目 录

装

订

线

摘要：随着疫情逐渐得到控制，公众对于文体活动的需求迅速增加。在这个背景下，本项目旨在通过设计并开发一个文体活动管理系统，以提供更加便捷的活动信息查询、参与和管理服务。传统的文体活动信息分散在不同平台上，给用户带来了不便，而本系统通过整合各类文体活动信息，搭建了一个统一的、智能化的服务平台。系统支持三类主要用户角色：普通游玩者、活动举办者和城市管理者。普通游玩者能够查询各类活动信息、记录参与活动并进行评价，同时可以发布寻求搭子的请求；活动举办者可以发布和管理活动信息，并查看用户反馈；城市管理者则可以对活动进行合规性审核和管理。

本系统采用了 Vue3、Vite 和 Element Plus 作为前端框架，提供了高效的开发体验和丰富的 UI 组件；后端则使用 Flask 和 Flask-SQLAlchemy，结合轻量级的 Flask 框架，实现了简洁而高效的 API 设计。此外，系统基于 SQLite 数据库存储和管理数据，保证了数据的一致性和可查询性。通过这个系统，用户能够便捷地获取活动信息、发布评价和搭子请求，同时活动举办者和城市管理者能够高效管理和监督文体活动。系统的设计和实现不仅提升了文体活动的管理效率，也为智慧城市建设中的文化和娱乐服务提供了有效的支持。

关键词：数据库，文体活动，flask，vue

装 订 线

1 引言

1.1 背景和意义

随着疫情的结束，人们对于文体活动的渴望和需求迅速增长。在经历了长时间的居家隔离和社交限制后，公众对于参与各类文化和体育活动表现出了极高的热情。这种需求的增长促使社会和市场寻求更加便捷、高效的活动参与方式。

当前，各类文体活动的信息分散在不同的平台和应用。例如，博物馆、公园、体育场等场馆的预约信息，演唱会、音乐剧等活动的时间和票务信息，往往需要用户分别访问不同的网站或应用才能获取。这种分散的信息源给用户带来了不便，也影响了活动参与的效率。同时，对城市管理者而言，文体活动通常具有人流量大等特点而是管理的重点；对活动举办者而言，让更多人知道活动信息也尤为重要。

在智慧城市建设的大背景下，城市基础设施和服务的数字化转型成为趋势。文体活动作为城市文化生活的重要组成部分，其信息的整合与智能化管理显得尤为重要。随着大数据技术推动城市计算的发展，可通过用户对文体活动的选择构建的数据库，在社交方面计算“游玩搭子”；可通过对过往人流量的数据，在交通方面，预测未来文体场所的人流量以调控交通；在文娱建设方面，可通过用户对各文体活动的喜爱程度计算未来文体活动举办方向。

综上，本项目旨在收集汇总文体活动相关数据，设计文体活动管理系统为游玩者，文体活动举行者，城市管理者提供便捷。

1.2 项目概述

本项目是一款基于关系数据库模型设计的文体活动管理系统，用户可分为普通游玩者，活动举办者，城市管理者，对于这三类用户该系统具有以下功能：普通游玩者可通过此系统查询各类文体活动信息，包括场所，购票信息，人流量等，并可记录自己所参与的活动并留下评价，发布寻求搭子的请求和查询可以成为搭子的其他用户；活动举办者可发布活动信息并查看游玩者的反馈等；城市管理者可查询各项活动的信息，并对个活动是否合规进行评判。

2 需求分析

2.1 功能分析

对于普通游玩者，活动举办者，城市管理者这三类用户分别有不同功能需求，下面将分别论述。

2.1.1 通用功能

1. 用户管理功能：

- 注册新用户：用户可以通过系统注册新账号，提供用户名、密码、邮箱等信息,并选择自己的身份。

登录：已注册用户可以通过用户名和密码登录系统。

- 用户角色管理：根据不同角色（普通游玩者，活动举办者，城市管理者）有不同的权限。

2. 数据管理和查询功能：

- 用户可查询某文体活动的场所时间人流量等各向信息，也可通过场所，类型等查询满足条件的文体活动。
- 用户可查询自己过往的查询记录。

2.1.2 普通游玩者功能

1. 普通游玩者可查看某文体活动的评分，也可对某文体活动打分或发布评价，系统能根据评分计算平均分等。
2. 普通游玩者可记录自己参与过的活动，并查询自己的参与历史。
3. 普通游玩者可对某一活动发布寻找搭子的信息，或查询某一活动需求搭子的用户的招募信息。

2.1.3 活动举办者功能

1. 活动举办者可发布活动，并有修改维护活动各项信息的权利
2. 活动举办者可查看活动的评分但不可参与打分。

2.1.4 城市管理者功能

1. 城市管理者可修改活动是否合规的相关数据。
2. 可查看活动发布者的相关信息，以检查其是否合规。

2.2 流程分析

2.2.1 登录与注册

用户访问系统，若是新用户则进行注册，若已注册则使用已有账号进行登录。用户注册提供用户名，密码，邮箱，选择身份，系统验证用户提供的注册信息，确保用户名唯一，密码符

合规定，邮箱格式正确，后台将用户信息放入数据库。注册成功后，用户可以使用用户名和密码登录系统。流程图如下：

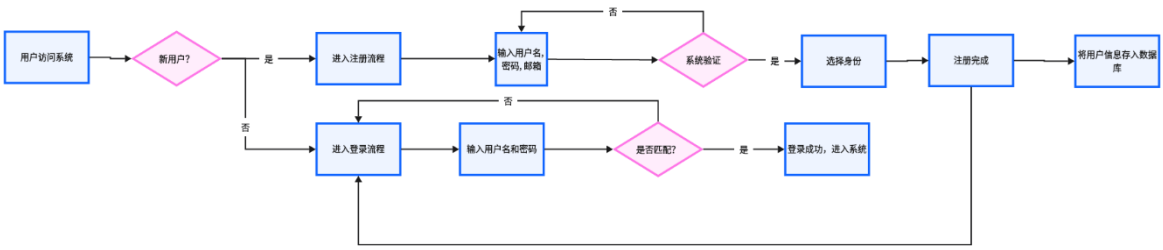


图 2.1 登录与注册流程图

2.2.2 评价与打分系统

普通游完者可发表自己的评价与评分，每记录一条评分，系统更新数据库，自动计算并更新该景点的最新平均分。活动发布者和城市管理者只可查看不可发表评价。

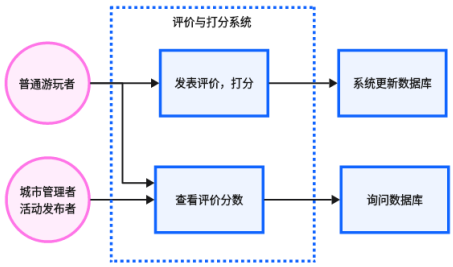


图 2.2 评价与打分流程图

2.2.3 活动信息管理系统

对于活动信息，每个不同角色的用户有不同权限，每个用户都可查看各项信息，但普通游玩者只能对某个活动发布招募搭子的信息，活动发布者可发布或更改自己发布的活动的信息，城市管理者可对发布的活动进行审核并更改错误的的信息。

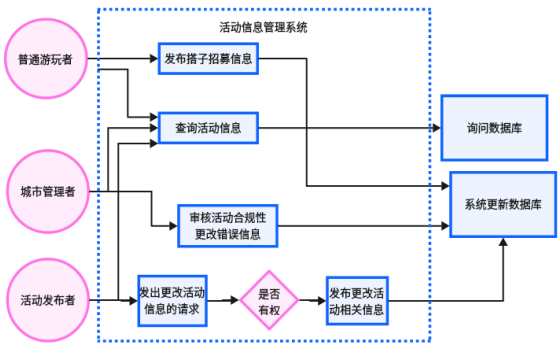


图 2.3 活动信息管理流程图

2.2.4 用户信息管理系统

不同角色的用户所维护的数据不同，除了基本信息（用户名，密码，邮箱等），普通游玩者可查询记录自己参与的活动，想去的活动，评价的活动；城市管理者可查询自己审核的活动，同时可赋予活动发布者可修改活动信息的权限；活动发布者记录查询自己发布的活动和自己有权限更改信息的活动。

总体流程来看，用户进入系统，首先进入登录与注册子系统，根据注册信息匹配角色以及相应权限，然后根据需求选择进入活动信息管理子系统，用户信息管理子系统，或评价与打分系统，如图所示：

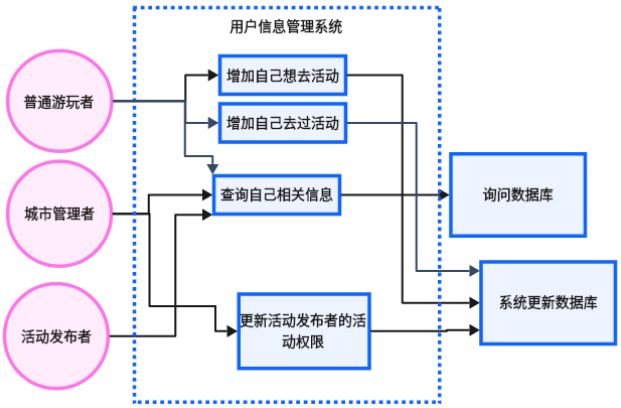
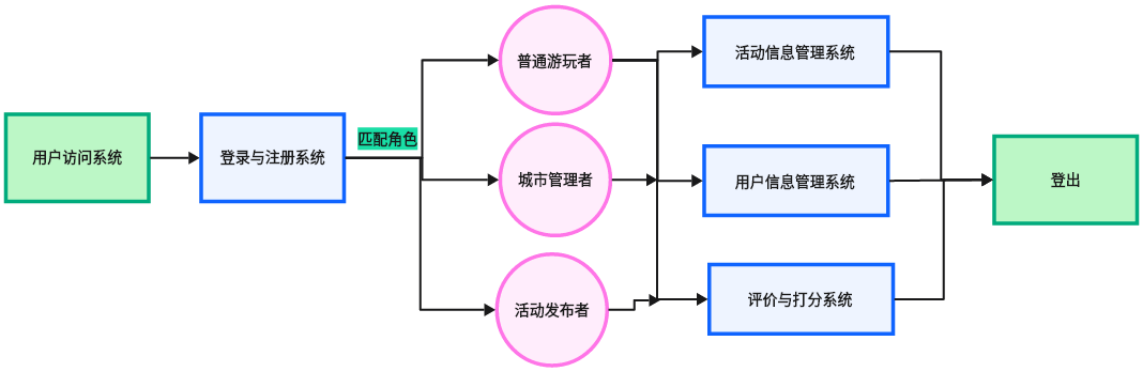


图 2.4 用户信息管理流程图

总体流程如下：



2.3 非功能需求分析

2.3.1 性能分析

（1）实时性需求：

系统应能快速响应用户请求，确保用户体验方便快捷。在面对用户数量较大时也应该拥有较快的反应速度。

（2）安全性需求：

系统应采用加密技术保护用户数据和通信安全，防止非法访问。同时，部署防火墙和其他安全措施，以预防网络攻击，保障系统安全。定期对数据库进行备份，确保数据的可用性和一致性，以应对数据丢失。系统应具备错误自动检测和自我恢复功能，确保稳定运行。

（3）准确性需求：

实施数据验证机制，确保所有上传和更新的活动信息准确无误。对于格式不合规的数据，提供清晰的错误提示，帮助用户纠正输入错误，提高数据的准确性和可靠性。

（4）稳定性需求：

系统应该具备在主流操作系统和设备上运行的能力，以保证广泛的可用性。同时，系统可在后续进行维护和更新，以及时修改新发的问题，也能增加新的功能和应用新的技术已契合城市计算的大背景。

2.2.3 可视化需求

（1）用户界面友好：

界面设计应考虑不同用户群体的需求，提供易于理解和操作并且简洁直观的图形界面，降低新用户的学习成本，提高用户的使用效率。使用图标、颜色等视觉元素，帮助用户快速理解信息，减少操作错误。界面布局应合理，避免信息过载，确保用户能够轻松找到所需功能。并能支持用户根据个人喜好调整界面布局、颜色等，增强个性化体验。在用户操作过程中提供帮助信息，如操作提示、功能说明，降低用户学习成本。

（2）数据可视化

本系统中对于城市管理者的使用需要对人流量，评分等数据进行查询且这些数据较多，因此以图表形式展现会更加直观，易于统计分析。同时本系统中地点作为重要数据，若能与地图结合展现，对于使用者而言会更加直观和方便。

装 订 线

3 概念设计

3.1 实体属性设计：

3.1.1 用户相关实体设计

（1）普通游玩者实体：

以普通游玩者身份注册系统后，普通游玩者实体的属性有 ID，密码，昵称，邮箱和喜爱的活动类型，其中 visitorID 作为主键，喜爱的活动类型在文化类，体育类，娱乐类中选择。

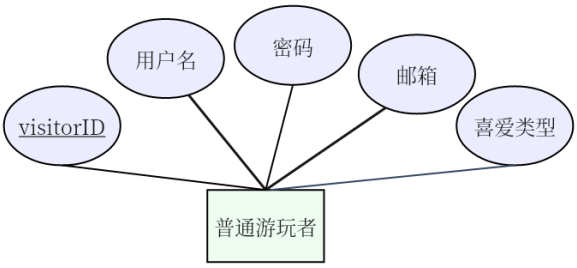


图 3.1 普通游玩者实体图

（2）活动发布者实体

以活动发布者身份注册系统后，活动发布者实体的属性有唯一 ID，密码，昵称，邮箱和是否合规的状态，其中 organizerID 作为主键，是否合规是一个 bool 变量，可由城市管理者更改。

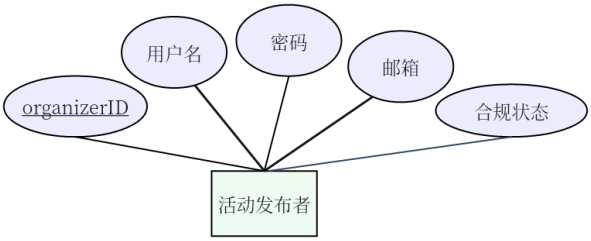


图 3.2 活动发布者实体图

（3）城市管理者

以城市管理者身份注册系统后，活动发布者实体的属性有唯一 ID，密码，名字，邮箱，其中 managerID 作为主键。

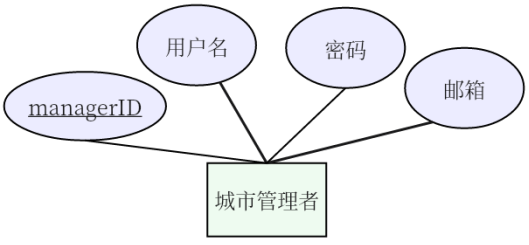


图 3.3 城市管理者实体图

3.1.2 活动相关实体设计

（1）活动实体设计

活动实体得属性有作为主键的编号，其他基本信息包括活动名字，开始时间，结束时间，活动类型，活动详细信息（text 的形式存储），是否结束状态，人流量，报名相关信息。报名相关信息作为复合属性包括报名渠道，报名开始时间，报名结束时间，是否报名结束状态。其中着两个 bool 表示的状态量依赖于结束时间和报名结束时间，是其派生属性，用虚线圈表示。

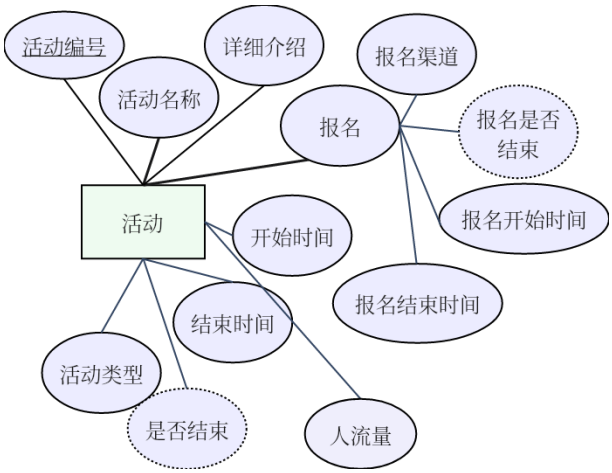


图 3.4 活动实体图

（2）场所实体设计

场所实体的属性中主键为场所编号，其他属性有场所名称，详细介绍（text 形式）和位置，位置作为复合属性包括城市，县区，街道，街道编号，备注，经度，纬度。如地址上海市嘉定区曹安公路 4800 号，城市为上海市，县区为嘉定区，街道为曹安公路，街道编号为 4800 号，备注可为空。

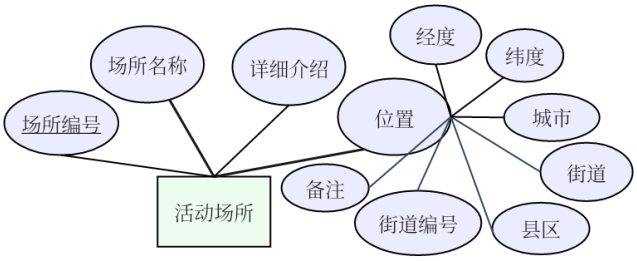


图 3.5 活动场所实体图

3.2 实体关系设计

（1）普通游玩者评论活动：

关系类别：多对多，即一个普通游玩者可以对多个活动发表评论，一个活动可以被多个普通游玩者评论。

外键：visitorID 和活动编号

属性说明：一个普通游玩者只能对一个活动发表一个评论，因此 visitorID 和活动编号合在一起具有唯一性可作为主键，另外，评论的其他属性有评论内容，评分，评价发布时间。

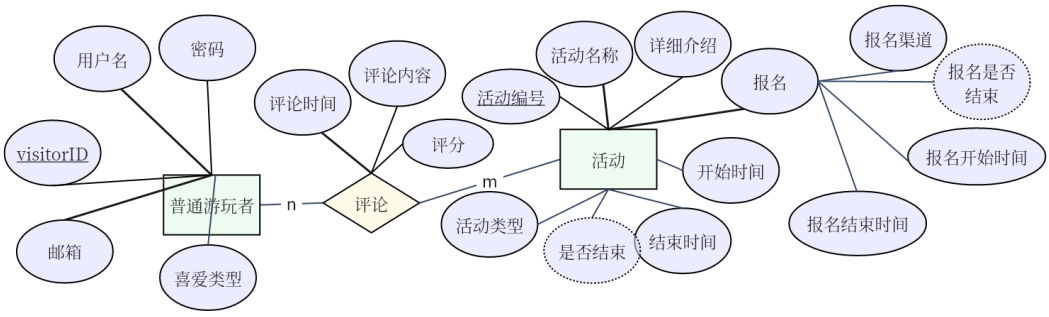


图 3.6 评论关系 ER 图

（2）普通游玩者对某个活动发布寻找搭子的请求

关系类别：多对多，即一个普通游玩者可以对多个活动发布寻找搭子的请求，一个活动可以被多个普通游玩者发布寻找搭子的请求。

外键：visitorID 和活动编号

属性说明：一个普通游玩者对一个活动只能发布一个寻找搭子的请求，因此 visitorID 和活动编号合在一起具有唯一性，可作为主键，另外，其他属性有搭子要求，发布时间，以及 bool 型数据表示是否找到即代表是否还需继续寻找。

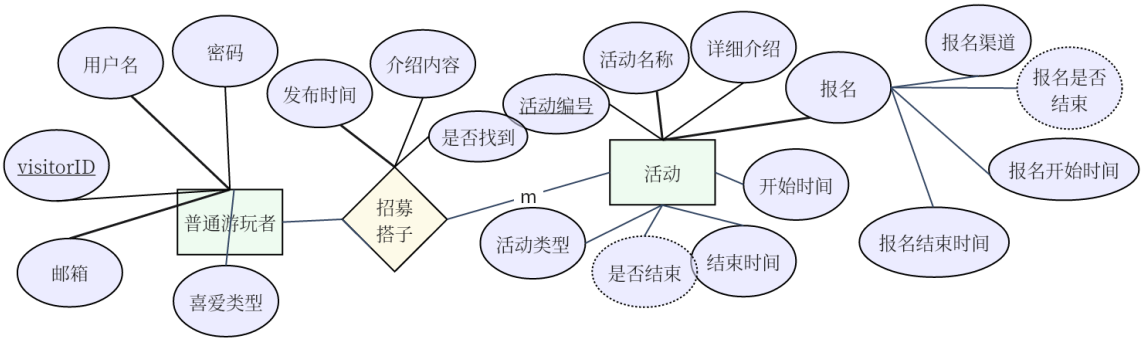


图 3.7 招募搭子关系 ER 图

（3）活动发布者负责的活动

关系类别：一对多，即一个活动发布者管理多个活动，一个活动只有一个活动发布者管理。

外键：organiserID 和活动编号

属性说明：由于一对多的关系，因此活动编号对这个关系集具有唯一性，可作为主键。最新操作指活动发布者对某活动的最新更改，有如下几项增添活动，更改介绍，更改报名信息，更改时间，并记录最新操作时间。

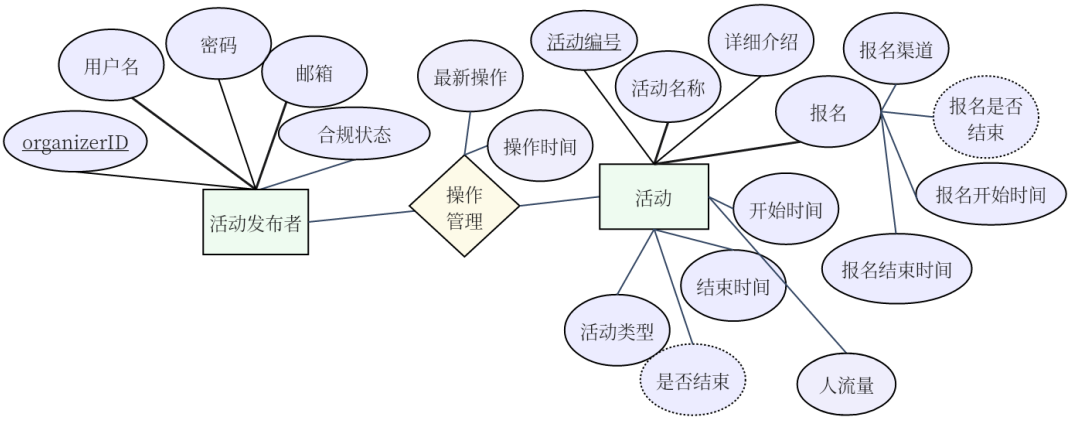


图 3.8 管理关系 ER 图

（4）城市管理者管理活动发布者

关系类别：一对多，即一个城市管理者管理多个活动发布者，一个活动发布只有一个城市管理者管理。

外键：organiserID 和 managerID

属性说明：由于一对多的关系，因此活动发布者对这个关系集具有唯一性，可作为主键。

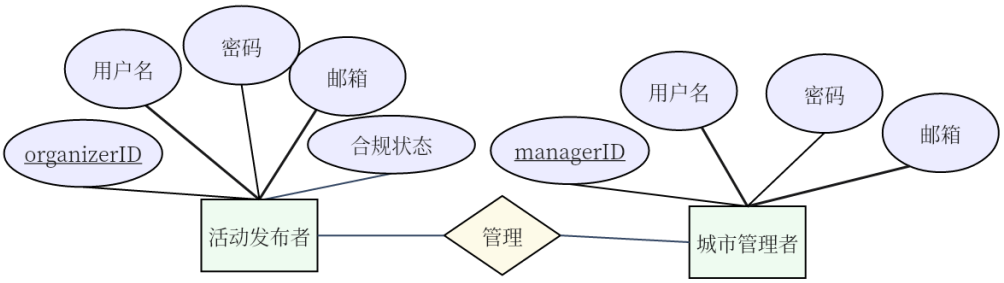


图 3.9 城市管理者管理活动发布者关系 ER 图

（5）活动与场所关系

关系类别：一对多，即一活动位于一个场所，一个场所有多个活动发生。

外键：活动编号和场所编号

属性说明：由于一对多的关系，因此活动编号对这个关系集具有唯一性，可作为主键。

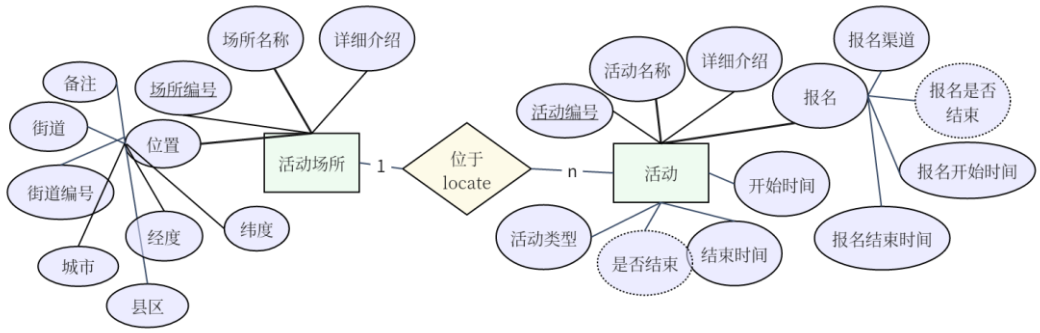


图 3.10 活动与场所关系 ER 图

3.3 全局 ER 图

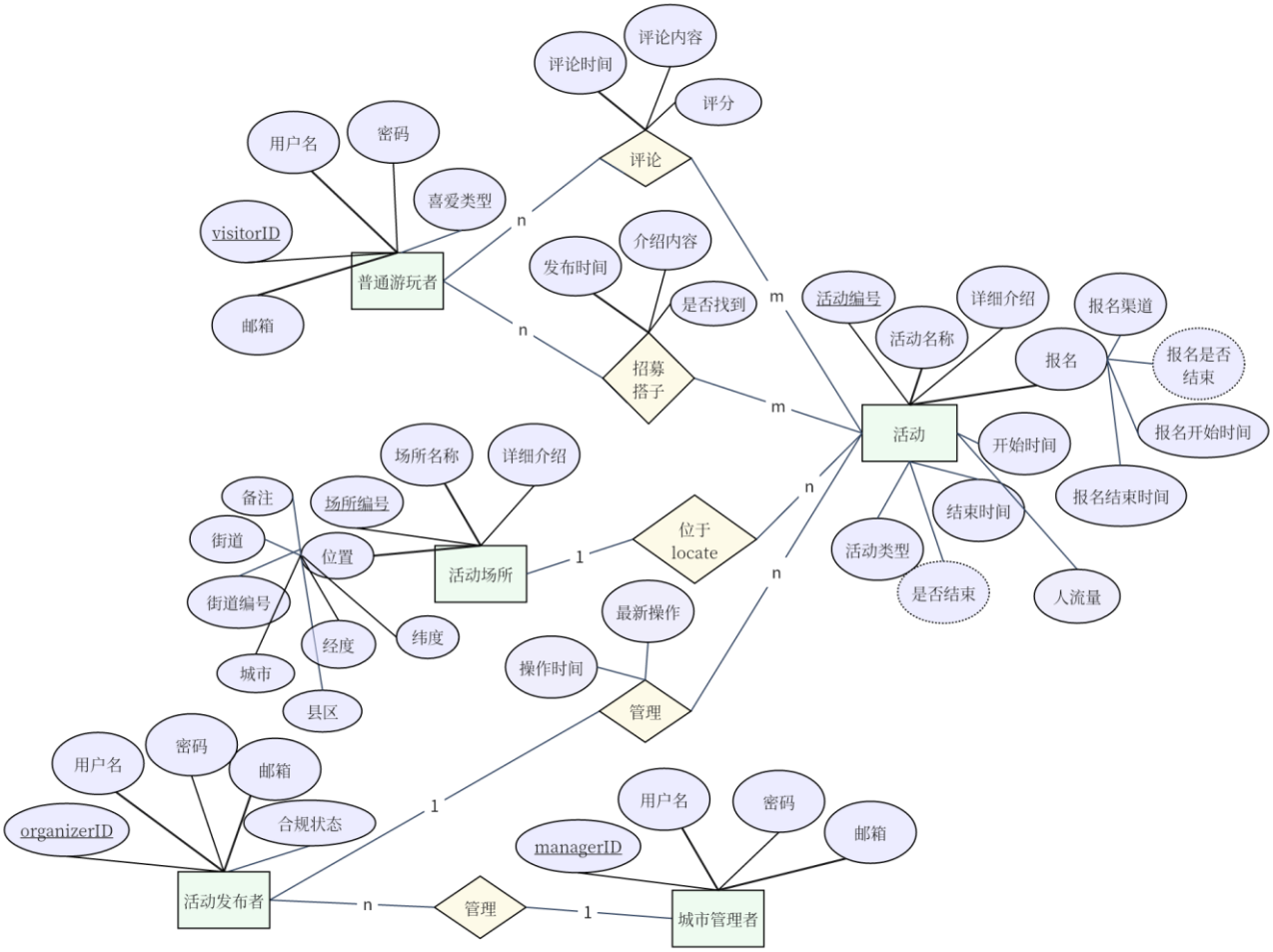


图 3.11 全局 ER 图

4 逻辑设计

4.1 关系模式和表设计

在这个部分根据 ER 图将其转换为关系模式和表，同时分析是否符合 1NF，2NF，3NF。1NF 即第一范式要求每一列都只包含单一值，即数据表中的每个字段都应该是原子的，不允许包含重复的列，也不允许包含多值属性。第二范式在符合第一范式的基础上，要求消除部分依赖，即非主属性必须完全依赖于主键。第三范式在符合第二范式的基础上，要求消除传递依赖，即非主属性不能依赖于其他非主属性。

4.1.1 普通游玩者表

关系模式：visitor(Name, Password, Email, visitorID, FavoriteActivityType)
每一个属性都是单一值，符合第一范式，visitorID 作为唯一主键，所有非主属性都完全依赖于 visitorID，所以符合第二范式，且不依赖于其他非主属性，所以符合第一范式。

表 4.1 普通游玩者表

属性名	说明	数据类型	约束	主键	外键
visitorID	普通游玩者唯一标识	INT	not null, auto_increment	YES	-
Name	普通游玩者昵称	VARCHAR(25)	not null	-	-
Password	密码	VARCHAR(100)	加密存储, not null	-	-
Email	用户邮箱	VARCHAR(100)	-	-	-
FavoriteActivityType	喜爱的活动类型	ENUM('文化类', '体育类', '娱乐类')	not null	-	-

4.1.2 活动发布者表

关系模式：organizer(Name, Password, Email, organizerID, ComplianceStatus)
每一个属性都是单一值，符合第一范式，organizerID 作为唯一主键，所有非主属性都完全依 organizerID，所以符合第二范式，且不依赖于其他非主属性，所以符合第三范式。

表 4.2 活动发布者表

属性名	说明	数据类型	约束	主键	外键
organizerID	活动发布者唯一标识	INT	not null, auto_increment	YES	-
Name	活动发布者姓名	VARCHAR(25)	not null	-	-
Password	密码	VARCHAR(100)	加密存储, not null	-	-
Email	用户邮箱	VARCHAR(100)	-	-	-
ComplianceStatus	是否合规	BOOLEAN	not null, default 0	-	-

4.1.3 城市管理者表

关系模式：manager(Name, Password, Email, managerID)

每一个属性都是单一值，符合第一范式，managerID 作为唯一主键，所有非主属性都完全依 managerID，所以符合第二范式，且不依赖于其他非主属性，所以符合第三范式。

表 4.3 城市管理者表

属性名	说明	数据类型	约束	主键	外键
managerID	城市管理者唯一标识	INT	not null, auto_increment	YES	-
Name	城市管理者姓名	VARCHAR(25)	not null	-	-
Password	密码	VARCHAR(100)	加密存储, not null	-	-
Email	用户邮箱	VARCHAR(100)	-	-	-

4.1.4 活动表

关系模式：activity(ActivityID, Name, StartTime, EndTime, Type, Description,Flow, RegistrationChannel, RegistrationStartTime, RegistrationEndTime)

每一个属性都是单一值，符合第一范式，ActivityID 作为唯一主键，所有非主属性都完全依 ActivityID，所以符合第二范式，在 ER 图中还有表示是否结束（IsFinished）和是否报名结束（IsRegistrationClosed）的派生属性，为满足 3NF 设计，在这里去掉，但可设置视图，以方便查询

表 4.4 活动表

属性名	说明	数据类型	约束	主键	外键
ActivityID	活动唯一标识	INT	not null, auto_increment	YES	-
Name	活动名称	VARCHAR(100)	not null	-	-
StartTime	开始时间	DATETIME	not null	-	-
EndTime	结束时间	DATETIME	not null	-	-
Type	活动类型	ENUM('文化类', '体育类', '娱乐类')	not null	-	-
Description	活动详细信息	TEXT	-	-	-
Flow	人流量	INT	-	-	-
RegistrationChannel	报名渠道	VARCHAR(100)	-	-	-
RegistrationStartTime	报名开始时间	TIMESTAMP	-	-	-
RegistrationEndTime	报名结束时间	TIMESTAMP	-	-	-

4.1.5 场所表

关系模式：venue(VenueID, Name, Description, City, County, Street, StreetNumber, Note, Longitude, Latitude)

每一个属性都是单一值，符合第一范式，VenueID 作为唯一主键，，所以符合第二范式，且所有非主属性都完全依赖 VenueID，所以符合第三范式。

表 4.5 场所表

属性名	说明	数据类型	约束	主键	外键

VenueID	场所唯一标识	INT	not null, auto_increment	YES	-
Name	场所名称	VARCHAR(100)	not null	-	-
Description	场所详细介绍	TEXT	-	-	-
City	城市	VARCHAR(100)	not null	-	-
County	县区	VARCHAR(100)	not null	-	-
Street	街道	VARCHAR(100)	not null	-	-
StreetNumber	街道编号	VARCHAR(50)	not null	-	-
Note	备注	VARCHAR(255)	-	-	-
Longitude	经度	DECIMAL(10, 7)	-	-	-
Latitude	纬度	DECIMAL(10, 7)	-	-	-

4.1.6 评价表

关系模式：comment (visitorID, ActivityID, CommentContent, Rating, CommentTime)
每一个属性都是单一值，符合第一范式，visitorID 和 ActivityID 作为联合主键（一个游玩者只能评论一个活动一次，因此 visitorID 和 ActivityID 的组合是唯一的），没有非主属性只依赖 visitorID 或 ActivityID，所以符合第二范式，且不依赖于其他非主属性，所以符合第三范式。

表 4.6 评价表

属性名	说明	数据类型	约束	主键	外键
visitorID	普通游玩者唯一标识	INT	not null	YES	references Visitor(visitorID)
ActivityID	活动唯一标识	INT	not null	YES	references Activity(ActivityID)
CommentContent	评论内容	TEXT	-	-	-
Rating	评分	INT	-	-	-
CommentTime	评价发布时间	TIMESTAMP	not null	-	-

4.1.7 搭子请求表

关系模式：visitor_request_buddy(visitorID, ActivityID, BuddyRequirement, RequestTime, IsFound)
每一个属性都是单一值，符合第一范式，visitorID 和 ActivityID 作为联合主键（一个游玩者只能在一个活动下发布一个请求，因此 visitorID 和 ActivityID 的组合是唯一的），没有非主属性只依赖 visitorID 或 ActivityID，所以符合第二范式，且不依赖于其他非主属性，所以符合第三范式。

表 4.7 搭子请求表

属性名	说明	数据类型	约束	主键	外键
visitorID	普通游玩者唯一标识	INT	not null	YES	references Visitor(visitorID)
ActivityID	活动唯一标识	INT	not null	YES	references Activity(ActivityID)
BuddyRequirement	搭子要求	TEXT	-	-	-
RequestTime	发布时间	DATETIME	not null	-	-
IsFound	是否找到	BOOLEAN	-	-	-

4.1.8 活动发布者操作管理表

关系模式：organizer_manages_activity(organizerID, ActivityID, LatestOperation, LatestOperationTime)

每一个属性都是单一值，符合第一范式，ActivityID 作为唯一主键（一个活动发布者管理一个活动只记录其最新操作，一个活动只会被一个活动发布者操作，因此 ActivityID 是唯一的）所以符合第二范式，且其他非主属性不依赖于其他非主属性，所以符合第三范式。

表 4.8 活动发布者操作管理表

属性名	说明	数据类型	约束	主键	外键
organizerID	活动发布者唯一标识	INT	not null	-	references Organizer(organizerID)
ActivityID	活动唯一标识	INT	not null	YES	references Activity(ActivityID)
LatestOperation	最新操作	ENUM(增添活动，更改介绍，更改报名信息，更改时间)	-	-	-
LatestOperationTime	最新操作时间	TIMESTAMP	-	-	-

4.1.9 城市管理者管理活动发布者关系表

关系模式：manager_manages_organizer(managerID, organizerID)

每一个属性都是单一值，符合第一范式，organizerID 作为唯一主键（一个活动发布者只会被一个城市管理者管理，因此 organizerID 是唯一的）所以符合第二范式，且其他非主属性不依赖于其他非主属性，所以符合第三范式。

表 4.9 城市管理者管理活动发布者关系表

属性名	说明	数据类型	约束	主键	外键
organizerID	活动发布者唯一标识	INT	not null	YES	references Organizer(organizerID)
managerID	城市管理者唯一标识	INT	not null	-	references Activity(ActivityID)

4.1.10 活动位于场所表

关系模式：activity_in_venue(ActivityID, VenueID)

每一个属性都是单一值，符合第一范式，ActivityID,作为唯一主键（一个活只会在一个地方，因此 ActivityID 是唯一的）所以符合第二范式，且其他非主属性不依赖于其他非主属性，所以符合第三范式。

表 4.10 活动位于场所表

属性名	说明	数据类型	约束	主键	外键
ActivityID	活动唯一标识	INT	not null	YES	references Activity(ActivityID)
VenueID	场所唯一标识	INT	not null	-	references Venue(VenueID)

装

订

线

5 物理设计

5.1 物理设计概述

物理设计是对逻辑模型进行存储结构设计的过程。物理设计中除了要确定数据的存储方式（这在逻辑设计中已经完成），还要建立起数据库的索引结构。创建索引意味着数据库会创建 B-tree 以方便查询，灵活的索引结构能显著提高数据库查询的效率，但索引并不是越多越好，过多的索引会影响写操作的性能。因此是物理设计中重要的一环。该部分将重点介绍物理设计中数据库索引的建立。

5.2 索引设计方法概述

5.2.1 整理查询条件

设计索引的主要目的是为了加快查询速度，因此，设计索引的第一步是整理需要用到的查询条件，也就是我们会在 WHERE 子句、JOIN 连接条件中使用的字段。整理程序中除 INSERT 语句之外的所有 SQL 语句，按不同的表分别整理出每张表上的查询条件。也可以根据对业务的理解，添加一些暂时还没有使用到的查询条件。

5.2.2 分析字段的可选择性

字段的可选择性指的是字段值的区分度。通常对于复合索引，优先选择可选择性高的字段放在前面，可选择性低的字段放在后面，如果可选择性非常低，通常不会将这样的字段放到索引里。根据字段的可选择性对索引字段进行确定是十分重要的，好的索引字段起到事半功倍的效果。

5.3 本数据库索引设计

5.3.1 分析

（1）Visitor 表

visitorID：设为主键索引。主键索引是必需的，用于唯一标识每个用户，提高了基于主键的查询速度。

Email：设为唯一索引。邮箱在用户注册时是唯一的，通过为该字段建立索引，可以快速定位用户，避免重复注册。在用户登录时会使用此查询。

（2）Organizer 表

organizerID：设为主键索引。主键索引确保每个活动发布者的唯一性，加快基于主键的查询。

Email：设为唯一索引。保证每个活动发布者的邮箱唯一性，提高查询效率，防止重复注册。在用户登录时会使用此查询。

（3）Manager 表

managerID：设为主键索引。主键索引确保每个城市管理者的唯一性，便于快速查询。

Email：设为唯一索引。保证城市管理者邮箱唯一性，提高查找效率，防止重复注册。在用户登录时会使用此查询。

（4）Activity 表

ActivityID：设为主键索引。用于唯一标识每个活动，加快活动的检索速度。

Type：设为普通索引。活动类型通常用于分类查询，通过索引可以加快按类型查找活动的速度。

StartTime 和 EndTime：设为组合索引。活动的开始时间和结束时间经常用于查询活动的时间段，通过组合索引可以提高时间范围查询的效率。

（5）Venue 表

VenueID：设为主键索引。确保场所的唯一性，加快场所信息的检索。

City：设为普通索引。场所的城市信息用于按地区分类查询，通过索引加快按城市查找场所的速度。

（6）Comments 表

visitorID 和 ActivityID：设为复合主键索引。复合主键索引保证同一个用户对同一个活动只能评论一次，并提高联合查询的效率。同时，这些索引用于加快基于单个字段的查询，如根据用户 ID 查找评论，或根据活动 ID 查找评论。

（7）VisitorRequestBuddy 表

visitorID 和 ActivityID：设为复合主键索引。复合主键索引确保同一个用户对同一个活动只能发布一次寻找搭子的请求，且查询效率高。同时，这些索引用于加快单字段查询，如根据用户 ID 查找请求，或根据活动 ID 查找请求。

（8）OrganizerManagesActivity 表

organizerID：设为外键索引。用于关联活动发布者的信息，加快基于活动发布者的查询。

ActivityID：设为主键索引和唯一索引。确保活动 ID 的唯一性，且每个活动只能由一个发布者管理，同时加快基于活动 ID 的查询。

（9）ManagerManagesOrganizer 表

managerID：设为外键索引。用于关联城市管理者的信息，优化基于城市管理者的查询。

organizerID：设为主键索引和唯一索引。确保活动发布者在管理者下的唯一性，加快查询速度。

（10）ActivityInVenue 表

ActivityID：设为主键索引和唯一索引。确保活动 ID 的唯一性，加快基于活动 ID 的查询。

VenueID：设为外键索引。用于关联场所的信息，优化基于场所的查询。

5.3.2 索引表

表名	属性	索引类型	说明
Visitor	visitorID	主键索引	用户唯一标识
	Email	唯一索引	用户邮箱
Organizer	organizerID	主键索引	活动发布者唯一标识
	Email	唯一索引	活动发布者邮箱
Manager	managerID	主键索引	城市管理者唯一标识
	Email	唯一索引	城市管理者邮箱
Activity	ActivityID	主键索引	活动唯一标识
	Type	普通索引	活动类型
	StartTime，EndTime	组合索引	活动开始结束时间

Venue	VenueID	主键索引	场所唯一标识
	City	普通索引	城市
VisitorCommentsActivity	visitorID，ActivityID	复合主键索引	普通游玩者唯一标识，活动唯一标识
VisitorRequestBuddy	visitorID，ActivityID	复合主键索引	普通游玩者唯一标识，活动唯一标识
OrganizerManagesActivity	organizerID	外键索引	活动发布者唯一标识
	ActivityID	主键索引	活动唯一标识
ManagerManagesOrganizer	managerID	外键索引	城市管理者唯一标识
	organizerID	主键索引	活动发布者唯一标识
ActivityInVenue	ActivityID	主键索引	活动唯一标识
	VenueID	外键索引	场所唯一标识

装 订 线

6.项目管理

6.1 框架选择

6.1.1 前端框架

vue3+vite+element-plus+ vue-amap

在本项目中，选择了 Vue 3 作为前端框架，因其提供了强大的响应式数据绑，使得代码更具可维护性和逻辑复用性，同时高效的虚拟 DOM 机制也提高了页面的渲染速度。Vite 作为构建工具，能够提供极快的开发启动速度和高效的热更新功能，通过使用原生 ES 模块，使得开发体验更加流畅。选择 Element Plus 为 UI 组件库，因为它提供了丰富的组件、简洁的设计风格，以便快速构建现代化的用户界面。为了集成地图功能，使用了 Vue AMap，它能够简化高德地图的集成过程，同时充分利用 Vue 的响应式特性，便于快速实现地图相关的交互和展示。

6.1.2 后端框架

Flask 是一个轻量级的 Python Web 框架，选择它是因为其简洁性和灵活性。Flask 不强制任何约定，开发者可以根据项目需求选择适合的扩展库，这有助于灵活地控制项目的结构和功能实现。Flask 提供了丰富的插件和扩展，支持快速集成数据库、认证等常用功能，同时它的学习曲线较低，文档丰富，易于上手，特别适合用于小型到中型的 Web 应用开发。Flask 与前端框架 Vue 以及数据库 SQLite 的兼容性良好，能够快速高效地响应请求，处理业务逻辑，满足本项目的需求。Flask-SQLAlchemy 是 Flask 的一个扩展，它提供了对 SQLAlchemy 的支持，是一个强大的 SQL 工具包和对象关系映射（ORM）系统。使用 FlaskSQLAlchemy，可以以面向对象的方式操作数据库。

6.1.3 数据库

SQLite 是一个轻量级的关系型数据库无需安装额外的数据库服务器，且数据存储在一个文件中，易于管理和部署。SQLite 提供了高效的性能，尤其在处理小型和中型应用的数据存储时表现良好，而且无需复杂的配置，开发者可以快速开始使用。由于其零配置的特性，SQLite 特别适合用于开发阶段以及轻量级的 Web 应用，在与 Flask 框架结合时，能够简化数据库操作，满足本项目对数据库存储的需求。

6.2 开发平台

系统环境：Windows11 64 位

后端：pycharm

前端：vscode

包管理：npm

7. 系统实现

7.1 系统架构

```
├─back # 后端代码目录
│ │ app.py # 接口的实现
│ │ model.py # 数据库模型的定义
│ │ add_data.py # 处理场所数据
│ │ Douban_events.py # 爬虫获取活动信息
│ │ test.http# 测试接口的文件
│ └─instance # 存放实例数据的目录，如数据库文件
│      └─activities.sqlite# 数据库文件
└─front # 前端代码目录
 │ .gitignore # Git 忽略文件列表
 │ index.html # 前端的入口文件,设置系统名称等
 │ package-lock.json # npm 包管理工具自动生成的锁文件
 │ vite.config.js # Vite 的配置文件
 ├─node_modules # 存放 npm 或 pnpm 安装的依赖包的目录
 ├─public # 存放静态资源文件，
 │ logo.png # 应用的 logo 图片文件
 │
 └─src # 源代码目录，存放主要的前端代码
 │ App.vue # Vue 应用的根组件文件
 │ main.js # 应用的入口文件，用于启动 Vue 应用
 │
 ├─assets # 存放静态资源文件，如图片、SVG 等
 ├─components # 存放 Vue 组件文件
 │  ActivityOverview.vue # 活动地图组件
 │   ActivityList.vue # 活动列表组件
 │   SelfPage.vue # 个人页面
 │   navbar.vue # 导航栏组件
 ├─api # 用于存放与 Axios 相关的配置
 │ config.js # Axios 的配置文件
 ├─router # 存放 Vue Router 的配置文件
 │ index.js # Vue Router 的入口文件，配置路由
 └─views # 存放 Vue 组件文件，代表不同的视图页面
 │ DashBoard.vue # 导航栏和其他组件的布局
 │ Login.vue # 登录页面
```

7.2 系统设计

7.2.1 页面设计

- 登录注册页面
  选择角色登录进入系统
  若没有账号则注册新账号
- 活动地图页面

显示标注文旅场所的地图
点击文旅场所，获取此场所的相关信息

- 活动列表页面
  显示所有活动并可查询某个活动
  显示选择的活动详细信息，评论，搭子请求信息
  可发布评论
  可发布搭子请求
- 个人主页
  游玩者可以查看自己的搭子请求，可以修改是否找到搭子状态
  活动发布者此界面可点击发布活动，在弹窗中填写相关信息
  城市管理者显示活动发布者相关信息，对其是否合法状态进行更改

7.2.2 接口设计

根据页面设计完成接口的设计，以实现前后端的连接。以下表为接口设计：

功能模块	接口	请求方法	参数	描述
用户注册	/visitor/register	POST	name, password, email, favorite_activity_type	游客注册接口，接收游客信息并完成注册。
	/organizer/register	POST	name, password, email	组织者注册接口，接收组织者信息并完成注册。
	/manager/register	POST	name, password, email	管理员注册接口，接收管理员信息并完成注册。
用户登录	/visitor/login	POST	name, password	游客登录接口，验证游客身份并返回登录结果。
	/organizer/login	POST	name, password	组织者登录接口，验证组织者身份并返回登录结果。
	/manager/login	POST	name, password	管理员登录接口，验证管理员身份并返回登录结果。
场所管理	/venues	GET	无	获取所有场所的基本信息。
	/venuesDetail	POST	venue_id	获取指定场所的详细信息。
活动管理	/activities	GET	无	获取所有活动的基本信息。
	/activitiesDetail	POST	activity_id	获取指定活动的详细信息、评论和搭子信息。
评论管理	/addComments	POST	activity_id, user_id, content, time	用户发布活动评论。
搭子管理	/addPartners	POST	activity_id, user_id, buddy_requirements, time	发布搭子请求，寻找活动伙伴。
	/visitor_buddy	POST	visitor_id	获取游客发布的搭子信息。
	/changeBuddyStatus	POST	buddy_id, status	更改搭子的状态（例如确认、取消等）。
	/deleteBuddy	POST	buddy_id, user_id	删除搭子请求或已接受的搭子。
组织者管理	/organizers	GET	无	获取所有组织者的基本信息。
	/changeOrganizerStatus	POST	organizer_id, status	更改组织者的状态（启用/禁用）。
活动管理（组织者）	/organizer_activities	POST	organizer_id	获取组织者的所有活动信息。
	/deleteActivities	POST	activity_id	组织者删除指定活动。
	/addActivities	POST	activity_info	组织者增加新的活动。
	/editActivities	POST	activity_id, new_activity_info	组织者编辑现有活动的信息。

装 订 线

7.2.3 数据获取：

地点数据获取：https://www.poilist.cn/。在这个网址中可以获得文旅场所的信息，其中经纬度信息可以使其被标注在地图上，同时，此网址的信息较为杂乱，需要用 python 对数据进行清理后与所设计的数据库相匹配。

活动数据获取：https://shanghai.douban.com/events/week-all。在豆瓣网中可以获取一周活动，为获得相关信息写了爬虫来获取数据，可以获取活动的名字，时间，渠道，费用（写入数据库的描述中），但对与场所的获取没有成功。

数据库初始化：

为方便测试，设置一个数据库重置命令，重置时会读取已清洗好的地点数据，但还需手动运行脚本来获取活动数据，如下：

```
(venv) PS E:\桌面\数据库\mycode\back> flask reset
(venv) PS E:\桌面\数据库\mycode\back>
```

7.2.4 UI 设计

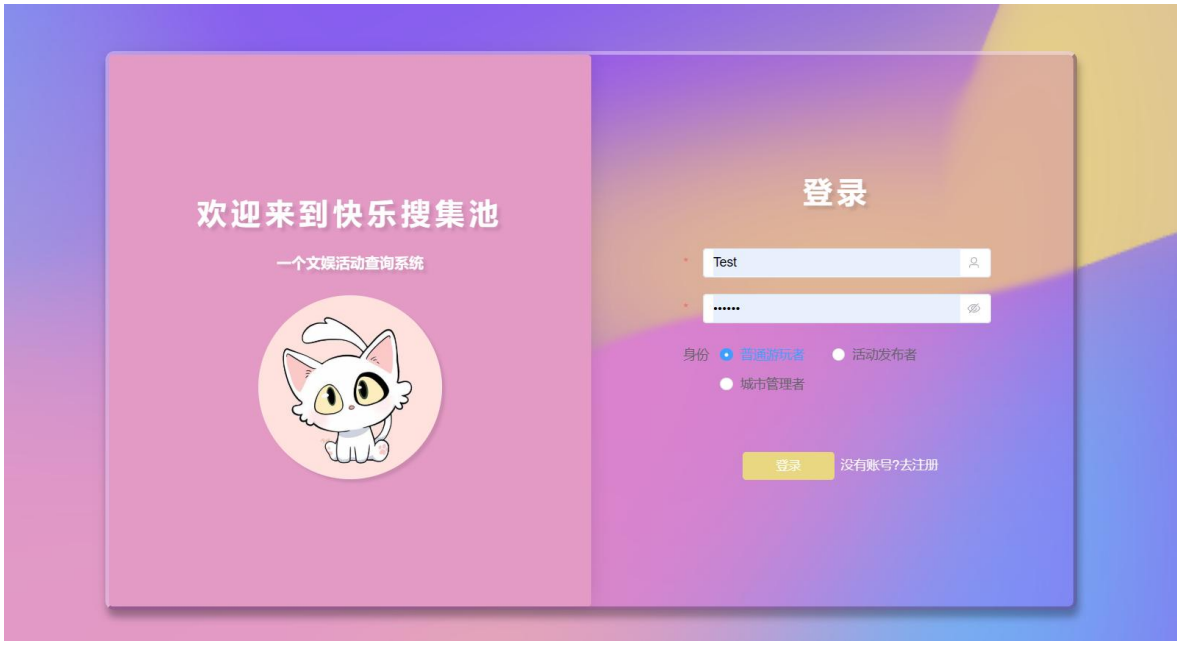
为契合给系统所取名字——快乐搜集池，选取了波浪动画作为背景，主要依靠 css 实现，实现 css 代码如下：

```css
canvas#canvas {
  z-index: -1;
  position: absolute;
  width: 100%;
  height: 100%;
  transform: rotate(0deg) scale(2) translateY(0%);
  --gradient-color-1: #E39AC5;
  --gradient-color-2: #6ec3f4;
  --gradient-color-3: #875EF0;
  --gradient-color-4: #EAD781;
  --gradient-speed: 0.000006;
}
```

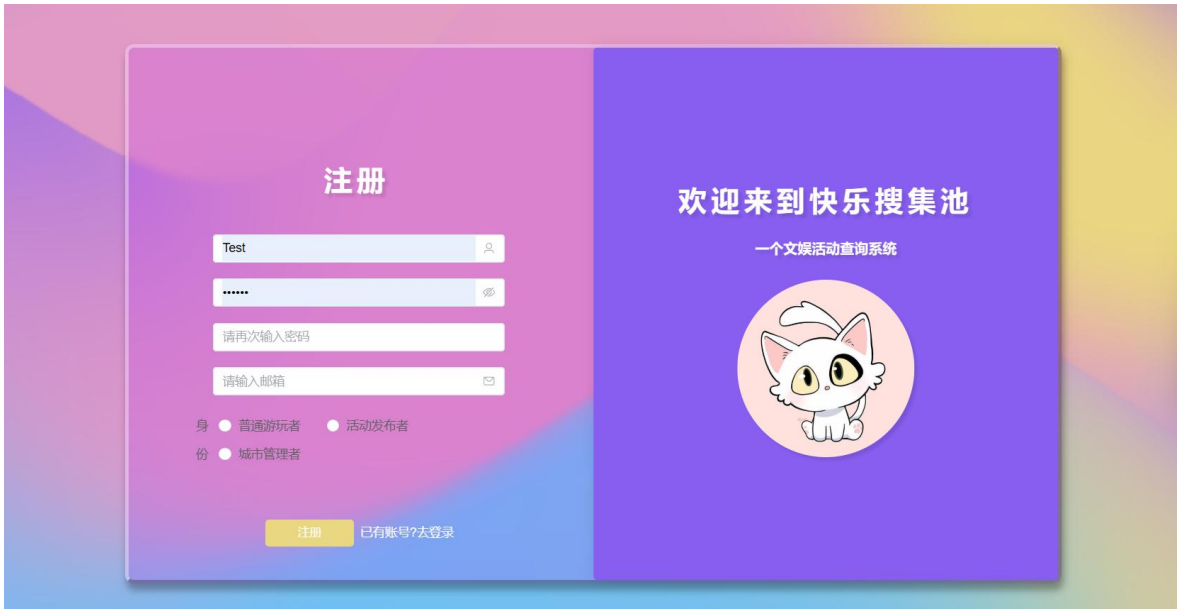
7.3 功能细节展示

7.3.1 登录注册

登录：输入用户名和密码，并选择身份

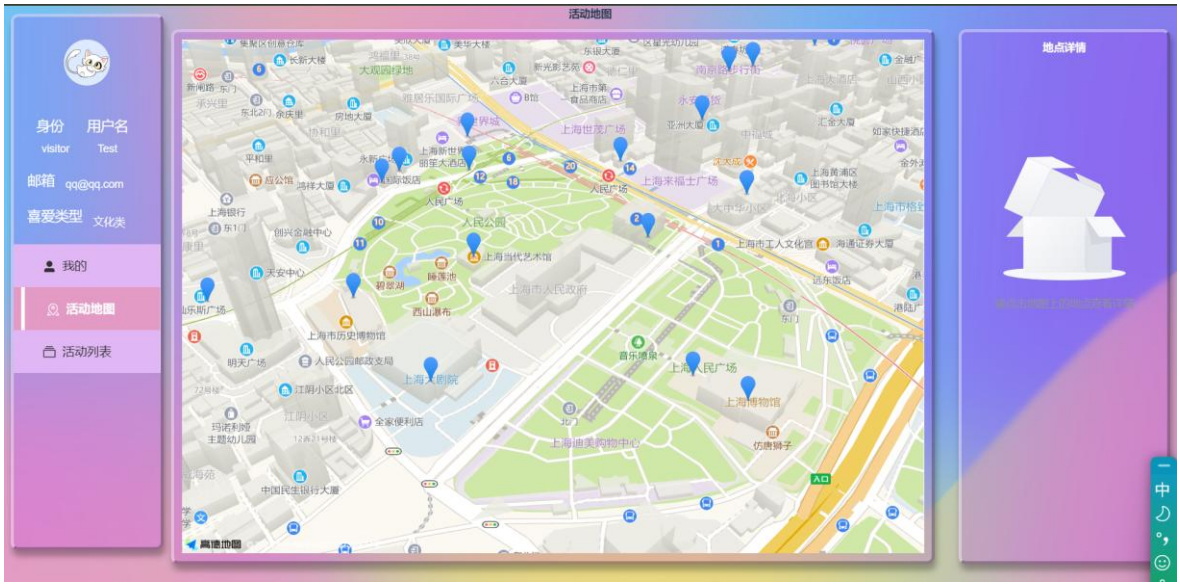


若没有注册账户，则注册账户，输入名字，密码，确认密码，邮箱，并选择身份，

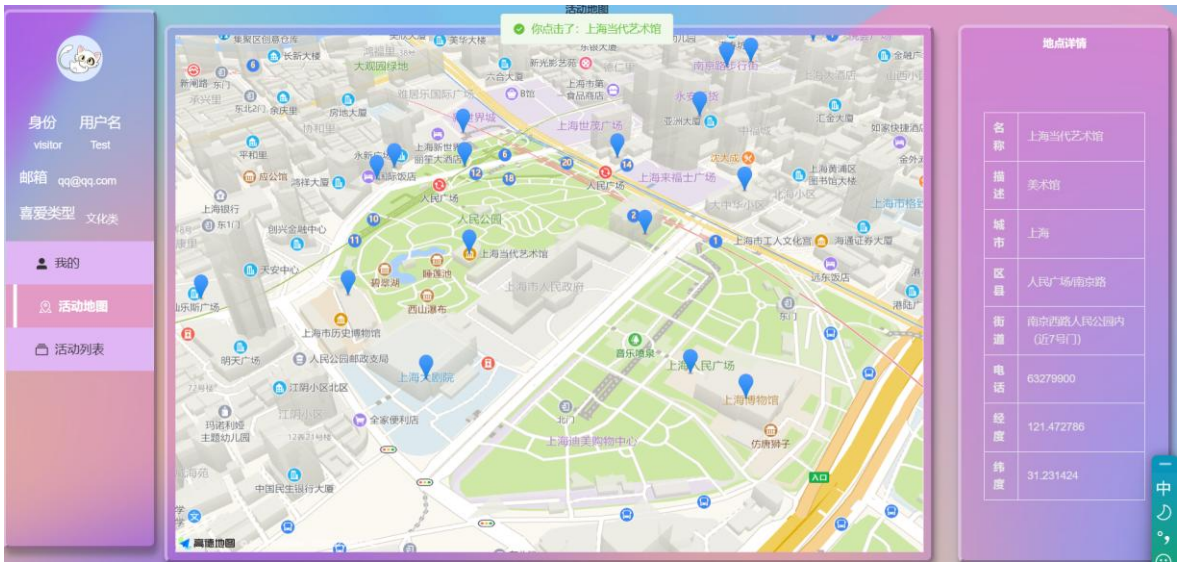


7.3.2 以游玩者登录

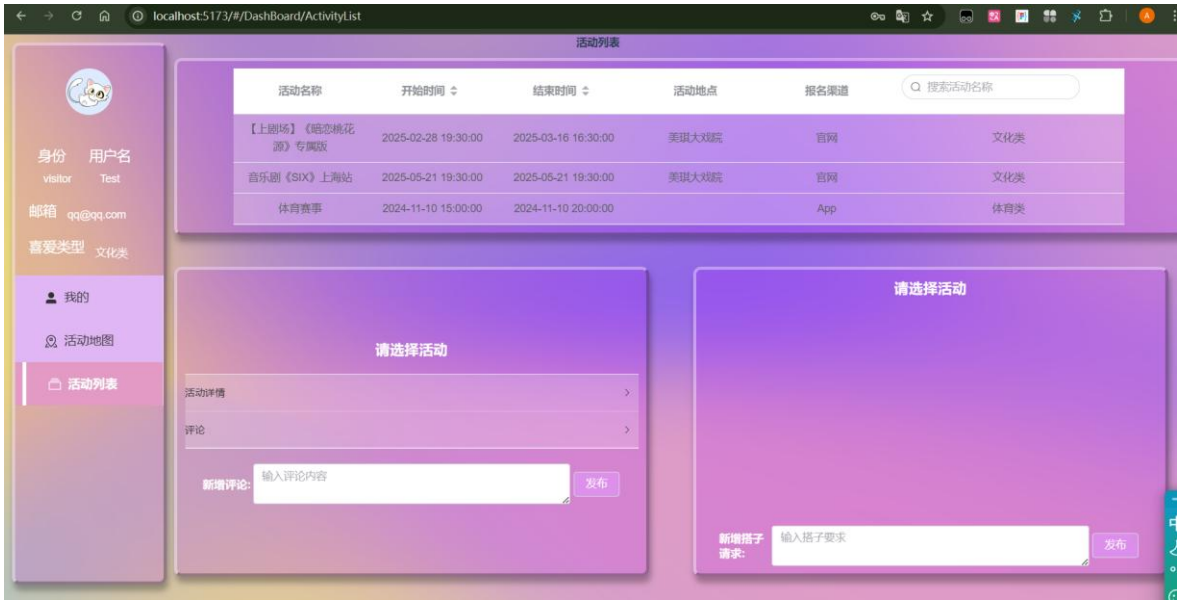
首先进入活动地图，显示地图并标注文旅场所，右侧导航栏显示个人信息



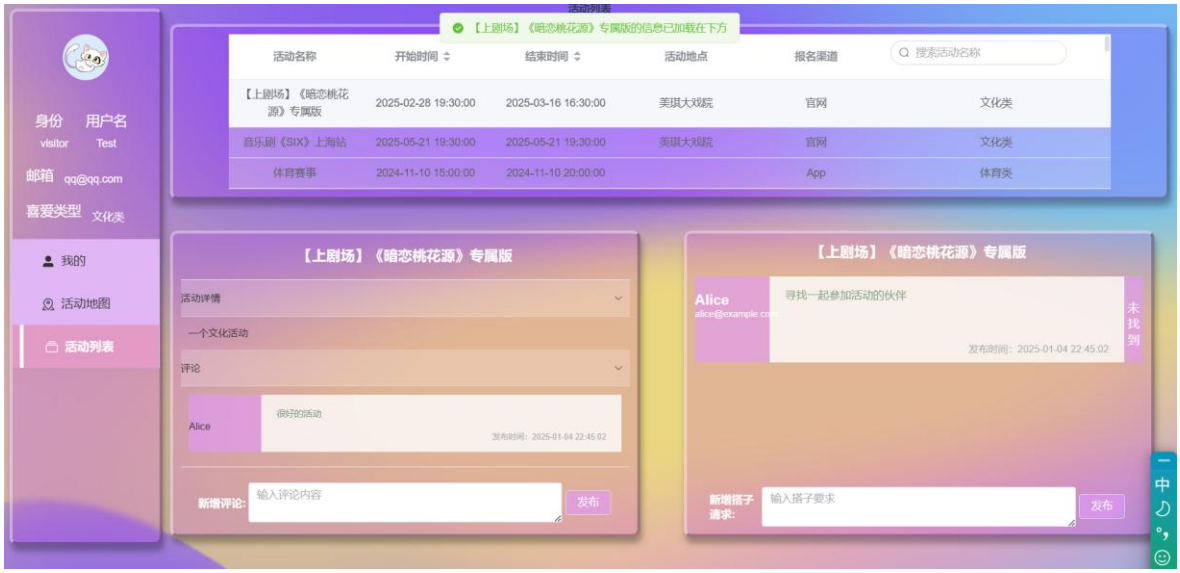
点击蓝色的标注，将显示地点详情



进入活动列表（路由改变），显示所有活动：



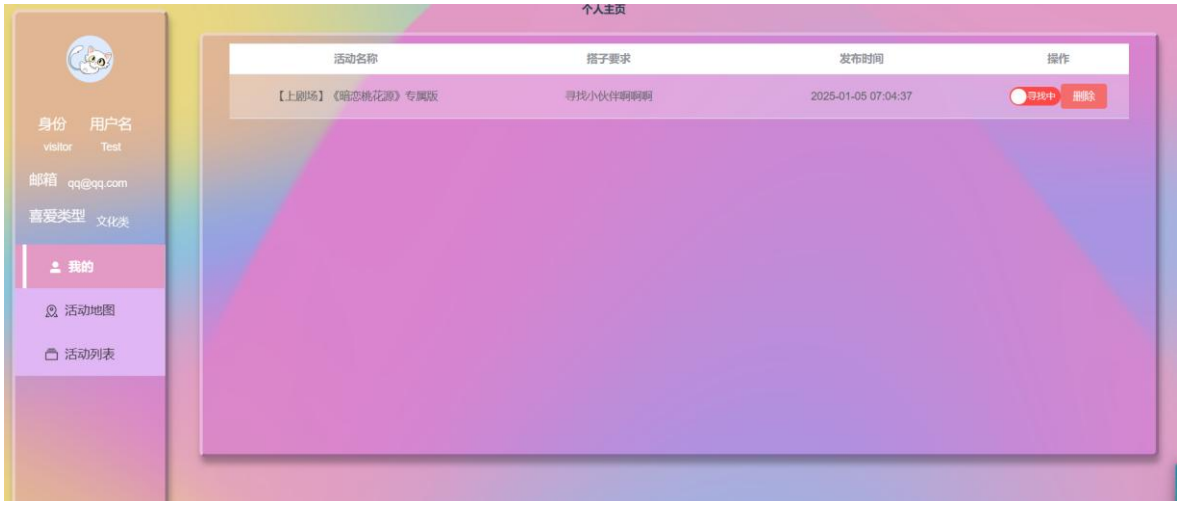
点击表格的行，将显示这一行活动的详情和评论，搭子请求：



可以对活动进行搜索：

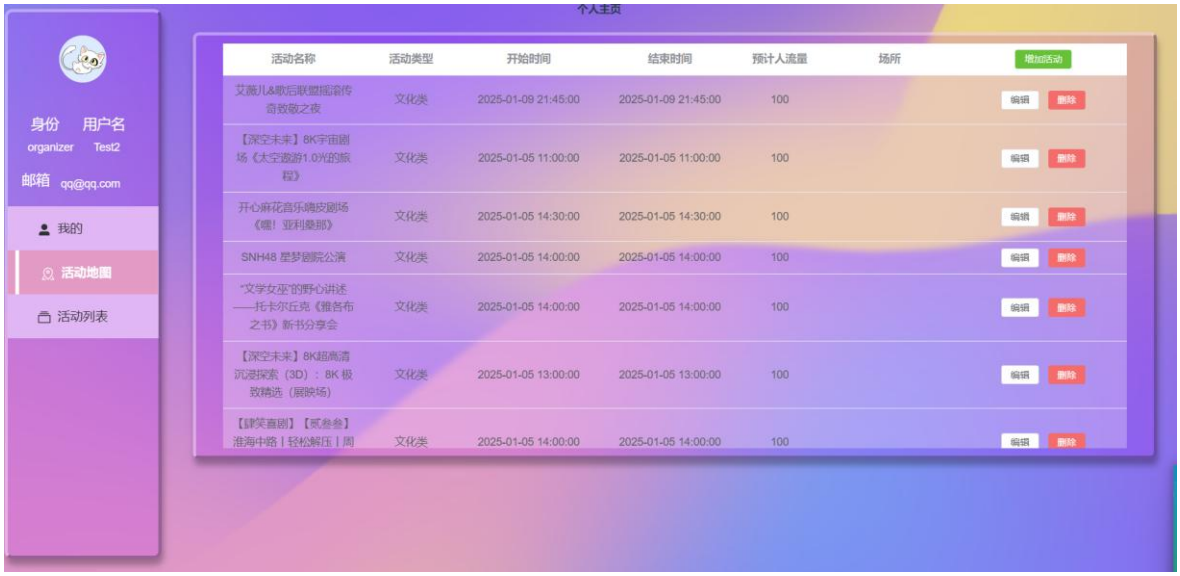
可以发布评论，搭子请求：

在个人界面，可以看到自己发布的搭子请求，可以调整寻找状态，或删除：

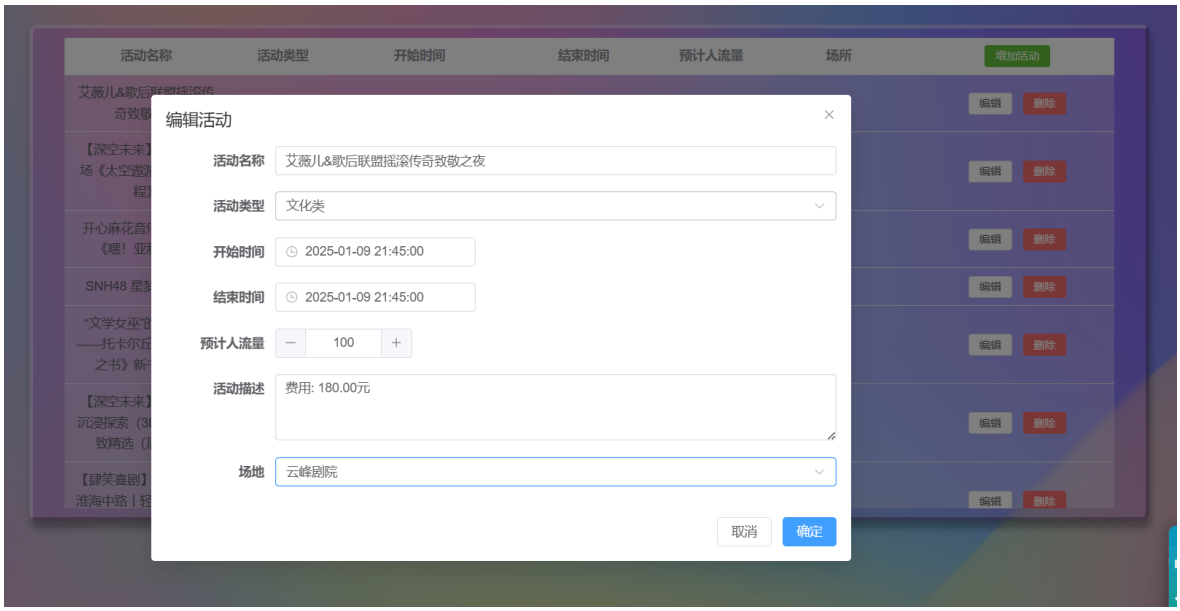


7.3.3 以活动发布者登录：

较之游玩者，最主要的区别在于个人界面，个人界面将显示所有发布的活动（因用爬虫获取的活动没有发布者也没有加场所，这里默认为这次登录用户，换一个账户登录则不会显示这些活动了）

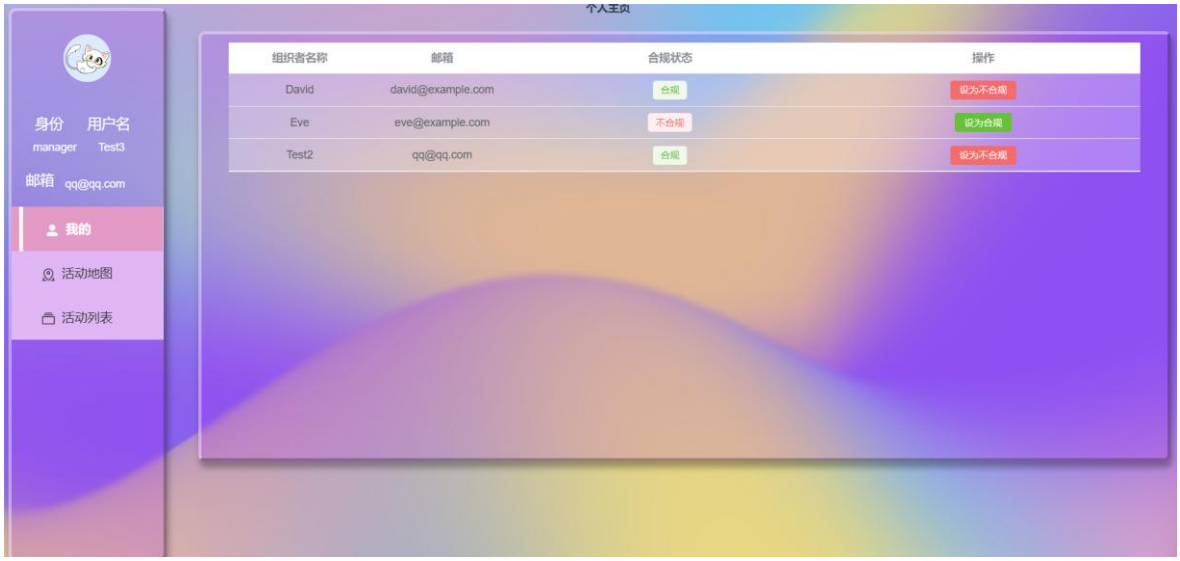


可以在这个页面修改，删除，添加活动，修改添加将弹出对话框，输入相关信息：



7.3.4 以管理者登录

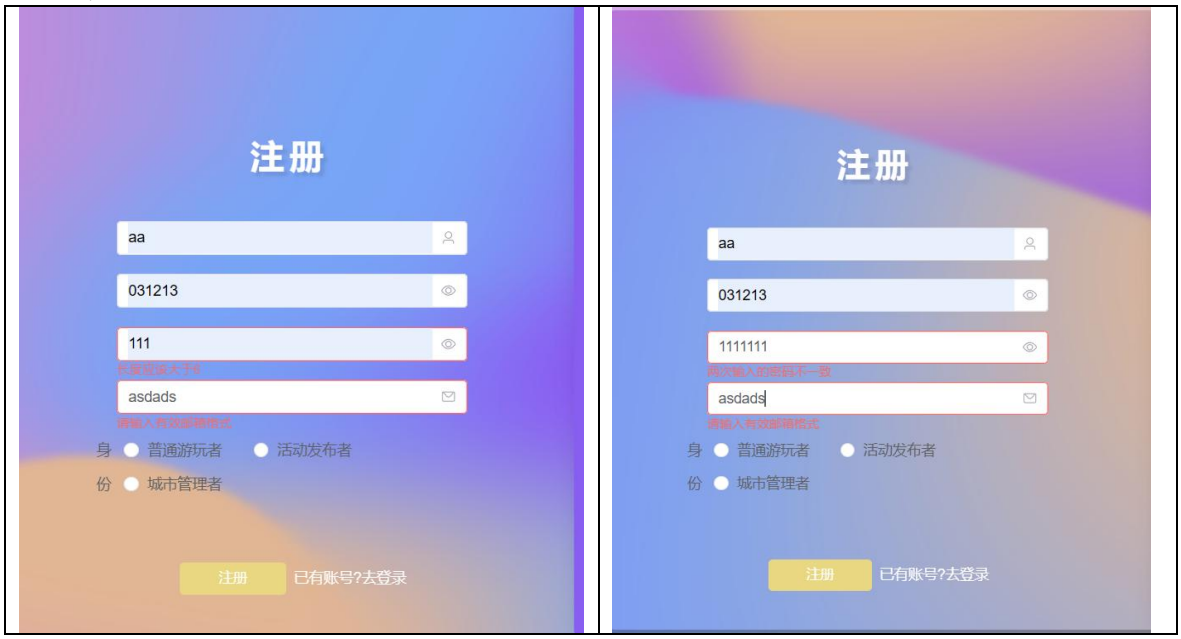
个人界面将显示所有活动发布者，并可更改其是否合规状态：



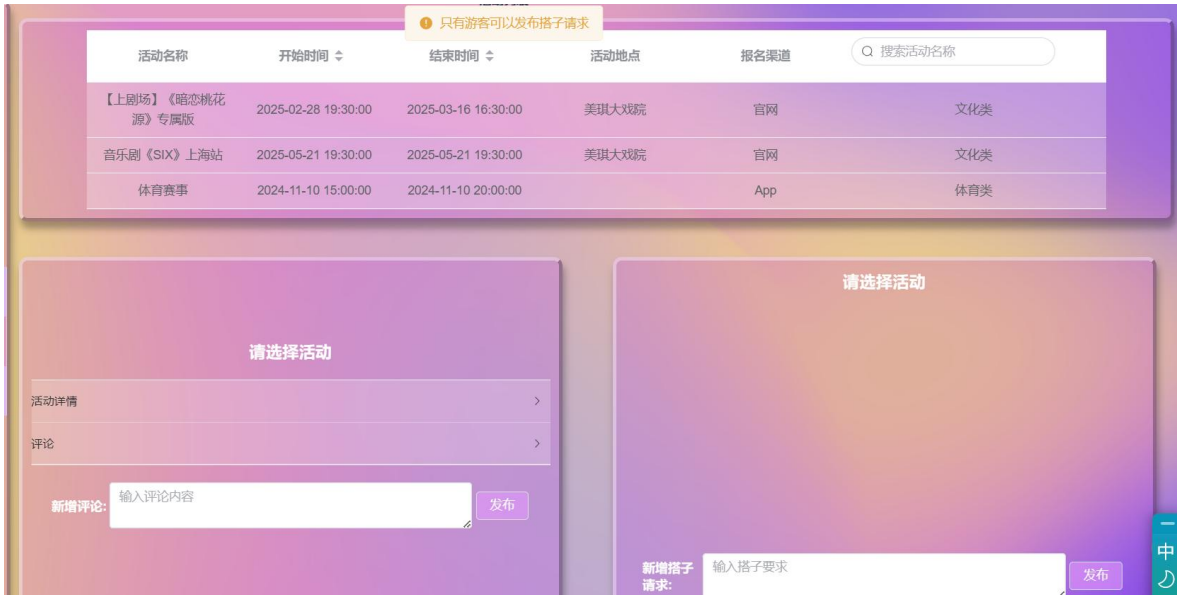
7.4 系统测试

主要进行边界测试，对一些输入输出进行错误提示：

注册时，对密码的确认和邮箱格式进行检验，对密码长度进行检验，对所有输入的非空进行检验,同时每次注册都需保证用户名的唯一性：

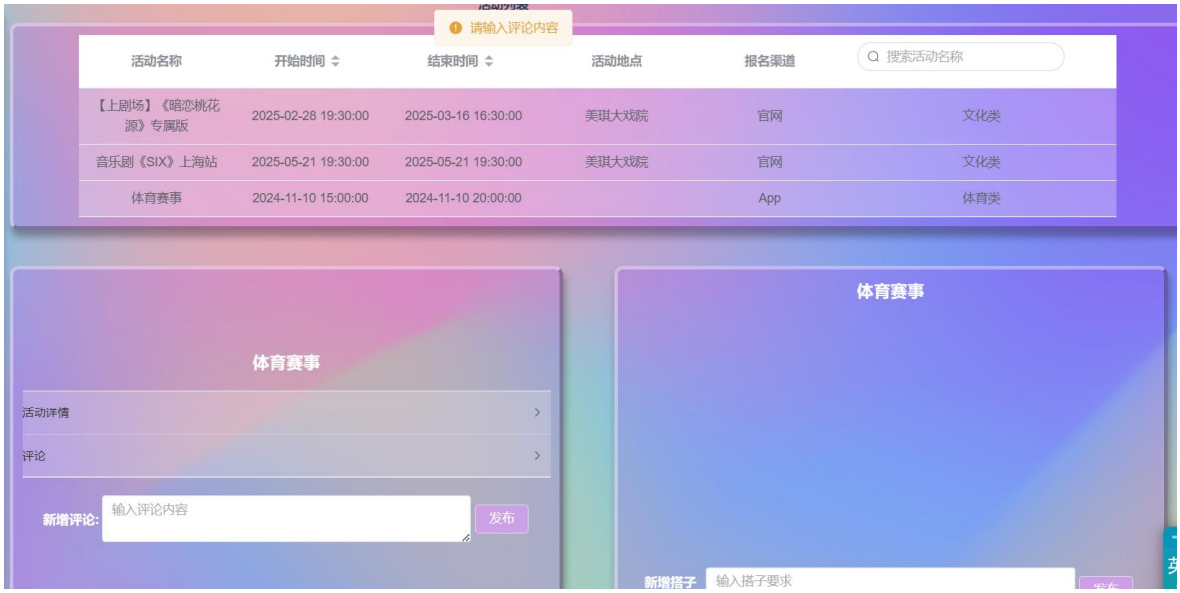


在发布评论和搭子请求时，只有游客身份可以：

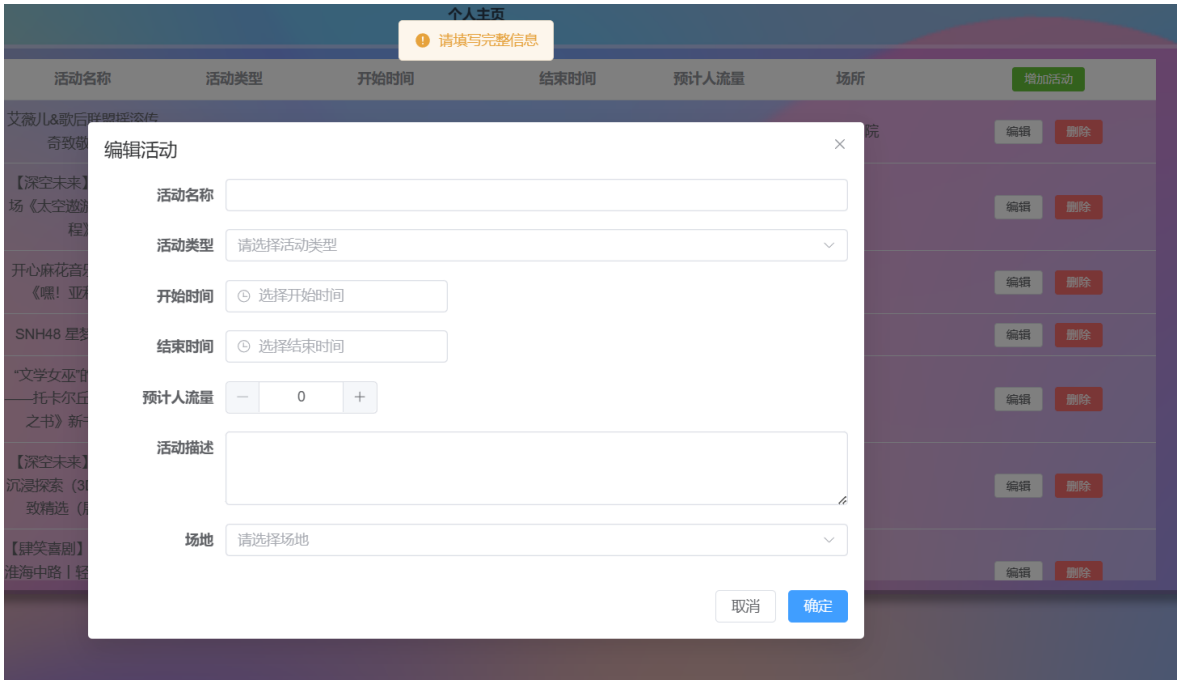


必须选择活动后才可以评论和活动：

输入框空时也不能发布：



在增加活动时，名字不能为空：



7.5 部署说明

数据库重置+数据爬取：flask reset+运行爬虫文件

后端启动：python.exe -m flask run

前端启动：npm run dev

后端启动（打印路由）如图：

```
E:\桌面\数据库\mycode\back\venv\Scripts\python.exe -m flask run
Map([<Rule '/static/<filename>' (OPTIONS, HEAD, GET) -> static>,
 <Rule '/' (OPTIONS, HEAD, GET) -> hello_world>,
 <Rule '/organizer/register' (POST, OPTIONS) -> organizer_api>,
 <Rule '/organizer/login' (POST, OPTIONS) -> organizer_api>,
 <Rule '/manager/register' (POST, OPTIONS) -> manager_api>,
 <Rule '/manager/login' (POST, OPTIONS) -> manager_api>,
 <Rule '/visitor/register' (POST, OPTIONS) -> visitor_api>,
 <Rule '/visitor/login' (POST, OPTIONS) -> visitor_api>,
 <Rule '/venues' (OPTIONS, HEAD, GET) -> venue_api>,
 <Rule '/venuesDetail' (POST, OPTIONS) -> post_venues_id>,
 <Rule '/visitor_buddy' (POST, OPTIONS) -> get_visitor_buddy_requests>,
 <Rule '/changeBuddyStatus' (POST, OPTIONS) -> change_buddy_status>,
 <Rule '/deleteBuddy' (POST, OPTIONS) -> delete_buddy>,
 <Rule '/organizer_activities' (POST, OPTIONS) -> get_organizer_activities>,
 <Rule '/organizers' (OPTIONS, HEAD, GET) -> get_manager_organizers>,
 <Rule '/changeOrganizerStatus' (POST, OPTIONS) -> change_organizer_status>,
 <Rule '/activities' (OPTIONS, HEAD, GET) -> get_activities>,
 <Rule '/activitiesDetail' (POST, OPTIONS) -> get_activity_detail>,
 <Rule '/addActivities' (POST, OPTIONS) -> add_activity>,
 <Rule '/addComments' (POST, OPTIONS) -> add_comment>,
 <Rule '/addPartners' (POST, OPTIONS) -> add_partner>,
 <Rule '/editActivities' (POST, OPTIONS) -> edit_activity>,
 <Rule '/deleteActivities' (POST, OPTIONS) -> delete_activity>])
 * Serving Flask app 'app.py'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

前端启动，如图：

```
E:\桌面\数据库\mycode\front>npm run dev

> front@0.0.0 dev
> vite


  VITE v5.4.10  ready in 585 ms

  ➜  Local:   http://localhost:5173/
  ➜  Network: use --host to expose
  ➜  press h + enter to show help
```

装 订 线

8 总结

8.1 项目总结与不足分析

在本次项目中，我实现了一款基于 Web 网页的上海市文体活动系统，完成了对数据库基本的增删改查操作，实现了预期的用户管理、景点与活动信息管理、发布评论与发布搭子请求四大功能模块。

在需求分析阶段，我深入理解了本项目中用户对活动管理平台的需求，明确了系统的主要用户群体，包括游客、组织者和管理员。通过对不同角色的需求进行详细分析，我确定了用户注册与登录、活动查询、评论与搭子管理等核心功能。此外，我认识到数据的高效展示与实时交互对于用户体验的重要性，这为后续的系统设计奠定了基础。概念设计阶段，我通过绘制 E-R 图，明确了系统中各个实体及其关系，为逻辑设计和物理设计提供了清晰的指导。在逻辑设计中，我遵循数据库的规范化原则，对数据模型进行了优化，以确保数据的一致性和减少冗余。物理设计阶段，我重点对查询性能进行了考虑，使用合适的索引设计，提升了系统的响应速度，这让我认识到数据库性能优化对系统整体效率的影响。系统实现方面，我采用了 Vue3 和 Element Plus 作为前端技术栈，Flask 与 Flask-SQLAlchemy 作为后端框架，成功实现了一个功能完善的活动管理平台。在这个过程中，我深入学习了前后端分离的开发模式，掌握了 HTTP 协议、数据库操作等技术，特别是在接口设计与数据存取优化方面的实践经验，使我对 Web 开发架构和性能优化有了更加深刻的理解。

本次项目还是有很多不足需要改进，比如在数据获取方面，没找到与设计数据库完全相同的数据，没有实地部署在服务器，对性能测试不够详尽等，界面设计在不同设备上兼容性不够好的问题，这也意味着我还需要不断学习，维护提升此系统 。

8.2 收获

这是第一次完整独自完成前后端项目，从需求分析，到数据库设计，再到页面划分与设计，再设计接口，爬取数据，完成前后端配对，从对前后端项目一无所知，到选择前后端技术，再到学习 vue，JS，CSS，flask，直到完完整整完成整个项目，经历各种各样的 bug，比如几乎每个接口都会遇到的跨域问题，比如 flask 更改后需终止现有运行（有时需要使用 taskkill 命令才有效）再重启否则更改无效，比如 css 不断调整反而越调越奇怪，比如有时候问题不在代码上而是前后端运行时过于占用电脑内存导致内存爆满，以至于电脑蓝屏。在完成项目的过程中，伴随着各种 bug，秉持着“兵来将挡，水来土掩”的原则，索幸这些 bug 都解决了。

在整个开发过程中，也实实在在学到了很多，对 web 项目的构架有了更清晰明了的学习，同时在完成项目的同时也是对计算机网络，软件工程课程知识的应用。我不仅学会了如何设计和实现后端接口，还理解了如何规划前端与后端的交互，使得数据能够顺畅地流动。通过不断测试和调试，我提高了对 Web 项目调试和错误处理的能力。此外，项目的实施让我深入理解了前端与后端框架之间的协作，以及如何高效地管理请求与响应，确保前后端分离的架构能够顺利运行。

总而言之，这个项目让我对 Web 开发的各个环节有了更全面的了解，从技术栈的选择到具

体功能的实现，我积累了丰富的实践经验，也提升了自己的解决问题的能力。未来，我希望能够在此基础上继续深入学习，完善自己的开发技能，承担更复杂的项目任务。