3.1.1. Authentication (Angelos). A secure authentication system shall be provided for users to access GameEye. The following requirements shall be met:

3.1.1.1. Account Creation. The system shall allow new users to create an account. Accounts shall be created in one of two ways:

1. New users shall be able to use their email and a password of their choice to create an account.

2. New users shall be able to use an existing third-party account to associate with GameEye. The following external providers shall be supported:

a. Google.

b. Microsoft.

3.1.1.2. Login. The system shall allow existing users to login. Login shall be accomplished in one of two ways:

1. Users shall be able to use their email and password to login.

2. Users shall be able to login with an existing third-party account. The following external providers shall be supported:

a. Google

b. Microsoft

3.1.1.3. Email Verification. The system shall ensure that users use an email address they control before they can access the platform to reduce fraud. The following requirement shall be met:

1. Users shall be able to change their email address while they have an unverified email address.

3.1.1.4. Password Reset. The system shall provide a way for users to reset their password if they have forgotten it.

1. Users shall be sent a password reset email.

3.1.1.5. Persistent Sessions. The system shall ensure that users do not have to sign in again indefinitely. The users will have to sign in again if one of the following events occur:

1. The user logs out.

2. A major account change is detected, such as email or password change.

3. The user is deleted.

4. The user is disabled.

3.1.2. Account Management (Angelos). Standard account management features shall be provided by GameEye so that users have control over their account information. The following requirements shall be met:

3.1.2.1. Change Name. The system shall provide a way for users to change their names or provide one since GameEye does not require that users provide a name on account creation.

1. Users shall be able to set a name, even if they have not previously specified one

3.1.2.2. Change Email. The system shall provide a way for users to change their email. The following requirement shall be met:

1. The system shall ensure that users must verify any new email address they use.

2. Users shall not be able to use GameEye until they verify their new email address.

3. Users shall be able to change their email address while they have an unverified email address.

3.1.2.3. Change Password. The system shall allow users to change their password once they are signed in.

1. Users shall be sent a password reset email.

3.1.2.4. Reauthentication. The system shall require users to reauthenticate themselves before changing important account information such as their email or password.

1. Users shall reauthenticate with their email and password if they used that

2. Users shall reauthenticate with their Google account if they used that

3. Users shall reauthenticate with their Microsoft account if they used that

3.1.3. Game Searching (Angelos, Jacob). The system shall allow users to search for games to add to their watchlists. The following requirements shall be met:

3.1.3.1. Autocompletion Suggestions. Searching shall support the autocompletion of partially inputted game titles. Game title suggestions shall be offered based on what has been typed so far by the user. The following requirements shall be met:

1. The system shall be able to offer relevant autocompletion suggestions even if a user makes spelling errors or misses some characters.

2. The system shall be able to offer autocompletion suggestions in real-time, as the user is typing in the search box.

3.1.3.2. Thumbnails. The system shall display a thumbnail of the game logo next to search results, whenever possible, to make it easier for users to know exactly what games come up as results. The following shall be met:

1. Thumbnails shall be compressed to allow fast transfer

2. Thumbnails shall be a scaled-down version of the original image

3.1.3.3. Searching Backend. Searching shall be implemented using the Spring Boot primary GameEye backend and an ElasticSearch database.

1. The primary backend shall call the ElasticSearch database to get autocompletion suggestions.

3.1.4. Article Impact Scoring (Angelos, everyone for labeling). The system shall compute an impact score for every news article title using machine learning. The following requirements shall be met:

3.1.4.1. Training Dataset. The training dataset shall consist of 11,000 article titles. The following requirements shall be met:

1. Every title shall be labeled with an impact score from 1 to 3.

2. Titles shall be game-specific

3. Shall not contain any titles present in the training dataset

3.1.4.2. Test Dataset. The evaluation dataset shall consist of 1,000 article titles. The following requirements shall be met:

1. Every title shall be labeled with an impact score from 1 to 3.

2. Titles shall be game-specific

3. Shall not contain any titles present in the training dataset

3.1.4.3. Machine Learning Framework. The impact scoring component shall be implemented using Python with TensorFlow and Keras.

3.1.4.4. Model Serving. The machine learning model shall be served using TensorFlow Serving. The following requirements shall be met:

1. The machine learning system shall provide a REST API to call the machine learning model

2. The machine learning system, upon receiving an article title, shall return an impact score

3.1.5. Push-Notifications (Angelos, Chris, Brenden). The system shall prepare and send notifications to users when new articles have been found for a game in their watchlist. The following requirements shall be met:

3.1.5.1. Notification Backend. Notifications shall be sent by the primary GameEye backend via Firebase Cloud Messaging.

3.1.5.2. Process. Notifications shall be sent immediately after new articles have been found. The following requirements shall be met:

1. Scraped article titles shall be scanned to find which specific game they refer to. This shall be done by finding the longest possible game title contained in an article title from all game titles stored in the database.

2. Articles that do not refer to a specific game shall be deleted.

3. The titles of the articles that remain shall be sent to the machine learning backend for impact score calculation.

4. For all new articles, a list of users that have the games they refer to in their watchlists shall be compiled.

5. Notifications shall be sent to only the users that watch games for which new articles have been found.

3.1.5.3. Game-Specific Notifications. Notifications shall be game-specific when news articles for only a specific game has been found. The following requirements shall be met:

1. A notification shall include the name of the game and the types of scraped resources, such as “news article”.

2. A notification shall include the types of scraped resources, such as “news article”.

3. A notification shall include a thumbnail of the game if one exists.

3.1.5.4. Notification Grouping. Notifications for several different games shall be grouped and sent as a single notification to avoid flooding the user’s inbox if multiple games had new articles during a scraping cycle. The following notification content is required:

1. Grouped notifications shall mention that new game information has been found for several games.

3.1.5.5. Cross-Platform Notifications. Users shall receive notifications on both desktop and mobile devices.

3.1.6. Primary Backend Endpoints (Angelos, Ben, Jacob). The primary backend serves requests from the frontend, coordinates web scraping, and interacts with the database. The following requirements shall be met:

3.1.6.1. User Profile Endpoints. A set of endpoints for handling user profile operations shall be created. The following endpoints shall be implemented:

3.1.6.1.1. User Profile Creation. An endpoint for creating a user profile in the GameEye database shall be created. The following requirements shall be met:

1. Get the user ID from the ID token contained in the request.

2. Create a user document if the user ID does not match any within the database

3.1.6.1.2. User Profile Deletion. An endpoint for deleting a user profile shall be created. The following requirements shall be met:

1. Get the user ID from the ID token contained in the request.

2. Delete the user profile matching the received user ID from the database.

3. Delete the user account matching the received user ID from Firebase.

3.1.6.1.3. User Profile Settings. An endpoint for handling user profile settings shall be created. The following requirements shall be met:

1. Get the user ID from the ID token contained in the request.

2. Receive new user profile settings from the request

3. Update any settings in the database that differ to match the new user profile settings.

3.1.6.2. Watchlist Endpoints. A set of endpoints for handling operations on user watchlists shall be created. The following endpoints shall be implemented:

3.1.6.2.1. Get Watchlist Games. An endpoint for getting all games a user has in their watchlist shall be created. The following requirements shall be met:

1. Get the user ID from the ID token contained in the request.

2. Return data for all games in the user's watchlist. The following data is required:

a. Game title

b. Game thumbnail or logo

c. Notification count

3.1.6.2.2. Add Watchlist Game. An endpoint for adding a new game to a user's watchlist shall be created. The following requirements shall be met:

1. Get the user ID from the ID token contained in the request.

2. Receive a game ID from the request

3. Add the game referenced by the received game ID to the user's watchlist

3.1.6.2.3. Delete Watchlist Game. An endpoint for deleting a game from a user's watchlist shall be created. The following requirements shall be met:

1. Get the user ID from the ID token contained in the request.

2. Receive a game index from the request

3. Delete the game under the received index from the user's watchlist.

3.1.6.3. Searching Endpoints. A set of endpoints for searching games using autocompletion shall be created. The following endpoints shall be implemented:

3.1.6.3.1. Game Title Autocompletion. An endpoint that returns a list of autocompleted game titles given a partial game title shall be created. The following requirements shall be met:

1. Get a game title fragment from frontend.

2. Query the ElasticSearch database and get a list of autocompleted game titles.

3. Get the IDs of the autocompleted game titles from the GameEye database.

4. Return the list of autocompleted game titles and their IDs.

3.1.6.4. IGDB Endpoints. A set of administrative endpoints for interfacing with IGDB game data shall be created. The following endpoints shall be implemented:

3.1.6.4.1. IGDB Replication Endpoint. An endpoint shall be created that accepts minimum and maximum IGDB IDs. The endpoint shall populate the GameEye database with IGDB game data for games within the inclusive minimum-maximum range. The endpoint shall update existing game data. The following requirements shall be met:

1. Get the minimum and maximum IGDB game IDs contained in the request.

2. Clone game data from IGDB to the GameEye database. Game data from IGDB shall be copied into the games collection. The following GameEye database fields shall be populated from the IGDB database:

a. igdbId

b. title

c. platforms

d. genres

e. sourceUrls

3. Update existing GameEye database game data. Existing games in the GameEye database shall be updated from IGDB data. Only the following fields shall be updated from the IGDB database:

a. title

b. platforms

c. genres

d. sourceUrls

4. Update ElasticSearch with game titles. Game title shall be updated to the ElasticSearch database when the following events occur:

a. Clone game data from IGDB to the GameEye database

b. Update existing GameEye database game data

3.1.6.5. Game Endpoints. A set of endpoints for getting game information shall be created. The following endpoints shall be implemented:

3.1.6.5.1. Game Logo. An endpoint shall be created that returns the logo of the respective game. The following requirements shall be met.

1. Get the game ID contained in the request.

2. Return the URL of the logo of the game referenced by the passed id

3.1.6.5.2. List of News Articles for Game. An endpoint shall be created that given a game ID returns all articles pertaining to that game. The following requirements shall be met:

1. Get the game ID contained in the request.

2. Get all the article data for the game that has the ID from the request. The following data shall be returned for each article:

a. Title

b. Text snippet

c. Publishing date

d. Publisher logo

e. Thumbnail

f. Impact score

g. Source URL

3.1.6.5.3. Most-Watched Games List. An endpoint shall be created that returns the most-watched games on the platform. The following requirements shall be met:

1. Query the database to get the top 50 games with the highest amount of watchers

2. Return a list of the top 50 game titles along with their number of watchers to the frontend.

3.1.7. User Settings (Jonathan). Users shall customize the content that is shown to them and the notifications that they receive.

3.1.7.1. Content. Users shall choose what type of content they want to receive.

1. Users shall be provided the option of choosing to see old resources that have been archived or not to see archived resources at all.

2. Users shall choose whether they want to see the impact scores of articles.

3.1.7.2. Notifications. Users shall be able to customize their notification preferences.

1. Users shall be able to choose whether to receive notifications at all.

2. Users shall be able to choose for which impact scores they will receive notifications for.

3.1.8. GameEye Database (Jacob). The GameEye database shall store news website data, user data, game data, and image data. Data shall be stored in four collections in MongoDB using the BSON document format. The BSON document format supports specific data types. Each field shall be provided in the specified data type. Note that each collection shall be named case-sensitively according to the functional requirements. The following database structures shall be implemented:

3.1.8.1. newsWebsites Collection. The newsWebsites collection shall store data related to new websites. The web scrapers use data in the newsWebsites collection to retrieve news articles for video games. The following fields and corresponding data types shall be provided:

Name	Data Type
_id	ObjectId
name	String
logo	Binary
siteUrl	String
rssFeedUrl	String
lastUpdated	Date

3.1.8.2. users Collection. The users collection shall store data related to users. User data includes the authentication data. Personal data for following video games are stored in the users collection. The users collection shall store personal preferences for each user. The users collection shall store single-layer fields and multi-layer fields. The following database structures shall be implemented:

3.1.8.2.1. Single-Layer Fields. The users collection shall contain the following single-layer fields and corresponding data types:

Name	Data Type
_id	ObjectId
status	String
plan	String

3.1.8.2.2. Multi-Layer Fields. The users collection shall contain the following multi-layer fields and corresponding data types:

1. preferences – Object
    a. contentPreferences - Object
        i. showArvhivedResources – Boolean
        ii. showImpactScores – Boolean
        iii. impactScores – Boolean Array
    b. notificationPreferences – Object
        i. showArticleResources – Boolean
        ii. showImageResources – Boolean
2. watchList - Object Array; each object shall contain the following fields:
    a. gameId – String
    b. notificationCount – Integer
    c. resourceNotifications – Object
        i. articleNotifications – Object
            1. count – Integer
            2. articleIds – String Array
        ii. imageNotifications – Object
            1. count – Integer
            2. imageIds – String Array

3.1.8.3. games Collection. The games collection shall store data gathered from web scraping. Data from external databases such as IGDB shall be stored in the games collection. The games collection shall store single-layer fields and multi-layer fields. The following database structures shall be implemented:

3.1.8.3.1. Single-Layer Fields. The games collection shall contain the following single-layer field**s:**

Name	Data Type
_id	ObjectId
igdbId	String
title	String
watchers	Integer
platforms	String Array
status	String
lastUpdated	Date
genres	String Array

3.1.8.3.2. Multi-Layer Fields. The users collection shall contain the following multi-layer fields:

1. sourceUrls – Object
    a. publisherUrl – String
    b. steamUrl – String
    c. subRedditUrl – String
    d. twitterUrl – String
2. resources – Object
    a. images – Object Array
        i. _id – ObjectId
        ii. title – String
        iii. imageId – String
        iv. lastUpdated – Date
    b. articles – Object Array
        i. _id – ObjectId
        ii. title – String
        iii. url – String
        iv. newsWebsite – DBRef to the newsWebsites collection
        v. thumbnail – DBRef to the images collection
        vi. snippet – String
        vii. publicationDate – Date
        viii. lastUpdated – Date
        ix. impactScore – Integer

3.1.8.4. images Collection. The images collection shall store images for news articles, news website logos, and games. Documents in the games collection reference the images collection for stored images via foreign key. The following fields and corresponding data types shall be provided:

Name	Data Type
_id	ObjectId
type	String
data	Binary

3.1.9. Web Scraping (Chris and Brenden) The system shall retrieve information from video game news RSS feeds. The following requirements shall be met:

3.1.9.1. Interval. Information shall be retrieved from the RSS feeds twice a day.

3.1.9.2. Sources Information shall be retrieved from the following news sources:

1. IGN

2. GameSpot

3. Eurogamer

4. PC Gamer

5. GameEye Mock News

3.1.9.3. Information The following details shall be scraped from each article in the RSS feed:

1. Title of article

2. Snippet of article

3. Date of publication

4. Article URL

5. Images

3.1.9.4. Force Scrape. The system shall have functionality for force scraping for demonstration purposes.

3.1.9.5. Organization. Articles shall be organized by which game title they are referencing from the database.

3.1.9.6. Reference Game Search. Each scraped article shall be analyzed to determine which specific game it is referencing. The following requirements shall be met:

1. A search shall be made to ElasticSearch using the article title as the query.

2. The search results shall include the top 10 most likely games to be referenced in the article title.

3. Each returned game title shall be searched in an article's title.

4. The longest exactly matching game title shall be set as the article's reference game.

5. If there are multiple exact matches, a scraped article shall be set to reference multiple games

3.1.9.7. Data Clean-up Web scrapers shall delete recently scraped articles not explicitly related to any games contained within GameEye's *games collection* in the GameEye database.

3.1.9.8. Duplicates Web scrapers shall delete any scraped articles already present in the GameEye database.

3.1.9.9. Database Integration The final batch of data collected from web scraping will be pushed to GameEye's database. The following requirements shall be met:

1. Articles that contain games not already present in the GameEye database will be added to GameEye's *games collection.*

2. Images attached to scraped articles will be stored in GameEye's *images collection*.

3. Websites used by web scrapers will be stored in GameEye's *newsWebsites collection.*

3.1.10. Watchlist (Adrian). The system shall keep track of the games a user has selected to follow, known as their watchlist. The following requirements shall be met:

3.1.10.1. Add Game. The system shall allow users to add new games to their watchlist. The following requirements shall be met:

1. Provide the user with a search bar to find games they wish to follow.

2. The search bar shall have an auto-completion feature, which displays suggestions beneath the search bar.

3. Add the selected games to the user's watchlist.

3.1.10.2. Remove Game. The system shall allow users to remove games from their watchlist.

3.1.10.3. Watched Games. The system shall display the user's current watchlist. The following requirements shall be met:

3.1.10.3.1. Watchlist Entries The system shall implement material cards to represent watchlist entries. The following requirements shall be met:

1. Each card shall describe the game it represents with the title and a thumbnail.

2. Each card shall include the notification count of the total amount of new game updates.

3. The user shall be able to access the updates for a specific game by clicking on the corresponding card.

3.1.10.3.2. Game Updates. The system shall organize the update information into specific categories. The following requirements shall be met:

3.1.10.3.2.1. Update Card. The system shall implement material cards to represent each update category. The following requirements shall be met:

1. Each card shall describe the category it represents with a title and a thumbnail.

2. Each card shall include a notification count of the new game updates for the category it represents.

3.1.10.3.2.2. News Articles. The system shall list the recent articles it has collected along with relevant descriptors. The following requirements shall be met:

1. The title of the article as it appears on the source site.

2. Article impact scores shall be displayed for each listing.

3. Each listing should contain the logo from the site the article was published.

4. Each listing should indicate the amount of time that has passed since the article was published.

5. The user shall be directed to the source site by clicking on an article's designated region.

3.1.11. Most-Watched Games List (Angelos, Adrian, Jonathan). GameEye shall provide a list of the most-watched games on the platform. The following requirements shall be met:

3.1.11.1. Watcher Counter. The backend shall keep track of how many users are watching a particular game. The following requirements shall be met:

1. When a user adds a new game to their watchlist, the watcher counter for the appropriate game shall be incremented

2. When a user removes a game from their watchlist, the watcher counter for the appropriate game shall be decremented

3.1.11.2. Watcher Counter Sorting. The backend shall query the database to get the games with the highest watcher counters. The following requirements shall be met:

1. The backend shall query the database to get the top 50 most watched games.

2. The game titles along with their watcher count shall be available for the frontend.

3.1.12. Cross-platform (Angelos, Adrian, Jonathan). GameEye shall be implemented as a cross-platform web application. The following requirement shall be met:

3.1.12.1. Responsiveness. GameEye shall use responsive web design, making it able to adjust to various screen sizes. The following requirements shall be met:

1. GameEye shall be usable from devices with large-sized screens such as desktop computers

2. GameEye shall be usable from devices with medium-sized screens, such as laptops.

3. GameEye shall be usable from devices with large-sized screens, such as mobile phones.

3.1.12.2. Browser Support. GameEye shall be able to run on major web browsers. The following browsers shall be officially supported:

1. Google Chrome

2. Microsoft Edge

3. Mozilla Firefox

4. Opera

5. Brave

6. Safari

3.1.13. Testing (Everyone). Extensive testing shall be implemented, including with mock data, to ensure that the front-end and back-end GameEye components work as intended. The following requirement shall be met:

3.1.13.1. Backend Testing. Automated tests for the backend shall be implemented to test web scraping, endpoints, services, and integration of the various backend components.

1. Unit tests shall be implemented for isolatable backend components. The following components shall be tested:

a. API endpoints.

b. Services.

2. Integration tests shall be implemented for component interactions. The following interactions shall be tested:

a. Primary backend and database

b. Primary backend and machine learning backend

3.1.13.2. Frontend Testing. Automated tests for the frontend shall be implemented to ensure that frontend services and user flow work as intended.

1. Unit tests shall be implemented for isolatable frontend services.

2. Automated End-to-End (E2E) tests shall be implemented for the frontend to test it working as a whole. These tests shall simulate user interaction in a real browser.

3.1.13.3. Mock News Website. A mock news website shall be created that will contain news article titles. The following requirements shall be met

3.1.13.3.1. Articles Page. The mock news website shall display a list of mock news articles. The following requirements shall be met:

1. Each article shall have a title
2. Each article shall have a publication date
3. Each article shall have a short description of its content

3.1.13.3.2. Content Management System. The mock news website shall include a content management system (CMS) for managing mock news articles. The following requirements shall be met:

1. Users shall be able to add new articles
2. Users shall be able to edit existing articles
3. Users shall be able to upload images to use within articles
4. All articles shall be stored on a public GitHub repository