5. Frontend Development

React Application Structure The project features a React application for the frontend using Typescript. This is designed to provide and engaging and interactive experience for the users. It has a landing page with multiple sections, and navigation to each implemented game. Furthermore it is developed in a modular way, ensuring that scaling the project up and down can be done easily.

Key sections and features The landing page is the first page of the application and where the user will start when they enter. It is built up by using several components that form the landing page when put together. The header is at the top of the landing page, and serves as a navigation bar. It has the logo of the application and then links to different sections of the application: Home, About, Features and Contact. Below that is the hero section, where you would typically have a banner, welcoming the user onto the page and inviting them to navigate around the application to explore the different features. Below that is the games section where the different implemented games are listed along with buttons that navigate to said games. Below that there is a "Start" section, which is component that invites the user to sign up in order to play the games. Lastly a footer is implemented, which is a generic footer containing copyright information and links to resources such as terms of service and privacy policy. The application also has seperate game pages, which when navigated to, shows each of the games.

Navigation and Routing

For routing in the application react-router-dom is used to enable navigation between different pages. The routes are defined in the index.tsx file, and currently contains routes for blackjack, coinflip and the landing page. This setup makes it easy if new games are added, because they can just be added to the index file.

TypeScript Integration The project is built with TypeScript, which makes it possible to do static type checking. This is super helpful in order to catch errors during development instead of having to catch them at runtime. Typescript and the static type checking is best used when defining interfaces and types for components props and states. This limits the data passed between componenets such that it has to be structured properly for the component to accept it.

Vite Build Tool Implementation for building the application this project uses Vite. This is a popular build tool with some advantages mainly aimed at ease of use for the developers. The advantages include a really fast development server providing instant updates when changes are made to the code. It supports TypeScript out of the box, which minimized the amount of work developers have to focus on configuring build pipelines.

State Management In this project, Reacts built in state management hooks are used. UseNavigation is used from react-router-dom is used to handle the navigation state and allowing the user to navigate to different pages. Local states are also being used within the components with UseState. UseState can be used to handle specific UI states, for example when users are interacting with the application, or when dynamic content rendering is used.

Component Architecture

Security Features

Token Security Implementation

Package Management