Lab 1 – GameEye Product Description

Adrian Wilson

Old Dominion University

CS411W

Professor Brunelle

4 October 2020

Final Version 2

Table of Contents

List of Figures

List of Tables

1 Introduction

With nearly one third of the world actively gaming, the future of the industry is bright. The gaming industry is thriving and consistently expanding. The industry will soon be worth over 200 billion dollars (Gough). The large community and potential profits have provided many opportunities for developers to succeed. There are a multitude of developers, as well as platforms for which games can be produced; from designated gaming consoles like Sony's PS4 or Nintendo's DS, to PC's and mobile phones. As a result, thousands of games are released every year (Gough). On a game distributer for PC called Steam, 9050 games were published in 2018 (Gough). The large number of published games is not exclusive to PCs, the Google Play store, which is only available on android devices, saw an increase of nearly 100,000 in the year 2017(Gough).

While games developed for the web, and mobile can be completed in a relatively short amount of time, that is not always the case for console. Years of development can be spent on games that are developed mainly for consoles and the PC. To build anticipation, publishers routinely announce upcoming project and release brief details; this can happen long before the release date. In the years in between, gamers may struggle to keep up with all of the information released on a game that interest them.

If the news about a pending release dwindles, gamers may forget completely. This can negatively affect small independent developers. Without the marketing dollars, they can struggle to maintain interest when they have to fund their development of games. As if the amount of information before the release was not enough, more often than not, there is more after the release. This is because of the countless: updates, bug-fixes, and other news-worthy changes that are made to a game. The amount of information is not the only problem there are also too many

sources. Sites such as Twitter, Facebook, and Reddit which has subreddits devoted to specific games, all could contain relevant information; not to mention the news sites which are specifically devoted to gaming, i.e. Game Informer. The spread of information across many mediums poses an additional challenge to gamers. When time and effort are the focal point in gathering gaming information, the tools and methods currently available do not suffice.

The solution is GameEye, an aggregator for gaming news. Users shall be able to search for and track their favorite games. After searching for the game, users have the option to add it to their personal watchlist. When news is released, GameEye shall alert the user. GameEye shall collect and analyze the data it gathers. With the use of machine learning, articles and tweets are categorized based on their subject matter. The articles and tweets shall be given a score on their importance, judged by a multitude of factors.

2 Game Eye Product Description

GameEye is a cross-platform web application available for desktop and mobile. It scours the web for gaming news, organizing and consolidating it for the user. The news is tailored to their gaming interests and is displayed as part of six categories: important updates, news articles, tweets, Reddit posts, images, and videos. After scouring the web for new game content, GameEye uses artificial intelligence to categorize what it found.

2.1 Key Product Features and Capabilities

By developing GameEye using PWA (Progressive Webs App) technology, cross-platform use across desktop and mobile is achieved. This allows for offline support and a local cache allows game information to be displayed when the user is offline. GameEye uses authentication architectures to provide the user with a secure experience registering and logging in. An account used on external sites can be used to register as well.

A user's session shall persist after exiting the app, so that repeated logins are not necessary. Two-factor authentication ensures that user logins are genuine. And in the event that a user forgets their password, a method of recovery is available. After creating an account, users have the ability to change their password and profile details. In the event they want to delete their account, that feature is available. Each user shall have a personal watch list consisting of games they selected to follow. Games currently being followed shall be displayed on the watchlist tab along with a thumbnail of the game. Game Eye shall monitor the games on the watchlist classifying new content it retrieves by important updates, news articles, tweets, Reddit posts, images, and videos. Thumbnails for news articles and tweets shall be included. In an effort to introduce new games into a user's watchlist, GameEye shall display a list of the most-watched games on their desired platform.

In order to fill their watchlist, the user is able to search for video game titles. The search bar is located in the Watchlist tab and has autocompletion. Aside from saving some time, users shall be able to find games easier if they are unsure of its name. To provide relevant news to the user, GameEye uses Web Scraping. The data gathered shall be classified into one of five categories: news articles, Reddit post from designated subreddits, tweets from trustworthy Twitter feeds, YouTube videos from designated game accounts, and images. Push notifications are available for both mobile and desktop to alert users about game updates. Users shall be able to see the number of notifications for a certain category, i.e. tweets or news articles. GameEye also suggests notifying users of game updates on the most-watched list. GameEye allows users to choose whether or not they want to: see archived resources, see importance scores of articles or tweets, and if they want notifications at all. The user also has the option to submit feedback.

To automate the resource collection, GameEye uses machine learning in conjunction with web scraping. The interface is provided with labeled data it can learn from. This allows for the data collected from web scraping to be classified as: release date announcement, delay, major game update, minor game update, etc. In addition, the machine learning interface shall be responsible for extracting information out of articles and tweets, as well as scoring its importance.

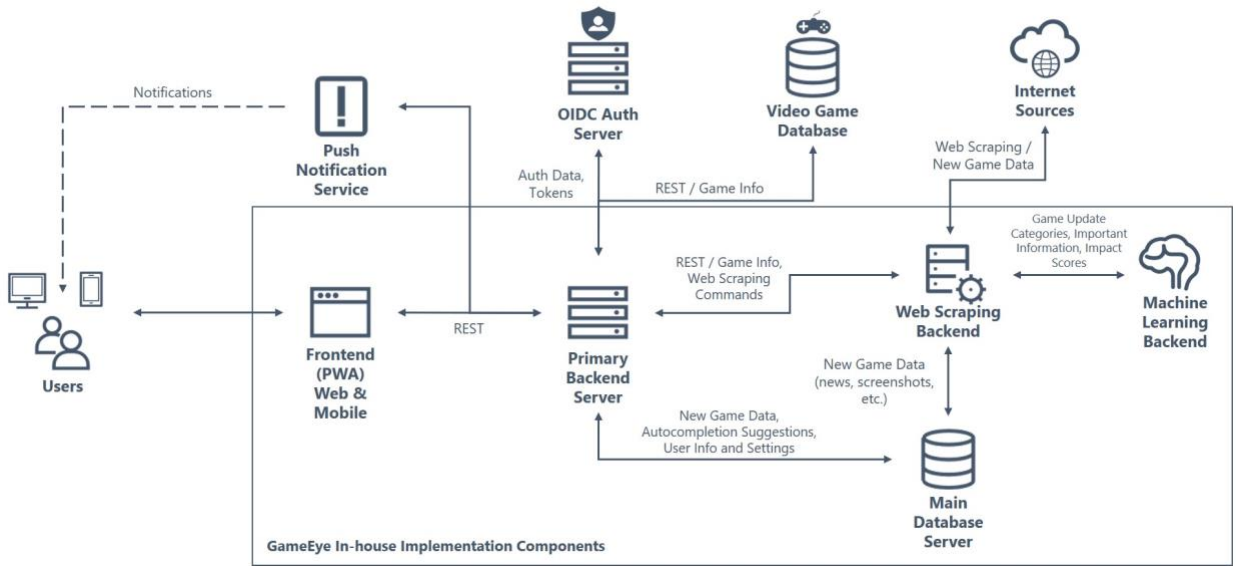
2.2 Major Components (Hardware/Software)

GameEye is comprised of five hardware components: frontend sever, main backend server, webscraping backend server, machine learning backend server, and the main database server. The frontend sever is responsible for what is displayed to the user on the mobile/desktop application. The main backend uses REST API to manage interactions between the other servers. Works with the web scraping backend to gather necessary for the frontend to be meaningful The webscraping backend server scours the web with operations run by REST API. It provides the data for the machine learning backend. The machine learning backend uses a REST API so that a machine learning interface can be run. Works with the webscraping backend so that it stores the correct information. The main database server is provided by MongoDB and it contains all of the application data. The major functional component diagram shown in the below shows the interaction and flow of data between components.

[This space is left blank intentionally]

Figure 1

Major Functional Component Diagram 1



WebStorm is the developmental environment that will be used to program the frontend. Angular framework allows GameEye to be developed efficiently as a single-paged application. Angular incorporates: HTML, CSS/SASS, and TypeScript. Google Workbox PWA libraries will be used provide offline support. IntelliJ IDEA is the development environment that will be used to program the backend. Spring framework will be used to build the backend Java application. Jsoup is an HTML parser will be used to webscrape. MongoDB Java driver allows for communication with the MongoDB sever.

Jest JavaScript Framework will be used to test the front end. Junit Java Framework will be used to test on the backend. Python shall be used to implement the machine learning the GameEye requires, along with the help of a few libraries. TensorFlow, Keras, and scikit-learn are the machine learning libraries that shall be used to accomplish machine learning. The text

gathered from articles, tweets, and etc. has to translated into usable data. To do so, GameEye shall use Python along with spaCy library. To store data, GameEye shall use the MongoDB database along with its Compass GUI. To retrieve game titles, GameEye shall use the IGDB REST API. The only third-party software GameEye shall use is Firebase Cloud Messaging, allowing notifications to be sent to users. MFCD figure 1 details the major functional components present in the GameEye prototype, and the flow of data in between them.

3 Identification of Case Study

GameEye targets customers who want to stay up to date with information about the games that interest them. Gamers come in all shapes and sizes; college students happen to account for a significant part of the community. ODU for example, has Esports teams for specific games like Fortnite and CS: GO. Not to mention, before COVID-19 many college gamers would gather at the center of the Webb Center to game with each other. This presents an opportunity for a case study.

A small group of ODU students who are affiliated with an Esports team, or that just have a passion for gaming, shall be given access to the prototype. They can create an account through the web or mobile and test out the account management features. Most importantly they shall be able to search for and track game information. Once a game is added to their personal watchlist, they shall be able to view information on that game and be notified when new information is published on the web. These students, as well as others on the ODU campus, will also serve as the initial user base. In the future GameEye shall be marketed to gamers all over, as it is going to be available on the Google Play Store, Apple App Store, and the web.

4 Game Product Prototype Description

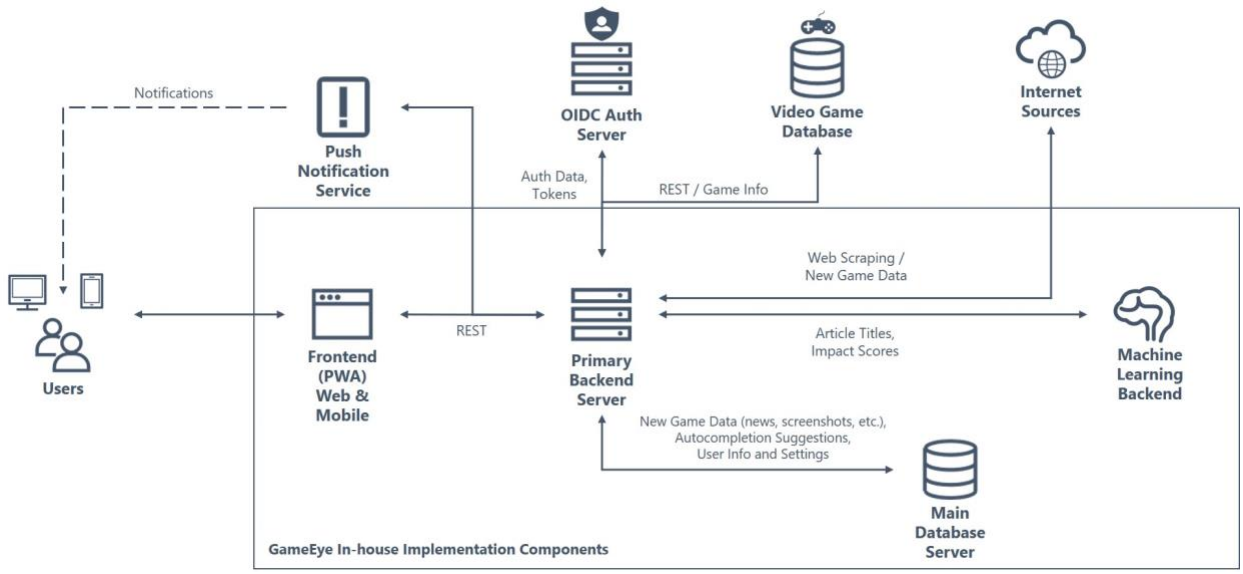
The GameEye prototype is functional but limited in features compared to the RWP; it shall still provide an adequate experience for the user. They shall be able to use GameEye for its intended purpose, which is to track games on their personal watchlist. Notifications shall be sent to users when new information arises about a game they follow. Through machine learning, GameEye shall categorize and weight the importance of the information it gathers. The prototype shall also provide users with basic account management features such as modifying profile information.

4.1 Prototype Architecture (Hardware/Software)

The prototype will utilize the same architecture as the real-world product except for the replicated servers for load balancing that would be present in the real-world product.

Figure 2

Major Functional Component Diagram 2



The prototype MFCD pictured above shows there are some differences compared to the RWP MFCD. The biggest change is the removal of the web scraping backend.

4.2 Prototype Features and Capabilities

The features available in the prototype shall differ from those in the RWP. Due to the time constraints, some features in available in the RWP shall be partially or non-functional in the prototype. The three tables below list those differences in the functionality of certain features.

Table 1

RWP vs Prototype 1

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

[This section is left blank intentionally]

Table 2

RWP vs Prototype 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

Table 3

RWP vs Prototype 3

Category	Feature	RWP	Prototype
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

4.3 Prototype Development Challenges

The first challenge is learning the language and frameworks necessary to develop the application and decide how they will be incorporated. Teaching the machine learning models will also be a challenge, because a sufficient amount of data is needed to ensure their accuracy. Since the backend has multiple servers, getting them to work together will require effort. To provide connectivity interruption resiliency, a cache must be implemented for data on the frontend. Webscraping needs to be flexible, as not to cause errors if a site changes in the future.

[This space is left blank intentionally]

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® subsidiary that provides on-demand cloud computing platforms 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.

IGDB: Database of known video games, accessed by 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.

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: Information/news articles shown that caters to an individual's content preferences.

OIDC Authentication: Authentication protocol based on the OAuth2.0 family of specifications.

PWA: Progressive Web Application; 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: Software architectural style used in creating web services.

RSS Feed: Web feed that allows users and applications to access updates to websites in a standardized, computer-readable format.

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.

Web Scraping: Data scraping for extracting data from websites.

WebStorm: IDE developed by JetBrains to write JavaScript code.

[This space is left blank intentionally]

6 References

Anderton, K. (2019, June 26). The Business Of Video Games: Market Share For Gaming Platforms in 2019 [Infographic]. 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. https://www.metacritic.com/feature/games-that-shed-vaporware-status

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

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

Gough, C. (2019, October 9). Google Play: Number of available games by quarter 2019. 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. 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. https://www.gamasutra.com/view/news/217583/How_the_surge_of_Steam_releases_will_affect_game_developers.php