Lab 1 Draft – Sections 1, 2, 3

Angelos Angelopoulos

Department of Computer Science, Old Dominion University

CS 410: Professional Workforce Development I

Professor Janet Brunelle

May 4, 2020

Table of Contents

1. Introduction

Technology has greatly advanced over the past couple of decades and with ever-improving hardware and software as well as technology-literate generations an industry grew alongside it: video games. Video games have flourished, especially in the last decade. There are more than 2.47 billion gamers worldwide (Gough, 2019), and the video game industry is projected to reach a value of $196 billion in 2022, surpassing the film industry. Thousands of video games are released every year (Anderton, 2019). Just in 2018, more than nine thousand games were released for the PC on the Steam platform, and this number has been increasing in the past years (Gough, 2019). Of course, by also taking into consideration console and mobile games, this number increases to tens of thousands (Gough, 2019). With so many games coming out annually, a problem starts to surface. Gamers might want to play many of those new games but staying up to date for all of them can be time-consuming.

Video games are costly to produce, both in money and time. A typical video game takes one to seven years to make, and some games take even longer than that. For example, the popular game *Team Fortress 2*, was in development for nine years (Dietz, 2011). Because video games often have long development times, it can get tiresome to look for news online over such long timespans. Part of a game’s development budget includes marketing, as getting the word out for a video game is crucial for driving sales. Large development studios spend millions on marketing. This gives rise to another problem: indie video games (games from small development studios) may lose the public spotlight as time goes on and may even be forgotten. There are several reasons for this, from indie studios having smaller marketing budgets and staying in development for too long, to large development studios attracting more players and away from other games, hurting indie developers in the process.

The last problem is that information and news about video games is very decentralized and spread over many different sources. There are many independent video games news websites reporting on the industry, developers officially reporting on Twitter and their websites, developers and individuals uploading videos on YouTube, and individuals posting on Reddit. Even the news websites cannot keep track of the sheer number of games out there. The result is an increased burden on gamers, who must spend more time looking for information online.

To solve these problems, a way to aggregate, categorize, and rank all this information is needed. The solution is *GameEye*, a platform where users can search for and follow/watch video games. Users will have a personal watchlist of video games, and by following a game, *GameEye* will keep track of it and notify when new content is released online about it. *GameEye* will collect and categorize video game data from the internet, including news articles, tweets, Reddit posts, images, and YouTube videos. However, the application does not stop there. Using machine learning, news articles and tweets will be classified depending on what they are about, and a multi-factor score will be computed and assigned based on how important they are for users to know. *GameEye* aims to be the gamer's eye into the video game industry.

2. GameEye Product Description

GameEye has five core functionalities. The first is the aggregation of online resources about video games, including news articles, tweets, Reddit posts, images, and videos. The second is the organization of them by game and based on what type of resource they are (e.g. grouping news articles, grouping tweets). The third is the classification and ranking of news articles and tweets based on what they are about and their importance to the user, using machine learning. The fourth is the notification of users about new content. The fifth is the ability for users to search for games to add to their watchlists, with support for autocompleting game titles.

The goals of the application are to make the lives of video game players easier, as well as to help game developers maintain more attention to their games. *GameEye* aims to be the middleman between the gamer and the video game industry, performing tasks and analysis that the gamer would otherwise have to do manually. The application seeks to automate the discovery of new video game information and the categorization and ranking of it, with the goal of delivering to users what they need to know.

2.1. Key Product Features and Capabilities

GameEye is superior to other video game aggregators, like *N4G* (2020), because it aggregates a larger variety of resources and aggregates and displays information with finer granularity by default: per-game instead of per-genre or per-platform. Another main aspect in which *GameEye* is superior to other aggregators, especially *N4G,* is that it uses automatic aggregation, whereas *N4G* relies on contributors to search for and submit new content. Furthermore, the application is superior to its competition because it uses notifications. Other gaming aggregators, like *N4G*, do not offer notifications for new content. On top of all these, the application has features that make it into more than just a regular aggregator. It uses machine learning to classify news articles and tweets based on what they are about as well as to rank them based on their importance to users. The beforementioned features are the unique selling points of the application, and there is no other solution that offers them. The innovation of *GameEye* lies in its finer-level automated aggregation and machine learning capabilities. Features are described in more detail in the next few paragraphs.

General: The application will be designed and built as a progressive web application, a new generation of web applications intended to work anywhere that a web browser can work and provide offline support. This means that the application will be cross-platform, targeting desktops and mobile devices, and will include offline support, local caching, and connection interruption resiliency features.

Authentication: Using a secure third-party authentication provider, the application will provide secure login and registration for users and will support login and registration using external provider accounts such as social media, further increasing accessibility to users. Persistent sessions so that users do not have to login every time, as well as two-factor authentication and password recovery mechanisms, are to be implemented.

Account Management: Standard account management functionality including the ability for users to change their passwords, modify their profile information, as well as the ability to delete their accounts is planned.

Game Tracking: Every user will have a personal watchlist of video games they want to stay updated about. Aggregation of game thumbnails will be implemented to also add a visual cue about games next to their titles. Aggregated information will be organized by important updates (powered by machine learning), news articles, tweets, Reddit posts, images, and videos. Thumbnail images will be included for news articles and tweets. Finally, a list of the most-watched games on the platform will be implemented so that users can discover potentially good games.

Searching: Using data from the IGDB (Twitch, 2020) video game database, the application will provide the ability to search for video games based on their title. Support for autocompletion of game titles will also be implemented to enhance user experience.

Web Scraping: The application will automatically aggregate new information about video games watched by users at regular time intervals. Aggregated information includes news articles from video game news websites, tweets from official game Twitter feeds, Reddit posts from official game subreddits, videos from official game YouTube channels, and images. New game information is kept for 90 days and information older than 90 days will be archived but will still be accessible to users.

Machine Learning: The application will use machine learning models and natural language processing to classify news articles and tweets based on what they are about (e.g. release date announcement, major game update, minor game update). An importance score based on the importance of an article or tweet for the user will also be computed. Lastly, the application will extract important information (such as version numbers and dates) from news articles and tweets based on their classification.

Notifications: The application will send out cross-platform push-notifications to users for new game updates. A UI count of the notifications for each game and each resource category will also be displayed. Notifications about suggested video games based on the most-watched games list are also planned.

Settings: The application will be customizable in that users will be able to choose if they want to see archived resources, if they want to see importance scores, if they want to receive notifications, and if so, for which resource categories to be notified about.

Through all the above features, *GameEye* seeks to make it effortless for gamers to stay up to date about their favorite games. The automated web scraping of the application, as well as its machine learning capabilities, seek to minimize human intervention and save time. *GameEye* will scrape internet sources at regular time intervals, analyze the scraped data using machine learning, and notify users about what it retrieved. A direct result is that users will get all the information delivered to them instead of having to go look for it.

2.2. Major Components (Hardware/Software)

GameEye consists of several in-house and third-party components. The in-house components are: (**1**) frontend, (**2**) primary backend, (**3**) web scraping backend, (**4**) machine learning backend, and (**5**) main database server. The third-party components are: (**1**) *Auth0*, an OIDC authentication service (Auth0 Inc., 2020), (**2**) *IGDB*, a video game database (Twitch, 2020), and (**3**) *Firebase Cloud Messaging*, a push notification service (Google Inc., 2020). All the above components are shown in *Figure 1* and described in more detail in the following paragraphs.

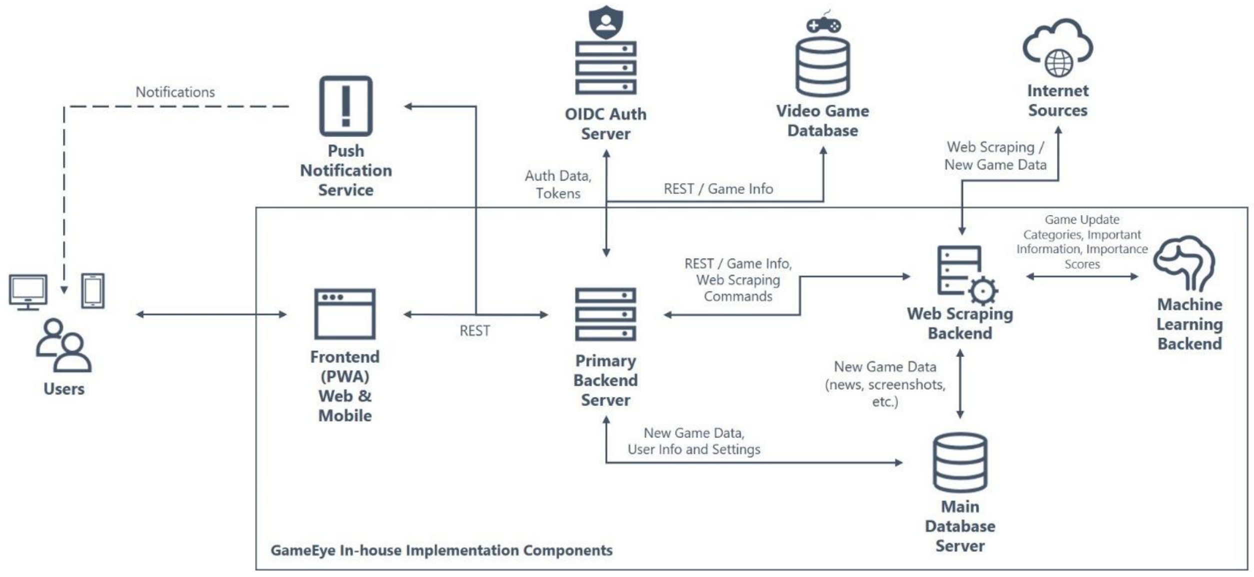
Notifications
Push Notification Service
OIDC Auth Server
Video Game Database
Internet Sources
Auth Data, Tokens
REST / Game Info
Web Scraping / New Game Data
Game Update Categories, Important Information, Importance Scores
REST / Game Info, Web Scraping Commands
Web Scraping Backend
Machine Learning Backend
Users
Frontend (PWA) Web & Mobile
REST
Primary Backend Server
New Game Data (news, screenshots, etc.)
New Game Data, User Info and Settings
Main Database Server
GameEye In-house Implementation Components

Figure 1: Shows the major functional components of *GameEye* and their interaction.

***Auth0*, OIDC Authentication Service (3rd Party):** An authentication platform offering flexible drop-in authentication and authorization services to applications. *GameEye* will use it to securely authenticate users and store passwords.

***IGDB*, Video Game Database (3rd Party):** A large video game database owned by *Twitch* (Humphries, 2019) and by extension *Amazon*. It contains a lot of information about video games, such as important URLs, platforms, genres, and descriptions. *IGDB* offers a REST API to access the content of the database. *GameEye* will use it to retrieve video game titles and URLs for web scraping.

Firebase Cloud Messaging, Push-Notification Service (3rd Party): A cross-platform messaging solution developed by *Google* that can be used to send notifications. *GameEye* will use it to send push-notifications.

Frontend (In-House): A progressive web application (PWA) (Google Inc., 2020) built with *Angular* (Google Inc., 2020), *Workbox* (Google Inc., 2020), HTML, CSS, and TypeScript that will serve as the UI for *GameEye*. Because *GameEye* will be a PWA, it will be cross-platform and will target desktop and mobile devices. It is an external-facing web server.

Primary Backend (In-House): The backend that communicates with the frontend, powers its dynamic functionality, and coordinates the web scraping backend. Built with *Spring* framework (VMware Inc., 2020) and Java. It is an external-facing web server and provides a REST API for communication with the frontend, web scraping backend, databases, and other third-party services.

Web Scraping Backend (In-House): The backend that performs web scraping. Communicates with the machine learning backend to classify scraped video game data and extract important information before putting it in the database. Built with *Spring* framework, *jsoup* (Hedley, 2020) for web scraping, and Java. It is an external-facing web server exposing a REST API for launching web scraping operations.

Machine Learning Backend (In-House): The backend that performs machine learning. It communicates with the web scraping backend to classify video game data and extract important information. Built with Python, Keras (Chollet et al., 2020), scikit-learn (Cournapeau et al., 2020), and spaCy (Explosion AI, 2020). It is an internal GPU-powered *TensorFlow Serving* (Google Inc., 2020) server exposing a REST API for machine learning inference.

Main Database Server (In-House): The internal *MongoDB* (MongoDB Inc., 2020) database that stores all the data related to *GameEye*, including data retrieved from web scraping.

All the in-house components ideally require dedicated hardware for optimal performance. Thus, the required hardware for the application are servers to host the in-house components. An alternative to dedicated hardware is *Docker* (Docker Inc., 2020), for running multiple components as containers on a single or a couple of high-performance servers.

3. Identification of Case Study

GameEye is being developed for casual and highly active gamers alike. Both types of end-users benefit from the time-saving the application offers. Casual gamers who play a few games less frequently would value a plug-and-play solution where they can enter a few games they are interested in and get notified about important news leaving them more time to focus on other matters. Highly active gamers who frequently play and are interested in many games will also benefit from the application as they will not have to manually keep track of so many games. The future users of the application are not expected to change without significant updates to the feature set of the application.

4. References

Anderton, K. (2019, June 26). *The Business of Video Games: Market Share for Gaming Platforms in 2019 [Infographic]*. Retrieved from: https://www.forbes.com/sites/kevinanderton/2019/06/26/the-business-of-video-games-market-share-for-gaming-platforms-in-2019-infographic/#c1793427b254

Dietz, J. (2011, June 23). *30 Games That Emerged from Development Hell*. Retrieved from: https://www.metacritic.com/feature/games-that-shed-vaporware-status

Gough, C. (2019, August 9*)*. *Number of games released on Steam 2018*. Retrieved from: https://www.statista.com/statistics/552623/number-games-released-steam/

Gough, C. (2019, August 9). *Number of gamers worldwide 2021*. Retrieved from: https://www.statista.com/statistics/748044/number-video-gamers-world/

Gough, C. (2019, October 9). *Google Play: Number of available games by quarter 2019*. Retrieved from: https://www.statista.com/statistics/780229/number-of-available-gaming-apps-in-the-google-play-store-quarter/

Humphries, M. (2019, September 18). *Twitch Acquires Gaming Database Website IGDB*. Retrieved from: https://www.pcmag.com/news/twitch-acquires-gaming-database-website-igdb

Rose, M. (2014, May 15). *How the surge of Steam releases will affect game developers*. Retrieved from: https://www.gamasutra.com/view/news/217583/How_the_surge_of_Steam_releases_will_affect_game_developers.php

Google Inc. (2020). *Progressive Web Apps*. Retrieved from: https://web.dev/progressive-web-apps/

Twitch. (2020). *IGDB*. Retrieved from: https://www.igdb.com/discover

Auth0 Inc. (2020). *Auth0*. Retrieved from: https://auth0.com/

Google Inc. (2020). *Firebase Cloud Messaging*. Retrieved from: https://firebase.google.com/products/cloud-messaging

Google Inc. (2020). *Angular*. Retrieved from: https://angular.io/

Google Inc. (2020). *Workbox*. Retrieved from: https://developers.google.com/web/tools/workbox

VMware Inc. (2020). *Spring Framework*. Retrieved from: https://spring.io/

MongoDB Inc. (2020). *MongoDB*. Retrieved from: https://www.mongodb.com/

Google Inc. (2020). *TensorFlow Serving*. Retrieved from: https://www.tensorflow.org/tfx/guide/serving

Chollet, F. et al. (2020). *Keras*. Retrieved from: https://keras.io/

Cournapeau, D. et al. (2020). *Scikit-learn*. Retrieved from: https://scikit-learn.org/stable/

Explosion AI. (2020). *spaCy*. Retrieved from: https://spacy.io/

Docker Inc. (2020). *Docker*. Retrieved from: https://www.docker.com/

Hedley, J. (2020). *Jsoup HTML parser*. Retrieved from: https://jsoup.org/

N4G. (2020). *N4G*. Retrieved from: https://n4g.com/