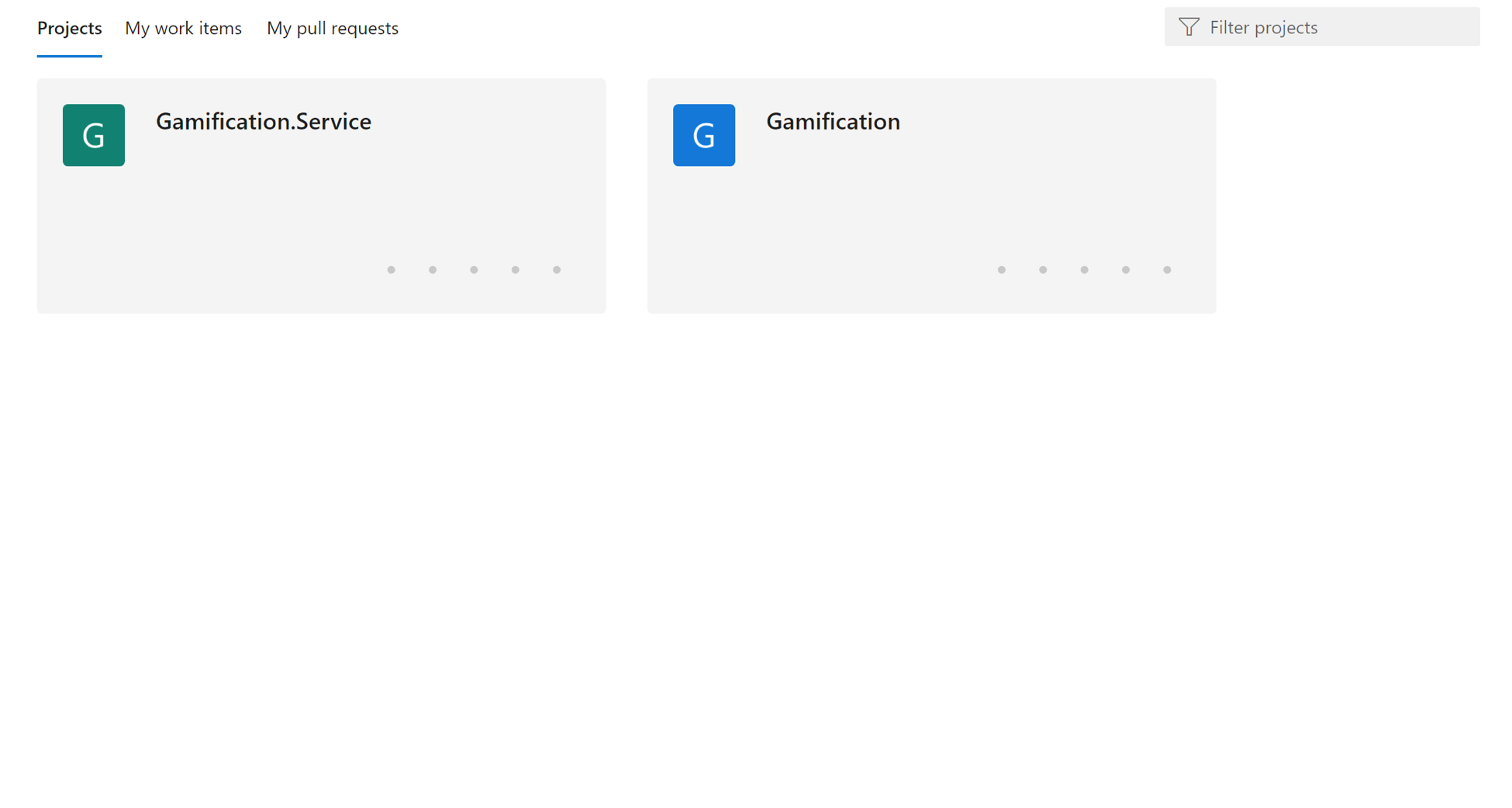
#### Setting up the environment:

##### How to clone the project

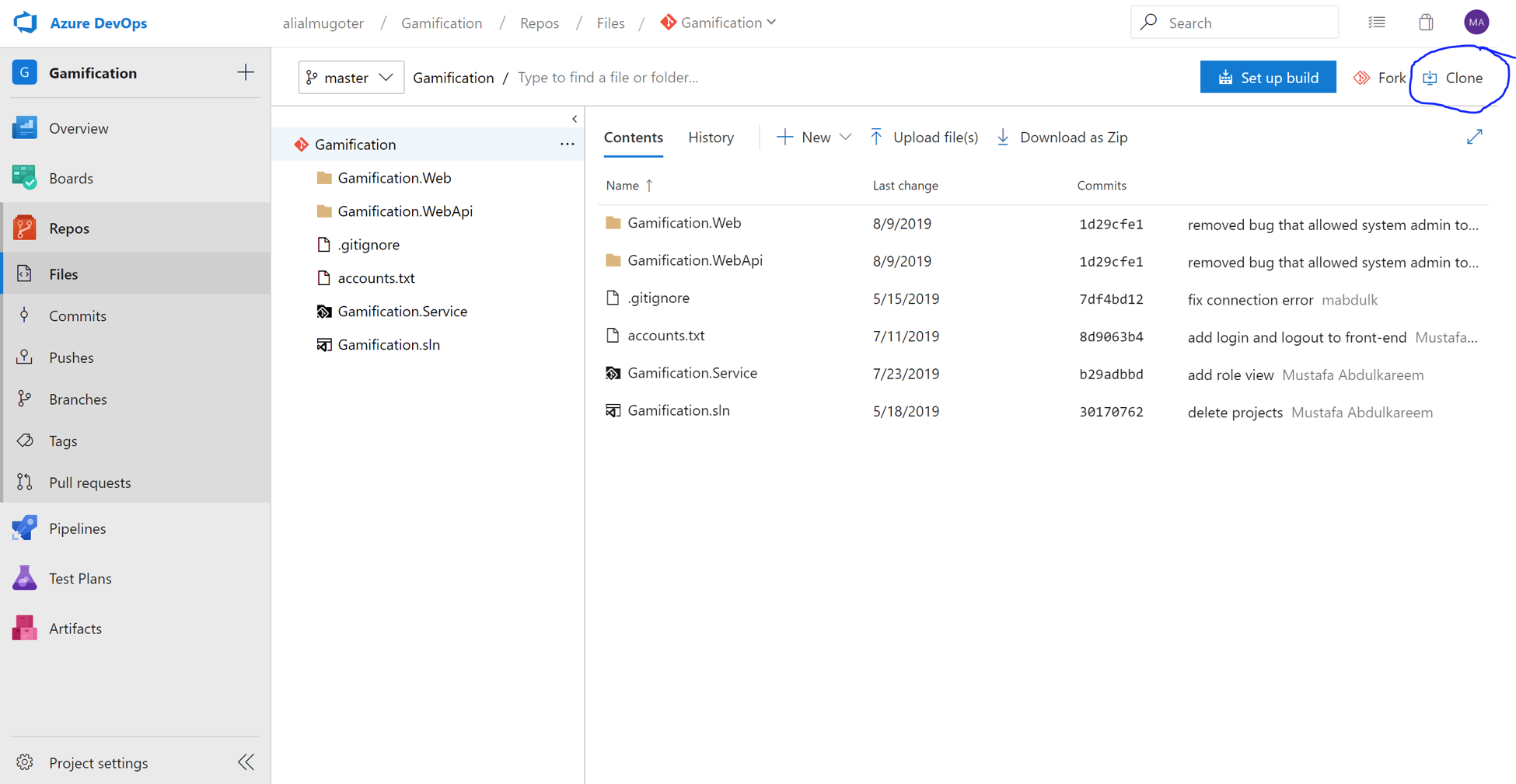
In the Azure DevOps website. You must be granted access to be able to view the projects.

After signing in you will have access to two projects

Gamification and Gamification.Service



You then clone the two projects to the same local repository.



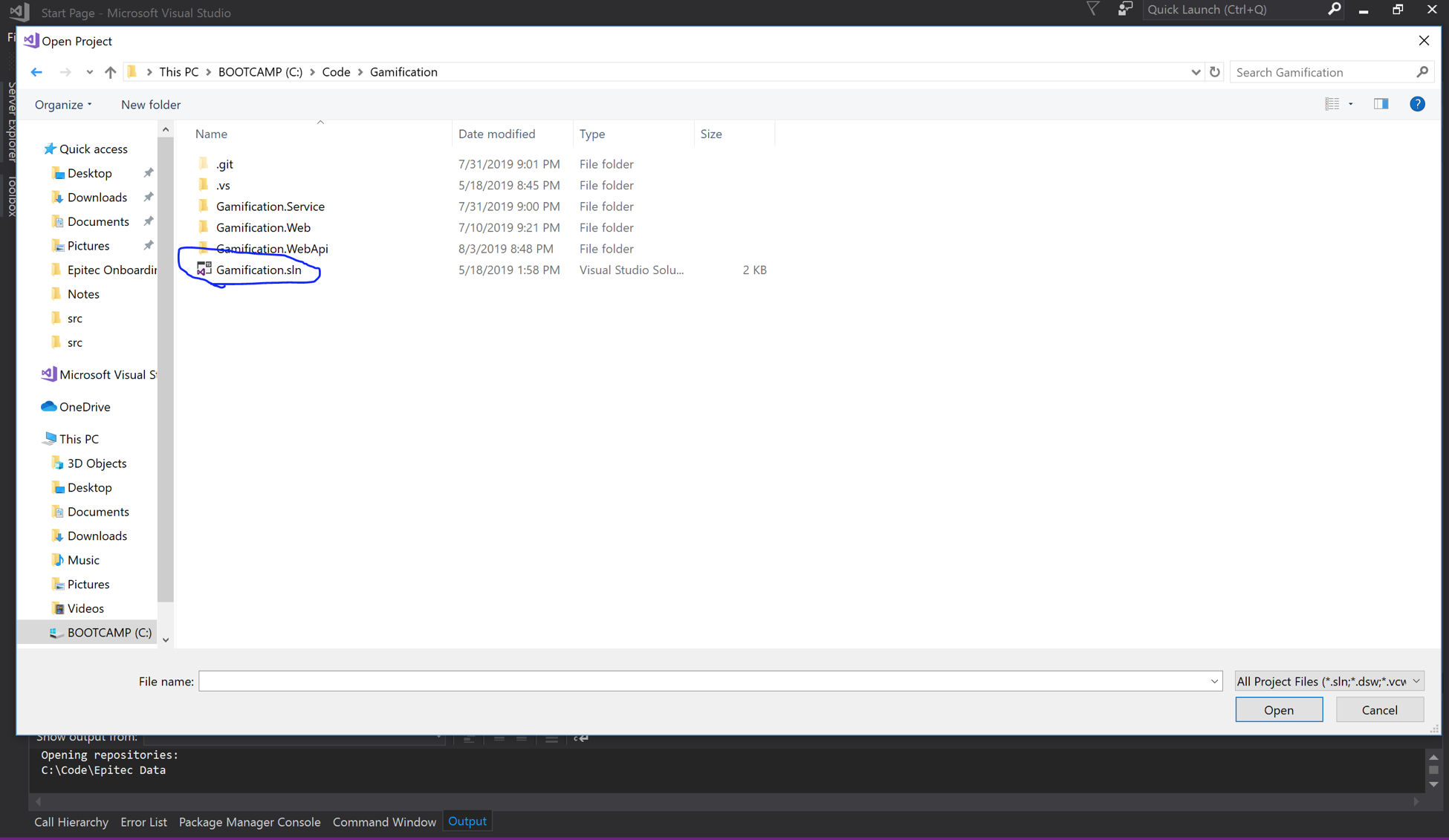
##### What tools you need

· Visual Studio

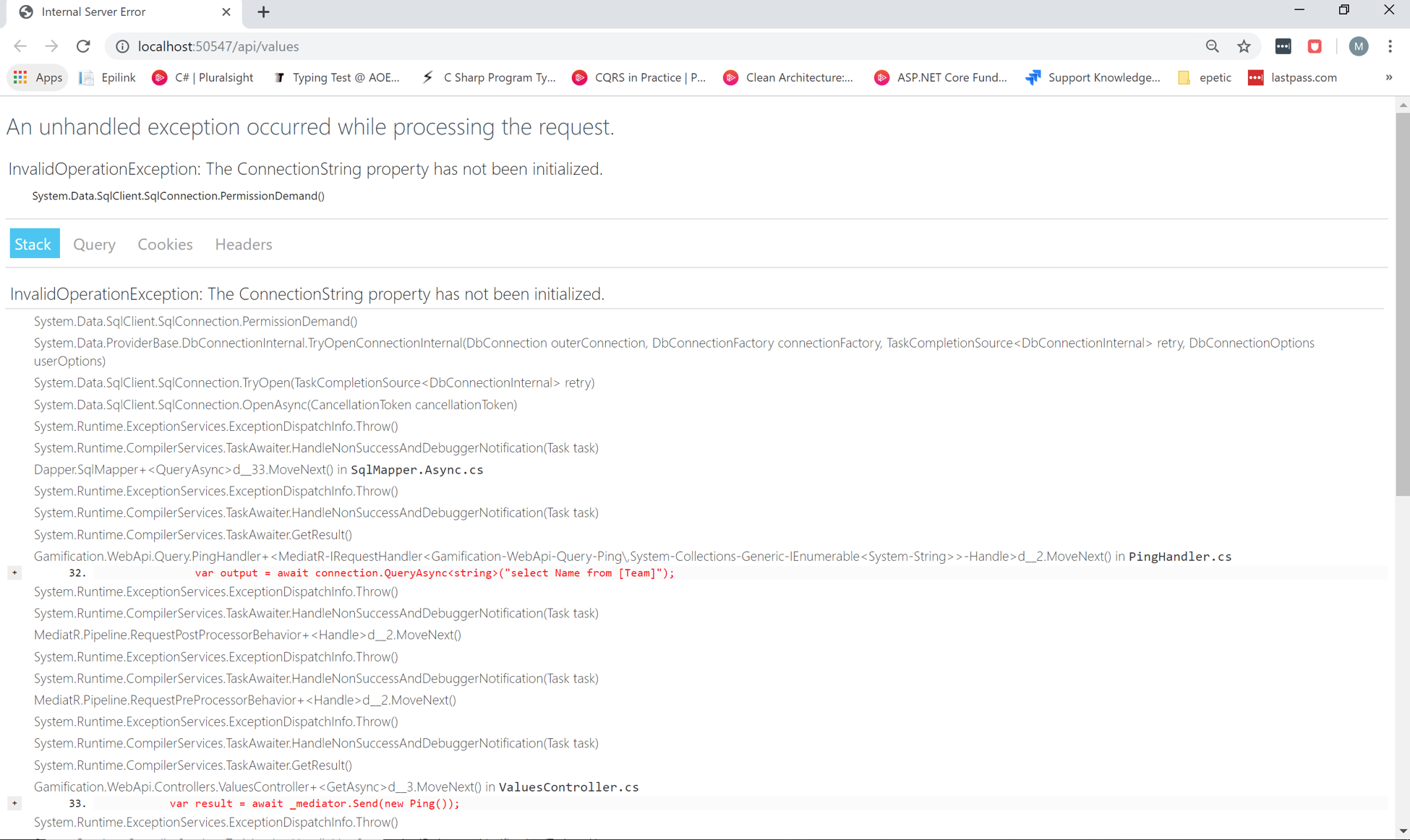
· Visual Studio Code

· Microsoft SQL Server Management Studio

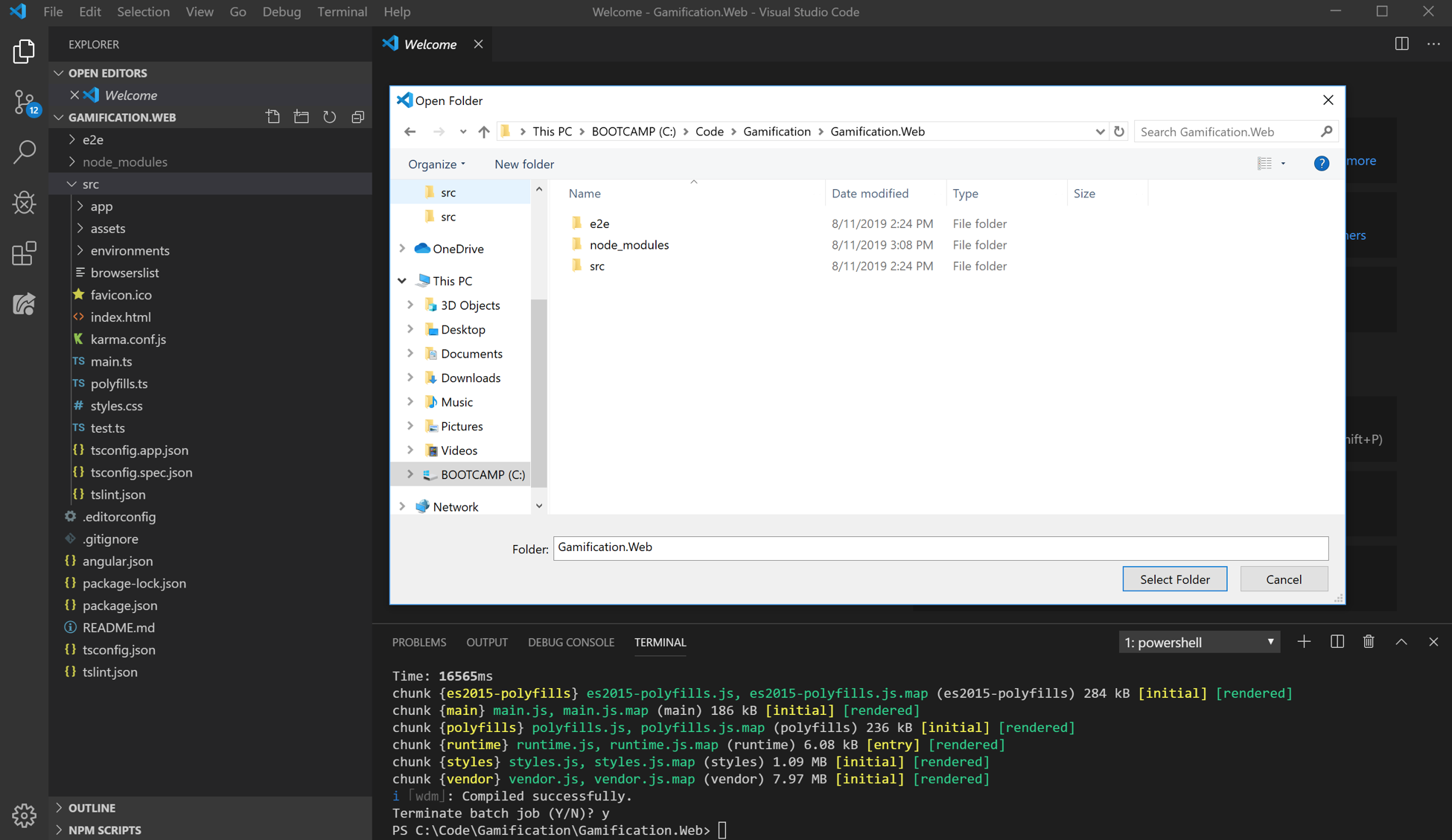
Gamification: you open visual studio and get to the directory Gamification and select Gamification.sln



this will open the project solution for you. It contains the backend of the project (web api .net core). When you run the project you should see the browser opening with the local port number.



FrontEnd: to run the front-end you first open Visual Studio Code, you select the following folder to open (Gamification.Web)



After selecting the folder you open the terminal command and run the following command

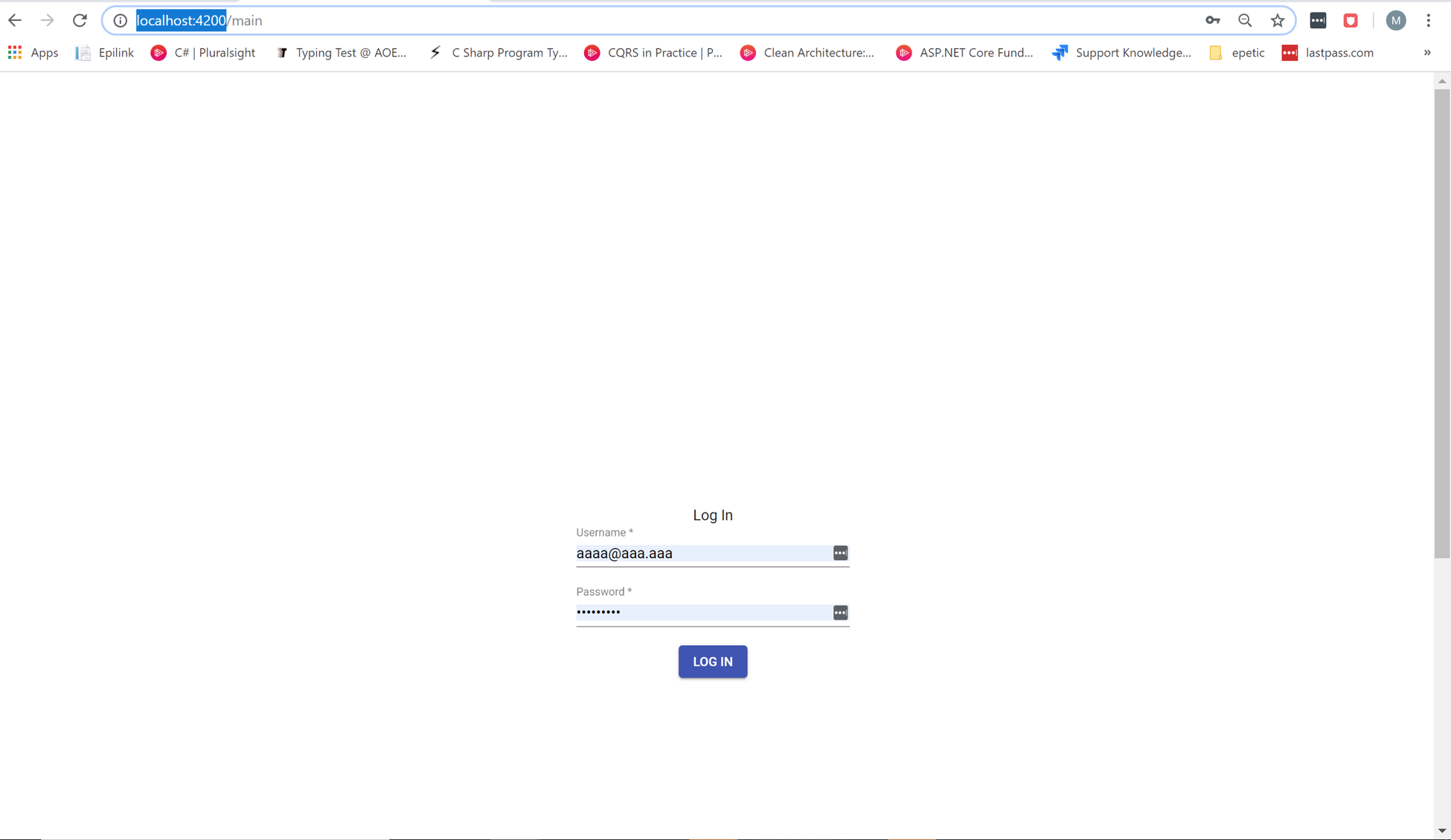
Npm install

//this will install the Nuget package

Npm start

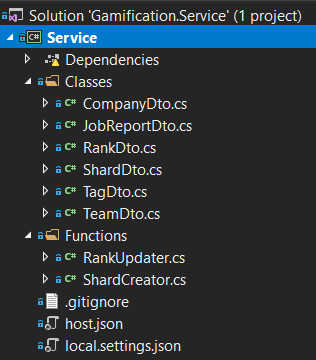
// this will run the front-end

Then you type in [http://localhost:4200](http://localhost:4200/) in the browser to view the website

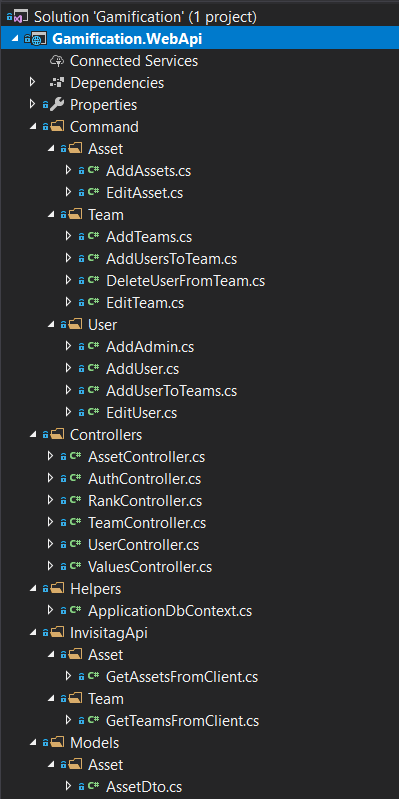


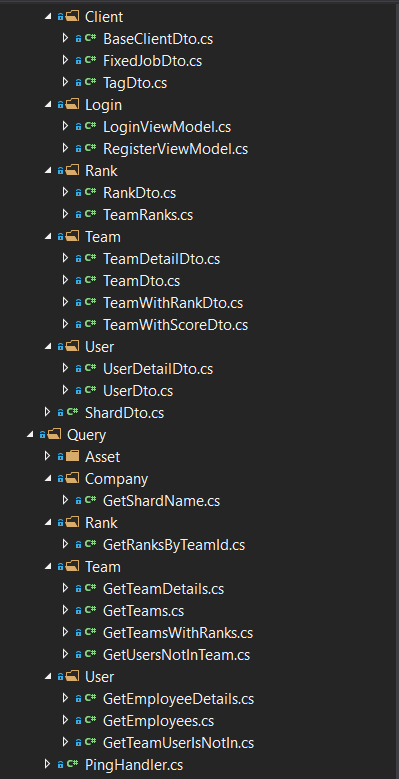
### Appendix B - Program Listing

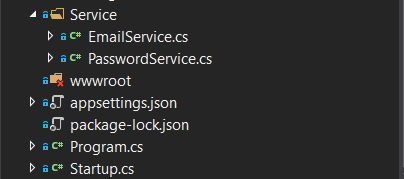
Gamification.Service



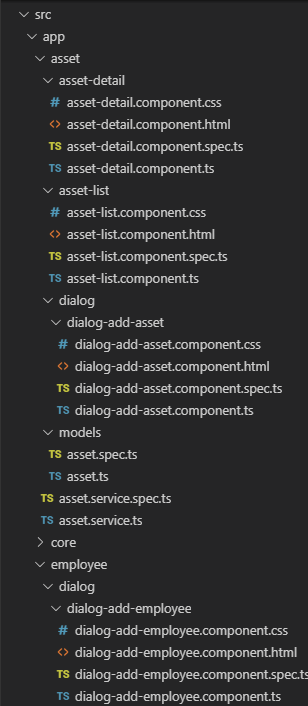
Gamification.WebApi

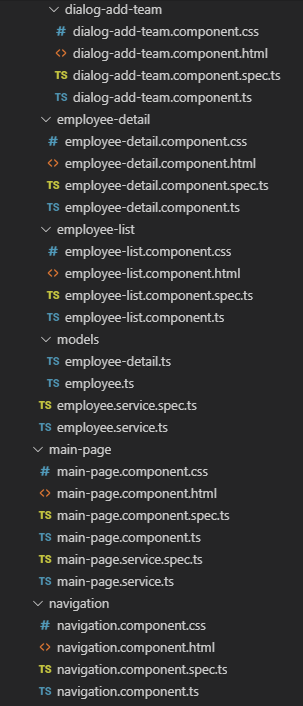


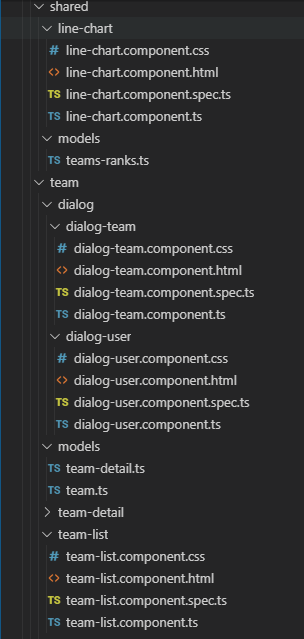


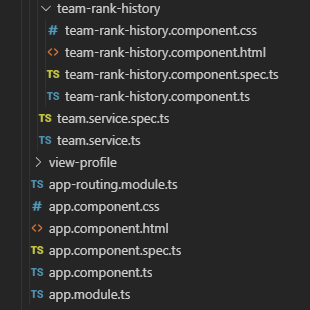


Gamification.Web



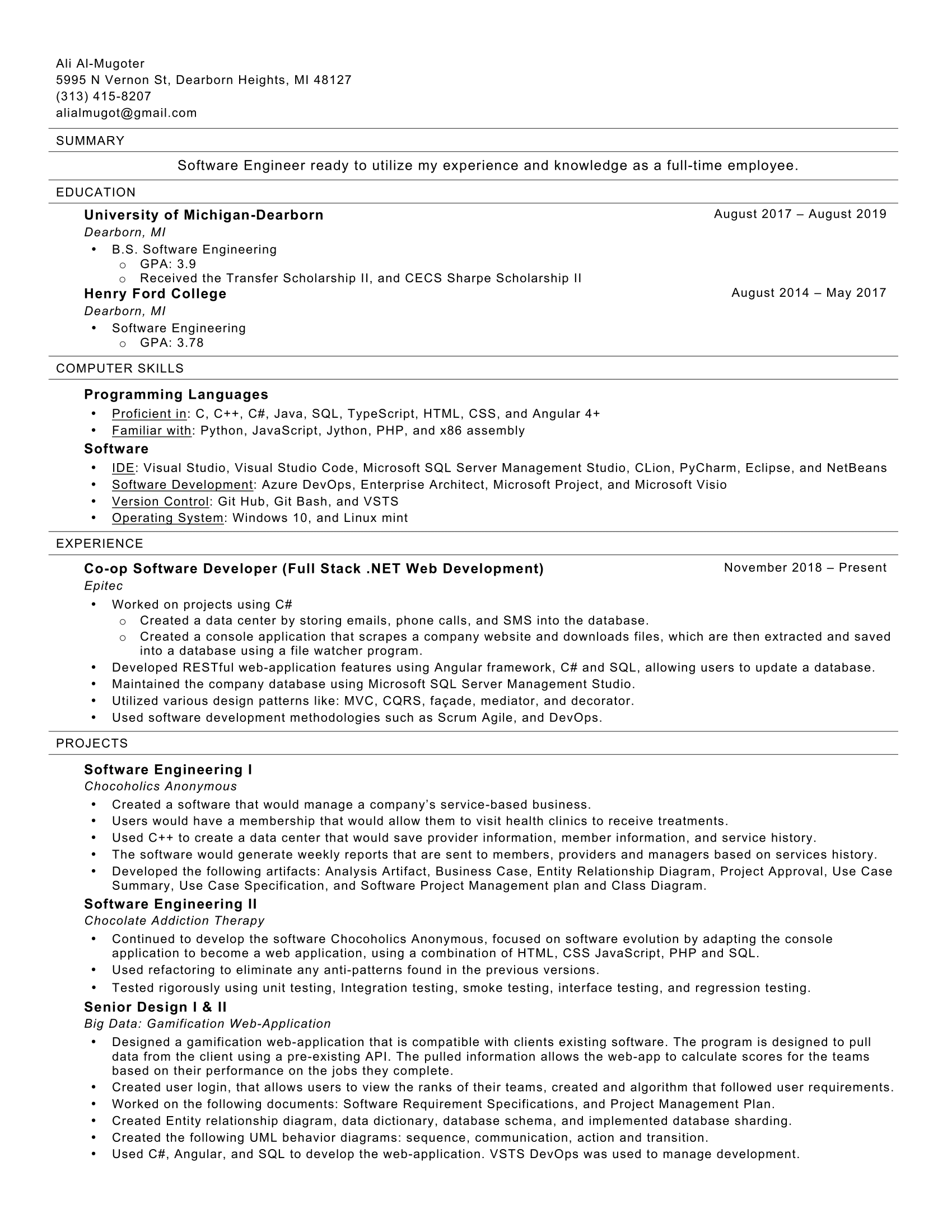






### Appendix C - Team Member Resumes

#### Ali Al-mugoter

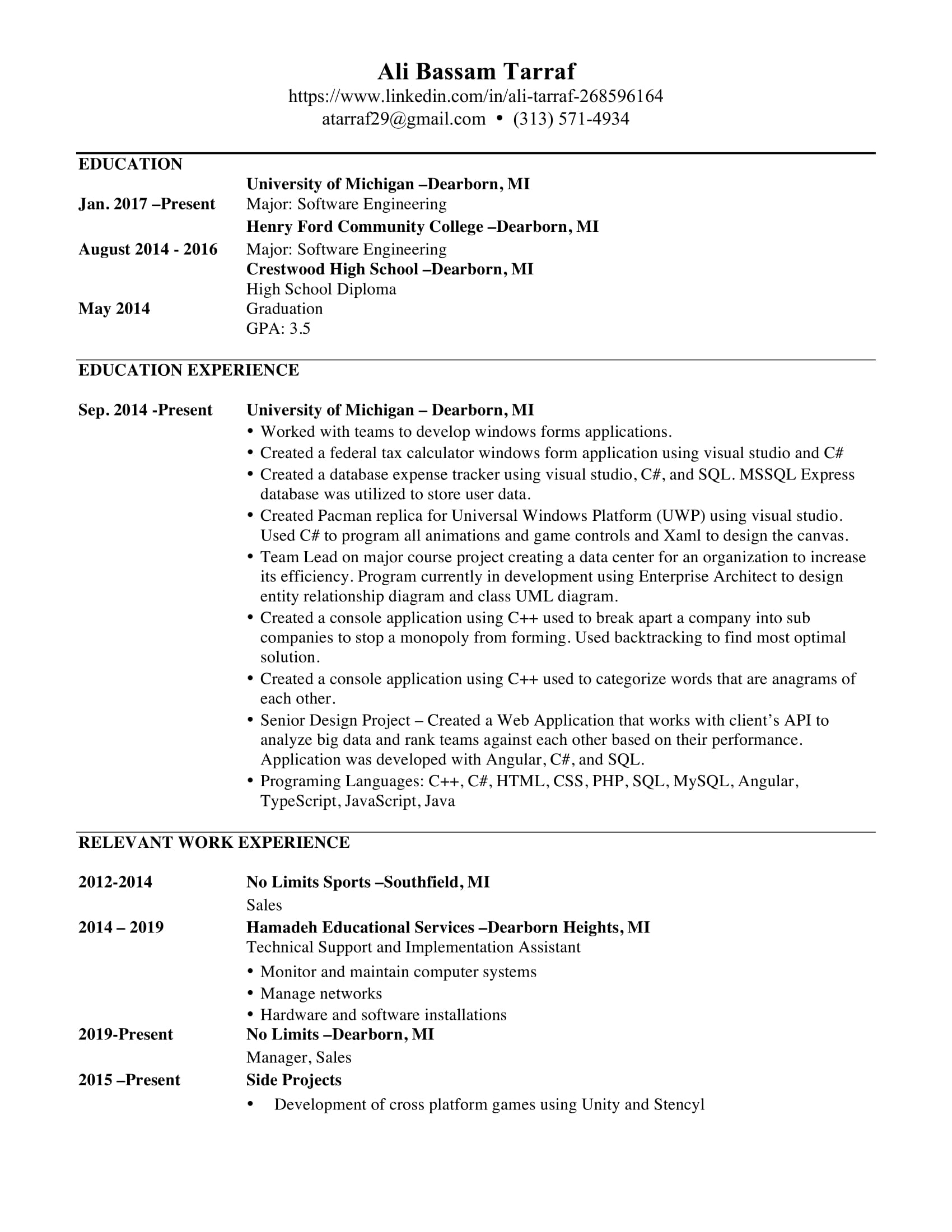


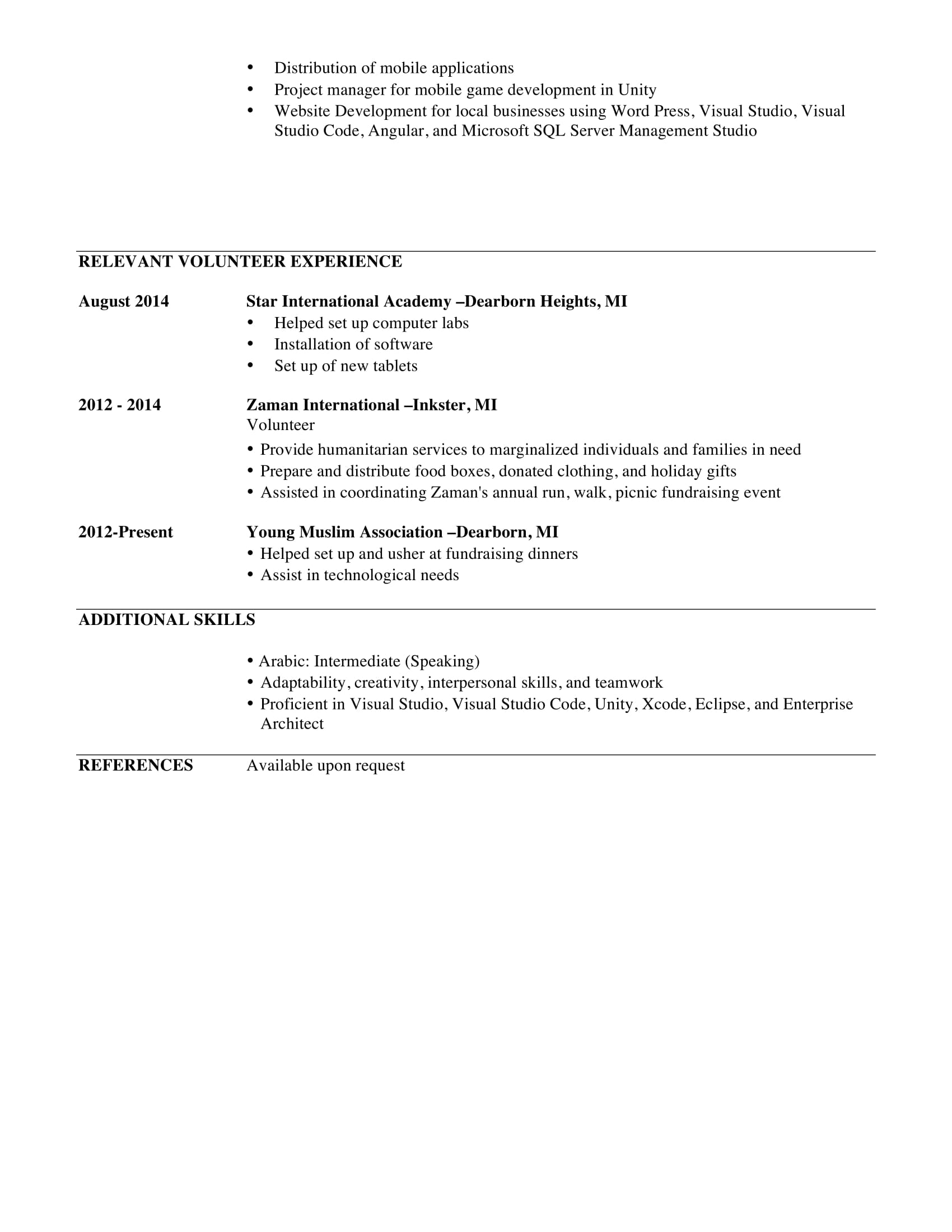
#### Steve Karim





#### Ali Tarraf





### Appendix D - Project Plan & Log Book

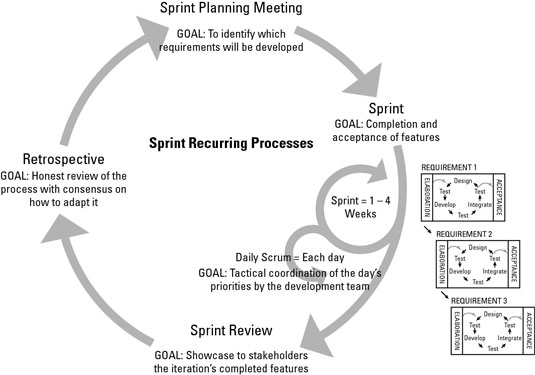
#### Project task set

Process Model: Scrum Agile

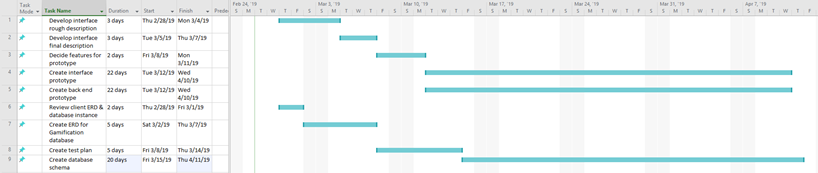
The scrum approach includes assembling the project’s requirements and using them to define the project. Then plan the necessary sprints, and divide each sprint into its own list of requirements. Daily/weekly scrum meetings help keep the project on target as do regular inspections and reviews. At the end of every sprint, we hold a sprint retrospective to look for ways to improve the next sprint.

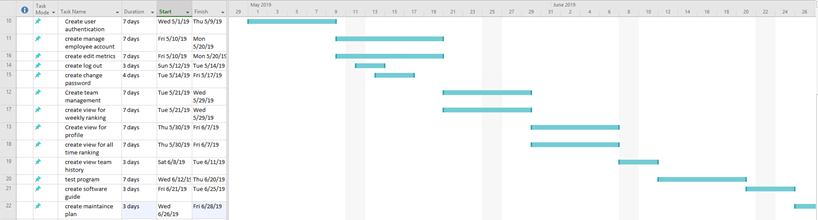
Within each sprint, we build and tests a functional part of the product until the product owner accepts it and the functionality becomes a potentially shippable product. When one sprint finishes, another sprint starts. Product features are delivered in increments at the end of each sprint. A product release occurs at the end of a sprint or after several sprints.

* We use the tenets of inspection and adaptation on a daily basis as part of a scrum project:
* During a sprint, we conduct constant inspections to assess progress toward the sprint goal, and consequently, toward the release goal.
* We hold a daily scrum meeting to organize the day by reviewing what the team completed yesterday and what it will work on today. Essentially, the scrum team inspects its progress toward the sprint goal.
* At the end of the sprint, you use a retrospective meeting to assess performance and plan necessary adaptations.



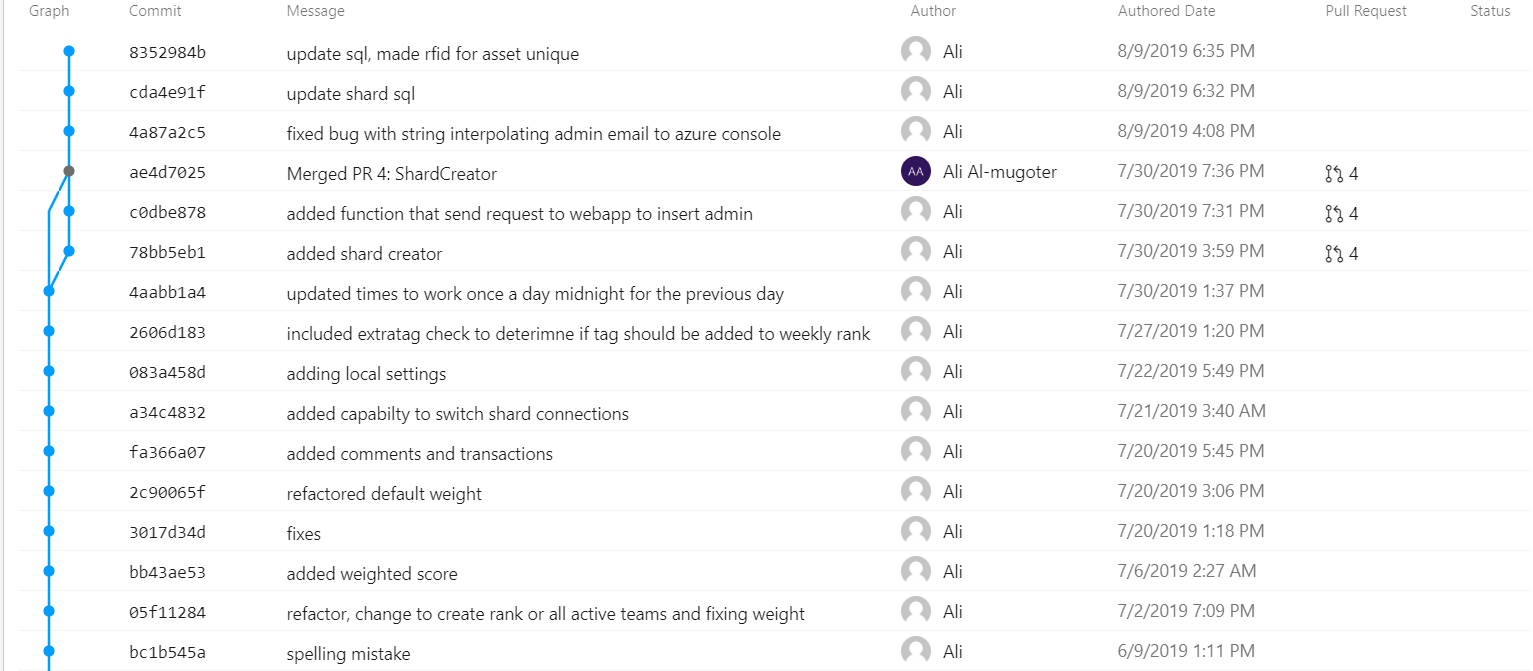
|  |  |
| --- | --- |
| **Framework Activities** | **Task Set** |
| Modeling: Analysis of requirements | Software Interface Description  Validation Criteria  Milestone |
| Modeling: Design | Entity relationship Diagram (Database)  Prototype  Milestone |
| Construction: Code Generation | Database  Frontend & Backend  Milestone |
| Construction: Testing | Testing Result  Milestone |
| Deployment | Software Guide  Maintenance Plan  Milestone |

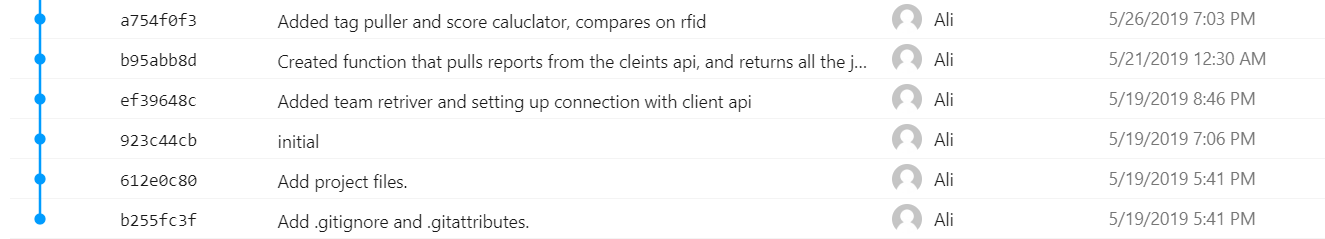
Original Timeline



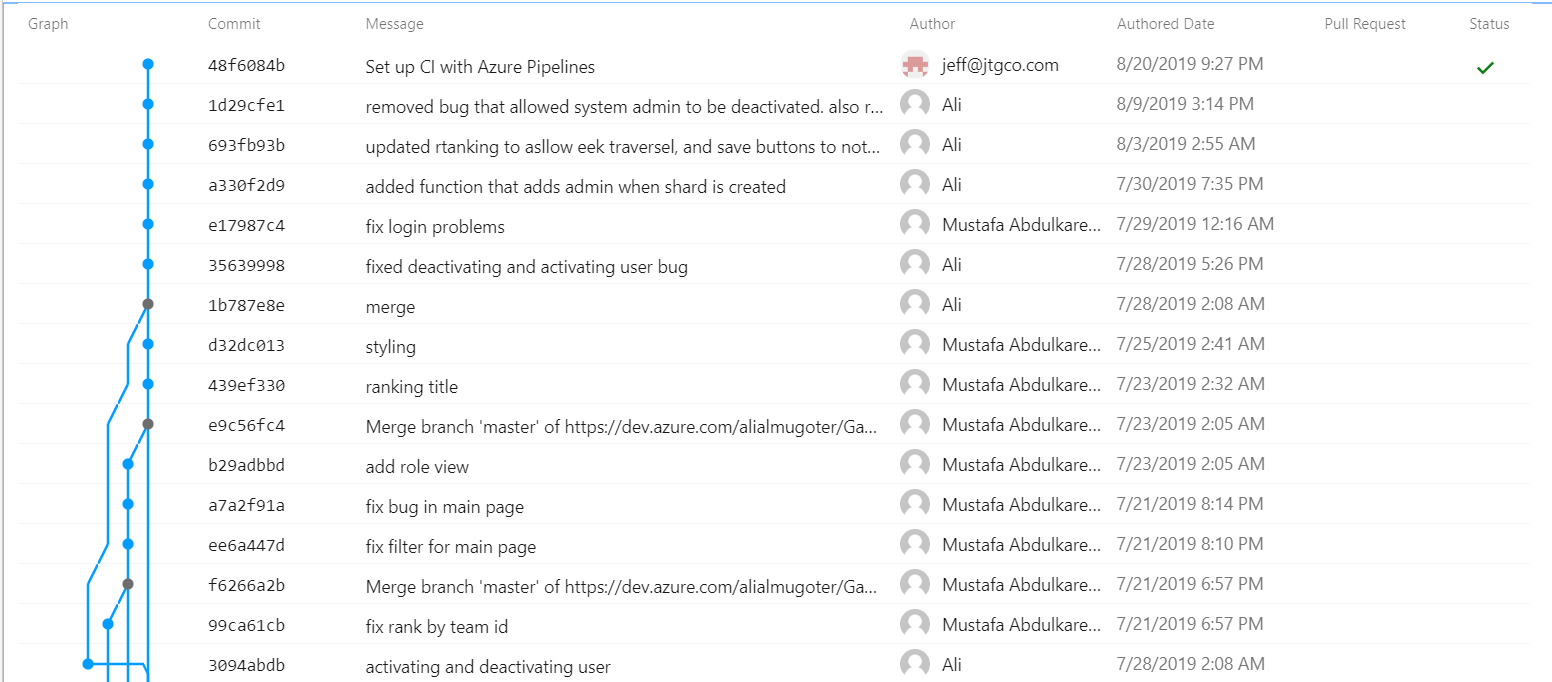
Log book

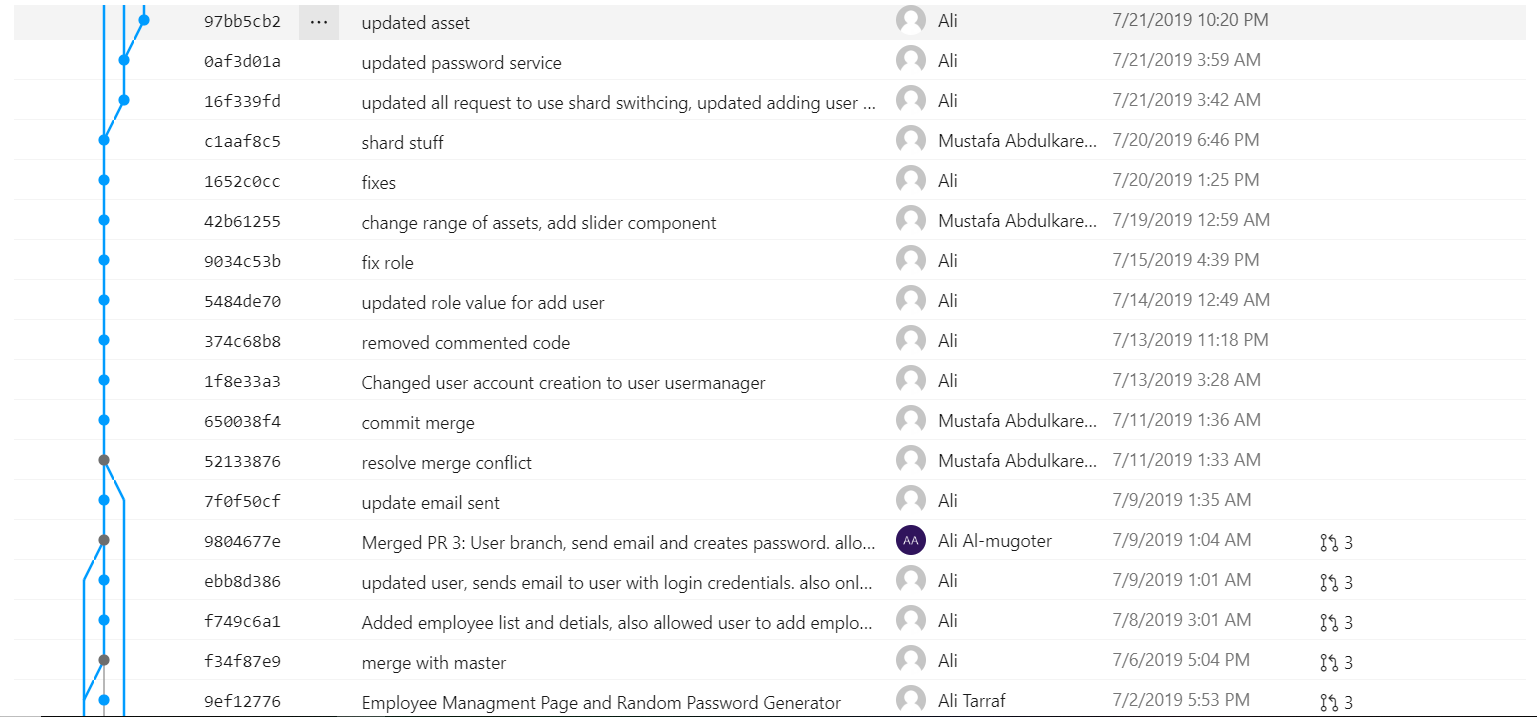
Gamification.Service

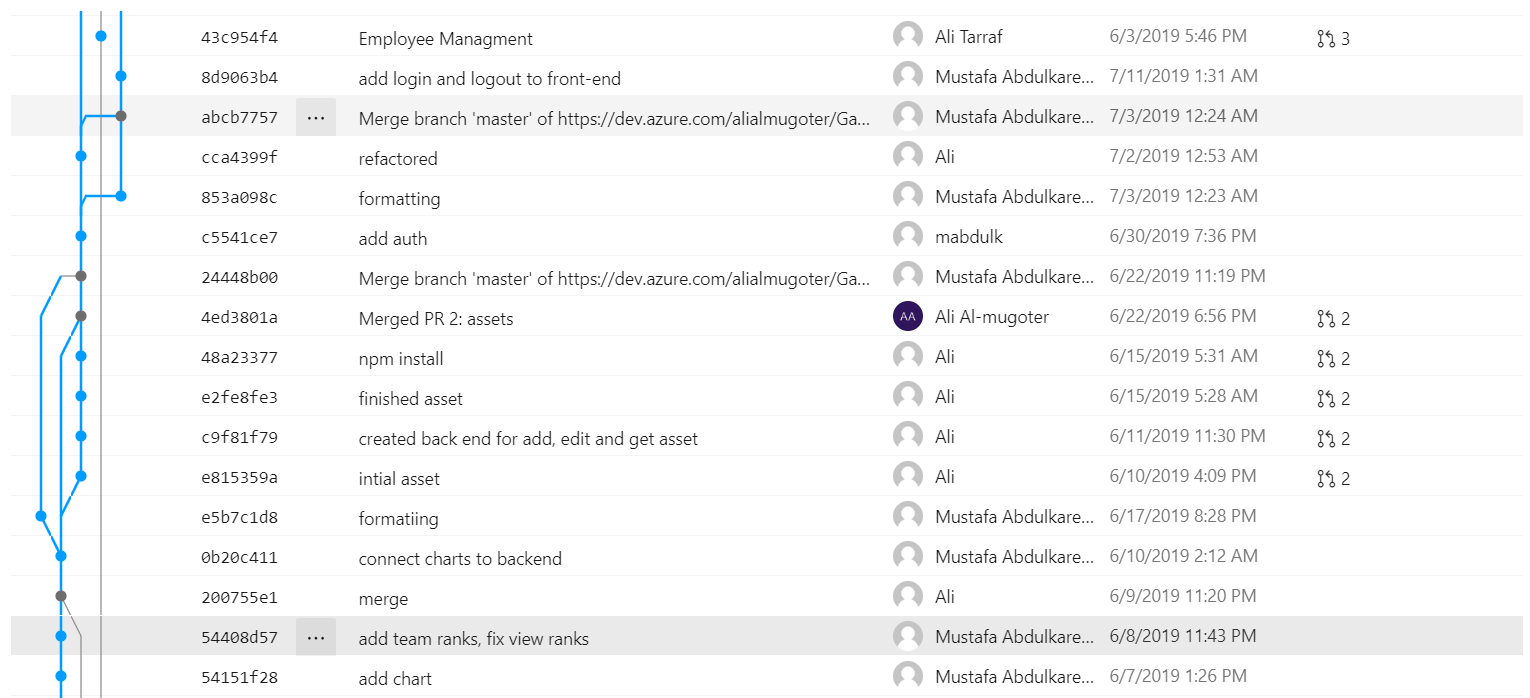


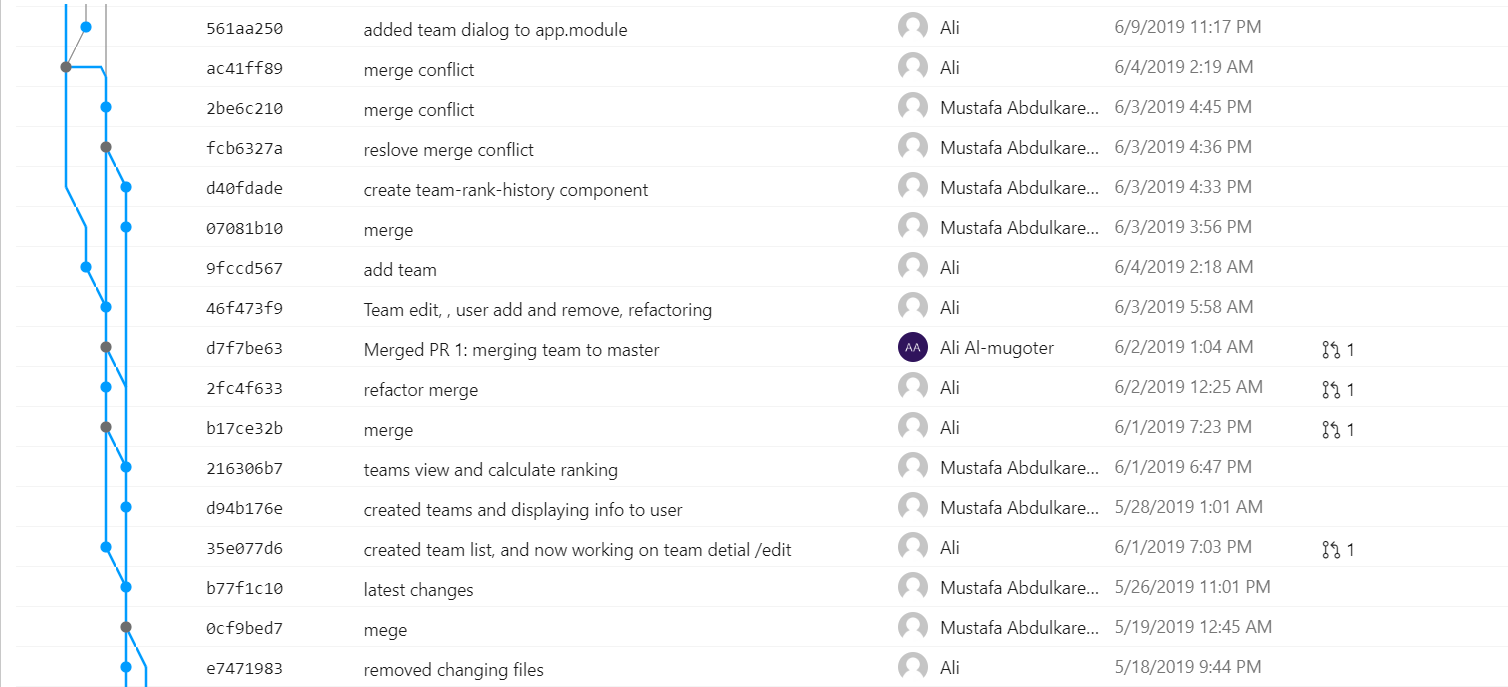


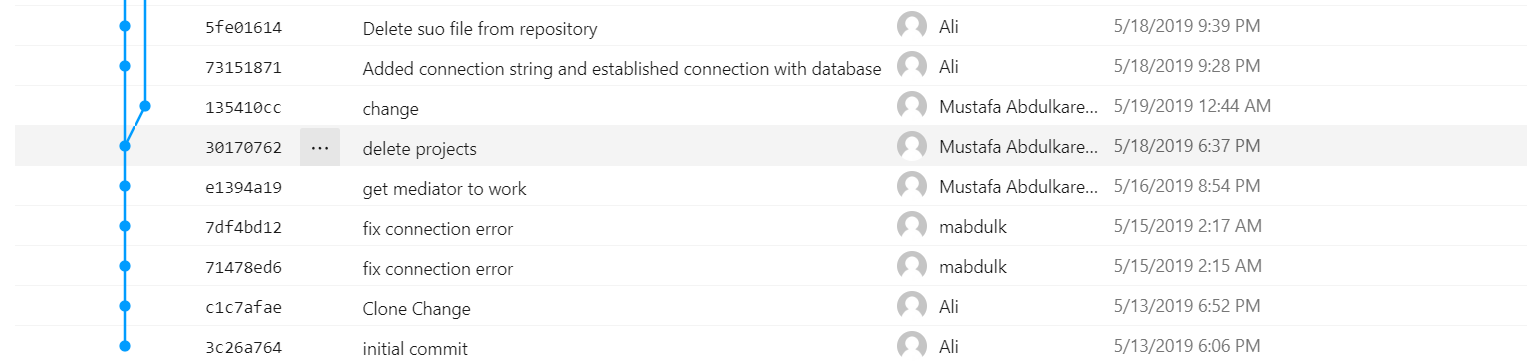
Gamification











### Appendix E - Project Demo Notes

* + - * Demonstrating Empoyee login view:

This shows the website from the view of the employee, the employee has limited access. Only able to view the main page and profile page.

* + - * Demonstrating Admin view. The admin has the ability to view all the features of the website including adding emplyees to teams, adding assets and changing the weight of the assets
      * Signing in for a user from a different company demonstrating the sharding feature implemented in the project.
        + This will show the different teams related to the different company
      * Go over the architecture of the website demonstrating all the feature varying from the ranking to the graph of the team ...etc

### Appendix F - Final Presentation Slides

