GameEye: The Gamer’s Eye Into the Gaming Industry

Brenden Lewis

Old Dominion University

CS411W

Professor Janet Brunelle

Team Yellow

Version 1

Table of Contents

List of Figures

1 Introduction

1.1 Societal Problem

Given the scale of the web and how far information can be spread, it can be difficult for avid gamers to follow their favorite games. Thousands of games are released annually (Gough, 2019a; Gough, 2019b), with some of those games spending up to 7-9 years in development (Dietz, 2011). Information regarding game development is often decentralized and widespread across many different news sites, developer blogs, streaming platforms, and social media. This spread of information can make it both difficult and time-consuming for gamers to find new information for games in development.

Figure 1

Current Process Flow

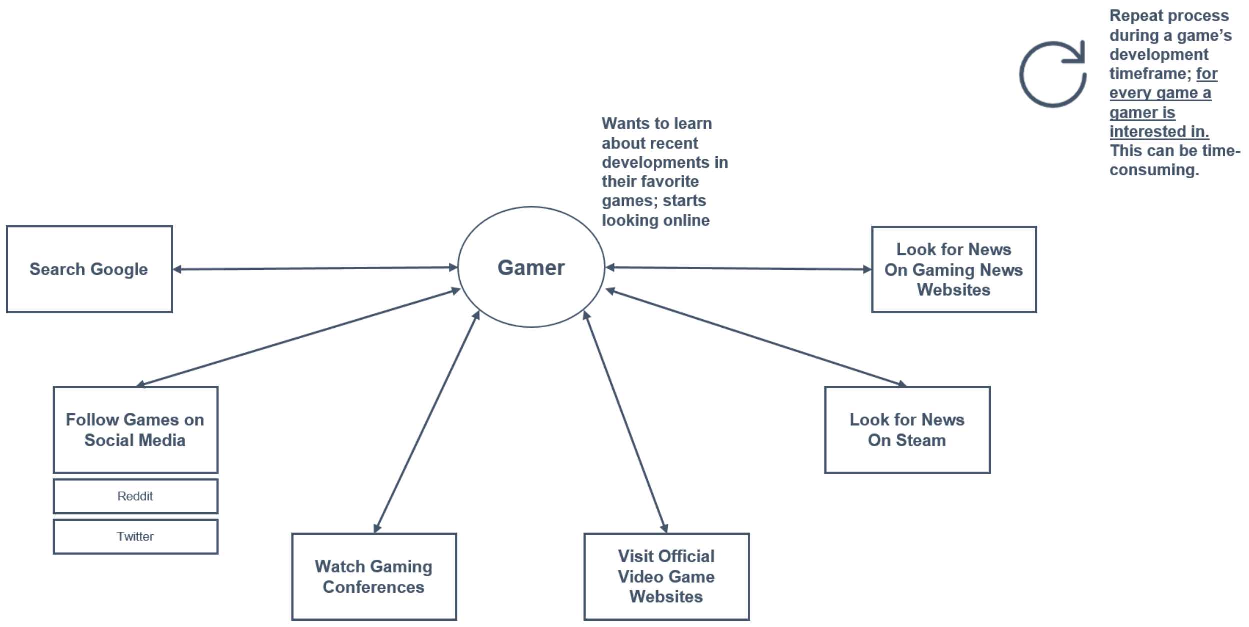

Repeat process during a game’s development timeframe; for every game a gamer is interested in. This can be time-consuming.
Wants to learn about recent developments in their favorite games; starts looking online
Gamer
Search Google
Look for News On Gaming News Websites
Follow Games on Social Media
Reddit
Twitter
Look for News On Steam
Watch Gaming Conferences
Visit Official Video Game Websites

Figure 1 shows the current process flow of a gamer looking for information about an upcoming game. To stay up-to-date with the game’s development prior to release, the gamer will have to actively and regularly check each of these different sources in case any new information is released; this is an active process that involves dealing with a lot of “noise” both before and after a game is released, and can be applicable to multiple games at once. This kind of routine can be both time-consuming and exhausting, which can lead to gamers doing less news-searching and risk missing out on new information. As a result, gamers can often forget about the games they would have enjoyed playing. This poses a huge challenge for small, independent (indie) developers to keep attention on their games, both released and upcoming, as they become overshadowed by larger, more popular triple-A (AAA) franchises.

1.2 Solution Description

In order to alleviate having to keep track of all this information, gamers need a singular location to find all the news for their favorite games; they need a platform that is accessible, customizable, and capable of notifying them when new news is released while also filtering out any unwanted information. This will keep gamers up-to-date with the games they follow and help prevent any loss in retention for smaller games that struggle with player populations.

The solution is GameEye, a progressive web application (PWA) for computers and mobile devices that allows user to craft a personal watch-list of their favorite games and receive notifications when any news or updates for those games are released. GameEye utilizes web scraping to find news for games online and links the user directly to the sources. Machine learning is also utilized to classify these new sources using a multi-factor scoring system to determine the importance of news updates for use in filtering unimportant content and notification customization. Users will be able to easily search for and add games, both current and unreleased, to their personal, customizable watch-list.

GameEye is intended for those who play video games either as serious hobby or casually. These gamers will want to stay informed about their favourite games throughout their development cycles; they will be updated and notified about new information as it is released.

Small indie developers can be considered an indirect customer as they benefit from their games gaining more exposure and player retention over a longer period of time as they release new games or updates to existing titles.

2 GameEye Product Description

2.2 Key Product Features and Capabilities

GameEye is able to scour the internet to find news about specified games and consolidate it in one location for convenient viewing.

Figure 2

Solution Process Flow

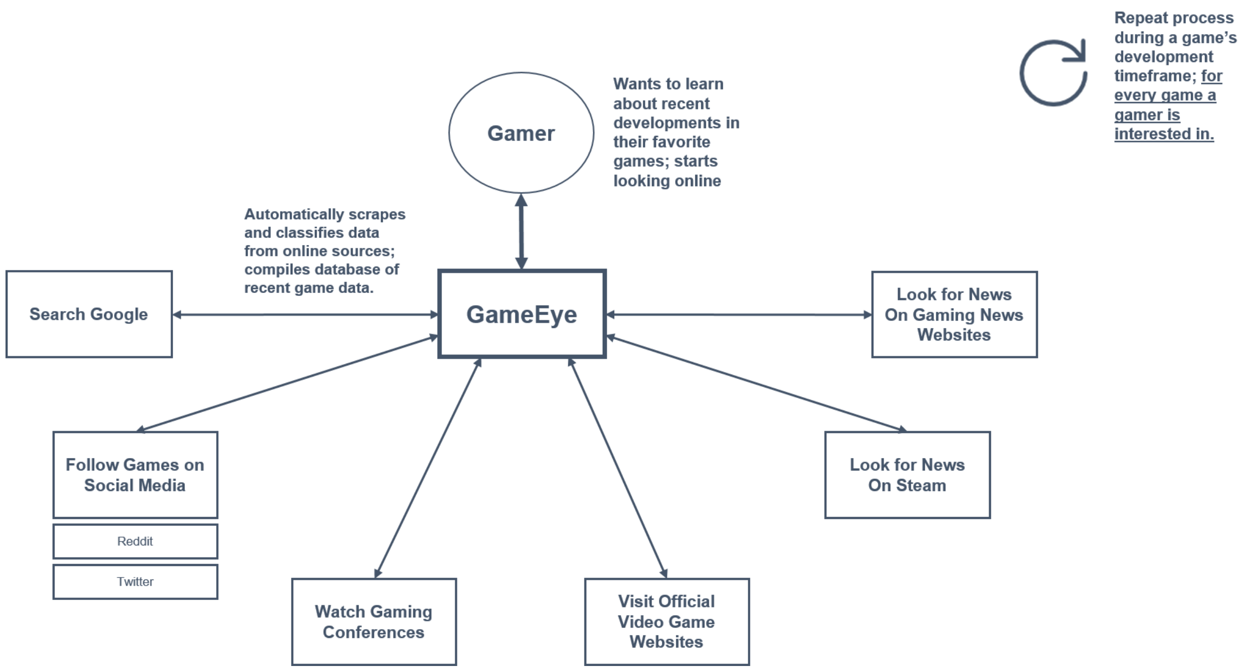
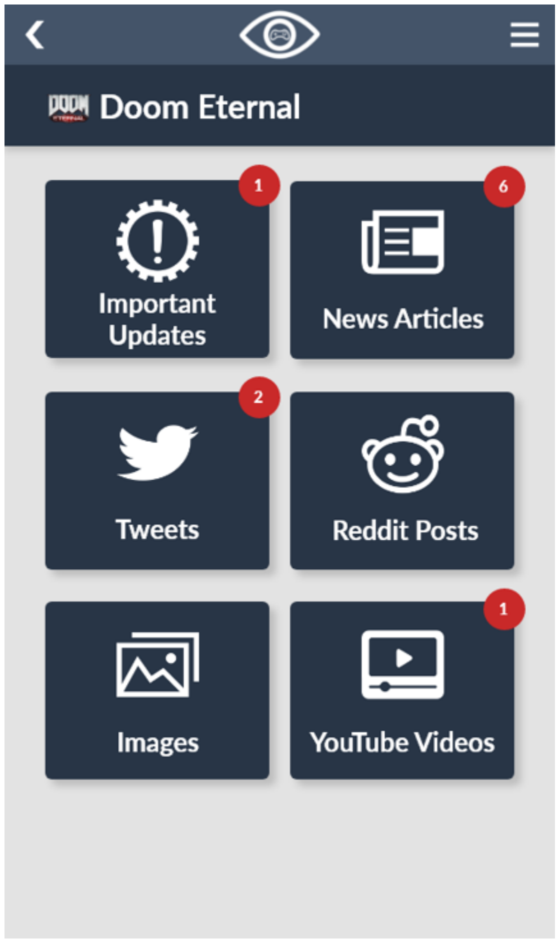

Repeat process during a game's development timeframe; for every game a gamer is interested in.
Gamer
Wants to learn about recent developments in their favorite games; starts looking online
Automatically scrapes and classifies data from online sources; compiles database of recent game data.
Search Google
GameEye
Look for News On Gaming News Websites
Follow Games on Social Media
Reddit
Twitter
Look for News On Steam
Watch Gaming Conferences
Visit Official Video Game Websites

Figure 2 represents the updated process flow with the use of GameEye; instead of the gamer having to constantly navigate between news sources to find information, GameEye handles the same workload so the gamer can view the information in one location. This cuts out both the time and effort needed to stay up-to-date with a game's development.

The user is able to build their own watch-list of games; these can be games that are either already released or still in development. Each game in a user's watch-list contains categorized information as represented in Figure 3. When a new piece of information is found, it is given a label depending on source information and a notification is sent to the user.

Figure 3

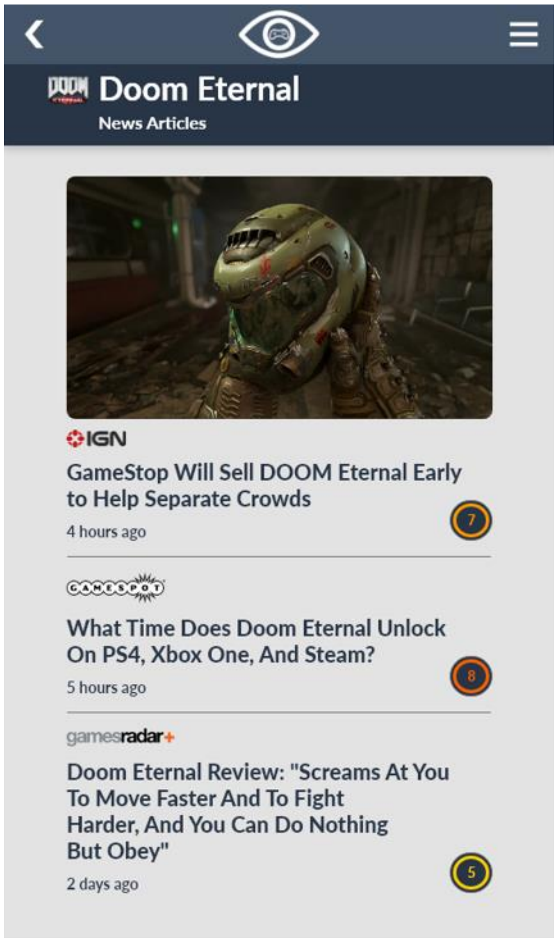
Information Categories Mock-Up



The categories of posts represented in Figure 3 include important updates such as changes in release dates or new game updates, news articles pertaining to the game, social media posts from development teams, official or community artwork, and videos on streaming sites such as trailers or developer documentaries. When new content is discovered, notifications are sent to the user through push-notifications or email if the user specifies. On the application itself, the number of new notifications since the user previously logged in are shown next to each category.

Figure 4

News Articles Mock-Up



Doom Eternal

News Articles

IGN

GameStop Will Sell DOOM Eternal Early to Help Separate Crowds

4 hours ago

7

GAMESPOT

What Time Does Doom Eternal Unlock On PS4, Xbox One, And Steam?

5 hours ago

8

gamesradar+

Doom Eternal Review: "Screams At You To Move Faster And To Fight Harder, And You Can Do Nothing But Obey"

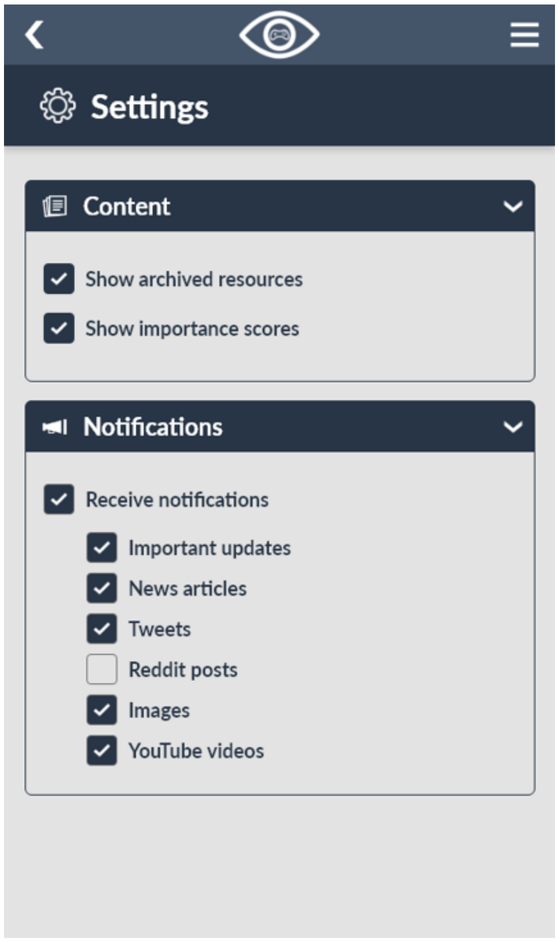
2 days ago

5

Figure 4 shows how the articles are represented when visiting the 'news articles' category. If available, an image from the most recent article is provided for viewing above the articles listed, and each article is accompanied by the name of the news outlet. Each article is given a score based on its importance generated by machine learning algorithms, and directs the user to the original news source. Any content in these categories that are over ninety days old will be removed from the listing and archived for later viewing at the user's discretion.

Figure 5

Settings Mock-Up



Content
- ✓ Show archived resources
- ✓ Show importance scores

Notifications
- ✓ Receive notifications
  - ✓ Important updates
  - ✓ News articles
  - ✓ Tweets
  - ☐ Reddit posts
  - ✓ Images
  - ✓ YouTube videos

Shown in Figure 5, the user is able to modify notification settings to change what kind of notifications they get and the frequency in which they are notified. Users have the option to customize the notifications they receive from the different categories; and, if desired, users have the capability to enable or disable the scoring for incoming news and the ability to view archived information. Users are also given the option to customize which games from their watch-list they wish to receive notifications from.

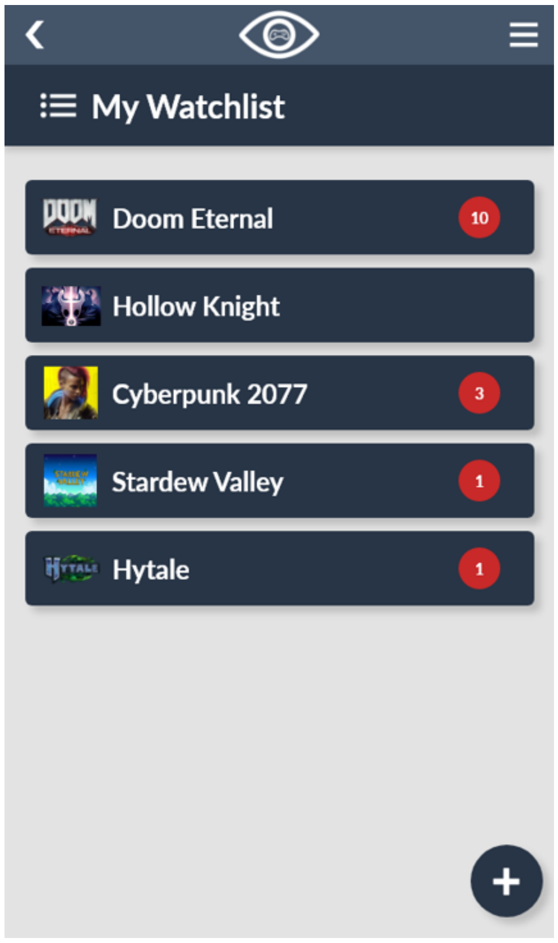
GameEye requires registering an account with an email and password in order to access its features. GameEye users take on one of two main roles: guest and registered user. A guest is anyone that visits the application who is not registered; a guest is only able to register for an account and learn more about the GameEye application. Registered users are those who have registered an account and can begin curating their watch-list and receiving notifications.

Games are tracked through the user’s watch-list as represented in Figure 6. Through the watch-list, a user can see all the games they follow and if any new content is available for viewing; interacting with a title in the watch-list takes the user to the categorized information related to that game. The watch-list can also be customized to fit the user preferences such as sorting order and which games can send alerts.

[This space intentionally left blank]

Figure 6

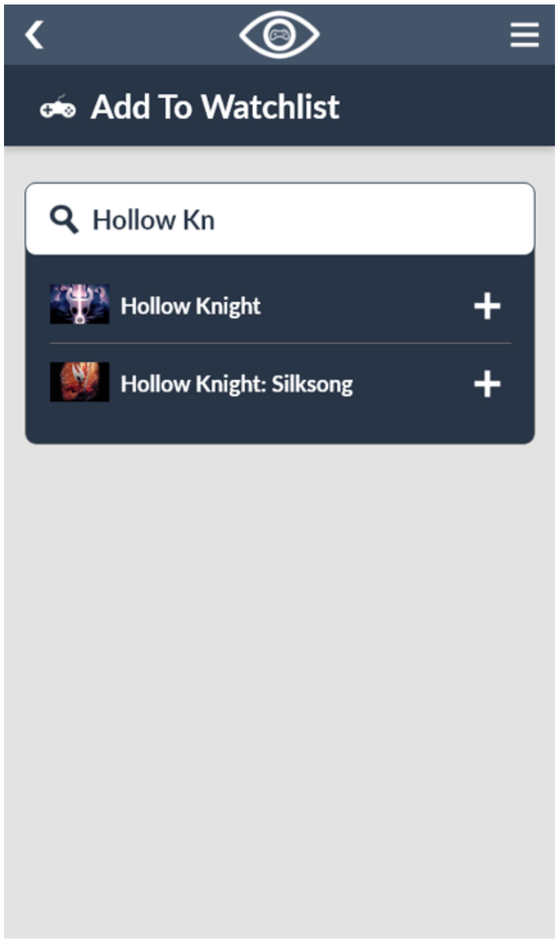
User Watch List Mock-Up



[This space intentionally left blank]

Figure 7

Searching Mock-Up

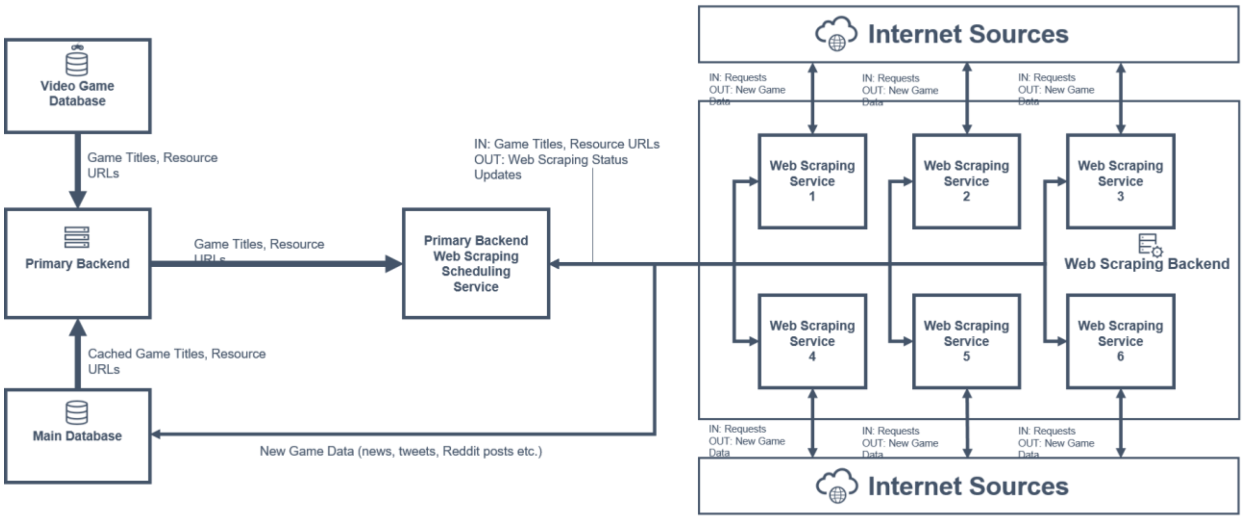


When searching for games to add to the watch-list, GameEye utilizes a search bar with auto-completion, as demonstrated in Figure 7, to find games and directly add them to the watch-list. When searching for a game, game titles and their respective information is retrieved from an existing video game database, such as the IGDB database or potentially a curated database of games. GameEye also provides a list of current popular games based on user activity to serve as suggestions for users.

Figure 8 shows the flow diagram for how the web scraping works on GameEye's backend.

Figure 8

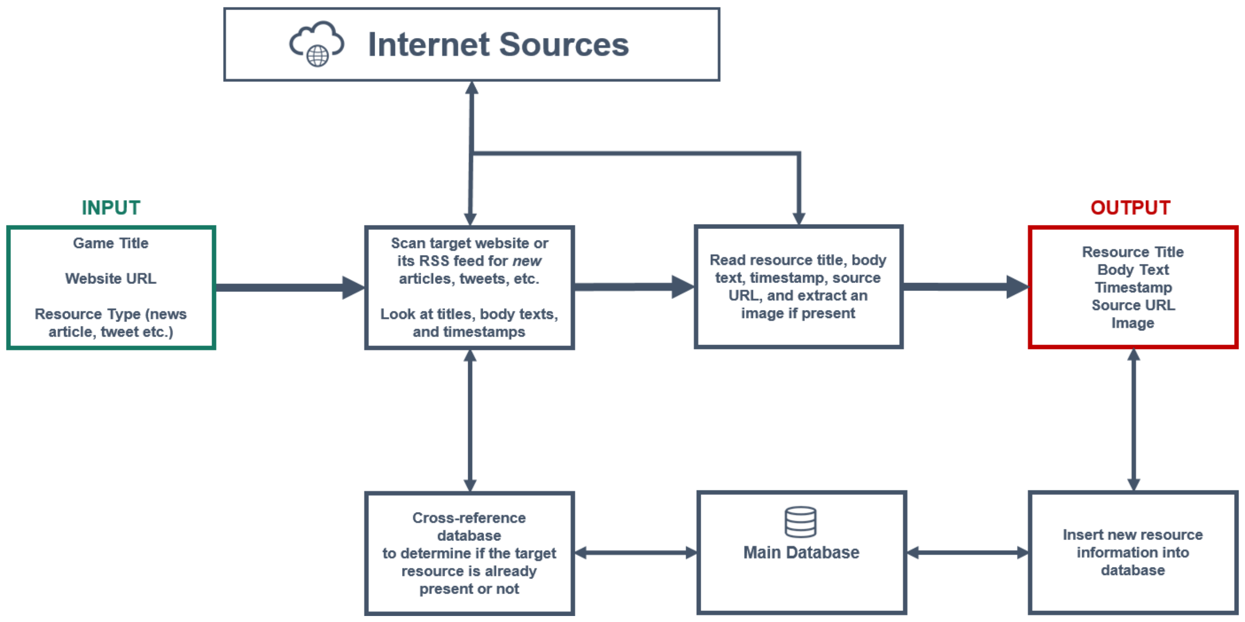
Web Scraping Flow Diagram



Web scraping services are developed to monitor different sources of information across the internet such as news aggregators, social media, online store pages like Steam, and streaming services. When new information is found, the scraper is able to provide collectively the title of resource found, the text tied to the resource, the resource URL, a timestamp of when the resource was found, and any potential thumbnail back to the GameEye database to be used for processing; this process is represented in Figure 9. From there, the resource type will be determined and stored in the appropriate category under the related game.

Figure 9

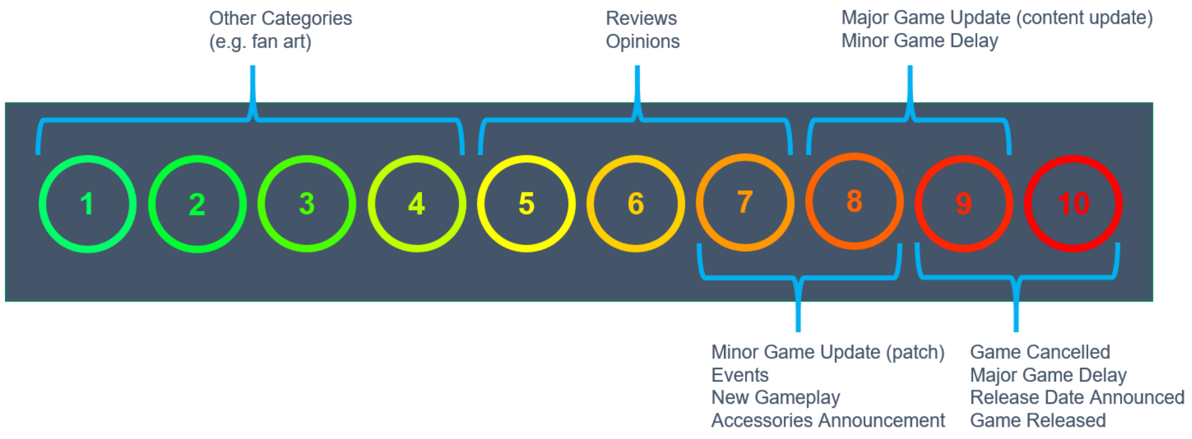
Web Scraping Service Algorithm


Internet Sources
INPUT
Game Title
Website URL
Resource Type (news article, tweet etc.)
Scan target website or its RSS feed for *new* articles, tweets, etc.
Look at titles, body texts, and timestamps
Read resource title, body text, timestamp, source URL, and extract an image if present
OUTPUT
Resource Title
Body Text
Timestamp
Source URL
Image
Cross-reference database to determine if the target resource is already present or not
Main Database
Insert new resource information into database

With the information provided by the web scrapers, GameEye will utilize machine learning algorithms to observe the resources and determine "impact" scores to help users determine how significant the information is.

Figure 10

Impact Score Scale



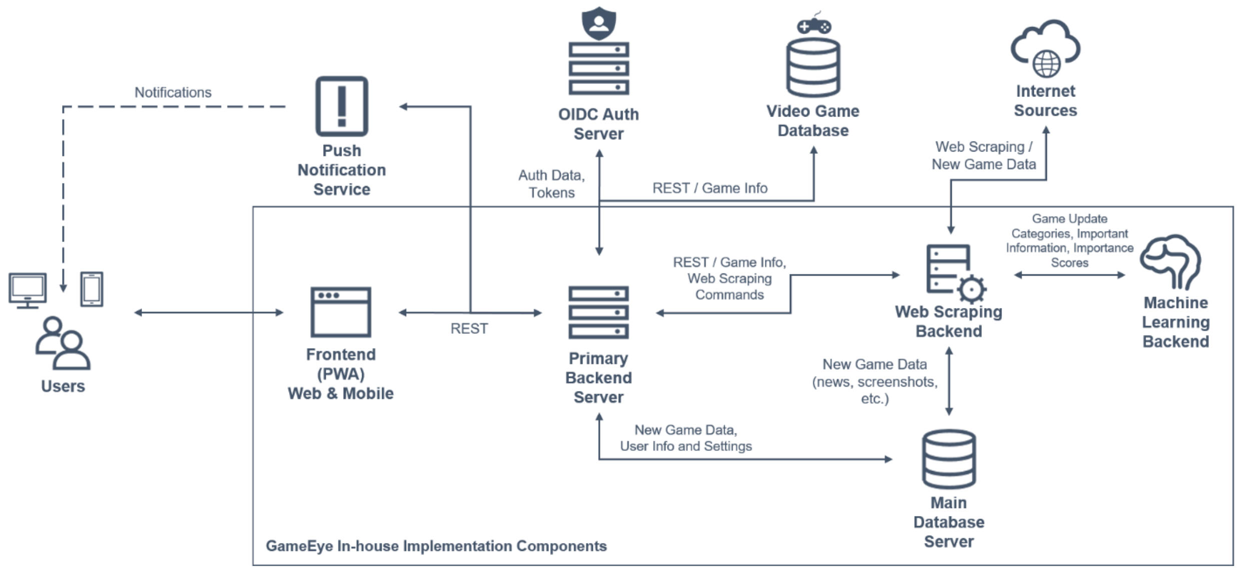
These scores range from one to ten with one being of low impact that may be of little interest to the user, and ten being high impact which the user would need to know in regards to development. This scale is shown in Figure 10.

2.2 Major Components (Hardware/Software)

Figure 11 shows the Major Functional Component Diagram (MFCD) for GameEye. The main components outside of the frontend, backend, and main servers are the video game database to retrieve video game titles, the web scraping backend, and the associated machine learning backend.

Figure 11

Major Functional Component Diagram


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

The hardware used by GameEye is the frontend server, main backend server, web scraping server, main database server, and machine learning backend server. The frontend server is an external-facing web server responsible for handling the frontend of the web application. The main backend is an external-facing web server exposing a REST API for communication between the frontend, web scraping backend, databases, and other third-party services. The web scraping server is also external-facing, but doesn’t monitor incoming traffic; this server also exposes a REST API for launching web scraping operations. The machine learning backend is an internal GPU-powered “TensorFlow Serving” server exposing a REST API for machine learning inference; it communicates directly with the web scraping server to extract information and classify data. The main database server is an internal MongoDB server that contains all data related to the GameEye application.

GameEye is a PWA that utilizes the WebStorm IDE, Angular Framework, and Google Workbox to handle the frontend work. The main programming languages used for the frontend are HTML, SASS/CSS, and TypeScript. The backend of GameEye uses the IntelliJ IDEA IDE, Spring Framework, jsoup library, and the MongoDB Java Driver. The main programming language used for the backend is Java. Testing for GameEye and all its functions will be done in the JUnit Java Framework and the Jest JavaScript Framework.

MongoDB, MongoDB Compass and the IGDB REST API will be used to build and maintain the GameEye databases. For third-party software, GameEye uses the Auth0 Single Page App SDK for authentication and Firebase Cloud Messaging to handle notifications on mobile devices. The software used to handle the machine learning algorithms are the Keras library, scikit-learn library, and a TensorFlow Serving model server. The primary programming language that GameEye uses for all forms of machine learning is Python. For natural language processing, GameEye will utilize the spaCy library which is used with Python.

3 Identification of Case Study

Upon initial deployment, the case study group will be comprised of gamers who are looking forward to new game releases over the course of 2021 and beyond. The users in this group will be tracking upcoming games and receiving notifications as news for these games become available. The users will be able to submit feedback within the application on current, live features and features they would like to have add in the future. There will also be an option for these users to request new games to follow if they are not present in the database; this will assist in further expanding GameEye's searching functionality and improving web scraper efficiency.

4 Prototype Description

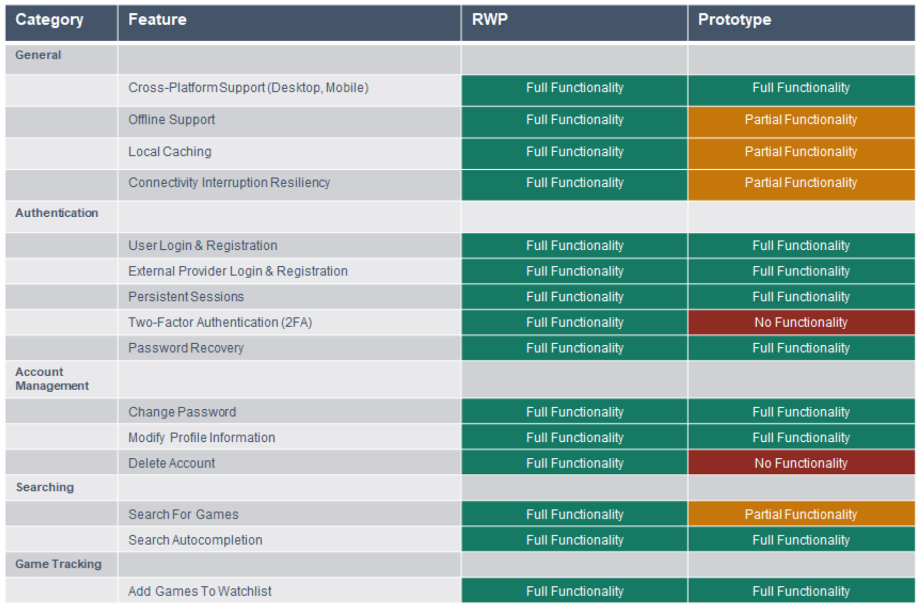
4.1 Features and Capabilities

The GameEye prototype will provide support for user accounts. The prototype will provide robust user authentication using external authentication support and account management features to allow users to modify profile information and change their passwords. Users will have access to a personal watchlist, which they can curate by searching for games within the application that will feature auto-completion. Users will also have access to a list of the current most-watched games on the platform by all users. There will be settings available for users to customize notifications they receive and desired frequency, the toggling of the archived resource visibility, and the toggling of impact score visibility.

[This space intentionally left blank]

Figure 12

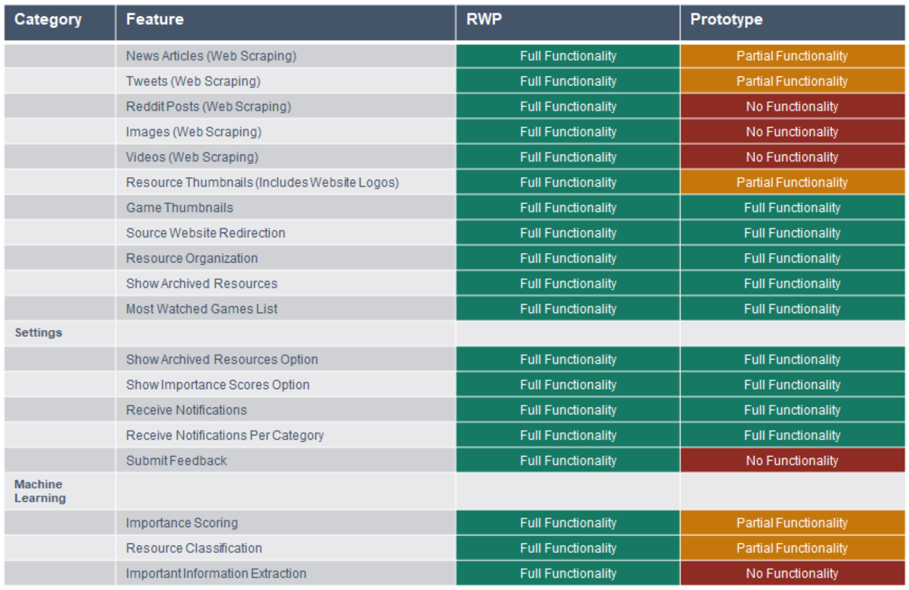
RWP vs. Prototype Features (1)



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

Figure 13

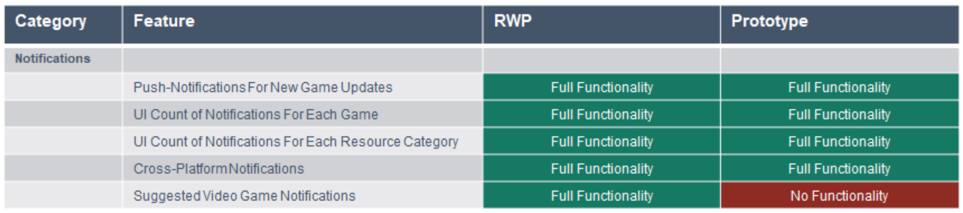
RWP vs. Prototype Features (2)



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

Figure 14

RWP vs. Prototype Features (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

The GameEye will utilize the use of web scrapers to scrape data from multiple online sources; this data includes news articles and their content, which are then classified and scored using machine learning algorithms. Links to these articles and any thumbnails associated with them are provided to the user for viewing, which will then redirect users from the application to the source URL outside when interacted with. The prototype will have cross-platform support between desktop and mobile platforms, and provide push notifications viewable to the user. Figures 12-14 shown above represent the differences in functionality and features between the prototype and real-world product.

4.1 Risk Mitigation

The GameEye prototype will utilize several techniques behind-the-scenes to mitigate potential risks related to database failures, web scraping failure, and security breaches. To protect against any kind structural changes to source websites that could affect the web scrapers, RSS feeds will be scraped to ensure web scraping does not fail. To prepare for a situation where the IGDB is down, content from the IGDB will be stored on the GameEye main database for redundancy. In an instance where the GameEye main database fails, content will be cached on user devices so the application does not appear blank in case of a failure.

Security measures will be in place to ensure user information is safeguarded and the application is protected against different forms of cyberattack. To protect user account information, thorough database encryption will be used for login information alongside the use of a 3rd party authentication provider. The GameEye platform will be made scalable with the use of load balancers and multiple instances to ensure protection of the main database and server in case of a heavy load or outright DDOS attack. Both the back and front end developers will also ensure the use of proper coding practices to prevent cyber-attacks on the application such as NoSQL Injection, data exposure, and broken authentication and/or access controls.

4.2 Architecture

The prototype will be utilizing the same architecture as the real-world product; however, the prototype will not be utilizing the replicated servers that would be present in the real-world product.

4.3 Development Challenges

The largest expected obstacle while developing the prototype is learning the necessary technology, languages, and frameworks required for implementation. For both the front and backend developers, this prototype is in unfamiliar territory and requires time to become fully acclimated with the involved systems. Part of this process includes properly securing the necessary connections between multiple backend repositories to minimize issues. For the machine learning algorithm, a large magnitude of data needs to be manually collected and labeled to ensure the scraped data is labeled and scored accurately. In order to improve application connectivity and performance, the GameEye frontend developers are challenged with having to find the best way to vigorously cache content on the frontend. The backend developers are challenged with implementing the web scrapers in way to prevent source links and their content from breaking or becoming unviable in the case the changes are made to the platform where the sources originate (i.e. a website changes their structure or design).

5 Glossary

AAA: Classification of video games produced by a mid-sized or major game publisher that typically have large development and marketing budgets; analogous to “blockbuster” in film.

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.

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

Hit List: 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: Internet Games Database; 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: OpenID Connect; 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.

TensorFlow: Open-source software library for differential programming and dataflow; used in machine learning applications.

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.

6 References

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

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

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

Gough, C. (2019, October 9*). Google Play: Number of available games by quarter*. Statista. 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. *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

WePC. (2020, January 1). *2020 Video Game Industry Statistics, Trends & Data*. WePC. Retrieved May 4, 2020, from https://www.wepc.com/news/video-game-statistics/#video-game-monetization