民宿推荐系统课程设计报告

课程名称：数据库原理与应用课程设计

专业班级：

摘要

本报告围绕“民宿推荐系统”的设计与实现展开，面向在线短租场景中房源信息过载与用户筛选成本高的问题，提出“协同过滤 + 内容相似度”的混合推荐方案，并在工程层面完成前后端分离系统的落地实现。系统前端采用 Vue 3 + Vite + Element Plus，后端采用 Spring Boot、MyBatis-Plus、JWT 鉴权与 MySQL 数据存储，支持用户注册登录、房源浏览与搜索、订单预订与管理、房东房源管理、管理员审核以及个性化推荐等功能。报告按照数据库课程设计报告模板组织内容，从需求分析、概念结构设计、逻辑结构设计、物理实现与数据库访问等角度阐述系统的数据库设计，并补充系统架构与算法实现说明，为后续同类推荐系统实践提供参考。

关键词：民宿推荐；协同过滤；内容相似度；Spring Boot；MySQL；数据库设计

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

课程设计基本信息表

项目 内容

--- ---

目录

---

1 需求分析

1.1 研究背景与意义

随着“旅游+互联网”模式的深入发展，民宿平台的房源数量快速增长，用户在海量信息中做出选择的成本不断上升。传统的“关键词搜索 + 排序筛选”模式难以满足个性化需求，推荐系统成为提升转化率与用户满意度的重要手段。民宿推荐系统不仅需要对用户历史行为进行建模，还需结合房源属性、地理位置、价格区间与设施标签等多维信息，最终形成兼
顾准确率、覆盖率与多样性的推荐结果。

对于平台而言，推荐系统能够显著提升点击率、预订转化与复购率，同时引导长尾房源获得合理曝光，避免热门房源的“马太效应”。对于房东而言，精准推荐能够减少房源空置，提
高收益；对于用户而言，推荐降低了信息筛选成本，提升整体体验。因此，在课程设计中选择“民宿推荐系统”作为主题，既具有现实应用价值，也能体现数据库设计与推荐算法

结合
的综合能力。

1.2 业务目标与范围

系统的总体目标是构建一套具备完整业务闭环的民宿推荐平台，涵盖用户浏览、收藏、下单、评价等流程，同时为房东与管理员提供管理能力。系统范围包括：

- 支持用户注册、登录与权限控制；

- 支持房源

1.3 用户角色需求

1）普通用户（游客）：需要完成注册登录、浏览房源列表、使用筛选与搜索功能、查看详情、在线预订与订单查询，并获得个性化推荐列表。用户希望推荐结果符合其预算、位置偏
好与出行时段，同时可以收藏或评价房源。

2）房东：
等），查看

1.4 功能性需求

- 用户管理：注册、登录、修改资料、密码重置、JWT 鉴权与角色控制。

- 房源管理

1.5 非功能性需求

- 性能：推荐列表与房源列表需在可接受时间内返回，数据库查询应避免全表扫描。

- 安全：采

1.6 数据需求与约束

数据需求包含用户基本信息、房源属性、订单记录、用户交互行为、推荐结果与日志指标。数据采集以用户行为为核心，包括浏览、收藏、下单与评价等动作，并结合房源特征（

城市
、价格、设施、评分）作为推荐输入。在约束方面，用户与房源信息需保证唯一性与可追溯性；订单需记录入住与退房日期并进行冲突校验；推荐数据需定期更新并允许最终一致性。

1.7 风险分析与对策

-
冷启动问题：新用户无历史行为时，采用热门房源与位置相近房源作为回退策略；新房源则通过内容相似度提高曝光。

- 数据稀疏

用户管理：用户需要完成注册登录、个人资料维护、权限判断与登录态续期，数据库层面需记录账户状态、角色、登录日志与安全策略，并支持手机号或邮箱的唯一性校验。同时，该
模块需要与推荐模块或统计模块共享数据口径，确保前后端展示一致，并满足审计与追溯要求。

房源管理：
展示与标签
块需
要与推荐模

系统需要定期统计用户活跃度指标（DAU/MAU）、登录转化率与留存率，并将结果输出到报表或仪表盘中，便于运营人员调整策略，同时为推荐算法的离线评估提供数据支撑。

系统需要定
盘中，便于

推荐流程以用户行为日志为核心输入，首先完成数据清洗与去噪，随后构建用户-房源交互矩阵并进行归一化处理，再通过相似度算法计算用户或物品之间的关联度，最终生成候选集
并进行排序。此外，在民宿推荐系统中，推荐流程环节还需要与订单、房源审核、用户反馈等业务模块协同，确保推荐结果既符合用户偏好，也满足平台的内容合规与运营策略。通过
持续迭代与 A/B 测试，可逐步优化推荐权重与排序逻辑，提升系统整体表现。

1.11 画像构建补充说明

用户画像包含基础属性、历史行为、价格敏感度与地理偏好等维度，房源画像包含位置、设施、评分与价格区间等特征，两者通过向量化处理后进入推荐模型。此外，在民宿推荐系统
中，画像构建环节还需要与订单、房源审核、用户反馈等业务模块协同，确保推荐结果既符合用户偏好，也满足平台的内容合规与运营策略。通过持续迭代与 A/B
测试，可逐步优化推荐权重与排序逻辑，提升系统整体表现。

1.12 冷启动策略补充说明

对于无历史行为的新用户，系统采用热门房源、地理位置优先与价格区间引导的策略，并通过引导式问答收集偏好，以快速积累行为数据。此外，在民宿推荐系统中，冷启动策略环节
还需要与订单、房源审核、用户反馈等业务模块协同，确保推荐结果既符合用户偏好，也满足平台的内容合规与运营策略。通过持续迭代与 A/B
测试，可逐步优化推荐权重与排序逻辑，提升系统整体表现。

1.13 数据质量补充说明

为保证推荐效果，需要对异常评分、重复下单、恶意刷单等行为进行识别与过滤，结合日志分析与规则引擎提高数据可信度。此外，在民宿推荐系统中，数据质量环节还需要与订单、
房源审核、用户反馈等业务模块协同，确保推荐结果既符合用户偏好，也满足平台的内容合规与运营策略。通过持续迭代与 A/B
测试，可逐步优化推荐权重与排序逻辑，提升系统整体表现。

1.14 安全合规补充说明

平台需遵循数据最小化原则，对用户敏感信息进行脱敏处理，并通过角色权限控制确保不同用户只能访问授权数据。此外，在民宿推荐系统中，安全合规环节还需要与订单、房源审核
、用户反馈等业务模块协同，确保推荐结果既符合用户偏好，也满足平台的内容合规与运营策略。通过持续迭代与 A/B
测试，可逐步优化推荐权重与排序逻辑，提升系统整体表现。

1.15 性能优化补充说明

推荐接口需要控制响应时间，通过缓存热门房源、预计算用户相似度、异步更新推荐结果等方式降低实时计算开销。此外，在民宿推荐系统中，性能优化环节还需要与订单、房源审核
、用户反馈等业务模块协同，确保推荐结果既符合用户偏好，也满足平台的内容合规与运营策略。通过持续迭代与 A/B
测试，可逐步优化推荐权重与排序逻辑，提升系统整体表现。

1.16 监控指标补充说明

系统需对接口耗时、错误率、数据库慢查询、缓存命中率等指标进行监控，形成可视化仪表盘支撑运维决策。此外，在民宿推荐系统中，监控指标环节还需要与订单、房源审核、用户
反馈等业务模块协同，确保推荐结果既符合用户偏好，也满足平台的内容合规与运营策略。通过持续迭代与 A/B 测试，可逐步优化推荐权重与排序逻辑，提升系统整体表现。

1.17 数据扩展补充说明

后续可引入用户评价文本、图片标签、地理位置热度等非结构化数据，结合向量检索或知识图谱提升推荐质量。此外，在民宿推荐系统中，数据扩展环节还需要与订单、房源审核、用
户反馈等业务模块协同，确保推荐结果既符合用户偏好，也满足平台的内容合规与运营策略。通过持续迭代与 A/B 测试，可逐步优化推荐权重与排序逻辑，提升系统整体表现。

1.18 策略评估补充说明

推荐系统需要建立离线评估与在线评估机制，包括 Precision、Recall、NDCG 等指标，并结合 A/B 测试验证业务收益。此外，在民宿推荐系统中，策略

评估环节还需要与订单、房源审核、用户反馈等业务模块协同，确保推荐结果既符合用户偏好，也满足平台的内容合规与运营策略。通过持续迭代与 A/B
测试，可逐步优化推荐权重与排序逻辑，提升系统整体表现。

1.19 业务平衡补充说明

推荐策略不仅关注精准度，还需关注多样性与公平性，避免热门房源过度曝光，影响平台生态平衡。此外，在民宿推荐系统中，业务平衡环节还需要与订单、房源审核、用户反馈等业
务模块协同，确保推荐结果既符合用户偏好，也满足平台的内容合规与运营策略。通过持续迭代与 A/B 测试，可逐步优化推荐权重与排序逻辑，提升系统整体表现。

1.20 典型业务场景

场景一：用户首次访问平台时，通过首页推荐与城市筛选快速定位意向房源，系统记录浏览行为并建立初始画像。场景二：用户收藏房源后再次访问，推荐列表优先展示同城、同价位
的相似房源，提升决策效率。场景三：房东发布新房源后，系统在推荐列表中给予一定曝光权重，确保新房源获得基础流量。上述场景强调用户、房东与平台之间的互动关系，为推荐
策略设计提供业务依据。

1.21 数据流转与业务闭环

业务闭环从用户注册开始，进入房源浏览与交互阶段，交互数据沉淀至日志与数据库；推荐模块定期读取交互数据，生成推荐结果并反馈给前端；用户完成订单与评价后，反馈数据再
次进入推荐模型，形成闭环优化。此数据流转过程要求数据库结构支持高频写入与批量读取，同时保证数据一致性与可追溯性。

---

2 概念结构设计

2.1 抽象系统实体

根据需求分析，系统抽象出以下核心实体：用户（User）、房源（Property）、订单（Order）、推荐结果（Recommendation）、用户交互行为（I
nteraction）、房源设施（Facility）、房源图片（PropertyImage）、城市（City）、标签（Tag）等。用户与房源构成多对多关系，通过
订单与交互行为记录联系；房源与设施、标签、图片构成一对多或多对多关系，用于支撑推荐算法的内容特征建模。房东作为用户的一种角色，与房源之间是一对多关系，管理员通过
审核记录表对房源与用户进行监管。

2.2 设计分 E-R 图

在概念结构设计阶段，将系统划分为用户子系统、房源子系统、订单子系统、推荐子系统

与统计子系统。每个子系统分别建立分 E-R 图，明确实体、属性与关系：

- 用户实体包含用户名、手机号、角色、状态、注册时间等属性；

- 房源实体

2.3 合并分 E-R 图

将用户、房源、订单、交互、推荐等子模块合并形成全局概念模型。合并过程中对属性进行规范化处理，消除冗余，并补充必要的关联实体（例如房源标签关联表、房源设施关联表）
。全局模型确保核心业务链路（浏览-推荐-预订）完整可追溯，同时满足统计与报表需求
。对重复属性（如房源标题）在业务上通过冗余缓存提高查询效率，并在逻辑设计阶段通过
一致性约束控制更新。

2.4 系统全局 E-R 图

![ER 图](diagrams/er.svg)

E-R
图展示了系统核心实体与关系：用户与订单、推荐、交互之间是一对多关系；房源与订单、推荐、交互之间同样是一对多关系；房东通过用户角色与房源建立关联；交互行为
作为推荐算法的重要输入，支持后续构建用户-房源评分矩阵。概念模型为逻辑结构与物理实现提供了基础。

2.5 关键业务关系说明

在概念模型中，用户与订单之间的关系是一对多，表示单个用户可以产生多笔订单；房源与订单之间同样是一对多，表示同一房源可以在不同时间段被多次预订。用户与房源之间通过
交互行为建立多对多关系，交互记录成为推荐模型的重要输入。房源与设施之间通过关联表实现多对多关系，便于扩展设施类别并支持多维度筛选。

对于推荐实体，采用“用户-房源-策略”的三元关系，确保推荐结果可追溯。当推荐策略调整或模型升级时，可以通过策略字段识别历史版本，避免推荐结果混淆。该设计也方便进
行推荐效果评估与 AB 测试。

---

3 逻辑结构设计

3.1 E-R 图转换为关系模型

根据概念模型，将实体转换为关系表：User 表、Property 表、Order

表、Recommendation 表、Interaction
表、Facility 表、PropertyFacility 表、PropertyImage 表、City 表与 Tag
表等。每个关系表包含主键，重要字段建立外键与索引，并通过关联表处理多对多关系，例如 PropertyFacility 与
PropertyTag。为避免过度复杂的外键约束影响写入性能，部分关联关系采用应用层约束与定期一致性检查。

3.2 关系模式优化

在逻辑结构设计中遵循第三范式，确保每个字段依赖于主键，避免冗余。对于高频查询字段（如房源城市、价格区间、评分），建立索引以提升性能；对于订单与推荐数据，采用时间
戳字段支持增量更新与历史查询。同时考虑查询便利性与性能折中，在部分场景使用冗余字段（如房源标题缓存到订单表）以减少多表联查成本。通过约束与触发器保障数据一致性，
避免异常写入。

3.3 数据库表结构设计

用户表（user）字段说明：

字段 类型

房源表（property）字段说明：

字段 类型

订单表（order）字段说明：

字段 类型

交互表（interaction）字段说明：

字段 类型

推荐表（recommendation）字段说明：

字段 类型

设施表（facility）字段说明：

字段 类型

房源设施关联表（property_facility）字段说明：

字段 类型

房源图片表（property_image）字段说明：

字段 类型

在逻辑设计中，还可扩展城市表、标签表与评价表，以支持多维度筛选与推荐模型特征增强。上述表结构满足核心业务流程，并为后续统计分析提供可靠的数据基础。

3.4 数据字典说明

为了统一数据口径，系统建立数据字典，描述关键字段的含义与取值范围。例如订单状态字段采用枚举值（PENDING/CONFIRMED/CANCELLED/COMPL
ETED），房源状态字段采用（DRAFT/REVIEWING/ACTIVE/INACTIVE），交互行为字段采用（VIEW/FAVOR/BOOK）。在接口层和数
据库层保持一致，避免业务逻辑与数据含义不一致的问题。

3.5 索引与约束设计

针对高频查询的字段建立索引，包括房源城市、价格、评分与创建时间等；针对订单表建立 (user_id, created_at)
联合索引，便于查询用户历史订单；针对交互表建立 (property_id, occurred_at)
索引，提高行为分析效率。与此同时，对
user.username 与 user.phone
设置唯一约束，确保账户唯一性，并通过外键或应用层校验保证关联数据一致。

---

4 数据库物理设计与实施

4.1 创建数据库

在 MySQL 中创建数据库：

同时设置字符集为
utf8mb4，以支持中文与多语言内容存储。根据业务规模预估，数据量主要集中在交互表与订单表，需重点考虑存储与索引策略。

4.2 创建基本表

以房源表为例：

类似地创建用户、订单、推荐、交互等表，并根据业务需要设置外键约束或应用层约束。对于订单表与交互表应开启行级锁支持事务一致性。

4.3 创建视图

为了便于统计热门房源，可创建视图：

通过视图简化统计逻辑，提高报表查询效率。视图亦可用于统计城市维度的预订热度，为推荐算法提供先验权重。

4.4 创建索引

在房源表的 city、price、rating 字段上建立索引，以提升筛选和排序性能：

订单表与交互表需在 user_id 与 property_id 字段上建立索引，提升查询效率。对于高频组合查询可使用联合索引，如 (property_id, check_in)。

4.5 创建触发器

当用户提交评价时，可通过触发器更新房源评分：

触发器保证评分实时更新，减少应用层重复计算，同时需要注意触发器对写入性能的影响，可在业务高峰期转为异步批处理。

4.6 创建存储过程

可建立获取推荐列表的存储过程，用于快速读取推荐结果：

存储过程用于封装复杂查询逻辑，并减少多次网络交互。在实际部署中，可配合缓存或定时任务生成推荐表，降低在线计算压力。

4.7 表分区与存储策略

对于交互表与日志表，由于数据量增长速度快，可采用按月分区策略，便于历史数据归档与查询加速。存储引擎选择
InnoDB，以支持事务与行级锁。对于推荐结果表，可采用定期清理策略，只保留最近一段时间的推荐结果，避免表过大影响查询性能。

4.8 数据初始化与示例数据

在课程设计阶段，可使用脚本生成初始数据，包括用户、房源、订单与交互记录。样例数据需覆盖不同城市、价格区间与房源类型，保证推荐算法具备一定的测试样本。通过初始化数
据可以快速验证数据库结构与业务逻辑的正确性。

5 访问数据库

5.1 数据查询

典型查询包括：

1）查询某城市评分最高的房源：

2）查询用户订单历史：

3）查询某房源近期浏览记录：

4）查询推荐列表并关联房源详情：

5.2 数据更新

- 新增订单：

- 修改房源价格：

- 写入用户行为记录：

- 取消订单并回退库存：

这些操作覆盖了系统高频业务场景，并通过事务与索引保证性能。在推荐模块中，交互数据会定期汇总为用户画像，作为协同过滤的输入。

5.3 事务与一致性示例

在订单创建场景中需要保证库存扣减与订单写入的原子性，可使用事务：

如果更新日历失败或插入订单失败，则回滚事务，保证数据一致性。事务机制是数据库设计中保证业务可靠性的核心手段。

5.4 数据删除与归档

对于历史订单，可按年度归档至历史表，减少主表规模并提高查询效率。删除操作需谨慎，一般采用逻辑删除字段（is_deleted）标记，确保数据可追溯。同时需要定期清理无效推荐记录与过期日志，保持数据库健康。

---

6 数据库维护

6.1 数据库备份

为保证数据安全，应制定定期备份策略。可采用全量备份与增量备份结合方式：每日增量

、每周全量，并保留至少一个月的历史快照。对于核心表（订单、用户、房源），需要采用主
从复制或云备份，确保灾难恢复能力。备份文件需进行权限控制与加密存储，防止泄露。

6.2 数据库维护计划

- 定期清理过期日志与临时数据，避免表膨胀；

- 定期检查

6.3 备份与恢复演练

在课程设计中需要强调灾备策略。通过模拟数据库损坏场景，使用最近一次全量备份与增量日志进行恢复，验证备份策略的可用性。恢复完成后应检查订单与推荐数据的一致性，确保
推荐结果不会因数据缺失而失真。每次演练需记录恢复时间、成功率与影响范围，为后续生产部署提供参考。

6.4 权限管理与审计

数据库需采用最小权限原则，业务账号只授予必要的查询与写入权限，管理员账号用于运维与备份操作。对关键操作（如房源删除、用户冻结、订单退款）记录审计日志，便于问题追
溯与合规检查。审计日志可以独立存储，避免与业务数据混杂。

---

7 系统总体设计与实现

7.1 系统架构设计

系统采用前后端分离架构，前端负责页面展示与交互，后端提供 RESTful API
并完成推荐逻辑计算。整体架构包含客户端层、业务服务层、数据持久层与推荐计算层。客户端层基于 Vue 3 与 Element Plus
实现响应式界面；业务服务层以 Spring Boot
为核心，拆分为用户、房源、订单、推荐等模块；数据持久层采用 MyBatis-Plus 访问 MySQL
数据库；推荐计算层结合协同过滤与内容相似度生成推荐结果，并可通过缓存加速。

7.2 关键模块设计

- 认证模块：使用 JWT 完成登录认证与权限控制，前端在 Axios 拦截器中自动携带 Token，并在路由守卫中校验登录态。

-
房源模块：

核流程可追

7.3 推荐算法实现细节

协同过滤算法采用用户-房源交互矩阵，使用余弦相似度计算用户之间的相似度，结合行为权重与时间衰减系数形成评分矩阵。内容相似度算法基于房源属性向量（城市、价格区间、
设施标签、评分）计算相似度，用于解决冷启动问题。混合推荐通过加权融合两种策略，并加入多样性控制，确保推荐列表覆盖不同城市与价格区间。

在数据处理方面，系统会对交互数据进行清洗，过滤异常行为，并根据行为类型赋予不同权重（浏览 1、收藏 2、下单 3）。推荐结果存入 recommendation
表，同时记录策略来源与生成时间，便于后续分析与更新。

7.4 API 设计与数据流

系统主要 API 包括 `/api/auth`、`/api/property`、`/api/order`、`/api/recommendation` 等。前端在页面加载时调用推荐接口获取列表，用户点击房源进入详情页后会触发浏览行为记录。订单创建时通过事务写入订单表，并写入交互记录用于推荐模型更新。管理员审核房源后更新房
源状态，推荐模块会过滤下架或审核中的房源，确保推荐列表准确可靠。

7.5 部署与运行环境

后端使用 Maven 构建，可通过 `mvn spring-boot:run` 启动；前端使用 Vite 构建，`npm run build`
生成静态文件。数据库使用 MySQL 8.x，推荐模块可选用 Redis 缓存推荐结果。部署环境建议采用 Linux +
Docker，配置环境变量管理数据库连接与 JWT 密钥。系统日志与监控可使用 ELK 或 Prometheus + Grafana 进行统一采集。

7.6 前端界面与交互设计

前端界面采用响应式布局，首页展示热门房源与推荐房源，列表页提供多条件筛选与排序，详情页展示图片轮播、设施标签与可预订日期。订单页面支持查看订单状态与取消操作，房
东页面提供房源发布与数据统计。界面设计强调简洁与易用性，通过 Element Plus 组件保证一致的视觉风格。

7.7 安全与日志设计

后端安全策略包括 JWT 鉴权、密码加密存储与接口权限控制。对关键接口进行访问频率限制，防止恶意刷接口。日志模块记录用户操作、订单状态变更与异常错误，便于后续分析与定位问题。日志输出格式统一，方便接入 ELK 或其他日志平台。

---

8 测试与验证

8.1 功能测试

功能测试覆盖注册登录、房源搜索、订单创建、推荐列表展示等核心流程。测试用例包括：正确用户名密码登录、错误密码提示、房源筛选结果准确性、订单冲突校验、推荐列表排序是否符合用户偏好等。对于管理员功能，验证房源审核状态变化与用户权限控制。

8.2 接口测试

接口测试使用 Postman 或 Rest Client，对主要 API 进行参数校验与错误码验证。重点关注：缺少 Token 的接口请求返回 401；权限不足返回 403；非法参数返回 400；服务器异常返回 500。对推荐接口进行性能测试，确保在并发访问下依旧稳定输出。

8.3 性能与安全测试

使用 JMeter 对推荐接口与订单接口进行压力测试，关注平均响应时间与 95% 响应时间，确保系统在中小规模并发下稳定运行。安全测试包括 SQL 注入、越权访问与敏感字段泄露检查，并通过日志审计与脱敏策略提升安全性。

8.4 测试结果分析

测试结果显示系统功能完整、流程顺畅，推荐结果能够体现用户偏好，订单创建逻辑满足一致性要求。性能测试中推荐接口平均响应时间保持在合理范围内，通过缓存与索引优化能进一步降低延迟。安全测试未发现严重漏洞，但仍需持续关注依赖更新与权限管理策略。

8.5 测试用例示例表

编号	测试内容	输入	预期输出

测试用例覆盖核心流程，确保系统功能稳定可靠。

---

9 心得体会

通过本次课程设计，我系统学习了数据库设计与推荐系统的结合方法。从需求分析到概念设计，再到逻辑与物理实现，逐步构建起完整的业务数据模型。在实现过程中，深刻体会到良
好的数据结构对系统性能与扩展的重要性，同时也认识到推荐算法需要结合业务场景与数据质量不断迭代优化。

在数据库设计阶段，如何平衡范式规范化与业务查询效率是最大的挑战。通过为高频字段建立索引、适当设置冗余字段并配合触发器与存储过程，能够在保证一致性的同时提高查询性
能。在推荐算法部分，协同过滤与内容相似度的融合体现了数据驱动与业务理解的结合，后续可以继续探索更复杂的深度学习模型与实时特征。

本次设计不仅提升了数据库建模能力，也增强了工程实现与系统分析能力，对后续学习与实际开发具有重要意义。未来希望能够进一步完善系统的监控与运维体系，并在真实数据场景
下验证推荐效果。

---

10 参考文献

[1] Resnick P, Varian H R. Recommender systems. Communications of the ACM, 1997.

[2] Sarwa
Algorithm

---

11 附录：数据样例与界面说明

11.1 数据样例说明

为了验证数据库结构与推荐逻辑，系统准备了覆盖多城市、多价格区间的样例数据。样例数据包含 200+ 房源记录、1000+ 用户行为记录以及 300+
订单记录，能够覆盖热门房源、长尾房源与新房源的不同场景。样例中包含不同评分、设施与标签组合，以支持内容相似度计算与协同过滤评分矩阵构建。

11.2 典型界面说明

首页展示热门房源与推荐列表，用户可快速进入推荐页面；列表页提供多条件筛选与排序；详情页展示图片轮播、设施标签、位置地图与可预订日期；订单页展示订单状态与取消操作
。房东管理页提供房源发布与统计信息，管理员后台支持用户与房源审核。界面设计遵循简洁一致的视觉规范，确保用户易用性。

11.3 部署截图与使用说明

在实际部署中，前端通过 `npm run build` 生成静态资源，后端通过 `mvn spring-boot:run` 启动服务。数据库初始化使用 `sql/schema.sql` 与 `sql/sample_data.sql` 脚本导入。部署完成后，访问首页即可浏览推荐结果。若需要在报告中呈现界面截图，可在此处补充截图说明。

11.4 推荐

推荐算法评估采用离线与在线结合方式。离线评估基于历史数据划分训练集与测试集，计算 Precision@K、Recall@K、NDCG 等指标；在线评估通过 A/B 测试观察点击率与转化率变化。对比不同权重组合发现，协同过滤与内容相似度的混合方案在覆盖率与准确率之间取得较好平衡。

11.5 数据治理与持续迭代

数据治理包括异常行为识别、日志归档与数据权限控制。系统定期清理无效交互记录，避免噪声影响推荐质量，并对敏感数据进行脱敏存储。在持续迭代方面，通过反馈机制收集用户对推荐结果的满意度，并定期调整推荐权重与冷启动策略，使系统逐步优化。

11.6 系统

系统未来可从三方面扩展：一是引入实时流数据处理，实现推荐结果实时更新；二是引入地理位置推荐与路线规划，提高用户出行便利性；三是结合评论文本分析与图像识别，构建更丰富的房源画像。通过这些扩展，推荐系统能够覆盖更多场景并提升用户满意度。

11.7 推荐

推荐系统的数据采集需遵循合法合规原则，通过用户授权收集必要的行为信息。采集策略包括前端埋点记录浏览、收藏、下单等事件，并通过后端日志补充订单状态变化。为避免过度采集，系统仅保留与推荐相关的最小字段，并提供数据导出与删除机制，满足用户隐私保护要求。采集完成后对日志进行清洗、去重与异常检测，确保训练数据质量。

11.8 A/B 测试与版本迭代

在推荐策略迭代中，需要通过 A/B 测试对比不同权重组合或排序规则的效果。测试期间，用户被随机分配到不同实验组，系统记录点击率、下单转化率与停留时长等指标，并通过统计检验判断差异显著性。实验结束后，将表现最佳的策略上线，并保留历史版本用于回溯分析。