Lab 1 - GameEye Product Description

Angelos Angelopoulos

Old Dominion University

CS 411W: Professional Workforce Development II

Professor Janet Brunelle

October 4, 2020

Version 2

Table of Contents

1. Introduction 3
2. GameEye Product Description 4
 2.1. Key Product Features and Capabilities 5
 2.2. Major Components (Hardware/Software) 8
3. Identification of Case Study 12
4. GameEye Product Prototype Description 12
 4.1. Prototype Architecture 13
 4.2. Prototype Features and Capabilities 14
 4.3. Prototype Development Challenges 17
5. Glossary 19
6. References 21

List of Figures

Figure 1: Major Functional Components of GameEye 9
Figure 2: Major Functional Components of the GameEye Prototype……………………..…13

List of Tables

Table 1: Features that will be available in the real-world product and the prototype………….. 12

1. Introduction

The video game industry has seen incredible growth in the past decades. Today, there are more than 2.47 billion gamers worldwide (Gough, 2019b), and the video game industry is projected to reach a value of $196 billion in 2022 (Webb, 2019), surpassing the film industry. Thousands of video games are released every year (Anderton, 2019). Just in 2018, more than 9000 games were released for personal computers on the Steam platform, and this number has been increasing in the past years (Gough, 2019a). By also considering console and mobile games, this number increases to tens of thousands (Gough, 2019c). With so many games releasing annually, a problem appears. Gamers may desire to play many upcoming games but staying up to date with 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 becomes tiresome to look for news online. Part of a game's development budget includes marketing, as promoting 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 and may even be forgotten with time. There are several reasons for this: (a) indie studios having smaller marketing budgets, (b) games staying in development for too long, and (c) large development studios attracting more players and away from other games hurting indie developers in the process.

Information and news about video games are very decentralized and spread over many different sources. There are many independent video games news websites that report on the industry. Game developers officially report on Twitter and their official game websites, developers and gamers upload videos on YouTube, and developers and gamers post on Reddit. Even the news websites cannot keep track of the sheer number of games. 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 game information is needed. The solution is GameEye, a platform where users can search for and follow video games. Users will have a personal watchlist of video games. GameEye will keep track of followed games and notify users when new content is released online about them. GameEye will collect and categorize video game data from the internet, including news articles, tweets, Reddit posts, images, and YouTube videos. Using machine learning, news articles and tweets will be classified depending on their content, and a multi-factor impact score will be computed and assigned based on how impactful the news is for a game and its users. GameEye aims to be the gamer’s eye in the video game industry.

2. GameEye Product Description

GameEye is a product that seeks to organize the vast amount of information in the video game industry. This is achieved by automating the discovery of new video game information in various formats such as news articles, tweets, Reddit posts, images, and videos. GameEye processes and analyzes discovered information using machine learning to reduce noise and deliver relevant information to users. 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.

2.1. Key Product Features and Capabilities

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 online resources by game and type (e.g. grouping news articles, grouping tweets). The third functionality uses machine learning to classify and rank news articles and tweets based on their subject and their impact on the game and the user. 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.

GameEye is superior to other video game aggregators because it aggregates a larger variety of resources 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 aggregation services is that it uses automatic aggregation. This means that the content is not user-curated but is collected and processed automatically. Furthermore, the application uses notifications to inform users about new content. Other gaming aggregation services do not offer notifications for new content. GameEye also uses machine learning to classify information based on the subject and to rank it based on its impact on games and the users. These features are unique selling points (USPs) of the application. The innovation of GameEye lies in its finer-level automated aggregation and machine learning capabilities.

The application is designed and built as a progressive web application. Progressive web applications are a new generation of web applications. Unlike traditional web applications, progressive ones can work offline and provide features while the user is offline by caching previously retrieved data. GameEye is cross-platform, targeting desktops and mobile devices, and includes offline support, local caching, and connection interruption resiliency features.

GameEye provides modern and robust authentication features. Using a secure authentication provider, the application provides secure login and registration for users and supports 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. 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 available.

The core feature of GameEye is allowing users to track games. Every user has a personal watchlist of video games for which they want to stay updated. Game thumbnails are aggregated and displayed to add a visual cue about games next to their titles. Aggregated information is organized by news articles, tweets, Reddit posts, images, and videos. Thumbnail images are included for news articles and tweets. Finally, a list of the most-watched games on the platform are implemented so that users can easily discover promising games.

GameEye is a data-driven application and makes use of a large amount of video game data. Using data from the IGDB video game database, the application provides the ability to search for video games based on their title. Support for autocompletion of game titles enhances the user experience by allowing users to partially type a game title. Even if a user misspells a title, the autocompletion returns results.

Web scraping and machine learning are the bread and butter of GameEye. Using this analogy, web scraping is the bread or raw data, whereas machine learning is the butter that enhances the data. The application uses web scraping to automatically aggregate new video game information for 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 is archived but is still accessible to users. Most users care about recent information, so there is no need to maintain an extensive active list of video game news.

The application uses machine learning models and natural language processing to classify news articles and tweets based on their subject category. Categories include release date announcements, major game updates, minor game updates, and more. An impact score is also computed. This score is based on the impact of the news that an article reports on its reference game or its players. The impact score acts as the noise reduction element of the application, as users can choose the impact score range for articles they want to receive notifications for. For example, users may wish to ignore low-impact articles which are not very significant. Lastly, the application extracts important information, such as version numbers and dates, from news articles and tweets based on their classification.

For users to learn about recent news they need to be notified. Because of this, notifications are a critical feature for GameEye. The application sends out cross-platform push-notifications to users for any news it finds. The number of notifications for each game and each resource category is also displayed so that users can know exactly what type of new information was found and for what game. Notifications about suggested video games based on the most-watched games list are also sent. These notifications serve as both a means for users to discover potential games to play and as a way for GameEye to advertise specific games.

The application provides a range of customization options. Users can choose if they want to see archived resources, if they want to see impact scores, and if they want to receive notifications. Users are also able to select which resource categories to be notified about. Finally, users can choose the range of impact scores to receive notifications for, if they have selected that they want to see impact scores. By controlling the range of impact scores, users also control the number of articles they are notified about, which helps reduce noise.

GameEye seeks to make it easy 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 by increasing the relevance of information and making it easier for users to find it. GameEye scrapes internet sources at regular time intervals, analyzes the scraped data using machine learning, and notifies users about what it retrieved. A direct result is that users get all the information delivered directly to them.

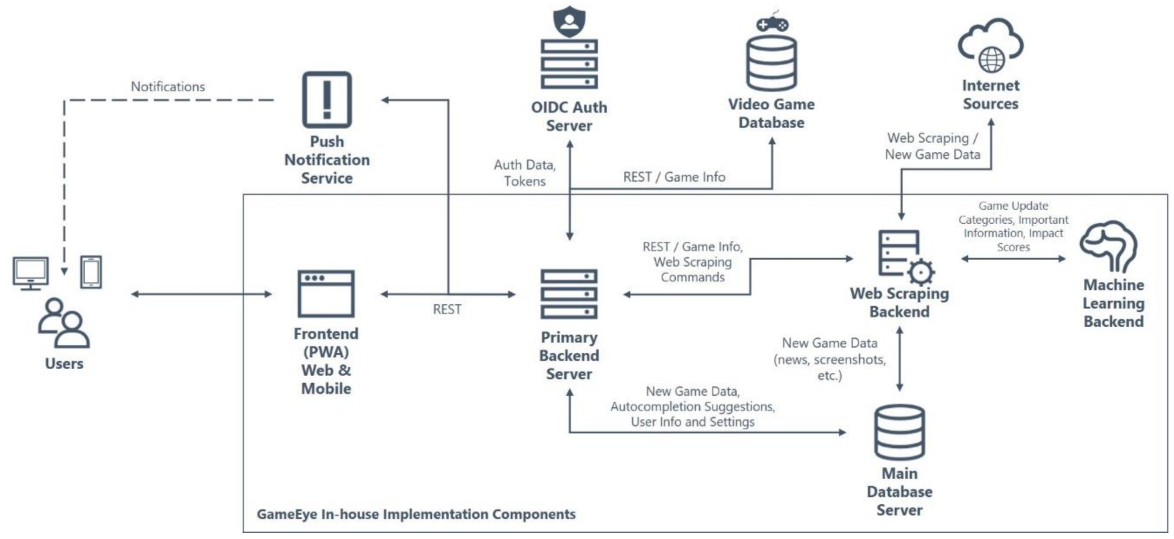
2.2. Major Components (Hardware/Software)

GameEye consists of several in-house and third-party components. The in-house components are: (a) frontend, (b) primary backend, (c) web scraping backend, (d) machine learning backend, and (e) main database server. The third-party components are: (a) IGDB, a video game database, (b) Firebase Authentication, a Google authentication service, and (c) Firebase Cloud Messaging, a push notification service. These components are shown in Figure 1.

[This space is intentionally left blank]

Figure 1

Major Functional Components of GameEye.


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, Impact 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, Autocompletion Suggestions, User Info and Settings
Main Database Server
GameEye In-house Implementation Components

The GameEye frontend is a progressive web application built with, Workbox, HTML, CSS, and TypeScript. It serves as the user interface for GameEye. As a progressive web application, it is cross-platform and targets desktop and mobile devices. The frontend must be cross-platform as this allows better coverage of the massive audience of video game players no matter the platform they are using. The frontend is the same for every platform and the codebase is shared, which eliminates the need for porting the application to different platforms.

The frontend communicates with the primary backend to retrieve data. The primary backend also coordinates and schedules web scraping. The backend is built with the Spring Boot framework and Java. It provides a REST API for communication with the frontend, web scraping backend, databases, and other third-party services. All endpoints are secured using a role-based system, which allows GameEye to have different types of users, such as regular users, premium users, and administrators.

Web scraping is implemented as a dedicated backend because it is computationally expensive. The web scraping backend exposes a REST API that allows the primary backend to control it. Essentially, the primary backend maintains a schedule and orders the web scraping backend to search and retrieve new video game information at regular intervals. The web scraping backend also communicates with the machine learning backend to classify scraped video game data and extract important information before putting it in the database. It is built with Spring Boot framework, jsoup, and Java. In a real-world environment, the web scraping backend will be implemented as a cluster of computers, so that web scraping tasks can be evenly distributed by the primary backend overseer.

Machine learning is also implemented as a dedicated backend. It communicates with the web scraping backend to process new data. This backend is responsible for computing impact scores, classifying news articles and tweets, as well as extracting important information. It is built with Python, Keras, scikit-learn, and spaCy. The machine learning backend uses a GPU-powered TensorFlow Serving server exposing a REST API for high-performance machine learning inference.

GameEye has extensive data storage requirements and requires a dedicated database server. This is called the main database server. It includes a MongoDB database that stores all the data related to GameEye, including data retrieved from web scraping. Notably, it stores data retrieved from IGDB for increased performance and redundancy. It also includes an ElasticSearch database that stores game titles. More importantly, it maintains complex data structures for fast autocompletion. Because of this, the ElasticSearch database provides autocompletion suggestions for game titles. Both the primary backend and the web scraping backend use the databases.

To navigate the vast amounts of video game information, GameEye makes use of IGDB, a large video game database. IGDB is owned by Twitch (Humphries, 2019) and by extension Amazon. It contains a large amount 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 uses IGDB to retrieve video game information necessary for web scraping, such as game titles and URLs. Notably, IGDB permits creating a competing product, which essentially allows the replication of their database and the storage of the information in GameEye's main database, bypassing the need to use their API in the future.

User authentication is a complex matter that is difficult to get right. For this reason, GameEye makes use of Firebase Authentication, an authentication service offered through Google's Firebase platform. Using Firebase Authentication, GameEye allows users to login with an email and password, login with external providers like their Google accounts, as well as register for new accounts.

Lastly, GameEye uses Firebase Cloud Messaging to send notifications to users. When the primary backend receives new video game data from the web scraping backend, it creates notifications for every user that watches games for which new data was found. The primary backend uses Firebase Cloud Messaging to dispatch these notifications on a large scale.

The frontend, backends, and main database server 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 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 application. Casual gamers who play a few games less frequently would value a plug-and-play solution where they can enter a few games that they are interested in and save time by getting notified about important news. Highly active gamers who frequently play and are interested in many games 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.

The case study for the application is the ODU Gaming Club. This user group is composed of ODU students who are interested in video games. These users may either be casual or highly active gamers and fit GameEye's target groups. Around 4 people are needed to test the application. Each user will build a personal watchlist and over some time observe the behavior of GameEye. The necessary feedback is: (1) timeliness of GameEye in discovering news articles and sending notifications, (2) accuracy of the impact scores assigned by GameEye, (3) relevance of the articles found by GameEye, (4) presence of articles that refer to the wrong game, and (5) overall changes they would like to see. ODU Gaming Club members make for a great group of testers who can provide feedback to improve GameEye. It is important that their insight is obtained during development, and thus are a valuable case study.

4. GameEye Product Prototype Description

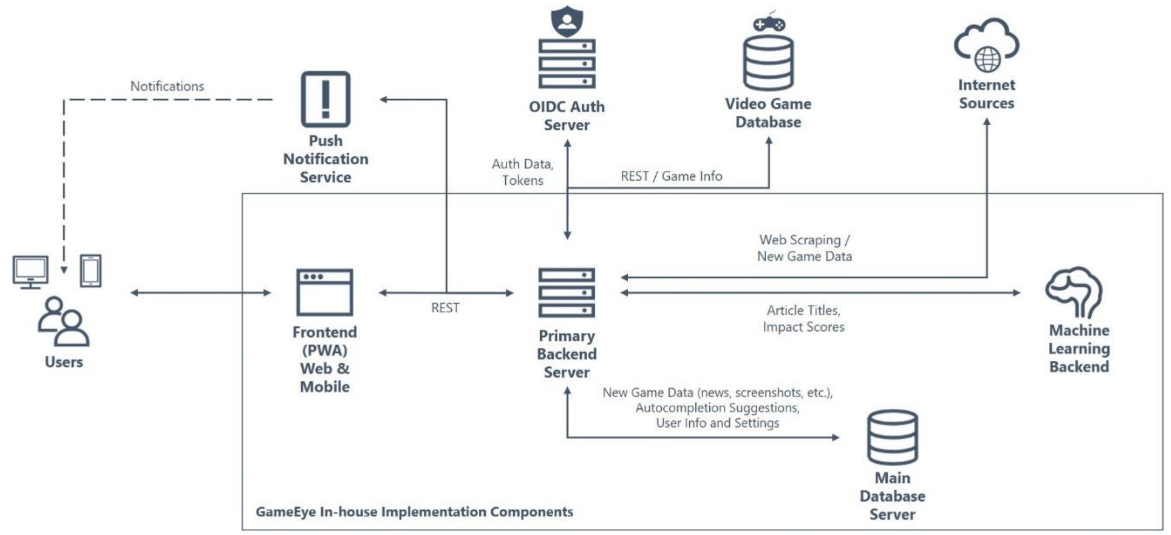
The prototype contains a small subset of features of the real-world product. It is focused on functionality related to news articles. The biggest difference between the prototype and the real-world product is the number of resources that are supported. The real-world product supports news articles, tweets, Reddit posts, images, and videos. The prototype, however, only supports news articles. Additionally, machine learning features in the prototype are limited to impact scoring. Table 1 lists the features of the real-world product and the prototype in more detail.

4.1. Prototype Architecture

The prototype uses the same architecture as the real-world product. The primary difference is that all 5 of the in-house components are hosted on one physical server, the virtual machine provided by the CS department, instead of dedicated ones. Furthermore, there are no replicated servers and no load balancing, which would be present in the real-world product. There is only one instance of the frontend server, the primary and machine learning servers, and the database. The software components are: (a) frontend, (b) primary backend, (c) machine learning backend, (d) MongoDB database, and (e) ElasticSearch database. The primary backend also performs web scraping. The two databases are part of the main database server in the MFCD. The prototype MFCD is shown in Figure 2.

Figure 2

Major Functional Components of the GameEye Prototype.


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

4.2. Prototype Features and Capabilities

The prototype aims to demonstrate the real-world applications of GameEye. It primarily demonstrates how a user: (a) can access the platform, (b) add games to their watchlist, (c) see recent news articles for each game and, (d) see impact scores for each article, and (e) receive notifications for news articles. Furthermore, the prototype demonstrates how GameEye functions internally, and how web scraping, machine learning, game title autocompletion, and user authentication and authorization are performed. Overall, the prototype seeks to show how users can save time by using the application to keep track of video game news.

From a functional standpoint, the goals are to: (a) have a functional authentication system, (b) have functional news article web scraping, (c) have functional watchlists and game tracking, (d) have functional machine learning impact scoring, and (e) have functional news article notifications. Optimally, the prototype needs to be a usable product, albeit for a limited number of games and tracking news articles only.

Table 1

Lists the features that will be available in the real-world product and the prototype.

Category	Feature	RWP	Prototype
General			
	Cross-Platform Support (Desktop, Mobile)	Full Functionality	Full Functionality
	Offline Support	Full Functionality	No Functionality
	Local Caching	Full Functionality	No Functionality
	Connection Interruption Resiliency	Full Functionality	No Functionality
Authentication			
	User Login	Full Functionality	Full Functionality
	User Registration	Full Functionality	Full Functionality
	External Provider Login & Registration	Full Functionality	Full Functionality
	Persistent Sessions	Full Functionality	Full Functionality

	Password Recovery	Full Functionality	Full Functionality
Account Management			
	Change Profile Information	Full Functionality	Full Functionality
	Change Password	Full Functionality	Full Functionality
	Delete Account	Full Functionality	No Functionality
Searching			
	Search for Games	Full Functionality	Partial Functionality
	Search Autocompletion	Full Functionality	Partial Functionality
Game Tracking			
	Add Games to Watchlist	Full Functionality	Partial Functionality
	News Articles (Web Scraping)	Full Functionality	Partial Functionality
	Tweets (Web Scraping)	Full Functionality	No Functionality
	Reddit Posts (Web Scraping)	Full Functionality	No Functionality
	Images (Web Scraping)	Full Functionality	No Functionality
	Videos (Web Scraping)	Full Functionality	No Functionality
	Resource Thumbnails	Full Functionality	Partial Functionality
	Game Thumbnails	Full Functionality	Partial Functionality
	Source Website Redirection	Full Functionality	Partial Functionality
	Resource Organization	Full Functionality	Full Functionality
	Archived Resources	Full Functionality	No Functionality
	Most-Watched Games List	Full Functionality	Full Functionality
Settings			
	Show Archived Resources Option	Full Functionality	No Functionality
	Show Impact Scores Option	Full Functionality	Full Functionality
	Receive Notifications Option	Full Functionality	Full Functionality
	Receive Notifications Per Category Option	Full Functionality	No Functionality
	Receive Notifications Per Impact Score Option	Full Functionality	Full Functionality
Machine Learning			
	Impact Scoring	Full Functionality	Partial Functionality
	Resource Classification	Full Functionality	No Functionality
	Important Information Extraction	Full Functionality	No Functionality
Notifications			
	Push-Notifications for New Game Updates	Full Functionality	Full Functionality
	UI Count of Notifications for Each Game	Full Functionality	Full Functionality
	UI Count of Notifications for Each Resource Category	Full Functionality	Full Functionality

	Cross-Platform Notifications	Full Functionality	Full Functionality
	Suggested Video Game Notifications	Full Functionality	No Functionality

The prototype will be a functional product that is usable for a limited scope. However, it will lack many of the features of the real-word product, as seen in Table 1. The prototype will be a cross-platform web application but will not have many of the features necessary for a progressive web application, such as offline support, local caching, and connection interruption resiliency. The prototype will offer a full set of authentication features that are equivalent to those of the real-world product. For account management, the prototype will allow users to change their profile information, such as their email and name, as well as their password. No functionality will be provided for deleting accounts, however. Game searching will be limited in scope, as only a subset of games that exist will be discoverable.

The most important features of GameEye are in the “Game Tracking” category of Table 1. The prototype will only allow the tracking of news articles. News article and game thumbnails will be supported. Archived resources will not be implemented. In terms of settings, there will be no options for displaying archived resources and choosing which resource categories to receive notifications for. Machine learning will be partially functional, with only impact scores having functionality. However, the accuracy of the impact score model cannot be guaranteed due to the lack of training data. Finally, for notifications, the prototype will support all features of the real-word product except notifications for suggested video games.

Success will be demonstrated by building a high-quality prototype that demonstrates how GameEye works and how it can be extended to support new features. On the frontend side, the goal is to deliver an intuitive and modern user interface using elements of Material design with a simple user flow and a straightforward user experience. On the backend side, the goal is to deliver high-performance logic, endpoint security, and reliability through robust exception handling and test-driven design.

Appropriate steps will be taken to mitigate the several identified risks. News article web scraping will be performed on RSS feeds, which rarely change. Portions of IGDB will be replicated into our database to increase redundancy, performance, and ensure that GameEye can work even if IGDB goes offline. To ensure user security, Firebase Authentication as well as several third-party authentication providers, like Google and Microsoft, will be used for authenticating users.

4.3. Prototype Development Challenges

GameEye is a complex application and involves a large amount of data. Several challenges must be overcome during prototype development. First, new technologies and frameworks which are required for implementation need to be learned. Second, around 12,000 news article titles need to be collected and labeled for training the machine learning model for impact scoring. Third, robust web scraping that is easy to modify in case source websites change needs to be implemented. Fourth, as the application works with a large amount of data, a database schema that minimizes data duplication needs to be devised. Fifth, a portion of the IGDB database needs to be replicated, which is challenging as the REST API is extensive and the database contains a large amount of data. Sixth, because GameEye is a cross-platform application, we need to ensure that it works in a variety of platforms and screen sizes. Lastly, GameEye tries to solve a multiplex machine learning problem that involves analyzing the text of news article titles to deduce the impact of the news they report. The combinatoric space of possible words is large. As a result, a great amount of training data is needed to achieve high accuracy. It is not possible to collect and label such a large amount of data for the prototype, so the accuracy of the machine learning model cannot be guaranteed.

[This space is intentionally left blank]

5. Glossary

Aggregator: A platform or service that collects and centralizes data.

Angular Framework: Platform for building mobile and desktop applications.

API: Application Programming Interface; a set of functions and procedures allowing the creation of applications that access the features or data of an operating system, application, or other services.

Amazon Web Services (AWS): Amazon subsidiary that provides on-demand cloud computing services and APIs.

CSS: Cascading Style Sheets; used to stylize webpages.

Guest: Initial role for users who have not created an account on GameEye.

Hitlist: List of highly watched video games by users.

HTML: HyperText Markup Language; used as markup for documents meant to be displayed in a web browser.

Impact Score: A score from 1 to 3 on the impact some news has on a game and its players. It is computed using machine learning.

Internet Games Database (IGDB): Database of known video games, accessed through a REST API to populate GameEye’s database.

Indie Games: Games developed by individuals or smaller teams of people without the financial support of larger game publishers.

IDE: Integrated Development Environment. A software application that includes a set of programming tools, notably a code editor and a debugger interface, to facilitate programming.

IntelliJ IDEA: IDE developed by JetBrains to write Java applications and will be used in the backend development of GameEye.

JavaScript: Object-oriented programming language used to create dynamic behavior on webpages.

JSoup Library: Java library for working with real-world HTML.

JUnit Java Framework: A testing framework for Java.

Keras: An open-source neural-network library written in Python.

MongoDB: A cross-platform document-oriented database program.

Noise Filtering: Removal of news articles and other content that is irrelevant or unimportant to the user.

OIDC Authentication: Open-ID Connect (OIDC) is an authentication protocol based on the OAuth 2.0 family of specifications.

Progressive Web Application (PWA): A type of application software delivered through the web which is built using common web technologies including HTML, CSS, and JavaScript. PWAs are an evolution of traditional web applications and can be used to a certain degree while offline.

Python: An interpreted, high-level, general-purpose programming language.

Representational State Transfer (REST): A software architectural style used in creating web services.

RSS Feed: Really Simple Syndication (RSS) is a web feed that allows users and applications to access updates to websites in a standardized, computer-readable format.

Scikit-learn: Open-source machine learning library for the Python programming language.

SpaCy: Open-source library for advanced natural language processing.

Spring Framework: Application framework and inversion of control container for the Java platform.

Tester: GameEye prototype users who will provide feedback on their experience with the application.

Web Scraping: Automated extraction of data from websites.

WebStorm: IDE developed by JetBrains for writing JavaScript and web-related code.

6. 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

Web, K (2019, Oct 1). *The $120 billion gaming industry is going through more change than it ever has before, and everyone is trying to cash in*. Retrieved from: https://www.businessinsider.com/video-game-industry-120-billion-future-innovation-2019-9