Capstone Project: Ft_Transcendence

Team Members:
Abdul Samad Pila Valappil (apila-va)
Ghaiath Abdoush (gabdoush)
Nousheen Ali (nali)
Sumaiya Fathima (sfathima)
Yonatan Moges (yonamog2)

Date of Submission: 14-December-2023

Abstract

Spin Masters is an online multiplayer pong game that offers its users an immersive and real-time gaming experience. Its visually captivating interface provides a social platform for gamers to interact with like-minded individuals, build a community, and enjoy a gaming environment in a robust and secure technology stack.

Crafted as a tribute to the timeless 1972 classic Pong game, this application offers a seamlessly immersive multiplayer gaming experience. Users can personalize their gaming adventure by customizing the game's appearance, adding a personal touch to the excitement. With an integrated chat functionality and a friend management system, the platform not only promotes social interaction but also fosters the formation of vibrant gaming communities. It sets the stage for endless hours of fun and competitive gameplay.

Spin Masters is specifically designed for users with a 42-school login along with an option to enable 2FA enhancing the security of the user accounts. The platform is designed to be compatible with the latest stable version of Google Chrome and Mozilla, ensuring that every player can enjoy the game without compromise.

This Application aims to blend the nostalgia of classic Pong with modern multiplayer dynamics, fostering a vibrant community, and prioritizing user security through features like 2FA.

Table of Contents

List of Figures

List of Tables

List of Abbreviations Used

ABBREVIATIONS	DEFINITIONS
2FA	Two-Factor Authentication
OAuth	Open Authorization
API	Application Programming Interface
DTO	Data Transfer Objects
JWT	JSON web Token
CORS	Cross-Origin Resource Sharing
NPM	Node Package Manager
SSR	Server-Side Rendering
SSG	Static Site Generation
SQL	Structured Query Language
JSON	JavaScript Object Notation

1 Introduction

Spin Masters is an innovative and entertaining online platform crafted to enable users to engage in real-time Pong matches. With an attractive user interface, this application provides a fluid experience, allowing users to play, chat, connect with friends, and enjoy moments of fun.

At the core of this application is NestJS backend which ensures robustness, scalability, and efficiency through its modular architecture. NestJS with Prisma establishes a clean and reliable approach to managing PostgreSQL database interactions ensuring security and efficiency. Real-time communication is powered by WebSockets, enabling instantaneous updates not only in the game but also in the integrated chat system and notifications.

The front end is built on the Next.js framework, which seamlessly integrates WebSockets and the Phaser game engine while offering a responsive and dynamic interface.

The use of the Phaser game engine elevated the game mechanics and visuals. It ensures a fast and flexible 2D gaming experience with easy placement of scenes, audio, and sprites. Combined with WebSockets, it delivers real-time synchronization of game events, creating a wonderful gaming experience

Some Key Features of the Application:

- **Dashboard:** Provides an extensive overview of the player's achievements within the application. The page shows one's game history and their outcomes along with live games.
- **Leaderboard:** Showcases the list of players as a ranked list giving a comprehensive view of the entire player community. Additionally shows the top-ranked players in the page's header banner.
- **Friends Management:** Enables users to find and interact with other users within the system. After establishing a connection, users can communicate and engage in gaming.
- **Integrated Chat:** Provides opportunities to interact with friends and create communities through our channels feature. The chat interface extends beyond communication, allowing users to seamlessly invite others to play Pong, access player profiles, and create a dynamic and inclusive gaming community.
- **Multiplayer Pong Game:** a canvas game that pays homage to the 1972 classic Pong, offering matchmaking and customization features.

- **Two-Factor Authentication(2FA):** Security feature to provide an additional layer of security to the user's account. Users require a 42 login to create an account on our platform and have the option to enable Google authenticators authentication.

Spin Masters aims to provide a comfortable gaming experience while fostering a vibrant and inclusive community for endless moments of fun and competition.

2 Software Development Life Cycle (SDLC)

The objective of this project is to create a single-page application with an integrated Chat, friend Management system, and an online multiplayer Pong Game. The mandates included a containerized application that employs a PostgreSQL database with a NestJS backend and any Typescript frontend.

2.1 Requirement Analysis

During this phase, a detailed analysis of functional and non-functional requirements for the application was conducted to identify the skills and technologies needed in each phase. Decisions on the technology stack and overall application structure were finalized.

Next.js was selected as the frontend framework since it supports client-side navigation (allowing faster transitions between pages), and breaking code into smaller components thereby optimizing performance.

Phaser was chosen as the game engine as it is designed to run on the client side thereby reducing the load on the server for multiplayer games. Its compatibility with Web Sockets, an important aspect of multiplayer gaming, was also a crucial factor in finalizing it.

Socket.IO was chosen as the WebSocket library considering its compatibility with both the server-side (Nest.js) and client-side (Next.js, Phaser) implementations. Authentication has been efficiently managed through the Nest.js backend, ensuring a cohesive and secure integration within the web application.

2.2 Design

During this phase, the following tasks were completed:

- A detailed prototype was created for the Spin Masters website using Figma, a leading collaborative design tool. This facilitated Identifying the various components and their interaction
- A GitHub repository was set up for version control and code reviewing purposes.
- A JIRA account was maintained for smooth project management. This facilitated the seamless allocation of tasks and provided a streamlined tracking mechanism, enhancing overall project organization and collaboration.
- Created a Risk Register to identify, assess, and manage potential risks that could impact the successful execution of a project

- Design the database schema to efficiently store and retrieve data.

2.3 Implementation

The implementation started with establishing the foundational structure of the front end. The focus was on creating skeletal components capable of rendering placeholder data. This helped determine the necessary data types for each component.

Once the front end was ready, it facilitated the segregation of backend modules. The integration of the Prisma package helped streamline interactions with the PostgreSQL database, enhancing data handling and retrieval. API Endpoints were created and rigorously tested using Postman to validate functionality and responsiveness. Data Transfer Objects (DTOs) were maintained for all NESTJS to ensure a clear and controlled flow of information and prevent unnecessary exposure of internal structures. Exception handling was incorporated to prevent unnecessary program crashes. This approach enhances the application's robustness by gracefully managing unexpected errors

The next step involved integrating these API endpoints with their corresponding frontend components. The login system was implemented using 42 OAuth API together with 2FA using Google Authenticator for added security. To handle real-time communication required for Chat, Game, and Notifications, the WebSocket library Socket.io was used. The dedicated Gateways, services, and Controllers were added in the backend for these modules for smooth workflow.

Finally, token authentication was added to ensure security and robustness. To ensure code quality, multiple levels of code reviews were conducted at every stage of the implementation, emphasizing the promotion of code reusability and adherence to coding standards.

2.4 Testing

The testing phase started with testing individual modules. The components were tested to validate their functionalities and correctness. This involved subjecting various scenarios specifically targeting the Game, Chat, Notifications, and User Profile modules. This was to ensure their reliability and adherence to the specified requirements.
Subsequently, in Integration testing, it was ensured that different parts of the application worked seamlessly and cohesively. It was made sure all requests from the front end to the back end involved token credential verifications preventing unauthorized access.

Performance Testing involved testing the application on various browsers to ensure a consistent and responsive user experience. The game was tested with multiple game sessions.

2.5 Evolution

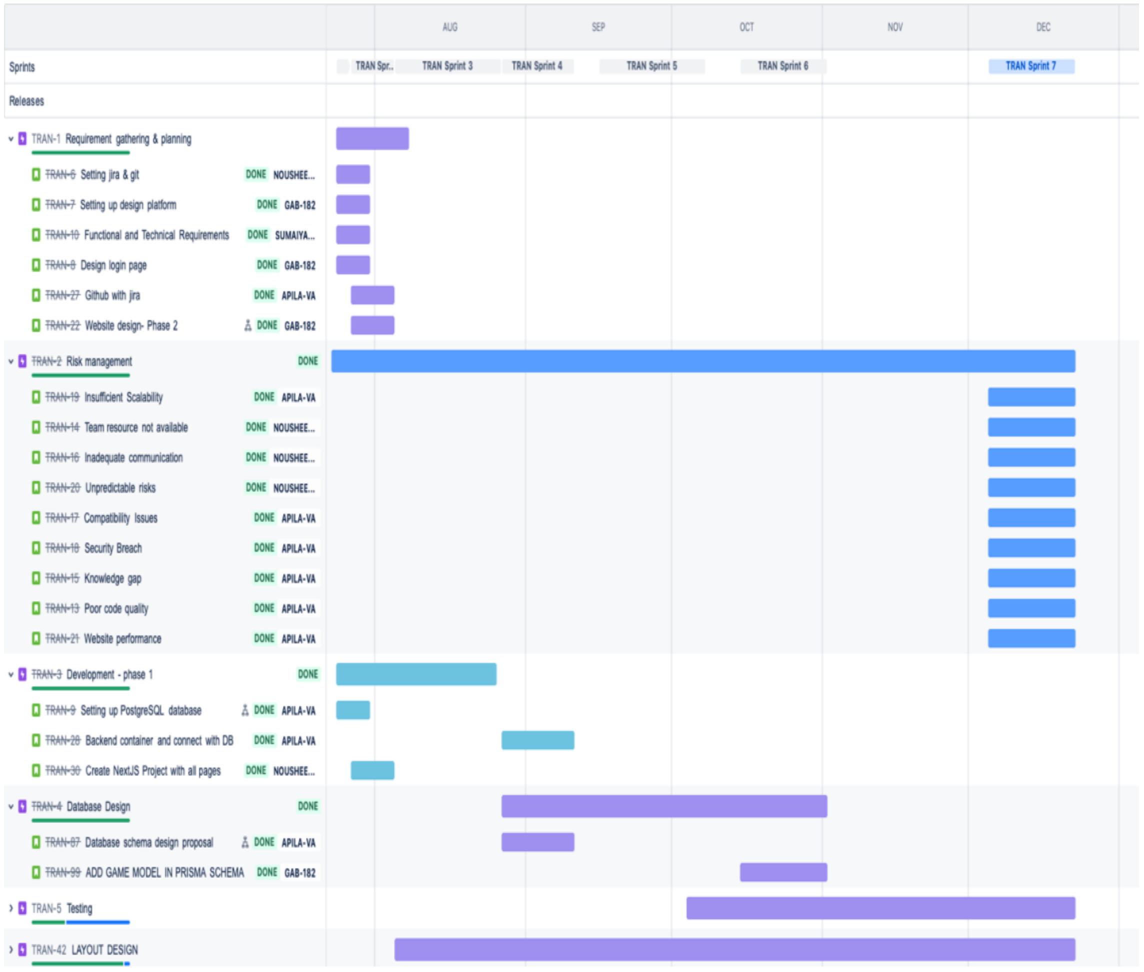
This Phase involved the following actions:

- Addressed issues and bugs identified during the testing phase.
- Game performance optimization was done with the 'WebGLrenderer' option in Phaser. This leverages hardware acceleration for graphics rendering thereby improving visual quality.
- The application was containerized using docker thereby streamlining the deployment process.
- Documentation was updated to reflect changes made in the previous phases.

3 Additional Requirements in SDLC

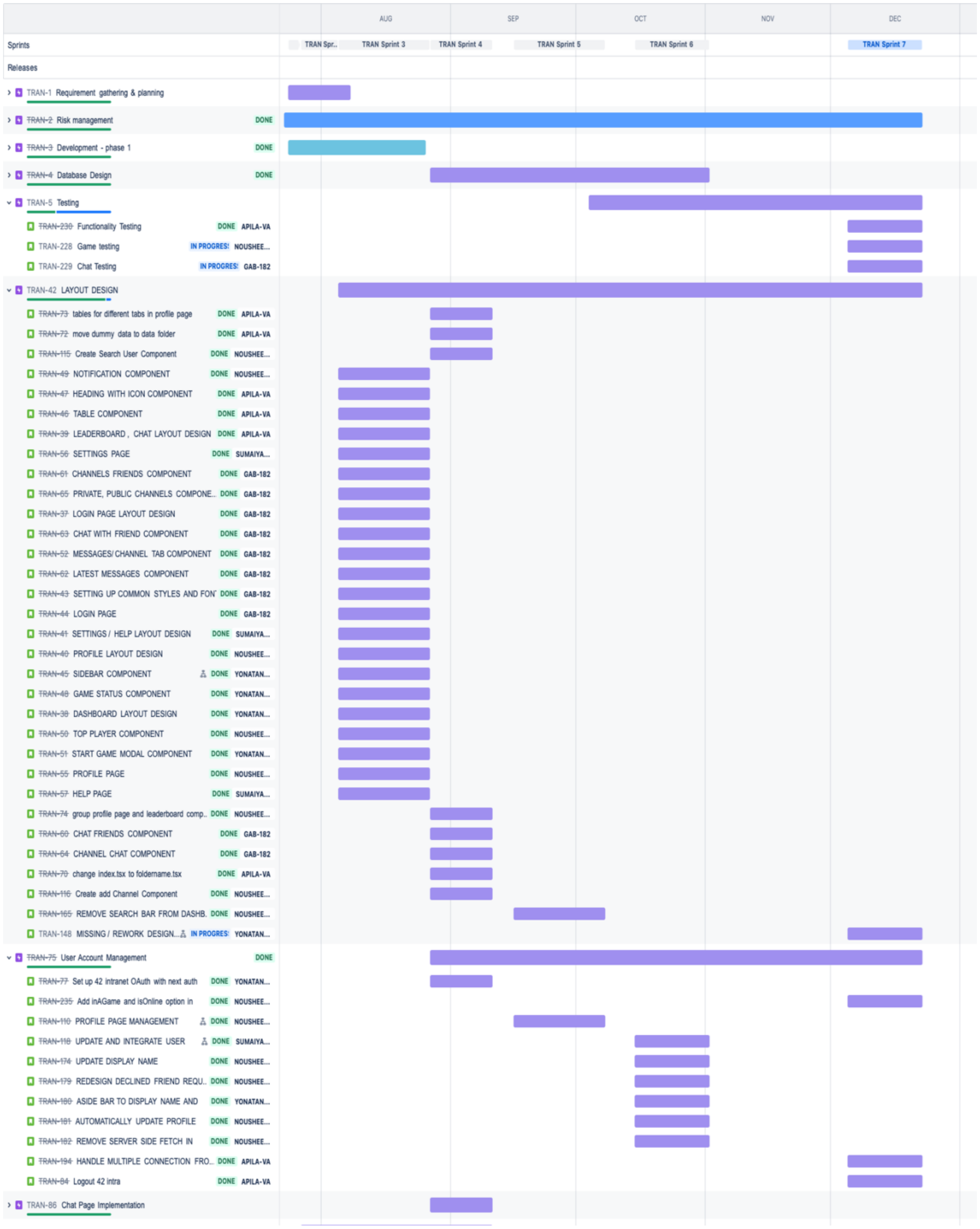
3.1 Gantt Chart

This project was divided into multiple epics and was completed in 7 sprints taking close to 4 months for completion. The following images show a visual representation of how task allotment was done in each epic and the time taken to complete them, with the help of a Gantt chart. These Images were generated using the Project Management software JIRA.



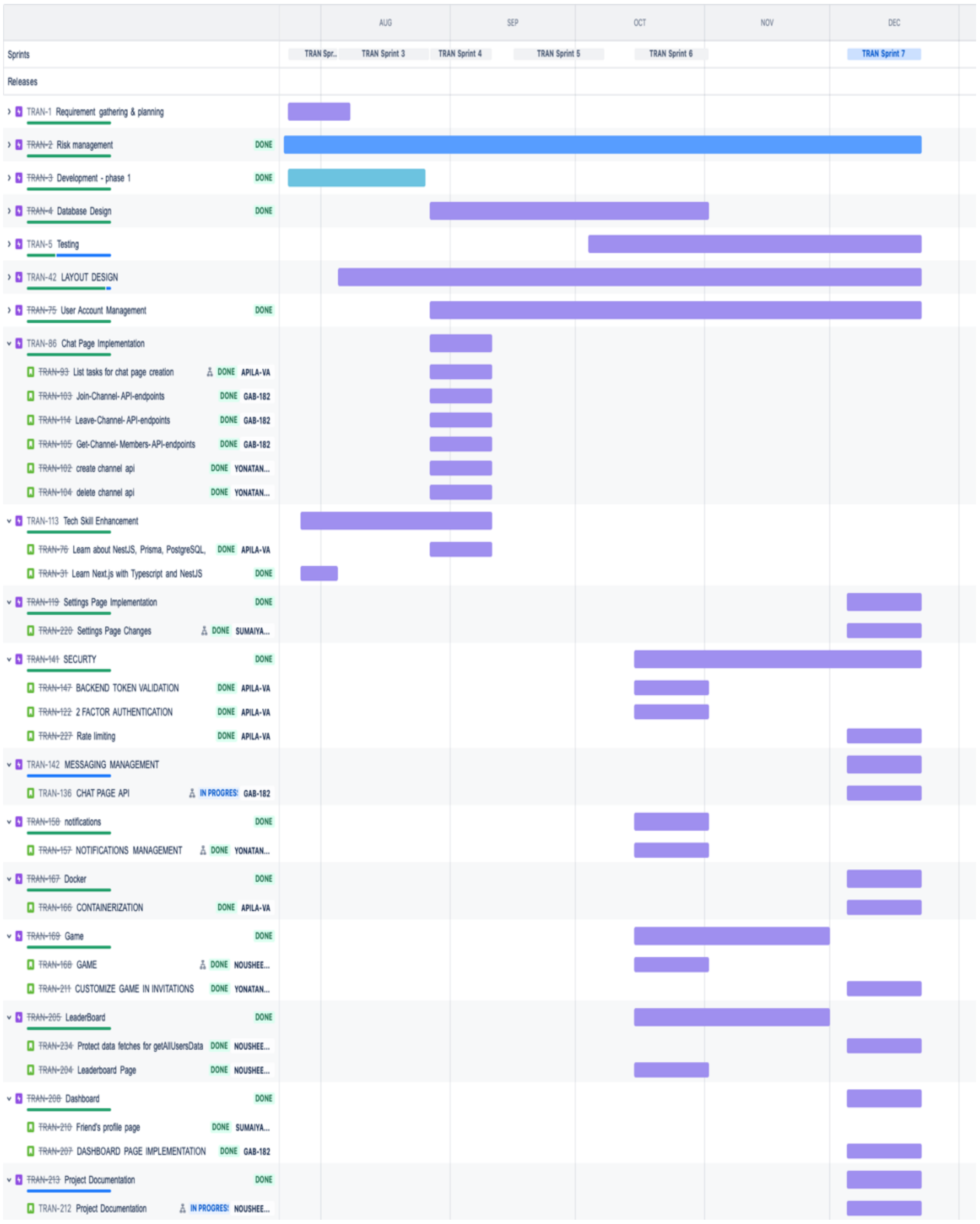
TRAN-1 Requirement gathering & planning		
~~TRAN-6~~ Setting jira & git	DONE	NOUSHEE...
~~TRAN-7~~ Setting up design platform	DONE	GAB-182
~~TRAN-10~~ Functional and Technical Requirements	DONE	SUMAIYA...
~~TRAN-8~~ Design login page	DONE	GAB-182
~~TRAN-27~~ Github with jira	DONE	APILA-VA
~~TRAN-22~~ Website design- Phase 2	DONE	GAB-182
~~TRAN-2~~ Risk management	DONE	
~~TRAN-19~~ Insufficient Scalability	DONE	APILA-VA
~~TRAN-14~~ Team resource not available	DONE	NOUSHEE...
~~TRAN-16~~ Inadequate communication	DONE	NOUSHEE...
~~TRAN-20~~ Unpredictable risks	DONE	NOUSHEE...
~~TRAN-17~~ Compatibility Issues	DONE	APILA-VA
~~TRAN-18~~ Security Breach	DONE	APILA-VA
~~TRAN-15~~ Knowledge gap	DONE	APILA-VA
~~TRAN-13~~ Poor code quality	DONE	APILA-VA
~~TRAN-21~~ Website performance	DONE	APILA-VA
~~TRAN-3~~ Development - phase 1	DONE	
~~TRAN-9~~ Setting up PostgreSQL database	DONE	APILA-VA
~~TRAN-28~~ Backend container and connect with DB	DONE	APILA-VA
~~TRAN-30~~ Create NextJS Project with all pages	DONE	NOUSHEE...
~~TRAN-4~~ Database Design	DONE	
~~TRAN-87~~ Database schema design proposal	DONE	APILA-VA
~~TRAN-99~~ ADD GAME MODEL IN PRISMA SCHEMA	DONE	GAB-182
TRAN-5 Testing		
TRAN-42 LAYOUT DESIGN		

Figure 1: Gantt Chart



Sprints			
Releases			
TRAN-1	Requirement gathering & planning		
~~TRAN-2~~	Risk management	DONE	
~~TRAN-3~~	Development - phase 1	DONE	
~~TRAN-4~~	Database Design	DONE	
TRAN-5	Testing		
~~TRAN-230~~	Functionality Testing	DONE	APILA-VA
TRAN-228	Game testing	IN PROGRES:	NOUSHEE...
TRAN-229	Chat Testing	IN PROGRES:	GAB-182
TRAN-42	LAYOUT DESIGN		
~~TRAN-73~~	tables for different tabs in profile page	DONE	APILA-VA
~~TRAN-72~~	move dummy data to data folder	DONE	APILA-VA
~~TRAN-115~~	Create Search User Component	DONE	NOUSHEE...
~~TRAN-49~~	NOTIFICATION COMPONENT	DONE	NOUSHEE...
~~TRAN-47~~	HEADING WITH ICON COMPONENT	DONE	APILA-VA
~~TRAN-46~~	TABLE COMPONENT	DONE	APILA-VA
~~TRAN-39~~	LEADERBOARD , CHAT LAYOUT DESIGN	DONE	APILA-VA
~~TRAN-56~~	SETTINGS PAGE	DONE	SUMAIYA...
~~TRAN-61~~	CHANNELS FRIENDS COMPONENT	DONE	GAB-182
~~TRAN-65~~	PRIVATE, PUBLIC CHANNELS COMPONE...	DONE	GAB-182
~~TRAN-37~~	LOGIN PAGE LAYOUT DESIGN	DONE	GAB-182
~~TRAN-63~~	CHAT WITH FRIEND COMPONENT	DONE	GAB-182
~~TRAN-52~~	MESSAGES/ CHANNEL TAB COMPONENT	DONE	GAB-182
~~TRAN-62~~	LATEST MESSAGES COMPONENT	DONE	GAB-182
~~TRAN-43~~	SETTING UP COMMON STYLES AND FON'	DONE	GAB-182
~~TRAN-44~~	LOGIN PAGE	DONE	GAB-182
~~TRAN-41~~	SETTINGS / HELP LAYOUT DESIGN	DONE	SUMAIYA...
~~TRAN-40~~	PROFILE LAYOUT DESIGN	DONE	NOUSHEE...
~~TRAN-45~~	SIDEBAR COMPONENT	DONE	YONATAN...
~~TRAN-48~~	GAME STATUS COMPONENT	DONE	YONATAN...
~~TRAN-38~~	DASHBOARD LAYOUT DESIGN	DONE	YONATAN...
~~TRAN-50~~	TOP PLAYER COMPONENT	DONE	NOUSHEE...
~~TRAN-51~~	START GAME MODAL COMPONENT	DONE	YONATAN...
~~TRAN-55~~	PROFILE PAGE	DONE	NOUSHEE...
~~TRAN-57~~	HELP PAGE	DONE	SUMAIYA...
~~TRAN-74~~	group profile page and leaderboard comp..	DONE	NOUSHEE...
~~TRAN-60~~	CHAT FRIENDS COMPONENT	DONE	GAB-182
~~TRAN-64~~	CHANNEL CHAT COMPONENT	DONE	GAB-182
~~TRAN-70~~	change index.tsx to foldername.tsx	DONE	APILA-VA
~~TRAN-116~~	Create add Channel Component	DONE	NOUSHEE...
~~TRAN-165~~	REMOVE SEARCH BAR FROM DASHB.	DONE	NOUSHEE...
TRAN-148	MISSING / REWORK DESIGN...	IN PROGRES:	YONATAN...
~~TRAN-75~~	User Account Management	DONE	
~~TRAN-77~~	Set up 42 intranet OAuth with next auth	DONE	YONATAN...
~~TRAN-235~~	Add inAGame and isOnline option in	DONE	NOUSHEE...
~~TRAN-110~~	PROFILE PAGE MANAGEMENT	DONE	NOUSHEE...
~~TRAN-118~~	UPDATE AND INTEGRATE USER	DONE	SUMAIYA...
~~TRAN-174~~	UPDATE DISPLAY NAME	DONE	NOUSHEE...
~~TRAN-179~~	REDESIGN DECLINED FRIEND REQU..	DONE	NOUSHEE...
~~TRAN-180~~	ASIDE BAR TO DISPLAY NAME AND	DONE	YONATAN...
~~TRAN-181~~	AUTOMATICALLY UPDATE PROFILE	DONE	NOUSHEE...
~~TRAN-182~~	REMOVE SERVER SIDE FETCH IN	DONE	NOUSHEE...
~~TRAN-194~~	HANDLE MULTIPLE CONNECTION FRO...	DONE	APILA-VA
~~TRAN-84~~	Logout 42 intra	DONE	APILA-VA
TRAN-86	Chat Page Implementation		

Figure 2: Gantt Chart (continued...)



	AUG	SEP	OCT	NOV	DEC
Sprints	TRAN Spr.. TRAN Sprint 3	TRAN Sprint 4	TRAN Sprint 5	TRAN Sprint 6	TRAN Sprint 7
Releases					

TRAN-1 Requirement gathering & planning		
~~TRAN-2~~ Risk management	DONE	
~~TRAN-3~~ Development - phase 1	DONE	
~~TRAN-4~~ Database Design	DONE	
TRAN-5 Testing		
TRAN-42 LAYOUT DESIGN		
~~TRAN-75~~ User Account Management	DONE	
TRAN-86 Chat Page Implementation		
~~TRAN-93~~ List tasks for chat page creation	DONE	APILA-VA
~~TRAN-103~~ Join-Channel- API-endpoints	DONE	GAB-182
~~TRAN-114~~ Leave-Channel- API-endpoints	DONE	GAB-182
~~TRAN-105~~ Get-Channel- Members-API-endpoints	DONE	GAB-182
~~TRAN-102~~ create channel api	DONE	YONATAN...
~~TRAN-104~~ delete channel api	DONE	YONATAN...
TRAN-113 Tech Skill Enhancement		
~~TRAN-76~~ Learn about NestJS, Prisma, PostgreSQL,	DONE	APILA-VA
~~TRAN-31~~ Learn Next.js with Typescript and NestJS	DONE	
~~TRAN-119~~ Settings Page Implementation	DONE	
~~TRAN-220~~ Settings Page Changes	DONE	SUMAIYA...
~~TRAN-141~~ SECURTY	DONE	
~~TRAN-147~~ BACKEND TOKEN VALIDATION	DONE	APILA-VA
~~TRAN-122~~ 2 FACTOR AUTHENTICATION	DONE	APILA-VA
~~TRAN-227~~ Rate limiting	DONE	APILA-VA
TRAN-142 MESSAGING MANAGEMENT		
TRAN-136 CHAT PAGE API	IN PROGRES!	GAB-182
~~TRAN-158~~ notifications	DONE	
~~TRAN-157~~ NOTIFICATIONS MANAGEMENT	DONE	YONATAN...
~~TRAN-167~~ Docker	DONE	
~~TRAN-166~~ CONTAINERIZATION	DONE	APILA-VA
~~TRAN-169~~ Game	DONE	
~~TRAN-168~~ GAME	DONE	NOUSHEE...
~~TRAN-211~~ CUSTOMIZE GAME IN INVITATIONS	DONE	YONATAN...
~~TRAN-205~~ LeaderBoard	DONE	
~~TRAN-234~~ Protect data fetches for getAllUsersData	DONE	NOUSHEE...
~~TRAN-204~~ Leaderboard Page	DONE	NOUSHEE...
~~TRAN-208~~ Dashboard	DONE	
~~TRAN-210~~ Friend's profile page	DONE	SUMAIYA...
~~TRAN-207~~ DASHBOARD PAGE IMPLEMENTATION	DONE	GAB-182
~~TRAN-213~~ Project Documentation	DONE	
TRAN-212 Project Documentation	IN PROGRES!	NOUSHEE...

Figure 3: Gantt Chart (continued...)

3.2 Sprint Task Assignment

The following table lists the epics and tasks undertaken during each sprint. Project Management software JIRA was used to allocate and track progress during the project timeline.

Table 1: Sprint Task Allocation

SPRINT	TASKS
Sprint 1 **Start: 24-JUL-2 023** **End: 30-JUL-2023** **Epic:** **REQUIREMENT GATHERING & PLANNING**	• Setting jira & git • Setting up design platform • Design login page • Functional and Technical Requirements • Setting up PostgreSQL database
Sprint 2 **Start: 27-JUL-2023** **End: 04-AUG-2023** **Epic:** **REQUIREMENT GATHERING & PLANNING / TECH SKILL ENHANCEMENT**	• Website design- Phase 2 • Github with jira • Create NextJS Project with all pages • Learn Next.js with Typescript and NestJS
Sprint 3 **Start: 5-AUG-2023** **End: 26-AUG-2023** **Epic:** **LAYOUT DESIGN / USER ACCOUNT MANAGEMENT**	• Backend container and connect with DB • LOGIN PAGE LAYOUT DESIGN • DASHBOARD LAYOUT DESIGN • LEADERBOARD, CHAT LAYOUT DESIGN • PROFILE LAYOUT DESIGN • SETTINGS / HELP LAYOUT DESIGN • SETTING UP COMMON STYLES AND FONT • LOGIN PAGE • SIDEBAR COMPONENT • TABLE COMPONENT • HEADING WITH ICON COMPONENT • GAME STATUS COMPONENT • NOTIFICATION COMPONENT • TOP PLAYER COMPONENT • START GAME MODAL COMPONENT • MESSAGES/CHANNEL TAB COMPONENT

	• PROFILE PAGE • SETTINGS PAGE • HELP PAGE • CHAT FRIENDS COMPONENT • CHANNELS FRIENDS COMPONENT • LATEST MESSAGES COMPONENT • CHAT WITH FRIEND COMPONENT • CHANNEL CHAT COMPONENT • PRIVATE, PUBLIC CHANNELS COMPONENTS • change index.tsx to foldername.tsx • move dummy data to data folder • tables for different tabs in profile page • group profile page and leaderboard components
SPRINT 4 **Start: 27-AUG-2023** **End: 10-SEP-2023** **Epic:** **LAYOUT DESIGN / TECH SKILL ENHANCEMENT/ DATABASE DESIGN/ USER ACCOUNT MANAGEMENT / CHAT PAGE IMPLEMENTATION**	• change index.tsx to foldername.tsx • CHAT FRIENDS COMPONENT • move dummy data to data folder • tables for different tabs in profile page • group profile page and leaderboard components • CHANNEL CHAT COMPONENT • Backend container and connect with DB • Learn about NestJS, Prisma, PostgreSQL, Authentication, Models/Types • Set up 42 intranet OAuth with next auth integration • Database schema design proposal • List tasks for chat page creation • create channel API • Join-Channel-API-endpoints • delete channel API • Get-Channel-Members-API-endpoints • Leave-Channel-API-endpoints • Create Search User Component • Create add Channel Component
SPRINT 5 **Start: 16-SEP-2023** **End: 14-OCT-2023** **Epic:** **LAYOUT DESIGN / DATABASE DESIGN/ USER ACCOUNT MANAGEMENT**	• PROFILE PAGE MANAGEMENT • Logout 42 intra • ADD GAME MODEL IN PRISMA SCHEMA • UPDATE AND INTEGRATE USER SETTINGS PAGE WITH UPDATE USER API • FACTOR AUTHENTICATION • CHAT PAGE API • BACKEND TOKEN VALIDATION • MISSING / REWORK DESIGN ELEMENTS • NOTIFICATIONS MANAGEMENT • REMOVE SEARCH BAR FROM DASHBOARD AND LEADERBOARD

SPRINT 6 **Start: 15-OCT-2023** **End: 6-DEC-2023** **Epic:** **API INTEGRATION / AUTHENTICATION SETUP / USER, NOTIFICATION, CHAT, CHANNEL MANAGEMENT / SECURITY/ NOTIFICATIONS/ GAME /LEADERBOARD**	● 2 FACTOR AUTHENTICATION ● Logout 42 intra ● ADD GAME MODEL IN PRISMA SCHEMA ● UPDATE AND INTEGRATE USER SETTINGS PAGE WITH UPDATE USER API ● NOTIFICATIONS MANAGEMENT ● MISSING / REWORK DESIGN ELEMENTS ● BACKEND TOKEN VALIDATION ● CHAT PAGE API ● CONTAINERIZATION ● GAME ● UPDATE DISPLAY NAME ● REDESIGN DECLINED FRIEND REQUEST ● ASIDE BAR TO DISPLAY NAME AND IMAGE WITH API REQUEST ● AUTOMATICALLY UPDATE PROFILE PAGE ON BUTTON CLICK EVENT ● REMOVE SERVER-SIDE FETCH IN PROFILE PAGE ● HANDLE MULTIPLE CONNECTION FROM DIFFERENT BROWSERS OR TABS USING SOCKETS AND STATE MANAGEMENT ● Leaderboard Page ● DASHBOARD PAGE IMPLEMENTATION
SPRINT 7 **Start: 5-DEC-2023** **End: 22-DEC-2023** **Epic:** **RISK MANAGEMENT/ TESTING/ LAYOUT DESIGN/ USER ACCOUNT MANAGEMENT /SETTINGS PAGE IMPLEMENTATION / SECURITY/MESSAGING MANAGEMENT / DOCKER/ DASHBOARD / PROJECT DOCUMENTATION**	● CONTAINERIZATION ● HANDLE MULTIPLE CONNECTION FROM DIFFERENT BROWSERS OR TABS USING SOCKETS AND STATE MANAGEMENT ● Logout 42 intra ● DASHBOARD PAGE IMPLEMENTATION ● MISSING / REWORK DESIGN ELEMENTS ● CHAT PAGE API ● CUSTOMIZE GAME IN INVITATIONS ● Project Documentation ● Settings Page Changes ● Rate limiting ● Risk Management ● Game Testing ● Chat Testing ● Functional Testing

3.3 Risk Register

A Risk Register was maintained to identify, assess, and manage potential risks that could impact the successful completion of the project. The following table shows the steps implemented to mitigate the risks mentioned in the Risk Register.

Table 2: Risk Register

RISK	LIKELI-HOOD (1-10)	SEVERITY (1-10)	IMPACT	OWNER	MITIGATION STEPS
Website Performance	8	7	56	Apila-va	• Regularly review and optimize the codebase. • Utilize a performant WebSocket library, such as socket.io, to handle real-time communication efficiently. • Run Next js after building the project instead of running development mode
Insufficient Scalability	7	8	56	Apila-va	• Optimize your PostgreSQL database by ensuring proper database schema design • Use WebSocket library (e.g., Socket.io) that supports scalability to ensure concurrent connections • Choose a game engine (Phaser) designed to run on the client side thereby reducing the load on the server. This is crucial for scalability, especially for many concurrent users. • Choose Next.js as the frontend framework. This supports client-side navigation (allowing faster transitions between pages), and breaking code into smaller components.
Security Breach	8	10	80	Apila-va	• Keep all libraries, frameworks, and dependencies up to date to ensure that you benefit from security patches and updates. • Implement robust user authentication and authorization mechanisms. Use secure authentication protocols like OAuth or JWT (http only cookie)

					• Sanitize and validate all user inputs to prevent common vulnerabilities like SQL injection • Cross-Origin Resource Sharing (CORS) for requests from another domain
Compatibility Issues	5	7	35	Apila-va	• Verify that the features in frontend code especially game engine is supported in target browsers • Ensure that communication between different services (Next.js frontend, Nest.js backend, WebSocket server) is well-defined. • WebSocket library is compatible with both the server-side (Nest.js) and client-side (Next.js, Phaser) implementations
Knowledge Gap	3	6	18	Apila-va	• Comprehensive analysis of project requirements to identify skills needed in each phase. • Regular team meetings to discuss progress and challenges. • Share useful resources • Conduct code reviews.
Poor Code Quality	8	9	72	Apila-va	• Follow best practices and coding standards • Include exception handling • Keep code modular and maintainable. • Use version control systems such as GIT to track changes and collaborate. Conduct multiple code reviews before merging the code to the main branch
Team resource not available	6	8	48	nali	• Adopt an agile methodology with short development cycles(sprints) • Distribute tasks to avoid dependency on a single member. • Use project management system (JIRA) and version control system (GIT) to ensure project data is accessible to all members • Maintain clear communication
Inadequate Communication	4	6	24	nali	• Schedule regular meetings to communicate tasks and challenges

					• Maintain a project management system such as JIRA to allocate tasks and track progress. • Use version control systems such as GIT to track changes and collaborate. • Define the roles and responsibilities of each member
Unpredictable Risks	9	9	81	nali	• Allocate buffer resources(time) when project planning to accommodate project delays. • Conduct thorough risk identification and maintain regular communication. • Multiple users testing and reviewing the code

Based on their score, high, medium, and low-impact risks are:

High Risks (Impact Score > 60):

- Security Breach (80)
- Unpredictable Risks (81)
- Poor Code Quality (72)

Medium Risks (40 < Impact Score ≤ 60):

- Insufficient Scalability (56)
- Team Resource Not Available (48)
- Website Performance (56)

Low Risks (Impact Score ≤ 40):

- Compatibility Issues (35)
- Knowledge Gap (18)
- Inadequate Communication (24)

3.4 Functional Requirements

The project sets out specific requirements that demand some essential functional requirements that the system should offer. All these functionalities ensure the end user performs operations and navigates through the application with ease. These cover a range of operations including user registration, logging in and logging out of the application, real-time chatting, and a live ping-pong game with a matchmaking system and customization features.

Security concerns/Management:

- **User authentication feature:** The project requires user authentication using the 42 intranet OAuth system for enhanced security ensuring secure password/credential storage.
- **Enhanced security feature:** The platform should offer(offers) a two-factor authentication feature, preventing malicious activities.
- **System termination:** Implement a secure session termination mechanism to ensure user data and sessions are closed properly and enhance security by preventing unauthorized access.

User Account Management:

- **User profile customization:** Enable users to personalize their profile, allowing them to customize their avatar and name of choice and permitting other users to access this personalized profile.
- **Real-time interaction:** Users should be able to add other users as friends and view their real-time status and stats.
- **Performance metric:** Enhance user engagement, the application provides a detailed gaming journey feature including wins, losses, and achievements of all the games played.

Chat System:

- **Direct messaging:** The project allows users to enjoy direct messaging capabilities along with the ability to block other users for a personalized chatting experience.
- **Channel creation:** Allow users to initiate a variety of chatting experiences by providing them the ability to create public, private, and password-protected chat channels.
- **Channel administration:** Empower channel owners to manage the spaces they create by setting and changing the channel's password along with the ability to ban, kick, and mute specific users.
- **In-chat game invitation:** Users should be able to extend gaming invitations through the chat interface, enhancing the social gaming experience.

- **User profile insight:** Enable users to view the detailed profiles of their friends through the chat interface.

Game System:

- **Live gaming experience:** The platform should allow users to play a live game of pong against another player on the website.
- **Match-making system:** Enhance user interaction by allowing an automatic match-making gaming system with the users waiting in the queue.
- **Customized gaming experience:** The users should be able to customize their game along with the option of a default gaming version.
- **Optimized performance:** The game must be designed to be responsive and lag-free for the best user experience.

3.5 Technical Requirements

To establish peak performance a set of technical requirements were decided for the smooth operation of the Spin Masters Website. These made the building blocks for a stronger and smarter application. After careful comparison among various technologies, the following were finalized. The front end is designed for a user-friendly experience using Next.js, whereas the back end is NestJS, providing a well-structured and efficient server-side implementation. The database selected is PostgreSQL as per the project requirement. When preparing to deploy, Docker is used to make the process flexible and easy to manage. Each of these technical requirements works together to make the Spin Masters project a success story.

Frontend Framework:
The project gave the flexibility to select the frontend framework, provided it was based on Typescript. With server-side rendering (SSR), static site generation (SSG), dynamic routing, and seamless integration features, Next.js was the ideal framework for the application. Next.js allows a better user experience by enhancing page responsiveness. The possibility of seamless integration of web sockets and the phaser game engine into this framework was an added advantage. Next.js suitability to create single-page applications (SPA), another mandatory requirement for the project, was the final deciding factor.

Backend Framework:
The technical requisites for the application include the website's backend being written in NestJS. The Nest JS framework is known for its scalability and efficiency. The opinionated architecture, combined with the industry's best practices, makes it easy for developers to write clean and maintainable code. Additionally, NestJS supports authentication,

authorization, encryption, and hashing, and is easy to integrate with other libraries and technologies.

Databases:
PostgreSQL is a powerful, open-source object-relational database system that has earned it a strong reputation for reliability, feature robustness, and performance. The choice of this database system was also a technical requisite for this project. PostgreSQL comes with many features aimed to help us build applications to protect data integrity and build fault-tolerant database environments.

Containerization:
The application is required to be containerized. This is done using Docker, a popular containerization platform that provides the ability to package and run our application in a loosely isolated and lightweight environment. This enables portability, scalability, and manageability. The continuous integration and continuous delivery (CI/CD) workflow made it easy for us to work.

Game Engine:
Phaser is a popular JavaScript framework specifically designed for game development. One of the key features that make Phaser stand out is its ease of use, flexibility, and ease of integration with Next.js since it is primarily written in JavaScript. The game engine's active developer community and vast reference materials made it an obvious choice.

WebSocket Technology:
The need for real-time interactions and updates discussed in the functional requirements was met using web sockets. WebSocket is a duplex protocol used mainly in the client-server communication channel. It is bidirectional in nature which means it facilitates two-way communication between the client and server, thus enabling features like real-time chat and multiplayer gaming environments.

3.6 Wireframes

Wireframes are visual representations or skeletal outlines of a web page or application, illustrating the basic structure, layout, and placement of elements. Using Figma design software a detailed prototype of Spin Masters web application was created to aid the subsequent development process. Attached here is the link to the project Figma page.

[FIGMA LINK]()

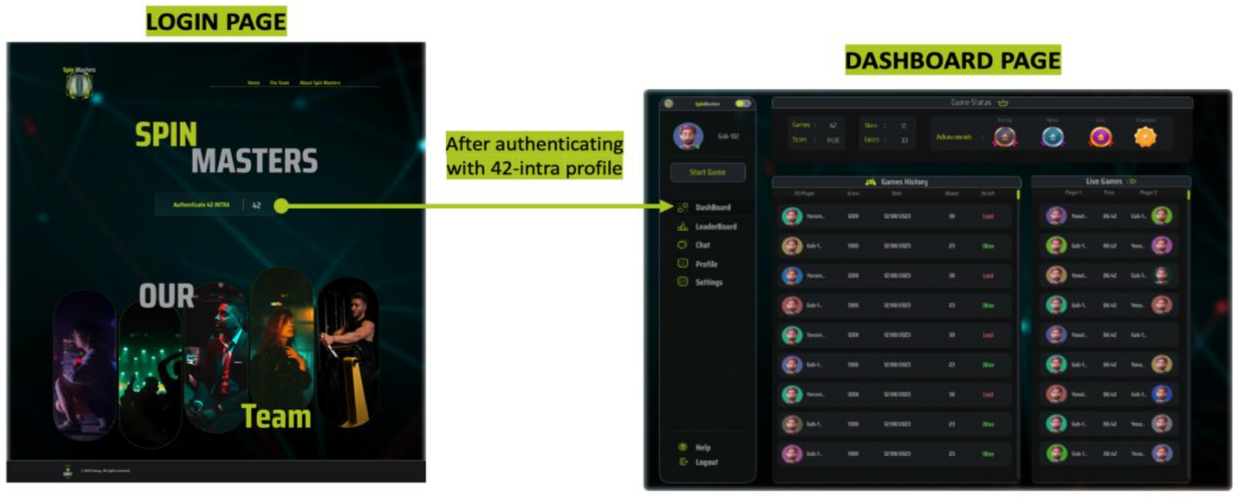


Figure 4: Login Page

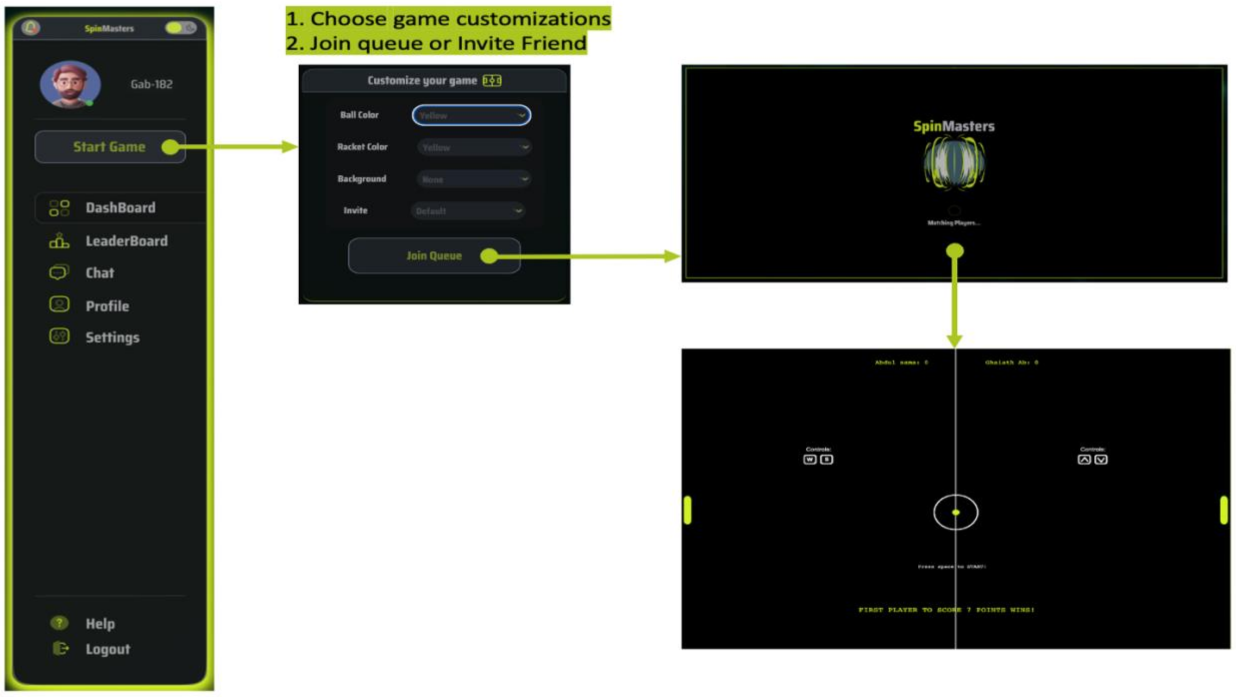


Figure 5: Game Page

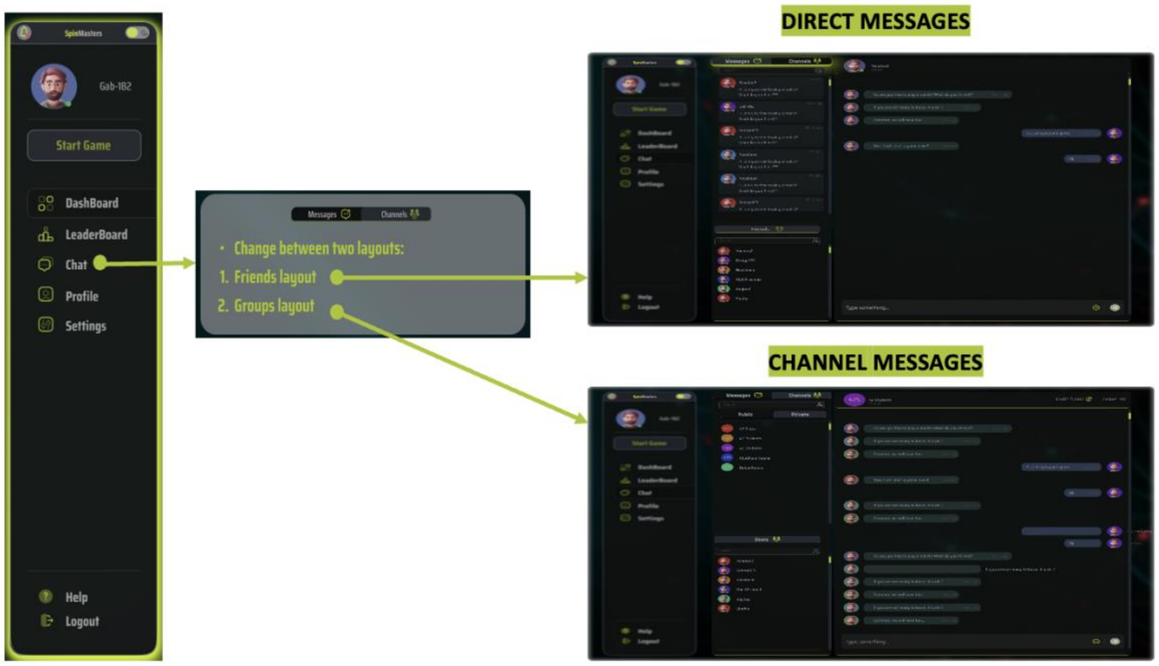


Figure 6: Chat Page

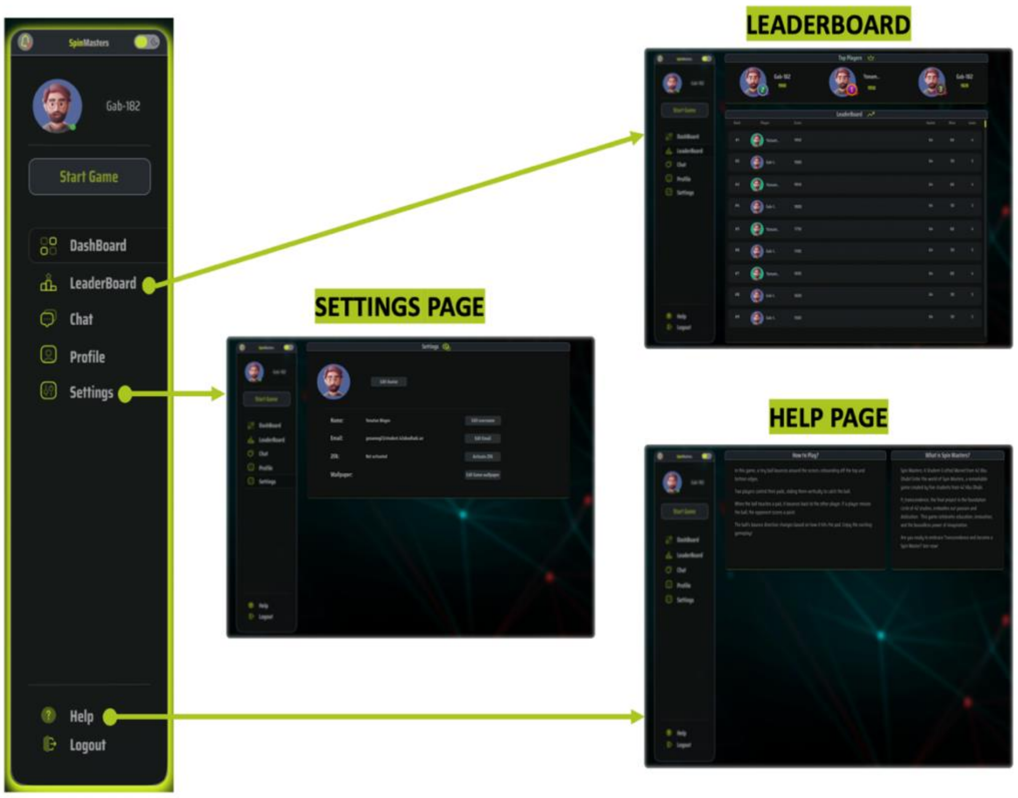


Figure 7: Leaderboard, Settings Page, Help Page

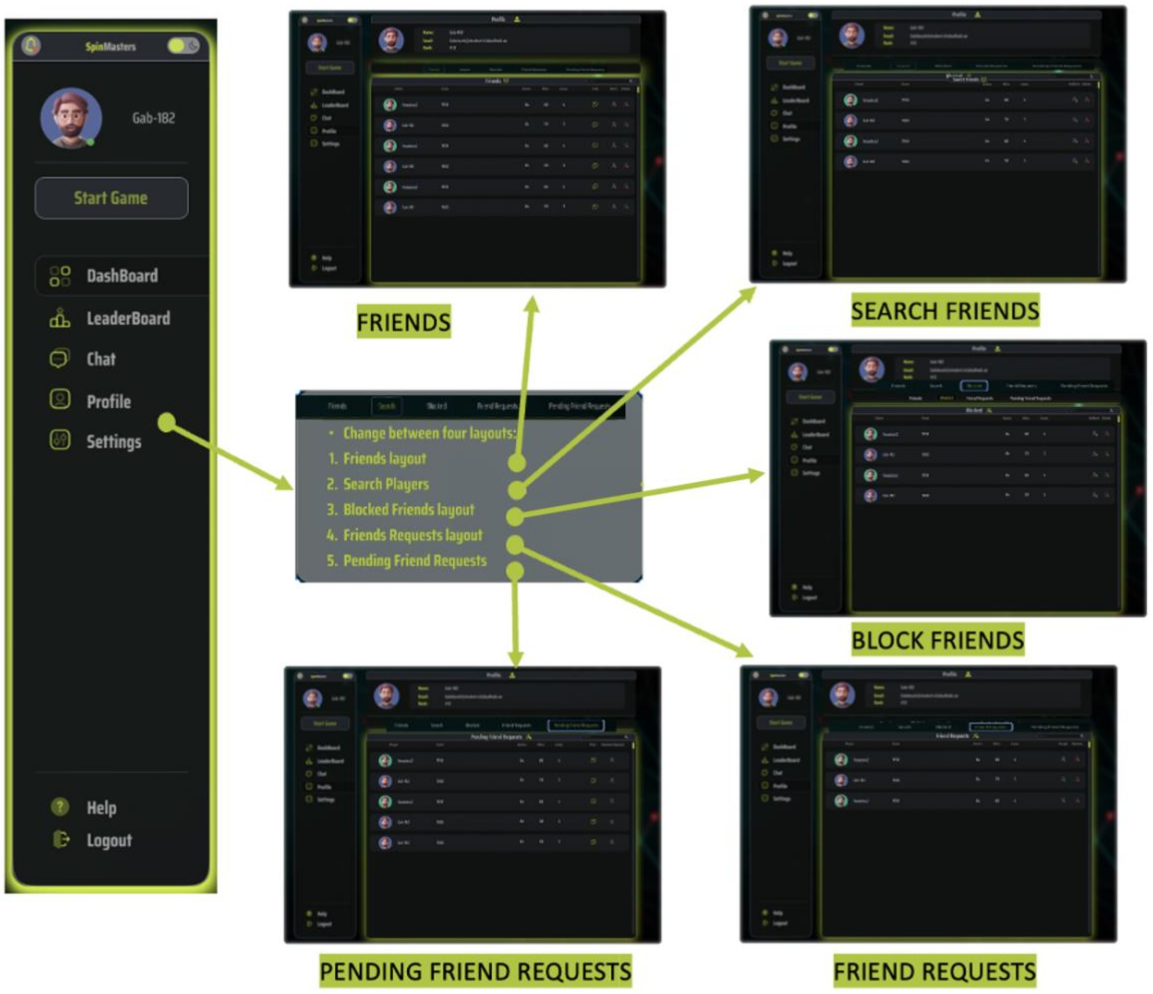


Figure 8: Profile Page

3.7 Flowcharts

The following flowchart illustrates the process flow for the entire Spin Masters application.

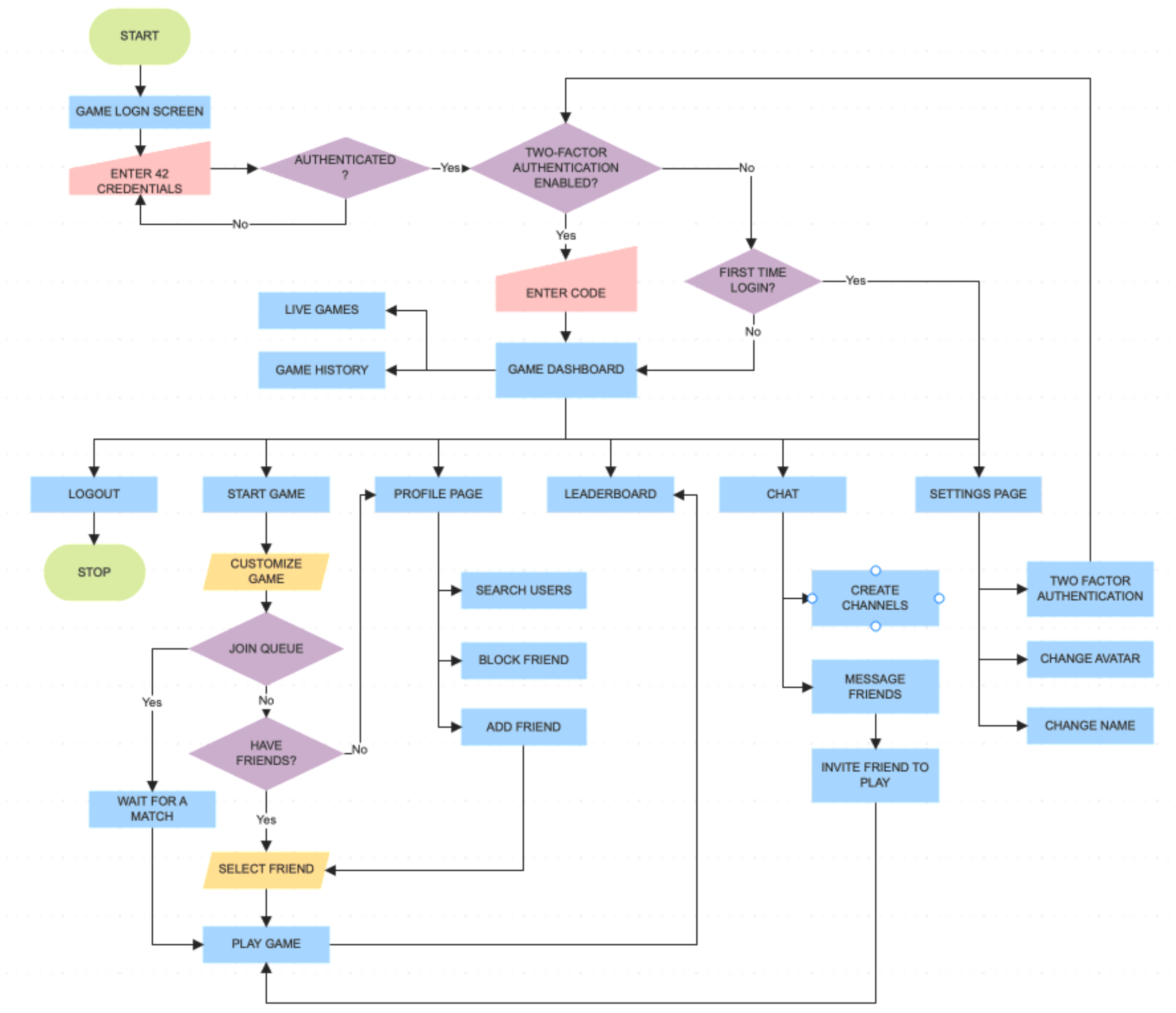


Figure 9: Spin Masters Application Process Flow

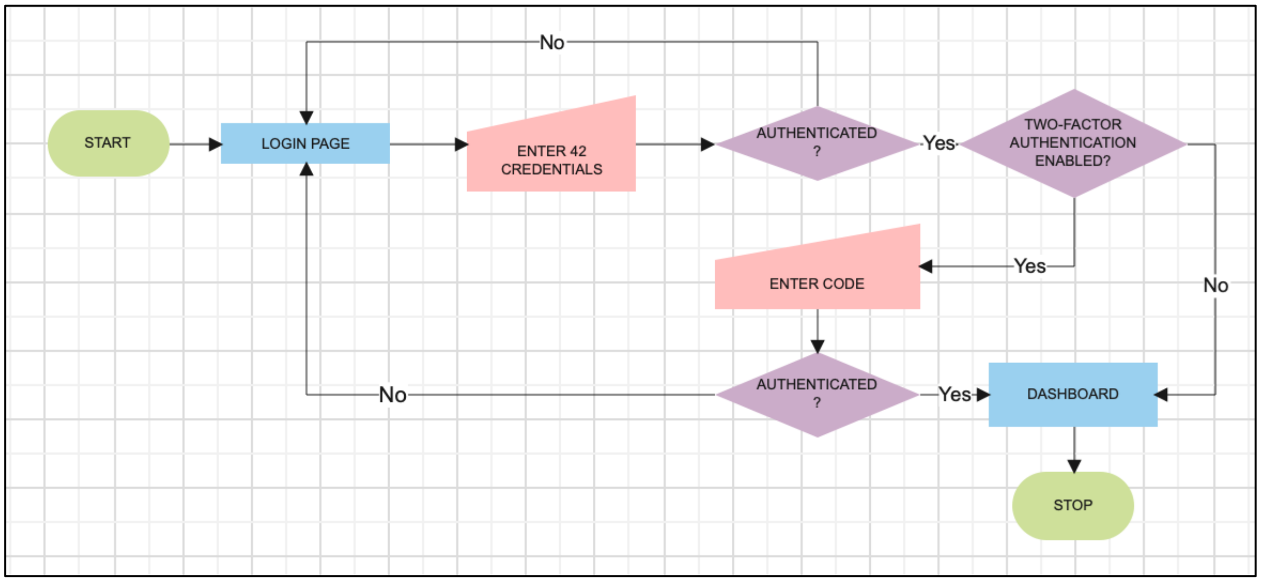


Figure 10: Login Page Process Flow

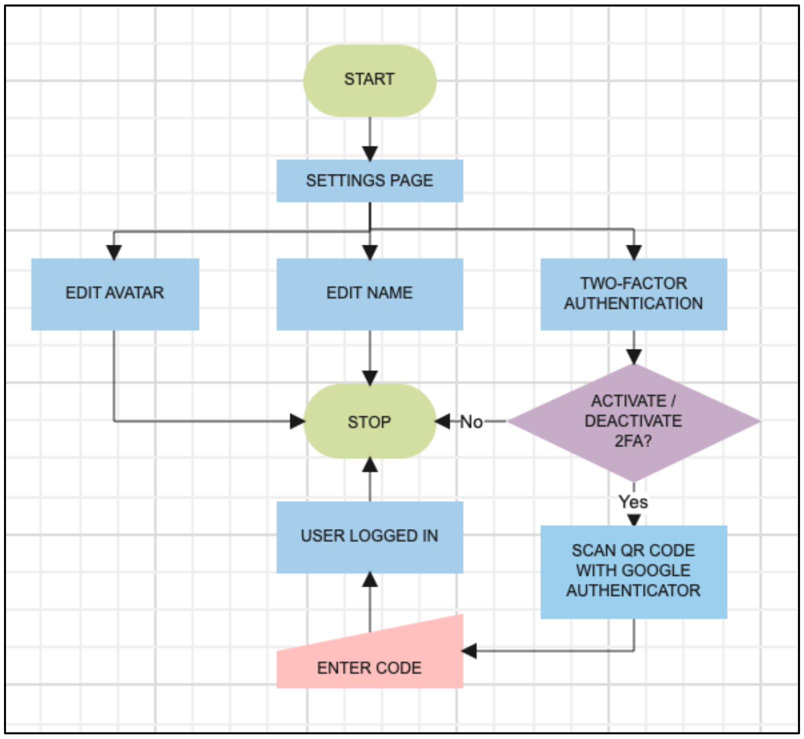

START
SETTINGS PAGE
EDIT AVATAR
EDIT NAME
TWO-FACTOR AUTHENTICATION
STOP
No
ACTIVATE / DEACTIVATE 2FA?
Yes
USER LOGGED IN
SCAN QR CODE WITH GOOGLE AUTHENTICATOR
ENTER CODE

Figure 11: Settings Page Process Flow

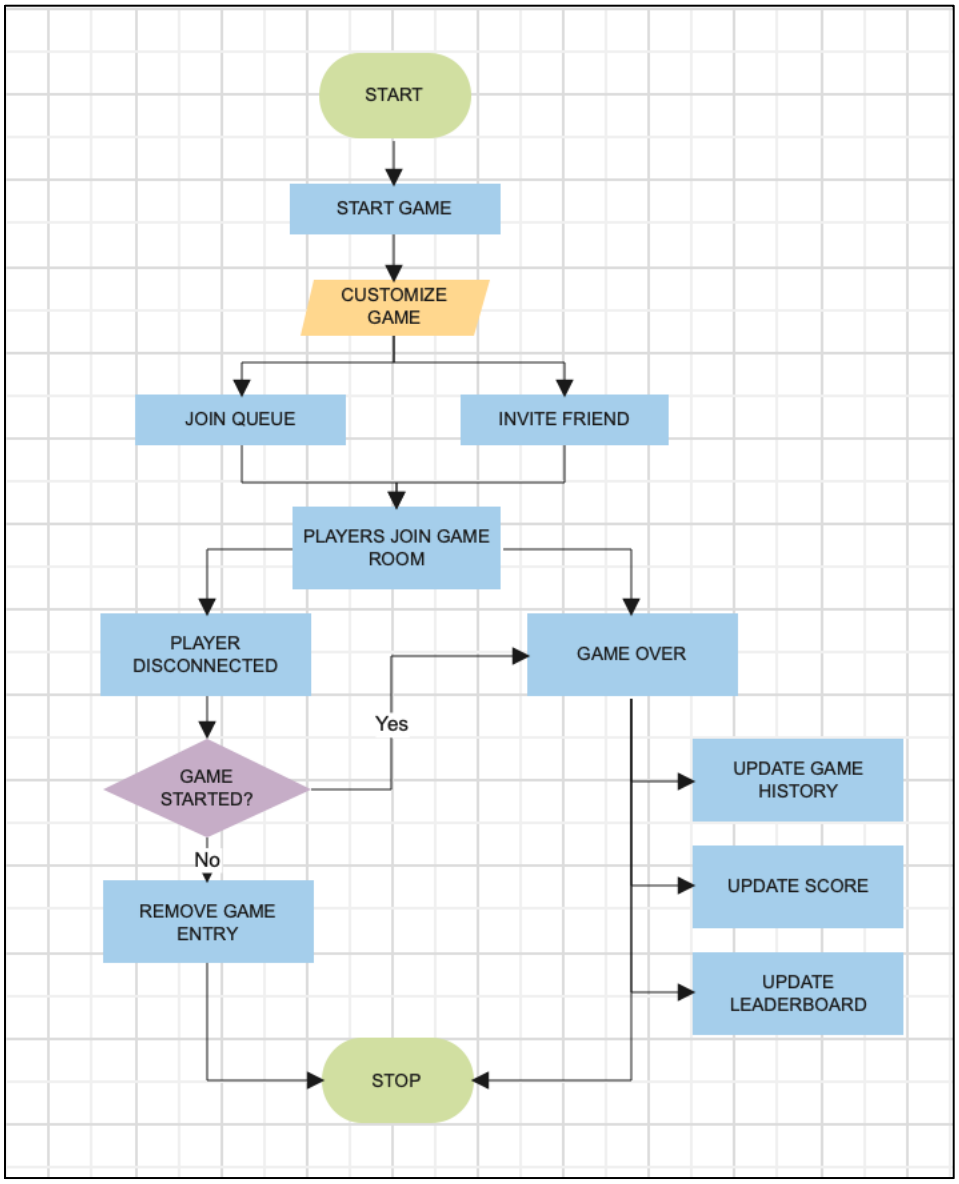


Figure 12: Game Process Flow

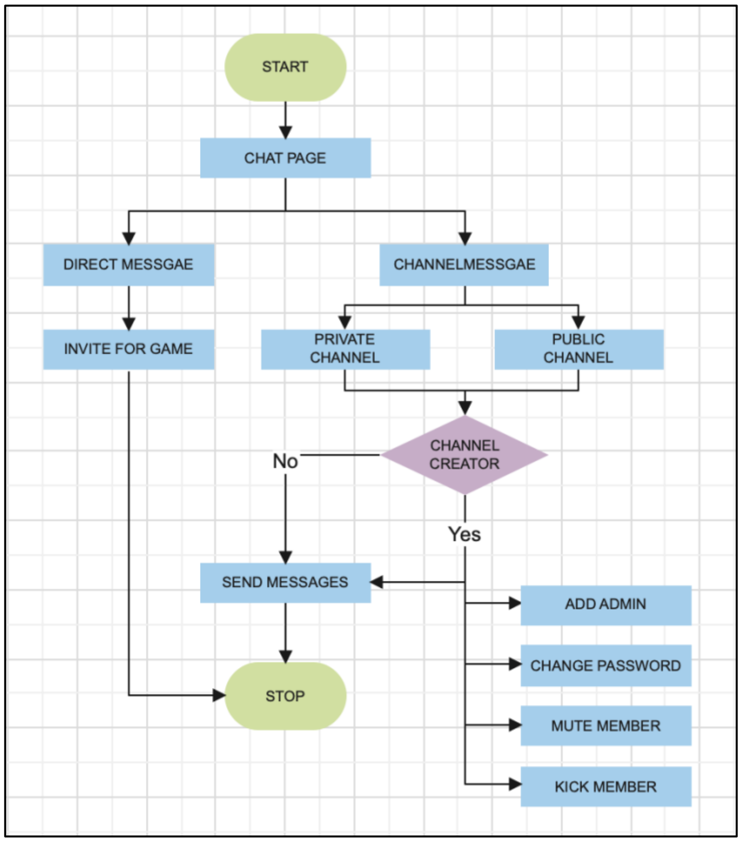


Figure 13: Chat Page Process Flow

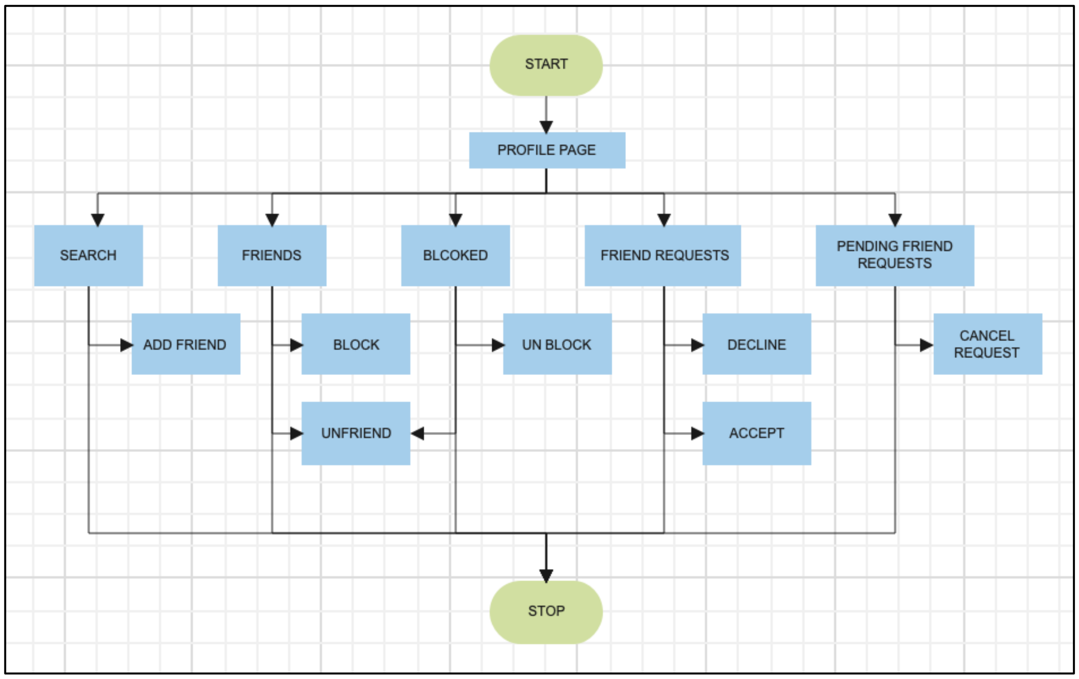


Figure 14: Profile Page Process Flow

4 Conclusion

The journey to creating a web application that seamlessly integrates an efficient friend management system, an engaging chat feature, and an entertaining multiplayer pong game was the result of innovation, dedication, collaboration, and technical expertise. The objective of the project was not only achieved but exceeded by the combined use of technologies such as Next.js, NestJS, Prisma, Web sockets, and Phaser to create a feature-rich and dynamic application.

The project took four months to complete, thanks to the meticulous planning during the initial phase of the project. The use of Figma design software aided in creating a detailed prototype of the web application, making the subsequent processes easier. The project Management software JIRA helped the team effectively in task allocation and progress tracking. The Version control system, GitHub, provided ease in code review and collaboration. Weekly team meetings and knowledge sharing ensured a synchronized and organized development process.

As we come to the end of the project, it is important to acknowledge that a strong foundation has been established by the technology building blocks of this application. Giving the possibility of scalability and enhancements in the future.

In summary, the project's successful completion is evidence of the balanced combination of cutting-edge technology, careful planning, and teamwork. The skills gained during this journey, both technical and non-technical, will surely impact our future endeavors positively.

5 Appendices

5.1 Website Prototype

LINK TO FIGMA PROTOTYPE PAGE

5.2 Database Schema

The following Image is the Database schema for the Spin Masters application illustrating the organization of database tables and their relationships. It was designed with a focus on optimizing data retrieval and enhancing database performance.

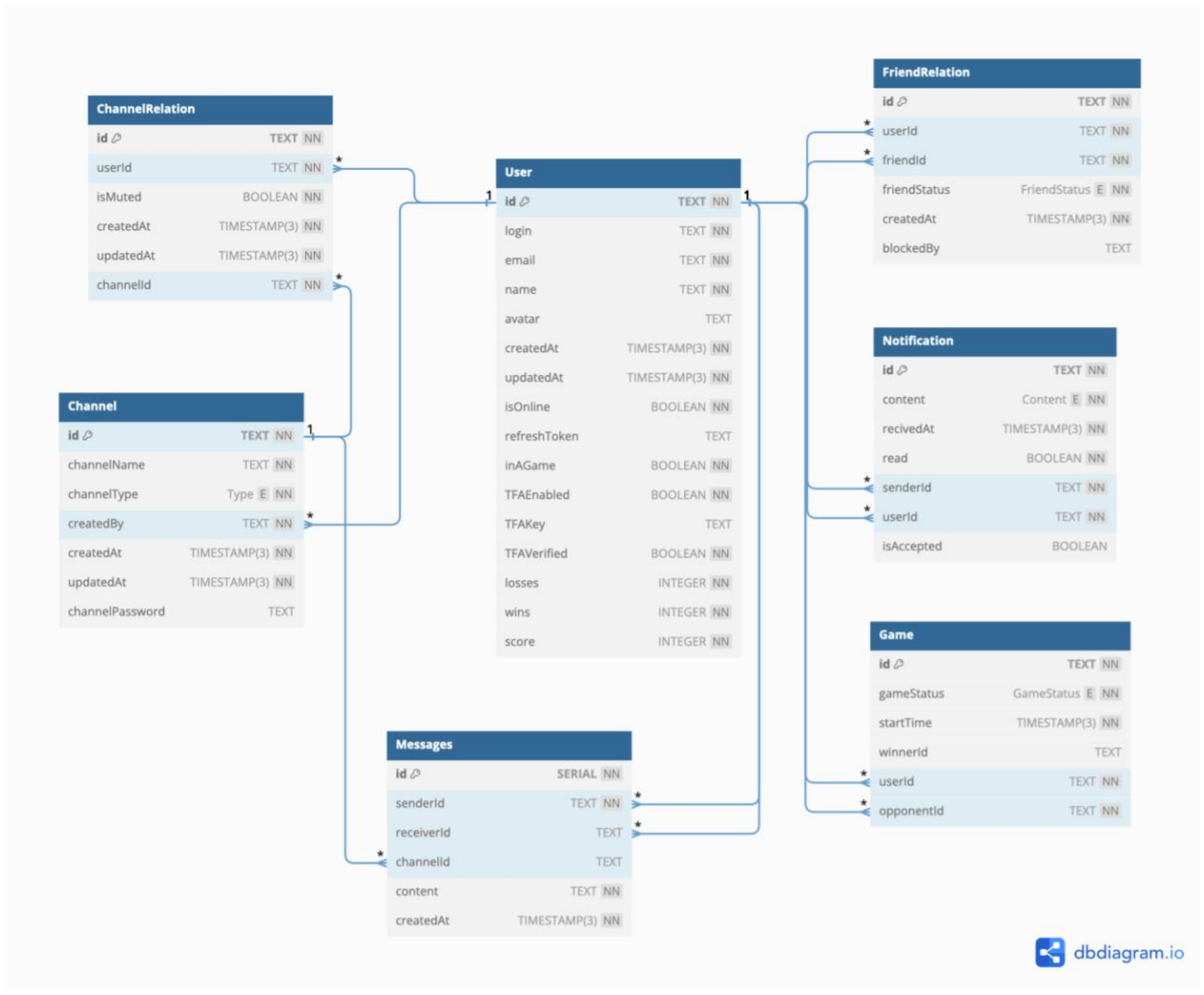


Figure 15: Spin Masters Database Schema

Link to Database Schema

6 References

Agaev, V. (2022, August 18). *How to use Prisma in NestJS*. Www.Codewithvlad.com/. https://www.codewithvlad.com/blog/how-to-use-prisma-in-nestjs

[Dave Gray]. (2023, May 19). *Next.Js Full Course for Beginners | Nextjs 13 Tutorial | 7 Hours*[Video]. Www.Youtube.com. https://www.youtube.com/watch?v=843nec-IvW0

Davey, R. (2018, February 20). *Making your first Phaser 3 game*. Phaser.io. https://phaser.io/tutorials/making-your-first-phaser-3-game/part1

[FreeCodeCamp.org]. (2022, February 22). *NestJs Course for Beginners - Create a REST API* [Video]. Www.Youtube.com. https://www.youtube.com/watch?v=GHTA143_b-s&t=5728s

Ishmam, T. (2023, March 23). *Building a REST API with NestJS and Prisma: Handling Relational Data*. Www.Prisma.io. https://www.prisma.io/blog/nestjs-prisma-relational-data-7D056s1kOabc#add-a-user-model-to-the-database

[LogRocket]. (2020, June 9). How to build pong with Phaser3[Video]. Www.Youtube.com. https://www.youtube.com/watch?app=desktop&v=itXXERREvx8

M. W. (2023, May 23). *Building Real-time Applications with Nest.Js and WebSockets*. Www.Medium.com. https://medium.com/@waqarilyas/building-real-time-applications-with-nest-js-and-websockets-2a7114dc4544

[Marius Espejo]. (2022, April 3). *Easiest way to build real-time web apps? WEBSOCKETS with NestJS* [Video]. Www.Youtube.com. https://www.youtube.com/watch?v=atbdpX4CViM

n.d. (2020, February 9). NestJS For Absolute Beginners. Www.Codingthesmartway.com. https://www.codingthesmartway.com/nestjs-for-absolute-beginners/

[Stewart Gauld]. *HOW TO USE JIRA | Free Agile Project Management Software (Jira tutorial for Beginners)* [Video]. Www.Youtube.com. https://www.youtube.com/watch?v=GWxMTvRGIpc

[Yugen]. (2022, August 22). Building A Game with Phaser JS and NextJS [Video]. Www.Youtube.com. https://www.youtube.com/watch?v=xRJ787usR5s

7 Glossary

TERM	DEFINITION
Jira	A project management and issue tracking tool that helps teams plan, track, and manage their work efficiently.
Figma	A collaborative cloud-based design tool that allows users to create, share, and prototype digital designs in real-time.
Agile	An iterative and incremental approach to software development that prioritizes continuous collaboration and improvement. Teams follow a cycle of planning, executing, and evaluating.
Epic	In Agile terminology, an epic is a large, high-level user story that represents a significant piece of work, often broken down into smaller, more manageable tasks.
Sprint	In Agile terminology, a sprint is a time-boxed iteration, typically lasting two to four weeks, during which a development team works to complete a set amount of work from the product backlog.