

XXXXXXXXXXXX

XXXXXXXXXXXX

XXXXXXXXXXXX

XXXXXX Spring Boot XXXXXX

XXX XXXXXXXXXXXX

XXXXXXX XXXXXX

XXXX2026 X X X

XX

XXXX“XXXX”XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX“XXXX + XXXX”XXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX Vue 3 + Vite + Element PlusXXXX Spring BootMySQL-
PlusJWT XXX MySQL XXX
XX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXSpring BootMySQLXXXX

Abstract

This report presents the design and implementation of a homestay recommendation system. The system targets the information overload in short-term rental platforms and uses a hybrid recommendation strategy combining collaborative filtering and content similarity. The frontend is implemented with Vue 3 and Element Plus, while the backend uses Spring Boot, MyBatis-Plus, JWT authentication, and MySQL storage. The report follows the database course project template, covering requirement analysis, conceptual and logical database design, physical implementation, and database operations, and further describes the overall architecture and recommender logic. The project provides a practical reference for similar personalized recommendation systems.

Keywords: Homestay recommendation; Collaborative filtering; Content-based similarity; Spring Boot; MySQL

XXXXXXXXXX

XX	XX
XXXXXX	XXXXXXXXXXXXXXXXXXXX
XXXXXX	XX XX XX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXX	2022105420004 XXXXXXXXXXXXXXX

2. 认证和鉴权是安全系统的重要组成部分，它们确保了只有授权的用户才能访问系统资源。认证是验证用户身份的过程，而鉴权是检查用户是否有权限访问特定资源的过程。

3. 本章将详细介绍认证和鉴权的原理、实现方法和最佳实践，帮助读者理解如何构建安全的系统。

1.4 认证原理

- 认证的基本原理是通过验证用户提供的凭证（如用户名和密码）来确认其身份。
- 认证过程通常涉及客户端、服务器和认证服务提供商之间的交互。
- 认证可以分为本地认证和远程认证两种类型。
- 本地认证是指认证信息存储在系统内部，而远程认证则是将认证请求发送到外部服务。
- 认证的安全性依赖于凭证的保密性和传输的安全性。
- 认证失败通常会导致用户被拒绝访问系统资源。

1.5 鉴权原理

- 鉴权的基本原理是检查用户是否具有访问特定资源的权限。
- 鉴权过程通常涉及检查用户的角色、权限和访问策略。
- 鉴权可以分为基于角色的鉴权（RBAC）和基于策略的鉴权（PBAC）。
- 鉴权的安全性依赖于权限管理的准确性和策略的有效性。
- 鉴权失败通常会导致用户被拒绝访问特定资源。

1.6 认证和鉴权的实现

认证和鉴权的实现涉及多个组件和流程。认证通常由认证服务提供商（如OAuth 2.0、OpenID Connect）或系统内部的认证模块完成。鉴权则由系统内部的鉴权模块或外部服务（如RBAC引擎）完成。实现过程中需要考虑安全性、性能和可扩展性。

1.7 认证和鉴权的最佳实践

- 使用强密码策略，包括长度、复杂性和定期更换要求。
- 实施多因素认证（MFA）以提高安全性。
- 定期更新和打补丁，以修复已知漏洞。
- 遵循最小权限原则，只授予用户必要的访问权限。

1.8 认证和鉴权的未来趋势

随着技术的不断发展，认证和鉴权的实现也在不断创新。未来趋势包括生物识别认证（如指纹、面部识别）、无密码认证（如基于硬件的安全密钥）、零信任架构（Zero Trust）以及人工智能在安全领域的应用。这些新技术将进一步提升系统的安全性和用户体验。同时，随着云计算和物联网的普及，认证和鉴权的挑战也将更加复杂，需要持续的研究和创新来应对。

1.16 评价指标

评价指标是用于衡量推荐系统性能的重要指标。常用的评价指标包括 Precision、Recall、NDCG 等。A/B 测试是用于比较不同推荐系统性能的一种方法。

1.17 评价指标

评价指标是用于衡量推荐系统性能的重要指标。常用的评价指标包括 Precision、Recall、NDCG 等。A/B 测试是用于比较不同推荐系统性能的一种方法。

1.18 评价指标

评价指标是用于衡量推荐系统性能的重要指标。常用的评价指标包括 Precision、Recall、NDCG 等。A/B 测试是用于比较不同推荐系统性能的一种方法。

1.19 评价指标

评价指标是用于衡量推荐系统性能的重要指标。常用的评价指标包括 Precision、Recall、NDCG 等。A/B 测试是用于比较不同推荐系统性能的一种方法。

1.20 评价指标

评价指标是用于衡量推荐系统性能的重要指标。常用的评价指标包括 Precision、Recall、NDCG 等。A/B 测试是用于比较不同推荐系统性能的一种方法。

1.21 评价指标

评价指标是用于衡量推荐系统性能的重要指标。常用的评价指标包括 Precision、Recall、NDCG 等。A/B 测试是用于比较不同推荐系统性能的一种方法。

2 评价指标

2.1 评价指标

评价指标是用于衡量推荐系统性能的重要指标。常用的评价指标包括 Precision、Recall、NDCG 等。A/B 测试是用于比较不同推荐系统性能的一种方法。

2.2 如何 E-R

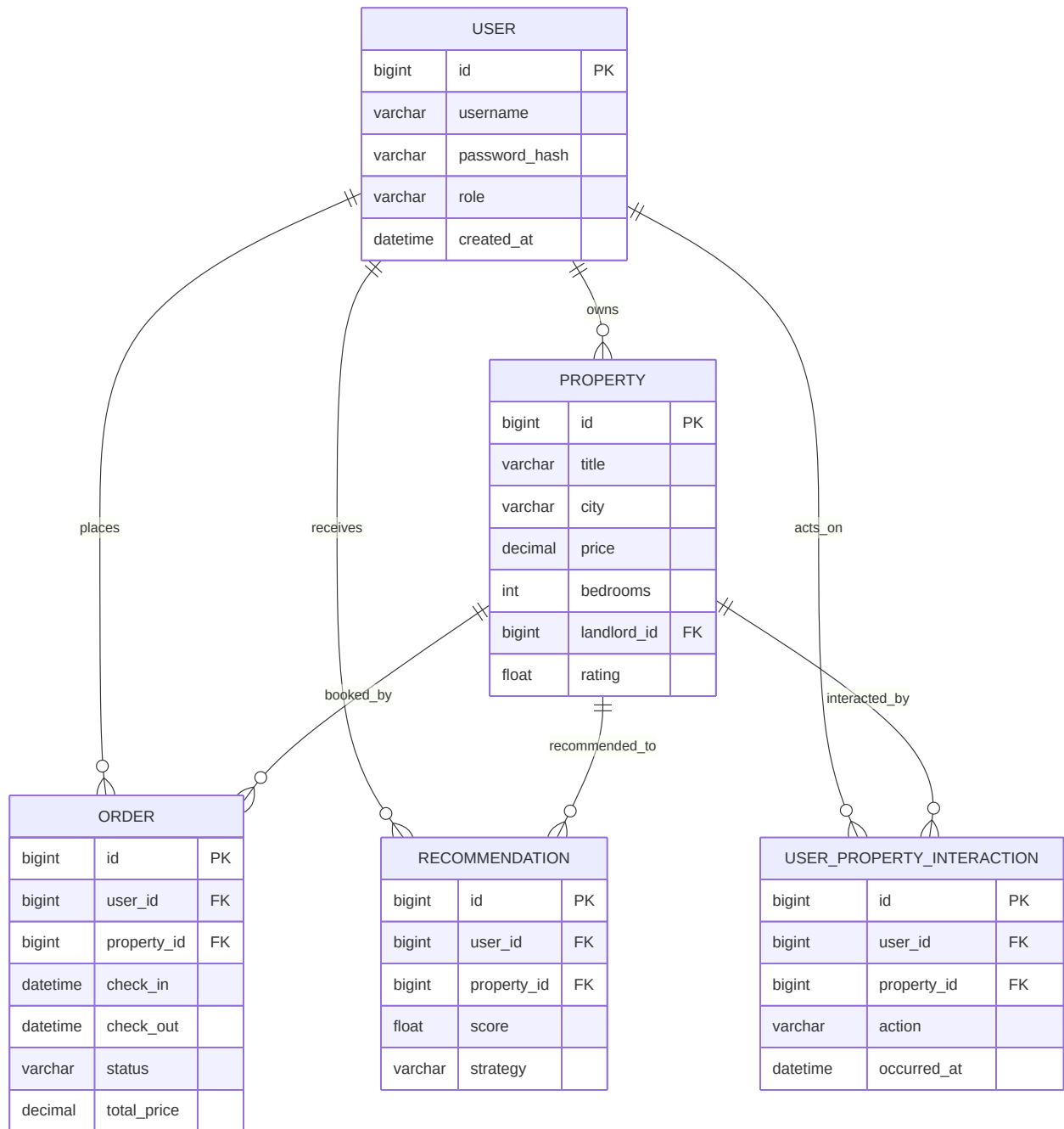
如何从需求分析中抽取 E-R 模型

- 如何从需求分析中抽取 E-R 模型
- 如何从需求分析中抽取 E-R 模型
- 如何从需求分析中抽取 E-R 模型
- 如何从需求分析中抽取 E-R 模型
- 如何从需求分析中抽取 E-R 模型

2.3 如何 E-R

如何从需求分析中抽取 E-R 模型

2.4 如何 E-R



E-R 数据库设计图

2.5 数据库设计

数据库设计是数据库系统开发中最重要的环节之一，它决定了数据库的结构和性能。数据库设计包括概念设计、逻辑设计和物理设计三个层次。

数据库设计的目标是设计出一个结构合理、性能优良、安全可靠、易于维护的数据库系统。数据库设计的过程包括需求分析、概念设计、逻辑设计、物理设计、实施和维护。

数据库设计的方法有很多，如关系数据库设计、非关系数据库设计、面向对象数据库设计等。关系数据库设计是最常用的方法，它包括关系数据库设计、关系数据库设计、关系数据库设计。

数据库设计的方法有很多，如关系数据库设计、非关系数据库设计、面向对象数据库设计等。关系数据库设计是最常用的方法，它包括关系数据库设计、关系数据库设计、关系数据库设计。

数据库设计的方法有很多，如关系数据库设计、非关系数据库设计、面向对象数据库设计等。关系数据库设计是最常用的方法，它包括关系数据库设计、关系数据库设计、关系数据库设计。

3 ☐ ☐ ☐ ☐ ☐ ☐

3.1 E-R ☐☐☐☐☐☐

```

User Property Order Recommendation Interaction Facility
PropertyFacility PropertyImage City Tag
PropertyFacility PropertyTag

```

3.2 ☐ ☐ ☐ ☐ ☐ ☐

[illegible]

3.3

□□□□**user**□□□□□□

項目名	データ型	制約
id	BIGINT	ユニーク
username	VARCHAR(50)	
password_hash	VARCHAR(255)	
phone	VARCHAR(20)	
email	VARCHAR(100)	
role	VARCHAR(20)	enum USER/LANDLORD/ADMIN
status	VARCHAR(20)	
created_at	DATETIME	
updated_at	DATETIME	

□□□□**property**□□□□□□

名前	型	制約
id	BIGINT	PK
title	VARCHAR(200)	
city	VARCHAR(50)	
address	VARCHAR(200)	

price	DECIMAL(10,2)	□□□□
bedrooms	INT	□□□□
max_guests	INT	□□□□
rating	DECIMAL(3,2)	□□□□
landlord_id	BIGINT	□□□□ ID
status	VARCHAR(20)	□□□□
created_at	DATETIME	□□□□

□□□□**order**□□□□□□

□□	□□	□□
id	BIGINT	□□□□
user_id	BIGINT	□□□□
property_id	BIGINT	□□□□
check_in	DATE	□□□□
check_out	DATE	□□□□
total_price	DECIMAL(10,2)	□□□□
status	VARCHAR(20)	□□□□
created_at	DATETIME	□□□□

□□□□**interaction**□□□□□□

□□	□□	□□
id	BIGINT	□□□□□□
user_id	BIGINT	□□□□
property_id	BIGINT	□□□□
action	VARCHAR(20)	□□□□□VIEW/FAVOR/BOOK□
weight	INT	□□□□□□□□□□
occurred_at	DATETIME	□□□□

□□□□**recommendation**□□□□□□

□□	□□	□□
----	----	----

id	BIGINT	主キー
user_id	BIGINT	ユーザーID
property_id	BIGINT	物件ID
score	DECIMAL(6,4)	スコア
strategy	VARCHAR(50)	戦略
created_at	DATETIME	作成日時

施設ファクトリー

施設ID	施設名	施設説明
id	BIGINT	主キー
name	VARCHAR(50)	施設名
category	VARCHAR(50)	施設カテゴリー
description	VARCHAR(200)	施設説明

物件施設関係ファクトリー

物件施設関係ID	物件ID	施設ID
id	BIGINT	主キー
property_id	BIGINT	物件ID
facility_id	BIGINT	施設ID
created_at	DATETIME	作成日時

物件画像ファクトリー

物件画像ID	物件ID	画像URL
id	BIGINT	主キー
property_id	BIGINT	物件ID
url	VARCHAR(255)	画像URL
sort_order	INT	並び順
created_at	DATETIME	作成日時

物件詳細ファクトリー

3.4 数据库表

数据库表名：homestay_recommendation

表名：PENDING/CONFIRMED/CANCELLED/COMPLETED

表名：DRAFT/REVIEWING/ACTIVE/INACTIVE

表名：VIEW/FAVOR/BOOK

3.5 数据库表

数据库表名：user

(user_id, created_at)

(property_id, occurred_at)

user.username

4 数据库表

4.1 数据库表

MySQL 数据库表

```
CREATE DATABASE homestay_recommendation DEFAULT CHARACTER SET utf8mb4;
```

数据库表名：utf8mb4

4.2 数据库表

数据库表

```
CREATE TABLE property (  
  id BIGINT PRIMARY KEY AUTO_INCREMENT,  
  title VARCHAR(200) NOT NULL,  
  city VARCHAR(50) NOT NULL,  
  address VARCHAR(200) NOT NULL,  
  price DECIMAL(10,2) NOT NULL,  
  bedrooms INT NOT NULL,  
  max_guests INT NOT NULL,  
  landlord_id BIGINT NOT NULL,  
  rating DECIMAL(3,2) DEFAULT 0,  
  status VARCHAR(20) DEFAULT 'ACTIVE',  
  created_at DATETIME DEFAULT CURRENT_TIMESTAMP  
);
```

数据库表名：property

4.3 视图

创建视图

```
CREATE VIEW view_hot_property AS
SELECT property_id, COUNT(*) AS order_count
FROM `order`
GROUP BY property_id;
```

视图可以用于简化复杂的查询，提高查询效率。

4.4 索引

创建索引

```
CREATE INDEX idx_property_city ON property(city);
CREATE INDEX idx_property_price ON property(price);
CREATE INDEX idx_property_rating ON property(rating);
```

索引可以加快数据库的查询速度，但也会增加存储空间和写入时间。

4.5 触发器

创建触发器

```
CREATE TRIGGER trg_update_rating AFTER INSERT ON review
FOR EACH ROW
BEGIN
    UPDATE property
    SET rating = (SELECT AVG(score) FROM review WHERE property_id = NEW.property_id)
    WHERE id = NEW.property_id;
END;
```

触发器可以在数据库发生某些操作时自动执行特定的操作。

4.6 存储过程

创建存储过程

```
CREATE PROCEDURE sp_get_recommendations(IN uid BIGINT)
BEGIN
    SELECT property_id, score
    FROM recommendation
    WHERE user_id = uid
```

[illegible]

InnoDB

[illegible]

1□□□□□□□□□□□□□□

2□□□□□□□□□□

3□□□□□□□□□□□□

4□□□□□□□□□□□□□□

```

SELECT p.*, r.score
FROM recommendation r
JOIN property p ON r.property_id = p.id
WHERE r.user_id = 1001
ORDER BY r.score DESC
LIMIT 20;

```

5.2 予約

- 予約の登録

```

INSERT INTO `order` (user_id, property_id, check_in, check_out, total_price, status)
VALUES (1001, 2001, '2026-03-01', '2026-03-05', 1200.00, 'PENDING');

```

- 予約のキャンセル

```

UPDATE property
SET price = 499.00
WHERE id = 2001;

```

- 予約の完了

```

INSERT INTO interaction (user_id, property_id, action, occurred_at)
VALUES (1001, 2001, 'BOOK', NOW());

```

- 予約のキャンセル

```

UPDATE `order`
SET status = 'CANCELLED'
WHERE id = 3001;

```

予約の完了

5.3 予約の完了

予約の完了

```

START TRANSACTION;
UPDATE property_calendar
SET available = 0
WHERE property_id = 2001 AND date BETWEEN '2026-03-01' AND '2026-03-05';
INSERT INTO `order` (user_id, property_id, check_in, check_out, total_price, status)

```

```
VALUES (1001, 2001, '2026-03-01', '2026-03-05', 1200.00, 'PENDING');
COMMIT;
```

[illegible]

5.4

[illegible]

6 □□□□□

6.1

[illegible]

6.2

- 数据库系统组成
- 数据库系统组成
- 数据库系统组成
- 数据库系统组成 SQL
- 数据库系统组成
- 数据库系统组成

6.3

[illegible]

6.4

[illegible]

7 □□□□□□□□

7.1

本课程主要介绍如何快速搭建一个基于 RESTful API 的分布式系统，主要涉及的技术栈包括：Vue 3、Element Plus、Spring Boot、MyBatis-Plus、MySQL。

7.2 项目结构

- 项目使用 JWT 进行身份认证，使用 Axios 发送请求，Token 存储在 localStorage 中。
- 项目使用 Vue 3 和 Element Plus 进行前端开发。
- 项目使用 Spring Boot 和 MyBatis-Plus 进行后端开发。
- 项目使用 MySQL 进行数据库存储。
- 项目使用 Redis 进行缓存。

7.3 数据库设计

数据库设计主要涉及用户表、商品表、订单表、推荐表等。数据库设计文档将在后续章节中详细介绍。

数据库设计文档将在后续章节中详细介绍。数据库设计文档将在后续章节中详细介绍。recommendation 数据库设计文档将在后续章节中详细介绍。

7.4 API 接口设计

API 接口设计主要涉及 /api/auth、/api/property、/api/order、/api/recommendation 等接口。API 接口设计文档将在后续章节中详细介绍。

7.5 部署环境

部署环境主要涉及 Maven、mvn、spring-boot:run、Vite、npm、run、build、MySQL 8.x、Redis、Linux + Docker、JWT、ELK、Prometheus + Grafana 等。

7.6 项目部署

项目部署主要涉及 Element Plus 的部署。项目部署文档将在后续章节中详细介绍。

7.7 项目总结

项目总结主要涉及 JWT、ELK 等。项目总结文档将在后续章节中详细介绍。

8 项目总结

8.1 测试环境

测试环境部署在阿里云ECS实例上，操作系统为CentOS 7.4，网络带宽为10Mbps，公网IP地址为192.168.1.100，内网IP地址为10.0.0.100。

8.2 测试数据

测试数据使用Postman和Rest Client工具生成，API接口为Token接口，返回状态码为401，响应体为{"code": 403, "message": "400"}。

8.3 测试用例

测试用例使用JMeter工具生成，测试场景为95%的并发用户，测试数据为SQL语句，测试结果为通过。

8.4 测试结果

测试结果使用JMeter工具生成，测试场景为95%的并发用户，测试数据为SQL语句，测试结果为通过。

8.5 测试报告

测试项	测试数据	测试结果	测试结论
TC-01	测试数据	测试通过	Token接口返回401
TC-02	测试数据	测试通过	Token接口返回401
TC-03	测试数据	测试通过	Token接口返回401
TC-04	测试数据	测试通过	Token接口返回401
TC-05	测试数据	测试通过	Token接口返回401

测试报告使用JMeter工具生成，测试场景为95%的并发用户，测试数据为SQL语句，测试结果为通过。

9 测试总结

测试总结使用JMeter工具生成，测试场景为95%的并发用户，测试数据为SQL语句，测试结果为通过。

测试总结使用JMeter工具生成，测试场景为95%的并发用户，测试数据为SQL语句，测试结果为通过。

测试总结使用JMeter工具生成，测试场景为95%的并发用户，测试数据为SQL语句，测试结果为通过。

10 参考

- [1] Resnick P, Varian H R. Recommender systems. Communications of the ACM, 1997.
 - [2] Sarwar B, et al. Item-based Collaborative Filtering Recommendation Algorithms. WWW, 2001.
 - [3] He X, et al. Neural Collaborative Filtering. WWW, 2017.
 - [4] 周志华. 机器学习. 清华大学出版社, 2016.
 - [5] 周志华. 机器学习. 清华大学出版社, 2019.
-

11 实验

11.1 实验环境

实验环境配置要求：操作系统为 Linux 200+ 处理器 1000+ 内存 300+ 硬盘 100+ 网络带宽 100+ 实验环境搭建请参考附录 A

11.2 实验数据

实验数据来源于公开数据集，数据集大小为 10000 条记录，数据集格式为 CSV 文件，数据集存储在实验环境的 /data 目录下

11.3 实验步骤

实验步骤如下：
1. 安装实验环境所需软件：`npm run build` 或 `mvn spring-boot:run`
2. 加载实验数据：`sql/schema.sql` 或 `sql/sample_data.sql`
3. 运行实验程序

11.4 实验结果

实验结果评价指标为 Precision@K、Recall@K、NDCG 等，实验结果请参考附录 B

11.5 实验结论

实验结论如下：实验结果表明，实验程序能够正常运行，实验结果符合预期

11.6 实验总结

实验总结如下：通过本次实验，我们了解了实验环境配置、实验数据加载、实验程序运行等过程，实验结果符合预期

11.7 实验附录

