Project – Memory Game

1. OBJECTIVE

This project's objective is to implement a single-page application (SPA) for the "Memory Game" platform, using Vue.js framework for the development of the web client, a Restful API server and a WebSocket server.

2. MEMORY GAME

The aim of the Memory Game is to find pairs of matching tiles, where each tile has a face with an image that represents it, and a back with a common image shared by all the tiles. The image of the face is visible when the tile is face up, while the image of the back is displayed when the tile is face down (the face is hidden). Each game features a set of paired tiles, with each pair consisting of two tiles that share the same face image. A face image used by one pair cannot be used by any other pair in the game. All tiles must have a matching pair - there can be no individual tiles or sets of three or more tiles with the same front face image.

When the game starts, all tiles are facing down (all faces are hidden) and randomly distributed on a game board. Players have to find all pairs of tiles, by selecting 2 tiles per turn. If the selected tiles form a pair, they are removed from the game board. The game ends when all the pairs have been found.

The game can be played by one or more players. In the single-player version, the player aims to find all pairs of matching tiles in the shortest time possible. In the multiplayer version, players take turns, aiming to find as many pairs of matching tiles as possible. When the game starts, players are arranged in a random order, and turns are taken in that sequence.

For each turn of the game, the active player picks the first tile, which is immediately turned face up (making the face image visible). Then, the player selects a second tile, which is also turned face up. If the two selected tiles form a pair (they have the same face image), they are removed

from the game board - their images remain visible for a brief moment (e.g., one second or less) before removal. In this case, the active player continues to play the next turn.

If the two selected tiles do not form a pair (they have different face images), they remain face up for a short period (e.g., one to two seconds) to allow all players to memorize their positions. After this time, both tiles are turned face down (showing the back image), and the next player takes their turn (in multiplayer games).

The game ends when there are no more tiles on the game board. In the multiplayer version, the player who has found the most pairs wins. If two or more players discover the same number of pairs, the player who found them first wins, regardless of the order in which the game was played.

In the single-player version, when the game ends, it will display the 'game time' - the interval from the selection of the first tile in the first turn to the removal of the last tile from the board.

If the active player takes longer than 20 seconds to make a move (the time between becoming active for the turn and clicking on the second tile), they are considered to have given up. In this case, the next player takes their turn, and the game continues with the remaining players. If only one player is left, that player is declared the winner.

3. MEMORY GAME PLATFORM

The Memory Card platform allow users to play the memory game, either as a single-player or on multiplayer games. Anonymous users are limited to single-player games on a board size of 3 columns by 4 rows. In contrast, registered users can play both single and multiplayer games with any board size supported by the platform. At a minimum, the platform should accommodate the following board sizes: 3 x 4 (accessible to all users), 4 x 4, and 6 x 6, with the possibility of additional sizes at the discretion of students.

The platform features a credit system where users can purchase "brain coins" using real money at a conversion rate of 1 euro for 10 brain coins. Newly registered users receive a welcome bonus of 10 brain coins. Users can buy additional brain coins in multiples of ten (1 €). Brain coins cannot be converted back into real money.

These brain coins are used to play games and access other in-game items and features (if implemented) determined by students. Each single-player game costs 1 "Brain Coin", while each multiplayer game costs 5 brain coins. The winner of a multiplayer game receives all brain coins in the game, minus 3 brain coins which revert to the platform as a commission. Costs of in-game items and features, if implemented, will be determined by students.

The platform should track all transactions (welcome or other bonuses, brain coins purchases, brain coins spent and earned on games or in-game items and features), as well as games played by registered users (ignoring games played by anonymous users). This enables users to access their transaction and game history. Additionally, the platform should maintain personal scoreboards for each user, and global scoreboards available to all users, including anonymous users.

Platform management is handled by administrators, who can access the list of all users, including players and other administrators. They have the authority to block and unblock players, create additional administrator accounts, and remove any account except their own. Players' accounts can be removed by any administrator or by themselves. If a player has conducted at least one transaction or played one game, their account must be soft deleted.

> *Note: when a player account is soft deleted, it maintains all related data (user profile, brain coins, transactions and games) on the database, but the application behaves as if the account does not exist.*

Administrators should also have view-only access to all transactions and games on the platform, as well as summary and statistical information on platform usage.

4. DAD PROJECT

The objective of the DAD project is to develop and deploy a web application for the Memory Game Platform. This application will utilize a single-page application (SPA) model and the Vue.js framework for the web client, and it will be hosted on a designated school internal server.

The project will also include a backend with a database, a RESTful API server, and a WebSocket server. Students are free to choose the technologies for the backend, provided they are open source or free external services, and can be installed on or consumed from the designated server.

5. FEATURES

The web application must implement a set of groups of features, which will be described in this section. Their implementation must consider all relevant information about the business model outlined previously, the descriptions of the groups of features (in current section), the constraints (section 6) and non-functional requirements (section 7), along with all applicable standards and best practices for the technologies and patterns involved. Students are free to determine the user interface and usability, as well as aspects of the business not explicitly detailed.

G1. USER REGISTRATION, AUTHENTICATION, PROFILE, END SESSION, ACCOUNT REMOVAL

Any user can register on the platform, provided they supply a unique email address and unique nickname during the registration process. Users must also provide their name, an optional photo or avatar, and a password (with a minimum length of 3 characters).

The credentials for authenticating the application (login) include the email and password. An authenticated user can access their profile, where they can view and update their email, nickname, name, photo or avatar, and password.

Users can also end their current session (sign out) or remove their own account. Account removal requires confirmation, which may involve inputting the current password, nickname, or a specific word provided by the application. When the account is removed, the user loses all brain coins associated with that account.

Administrators will also have access to their own profile, where they can view and perform all the same actions as any other user, except for the option to remove their own account.

G2. BRAIN COINS AND TRANSACTIONS

Registered users should be able to easily view their total brain coins and purchase brain coins within the platform. The platform will utilize an external API to simulate the financial transaction involved on a purchase. Additionally, while playing, users will spend or earn brain coins on games or in-game items and features.

All transactions involving brain coins—including bonuses, purchases, internal spending, and internal earnings—should be tracked by the platform and presented to the user as their transaction history.

Each player has access only to their own transaction history, while administrators can access the transaction histories of all players on the platform. However, administrators cannot create, modify, or delete transactions; they can only view them.

G3. SINGLE-PLAYER GAMES

The application allows users to play the memory game in a single-player mode. Single-player games track the game time and display it when the game finishes. The platform supports the following board sizes: 3x4 (3 columns and 4 rows), 4x4, and 6x6. Students may choose to implement additional board sizes at their discretion, provided they follow game rules.

When a registered user plays on any board except the 3x4 board, they will pay one brain coin at the start of the game. This brain coin is non-recoverable, even if the game is not completed. If the user has no brain coins in their account, they will be restricted to playing on the 3x4 board.

Single-player games played by registered users should be tracked on the platform, so that the user has access to their game history, as well as providing the information needed to maintain personal and global scoreboards.

G4. MULTIPLAYER GAMES

The application allows users to play the memory game in multiplayer mode, where two or more players take turns trying to find matching pairs of tiles, as detailed in section "2. Memory Game." Only registered users with at least 5 brain coins can participate, and each game costs 5 brain coins to enter. The winner of a multiplayer game receives all brain coins from the game, minus a commission of 3 brain coins that revert to the platform.

The application supports a minimum of two players per game, with the potential for more depending on the students' implementation. To enter a multiplayer game, players must go to the lobby, where they can join a pending game or create a new one and wait for others to join. A pending game is a multiplayer game that has not yet started because it is waiting for players. The game begins when the number of joined players reaches the game limit (e.g., if the platform only supports 2 players, the game will start when the second player joins) or when the player who created the game decides to start it, provided at least 2 players have joined.

The platform must support 4x4 and 6x6 boards for multiplayer games. Students may choose to implement additional board sizes at their discretion, provided they follow game rules. Note that the maximum number of players per game may depend on the board size.

Multiplayer games are also tracked on the platform, guaranteeing that users have access to their full game history, including both single-player and multiplayer games.

G5. GAME HISTORY AND SCORING BOARDS

All games played by registered users will be tracked by the platform. This means that any user can access their game history, which includes summary information about each game. Each game will include the starting date and time, board size, status: (pending, playing, ended, interrupted), total game time (the time interval from the first move until the game ends), and the number of turns played. Multiplayer games will also provide information about the players involved, including the player who created the game and the winner of the game. Note that the information about a specific

multiplayer game should be accessible to all players involved in that game. Students may choose to include additional details about the games, enhancing the overall experience.

Each player only has access to their own game history, while administrators can access the history of all games from all players on the platform.

Using the information tracked by the platform, the application should also present personal and global scoreboards. Personal scoreboards pertain to a specific user's games and are only available to that user. Global scoreboards encompass all games of all users and are accessible to everyone on the platform, including administrators and anonymous users. Global scoreboards should include the nickname of the player associated with a given performance (e.g., the nickname of the player with the best time on a 4x4 board).

When comparing performances on different board sizes is not applicable, the scoreboard must be organized by board size. The platform should present the following personal scoreboards for single-player games: best times for each board size and minimum turns for each board size; and for multiplayer games: total victories and total losses (of the player).

For global scoreboards, the platform should present the following for single-player games: the best time for each board size and the minimum turns for each board size. For multiplayer games, the platform should show the top 5 players with the most victories in any board. Note that all three global scoreboards mentioned should include the nickname of the player. Additionally, when two or more players have the same result (e.g., the same number of victories), the first to obtain that result is considered the best; the scoreboard is updated only when a player surpasses the current results. Students are free to include other scoreboards or summaries related to game performance.

G6. ADMINISTRATION

Platform administrators will be responsible for monitoring and maintaining the game platform. They are not allowed to play games or have any brain coins; however, they have access to their own profile, where they can view and perform all the same actions as any other user, except for the option to remove their own account (they can only remove the accounts of other users).

Administrators have access to the list of all users, including players and other administrators. They can block and unblock players, create other administrator accounts, as well as remove any other account, whether player or administrator, except their own. An administrator account must always be created by another administrator; the platform user registration process does not allow for the creation of administrator accounts.

Administrators should also have view-only access to all transactions and games on the platform, as well as summary and statistical information on platform usage.

G7. STATISTICS

The application should provide all users with statistical information, both visually (with graphs) and textually (with plain text or tables). Anonymous users and players can only view generic anonymized information, such as the total number of players registered, total games played, games played last week or last month, etc. Conversely, administrators have access to all information, including sensitive and non-anonymized data, such as total purchases by time period (weeks or months), total purchases by player, etc.

> *Students are responsible for deciding which statistics to show, for the type and quality of information extracted from the platform, and for the display format presented to end users.*

G8. CUSTOM FEATURES AND IMPLEMENTATION DETAILS

To distinguish their projects from one another, students should add custom features not specified in the project statement or include implementation details that meet one of the following criteria: use technologies or techniques not covered in classes; optimize performance; improve architecture; enhance implementation structure or quality.

> *Students are responsible for specifying their own custom features and implementation details. The objective is to promote innovation and creativity.*

As a reference, here are some examples of custom features: enabling games with three or more identical tiles (instead of only pairs); allowing full playback of previous games; incorporating multiple sets of tile images; introducing difficulty levels for single-player games; permitting players to add their own tile sets; adding sound effects to the game, among others.

> *The evaluation of this group of features will be cumulative and based on empirical assessment. The teacher will evaluate the quality and complexity of each feature or implementation detail and weight it accordingly. Grading this group to 100% may require several custom features or implementation details or just one, depending on the quality and complexity of what was implemented.*
>
> *Technologies or features applied in the context of other disciplines (TAES or other) are not considered on the evaluation of this group of features.*

6. CONSTRAINTS

The project's **mandatory constraints** are as follows:

C1. The application must exclusively use the single-page application (SPA) paradigm.

C2. The application must use the Vue.js framework for client-side code.

C3. External transactions (for purchasing brain coins) must be handled by the provided external RESTful API, the "Payment Gateway Service."

C4. The application must be deployed on a designated internal school server and accessible via a desktop browser.

C5. Backend technologies for the database, RESTful API server, and WebSocket server must have an open-source license or be available as a free external service.

C6. Backend technologies are constrained by the technical characteristics of the designated server as they must be installed on or accessible from the designated server. If the server does not support a specific technology, students must replace it with a valid alternative.

C7. The database of the deployed application must be seeded with data that simulates real-life usage in terms of both size and content, reflecting typical user interactions and data patterns.

7. NON-FUNCTIONAL REQUIREMENTS

The project's non-functional requirements are as follows:

NF1. The client application code and structure must adhere to the principles and best practices of the Vue.js framework.

NF2. The REST API server must follow the principles and best practices of RESTful services.

NF3. The visual appearance and layout must be consistent throughout the application and adapted to the application's objective and usage context.

NF4. All implemented features must be correctly integrated and function consistently across the application.

NF5. The application should offer optimal usability, ensuring that information and available operations are clearly presented and that user interactions are straightforward, utilizing standard procedures and requiring minimal effort to achieve desired outcomes.

NF6. User input must always be validated on both the client and server according to rules derived from the business model and feature groups.

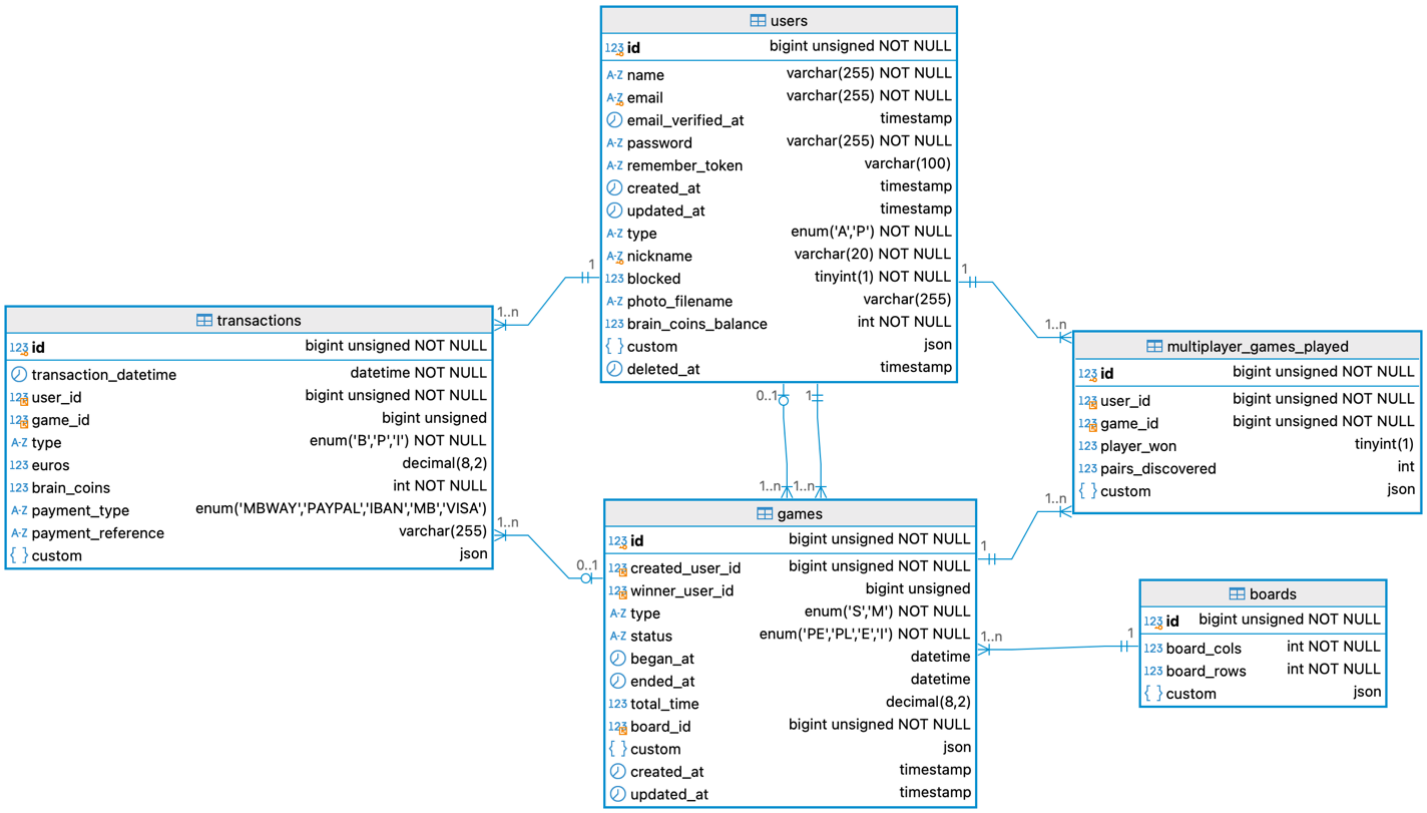
NF7. The application should be reliable and optimized for performance. Techniques may include minimizing internet bandwidth by reducing the number of HTTP requests and the size of HTTP responses, as well as limiting dataset sizes on the client while maximizing performance for data navigation and filtering using JavaScript. Other optimization techniques should also be applied for both the client and server.

NF8. The application must prioritize security, ensuring the protection of user data and privacy, and preventing unauthorized access to sensitive information. Authentication and authorization must comply with established principles and best practices for REST APIs.

NF9. The performance and security of WebSocket full duplex communication must be optimized, including minimizing message sizes and sending messages only to clients that require them, avoiding unnecessary broadcasts.

8. DATABASE

As mentioned earlier, the project will include a database that uses technology selected by the students, provided it is an open-source database system or a free external service. Additionally, a fully functional database will be available for students to use directly or modify as needed. The database structure is defined through Laravel migrations and populated with sample data using a provided Laravel seeder. The database structure is as follows:



*Note: All users have the same password: "**123**"*

TABLES AND COLUMNS

In this section we will describe all tables and the most relevant columns of the database.

USERS

Table with all users (players and administrators).

- **type** – User type ('A' for administrator; 'P' for player),
- **nickname -** User's nickname – used for multiplayer games and scoreboards.
- **blocked** – Boolean indicating whether the user is blocked.
- **photo_filename** – Filename of the user's photo/avatar.
- **brain_coins_balance** – Current balance of "brain coins" for the player.
- **custom** – JSON field for students to include any additional information they wish to add.

BOARDS

Table with all boards supported by the platform.

- **board_cols** – Number of columns of the board.
- **board_rows** – Number of rows of the board.
- **custom** – JSON field for students to include any additional information they wish to add.

GAMES

Table of games played by registered users:

- **type** – Type of game ('S' for single-player; 'M' for multiplayer).
- **created_user_id** – User ID of the player who created the game.
- **winner_user_id** – User ID of the winning player (null for single-player games).
- **status** – Game status ('PE' for pending, 'PL' for in progress, 'E' for ended, 'I' for interrupted).
- **began_at** – Date and time when the game started (first tile selection).
- **ended_at** – Date and time when the game ended (only when status = 'E', null for other statuses).
- **total_time** – Total duration of the game (in seconds).
- **board_id** – ID of the board used in the game.

- **custom** – JSON field for students to include any additional information they wish to add.

MULTIPLAYER_GAMES_PLAYED

Table of players in multiplayer games (pivot table between players and games):

- **user_id** – User ID of the player who participated in the game.
- **game_id** – Game ID of the multiplayer game played by the user.
- **player_won** – Boolean indicating whether the player won the game.
- **pairs_discovered** – Total number of pairs discovered by the player.
- **custom** – JSON field for students to include any additional information they wish to add.

TRANSACTIONS

Table of all transactions:

- **type** – Type of transaction ('B' for bonus, 'P' for purchases, 'I' for internal spending/earnings related to a game).
- **transaction_datetime** – Date and time of the transaction.
- **user_id** – User ID of the player associated with the transaction.
- **game_id** – Game ID linked to the transaction (only applicable for type 'I').
- **euros** – Value of the purchase transaction (in euros; only applicable for type 'P').
- **payment_type** – Type of payment for the purchase transaction (only applicable for type 'P'). Possible values: MBWAY, IBAN (bank transfer), MB (Multibanco payment) and VISA.
- **payment_reference** – Reference for the payment of a purchase transaction (only applicable for type 'P').
- **brain_coins** – Value of the purchase transaction in brain coins; this can be a positive or negative integer, indicating increments or decrements in the player's brain coins balance.
- **custom** – JSON field for students to include any additional information they wish to add.

9. PAYMENT GATEWAY SERVICE

The platform will use an external payment gateway service to handle all financial transactions which is available on this URI: **`https://dad-202425-payments-api.vercel.app`**

This service is a simulated "fake" service that only mimics external financial transactions—no actual transactions occur. The service has only one endpoint:

`post`	`<service URI>/api/debit`	Creates a new debit on the external entity

When a user purchases brain coins, a debit should be created on the external entity. The endpoint expects a JSON object in the request payload with the following format (example):

```
{
     "type": "MB",
     "reference": "45634-123456789",
     "value": 3
 }
```

Upon a successful debit operation, the service returns an HTTP response with the status code `201 Created`. If the debit operation fails due to an invalid request payload, unrecognized reference, or exceeding the value limit, the service returns an HTTP response with error messages and a status code of `422 Unprocessable Entity`. Other error status codes may be returned for different issues.

VALIDATION

The properties "type," "reference," and "value" in the request payload are mandatory and are validated on the Payment Gateway Service. They should also be validated on the game platform:

- **type**: Accepts the following values: "MBWAY", "PAYPAL", "IBAN", "MB", and "VISA".
- **reference**: Validation rules depend on the type:
    - MBWAY: 9 digits starting with 9 (e.g., "915785345")
    - PAYPAL: A valid email (e.g., "john.doe@gmail.com")
    - IBAN: 2 letters followed by 23 digits (e.g., "PT5012345678123456781234 9")
    - MB: 5 digits (entity number), a hyphen ("-"), and 9 digits (e.g., "45634-123456789").
    - VISA: 16 digits starting with 4 (e.g., "4321567812345678").
- **value**: Accepts any positive integer greater than 0 and less than 100 – (e.g., 3 is valid; 3.01 is invalid).

SIMULATION

To simulate valid and invalid debit operations, the “fake” service uses a set of simple rules designed for testing. These rules **cannot be validated within the game platform code**; rather, they exist solely to demonstrate how the platform reacts to errors returned by the external service.

The following reference values are considered invalid by the service, simulating that there are no associated MBWay, PayPal, IBAN, MB, or Visa accounts:

- MBWAY – phone numbers (reference) starting with "90" – e.g., "901645932"
- PAYPAL – emails that start with the substring "xx" – e.g., xx.john.doe@gmail.com
- IBAN – references that start with "XX" – e.g., "XX5012345678123456781234 9"
- MB – entity numbers starting with 9 – e.g., "95634-123456789"
- VISA – references starting with "40" – e.g., "4021567812345678"

To simulate fund limitations (insufficient account balance), the service applies the following limitations:

- MBWAY - MbWay account funds are limited to 5€.
- PAYPAL - PayPal account funds are limited to 10€
- IBAN - IBAN account funds are limited to 50€
- MB - MB account funds are limited to 20€
- VISA – Visa account funds are limited to 30€

10. DELIVERY AND DEPLOYMENT

The application must be deployed on the designated internal school server and be accessible on ESTG intranet via a desktop browser (testing will be conducted on Google Chrome). The functional evaluation will occur exclusively on the deployed application, so **deployment is mandatory**.

When deploying the project, all features and configurations should closely resemble the application's final production stage. Deployment must include all performance optimizations and utilize data that simulates real-life application usage.

Project delivery must include three files ("code," "report," and "report complement") per group, as well as one individual report file per student. For example, if the group has four students, a total of seven files must be submitted (three group files and four individual report files). One student will deliver four files (three group files and their own individual report), while the other three students will submit only their individual reports.

Files to be submitted:

- **prj_GG.zip** – A zip file that includes a copy with all folders and files (source code and other files) of all projects of the platform, excluding the “vendor,” “node_modules,” and ".git" folders (the ".git" folder, if present, is typically hidden).
- **grp_report_GG.xlsx** – The group report file, which includes group identification elements and generic information about the project implementation. A model for the report will be provided.
- **grp_report_complement_GG.zip** – A zip file containing documents and images that complement the report. It should include at least one image of the database diagram used in the project but may contain additional images or documents that help the teacher understand the project's internal structure and implementation details.
- **individual_report_NNNNNNN.xlsx** – The individual report file, which includes self-assessment and peer evaluation. A model for the report will be provided.

Note: Replace GG with your group number and NNNNNNN with your student number.

11. EVALUATION

The evaluation of the project will consider the compliance with the mandatory constraints (C1… C7), Groups of features (G1 … G6) and Non-Functional Requirements (NF1 … NF10).

Mandatory Constraints (C1…C7)

These are mandatory constraints. If the project does not comply with all these constraints (including application deployment), then it will have a **classification of zero**.

Groups of features (G1…G8)

For the classification of each group of features, it will be considered the compliance with the business model, the usability, the reliability and the integration with the application, the quality and structure of the project's source code to implement it, as well as the compliance of non-functional requirements applicable to the user story.

The classification of all groups of features is valued **90%** of the project. Individual values for each group of features are specified on the table with the weight of all criteria.

Non-Functional Requirements (NFR1…NFR10)

Non-functional requirements are valued **10%** of the total classification of the project. These requirements are evaluated as a whole (there is no individual classification for each non-functional requirement) through the entire application. The evaluation of these requirements is made in parallel with the evaluation of the functional requirements, by analyzing the code and structure of the project and by testing (manually or automatically) chosen Web API endpoints.

Weight of all evaluation criteria:

	Groups of features	**Weight**
G1	User registration, authentication, profile, end session, account removal	**10%**
G2	Brain coins and Transactions	**10%**
G3	Single-player games	**10%**
G4	Multiplayer games	**20%**
G5	Game history and scoring boards	**10%**
G6	Administration	**10%**
G7	Statistics	**10%**
G8	Custom features and implementation details	**10%**
NFR	All Non-Functional Requirements	**10%**