PROJECT REPORT

OF

“Two Truths and a Lie Game”

Software Engineering (CS330)

Bachelor of Technology (CSE)

By

Mahi V Prajapati – 22000996

Nidhi A Patel – 22001004

Third Year, Semester 5

Course In-Charge: Prof. Darshan Parmar

NAVRACHANA UNIVERSITY

a UGC recognized University

Department of Computer Science & Engineering

School Engineering and Technology

Navrachna University, Vadodara

Autumn Semester 2024

Table of Contents

List of Figures

List of Tables

I Project Description

1 Project Overview

The "Two Truths and a Lie" game is a popular icebreaker and party game that helps participants get to know each other better. Each player shares three statements about themselves—two of which are true, and one is false. The objective for the other players is to identify the lie. This project aims to digitize this engaging game, providing an interactive platform for users to play with friends or others online. The project will involve designing a user-friendly interface, implementing game logic, and ensuring smooth, real-time interaction between players.

2 The Purpose of the Project

2a The User Business or Background of the Project Effort

Content

This project is focused on creating a digital version of the "Two Truths and a Lie" game, a popular social activity designed to break the ice in group settings. The development effort was triggered by the increasing demand for online social interaction tools, particularly those that can be used in both casual and professional environments. The project aims to deliver a platform that enhances user engagement, improves accessibility across various devices, and ensures user privacy, thereby addressing the needs of both individual users and event organizers.

Motivation

Without this project, there would be limited options for people to engage in such social activities online, especially in a secure and accessible manner. The digital platform will provide a way for users to connect and interact, regardless of their physical location, while ensuring their data is protected. This provides justification and direction for the project, as it meets a growing demand in the market.

Considerations

The user problem is significant, as there is a clear need for more engaging and secure online platforms for social interaction. Addressing

this problem is essential, as it fills a gap in the market for a fun and accessible tool that can be used in various settings, from casual get-togethers to corporate events.

2b Goals of the Project

Content

The goal of this project is to develop an interactive and secure platform for the "Two Truths and a Lie" game that is accessible across multiple devices. The platform will prioritize user engagement, ease of use, and the protection of personal data.

Motivation

There is a risk that the project could lose focus as new features or ideas emerge during development. It is crucial to remain aligned with the core objectives of creating a user-friendly, accessible, and secure platform. Regular review sessions will ensure that the project does not deviate from its goals and maintains its intended purpose.

Examples

- Enhance User Interaction: Create a platform that encourages users to engage with each other in a fun and meaningful way.
- Improve Accessibility: Ensure the game can be played across various devices and platforms, including mobile and desktop.
- Maintain User Privacy: Implement secure login and data management features to protect users' information.
- Facilitate Social Connection: Provide features that allow users to connect with friends or find new players easily.

2c Measurement

The success of the project will be measured by several key metrics:

- **User Engagement:** Number of games played per user and session length.
- **User Satisfaction:** Feedback collected through in-app surveys and user reviews.
- **Platform Performance:** Load times, responsiveness, and uptime statistics.
- **User Growth:** Number of new users and retention rates over time.
- **Privacy Compliance:** Ensuring there are no data breaches or user complaints related to privacy issues.

SDLC Model Selection for "Two Truths and a Lie Game" Project

Selection of SDLC Model: Agile Methodology

For the "Two Truths and a Lie" game project, the **Agile methodology** is the most suitable Software Development Life Cycle (SDLC) model due to the following reasons:

- **Flexibility and Adaptability:**
    - The Agile model allows for iterative development, which is crucial for a project that may evolve based on user feedback and testing. Since the game involves user interaction and social features, it’s important to be able to adapt to user needs and preferences quickly.
- **Incremental Development:**
    - Agile promotes incremental delivery of the project, where each iteration adds functional components. This approach allows for early detection of issues and continuous integration of improvements, ensuring that the core features of the game (user interaction, accessibility, privacy) are built and refined progressively.

- **User-Centric Development:**
    - The Agile model emphasizes regular user feedback, which is essential for a project focused on user engagement. By involving users in the development process, the project can ensure that the final product meets user expectations and provides a seamless and enjoyable experience.
- **Risk Management:**
    - Agile's iterative nature helps in identifying and addressing risks early in the development process. This is particularly important for maintaining the integrity of the project, ensuring that the platform remains secure and user-friendly throughout its development.
- **Continuous Improvement:**
    - Agile encourages ongoing improvement and adaptation, allowing the project team to refine the platform continuously based on user feedback and performance metrics. This approach aligns with the project's goals of enhancing user interaction, improving accessibility, and maintaining privacy.

3. The Scope of the Work

The scope of this project involves the development of a "Two Truths and a Lie" game application. The application will feature a user-friendly interface, multiplayer functionality, and customizable game settings. The scope also includes integrating social sharing features and providing analytics for tracking user engagement. The project will be executed using agile methodologies, ensuring iterative development and continuous feedback.

3a The Current Situation

Content

- **Current Situation:** This section describes the existing state or situation relevant to the development of the “Two Truths and a Lie” game

application. It outlines any current systems, practices, or tools that the new application will interact with or replace.

Motivation

- Understanding the current situation helps in identifying the gaps or issues that the new application aims to address. It also provides a baseline for measuring the impact and success of the new system.

Examples

- **Current State:** Currently, there is no dedicated digital platform for playing "Two Truths and a Lie" in a structured manner. Users may play informally through social media or messaging apps, but these methods lack organized gameplay features, score tracking, and user management.
- **Existing Tools:** Players might use general-purpose communication tools or manual methods to facilitate the game, which can be cumbersome and inefficient.

3b The Context of the Work

Content

- **Context of the Work:** This section describes the broader environment and circumstances in which the "Two Truths and a Lie" game application will be developed and used. It includes any relevant factors such as target audience, technological trends, and organizational goals.

Motivation

- Understanding the context helps ensure that the application is designed to meet the needs of its intended users and align with current trends and technologies. It also provides insight into the potential impact and relevance of the application.

Examples

- **Target Audience:** The primary users are likely to be individuals and groups looking for a fun and engaging social game, including friends and family members, as well as potentially educational settings where the game can be used as an icebreaker or team-building exercise.
- **Technological Trends:** Increasing use of mobile devices and online gaming platforms creates an opportunity for the application to leverage these technologies to provide a seamless and interactive experience.
- **Organizational Goals:** If developed as part of a course or project, the application aims to demonstrate practical skills in app development and game design.

Considerations

- Assess user preferences and requirements to ensure the game features align with their expectations.
- Consider how emerging technologies, such as augmented reality or integration with social media, could enhance the game experience.

3c Work Partitioning

Content

- **Work Partitioning:** This section outlines how the development work for the “Two Truths and a Lie” game application will be divided into manageable tasks or phases. It details the major components or modules of the project and assigns responsibilities.

Motivation

- Effective partitioning helps in organizing the development process, assigning tasks to team members, and managing project timelines. It ensures that each part of the project is given appropriate attention and resources.

Examples

- **Design Phase:** Creating wireframes and prototypes for the user interface and game mechanics.
- **Development Phase:** Coding the core game logic, user authentication, and user interface components.
- **Testing Phase:** Conducting unit tests, integration tests, and user acceptance testing to ensure the application functions as expected.
- **Deployment Phase:** Releasing the application to users and providing support for any post-launch issues.

Considerations

- Define clear milestones and deliverables for each phase to track progress and ensure timely completion.
- Allocate resources and assign tasks based on team members' skills and expertise.

3d Competing Products

Content

- **Competing Products:** This section reviews other similar applications or platforms that offer "Two Truths and a Lie" or similar game experiences. It identifies their strengths, weaknesses, and features that may influence the development of your application.

Motivation

- Analysing competing products helps in understanding the market landscape and identifying opportunities for differentiation. It also provides insights into features that are popular or lacking in existing solutions.

Examples

- **Competitor 1:** Social media platforms where users play “Two Truths and a Lie” informally, but lack structured gameplay and scoring.
- **Competitor 2:** Existing mobile game apps that offer similar games but may not have the specific features or user experience envisioned for your application.

Considerations

- Identify unique features or improvements that your application can offer to stand out from competitors.
- Consider user feedback on competing products to address common pain points or preferences.

4 Product Scenarios

This section outlines the various scenarios in which the "Two Truths and a Lie" game application could be used. These scenarios will help guide the development process by providing a clear understanding of the different contexts in which the product may be deployed.

4a Product Scenario List

The product scenarios for the "Two Truths and a Lie Game" application include:

- **Scenario 1:** A user wants to create a new game session and invite friends to participate.
- **Scenario 2:** A user joins an existing game and guesses which of the three statements is a lie.
- **Scenario 3:** A user views the results of a completed game, including scores and correct answers.
- **Scenario 4:** A user shares their game results on social media.
- **Scenario 5:** A user wants to browse through their game history and see past performance.

4b Individual Product Scenarios

- **Scenario 1: Creating a New Game**
  - The user opens the app and selects the "Create Game" option. They enter three statements, two of which are true and one is a lie. The user can then invite friends via a shareable link or direct app invite.
- **Scenario 2: Joining an Existing Game**
  - The user receives an invitation to join a game and clicks the link, which opens the app and directs them to the game. The user reads the three statements and selects the one they believe is a lie.
- **Scenario 3: Viewing Results**
  - After the game ends, the user is presented with the correct answers and a summary of how many players guessed correctly. The user can also see their overall score and ranking within that session.
- **Scenario 4: Sharing Game Results**
  - The user has the option to share their game results directly from the app to social media platforms like Facebook, Twitter, or Instagram.
- **Scenario 5: Viewing Game History**
  - The user can access their game history from the app's main menu. Here, they can see all the games they have participated in, along with the outcomes and scores.

5 Stakeholders

5a The Client

Content

- The client is the entity or individual who has commissioned the development of the "Two Truths and a Lie" game application. They are primarily interested in the successful completion of the project within the agreed-upon scope, budget, and timeline.

Motivation

- The client's satisfaction is paramount as they are funding the project and have a vested interest in its success. Understanding the client's goals and requirements ensures that the development team delivers a product that aligns with their vision.

Considerations

- The client's expectations must be clearly defined and documented at the start of the project. Regular progress updates and meetings should be scheduled to keep the client informed and involved in key decisions. Flexibility in accommodating any changes or feedback from the client is also crucial.

5b The Customer

Content

- The customer refers to the end-user who will be purchasing or downloading the "Two Truths and a Lie" game application. This group includes casual gamers, educators, and social influencers who will interact with the product regularly.

Motivation

- The customer's experience with the product will directly influence its success in the market. By understanding customer needs, preferences, and pain points, the development team can create a product that meets or exceeds expectations, leading to higher satisfaction and retention rates.

5c Hands-On Users of the Product

Content

- Hands-on users are those who will interact with the application on a daily basis. This group includes gamers, educators using the app in classrooms, and social media influencers engaging their audience through the app.

Motivation

- Understanding the specific needs and behaviors of hands-on users is critical to the product's design. These users will drive the success of the app through their engagement and feedback, helping to refine and enhance the product over time.

Examples

- **Gamers:** Casual players using the app for entertainment in social settings.
- **Educators:** Teachers using the game as an educational tool to foster interaction and critical thinking in students.
- **Influencers:** Social media personalities who use the app to create content and engage with their followers.

5d Priorities Assigned to Users

Content

- Different user groups may have varying levels of importance in the project's hierarchy. Prioritizing these users helps ensure that the most critical features and functionalities are developed to meet their needs first.

Motivation

- Assigning priorities allows the development team to focus on the most impactful areas of the project, ensuring that key user groups have their needs addressed promptly. This approach helps in managing resources effectively and delivering a product that meets core requirements.

5e User Participation

Content

- User participation involves actively engaging users in the development process, from initial requirements gathering to beta testing. Their input is invaluable in shaping the product to meet real-world needs.

Motivation

- Involving users throughout the development process increases the likelihood of delivering a product that resonates with the target audience. It also helps identify potential issues early, allowing for timely adjustments and refinements.

5f Maintenance Users and Service Technicians

Content

- Maintenance users and service technicians are responsible for ensuring the ongoing functionality and performance of the application post-launch. This includes routine updates, troubleshooting, and addressing any technical issues that arise.

Motivation

- Ensuring the app is well-maintained is essential for long-term success. A reliable and well-supported application builds trust with users, leading to higher retention and positive word-of-mouth.

5g Other Stakeholders

Content

- Other stakeholders include investors, regulatory bodies, and marketing teams who have a vested interest in the project's outcome. While they may not interact with the product directly, their influence and support are crucial to the project’s success.

Motivation

- Recognizing the interests of these stakeholders ensures that the project aligns with broader organizational goals and complies with relevant regulations. Their support can also be critical in securing funding and driving the product’s market success.

6 Mandated Constraints

Mandated constraints refer to the specific limitations and requirements imposed on the development of the "Two Truths and a Lie" game application. These constraints include technological, design, financial, and scheduling factors that directly impact the project.

6a Solution Constraints

Content

- The "Two Truths and a Lie" game application must adhere to specific solution constraints, including the design and implementation of core features, user experience, and performance. These constraints define the boundaries within which the development team must operate.

Motivation

- Adhering to solution constraints ensures that the game remains within scope, aligns with user expectations, and delivers a consistent experience across various devices. This helps maintain the integrity and quality of the application.

Examples

- The game must support both single-player and multiplayer modes.
- It should be compatible with Android and iOS platforms, ensuring a seamless experience on both.
- The user interface must be intuitive, with easy-to-navigate menus and clear instructions for gameplay.

Considerations

- The development team should evaluate the feasibility of meeting these constraints given the project timeline and budget. Any potential conflicts or challenges must be addressed early to prevent delays or compromises in the final product.

6b Implementation Environment of the Current System

Content

- The implementation environment includes the software development tools, platforms, and infrastructure necessary to create, test, and deploy the "Two Truths and a Lie" game application.

Motivation

- A thorough understanding of the implementation environment ensures that the application is developed in a way that is compatible with the existing infrastructure and can be deployed smoothly.

Examples

- The development will be done using IDEs like Android Studio and Xcode, ensuring compatibility with both Android and iOS devices.
- Testing will be conducted on a variety of devices with different screen sizes and processing capabilities to ensure broad compatibility.
- The application will be deployed on cloud servers for scalability and performance optimization.

Considerations

- The development team must ensure that the implementation environment supports all necessary tools and technologies. Any limitations or potential issues should be identified and mitigated to avoid disruptions in the development process.

6c Partner or Collaborative Applications

Content

- The "Two Truths and a Lie" game may need to integrate with third-party applications or services to enhance functionality, such as social media platforms for sharing game results or analytics tools for tracking user engagement.

Motivation

- Integrating with partner applications allows the game to offer additional features and improve user engagement by leveraging existing technologies.

Examples

- Integration with social media platforms like Facebook and Twitter for sharing scores and game results.
- Using Google Analytics to monitor user activity and gather insights on gameplay trends.
- Integration with in-app purchase systems to unlock additional game content or features.

Considerations

- The development team must ensure that these integrations are seamless and that the partner applications are reliable and secure. Any changes or updates to these applications should be monitored to avoid affecting the game's functionality.

6d Off-the-Shelf Software

Content

- Off-the-shelf software refers to existing software products that can be used in the development of the "Two Truths and a Lie" game application. This includes tools and libraries that can help speed up development and reduce costs.

Motivation

- Using off-the-shelf software allows the team to focus on developing unique features for the game while relying on proven solutions for standard functionalities, which can save time and resources.

Examples

- Utilizing game development engines like Unity for creating the game environment.
- Implementing pre-built UI/UX components to ensure a consistent and professional interface.
- Using Firebase for backend services such as real-time database and authentication.

Considerations

- The development team must carefully evaluate the suitability of off-the-shelf software for the project's needs, considering factors like licensing, compatibility, and support to ensure it aligns with the game’s goals.

6e Anticipated Workplace Environment

Content

- The anticipated workplace environment describes the conditions under which the development team will work, including the tools, collaboration methods, and team dynamics necessary to develop the "Two Truths and a Lie" game application.

Motivation

- A well-organized workplace environment is essential for maintaining productivity and fostering collaboration among team members, which is crucial for the successful completion of the project.

Examples

- The team will utilize tools like Slack for communication, Trello for task management, and GitHub for version control.
- Regular stand-up meetings will be held to track progress and address any roadblocks.

- Remote work flexibility will be provided, allowing team members to contribute from different locations.

Considerations

- The team must ensure that all members have access to the necessary tools and resources to perform their tasks effectively. Any challenges related to remote collaboration or communication should be addressed promptly to maintain project momentum.

6f Schedule Constraints

Content

- Schedule constraints refer to the deadlines, milestones, and overall timeline for the "Two Truths and a Lie" game project. These constraints are crucial for ensuring that the game is developed and delivered on time.

Motivation

- Adhering to schedule constraints is vital to meet client expectations and avoid delays that could affect the game's release and market success. It also helps maintain team discipline and focus throughout the project.

Examples

- The project must be completed within a 4-month timeline, with major milestones including the completion of game design, development, and testing.
- A beta version of the game should be available by the end of the second month for user feedback.
- Final delivery is scheduled for the end of the fourth month, with all major features implemented and tested.

Considerations

- The team must regularly monitor progress against the schedule, adjusting as necessary to accommodate any delays or unforeseen

challenges. Buffer time should be built into the schedule to allow for final testing and any last-minute changes.

6g Budget Constraints

Content

- Budget constraints refer to the financial limitations placed on the "Two Truths and a Lie" game project, including costs related to development, licensing, marketing, and deployment.

Motivation

- Staying within the budget is essential for ensuring the financial viability of the project. Proper budget management allows the team to allocate resources effectively and ensures that the project delivers value without overspending.

Considerations

- The team must prioritize features and functionalities based on their cost and impact on the overall project. Regular budget reviews should be conducted to ensure that the project remains financially on track, and any potential cost overruns should be identified and addressed early.

7. Naming Conventions and Definitions

7a. Definitions of Key Terms

Content

- **Game Logic:** Refers to the rules and mechanics that govern how the "Two Truths and a Lie" game operates, including how truths and lies are handled.
- **User Profile:** Represents the data associated with a player, including their username, scores, and game history.
- **Session:** An instance of the game where players participate in a single round of "Two Truths and a Lie."

- **Scoreboard:** A feature that displays the current scores of all participants in the game.

Motivation

- Defining key terms helps ensure clear communication and understanding among developers, designers, and stakeholders. It also aids in maintaining consistency across the documentation and the application itself.

Examples

- **Game Logic:** The function that validates whether a statement is a truth or a lie.
- **User Profile:** User Profile class in the codebase containing fields like username, total Score, and played Games.

Considerations

- Ensure that the definitions are specific enough to avoid ambiguity and are aligned with the actual functionality of the game. Review definitions periodically to incorporate any new features or changes.

7b. UML and Other Notation Used in This Document

Content

- **UML (Unified Modelling Language):** Used for modeling the system's architecture, including class diagrams, sequence diagrams, and activity diagrams.
- **Class Diagrams:** Represent the structure of the application by showing classes, attributes, methods, and relationships between objects.
- **Sequence Diagrams:** Illustrate the sequence of interactions between system components during a particular operation or use case.
- **Activity Diagrams:** Visualize the flow of control in the system, representing different activities and decision points.

Motivation

- Using UML and other notations helps in visualizing the system architecture, data flow, and interactions among components. This makes it easier to understand and communicate the design and functionality of the application.

Examples

- **Class Diagram:** Illustrates the relationships between the Game, User Profile, and Session classes.
- **Sequence Diagram:** Shows how a user interacts with the game during a session.

Considerations

- Choose notations that best represent the structure and behavior of the application. Keep diagrams updated with changes in the system design and ensure they are easily understandable.

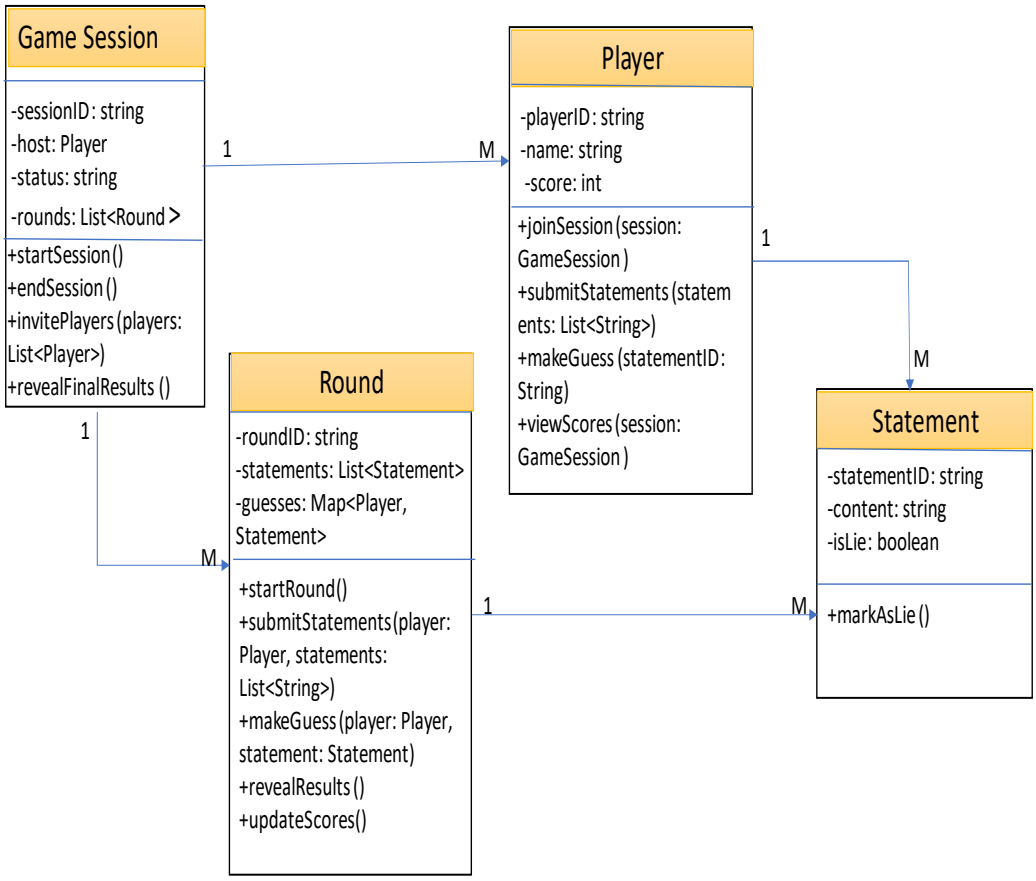

Figure 1 - Class diagram

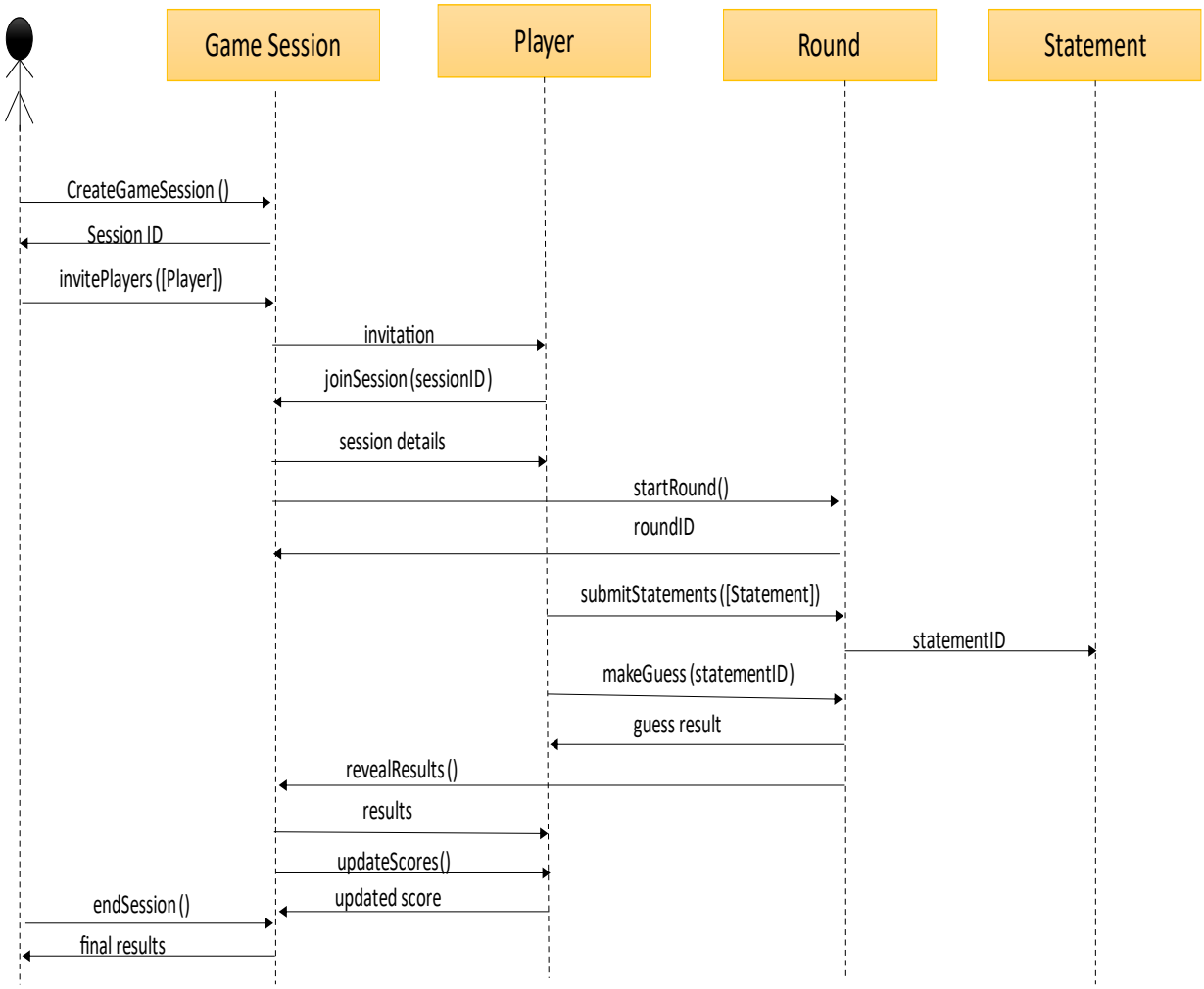

Figure 2 - Sequence diagram

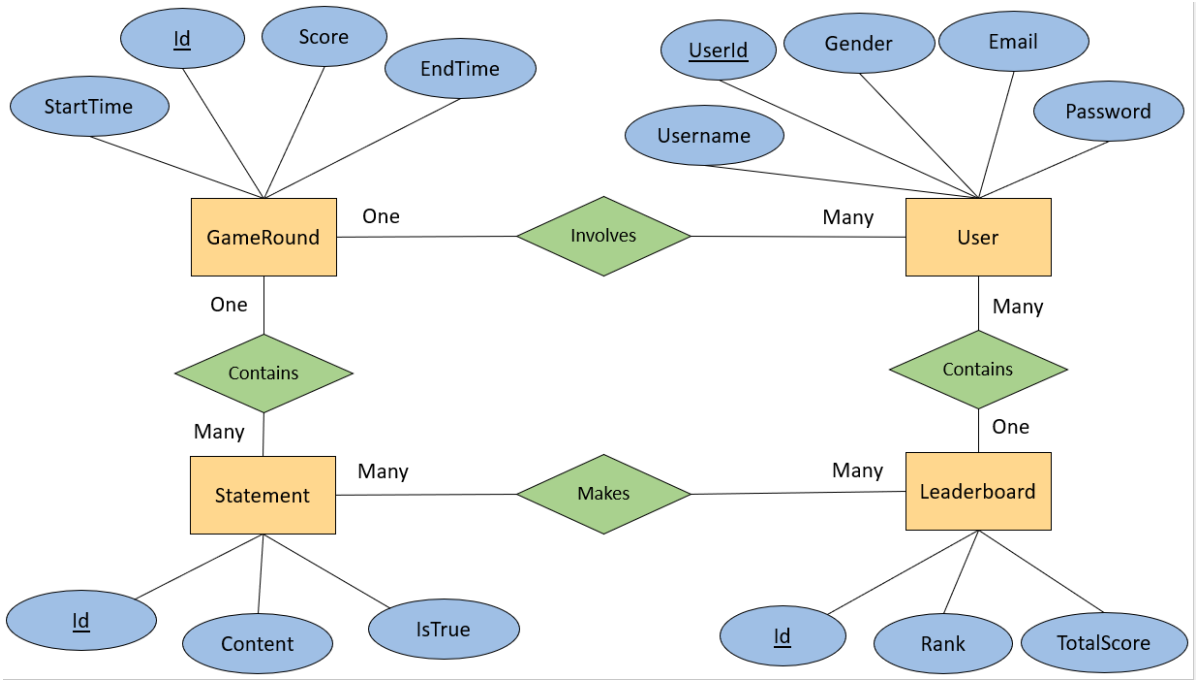

Figure 3 - ER diagram

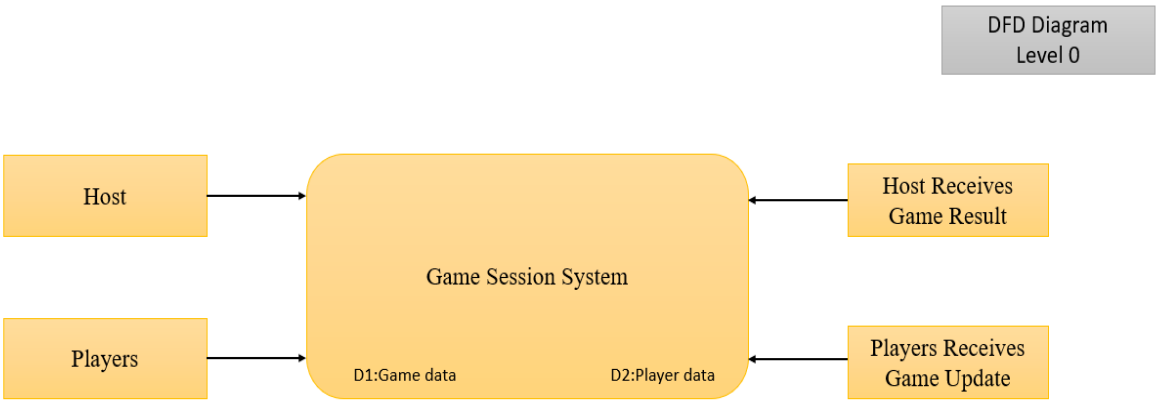

Figure 4 – DFD Level - 0

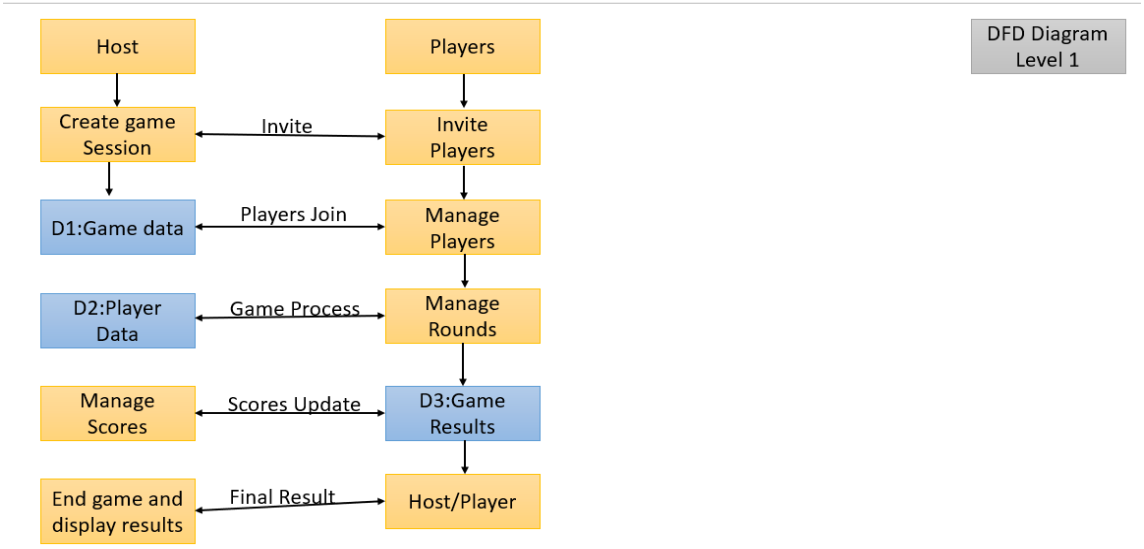

Figure 5 – DFD Level – 1

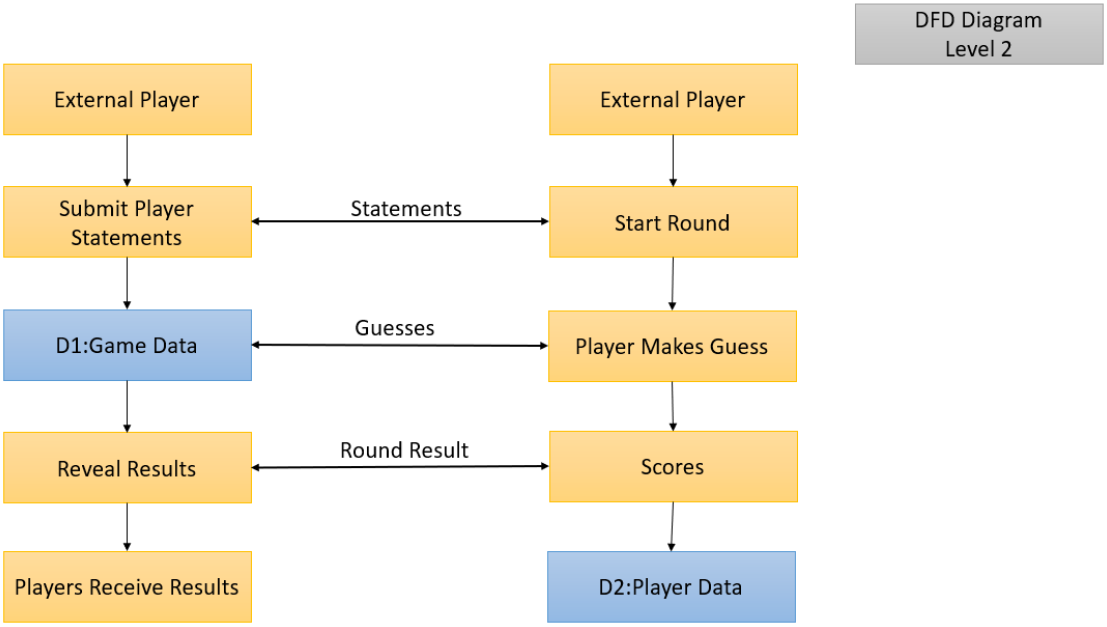

Figure 6 – DFD Level – 2

7c. Data Dictionary for Any Included Models

Content

- **Data Dictionary:** A comprehensive list of all data elements used in the application, including their names, types, and descriptions.
- **Model Definitions:** Detailed explanations of data models used in the game, including attributes and relationships.

Motivation

- A data dictionary helps in standardizing the data elements, ensuring consistency, and facilitating easier integration and maintenance of the application.

Examples

- **Player_ID**: Unique identifier for each player in the system. Data type: Integer.
- **Game_ID**: Unique identifier for each game session. Data type: Integer.
- **Statement_Text**: The text of a statement provided by a player. Data type: String.
- **Is_Lie**: A boolean value indicating whether the statement is a lie. Data type: Boolean.
- **Score_Value**: The score attributed to a player's correct guess. Data type: Integer.

Considerations

- Regularly update the data dictionary to reflect any changes in data models or new data elements. Ensure that it is comprehensive and accurately represents all data used in the application.

8. Relevant Facts and Assumptions

8a. Facts

Content

- **Game Concept:** The game "Two Truths and a Lie" involves players presenting three statements where two are true and one is a lie. The goal is for other players to identify the lie.
- **Technology Stack:** The application is built using HTML, CSS, JavaScript, and Python, with a MySQL database for storing user data and game sessions.
- **User Interaction:** Players can create profiles, join games, and participate in multiple sessions. Scores are tracked and displayed on a scoreboard.

Motivation

- Stating relevant facts provides a clear and factual basis for the design and implementation of the application. It ensures that all stakeholders have a shared understanding of the game's core features and technology.

Examples

- **Game Concept Fact:** In every game session, players are required to submit three statements, which will be evaluated to determine the lie.
- **Technology Stack Fact:** The backend is powered by Python with Flask, and MySQL is used for data persistence.

8b. Assumptions

Content

- **User Engagement:** It is assumed that players will actively participate and regularly play the game, leading to frequent updates to user profiles and scores.

- **Internet Connectivity:** The game assumes a stable internet connection for accessing online features, such as profile management and multiplayer sessions.
- **Device Compatibility:** The application is assumed to be used on modern web browsers and devices, which support HTML5, CSS3, and JavaScript.

Motivation

- Documenting assumptions helps in setting expectations and planning for potential challenges. It also helps in risk management and determining the scope of testing.

Examples

- **User Engagement Assumption:** Assumes that players will log in and engage with the game at least once a week, influencing the design of features like notifications and updates.
- **Internet Connectivity Assumption:** Assumes that users have access to a reliable internet connection, affecting the design of offline capabilities and error handling.

Considerations

- **User Engagement:** Consider implementing features that encourage regular participation, such as notifications or rewards.
- **Internet Connectivity:** Plan for scenarios with intermittent connectivity, such as implementing local storage or offline mode features.
- **Device Compatibility:** Ensure thorough testing across different devices and browsers to provide a consistent user experience.

II. Requirements

9. Product Use Cases

Visual representations that show the interactions between users (actors) and the system (use cases). These diagrams illustrate the primary functions of the application and how different user types interact with those functions.

9a. Use Case Diagrams

The use case diagrams for the "Two Truths and a Lie Game" application illustrate the interactions between users (players) and the system. The primary actors include the Player, Host, and the System (Application). The key use cases depicted are:

- **Start Game Session**: The Host initiates a new game session.
- **Join Game Session**: A Player joins an existing game session.
- **Submit Statements**: A Player submits two truths and one lie during their turn.
- **Guess the Lie**: Players make guesses to identify the lie among the statements presented.
- **View Scores**: Players check their scores after each round or at the end of the game.

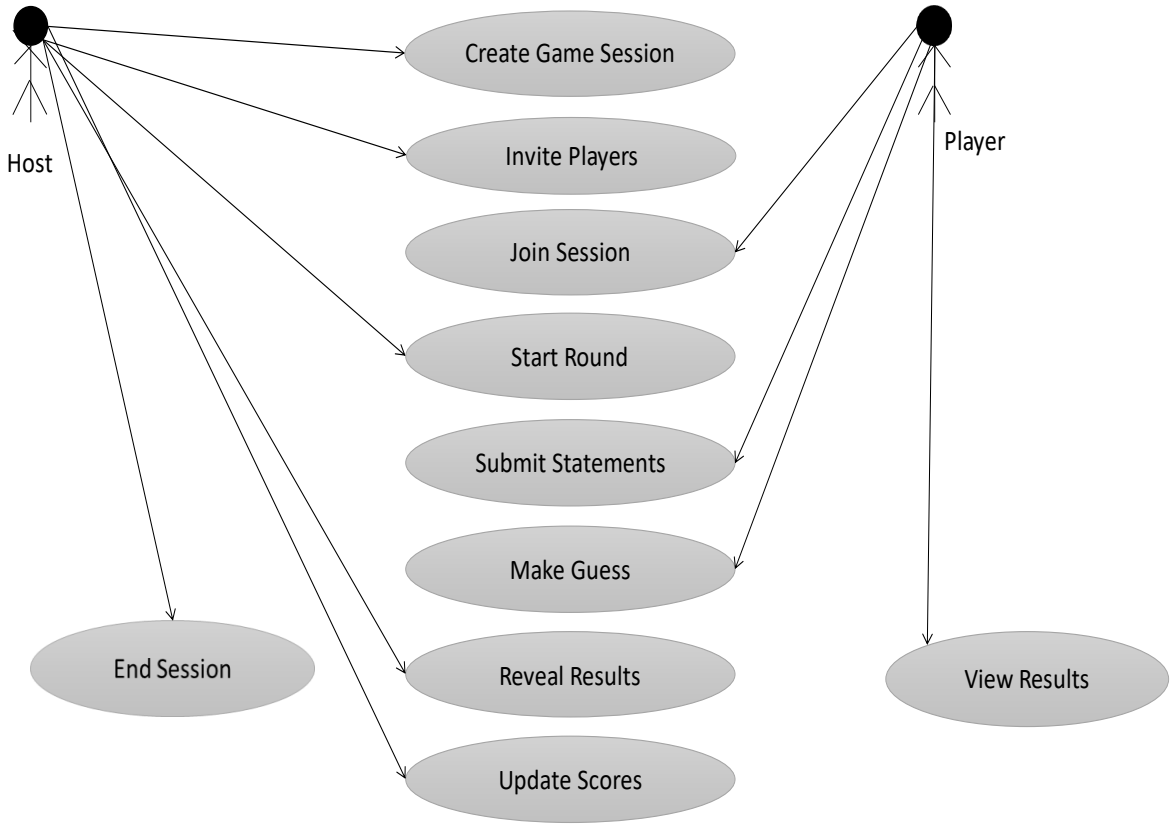

Figure 7 - Use Case diagram

9b. Product Use Case List

- **Start Game Session**: Host creates and starts a new game session.
- **Invite Players**: Host invites other players to join the session.
- **Join Game Session**: Players join a game session via a unique code or invitation link.
- **Submit Statements**: Player submits their set of statements (two truths and one lie).
- **Guess the Lie**: Players make guesses about which statement is the lie.
- **Reveal Results**: The system reveals the correct answer and updates scores.

- **View Scores**: Players can view their scores and ranking at the end of each round or game.
- **End Game Session**: The Host ends the game session and final results are displayed.

9c. Individual Product Use Cases

- **Start Game Session**:
    - **Actor**: Host
    - **Description**: The Host initiates a new game session by clicking the "Start Game" button, generating a unique session ID for players to join.
    - **Preconditions**: Host must be logged into the application.
    - **Postconditions**: A new game session is created and ready for players to join.
- **Submit Statements**:
    - **Actor**: Player
    - **Description**: During their turn, the Player submits two truths and one lie. These statements are displayed to other players for guessing.
    - **Preconditions**: The Player must be in an active game session.
    - **Postconditions**: The statements are submitted and await guesses from other players.
- **Guess the Lie**:
    - **Actor**: Player

- **Description**: Players guess which of the three statements presented by another player is the lie.
- **Preconditions**: The Player must be in an active game session and it must be their turn to guess.
- **Postconditions**: The guess is recorded, and the system awaits all players' guesses before revealing the correct answer.

10. Functional Requirements

Content

- **Functional Requirements:** These define the specific behaviors and functions the application must support. Each requirement should describe a feature or capability that the system must provide.

Motivation

- Clearly defined functional requirements are crucial for ensuring that all stakeholders understand what the application is expected to do. They guide the development process and help in validating that the final product meets user needs and expectations.

Examples

- **Requirement 1:** User Registration
    - **Description:** The system must allow users to create an account by providing a username, email, and password.
- **Requirement 2:** Game Session Management
    - **Description:** The system must support the creation, joining, and management of game sessions, including handling multiple players.
- **Requirement 3:** Statement Submission

    - **Description:** Users must be able to submit three statements in each game session, including two truths and one lie.

- **Requirement 4:** Scoring System

    - **Description:** The system must calculate and display scores based on user performance, such as correct identification of lies.

Fit Criterion

- **Requirement 1:** Users can successfully register and receive a confirmation email.
- **Requirement 2:** Game sessions can be created, joined, and managed with a maximum of 10 players.
- **Requirement 3:** Users can submit their statements without errors, and the system correctly identifies and processes them.
- **Requirement 4:** Scores are updated in real-time and accurately reflect player performance.

Considerations

- Ensure all functional requirements are testable and traceable to ensure they are properly implemented.
- Consider potential edge cases and how the system will handle unexpected inputs or scenarios.
- Regularly review and update requirements to accommodate any changes in user needs or system capabilities.

11. Data Requirements

Content

- **Data Requirements:** These specify the data that the system must collect, store, and manage. They include details about data types, data sources, and data handling processes.

Motivation

- Defining data requirements ensures that the application can effectively manage the information it needs to operate. It also helps in designing the database schema and ensuring data integrity and security.

Examples

- **Requirement 1:** User Data
    - **Description:** The system must store user information including usernames, email addresses, hashed passwords, and user scores.
- **Requirement 2:** Game Data
    - **Description:** The system must track game sessions, including participant lists, statements submitted, and scores for each session.
- **Requirement 3:** Session History
    - **Description:** The system must maintain a history of completed sessions, including details of the statements and outcomes for future reference.

Considerations

- **Data Security:** Ensure that sensitive user information, such as passwords, is encrypted and protected against unauthorized access.
- **Data Integrity:** Implement validation checks to maintain accurate and consistent data throughout the application.
- **Scalability:** Design data storage solutions to handle increasing amounts of data as the user base and number of game sessions grow.
- **Compliance:** Ensure that data handling practices comply with relevant data protection regulations and standards.

12. Performance Requirements

12a. Speed and Latency Requirements

Content

- **Speed and Latency Requirements:** These define the acceptable performance metrics for the application, including response times and processing speeds that ensure a smooth user experience.

Motivation

- Performance requirements are essential to guarantee that the application operates efficiently and provides a responsive experience for users. They help in setting benchmarks for acceptable performance and identifying potential areas for optimization.

Examples

- **Requirement 1:** Game Startup Time
    - **Description:** The application should start a new game session within 5 seconds after the user initiates it.
- **Requirement 2:** Statement Submission
    - **Description:** Statements submitted by users should be processed and displayed within 2 seconds.
- **Requirement 3:** Score Update
    - **Description:** Scores should be updated and displayed to all players within 3 seconds of a game event.

Fit Criterion

- **Requirement 1:** Game sessions are consistently started within 5 seconds, as measured during peak usage times.
- **Requirement 2:** The system processes and displays submitted statements within 2 seconds, verified through performance testing.

- **Requirement 3:** Scores are updated in real-time with no more than a 3-second delay.

Considerations

- Optimize backend services and database queries to meet speed requirements.
- Monitor performance under varying loads to ensure the application remains responsive.
- Implement caching strategies to reduce latency and improve performance.

12b Precision or Accuracy Requirements

Content

- **Precision or Accuracy Requirements:** These specify the level of correctness and detail the system must achieve in processing data and delivering results.

Motivation

- High precision and accuracy are critical for maintaining the integrity of the game and ensuring that user interactions are handled correctly. This helps in providing a fair and reliable gaming experience.

Examples

- **Requirement 1:** Statement Validation
    - **Description:** The system must accurately differentiate between two truths and one lie in submitted statements, with a precision of 100%.
- **Requirement 2:** Score Calculation
    - **Description:** The score calculation must be accurate to within 0.1 points to ensure fair scoring.

Considerations

- Implement rigorous testing to ensure that the system meets accuracy requirements.
- Regularly review and update algorithms to maintain precision.
- Address any issues promptly to prevent inaccuracies from affecting gameplay.

12c Capacity Requirements

Content

- **Capacity Requirements:** These outline the system's ability to handle various levels of user load, data volume, and other capacity-related factors.

Motivation

- Capacity requirements ensure that the system can accommodate the expected number of users and data volume without performance degradation. They help in planning for scalability and resource allocation.

Examples

- **Requirement 1:** Concurrent Users
    - **Description:** The system must support up to 500 concurrent users without performance issues.
- **Requirement 2:** Data Storage
    - **Description:** The application must handle up to 1 TB of game data, including user profiles, game sessions, and scores.

Fit Criterion

- **Requirement 1:** The application can handle up to 500 concurrent users with no significant performance degradation, as verified through stress testing.
- **Requirement 2:** Data storage is scalable to accommodate up to 1 TB of data, with efficient data management and retrieval.

Considerations

- Plan for horizontal scaling and load balancing to manage high user volumes.
- Implement data archiving strategies to handle large volumes of historical data.
- Monitor system performance and capacity regularly to identify and address potential bottlenecks.

13 Dependability Requirements

13a Reliability Requirements

Content

- **Reliability Requirements:** These specify the expected performance of the application in terms of its ability to consistently perform its functions correctly over time without failures.

Motivation

- Reliability is crucial for ensuring that the application performs its intended functions correctly and consistently, providing a stable and dependable user experience.

Examples

- **Requirement 1:** Error Rate
    - **Description:** The application should have an error rate of less than 1% for all user interactions, including game creation, statement submission, and score calculations.
- **Requirement 2:** System Uptime
    - **Description:** The system should be operational and performing as expected 99.9% of the time.

Considerations

- Implement thorough testing and quality assurance processes to minimize bugs and errors.
- Use monitoring tools to track the error rate and system performance in real-time.
- Regularly review and update the system to address any identified reliability issues.

13b Availability Requirements

Content

- **Availability Requirements:** These define the percentage of time the application should be accessible and operational to users, including any planned downtime for maintenance.

Motivation

- High availability ensures that users can access and use the application whenever they need it, which is essential for maintaining user satisfaction and trust.

Examples

- **Requirement 1:** Uptime
    - **Description:** The application should be available 24/7 with an uptime of 99.9%.
- **Requirement 2:** Maintenance Downtime
    - **Description:** Planned maintenance should not exceed 2 hours per month, and users should be notified in advance.

Considerations

- Implement redundant systems and failover mechanisms to maintain availability during unexpected outages.
- Schedule maintenance during off-peak hours to minimize user impact.
- Use automated monitoring and alerting systems to quickly identify and address availability issues.

13c Robustness or Fault-Tolerance Requirements

Content

- **Robustness or Fault-Tolerance Requirements:** These define the system's ability to handle and recover from unexpected errors or failures without significant disruption to the user experience.

Motivation

- Robustness and fault-tolerance are crucial for ensuring that the application can handle errors gracefully and continue operating smoothly, even when issues arise.

Examples

- **Requirement 1:** Error Handling
    - **Description:** The system should handle and recover from errors such as invalid inputs or network failures without crashing.
- **Requirement 2:** Data Recovery
    - **Description:** The system should be able to recover data and resume operations within 5 minutes in the event of a database failure.

Considerations

- Implement error handling and exception management to gracefully handle and log errors.
- Regularly test fault-tolerance mechanisms to ensure they work as expected under different failure scenarios.
- Ensure data backup and recovery procedures are in place and tested regularly.

13d. Safety-Critical Requirements

Content

- **Safety-Critical Requirements:** These specify the measures that need to be in place to ensure that the application does not cause harm to users or result in unsafe conditions.

Motivation

- While “Two Truths and a Lie” is generally not safety-critical, it's important to consider safety aspects to protect user data and ensure that the application does not inadvertently cause harm.

Examples

- **Requirement 1:** Data Protection
    - **Description:** The application must ensure user data is securely stored and transmitted, adhering to data protection regulations.
- **Requirement 2:** User Privacy
    - **Description:** The system should not expose any personal information of users without their consent.

Fit Criterion

- **Requirement 1:** User data is encrypted both at rest and in transit, and regular security audits are conducted.
- **Requirement 2:** User privacy is maintained by implementing strict access controls and data protection policies.

Considerations

- Implement robust data encryption and access controls to protect user information.
- Ensure compliance with relevant data protection laws and regulations.
- Conduct regular security assessments to identify and address potential vulnerabilities.

14. Maintainability and Supportability Requirements

14a. Maintenance Requirements

Content

- **Maintenance Requirements:** These specify how the system should be maintained over its lifecycle, including the ease of updating, fixing bugs, and performing upgrades.

Motivation

- Effective maintenance practices ensure that the application remains functional, secure, and up-to-date. It helps in managing technical debt and responding to user feedback and evolving needs.

Examples

- **Requirement 1:** Code Documentation
    - **Description:** All code must be well-documented with inline comments and comprehensive developer guides to facilitate easy updates and bug fixes.
- **Requirement 2:** Modular Architecture
    - **Description:** The application must be designed using modular architecture to isolate components and simplify maintenance and upgrades.

Considerations

- Establish a process for regular code reviews and updates.
- Provide training and documentation for developers to ensure smooth maintenance operations.
- Plan for periodic refactoring to address technical debt and improve code quality.

14b. Supportability Requirements

Content

- **Supportability Requirements:** These define how the application should support users and administrators, including help resources, troubleshooting, and user support channels.

Motivation

- Supportability ensures that users and administrators can get assistance when needed, which is critical for resolving issues quickly and maintaining user satisfaction.

Examples

- **Requirement 1:** Help Documentation
    - **Description:** The application must include comprehensive help documentation and user guides accessible from within the app.
- **Requirement 2:** Support Channels
    - **Description:** Users should have access to support channels such as email, chat, or forums for reporting issues and seeking assistance.

Considerations

- Implement a ticketing system for tracking and managing support requests.
- Regularly update help documentation and FAQs based on user feedback and common issues.
- Train support staff to handle a wide range of user queries and technical issues.

14c. Adaptability Requirements

Content

- **Adaptability Requirements:** These specify how the application should accommodate changes in user needs, technology, or business processes.

Motivation

- Adaptability ensures that the application can evolve in response to new requirements or changes in the environment without requiring major rework or redesign.

Examples

- **Requirement 1:** Feature Toggles
    - **Description:** The system should support feature toggles to enable or disable features without requiring code changes or redeployment.
- **Requirement 2:** Configurable Settings
    - **Description:** The application should provide configurable settings that allow administrators to adjust parameters and features as needed.

Fit Criterion

- **Requirement 1:** New features can be enabled or disabled via configuration settings without deploying new versions.
- **Requirement 2:** Administrators can adjust application settings through a user-friendly interface.

Considerations

- Design the system with flexibility in mind to accommodate future changes.
- Use configuration management tools to handle application settings and features.
- Document adaptability features to ensure they are used effectively.

14d. Scalability or Extensibility Requirements

Content

- **Scalability or Extensibility Requirements:** These define how the application should handle increasing loads or be extended with new features and functionalities.

Motivation

- Scalability and extensibility ensure that the application can grow with user demand and incorporate new features without significant rework or performance issues.

Examples

- **Requirement 1:** Horizontal Scaling
    - **Description:** The application should support horizontal scaling by adding more servers or instances to handle increased user load.
- **Requirement 2:** Plugin Architecture
    - **Description:** The system should use a plugin architecture to allow for easy integration of new features or third-party services.

Considerations

- Implement load balancing and clustering to manage increased traffic and user load.
- Design the system with modular components that can be extended or replaced without affecting the core functionality.
- Regularly review and test scalability strategies to ensure they meet performance and reliability goals.

14e. Longevity Requirements

Content

- **Longevity Requirements:** These specify the expected lifespan of the application and its ability to remain relevant and functional over time.

Motivation

- Ensuring longevity involves planning for the long-term maintenance and evolution of the application to ensure it continues to meet user needs and remain compatible with evolving technologies.

Examples

- **Requirement 1:** Legacy Support
    - **Description:** The application should support legacy systems and data formats to ensure compatibility with older versions and systems.
- **Requirement 2:** Technology Upgrades
    - **Description:** The system should be designed to facilitate technology upgrades, including updating libraries and frameworks with minimal disruption.

Considerations

- Plan for regular technology reviews to address deprecated technologies and update components as needed.
- Implement strategies for data migration and compatibility to support legacy systems.
- Ensure that the application can adapt to new technologies and industry standards.

15. Security Requirements

Security is crucial for the "Two Truths and a Lie Game" application to protect user data, ensure the integrity of the game, and maintain the confidentiality of personal information. The following subsections detail the specific security requirements.

15a. Access Requirements

Content

- **Access Requirements:** Define how users and administrators access the application and its features, including authentication mechanisms and user roles.

Motivation

- Proper access controls ensure that only authorized users can access or modify sensitive features and data, preventing unauthorized actions and maintaining system integrity.

Examples

- **Requirement 1:** User Authentication
    - **Description:** The application must require users to log in using a secure authentication mechanism (e.g., username and password) before accessing game features.
- **Requirement 2:** Role-Based Access Control
    - **Description:** Different user roles (e.g., regular players, administrators) should have appropriate access levels to various features and administrative functions.

Fit Criterion

- **Requirement 1:** Users must be able to log in securely with their credentials, and failed login attempts are logged.
- **Requirement 2:** Access controls are enforced, with role-specific permissions accurately implemented and verified.

Considerations

- Implement multi-factor authentication (MFA) for added security.
- Regularly review and update access controls based on user role changes.
- Ensure secure storage and handling of authentication credentials.

15b. Integrity Requirements

Content

- **Integrity Requirements:** Define measures to ensure the accuracy and consistency of data within the application, preventing unauthorized modifications or data corruption.

Motivation

- Data integrity is critical to ensure that user-generated content, game results, and other data remain accurate and trustworthy.

Examples

- **Requirement 1:** Data Validation
    - **Description:** The application must validate user inputs to prevent invalid or malicious data from being processed.
- **Requirement 2:** Data Integrity Checks
    - **Description:** Implement mechanisms to check for and prevent unauthorized changes to game data and user information.

Considerations

- Use input validation and sanitization techniques to prevent injection attacks.
- Implement checksums or hashing to verify data integrity.
- Regularly audit data integrity and address any detected issues.

15c. Privacy Requirements

Content

- **Privacy Requirements:** Define how user data is protected to ensure privacy and comply with relevant data protection regulations.

Motivation

- Protecting user privacy is essential to maintain trust and comply with legal and regulatory requirements regarding data protection.

Examples

- **Requirement 1:** Data Encryption
    - **Description:** All sensitive user data must be encrypted both at rest and in transit.
- **Requirement 2:** Data Minimization
    - **Description:** The application should collect only the minimum amount of personal data necessary for its functionality.

Considerations

- Regularly review privacy policies and practices to ensure compliance with data protection regulations.
- Implement secure encryption protocols and regularly update them as needed.
- Provide users with clear information about data collection and usage practices.

15d. Audit Requirements

Content

- **Audit Requirements:** Define the mechanisms for tracking and reviewing system activities to detect and investigate potential security issues or breaches.

Motivation

- Auditing is essential for monitoring system activity, detecting suspicious behaviour, and ensuring accountability for actions taken within the application.

Examples

- **Requirement 1:** Audit Logging
    - **Description:** The system must maintain detailed logs of user actions, system events, and administrative changes.
- **Requirement 2:** Log Review
    - **Description:** Implement regular review processes for audit logs to identify and respond to potential security incidents.

Considerations

- Ensure audit logs are protected from tampering and unauthorized access.
- Implement automated tools to analyse logs for unusual activity.
- Define procedures for responding to and investigating security incidents.

15e. Immunity Requirements

Content

- **Immunity Requirements:** Define measures to protect the application from known vulnerabilities and attacks, ensuring it remains resilient to threats.

Motivation

- Protecting the application from attacks and vulnerabilities is crucial for maintaining its security and preventing unauthorized access or damage.

Examples

- **Requirement 1:** Vulnerability Management
    - **Description:** Regularly update the application and its components to address known vulnerabilities and apply security patches.
- **Requirement 2:** Penetration Testing
    - **Description:** Conduct regular penetration tests to identify and address potential security weaknesses.

Considerations

- Stay informed about the latest security threats and vulnerabilities relevant to your application.
- Implement a process for applying security patches and updates in a timely manner.
- Engage with security professionals to perform thorough testing and review security practices.

16. Usability and Humanity Requirements

Usability and humanity factors are key to ensuring that the "Two Truths and a Lie Game" application is user-friendly and accessible to a wide audience. The following subsections outline the requirements related to ease of use, personalization, learning, and accessibility.

16a. Ease of Use Requirements

Content

- **Ease of Use Requirements:** These specify how user-friendly and intuitive the application should be, ensuring that users can easily navigate and interact with the game.

Motivation

- An easy-to-use interface improves user satisfaction and engagement by making it straightforward to start and play the game. This reduces the learning curve and enhances the overall experience.

Examples

- **Requirement 1:** Simple Navigation
    - **Description:** The game interface should have a clear and intuitive layout, allowing users to start a new game, join existing sessions, and submit their statements with minimal effort.
- **Requirement 2:** Consistent Design
    - **Description:** The application should maintain a consistent design across all screens, using familiar icons and language to guide users.

Fit Criterion

- **Requirement 1:** Users can start or join a game session and submit statements within 3 clicks or taps.
- **Requirement 2:** Design elements such as buttons and icons are consistent and recognizable throughout the application.

Considerations

- Conduct usability testing to identify and address any navigation issues.
- Gather user feedback to refine the interface and improve ease of use.
- Ensure that visual and interactive elements are intuitive and accessible.

16b. Personalization and Internationalization Requirements

Content

- **Personalization and Internationalization Requirements:** These define how the application should support user preferences and accommodate different languages and cultures.

Motivation

- Personalization enhances user engagement by tailoring the game experience to individual preferences. Internationalization ensures that the application is accessible to a global audience by supporting multiple languages and cultural norms.

Examples

- **Requirement 1:** User Profiles
    - **Description:** Users should be able to personalize their profiles with avatars, usernames, and preferences for game settings.
- **Requirement 2:** Language Support
    - **Description:** The application should support multiple languages and allow users to switch between them easily.

Considerations

- Implement localization strategies to support different languages and cultural contexts.
- Provide options for users to customize their experience, such as theme preferences or avatar choices.
- Regularly update language translations and cultural content to ensure accuracy and relevance.

16c. Learning Requirements

Content

- **Learning Requirements:** These specify how the application should facilitate learning for new users and help them understand how to play the game effectively.

Motivation

- Effective learning resources and onboarding processes help new users quickly become familiar with the game's rules and features, leading to a more enjoyable experience.

Examples

- **Requirement 1:** Tutorial
    - **Description:** The application should offer an interactive tutorial or onboarding experience that explains the game rules and how to play.
- **Requirement 2:** Help Resources
    - **Description:** Users should have access to a help section with clear explanations and examples of game mechanics.

Considerations

- Design the tutorial to be engaging and informative, covering all essential aspects of gameplay.
- Provide easily accessible help resources within the app to assist users as needed.
- Update learning materials based on user feedback and changes to game features.

16d. Understandability and Politeness Requirements

Content

- **Understandability and Politeness Requirements:** These define how clearly the application should communicate with users and how it should maintain a polite and respectful tone.

Motivation

- Clear communication and a polite tone improve user satisfaction and help create a positive experience. Ensuring that all messages and instructions are easily understood helps users engage with the application effectively.

Examples

- **Requirement 1:** Clear Instructions
    - **Description:** All in-game instructions, notifications, and error messages should be written in clear and simple language.
- **Requirement 2:** Polite Interaction
    - **Description:** The application should use polite and respectful language in all interactions with users.

Considerations

- Review all user-facing text to ensure clarity and politeness.
- Conduct user testing to identify any areas where communication could be improved.
- Regularly update text and messages based on user feedback and evolving standards.

16e. Accessibility Requirements

Content

- **Accessibility Requirements:** These define how the application should be designed to be accessible to users with disabilities, ensuring that everyone can participate in the game.

Motivation

- Accessibility ensures that users with various disabilities can enjoy the game, promoting inclusivity and compliance with accessibility standards.

Examples

- **Requirement 1:** Screen Reader Compatibility
    - **Description:** The application should be compatible with screen readers and provide alternative text for visual elements.
- **Requirement 2:** Keyboard Navigation
    - **Description:** The game should support full keyboard navigation for users who cannot use a mouse.

Considerations

- Follow accessibility guidelines such as WCAG (Web Content Accessibility Guidelines) to ensure compliance.
- Test the application with different assistive technologies to identify and address accessibility issues.
- Regularly review and update accessibility features to accommodate new standards and user needs.

16f. User Documentation Requirements

Content

- **User Documentation Requirements:** These specify the documentation that should be provided to users, including manuals, FAQs, and online help resources.

Motivation

- User documentation helps users understand how to use the application and troubleshoot common issues, enhancing their overall experience and reducing support requests.

Examples

- **Requirement 1:** User Manual
    - **Description:** The application should include a user manual or online guide that covers all game features and functionalities.
- **Requirement 2:** FAQ Section
    - **Description:** The application should have an FAQ section addressing common questions and issues.

Considerations

- Ensure that user documentation is comprehensive, up-to-date, and easy to understand.
- Provide documentation in multiple languages if the application supports international users.
- Update documentation based on user feedback and changes to the application.

16g. Training Requirements

Content

- **Training Requirements:** These specify the training materials and resources needed to help users and administrators effectively use and manage the application.

Motivation

- Training materials ensure that users and administrators understand how to use the application effectively, which is essential for maximizing its benefits and minimizing issues.

Examples

- **Requirement 1:** Training Videos
    - **Description:** Provide training videos demonstrating key features and gameplay strategies.
- **Requirement 2:** Administrator Training
    - **Description:** Offer training resources for administrators on managing game sessions, user accounts, and support tasks.

Considerations

- Develop training materials that are engaging and easy to follow.
- Offer training resources in multiple formats (videos, manuals, interactive guides) to accommodate different learning styles.
- Provide ongoing training opportunities as the application evolves and new features are introduced.

17. Look and Feel Requirements

The look and feel of the "Two Truths and a Lie Game" application are essential to creating a visually appealing and engaging user experience. The following subsections outline the specific requirements for appearance and style.

17a. Appearance Requirements

Content

- **Appearance Requirements:** Define how the game's user interface (UI) should look, including layout, color schemes, typography, and graphical elements.

Motivation

- A well-designed appearance ensures that the application is visually appealing and user-friendly, enhancing the overall user experience and engagement.

Examples

- **Requirement 1:** Color Scheme
    - **Description:** The application must use a color scheme that is consistent with the game's theme and accessible to users with color blindness.
- **Requirement 2:** Layout
    - **Description:** The UI should have a clean and intuitive layout with easily identifiable game controls, player information, and feedback areas.

Fit Criterion

- **Requirement 1:** The color scheme is tested with color blindness tools and is consistent across all screens.
- **Requirement 2:** User feedback confirms that the layout is intuitive and that users can easily navigate and interact with game elements.

Considerations

- Ensure the design is responsive and looks good on various devices and screen sizes.
- Test the design with real users to gather feedback and make necessary adjustments.

17b. Style Requirements

Content

- **Style Requirements:** Define the visual and stylistic elements of the game, including design themes, iconography, and animations.

Motivation

- Consistent styling helps in creating a cohesive and immersive user experience, making the game visually engaging and memorable.

Examples

- **Requirement 1:** Iconography
    - **Description:** Use consistent and recognizable icons for actions like submitting statements, viewing scores, and starting new games.
- **Requirement 2:** Animations
    - **Description:** Include smooth animations for game transitions, score updates, and feedback to enhance the user experience.

Fit Criterion

- **Requirement 1:** Icons are designed to be easily recognizable and are used consistently throughout the application.
- **Requirement 2:** Animations are smooth and enhance the gameplay experience without causing delays or distractions.

Considerations

- Follow design guidelines for accessibility to ensure animations and visual elements are usable for all users.
- Maintain a balance between aesthetics and functionality to avoid overcomplicating the UI.

18. Operational and Environmental Requirements

Operational and environmental factors play a critical role in ensuring the smooth functioning of the application in different settings and conditions. The following subsections describe the operational and environmental requirements.

18a. Expected Physical Environment

Content

- **Expected Physical Environment:** Specify the physical conditions under which the application will operate, including hardware and network environments.

Motivation

- Understanding the physical environment helps in designing the application to perform optimally under various conditions and ensures compatibility with the target user devices.

Examples

- **Requirement 1:** Device Compatibility
    - **Description:** The application must be compatible with both desktop and mobile devices, including different screen resolutions and operating systems.
- **Requirement 2:** Network Conditions

- **Description:** The application should perform well under typical home and office network conditions, including various internet speeds.

Considerations

- Test the application on different devices and network conditions to ensure consistent performance.
- Optimize the application to handle lower bandwidth scenarios gracefully.

18b. Requirements for Interfacing with Adjacent Systems

Content

- **Requirements for Interfacing with Adjacent Systems:** Define how the application should interact with other systems, such as social media platforms or third-party services.

Motivation

- Proper interfacing ensures that the application can integrate with other systems as needed, providing additional features or data.

Examples

- **Requirement 1:** Social Media Integration
    - **Description:** The application should allow users to share their game achievements on social media platforms like Facebook and Twitter.
- **Requirement 2:** Third-Party Services
    - **Description:** The application must be able to integrate with third-party services for user authentication or data storage if applicable.

Fit Criterion

- **Requirement 1:** Users can successfully share their achievements on social media without errors.
- **Requirement 2:** Integration with third-party services works seamlessly and does not affect the application's performance.

Considerations

- Ensure secure and reliable communication with adjacent systems.
- Handle any potential errors or issues gracefully to maintain a smooth user experience.

18c. Productization Requirements

Content

- **Productization Requirements:** Define the steps and requirements for preparing the application for release, including packaging, distribution, and deployment.

Motivation

- Proper productization ensures that the application is ready for release and can be easily installed and used by the end-users.

Examples

- **Requirement 1:** Packaging
    - **Description:** The application should be packaged in formats suitable for distribution on various platforms (e.g., .apk for Android, .ipa for iOS).
- **Requirement 2:** Deployment
    - **Description:** Provide clear instructions for deployment and installation, including any prerequisites or dependencies.

Fit Criterion

- **Requirement 1:** The application can be packaged and distributed without issues on all target platforms.
- **Requirement 2:** Deployment instructions are clear, accurate, and lead to successful installation and setup.

Considerations

- Test the deployment process thoroughly to identify and resolve any potential issues.
- Ensure that packaging and distribution comply with platform guidelines and requirements.

18d. Release Requirements

Content

- **Release Requirements:** Define the criteria for releasing the application, including versioning, release notes, and post-release support.

Motivation

- Clearly defined release requirements ensure that the application is properly prepared for launch and that users have access to necessary information and support.

Examples

- **Requirement 1:** Versioning
    - **Description:** Implement a versioning system to track updates and changes to the application.
- **Requirement 2:** Release Notes
    - **Description:** Provide detailed release notes with each version, including new features, bug fixes, and any known issues.

Fit Criterion

- **Requirement 1:** The application uses a consistent and clear versioning system for all releases.
- **Requirement 2:** Release notes are comprehensive and accurately reflect the changes and updates in each version.

Considerations

- Plan and test the release process to ensure a smooth transition from development to production.
- Prepare for post-release support, including bug fixes and user feedback handling.

19. Cultural and Political Requirements

Cultural and political factors may influence the design and functionality of the application, especially when targeting a global audience. The following subsections address the relevant cultural and political requirements.

19a. Cultural Requirements

Content

- **Cultural Requirements:** Define how the application should address cultural considerations, including language, symbols, and cultural norms.

Motivation

- Addressing cultural requirements ensures that the application is inclusive and respectful of diverse user backgrounds, enhancing its acceptance and usability.

Examples

- **Requirement 1:** Localization
    - **Description:** The application should support multiple languages to accommodate users from different regions.

- **Requirement 2:** Cultural Sensitivity
    - **Description:** The game's content and design should be reviewed to avoid cultural insensitivity or offensive symbols.

Considerations

- Conduct cultural reviews and user testing to identify and address potential issues.
- Provide options for users to select their preferred language and cultural settings.

19b. Political Requirements

Content

- **Political Requirements:** Define any considerations related to political sensitivities, including compliance with local regulations and avoidance of controversial content.

Motivation

- Addressing political requirements helps ensure that the application complies with legal and regulatory standards and avoids content that could be politically sensitive or controversial.

Examples

- **Requirement 1:** Regulatory Compliance
    - **Description:** Ensure that the application complies with local laws and regulations related to data privacy, content, and user interactions.
- **Requirement 2:** Content Moderation
    - **Description:** Implement moderation mechanisms to prevent the inclusion of politically sensitive or controversial content in user-generated statements.

Considerations

- Stay informed about relevant laws and regulations in the target markets.
- Provide clear guidelines for content creation and moderation to maintain compliance and avoid issues.

20. Legal Requirements

Compliance with legal requirements is essential to protect the application and its users from legal issues. The following subsections outline the legal requirements for the application.

20a. Compliance Requirements

Content

- **Compliance Requirements:** These define the legal and regulatory obligations that the application must adhere to, including data protection, privacy laws, and any other relevant regulations.

Motivation

- Ensuring compliance with legal requirements is crucial to avoid legal issues, protect user data, and maintain trust with users. It helps in aligning the application with industry standards and legal frameworks.

Examples

- **Requirement 1:** Data Protection Regulations
    - **Description:** The application must comply with data protection laws such as the General Data Protection Regulation (GDPR) or the California Consumer Privacy Act (CCPA). This includes obtaining user consent for data collection, providing data access and deletion options, and ensuring data security.
- **Requirement 2:** Children's Online Privacy Protection

- **Description:** If the application is accessible to children under 13, it must comply with the Children's Online Privacy Protection Act (COPPA), including obtaining parental consent before collecting personal information from children.

Fit Criterion

- **Requirement 1:** The application obtains user consent for data collection and provides users with options to access and delete their data, as verified through compliance audits.
- **Requirement 2:** For users under 13, the application has mechanisms for obtaining parental consent and ensures compliance with COPPA regulations.

Considerations

- Regularly review and update compliance practices to align with changes in legal requirements.
- Implement data protection measures such as encryption and secure storage to protect user information.
- Provide clear privacy policies and terms of service to inform users about data handling practices.

20b. Standards Requirements

Content

- **Standards Requirements:** These specify the industry standards and best practices that the application must follow to ensure quality, security, and interoperability.

Motivation

- Adhering to industry standards helps ensure the application meets high-quality benchmarks, is secure, and integrates well with other systems. It also facilitates user trust and system reliability.

Examples

- **Requirement 1:** Security Standards
    - **Description:** The application should follow security standards such as the OWASP Top Ten to protect against common vulnerabilities and ensure secure coding practices.
- **Requirement 2:** Accessibility Standards
    - **Description:** The application should comply with Web Content Accessibility Guidelines (WCAG) to ensure it is accessible to users with disabilities.

Fit Criterion

- **Requirement 1:** The application is tested and found to be compliant with OWASP Top Ten recommendations, including measures against SQL injection, cross-site scripting, and other vulnerabilities.
- **Requirement 2:** The application meets WCAG 2.1 AA standards, as verified by accessibility testing tools and user feedback.

Considerations

- Implement regular security assessments and updates to address vulnerabilities and comply with evolving security standards.
- Ensure accessibility features are tested with actual users who have disabilities to verify compliance with accessibility standards.
- Stay informed about updates to standards and best practices to maintain compliance and enhance the application.

III. Design

21. System Design

The system design for the "Two Truths and a Lie Game" application is centered around creating a user-friendly, engaging, and efficient platform that allows users to play the game seamlessly. The design aims to balance simplicity and functionality while ensuring a smooth user experience across various devices.

21a. Design Goals

Content

- The primary goal of the system design is to create an interactive game environment where users can enjoy the "Two Truths and a Lie" format. The game requires users to input three statements, and other players have to identify which one is false. The design goals focus on:
    - **Usability:** The system must be easy to use, with a clear and intuitive user interface.
    - **Scalability:** The application should handle multiple users playing simultaneously without affecting performance.
    - **Responsiveness:** The design must ensure a smooth experience on both mobile devices and desktops.
    - **Security:** User data, including game progress and personal information, must be securely stored and processed.
    - **Engagement:** The design should incorporate elements such as notifications, score tracking, and leaderboards to keep users engaged.

Motivation

- The motivation for this design is to create an enjoyable and accessible experience for users of all ages. By focusing on a clean interface and fun

gameplay mechanics, the game encourages social interaction and creativity, as users come up with unique truths and lies. The system's design also aims to support social connections through multiplayer capabilities and sharable game results.

Considerations

- **User Interface (UI):** The interface needs to be simple, with minimal distractions, while offering a vibrant and fun aesthetic to attract players. Icons, buttons, and text should be well-placed and sized for ease of use.
- **Game Logic:** The design should focus on creating a seamless flow of gameplay, from inputting statements to voting on the lie, ensuring minimal delay between game phases.
- **Multiplayer Support:** The system should allow multiple players to join a game room either locally or remotely, and the design should include real-time updates for all players during gameplay.
- **Data Storage:** Player data, including scores, game history, and user profiles, should be securely stored in a cloud database, with quick retrieval for future game sessions.
- **Error Handling:** The design should handle errors smoothly, such as missing inputs, network issues, or incorrect data submissions, to ensure a consistent game experience.

Example

For instance, a user opens the application and navigates to a new game. They input three statements:

- "I have visited 10 countries."
- "I can speak 4 languages."
- "I have a twin sibling."

The system presents these statements to other players, who then select which one they believe is the lie. After all players submit their guesses, the system reveals the correct answer, updates scores, and moves to the next round. The design ensures that this process is smooth and quick, with minimal load times and clear visual feedback for each phase of the game.

22. Current Software Architecture

Content

- The current software architecture of the "Two Truths and a Lie" game application is designed to be simple, scalable, and user-friendly. The architecture follows a client-server model where the frontend is responsible for user interaction, while the backend handles the game logic, user data, and communication between users during the game.

Motivation

- The main motivation behind this architecture is to ensure a seamless and interactive user experience. By separating the frontend and backend, the application can efficiently manage multiple users, process game rounds, and allow real-time updates without slowing down the interface. This design also ensures that the game can scale with additional features or a larger user base.

Considerations

Key considerations for the architecture include:

- **Performance:** The game should handle multiple users playing simultaneously without performance degradation. This requires efficient data handling and minimal latency.
- **Scalability:** As more users join, the application must be capable of scaling horizontally by adding more servers or resources.

- **Security:** User data, such as player scores and login credentials, must be protected. Secure authentication and data encryption are critical considerations.
- **User Interface (UI):** The architecture needs to support a responsive and engaging UI that works well across devices (mobile, desktop).
- **Maintenance and Extensibility:** The system should be easy to maintain, and adding new features like leaderboards or advanced game modes should not require a major overhaul.

Example

An example of the architecture could involve a three-tier design:

- **Frontend**: The user interface is developed using technologies like HTML, CSS, and JavaScript (React or Angular). It presents game elements, including options for entering truths and lies, game results, and user statistics.
- **Backend (API Layer)**: The backend is built using a framework like Python (Flask or Django), handling game logic, real-time interaction, and user authentication. It exposes APIs for frontend communication and integrates with the database to store game-related data.
- **Database**: A MySQL or Firebase database is used for storing user profiles, game records, and gameplay statistics. It ensures fast and reliable data retrieval for real-time updates during gameplay.

23. Proposed Software Architecture

23a. Overview

The proposed software architecture for the "Two Truths and a Lie" game is designed to enhance performance, scalability, and user experience by leveraging modern technologies and best practices in software engineering. The architecture follows a microservices-based model, which divides the application into smaller, independent services that can be developed, deployed, and scaled independently.

Content

The core components of the proposed architecture include:

- **Frontend (Client-side):** A responsive and interactive user interface built with a modern JavaScript framework (e.g., React or Vue.js) that works across devices and platforms.
- **Backend (Server-side):** A microservices-based backend to handle game logic, user management, and real-time communication between players.
- **Database:** A scalable, distributed database system (e.g., MongoDB, PostgreSQL) for storing user data, game history, and other persistent data.
- **APIs:** RESTful APIs for communication between the frontend, backend, and database, and WebSocket APIs for real-time game updates.
- **Authentication:** Secure authentication services (e.g., OAuth, JWT) to manage user sessions and secure game data.
- **Game Server:** A dedicated game server to manage game state, handle user inputs, and synchronize game data between players in real time.

Motivation

The motivation for this architecture is to build a robust and scalable system that can handle a large number of concurrent users and provide a smooth, real-time

gaming experience. The microservices approach allows each component to be independently developed, tested, and scaled, leading to faster development cycles, better fault isolation, and easier integration of new features.

Key motivations include:

- **Performance:** Enhancing the real-time interaction experience with minimal latency.
- **Scalability:** Ensuring that the system can handle thousands of concurrent users as the game grows in popularity.
- **Maintainability:** Facilitating easier updates and bug fixes without affecting the entire application.
- **Extensibility:** Allowing for the addition of new features, such as leaderboards, multiplayer modes, and user-generated content, without overhauling the entire system.

Considerations

Key considerations for the proposed architecture include:

- **Real-time Communication:** The game requires low-latency real-time communication between players, which can be achieved through WebSocket's or real-time database solutions (e.g., Firebase, Socket.io).
- **Scalability:** The microservices architecture enables horizontal scaling. Individual services, such as the game server or user authentication, can be scaled independently to meet demand.
- **Security:** Strong authentication mechanisms like OAuth and JWT tokens should be implemented to protect user data and ensure secure game sessions.
- **Database Scalability:** A NoSQL database (e.g., MongoDB) or a distributed SQL solution (e.g., PostgreSQL with sharding) can be used to manage large datasets and maintain high performance under heavy loads.

- **Fault Tolerance and Redundancy:** The architecture should be designed with redundancy to ensure that failures in one service do not bring down the entire application.

Example

A proposed implementation of the software architecture for "Two Truths and a Lie" might look like this:

- **Frontend (Client-side):**
  - A modern frontend framework (e.g., React or Vue.js) provides a fast, dynamic, and responsive user interface.
  - User actions (e.g., submitting truths and lies, voting) are handled via API calls and real-time WebSocket connections.
- **Backend (Microservices-based Server):**
  - **User Management Service:** Handles user authentication, session management, and profile updates.
  - **Game Logic Service:** Processes game rounds, verifies user inputs, and calculates game results.
  - **Real-Time Communication Service:** Manages real-time interactions between players via WebSocket's or Firebase.
- **Database:**
  - **MongoDB:** Stores user profiles, game history, and game results, offering scalability and flexibility in handling complex data structures.
  - **Redis:** Used for caching game state and user sessions to improve performance and reduce database load.
- **API Gateway:**
  - Acts as the entry point for all frontend requests, routing them to the appropriate microservice.
  - Ensures secure communication and handles load balancing.
- **Authentication:**

- Uses OAuth 2.0 for secure login and session management, ensuring that users' game data is protected.

23b. Class Diagrams

Content

Class diagrams are essential for modeling the structure of the "Two Truths and a Lie" game by showcasing the relationships between different objects and their properties and methods. The proposed class diagram will focus on key components of the game, including **Player**, **Game**, **Round**, **TruthsAndLies**, and **Score**.

- **Player Class**: Handles user information such as username, ID, score, and game history.
- **Game Class**: Manages the overall game flow, including the number of rounds, players, and game state.
- **Round Class**: Represents each round of the game, which includes a player's two truths and one lie.
- **TruthsAndLies Class**: Contains the specific details of the truths and the lie submitted by each player.
- **Score Class**: Calculates and stores each player's score based on the accuracy of guesses.

Motivation

The motivation behind the class diagram is to provide a clear and structured representation of how the core entities in the "Two Truths and a Lie" game interact. By creating a well-defined class diagram, the development team can ensure modularity, reusability, and clarity in code implementation. This diagram will help to visualize object-oriented relationships, making it easier to maintain and expand the game in the future.

The class diagram:

- Clarifies the structure of the software system.
- Helps break down complex logic into manageable components.
- Aids in understanding the flow and relationships of key entities within the game.

Considerations

When designing the class diagram for the game, key considerations include:

- **Modularity:** Each class should represent a distinct component of the game with a single responsibility.
- **Reusability:** Classes should be designed in a way that makes them reusable in different contexts, such as extending game modes or adding features like leaderboards.
- **Abstraction:** The class design should abstract away unnecessary complexity, focusing only on the properties and methods relevant to the core game mechanics.
- **Scalability:** The class design should allow for the game's growth, such as the introduction of new game types or variations.
- **Interaction Between Classes:** The relationships between classes, such as inheritance or associations, should be carefully modeled to reflect how data is shared and used within the game.

Class Diagram Representation:

In the diagram:

- **Player** has an association with **Game** (a player participates in multiple games).
- **Game** aggregates **Round** (a game contains multiple rounds).
- **Round** contains **TruthsAndLies** and **Score** (each round involves a player's submission of truths and lies and scoring based on guesses).

23c. Dynamic Model

Content

A dynamic model focuses on the behaviour of the system over time, especially how objects interact and respond to events. In the "Two Truths and a Lie" game, the dynamic model represents the sequence of interactions between players and the system, such as submitting truths and lies, making guesses, and receiving scores.

Key elements in the dynamic model include:

- **Player Interactions:** Players submit their two truths and a lie, make guesses about other players' statements, and receive feedback.
- **Game Flow:** The game progresses through multiple rounds, and the dynamic model outlines how the system handles transitions from one game phase to another.
- **System Response:** The model also demonstrates how the system validates inputs, processes guesses, calculates scores, and updates game states.

Motivation

The motivation behind the dynamic model is to ensure that the game system functions as expected when players interact with it in real-time. A dynamic model is crucial for visualizing the sequence of actions, message exchanges, and state transitions within the game. It helps in:

- **Clarifying workflows:** Ensuring the game's progression from one stage to another is smooth.
- **Debugging:** Identifying possible issues with the flow of interactions.
- **Optimization:** Improving the responsiveness of the system by analysing bottlenecks in interactions.

The dynamic model provides developers with a clear understanding of how different entities behave during game execution, helping in implementing robust game logic and real-time features.

Considerations

When designing the dynamic model for the game, several key considerations must be addressed:

- **Real-time Interaction:** The dynamic model should account for real-time feedback between players during guessing rounds and score updates.
- **Synchronous vs. Asynchronous Events:** Determining which actions (e.g., submitting guesses, processing results) should occur synchronously and which can be handled asynchronously to avoid latency.
- **Error Handling:** The dynamic model should incorporate how the system reacts to invalid inputs, connection issues, or incomplete rounds.
- **State Transitions:** Carefully manage state transitions, such as moving from the submission phase to the guessing phase, to avoid inconsistencies.

Example

An example of the dynamic model in action for a single round of the "Two Truths and a Lie" game might involve the following interactions:

- **Submission Phase:**
  - **Player Action:** A player submits two truths and one lie.
  - **System Response:** The system validates the submission to ensure the format is correct (i.e., exactly two truths and one lie). If valid, the game state moves to the next phase.
- **Guessing Phase:**

- **Player Action:** Other players receive the submitted truths and lie and make guesses about which statement is the lie.
- **System Response:** The system records each player's guess.

- **Processing Phase:**
  - **System Action:** The system processes the guesses and determines whether players correctly identified the lie. Scores are updated accordingly.
- **Feedback Phase:**
  - **System Response:** The system sends feedback to each player, indicating the correct answer and their updated score.
  - **Player Action:** Players receive their updated scores and see whether they guessed correctly.
- **Transition to Next Round:**
  - **System Action:** Once all players have completed the round, the system automatically transitions to the next round, repeating the process.

Dynamic Model Representation Example:

Here's how the interactions would be represented:

- **Sequence Diagram:**
  - **Actors:** Player, Game System
  - **Messages:**
    - Player sends truths and lie to the system.
    - System validates input.
    - Other players receive the truths and lie.
    - Players submit their guesses.
    - System processes guesses and calculates scores.

- System sends feedback to players.

- **State Diagram:**
  - **States:**
    - Waiting for Submission → Validating Submission → Waiting for Guesses → Processing Guesses → Updating Scores → Next Round
  - **Transitions:** Players' actions (submitting, guessing) trigger transitions between game states, ensuring smooth progression from one phase to the next.

23d. Subsystem Decomposition

Content

Subsystem decomposition involves breaking down the "Two Truths and a Lie" game into smaller, manageable components or subsystems that work together to deliver the game's functionality. Each subsystem handles a specific part of the application, such as user management, game logic, real-time communication, and scoring.

The key subsystems for this game include:

- **User Management Subsystem:** Handles user authentication, profiles, and sessions.
- **Game Management Subsystem:** Manages the game flow, including player turns, rounds, and game states.
- **Truth and Lie Subsystem:** Processes the submission of truths and lies and manages validation.
- **Guessing and Scoring Subsystem:** Handles the process of player guesses and score calculation.

- **Real-Time Communication Subsystem:** Manages real-time interactions between players, such as WebSocket communication.
- **Database Subsystem:** Responsible for data storage, such as user profiles, game history, and scores.

Motivation

The motivation for subsystem decomposition is to divide the application into smaller, self-contained components, each with a single responsibility. This makes the system more maintainable, scalable, and easier to understand. Decomposing the system also allows for parallel development, where different teams can work on different subsystems independently.

The key goals of subsystem decomposition are:

- **Maintainability:** Subsystems make the system easier to manage, allowing for isolated updates and debugging.
- **Scalability:** Each subsystem can be scaled independently, optimizing resource usage.
- **Flexibility:** The system can be easily extended by adding new features to individual subsystems without affecting the whole architecture.
- **Modularity:** Subsystems create a clear separation of concerns, making each part of the system modular and easier to understand.

Considerations

When designing subsystems, the following considerations must be taken into account:

- **Inter-Subsystem Communication:** Subsystems must communicate with each other efficiently, especially for real-time interactions. This can be achieved using APIs, message queues, or event-driven architectures.

- **Loose Coupling:** Subsystems should be loosely coupled to ensure changes in one subsystem do not heavily impact others. This promotes flexibility and easier maintenance.
- **Security:** Each subsystem, especially user management and game data handling, should ensure data security and prevent unauthorized access.
- **Performance Optimization:** The decomposition should optimize the performance of each subsystem, ensuring that real-time communication and processing tasks are handled efficiently.
- **Redundancy and Fault Tolerance:** Each subsystem should be designed to handle failures gracefully without affecting the overall system.

Example

Here is an example of how the "Two Truths and a Lie" game could be decomposed into subsystems:

- **User Management Subsystem**
  - **Responsibilities:**
    - User registration and login.
    - Session management.
    - Storing and retrieving user profiles.
  - **Technologies:** OAuth or JWT for authentication, integrated with a user database (e.g., MySQL, Firebase).
- **Game Management Subsystem**
  - **Responsibilities:**
    - Creating new games and managing existing games.

        - Handling game states (e.g., submission phase, guessing phase).
        - Managing player turns and transitions between game rounds.
    - **Technologies:** REST APIs or gRPC for managing game logic and state transitions.
- **Truth and Lie Subsystem**
    - **Responsibilities:**
        - Processing player submissions of two truths and one lie.
        - Validating the format of submissions.
        - Displaying the statements to other players.
    - **Technologies:** Backend service written in Python/Node.js to handle validation and game logic.
- **Guessing and Scoring Subsystem**
    - **Responsibilities:**
        - Accepting and processing player guesses about which statement is a lie.
        - Calculating scores based on correct or incorrect guesses.
        - Updating the score for each player at the end of a round.
    - **Technologies:** Scoring algorithms implemented on the server-side, with results stored in the database.
- **Real-Time Communication Subsystem**

- **Responsibilities:**
    - Handling real-time interaction between players during each round (e.g., submitting guesses, receiving results).
    - Ensuring low-latency updates for actions such as guesses and score updates.
- **Technologies:** WebSocket protocol or Firebase real-time database to enable real-time communication.

- **Database Subsystem**
    - **Responsibilities:**
        - Storing user data, game history, and score records.
        - Managing database operations such as queries and updates.
    - **Technologies:** NoSQL (e.g., MongoDB) or SQL-based systems (e.g., MySQL) to store game and user-related data.

Subsystem Decomposition Example:

In this example, the **Game Management Subsystem** would interact with the **Truth and Lie Subsystem** to validate player submissions, then hand off the process to the **Guessing and Scoring Subsystem** for scoring. The **Real-Time Communication Subsystem** would ensure that all players receive real-time updates on guesses and scores.

Each subsystem communicates via well-defined APIs, ensuring loose coupling and allowing each to be developed, tested, and deployed independently. For instance:

- When a player submits a guess, the **Real-Time Communication Subsystem** forwards the guess to the **Guessing and Scoring Subsystem**,

which processes the input and updates the scores, then informs the **Game Management Subsystem** to transition to the next phase or round.

23e. Hardware/Software Mapping

Content

Hardware/software mapping assigns software components to hardware, ensuring optimal performance and scalability. In "Two Truths and a Lie," key components include:

- **Frontend Devices:** Player interactions via mobile/web apps.
- **Backend Servers:** Handle game logic, user management, and scoring.
- **Database Servers:** Store user profiles, game history, and scores.
- **Real-Time Communication:** Supports live updates using WebSocket or Firebase.
- **Load Balancers:** Distribute traffic across multiple servers for efficiency.

Motivation

Mapping ensures that software components are deployed in a way that optimizes performance, reliability, and scalability. It helps manage resource usage efficiently, ensures real-time interactions, and allows for system scaling to handle more users.

Considerations

- **Performance:** Ensuring fast response times for real-time features.
- **Scalability:** Ability to add more servers as user load increases.
- **Security:** Protecting sensitive user data.

- **Cost Efficiency:** Balancing performance with infrastructure costs.

Example

- **Frontend Devices:** Web/mobile app on user smartphones.
- **Backend Servers:** Game logic on cloud (e.g., AWS EC2).
- **Database:** MySQL on cloud (e.g., AWS RDS).
- **Real-Time:** WebSocket server or Firebase for live interactions.
- **Load Balancers:** AWS Elastic Load Balancer to distribute traffic.

23f. Data Dictionary

Content

A data dictionary is a centralized repository of information about the data used in the system. It defines all data elements, including their types, structures, and relationships. In the "Two Truths and a Lie" game, the data dictionary includes details about users, game sessions, guesses, scores, and more.

Typical entries include:

- **Field Name:** The name of the data element (e.g., username, game_id).
- **Data Type:** The type of data (e.g., string, integer, boolean).
- **Description:** A brief description of what the field represents.
- **Constraints:** Any restrictions or validations (e.g., username must be unique).

Motivation

The data dictionary is crucial for maintaining consistency across the system and ensuring all team members have a clear understanding of the data being handled. It helps in:

- **Standardizing data usage:** Ensures consistent data definitions across different modules.
- **Facilitating communication:** Developers, database administrators, and analysts can refer to it for clarity on data structures.
- **Supporting maintenance:** Makes it easier to debug, update, or scale the system by providing clear documentation of data entities.

Considerations

- **Completeness:** The dictionary should cover all critical data fields used across the system.
- **Consistency:** Ensure that field names and data types are consistent across different subsystems.
- **Flexibility:** The dictionary should be flexible enough to accommodate future changes or additions.
- **Security:** Sensitive data (e.g., passwords) should have encryption details specified.

Example

Here's an example data dictionary for key entities in the game:

Field Name	Data Type	Description	Constraints
user_id	Integer	Unique identifier for each user	Primary Key, Auto-increment
username	String	Username chosen by the player	Unique, Max 30 characters
game_id	Integer	Unique identifier for each game	Primary Key, Auto-increment
statement_1	String	First truth submitted by the user	Max 200 characters
is_guess_correct	Boolean	Tracks if the player guessed correctly	Default: false
score	Integer	Player's score for the game	Non-negative

Table 1 – Data Dictionary

23g. Persistent Data Management

Content

Persistent data management involves storing and maintaining important data, such as user profiles, game history, and scores, in a database. This ensures that data is saved and accessible across sessions and system restarts. It encompasses:

- **Data Storage:** Using a database to store information.
- **Data Retrieval:** Accessing stored data when needed.
- **Data Updates:** Modifying data as required.
- **Data Backup:** Regularly backing up data to prevent loss.

Motivation

Effective persistent data management is essential for:

- **Continuity:** Ensures user progress and game states are preserved.
- **User Experience:** Allows players to resume games and review past interactions.
- **Scalability:** Supports growing amounts of data without performance issues.
- **Reliability:** Prevents data loss and maintains data integrity.

Considerations

- **Database Choice:** Select between SQL (e.g., MySQL) for structured data or NoSQL (e.g., MongoDB) for flexible data models.
- **Backup Strategy:** Implement regular backups to safeguard against data loss.
- **Data Integrity:** Ensure data is accurate and secure from unauthorized access or corruption.

Example

For the "Two Truths and a Lie" game, user profiles and game data might be stored in a MySQL database. This setup allows players to access their game history and scores across different sessions, ensuring a seamless experience.

23h. Access Control and Security

Content

Access control and security involve protecting the system and its data from unauthorized access and ensuring only authorized users can perform specific actions. This includes:

- **Authentication:** Verifying user identities (e.g., login credentials).
- **Authorization:** Granting access based on user roles and permissions.
- **Data Encryption:** Securing data during transmission and storage.
- **Audit Logging:** Tracking user activities for security monitoring.

Motivation

Proper access control and security are crucial for:

- **Data Protection:** Safeguarding sensitive information from unauthorized access.
- **Privacy:** Ensuring user data is kept confidential.
- **Integrity:** Preventing data tampering or unauthorized changes.
- **Compliance:** Meeting legal and regulatory requirements for data protection.

Considerations

- **Authentication Methods:** Use strong methods like multi-factor authentication (MFA).
- **Role-Based Access Control (RBAC):** Define roles and permissions clearly.
- **Encryption Standards:** Apply robust encryption for data at rest and in transit.
- **Regular Audits:** Conduct regular security audits to identify vulnerabilities.

Example

For the "Two Truths and a Lie" game, implementing OAuth for authentication and using HTTPS for data transmission ensures that user credentials and game data are secure. Role-based access control restricts admin functions to authorized personnel only, and audit logs track user activities to detect any unauthorized actions.

23i. Global Software Control

Content

Global software control involves managing and overseeing the software development and deployment process across different environments and stages. This includes:

- **Version Control:** Tracking changes in the codebase using systems like Git.
- **Configuration Management:** Managing software configurations and environments.

- **Continuous Integration/Continuous Deployment (CI/CD):** Automating testing and deployment processes.
- **Change Management:** Handling changes to the software and coordinating updates.

Motivation

Effective global software control is essential for:

- **Consistency:** Ensures that changes are systematically managed and integrated.
- **Quality Assurance:** Automates testing to catch issues early and maintain software quality.
- **Efficient Deployment:** Streamlines deployment processes and reduces downtime.
- **Collaboration:** Facilitates team coordination and version management.

Considerations

- **Version Control Systems:** Choose a reliable system (e.g., Git) for tracking code changes.
- **CI/CD Tools:** Implement tools (e.g., Jenkins, GitHub Actions) to automate builds and deployments.
- **Environment Management:** Ensure consistent configurations across development, testing, and production environments.
- **Change Control Processes:** Define procedures for managing and approving changes.

Example

In the "Two Truths and a Lie" game, using Git for version control helps track code changes, while CI/CD pipelines (e.g., GitHub Actions) automate testing

and deployment. Configuration management tools ensure consistency across development and production environments, and a change management process ensures that updates are thoroughly reviewed and tested before release.

23j. Boundary Conditions

Content

Boundary conditions define the limits and constraints of the software system, specifying how the system interacts with external environments and handles edge cases. For the "Two Truths and a Lie" game, boundary conditions include:

- **User Limits:** Maximum number of players in a game.
- **Data Limits:** Maximum length for truths and lies, file size for uploads.
- **Performance Limits:** Response times and system load capacities.
- **Operational Constraints:** Conditions under which the system operates (e.g., network availability).

Motivation

Defining boundary conditions is crucial for:

- **System Stability:** Ensures the system behaves correctly within its defined limits and handles edge cases gracefully.
- **Resource Management:** Helps in optimizing performance and managing system resources efficiently.
- **User Experience:** Provides clear guidelines on system limitations to avoid user errors and frustration.
- **Reliability:** Ensures the system operates reliably under various conditions.

Considerations

- **Edge Cases:** Identify and handle unusual or extreme scenarios that might affect system performance.
- **Scalability:** Ensure the system can handle varying loads and numbers of users without degradation.
- **Error Handling:** Implement robust error handling for situations that exceed boundary limits.
- **Testing:** Thoroughly test boundary conditions to verify system behavior under constraints.

Example

In the "Two Truths and a Lie" game:

- **User Limits:** Restrict each game to a maximum of 10 players.
- **Data Limits:** Allow each truth or lie to be up to 200 characters long.
- **Performance Limits:** Ensure the system can handle up to 1,000 simultaneous users without performance issues.
- **Operational Constraints:** The game should function properly with a minimum network speed of 500 kbps.

24. Subsystem Services

Content

Subsystem services are distinct functionalities provided by various subsystems within a software system. They define the specific operations and tasks each subsystem performs. For the "Two Truths and a Lie" game, subsystem services might include:

- **User Management:** Handles user registration, login, and profile management.
- **Game Management:** Manages game creation, player participation, and game state.
- **Scoring System:** Calculates and updates scores based on player guesses.
- **Real-Time Communication:** Facilitates live interactions between players, such as submitting guesses and updating scores.
- **Data Storage:** Manages the saving and retrieval of game data and user information.

Motivation

Defining subsystem services is important for:

- **Modularity:** Clearly separates different functionalities, making the system easier to manage and update.
- **Scalability:** Allows individual subsystems to be scaled independently based on demand.
- **Maintainability:** Simplifies debugging and enhances system maintenance by isolating specific functionalities.
- **Reuse:** Enables reuse of subsystems or services in different contexts or projects.

Considerations

- **Interdependencies:** Manage interactions between subsystems to ensure smooth operation.
- **Performance:** Optimize each service for efficiency and responsiveness.
- **Security:** Ensure that each subsystem is secure and handles data appropriately.
- **Scalability:** Design subsystems to handle increased load as the user base grows.

Example

For the "Two Truths and a Lie" game:

- **User Management Service:** Handles user login, registration, and profile updates.
- **Game Management Service:** Manages game creation, player additions, and game progress.
- **Scoring Service:** Calculates scores based on player guesses and updates them in real-time.
- **Real-Time Communication Service:** Uses WebSockets or Firebase to facilitate live interactions during gameplay.
- **Data Storage Service:** Utilizes a database (e.g., MySQL) to store user profiles, game history, and scores.

25. User Interface

Content

The user interface (UI) encompasses the design and layout of elements that users interact with. It includes visual components, navigation, and user interactions. For the "Two Truths and a Lie" game, the UI might include:

- **Home Screen:** Displays game options and user login/register buttons.
- **Game Screen:** Shows the current game, player inputs for truths and lies, and game status.
- **Profile Page:** Allows users to view and edit their profile information and view their game history.
- **Leaderboard:** Displays rankings and scores of players.

Motivation

A well-designed UI is crucial for:

- **User Experience:** Ensures that interactions are intuitive and enjoyable.
- **Accessibility:** Makes the game usable for people with different abilities.
- **Engagement:** Attracts and retains players through a visually appealing and functional design.
- **Efficiency:** Allows users to complete tasks and navigate the game easily.

Considerations

- **Usability:** Design for ease of use, ensuring that users can navigate and interact with the game effortlessly.

- **Aesthetics:** Create an appealing visual design that enhances user enjoyment.
- **Responsiveness:** Ensure the UI adapts well to different screen sizes and devices (e.g., mobile, tablet, desktop).
- **Feedback:** Provide clear feedback for user actions (e.g., button clicks, form submissions).

Example

For the "Two Truths and a Lie" game:

- **Home Screen:** Features large, clear buttons for "Start Game," "Login," and "Register," with a welcoming design.
- **Game Screen:** Displays player statements, input fields for guesses, and a timer with an easy-to-read layout.
- **Profile Page:** Includes editable fields for user details and a section showing past games and scores.
- **Leaderboard:** A clean table layout showing player names, scores, and rankings, with sorting options.

26. Object Design

26a. Object Design Tradeoffs

Content

Object design tradeoffs involve balancing various factors when creating object-oriented designs, focusing on how to best structure objects and their interactions. This includes choosing between different design options to meet system requirements while addressing constraints and goals.

Motivation

Considering tradeoffs is essential for:

- **Optimization:** Achieving the best balance between performance, maintainability, and scalability.
- **Flexibility:** Ensuring the design can adapt to changing requirements and future enhancements.
- **Resource Management:** Efficiently using system resources (e.g., memory, processing power).

Considerations

- **Complexity vs. Simplicity:** More complex designs can be more flexible but harder to maintain. Simpler designs may be easier to manage but might lack flexibility.
- **Performance vs. Maintainability:** Optimizing for performance can make the design more complex and harder to maintain, while a focus on maintainability might impact performance.
- **Coupling vs. Cohesion:** High coupling (dependencies between objects) can simplify some interactions but make the system less modular. High cohesion (related functionality within objects) enhances modularity but might complicate interactions.

Example

For the "Two Truths and a Lie" game API:

- **API Endpoint:** /submit_guess
    - **Description:** Submits a player's guess in a game.
    - **Usage Example:** A developer wants to allow users to submit their guesses during gameplay. They use the /submit_guess endpoint to send the guess data.
    - **Error Handling:** If a developer encounters a "Game not found" error, they need to check if the game_id is correct.

26b Interface Documentation Guidelines

Content

The "Two Truths and a Lie Game" is a classic party game where players take turns stating three statements about themselves: two truths and one lie. The goal is for other players to guess which statement is the lie. The application brings this engaging game to mobile platforms, allowing users to play with friends or random players online. The interface provides a user-friendly design for inputting statements, guessing the lie, and viewing the results.

Motivation

The motivation behind creating the "Two Truths and a Lie Game" application is to provide a fun and interactive way for people to connect. The game encourages players to share interesting facts about themselves, leading to better social interactions and fostering a sense of camaraderie. By digitizing this game, the app aims to make it accessible anytime, anywhere, while adding features like scoring, player matching, and customized game modes to enhance the gaming experience.

Considerations

Several key considerations were taken into account while designing the user interface for the application:

- **User Friendliness**: The interface should be intuitive and easy to navigate, allowing players of all ages to enjoy the game without a steep learning curve.
- **Mobile Compatibility**: The app needs to work seamlessly on both Android and iOS devices, with responsive design to accommodate various screen sizes.
- **Real-Time Interaction**: Given that the game involves player interactions, the interface must support real-time updates and responses to keep the gameplay engaging.
- **Customization**: Allowing users to customize their avatars, create private game rooms, or modify game settings adds to the personalization of the experience.
- **Data Privacy**: Since the game involves user-generated content, privacy measures must be implemented to protect user data, including age-appropriate content filtering.

Example

Consider a scenario where a user wants to start a game. The interface will follow these steps:

- **Home Screen**: The user is presented with options to "Start a New Game," "Join a Game," or "View Leaderboard."
- **New Game Setup**: When selecting "Start a New Game," the interface guides the user to create a game room. They can invite friends or allow random players to join.

- **Game Play Screen**: Once the game starts, players take turns entering three statements. The screen will display the statements with interactive options for other players to vote on which one is the lie.
- **Results Display**: After all players have voted, the results are shown, including who guessed correctly, along with a score tally.
- **End of Round Summary**: Players can see a summary of the round, with options to start another round or end the game.

26c. Packages

Content

Packages are organizational units in software design that group related classes and interfaces. They help in structuring and managing code by encapsulating functionalities into logical units. In the "Two Truths and a Lie" game, packages might include:

- **User Management:** Handles user registration, login, and profile management.
- **Game Logic:** Manages game rules, state, and interactions.
- **Scoring System:** Calculates and tracks player scores.
- **Communication:** Facilitates real-time interactions between players.

Motivation

Using packages is important for:

- **Modularity:** Enhances code organization by grouping related classes and functions.
- **Maintainability:** Simplifies updating and managing code by isolating different functionalities.

- **Reusability:** Encourages reuse of code across different parts of the system or in other projects.
- **Scalability:** Makes it easier to expand and scale the system by adding or modifying packages without affecting other parts.

Considerations

- **Package Boundaries:** Clearly define the scope and responsibilities of each package.
- **Dependencies:** Manage dependencies between packages to avoid tight coupling.
- **Naming Conventions:** Use meaningful and consistent names for packages to improve readability and understanding.
- **Encapsulation:** Ensure that packages expose only necessary functionality and hide internal details.

Example

For the "Two Truths and a Lie" game:

- **com.game.usermanagement**: Contains classes like User, UserProfile, and AuthenticationService.
- **com.game.gamelogic**: Includes classes like Game, GameState, and GameRules.
- **com.game.scoring**: Features classes such as ScoreManager, ScoreCalculator, and Scoreboard.
- **com.game.communication**: Manages real-time communication with classes like ChatService and NotificationManager.

26d. Class Interfaces

Content

Class interfaces specify the methods and properties that a class exposes, defining how other parts of the system can interact with it. This includes:

- **Method Names:** What operations the class can perform.
- **Parameters:** The inputs required for these operations.
- **Return Types:** The outputs or results of the operations.
- **Exceptions:** Any errors that the methods might throw.

Motivation

Class interfaces are important for:

- **Encapsulation:** Hides internal details while exposing necessary functionalities.
- **Consistency:** Provides a clear contract for how classes should be used.
- **Flexibility:** Allows implementation changes without affecting other parts of the system.
- **Testing:** Simplifies unit testing by defining expected interactions.

Considerations

- **Method Signatures:** Clearly define what methods are available, their parameters, and their return values.
- **Exception Handling:** Document any potential exceptions a method might throw.
- **Contract Clarity:** Ensure that the interface specifies unambiguous expectations.

- **Documentation:** Include comprehensive descriptions of methods and their purposes.

Example

For the "Two Truths and a Lie" game, the class interfaces might include:

- **User Interface:**
  - **Description:** Manages user-related operations.
  - **Methods:**
    - **Register:** Allows new users to create an account.
    - **Login:** Authenticates existing users.
    - **GetProfile:** Retrieves user profile details.
- **Game Interface:**
  - **Description:** Handles game operations.
  - **Methods:**
    - **CreateGame:** Initializes a new game session.
    - **SubmitGuess:** Allows players to submit their guesses.
    - **GetGameState:** Provides the current state of the game.
- **Scoring Interface:**
  - **Description:** Manages scoring details.
  - **Methods:**
    - **UpdateScore:** Updates a player's score based on their performance.
    - **GetScore:** Retrieves the current score for a player.

- **Communication Interface:**
    - **Description:** Facilitates real-time communication between players.
    - **Methods:**
        - **SendMessage:** Sends messages between players.
        - **ReceiveMessages:** Retrieves messages from the game.

IV Test Plans

27. Features to be Tested / Not to be Tested

Content

This section outlines the specific features that will be included in the testing process, as well as those that are excluded. It helps clarify the scope of testing to ensure focus on critical components while explaining why certain features are not tested. Features to be tested include core functionality and critical integrations, while those not to be tested may involve external dependencies or non-essential features.

Motivation

Clearly defining what is tested helps:

- **Prioritize Testing:** Focuses on high-risk and high-importance features.
- **Resource Allocation:** Ensures testing resources are used efficiently by avoiding unnecessary tests on minor features.
- **Transparency:** Provides a clear understanding of the test coverage, reducing misunderstandings during development.

Considerations

- **Feature Importance:** Prioritize testing based on the criticality of the feature to the system.
- **Risk Assessment:** Test high-risk areas where failure could significantly impact the system.
- **External Dependencies:** Exclude features outside the team's control, such as third-party integrations, unless they are mission-critical.
- **Cost vs. Benefit:** Balance testing costs with the benefits of testing minor or less-used features.

Example

For the "Two Truths and a Lie" game:

- **To be Tested:**
    - **User Authentication:** Ensure login and registration work correctly.
    - **Game Logic:** Validate that players can submit truths and lies, and guesses are processed correctly.
    - **Scoring System:** Verify that points are awarded and deducted based on correct or incorrect guesses.
    - **Real-Time Updates:** Ensure that updates (like score changes) appear instantly for all players in a game.
- **Not to be Tested:**
    - **Third-Party Payment Gateway:** Testing of the payment provider's functionality will be excluded as it is out of the team's control.
    - **Browser-Specific Issues:** Only core browsers (Chrome, Firefox) will be tested, and not every possible browser version.

28. Pass/Fail Criteria

Content

Pass/Fail criteria define the conditions that determine whether a test case has succeeded or failed. Each feature or functionality being tested must meet specific expectations, and these criteria ensure consistency in evaluating test results. Common metrics include:

- **Pass:** The system behaves as expected, with no critical issues.
- **Fail:** The system does not meet the expected behavior or results in errors.

Motivation

Clear Pass/Fail criteria are essential because:

- **Consistency:** Ensures uniform assessment of test results across different features and developers.
- **Objectivity:** Provides a straightforward and unbiased way to determine whether a test meets the required standards.
- **Efficiency:** Speeds up the testing process by providing clear guidelines on when tests succeed or fail.
- **Quality Assurance:** Ensures that only features that meet performance and functional requirements are considered "passed."

Considerations

- **Feature Requirements:** Ensure the criteria are based on the predefined requirements for each feature.
- **Error Tolerance:** Determine if any level of non-critical errors or warnings can still result in a pass.
- **Performance Expectations:** Consider response times, load handling, and overall performance.

- **User Impact:** A failure in user-facing components (e.g., login, gameplay) is critical, whereas backend failures might be acceptable depending on the context.
- **Automation:** For automated testing, clearly define thresholds for acceptable automated test results (e.g., test execution time, accuracy).

Example

For the "Two Truths and a Lie" game:

- **Pass Criteria:**
    - **User Login:** The user can log in successfully without errors, and all expected profile details are displayed.
    - **Game Creation:** A new game is successfully created, all players can join, and the game proceeds without crashes or errors.
    - **Truth/Lie Submission:** Players can submit their truths and lies, and the system records them correctly.
    - **Score Updates:** Scores are updated in real-time after each round without delays or inaccuracies.
- **Fail Criteria:**
    - **User Login Failure:** If the user cannot log in, or the system displays incorrect information after login.
    - **Game Creation Failure:** If the game crashes or users are unable to join, the test is considered a fail.
    - **Data Inconsistency:** If player inputs are not correctly saved or retrieved, leading to incorrect game outcomes.
    - **Delayed Score Updates:** If scores are not updated in real-time, it results in a test failure.

29. Approach

Content

The **approach** section outlines the testing strategy used to evaluate the system's performance and functionality. It describes the methodologies, tools, and processes that will be employed during testing. This can include:

- **Testing Types:** Unit testing, integration testing, system testing, and user acceptance testing (UAT).
- **Testing Techniques:** Manual or automated testing, black-box or white-box testing.
- **Tools:** The specific tools (e.g., Selenium, JUnit) that will be used for automated tests.
- **Execution Plan:** How and when tests will be executed, including the order of test cases and test cycles.

Motivation

A well-defined testing approach is important to:

- **Organize Testing Efforts:** Ensures that testing is structured, repeatable, and covers all necessary components.
- **Maximize Coverage:** Ensures that all aspects of the system are tested thoroughly, from individual units to the integrated system.
- **Align Resources:** Ensures that resources (time, personnel, tools) are used efficiently.
- **Mitigate Risk:** Identifies potential risks and ensures that testing is focused on areas with higher risk.

Considerations

- **Scope of Testing:** Decide the breadth and depth of the testing process, from unit tests to full integration.

- **Risk-Based Testing:** Focus more on high-risk areas, like user authentication or gameplay, to minimize system failure risks.
- **Automation vs. Manual Testing:** Balance the need for automated tests (for repetitive tasks) and manual tests (for more complex user interactions).
- **Continuous Integration:** Integrate testing into the development pipeline to catch issues early and ensure regular checks.
- **Test Environment:** Ensure the test environment mimics production closely in terms of hardware, software, and network settings.

Example

For the "Two Truths and a Lie" game:

- **Testing Types:**
    - **Unit Testing:** Test individual components like the scoring system, game logic, and data validation.
    - **Integration Testing:** Check how the game engine interacts with the user database and real-time update systems.
    - **System Testing:** Validate the entire game flow from login to submitting guesses, scoring, and declaring a winner.
    - **User Acceptance Testing (UAT):** Involve real users to ensure the game is intuitive and functions as expected in real-world conditions.
- **Execution Plan:**
    - **Phase 1 (Unit Testing):** Conduct unit tests for individual modules (e.g., game logic, user login).
    - **Phase 2 (Integration Testing):** Test how modules work together, such as user inputs affecting game mechanics.

- **Phase 3 (System Testing):** Execute end-to-end tests in a near-production environment.
- **Phase 4 (UAT):** Allow players to try the game and provide feedback before final deployment.

30. Suspension and Resumption

Content

This section outlines the conditions under which testing may be **suspended** (paused) and the criteria for **resumption** (continuing). Suspension happens when critical issues arise, while resumption occurs after those issues are resolved or conditions for testing readiness are restored.

Motivation

Defining suspension and resumption criteria helps:

- **Minimize Wasted Effort:** Pauses testing when critical issues prevent further progress.
- **Ensure Readiness:** Ensures testing resumes only when the system is stable and issues are resolved.
- **Resource Management:** Avoids spending time on testing a broken or incomplete system.

Considerations

- **Critical Issues:** Suspend testing if critical bugs or system crashes occur.
- **Dependencies:** Pause if external systems or resources (e.g., third-party APIs) are unavailable.
- **Resumption:** Testing resumes once fixes or updates are confirmed, or external dependencies are restored.

Example

For the "Two Truths and a Lie" game:

- **Suspend:** If a major bug causes incorrect score calculations, testing is paused.
- **Resume:** After the bug is fixed and validated, testing resumes from the point of suspension.

31. Testing Materials (Hardware/Software Requirements)

Content

This section specifies the hardware and software required for testing. It includes the necessary devices, operating systems, browsers, testing tools, and any specific environments or configurations.

Motivation

Defining hardware and software requirements ensures:

- **Consistency:** Testing occurs in environments similar to the production setup.
- **Comprehensive Coverage:** Ensures compatibility across different platforms.
- **Efficiency:** Prepares teams with the correct tools and environments to avoid delays.

Considerations

- **Hardware:** List the devices (PCs, mobile phones, tablets) for testing.
- **Software:** Include required operating systems, browsers, testing tools, and versions.

- **Environment:** Ensure the setup mirrors the production environment (e.g., network conditions, data configurations).

Example

For the "Two Truths and a Lie" game:

- **Hardware:** PCs with Windows and macOS, mobile devices (Android, iOS).
- **Software:** Chrome, Firefox, Safari, Selenium for automated tests.
- **Environment:** Test servers configured to mirror the production environment.

32. Test Cases

Test Case 1

- **Test Case ID: TC001**
- **Description: Test creation of a new multiplayer game session.**
- **Preconditions: User is logged in to the application.**
- **Test Steps:**
    1. **Navigate to the "Multiplayer Mode" in the application.**
    2. **Click on "Start New Game."**
- **Expected Results: A new multiplayer session should be created, and other players should receive an invitation.**
- **Actual Result: A new multiplayer session was created successfully, and other players received the invitation notification.**
- **Status: Pass**

Test Case 2

- **Test Case ID: TC002**
- **Description: Test player joining functionality for a new session.**
- **Preconditions: A multiplayer game session is already created.**
- **Test Steps:**
    1. **Player receives an invitation to join the session.**
    2. **Clicks on the invitation link.**
- **Expected Results: Player should successfully join the session, and their name should appear in the lobby.**
- **Actual Result: Player joined the session successfully, and their name appeared in the lobby as expected.**
- **Status: Pass**

Test Case 3

- **Test Case ID: TC003**
- **Description: Test the game initiation after all players join.**
- **Preconditions: All invited players have joined the session.**
- **Test Steps:**
    1. **Host clicks "Start Game."**
- **Expected Results: Game should initiate with the first player prompted to enter two truths and one lie.**
- **Actual Result: Game initiated successfully, and the first player was prompted to enter their statements.**
- **Status: Pass**

Test Case 4

- **Test Case ID: TC004**
- **Description: Test submission of statements in multiplayer mode.**
- **Preconditions: Game is in progress, and it is Player 1's turn.**
- **Test Steps:**
    1. **Player 1 enters two truths and one lie.**
    2. **Clicks "Submit."**
- **Expected Results: Statements are submitted successfully, and other players are prompted to guess the lie.**
- **Actual Result: Player 1's statements were submitted successfully, and other players were prompted to guess the lie.**
- **Status: Pass**

Test Case 5

- **Test Case ID: TC005**
- **Description: Test guessing functionality by other players.**
- **Preconditions: Statements are visible to all players, and it's time for players to guess.**
- **Test Steps:**
    1. **Each player selects a statement they believe is a lie.**
    2. **Clicks "Submit Guess."**
- **Expected Results: Each player's guess is submitted, and the system displays if they were correct or incorrect.**

- **Actual Result: Each player's guess was submitted successfully, and the system displayed whether each guess was correct or incorrect.**
- **Status: Pass**

Test Case 6

- **Test Case ID: TC006**
- **Description: Test scoring calculation for correct and incorrect guesses.**
- **Preconditions: Players have submitted their guesses.**
- **Test Steps:**
    1. **System calculates points based on the guesses.**
- **Expected Results: Correct guesses should add points to the players' scores, and updated scores should display on the screen.**
- **Actual Result: Points were calculated accurately based on the guesses, and the updated scores were displayed on the screen.**
- **Status: Pass**

33. Testing Schedule

Content

The testing schedule outlines the timeline for all testing activities, from unit testing to system and user acceptance testing (UAT). It includes key milestones, deadlines, and testing phases to ensure the project stays on track.

Motivation

A clear testing schedule:

- **Organizes Workflows:** Helps teams know when and what to test.
- **Tracks Progress:** Ensures testing is aligned with development timelines.
- **Allocates Resources:** Ensures adequate time and resources for each phase of testing.

Considerations

- **Dependencies:** Coordinate testing phases with development milestones.
- **Buffer Time:** Include time for fixing issues and re-testing.
- **Parallel Testing:** Plan for concurrent tests (e.g., automated and manual testing) to optimize time.

Example

For the "Two Truths and a Lie" game:

- **Week 1:** Unit testing for game logic and scoring.
- **Week 2:** Integration testing for login, gameplay, and real-time updates.
- **Week 3:** System testing and bug fixes.
- **Week 4:** UAT with real users.

V Project Issues

34. Open Issues

Content

This section lists any unresolved problems or challenges that have emerged during the project but have not yet been addressed. These issues could be related to technical bugs, incomplete features, or other project-related obstacles.

Motivation

Documenting open issues is important because:

- **Tracking Progress:** Helps track known problems and their resolution status.
- **Prioritization:** Ensures critical issues are prioritized for resolution.
- **Transparency:** Provides visibility into unresolved problems, allowing stakeholders to understand the current state of the project.

Considerations

- **Severity:** Focus on critical issues that block key functionality.
- **Dependencies:** Note any external dependencies affecting issue resolution.
- **Timeline:** Estimate how long it will take to resolve the issue and its potential impact on project deadlines.

Example

For the "Two Truths and a Lie" game:

- **Open Issue:** Incorrect score calculation when multiple players guess simultaneously.
    - **Impact:** Affects gameplay experience, requires debugging and testing.

- **Resolution Status:** In progress, expected to be fixed by next release.

35 Off-the-Shelf Solutions

35a Ready-Made Products

Content

Ready-made products refer to existing software, tools, or services that can be directly utilized in the development of the "Two Truths and a Lie Game" application. These products typically come with pre-built functionalities that can help accelerate the project. Examples of such products include pre-made game engines, user interface (UI) design libraries, and backend services for user authentication or data storage.

Motivation

Using ready-made products can significantly reduce the time and effort needed for development by providing pre-tested and well-documented functionalities. This approach allows the development team to focus on customizing the application and improving the user experience instead of building everything from scratch.

Considerations

- Licensing Costs: Some ready-made products may require a subscription or licensing fee, which needs to be considered when budgeting.
- Customization Limits: The extent to which the products can be customized might be limited. It’s important to evaluate whether these limitations will affect the overall application.
- Integration Effort: Although these products are pre-made, integrating them into the system may still require effort to ensure compatibility.

35b Reusable Components

Content

Reusable components are modular pieces of software that can be used across different parts of the application or even in future projects. In the context of the "Two Truths and a Lie Game" application, these could include libraries for managing user profiles, game logic modules for scoring and tracking user progress, or UI elements like buttons and animations.

Motivation

Incorporating reusable components promotes code reusability and maintainability. It also helps standardize certain features across the application, making development more efficient and consistent.

Considerations

- Compatibility: Ensuring that the components are compatible with the development stack being used is crucial.
- Modularity: The components should be flexible enough to allow for modifications to suit specific needs.
- Versioning: Keeping track of component versions and updates is important to avoid breaking changes.

35c Products That Can Be Copied

Content

Some products or existing open-source projects can serve as inspiration or even be adapted for the development of the "Two Truths and a Lie Game" application. For example, existing trivia or quiz games with similar functionalities could provide code or design patterns that can be modified to fit this project's requirements.

Motivation

Using products that can be copied or adapted can offer a head start by providing a foundation that can be customized. This approach reduces development time and leverages existing best practices.

Examples

- Open-Source Trivia Game Engines: There are several open-source game engines specifically designed for trivia or quiz games. These engines can provide the underlying game logic for scoring, timing, and question management.
- Template UI Designs: Pre-made user interface templates can be adapted to match the branding and design preferences of the "Two Truths and a Lie Game" application.

Considerations

- Legal Implications: Ensure that any copied or adapted code complies with copyright and licensing agreements.
- Technical Debt: Relying on external code may introduce dependencies that need to be maintained or updated.
- Quality Assurance: Even if the original product is well-tested, modifications might introduce bugs that need thorough testing.

36 New Problems

36a Effects on the Current Environment

Content

The integration of a new mobile game application into an existing digital ecosystem may cause changes or disruptions to the current environment. For example, the application could increase server load or require additional resources for maintenance and updates.

Motivation

Understanding the potential effects on the current environment helps prepare for any resource scaling or technical adjustments that may be required to accommodate the new application.

Examples

- **Increased Network Traffic:** The application's usage may lead to higher data demands, especially during peak times.
- **Server Resource Allocation:** More backend resources may be needed to handle user authentication, data storage, and real-time game activities.
- **Security Considerations:** Any application that involves user data needs to adhere to data protection regulations, potentially requiring updates to current security measures.

Considerations

- **Scalability:** The system should be able to handle increased usage without significant degradation in performance.
- **Data Privacy:** The new system must comply with relevant privacy laws, such as GDPR, to protect user data.
- **System Downtime:** There should be a plan to manage any downtime required for system upgrades or maintenance.

36b Effects on the Installed Systems

Content

Integrating the "Two Truths and a Lie Game" application may impact existing systems already in use. These effects could include compatibility issues, additional system requirements, or performance bottlenecks.

Motivation

It is important to analyse how the new application might interact with existing systems to avoid conflicts and ensure smooth operation.

Considerations

- **Compatibility Checks:** The new application should be compatible with existing platforms and services used by the target audience.
- **Integration Testing:** Testing should be conducted to identify any potential issues with the installed systems.

36c Potential User Problems

Content

Users may encounter difficulties when adopting the "Two Truths and a Lie Game" application. These problems could arise from usability issues, technical glitches, or accessibility concerns.

Motivation

Addressing potential user problems early in the development process can improve the user experience and increase adoption rates.

Considerations

- **Usability Testing:** Conducting user testing can help identify areas of the application that may confuse or frustrate users.
- **User Feedback:** Implementing a mechanism for collecting user feedback will be crucial for identifying and addressing issues post-launch.

36d Limitations in the Anticipated Implementation Environment That May Inhibit the New Product

Content

Certain limitations in the anticipated implementation environment could hinder the performance or adoption of the "Two Truths and a Lie Game" application. This could include hardware limitations, network connectivity issues, or platform-specific restrictions.

Motivation

Understanding these limitations allows for pre-emptive planning and optimization to ensure the application performs well across different environments.

Examples

- **Hardware Limitations:** Some older mobile devices may not support advanced graphics or intensive processing.
- **Network Connectivity:** In areas with poor internet connections, the app may not function smoothly, particularly if it requires real-time updates.
- **Platform Restrictions:** Differences in iOS and Android operating systems may necessitate platform-specific modifications.

Considerations

- **Optimization for Low-End Devices:** Ensuring the app runs on a wide range of devices will broaden its potential user base.
- **Offline Capabilities:** Adding offline features can make the app more usable in areas with limited connectivity.
- **Cross-Platform Testing:** Testing on both Android and iOS platforms is essential to identify and resolve any platform-specific issues.

36e Follow-Up Problems

Content

Follow-up problems may arise after the application is launched, such as software bugs, security vulnerabilities, or new feature requests from users.

Motivation

Preparing for follow-up problems ensures that resources are in place for ongoing maintenance, updates, and user support.

Considerations

- **Support Infrastructure:** There should be a plan for providing user support and addressing issues post-launch.
- **Regular Updates:** Implementing a schedule for regular updates and security patches is crucial to maintain the app's performance and security.
- **User Expectations:** Managing user expectations regarding new features and bug fixes can help maintain a positive relationship with the user community.

37 Tasks

37a Project Planning

Content

Project planning involves outlining the project's scope, defining objectives, and creating a timeline for development. It includes identifying key milestones, assigning responsibilities, and establishing a budget for the project.

Motivation

Effective project planning provides a clear roadmap for the development process, ensuring that the project stays on track and meets deadlines. It also helps in resource allocation and risk management.

Considerations

- **Risk Assessment:** Identifying potential risks early in the planning phase allows for mitigation strategies to be put in place.
- **Resource Allocation:** Ensuring that the necessary resources (developers, tools, budget) are available at each stage of the project.
- **Stakeholder Involvement:** Regular updates and communication with stakeholders ensure that the project meets user and business expectations.

37b Planning of the Development Phases

Content

Planning the development phases involves breaking down the project into smaller, manageable stages, such as requirements analysis, design, development, testing, and deployment. Each phase has specific goals and deliverables.

Motivation

Dividing the project into phases allows for better tracking of progress and ensures that each stage is completed before moving on to the next. This structured approach helps manage complexity and improves the overall quality of the application.

Fit Criterion

- **Phase Completion:** Each phase should have clear criteria for completion, such as passing all unit tests during the testing phase.
- **Milestone Achievement:** Progress should be measured by achieving predefined milestones within each phase.

Considerations

- **Iteration Needs:** Some phases may require iterative development, such as repeated testing and refinement.
- **Phase Overlap:** In some cases, phases may overlap, requiring coordination between teams working on different aspects of the project.
- **Dependency Management:** Certain tasks may be dependent on the completion of others, so dependencies should be managed carefully.

38 Migration to the New Product

38a Requirements for Migration to the New Product

Content

The migration process requires specific steps to be taken, such as setting up the new system environment, transferring data, and ensuring compatibility with existing infrastructure. It may also involve training users to adapt to the new system.

Motivation

Proper planning and execution of migration are essential to minimize downtime, data loss, and user disruption. It ensures a smooth transition and helps achieve the full benefits of the new product.

Considerations

- **System Compatibility:** Ensuring that the new application integrates well with existing platforms and services.
- **Data Integrity:** Maintaining data accuracy and completeness during the migration process.
- **User Training:** Providing adequate training to users to ensure they are comfortable with the new application.

38b Data That Has to Be Modified or Translated for the New System

Content

Data migration may involve converting existing data into a format that is compatible with the new application. This could include modifying user data, game records, or configuration files to align with the new system's data structures.

Motivation

Ensuring that data is properly formatted and validated before migration helps prevent data-related issues and ensures that the new system can function effectively.

Fit Criterion

- **Data Accuracy:** All migrated data should be verified for accuracy and completeness.
- **Compatibility Checks:** Data should be tested in the new environment to ensure it functions as expected.

Considerations

- **Data Mapping:** Mapping existing data fields to the new data structures to avoid inconsistencies.
- **Testing Migration:** Running tests to validate that the data migration has been successful and that all necessary data is accessible in the new system.
- **Backup Plans:** Creating backups of existing data before migration to avoid data loss.

39 Risks

Risks refer to potential problems or challenges that could arise during the development or operation of the "Two Truths and a Lie Game" application. They can affect the project's scope, schedule, budget, quality, or user experience. Identifying risks early in the project helps in developing strategies to mitigate or avoid them.

Common risks include:

- **Technical Risks:** Issues related to software compatibility, bugs, or technical debt.
- **Project Management Risks:** Delays in timelines, resource unavailability, or miscommunication among team members.
- **Security Risks:** Vulnerabilities in the application that could lead to data breaches or unauthorized access.
- **User Adoption Risks:** Challenges in getting users to download and use the application, or problems with the user interface that might hinder user engagement.

Risk Management Strategies:

- Regular testing and code reviews to identify bugs early.
- Creating a detailed project timeline with buffer periods for unexpected delays.
- Implementing data encryption and secure authentication methods.
- Conducting user acceptance testing (UAT) to gather feedback and improve usability.

40 Costs

Costs refer to the financial resources required to develop, deploy, and maintain the "Two Truths and a Lie Game" application. These costs can be categorized into various types:

- **Development Costs:** Expenses for software development, including salaries for developers, designers, and project managers.
- **Operational Costs:** Ongoing expenses for hosting, data storage, and maintenance.
- **Marketing Costs:** Funds needed for promoting the app to increase visibility and user adoption.
- **Support Costs:** Expenses related to user support, updates, and bug fixes.

41 Waiting Room

Content

The "Waiting Room" section contains features, enhancements, or ideas that are not included in the current version of the "Two Truths and a Lie Game" application but may be considered for future updates. These items are kept in a backlog and can be revisited once the initial project goals are met.

Motivation

Having a "Waiting Room" for future ideas ensures that potentially valuable features are not forgotten and can be prioritized based on user feedback or changing project requirements. This approach allows for continuous improvement without overwhelming the development team with too many tasks at once.

Considerations

- **Prioritization:** Features in the Waiting Room should be periodically reviewed and prioritized based on user demand and project resources.
- **Resource Allocation:** Ensure there are enough resources to implement new features without compromising existing functionalities.
- **User Feedback Integration:** Use feedback from users to guide which features should be moved from the Waiting Room to active development.

42 Ideas for Solutions

Content

This section discusses potential solutions to address identified challenges in the "Two Truths and a Lie Game" application. Solutions may involve technical implementations, design changes, or new features to improve user experience, performance, or security.

Motivation

Exploring various solutions encourages innovative thinking and ensures that different approaches are considered when addressing challenges. This can lead to more effective and efficient problem-solving.

Considerations

- **Feasibility:** Assess the technical and financial feasibility of each proposed solution.
- **Impact:** Evaluate the potential impact on users and the system before implementing changes.
- **Scalability:** Solutions should be scalable to accommodate future growth and additional features.

43 Project Retrospective

Content

The project retrospective provides an evaluation of the "Two Truths and a Lie Game" application development process, focusing on what went well, what could be improved, and lessons learned. It helps in understanding the strengths and weaknesses of the project execution.

Motivation

Conducting a retrospective allows the team to reflect on their performance, identify areas for improvement, and apply those insights to future projects. It also helps in documenting best practices and avoiding mistakes in future development efforts.

Considerations

- **Team Feedback:** Collect feedback from all team members to get a well-rounded view of the project's success and challenges.
- **Key Performance Indicators (KPIs):** Measure the project's success against the initial goals and timelines.
- **Actionable Outcomes:** The retrospective should result in concrete action items for improving processes in future projects.

VI Glossary

Two Truths and a Lie is a fun guessing game where players reveal two true statements and one sneaky lie about themselves. The challenge is to spot the lie—testing intuition, creativity, and knowledge of friends or new acquaintances. With each round, players can gain insights, score points, and enjoy unexpected surprises. Perfect for breaking the ice, sparking laughs, and uncovering hidden truths, Two Truths and a Lie brings people together in a light-hearted game of wits and revelations.

VII References / Bibliography

- https://youtube.com/
- https://www.geeksforgeeks.org/
- https://www.javatpoint.com/
- https://play.google.com/store/apps/details?id=com.vanilla.truthlie&pcampaignid=web_share

VIII Plagiarism Screenshots

1. Module 1 to 3c

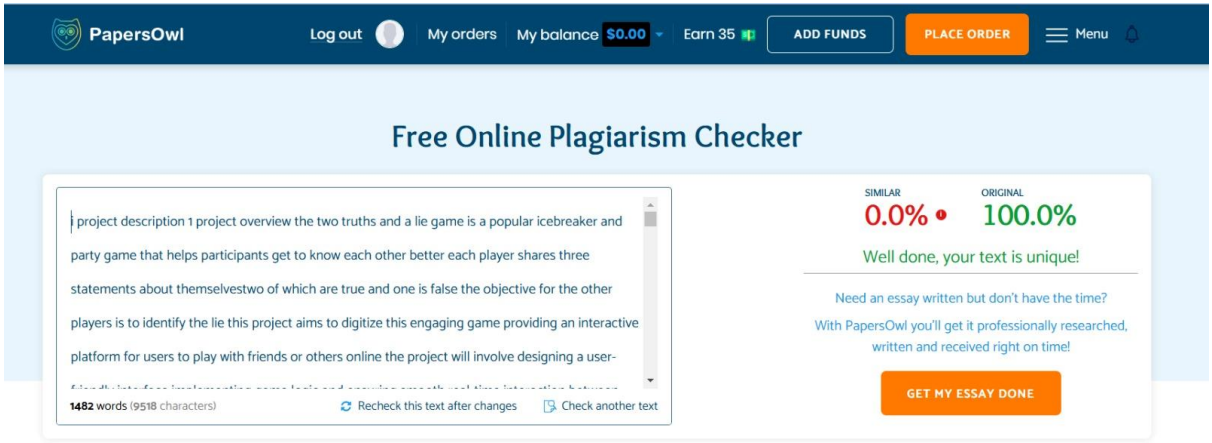

Figure 8 – 1 to 3c

2. Module 3d to 5d

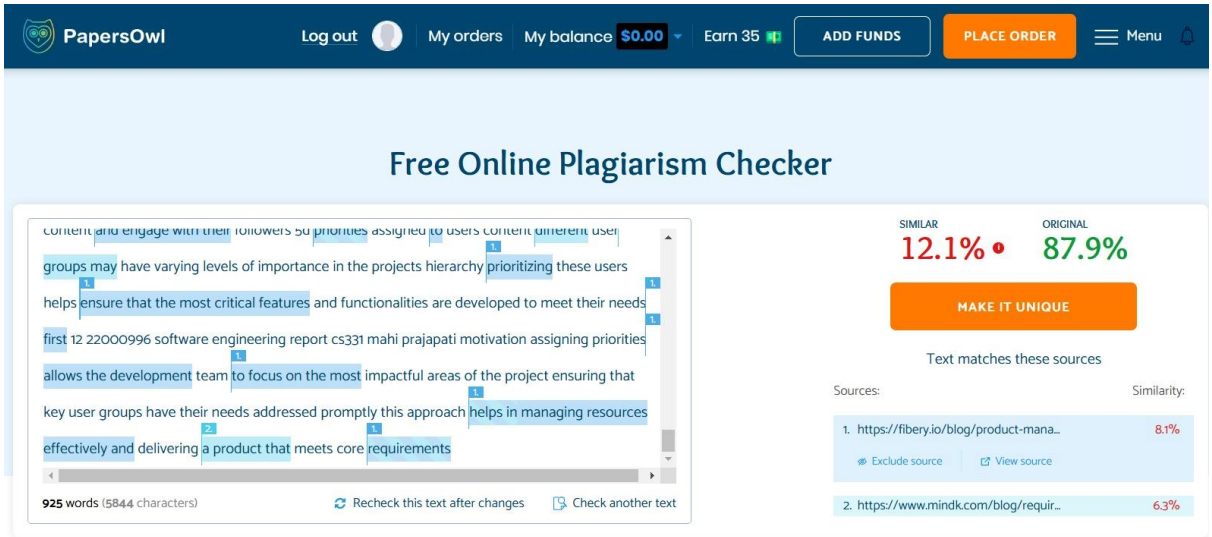

Figure 9 – 3d to 5c

3. Module 5e to 6g

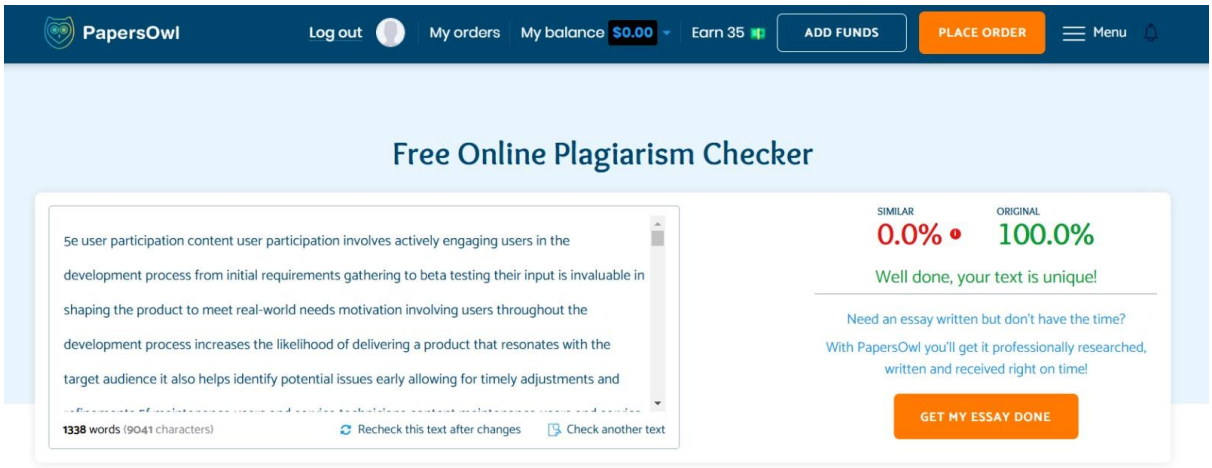

Figure 10 – 5e to 6g

4. Module 7 to 8b

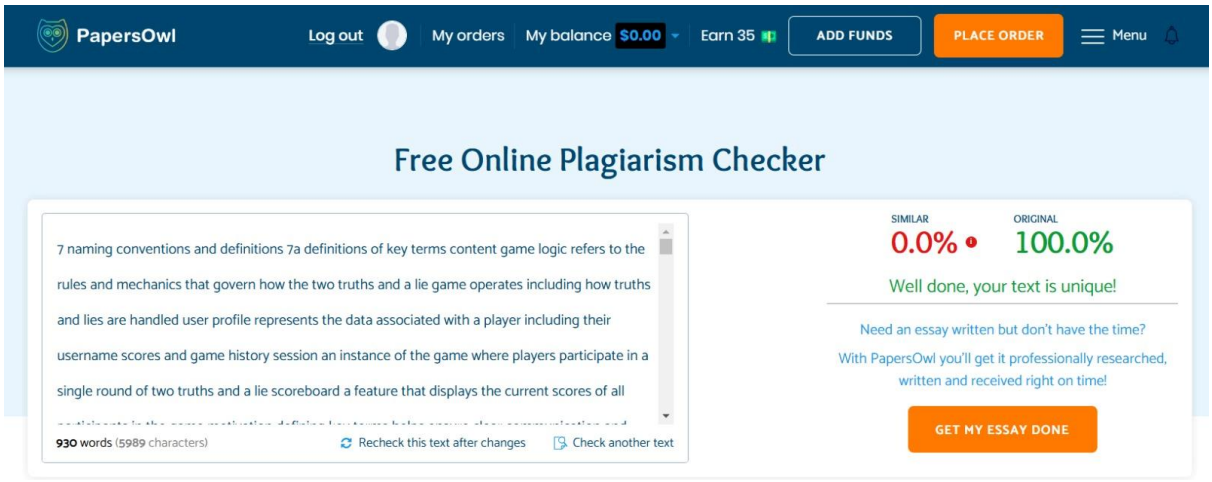

Figure 11 – 7 to 8b

5. **Module 9 to 11**

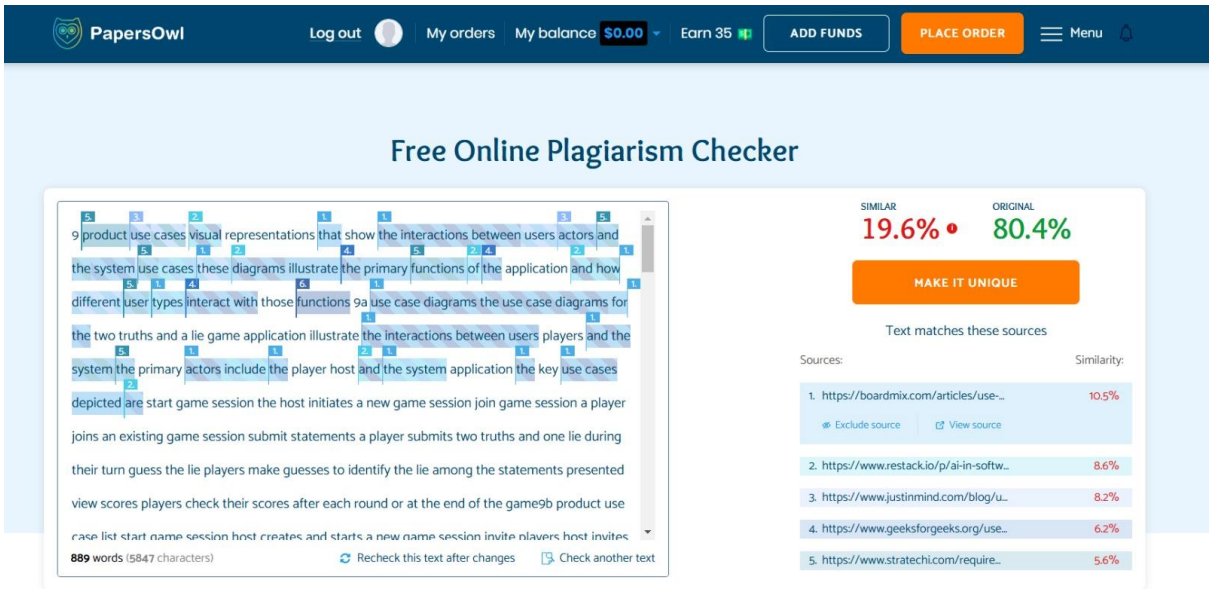

Figure 12 – 9 to 11

6. **Module 12 and 13**

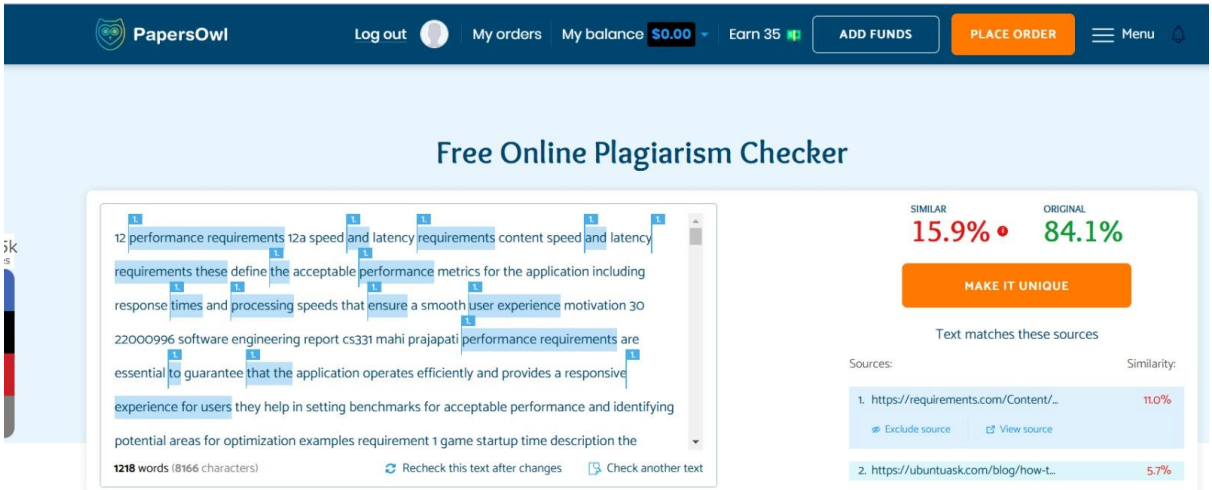

Figure 13 – 12 and 13

7. Module 14 and 15

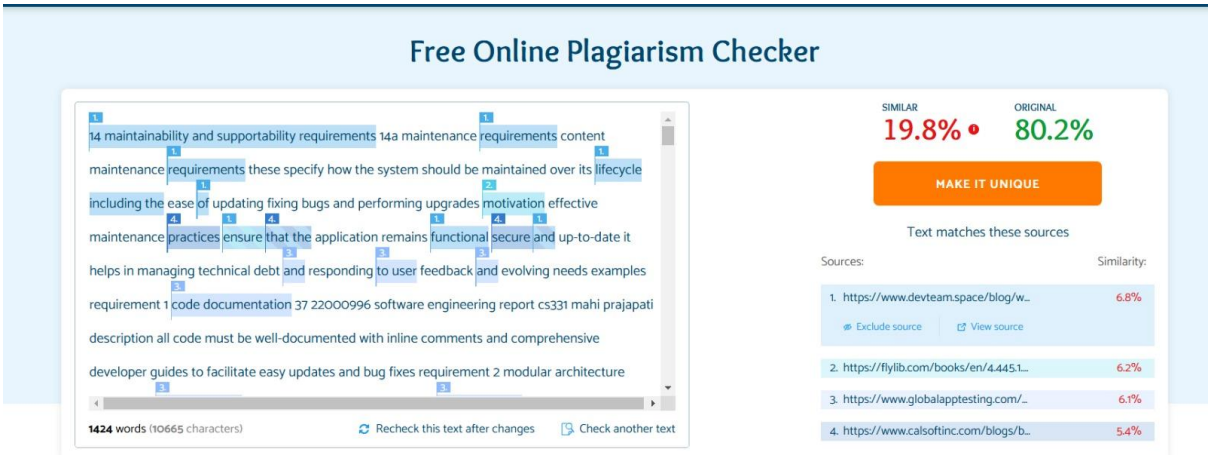

Figure 14 – 14 and 15

8. Module 16 and 17

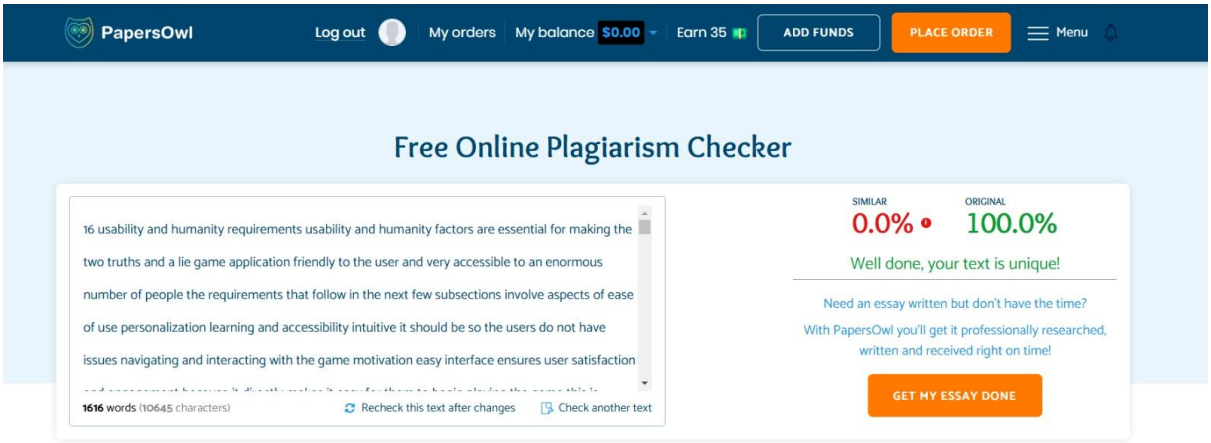

Figure 15 – 16 and 17

9. **Module 18 and 19**

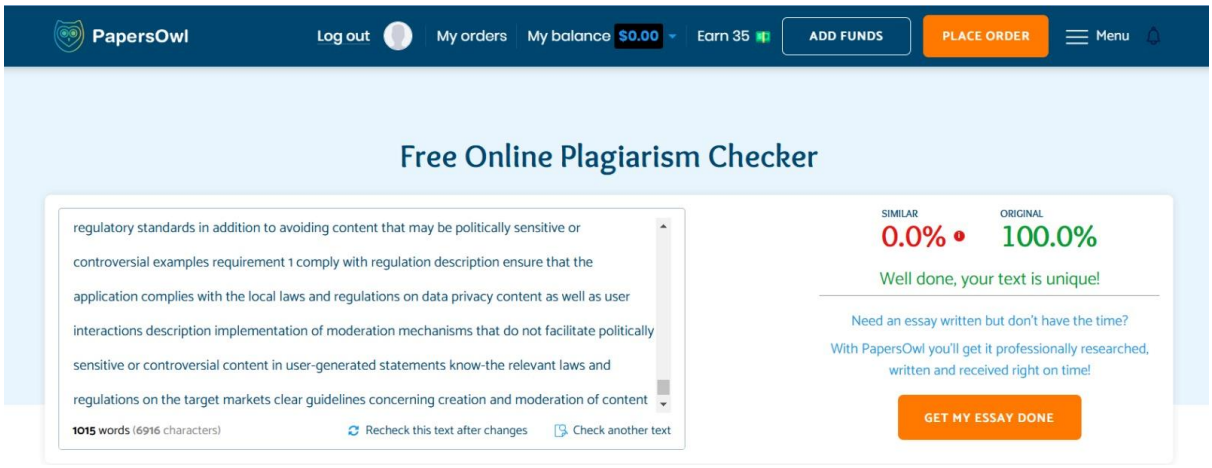

Figure 16 – 18 and 19

10. **Module 20,21 and 22**

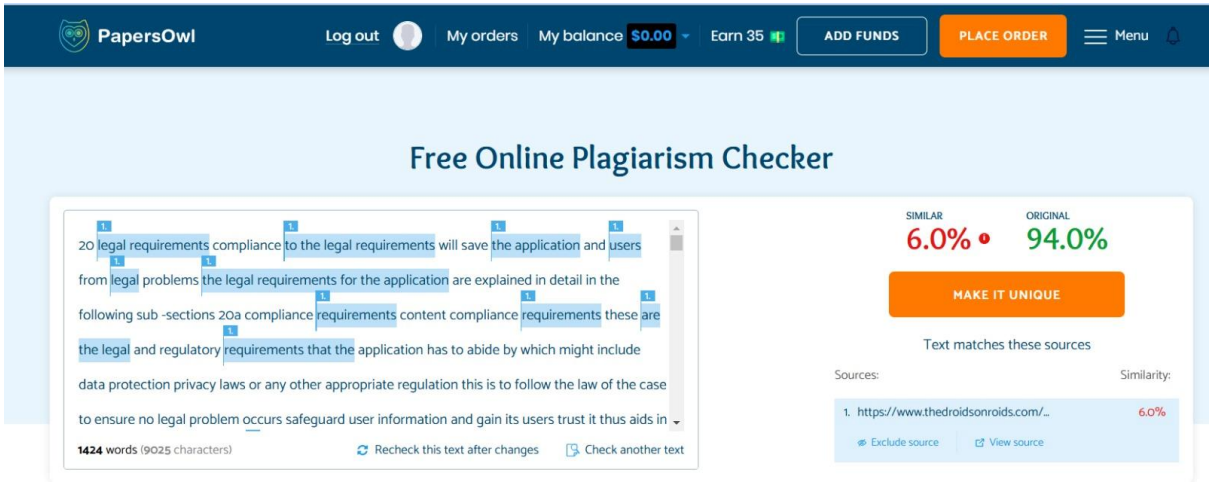

Figure 17 – 20,21 and 22

11. Module 23a to 23d

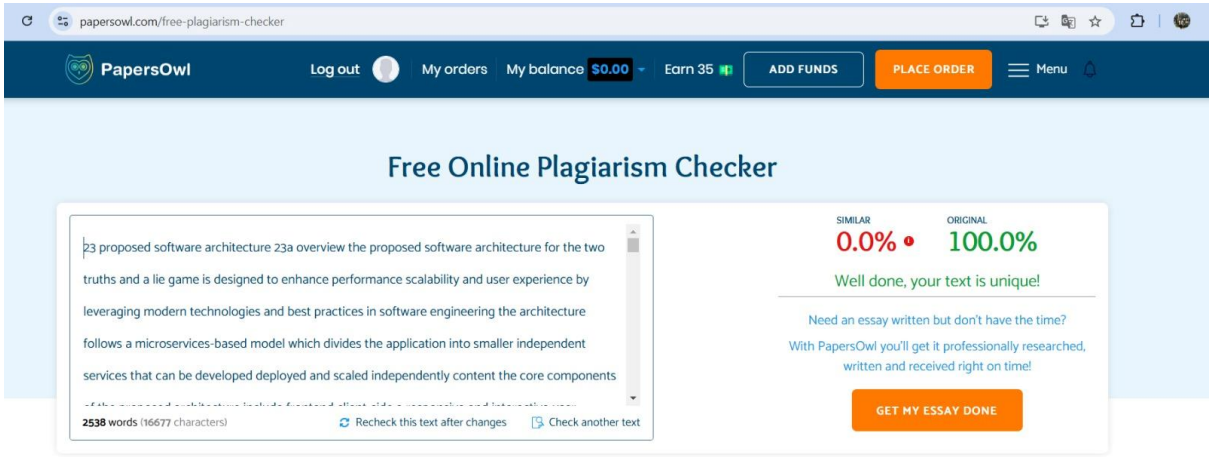

Figure 18 – 23a to 23d

12. Module 23e to 23h

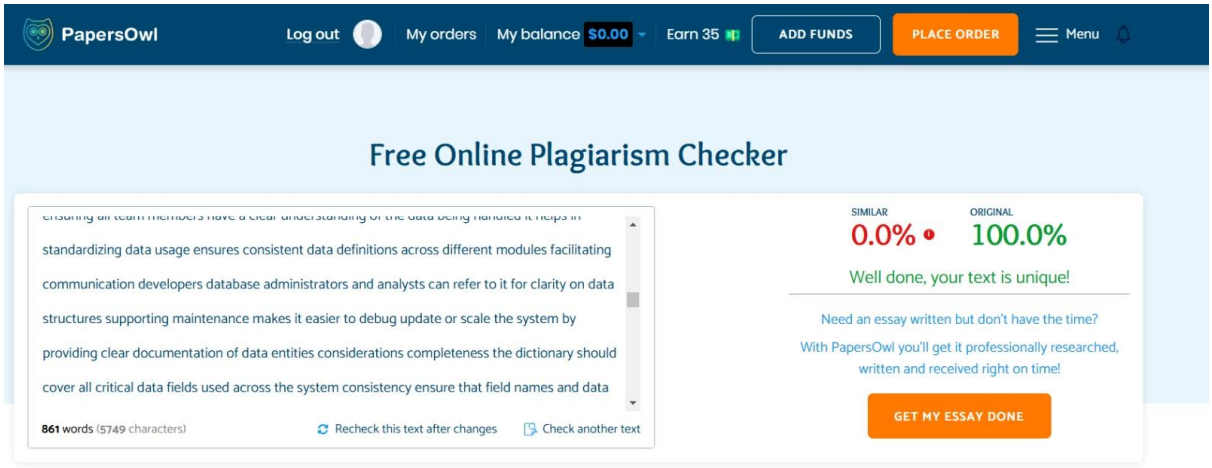

Figure 19 – 23e to 23h

13. **Module 24 and 25**

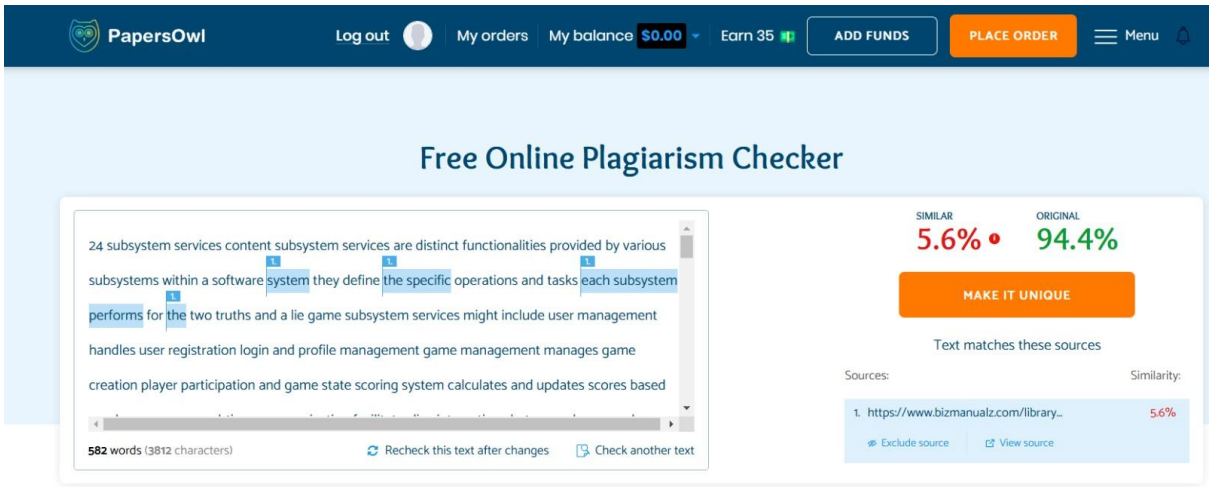

Figure 20 – 24 and 25

14. **Module 26**

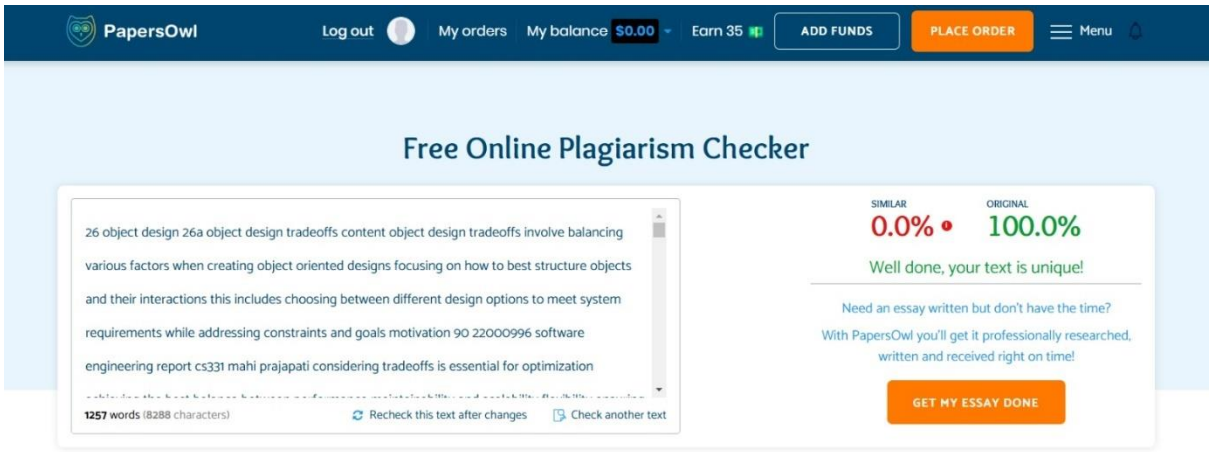

Figure 21 – 26

15. **Module 27 and 28**

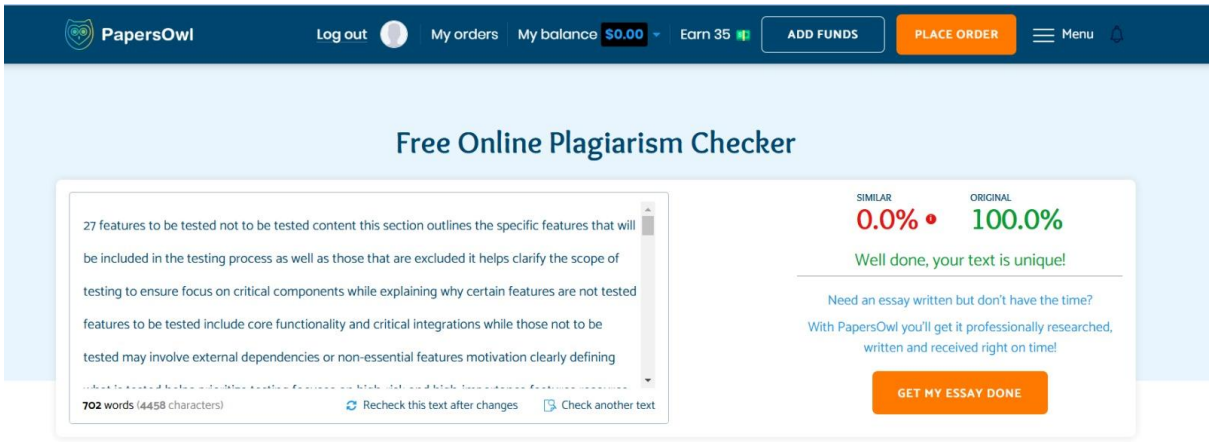

Figure 22 – 27 and 28

16. **Module 29 and 30**

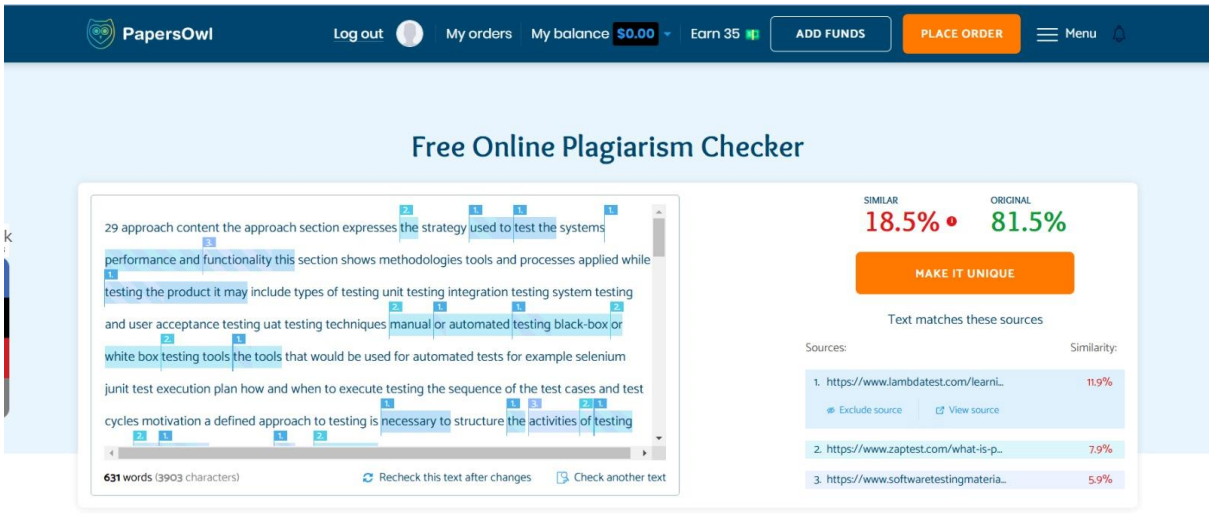

Figure 23 – 29 and 30

17. **Module 31,32 and 33**

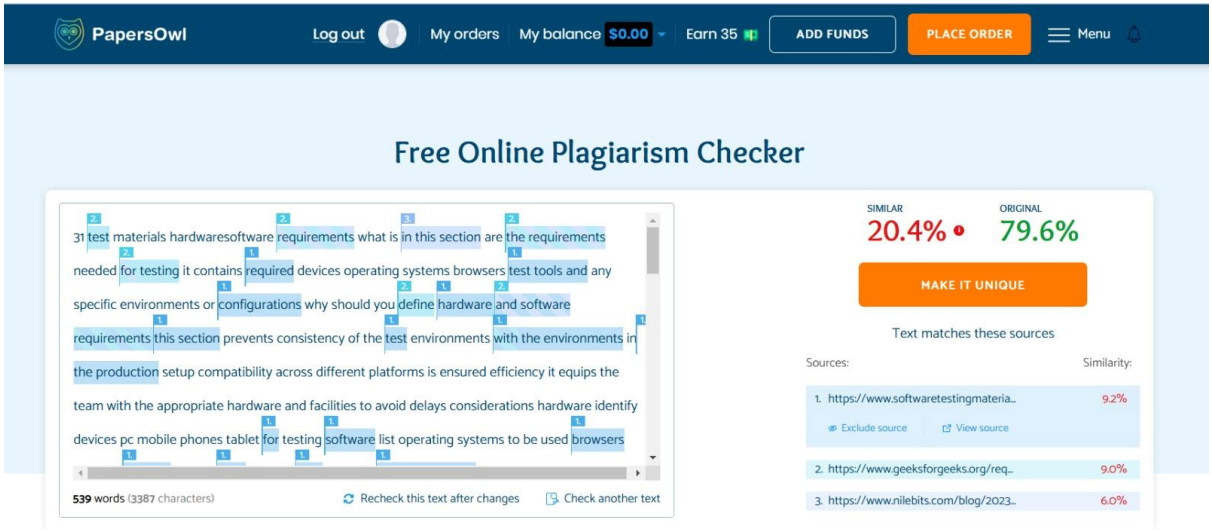

Figure 24 – 31,32 and 33

18. **Module 34 and 35**

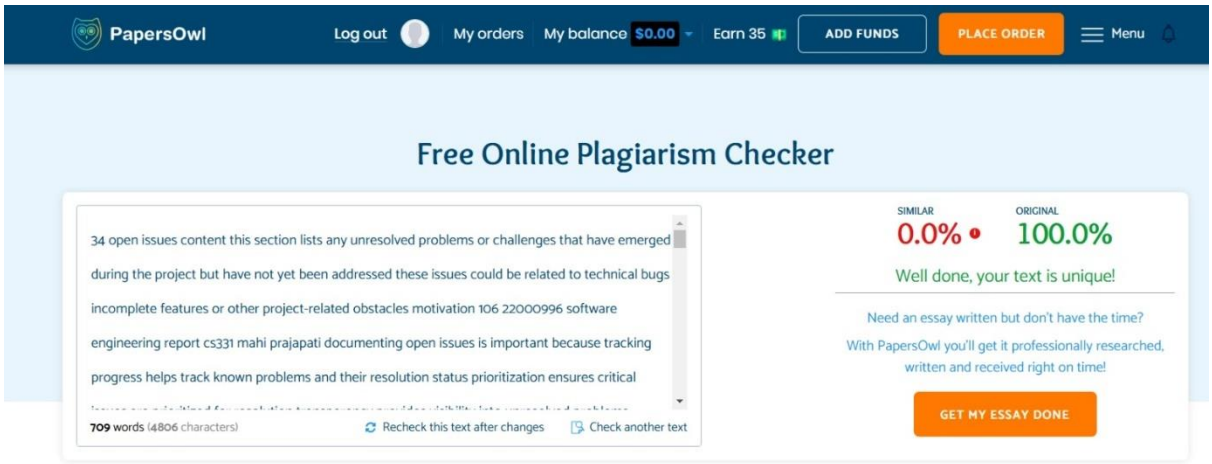

Figure 25 – 34 and 35

19. **Module 36 and 37**

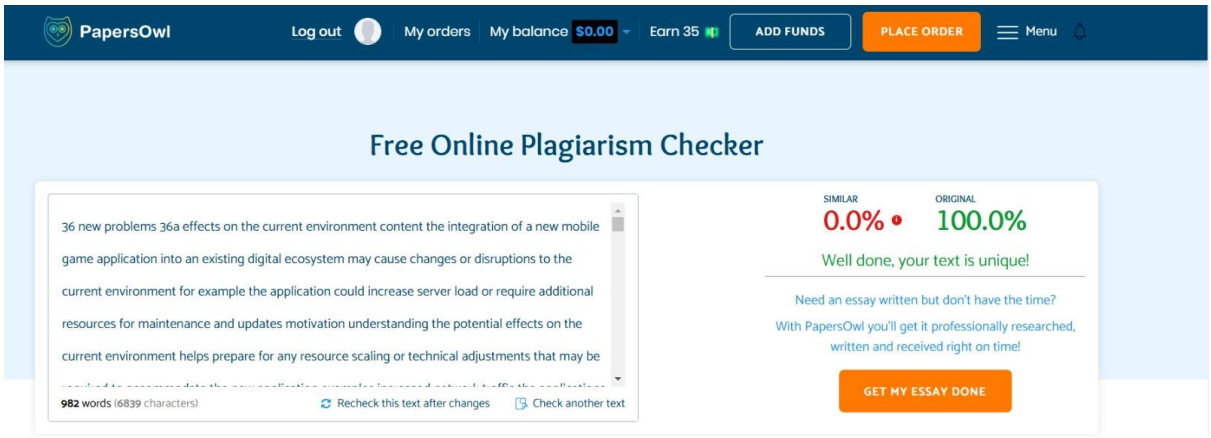

Figure 26 – 36 and 37

20. **Module 38 and 39**

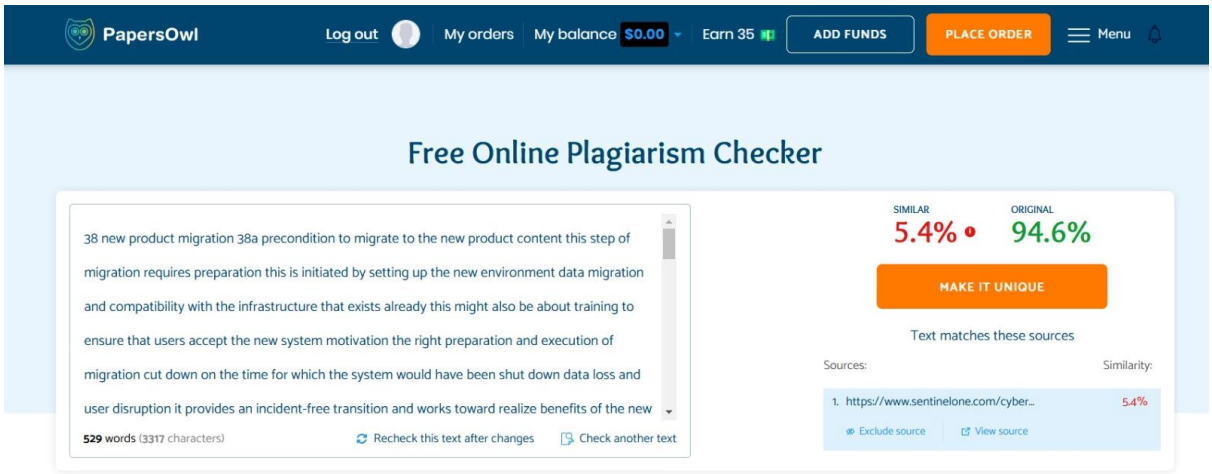

Figure 27 – 38 and 39

21. **Module 40,41,42 and 43**

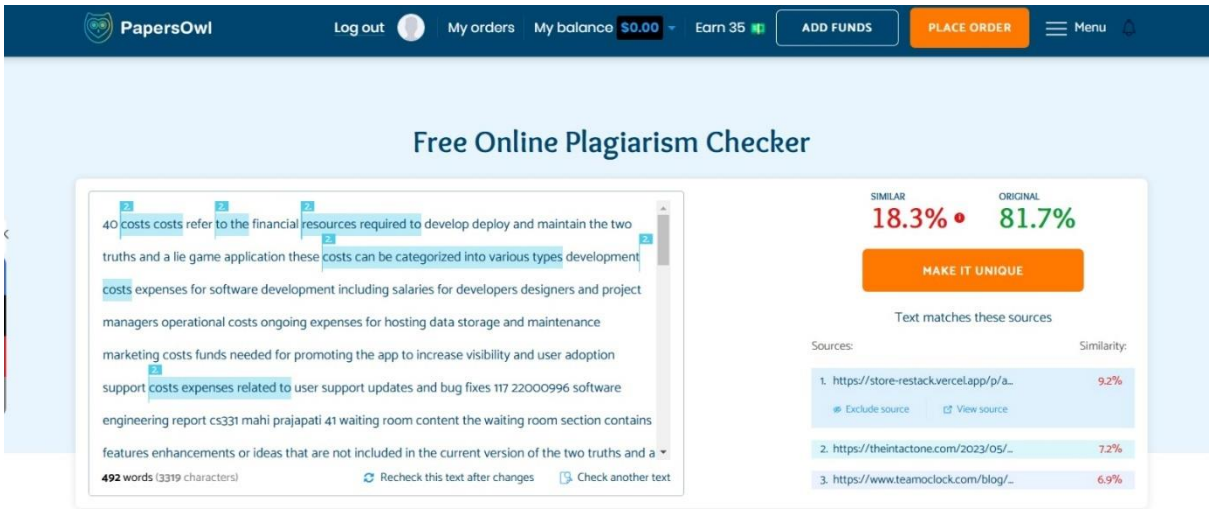

Figure 28 – 40,41,42 and 43