网络应用开发课程设计

网页游戏合集项目报告

**组员信息：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 姓名 | 学号 | 成员贡献 | 课程设计成绩 |
| 1 | 杨轶洋 | 2022280067 | 注册登录功能、数据库、游戏选择界面及功能、连连看、部分贪吃蛇、框架设计、报告撰写 |  |
| 2 | 刘沛 | 2022280065 | 贪吃蛇、2048 |  |
| 3 | 钟万乐 | 2022280066 | 连连看、素材收集 |  |
| 4 | 任炳儒 | 2022280054 | 贪吃蛇样式 |  |
| 5 |  |  |  |  |

<https://github.com/Kagari-Yang/html_game>

# **引言**

本项目是设计一个包含多个小游戏的网页应用，玩家用户可以通过注册账号，登录到游戏内，选择自己想玩的游戏。目标是提供一个用户友好且互动性强的游戏平台，允许用户在一个集成环境中体验不同的游戏。

# 功能与设计

## 功能介绍

1. 用户登录和注册功能。

用户首先需要注册一个账号才能登录，注册信息包括用户名和密码，注册账号之间的用户名不允许重复。注册完成后，用户可以通过输入匹配的用户名和密码登录进入游戏界面。

1. 游戏选择界面，允许用户在不同游戏间切换。

在游戏选择界面，用户可以选择进入贪吃蛇或者连连看游戏，也可以退出登录，返回注册登录界面。

1. 三款游戏：

贪吃蛇：玩家控制一条不断增长的蛇，目标是吃掉出现在屏幕上的果实。每当蛇吃掉一个果实，它就会变长。

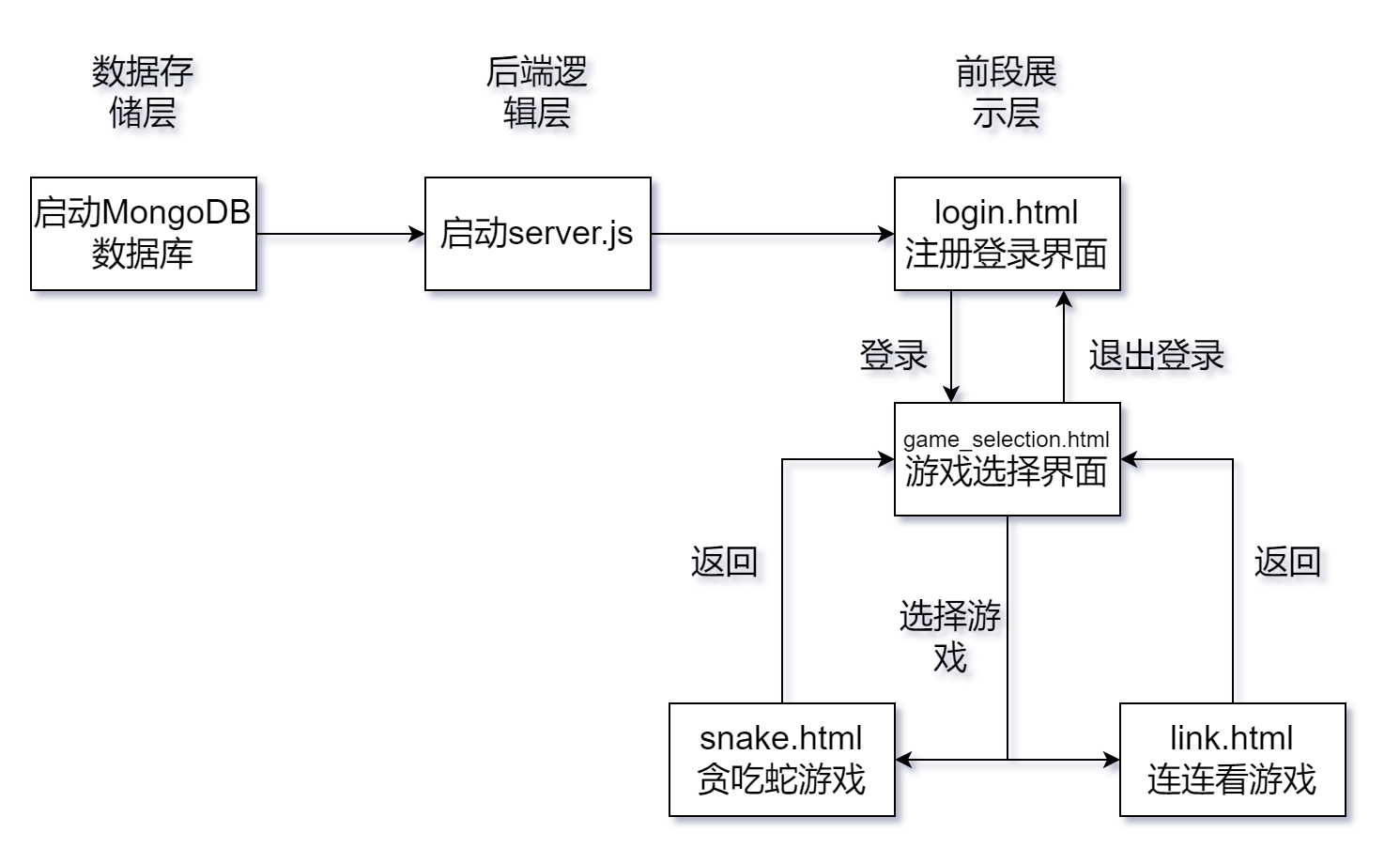
连连看：玩家需要在时间内，找到并消除所有相同图案的方块。

2048：用键盘方向键移动数字方块。当两个具有相同数字的方块接触时会合并，在盘格填满前合并出2048即为通关

## 系统设计

* + 1. 系统架构

本系统采用Web的三层架构，包括前端展示层（HTML/CSS/JavaScript），后端逻辑层（Node.js），以及数据存储层（MongoDB）。具体框架如下图所示：



前端展示层：采用HTML、CSS和JavaScript构建，负责用户界面的渲染和用户交互。HTML提供页面结构，CSS负责样式设计，JavaScript处理用户事件和与后端的数据交互。

后端逻辑层：使用Node.js框架搭建，主要处理业务逻辑，如用户认证、数据处理和与数据库的交互。

数据存储层：采用MongoDB数据库，存储用户信息和游戏数据。MongoDB的文档导向特性使得数据的存储和查询更加高效。

* + 1. 逻辑流程

（1）用户注册与登录：用户首先在登录页面进行注册或登录。系统后端通过bcrypt进行密码加密处理，保障安全性。

（2）进入游戏选择界面：登录成功后，用户跳转到游戏选择界面，目前完成了贪吃蛇和连连看两个游戏选项。

（3）游戏互动：用户选择游戏后，会加载对应的游戏页面。在贪吃蛇游戏中，玩家控制蛇的移动，吃果实增长；在连连看游戏中，玩家需在限定时间内找出并消除所有配对图案。

（4）游戏状态管理：游戏过程中的得分、时间等状态信息实时更新，并与后端通信保存。

（5）退出登录：用户可随时选择退出登录，回到登录界面。

* + 1. 通信协议

我们小组的项目使用HTTP协议进行前后端通信。用户认证信息和游戏状态等数据采用JSON格式交换。前端通过AJAX请求与后端交互，实现异步数据传输，从而提高用户体验和应用性能。这种方式使得前后端分离，前端专注于用户界面和交互，后端则负责逻辑处理和数据持久化。

# **功能实现**

前端：HTML、CSS、JavaScript提供用户界面和游戏逻辑。

后端：Node.js和Express框架处理HTTP请求和响应。

数据库：MongoDB用于存储用户数据。

关键技术说明：

* 1. AJAX异步通信

在login.js文件中，登录功能通过AJAX实现。当用户填写用户名和密码，点击登录按钮时，触发 login() 函数。此函数中，使用 fetch API 发送一个POST请求到后端的 /login 路由。

fetch API 是AJAX技术的一部分，允许前端以异步方式发送HTTP请求。它不会导致页面刷新，而是在后台处理HTTP请求和响应。请求的数据（用户名和密码）以JSON格式发送，后端验证用户信息后，返回相应的JSON格式响应。

如果登录成功，前端会接收到后端的响应，并据此跳转到游戏选择界面，而整个过程无需页面刷新。

同理，在login.js中，用户注册也通过AJAX实现。当用户填写注册信息并点击注册按钮时，触发 register() 函数。该函数同样使用 fetch API 发送POST请求到 /register 路由，传递用户的注册信息。后端接收到这些信息后，进行处理（如用户信息存储）并返回相应的JSON响应。这一过程也是异步的，不会导致页面刷新。

* 1. Express框架

Express是一个灵活且轻量的Node.js Web应用框架，提供了一系列强大的功能来帮助创建各种Web应用和API。它是Node.js最流行的Web框架之一，大幅简化了Node.js服务器端的开发。

在项目的入口文件server.js中，使用Express创建了一个Web服务器。通过调用express()函数，实例化了一个Express应用对象，代表了服务器本身。

Express的一个关键特性是其路由功能。项目中定义了多个路由来处理不同的HTTP请求。例如：

注册路由：使用app.post('/register', async (req, res) => {...})定义了一个处理POST请求的路由。当用户提交注册信息时，此路由激活并处理注册逻辑。

登录路由：同样使用app.post('/login', async (req, res) => {...})定义了登录功能的路由。它监听登录请求并验证用户信息。

* 1. MongoDB数据库

本项目利用Mongoose与MongoDB数据库建立连接。通过mongoose.connect()函数，指定了数据库的URL和必要的配置选项。这样，项目后端就可以访问并操作数据库。

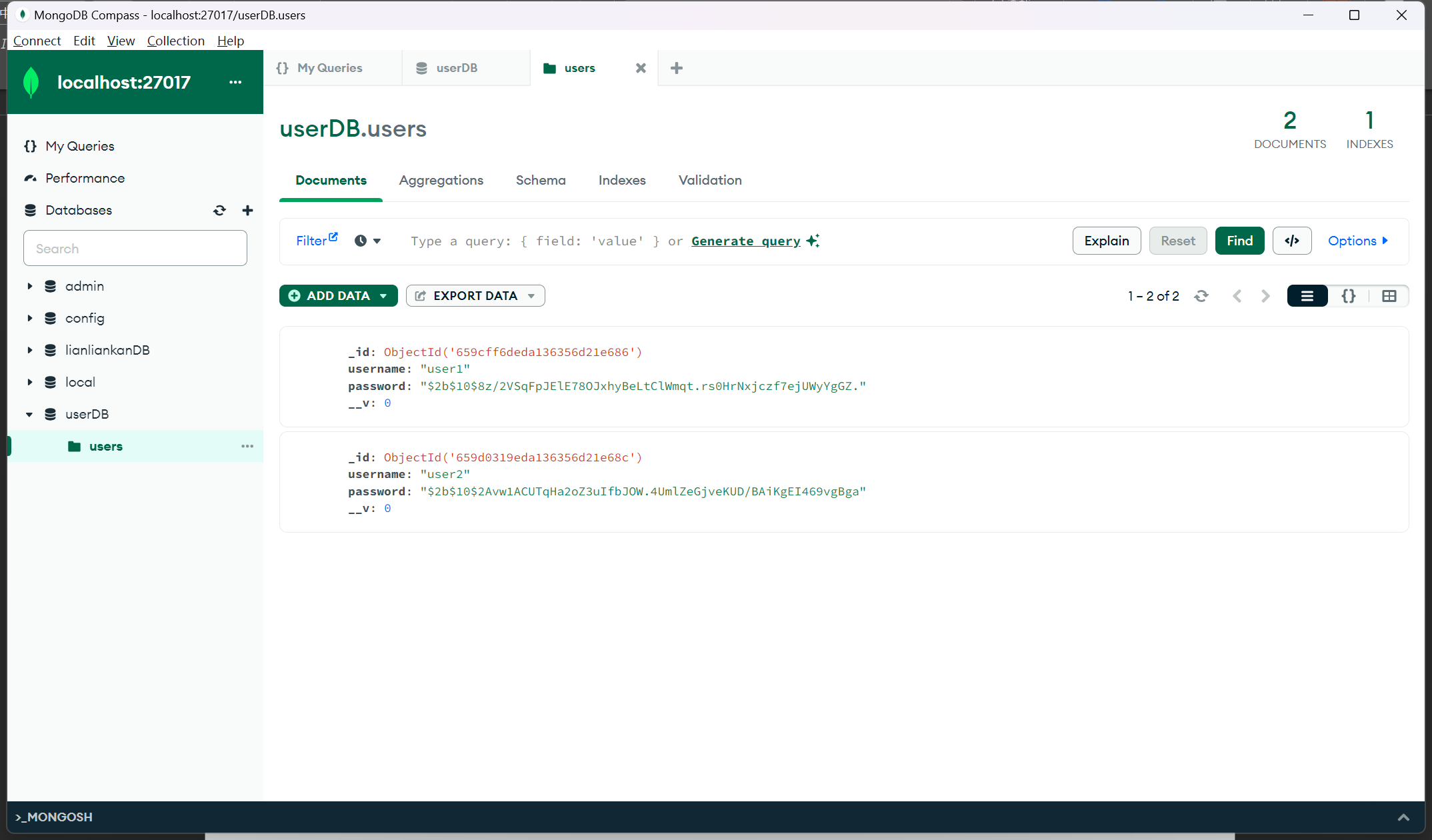
在Mongoose中，根据Schema构建模型。Schema定义了文档的结构和属性类型。在项目中，定义了userSchema，它规定了用户数据的结构，包括用户名和加密后的密码。

Mongoose提供了丰富的方法来查询和操作数据库。例如，在注册路由中，使用User.findOne({ username: req.body.username })查询数据库中是否已存在同名用户。在用户注册时，又使用new User({...})和user.save()来创建新用户并将其保存到数据库。

# **功能展示与评测**

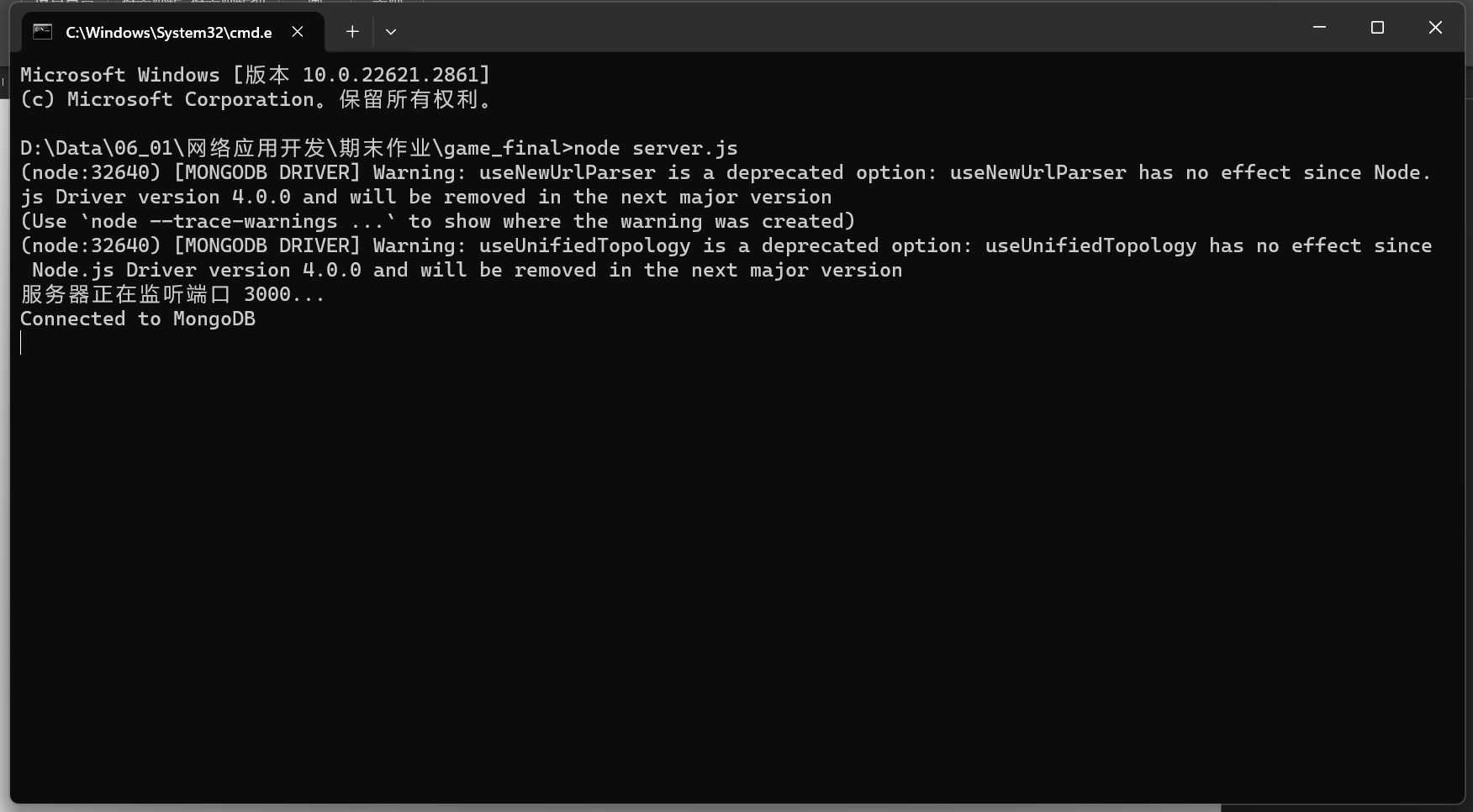
1. 启动MongoDB

利用MongoDB Compass查看数据库的情况，本项目数据库为userDB，每项数据有2个字段，分别是username和password，如下图所示：



1. 启动nodejs服务器

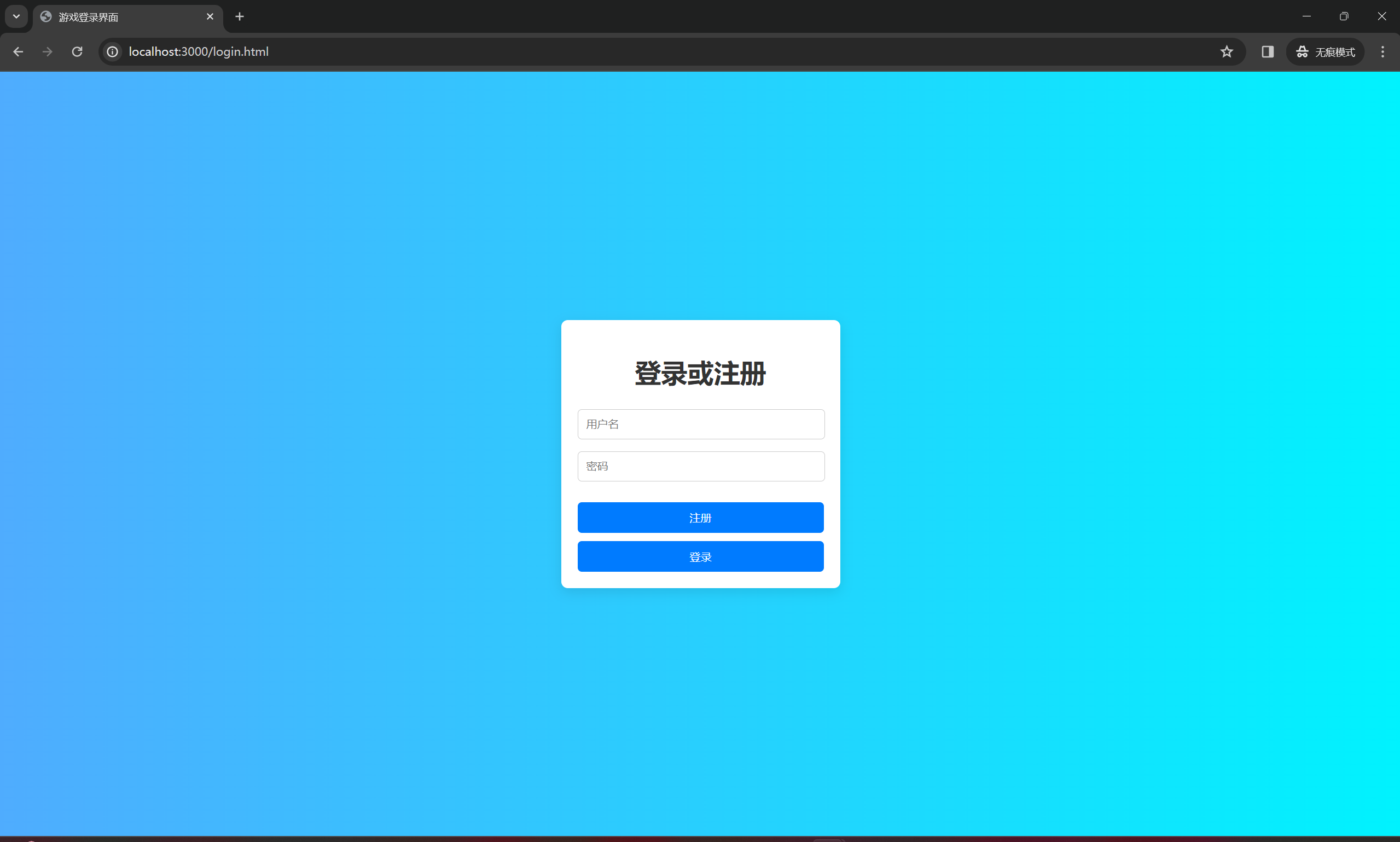
在项目根路径下，打开cmd面板，输入命令node server.js，来启动项目的后端功能，如下图所示：



1. 注册与登录

在浏览器中访问 <http://localhost:3000/login.html>，（注意不要在本地直接打开login.html），

进入login界面，如下图所示：



成功注册：



失败注册：



失败登录：

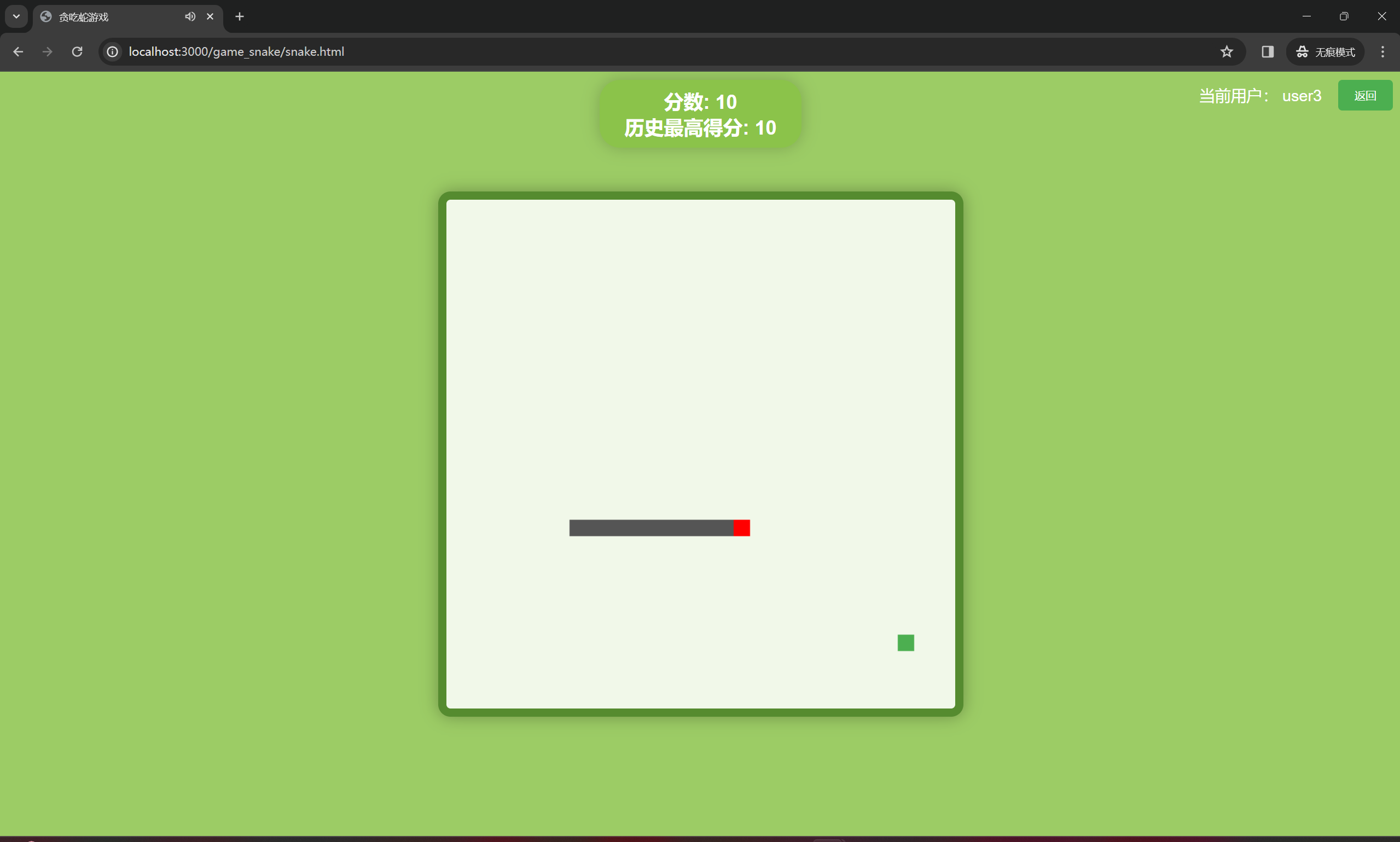


1. 游戏选择界面

用户成功登录后，进入游戏选择界面。

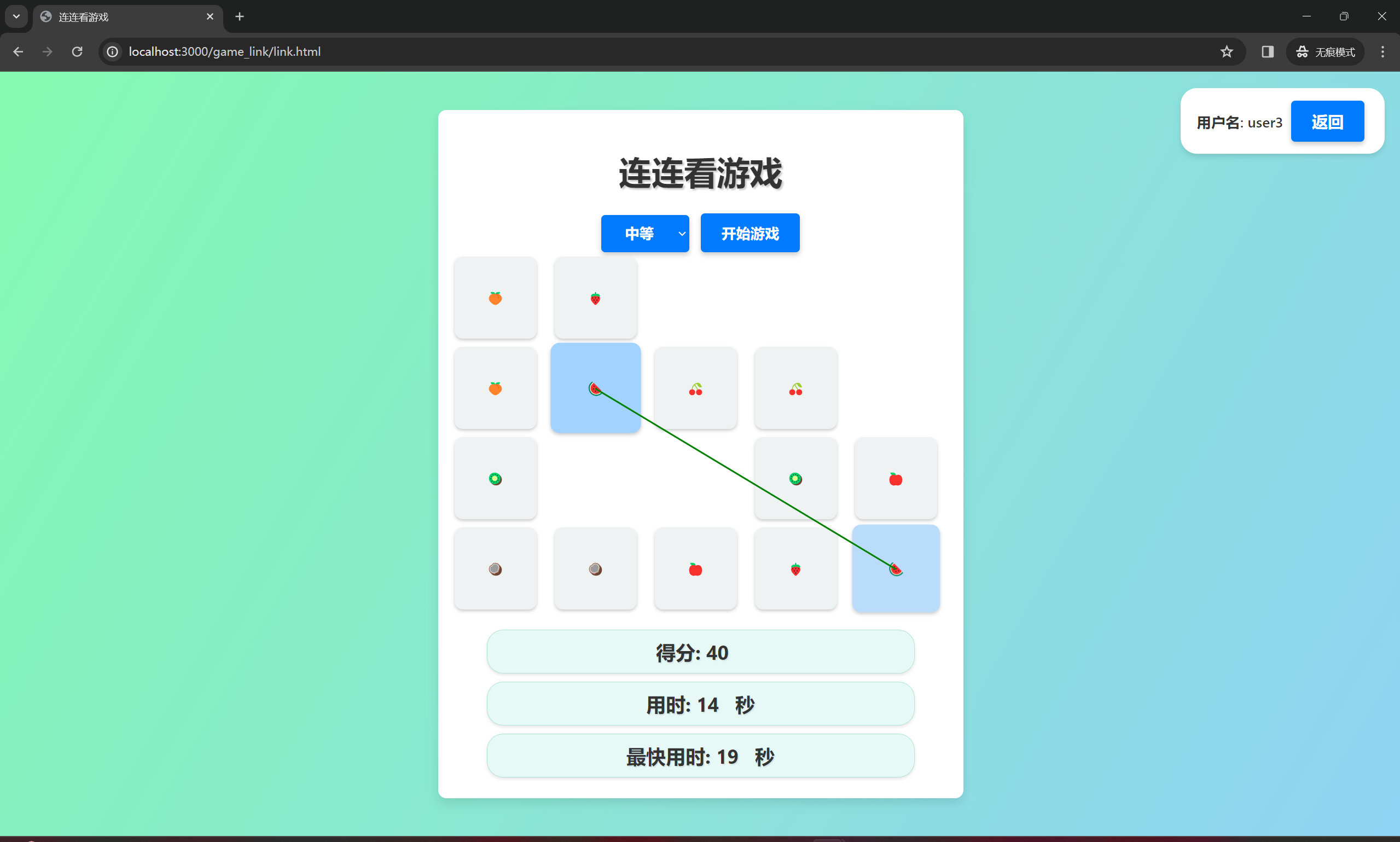


1. 贪吃蛇游戏



玩家通过键盘“上下左右”控制蛇的移动，使得蛇能够吃到果子，蛇每吃到一次果子长度都加一，并且得分加一，会播放eat.mp3音效；蛇自己咬到自己则视作死亡，游戏重新开始，分数归0。该游戏会统计玩家历史最高得分，玩家若超过高得分，则会刷新纪录。

1. 连连看游戏



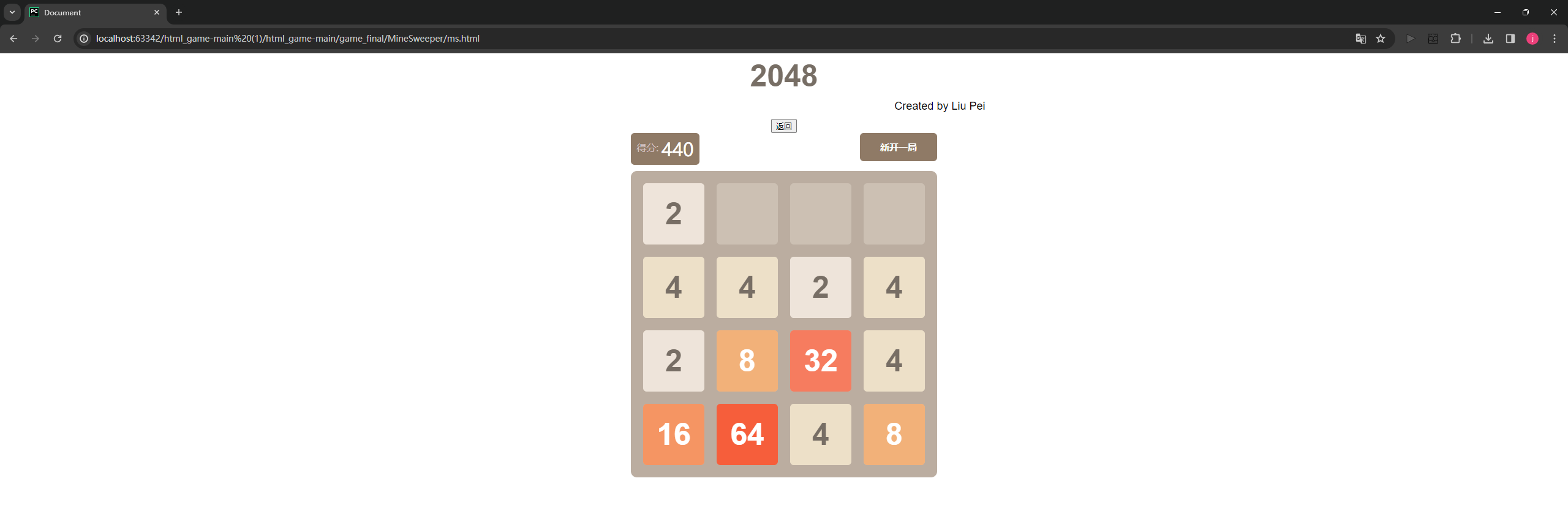


玩家可以选择3种难度，简单、中等、困难，随着难度的增加，地图会不断变大。每正确连接2个相同的方块，会得10分，成功连接会显示绿色的线，并播放match.mp3音效；失败连接（选择2个不相同的方块），会显示红色的线，并播放mismatch.mp3音效；成功连接完所有方块会播放victory.mp3音效。

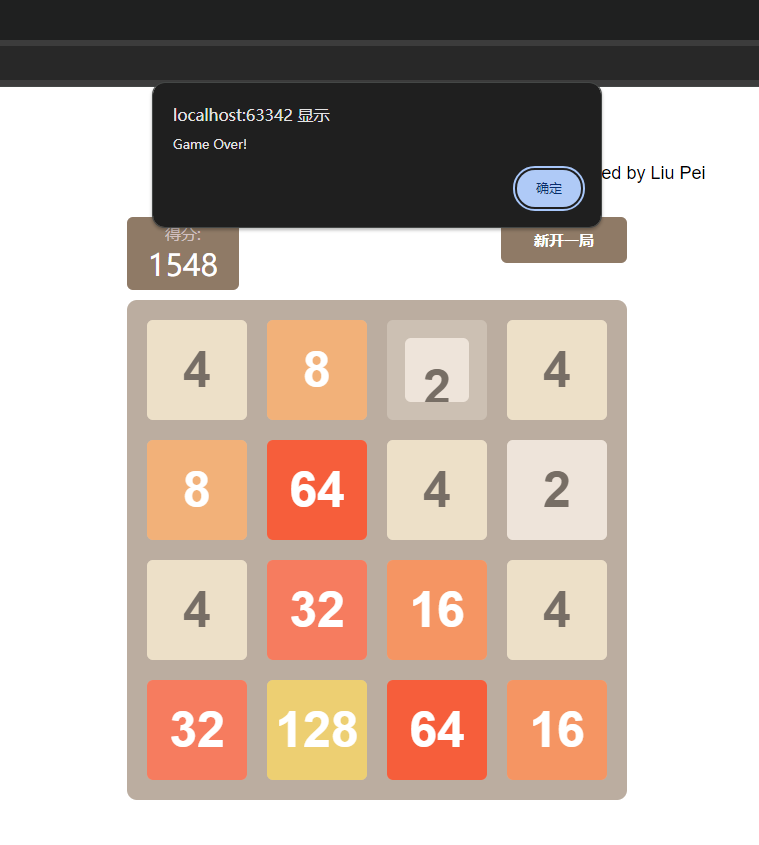
游戏会统计玩家最快用时，玩家在下一次游玩若更快完成，会刷新纪录。

（7）2048游戏

从游戏选择页面选择2048，开始游戏，输入键盘的方向键，将相同数字合并



在填满之前尽可能合并出更大的数字，达到2048即可通关，未能达到则game over,左上角为当前分数，点击新开一局重新游戏，点击返回进入游戏选择页面。



# **总结**

我们组的项目有较高的可拓展性，由于前后端分离做得比较清晰，如果想添加新的游戏，只需要完成js,html,css三件套之后，在game\_selection.html中添加对应的链接按钮即可。

在技术方面，我们采用了HTML、CSS和JavaScript作为主要的前端技术，结合Node.js和Express框架搭建后端服务。使用MongoDB作为数据库存储用户信息，利用bcrypt进行密码加密，确保用户信息安全。此外，通过RESTful API设计和AJAX技术实现了前后端的高效通信。这些技术的结合不仅提升了系统的性能和用户体验，也确保了项目的可扩展性和可维护性。

我们的项目还有一些地方可以改进，（1）可以把项目放到云端上，不需要买域名，通过公网ip即可完成完成访问，这一点不难，可惜时间有点紧张就没完成。这样可以解决其他人从github下载后，想运行项目还得配置nodejs和MongoDB的问题。

（2）数据库设计还是比较简单，可以增加一个登录状态的布尔值，若用户已经登录就设为true，其他人若想再登录这个账号就会被服务器拒绝；还可以对每个用户设计最高分和最短时间，这样显示游戏纪录时，可以显示创造纪录对应的用户，增强玩家之间的互动性。