RamBluff: Online 21

Group 9
Sean Berlin, Kirtan Chavda, Jared Colletti, Zachary Leopold

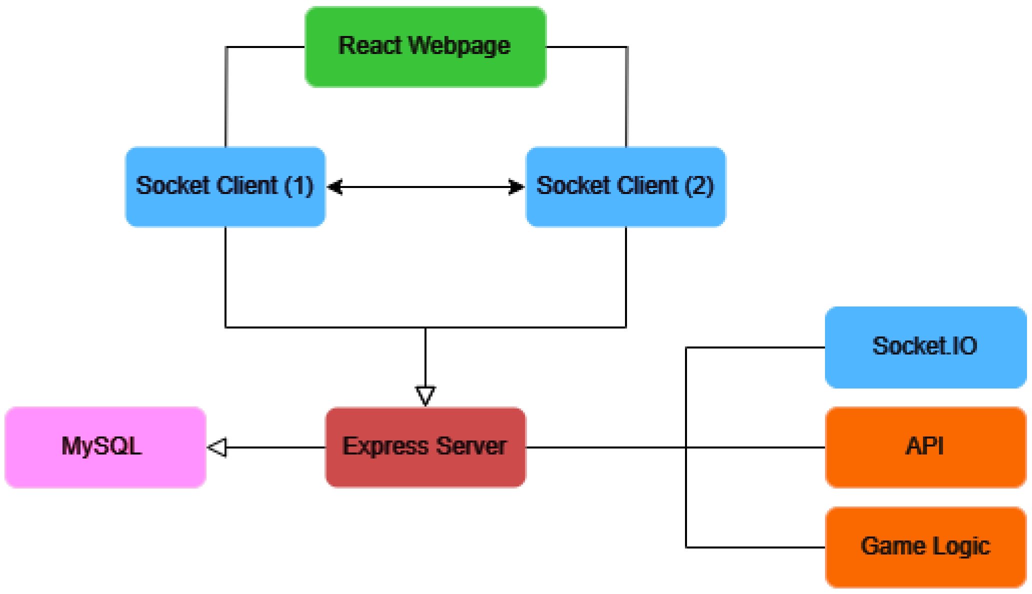
Chapter 1: Vision

Introduction

RamBluff is an online poker web application that delivers a seamless and immersive gaming experience to users. Our platform will utilize cutting-edge technologies to create a dynamic environment where players can engage in poker games with ease and convenience. The games played within our application will contain a user-friendly interface while securely managing player and gamestate data, and comply with the Texas Hold'em ruleset.

RamBluff: Online Poker

Figure 1



Chapter 2: Technical Proposal

2.0 Architecture Overview

Our proposed architecture consists of three main components: the frontend UI developed with React, the backend server powered by Node.js, and the database management system using MySQL.

2.1 Frontend Development (React)

We will adopt a component-based architecture in React to facilitate code reuse, improve maintainability, and enhance development efficiency. With the utilization of CSS, we will ensure that the UI is responsive and optimized for all screen sizes to provide a consistent user experience. Along with that, we will be employing React state / useEffect, socketClient, and Axios to orchestrate the synchronization with our backend services. Our users will be presented with a homepage where they will be prompted to create a game instance. To invite others to play, the user will be provided with an invite link to share.

2.2 Backend Development (Node.js)

Our team is designing a RESTful API to handle client-server communication, allowing for seamless interaction between the frontend and backend components. In order to enable real-time communication between players and the server, we will be using Socket.IO; this will allow live game updates and possible chat functionalities. We plan to use middleware libraries, such as Express, to define our routes for the necessary HTTP methods (GET,POST,PUT,DELETE,etc.)

The functionality of the poker game will be powered by the ‘Table’, ‘Dealer’, and ‘Player’ classes. Where ‘Table’ will be written in Node in order to make use of an open-source API, made by “goldfire”, that will handle player hand rankings. Along with that, ‘Table’ will manage the pot, the rotation of the blinds, and current players. Adversely, the ‘Dealer’ and ‘Player’ classes will be written in Python. ‘Dealer’ will handle the shuffling and dealing of cards (Both player and community cards), while ‘Player’ is utilized for player actions and states.

2.3 Database Management (MySQL)

We will design a relational database schema using MySQL to store game states and player profiles(along with any other relevant data). After a player completes a successful sit down at a table, they will be given a UserID that corresponds to the seat of the particular table at which they are sitting. This will allow us to maintain which users are in which games. Lastly, to ensure efficient data retrieval and storage, we will be using proper normalization and indexing techniques.

2.4 Conclusion

By implementing these technical strategies, we aim to develop a robust and scalable online poker platform that delivers a superior gaming experience to our users while maintaining the highest standards of security, reliability, and entertainment.

Chapter 3: Intermediate Milestones

3.0 Overview

The development of RamBluff has dealt its share of progress and challenges (specifically, the implementation of Socket.io). When attempting to leverage the library for real-time communication via gamestate updates, it proved difficult to correctly set up the Socket.io rooms. These rooms are what enable us to host multiple running poker games. The logic of creating our player objects and associating them with particular tables had to be tweaked multiple times. However, we have solved this issue. By utilizing a hashmap in our back end, where the Table ID is the key and the list of player objects is the data. When a user performs an action, such as sitting down, our frontend sends an emission with the Table ID to the backend that, in this example, creates a new player object such that it is only associated with that table (socket room). This allows for the multiple games we previously mentioned.

Currently our users are able to sit down and leave unique tables, which is visualized on all clients associated with that table. Our team is working on the implementation of the game logic and the cloud deployment. Where, the game logic is being implemented and tested locally, but the cloud deployment is our main focus. With the structure for our socket rooms in place, our game logic (betting, calling, checking, folding) will be easily tied to each individual user at each unique table. As it stands, our team is fully confident in the completion of RamBluff by the end of the semester.

In regards to the cloud deployment, we will be utilizing Docker and its image functionality. RamBluff requires the construction of a Docker image for each of the three main components: the React frontend application, the Node.js server, and the MySQL database. Where all require a systematic approach to ensure efficiency, security, and scalability. The following sections detail the procedures for building these Docker images, along with insights on preliminary testing and data management strategies.

3.1 React App (Frontend)

To create our Dockerfile for the React component, we will be hosting our Dockerfile in the frontend’s root directory. This file will use the official Node image, specifically node:alpine, for the build stage. This stage will handle installing our dependencies and building the static files for the React application. For the serving stage, we will be switching to nginx:alpine, a lightweight server, to host the built application. We are ensuring that the nginx configuration is optimized for serving a single-page application (SPA) and can handle our routing correctly. Once the Dockerfile is configured the docker build command is used to create the image. This image will be tagged appropriately for future use with Kubernetes and will also be stored on DockerHub. For preliminary testing purposes, the container is ran locally to confirm that the React app is served correctly and connects to our backend API and Socket.io without any issues.

3.2 Node.js Server (Backend)

Our backend Dockerfile will also begin with a base image of node:alpine. It will include copying RamBluff's source code into the container and installing the necessary dependencies. The configuration details, such as database connection strings, are managed using environment variables, which can be defined in our Kubernetes deployment configurations. The docker build command will be used to build our image with the appropriate tags and to ensure the Dockerfile specifies the command to run the Node.js server. For preliminary testing purposes, the container is ran locally to confirm that our API and Socket.io endpoints are accessible and operating as intended.

3.3 MySQL (Database)

The database setup involves using the official MySQL Docker image, eliminating the need for a custom Dockerfile. Configuration tasks, such as setting up our database schemas are achieved by mounting SQL scripts into the Docker container at /docker-entrypoint-initdb.d.Key environment variables like MYSQL_DATABASE, MYSQL_USER, and MYSQL_PASSWORD are utilized to configure the database instance. When building our image we are preparing the Docker Compose file instead of creating a new image, given the direct use of the official MySQL image. During testing we are ensuring the accessibility of the database from the Node.js backend container and are conducting tests to confirm smooth execution of initial data loading and schema creation scripts.

3.4 Data Collection/Creation and Tables Management

For RamBluff, data management involves handling user profiles, game states, and other relevant information within the MySQL database. Proper schema design is critical to support efficient data access and updates during gameplay. Our system does not require a user to sign-up. Therefore, we will never be storing sensitive information such as emails and passwords. Our schemas consist of two tables; Players and Games. Players will hold the name, stack size, cards, and the table ID at which they are seated. Games will hold the status (active / inactive), the gamestate, the time created, and the time of the last update. The gamestate will utilize the players table to access the players seat, stack size, etc along with the current round of betting such that when a new player joins they will be able to see the current game state. The schema designs are made in a way to minimize redundancy and optimize query performance. We are using the best practices of normalization while also considering practical denormalization for frequently accessed data. In our testing, we ensure that the database properly updates the gamestate information with the use of SQL scripts to populate our tables with initial data.

3.5 Conclusion

We are making headway on our project and are tackling each issue as it comes. By carefully implementing our socket emissions and constructing Docker images for each component of RamBluff, we ensure a scalable, secure, and manageable deployment process. While we are in our preliminary testing phase it is crucial to identify and resolve any integration issues early in our deployment cycle. Through thoughtful code and data schema design and management, RamBluff is positioned to offer a robust and engaging gaming experience.

Chapter 4: Final Results

4.0 Overview

The original idea for RamBluff was an online multiplayer poker game that offered features such as a leaderboard, user authentication, and a more enhanced UI. Throughout the development process, RamBluff transitioned away from poker and instead offered the game 21. RamBluff is made of 3 components : [Frontend, Backend, Database] utilizing React, Node.js and mysql. RamBluff offers users a real time gaming experience using Socket.IO, which allows users to create multiple games and invite additional members simply by sharing a link.

4.1 Frontend

The React application architecture encompasses two distinct components: a homepage and a game page, each dynamically rendered through React Router. Facilitating seamless API communication, Axios (a renowned library for executing asynchronous HTTP requests to REST endpoints) is employed. SocketIO-client is leveraged to establish and maintain a WebSocket connection to the backend, initializing upon the user's initial access to the homepage. Notably, the homepage hosts a POST request mechanism for game creation.

For the game page, a significant portion of its content is generated dynamically and in real-time, orchestrated through state objects, useEffect hooks, and synchronized with socket events. This sophisticated setup ensures responsive and synchronized interactions, elevating the application's user experience.

The user interface (UI) is crafted with CSS styling techniques to optimize visual presentation. Each page is structured for clarity and navigational efficiency. Furthermore, a carefully curated casino-themed color palette is employed to enhance the aesthetic appeal and coherence of the interface. Our frontend has met all of our technical proposal requirements.

4.1 Challenges

Establishing event listeners and emitting events between the client and server marked only the initial phase of providing a real-time gaming experience. Upon successful establishment of the client-server connection, management of state variables within the React page became imperative. Tracking the state of each value on the React page and ensuring synchronized updates demanded intricate handling of data flow and real-time synchronization mechanisms.

4.2 Backend

In the backend architecture, Node.js serves as the foundational framework for executing a myriad of crucial services. It orchestrates the server instance, overseeing its configuration and management to ensure optimal performance and reliability. Additionally, Node.js facilitates seamless handling of API requests, leveraging its asynchronous nature to efficiently process incoming and outgoing data flows.

Furthermore, Node.js plays a pivotal role in facilitating Socket.io communication, pivotal for enabling real-time bidirectional interactions between clients and the backend infrastructure. This critical functionality synergizes seamlessly with the game page component of the React application, ensuring the seamless delivery of live updates to users engaged in gameplay. Upon the creation of a game instance, the unique identifier is created, added to the database, and then is used as the designated socket room identifier. When a client joins a game session, their corresponding socket instance is dynamically added to the pertinent socket room. Leveraging socket rooms empowers RamBluff to concurrently host and manage multiple game instances. Both the server and the backend are constantly listening for emissions from each other and essentially ping-pong between themselves. When a user sits down, the back end receives that, applies the necessary storage, then emits an event back to the frontend where it will be handled. In the case of sitting down, the front end would update the players at the table.

Beyond communication protocols,the backend is also responsible for MySQL management, facilitating database connection and information retrieval .

Lastly, the backend houses intricate logic dictating the operations of the game 21. The implementation is architected around distinct player and dealer classes, each meticulously designed to encapsulate the core functionalities of their respective roles within the game. Every instantiated player and dealer entity is associated with a designated socket room to facilitate streamlined communication and synchronization between the backend and connected clients. These classes encapsulate a plethora of crucial procedures pivotal for gameplay, ranging from card dealing and player transitions to betting actions, and beyond.

Our backend has met our goals within the technical proposal, with slight shifts. We originally thought we would have to have a separate class for dealer and table, but we were able to combine the two into one. Along with that, all of the logic regarding win condition (hand rankings) and player actions has shifted from poker to that of 21(blackjack).

4.2 Challenges

The biggest challenge with the backend was implementing the logic for poker. Poker presents a nuanced and intricate landscape in its gameplay. While the fundamental concept of the game appears straightforward, its intricacies encompass a plethora of edge cases, including but not limited to kicker pots, player actions such as betting, folding, calling, raising, among others. The complexity introduced by these multifaceted scenarios exceeded the scope designated within the timeframe allocated for this project. As a result, our team decided to implement a simpler game, 21.

4.3 Database

The database architecture is characterized by its simplicity and efficiency. Within its schema, game entities are stored alongside essential metadata such as timestamps denoting creation and last updates. As we originally planned, we are storing player objects and associating them with their game. However, these stored players are not being used and have been overtaken by a hashmap implementation.

4.4 Deployment

Every component has been containerized, each with its own Dockerfile. Each container is running and the docker images are updated from the main branch of the GitHub repository. There were no complications regarding creating each of the three Docker images. All three Dockerfiles follow a similar structure, starting with an official base image tailored to their specific requirements. The first Dockerfile targets a Node.js runtime for a React application, setting up the working directory, installing dependencies, copying application code, and serving the built React app using a lightweight Node.js server. The second Dockerfile also utilizes a Node.js base image, focusing on backend functionality, while the third Dockerfile starts with a MySQL image for database management. Each Dockerfile encapsulates the necessary steps to create a self-contained environment for its respective components within the overall application architecture. The current state of deployment is that individual units are able to run simultaneously from both docker-compose and Kubernetes(in the form of individualized pods) and are able to communicate with one another. The containers (docker-compose) and the pods (Kubernetes) can ping one another successfully. However, they are not sending the proper information back and forth to make the application run with complete functionality.

4.4 Challenges

The current challenge faced is enabling the individualized units (containers when using Docker and Pods when using Kubernetes) to communicate the needed information to enable app functionality. The units themselves are able to communicate, but they are not communicating the needed live input information. The main reason for this challenge is due to the complexity of using web sockets. The application code in both the frontend and backend has hard-written code specifying where to send a receive data. As it stands now, these areas of code are static using the URL of the Local Host. The believed and proper solution is to use environmental variables within the frontend and backend programs. This would allow the code to be dynamically changed according to the deployment vector. Then, of course, set these environmental variables in both the docker-compose and Kubernetes manifest files accordingly. This solution is currently in the works of being implemented with the modified docker-compose and Kubernetes manifest files ready. Next, would be to set the environmental variables in the frontend and backend application. Finally, testing and monitoring the deployment would follow. To be clear, it is not guaranteed the proposed solution is one hundred percent fault-proof due to the complexities of web sockets, but it is believed to be the proper solution.

Throughout the development journey of RamBluff, our team accumulated invaluable expertise not only in crafting a comprehensive cloud-based application but also in crucial domains such as task delegation, feature prioritization, seamless team communication, and the intricate art of debugging. As we gaze towards the horizon of possibilities, numerous avenues unfurl for the future evolution of RamBluff. Primarily, refining the user interface stands as a pivotal endeavor to distinguish our application amidst its peers. Furthermore, expanding RamBluff's repertoire to encompass additional gaming offerings like poker, slots, baccarat, among others, holds promise for enriching user experiences. Presently, users are tethered to their socket client connections; however, envisioning a future where users possess dedicated RamBluff accounts to track their earnings and historical gaming performances emerges as a cornerstone of our roadmap.

4.5 Live Demo

If you are interested in viewing a live demo of our current application it can be found at the link below. Along with that, within our github repository, you can find our docker images, and docker-compose.yaml files.

https://drive.google.com/file/d/14oh7T5do-tkL1v0x9NSxajMuiemzafxI/view?usp=drive_link

Chapter 5: Conclusion

5.0 Reflections

Throughout the development journey of RamBluff, our team accumulated invaluable expertise not only in crafting a comprehensive cloud-based application but also in crucial domains such as full-stack development, socket.io implementation, task delegation, feature prioritization, seamless team communication, and the intricate art of debugging. As we gaze towards the horizon of possibilities, numerous avenues unfurl for the future evolution of RamBluff.

5.1 Future Frontend

Refining the user interface stands as a pivotal endeavor to distinguish our application amidst its peers. We can improve our app to allow users to join ongoing games, showcase a ledger of individual players' stack count plus or minus, incorporate a rebuy system, along with many other polishing features. Along with that, we would be able to have distinct game screens for future gamemode implementations. Doing so, would lead us to allow users to have distinct RamBluff accounts that track their earnings and their play history.

5.2 Future Gamemode Implementation

Although we shifted away from poker, due to it's intricacies, it's implementation is not impossible. With our socket infrastructure in tact, we would be able to implement poker alongside other games such as: slots, baccarat, and in theory any other card game. These inclusions would enrich user experiences.

5.3 Future Cloud Deployment

The RamBluff team plans to debug our cloud deployment in order to allow users from anywhere to access and enjoy our application. We are still unsure of what exactly our issue is, but we will work diligently to complete this aspect of our project. As, it will not only give us valuable experience, but give our potential users the ability to play with their friends.

References

Docker Documentation: https://docs.docker.com/
Express.js Documentation: https://expressjs.com/en/guide/routing.html
GitHub Repository: https://github.com/zleopold/RamBluff/tree/main
Kubernetes Documentation: https://kubernetes.io/docs/home/
Node Documentation: https://nodejs.org/docs/latest/api/
React Documentation: https://react.dev/reference/react
Socket.IO Documentation: https://socket.io/docs/v4/

Sean Berlin

SeanBerlin724@gmail.com | Graduating Spring 2024| linkedin.com/in/sean-berlin-571778201/

EDUCATION

West Chester University **Graduating Spring 2024**
B.S Computer Science- Certificate in Computer Security (NCAE-C) **GPA 3.5**

- **Concentrations:** Computer Science with Cybersecurity Specialization
- **Related Coursework:** Computer Security & Ethics, Software Security, Modern Malware Analysis, Data Communications and Networking, Intro to Cloud Computing, Software Engineering, Data Structures & Algorithms, Database Management Systems

SKILLS

Operating Systems: Windows, Linux
Languages: Java, Python, C#, SQL, OCaml, x86 Assembly, Bash Shell Scripting
Microsoft: Access, Excel, Word, PowerPoint
General IT Security: Basic knowledge of cybersecurity principles, user account management, and access control
Communication: Excellent verbal and written communication skills with ability to convey technical information

COLLEGE INVOLVEMENT

Clubs and Activities from CS Department

- Multiple-time participant in the West Chester University Programming Contest (WCPC) and Computer Science Robotics Camp, showcasing problem-solving skills and algorithm development expertise
- Member of the department's Cyber Security Club, actively participating in hackathons and staying informed about cyber security events

Fraternity

- Member and Community Service Chair for Alpha Chi Rho, organizing and leading community service initiatives to give back within the fraternity and the broader community

JOB EXPERIENCE

Lee's Hoagie House **June 2019 - Current**
Assistant Manager with Third-Party Ordering Oversight

- Supervised and trained a team of up to 10 employees, ensuring a high level of customer service and quality food preparation.
- Maintained a clean and organized shop environment, ensuring compliance with health and safety regulations.
- Managed the implementation and operation of third-party delivery services, ensuring seamless integration with the business's ordering system and procedures.

Camp Invention (STEM) Leadership Intern
Instructor

- Led and supervised groups of up to 30 elementary school students during STEM-focused summer camps, ensuring safety and engagement at all times.
- Developed and facilitated hands-on activities and experiments that fostered creativity, problem-solving, and critical thinking skills among campers.
- Mentored and trained junior counselors to promote leadership development and effective teamwork.

Kirtan Chavda

123-456-7890 | kirtanchavda2003@gmail.com | linkedin.com/in/kirtan-chavda-5561b823b |

Education

West Chester University West Chester, PA
Bachelors in Computer Science *Aug. 2021 – May 2025*

Experience

Manager July 2021 – Present
Best Eye Vision *Malvern, PA*

- Supervised the business and the departments to help function the clinic
- Utilized skills for financial decisions, marketing and growth of the doctor
- Worked with several companies to give services

Volunteer

BAPS Jan 2018 – Present

- Teach younger kids the way of living and serving the community
- Organizing several events to generate donations and run the non-profit organization
- Collaborating with other volunteers to construct ways of growth
- Giving speeches at different locations to motivate the volunteers

Akshardham March 2023 – Aug 2023

- Served in the construction of a temple in New Jersey
- Helped the department of fabrication where I got experience of working in a team
- Developed and completed several projects

Technical Skills

Languages: Java, C/C++
Concepts: Ethics, Security, Data Structures and Algorithms
Management
Public Speaking

Jared Colletti

jaredcolletti123@gmail.com • https://www.linkedin.com/in/jared-colletti-975316215/ • github.com/JaredC71

EDUCATION

West Chester University | **West Chester, PA**

B.S. in Computer Science | **GPA:** 3.3/4.0 **May 2024**

Relevant Courses: Operating Systems, Data Structures, Foundations of Programming, Algorithms & Analysis

Certificate: University of Pennsylvannia's LPS Coding Bootcamp

TECHNICAL SKILLS

Programming Languages: Java, Python, JavaScript, C, Haskell

Operating Systems: Windows XP/Vista/7/8/10/11, Linux

PROJECTS

Online Poker Game | University Project **May 2024**

- Collaborated with a team of 3 and developed a fully functional web application with a dynamic **Javascript-based** front-end, a RESTful server-side API, a relational database that stores back-end application state, and an ORM layer to access data in the database

Unix Shell | University Project **Dec 2023**

- C implementation of terminal-based Unix Shell
- Features supported : redirection, parallel processing, built-in commands

PROFESSIONAL EXPERIENCE

Programming Instructor | CodeWizardsHQ | Remote **Nov 2021 - May 2022**

- Create lessons plans
- Teach curriculum to 6 - 14 student classes
- Grade assignments, projects, submit feedback, and attend virtual training sessions

Zachary Leopold

zacleopold1@gmail.com • linkedin.com/in/zacharyleopold • github.com/zleopold

CAREER OBJECTIVE

As a dedicated soon-to-be graduate with a Bachelor's degree in Computer Science, I am eager to leverage my academic knowledge and skills in software development/engineering to contribute effectively to a dynamic team. Passionate about solving complex problems and staying up to date with emerging technologies. Seeking an entry-level position where I can grow professionally, learn from experienced colleagues, and make meaningful contributions to innovative projects.

EDUCATION

West Chester University of Pennsylvania | West Chester, PA

B.S. in Computer Science | GPA: 3.76 **Exp. Grad. Spring 2024**

Member of Upsilon Pi Epsilon, International Computer Science Honor Society

- **Relevant Courses:** Cloud Computing, Data Structures & Algorithms, Software Engineering, Computer Security
- **Dean's List:** Fall 2022, Spring 2023, Fall 2023
- **Clubs:** Competitive Programming, Computer Science, Cyber Security, WCU Esports

Millersville University of Pennsylvania | Millersville, PA

B.S. in Computer Science | Major GPA: 3.9 **Aug 2019 - May 2021**

Invited to Honor Society sophomore year

TECHNICAL SKILLS

Programming Languages: Java (Advanced), Python (Intermediate), C (Intermediate), GO (Beginner), JavaScript (Beginner)
Operating Systems: Windows, Linux
Software/Tools: Visual Studio, Eclipse, Git/Github, MongoDB

PROJECTS

GalaxyMentor | Independent Study Project **September 2023 - Present**

- Web application, created by Dr. Si Chen, that utilizes artificial intelligence to teach students across varying topics with a focus on Computer Science
- Studied and assisted with the implementation of features such as storing and accessing information within a database
- Use of GO, JavaScript, CSS, MongoDB

EXPERIENCE

Semi-Pro Counter-Strike Player **May 2016 - Present**

- Facilitated structured practice, analyzed gameplay and developed strategies with team members within team-set deadlines in order to perform to our highest abilities
- Collaborated with the team to consistently push the current ideologies and playstyles for upwards of seven (7) different maps
- Promoted and enforced the improvement of mechanical, conceptual, and communication skills

Hardscaper | EPLandscaping | Garnet Valley, PA *(Summers)* **June 2016 - August 2019**

- Completed projects that complied with given specifications, standards, and contractual obligations
- Consulted with customers to explain the methodology and decisions made throughout the project to ensure satisfaction