数据库系统原理个人课程设计报告

城市文旅出行查询平台

学　号 2054080

姓　名 林奕如

专　业 计算机科学与技术

授课老师 关佶红

目录

1　简介

1.1 项目背景

随着人们物质生活水平不断提高，交通不断便利，越来越多人选择在休息日和假期结伴出门旅行，放松身心。时长三天以内、地点为某一城市的自驾旅行成本小、自由度高，受到很多年轻人的偏爱。如今不少城市在文旅消费建设投入可观，加上网络自媒体的营销与分享，在原有老牌景区的基础上又涌现出一批新走红的旅游打卡点。对于各种各样的景点，即使是本地人在出行时也会遇到不熟悉的情况，造成出行不便。因此，在互联网技术发达的今天，许多旅游服务平台应运而生。

此类平台收录了城市内可供吃喝玩乐的景点信息，提供游玩攻略和交通等相关信息，旅客也可通过平台查看景点发布的通知等。这样集多功能为一体的综合旅行平台能让旅客出行变得更加便捷，也能促进旅客消费，带动城市旅游业。

1.2 项目概述

本项目是一款基于关系型数据库模型设计的文旅出行查询平台，该平台依托城市服务平台，旨在为用户提供便捷的文旅出行服务。景点可由平台管理员添加创立与维护，管理员可通过平台发布每个景点的通知。对于每个景点，数据库记录其名称、类型、所在地等详细信息。平台为游客提供多样化的景点查询和打分评价。

2　需求分析

2.1 项目前景

从游客的角度来看，该平台通过记录景点的类型、地址、旅客评分、评论等相关信息，为游客提供全面准确的旅游信息，帮助游客更快速、更便捷地规划旅游行程，同时展现实时准确的旅游信息和评论，帮助游客更好地了解景点的实际情况，提高游客的旅游体验。

从景点负责人的角度来看，该平台可以为景点提供更多的曝光机会，帮助景点吸引更多的游客，提高景点的知名度和曝光率，同时帮助景点负责人更好地管理景点信息和评论，及时了解游客的反馈和评价，优化景点管理和服务。通

过了解游客的需求和反馈，景点负责人可以更好地满足游客的需求，提高游客的满意度，进而提高景点的口碑和吸引力。

2.2 用户特征

2.2.1 游客

该文旅查询平台的目标用户群体之一是年龄在 20 岁至 50 岁之间的游客，他们通常具有一定的旅游经验，对文旅出行有一定的了解和认识。游客的旅行目的一般是为了放松心情，开阔视野，增加知识和体验当地的文化。在使用平台的过程中，游客希望能够获得更好的出行服务和体验。例如，他们希望能够快速、准确地查询景点信息，了解景点的详细情况和评价，以便更好地规划旅游行程，也希望能够针对景区发表自己的评价，供其他用户参考。此外，游客也希望能够关注自己感兴趣的景点，更加快速的查收景区发布的通知。

2.2.2 景点负责人

景区负责人是具备丰富的景区管理经验、熟悉本地旅游资源和市场状况、能够熟练使用电脑的人士。景区负责人希望平台提供一个方便快捷的界面，让他们能够轻松地在平台上新建景区，包括所在地区、所属类型等信息。平台需要提供一个完善的数据管理系统，让景区负责人能够随时维护景区信息，向游客发布重要通知和公告，如特别活动、天气预报等信息，提高游客的满意度和体验。同时，景区负责人可以查看游客的评价和反馈，及时做出改进和调整。

2.3 系统用例

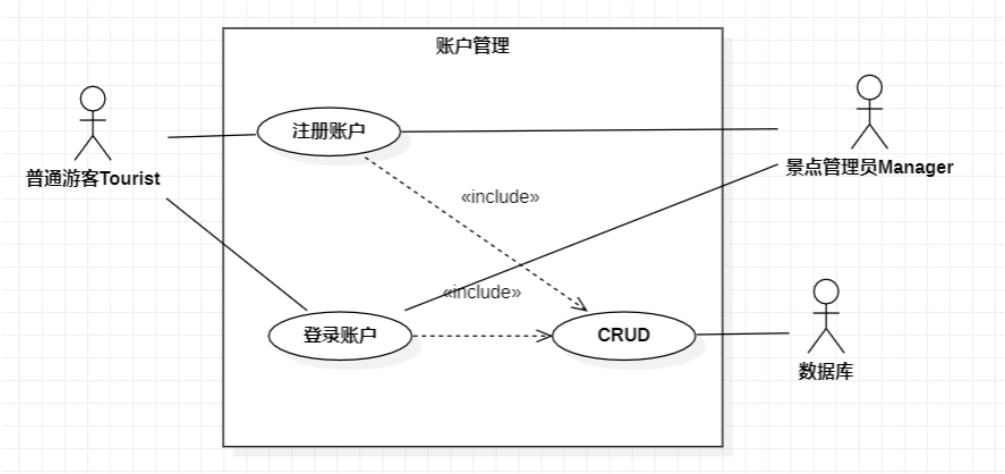
根据平台的需求分析与用户特征，结合对开发技术的学习和调研，我计划将该文旅出行系统划分为以下四个功能用例分别实现：

2.3.1 账号管理

账号管理模块适用于所有用户，他们需要登录平台来访问平台信息、使用平台服务。此外，景区负责人也需要登录平台来管理景区信息和发布通知公告。

用户需要先注册账号才能登录平台，普通游客的注册的信息包括用户名、密码，景点负责人注册时还需添加初始的自己管理的景点的名称。随后，用户使用注册时设置的用户名和密码登录平台。平台需要对用户输入的信

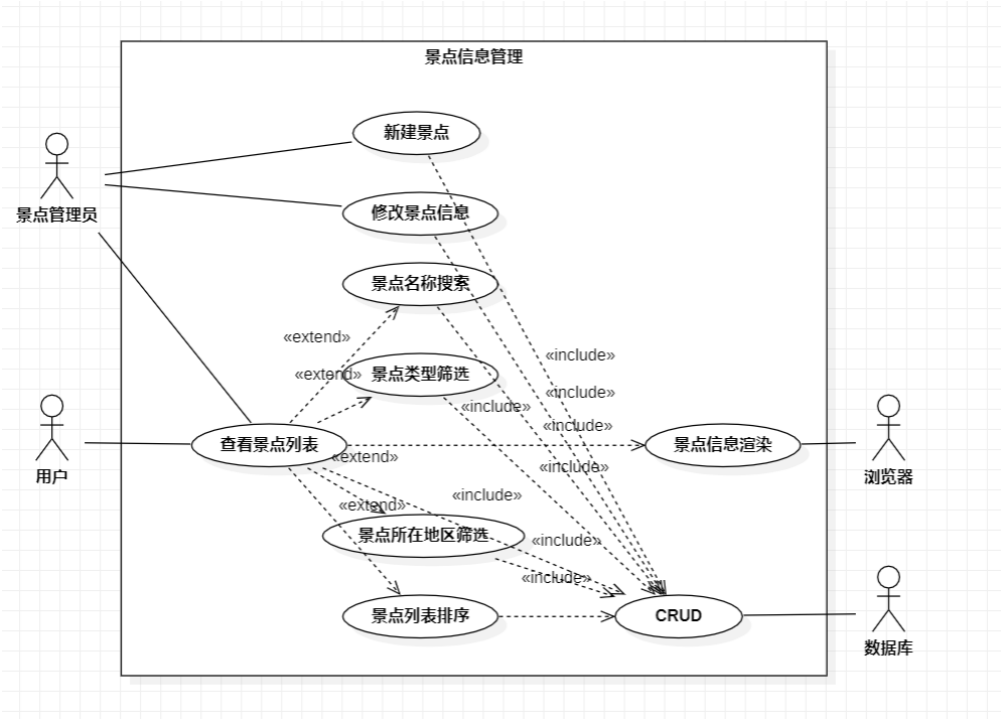
息进行校验，确保用户信息的安全性。



2.3.2 景点信息管理

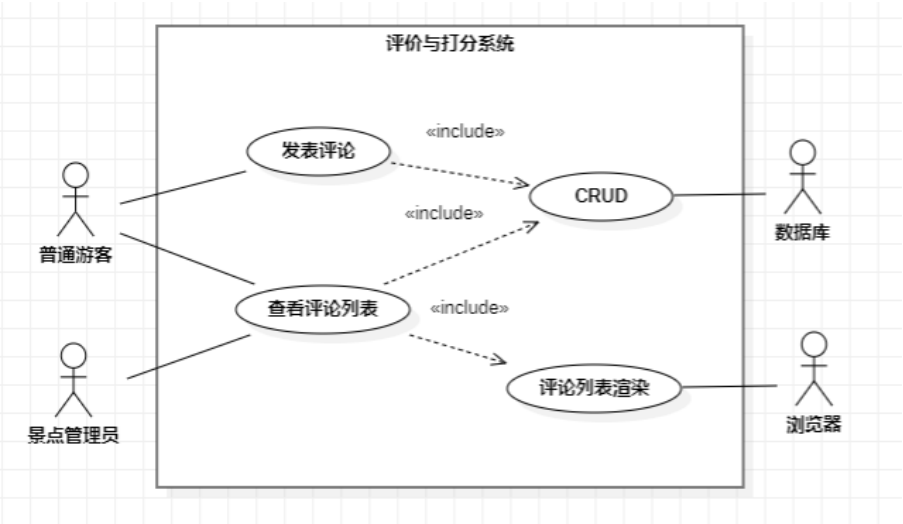
普通游客在登陆后可按需查看并检索景点列表，获取如景点名称、类型、所在地等基本信息；用户可以输入景点名称作为关键字在平台上进行景点信息检索，平台会返回相关的景区信息，也可选择景点类型与所在地区，平台筛选后返回。除此之外，平台还提供排序功能，让普通游客可以按照不同顺序查看景点列表。

景点管理员可查看自己管理的景点信息，进行基本信息维护，包括修改景点类型与所在地区，也可新增景点。



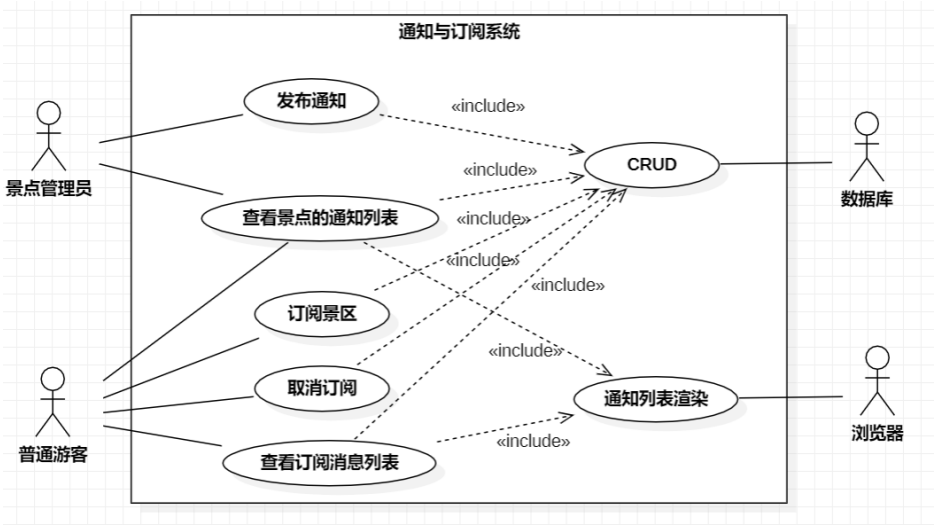
2.3.3 评价打分系统

用户可以在景点详情页查看该景点的相关评价，包括评分与具体评价内容。普通游客可发表自己的评价，每记录一条评分，平台自动计算并更新该景点的最新平均分。景点管理员只可查看不可发表评价。



2.3.4 通知与订阅系统

用户可以在景点详情页查看该景点的相关通知，包括通知日期与具体通知内容。普通游客可以订阅关注感兴趣的景点，并直接查看所有订阅的景点的通知。景点管理员可以发布通知。



2.4 可行性分析

2.4.1 一般性限制

在系统设计与开发过程中，必须遵守软件开发的一些一般性规范和限制。包括如下几点：

- 软件可靠性：软件必须具有一定的可靠性，包括在各种情况下运行稳定、数据准确无误、避免系统崩溃等多个方面。
- 软件易用性：软件必须具有一定的易用性，包括界面友好、操作简便、功能齐全等多个方面。
- 软件兼容性：软件必须具有一定的兼容性能，能够在不同的操作系统、不同的硬件平台上运行稳定。
- 软件可维护性：软件必须具有一定的可维护性能，包括代码可读性、易于维护、易于扩展等多个方面。

2.4.2 安全性分析

系统的安全性需求主要包括网络安全、系统安全和数据安全。网络安全方面，要确保服务器、交换机、防火墙等硬件设备的安全，谨防用户的个人信息泄露。数据安全方面，为数据库及时做好外部存储设备的备份，起到稳定存储的作用，防止因本地计算机出现故障而导致数据丢失难以恢复。

2.4.3 性能分析

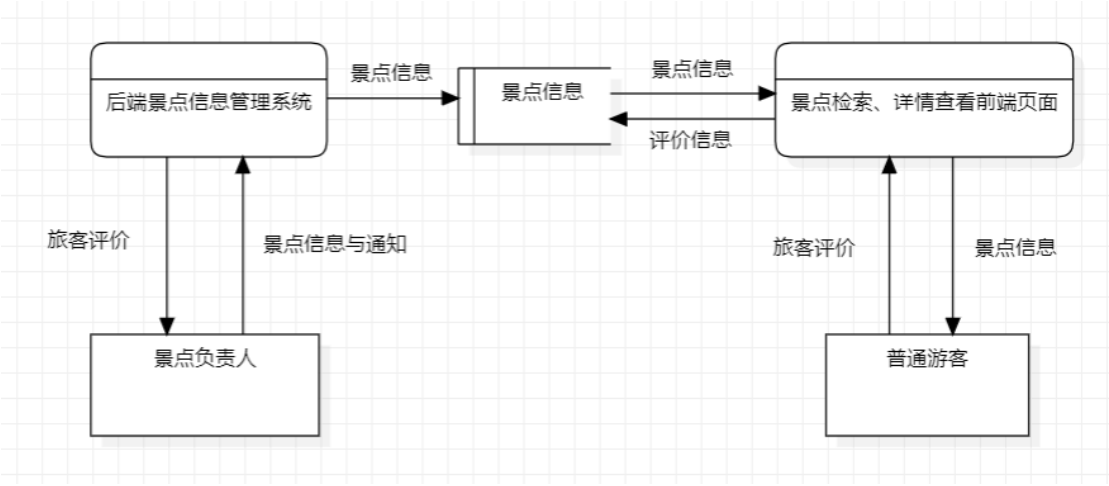
本系统基于城市服务平台，为全体有出行需求的市民和旅客提供信息查询与票务订购服务，同时实现管理人员对全市旅游出行点的管理，在节假日等旅游高峰期，可能出现大量用户集中访问系统的情况，造成访问量和信息量多。为了给用户更好的体验，网站每个页面响应时间<1 分钟，数据查询响应时间<30 秒。

3　数据库设计

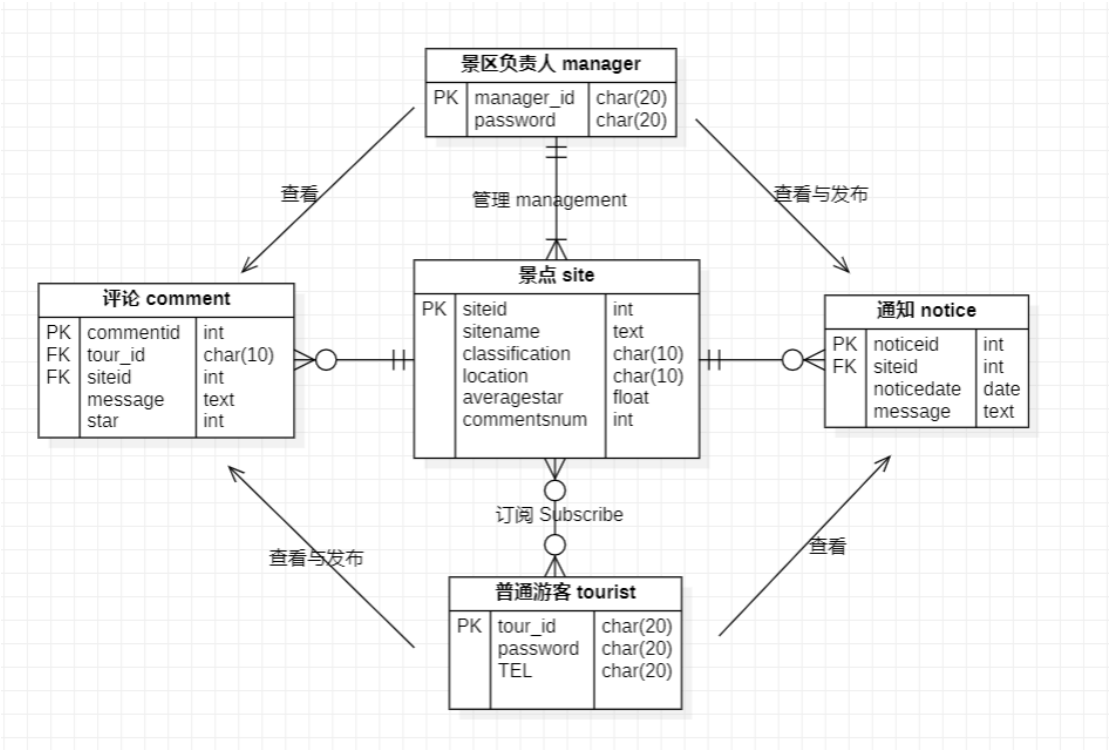
3.1 总体设计

3.1.1　总数据流图

系统以景点信息为核心，结合评价系统与通知系统两个子信息系统，划分景点负责人和普通游客两大用户群体，对三表进行数据的访问与增删改。总体数据流图如下：



3.1.2　总体 E-R 图



3.2 概念设计

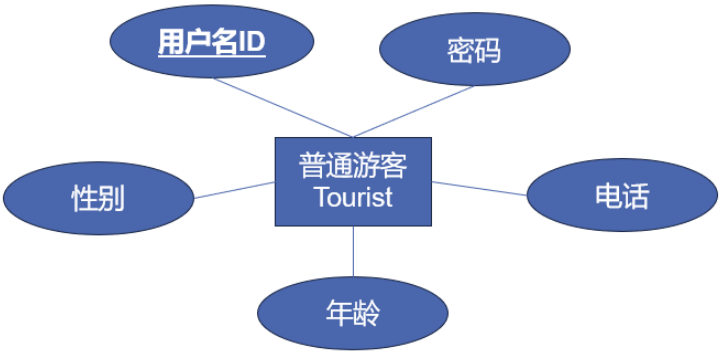
3.2.1 实体设计

- 景点

景点实体包含了景点的基本信息，如景点 ID、名称、分类、所在地、开放时间、评分、评论数、公告数等。这个表格的设计可以作为景点信息管理系统的一部分。主键是景点编号。

- 普通游客

普通游客的基本信息包括唯一游客 ID、密码、手机号码等。这个表格的设计可以作为游客信息管理系统的一部分，为游客提供方便的登录和注册服务。主键是游客 ID。

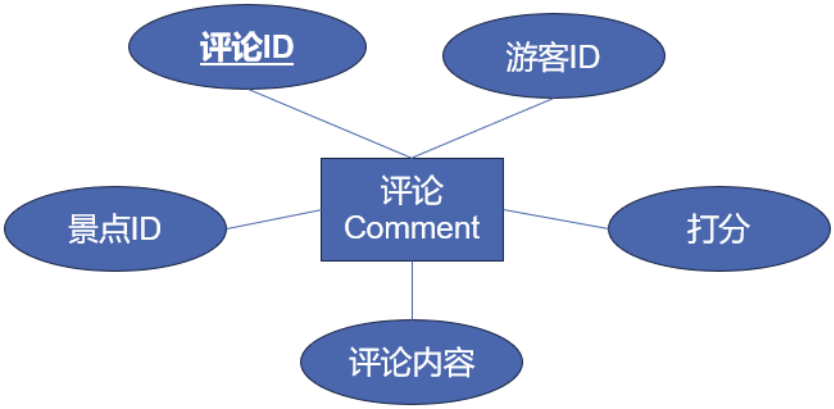


- 景点负责人

景点负责人的基本信息包括唯一负责人 ID、密码等。这个表格的设计可以作为景点管理信息系统的一部分，为景点管理人员提供登录和管理服务。

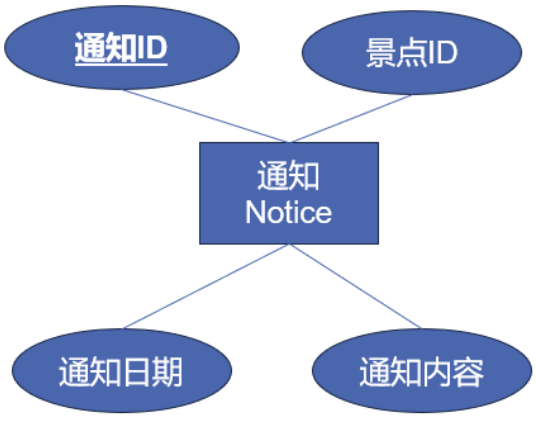
- 评论

  评论的基本信息包括评论 ID、游客 ID、景点 ID、评论内容、评论评分等。这个表格的设计可以作为景点评论系统的一部分，为游客提供评论服务。



- 通知

  通知的基本信息包括通知 ID、景点 ID、通知内容、通知日期等。这个表格的设计可以作为景点通知系统的一部分，为游客提供景点公告、活动等信息，同时也为景点管理人员提供了发布和管理通知的平台。

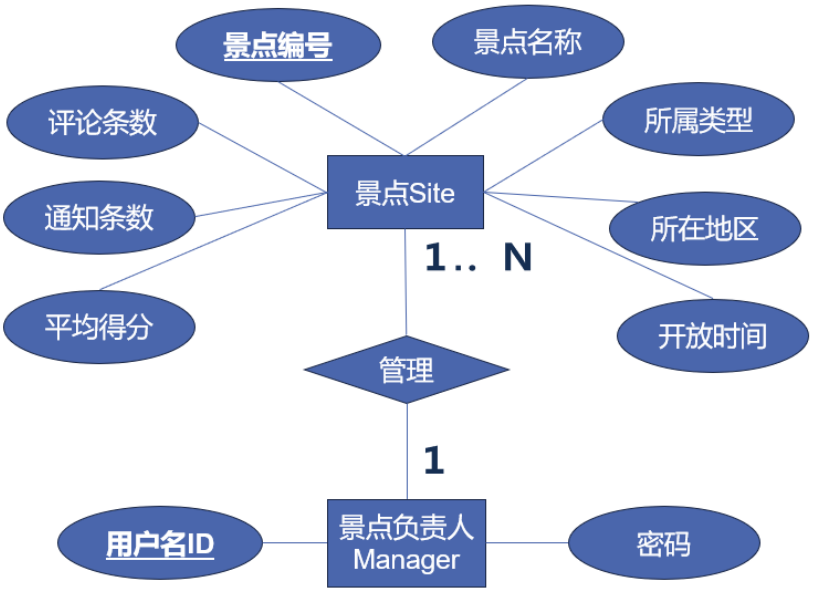


3.2.2 联系设计

- **管理关系**

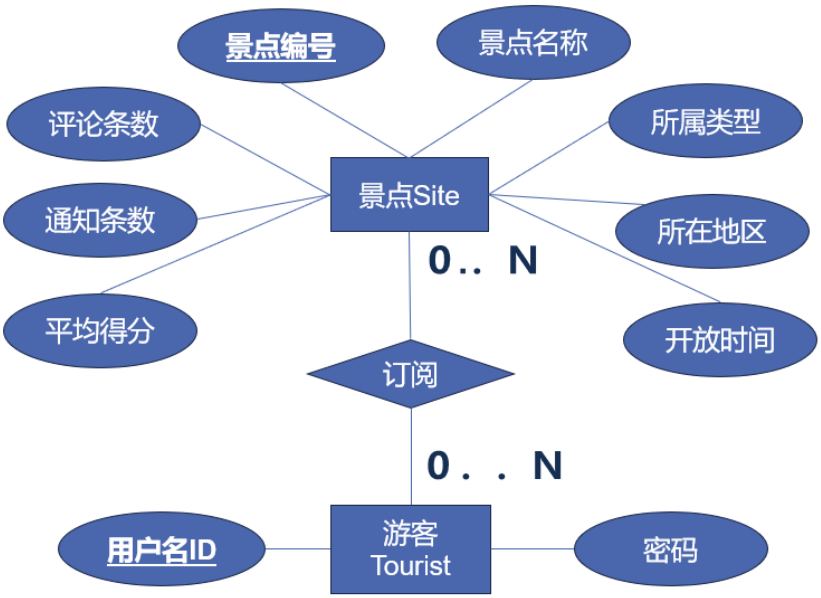
  创建了一个管理关系表，用于记录景点负责人与景点之间的关系。通过设

置复合主键，一个景点负责人可以管理多个景点，一个景点只可被一位景点负责人所管理。



- **订阅关系**

创建了一个订阅关系表，用于记录普通游客与景点之间的关系。通过设置复合主键，一位游客可以关注多个景点，一个景点也可以被多名游客关注。



3.3 逻辑设计

3.3.1 关系模式设计

设计出的关系 E-R 图可以转换为如下关系模型：

旅客用户实体 Tourist_user：Tourist_user(**Tour_id**, Password, TEl)，其中，Tour_id、Password、TEl 分别对应关系模式中的三个属性，表示旅客用户的用户名、登录密码和联系方式。

景点负责人实体 Manager_user：Manager_user(**Manager_id**, Password)，其中，Manager_id、Password 分别对应关系模式中的两个属性，表示景点负责人的用户名和登录密码。

景点实体 Site：Site(**SiteID**, SiteName, Classification, Location, OpenTime, AverageStar, CommentsNum, NoticesNum)，其中，SiteID、SiteName、Classification、Location、OpenTime、AverageStar、CommentsNum、NoticesNum 分别对应关系模式中的八个属性，表示景点的唯一标识符、名称、分类、地理位置、开放时间、平均评分、评论数和通知数。

评论实体 Comment：Comment(**CommentID**, TouristID, SiteID, Content, Star)，其中，CommentID、TouristID、SiteID、Content、Star 分别对应关系模式中的五个属性，表示评论的唯一标识符、游客的唯一标识符、景点的唯一标识符、评论内容和评分。

通知实体 Notice：Notice(**NoticeID**, SiteID, Message, NoticeDate)，其中，NoticeID、SiteID、Message、NoticeDate 分别对应关系模式中的四个属性，表示通知的唯一标识符、景点的唯一标识符、通知内容和通知日期。

订阅关系 Subscribe：Subscribe(**TouristID**, **SiteID**),其中，TouristID、SiteID 分别对应关系模式中的两个属性，表示订阅的游客的唯一标识符和景点的唯一标识符。

管理关系 Management：Management(**ManagerID**, **SiteID**),其中，ManagerID、SiteID 分别对应关系模式中的两个属性，表示管理关系的景点负责人的唯一标识符和管理的景点的唯一标识符。

3.3.2 实体存储结构

- 普通游客

Field Name	Type	Null	Key	Default	Extra	Info
tour_id	VARCHAR(20)	NO	PRI			旅客唯一用户名
Password	VARCHAR(20)	NO				密码
TEl	VARCHAR(20)	NO				手机号码

- 景点负责人

Field Name	Type	Null	Key	Default	Extra	Info
Manager_id	VARCHAR(20)	NO	PRI			管理员唯一用户 ID
Password	VARCHAR(10)	NO				管理员账户密码

- 景点

Field Name	Type	Null	Key	Default	Extra	Info
siteid	int	NO	PRI	NULL	auto_increment	景点 ID
sitename	text	NO		NULL		景点名称
classification	CHAR(10)			'其他景点'		景点分类
location	CHAR(64)			'其他地区'		景点所在地区
opentime	CHAR(64)			'暂无'		景点开放时间
averagestar	FLOAT			0		景点平均评分
commentsnum	INT			0		景点评论数
noticesnum	INT			0		景点公告数

- 评论

Field Name	Type	Null	Default	Extra	Info
comment_id	int	NO	NULL	auto_increment	评论 ID
tour_id	char(10)	YES	NULL	外键，关联 tourist_user 表的 tour_id 字段	评论者唯一用户 ID
siteid	int	YES	NULL	外键，关联 site 表的 siteid	景点 ID

				字段，无更新和删除规则	
content	text	NO	NULL		评论内容
star	float		0		评分

- 通知

Field Name	Type	Null	Extra	Info
noticeid	int	NO	auto_increment	公告 ID
siteid	int	YES	外键，关联 site 表的 siteid 字段	景点 ID
message	TEXT	YES		公告内容
noticedate	CHAR(20)	YES		公告日期

3.3.3 关系表设计

- 订阅关系表

Field Name	Type	Null	Key	Default	Extra	Info
tour_id	CHAR(20)	NO	PRI	NULL	外键，关联 tourist_user 表的 tour_id 字段	用户 ID
siteid	int	NO	PRI	NULL	外键，关联 site 表的 siteid 字段	景点 ID
Primary key			PK		主键，由 tour_id 和 siteid 两列组成	

- 管理关系表

Field Name	Type	Null	Key	Default	Extra	Info
manager_id	CHAR(20)	NO	PRI	NULL	外键，关联 manager_user 表的 manager_id 字段	管理员 ID
siteid	int	NO	PRI	NULL	外键，关联 site 表的 siteid 字段	景点 ID

Primary key			PK		主键，由 manager_id 和 siteid 两列组成	

3.4 物理设计

3.4.1 事务数据访问特性

① 数据库查询事务

- 用户登录

【事务内容】在登陆页面用户输入用户名和密码，系统在用户表中查找是否匹配。

【操作频率】10000 次/日。无论是普通游客还是景区负责人都需要登陆后才能使用平台的功能。

【性能要求】2s/操作

- 用户查询景点信息

【事务内容】用户查看某个景点信息，包括景点名称、类型、地址、开放时间、平均分等。此项操作需在订根据用户的查询方式检索景点表，又可能是对景区名的模糊检索或对某些属性的筛选。

【操作频率】10000 次/日。 这是平台最主要实现的功能，访问量会非常大，

【性能要求】3s/操作

- 游客和景区负责人查询景点详情

【事务内容】游客和景点负责人查看某个景点的详细信息，主要包括游客评价与打分以及该景点发布的通知。

【操作频率】5000 次/日。 游客在筛选出感兴趣的景点时往往会进一步查看评论区与通知。因此访问量不低。

【性能要求】2s/操作

- 游客查询已关注景区通知

【事务内容】游客打开个人消息，一次性接收并查看自己关注的所有景点发布

的所有通知列表。 系统同时访问订阅表与通知表，关联查找出数据。

【操作频率】1000 次/日。

【性能要求】2s/操作

② 数据增加事务

- 用户注册

【事务内容】用户在平台的注册页面输入账户名及密码等基本信息，系统在用户表里添加相应的记录。

【操作频率】1000 次/日。

【性能要求】2s/操作

- 新建景点

【事务内容】平台管理员新建景点，插入到景点表中。新建的景点必须包含姓名这项基本信息，其他信息由默认约束补全。

【操作频率】100 次/日。

【性能要求】3s/操作

- 游客发布打分与评价

【事务内容】游客在景区的详情页面中发布自己对该景区的打分与评价，发布后的数据插入到评论表中，并实时显示在评论区。

【操作频率】5000 次/日。

【性能要求】3s/操作

- 景点负责人发布景点通知

【事务内容】景点负责人在景区的详情页面中发布景区的通知，发布后的数据插入到通知表中，并实时显示在消息列表中。

【操作频率】1000 次/日。

【性能要求】3s/操作

- 游客关注景区

【事务内容】游客在景区的详情页面中点击“订阅”按钮，系统在订阅表中新建一条数据表示该用户与该景区的订阅关系。此后用户可以在消息列表中直接查收该景区的通知。

【操作频率】1000 次/日。

【性能要求】3s/操作

③ 数据更新事务

- 维护景点信息

【事务内容】平台管理员选择某项自己管理的景点，更新该景点的所属类型、所在地点等基本信息，后台修改对应表项。

【操作频率】100 次/日。

【性能要求】3s/操作

- 景区平均分计算

【事务内容】 每当有游客上传新的打分时，系统自动对该景点的平均分进行依次更新。平均分计算公式为当前所有评论打分的算术平均值。

【操作频率】5000 次/日。

【性能要求】3s/操作

④ 数据删除事务

- 景区负责人删除景点

【事务内容】平台管理员删除景点，自动移除该景点在所有表中的对应数据。

【操作频率】10 次/日。

【性能要求】3s/操作

- 游客取消关注景区

【事务内容】游客在景区的详情页面中点击“取消订阅”按钮，系统在订阅表中删除表示该用户与该景区的订阅关系的数据。此后用户的消息列表不再接收

该景点的通知。

【操作频率】1000 次/日。

【性能要求】3s/操作

3.4.2 安全性设计

数据库安全性设计是保护数据库中数据安全的重要步骤。以下是本项目在设计过程中计划采取的措施：

- 访问控制：限制用户对数据库的访问权限，只有授权用户才能够访问数据库。可以通过使用用户名、密码、角色或其他身份验证机制来实现访问控制。
- 数据备份和恢复：对数据库中的数据进行备份，以防止数据丢失或损坏。同时，也需要建立完善的数据恢复机制，以便在数据出现问题时能够及时恢复数据。
- 数据操作：避免直接在生产环境上操作数据，而是通过测试、预发布等环境来进行验证，确保数据安全。

3.4.3 完整性设计

- 普通游客

  tour_id VARCHAR(20) PRIMARY KEY NOT NULL：游客 ID，为 varchar 类型，长度为 20，是主键，不能为空。

  Password VARCHAR(20) NOT NULL：游客密码，为 varchar 类型，长度为 20，不能为空。

  TEl VARCHAR(20) NOT NULL：游客手机号码，为 varchar 类型，长度为 20，不能为空。

- 景点负责人

  Manager_id VARCHAR(20) PRIMARY KEY：景点负责人 ID，为 varchar 类型，长度为 20，是主键。

  Password VARCHAR(10) NOT NULL：景点负责人密码，为 varchar 类型，长度为 10，不能为空。

- 景点

  siteid int PRIMARY KEY AUTO_INCREMENT：景点 ID，为 int 类型，是主键，且自动递增。

  sitename text not null：景点名称，为 text 类型，不能为空。

  classification CHAR(10) default '其他景点'：景点分类，为 char 类型，长度为 10，如果未指定分类，则默认为“其他景点”，取值范围=[人文景观、自然风光、主题乐园、消费场所、拍照打卡、其他景点……]。

  location CHAR(64) default '其他地区'：景点所在地，为 char 类型，长度为 64，如果未指定地区，则默认为“其他地区”，取值范围为上海市主要的行政区和其他地区。

  opentime CHAR(64) default '暂无'：景点开放时间，为 char 类型，长度为 64，如果未指定开放时间，则默认为“暂无”。

  averagestar FLOAT default 0：景点评分，为 float 类型，如果未有评分，则默认为 0。

  commentsnum INT DEFAULT 0：景点评论数，为 int 类型，如果未有评论，则默认为 0。

  noticesnum INT DEFAULT 0：景点公告数，为 int 类型，如果未有公告，则默认为 0。

  AUTO_INCREMENT=3：表示从 id 为 3 的行开始自动递增。

  DEFAULT CHARSET=utf8mb4：设定编码方式为 utf8mb4，以支持中文等多种字符集。

- 评论

  comment_id int PRIMARY KEY AUTO_INCREMENT：评论 ID，为 int 类型，是主键，且自动递增。

  tour_id CHAR(10)：游客 ID，为 char 类型，长度为 10。

  siteid int：景点 ID，为 int 类型。

  content TEXT NOT NULL：评论内容，为 text 类型，不能为空。

  star FLOAT DEFAULT 0：评论评分，为 float 类型，默认值为 0。

FOREIGN KEY (tour_id) REFERENCES tourist_user(tour_id)：外键约束，关联 tourist_user 表格中的 tour_id 字段。

FOREIGN KEY (siteid) REFERENCES site(siteid)：外键约束，关联 site 表格中的 siteid 字段。

AUTO_INCREMENT=3：表示从 id 为 3 的行开始自动递增。

DEFAULT CHARSET=utf8mb4：设定编码方式为 utf8mb4，以支持中文等多种字符集。

- 通知

  noticeid int primary key AUTO_INCREMENT：通知 ID，为 int 类型，是主键，且自动递增。

  siteid int：景点 ID，为 int 类型。

  message TEXT：通知内容，为 text 类型。

  noticedate char(20)：通知日期，为 char 类型，长度为 20。

  FOREIGN KEY (siteid) REFERENCES site(siteid)：外键约束，关联 site 表格中的 siteid 字段。

  AUTO_INCREMENT=3：表示从 id 为 3 的行开始自动递增。

  DEFAULT CHARSET=utf8mb4：设定编码方式为 utf8mb4，以支持中文等多种字符集。

- 订阅

  tour_id CHAR(20) NOT NULL：这是一个 CHAR 类型的属性，用于存储旅客用户的唯一标识符，长度为 20，且不能为空。

  siteid int NOT NULL：这是一个 INT 类型的属性，用于存储景点的唯一标识符，且不能为空。

  PRIMARY KEY (tour_id, siteid)：设置了复合主键，由 tour_id 和 siteid 两个属性组成。

  FOREIGN KEY (tour_id) REFERENCES tourist_user(tour_id)：设置了外键约束，将 tour_id 属性与 tourist_user 表格中的 tour_id 属性关联，以确保在插入

或更新数据时，tour_id 属性的值必须在 tourist_user 表格中存在。这样可以保证订阅关系表中的旅客用户 ID 的正确性和一致性。

FOREIGN KEY (siteid) REFERENCES site(siteid)：这一行代码设置了外键约束，将 siteid 属性与 site 表格中的 siteid 属性关联，以确保在插入或更新数据时，siteid 属性的值必须在 site 表格中存在。这样可以保证订阅关系表中的景点 ID 的正确性和一致性。

- 管理

  manager_id CHAR(20) NOT NULL：这是一个 CHAR 类型的属性，用于存储景点负责人的唯一标识符，长度为 20，且不能为空。

  siteid int NOT NULL：这是一个 INT 类型的属性，用于存储景点的唯一标识符，且不能为空。

  PRIMARY KEY (manager_id, siteid)：设置复合主键，由 manager_id 和 siteid 两个属性组成。这意味着在表格中，同一个景点也不能被不同的景点负责人管理。

  FOREIGN KEY (manager_id) REFERENCES manager_user(manager_id)：设置了外键约束，将 manager_id 属性与 manager_user 表格中的 manager_id 属性关联，以确保在插入或更新数据时，manager_id 属性的值必须在 manager_user 表格中存在。这样可以保证管理关系表中的景点负责人 ID 的正确性和一致性。

  FOREIGN KEY (siteid) REFERENCES site(siteid)：设置了外键约束，将 siteid 属性与 site 表格中的 siteid 属性关联，以确保在插入或更新数据时，siteid 属性的值必须在 site 表格中存在。这样可以保证管理关系表中的景点 ID 的正确性和一致性。

4　系统设计

4.1 系统构建

4.1.1　开发环境

在项目需求分析与技术调研的基础上，计划采取 Web 端网页的形式进行项目本地开发调试，具体开发环境如下：

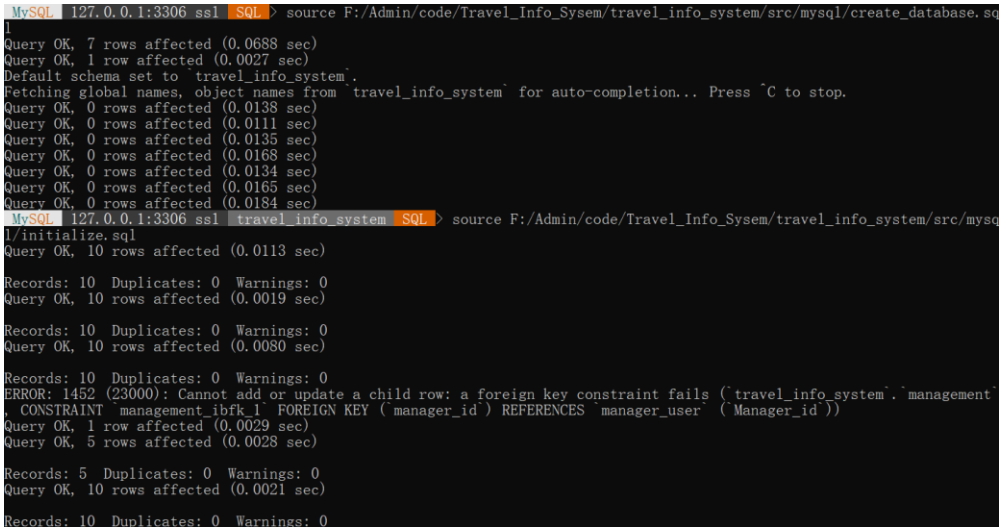
- 前端框架：vue 框架（基于 html+css+JavaScript）+ Element UI 组件
- 后端框架：Express + Node.js
- 接口 API 管理：apiFox
- 数据库管理：MySQL 关系型数据库
- 本地环境：Window 10 64 位
- 开发工具：VS Code + Edge 浏览器 + MySQL WorkBench 8.0 CE

4.1.2　项目结构

```
+---build 、config 、 node_modules        //Vue 资源文件
+---server                                //后端服务相关文件
|   db.js                                 //数据库配置
|   index.js                              //Node.js 服务配置
|   \---api                               //数据库访问相关 API 接口
        commentApi.js
        managerApi.js
        noticeApi.js
        siteApi.js
        userApi.js
+---src                                   //前端 Vue 组件
|   +---mysql                             //创建并初始化数据库相关文件
|   +---components                        //主要页面
|   |   \---childcomponents               //重复利用的子组件
|   \---router                            //页面跳转路由配置
+---static                                //图片等静态资源
```

4.1.3 项目初始化

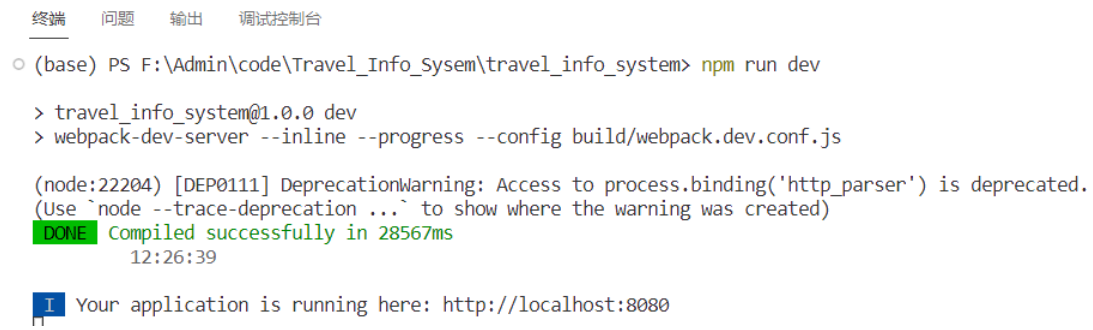
项目运行起始，需要对数据库进行初始化，主要的操作包括新建数据库、新建各数据表与默认的一批景点的插入。可在 MySQL Shell 中利用 Source 命令依次执行 src/mysql 文件夹下的 create_datebase.sql 与 initialize.sql 文件进行初始化。



完成数据库初始化后，可在命令行终端分别启动前后端服务。

前端启动命令：

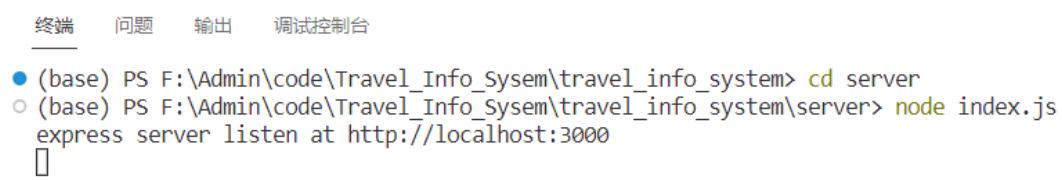
--- npm run dev



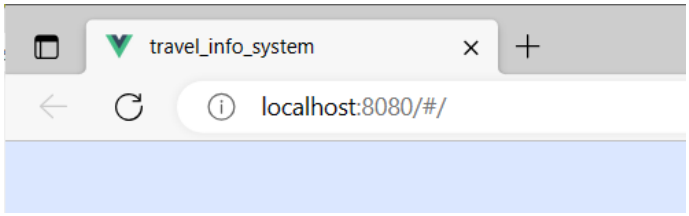
后端启动命令：

--- cd server

--- node index.js



运行成功后，项目开始工作，可以进入 locolhost:8080 端口体验系统。



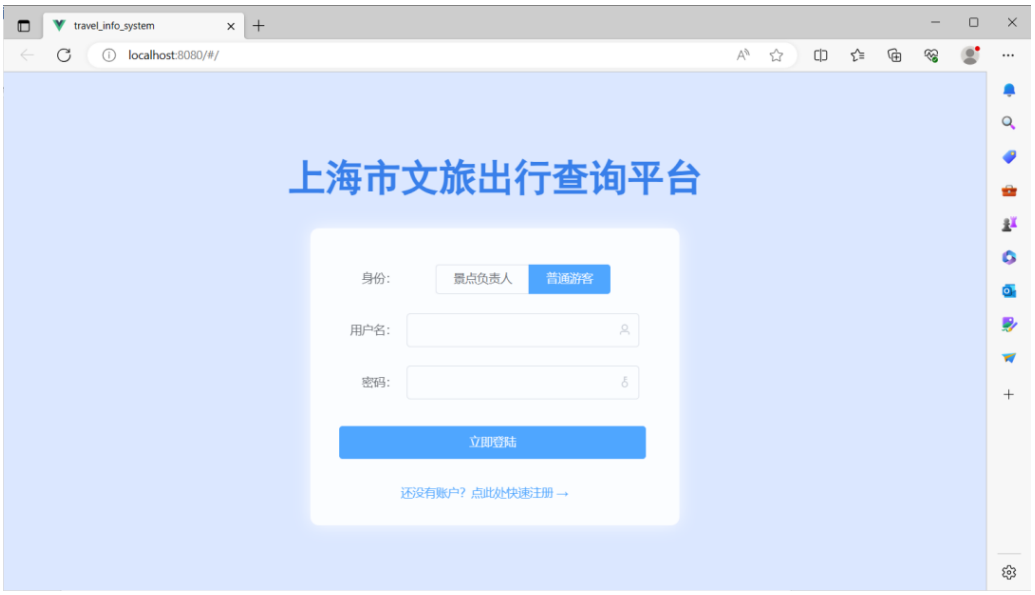
4.2 前端设计

本平台的前端设计采取 vue 脚手架配合 element UI 组件库的形式实现。本部分介绍其中的某些关键页面的实现。

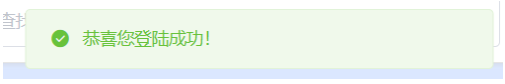
4.2.1 登陆界面

登录页面是一个用于普通游客和经典管理员的登录，包括网站标题、用户名和密码输入框、登录按钮、注册提示等组件。为了提高用户体验，登录页面的设计从以下几个方面入手：

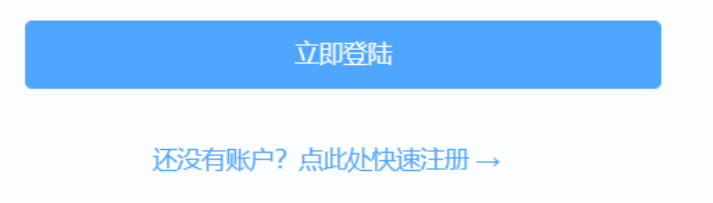
- 页面布局：登录页面的布局简洁明了，让用户能够快速找到登录入口。页面中上部为网站标题，随后为身份选择和用户名和密码输入框，登录按钮应该放在输入框下方，按钮下方给出注册提示。



- 输入框设计：输入框的设计需要考虑到用户的易用性和美观性。对于用户身份选择采取单选框形式，对于用户名和密码输入采用文本框的形式。输入框配有清晰的标签和提示信息，用户输入时应该有明显的反馈和提示。同时，输入框的样式也应该与整个页面的风格相一致。



- 按钮设计：登录按钮应该具有明显的颜色和样式，区别于其他按钮和组件。因此在这里采用醒目的蓝色。同时，按钮的大小和位置也应该与输入框相匹配，让整个页面的布局看起来协调和谐。



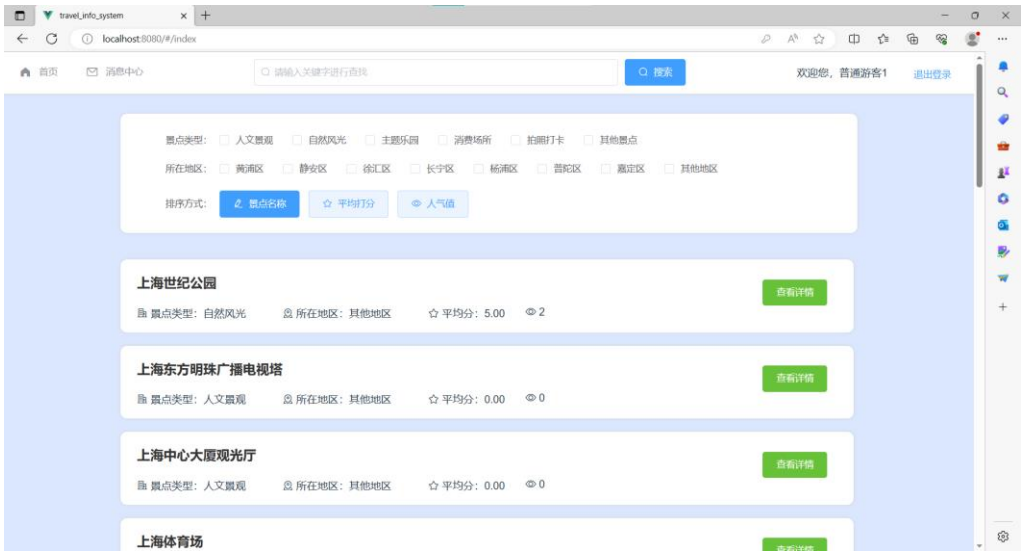
从登录界面衍生出用户的注册界面，可以在登陆界面中点击“注册”前往。注册界面的整体布局设计与登陆界面相似，保持整个平台的和谐统一。同时将原有“注册”跳转的位置改为“返回登陆界面”，方便用户操作。

- 跳转设计：为了简便用户的操作，给用户更好的体验，在这个界面我们设置当成功注册时，自动返回登录页面，当成功登录时自动跳转到网站主页，减少用户点击的次数

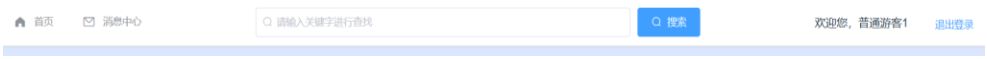
4.2.2 系统主页

系统主页是用户登陆后自动跳转的系统主界面，用于实现系统的核心功能：景点信息管理与查询。普通游客的系统主页的组成包括导航栏、查询条件以及景点列表。为了提高用户体验，系统主页的设计从以下几个方面入手：

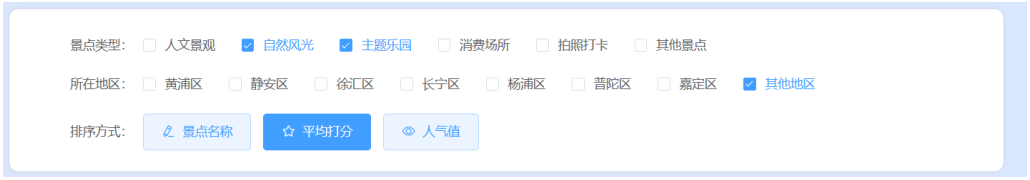
- 页面布局：系统主页应该具有整洁、全面的页面布局，让用户可以快速掌握平台的各种功能。一个好的主页首先应该设计有导航栏，方便用户进行快速的页面跳转。由于本网站的核心功能是景点查询，因此主页的主体部分为景点列表的显示。在列表与导航栏之间插入一个自定义筛选条件框，方便普通游客进行个性化的查询。



- 导航栏设计：导航栏位于页面的顶部，以便用户能够轻松找到需要的内容。导航栏的样式需要与整个页面的风格相一致，包括颜色、字体、图标等。导航栏左部是两个页面跳转，“首页”用于从其他页面跳转回首页，“消息中心”用于查看订阅消息。中部“搜索框”用于景点名称的模糊查找。右部为欢迎语句，通过访问用户名显示，最右为“退出登陆”的操作，可以返回登陆页面。导航栏组件被复用于本平台除登录界面以外的所有界面。



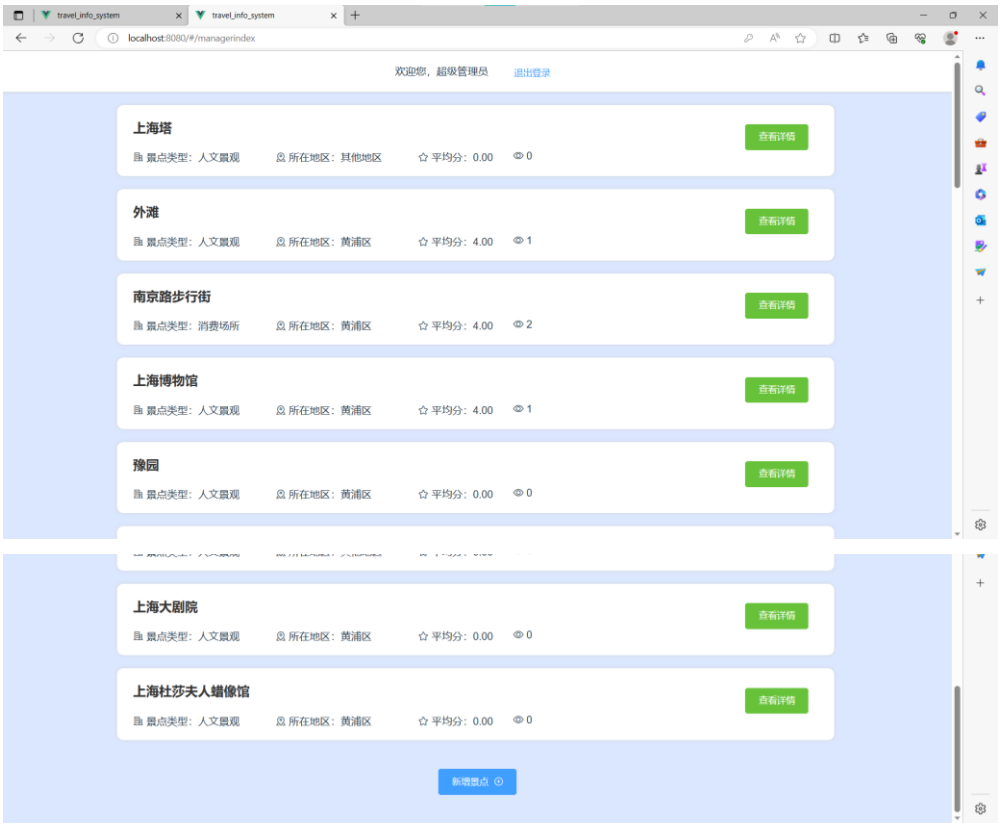
- 筛选框设计：筛选框部分用于实现普通游客的按需查找功能。为了减少用户的手动输入，采用多选项的形式进行交互。标签和选项标注清晰便于查找。排序采用按钮的形式，选中的方式变为亮色，表示当前排序按照的是这样的方式。



若按条件筛选出符合条件的景点个数为 0，则会提示“暂无符合条件的景区”并重置搜索条件：

- 展示列表设计：系统主页的主要功能为景点信息的展示。在页面主体部分采用卡片列表的形式。每张卡片表示一个景点，包括景点名称、类型、所在地区、平均分和人气值（当前评论数）。卡片右边部分为“查看详情”按钮，点击可以跳转到详情互动页面。

景点管理员端的系统主页与游客端的界面大致相同，区别在于简化了导航栏的内容，以及删去了自定义查询筛选框，页面的底部为添加景点按钮。

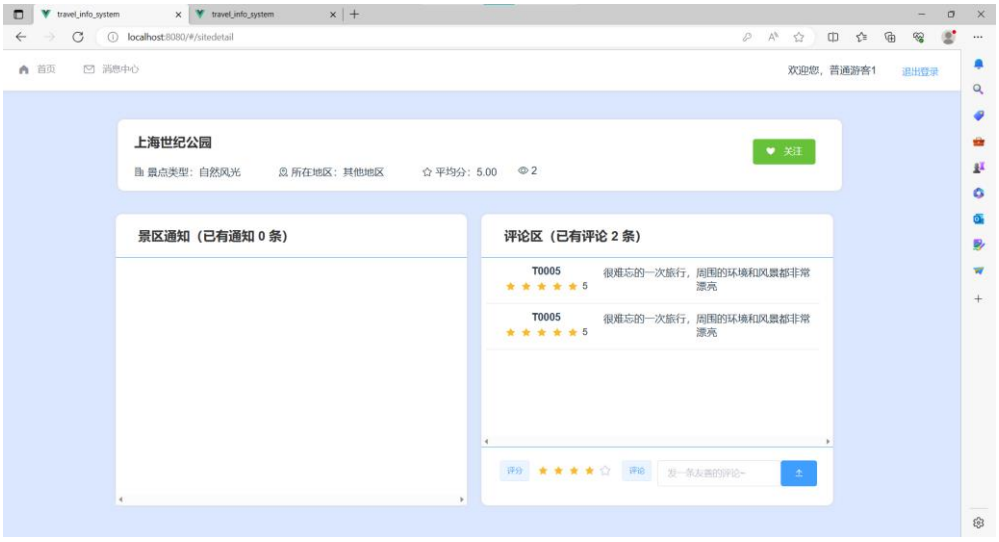


4.2.3 详情互动页面

详情互动页面展示的是某个景点的具体信息，包括导航栏、景点信息、评论区和通知区几个组件。普通游客可以选择订阅/取消订阅该景区，并发表评论。

景点管理员可以选择修改景点信息，并发布通知。为了提高用户体验，系统主页的设计从以下几个方面入手：

- 页面布局：游客端详情互动页面包括顶部的导航栏、首部的选择景点的简介，主体部分以左右两栏的形式展示景区通知列表以及评论区列表。



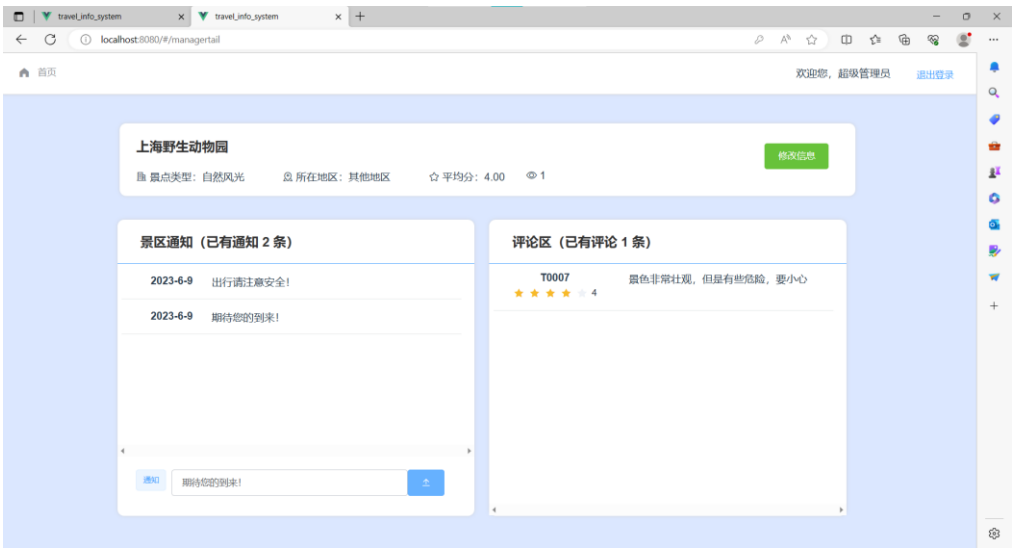
- 导航栏设计：本页面的导航栏与主页是同一个组件的复用，不过考虑到网页的实际功能，禁用并且不显示搜索框。

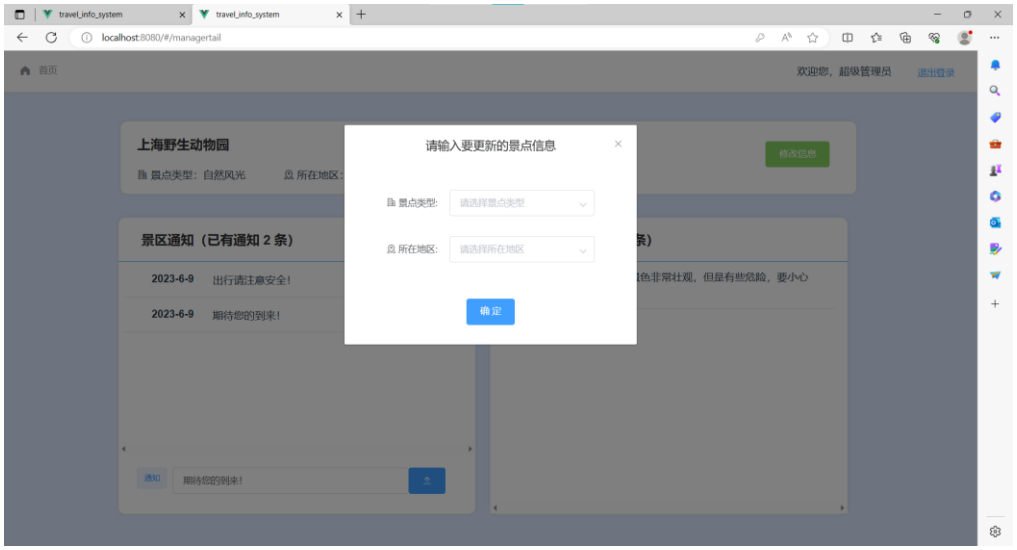
- 景点卡片设计：本页面仅展示某一特定景点的详细信息，因此游客端页面中，右端的按钮显示为“订阅”或“取消订阅”，游客通过点击按钮可以实现景点的订阅与取消。

- 通知区设计：通知区以列表的样式获取该景区发布的通知并展示，每条通知包含发布时间和内容。

- 评论区设计：评论区以列表的样式获取该景区收到的评论并展示，每条评论包含发布用户名、打分和内容。打分采用五星形式展现：

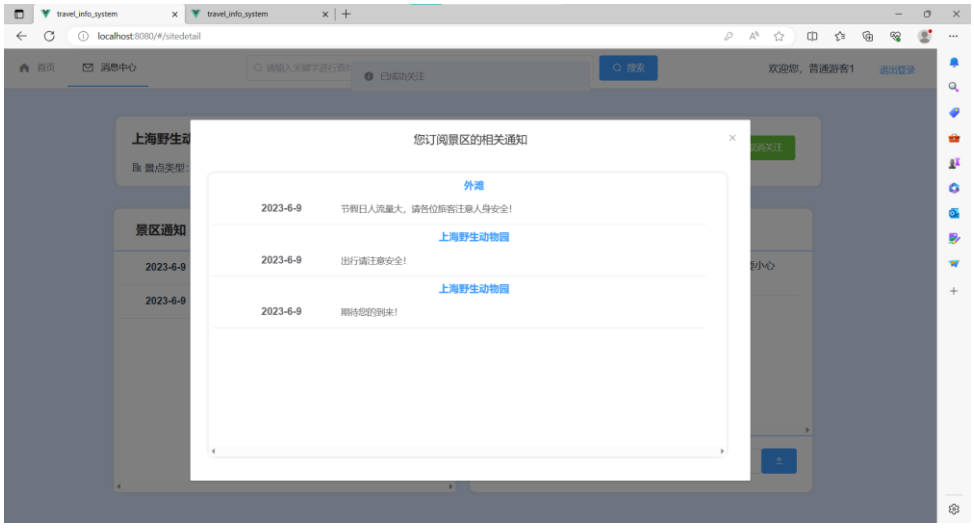
对于景点负责人端，则将订阅按钮切换为了修改景区信箱的按钮，点击会弹出修改表单。同时景区负责人可发布通知、不可发表评论。





4.2.4 订阅信箱

普通游客在订阅了一些列景区后，可以通过订阅信箱一次性查看所有订阅景区的通知。订阅信箱的跳转入口为导航栏中的“我的消息”，点击后弹出显示信箱。信箱内部布局与详情互动页中的通知区类似，增加了通知对应景点名称的显示。景点负责人端不含此页面。



4.3 后端设计

项目后端使用 Node.js 工具，与 Mysql 本地数据库建立连接。本部分选取项目中部分核心功能的后端逻辑进行介绍。

4.3.1 用户身份认证

用户身份认证是一个核心的后端功能，目的是验证用户身份并授权访问受保护的资源。登录功能需要保证用户的账号和密码安全，同时提供友好的反馈信息。前端收集到用户名与密码后通过请求后端接口，后端访问数据库中的旅游信息表或管理员信息表，首先判断该用户是否存在，若存在则进一步验证密码是否匹配。

4.3.2 唯一 ID 分配

在 MySQL 中，使用 AUTO_INCREMENT 可以通过在创建表时为主键列指定 AUTO_INCREMENT 属性，从而开启自动增长功能。

AUTO_INCREMENT 是 MySQL 数据库中常用的一个特性，用于自动管理表中整数类型的主键 ID。通常情况下，使用 AUTO_INCREMENT 可以使得每次插入新记录时，系统自动为记录生成一个唯一的主键 ID，省去了手动填写 ID 的麻烦，也避免了 ID 分配重复、出错。

本项目中采取 AUTO_INCREMENT 进行 ID 管理的表有景点表、评论表、通知表。ID 的起始值为 3.

4.3.3 景点筛查

本平台提供景点筛查选取功能，包括景点名称关键字模糊搜索、按所在地区筛选、按景点类型筛选以及按照不同的顺序进行列表的排序。前端以多选框的样式状态，获取用户的筛选行为，收集需要筛选的条件返回后端。用户勾选的所在地区和景点类型内部的关系为“或”关系，只要满足其中一个勾选条件，就算符合搜索条件的结果。而景点类型与所在地区之间的关系为“与”关系，必须同时符合类型与地区才算符合搜索条件的结果。景点名称关键字模糊搜索，通过文本输入框获取用户输入的关键字，并在数据库中查询景点名称含有此关键字字段的项返回。

4.3.4 景点评分计算

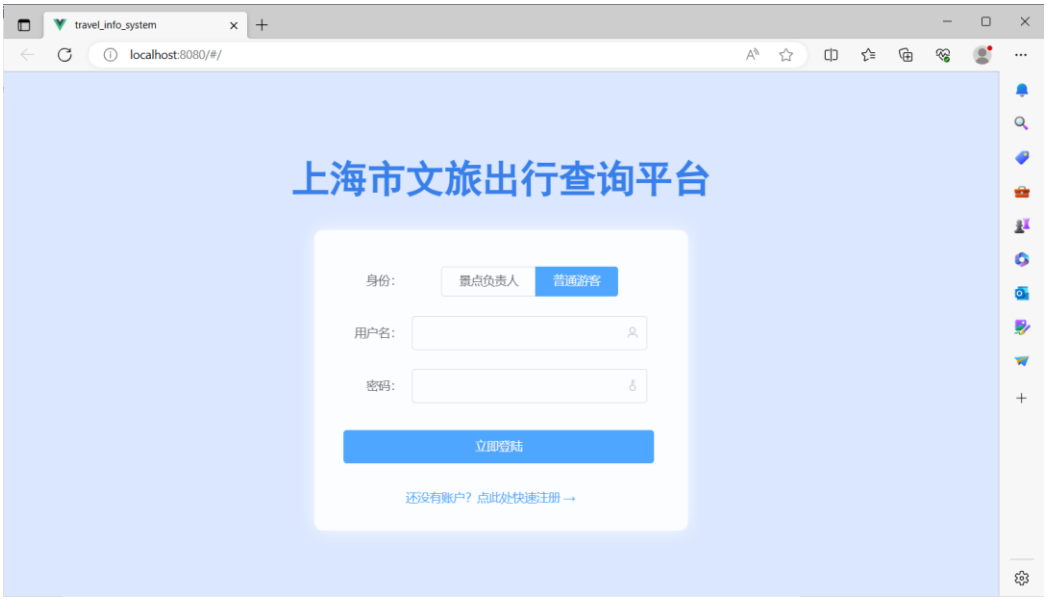
景点的评分由后台自动计算返回，计算的公式为将当前所有评论的打分相加，再除以评论总数，得到算术平均值。景点的打分按照五星制计算，每个游客在评论时可以给景点打一到五之间的任意整数颗星。在每次收到新的评论时，后台都会重新计算一次被评论景点的分数，刷新后就会实时的显示在网页里。

5 系统测试

5.1 项目展示

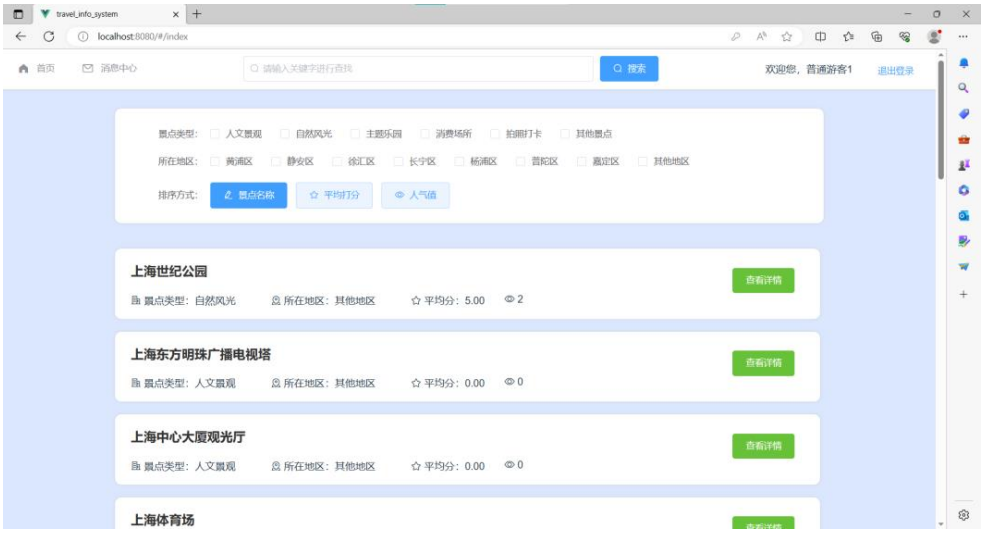
经过不断的调试与优化，最终实现了四大功能模块，搭建起主要的前端网页界面四个，建立与本地 MySQL 数据库的联系，通过数据库的 CRUD 操作实现网页内容的动态更新。页面展示如下，具体的内容可以观看前端设计部分的描述或录屏演示：

- 用户登陆页面

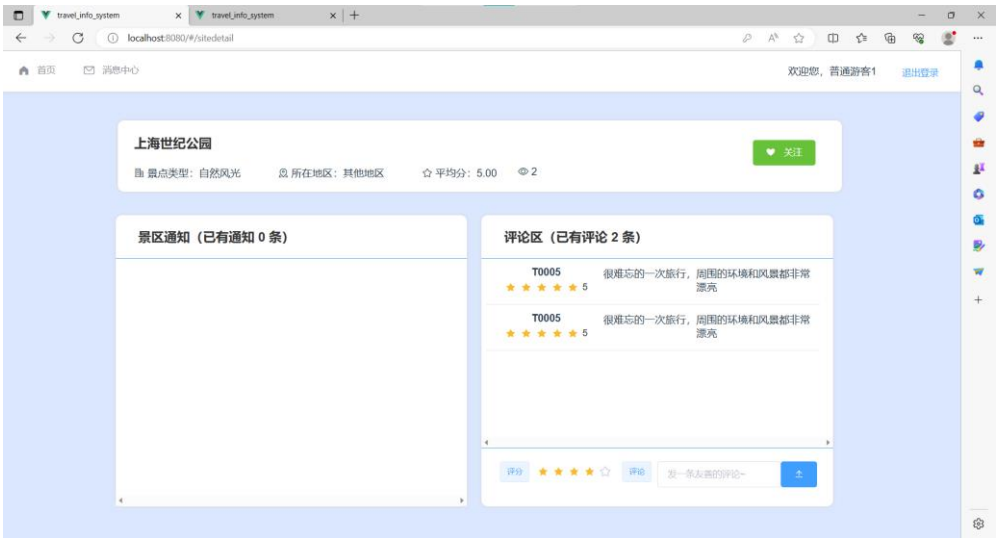


- 用户注册界面

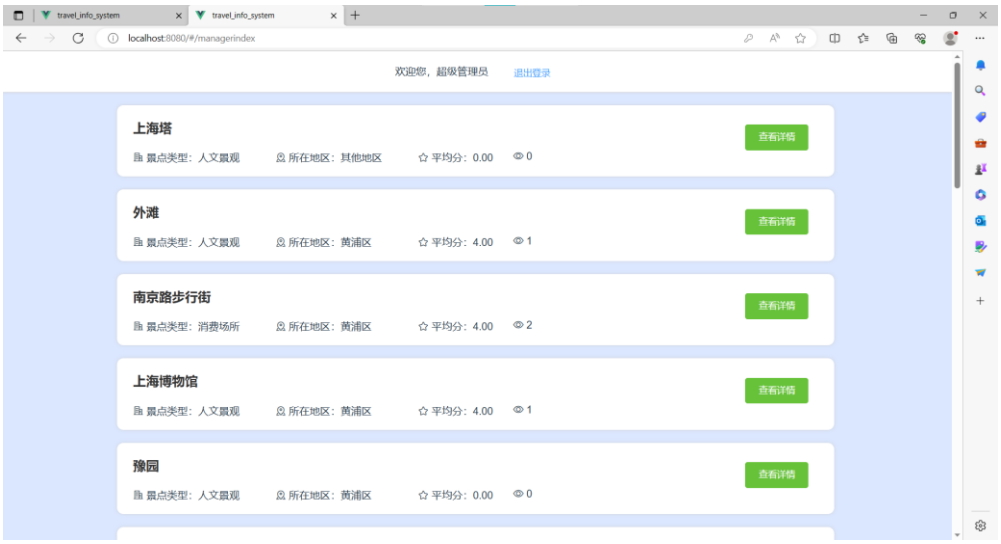
- 游客访问主页



- 游客互动页面

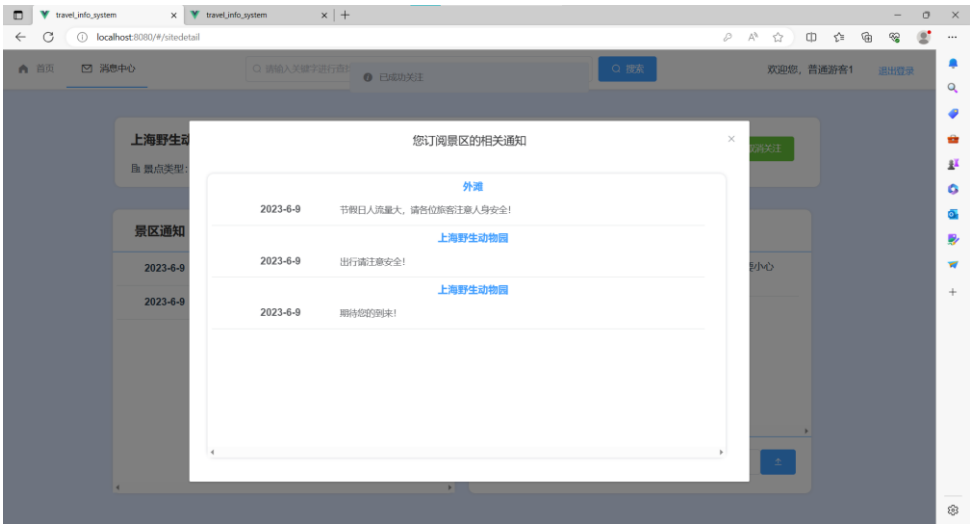


- 管理员主页



- 管理员互动页面

- 游客个人信箱



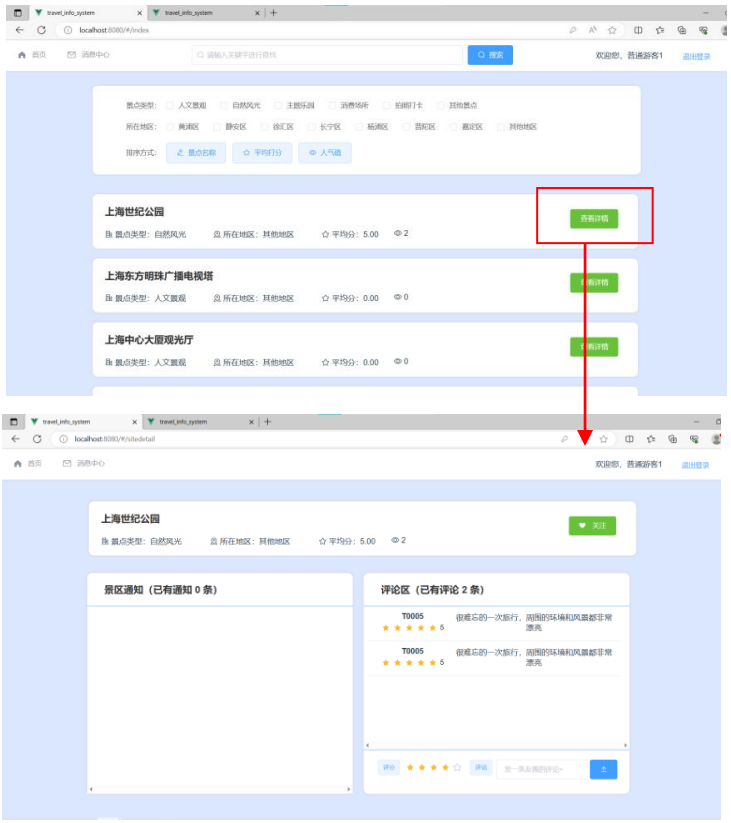
5.2 界面测试

测试方法为点击各页面中包含跳转操作的各个控件，观察跳转是否符合预期。经过操作，所有组件均通过测试。具体的界面跳转情况可观看录屏演示。

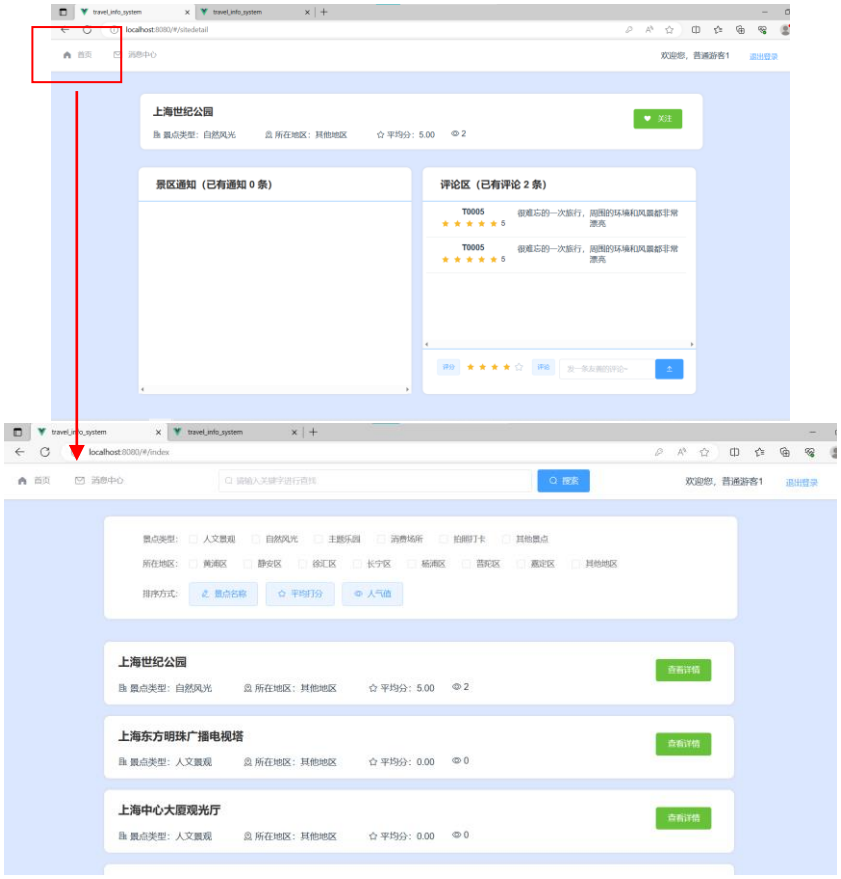
- 登录界面与注册界面的相互切换

- 成功登陆后自动跳转至主页

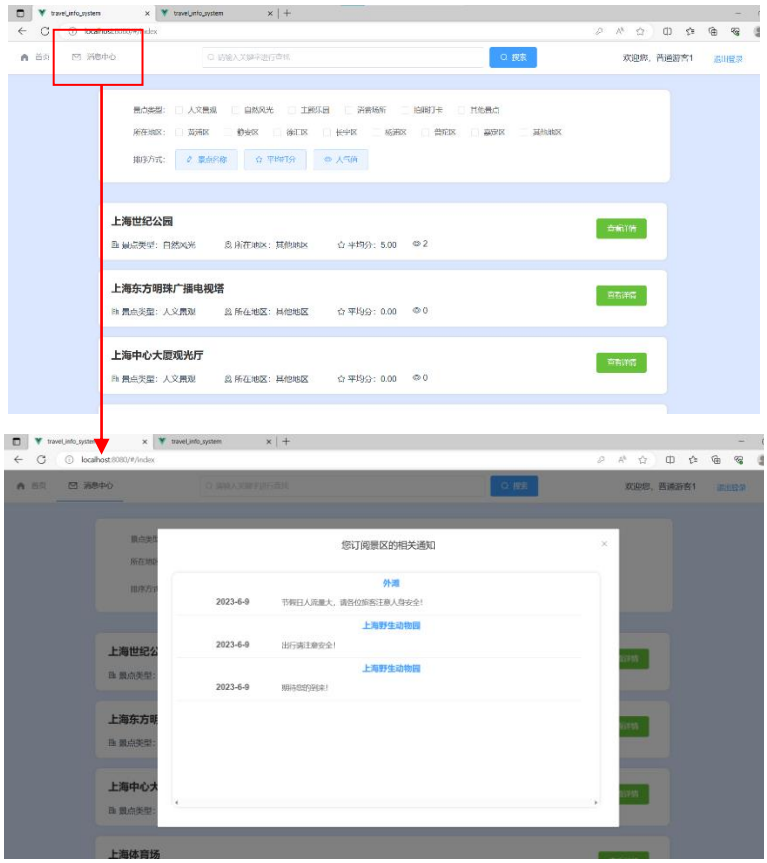
- 跳转至景点详情互动页面



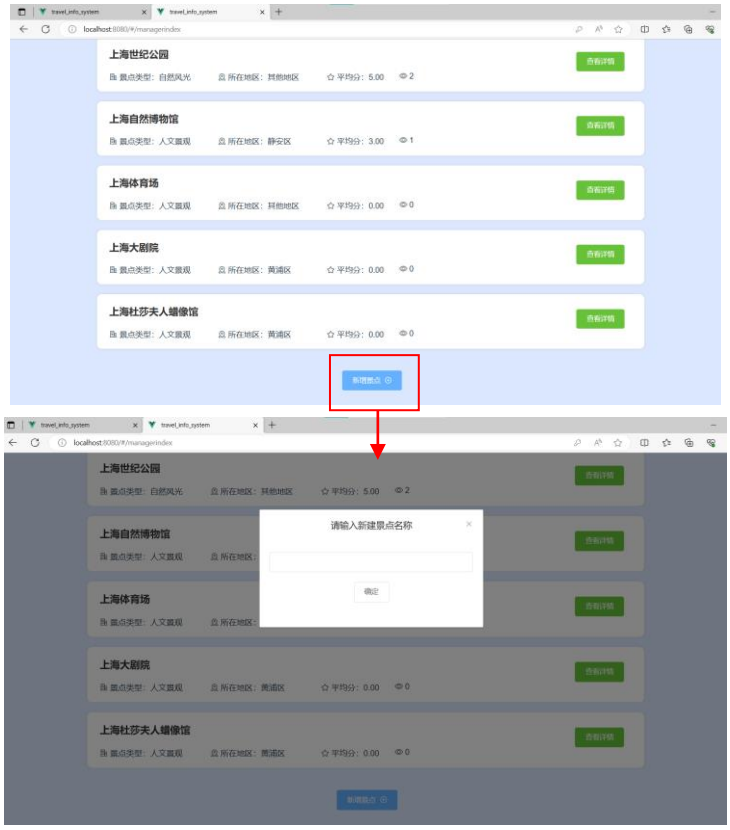
- 点击导航栏中“首页”返回



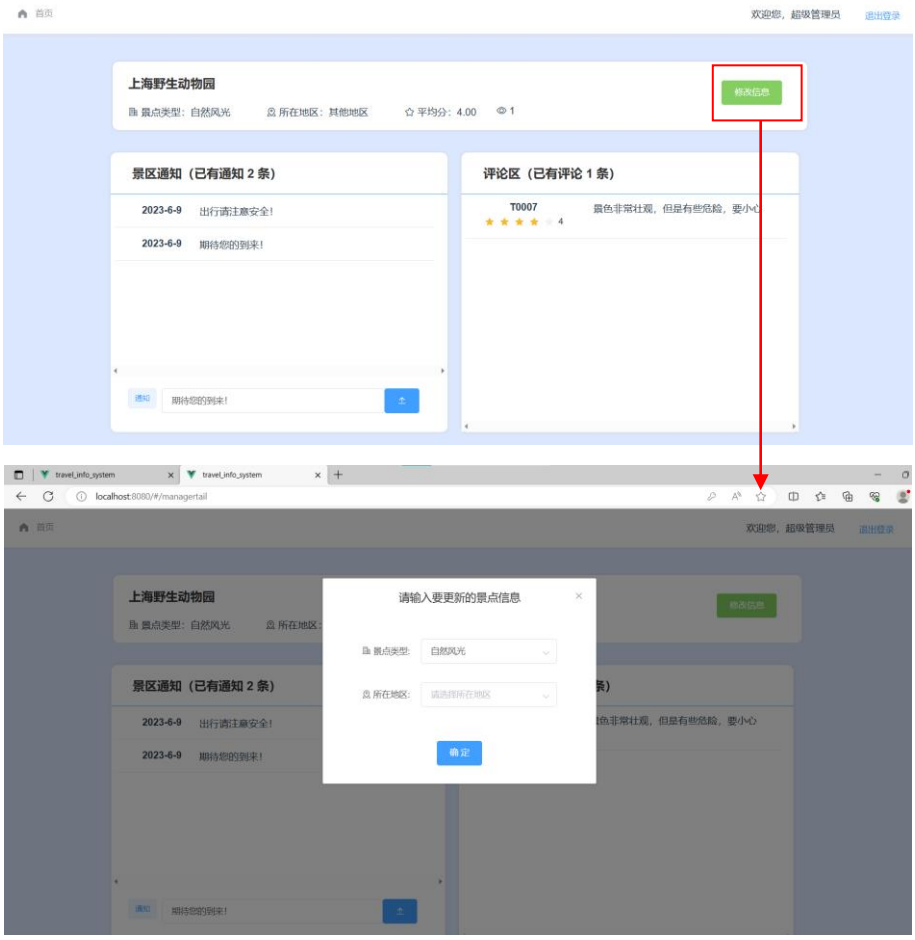
- 点击导航栏中的“消息中心”查看订阅景点通知



- 管理员新增景点



- 管理员修改景点信息



5.3 功能测试

对于平台的功能测试我们采用黑盒测试的方式。黑盒测试，顾名思义，就是把测试对象作为一个整体，看作一个黑盒子。我们只关注盒子的输入和输出是否正确，中间是如何实现的，具体的实现步骤，处理过程和对象的内部结构在测试过程中我们都不关心。这样很好的测试软件功能的正确性。根据平台的功能需求与实际特性，可以划分为以下几个测试用例进行测试：

序号	功能名称	功能描述	输入	输出
1	用户注册	给对应身份的用户表新增一条记录。	输入正确的用户名、密码和确认密码，测试注册是否成功。	注册成功
2	用户登录	①输入正确的用户名密码则登录。	输入正确的用户名和密码	登陆成功
			输入不存在的用户名	提示用户不存在

		②输入错误则提示用户名密码误。③用户没有注册用户名登录，则提示用户名不存在。	输入错误的密码	提示密码错误
3	管理员新建景点	给景点表新增一条记录，并相应的给管理表新增一条记录	输入景点名称，测试是否成功	景点列表中新增景区
4	游客查看景点信息	按照景点类型、所在地区、关键字搜索以及不同排序方式筛选景点	不选择任何筛选条件	成功显示所有景区
			勾选筛选几个条件	成功显示符合条件的景区
			勾选几个已知的无符合条件景点的条件	显示无符合条件的景点
			关键字查询	显示名字中含关键字的景区
			按景点名称/平均分/人气值排序	成功按序展示景点列表
5	游客查看订阅信息	展示所有订阅景点的通知列表	关注某几个景点，打开消息中心	成功展示订阅景点的通知
6	游客订阅景点	建立游客与景点的订阅关系	关注某个景区，查看消息列表中是否有新增通知	成功新增该景区通知
			取消关注某个景区，查看消息列表中是否还有收到该景区的通知	不再收到该景区通知
7	游客发表打分与评论	在评论表中新增一条数据，更新景点的平均分	发表打分与评论	新评论展示在评论区，景区平均分正确更新
8	管理员修改景点信息	修改景点表中对于项的	随意选择要修改的结果	景点卡片列表里显

		属性值		示的详细信息更新
9	管理员发布通知	在通知表中新建一条通知	选择景点，输入通知的内容	订阅游客的消息中心与景点通知区均显示新发表的消息。

功能测试的具体表现可观看演示视频。

5.4 性能测试

5.4.1 景点表查询性能

作为本平台的核心功能，景点表查询的性能表现影响着整个网页的使用体验。在测试过程中，我测试了 5 组操作，每组随机进行 10 次条件勾选筛选操作，得到数据如下：

序号	1	2	3	4	5
用时(s)	15.27s	11.58s	13.89s	15.46s	10.98s

每组筛查的平均用时为 13.44 秒，即每次查询的平均时间为 1.34 秒。影响时长的最主要因素是空集的情况。

5.4.2 景点详情获取性能

景点详情获取指从数据库中获取景点的评论区与通知区。本步骤数据访问量大，若性能不好对网页的流畅性有很大的削弱。在测试过程中，我测试了 5 组操作，每组操作中随机点开了 10 个景点的详情互动页面得到数据如下：

序号	1	2	3	4	5
用时(s)	20.83s	25.76s	23.83s	26.36s	23.58s

每组详情获取的平均用时为 24.07 秒，即每次查询的平均时间为 2.41 秒。由于手工测量还包含返回主页的时间，因此此数据虚高，但也可以看出系统的整体性能不高。

6　维护与完善

6.1 系统运行维护

6.1.1　安全性维护

我们将持续关注本平台的安全性，并确保平台的安全性能得到最大的保障。具体维护措施包括：定期进行系统漏洞扫描和安全测试，及时修复漏洞和强化安全防护措施。对于用户密码和敏感信息进行加密存储和传输，防止数据泄露和信息被篡改。对于平台的重要功能和关键数据进行备份和恢复，以防止数据丢失和系统瘫痪。

6.1.2　功能升级和优化

我们将持续关注用户反馈和市场需求，及时进行功能升级和优化，提升平台的用户体验和性能。具体维护措施包括：收集和分析用户反馈和市场需求，及时优化和升级平台的功能和界面设计。对于平台的性能和稳定性进行监控和分析，及时优化和升级系统架构和技术方案。对于平台的数据库进行优化和清理，提升查询速度和响应时间。

6.1.3　数据更新和维护

我们将持续关注旅游行业的最新动态和数据变化，及时更新和维护平台的相关数据和信息。具体维护措施包括：对于平台的旅游景点、酒店和交通信息进行定期更新和维护，保证数据的准确性和时效性。对于平台的数据源和数据接口进行监控和维护，及时修复和处理数据异常和错误。对于平台的数据备份和恢复进行定期检查和测试，防止数据丢失和系统瘫痪。

6.2 不足点分析

由于本项目是我第一次进行 web 全栈开发，缺乏实际经验，对相关技术运用不够熟练，且时间准备不充分，因此还存在着许多的不足。经过反思后列举如下：

- 表格结构设计未优化

本次系统的所有数据库表结构由我手动推导设计，由于对数据库范式这块知识掌握的不是很熟练，我在设计时可能有存在不少疏漏，导致表结构冗余与缺

漏。有些实体表的设计属性过于简单，不贴合实际。比如，用户表只记录了用户名、密码和电话等信息，并未添加更详细的性别、年龄等更具有实际意义的信息。

- 系统功能单一

由于对后端知识的缺乏，我在设计功能时并未用到许多复杂的后端逻辑，只是对数据库进行简单的 CRUD 操作，导致最终我的项目平台呈现出来的核心功能都是围绕数据库的增删改查展开的。这些单纯的数据操作在实际出行过程中是远远不够使用的，没有办法给用户带来更好的旅行规划等服务。

- 系统性能不佳

在后端实现数据库访问操作时，我都是采用最直接的方式，每当需要进行数据操作时，都会使用 MySQL 语句进行一次数据库的访问，这在实际应用中会让我的系统性能大大降低。因此，最终我测试出来的系统信息获取速度也比预期中的慢，需要经过一到两秒才能跳转到相应页面并获取相关信息，流畅性比预期中低。

- 系统数据不足

由于时间紧迫且对公开数据集了解较少，最终我的数据库初始化是由一系列手动构造的数据来完成的，这批数据只包含约 20 个景点和每个景点一到两条评论与通知。从测试的角度来说，由于数据的体量很小，所以无法测试出真实情况下数据特别多的情况。从应用的角度来说。仅有 20 个景区显然是无法覆盖整个上海的景区的，且手动添加的方式过于繁琐，显然很不实际。

6.3 改进方向

针对在 6.2 中分析的四个不足点，我们可以提出以下改进方案：

首先是对数据表结构进行优化，适当的增加表格的属性，添加更多的配套的功能，让表格的属性不闲置，使系统的服务更加完善。也能相应的缓解功能单一的问题。

若想使系统的服务不局限于数据库的增删改查操作，可以学习更多有关网页开发的知识。比如引用一些其他大型成熟网站的 API。引用百度地图的 API，可以让旅游景点的地区地理位置更加直观的显示在平台里，通过调用他们的服务实现一些路线的规划等高级功能。

有关系统的性能问题可以学习一下网页开发中使用的全局变量等相关知识，将获取到的信息存储全局变量，就不用重复访问数据库，只需获取有改动的部分即可。

数据集的扩充可以查询上海市公开数据平台，研究如何将数据转化为MYSQL语句进行导入。

7 总结

7.1 项目总结

在本次项目中，我实现了一款基于 Web 网页的上海市文旅景点出行平台，完成了对数据库基本的增删改查操作，基本实现了预期的用户管理、景点信息管理、评论与消息订阅四大功能模块。在系统功能、性能和数据库设计上都还有一些进步的空间，可在后续学生生活中不断学习完善。

7.2 心得体会

在项目初期，我明确了平台的目标和用户需求，包括提供旅游景点信息管理、评论与通知等功能。通过对相关网站页面设计与功能设计调研，我初步了解了用户的真实需求和市场的竞争情况，为后续的设计和开发奠定了坚实的基础。

在设计和开发阶段，我调研了当前网页开发采用的主流方法，选择了比较容易上手的 vue 框架和 JS 后端配合。我们使用了现代化的前端技术和后端技术，前后端分离，包括 HTML、CSS、JavaScript 等，以确保平台的性能和可靠性。在开发过程中，我注意对文件夹和文件名的管理，使开发更加流畅。

在测试和优化阶段，我采用黑盒测试的方法，对前后端分别进行全面的测试和优化，保证平台的质量和可靠性。

总结来说，通过本项目，我获得了丰富的实战经验和技术积累，包括：熟练掌握了现代化的前端技术和后端技术，包括 vue 框架和 element UI 的使用，Node.js 后端的使用等。掌握了敏捷开发的方法和项目管理技巧，以及测试和优化的方法和技术。了解了旅游行业的最新动态和市场需求，以及用户的真实需求和使用习惯。在未来，我将继续关注旅游行业的最新动态和市场需求，对平台进行维护与优化。