Lab 1 – GameEye Product Description

Jacob L. Cook

Old Dominion University

CS411W

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 4

2.2 Major Components (Hardware/Software) 7

3 Identification of Case Study 9

4 GameEye Product Prototype Description 10

4.1 Prototype Architecture (Hardware/Software) 11

4.2 Prototype Features and Capabilities 12

4.3 Prototype Development Challenges 15

5 Glossary 17

6 References 20

List of Figures

Figure 1: Real-World Product Major Functional Component Diagram 7

Figure 2: Prototype Major Functional Component Diagram 11

List of Tables

Table 1: Table of Comparison Between RWP and Prototype: Part 1 of 2 13

Table 2: Table of Comparison Between RWP and Prototype: Part 2 of 2 14

1 Introduction

Video games are one of the most popular forms of entertainment worldwide. According to Gough (2019a), the rate at which video games are released worldwide on Steam, a prominent video game platform, is increasing exponentially. Gough (2019a) provides data indicating that, worldwide, the number of new video games released on Steam grew from 4,207 new games in 2016 to 9,050 new games in 2018. In addition to the increasing video game release rate, the community of active gamers is expanding significantly. Gough (2019b) predicts that, by 2021, the number of video gamers worldwide will surpass 2.7 billion. For new and experienced gamers alike, the expanding library of video games is overwhelming.

The large number of video games and long development timeframes increase the difficulty for gamers to find new desirable games. Gamers struggle to maintain awareness throughout the development of new video games. There is great variation in the number of years from the date a game is announced to the date a game is released. Gamers are negatively impacted by long development timeframes for video games. Dietz (2011) illustrates that a prime example of a game with a long development timeline is Team Fortress 2, which was announced in 1998 and eventually released in 2007. While under development, games have distinct statuses, such as “announced,” “delayed,” and “released.” For games with extensive development timeframes, gamers face the difficulty of staying apprised of new information and status updates.

Game developers, especially those with small marketing budgets, suffer financially from uninformed video game consumers. To maintain awareness of video game news, gamers use social media platforms, game news websites, and game databases. Common video game news sources include Twitter, YouTube, Reddit, Steam, Google, IGN, and IGDB. Due to the overwhelming amount of game data, consumers need a centralized platform to maintain awareness of video game news.

GameEye is a platform for consumers to access game news and updates from a central location. Users can search for their favorite games and curate a personal watchlist. Each user can add a game they wish to follow to their watchlist. Once a game is added to the user's watchlist, they will receive notifications when GameEye finds new related content. Rather than bombarding users with all information related to a game, machine learning classifies news articles and Tweets into specific content categories. In addition to content categorization, each news resource is prioritized by a multifactor score computed by machine learning. GameEye informs users of the most popular games currently added to other watchlists. By centralizing game news, providing users with personal watchlists, and ranking information based on impact, GameEye is the gamer's eye into the video game industry.

2 GameEye Product Description

GameEye's most important goals are to provide personalized user watchlists and notify user about new video game content, such as news articles and Tweets. Personal watchlists are populated with video game titles that users find within the GameEye search feature. Users will receive notifications when new content is found for a game in their watchlist. In addition, video game content is ranked by impact to eliminate irrelevant information. GameEye's design leverages progressive web application (PWA) technologies – enabling usage on desktop and mobile devices. PWA architecture uses common web technologies to deliver the same user experience on any device with a modern web browser.

2.1 Key Product Features and Capabilities

GameEye implements several authentication features to protect users while facilitating a user-friendly experience. Users have access to a secure login and registration portal through the application. Login and registration features are available through GameEye via Firebase, or through external providers including Google and Microsoft. Persistent sessions, which remember

users who have recently signed in remove the inconvenience of constant authentication. To increase the security for users, two-factor authentication is a core feature of GameEye. GameEye employs recovery mechanisms to mitigate the risk of users forgetting their passwords.

Users can manage several aspects of their account through GameEye's account management portal. For instance, users can change their passwords. Users can also modify their profile information including first name, last name, and email address. Finally, users can delete their GameEye accounts.

Customizable watchlists are the core feature that GameEye provides to users for game tracking. When users identify a game of interest, they add the game to their watchlist. The personal watchlist allows users to maintain awareness of games of interest while ignoring games they find personally irrelevant. User watchlists contain a list of game titles that a user is following and a thumbnail image representing the game.

To structure game content, GameEye organizes information into six distinct categories: important updates, news articles, Tweets, Reddit posts, images, and videos. Game content is populated in the user's watchlist for each game they are following. Users can find a link to full content in each category within their personal watchlist along with a thumbnail image for news articles and Tweets. To showcase popular games, the most-watched games list is available for users. GameEye determines game popularity by the total number of followers for a specific game title. GameEye allows users to search for video games by title and quickly add a game to their watchlist. As a user types the game title, auto-completion assists the user in locating the desired video game. For users without a specific game title in mind, browsing the most-watched games list serves as another search method.

GameEye uses web scrapers to retrieve the content that populates a user's watchlist categories. Web scraping gathers news articles from video game news websites – removing the

need for users to navigate to dozens of web sources. GameEye locates Tweets about video games from official publisher Twitter feeds via web scraping. Web scrapers find Reddit posts from official game subreddits. After scraping YouTube for videos about video games, users are provided with a list of curated video content. Web scraping enables GameEye to show users images relevant to their watched video games.

Users receive push-notifications for new game updates whether using GameEye on desktop or mobile systems. The user interface provides a count of notifications for each game and category in a user's watchlist. Users receive notifications to suggest games on the most-watched games list in which they may be interested.

GameEye provides users with a customizable experience managed through content, notification, and general settings. Content preferences allow users to determine if they would like to view archived resources, which are resources older than 90 days. Impact scores assigned to news articles and Tweets can be enabled or disabled. Users can configure notification preferences by resource categories, or they can disable notifications entirely. To improve the user experience, users can submit feedback about GameEye.

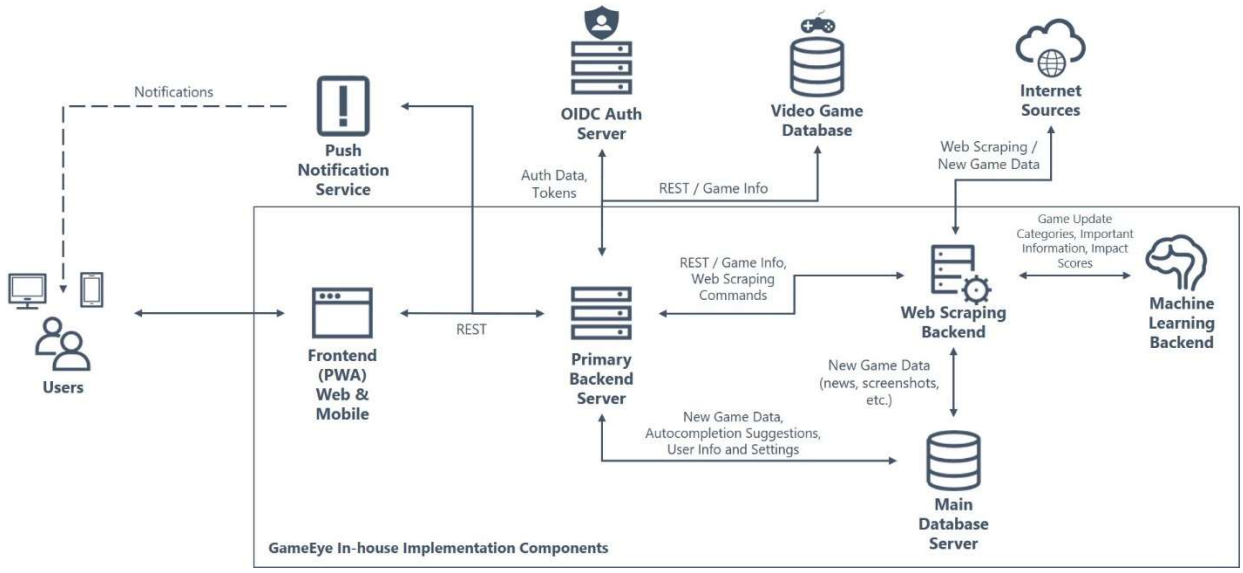
Using game data, machine learning algorithms classify game updates into meaningful categories. Game update categories include release date announcement, release date delay, major game update, and minor game update. GameEye determines the relative impact of news articles and Tweets by assigning an "impact score." The impact score measures how relevant an event, which is covered by an article or Tweet, is to a specific video game. Machine learning enables the accurate generation of impact scores for resources. GameEye also employs machine learning to extract important information from news articles and Tweets. Users receive important information and have the option to view impact scores of resources through GameEye's machine learning component.

2.2 Major Components (Hardware/Software)

GameEye's hardware components, as depicted in Figure 1, include a frontend server, a main back-end server, a web scraping back-end server, a machine learning backend server, and a main database server. The front-end server allows users to connect directly to GameEye on a desktop or mobile system. Connections between the front-end, push-notifications, main database, web scraping, and authentication service are available through the main back-end server. The web scraping back-end server is responsible for scraping the web for game data and relaying data to the main database server and machine learning back-end server.

Figure 1

Real-World Product Major Functional Component Diagram



The machine learning back-end server receives raw data from the web scraping back-end server and produces game updates, important information, and impact scores. Containing user and game data, the main database server provides the main back-end server with game data and user information. In addition, the main database server receives game data from the web scraping back-end server.

GameEye's front-end will be implemented with several technologies and tools. HTML, SASS/CSS, and TypeScript will be the front-end development languages. SASS, an extension of CSS, will allow quick development of style sheets for the front-end user interface. TypeScript, which is similar to JavaScript, will allow large-scale JavaScript applications for modern web browsers. The IDE, WebStorm, will be used for web development in the outlined programming languages. Angular Framework and Google Workbox, a set of PWA libraries, will provide a diverse array of web development technologies.

Java will be the primary programming language to implement GameEye. IntelliJ IDEA IDE will be used for Java development, and Spring Framework will be employed to ensure GameEye is secure and responsive. The jsoup library will be used for web scraping. For database integration, the MongoDB Java Driver will be used to directly interface with the main database from within Java. Data models and interfaces will be implemented to connect the main database to the backend application.

Test driven development (TDD) will be followed by using the Junit Java Framework for unit testing. For front-end testing, Jest JavaScript Framework will be employed to test JavaScript code. Unit testing will occur throughout each stage of development, and system and stress testing will occur in the final phase of development.

Python, an interpreted, high-level programming language, will be used to implement the machine learning functionality of GameEye. GameEye will rely on two primary machine learning libraries: Keras, a deep learning library, and scikit-learn library. Keras will enable fast deployment of machine learning models to compute impact scores. The machine learning library, scikit-learn, will allow GameEye to classify game data. TensorFlow Serving is used to interact with the REST API, as shown in Figure 1. Machine learning is the core of GameEye's innovation – allowing users to receive important information. Python will also be used to implement the

natural language processing (NLP) necessary to determine impact scores for articles and Tweets. The spaCy library will be used for GameEye's NLP component.

GameEye's main database will be implemented using MongoDB, a nonrelational database language and platform. GUI access will be provided through MongoDB Compass. The main database will connect to IGDB via a REST API. MongoDB enables GameEye to have a fast and responsive database. In addition, flexible database schema allows for growth in the structure of the types of data that GameEye collects. Mongo provides the ability to write and enforce data, or schema, validation rules for collections in the database. GameEye will require collections, which are equivalent to tables in relational databases, for users, games, news websites, images, and archived resources. To enable fast search auto-completion, GameEye will implement Elasticsearch, which integrates with the schema design of the MongoDB documents.

GameEye will employ two commercial, off-the-shelf, or third-party, software packages: Firebase Authentication and Firebase Cloud Messaging, which uses the Firebase Admin Java SDK. Firebase Authentication will allow GameEye to authenticate users via auth tokens, which are credentials for a login session – off-loading the security risks of storing user passwords or password hashes to a third-party service. Firebase Authentication provides a software development kit (SDK) for implementation and integration into GameEye's application. GameEye will employ Firebase Cloud Messaging to send push-notifications to users when new content is found.

3 Identification of Case Study

GameEye is designed for gamers seeking to focus on updates for games of personal interest. This target market includes experienced and novice gamers. GameEye is designed to provide any gamer with notifications for their favorite video games. Gamers would primarily use the product for game update notifications and video game tracking. Individuals actively playing

video games would be sufficient for a representative case study. GameEye would be initially tested on a small group of gamers, such as a local video game club at a university. The ODU Esports Club would be a perfect candidate for a case study.

During the case study, user activity would be monitored, and feedback would be gathered from interviews and surveys. The ODU Esports Club is interested in competitive video games. GameEye accounts can be created for each club member, and each member can populate a watchlist with competitive video games they wish to track. Notifications would be sent to the case study group members and surveys and interviews would be conducted to gauge the effectiveness of GameEye. After testing GameEye with the ODU Esports Club, the app can be deployed to a larger market. GameEye could then be distributed through video game websites, YouTube videos, Google Ads, advertisements, and Google Search. The characteristics of GameEye's target audience are not likely to change in the future.

4 GameEye Product Prototype Description

The GameEye prototype will demonstrate proof of concept by implementing the core product features. Product features such as web scraping of Reddit posts, images, and videos will be removed for the prototype. Users will have the ability to authenticate to the GameEye frontend via third-party services including Google and Microsoft. Video game data will be largely derived from actual entries in IGDB and populated into the MongoDB database. Users will be able to search for video game titles with the auto-completion feature supported by Elasticsearch. Users will be able to perform account management tasks, such as changing passwords and updating profile information.

In the prototype, each user will have a personal video game watchlist. Users will be able to configure their account settings such as visibility of archived resources, visibility of impact scores, and notification preferences. The prototype will provide web scraping capabilities in the

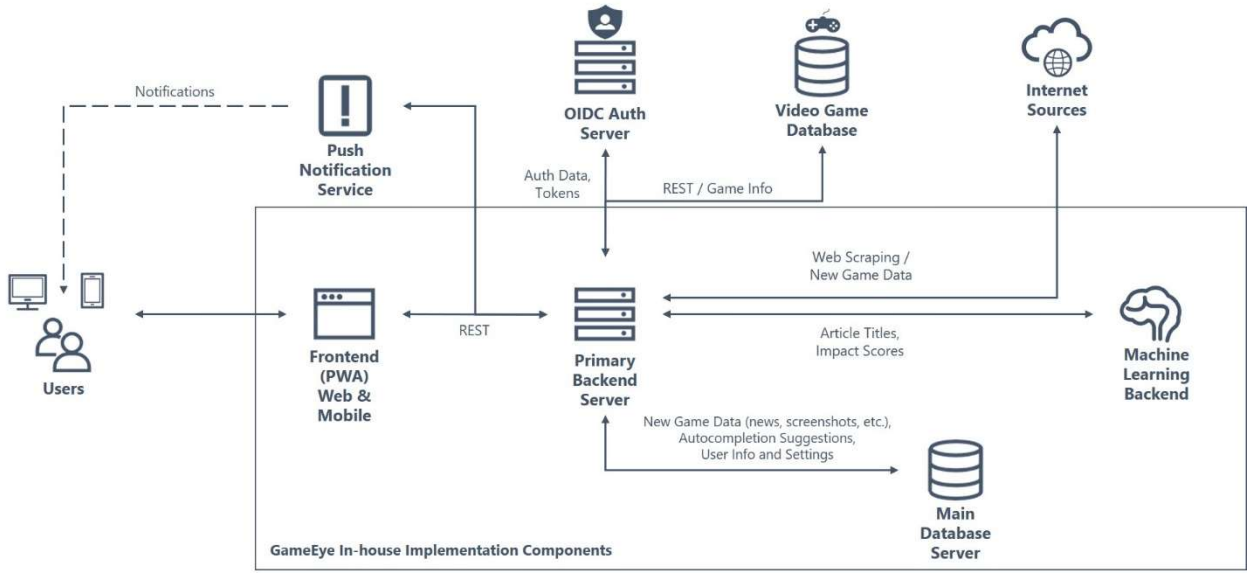
backend for obtaining news article data. Many of the challenges from the real-world product version of GameEye are mitigated by a reduced number of product features. The architecture, including the MFCD, of the GameEye prototype is still functionally equivalent to the real-world product.

4.1 Prototype Architecture (Hardware/Software)

The Spring Framework will be used to develop the backend service including the administrative shell for management of the system. Java is the primary programming language for the backend components. A Linux virtual machine on the ODU CS department network will host the MongoDB database instance. The GameEye backend service will connect to the MongoDB main database instance. The GameEye backend service will connect to the MongoDB main database via the Spring Framework. MongoDB will be managed via Java data models and interfaces and the Compass GUI application.

Figure 2

Prototype Major Functional Component Diagram



Frontend components will be implemented in the WebStorm environment with HTML, CSS, and JS. Java interfaces will be designed and implemented for web scraping. User

authentication through the “OIDC Auth Server” component will be implemented via the third-party service, Firebase Authentication. The “Machine Learning Backend” component will connect to the “Primary Backend Server” component to process game data and determine impact scores. Server replicas, which provide load balancing and redundancy, will be omitted for the purposes of the GameEye prototype.

4.2 Prototype Features and Capabilities

The prototype will implement most of the same features in the real-world product and mitigate identified risks. Offline support, local caching, and connectivity resiliency will be partially implemented in the prototype version of GameEye, as shown in Table 1. Cross-platform support for desktop and mobile systems will be functional in the prototype. The ability to provide two-factor authentication is not essential to GameEye’s core purpose and will not be implemented. Although users can create accounts in the prototype version, the ability to delete a user account will not be available.

Table 1

Table of Comparison Between RWP and Prototype: Part 1 of 2

Category	Feature	RWP	Prototype
General			
	Cross-Platform Support (Desktop, Mobile)	Full Functionality	Full Functionality
	Offline Support	Full Functionality	Partial Functionality
	Local Caching	Full Functionality	Partial Functionality
	Connectivity Interruption Resiliency	Full Functionality	Partial Functionality
Authentication			
	User Login & Registration	Full Functionality	Full Functionality
	External Provider Login & Registration	Full Functionality	Full Functionality
	Persistent Sessions	Full Functionality	Full Functionality
	Two-Factor Authentication (2FA)	Full Functionality	No Functionality
	Password Recovery	Full Functionality	Full Functionality
Account Management			
	Change Password	Full Functionality	Full Functionality
	Modify Profile Information	Full Functionality	Full Functionality
	Delete Account	Full Functionality	No Functionality
Searching			
	Search For Games	Full Functionality	Partial Functionality
	Search Autocompletion	Full Functionality	Full Functionality
Game Tracking			
	Add Games To Watchlist	Full Functionality	Full Functionality

Game tracking, which is one of GameEye's crucial components will be implemented. Users will be able to search games with auto-completion and add game titles to their personal watchlist. The sources that populate a user's watchlist will be scraped from the web. Web scraping sources partially include video game news websites and Twitter feeds for games. The important user settings such as notification preferences will be implemented and help to demonstrate that GameEye can be configured to meet a user's specific needs. The prototype will demonstrate successfully the ability to sends a user push-notifications when new updates are found for video games in their watchlist.

Table 2

Table of Comparison Between RWP and Prototype: Part 2 of 2

Category	Feature	RWP	Prototype
	News Articles (Web Scraping)	Full Functionality	Partial Functionality
	Tweets (Web Scraping)	Full Functionality	Partial 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 (Includes Website Logos)	Full Functionality	Partial Functionality
	Game Thumbnails	Full Functionality	Full Functionality
	Source Website Redirection	Full Functionality	Full Functionality
	Resource Organization	Full Functionality	Full Functionality
	Show Archived Resources	Full Functionality	Full Functionality
	Most Watched Games List	Full Functionality	Full Functionality
Settings			
	Show Archived Resources Option	Full Functionality	Full Functionality
	Show Importance Scores Option	Full Functionality	Full Functionality
	Receive Notifications	Full Functionality	Full Functionality
	Receive Notifications Per Category	Full Functionality	Full Functionality
	Submit Feedback	Full Functionality	No Functionality
Machine Learning			
	Importance Scoring	Full Functionality	Partial Functionality
	Resource Classification	Full Functionality	Partial 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

Identified risks will be mitigated in the GameEye prototype, which help to show proof of concept. The prototype will retrieve game information using web scraping interfaces in the form of RSS feeds. RSS feeds are static and mitigate the risk that a website's structure may change significantly over time. Game data from IGDB will be stored in the MongoDB main database – offering a mitigation to the possibility that IGDB experiences an outage. Partial application data caching will be implemented to address the risk that the GameEye database is unavailable to users for a brief period. User's PII (personally identifiable information) will be protected by

third-party authentication via Firebase Authentication, which uses end-to-end encryption. Each user will have access to a product manual – mitigating the risk that users cannot operation the application. The MongoDB database will be protected from NoSQL injection attacks via proper coding practices.

4.3 Prototype Development Challenges

The GameEye prototype, while offering fewer features than the real-world product, presents multiple development challenges. Design and implementation of GameEye requires sophisticated technology and frameworks, such as Angular, Spring, and MongoDB. The technologies implemented for the prototype have sufficient support from the developers and maintainers for reference during the prototype implementation. A core functionality of the prototype is establishing communication between the multiple backend components. The MongoDB database must communicate to the web scraping backend and the primary backend server as illustrated in Figure 2. Another importation communication channel exists between the web scraping backend and the primary backend server. The machine learning backend must communicate to the web scraping backend. A major priority and challenge will be establishing communications between each backend component.

Other challenges exist, such as local caching of frontend user content and resilient web scraping interface designs and implementations. Mitigating the possibility of service outages is accomplished by caching frontend user content on users' devices. Caching provides service resiliency for a short period and increases the overall performance of the GameEye frontend components. Web scraping is critical to maintaining current data in the database and user's watchlists. Designing and implementing web scraping interfaces that are reliable over a long

period of time will be challenging due to the dynamic nature of the Internet. The prototype design will confirm the resiliency of web scraping implementations via proper testing and validation.

5 Glossary

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.

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

CSS: Cascading Style Sheets; used to stylize webpages.

Elasticsearch: Search and analytics engine that integrates with schema-less (e.g., MongoDB) database systems. Based on the Lucene library and provides an HTTP web interface.

Firebase Authentication: Backend authentication services and software development kits (SDKs) for integrating authentication into applications.

Google Workbox: A set of JavaScript libraries for offline support in PWA.

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.

IDE (Integrated Developer Environment): Application providing a set of tools and resources for programmers. Includes source code editors, automation tools, compilers, and debuggers.

IGDB (Internet Games Database): Database of known video games: accessed by REST API to populate GameEye's database.

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

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

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

JavaScript: Object-oriented language used to create dynamic, interactive effects on webpages.

Jest JavaScript Framework: Testing framework maintained by Facebook Inc.

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

JUnit Java Framework: A testing framework for Java.

Keras (Python Deep Learning Library): Open-source neural-network library written in Python.

MongoDB: A cross-platform document-oriented database program

Noise Filtering: The act of removing data that is determined to be "noise" (i.e., subjectively irrelevant data) to suit an individual's content preferences.

OIDC Authentication: Authentication protocol based on the OAuth2.0 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.

Python: Interpreted, high-level, general-purpose programming language.

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

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

SASS/CSS (Syntactically Awesome Style Sheets/Cascading Style Sheets): an extension of CSS used for more robust development of style sheets for web design.

Scikit-learn Library: Software machine learning library for the Python programming language.

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

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

Tester: GameEye beta testers; users of the application in its prototype phase who will provide feedback on their experience.

TypeScript: An open-source language which augments the capabilities of JavaScript. The language adds static type definitions and is compiled into JavaScript via the TypeScript compiler or Babel compiler.

Web Scraping: Data scraping for extracting data from websites.

WebStorm: An IDE developed by JetBrains to write JavaScript code.

6 References

Anderton, K. (2019, June 26). Video game market share and growth. [Infographic]. *Forbes.* https://www.forbes.com/sites/kevinanderton/2019/06/26/the-business-of-video-games-market-share-for-gaming-platforms-in-2019-infographic/#600d6dd97b25

Dietz, J. (2011, June 23). *30 Games that emerged from development hell.* Metacritic. https://www.metacritic.com/feature/games-that-shed-vaporware-status

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

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

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

Humphries, M. (2019, September 18). Twitch acquires gaming database website IGDB. *PCMag.* 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.* Gamasutra. https://www.gamasutra.com/view/news/217583/How_the_surge_of_Steam_releases_will_affect_game_developers.php