CEN 4010 Principles of Software Engineering

Spring 2023

Milestone 3 Vertical Prototype

Group 14: The Web Warriors

GGTracker: A web app to help you track your gaming habits.

Siobahn Devlin: [sdevlin2021@fau.edu](mailto:sdevlin2021@fau.edu)

Peyton Adkins: [padkins2020@fau.edu](mailto:padkins2020@fau.edu)

David Miloseski: [dmiloseski2020@fau.edu](mailto:dmiloseski2020@fau.edu)

Jacob Goldstein: [jacobgoldste2017@fau.edu](mailto:jacobgoldste2017@fau.edu)

Kanksha Parikh: [kparikh2020@fau.edu](mailto:kparikh2020@fau.edu)

GitHub Repository: <https://github.com/cen4010-sp23-g14/ggTracker>

JIRA Link: <https://fau-cen4010-group14.atlassian.net/jira/software/projects/GGTRAC/boards/34>

03/03/2023

|  |  |
| --- | --- |
| Revision Date | Description |
| 03/03/2023 | Initial document submission |
| 03/28/2023 | Adding more details for Vertical Prototype |
| 04/02/2023 | Finishing up Milestone 3 Report |

2 - Executive Summary

One hundred years ago, Baseball was considered America's national pastime. However, in this modern age of technology, video games have surpassed it as our favorite leisure time activity. With thousands of games accessible from a multitude of devices at any given time, it can be disorientating trying to keep track of every game that is of interest to the user.

Our team is introducing GG Tracker. A state-of-the-art application for the user to stay informed and discover all their video game needs. GG Tracker allows the user to share, rate and discover new games circling the web. We intend to appeal to those in the gaming industry who want a platform to express their opinion and stay informed on older and new video games. With some video game collection apps on the market currently, we expect mild competition and are confident we will be able to build a strong market position.

GG Tracker was created by Siobahn Devlin in 2022 and was inspired by her enjoyment of video games. In 2022, Siobahn was tasked with an assignment to create a mobile application with her team for the Mobile Apps course. Now in 2023, her and her new team are looking to expand on GG Tracker by adding new features, designing a faster and more efficient architecture, and redesigning the appearance for a more captivating and easier ease of use. Along with a motivated team with interests in video games, art, and technology, there is no limit to GG Trackers capabilities.

3 – Competitive Analysis

Main Competitor: GGapp

|  |  |  |
| --- | --- | --- |
|  | GG Tracker (Us) | GGapp (Competitor) |
| Game Library | ✔ | ✔ |
| Create Lists | ✔ | ✔ |
| Add Friends | **✖** | ✔ |
| Profile Customization | ✔ | ✔ |
| Improved Search Options | ✔ | **✖** |
| Comments Sections | **✖** | **✖** |

Comparing our own design with that of our main competitor GGapp (or GG|), we found many of our features to be shared universally. However, a major strength of GG Tracker lies in its improved search options, allowing users to better find the games they want. Originally, we wanted to include user comments in the implementation of this website. We have determined that it is not a feasible option with our current time constraints and technical experience with the backend in a web development environment. Also, the option to add friends did not seem to have many benefits outside of having quicker access to view another user’s profile, which we saw to be redundant. Our app will follow the best features most logically conjured from the base idea of a video game tracker, which so happens to align with other competitors.

4 – Data Definitions

**General Definitions**

|  |  |
| --- | --- |
| **Term** | **Definition** |
| Backlog List | A user’s list of games they own and wish to play |
| Custom List | A custom list that the user can add games to |
| Game | The video game being interacted with |
| Hours | The amount of time a user has played a certain game |
| Password | The password provided by the user and to be stored in the database |
| Profile | A user’s profile featuring their name and list of games |
| Rating | The rating of the game as provided by the API |
| Release Date | The release date of the game |
| Summary | The summary of the game as described by the API |
| Name | The name of the selected game |
| User | The person interacting with GGtracker |
| Username | The username provided by the user and to be stored in the database |
| Wishlist | A list of games the user wishes to purchase in the future |
|  |  |

**See next page for Data Definitions from the API**

**Data Definitions for a Game from the IGDB Database**

|  |  |  |
| --- | --- | --- |
| **Term** | **Data Type** | **Definition** |
| game | Object | Object containing the properties of each game |
| name | String | The name of the game |
| summary | String | A description of the game |
| first\_release\_date | Unix Time Stamp | The first release date for the game |
| artworks | Array of Int | An array of id’s referencing the artwork for the game as defined by the API |
| cover | Int | The id for the cover art of the game |
| genres | Array of Int | An array of integers referencing the Genre as defined by the API |
| platforms | Array of Int | An array of integers referencing the platform as defined by the API |
| rating | Double | Average IGDB user rating for the game |
| videos | Array of Int | The array of Integers referencing video URLs for the game as defined by the API |

**Data Definitions for a Cover from the IGDB Database**

Note: A cover is the “poster” art for a video game

|  |  |  |
| --- | --- | --- |
| **Term** | **Data Type** | **Definition** |
| game | Int | The game this cover is associated with. If empty, then it belongs to the game\_localization id. |
| game\_localization | Int | The game localization the cover might be associated with |
| image\_id | String | The ID of the image used to construct an IGDB image link |
| url | String | The website (url) of the item |
|  |  |  |

5 – Overview, Scenarios, and Use-Cases

GG Tracker is an online app that allows users to discover new video games on the market in real-time as well as see which games are most recently released. Imagine an avid video-game consumer looking for a clean website to search for recent games online where they are met with easy-to-use, efficient search results.

Users can search for games they are interested in playing where they can view their rating, description, screenshots, and trailers. The user can add the game to their own custom lists, a wish list, or their backlog list. For games the user has completed, they can rate the game, mark the game as “playing,” “in progress,” or “completed” to keep track of their backlog, and record how long it took to beat the game.

Users of this web application are expected to have little experience with applications like it, offering simple functionality and ergonomic design. Users are met with an abundance of game options without the feeling of being overwhelmed.

1. Use Case: Login.

This will describe how a user can access their account. The preconditions for this are a stable internet connection, the system being available, and the user has an account. The user will get onto the homepage of the website and click on the login button. There, they will enter their username and password and click login. They will then be redirected to the homepage, but while logged in. An invalid username or password will alert the user as such.

1. Use Case: Create Account.

This allows the user to create a new account in a similar pop-up fashion as logging in. Upon request, the user will be presented with a modular pop-up that asks them to fill in their username and password they wish to use for their account. The password must be entered twice for secure confirmation. When complete, the user is taken back to the homepage, only now they will be logged in.

1. Use case: Homepage Navigation.

Users will be presented with a grid-like structure containing game covers for games based on a specific or general search query/category. By default, the homepage will display games by relevancy/recent publicity, but this will dynamically change based on search queries or categorial selection. Each game cover can take the user to a detailed page containing more information about the game, or it can take the cover image and expand it in detail, creating a slideshow for the user to closely look at the covers for the games in the homepage.

1. Use Case: Create Lists.

The user can go to their account and click on create a list to be able to make a list of games that fit a theme or category of their choosing. In addition to creating a custom list, there is a default wish list the user can add games to. There is also a default backlog list which shows the games the user has already played. They can add games to a list by searching for the game and then clicking on add to list, followed by the list they want to add it to.

1. Use Case: Account Functionality.

The user’s account and account page will have functionality that includes creating lists like the above. Some other features will have the user be able to switch between dark and light mode, view their backlog, and view the lists they have created as well. In addition to these features, the user will also be able to make edits to their profile like change their profile picture as well as their username and password.

1. Use Case: Game Details

Once the user has selected a game from the homepage, they will be presented with an abundance of information. Some of this information includes its name, IMbD rating, ports, initial release date, etc. In addition, a gallery may be presented near the bottom of the page containing a diverse number of official images. Contained within sticky tabs along the sides of the screen, buttons will be presented for the user to add the game to their wish list, backlog, or any other custom list.

6 – High-Level Requirements

1. Priority (1) Login/Logout Functionality: The user can log in and out of their account. They will be able to see their personal backlog lists and favorite games. The user will be presented with a popover modal and the user will be able to either create a new account if needed or they can log into their account if they have already made one. If the user is logged in, they will be able to logout through a logout button on their account page.

2. Priority (1) Main home page will show a list of trending games that the user can click on. The games are arranged on a grid, showing their cover art. When a user hovers over a game, they will have the ability to click on it and be brought to the details page of the game. The user will also be able to hover over a game cover and expand it to see it in more detail.

3. Priority (2) Main home page can sort games based on trending, rating, or system. There will be a navigation bar with a dropdown allowing the user to select the category that they would like to filter by. Example categories would be genre, ratings, release date, and system.

4. Priority (1) When the user clicks on a game on the home page, they will be brought to a details page for the game that will have the following details: game summary, rating, screenshots, videos, and the ability to be added to the user's backlog, the ability to add to the user’s wish list. The details shown will depend on the data received from the IGDB API.

5. Priority (3) Ability to search for games: The user will be able to search for a game in the search bar and view information such as overall rating, description, and systems it is available on.

6. Priority (1) Ability to create custom lists: The user should be able to create a custom list to add games to. For example, this could be a top 10 list for a certain genre the user has played or a custom wish list.

7. Priority (2) User Account Functionality: The user will be able to view their backlog list, wish list, and create new lists / view any created custom lists.

8. Priority (3) Ability to change between dark and light mode: The user should have the ability to change the website’s theme between a dark and a light mode

7 – List of Non-functional Requirements

1. Performance: The web app will be expected to deliver query results to the user in under 2 seconds. Traversing between pages of the web app should be under 1 second.
2. Usability: The web app should be very intuitive for the user. The web app should follow familiar design patterns that other information-based websites use.
3. Accessibility: The web app should be compatible with screen reading technology and adhere to Web Accessibility Initiative recommendations
4. Expected load: Since this is a student project, we do not expect there to be heavy load on the site.
5. Security requirements: we will be using Firebase as a backend which handles login/logout functionality securely. There should be no personally identifiable information other than the user’s name.
6. Storage: little storage will be needed. Storage will be used for saving lists in a JSON format. Data such as images and descriptions of the games will be queried from the API every time they are needed and will not need to be stored in the database.

8 – High Level System Architecture

**High Level System Architecture**

Visual Studio Code: The primary editor of choice for the team. It was created by Microsoft and is a lightweight editor that can have its functionality extended by extensions. We have heavily relied on the Live server features and ESLint extension to enforce code standards.

HTML (Hyper Text Markup Language): We are using HTML as the skeleton for the front end.

CSS (Cascading Style Sheets): CSS will be used with Bootstrap for styling.

Bootstrap5: An open-source CSS framework that has many different templates for different web development components. It has been integral to implementing our front-end design.

JavaScript: The language we are using for all backend logic.

jQuery: A JavaScript library to help with DOM manipulation and AJAX. We are using AJAX to communicate between the front and backend servers.

Node.js: A JavaScript runtime environment that allows the user to easily create servers. We are relying on this technology to drive the backend requests.

Firebase: A set of cloud computing services offered by Google. We are using Firebase to deploy the front end of our website.

Firebase Functions: A subset of Firebase that allows you to run backend code and make network requests. This works together with the frontend hosted on Firebase to run the entire web app.

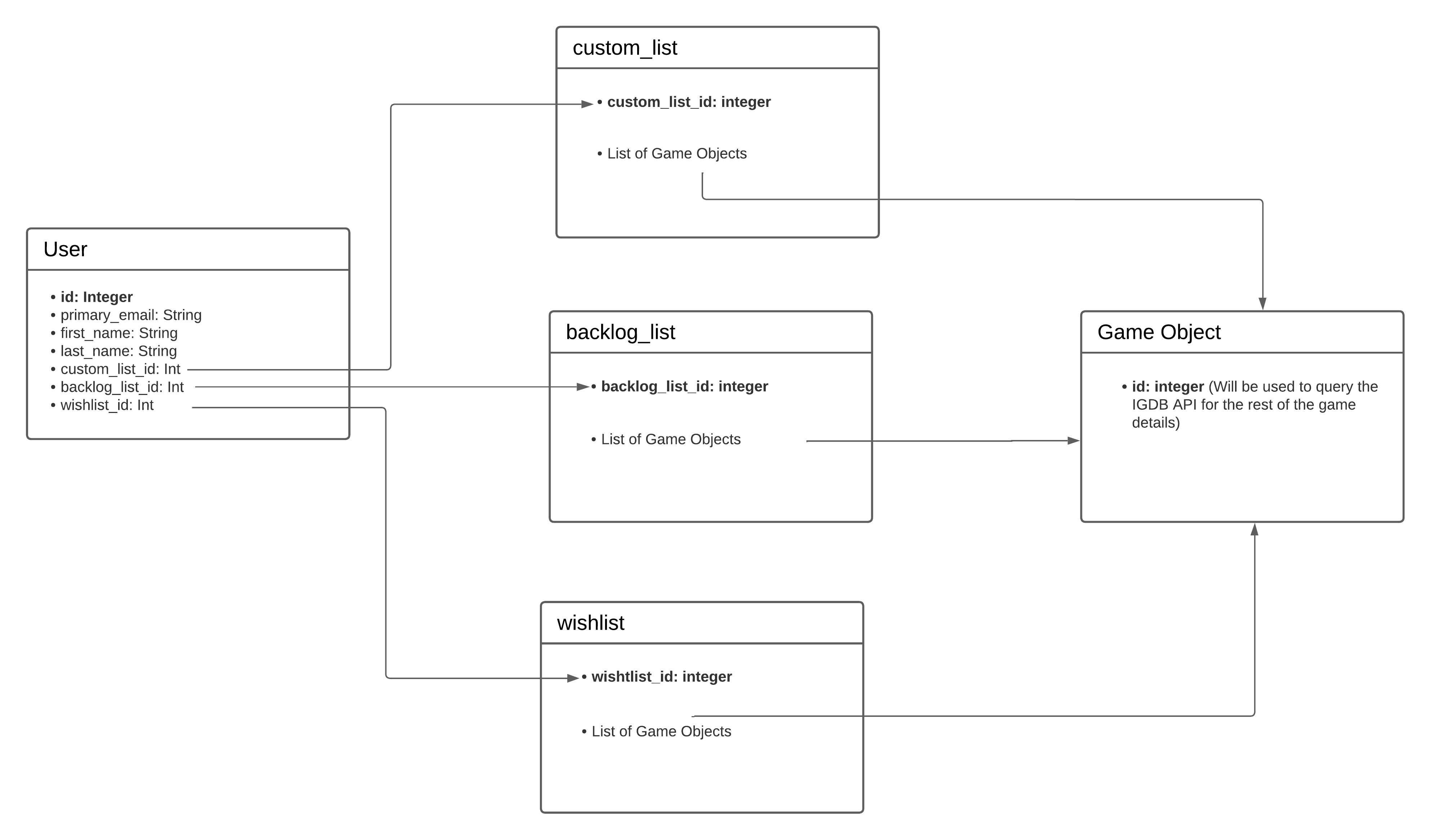
Firestore: A subset of Firebase that is a NoSQL database. It interacts seamlessly with the other Firebase products and is an ideal choice for our database needs.

IGDB Videogame API: The foundation of our app. This API provides extensive data on thousands of video games. It provides all the data needed to run our website.

GitHub: The cloud service used for backing up our code and managing different team member contributions.

JIRA: A task management software in the cloud to help manage tasks among a team.

**Database Organization**: The database has few tables. Each user will have their own entry in the database with their id as assigned by Firestore, an email address used to sign up, a first name, last name, a custom\_list\_id if they have created one, a backlog\_list\_id, and a watchlist\_id. The custom\_list\_id, backlog\_list\_id, and wishlist\_id are primary keys for the custom\_list table, backlock\_list table, and the wishtlist table. Each list will have a list of Game objects in the table that the user will have added to their lists on the site. For simplicity, we will only store the id associated with each game from the IGDB database and use the ids to query the API for any additional information needed to display the data to the user.



**Media Storage**: Images and videos will be queried directly from the IGDB database each time the user accesses a specific game. This allows us to keep our database storage needs with Firestore very small. In firestore, we will only be saving id numbers and strings used to query the IGDB database.

**Search Filter Architecture and Implementation**: The IGDB database has a search endpoint that we will use to implement our search bar. The user will enter the entire search query and then press search/enter. The search query will then be sent to the backend route called /search and an API call will be made to the IGDB database with the query. If there is a match in the API, a game object will be sent back to the front end to be displayed to the user.

**Your Own API:** We are using the IGDB Video Game Database. We did not need to write our own API for this project.

**Algorithms**: There is one major algorithm needed for this project, regarding HTML generation. The algorithm will be used for generating the posters and loading them into the DOM. It goes as follows:

generateHomeScreen():

Step 1: Query the IGDB API for the access token.

Step 2: Retrieve data from the API. This comes back as a list of objects containing the game information.

Step 3. After receiving the list of objects, loop through the list, check the object has a cover id. If it has a cover id, call getCovers() and retrieve the URL. If there is not a cover id, replace the field with “cover not found.”

Step 4. After checking for and generating the cover, create a new game object with the name, summary, release dates, cover URL link, genres, platforms, ratings, and video links.

Step 5. Append this game object to the gamesList

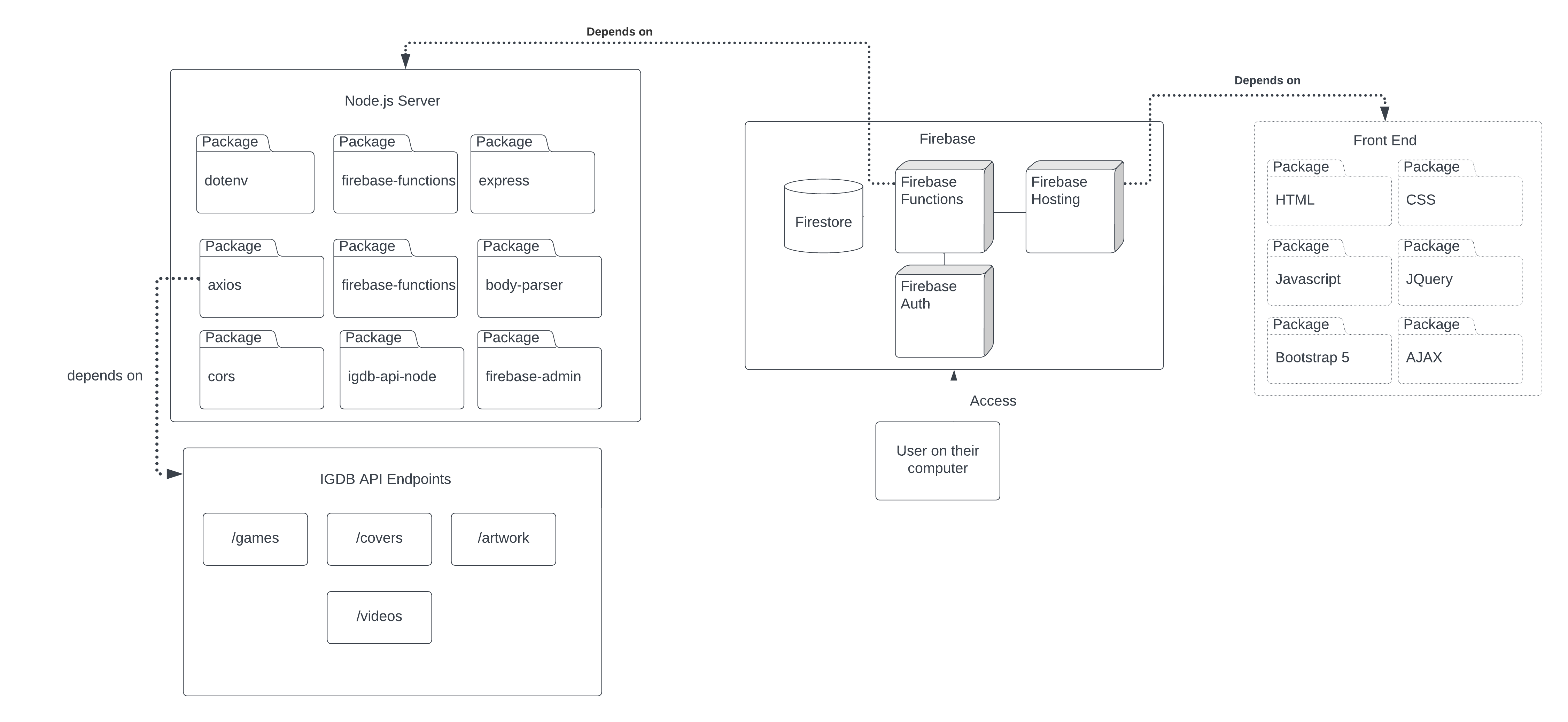
Step 6. Return the gamesList to the front end.

Step 7. In the front end, loop through the list and create a div for each game. Inside each div, place a link to the cover to be generated along with any necessary bootstrap tags from the template.

Step 8. Output newly generated HTML code to the front end.

9 – High Level UML Diagrams

High Level/ Component Diagram



10 – Key Risks

1. **Skill Risks**: (Risk Level: High) The current biggest risk regarding skills is having little knowledge among the team on how to deploy a project, write a backend for a project, and have those pieces working together. The team lead, Siobahn, is the only one with any experience programming asynchronous code at a professional level. However, Siobahn works in iOS and has little experience in web development. Since Siobahn has the most experience with this type of coding, she is responsible for the backend implementation and deployment. While other team members have been helping with debugging, they have little experience with web development. We plan to resolve this issue by having Siobahn teach the entire team the architecture of the project using the vertical slice and referring to documentation as much as possible. The other team members have been familiarizing themselves with Firebase documentation and JavaScript.
2. **Schedule Risks**: (Risk Level: High) The biggest schedule risk we have run into so far was a huge breakdown in the Firebase hosting functionality due to account permissions. While deploying the website, account permissions on Siobahn’s school google account were corrupted and caused errors that were difficult to debug and gave obscure error messages. A whole week was lost figuring out why the hosting stopped working. Since we lost a whole week of time to this bug, it has made delivering the vertical slice extremely difficult. We are resolving the schedule risks by removing features that we do not have the skillset to implement. For example, by removing the comments section of each game, we have removed a considerable time sink on trying to figure out how to implement this in a database.
3. **Technical Risks**: (Risk Level: Medium) The main technical risks come from our unfamiliarity with Firebase and using its services. None of us have implemented anything with Firebase before. Also, we do not currently know of a way to get a detailed error message from the backend if there is an error since it is deployed on the Firebase servers. This makes debugging the backend code difficult and is being done blind.
4. **Teamwork Risks**: (Risk Level: Non-existent) There are very few teamwork risks for this project. All team members are willing to help with problems even if they do not have experience with it. All team members are easy to reach through Discord or email and so far, there have been no problems.
5. **Legal Risks**: (Risk Level: Low) There are currently no legal risks. We are not making any money off this project and the API we are using, IGDB.com, is run through Twitch. Twitch grants us a limited, non-exclusive, worldwide, royalty-free, non-transferable, non-sublicensable, revocable license. This can be found here: <https://www.twitch.tv/p/en/legal/developer-agreement/>