GamesHub: A Web-based Game Sharing Platform

Abstract

GamesHub is a web application I developed for sharing and playing browser-based games. The platform allows developers to upload their games and players to try them instantly without downloading or installing anything. I built the backend using C# and ASP.NET Core and SQLite for the database, while the frontend uses Angular with Ionic for responsive design. The system includes user registration and login, game uploading with cover images, search and filtering features. Game files are stored locally on my computer during development, which keeps things simple and voids cloud storage costs. The report discusses how I built the platform, the technologies I used, the challenged I faced, and what could be improved in the future.

1. Introduction

1.1 Project Background

The idea for GamesHub came from my interest in game development at the time and how modern game engines like Unity and Godot can export games to run directly in web browsers. I was fascinated by how players could instantly try complex and well-built games without downloading anything, just by clicking a link. Platforms like Itch.io showed me that sharing games online doesn't have to be complicated, and I wanted to create my own version to learn full-stack web development.

During my internship, I had learned Angular and wanted to apply those skills in a real project. I also wanted to understand how file uploads work, how to manage users accounts, and how to build a complete web application from scratch. GamesHub became my learning project where I could combine all these interests into something practical and fun

1.2 Project goals

My main goal was to create a simple platform where:

- Developers could easily upload their indie games
- Played could browse and play games instantly
- Users could create accounts and manage their content
- The interface would work well on both computers and phones

I wanted to keep everything straight forward without overcomplicating the features. The focus was on learning how to build a working application rather than creating a commercial product. This meant choosing technologies I could understand and work with easily, like SQLite instead of a complex data base system, and local file storage instead of cloud services.

2. Related Work and Inspiration

2.1 Existing platforms

My biggest inspiration was itch.io, a popular platform for indie game developers. What I liked about it was how it made sharing games simple and accessible. Developers could upload their games, add description and images and players could try them right away. I wanted to capture the simplicity in my own project.

I also looked at other platforms like Kongregate and Newgrounds, which have been around for years hosting browser games. These sites showed me common features that players and developers expected to have, such as browsing by category, searching for specific games, and seeing which games are most popular. However, I noticed these platforms can be overwhelming with too many features, ads, and complex interfaces.

2.2 My Approach

Instead of trying to copy everything these platforms do, I decided to focus on core features that would make a function game sharing site. I wanted GamesHub to be clean and simple without ads, complex monetization systems or social media features. The goal was to understand the fundamental mechanics of how such platforms work, not to compete with established sites.

3. System Design and Architecture

3.1 Overall Structure

GamesHub follows a typical client server architecture. The backend handles all the data management, file storage and business logic, while the frontend provides the user interface. They communicate through REST API endpoints that send and receive JSON data.

I chose this structure it is widely used in web development and there are plenty of resources to learn and get help from. It also keeps the frontend and backend separate, which makes it easier to work on each part independently and potentially replace one without affecting the other.

3.2 Backend Design

The backend is built in ASP.NET Core using C#. I chose this because I was already familiar with C# from previous courses and projects and wanted to deepen my knowledge. For the database I used SQLite with Entity Framework as the ORM tool. SQLite was a perfect fit for this project because:

- It is just a single file, making it easy to move the database around
- It doesn't require installing a separate database server
- Its simple to set up and use during development
- Entity Framework generates all the SQL queries for me

The data base has four main tables:

- Users: Stores usernames and hashed passwords
- Games: Stores game information like name, description, file paths, upload date, genre, click count
- Categories: Stores the different game genres
- High scores: Stores the high scores for the games, consisted of player name and score

I store passwords using encryption for basic security, though I know this could be improved in a production environment. When users log in the system check their credentials and maintains their session.

For file storage, I simple save uploaded games in folders on my local computer, each game gets its own folder. This approach works well for development and testing, though obviously it wouldn't scare for real development and cloud storage would be much needed.

3.3 Frontend Design

The front end uses Angular with Ionic components. I chose angular because I recently learning it during my internship and wanted to practice more. Ionic helped me make the interface responsive without writing a lot of custom CSS for different screen sizes and components.

The main components include:

- Login and Register page

- Home page: Displays the most recent games and the top 10 games
- Game details: Place where users can play the game and get full description and high scores for the game
- Upload form: Allows developers to upload their games
- Search and filters: tools to find specific games

I also integrated Quill, a rich text editor, so developers can format their game description with bold text, lists, and other styling options. This makes game pages look more professional that plain text descriptions.

4. Implementation Details

4.1 User Management

The registration process is straight forward. Users provide an email, username and passwords, which gets saved to the database with the password encrypted. For the login the system compares the provided credentials with stored data. I keep track of logged in user using session storage, though this is quite basic compared to modern authentication systems like JWT tokens.

One change was making usernames are unique and handling case where someone tries to register with an existing username. I also had to ensure that only logged in users could upload games or delete their own games.

4.2 Game Upload Process

When a developer uploads a game, they provide:

- A ZIP file containing the game files
- Game name and description
- A cover image
- The display dimensions (width and height)
- The games categories / genres

The backend receives the ZIP file, extracts it to a designated folder and saves the metadata to the database. The cover image is stored separately and linked to the game record. I didn’t implement extensive validation, which is something that should be improved. Currently the system trusts that the users uploads are safe and actually games and not malicious content.

One interesting feature is that developers can specify the displays size for their game. This is important because different games might be designed for different resolutions, and forcing them all to the same size could break or hinder the gameplay experience which is of the utmost importance to players and developers.

4. 3 Browsing and Search Features

On the left side of the home page the most recent games are displayed as cards with their cover image, title, developer and a brief of the description. On the right side there is a list of the Top ten most popular games calculated by clicks on each game. If we chose browse games in the header we will go to the search and filter page. Here we can search games by their name and filter out categories like: RPG, Shooter, MMO, Adventure, Puzzle etc... Here only part of the games are loaded initially, to see more we can scroll down and activate the infinite scroll, this works by requesting games in batches using skip take in the backend to increase performance. Every time someone clicks on a game to play it, the games click count increases. This provides a simple metric without needing a complex rating system.

4.4 Playing Games

When a user clicks on a game card, the game opens in an embedded iframe that height and width are previously decided by the developer while he uploads the game. The game files are server from the local storage folders (cloud storage in the future). Since these are web exported games (HTML5, WebGL, WASM) they run directly in the browser without needing any other tools and / or plugins.

There is also a high score systems where scores can be submitted through an API call by the developer. On the game page we can see the high scores or the full description of the game that is made with a rich text editor.

5. Challenges and Learning Experiences

5.1 Technical challenges

The biggest challenge for me was learning a frontend framework while building the project. I initially started with React but during the summer I started an internship where I worked with Angular and Ionic, and decided to remake my front end from start using my newly learning skills. This proved to be a smart decision in the end.

Another hurdle was file uploads. I had to understand how the file uploading process works and where are the files stored, they cant be put in a simple database. There were some issues with the whole zipping and unzipping process the creation of subfolders and proper folder structure per user and game.

Making the interface responsive was also trickier than I initially expected. Even with Ionics help I had do adjust some stuff so everything looks good on desktop and mobile devices. The infinite scrolling feature also required careful implementation to avoid performance issues and loading duplicates.

6. Results and Current State

6.1 Working features

GamesHub successfully implements core functionalities like:

- Users can register and log in
- Developers can upload games with description and cover images
- Players can browse, search and filter games
- Games can be played directly in the browser
- The interface works well on desktop and mobile devices
- Infinite scrolling provides smooth browsing
- Click tracking shows game popularity

The platform demonstrates that it is possible to build a functional game sharing website with modern technologies. While its not ready for public deployment, it works well for its intended purposes.

6.2 Limitations

The current version has several limitations:

- No validation of uploaded files (security risks)
- Basic authentication without advanced security features
- No ability to edit accounts, except deleting uploaded games
- No comments, ratings, or reviews
- Search only works on game names, not description
- No moderation tools for inappropriate content
- Files stored locally, not suitable for real deployment
- No backup system for the database or files

These limitation don’t prevent the platform from functioning but would need to be addressed before production or a public release

7. Future Improvements

7.1 Security Enhancements

The first priority for improvement would be security:

- Implementing file validation
- Implementing CAPTCHA for registration to prevent bots
- Add rate limiting to prevent abuse
- Improve session management with proper tokens

7.2 Feature Additions

Several features would make the platform more complete

- Enhanced use profiles with avatar images and bio
- Comments and reviews on games
- Rating system (starts or likes)
- Follow system for developers
- Email notifications for new games from followed developers
- Better search that includes tags and description
- Game analytics for developers
- Ability to edit game information after upload

7.3 Technical Improvements

For deployment and scalability

- Move to game folders to a cloud storage
- Use a proper database system
- Implement caching for better performance
- Add automated backups
- Create an admin panel for moderation

- Implement proper logging and error tracking
- Add automated testing and a CI/CD pipeline

7.4 User Experience Enhancements

To make the platform more enjoyable

- Improve the upload interface by adding a progress bar
- Add game recommendations based on play history
- Create collections or playlists
- Implement achievements or badges for players and developers
- Add social sharing buttons
- Create a better game player with full screen support
- Add keyboard shortcuts support

8. Conclusions

Building GamesHub has been an invaluable learning experience in full stack web development. The project successfully demonstrates how modern technologies can create a functional platform for sharing and playing browser based games. Through this project, I gained practical experience with ASP.NET core, Entity Framework, Angular, Ionic and various other tools and libraries.

While the current implementation has limitations that prevent it from being production ready, it achieves its primary goal as a learning project. I now understand how file uploads work, how to manage user authentication, how to design REST APIs, and how to create a responsive web interface, The challenges I faced, from switching frameworks mid project to implementing various stuff, though me problem solving skills that will be valuable in future projects.

The Simplicity of the current implementation, using SQLite and local file storage, made it possible to focus on learning core concepts without getting overwhelmed by complex infrastructure. These choices, while not suitable for production, were perfect for a student project where the goal is understanding rather than scale.

Looking forward, GamesHub provides a solid foundation that could be expanded into a more complete platform. The modular architecture makes it easy to swap out components. For example replacing local with cloud storage or upgrading the

authentication system. The clean separation between frontend and backend means either could be rewritten without affecting the other.

This project reinforcement my interests in Web Development, it showed me that creating platforms to help others share their work is both challenging and rewarding. Whether or not GamesHub ever becomes a public platform, the skills and knowledge I gained from building it will certainly influence my future projects.