VOTAFUN

SYSTEM REQUIREMENT SPECIFICATIONS

Version 1.3
31/10/2023

Version History

Version #	Implemented By	Revision Date	Approved By	Approval Date	Reason
1.0	Ng Yue Jie Alphaeus	08/09/2023	Ng Yue Jie Alphaeus	08/09/2023	Initial system requirement specification draft
1.1	Ng Yue Jie Alphaeus	18/10/2023	Ng Yue Jie Alphaeus	20/09/2023	System requirement specification details added
1.2	Ng Yue Jie Alphaeus	09/10/2023	Ng Yue Jie Alphaeus	11/10/2023	Added additional details in sections 6 and 7.
1.3	Ng Yue Jie Alphaeus	31/10/2023	Ng Yue Jie Alphaeus	31/10/2023	Updated diagrams at section 3.4 and 10.2

Table of Contents

1. Problem Statement

In the context of various activities and choices, groups of friends face a recurring problem: they struggle to agree on a common activity due to their differing preferences and opinions. This diversity complicates decision-making, potentially leading to lengthy discussions and the inability to finalise plans efficiently.

The issue of indecision has real-world consequences. It results in inefficient use of time and undermines group cohesion. Prolonged debates can overshadow the enjoyment of shared activities, causing frustration and tension among friends. The initial enthusiasm and mutual respect can give way to disagreements and disrupt the group's harmonious dynamics.

A solution that balances individual preferences while streamlining decision-making is required. The goal of addressing these challenges is to restore efficiency to the process while maintaining the spirit of unity. Finding a method that respects individual input while allowing for faster decisions is critical for efficient planning.

2. Overview

2.1. Background

The project at hand is driven by the imperative need to improve decision-making efficiency among groups of friends when planning activities. The current situation often leads to time-consuming discussions and frustration. Our proposed design centres on the creation of a user-friendly platform while incorporating technology, specifically the integration of ChatGPT to facilitate structured discussions, activity proposals, and voting, all geared toward streamlining the decision-making process. This endeavour seeks to simplify communication by consolidating information within a dedicated platform, reducing misunderstandings and enhancing clarity. Moreover, the project aims to foster collaboration and consensus among friends with varying preferences, ultimately strengthening the overall decision-making process. Through these strategic design objectives, we aspire to revolutionise the way friends plan and decide on group activities, making it a more efficient and harmonious experience.

2.2. Overall Description

VotaFun is a web application designed for groups of people, particularly those looking to plan activities such as outdoor excursions, lunch plans or even movies to watch.

The following are objectives that VotaFun should achieve:

1. **Enhancing Decision-Making Efficiency**

   The project's primary goal is to improve decision-making among friends when planning group activities. The current situation of lengthy discussions and indecision frequently results in wasted time and frustration. Our goal in creating a user-friendly platform is to provide a structured environment where friends can collaboratively engage in discussions, propose activity ideas, and cast votes to streamline decision-making.

2. **Simplifying Communication**

   Current communication channels are fragmented. Groups may discuss activities on different platforms, resulting in a sprawl of information. This can lead to misunderstanding since information is discontinuous between users. Our goal is to improve communication by providing a platform dedicated solely to activity planning. This platform will consolidate information, proposals, discussions, and votes, making it easier for friends to stay informed and involved in decision-making.

3. **Collaboration in Promoting Consensus**

   We want to create a culture that values collaboration and compromise. Friends frequently have opposing viewpoints, making it difficult to reach an agreement. The project allows members in a group to vote anonymously. This encourages users to input their opinions when deciding on an activity when they might not have done so in an in-person setting. The app can then reach an activity which can best fit the group’s interests and can keep most of them happy.

3. Investigation & Analysis Methodology

3.1. System Investigation

VotaFun uses RESTful API calls to the backend in order to handle creation and joining of rooms. The server will allocate a room in the database in which the host can invite their friends over to join it via another API call. Once everyone has joined the lobby, the host can initiate a start to the game, in which VotaFun establishes and maintains a socket connection in order for the game to run and be in-sync with every client connected to the game room. While the session is in progress, API calls to OpenAI GPT model will be made, in which its responses are processed, and forwarded back to the clients.

3.2. Analysis Methodology

3.2.1. Feasibility study and requirements elicitation

There will be two teams organised for the purposes of this project, one that is familiar with developing enriching front-end websites, and a team familiar with working with socket programming as well as artificial intelligence. A series of interviews will be held for both teams. Interviews and feedback will be held for personnel who has worked in similar systems (ie. Kahoot) such that we are able to better understand their underlying implementation so that we can refine our system requirements. A feasibility and risk assessment study will be conducted to determine the best approach to developing this system.

3.3. System analysis and requirements specification

3.3.1. Perform an analysis of the problem using Object-Oriented techniques

An external view of VotaFun including Rooms, Users will be developed using Unified Modelling Language (UML). This System Requirement Specifications document will form part of the documentation for the project. More detailed documentation can be found in the accompanying Technical Specifications document. Some desired features of the system include:

- Ability to create/join room sessions
- Ability to manage room sessions (for host)
- Ability for users to vote for prompts in real-time
- Ability to generate prompts based on user responses using AI

3.3.2. Scope and Limitations

Analysis methodology will involve business analysis, requirement analysis, data analysis, process analysis (web) and application architecture:

- Business Analysis - State business rules, business system interfaces, business ownership, sponsorship and associated project budget requirement
- Requirement analysis - System I/O description, user requirement definition, functional and security requirement

- Process Analysis - Data/Process flow analysis, Networking communication analysis, process decomposition and system interface.
- Application Architecture - Analyse application structure, usability, user interface design, user experience and application implementation.

3.4. Object-oriented design using UML

A detailed object-oriented design for VotaFun will be developed. UML will be used again for graphical representation and documentation of a design. The system will primarily concern itself with the real-time room interactivity. At its core, once all users are connected via sockets to the server, the system will provide prompts in response to the users input. These prompts will be generated by AI in real-time and the responses are stored in a database. The user will be able to host sessions or join them. Creators of a new session, also known as the host, will be able to manage room settings as well as be able to remove users from the room if needed. The socket connection will be encrypted, thus ensuring security.

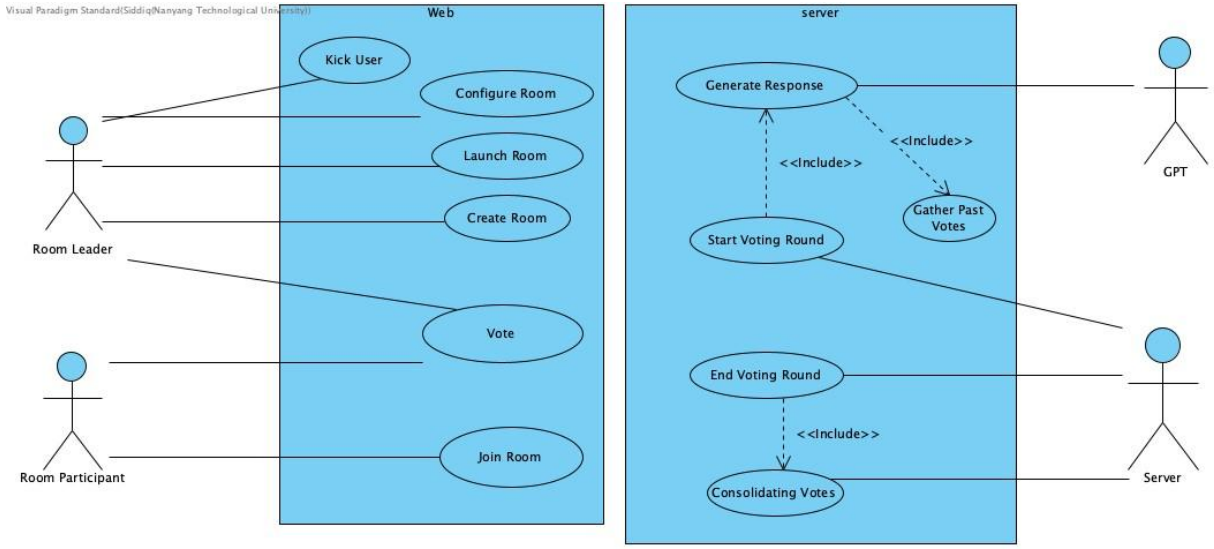

Figure 1. Use Case Diagram

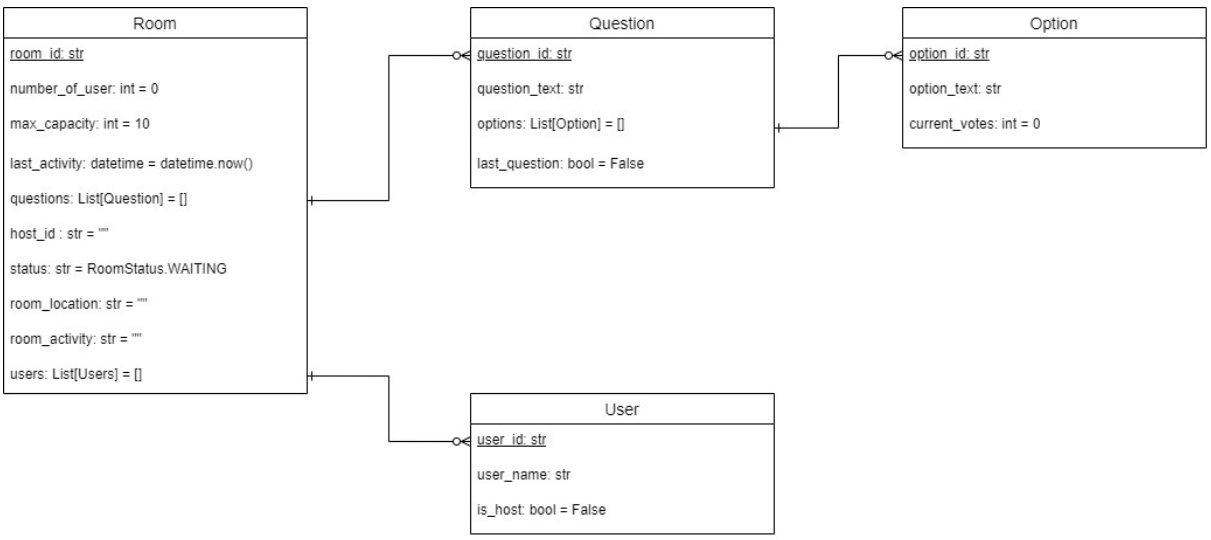

Figure 2. Entity Relation Diagram

3.5. Prototyping

Agile methodology and rapid prototyping will be used to develop a series of independent systems tackling the various technical requirements of this project. These systems include viability of socket programming in the context of this application, and a system with specifically crafted prompts to allow GPT to return responses in the expected format. A web interface will be developed as an end-user interface and a backend system is developed to handle all incoming connections as well as to manage room sessions. The prototype will be presented to the presentation team.

4. Constraints

4.1. Scalability

The web application must scale well with increasing demands, and up to 1000 concurrent users. The backend server must handle multiple API calls from the frontend within 30 seconds, while the database must be able to handle multiple concurrent reads and writes as room information is updated. Users must have a seamless experience and experience minimal disruptions even during increased loads.

4.2. Real-time Updates

Room information must be updated in real-time in the database. Room information that needs to be updated will include the number of users, the current users, the room host, and ChatGPT generated questions and options for a room. The database must update voting results as well.

The user interface must be updated in real-time whenever someone joins, leaves or when there is a change in the room host. It must also update when it receives questions and multiple options for that question from the backend server.

4.3. Project Schedule

The project will run for three months, starting on 24th August 2023, and ending on 26th October 2023.

5. Operational Requirements

5.1. Application Services and Technical Support

Emails will be the main channel of technical support for VotaFun’s users. Users can report technical issues such as application errors and slow system response. The developer team will respond as soon as possible.

The email channel will also be open to bug or vulnerability reporting, and users are encouraged to submit vulnerabilities found. Developers will have access to the source code to address technical issues, bugs and vulnerabilities raised by VotaFun’s users.

5.2. System hardware failover and routine backup

VotaFun is hosted on Heroku, a platform as a service (PaaS) cloud. Failover systems have been set up on our secondary cloud provider, Google Cloud, such that if Heroku goes down and users cannot access VotaFun, our secondary cloud provider will start to serve users.

VotaFun uses the concepts of rooms, which are short-lived and deleted after users have decided on their activity. Hence, there is no need for VotaFun to have a routine user data backup.

5.3. Audit Trail

VotaFun will log all user's activities, including web application access, users creating or joining rooms, API calls, and when the database stores data in a log file.

6. Functional Requirements

The VotaFun system is a web-based application system that involves real-time voting and storage of data.

6.1. Host privileges

The user who creates and joins a room will be deemed as the room host. Upon a user's disconnection, if the user was the room host, a new room host will be selected at random if there are existing users in the room, else the room will be closed.

6.1.1. The host can set the maximum amount of users in a room

6.1.2. The host can choose the activity type and location in the lobby page.

6.1.2.1. This selection will be visible to the other users in the lobby. Once finalised, the host can start the room by clicking the "Start room" button.

6.1.3. The room host must be able to generate an invite link to pass to other users to allow access into the room.

6.1.3.1. Users with an invalid invite link will not be able to join a room.

6.1.4. The room host can choose to close the room prematurely.

6.1.4.1. This will remove all users from the room and close the room instantly.

6.1.4.2. The users will be brought to the landing page with a message shown on their screens.

6.1.4.3. Room data associated with the closed room will be removed from the database.

6.1.5. The room host must be able to start a room by clicking the "Start room" button.

6.1.5.1. Once the room is started, the room status will be updated to "STARTED".

6.1.5.2. The room status, activity type, location of the room will be updated in the database.

6.1.5.3. All users will be notified of the updated activity type and the location of the activity.

6.1.6. The host can choose to remove a user from a room through a button.
6.1.6.1. The list of users in a room will be updated in the database.

6.1.7. The host will have an icon displayed beside his username to denote that he is the current host.
6.1.7.1. At any point in time, there can only be a single room host.
6.1.7.2. If the host leaves the room, a random user remaining in the room will be selected as the room host.

6.2. User functions (includes room host)

6.2.1. Users with a valid invite link must be able to join a valid room.
6.2.1.1. If the room status is “STARTED”, users cannot join the room.
6.2.1.2. If the room is at max capacity and new users cannot join that room.
6.2.1.3. The max capacity of the room can be changed by the host of the room.

6.2.2. Users are required to input their usernames before joining/ creating the room.
6.2.2.1. The username will be an alphanumeric string.
6.2.2.2. This username will be visible to the other users in the room in the lobby and game room pages.
6.2.2.3. The username will be updated in the database.

6.2.3. Users must be able to vote for their favourite option during each round.
6.2.3.1. This is done by clicking on the option displayed on the user interface before the timer in each round is up.
6.2.3.2. Users can only vote for one option each round.
6.2.3.2.1. The remaining options will not be clickable and users cannot select them anymore.
6.2.3.3. The option that was voted for will be stored in the database.

6.2.4. Users must be given the option to leave a room at any time.
6.2.4.1. The database will be updated to remove the user data from the room.
6.2.4.2. If the last user leaves a room, and there are no other users in a room, the room will be closed and the database will remove the room data associated with that room.

6.3. Interaction with application

To make the experience fun and interactive for our users, we require our application to fulfil the following points.

6.3.1. Each round, users will be shown a question and options generated by ChatGPT.
6.3.1.1. Users will have 15 seconds to cast their votes.
6.3.1.2. After 15 seconds is up, users must not be able to vote for any options.
6.3.1.3. The server will implement a countdown, and notify all clients in the room once the time is up.
6.3.1.4. The option that a user in the room has voted for will be updated in the database.

6.3.2. Users must be given at least 2 voting options for each question. No textual input is given.

6.3.2.1. Users will click on the option to vote for it.

6.3.2.2. Users can only vote for a single option.

6.3.2.3. The option that a user voted for will be stored in the database.

6.3.3. At the end of 5 rounds, users will be presented with 4 activities recommended by ChatGPT.

6.3.3.1. Users will also be given 15 seconds to vote for their favourite activity recommended by ChatGPT.

6.3.3.2. The activity that has the most votes will be displayed to all users in the room.

6.3.4. If there are any errors (network errors, server unavailability or server error) a popup will appear, along with a button “Return home”.

6.3.4.1. Users will be able to click on the “Return Home” button to return to the landing page in which a new session is created and the user attempts to reconnect to the backend socket once again.

6.4. Prompt engineering

The system must interact with ChatGPT via Prompt Engineering to generate question prompts used by the users.

6.4.1. The system must perform prompt engineering on ChatGPT.

6.4.1.1. The question and options generated at each round must be based on the votes of the previous round. This excludes the first round as users have not voted yet.

6.4.1.2. During each round, the past questions, options, and the number of votes for each option will be retrieved from the room to create the prompt for ChatGPT.

6.4.2. The system must receive and clean the prompt received from ChatGPT’s API before displaying it to users.

6.4.2.1. The system must only retrieve the questions, options and activities generated by ChatGPT. Any other information returned from ChatGPT must be removed.

6.4.2.2. If ChatGPT generates less than 2 options for a question, the system must call ChatGPT’s API again to try to regenerate the question and options.

6.4.2.2.1. The retry logic will run for a maximum of five times.

6.4.2.2.2. If ChatGPT is unable to give more than 2 options at the end of the retry logic, an error will be raised.

6.4.2.3. The questions and options generated by ChatGPT must be stored in the database, under the specific room.

6.4.3. At the end of 5 rounds, ChatGPT must recommend 4 activities for users to choose from.

6.4.3.1. The activities recommended must be based on the votings of the previous five rounds.

6.4.3.2. ChatGPT must include the location of the activity for the users.
6.4.3.3. These activities will be the final round for the users to vote from.
6.4.3.4. These activities will be stored in the database, under the specific room.

6.5. Socket-based communication

The system facilitates real-time, bidirectional data exchange between the server and the various clients to support the real-time voting functionality.

6.5.1. Socket Establishment
6.5.1.1. The system must establish and maintain socket connections between the backend and frontend to enable real-time communication.
6.5.1.2. The system must use a secure socket protocol (e.g., WebSocket or secure TCP/IP) to ensure data privacy and integrity.

6.5.2. Data Exchange
6.5.2.1. The system must support real-time data exchange, including text messages, multimedia content (e.g., images and videos), and system notifications.
6.5.2.2. It should handle incoming and outgoing messages efficiently to minimise latency and ensure smooth user experiences.

6.5.3. Message Handling
6.5.3.1. Messages sent between the frontend and backend must adhere to a defined message format to ensure consistency and ease of parsing.
6.5.3.2. The format should include fields for sender identification, timestamp, message content, and any associated metadata.

6.5.4. Error Handling
6.5.4.1. The system must have robust error handling mechanisms to address various issues that may arise during socket-based communication, including disconnections, network errors, and server unavailability.
6.5.4.2. It should provide informative error messages and gracefully recover from errors when possible.

7. Non-Functional Requirements

7.1. Reliability

7.1.1. The web application must be available for 99% of the time.
7.1.2. The web application must be able to handle errors in user inputs.
7.1.2.1. Errors include users entering an invalid room code.
7.1.3. Room data stored must be accurate and consistent.

7.2. Performance

7.2.1. Room data must load within 30 seconds of creating or joining a room on a 4G or Wi-Fi connection.

7.2.2. The web application must be able to support 500 concurrent users.

7.2.2.1. Each room must support a maximum of 10 users.

7.2.3. The user interface must render questions and options on the screen within 30 seconds after receiving data from the backend.

7.2.3.1. The backend must complete all application programming interface (API) calls within 30 seconds.

7.2.3.2. The backend must return any errors while processing API calls within 30 seconds.

7.2.4. The web application must be compatible with different web browsers with consistent performance.

7.3. Usability

7.3.1. The user interface must be clear and intuitive.

7.3.1.1. Users must be able to navigate the web application with minimal tutorials.

7.3.2. The user interface must be responsive to user interactions.

7.3.2.1. The user interface must be updated within 30 seconds when a user leaves the room, joins rooms, and votes for an option.

7.4. Security

7.4.1. The web application must encrypt all socket connections.

8. Input Requirements

8.1. Creating rooms

Room hosts must select from a list of options and choose the preferred type of activity (e.g. indoor, outdoor) and the location for the activity (e.g. North, South, East, West) before launching a room for users to join.

8.2. Joining rooms

Users must enter a valid room code to join a launched room. The room code is unique and maps to a launched room. Users cannot join a room if they enter an invalid room code that does not map to a room.

Users must also enter an alphanumeric string which will represent their username in the room.

8.3. Voting for options

The user interface will display a question and multiple options during each round. Users will vote for their favourite option by clicking on the corresponding button.

9. Process Requirements

The following requirements cover the platform's core functionality that the system must be able to handle.

9.1. Database Transactions

The system must manage transactions with the underlying database system, Redis, in real-time. This includes being able to send, receive, trigger, and complete transactions from the Redis database system.

9.2. Data Integrity

All processes have to function as expected by the user, and all data processed and stored must be accurate. It has to avoid committing corrupted or incomplete records to the database.

9.3. Data Validation

The system must perform extensive data validation and error-handling routines. This ensures that data errors resulting from both user inputs and database processing are handled correctly, reducing user frustration and improving activity planning clarity.

9.4. Performance

The system must resolve locking issues and efficiently handle concurrent platform usage on a 24x7 basis. In addition, the system must be designed to send, receive, and display user and system messages within 2-3 seconds.

10. Output Requirements

10.1. Room Information

When the user creates a room, the information should be stored in the Redis database.

10.2. Summary of Voting Results

After completing the voting rounds, the user will be presented with the final activity and the suggested location.

11. Hardware Requirements

11.1. Network

Users using VotaFun will need a stable internet connection (4G or Wi-Fi). Stable internet connection is required as the user is required to be connected to the server via socket at all times to ensure everything works as intended.

11.2. Computers

VotaFun must be able to be run on a computing device that is able to run web applications on a supported browser (Safari, Google Chrome, Firefox etc.). The host device should be kept up to date to ensure a smooth experience.

12. Software Requirements

12.1. Client Operating System

Unix (any distro), MacOS, Windows, iOS, iPadOS, Android

12.2. Web application

VotaFun must be able to run on any react compatible browser such as the following:

- Google Chrome
- Microsoft Edge
- Mozilla Firefox
- Opera
- Safari

12.3. Network system

Network software and protocols in order for the systems to communicate:

- TCP/IP
- HTTP
- HTTPS

12.4. Licences

Valid licences are required to run software from third-party vendors:

- To use application development tools.
- To use web server, application server and database software in development, test and production mode.