Man游项目详细设计说明书

1. 引言

1.1 目的
本说明书旨在详细描述“Man游”旅游规划应用的设计基础，明确系统的功能需求、技术规格及实现方式。通过本文件的编写，旨在为项目的开发、测试和验收提供清晰的指导，确保所有参与项目的人员对系统的业务逻辑、功能模块、数据结构等有统一的理解。该文档将作为设计与开发的依据、测试编写的指导，以及验收的关键标准。

1.2 面向读者
本说明书主要面向以下读者：
- **项目经理**：负责根据本需求文档制定项目计划、协调资源、跟踪进度，确保按时交付符合需求和质量标准的软件产品。
- **前后端开发人员**：依据本需求文档中的功能需求和技术规格进行系统设计与开发，确保按时交付符合要求的软件功能。
- **测试人员**：根据本需求文档中的功能需求和验收标准，设计测试用例并执行测试，确保软件功能和性能符合预期。
- **其他相关方**：如市场调研人员、用户体验设计师、产品所有者等，需根据本需求文档的需求和目标，提供反馈和支持，确保产品符合市场需求和用户期望。

1.3 参考资料
本项目的设计与开发将参考以下资料：
- **软件需求规格**：计算机软件需求规格说明规范。
- **前端设计**：Material Design 指南、Ant Design Vue 文档。
- **前端实现**：uni-app 官方文档、Vue 3 官方文档、Vuex 文档、Vue Router 文档。
- **后端实现**：Django 官方文档、Django REST framework 文档。
- **数据库管理**：PostgreSQL 文档、Django ORM 文档。
- **安全性与隐私保护**：Django 安全最佳实践指南、OWASP 指南、Python cryptography 库文档。
- **前后端交互**：Axios 文档、WebSocket 文档。
- **测试和调试**：Django 测试文档、Jest 文档、Chrome DevTools 文档。

1.4 项目背景
随着旅游市场的快速发展，用户对个性化、智能化的旅游规划需求日益增加。然而，当前市场上存在信息分散、模板化严重等问题，导致用户在制定旅游攻略时费时费力，难以高效地组织出行安排。Man游的目标是简化这一过程，提升用户体验，尤其是针对喜欢参考攻略、完全自主规划或懒于行前准备的不同类型旅行者。

1.5 项目目标
Man游旨在通过整合大语言模型、第三方平台的优质内容、实时天气预报等，提供个性化的旅游推荐和智能行程规划。项目的成功将为用户提供便捷的旅行体验，提升整体旅游满意度，并在市场中占据一席之地。

3. 系统功能模块详细设计

3.1 用户管理模块
- **功能**：实现用户的注册、登录、信息管理。
- **接口**：
  - **登录接口**：
    - **请求方法**
    - **请求参数**：
      - `phone_number`：用户手机号
      - `captcha`：验证码
    - **返回结果**：
      - `status`：登录状态（成功/失败）
      - `message`：提示信息
  - **注册接口**：
    - **请求方法**
    - **请求参数**：
      - `phone_number`：用户手机号
      - `password`：用户密码
    - **返回结果**：
      - `status`：注册状态（成功/失败）
      - `message`：提示信息
    - **界面原型展示**：

3.2 行程推荐模块
- **功能**：根据用户偏好和历史数据，提供个性化的行程推荐。
- **接口**：
  - **行程推荐接口**：
    - **请求方法**
    - **请求参数**：
      - `user_id`：用户ID
      - `preferences`：用户偏好
    - **返回结果**：
      - `recommended_itineraries`：推荐的行程列表
     - **界面原型展示**：

3.3 路线规划模块
- **功能**：根据用户选择的景点，提供最优路线规划。
- **接口**：
  - **路线规划接口**：
    - **请求方法**
    - **请求参数**：
      - `start_location`：起点
      - `end_location`：终点
      - `transport_mode`：交通方式
    - **返回结果**：
      - `optimal_route`：推荐的最优路线
      - `estimated_time`：预计时间
    - **界面原型展示**：

3.4 预算管理模块
- **功能**：帮助用户管理旅行预算，记录支出。
- **接口**：
  - **预算记录接口**：
    - **请求方法**
    - **请求参数**：
      - `user_id`：用户ID
      - `amount`：支出金额
      - `category`：支出类别
    - **返回结果**：
      - `status`：记录状态（成功/失败）
      - `message`：提示信息
    - **界面原型展示**：

3.5 天气信息模块
- **功能**：提供实时天气预报，帮助用户合理安排行程。
- **接口**：
  - **天气查询接口**：
    - **请求方法**
    - **请求参数**：
      - `location`：查询地点
    - **返回结果**：
      - `current_weather`：当前天气信息
      - `forecast`：未来几天的天气预报
    - **界面原型展示**：

3.6 预约提醒模块
- **功能**：根据用户的行程，自动检测是否需要预约并提醒用户。
- **接口**：
  - **预约提醒接口**：
    - **请求方法**
    - **请求参数**：
      - `user_id`：用户ID
      - `appointment_details`：预约详情
    - **返回结果**：
      - `status`：提醒设置状态（成功/失败）
      - `message`：提示信息
    - **界面原型展示**：

3.7 备忘录模块
- **功能**：支持用户记录账单、设置旅行预算以及整理行李清单等。
- **接口**：
  - **备忘录接口**：
    - **请求方法**
    - **请求参数**：
      - `user_id`：用户ID
      - `memo`：备忘录内容
    - **返回结果**：
      - `status`：记录状态（成功/失败）
      - `message`：提示信息
    - **界面原型展示**：
  ## 4. 性能设计

4.1 响应时间
- **系统登录时间**：不超过1秒。
- **页面请求时间**：不超过2秒。
- **数据处理时间**：不超过2秒。

4.2 并发用户数
- **高峰期并发用户数**：系统支持高峰期并发用户数达到xxx人。

4.3 数据处理能力
- **请求处理能力**：系统应能处理每秒xxx条请求，确保在高并发情况下的稳定性。

4.4 资源使用
- **内存使用**：系统应优化内存使用，确保在高负载情况下仍能保持流畅运行。
- **CPU使用**：系统应合理分配CPU资源，避免过载。

4.5 可扩展性
- **系统架构**：设计应支持水平扩展，以便在用户量增加时能够快速扩展资源。
- **模块化设计**：各功能模块应独立，便于后续功能的扩展和维护。

7. 系统出错处理设计

7.1 出错信息
- 系统在用户使用过程中出现错误时，需提供明确的错误提示信息，帮助用户理解问题所在

7.2 错误类型
- **数据库连接错误**：
  - **处理方式**：

- **用户输入错误**：
  - **处理方式**：

- **系统内部错误**：
  - **处理方式**：

7.3 补救措施

- **反馈渠道**：
  - 用户可以通过应用内反馈功能报告问题，开发团队将及时处理。

7.4 错误日志记录
- 系统应记录所有错误信息，包括时间戳、错误类型、用户ID等，以便后续分析和改进。
8. 系统处理规定

8.1 输入输出要求
- **输入数据有效性**：
  - 系统应确保所有输入数据的格式和内容有效，避免无效数据导致的错误。

- **输出数据格式**：
  - 系统输出的数据应符合预期格式，确保用户能够清晰理解。

8.2 数据管理能力要求
- **定期备份**：
  - 系统应具备定期备份和恢复数据的能力，确保数据安全，防止数据丢失。

- **数据一致性**：
  - 在数据操作过程中，系统应确保数据的一致性，避免出现脏数据。

8.3 故障处理要求
- **自动监测机制**：
  - 系统应具备自动监测和报警机制，及时发现并处理故障，确保系统的高可用性。

- **用户通知**：
  - 在发生故障时，系统应及时通知用户，并提供相应的解决方案或建议。

8.4 安全性要求
- **数据加密**：
  - 系统应对敏感数据进行加密处理，确保用户信息的安全性。

- **权限管理**：
  - 系统应实施严格的权限管理，确保用户只能访问其有权访问的数据和功能。