**学生饭卡管理系统详细设计**

**一、引言**

本详细设计文档是在学生饭卡管理系统概要设计的基础上，对系统各个模块的具体实现细节进行详细描述，包括类的设计、接口的实现、数据库操作、算法逻辑以及界面设计等方面，旨在为开发人员提供明确的编码指导和实现依据，确保系统能够按照预期的功能和性能要求进行开发。

**二、系统架构实现**

1. **技术选型**

后端开发采用 Java 语言，基于 Spring Boot 框架构建，利用其快速开发、自动配置等特性简化开发流程。

数据库选用 MySQL 关系型数据库存储数据，使用 MyBatis 作为数据持久层框架，方便与数据库进行交互操作。

前端采用 Vue.js 框架进行开发，结合 Element UI 组件库构建用户界面，实现良好的用户交互体验。

前后端通过 RESTful API 进行数据交互，采用 JSON 格式传输数据，确保数据传输的高效性和简洁性。

1. **项目结构**

**后端项目结构**：

src/main/java：存放 Java 源代码。

com.example.cardmanagementsystem.controller：控制器层，负责接收前端请求并调用相应的服务层方法处理请求，返回响应结果。

com.example.cardmanagementsystem.service：业务服务层，包含各个业务功能的实现逻辑，如饭卡开户、充值、消费等业务服务接口及其实现类。

com.example.cardmanagementsystem.dao：数据访问层接口，定义与数据库交互的方法，如查询学生信息、更新饭卡余额等操作的接口声明。

com.example.cardmanagementsystem.entity：实体类，对应数据库中的表结构，用于数据的传输和存储，如学生信息实体类、饭卡信息实体类等。

com.example.cardmanagementsystem.util：工具类，存放一些通用的工具方法，如密码加密工具、日期处理工具等。

src/main/resources：存放配置文件。

application.properties：Spring Boot 应用的配置文件，配置数据库连接信息、端口号、日志级别等参数。

mapper：存放 MyBatis 的 XML 映射文件，用于定义 SQL 查询语句和结果映射关系。

**前端项目结构**：

src：主要源代码目录。

assets：存放静态资源文件，如图片、样式文件等。

components：自定义组件目录，封装一些可复用的组件，如登录组件、饭卡信息展示组件等。

views：页面视图目录，每个页面对应一个 Vue 组件，如登录页面、饭卡管理页面、查询统计页面等。

router：路由配置文件，定义前端页面的路由规则，实现页面之间的导航。

store：Vuex 状态管理目录，用于管理全局共享的数据状态，如当前登录用户信息、饭卡余额等。

utils：前端工具函数目录，包含一些常用的工具函数，如请求封装函数、数据验证函数等。

**三、功能模块详细设计**

1. **用户登录模块**

**后端实现**：

在 LoginController 中定义登录接口方法 login，接收前端传来的用户名和密码参数。

在 UserService 中实现用户验证逻辑，根据用户名查询数据库中的用户信息，使用密码加密工具对输入密码进行加密后与数据库中存储的密码进行比对。

如果验证成功，生成一个 JWT（JSON Web Token）令牌，将用户信息和令牌返回给前端；如果验证失败，返回相应的错误信息。

**前端实现**：

在登录页面的 Vue 组件中，当用户点击登录按钮时，调用登录接口，将用户名和密码发送到后端。

接收后端返回的响应，如果登录成功，将 JWT 令牌存储在本地浏览器缓存（如 localStorage）中，并跳转到系统首页；如果登录失败，在页面上显示错误提示信息。

1. **饭卡管理模块**

**开户功能**：

**后端实现**：

在 CardController 中定义开户接口方法 openCard，接收前端传来的学生基本信息参数。

在 CardService 中实现开户逻辑，首先对学生信息进行合法性验证，如学号是否唯一、必填字段是否为空等。

如果验证通过，创建一个新的饭卡信息对象，设置初始余额（如 0 元）、卡状态为正常，并生成唯一的饭卡编号。

将学生信息和饭卡信息分别保存到数据库的学生信息表和饭卡信息表中，同时记录开户操作日志到日志表中。

返回开户成功的饭卡编号给前端。

**前端实现**：

在饭卡开户页面的 Vue 组件中，用户填写学生信息后点击开户按钮，调用开户接口，将信息发送到后端。

接收后端返回的饭卡编号，在页面上显示开户成功信息，并可选择是否跳转到饭卡信息详情页面查看新开饭卡的详细信息。

**充值功能**：

**后端实现**：

在 CardController 中定义充值接口方法 rechargeCard，接收饭卡编号和充值金额参数。

在 CardService 中实现充值逻辑，首先根据饭卡编号查询饭卡信息，验证饭卡是否存在且状态正常。

如果验证通过，计算充值后的余额，更新饭卡信息表中的余额字段，并在充值记录表中插入一条充值记录，包括充值时间、充值金额、操作员（可从登录用户信息中获取）等信息。

返回充值后的余额给前端。

**前端实现**：

在饭卡充值页面的 Vue 组件中，用户输入饭卡编号和充值金额后点击充值按钮，调用充值接口，将数据发送到后端。

接收后端返回的充值后余额，更新页面上显示的余额信息，并显示充值成功提示。

**消费功能**：

**后端实现**：

在 CardController 中定义消费接口方法 consumeCard，接收饭卡编号和消费金额参数。

在 CardService 中实现消费逻辑，根据饭卡编号查询饭卡信息，验证饭卡是否存在、状态正常且余额充足（余额大于等于消费金额）。

如果验证通过，扣除消费金额，更新饭卡信息表中的余额字段，并在消费记录表中插入一条消费记录，包括消费时间、消费地点（可从消费终端获取）、消费金额等信息。

返回消费后的余额给前端。

**前端实现**：

在食堂消费终端（可视为前端的一个特殊应用场景），通过刷卡设备读取饭卡编号，输入消费金额后调用消费接口，将数据发送到后端。

接收后端返回的消费后余额，更新消费终端显示的余额信息，并根据消费结果显示相应提示信息（如消费成功或余额不足等）。

**挂失解挂功能**：

**后端实现**：

在 CardController 中分别定义挂失接口方法 lostCard 和解挂接口方法 unlostCard，接收饭卡编号参数。

在 CardService 中实现挂失逻辑，根据饭卡编号查询饭卡信息，验证饭卡是否存在且未挂失，若满足条件则更新饭卡信息表中的卡状态为挂失，并记录挂失操作日志。

解挂逻辑类似，验证饭卡是否存在且已挂失，若满足条件则更新卡状态为正常，并记录解挂操作日志。

挂失或解挂操作成功后返回相应的成功信息给前端。

**前端实现**：

在饭卡挂失解挂页面的 Vue 组件中，用户输入饭卡编号后点击挂失或解挂按钮，调用相应的接口，将数据发送到后端。

接收后端返回的操作结果信息，在页面上显示提示信息，告知用户挂失或解挂是否成功。

1. **查询统计模块**

**个人查询功能**：

**后端实现**：

在 QueryController 中定义个人查询接口方法 queryStudentCardInfo，接收饭卡编号参数。

在 QueryService 中实现查询逻辑，根据饭卡编号查询饭卡信息表获取饭卡基本信息，查询充值记录表获取充值历史记录，查询消费记录表获取消费历史记录。

将查询到的信息封装成一个数据对象返回给前端。

**前端实现**：

在个人查询页面的 Vue 组件中，用户输入饭卡编号后点击查询按钮，调用查询接口，将数据发送到后端。

接收后端返回的查询结果数据，在页面上以列表或图表等形式展示饭卡基本信息、充值记录和消费记录，方便用户查看自己的饭卡使用情况。

**管理员查询功能**：

**后端实现**：

在 QueryController 中定义管理员查询接口方法 queryAllCardInfo，接收查询条件参数（如时间段、学号、班级等）。

在 QueryService 中实现查询逻辑，根据查询条件动态构建 SQL 查询语句，查询饭卡信息表、充值记录表和消费记录表获取相应的数据。

对查询到的数据进行统计分析，如计算总充值金额、总消费金额、各食堂消费占比等统计信息。

将查询结果数据和统计信息封装成一个数据对象返回给前端。

**前端实现**：

在管理员查询页面的 Vue 组件中，管理员设置查询条件后点击查询按钮，调用查询接口，将数据发送到后端。

接收后端返回的查询结果数据和统计信息，在页面上以表格、图表等形式展示所有饭卡的信息以及统计分析结果，方便管理员进行管理和决策。

1. **系统管理模块**

**数据备份与恢复功能**：

**后端实现**：

在 SystemController 中定义数据备份接口方法 backupData 和数据恢复接口方法 restoreData。

数据备份逻辑：使用 Java 的文件操作和数据库连接工具，将数据库中的数据以 SQL 脚本文件的形式导出到指定的备份目录下，备份文件命名可包含备份时间等信息，以便区分不同时间的备份。

数据恢复逻辑：读取指定的备份 SQL 脚本文件，使用数据库连接执行 SQL 语句，将数据恢复到数据库中。在恢复操作前，可先进行数据验证和提示操作，确保恢复操作的安全性和准确性。

备份和恢复操作成功后返回相应的成功信息给前端。

**前端实现**：

在系统管理页面的 Vue 组件中，设置数据备份和恢复的按钮，点击按钮时调用相应的接口，将请求发送到后端。

接收后端返回的操作结果信息，在页面上显示提示信息，告知管理员数据备份或恢复是否成功。

**系统参数设置功能**：

**后端实现**：

在 SystemController 中定义系统参数设置接口方法 setSystemParams，接收系统参数对象（包含饭卡有效期、消费限额、密码修改规则等参数）。

在 SystemService 中实现参数设置逻辑，将接收到的参数更新到系统配置表中。

返回参数设置成功信息给前端。

**前端实现**：

在系统管理页面的 Vue 组件中，设置系统参数的表单，管理员填写参数后点击保存按钮，调用参数设置接口，将数据发送到后端。

接收后端返回的成功信息，在页面上显示提示信息，告知管理员参数设置成功。

**四、数据库详细设计**

1. **数据库表结构**

**学生信息表（student\_info）**：

student\_id：VARCHAR (20)，主键，学号，唯一标识每个学生。

student\_name：VARCHAR (50)，学生姓名，不能为空。

class：VARCHAR (50)，班级信息。

gender：CHAR (1)，性别，取值为 'M'（男）或 'F'（女）。

password：VARCHAR (100)，密码，存储经过加密后的密码信息。



**饭卡信息表（card\_info）**：

card\_id：VARCHAR (20)，主键，饭卡编号，唯一标识每张饭卡。

student\_id：VARCHAR (20)，外键，关联 student\_info 表的 student\_id 字段，表明该饭卡所属的学生。

balance：DECIMAL (10, 2)，卡内余额，保留两位小数。

status：CHAR (10)，卡状态，取值为 'NORMAL'（正常）、'LOST'（挂失）、'CANCELLED'（注销）等。

open\_date：DATETIME，开户时间，记录饭卡开户的日期和时间。



**充值记录表（recharge\_record）**：

recharge\_id：VARCHAR (20)，主键，充值记录编号，唯一标识每条充值记录。

card\_id：VARCHAR (20)，外键，关联 card\_info 表的 card\_id 字段，表明该充值记录对应的饭卡。

amount：DECIMAL (10, 2)，充值金额，保留两位小数。

recharge\_date：DATETIME，充值时间，记录充值操作的日期和时间。

operator：VARCHAR (50)，操作员，记录进行充值操作的人员姓名或工号。



**消费记录表（consumption\_record）**：

consumption\_id：VARCHAR (20)，主键，消费记录编号，唯一标识每条消费记录。

card\_id：VARCHAR (20)，外键，关联 card\_info 表的 card\_id 字段，表明该消费记录对应的饭卡。

amount：DECIMAL (10, 2)，消费金额，保留两位小数。

consumption\_date：DATETIME，消费时间，记录消费操作的日期和时间。

consumption\_location：VARCHAR (100)，消费地点，记录消费发生的地点信息。



**系统配置表（system\_config）**：

config\_key：VARCHAR (50)，配置项键，唯一标识每个系统配置项，如 'CARD\_VALIDITY\_PERIOD'（饭卡有效期）、'CONSUMPTION\_LIMIT'（消费限额）、'PASSWORD\_CHANGE\_RULE'（密码修改规则）等。

config\_value：VARCHAR (100)，配置项值，存储对应配置项的值信息。

**操作日志表（operation\_log）**：

log\_id：VARCHAR (20)，主键，日志编号，唯一标识每条操作日志记录。

operation\_type：VARCHAR (50)，操作类型，如 'CARD\_OPEN'（饭卡开户）、'CARD\_RECHARGE'（饭卡充值）、'CARD\_CONSUME'（饭卡消费）、'CARD\_LOST'（饭卡挂失）、'CARD\_UNLOST'（饭卡解挂）等。

card\_id：VARCHAR (20)，饭卡编号，关联涉及的饭卡信息。

operation\_time：DATETIME，操作时间，记录操作发生的日期和时间。

operator：VARCHAR (50)，操作员，记录进行操作的人员姓名或工号。

1. **数据库索引设计**

在学生信息表的 student\_id 字段上创建唯一索引，提高根据学号查询学生信息的效率。

在饭卡信息表的 card\_id 字段上创建唯一索引，加快根据饭卡编号查询饭卡信息的速度。

在充值记录表的 card\_id 和 recharge\_date 字段上创建联合索引，便于按照饭卡编号和充值时间范围进行查询统计。

在消费记录表的 card\_id 和 consumption\_date 字段上创建联合索引，方便按饭卡编号和消费时间范围查询消费记录并进行统计分析。

**五、接口设计详细说明**

1. **登录接口（/api/v1/auth/login）**

**功能描述**：用于验证用户身份信息，返回 JWT 令牌以便后续接口请求的授权。

**请求方法**：POST

**请求参数**：

username（字符串类型，必填）：用户的账号，对于学生为学号，管理员为特定的管理账号。

password（字符串类型，必填）：用户密码，需进行加密传输。

**响应数据**：

成功登录：

{

"code": 200,

"message": "登录成功",

"data": {

"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV\_adQssw5c",

"user": {

"id": 12345,

"username": "2021001",

"role": "student",

"name": "张三"

}

}

}

登录失败：

{

"code": 401,

"message": "用户名或密码错误"

**（二）饭卡管理接口**

1. **饭卡充值接口（/api/v1/card/recharge）**

**功能描述**：为指定饭卡进行充值操作，更新饭卡余额并记录充值信息。

**请求方法**：POST

**请求参数**：

card\_id（字符串类型，必填）：要充值的饭卡编号，需确保该饭卡存在且状态正常。

amount（数值类型，必填）：充值金额，需大于 0。

operator（字符串类型，必填）：操作员信息，记录进行充值操作的人员姓名或工号，用于操作日志记录。

**响应数据**：

充值成功：

{

"code": 200,

"message": "充值成功",

"data": {

"card\_id": "FK20241213001",

"balance": 100.00 // 假设充值前余额为 0，充值了 100 元后的余额

}

}

充值失败：

{

"code": 400,

"message": "饭卡不存在"

}

1. **饭卡消费接口（/api/v1/card/consume）**

**功能描述**：处理饭卡消费操作，验证饭卡余额并扣除消费金额，记录消费信息。

**请求方法**：POST

**请求参数**：

card\_id（字符串类型，必填）：消费的饭卡编号，需确保该饭卡存在且状态正常。

amount（数值类型，必填）：消费金额，需大于 0 且小于等于饭卡当前余额。

consumption\_location（字符串类型，必填）：消费地点信息，如食堂窗口编号或具体位置描述，用于消费记录和统计分析。

**响应数据**：

消费成功：

{

"code": 200,

"message": "消费成功",

"data": {

"card\_id": "FK20241213001",

"balance": 50.00 // 假设消费前余额为 100 元，消费了 50 元后的余额

}

}

消费失败：

{

"code": 400,

"message": "饭卡余额不足"

}

1. **饭卡挂失接口（/api/v1/card/lost）**

**功能描述**：将指定饭卡设置为挂失状态，防止他人冒用。

**请求方法**：POST

**请求参数**：

card\_id（字符串类型，必填）：要挂失的饭卡编号，需确保该饭卡存在且未挂失。

operator（字符串类型，必填）：操作员信息，记录进行挂失操作的人员姓名或工号。

**响应数据**：

挂失成功：

{

"code": 200,

"message": "挂失成功"

}

挂失失败：

{

"code": 400,

"message": "饭卡已挂失"

}

1. **饭卡解挂接口（/api/v1/card/unlost）**

**功能描述**：将已挂失的饭卡恢复为正常状态，允许正常使用。

**请求方法**：POST

**请求参数**：

card\_id（字符串类型，必填）：要解挂的饭卡编号，需确保该饭卡存在且已挂失。

operator（字符串类型，必填）：操作员信息，记录进行解挂操作的人员姓名或工号。

**响应数据**：

解挂成功：

`json { "code": 200, "message": "解挂成功" } `

解挂失败：

``json { "code": 400, "message": "饭卡未挂失" } ```

**（三）查询接口**

1. **饭卡信息查询接口（/api/v1/card/info）**

**功能描述**：根据饭卡编号或学生学号查询饭卡的详细信息，包括饭卡余额、状态、开户时间等。

**请求方法**：GET

**请求参数**：

card\_id（字符串类型，可选）：饭卡编号，可根据此编号查询特定饭卡信息。

student\_id（字符串类型，可选）：学生学号，可根据此学号查询该学生对应的饭卡信息。至少提供一个参数，若两个参数都提供则优先使用 card\_id 进行查询。

**响应数据**：

查询成功：

{

"code": 200,

"message": "查询成功",

"data": {

"card\_id": "FK20241213001",

"student\_id": "2024001",

"balance": 100.00,

"status": "正常",

"open\_date": "2024-12-13T09:00:00Z"

}

}

查询失败：

{

"code": 400,

"message": "未找到对应的饭卡信息"

}

1. **充值记录查询接口（/api/v1/card/recharge/history）**

**功能描述**：查询指定饭卡的充值历史记录，可根据时间范围等条件进行筛选。

**请求方法**：GET

**请求参数**：

card\_id（字符串类型，必填）：要查询充值记录的饭卡编号。

start\_date（日期时间类型，可选）：查询开始时间，用于筛选指定时间范围后的充值记录。

end\_date（日期时间类型，可选）：查询结束时间，用于筛选指定时间范围前的充值记录。若只提供 start\_date，则查询该时间之后的所有充值记录；若只提供 end\_date，则查询该时间之前的所有充值记录；若两个时间都提供，则查询指定时间范围内的充值记录。

**响应数据**：

查询成功：

{

"code": 200,

"message": "查询成功",

"data": [

{

"recharge\_id": "RC20241213001",

"card\_id": "FK20241213001",

"amount": 100.00,

"recharge\_date": "2024-12-13T10:00:00Z",

"operator": "管理员001"

},

{

"recharge\_id": "RC20241214002",

"card\_id": "FK20241213001",

"amount": 50.00,

"recharge\_date": "2024-12-14T15:30:00Z",

"operator": "管理员002"

}

]

}

查询失败：

{

"code": 400,

"message": "未找到对应的充值记录"

}

1. **消费记录查询接口（/api/v1/card/consumption/history）**

**功能描述**：查询指定饭卡的消费历史记录，可根据时间范围、消费地点等条件进行筛选。

**请求方法**：GET

**请求参数**：

card\_id（字符串类型，必填）：要查询消费记录的饭卡编号。

start\_date（日期时间类型，可选）：查询开始时间，用于筛选指定时间范围后的消费记录。

end\_date（日期时间类型，可选）：查询结束时间，用于筛选指定时间范围前的消费记录。

consumption\_location（字符串类型，可选）：消费地点信息，用于筛选特定地点的消费记录。若同时提供时间范围和消费地点参数，则查询同时满足条件的消费记录；若只提供部分参数，则根据提供的参数进行相应筛选。

**响应数据**：

查询成功：

{

"code": 200,

"message": "查询成功",

"data": [

{

"consumption\_id": "CN20241213001",

"card\_id": "FK20241213001",

"amount": 30.00,

"consumption\_date": "2024-12-13T11:00:00Z",

"consumption\_location": "食堂一楼窗口 1"

},

{

"consumption\_id": "CN20241214002",

"card\_id": "FK20241213001",

"amount": 20.00,

"consumption\_date": "2024-12-14T16:00:00Z",

"consumption\_location": "食堂二楼窗口 5"

}

]

}

查询失败：

{

"code": 400,

"message": "未找到对应的消费记录"

}

**（四）系统管理接口（仅管理员可访问）**

1. **用户管理接口（/api/v1/admin/user）**

**功能描述**：对系统用户（学生和管理员）进行管理操作，包括创建、查询、修改和删除用户信息。

**请求方法**：根据具体操作而定（GET/POST/PUT/DELETE）

**请求参数**：

user\_id（字符串类型，可选）：用户编号，在查询、修改和删除操作时使用，用于指定特定用户。

username（字符串类型，可选）：用户名（学号或管理员账号），在查询操作时可作为筛选条件。

password（字符串类型，可选）：密码，在创建或修改用户密码时使用。

role（字符串类型，可选）：角色，取值为 "student" 或 "admin"，在创建或查询特定角色用户时使用。

其他用户信息字段（如学生的姓名、班级、性别，管理员的姓名、权限等）根据具体操作和需求而定，在创建或修改用户信息时使用。

**响应数据**：

操作成功：

code（整数类型）：200（查询操作）或 201（创建操作）或 204（删除操作），表示请求成功。对于修改操作，也可使用 200 表示成功。

message（字符串类型）：相应的成功消息，如 "用户查询成功"、"用户创建成功"、"用户删除成功" 或 "用户信息修改成功"。

data（对象类型或数组类型）：根据操作而定，查询操作返回符合条件的用户信息数组或单个用户对象；创建操作可返回创建成功的用户信息对象；修改和删除操作可根据需求返回相应的提示信息或空数据。

操作失败：

code（整数类型）：400（请求参数错误）或 401（未授权）或 403（禁止访问）或 404（未找到资源）或 409（冲突，如用户名已存在）等，表示请求失败。

message（字符串类型）：错误原因，如 "参数错误"、"未授权访问"、"禁止执行该操作"、"未找到对应的用户" 或 "用户名已存在，创建失败"。

1. **系统参数设置接口（/api/v1/admin/system/parameter）**

**功能描述**：设置系统相关参数，如饭卡有效期、消费限额、密码规则等。

**请求方法**：PUT

**请求参数**：

param\_name（字符串类型，必填）：参数名称，如 "card\_validity\_period"（饭卡有效期）、"consumption\_limit"（消费限额）、"password\_rule"（密码规则）等。

param\_value（字符串类型或数值类型，必填）：参数值，根据参数名称的不同，值的类型和格式可能不同。例如，饭卡有效期可能是一个数字（表示天数），消费限额是一个数值，密码规则可能是一个字符串描述（如 "至少 6 位字符，包含数字和字母"）。

**响应数据**：

设置成功：

code（整数类型）：200，表示请求成功。

message（字符串类型）："系统参数设置成功"。

设置失败：

code（整数类型）：400，表示请求参数错误或其他业务逻辑错误。

message（字符串类型）：错误原因，如 "参数名称错误" 或 "参数值无效"。