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

2. External Provider. 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. Email. Users shall be able to use their email and password to login.

2. External Provider. 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.

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

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.

3.1.2.2. Change Email. The system shall provide a way for users to change their email.

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

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

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

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

2. Real-Time Suggestions. 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.

3.1.3.3. Searching Backend. Searching shall be implemented using the Spring Boot primary GameEye backend and an ElasticSearch database. 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 and assign an impact score to 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. Every title shall be labeled with an impact score from 1 to 3.

3.1.4.2. Test Dataset. The evaluation dataset shall consist of 1,000 article titles. Every title shall be labeled with an impact score from 1 to 3.

3.1.4.3. Game-Specific Titles. Every article title in the datasets shall explicitly reference at least one game title.

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

3.1.4.5. Model Serving. The machine learning model shall be served using TensorFlow Serving.

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. Frequency. Notifications shall be sent immediately after new articles have been scraped and have been fully processed and assigned impact scores.

3.1.5.3. Game-Specific Notifications. Notifications shall be game-specific when news articles for only a specific game have been found. The following notification content is required:

1. Name of Game. Game-specific notifications shall include the name of the game and the types of scraped resources, such as “news article”.

2. Types of Scraped Resources. Game-specific notifications shall include the types of scraped resources, such as “news article”.

3. Thumbnail. Game-specific notifications 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. Several Games. 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.

INSERT HERE

3.1.6 Backend Endpoints (Benjamin, Jacob)

The backend shall act as a midpoint between the database and the frontend. The following functionality is required.

3.1.6.1 Account Creation

GameEye shall be able to store information on a user's GameEye account. To achieve this, the following must be implemented.

3.1.6.1.1 An operation to receive an ID token from the front end shall be created.

3.1.6.1.2 An operation to decode the ID token shall be created.

3.1.6.1.3 An operation to collect the user ID from the ID token shall be created.

3.1.6.1.4 An operation to search through the main database's user documents via the user ID as an index shall be created

3.1.6.1.5 An operation to create a user document shall be created if the user ID does not match any within the database

3.1.6.1.6 An operation to user settings from the frontend shall be created.

3.1.6.1.7 An operation to store user settings into the main database shall be created.

3.1.6.2 Article Endpoints

The backend shall act as a midpoint between the database and the frontend. To achieve the following must be implemented.

3.1.6.2.1 An operation to web-scrape twice a day shall be created.

3.1.6.2.2 An operation for the primary backend to receive new articles from web-scrapping shall be created.

3.1.6.2.3 An operation to send article titles from the primary backend to the machine learning backend shall be created.

3.1.6.2.4 An operation to have the primary backend receive the impact scores from the machine learning backend shall be created.

3.1.6.2.5 An operation to have the primary backend must store the modified data inside of the main database shall be created.

3.1.6.2.6 An operation to have the primary backend push notifications from the main database to the users shall be created.

3.1.6.2.7 An operation to have the primary backend send the article titles to the frontend shall be created.

3.1.6.2.7 An operation to have the primary backend send the article images to the frontend shall be created.

3.1.6.2.8 An operation to have the primary backend send the article summary to the frontend shall be created.

3.1.7. 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.

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

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

3.1.7.1.2.1. Impact score level. Users shall choose which level of impact score to receive notifications for a. None; b. 1; c. 2; d. 3; e. 1 and 2; f. 1 and 3; g. 2 and 3; h. 1 2 and 3.

3.1.7.2. Notifications. Users shall choose what type of notifications they want to receive.

3.1.7.2.1. Receive notifications. Users shall be provided the option of choosing to receive push notifications or not to receive push notifications.

3.1.8 GameEye Database (O: 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 functional requirements shall be provided:

3.1.8.1 newsWebsites Collection (O: Jacob)

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 (O: Jacob)

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 functional requirements shall be provided:

3.1.8.2.1 Single-Layer Fields (O: Jacob)

The users collection shall contain the following single-layer fields and corresponding data types:

Name	**Data Type**
_id	ObjectId
firebaseId	String
status	String
plan	String

3.1.8.2.2 Multi-Layer Fields (O: Jacob)

The users collection shall contain the following multi-layer fields and corresponding data types:

1. preferences – Object
    a. contentPreferences – Object
        i. showArchivedResources – 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 (O: Jacob)

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 functional requirements shall be provided:

3.1.8.3.1 Single-Layer Fields (O: Jacob)

The games collection shall contain the following single-layer field**s:**

Name	**Data Type**
_id	ObjectId
title	String
platforms	String Array
status	String
lastUpdated	Date
genres	String Array

3.1.8.3.2 Multi-Layer Fields (O: Jacob)

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 newsWebsites collection

v. thumbnail - DBRef to images collection
vi. snippet – String
vii. publicationDate – Date
viii. lastUpdated – Date
ix. impactScore - Integer

3.1.8.4 images Collection (O: Jacob)

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.

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 different news sources:

1. IGN
2. GameSpot
3. Eurogamer
4. PC Gamer

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

1. Title
2. Snippet of article
3. Date of publication
4. URL

3.1.9.4 Impact Score Articles shall be assigned a score based on the content of the article using machine learning.

3.1.9.5 Classification Articles shall be classified by which game they are referencing from the database.

3.1.9.6 Backend Implementation GameEye's web scrapers will be implemented in the main backend.

3.1.9.7 Article Storage Articles collected by each web scraper will be stored in private data structures.

3.1.9.8 Data Clean-up Web scrapers will use cross-referencing algorithms to remove stored articles not explicitly related to any games contained in GameEye's *games collection* in the main database.

3.1.9.9 Duplicates Web scrapers will eliminate duplicate articles stored before sending to the database.

3.1.9.10 Database Integration The final batch of data collected from web scraping will be pushed to GameEye's main database.

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

3.1.9.10.2 Storing Images Images attached to articles found by web scrapers will be stored in GameEye's *images collection*.

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

3.1.10 Watchlist (Adrian). The collection of games a user has selected to receive updates on.

3.1.10.1 Add Game. The system shall allow users to add new games to their watchlist.

3.1.10.1.1 Search Bar. The system shall provide a search bar for the user to find games to follow.

3.1.10.1.2 Auto-completion. The search bar shall have an auto-completion feature, which displays suggestion beneath the search bar.

3.1.10.2 Watched Games. The system shall show the user the games they are currently following.

3.1.10.2.1 Game Card. The system shall implement material cards to represent each game.

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

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

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

3.1.10.2.2 Game Updates. The system shall organize the update information into specific categories.

3.1.10.2.2.1 Update Card. The system shall implement material cards to represent each update category.

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

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

3.1.10.2.2.2 News Articles. The system shall list the recent articles it has collected along with relevant descriptors.

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

3.1.10.2.2.2.2 Impact Score. Article impact scores shall be displayed for each listing.

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

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

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

3.1.10.2.2.3 Tweets.

3.1.10.2.2.4 Reddit Posts.

3.1.10.2.2.5 Images.

3.1.10.2.2.6 YouTube Videos.

3.1.10.2.2.7 Important Updates.

3.1.10.3 Most Watched. The system shall display a list of games that are most frequently watched by all users.

3.1.11. 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:

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

a. API endpoints.

b. Services.

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

a. Primary backend and database

b. Primary backend and web scraping backend

c. Web scraping backend and database

d. Primary backend and machine learning backend

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

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

2. End-to-End Testing. 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.

3.1.11.3. Mock News Website. A mock news website shall be created that will contain news article titles and other metadata to test GameEye's web scraping, machine learning, and notification components more easily.

1. Article Insertion. A mechanism that allows the insertion of new articles shall be implemented to test how GameEye responds to new articles.

2. Backend. A backend for the news website that retrieves articles from a database shall be implemented.

3. Data Storage. Articles shall be stored in a database in the GameEye MongoDB server.