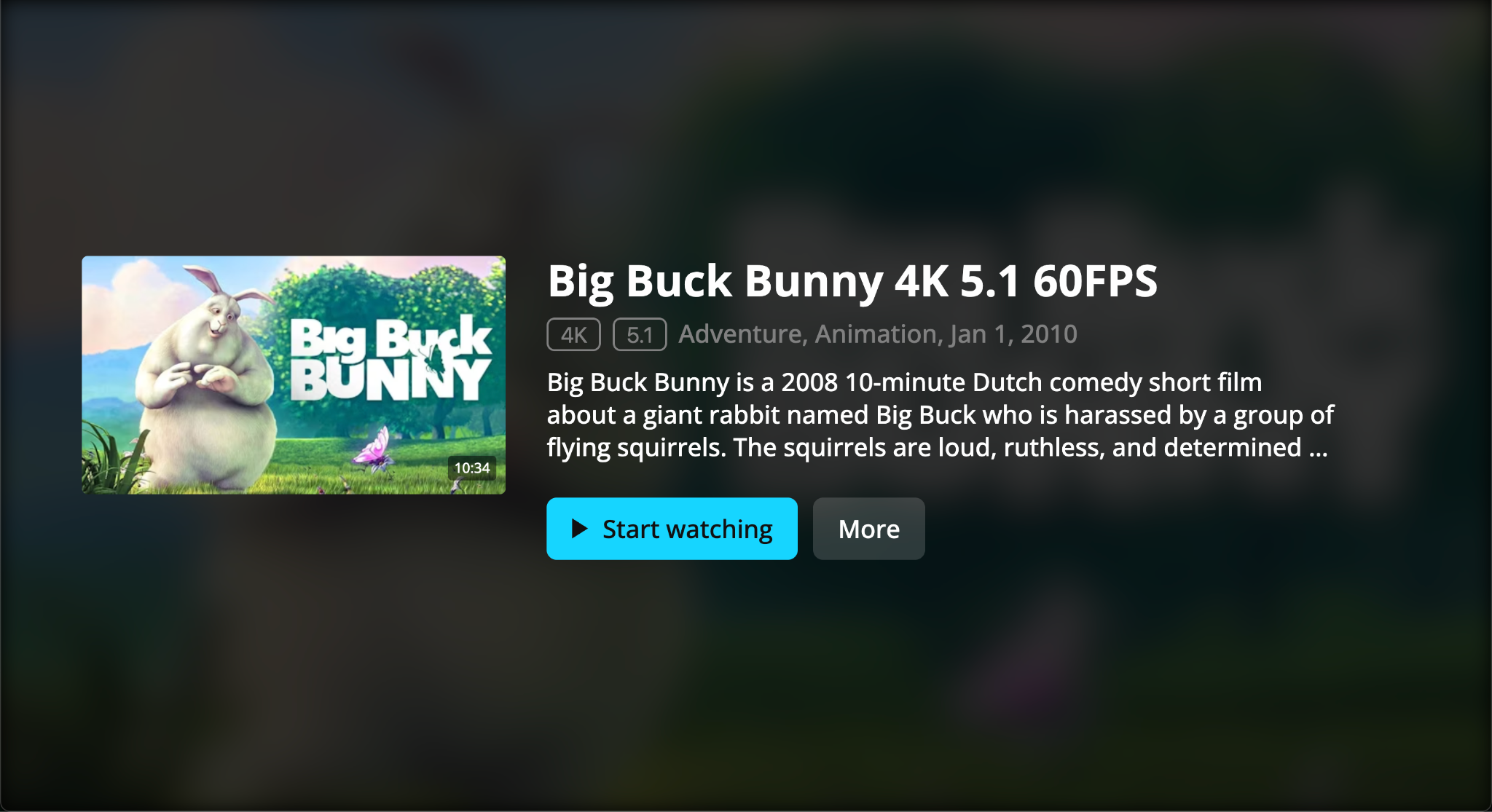
# Technical Challenge: Internet Archive API Facade

## Goal

The objective of this challenge is to design and build a simple backend API service that acts as a facade over the public Internet Archive API. This facade API will be consumed by a hypothetical client application (visual context provided via screenshots) designed for exploring and viewing video content from the Internet Archive.

## 

Explore Screen



Video Details Screen

## Context

You will be working with the [Internet Archive APIs](https://archive.org/developers/). We encourage you to explore their documentation to understand how to search for collections and retrieve metadata and files for specific items. **For this challenge, please focus your implementation on retrieving video content (e.g., items with mediatype:movies).**

We will provide you with:

1. A [Postman collection](https://pastebin.com/bzsyXAK4) containing example requests to the Internet Archive API (e.g., searching for collections, getting the videos/items within a collection). Using this collection is **optional**; you can import it into Postman or use it as a reference for crafting your own requests.
2. Two screenshots of the sample client application that would consume the API you build. These are meant to give you context on how the data might be used visually, helping you inform your API design decisions.

## Requirements

Your task is to implement an API service (using the backend language/framework of your choice) with the following endpoints:

1. **Explore Endpoint (e.g., GET /explore)**:
   * This endpoint should return data suitable for powering the main "explore" screen in the client app, which displays various categories or collections of videos (e.g., "Featured Videos", "Recently Added Movies", "Top Collections").
   * Consider how the client application will use this data. Your response should allow the client to display these thematic rows. Think about whether each row in the response should contain its list of video items directly, or if it should provide identifiers/information for the client to fetch the items separately using another endpoint.
2. **Collection Items Endpoint (e.g., GET /collections/{collection\_id}/items or similar)**:
   * This endpoint should return the list of video items belonging to a specific Internet Archive collection/identifier, potentially with pagination if you deem it necessary.
3. **Video Details Endpoint (e.g., GET /videos/{video\_id})**:
   * Given an Internet Archive item identifier for a video, this endpoint should return detailed information about that video.
   * **Required:** Include metadata such as title, description, publication date, creator, etc.
   * **Required:** Include information necessary to display a thumbnail image for the video.
   * **Playback URL Decision:** You need to decide whether to include video playback URL(s) directly in *this* response or use a dedicated endpoint (see below). Please be prepared to explain your choice (e.g., in your README).
4. **Playback URL Endpoint (Optional, e.g., GET /videos/{video\_id}/play)**:
   * As an alternative to including playback URLs in the details endpoint, you could implement this separate endpoint. Choose the approach you think is most appropriate for the API's design.

## Key Considerations

* **API Contract & Design:** A significant part of this challenge is designing a logical and consistent API contract (endpoint paths, request/response formats). Your design choices should serve the needs implied by the client app screenshots and consider client usability.
* **Internet Archive API Interaction:** You will need to figure out how to effectively use the Internet Archive APIs to fetch the necessary video data for your facade API.
* **Error Handling:** Your API should handle potential errors gracefully. Consider scenarios like invalid identifiers or issues communicating with the Internet Archive API, and return appropriate responses (e.g., meaningful status codes and error messages).
* **Technology:** You are free to use any backend programming language and framework you are comfortable with (e.g., Node.js, Python/Flask/FastAPI, Ruby/Rails, PHP/Laravel, Go, etc.).
* **Use of AI Tools:** You are permitted to use AI assistance tools (like ChatGPT, Gemini, Copilot, etc.) during this challenge. If you do use such tools, please briefly document *how* you used them in your README.md (e.g., "Used ChatGPT to generate boilerplate code for Flask routes", "Asked Gemini for alternative ways to parse the IA API response", "Used Copilot for code completion"). Transparency is key.

## Deliverables

* Source code for your API service, preferably shared via a Git repository (e.g., GitHub, GitLab).
* Clear instructions in a README.md file on how to set up and run your API service locally. Include any significant design decisions or trade-offs you made (particularly regarding /explore structure and playback URLs), and a section detailing any use of AI tools as described above.
* **(Optional)** API documentation for the endpoints you created. This could be in the form of a Postman collection, OpenAPI/Swagger specification, or simply clear descriptions within the README.