ENGTECH 4FD3 – Senior Engineering Project:

NBA Predictions Web Application

Design Documentation

**Instructor**: Dr. Seshasai Srinivasan

**Group Members**:

Branislav Kukulj

Wenbo Liu

Emily Ramanna

Mingming Zhang

**Date**: Feb 10 2022

**Table of Contents**

**1. Introduction 1**

**1.1 Project Purpose 1**

**1.2 Project Audience 1**

**1.3 Technologies Used in Project 1**

**2. Requirements 2**

**3. Scenario Viewpoint 4**

**4. Physical Viewpoint 5**

**5. Development Viewpoint 6**

**6. Logical Viewpoint 10**

**7. Process Viewpoint 12**

**8. Graph Database Details 14**

**9. Configuration Data and Starting Point 15**

**10. Additional References 15**

# 1. Introduction

## 1.1 Project Purpose

This project will produce a user-friendly web application that will allow people to predict wins and losses for NBA (National Basketball Association) games.

## 1.2 Project Audience

The target audience of this software will be adults (defined by the age of majority within the user’s residential jurisdiction) who are already comfortable using social media platforms and with others directly contacting them This software is primarily meant for those who are casually looking to meet new people/friends.

## 1.3 Technologies Used in Project

|  |  |  |
| --- | --- | --- |
| Technology | Role | Source |
| .NET Core Framework | Development Platform (is cross-platform to aid in portability) | <https://dotnet.microsoft.com/> |
| C# | Language that will be used to implement the console application | <https://docs.microsoft.com/en-us/dotnet/csharp/> |
| Mysql cloud | Sql database technology used for data storage | <https://clever-cloud.com/> |
| Python | Machine learning Models training | <https://python.com/> |
| Asp.Net Core 3.1 | Framework for building modern web application and services | [Download .NET Core 3.1 (Linux, macOS, and Windows) (microsoft.com)](https://dotnet.microsoft.com/en-us/download/dotnet/3.1) |
| JavaScript | Web programming language used for front stack development to interact with html, CSS etc. | [JavaScript.com](https://www.javascript.com/) |
| Bootstraps | One popular library for frontend toolkit to render pages responsive | [(getbootstrap.com)](https://getbootstrap.com/) |
| Google oAuth | OAuth is commonly used for web server, client-side, installed, and limited-input device applications | [Using OAuth 2.0 to Access Google APIs  |  Google Identity  |  Google Developers](https://developers.google.com/identity/protocols/oauth2) |

# 2. Requirements

|  |  |
| --- | --- |
| Functional Requirements | |
| R01 | A user shall be able to create, maintain, and login to an account that has a profile with the following pieces of information  1) unique username,  2) password, and  3) single piece of contact information. |
| R02 | Users can create a pool and invite others to join their pool. |
| R03 | Users can predict the day’s games; all user’s prediction results will be stored in the database. |
| R04 | Users can review past activity once logged in (i.e., see past predictions and their point total). |
| R05 | Users can review the leaderboard in the pool. |
| R06 | The application shall make use of a configuration file to read in database and API keys, credentials, and other information required for access. |
| R07 | The application shall record log messages to a database. |
| R08 | Users can choose to turn on auto select to make predictions on their behalf. |
| Nonfunctional Requirements | |
| R08 | Traceability: At least 90% of the application errors including database errors will create a log entry at runtime.  Validated via testing and verifying the log files. |
| R09 | Performance Requirement:  Database request (maximum 4 matchups or 8 teams) should be generated within 15 seconds (worst scenario performance metric).  Measured by testing and analyzing the running system: log the amount of time it takes between the request and receiving the data each time this list is requested, then verify that (number of requests that took longer than 6 seconds) / (total number of requests) \* 100 is less than or equal to 5%. |

# Unit testing of the webserver and the Python classes can also be part of the overall validation strategy.  UAT (user acceptance testing) scripts can be used to ensure that the user experience is functioning as  expected. Cross-validation can be performed when training the machine learning models to try to ensure  that the predictions are better than a coin flip.

# 3. Scenario Viewpoint

**Concern**: Foster greater understanding of the system’s core functionalities.

**Stakeholders**: This viewpoint can be useful to all stakeholders, but end-users will benefit from it especially.

**Modelling Technique**: UML Use Case Diagram

Primarily, a user can manage their account and game pool. In addition, the core functionalities involve inviting others to participate and review the results scores to select the best predictor.

# Diagram Description automatically generated

*Figure 2: Use Case Diagram (Scenario Viewpoint)*

# 4. Physical Viewpoint

**Concern**: Displaying which software runs on which hardware and showing which communication protocols and modules are used.

**Stakeholders**: Software architect/developers, deployment team/application support (they may need to know for troubleshooting errors regarding communication protocols or missing or corrupted files in the communication modules)

**Modelling Technique**: UML Deployment Diagram

The web application runs on the client device that has .NET Core installed. Since .NET Core is cross-platform, the operating system of the client device has not been specified.

Machine learning is done through python, which makes use of NBA\_API to make HTTP requests to nba.stats.

The core of whole application Mysql database is hosted in clever-cloud, which also provides backups and migrate and PHPMyAdmin access tools.

Diagram

Description automatically generated

*Figure 3: Deployment Diagram (Physical Viewpoint)*

# 5. Development Viewpoint

**Concern**: Software module organization

**Stakeholders**: Software developers, project manager (for allocating development resources), testing team (for decisions regarding automated testing)

**Modelling Technique**: UML Class Diagram

The Model-View-Controller (MVC) pattern handles the user interaction in this application. The controller and the view depend on the models, but the models are independent of the controller and the view.

Conventionally, the model in MVC handles data persistence too. This application makes use of the repository pattern (Microsoft Docs, 2018) to separate the model from a specific database implementation (e.g. MySQL, RedisGraph, etc

All classes related to the analysis service and the repository pattern will also be considered part of the Model layer because they will contain domain logic, which fits with the definition of the Model layer.

Design Pattern Summary

|  |  |  |
| --- | --- | --- |
| **Pattern** | **Role** | **Classes** |
| Singleton | Contains read-only configuration data. |  |
| Model-View-Controller | Handles user interaction. | Controller,  View,  Model Layer: |

# Diagram Description automatically generated

*Figure 4: Class Diagram (Development Viewpoint)*

# 6. Logical Viewpoint

**Concern**: Supporting the functional requirements that are available to end-users

**Stakeholders**: End-users, application testing team (ensuring that all end-user functionality is tested), software architect

**Modelling Technique**: UML Activity Diagram

The console application will present a sequence of prompts and menus that will enable end-users to perform the use cases presented in Figure 2.

Users will select options from the menus by entering a number that is associated with the option. All input and output between the user and the application will be text entered from the keyboard and displayed on the screen. Users will return to the menu they selected the activity from once that activity is complete. The progression of menus and activities is shown in Figure 5.

# Diagram Description automatically generated

*Figure 5: Activity Diagram (Logical Viewpoint)*

# 7. Process Viewpoint

**Concern**: Behaviour and communication at runtime

**Stakeholders**: Software architect, software developers, budget personnel (see where usage of paid services occurs)

**Modelling** **Technique**: UML Sequence Diagram

Figure 6 depicts the communication that happens between the application and the Sentiment Analysis API, and between the application and the MySQL database, once a user has decided to enter input.

The communication between the application and the database is noteworthy because it only needs to occur once per journal entry saved, even though multiple nodes need to be created.

Diagram

Description automatically generated

*Figure 6: Journal Entry Creation Communication Sequence Diagram (Process Viewpoint)*

Figure 7 depicts a generic sequence of communications that will occur when the user selects an option that requires access to the database. The View will pass along the selection to the Controller.

Diagram

Description automatically generated

*Figure 7: Generic Persistent Data Access Sequence Diagram (Process Viewpoint)*

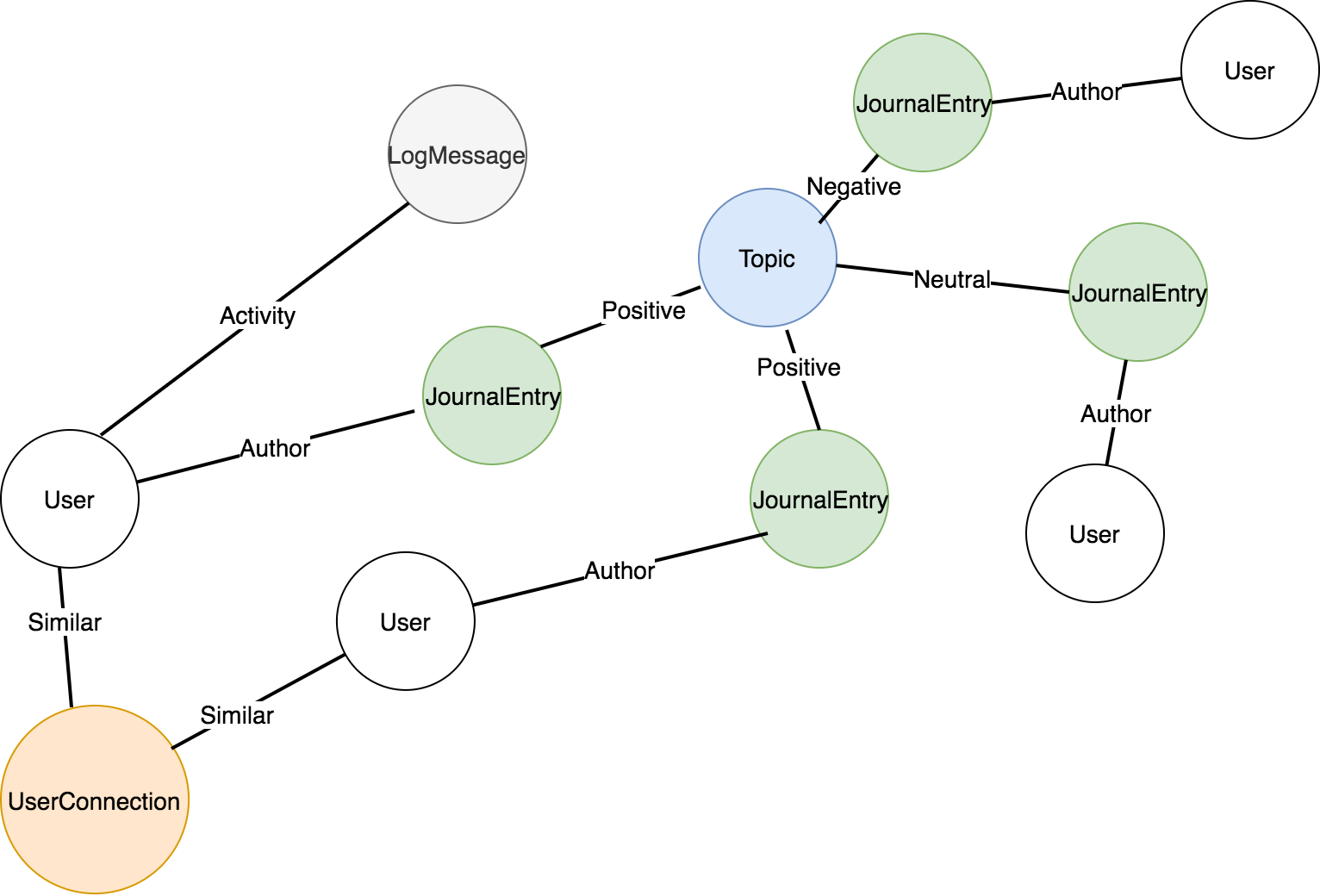
# 8. Graph Database Details

Types of Nodes:

* User
  + Stores information about a user
* JournalEntry
  + Stores the content and timestamp for a journal entry
* Topic
  + Stores a keyword or phrase that constitutes a topic
* UserConnection
  + Stores a GUID, topic, and sentiment that 2 users had
* LogMessage
  + Stores a string containing a log message

Types of Relationships:

* Sentiment
  + Relationship between a JournalEntry and a Topic (can be positive, negative, or neutral)
* Author
  + Relationship between a User and a JournalEntry
* Similar
  + Relationship between a User and a UserConnection
* Activity
  + Relationship between a User and a LogMessage



*Figure 8: Sample Graph Database Structure*

# 9. Configuration Data and Starting Point

The application will require a json file, named “config.json”, containing configuration data.

Example of required json file with key names:

{

"databaseHost": "redis-16263.c273.us-east-1-2.ec2.cloud.redislabs.com", "databasePort": 16263,

"databasePassword": "kT2ygstJVpJBASntruPq18ihhyyBgTcp",

"databaseName": "projectGraph",

"positiveSentiment": "p",

"negativeSentiment": "n",

"neutralSentiment": "none",

"APIURL": "https://api.meaningcloud.com/sentiment-2.1?",

"APIKey": "ef3686e753dfd83dbc9p4e45d8cc7cc8"

}

The “Main” method of the program is found in the file “Program.cs”.

# 10. Additional References

IBM Rational Software Architect. (n.d.). *Include relationships*. Retrieved June 26 2021,

from <https://www.ibm.com/docs/en/rational-soft-arch/9.7.0?topic=diagrams-include-relationships>

Microsoft Docs. (2018). *Design the infrastructure persistence layer*. Retrieved July

11 2021, from <https://docs.microsoft.com/en-us/dotnet/architecture/microservices/microservice-ddd-cqrs-patterns/infrastructure-persistence-layer-design>

Refactoring Guru. (n.d.). *Proxy*. Retrieved July 10, 2021, from

<https://refactoring.guru/design-patterns/proxy>