International Journal
Information Technology Applications (ITA)
Volume 13, Number 1, September 2024

INTERACTIVE WEB GAME

Michal Heban

Abstract:

This thesis deals with the development of a computer game, specifically a turn-based strategy adventure game, in which the player controls his character in a web-based environment. By harnessing the potential of web technologies, we seek to overcome the traditional barriers of computer games by enabling play anywhere. Thanks to the accessibility of the Internet, players can play the game on their favorite devices, regardless of location or time. The key concepts of our game will be uncovering maps, collecting a variety of gear, creating unique items, and engaging in battles using spells and various tactical elements. These elements are designed to provide a deep and immersive experience, encouraging players to strategize and plan their moves carefully. By designing comprehensive functionality and attractive visual design enriched with 2D graphical elements, we aim to create a visually appealing environment that enhances the overall player experience. The game's art style and user interface have been carefully crafted to ensure they are both engaging and intuitive, allowing players to easily navigate and enjoy the game.

Keywords:

React, game, pixel art, adventure, Next.js

Introduction

In the realm of computer gaming, we are witnessing a continual rise in popularity, driven by increased interest in interactive forms of leisure, often revolving around digital entertainment. With the advancement of the internet and its widespread availability, the use of web platforms for gaming is becoming increasingly common. The advantages of these platforms include the opportunity to play games without the need for downloading or installing necessary software.

Therefore, in our work, we will focus on the creation of an interactive RPG turn-based strategy implemented in a web environment. We will cover multiple domains starting with theoretical foundations essential for game development.

Following the theoretical groundwork, our next step involves determining the genre and style, refining the identity and aesthetics of the game. Subsequent to this stage, we move on to the practical process of prototyping various screens and user interfaces to materialize conceptual ideas.

After completing the prototyping phase our focus will shift towards the implementation stage, where we translate design into functional components, bringing our interactive web game to life within the web environment. Lastly we will conduct thorough testing to ensure the smooth functioning of all components, verifying that each element operates seamlessly withing the game.

1 Web Development

With the rise of web applications, the way people interact with software has undergone a significant transformation. Unlike traditional native applications that require installation and operate locally on specific operating systems, web applications only necessitate a functional web browser and internet access, regardless of the operating system, as they function on remote servers. This connectivity enables access from various devices, ranging from smartwatches and smartphones to computers and smart TVs.

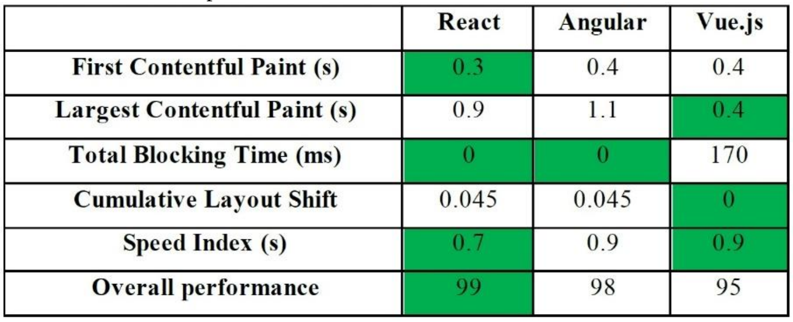
The web application operates on the client-server principle and consists of two main parts: frontend and backend. The frontend corresponds to the client-side, which focuses on the user and is responsible for the visual content of the application and its interaction with the user. The primary element is the graphical user interface (GUI), which includes buttons, multimedia content, and forms.

The backend, the server-side component of the application, is responsible for the application logic, data storage, and all associated calculations. It commonly communicates with the front end through HTTP requests, which are processed on the server side and returned to the client side as a response [1]. In some cases, the server may further connect to data sources or other services necessary for request processing.

In today's web application development, various frameworks are commonly used to assist programmers in building applications. These frameworks offer numerous advantages throughout the development process. They provide consistency, efficiency, maintainability, and scalability for the project.

Most popular frontend frameworks are React, Angular and Vue.js, where React is most popular among them according to trends on StackOverflow. In terms of performance, React also emerged as the winner. The comparison was conducted on three identical pages, where each was created in corresponding framework [2,3,4]. The comparison was utilizing Core Web Vital metrics obtained from the PageSpeed Insight tool. This includes First Contentful Paint, Largest Contentful Paint, Total Blocking Time, Cumulative Layout Shift, and Speed Index. The result can be seen in the table below (Tab.1).

Table 1. Comparison of Core Web Vitals for frontend frameworks.



	React	Angular	Vue.js
First Contentful Paint (s)	0.3	0.4	0.4
Largest Contentful Paint (s)	0.9	1.1	0.4
Total Blocking Time (ms)	0	0	170
Cumulative Layout Shift	0.045	0.045	0
Speed Index (s)	0.7	0.9	0.9
Overall performance	99	98	95

2 Main Concepts of the Game

In game development, genre and style are pivotal choices. We've opted for an adventurous RPG set in a fictional world dominated by magic. As for style, we're embracing a text-based approach with pixel graphics for a retro vibe, fostering player imagination. With a dark theme consisting of black background and white text, the game will be known as "Adventurer Chronicles".

Whole gameplay will be centered around player’s character, where player can choose from three classes, with each class offering three specializations. First class will be Warrior which is divided into these specializations: Guardian knight, Berserker and Shield Bearer.

Second class will be Mage, with specializations for being Elemental mage, Demonomancer or Druid. Lastly there will be Master of Weapons which offer these specializations: Sword Master, Sharpshooter, Assassin.

The game world will be expansive, comprising regions with towns which will be used for player to rest, get new gear, train new spells or taking quests. On other hand for battle and adventure there will be multiple locations, each with its own set of monsters specific to that area.

Attributes such as strength, agility, intelligence, endurance, and vitality further define the player's capabilities. Alongside these attributes, a player's strength will be determined by their level, equipment, and spells, which can vary in level and quality. Higher quality items provide greater strength. Players can advance their level by accumulating the necessary experience for leveling up. Attributes can be enhanced either through leveling up or by consuming specific items tailored for this purpose. Equipment can be acquired by defeating monsters, unlocking treasure chests, or purchasing them from in-game shops. Additionally, players will have the option to craft their own equipment and items using professions like alchemy or blacksmithing, further enriching the gameplay experience.

In the game, there are two types of currency: basic and premium. Basic currency is used for all aspects of the game, from buying items to training spells or abilities. It can be obtained through missions, looting monsters, or selling unwanted gear or items. Basic currency includes bronze, silver and gold coins. Premium currency, acquired from specific powerful monsters, is used for purchasing superior bonus items or enhancing player abilities. Premium currency includes diamonds for exclusive items and magic stones for exchanging or upgrading items.

Materials are items used in crafting equipment or consumable items. Players can acquire them through various means, similar to spells or equipment. They can be obtained from shops, missions, or as loot from monsters. Collecting materials from specific locations will also be a significant method of acquisition.

Each player will have access to three basic resources. Experience directly influences the player's level, serving as the main indicator of the player's success and progress in the game. Among the active resources crucial for the player's survival and success will be health and mana. Health represents the player's physical condition and determines how much damage the player can withstand before being defeated. Players will lose health during battles with monsters and will have the opportunity to replenish it using various items or actions. Mana is a mysterious energy serving as the fuel for all spells. Different levels of spells require different amounts of mana. Similar to health, mana can be replenished using items or actions.

Spells will serve as the primary resource for combat and will play a significant role in the game. Each spell will have its own level, which can be further enhanced to increase its potency. Additionally, spells can be upgraded to unlock higher tiers, making them even more powerful. However, casting spells will consume mana.

3 Prototypes and Data Model

Creating prototypes for game screens and defining a data model are vital steps in game development. They enable developers to gather feedback before final implementation, ensuring the final product meets expectations. Additionally, prototypes help maintain workflow continuity by providing clear direction for developers, reducing uncertainty about the next steps in the development process.

3.1 Prototypes

To create prototypes for our game, we utilized the MockFlow tool. We designed prototypes for all important sections of our game starting with home screen.

The home screen will serve as an introduction to the game, providing users with information about the game and options to either register or log in. Both the registration and login pages will feature simple forms.

Upon logging in, users will be directed to the dashboard, where they'll find comprehensive details about their character, including name, class, specialization, level, experience, health, and mana. Additionally, the dashboard will feature sections dedicated to character attributes, active spells, and in-game currencies (Fig.1). For new users without a created character, an option to do so will be provided.

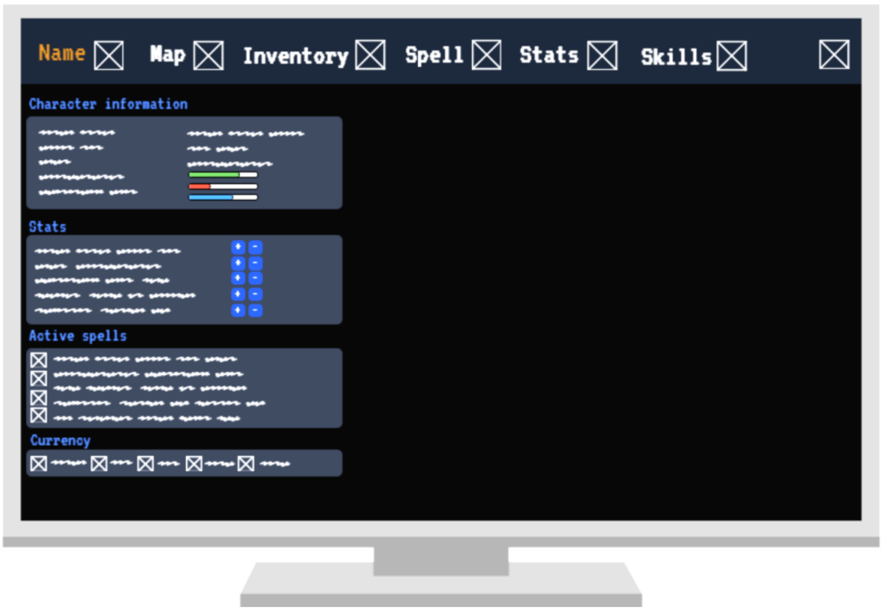


Fig.1. Prototype for dashboard together with header.

The next screen is the map screen. The map is divided into two main parts: crossroads and specific towns or locations. At the crossroads, players are located if they have not entered any town or location yet. On crossroad the player has the option to choose new location or town to travel to or to enter the current place. If the player is in a town, a brief description of the town along with its buildings will be displayed. Each building has its own screen, including its name, description, and actions based on the type of building. Buildings include, for example, the town hall for accepting and handing out quests, the training hall for learning and practicing spells, shops, or professional halls. The location contains a list of enemies along with the option to battle or gather resources.

The inventory will be divided into four main sections, which look almost identical, all having a filter along with a table of items or equipment that the player owns. These sections are for equipment, items, spells and materials.

On the spells screen, the player will switch between two sections: active spells and all learned spells. Active spells show how many spells the player can still activate, their level, name along with the number of experience points, and information about the spell. It also allows the player to upgrade the spell if all conditions are met. All learned spells will include a filter and the option to set some spells as active.

The attributes tab will serve to provide a more detailed view of all attributes, both primary and secondary, along with all character bonuses.

The appearance for each profession will be the same, differing only in text. Within each profession, there will be further categorization into research and crafting. Crafting will display a list of recipes with their levels, materials needed for their production, and crafting time. Below the recipes, there will be a current production list with completion times. In the research section, players will have the option to select materials and attempt to discover new recipes by clicking the research button.

3.2 Data Model

The data model for our game is expected to be quite robust, with an initial estimate of around eighty tables, some of which may contain over ten columns. We used the QuickDBD tool for creating the data model. The data model can be divided into several components.

The first set revolves around the player's character and the player themselves. It includes the table for the player's character, containing fields such as character name, class, specialization, level, experience, current health or mana, number of active spells, and location. This table also connects to several others. Among these tables is one for the player's currency status. There's also an attributes table, recording their values along with the number of points the character has available for allocation. Following that is the spells table, which holds information about the player's spells, including their level, experience, grade, spell status, and availability. While this table doesn't include other spell data, it references another table for that information. Similarly, a table for the player's equipment, materials, or items will be associated with them.

The subsequent set of tables is related to the game world. This may include records of cities, regions, locations, as well as buildings in the city.

Next in line are tables dealing with professions. The first one will contain data about players and their professions. The second will store information about the player's learned recipes or guides. The last set will comprise the list of products the player has entered for crafting. Auxiliary tables in this case will include the recipes themselves and the materials needed for their production.

4 Implementation

Before diving into implementation, it's crucial to choose some fundamental elements such as development environments, technologies, libraries, and programs we'll use for development. We'll start with the development environment. Given our experience with JetBrains development environments, we've decided to use WebStorm for web application development and DataGrip for database management. As a framework for building the application, we've opted for Next.js with TypeScript. Next.js is a library built upon React and provides many methods for server-side rendering, which can significantly boost the performance of our game. For styling, we've chosen the Tailwind CSS library, which allows us to quickly incorporate CSS classes for any element and achieve the desired appearance.

For code cleanliness and adherence to set rules, we relied on the ESLint library. For authentication and authorization, we utilized the NextAuth library, which seamlessly integrates into a Next.js project. Input field and form validation were handled using the zod library. For data handling we chose a PostgreSQL database and utilized the Prisma ORM library. Graphic elements were created using the Aseprite program.

The server-side portion was implemented using the serverless architecture provided by Next.js. Finally the whole application will be deployed to the Vercel platform for its simplicity in deploying Next.js projects, as it's a framework developed by Vercel.

All the screens were implemented based on visuals from prototyping, with minimal to no changes.

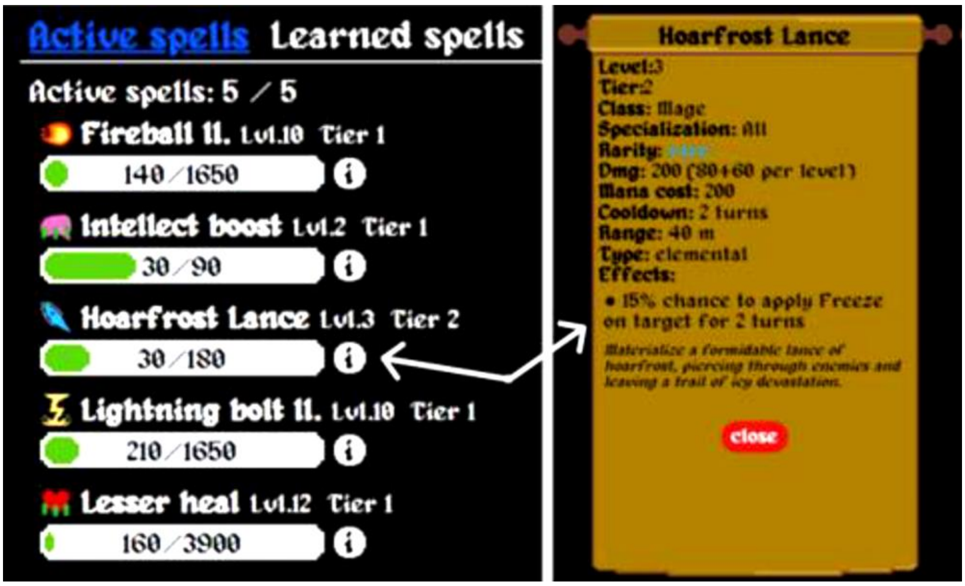
Data handling from the server was managed using the fetch function along with getServer-SideProps from the Next.js library, facilitating server-side rendering when needed. Some data which were used throughout the entire application was stored using the context functionality, allowing access from anywhere within our game. This was particularly beneficial for managing character information such as experience, health, mana, spells or equipment. Additionally, it ensured a consistent and efficient data flow, enhancing overall performance and user experience.

For registration and user sessions, we utilized the authentication function from the NextAuth library. This library not only stores data from the registration process but also manages user sessions, ensuring that players do not have to log in every time they return to the game. By leveraging the credential-based method for login, we provided a secure and seamless experience. This approach simplifies user access and enhances the overall usability of the game.

Overall performance was improved by utilizing memorization techniques, which helped to reduce redundant calculations and enhance efficiency. This optimization was particularly beneficial for handling frequently accessed data and rendering complex components. In our game it was used for data table as material or equipment list in inventory or some data calculations.

Navigation throughout the application was achieved using the routing capabilities provided by Next.js. This system utilizes the folder structure in the project, where every folder within the main app directory that contains a page.tsx file is recognized as a route. This approach simplifies the routing process and helps maintain an organized and easily navigable codebase. A similar approach is employed for the serverless architecture, where each folder within the api directory serves as an endpoint for API calls. Within these folders are files named route.ts, containing the logic executed when an API call to this endpoint is made.

We developed several reusable components for various aspects of the application, such as filters or custom bars for resource management. Additionally, each spell, material, equipment or building had a corresponding pixel art icon, enhancing the visual depth of our text-based game. Not only icon but also some bigger elements had its own art, for example scroll which shows spell information (Fig.2).



Active spells: 5 / 5

Fireball II. Lvl.10 Tier 1
140 / 1650

Intellect boost Lvl.2 Tier 1
30 / 90

Hoarfrost Lance Lvl.3 Tier 2
30 / 180

Lightning bolt II. Lvl.10 Tier 1
210 / 1650

Lesser heal Lvl.12 Tier 1
160 / 3900

Hoarfrost Lance
Level:3
Tier:2
Class: Mage
Specialization: All
Rarity:
Dmg: 200 (80+60 per level)
Mana cost: 200
Cooldown: 2 turns
Range: 40 m
Type: elemental
Effects:
- 15% chance to apply Freeze on target for 2 turns

Materialize a formidable lance of hoarfrost, piercing through enemies and leaving a trail of icy devastation.

Fig.2. Art style example of scroll with spell information and spell icons.

The main combat mechanics were implemented based on the principles of a turn-based game. Each side, whether player or monster, would take turns using spells, abilities, or items. Monster abilities were chosen dynamically based on an algorithm that selected the most suitable spell for the situation at that moment. The outcome of the fight was also influenced by buffs and debuffs that could be applied using spells or items. The battle concluded when the health of one side reached zero or below.

5 Testing

The entire application underwent manual testing, where we tested the complete functionality and addressed any errors. Additionally, we conducted usability testing with ten respondents playing the game on a test server. Two of them had no experience with gaming, three were occasional gamers and five were gaming enthusiasts.

Based on their feedback, we fine-tuned aspects of the game where issues or inconsistencies were found. The most significant concern among players, especially those with less experience, was the lack of a robust tutorial. Since the game featured numerous sections with actions, not all were thoroughly explained. Some players expected a confirmation email after registration, but this feature was not part of the implementation and was not added at the time. Four out of ten players expressed dissatisfaction with the limited number of items, spells, and equipment in the game, so additional content was developed. Crafting these items was time-consuming as all graphical assets were created manually. While this aspect did not affect gameplay, it enhanced the overall player experience.

By adhering closely to our concepts and prototypes without deviation, we encountered minimal errors throughout the development process.

Conclusion

The browser-based game Adventurer Chronicles combines elements of a text-based game enriched with 2D graphics, allowing players to control their character, complete tasks, collect spells, equipment, items and engage in turn-based combat with enemies. All project objectives, such as the game's appearance, functionality, and theme, were successfully achieved, as defined during the conceptualization process, including the design of individual screens and corresponding user interfaces. The resulting game is fully playable on all devices, from mobile phones to computers and smart TVs with internet access, without the need for additional downloads or installations.

Acknowledgement

When working on this project, I would like to express my gratitude to Mr. RNDr. Ján Lacko, PhD. who provided me with valuable advice and consultations throughout the process.

References

[1] Kornienko, D.V. et al. The Single Page Application architecture when developing secure Web services. In Journal of Physics: Conference Series 2021. Vol. 2091, no. 1, s. 012065.

[2] Vue Landing Page. Retrieved online, May 18, 2024 https://logrocket-vue-landing-page.ver-cel.app

[3] React App. Retrieved online, May 18, 2024 https://logrocket-react-landing-page.vercel.app

[4] LogrocketAngular. Retrieved online, May 18, 2024 https://logrocket-angular-landing-page.vercel.app

Authors

Bc. Michal Heban
Faculty of Informatics,
Pan-European University in Bratislava, Slovakia
michal.heban52@gmail.com

Student of Faculty of Informatics on Pan-European University, with extensive knowledge in modern web development.