Lab 2 - GameEye Product Specification Outline

Angelos Angelopoulos

Old Dominion University

CS 411W: Professional Workforce Development II

Professor Janet Brunelle

November 20, 2020

Version 2

Table of Contents

List of Figures

List of Tables

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). Thousands of video games are released every year (Anderton, 2019). Just in 2018, more than 9,000 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). Gamers may desire to play many upcoming games but staying up to date for all of them can be time-consuming. Simultaneously, video game developers spend millions on marketing. Indie developers face an increased challenge in maintaining the public spotlight as the marketing budgets of large video game developers overshadow theirs. Another issue is that information about video games is decentralized, which makes it more difficult to find news.

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 have a personal watchlist of video games. GameEye keeps track of followed games and notifies users when new content is released online about them. The application collects and categorizes video game data from the internet, including news articles, tweets, Reddit posts, images, and YouTube videos. Using machine learning, news articles and tweets are classified depending on their content, and a multi-factor impact score is 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.

1.1. Purpose

GameEye is a product with the purpose of organizing 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.

GameEye is a simple and intuitive solution. The user simply enters a few games they are interested in, and the application keeps track of them and notifies the user of any new information. The application is being developed for casual and highly active gamers alike as both types of end-users benefit from it. 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.

GameEye seeks to complement, not replace, current news sources and other media by aggregating and processing information from many different sources. Information from multiple media is scraped regularly every day and users are notified about relevant information. Links to navigate to the sources of information are always provided so that users can read from the source websites. In this way, information sources such as video game news websites benefit from the increased traffic GameEye brings to them. The application does not scrape data in real-time, nor allows users to have unlimited games in their watchlist, as both have high operational expenses.

1.2. Scope

GameEye is a smart content and news aggregator that uses machine learning to deliver relevant content to the user. It saves time by providing a central location for video game news and by notifying users about important developments. 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 by performing tasks and analysis that the gamer would otherwise have to do manually.

The prototype has a subset of the features of the real-world product. This subset is meant to demonstrate the applicability and benefits of the application. The following features are implemented in the prototype: (a) news article scraping, (b) machine learning importance scoring, (c) game title autocompletion, (d) user authentication and authorization, (e) user watchlists, (f) push notifications, (g) cross-platform support, (h) most-watched games list, and (i) game logos. A limited amount of games that mainly includes highly popular games is available to watch for demonstration purposes. Furthermore, to facilitate testing and demonstration, a mock news website is implemented. This website displays a list of articles to be scraped by GameEye and provides a simple content management system (CMS) to add, edit, and remove articles.

[This space is intentionally left blank]

1.3. Definitions, Acronyms, and Abbreviations

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.

Cheroot: A high-performance Python HTTP server that conforms to the WSGI specification.

CSS: Cascading Style Sheets; used to stylize webpages.

CMS: Content Management System. An application used to manage web content, allowing authors to create, edit, and remove content displayed on a website.

Extremely Randomized Trees (ExtraTrees): A type of Random Forest machine learning model. The key differences with Random Forest are that each decision tree in an ExtraTrees model is trained on the whole training dataset, and that top-down splitting is randomized. ExtraTrees is found to generalize better than Random Forests.

Flask: A micro web framework written in Python.

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 Framework: A testing framework for Java.

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

Netlify: A platform for hosting serverless abckend services for web applications and static websites.

Netlify CMS: An open-source content management system for static websites.

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.

Random Forest: A type of machine learning model that is an ensemble of decision trees. Each decision tree is trained independently, and in the end, their predictions are averaged to produce a final result.

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.

Single-page Application: A web application that interacts with the user by dynamically rewriting the current web page with new data from a web server instead of reloading new pages.

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

Spring Boot: 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.

Tf-idf: Short for term frequency-inverse document frequency. Tf-idf is a statistical method for computing the importance of a word for a document by also accounting for its overall frequency in a collection of documents.

Web Scraping: Automated extraction of data from websites.

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

WSGI: Short for Web Server Gateway Interface. WSGI is a calling convention for web servers to forward requests for Python web applications or frameworks.

[This space is intentionally left blank]

1.4. 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

Angelopoulos, A., Cook, J., Diasanta, C., Epps, J., Hund, B., Lewis, B., Wilson, A. (2020, October 4). *Lab 1 – GameEye Product Description Outline*. Old Dominion University. Retrieved from: https://www.cs.odu.edu/~411yello/pages/labs.html

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

[This space is intentionally left blank]

1.5. Overview

This product specification details the software and hardware components, features, development challenges, and external interfaces of the GameEye prototype. The information provided in the remaining sections of this document gives a detailed description of (a) the software and hardware used for implementing the prototype, (b) the architecture of the prototype, (c) the features that the prototype will provide including any limitations, (d) the testing methods and challenges of developing the prototype, and (e) the interfaces the prototype has with other software. The functional requirements of the prototype are enumerated and detailed in a separate document titled “Lab 2 Section 3 – GameEye Product Requirements”.

2. General 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 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. A subset of these functionalities is implemented in the prototype.

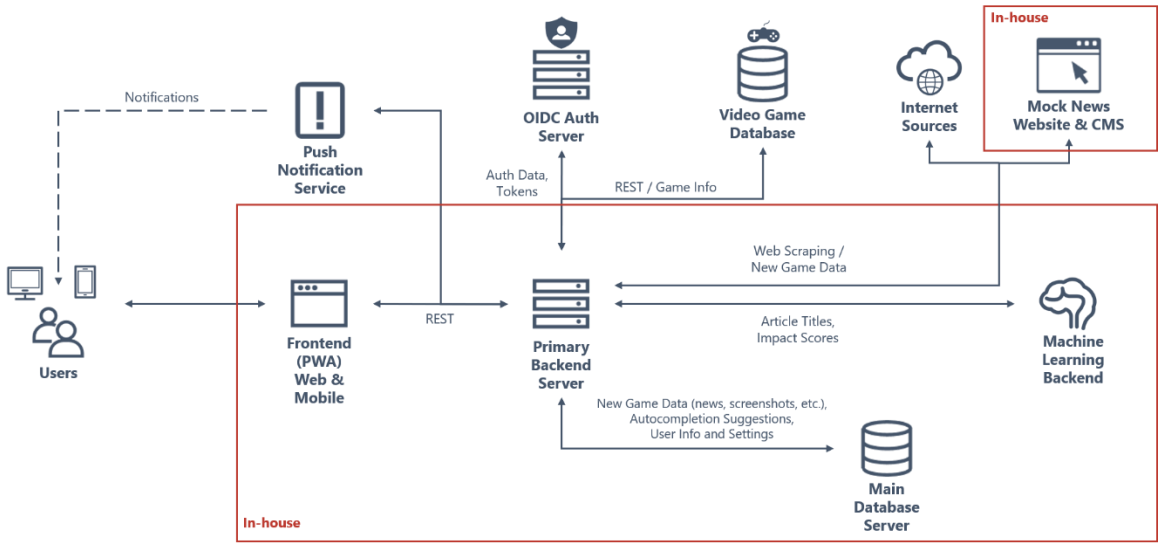
2.1. Prototype Architecture Description

The prototype for GameEye consists of several in-house and third-party components. The in-house components are: (a) the frontend, (b) the primary backend, (c) the web scraping backend, (d) the machine learning backend, (e) the main database server, and (f) the mock news website with a CMS for testing purposes. 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. The components and their interactions are illustrated in Figure 1.

Figure 1

Major Functional Components of the GameEye prototype.



All in-house components are hosted on a virtual machine provided by the ODU Computer Science department. The virtual machine is protected by the CS department's firewall and can only be accessed from within the network or by using a VPN. All servers hosted on the VM, including the frontend, backend, and databases, listening on separate ports. All the servers are secured and require some form of authentication to access.

The GameEye frontend is a single-page web application built with, HTML, SCSS, and TypeScript. It serves as the user interface for GameEye. As a web application, it is cross-platform and targets desktop and mobile devices. The frontend communicates with the primary backend to retrieve data. The primary backend also coordinates and schedules web scraping. It

provides a REST API for communication with the frontend, web scraping backend, databases, and other third-party services. The primary backend is built with the Spring Boot framework and Java.

Web scraping is performed by the primary backend in the prototype, but in the real-world product, it would be carried out by a cluster of computers. The primary backend orchestrates web scraping by launching the operations twice a day and processing scraped data before storing it in the database. A REST API is provided for controlling web scrapers, such as forcing web scraping outside of scheduled times, if necessary. The library used for web scraping is jsoup.

Machine learning is implemented as a dedicated backend. It communicates with the primary backend web scrapers to process new data. This backend is responsible for determining article importance. It is built with Python, scikit-learn, spaCy, Flask, and cheroot. The machine learning backend is a Flask application served with the cheroot HTTP server and provides a REST API for machine learning inference.

GameEye has extensive data storage requirements and would normally require a dedicated database server. For the prototype, the two databases used by GameEye are hosted on the same server together with the two backends and the frontend. The first database is a MongoDB instance 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. The second database is ElasticSearch, and it stores game titles. More importantly, it maintains complex data structures for efficient searching. Because of this, the ElasticSearch database is used to provide auto-completion suggestions for game titles. Both databases are accessed by the primary backend.

To facilitate testing, a mock news website is used as a data source by the GameEye web scrapers. The mock news website is designed to mimic a simple news website and displays a list of news articles. A content management system (CMS) is implemented to allow developers and testers to login with their GitHub accounts, and add, edit, or remove articles. This website allows the developers to more readily test GameEye by not only relying on official news websites to retrieve news articles. The mock news website is hosted on Netlify and utilizes the Netlify CMS.

The prototype implementation also makes use of a few third-party services: (a) IGDB, (b) Firebase Authentication, and (c) Firebase Cloud Messaging. IGDB is a large video game database that GameEye uses to obtain video game metadata. Firebase Authentication is used to implement authentication and authorization mechanisms. Finally, Firebase Cloud Messaging is used to send push-notifications to users.

2.2. Prototype Functional Description

The prototype provides a subset of features of the real-world product, as seen in Table 1. A full-fledged authentication system is implemented, allowing users to create accounts and login with an email and password or by using their Google or Microsoft accounts. Persistent sessions are implemented so that users do not have to login again after logging in once. The prototype also provides essential account management features, such as the ability for users to change information like their name, their email, and their password.

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
	Remove Games from Watchlist	Full Functionality	Full 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
Testing			
	Mock News Website	Full Functionality	Full Functionality
	Creation, Editing, and Removal of Articles from the Mock News Website (CMS)	Full Functionality	Full Functionality
	Web Scraping of the Mock News Website	Full Functionality	Full Functionality

The GameEye prototype includes functional watchlists, allowing users to add and remove games. Users can add games by searching for their title, and robust auto-completion suggestions are provided to make finding games easier. A most-watched games list is implemented, which displays the top 50 most-watched games on the platform. This provides a great resource for users to find interesting games. Regarding web scraping, the prototype only scrapes news articles. Articles are scraped from various popular video game news websites, including IGN, GameSpot, PCGamer, Eurogamer, and the mock news website. Scraped article titles are displayed for each game, together with how long ago they were published, a logo of their source website, and an indication if they are deemed important by the machine learning model. Users can click article titles to be redirected to their source websites to read the full articles.

Machine learning is a very important feature of GameEye. However, due to the complexity of the machine learning problems, the prototype implements a simplified version of impact

scoring that can achieve relatively high accuracy with a small training dataset. The prototype implements logic to determine whether an article is important or not using machine learning. The machine learning implements an Extremely Randomized Trees model trained on the term frequency-inverse document frequency (tf-idf) vectors of more than 3,000 preprocessed news articles dating from 2020 to 2012. The preprocessing of news articles includes lowercase conversion, tokenization, stop-word removal, digit removal, punctuation removal, and lemmatization to reduce noise and improve accuracy.

Cross-platform web push-notifications are implemented in the prototype. Whenever GameEye scrapes new articles, users that watch the related games are immediately notified on their computers and mobile devices. Settings are provided to users so that they can choose whether to receive notifications, and if so, whether they only want to be notified for articles deemed important by the machine learning model. The number of new notifications is displayed for each game in a user's watchlist.

An important component of the prototype is the mock news website. This website is used by GameEye as an additional news source for testing and demonstration purposes. Developers can create, edit, and remove articles using a CMS provided within the website. The mock news website allows the developers to readily create news articles for testing GameEye.

[This space is intentionally left blank]

2.3. Prototype Development Challenges

GameEye is a complex application that involves a large amount of data. The time for developing the prototype is constrained to a single semester, which makes it challenging to fully implement the desired features. The technologies used by GameEye are not simple and require some learning from developers to be able to participate in the development of the prototype. Developers are required to at least learn about Spring and Angular, both of which are large software frameworks.

A major challenge is the implementation of the machine learning component. The GameEye prototype tries to solve a multiplex machine learning problem that involves the analysis of news article titles to deduce the importance of the news they report. As a result, a great amount of training data is needed to achieve high accuracy. Creating a quality dataset for training and testing in such a constrained timeframe is difficult. While the collection of news articles is straightforward by using web scraping, labeling the news articles is very time-consuming. As a result, the prototype must implement a subset of machine learning features that can function reasonably well with a small dataset.

The GameEye prototype is a cross-platform web application, which leads to various challenges. The frontend is aimed to be usable by desktop computers, laptops, tablets, and mobile devices, all of which have different operating systems, browsers, and screen sizes. This means that the frontend UI must be functional, responsive, and look reasonably good regardless of the device. Thorough testing is required to ensure functionality on a variety of platforms.

The prototype also utilizes a large amount of video game data. All this data is retrieved from the IGDB database. A portion of IGDB needs to be replicated, which is challenging as the REST API is extensive and the database contains a large amount of data. Request limits make

the replication of IGDB more time-consuming, which requires the implementation of an efficient method for retrieving data for thousands of video games in a reasonable amount of time.

2.4. External Interfaces

This section identifies the physical, logical, and user interfaces used by the GameEye prototype. The characteristics of each interface are detailed.

2.3.1. Hardware Interfaces

The prototype does not make use of any specialized hardware interfaces. All components, except the mock news website, are hosted inside the ODU CS network and utilize its computer and networking infrastructure.

2.3.2. Software Interfaces

The prototype makes use of several software libraries. Angular is the framework used for implementing the frontend, while Spring Boot is used for implementing the backend. Machine learning is implemented using Scikit-learn, and the impact score model is served using Flask and Cheroot. The SpaCy library is used for text processing. Web scraping is implemented using the jsoup library, which is used for parsing HTML. The MongoDB database is accessed using a specialized MongoDB driver for Spring Boot. Similarly, the ElasticSearch database is also accessed using a specialized driver.

2.3.3. User Interfaces

The GameEye frontend is a web application. An internet connection is required for accessing it while a mouse and keyboard or a touchscreen are required to interact with it. The frontend is displayable on most screen sizes ranging from desktops to mobile devices. The

frontend officially supports popular web browsers such as Google Chrome, Mozilla Firefox, Microsoft Edge, Opera, Brave, and Apple Safari.

2.3.4. Communications Protocols and Interfaces

GameEye utilizes standard TCP/IP, HTTP, and REST for web communication. SSL/TLS is used for securing communication with Firebase services. OAuth 2.0 and OpenID Connect are the protocols used for authentication and integration with third-party authentication services, such as Google and Microsoft. Communication with MongoDB is done using a regular TCP/IP socket, while communication with ElasticSearch is carried out via its specialized Transport Client.

3. Product Requirements

The requirements for the GameEye prototype are included in a separate document titled “Lab 2 Section 3 – GameEye Product Requirements”. It includes the key functional requirements of the product, the necessary illustrations to expand on the concepts, as well as assumptions and constraints for the implementation.