Realizarea unui platforme de învățare bazată pe gamification (jeopardy) pentru asimilarea cunoștințelor aferente Modulelor 7-8

made by students from 321AC:

Tica Simeon (Project manager)

Bolohan Cristian

Zarnescu Dragos

Gavrila Adrian

Munteanu Vlad

Contents

Project Overview

We proposed to do a full stack application starting from the idea of the Jeopardy game where we are going to implement the logic behind using c++ and javascript.

The project aims to contribute to the educational digitization initiative by implementing a dedicated platform that serves as a new method of learning, specifically through gamification.

By developing a dedicated application and incorporating modern design ideas that align with the expectations of the new generation in the field of education, we will bridge the gap between traditional teaching methods and the highly promising and appreciated methods of the future, as endorsed by students, pupils, and teachers alike.

Motivation behind this project

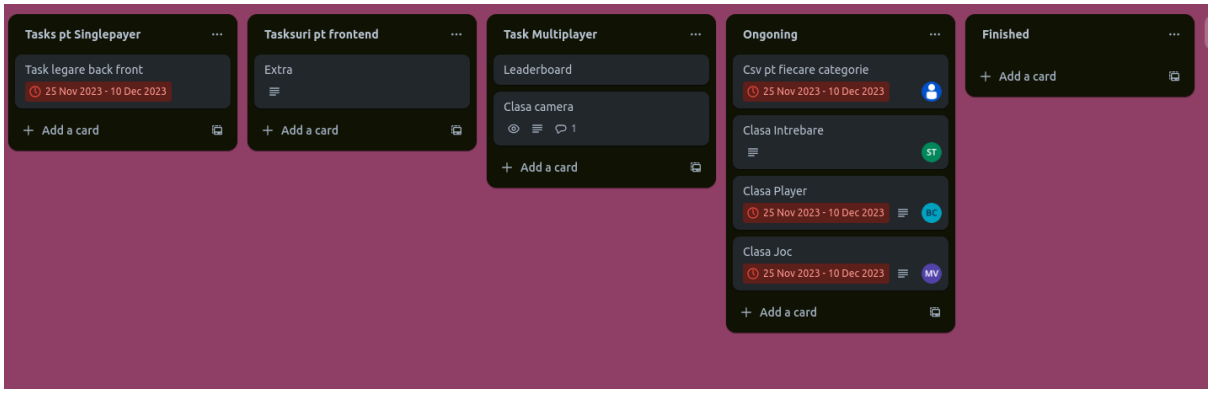
The choice of the project theme, "Jeopardy!", and the development of a platform that leverages gamification, stem from an observation of the current educational environment. Inspired by the concept presented by "Kahoot!", we opted for the "Jeopardy!" theme because it is an engaging trivia game where students can showcase the knowledge they have accumulated throughout their educational journey.

It is crucial to note that the format of this game, inspired by the well-known game show format in America, plays a significant role in honing one of the most important skills: critical thinking. Lastly, this theme sparks active participation from students by introducing an element of entertainment into the academic realm, thus easing the effort required to capture their interest.

The Team behind

We are an enthusiastic team of 5 members of 2nd year students in the Automatic and Computer Science Faculty that seek from this project the most of what they can learn in OOP class.

We have organized on Trello platform but anyone contributed where they felt like he could change.



The managing part of the project was done by Simeon Tica where he coordinated us in a way to build this project entirely.

Documentation

Why gamification?

Gamification in education refers to the integration of game elements and principles into the learning environment to enhance engagement, motivation, and overall educational outcomes. The decision to embrace gamification in educational settings is driven by a multitude of benefits that cater to the evolving needs and preferences of modern learners.

Here are some compelling reasons why gamification is an excellent choice for educational purposes:

- Enhanced Engagement: Gamification captivates students' attention by making learning enjoyable and interactive.

- Intrinsic Motivation: Games are inherently motivating, and gamification taps into this natural inclination. By incorporating elements like points, badges, and levels, educators can stimulate intrinsic motivation.

- Skill Development : Educational games often require problem-solving, critical thinking, and decision-making skills. By integrating these elements into the learning process, gamification helps students develop and strengthen these essential skills in a context that feels less like traditional instruction and more like a recreational activity.

- Immediate feedback : Gamification allows for instant feedback on performance. Whether through scoring systems or immediate responses to quiz answers, students receive timely information about their progress. This quick feedback loop is crucial for effective learning and enables students to identify and address areas that need improvement promptly.

All in order, Gamification encourages a culture of continuous learning. The desire to unlock new levels, achieve higher scores, or earn additional rewards promotes a mindset of ongoing improvement and mastery, instilling a love for learning that extends beyond the immediate educational context.

Why Jeopardy ?

Choosing "Jeopardy!" as a game for educational purposes can be a strategic and effective decision, given its unique characteristics and potential benefits in the learning environment

“Jeopardy!" encourages active participation and engagement. Students are not passive observers; they become active contestants, answering questions and making decisions. This interactivity enhances the learning experience by involving students directly in the educational content.

The format of "Jeopardy!" allows for the inclusion of diverse content in a single game. Questions can cover a wide range of topics, facilitating the integration of various subjects into one interactive session. This versatility makes it suitable for different educational disciplines.

“Jeopardy!" offers a dynamic and versatile platform for educational purposes. Its interactive, competitive, and adaptable nature makes it a valuable tool for educators seeking to create engaging and effective learning experiences for their students across a wide range of subjects and topics.

Rules of the game

The game consists of three rounds: Jeopardy!, Double Jeopardy!, and Final Jeopardy! Each round has a game board with categories and clues of increasing point values.

Categories: The game board features six categories, and each category has five clues of ascending point values.

Game Start: Three contestants compete in each episode. Contestants are selected through a qualifying process before the show.

Clue Selection: Contestants take turns selecting a clue from the game board. Clues are presented in the form of answers, and contestants respond with the corresponding questions.

Responding To Clues: Contestants use a buzzer system to signal that they want to respond. The first contestant to buzz in has the opportunity to answer. Responses must be in the form of a question (e.g., "What is...?" or "Who is...?").

Scoring: Correct responses earn the contestant the point value of the selected clue. Incorrect responses result in a deduction of the corresponding points. Contestants can "wager" or choose how much money they want to risk during certain rounds.

Daily Double: Two "Daily Double" clues are hidden on the board, allowing contestants to wager any or all of their current winnings on a single clue.

Double Jeopardy: In the second round, point values are doubled. The game board features new clues, and contestants can accumulate more points.
Final Jeopardy: After Double Jeopardy! contestants can wager any or all of their winnings on a single Final Jeopardy! clue. They write down their responses and wagers during a designated time.

Winning the Game: The contestant with the highest total at the end of Final Jeopardy! wins. Contestants who finish with zero or negative winnings do not advance to Final Jeopardy!
Tiebreaker: In the event of a tie, additional tiebreaker questions are used to determine the winner.

Solution Justifying

Our solution approach for a game consists in a full stack application that has 2 main features for the game. Singleplayer and Multiplayer

Singleplayer lets the user play all by himself the Jeopardy game. It shows a screen rendering different categories of questions each with 4 different sets of questions with random questions set off from module 7-8 taken from a csv.

For each question selected the user gets 4 answers to choose but only one correct. Then it shows if it is correct or not. After 5 questions the game is ended and shows the total score.

Multiplayer lets a player join a room or create one. A room consists of up to 4 players that can battle one against each other on how better he answers to the random selected questions. When one player is answering the rest are staying in a queue until their turn arrives. At the end after all players answers 5 questions the winner is shown

Technologies in Use

For this project we are going to use React framework for the user-application interface and Crow for the logic behind the application where we are going to apply OOP concepts to have the game done in place.

React

React is a free and open-source front-end JavaScript library for building powerful user interfaces based on components. It is maintained by Meta and a community of individual developers and companies.

React's component-based architecture allows for the creation of modular and reusable UI elements. Each component encapsulates its own logic and presentation, promoting a clean and organized code structure. This modularity aligns well with OOP principles, as each component can be considered an object with its own behaviour and state.

Leveraging React's state management capabilities enhances the dynamic nature of the "Jeopardy!" game. OOP concepts like encapsulation and state encapsulation can be applied, ensuring that the internal state of components is well-controlled and providing a clear separation of concerns.

Crow

Crow is a C++ framework for creating HTTP or Websocket web services. It uses routing similar to Python's Flask which makes it easy to use and has a built-in json component.

Applying OOP principles in the Crow backend can lead to a well-organized and extensible codebase. Utilize classes and objects to represent entities within the game, such as players, questions, and game sessions. Encapsulation, inheritance, and polymorphism can be applied to model and manage different aspects of the game's logic.

Define clear and well-structured object models for game entities in the Crow backend. For example, you can have classes representing a Game, Player, Question, and Score. This object-oriented approach helps in maintaining a conceptual model that aligns with the rules and dynamics of the "Jeopardy!" game.

Implementation Description

UML Diagram

Diagram UML of C++ objects

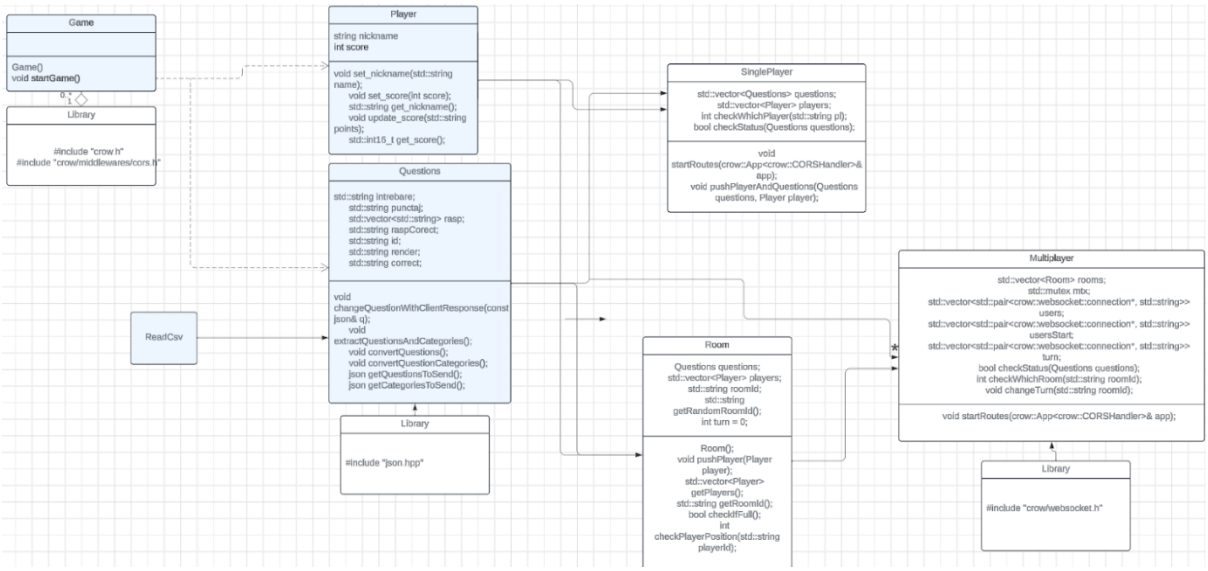
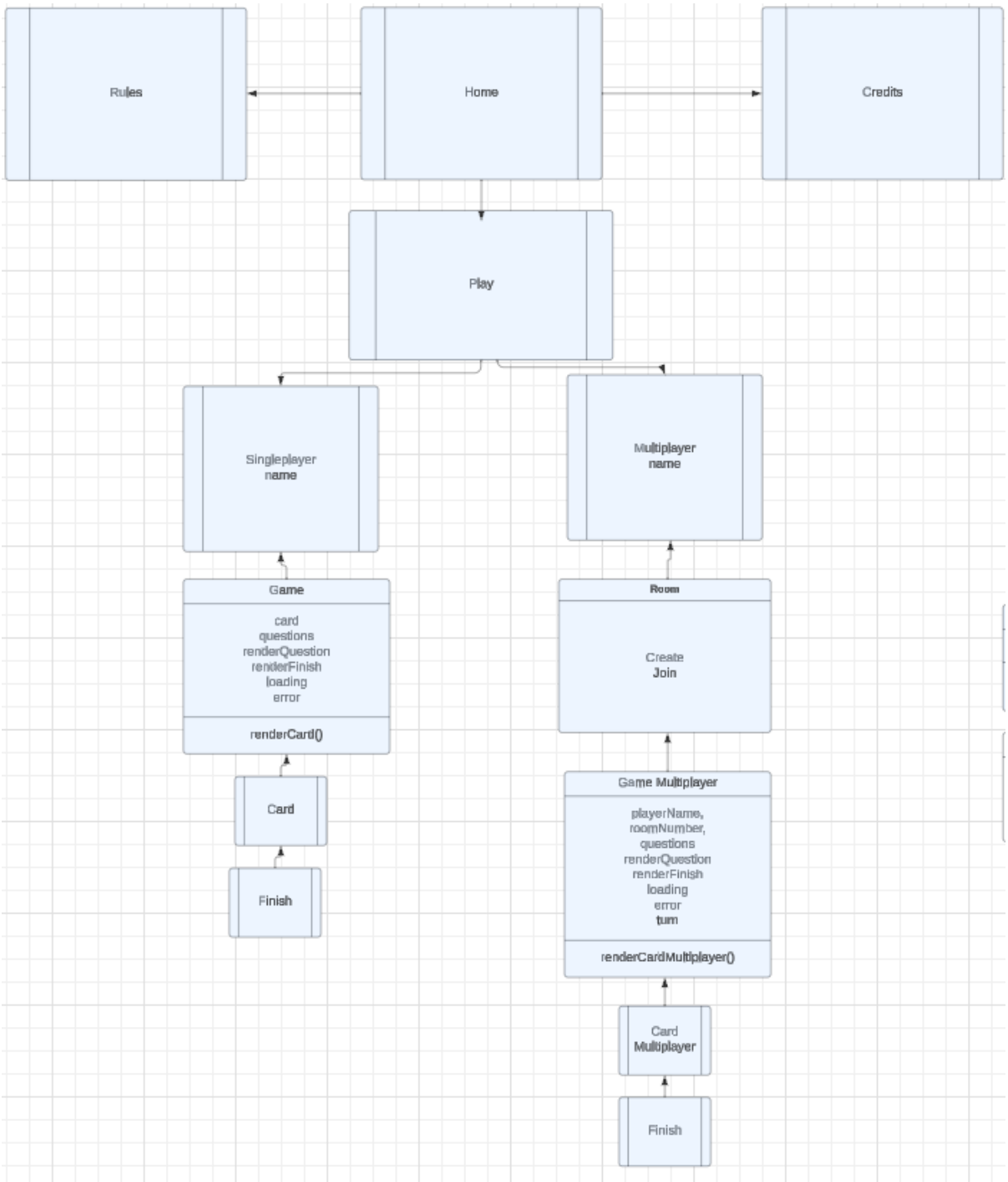


Diagram of React Components



OOP Principles

In the development of our gamification-based learning platform, we incorporated various elements of Object-Oriented Programming (OOP) principles, such as encapsulation, inheritance, and polymorphism.

- Encapsulation: One of the core OOP principles we utilized was encapsulation. This involved encapsulating the data and functionality of the platform into separate objects, such as the room, singleplayer or multiplayer. This allowed us to isolate different components of the platform, making the code more modular and easier to maintain.

- Polymorphism: We also incorporated polymorphism, which allows objects from different classes to be treated as objects of a common base class. This allowed us to create a single function capable of handling various types of inputs, such as different types of questions. This made the platform more versatile and adaptable to different types of content.

- Friend Class: This principle that we used involves granting a class special access to the private and protected members of another class. In our platform, we strategically used friend classes to establish privileged relationships between certain classes

- Inheritance : Inheritance is a core OOP concept that allows a new class (derived or child class) to inherit attributes and behaviors from an existing class (base or parent

class). In our learning platform, we leveraged inheritance to create a hierarchy of classes, promoting code reuse and the organization of functionalities. In our case we inherit Questions class from ReadCSV to access the attributes and methods more easily

Implementations Steps

Research

First, we started by seeing what this game consists of and what methods we can approach to make a full steady product that can be enjoyed for educational purposes.

After we seek information about the game, we think about a method in which we can implement it. First we wanted to do a game, but then it aroused the idea of a full-stack application where we could entrenched our skills in a context of real life application.

Then we started the research on how we can connect to a backend of c++, using an interface from React javascript. In our learning path we come across different frameworks that we could use such as “molybden”, but we committed to Crow as it was more easily to find documentation about it also we seek need for a json component to be rendering the questions and we used “nlohmann/json.hpp” library

Implementation Steps

Interface

For the interface we used node js react library. We issued which pages to be displayed then we started to create them by task allocation to each member of the team.

First, Cristi has created a mock-up for our page then we allocate the tasks this way :

- Vlad : Home, Credits and Rules screen
- Adrian : Singleplayer, Multiplayer screens
- Cristi : Singleplayer, Multiplayer games
- Dragos : Card rendering components for multiplayer and singleplayer
- Tica : Reviewing pages, fetching functions and fetching data in all pages

In our implementations, we come across different issues that we resolved by learning how the react behavior works, especially on the fetch side

Backend C++

When it comes to the backend, we come across several issues consisting in how the data was sent to the server. We first started to learn how the server side works and how we can post the data to a server. Then we had a brainstorming about how we wanted the logic of the app to be and then the Project Manager organized a schema of the project to respect it when we work on the project remotely.

In order to allocate the tasks :

- Adrian has worked on the question object.
- Vlad has worked on the file read object from csv
- Dragos has worked on Player Object and Room Creation
- Cristi has worked on developing singleplayer routes.
- Simeon has worked on creating multiplayer game routes and reviewed other works to be fully efficient.

Elements used in the implementation.

- In Game class we create an instance of the app where we are going to start the server.
    - We create instances of player and questions with the name of the user and respectively random questions for him and set up the route on the server for the player
    - We create instances of Multiplayer and Singleplayer and initialize their routes

- The Singleplayer class provides functionality for managing single-player game sessions. It includes features for retrieving questions, changing questions based on client responses, updating scores, and checking game status.

- Multiplayer Class provides functionality for creating and managing multiplayer game rooms. It includes features for creating game rooms, handling WebSocket connections, managing player turns, and updating game states.

Performance indices of the project

To ensure that our gamification-based learning platform is performant and meets design requirements, we have employed multiple performance indicators. These indicators have allowed us to gauge the platform's performance and identify any issues that need addressing concerning resource allocation.

They allow us to quantitatively measure aspects such as user engagement, completion rates, and bugs per session. Through these metrics, we gain valuable insights into how well our platform is meeting the needs of users and fulfilling its educational objectives.

Also deploy time of the crow server is a several aspect that we took in consideration and by that we could reduce the deploy time considerably and facilitate us a better debugging time of the code

Conclusions

In conclusion, the development of our gamification-based learning platform, spearheaded by the collaborative efforts of our dedicated team, marks a significant stride towards revolutionizing educational experiences. Our project, led by Project Manager Simeon Tica and the team members Cristian Bolohan, Dragos Zarnescu, Adrian Gavrila, and Vlad Munteanu, embodies a fusion of creativity, technical expertise, and a commitment to leveraging modern educational methodologies.

The team's commitment to embracing Object-Oriented Programming (OOP) principles showcases a sophisticated approach to code organization and functionality. This application of OOP principles not only enhances the modularity and maintainability of the codebase but also contributes to the adaptability of the platform to various types of educational content.

The implementation of a full-stack application, featuring singleplayer and multiplayer modes, further emphasizes the versatility and real-life application of the project. Leveraging technologies like React for the user interface and Crow for the backend, the team demonstrates a strategic use of frameworks aligned with OOP concepts.