

## JSC “Kazakh-British Technical University”

Faculty of Information Technology

**Course**: Introduction to Blockchain: Mathematical Foundations of Decentralized Systems

**Final Project**

**Kazakh Khanate**

Team name: **Hito**

Prepared by: Kazhina Ayazhan

Kazymov Ali

Kuandykova Aida

Zhandyrbai Medet

Alipbaev Bibarys

Sattaruly Yerassyl

Checked by: Mukash N.

Almaty, 2024

**CONTENT**

[INTRODUCTION 3](#_bookmark0)

[MAIN PART 4](#_bookmark1)

[CONCLUTION 5](#_bookmark2)

[REFERENCES 6](#_bookmark3)

[ANNEX 7](#_bookmark4)

Introduction

**1.1. Objective**  
 The objective of this project was to design and implement a game that integrates blockchain technology. The primary goal was to deepen our understanding of how to connect applications to blockchain networks, interact with smart contracts, and incorporate decentralized technologies into a functional product. By working on this project, we aimed to explore the practical aspects of blockchain integration, including managing accounts, executing transactions, and deploying contracts. This project is highly relevant in today’s world as blockchain continues to gain significance across industries, providing secure, transparent, and decentralized solutions. Learning how to harness this technology equips us with essential skills for contributing to modern technological advancements.

**1.2. Description**  
 This game is a historical strategy experience inspired by the Kazakh Khanate, where players manage their armies, acquire ***Batyrs***, and engage in battles. Blockchain technology is at the core of the game's functionality, enabling decentralized management of player accounts, transactions, and game data. Initially, the game operated using a local blockchain instance provided by Ganache, which allowed for testing and development in a controlled environment. Subsequently, the game was migrated to the Binance Smart Chain (BSC) Testnet, with wallet integration via Metamask, making it accessible and functional on a live blockchain network. This integration ensures that all game interactions, such as resource purchases and battles, are securely logged as blockchain transactions, reinforcing the decentralized nature of the game.

**1.3. Approach**  
 The development process began with setting up the local blockchain environment using Ganache and Hardhat to manage and test smart contracts. The game logic and user interface were developed alongside this setup, with the blockchain serving as the backend for transactions and data storage. Using Ganache allowed for seamless testing and debugging during the early stages of development. Once the functionality was validated, modifications were made to the codebase to integrate Metamask for account management and connect to the Binance Smart Chain Testnet. The game was redeployed and tested on the live network, ensuring all features worked as intended. Finally, the project was made accessible via GitHub Pages for demonstration purposes, showcasing the successful transition from a local development environment to a public blockchain.

Main Body

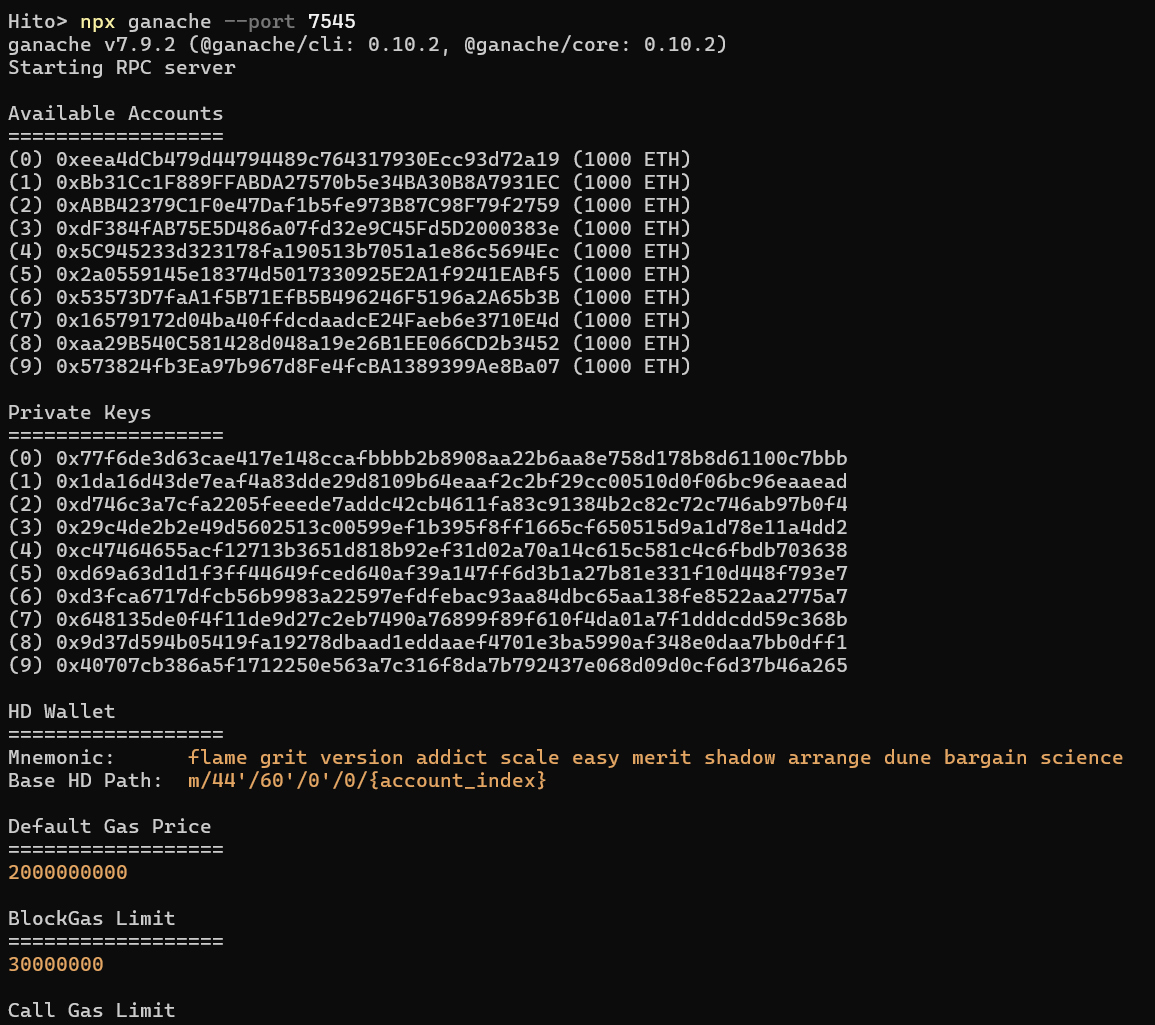
**Programs and Tools Used**

* **Ganache**: A local blockchain instance for testing and development.
* **Hardhat**: A development environment for writing, testing, and deploying smart contracts.
* **web3.js**: A JavaScript library used for interacting with the blockchain.
* **MetaMask**: A browser extension wallet for connecting to the Binance Smart Chain Testnet.
* **HTML, CSS, JavaScript**: For the game’s frontend design and logic.
* **GitHub Pages**: For deploying and showcasing the game on a public platform.
* **Node Package Manager (npm)**: Package manager for Node.js to install hardhat and ganache.
* **Visual Studio Code (IDE)**: Code editor that was used to write Solidity, HTML, CSS, and JavaScript code.

**Development Steps**

**Ganache.**  
The development process began by setting up a local blockchain environment with Ganache.





Ganache provided a private blockchain to test the integration of accounts, transactions, and smart contracts.

We then can deploy our smart contract to local Ganache using hardhat to start testing:



This is the configuration file *hardhat.config.js* that we used to deploy the contract to Ganache:

require("@nomicfoundation/hardhat-toolbox");

/\*\* @type import('hardhat/config').HardhatUserConfig \*/

module.exports = {

  solidity: "0.8.19",

  networks: {

    ganache: {

      url: "http://127.0.0.1:7545",

      chainId: 1337

    }

  },

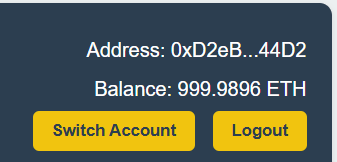
  paths: {

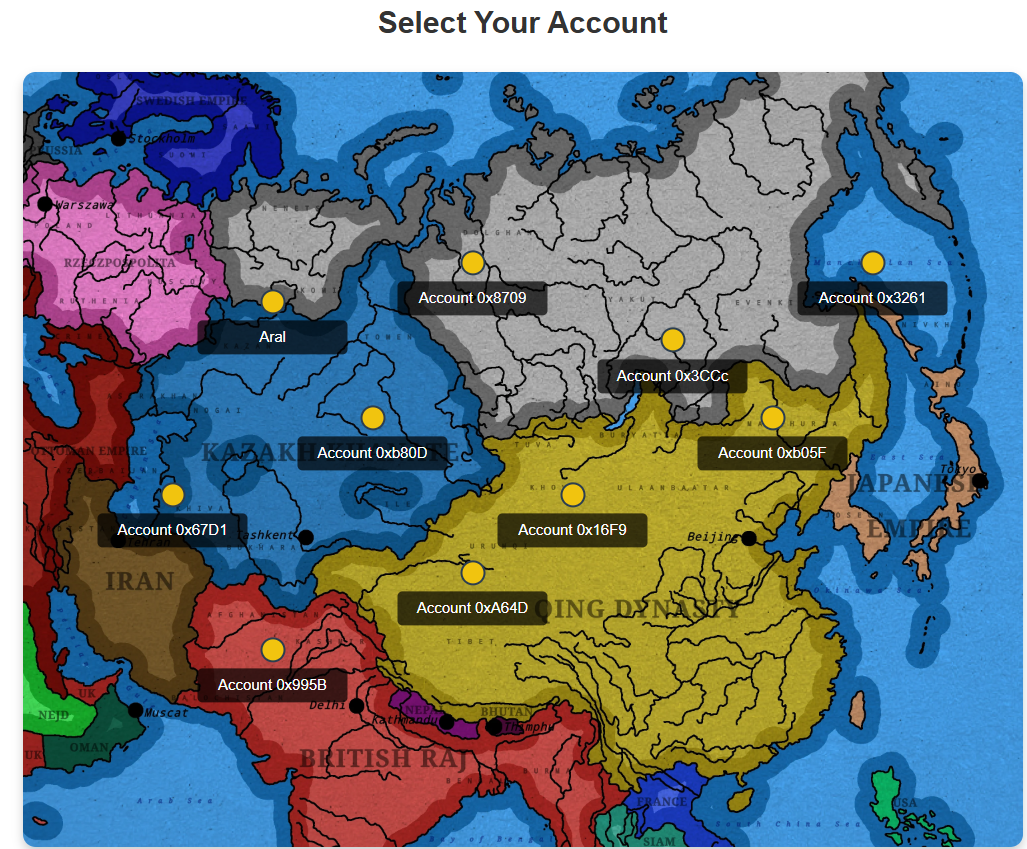
    artifacts: './src/artifacts',

  }

};

Ganache also provides free and unlimited number of tokens that we can use to test our application:



10 free accounts from Ganache for testing:  


This method significantly reduced time that we would have spent waiting for transactions on BSC Testnet, tBNB tokens spent, and accounts needed.

Using Hardhat, we wrote and deployed smart contracts tailored to handle game logic such as troop acquisition (archers and cavalry), battles (attack players with different addresses), and lucky box purchases (to win batyrs). The contracts were tested locally to ensure their functionality before integrating them into the game.

Below are the structures (data models) present in the ***KazakhKhanate.sol*** smart contract.

struct Khanate {

        string name;

        uint256 level;

        uint256 archers;

        uint256 cavalry;

        uint256 experience;

        uint256 wins;

        uint256 losses;

        uint256 successfulDefenses;

        bool initialized;

    }

 struct BatyrStats {

        string name;

        string ability;

        string description;

        uint256 power;

        uint256 archerBonus;

        uint256 cavalryBonus;

        bool canBattle;

    }

struct Battle {

        address attacker;

        address defender;

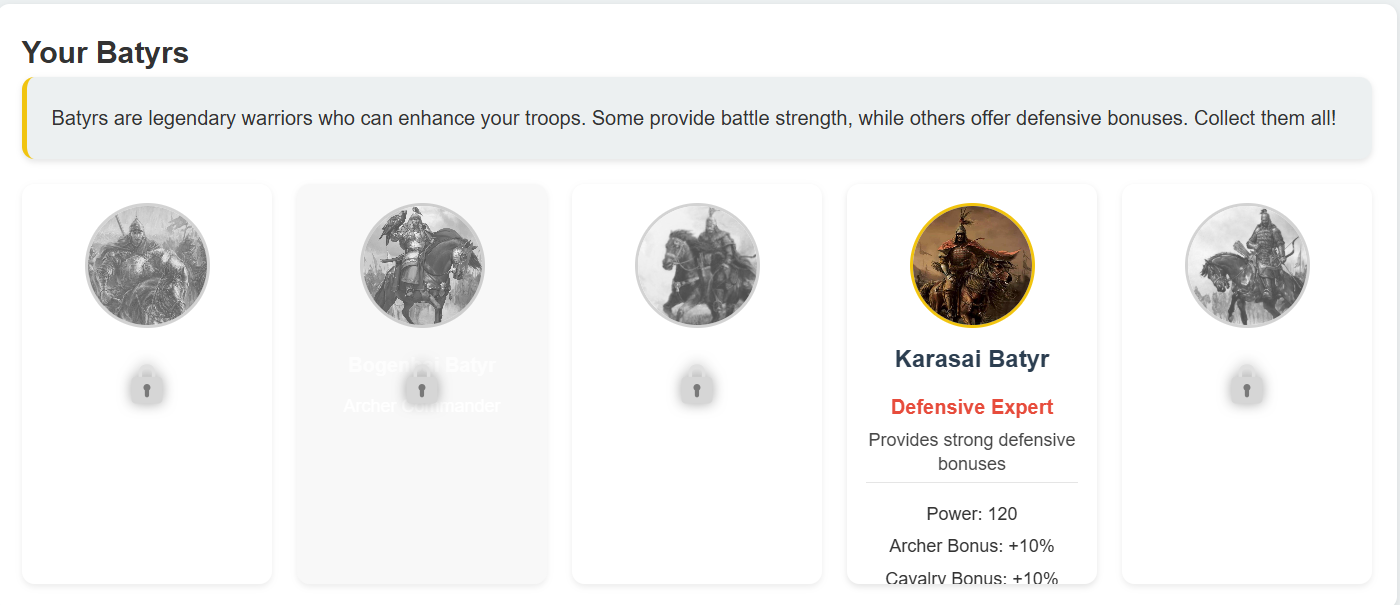
        uint256 startTime;

        bool resolved;

        uint256 travelTime;

    }

Batyrs here, for instance, serve as a valuable NFTs. They can improve army’s power or provide defense bonus. They can be acquired by purchasing lucky boxes in the game for some amount of BSC Testnet tBNB tokens.



Below is a function in the contract that is used to purchase troops, for example:

    function purchaseTroops(uint256 archerAmount, uint256 cavalryAmount) external payable onlyInitialized {

        uint256 totalCost = (archerAmount \* ARCHER\_PRICE) + (cavalryAmount \* CAVALRY\_PRICE);

        require(msg.value >= totalCost, "Insufficient payment");

        khanates[msg.sender].archers += archerAmount;

        khanates[msg.sender].cavalry += cavalryAmount;

        emit TroopsPurchased(msg.sender, archerAmount, cavalryAmount);

        if (msg.value > totalCost) {

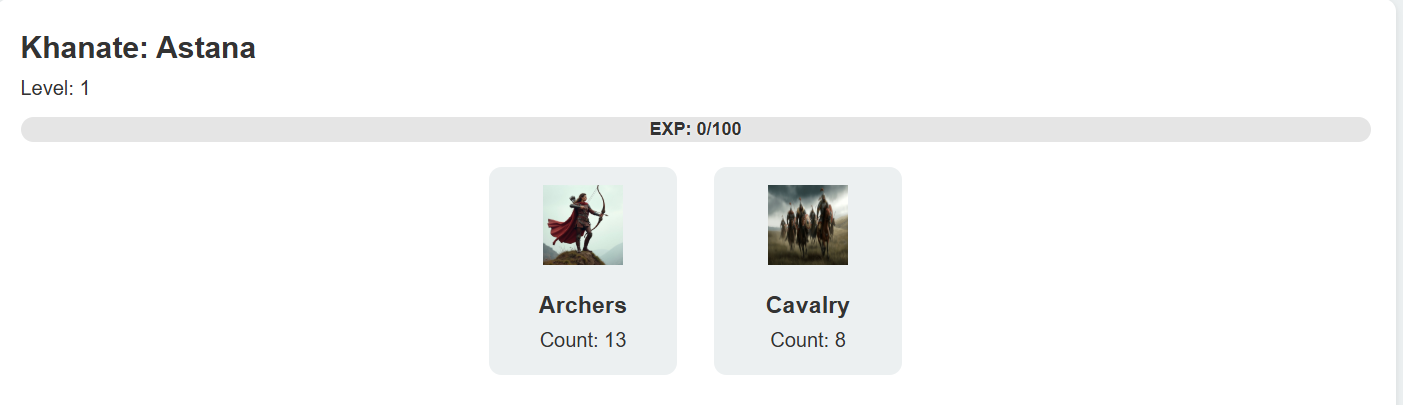
            payable(msg.sender).transfer(msg.value - totalCost);

        }

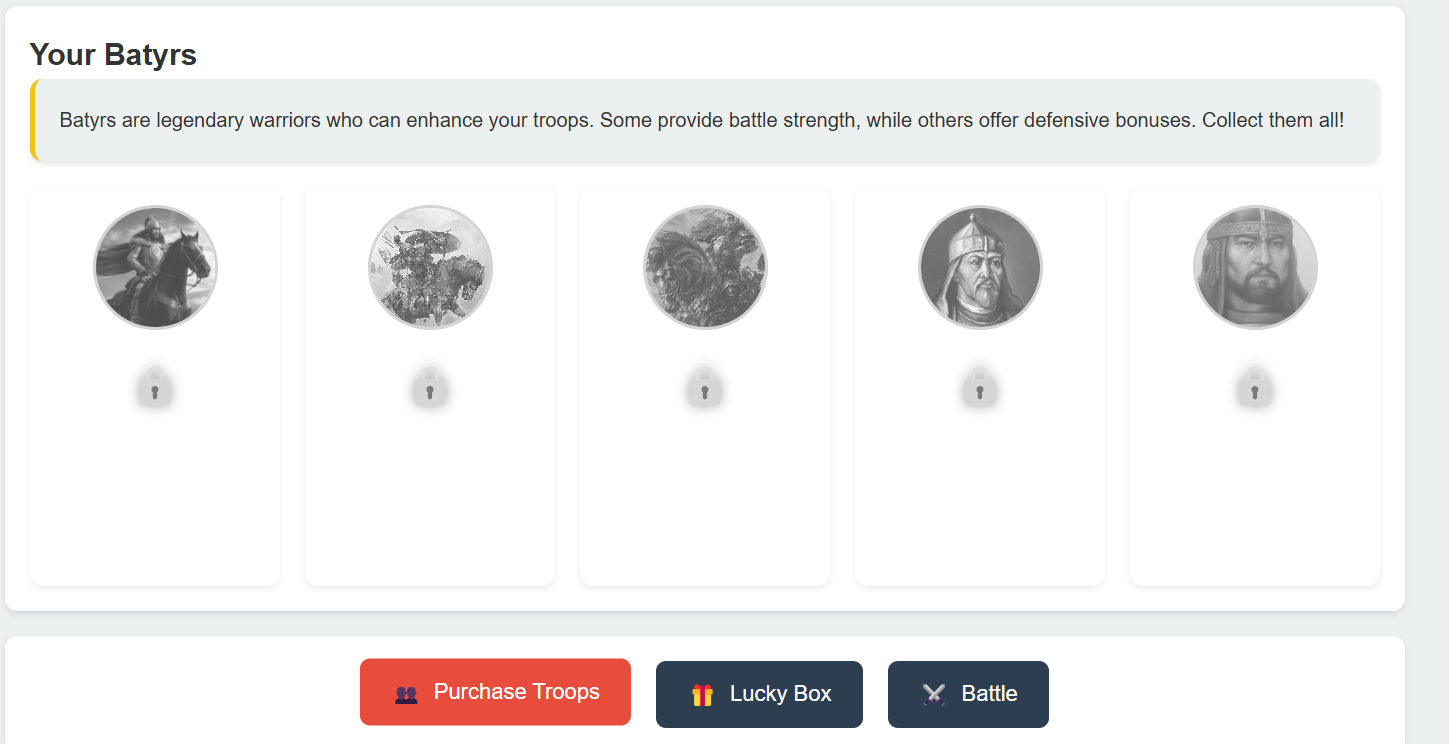
    }

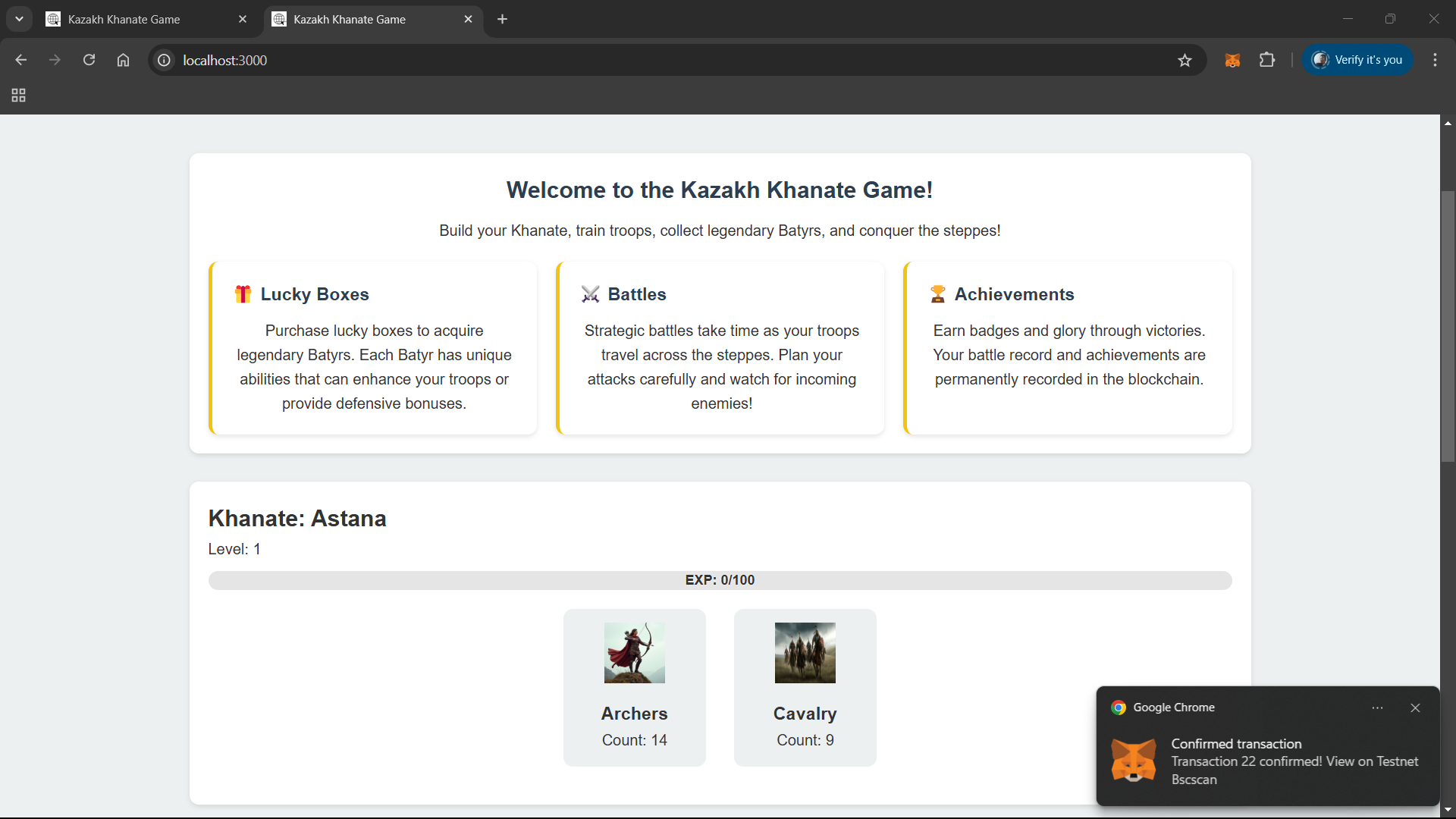
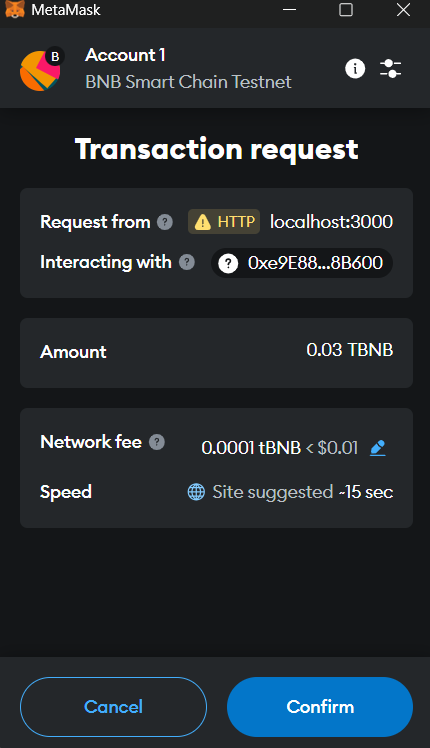
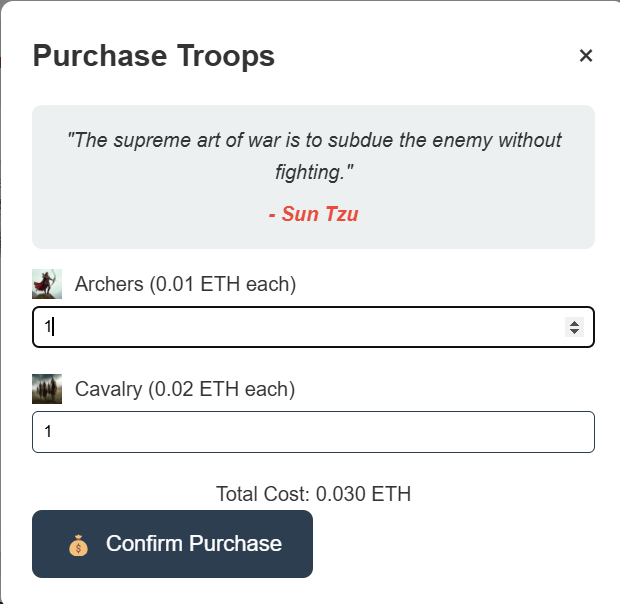
We increase the number of archers or cavalry if the owner of a Khanate purchases any.

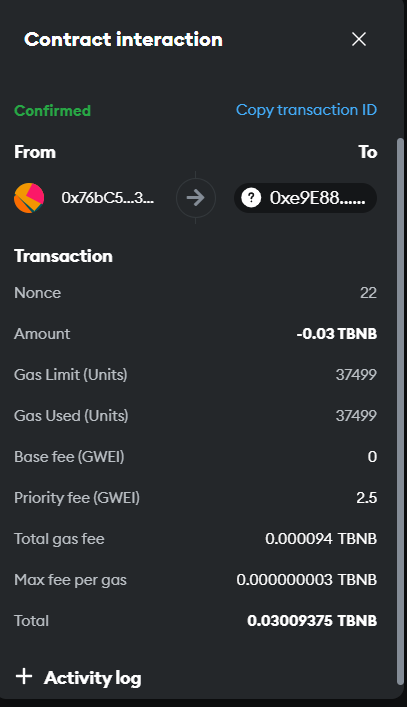
Below is an example of purchasing troops using MetaMask:



We can buy archers and cavalry using *Purchase Troops* button:

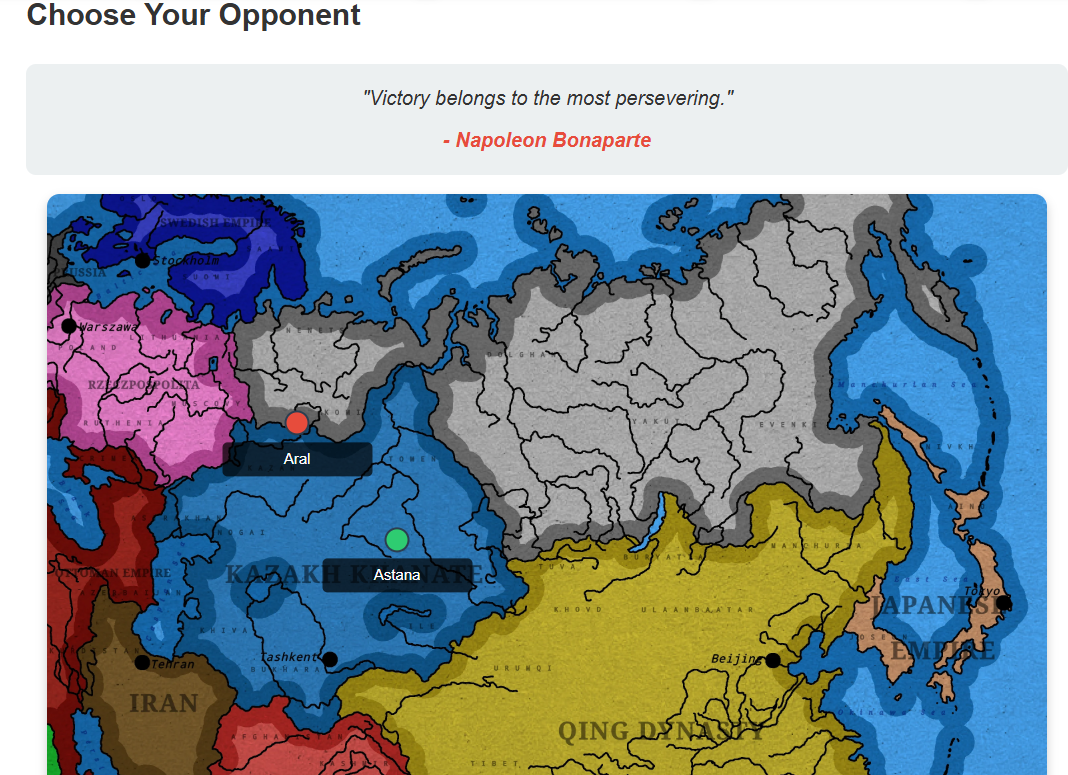


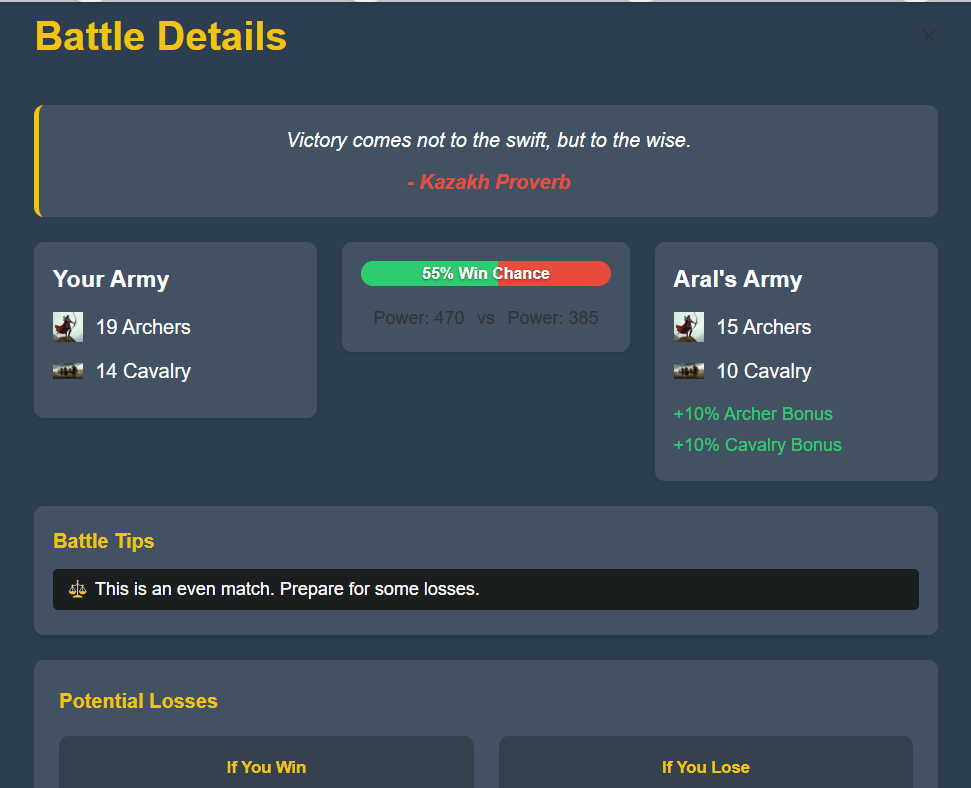




The frontend was developed using HTML, CSS, and JavaScript, with web3.js managing the interaction between the UI and the blockchain. Initial testing focused on connecting the game to Ganache to verify that transactions were being recorded correctly. For instance, logging into a player account, making purchases, and initiating battles were all tested locally with Ganache as the backend.

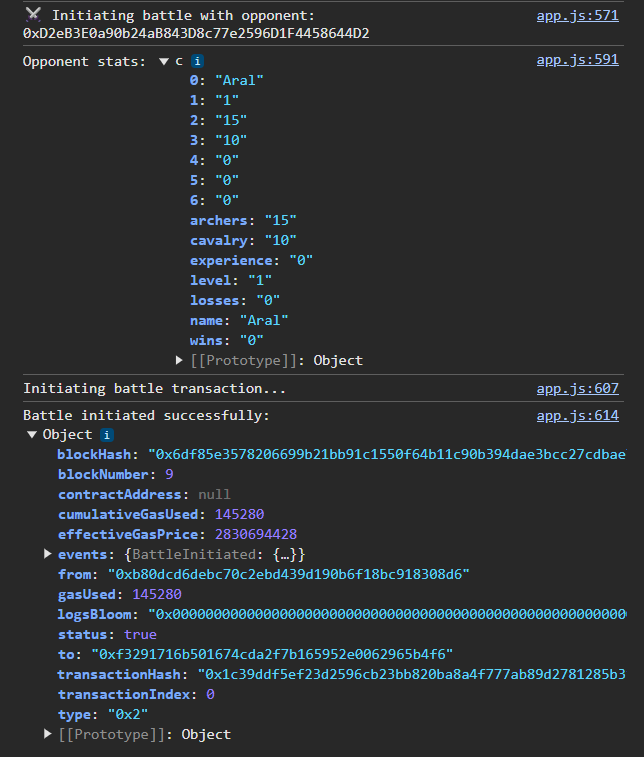
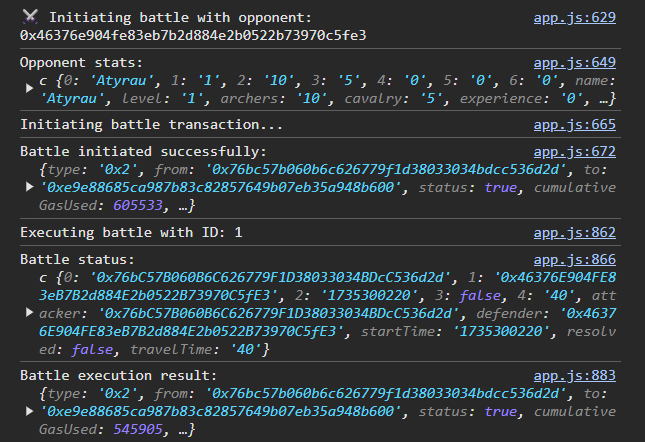
Here are the examples of attacking a Khanate located at another Ganache account address:



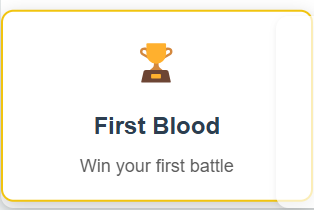




We can open Developer Tools (F12) and see logs:

And players can win achievements that provide motivation to playing the game:



Once the local testing phase was complete, the code (app.js) was modified to connect the game to the Binance Smart Chain Testnet using MetaMask. This required updating the *hardhat.config.js* *app.js* code and ensuring compatibility with the network:

require("@nomicfoundation/hardhat-toolbox");

require("dotenv").config();

/\*\* @type import('hardhat/config').HardhatUserConfig \*/

module.exports = {

  solidity: "0.8.19",

  networks: {

    bscTestnet: {

      url: "https://data-seed-prebsc-1-s1.binance.org:8545",

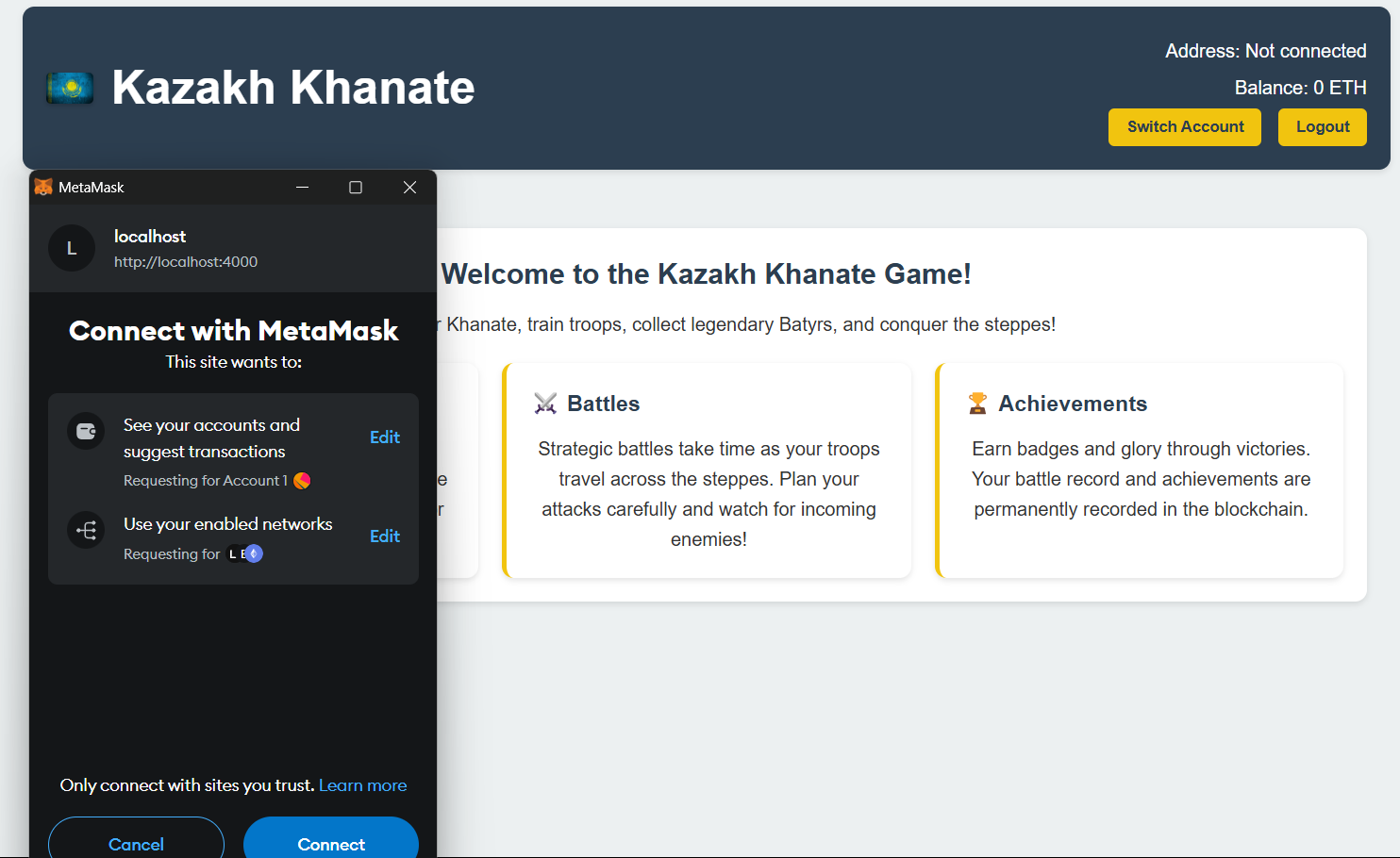
      chainId: 97,

      accounts: process.env.PRIVATE\_KEY ? [process.env.PRIVATE\_KEY] : [],

      gasPrice: 10000000000 // 10 Gwei

    }};

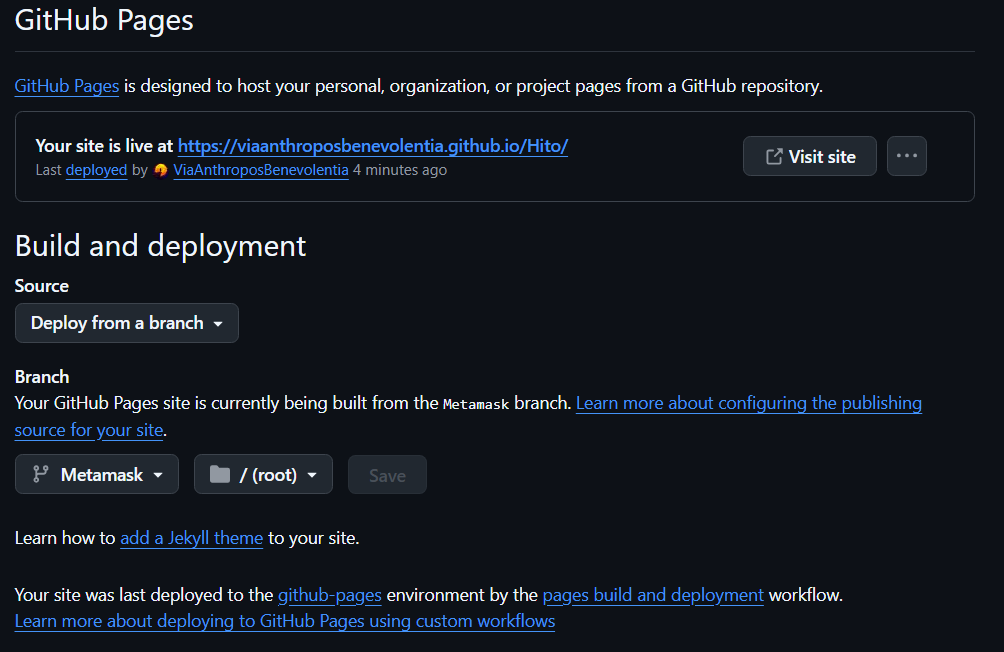
MetaMask was used for wallet integration, allowing users to manage their accounts and interact with the blockchain directly.



After successful deployment to the Binance Smart Chain Testnet, we verified all functionalities, including game logic, smart contract interactions, and transaction records.

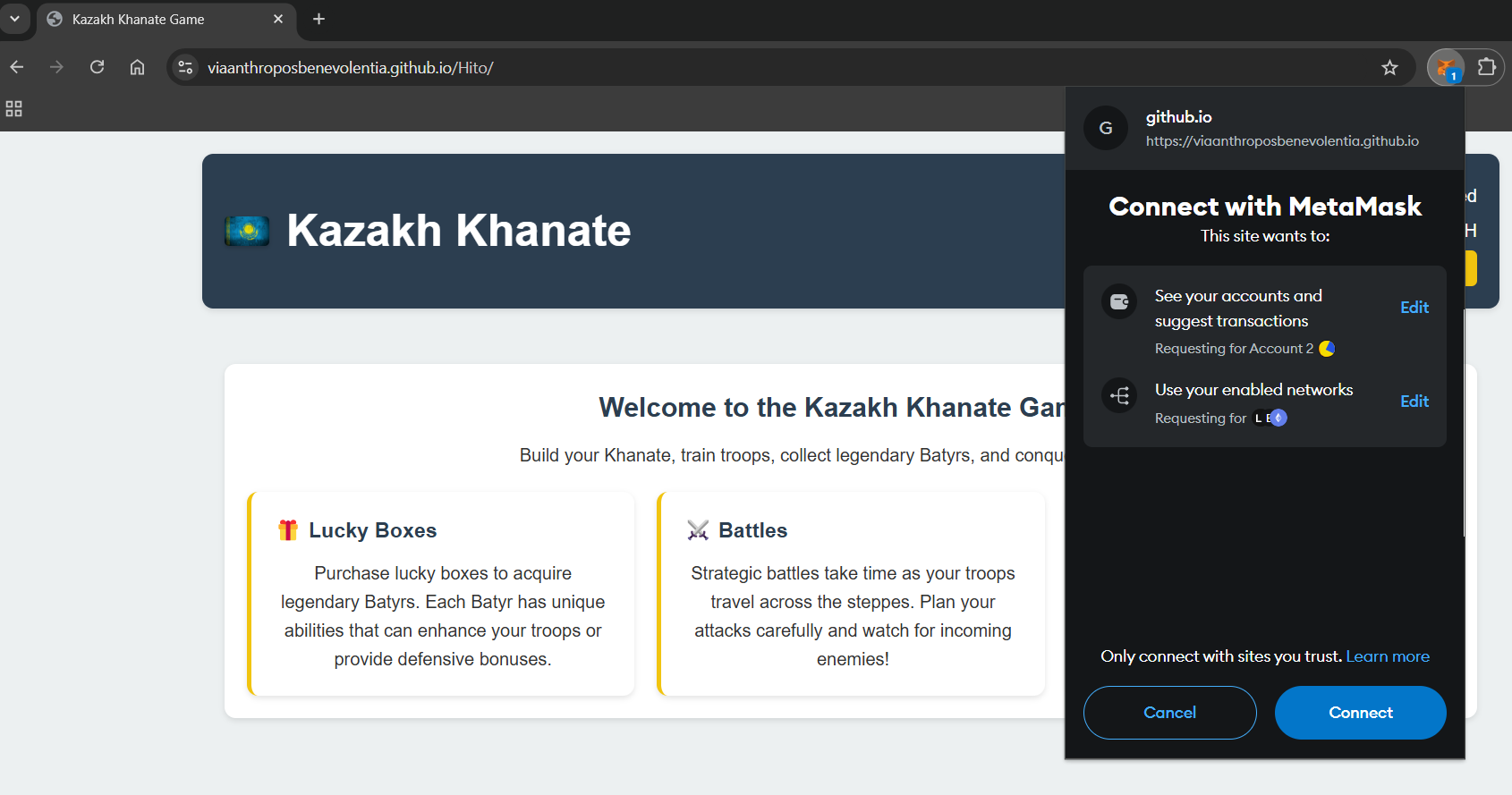
Finally, the game was deployed on *GitHub Pages* for easy access and demonstration purposes.

*GitHub Pages* offers free hosting:

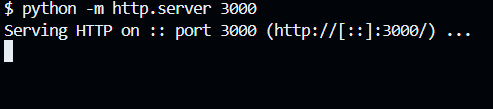


You can play the game at <https://viaanthroposbenevolentia.github.io/Hito/>

Here are the examples of connecting MetaMask wallet to a GitHub Pages website that we’ve just deployed:

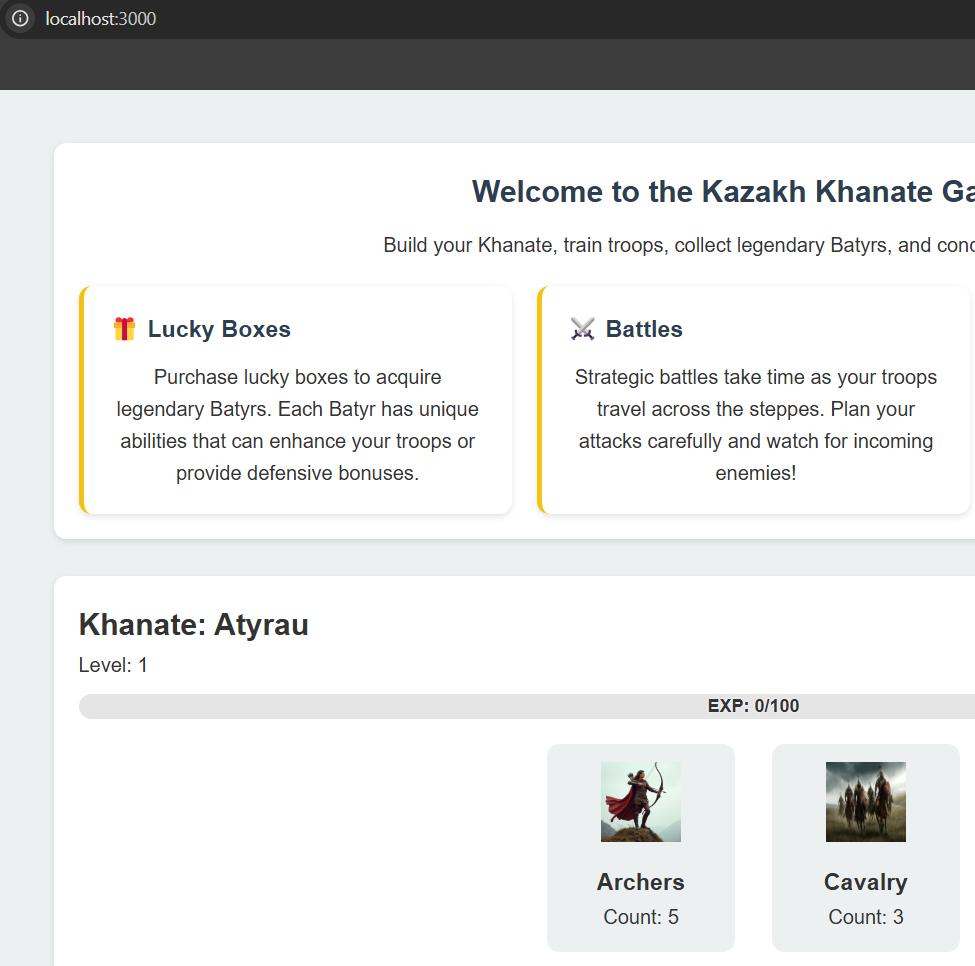


Before hosting our website on the cloud, we have tested connection to MetaMask using localhost:

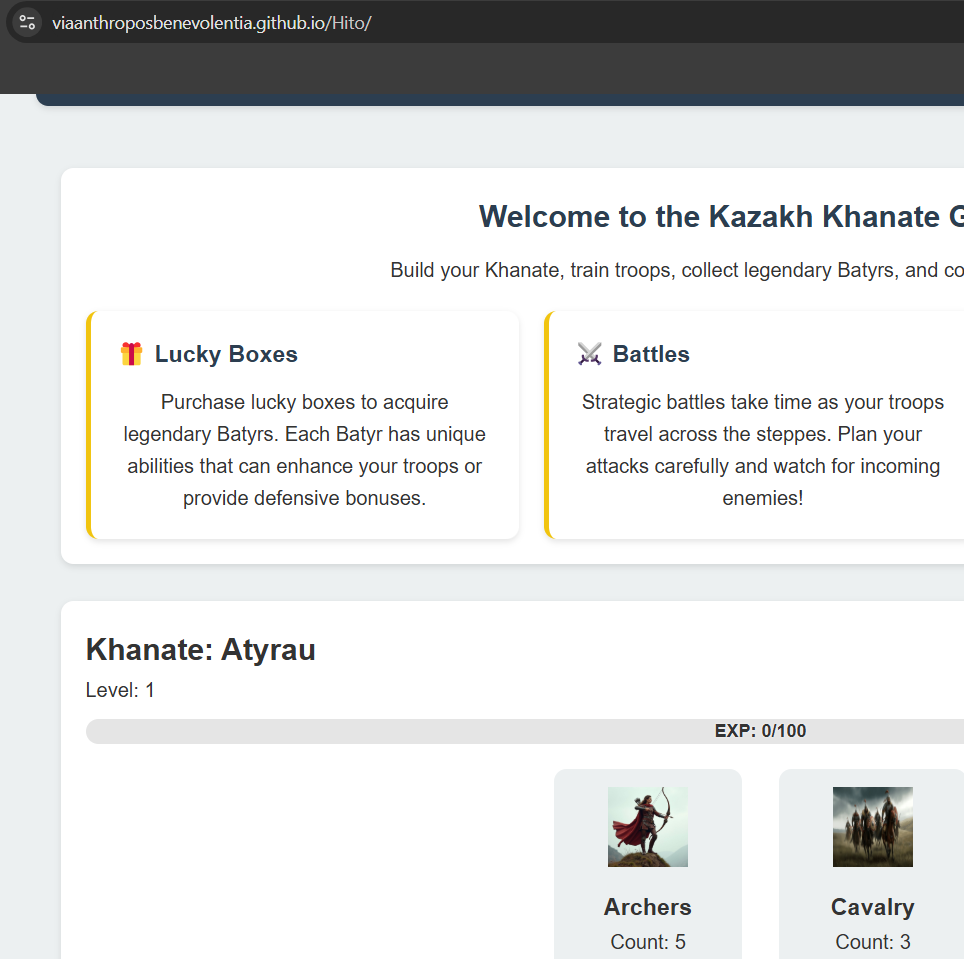


We then can access it in the browser at the *localhost:3000* address.

Previously, we’ve launched an attack from *Account1* and *Account2* loosed some troops because of defeat.



Now, when we’ve logged in to *Account2* from the public website, it still has the same number of troops, indicating that the number of troops, khanate name, level, battle information, etc. are securely stored on the BSC Testnet Networks:



Thus, we were able to build public game where players can buy troops, lucky boxes to win legendary Kazakh Batyrs, and launch attacks to other players.