《模拟桌游》系统架构设计说明书

**课题名称： 模拟桌游网页游戏**

小 组： 第1组

组 长： 谢子渊

组 员： 蔡祺辉

李欢

冉启航

指导教师： 杨波

**评阅成绩：**

提交报告时间：2025年4月日

目录

**[《模拟桌游》系统架构设计说明书 1](#_Toc880)**

**[一、 介绍 3](#_Toc3570)**

[1.1 编制目的 3](#_Toc8663)

[1.2 适用范围 3](#_Toc8281)

[1.4 定义和缩略语 3](#_Toc14577)

**[二、 系统概览 3](#_Toc10944)**

[2.1 系统功能 3](#_Toc31419)

[2.2 系统架构 3](#_Toc20996)

[2.3 技术选型 4](#_Toc4138)

[2.4 运行环境 4](#_Toc31912)

[2.5 用户界面 4](#_Toc10638)

**[三、 系统结构 4](#_Toc22061)**

[3.1 结构设计 4](#_Toc22287)

[3.2 分解描述 6](#_Toc7708)

**[四、 数据设计 12](#_Toc4618)**

[4.1 数据描述 12](#_Toc10352)

[4.2 数据字典 13](#_Toc26200)

**[五、 组件设计 15](#_Toc25158)**

[5.1 用户管理系统 15](#_Toc25280)

[5.2 游戏管理系统 16](#_Toc1264)

[5.3 UNO游戏逻辑系统 17](#_Toc29233)

[5.4 海龟汤游戏逻辑系统 18](#_Toc13038)

[5.5 AI处理系统 18](#_Toc2920)

[5.6 通信管理系统 19](#_Toc21844)

[5.7 数据管理系统 19](#_Toc21299)

**[六、 用户界面设计 19](#_Toc1439)**

[6.2.1 游戏选择界面 20](#_Toc28022)

[6.2.2 UNO创建/加入房间界面 20](#_Toc28432)

[6.2.3 UNO房间准备界面 20](#_Toc23443)

[6.2.4 UNO游戏界面 20](#_Toc11269)

[6.2.5 UNO游戏结算界面 20](#_Toc31470)

[6.2.6 海龟汤选择列表界面 20](#_Toc1766)

[6.2.7 海龟汤汤面详情页 21](#_Toc3225)

[6.2.8 海龟汤游戏界面 21](#_Toc922)

**[七、 需求矩阵 21](#_Toc14200)**

1. **介绍**

**1.1 编制目的**

本软件架构设计说明书旨在为《模拟桌游》项目的开发团队提供详细的架构设计指导，确保开发过程的顺利进行和最终产品的质量。通过本说明书，开发人员能够清晰了解系统的整体架构、模块划分、接口设计以及数据流等关键信息，从而高效地进行代码实现和系统集成。

**1.2 适用范围**

本说明书适用于《模拟桌游》项目的开发、测试和维护阶段。它为开发人员提供了系统架构的详细设计，为测试人员提供了测试策略和方法的参考，同时也为项目管理人员提供了项目进度和质量控制的依据。本说明书主要面向以下角色：

开发人员：负责系统的代码实现和模块开发。

测试人员：负责系统的功能测试、性能测试和安全测试。

项目管理人员：负责项目的进度管理和质量控制。

**1.4 定义和缩略语**

**1.4.1 UNO**

UNO游戏开始时，每位玩家随机获得7张牌，牌面朝上置于手中，同时在牌堆上方翻开一张牌作为起始牌，形成公共牌堆。玩家需根据公共牌堆顶牌的颜色或数字匹配出自己的手牌，若无法匹配则需从公共牌堆抽牌。特殊功能牌包括“跳过”、“反转”、“+2”等，使用后会触发相应效果。当某位玩家出完手牌时，游戏结束，该玩家获胜。若游戏过程中无人出完牌，则继续进行，直至某位玩家获胜。

**1.4.2 海龟汤**

在海龟汤游戏中，玩家从一系列神秘的汤面中选择一个开始游戏，每个汤面背后都隐藏着一个独特的故事。游戏过程中，玩家通过提问来探索和还原汤面背后的故事内容，但提问必须遵循特定规则：只能针对汤面相关的内容提问，且不能直接询问汤底（即故事的核心谜底）。AI会根据问题与汤底的相关性，给出“是”“否”或“无关”的回答，帮助玩家逐步缩小范围、推理出完整的故事。当玩家成功还原出汤底时，游戏结束。

1. **系统概览**

《模拟桌游》是一款集成了UNO和海龟汤两款经典小游戏的多人在线网页游戏，旨在为用户提供一个便捷、有趣的在线桌游平台，支持多平台浏览器访问。

**2.1 系统功能**

系统主要包含以下功能：

配置管理：管理用户账户信息，包括支持用户注册、登录、个人信息修改和房间历史查询；支持系统根据用户的选择流畅地在UNO和海龟汤模块间切换。

UNO游戏：用户可以创建或加入UNO游戏房间，进行实时对战；游戏支持标准UNO规则和AI玩家加入模式。

海龟汤游戏：用户可以选择想玩的汤面，由AI主持游戏过程，通过提问逐步还原汤底。

**2.2 系统架构**

系统采用分层架构设计，主要包括以下部分：

表现层：构建用户界面，实现响应式设计与游戏中的动态效果。

业务逻辑层：实现用户注册、登录、个人信息修改和房间历史查询；实现UNO和海龟汤游戏的核心规则；处理玩家之间的实时通信，确保游戏流畅性和实时性。

数据访问层：存储用户账户信息等结构化数据、海龟汤游戏的汤面信息和AI回答规则和游戏缓存数据等。

基础设施层：实现WebSocket通信，确保玩家之间的低延迟双向通信；处理高并发游戏事件，确保游戏的稳定性。

**2.3 技术选型**

**2.3.1 前端**

开发框架：Vue.js

状态管理：Vuex

样式框架：SCSS,Tailwind CSS

交互与动画：Vue Transitions,GSAP

通信与实时工具：Axios,Socket.io-client

**2.3.2 后端**

语言与框架：使用Golang语言，Gin框架和gorilla/websocket。

数据库：MySQL,Redis,NSQ或RabbitMQ消息队列。

**2.3.3 测试工具**

Jest,Vue Test Utils或Cypress。

**2.4 运行环境**

浏览器：兼容主流浏览器，包括Chrome、Firefox、Safari和Edge。

设备：支持桌面设备和移动设备，确保在不同设备上的良好用户体验。

**2.5 用户界面**

系统界面设计简洁直观，参考同类型网页游戏的UI风格设计，确保用户能够轻松上手。游戏大厅提供游戏列表和房间管理功能，游戏房间内提供实时互动和操作提示，确保用户在游戏过程中的流畅体验。

1. **系统结构**

**3.1 结构设计**

《模拟桌游》的系统架构采用模块化设计，将系统的功能划分为多个高内聚、低耦合的子系统，以实现系统的完整功能。以下是主要的子系统及其职责：

**3.1.1 用户管理系统**

该系统负责用户账户的管理，包括注册、登录、个人信息修改和房间历史查询。

其与通信管理系统协作，通过WebSocket实时通知用户登录状态和房间信息。

**3.1.2 游戏管理系统**

该系统负责游戏房间的创建、加入和管理，包括房间状态的维护和游戏的初始化。

其与通信管理系统协作，实时同步房间状态和游戏信息给所有玩家。

**3.1.3 UNO游戏逻辑系统**

该系统实现UNO游戏的核心规则，包括发牌、出牌验证、特殊牌处理、回合管理和胜负判定等。

其与游戏管理系统协作，接收游戏初始化请求并返回游戏状态；与通信管理系统协作，实时同步游戏状态给所有玩家。

**3.1.4 海龟汤游戏逻辑系统**

该系统实现海龟汤游戏的核心规则，包括汤面选择、用户提问、AI回答。

其与数据管理系统协作，从数据库中拉取文本显示和AI支持所需要的数据；与AI处理系统协作，获取AI回答与游戏结束判定。

**3.1.5 AI处理系统**

该系统为UNO和海龟汤游戏提供AI支持，分别包括参与UNO游戏，根据场上状态判断出牌和根据用户提问生成回答并判断游戏是否结束。

其与UNO游戏逻辑系统协作，处理场上游戏信息并返回AI玩家游戏回合操作；与海龟汤游戏逻辑系统协作，处理用户提问并返回AI回答。

**3.1.6 通信管理系统**

该系统负责玩家与服务器之间的实时通信，确保低延迟双向通信。

其与用户管理系统、游戏管理系统、UNO游戏逻辑系统和AI处理系统协作，实时同步用户和游戏状态。

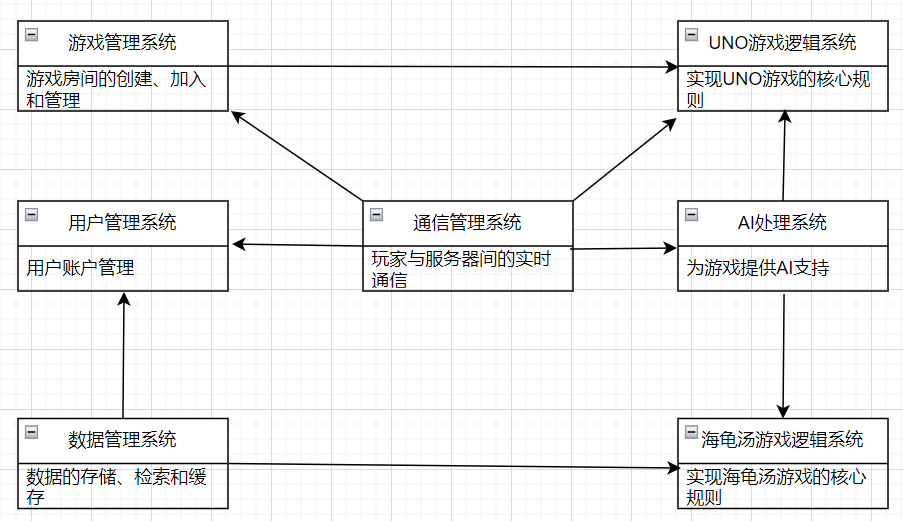
**3.1.7 数据管理系统**

该系统负责数据的存储、检索和缓存，包括用户数据和海龟汤游戏数据。

其与用户管理系统、游戏管理系统和海龟汤游戏逻辑系统协作，提供数据支持。

**3.1.8 系统架构图**

以下是系统架构的示意图，展示了主要的子系统及其相互关系。



**3.2 分解描述**

**3.2.1 用户管理系统**

用户管理子系统负责处理用户账户相关的所有操作，包括用户注册、登录、个人信息修改和房间历史查询。

**3.2.1.1 功能分解**

**3.2.1.1.1 用户注册模块**

功能：处理新用户的注册请求，包括验证用户输入的信息（如用户名、密码、邮箱等）是否合法。

数据流：接收前端传来的用户注册信息，验证后存储到数据库中，并返回注册结果。

接口：提供注册接口，接收用户注册信息并返回注册结果。

**3.2.1.1.2 用户登录模块**

功能：验证用户的登录信息（如用户名和密码），生成会话标识符。

数据流：接收前端传来的用户登录信息，验证后生成会话标识符并返回给前端。

接口：提供登录接口，接收用户登录信息并返回会话标识符。

**3.2.1.1.3 个人信息修改模块**

功能：允许用户修改个人信息，如昵称、头像等。

数据流：接收前端传来的用户修改信息，更新数据库中的用户信息，并返回更新结果。

接口：提供个人信息修改接口，接收用户修改信息并返回更新结果。

**3.2.1.1.4 房间历史查询模块**

功能：查询用户曾进入过的游戏房间历史记录。

数据流：接收前端传来的用户查询请求，从数据库中检索房间历史记录，并返回查询结果。

接口：提供房间历史查询接口，接收用户查询请求并返回查询结果。

**3.2.1.2 接口规范**

**3.2.1.2.1 用户注册接口**

输入参数：用户注册所用的信息

输出结果：注册成功或失败的布尔值消息

**3.2.1.2.2 用户登录接口**

输入参数：用户名、密码

输出结果：会话标识符及登录成功或失败的布尔值消息

**3.2.1.2.3 个人信息修改接口**

输入参数：用户ID、修改信息

输出结果：更新成功或失败的布尔值消息

**3.2.1.2.4 房间历史查询接口**

输入参数：用户ID

输出结果：房间历史记录列表

**3.2.2 游戏管理系统**

游戏管理系统负责处理玩家的游戏选择和UNO游戏房间相关操作，包括房间的创建、加入、状态管理和游戏初始化。

**3.2.2.1 功能分解**

**3.2.2.1.1 房间创建模块**

功能：处理房主创建游戏房间的请求，生成唯一的房间号，并将房间信息存储在服务器内存中。

数据流：接收前端传来的房间创建请求，生成房间号，存储房间信息，并返回房间号给前端。

接口：提供房间创建接口，接收房间创建请求并返回房间号。

**3.2.2.1.2 房间加入模块**

功能：验证玩家加入房间的请求，检查房间号的有效性，并将玩家信息添加到房间中。

数据流：接收前端传来的房间加入请求，验证房间号，更新房间状态，并返回加入结果。

接口：提供房间加入接口，接收房间加入请求并返回加入结果。

**3.2.2.1.3 房间状态管理模块**

功能：管理房间的状态，包括玩家列表、游戏是否开始等，并实时同步房间状态给所有玩家。

数据流：接收前端传来的状态更新请求，更新房间状态，并通过通信管理子系统实时同步状态给所有玩家。

接口：提供房间状态更新接口，接收状态更新请求并返回更新结果。

**3.2.2.1.4 游戏初始化模块**

功能：根据房间类型（UNO或海龟汤）初始化对应的游戏逻辑，包括调用相应的游戏逻辑子系统进行游戏设置。

数据流：接收游戏管理子系统的初始化请求，与UNO游戏逻辑子系统或海龟汤游戏逻辑子系统协作，完成游戏初始化，并返回初始化结果。

接口：提供游戏初始化接口，接收初始化请求并返回初始化结果。

**3.2.2.2 接口规范**

**3.2.2.2.1 房间创建接口**

输入参数：房主相关信息

输出结果：房间号及创建成功或失败的信息

**3.2.2.2.2 房间加入接口**

输入参数：房间号、玩家相关信息

输出结果：加入成功或失败的布尔值消息

**3.2.2.2.3 房间状态更新接口**

输入参数：房间号、新的状态信息

输出结果：更新成功或失败的布尔值消息

**3.2.2.2.4 游戏初始化接口**

输入参数：房间号

输出结果：初始化成功或失败的布尔值消息

**3.2.3 UNO游戏逻辑系统**

UNO游戏逻辑系统负责实现UNO游戏的核心规则和游戏流程，包括发牌、出牌验证、特殊牌处理、回合管理和胜负判定。

**3.2.3.1 功能分解**

**3.2.3.1.1 发牌模块**

功能：根据UNO游戏规则和场上玩家数量，随机为每个玩家发牌。

数据流：接收游戏初始化请求，生成随机牌堆，为每个玩家分配初始手牌，并返回初始牌局状态。

接口：提供发牌接口，接收游戏初始化请求并返回初始牌局状态。

**3.2.3.1.2 出牌验证模块**

功能：验证玩家的出牌是否符合当前游戏规则，包括颜色和数字匹配。

数据流：接收玩家出牌请求，验证出牌合法性，返回验证结果。

接口：提供出牌验证接口，接收玩家出牌请求并返回验证结果。

**3.2.3.1.3 特殊牌处理模块**

功能：处理玩家打出的特殊牌（如跳过、反转、+2等），根据规则改变游戏状态。

数据流：接收特殊牌出牌请求，更新游戏状态（如出牌顺序、当前花色等），返回更新后的游戏状态。

接口：提供特殊牌处理接口，接收特殊牌出牌请求并返回更新后的游戏状态。

**3.2.3.1.4 回合管理模块**

功能：管理游戏的回合流程，包括提示当前出牌玩家、检测可出牌、处理回合结束逻辑。

数据流：接收回合开始请求，激活当前玩家操作，检测可出牌，处理回合结束逻辑，更新牌堆顶显示，广播回合结果。

接口：提供回合管理接口，接收回合相关请求并返回处理结果。

**3.2.3.1.5 胜负判定模块**

功能：判断游戏是否结束，确定胜利玩家，并提供“再来一局”和“退出房间”选项。

数据流：接收游戏状态更新，判断胜负条件，返回胜负结果和“再来一局”和“退出房间”选项。

接口：提供胜负判定接口，接收游戏状态更新并返回胜负结果。

**3.2.3.2 接口规范**

**3.2.3.2.1 发牌接口**

输入参数：玩家数量

输出结果：初始牌局状态（包括每个玩家的手牌和公共牌堆）

**3.2.3.2.2 出牌验证接口**

输入参数：当前出牌玩家信息、出牌信息

输出结果：更新后的游戏状态

**3.2.3.2.3 特殊牌处理接口**

输入参数：当前出牌玩家信息、特殊牌信息

输出结果：更新后的游戏状态

**3.2.3.2.4 回合管理接口**

输入参数：当前出牌玩家信息、操作类型

输出结果：更新后的游戏状态

**3.2.3.2.5 胜负判定接口**

输入参数：当前游戏状态

输出结果：胜负结果

**3.2.4 海龟汤游戏逻辑系统**

海龟汤游戏逻辑系统负责实现海龟汤游戏的核心规则和游戏流程，包括汤面选择、用户提问、AI回答和游戏结束判定。

**3.2.4.1 功能分解**

**3.2.4.1.1 汤面选择模块**

功能：从题库中选择一个汤面，作为游戏的初始情境描述。

数据流：接收游戏初始化请求，从题库中选择一个汤面，并返回汤面信息。

接口：提供汤面选择接口，接收游戏初始化请求并返回汤面信息。

**3.2.4.1.2 用户提问模块**

功能：处理玩家的提问请求。

数据流：接收玩家提问，传递给AI处理模块，返回提问结果。

接口：提供用户提问接口，接收玩家提问并返回提问结果。

**3.2.4.1.3 AI回答模块**

功能：根据玩家提问，生成“是”“否”“无关”或“是也不是”的回答。

数据流：接收玩家提问，根据汤底规则生成回答，返回回答结果。

接口：提供AI回答接口，接收玩家提问并返回回答结果。

**3.2.4.1.4 游戏结束判定模块**

功能：根据AI的判定结果，结束游戏并返回完整汤底。

数据流：接收AI的判定结果，返回游戏结束结果。

接口：提供游戏结束判定接口，接收游戏状态更新并返回游戏结束结果。

**3.2.4.2 接口规范**

**3.2.4.2.1 汤面选择接口**

输入参数：汤面对应的唯一标识符

输出结果：汤面完整信息

**3.2.4.2.2 用户提问接口**

输入参数：玩家提问内容

输出结果：提问是否成功的布尔值消息

**3.2.4.2.3 AI回答接口**

输入参数：玩家提问内容

输出结果：AI回答结果

**3.2.4.2.4 游戏结束判定接口**

输入参数：AI判定的游戏是否结束的结果

输出结果：游戏结束的消息、完整汤底信息

**3.2.5 AI处理系统**

AI处理系统负责参与UNO游戏和主持海龟汤游戏。

**3.2.5.1 功能分解**

**3.2.5.1.1 AI UNO玩家模块**

功能：参与UNO游戏，根据既定的游戏规则和当前牌局状态进行出牌决策。

数据流：接收当前牌局状态，返回出牌决策。

接口：提供AI出牌接口，接收当前牌局状态并返回出牌决策。

**3.2.5.1.2 AI海龟汤主持模块**

功能：主持海龟汤游戏，根据玩家提问和海龟汤相关信息，生成回答结果和游戏结束判定结果。

数据流：接收玩家提问和海龟汤相关信息，根据既定规则生成回答，返回回答结果和游戏结束判定结果。

接口：提供AI回答接口，接收玩家提问和海龟汤相关信息并返回回答结果和游戏结束判定结果。

**3.2.5.2 接口规范**

**3.2.5.2.1 AI出牌接口（UNO）**

输入参数：当前牌局状态（包括公共牌堆顶牌、玩家手牌、其他玩家手牌数等）

输出结果：AI的出牌决策

**3.2.5.2.2 AI回答接口（海龟汤）**

输入参数：海龟汤信息、提问内容

输出结果：回答结果和游戏结束判定结果

**3.2.6 通信管理系统**

通信管理系统负责玩家与服务器之间的实时通信，确保低延迟双向通信。

**3.2.6.1 功能分解**

**3.2.6.1.1 WebSocket连接管理模块**

功能：建立和维护用户与服务器之间的WebSocket连接，确保实时通信的稳定性。

数据流：接收用户的连接请求，建立WebSocket连接，维护连接状态，处理连接断开。

接口：提供WebSocket连接接口，接收用户连接请求并返回连接状态。

**3.2.6.1.2 消息分发模块**

功能：接收来自用户的消息，根据消息类型分发到相应的处理模块。

数据流：接收消息，解析消息类型，将消息分发到相应的处理模块。

接口：提供消息分发接口，接收消息并返回分发结果。

**3.2.6.1.3 消息格式转换模块**

功能：将消息转换为统一的格式，确保消息在不同模块之间的兼容性。

数据流：接收原始消息，转换消息格式，返回转换后的消息。

接口：提供消息格式转换接口，接收原始消息并返回转换后的消息。

**3.2.6.1.4 通信安全模块**

功能：确保通信过程中的数据安全，包括数据加密和身份验证。

数据流：接收消息，加密消息，验证身份，返回安全的消息。

接口：提供通信安全接口，接收消息并返回加密后的消息。

**3.2.6.2 接口规范**

**3.2.6.2.1 WebSocket连接接口**

输入参数：用户信息、连接信息

输出结果：连接状态

**3.2.6.2.2 消息分发接口**

输入参数：消息内容、消息类型

输出结果：分发结果

**3.2.6.2.3 消息格式转换接口**

输入参数：原始消息

输出结果：转换后的消息

**3.2.6.2.4 通信安全接口**

输入参数：消息内容

输出结果：加密后的消息

**3.2.7 数据管理系统**

数据管理系统负责处理系统的数据存储、检索和缓存，确保数据的一致性和高效访问。

**3.2.7.1 功能分解**

**3.2.7.1.1 用户数据管理模块**

功能：管理用户账户信息和房间历史记录。

数据流：接收用户数据的增删改查请求，存储和检索用户数据。

接口：提供用户数据管理接口，接收用户数据操作请求并返回操作结果。

**3.2.7.1.2 海龟汤数据管理模块**

功能：管理海龟汤游戏的相关信息和AI回答规则。

数据流：接收相关数据的增删改查请求，存储和检索相关数据。

接口：提供相关数据管理接口，接收相关数据操作请求并返回操作结果。

**3.2.7.1.3 数据安全与备份模块**

功能：确保数据的安全性和完整性，定期备份数据。

数据流：接收数据备份和恢复请求，执行备份和恢复操作。

接口：提供数据安全与备份接口，接收数据操作请求并返回操作结果。

**3.2.7.2 接口规范**

**3.2.7.2.1 用户数据管理接口**

输入参数：用户ID、操作类型（增删改查）、数据内容

输出结果：操作结果（成功或失败）

**3.2.7.2.2 海龟汤数据管理接口**

输入参数：汤面ID、操作类型（增删改查）、数据内容

输出结果：操作结果（成功或失败）

**3.2.7.2.3 数据安全与备份接口**

输入参数：操作类型（备份或恢复）

输出结果：操作结果（成功或失败）

1. **数据设计**

**4.1 数据描述**

**4.1.1 用户数据**

用户数据存储在关系型数据库中，包含用户的基本信息和账户设置。结构如下：

用户ID（主键，整数类型）：唯一标识用户。

用户名（字符串类型）：用户注册时选择的用户名。

密码（字符串类型）：用户密码，加密存储。

昵称（字符串类型）：用户自定义的昵称。

头像（字符串类型）：用户上传的头像URL。

邮箱（字符串类型）：用户的邮箱地址。

个人简介（文本类型）：用户自定义的个人简介。

注册时间（时间戳类型）：用户注册的时间。

**4.1.2 游戏房间数据**

游戏房间数据存储在服务器内存中，包含房间的基本信息和状态。游戏房间表的结构如下：

房间ID（主键，整数类型）：唯一标识房间。

房间类型（枚举类型）：房间类型，如“UNO”或“海龟汤”。

房主ID（整数类型）：房主的用户ID。

玩家列表（JSON类型）：房间内玩家的ID列表。

游戏状态（枚举类型）：房间的游戏状态，如“等待中”或“进行中”。

创建时间（时间戳类型）：房间创建的时间。

**4.1.3 UNO牌局数据**

UNO牌局数据存储在服务器内存中，包含牌局的详细信息。UNO牌局表的结构如下：

牌局ID（主键，整数类型）：唯一标识牌局。

房间ID（整数类型）：对应的房间ID。

公共牌堆（JSON类型）：公共牌堆的牌列表。

玩家手牌（JSON类型）：每个玩家的手牌列表。

当前玩家ID（整数类型）：当前出牌的玩家ID。

出牌顺序（枚举类型）：出牌顺序，如“顺时针”或“逆时针”。

当前花色（字符串类型）：当前需要匹配的花色。

游戏状态（枚举类型）：牌局的游戏状态，如“进行中”或“已结束”。

**4.1.4 海龟汤数据**

海龟汤数据存储在关系型数据库中，包含海龟汤的详细信息。海龟汤表的结构如下：

汤面ID（主键，整数类型）：唯一标识汤面。

汤面描述（文本类型）：汤面的描述内容。

汤底答案（文本类型）：汤面的完整答案。

创建时间（时间戳类型）：汤面创建的时间。

**4.1.5 缓存数据**

缓存数据存储在Redis中，用于提升系统性能。缓存数据包括：

会话信息：存储用户的会话信息，如JWT Token。

实时游戏数据：存储实时游戏状态，如玩家手牌、公共牌堆顶牌等。

**4.2 数据字典**

**4.2.1 用户表**

| **字段名称** | **类型** | **描述** |
| --- | --- | --- |
| user\_id | 整数类型 | 用户的唯一标识符 |
| username | 字符串类型 | 用户的用户名 |
| password | 字符串类型 | 用户的加密密码 |
| nickname | 字符串类型 | 用户的昵称 |
| avatar\_url | 字符串类型 | 用户的头像URL |
| email | 字符串类型 | 用户的邮箱地址 |
| bio | 文本类型 | 用户的个人简介 |
| registration\_time | 时间戳类型 | 用户的注册时间 |

**4.2.2 游戏房间表**

| **字段名称** | **类型** | **描述** |
| --- | --- | --- |
| room\_id | 整数类型 | 房间的唯一标识符 |
| room\_type | 枚举类型 | 房间类型（如“UNO”或“海龟汤”） |
| owner\_id | 整数类型 | 房主的用户ID |
| player\_list | JSON类型 | 房间内玩家的ID列表 |
| game\_status | 枚举类型 | 房间的当前状态（如“等待中”或“进行中”） |
| creation\_time | 时间戳类型 | 房间的创建时间 |

**4.2.3 UNO牌局表**

| **字段名称** | **类型** | **描述** |
| --- | --- | --- |
| game\_id | 整数类型 | 牌局的唯一标识符 |
| room\_id | 整数类型 | 对应的房间ID |
| public\_deck | JSON类型 | 公共牌堆的牌列表 |
| player\_hands | JSON类型 | 每个玩家的手牌列表 |
| current\_player\_id | 整数类型 | 当前出牌的玩家ID |
| play\_direction | 枚举类型 | 出牌顺序（如“顺时针”或“逆时针”） |
| current\_color | 字符串类型 | 当前需要匹配的花色 |
| game\_status | 枚举类型 | 牌局的当前状态（如“进行中”或“已结束”） |

**4.2.4 海龟汤信息表**

| **字段名称** | **类型** | **描述** |
| --- | --- | --- |
| soup\_id | 整数类型 | 汤面的唯一标识符 |
| description | 文本类型 | 汤面的描述内容 |
| answer | 文本类型 | 汤面的完整答案 |
| creation\_time | 时间戳类型 | 汤面的创建时间 |

1. **组件设计**

**5.1 用户管理系统**

**5.1.1 用户注册**

function registerUser(username, password, email)

if 验证用户名和邮箱是否可用

password = 加密(password)

user\_id = 生成唯一用户ID

插入数据库: Users(user\_id, username, password, email, 默认昵称, 默认头像, 空简介, 当前时间)

返回: 注册成功

else

返回: 注册失败，用户名或邮箱已存在

**5.1.2用户登录**

function loginUser(username, password)

查询数据库: Users WHERE username = username

if 用户存在

if 验证密码正确

生成JWT Token

更新最后登录时间

返回: 登录成功，Token

else

返回: 登录失败，密码错误

else

返回: 登录失败，用户不存在

**5.1.3 个人信息修改**

function updateUserInfo(user\_id, new\_nickname, new\_avatar)

查询数据库: Users WHERE user\_id = user\_id

if 用户存在

更新数据库: Users SET nickname = new\_nickname, avatar\_url = new\_avatar WHERE user\_id = user\_id

返回: 更新成功

else

返回: 更新失败，用户不存在

**5.1.4 房间历史查询**

function queryRoomHistory(user\_id)

查询数据库: GameRooms WHERE player\_list 包含 user\_id

返回: 房间历史列表

**5.2 游戏管理系统**

**5.2.1 房间创建**

function createRoom(room\_type)

room\_id = 生成唯一房间ID

插入数据库: GameRooms(room\_id, room\_type, 当前用户ID, [当前用户ID], "等待中", 当前时间)

返回: room\_id

**5.2.2 房间加入**

function joinRoom(room\_id, player\_id)

查询数据库: GameRooms WHERE room\_id = room\_id

if 房间存在

if 房间状态为"等待中" 且 玩家数量未满

更新数据库: GameRooms SET player\_list = player\_list + [player\_id]

返回: 加入成功

else

返回: 加入失败，房间已满或游戏已开始

else

返回: 加入失败，房间不存在

**5.2.3 房间状态更新**

function updateRoomStatus(room\_id, new\_status)

查询数据库: GameRooms WHERE room\_id = room\_id

if 房间存在

更新数据库: GameRooms SET game\_status = new\_status

通过WebSocket推送状态更新给所有玩家

返回: 更新成功

else

返回: 更新失败，房间不存在

**5.2.4 游戏初始化**

function initializeGame(room\_id, game\_type)

查询数据库: GameRooms WHERE room\_id = room\_id

if 房间存在

if game\_type == "UNO"

调用UNO游戏逻辑子系统初始化牌局

else if game\_type == "海龟汤"

调用海龟汤游戏逻辑子系统初始化汤面

返回: 初始化结果

else

返回: 初始化失败，房间不存在

**5.3 UNO游戏逻辑系统**

**5.3.1 发牌**

function dealCards(player\_count)

初始化牌堆

洗牌

每个玩家分配7张牌

公共牌堆翻开一张牌

返回: 初始牌局状态

**5.3.2 出牌验证**

function validatePlay(current\_player\_hand, current\_top\_card, selected\_card)

if selected\_card.color == current\_top\_card.color 或 selected\_card.number == current\_top\_card.number 或 selected\_card.type == "万能"

返回: 合法

else

返回: 非法

**5.3.3 特殊牌处理**

function handleSpecialCard(selected\_card, current\_game\_state)

if selected\_card.type == "跳过"

跳过下一个玩家

else if selected\_card.type == "反转"

改变出牌顺序

else if selected\_card.type == "+2"

下一个玩家摸2张牌

更新游戏状态

返回: 更新后的游戏状态

**5.3.4 回合管理**

function manageTurn(current\_player\_id, action, card)

if action == "出牌"

if 验证出牌合法

移除玩家手牌

添加到公共牌堆

处理特殊牌效果

else

返回: 出牌失败

else if action == "摸牌"

从牌堆摸一张牌添加到玩家手牌

更新当前玩家ID

返回: 更新后的游戏状态

**5.3.5 胜负判定**

function determineWinner(player\_hands)

for 每个玩家的手牌 in player\_hands

if 手牌数量 == 0

返回: 该玩家胜利

返回: 游戏继续

**5.4 海龟汤游戏逻辑系统**

**5.4.1 汤面选择**

function selectSoup()

随机选择一个汤面ID

查询数据库: SoupFaces WHERE soup\_id = 汤面ID

返回: 汤面描述

**5.4.2 用户提问**

function submitQuestion(player\_id, question)

验证问题是否符合规则

添加问题到问题列表

返回: 提问成功

**5.4.3 AI回答**

function getAnswer(question, soup\_answer)

分析问题与汤底答案的相关性

if 相关性高

返回: "是"

else if 相关性中

返回: "否"

else

返回: "无关"

**5.4.4 游戏结束判定**

function checkGameEnd(player\_questions, soup\_answer)

分析玩家提问是否基本还原汤底

if 还原完成

返回: 游戏结束，完整汤底

else

返回: 游戏继续

**5.5 AI处理系统**

**5.5.1 AI出牌（UNO）**

function aiPlayCard(current\_game\_state)

分析当前牌局状态

选择最优出牌策略

返回: AI出牌决策

**5.5.2 AI回答（海龟汤）**

function aiAnswerQuestion(question)

分析问题与汤底的相关性

生成回答

返回: 回答结果

**5.6 通信管理系统**

**5.6.1 WebSocket连接**

function establishWebSocketConnection(player\_id)

创建WebSocket连接

存储连接状态

返回: 连接状态

**5.6.2 消息分发**

function dispatchMessage(message)

解析消息类型

根据类型调用相应处理模块

返回: 分发结果

**5.7 数据管理系统**

**5.7.1 用户数据管理**

function manageUserData(user\_id, operation, data)

if operation == "查询"

查询数据库: Users WHERE user\_id = user\_id

else if operation == "更新"

更新数据库: Users SET data WHERE user\_id = user\_id

返回: 操作结果

**5.7.2 游戏数据管理**

function manageGameData(game\_id, operation, data)

if operation == "查询"

查询数据库: UNOGames 或 SoupGames WHERE game\_id = game\_id

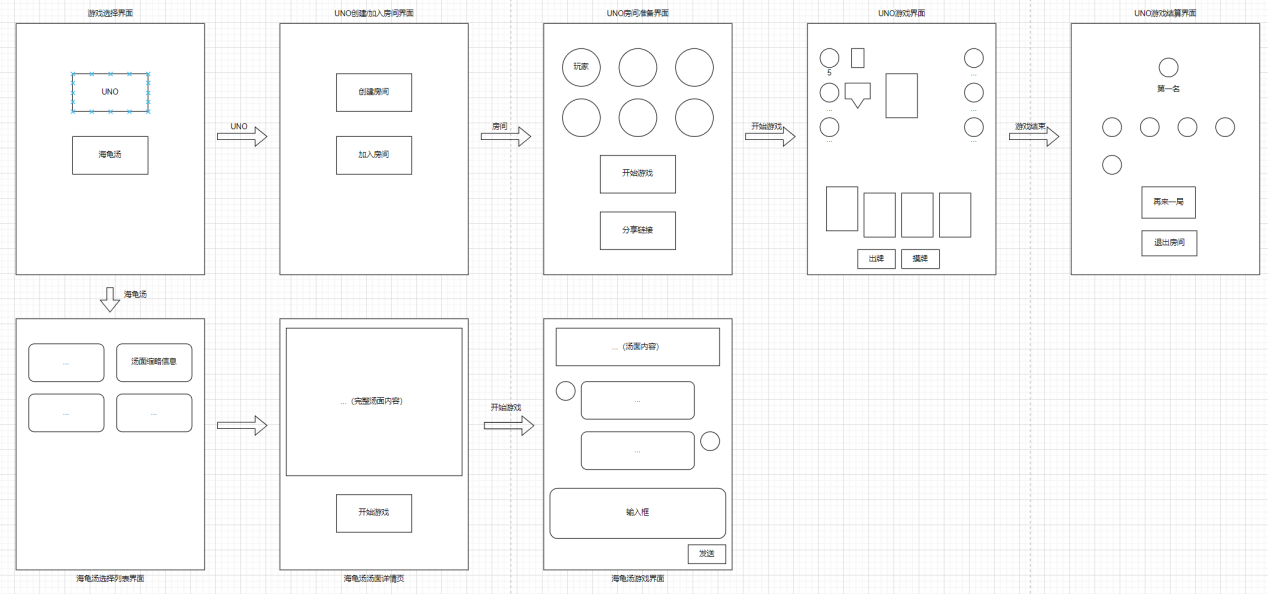
else if operation == "更新"

更新数据库: UNOGames 或 SoupGames SET data WHERE game\_id = game\_id

返回: 操作结果

1. **用户界面设计**

**6.1 用户界面原型图**



**6.2 各界面交互说明**

**6.2.1 游戏选择界面**

用户登录后进入的界面。用于选择要游玩的游戏。

“UNO”按键：点击后进入UNO创建/加入房间界面。

“海龟汤”按键：点击后进入海龟汤选择列表界面。

**6.2.2 UNO创建/加入房间界面**

选择游玩UNO游戏后进入的界面。用于选择创建新的UNO房间或加入他人的房间。

“创建房间”按键：创建新的房间，作为房主进入UNO房间准备界面。

“加入房间：按键：加入他人的房间，输入房间号后进入对应的UNO房间。

**6.2.3 UNO房间准备界面**

进入UNO游戏房间后的准备阶段所在的界面。

“开始游戏”按键：房主以当前房间成员开始游戏。

“分享链接”按键：生成游戏分享链接和当前房间号。

**6.2.4 UNO游戏界面**

UNO游戏开始后的界面。

在轮到对应玩家的回合时才可进行操作。

玩家可从自己的手牌中选中一张，在符合出牌规则的情况下点击“出牌”按键打出（如为特殊功能牌，还可选择其效果）并点击“出牌”按键，然后结束该回合。玩家也可选择“摸牌”，摸牌并结束该回合。

**6.2.5 UNO游戏结算界面**

当场上除最后一人外其余所有人都出完牌时进入的界面，显示游戏排名并提供游戏结束后的选项。

“再来一局”按键：回到原来的UNO房间准备界面。

“退出房间”按键：回到游戏选择界面。

**6.2.6 海龟汤选择列表界面**

显示所有的海龟汤缩略信息的界面。

玩家可点击其中一个缩略信息卡以进入海龟汤汤面详情页。

**6.2.7 海龟汤汤面详情页**

显示完整的海龟汤汤面信息的界面。

“开始游戏”按键：进入当前的海龟汤游戏界面。

**6.2.8 海龟汤游戏界面**

显示完整的海龟汤汤面信息和AI对话界面。

文本输入框：玩家在此处输入要询问AI的汤面相关的问题。

“发送”按键：当文本输入框内的内容发送给AI。

1. **需求矩阵**

**7.1 功能需求矩阵**

| **需求编号** | **需求名称** | **相关系统组件** |
| --- | --- | --- |
| SRS-0101 | 用户注册/登录 | 用户管理子系统 |
| SRS-0102 | 账号信息保存 | 用户管理子系统 |
| SRS-0103 | 个人信息修改 | 用户管理子系统 |
| SRS-0104 | 房间历史查询 | 用户管理子系统 |
| SRS-0105 | 游戏选择 | 游戏管理子系统 |
| SRS-0106 | 创建房间 | 游戏管理子系统 |
| SRS-0107 | 加入房间 | 游戏管理子系统 |
| SRS-0108 | 房间设置 | 游戏管理子系统 |
| SRS-0109 | AI加入模式 | 游戏管理子系统, AI处理子系统 |
| SRS-0110 | 发牌 | UNO游戏逻辑子系统 |
| SRS-0111 | 出牌验证 | UNO游戏逻辑子系统 |
| SRS-0112 | 特殊牌处理 | UNO游戏逻辑子系统 |
| SRS-0113 | 回合开始 | UNO游戏逻辑子系统 |
| SRS-0114 | 出牌阶段 | UNO游戏逻辑子系统 |
| SRS-0115 | 回合结束 | UNO游戏逻辑子系统 |
| SRS-0116 | 游戏结束 | UNO游戏逻辑子系统 |
| SRS-0117 | 再来一局选项 | UNO游戏逻辑子系统 |
| SRS-0118 | 退出游戏选项 | UNO游戏逻辑子系统 |
| SRS-0119 | 汤面推送 | 海龟汤游戏逻辑子系统 |
| SRS-0120 | 选择汤面 | 海龟汤游戏逻辑子系统 |
| SRS-0121 | 开始游戏 | 海龟汤游戏逻辑子系统 |
| SRS-0122 | 用户提问 | 海龟汤游戏逻辑子系统 |
| SRS-0123 | 游戏结束 | 海龟汤游戏逻辑子系统 |

**7.2 性能需求矩阵**

| **需求编号** | **需求名称** | **相关系统组件** |
| --- | --- | --- |
| SRS-0201 | 响应速度 | 通信管理子系统 |
| SRS-0202 | 并发能力 | 通信管理子系统 |
| SRS-0203 | 资源占用 | 前端 |

**7.3 非功能需求矩阵**

| **需求编号** | **需求名称** | **相关系统组件** |
| --- | --- | --- |
| SRS-0301 | 断线恢复 | 通信管理子系统 |
| SRS-0302 | 数据持久化 | 数据管理子系统 |
| SRS-0303 | 服务可用性 | 后端 |
| SRS-0304 | 数据安全 | 用户管理子系统, 数据管理子系统 |
| SRS-0305 | DDoS防护 | 后端 |
| SRS-0306 | 兼容性需求 | 前端 |