中山大學

SUN YAT-SEN UNIVERSITY

软件工程

Yatmosphere 智能家居控制系统项目管理

团 队 Yatmosphere

学 院 计算机学院

专 业 计算机科学与技术

2025 年 6 月 15 日

目录

1 项目概述

1.1 项目名称

Yatmosphere 智能家居控制系统

1.2 执行周期

2025 年 4 月 22 日–2025 年 6 月 15 日（约为 2 个月）

1.3 项目目标

本项目旨在开发一个高效、可靠、易用的智能家居控制系统——Yatmosphere，满足用户对智能生活场景的全方位控制需求。项目具体目标包括：

- 构建统一的全屋智能设备管理平台，实现多类型设备（如灯光、空调、传感器等）的一体化接入与统一控制；
- 实现核心功能模块：设备状态监测、远程控制、场景转换、自定义场景、设备自由添加、安全权限管理等；
- 采用前后端分离架构，前端使用 Vue 技术栈构建响应式 UI，后端基于 Java 与 Spring Boot 提供 RESTful API 服务，设备通信层采用 MQTT 协议，实现低延迟、高可靠的实时消息传输；
- 实现多端适配，系统支持在 PC、手机等终端自适应运行，保障用户在不同场景下的便捷访问；
- 强调用户体验与可用性，提供清晰的交互流程与友好的界面设计；
- 注重系统可扩展性与可维护性，为后续集成语音控制、智能推荐、数据可视化等功能预留接口；
- 加强数据安全与访问控制，保障用户隐私与设备控制安全。

2 项目计划制定

2.1 里程碑 & 时间节点

编号	名称	说明	开始	结束	责任人
M1	需求确认	完成需求规格说明书评审，明确多端适配与通信需求	04-22	04-28	马岱
M2	原型与 UI 设计	完成系统原型图与 UI 交互界面设计初稿	04-25	05-01	UI 设计师
M3	模块设计完成	提交前端、后端、MQTT 通信、安全模块设计方案	04-29	05-07	各模块负责人
M4	多端前端开发	完成 PC 端与移动端界面及交互逻辑开发	05-08	05-18	前端组
M5	后端核心接口实现	实现设备管理、控制、权限、用户管理等 REST 接口	05-08	05-20	后端组
M6	MQTT 通信与联调	完成 MQTT 消息通道接入与设备通信联调测试	05-15	05-25	后端 + 硬件接口组
M7	权限与安全机制完善	完成权限模型设计、加密通信机制与用户认证功能	05-20	05-28	安全负责人
M8	系统集成测试	全功能联调，完成黑盒测试、压力测试与性能调优	05-29	06-05	测试组 + 全体
M9	用户体验与演示准备	内部用户验收测试 (UAT)，优化 UI/交互体验	06-06	06-10	UI + 测试 + 前端
M10	上线部署与最终验收	项目部署至演示环境，完成演示与验收材料准备	06-11	06-15	运维 + 项目组

2.2 每周目标

周次	日期范围	主要目标
第 1 周	4 月 22 日—4 月 30 日	完成需求调研与用户访谈，分析典型智能家居使用场景；初步梳理功能模块（控制、监测、权限等）；搭建 Jira 看板与 GitHub 仓库；启动《需求规格说明书》撰写；制定甘特图（使用 Microsoft Project）
第 2 周	5 月 1 日—5 月 7 日	完成系统总体架构设计（前后端分离结构、MQTT 通信、Spring Boot 技术选型）；设计数据库 ER 图与数据流图；确定设备通信模型与 REST/MQTT 接口规范；绘制 UI 原型（响应式布局、多端适配）
第 3 周	5 月 8 日—5 月 14 日	各小组完成详细模块设计：前端（Vue 多端适配策略）、后端（Java Spring Boot 服务接口）、通信服务（MQTT Broker 封装与 Topic 策略）、安全模块（认证与权限控制）；确定各模块职责边界；初步完成 Swagger 接口文档；设计视觉稿与组件规范
第 4 周	5 月 15 日—5 月 21 日	初始化开发环境与框架配置：Vue3 + Vite 前端工程，Spring Boot 后端服务，Eclipse Mosquitto MQTT Broker；完成前端首页、设备控制页面结构开发；实现后端用户与设备模块基本接口；搭建联调用例与模拟数据源
第 5 周	5 月 22 日—5 月 28 日	开发前端核心交互模块（控制面板、场景联动配置、设备实时状态展示）；实现后端业务逻辑（场景触发器、状态订阅发布、用户权限管理）；实现 MQTT 模拟设备脚本；更新文档与测试用例

第 6 周	5 月 29 日—6 月 4 日	完成多端适配支持（响应式样式适配手机、平板端），引入 Element Plus 与自定义组件库优化 UI；后端实现与设备模拟器联动并支持并发测试；验证通信稳定性与接口完整性；统一接口校验与错误处理
第 7 周	6 月 5 日—6 月 11 日	联调与系统测试：集成测试所有模块通信流、状态更新、权限访问；进行性能测试（MQTT 吞吐、并发负载）；Bug 修复与优化，补全文档（技术文档、测试报告、部署说明）
第 8 周	6 月 12 日—6 月 15 日	容器化部署与上线（Docker Compose + Nginx 部署方案）；完成最终功能验收与冒烟测试；整理项目总结文档与技术移交材料；准备结题答辩

2.3 甘特图

符合以上要求的具体甘特图如下所示：

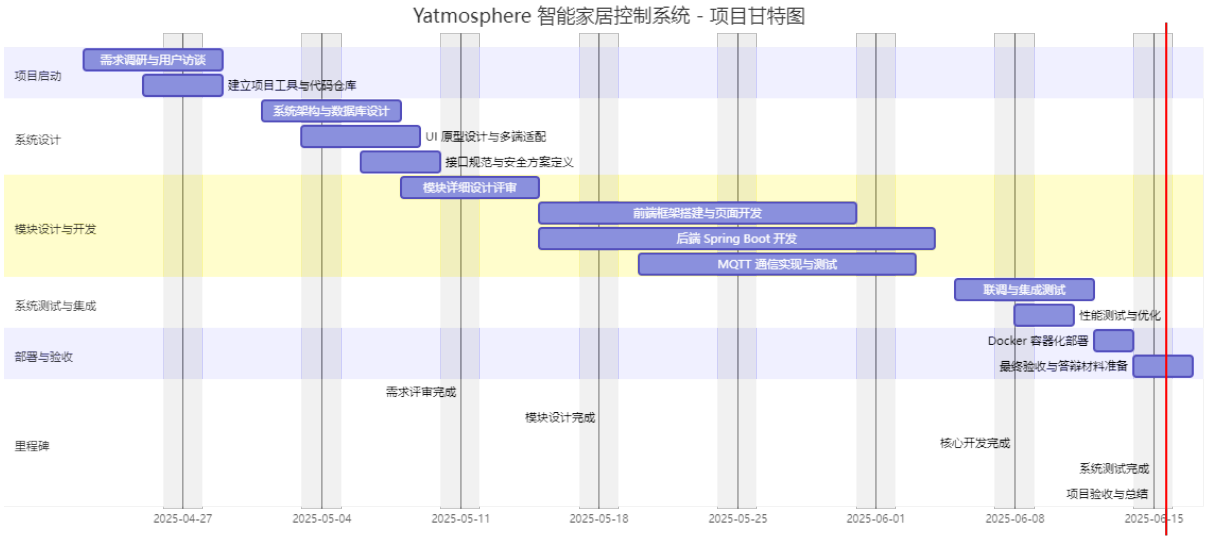

图 1: 甘特图

3 团队组织与分工

3.1 团队结构

为保障项目的高效推进与质量交付，项目团队采用模块化 + 跨职能协作模式，设置明确的岗位职责，并设立定期同步机制与协作流程。整体团队结构如下：

角色	**职责说明**
项目经理	统筹项目整体进度与资源调度，组织需求评审、里程碑验收会议，负责风险预警与协调；主导项目计划撰写与变更管理；确保 Jira 看板、日报、周报更新
前端工程师	基于 Vue 技术栈完成 Web 与移动端界面开发；对接 REST API 与 MQTT 通信层；优化多端适配与响应式布局，保障页面交互一致性和性能表现
UI 设计师	完成低保真原型、交互流程、高保真视觉稿设计，输出设计规范文档（组件风格、配色方案、动效指引）；协助前端工程师进行还原验收及适配调优
后端工程师	使用 Spring Boot 开发核心业务模块（设备控制、场景引擎、用户管理）；设计数据库模型与接口逻辑；集成 MQTT 协议处理模块；负责 API 文档编写与服务部署
接口开发	负责设备层通信协议的实现（MQTT 消息格式、主题结构）；测试并验证 MQTT 消息流转逻辑与服务稳定性；参与前后端接口联调，编写自动化测试脚本
安全工程师	设计用户认证与授权机制（JWT / OAuth2）；配置数据加密传输、接口签名验证；实施输入校验、权限边界测试、防御 XSS/CSRF 攻击；执行依赖扫描与渗透测试
测试工程师	编写系统测试计划（功能、性能、安全、接口）；维护自动化测试用例并持续执行；协助开发进行缺陷定位与回归测试；组织系统验收与用户验收阶段的测试记录
运维工程师	搭建 Docker 化部署流程，维护 CI/CD 管道；配置开发/测试/生产环境；实现日志采集与服务监控；制定备份与容灾策略，确保部署可靠与性能可观测性

3.2 协作机制

- **项目管理工具**：采用 Jira 管理任务分配与进度跟踪，配合 GitHub 进行代码版本管理与代码审查，保障开发过程可审计、可追踪；
- **沟通机制**：每周四召开一次站会（Scrum 例会），同步阶段进度、阻碍事项与下周计划；需求评审、测试评审等采用文档协同 + 实时会议形式；
- **版本协作**：采用 Git 分支策略（主干 + 开发 + 功能分支）进行协作，保障主干稳定，开发迭代有序推进；
- **任务分配与考核**：所有任务按模块划分，责任人明确并记录在 Jira 中，任务完成质量纳入阶段性评审，促进项目目标达成。

3.3 人员匹配与分工依据

根据团队成员的技术专长与兴趣方向进行了初期任务匹配与灵活调整：

- 技术背景匹配职责：如熟悉 Java 的同学主导后端业务模块，具备 Web 前端开发经验者承担 Vue 相关开发；
- 跨模块协作机制：设计、开发、测试各环节间设置对接点，如 UI 提供标注稿 → 前端还原开发 → 测试用例验收；
- 风险缓冲与替代策略：为关键岗位设置“协助成员”，提升关键路径抗风险能力。

团队通过职责明确、流程规范与工具支持，构建了高协作、高响应的开发机制，确保各阶段目标按期保质完成。

3.4 项目实际分工

项目经理：马岱

- 前端组 (4 名)
  - 前端工程师：谢泽中（负责人）、汪丁宇洋、张管文
  - 前端/移动集成：潘致远
- UI 设计 (1 名)
  - 视觉 & 原型：潘致远

- 后端组 (3 名)
  - 数据库设计：庄云皓（负责人）
  - 各接口设计：金鹏飞
  - mqtt 通信、大模型集成：马岱
- 设备层 & 中间件 (2 名)
  - 设备层 & 中间层：张贞蔚（负责人）、潘文磊
- 安全组 (1 名)
  - 用户管理 & 安全维护：黄集瑞
- 测试 & 部署 (2 名)
  - 测试、部署与运维：马岱、潘文磊

4 进度监控与风险管理

4.1 进度监控

为了确保项目进度的有效监控，我们将采取以下措施，结合项目管理工具和会议机制，实现实时跟踪与调整：

- **每日站会（Daily Stand-up）**：每个小组（前端、后端、测试）每日早上进行简短的站会，汇报前一天完成的工作、当日计划及当前阻碍点。确保每个人的任务对齐，并能够及时发现并解决问题。
- **周度评审（Weekly Review）**：每周四下午召开进度评审会议，全面检查各个模块的任务完成情况，更新任务的进展状态，并根据甘特图调整任务的优先级和时间节点。
- **看板管理（Kanban Management）**：项目管理工具（如 Jira）中将任务按 Epic → Story → Task 划分，每个任务分配负责人，并实时更新任务状态（Done / Doing / ToDo）。通过可视化看板确保每个任务的进展透明、清晰，所有团队成员都可以实时查看。
- **燃尽图（Burndown Chart）**：基于 Sprint 和任务的分配情况，每日更新燃尽图，实时跟踪剩余工作量，确保项目不会出现进度滞后的问题。通过数据分析提前发现项目风险。
- **Git 提交与回退机制（Version Control Conflict Management）**：所有代码将通过 Git 进行管理。若出现冲突，开发人员会及时解决并回退到可用的版本，确保版本迭代的稳定性。

4.2 风险识别与应对

项目中的风险识别与管理是确保项目按时交付的关键。我们通过定期评估并制定应对策略，以确保风险可控。具体措施如下：

- **定期风险评估：**项目经理每两周对项目中可能出现的风险进行重新评估，并调整应对策略。团队成员也会在每次站会上报告潜在风险。
- **风险监控与调整：**所有识别出的风险将被记录并纳入 Jira 看板，分配负责人并定期更新风险状态，必要时调整项目优先级和资源分配。

风险类别	风险描述	可能性	影响	应对策略
技术风险	MQTT、大数据量场景下性能瓶颈	中	高	提前搭建压力环境测试，优化 Broker/QoS，采用负载均衡与集群部署
接口兼容风险	前后端接口不一致导致联调失败	高	中	提前定义接口文档，使用 Swagger 提供 Mock 服务，并定期进行接口自动化测试
人员风险	关键人员离职或任务遗漏	低	高	设立双人负责制、加强知识共享；采用 Jira 等工具记录每个任务的细节，确保知识传递无障碍
进度风险	功能开发延迟	中	高	每周进行任务细化与优先级调整，确保核心模块优先开发，调整次要功能开发顺序
安全风险	权限绕过、数据泄露	低	高	安全团队执行渗透测试与代码审计，定期进行安全培训与漏洞修复，严格权限控制
部署风险	环境差异导致线上部署失败	低	中	使用 Docker 容器化部署，采用 CI/CD 完成自动化部署与环境一致性验证，定期进行灾难恢复演练

4.3 团队合作与风险管理细化

- **团队协作与知识共享：**定期组织技术分享会和回顾会议，确保团队成员对当前任务的技术细节有足够的了解，并及时传递关键知识，避免出现任务遗漏或理解偏差。
- **风险归属与责任人：**每个风险的应对策略都将指定专门的责任人，确保每一项风

险都有专人负责。风险评估和应对策略将在每周的进度评审会上进行更新和讨论。

- **应急响应机制：**对于高影响、高可能性的风险，建立应急响应小组，确保在风险发生时能够迅速反应并采取措施，避免对项目进度造成影响。

5 项目管理规范

为保障项目实施的规范性和一致性，我们制定以下管理标准，涵盖代码管理、文档维护、协作流程等方面：

- **分支策略**：采用 Git Flow 模型，包含 `main`（生产分支）、`develop`（开发集成分支）以及若干 `feature/name` 分支用于新功能开发，确保多人协作下的代码稳定性与可回滚性。
- **版本管理**：遵循 Semantic Versioning 语义化版本控制规范（如 `v1.2.0`），在每次重要发布时进行打 Tag 标记，便于历史追踪与回退。
- **代码评审（Code Review）**：所有合并至 `develop` 分支的 Pull Request（PR）必须经过至少 1 位项目成员 Review，并通过自动化测试后方可合并，保证代码质量与逻辑一致性。
- **提交规范**：使用 Conventional Commits 格式，例如 `feat: 添加设备控制接口`或 `fix: 修复 UI 样式错位问题`，方便自动生成 changelog 与分析版本变更。
- **文档管理**：使用 Markdown 统一编写项目文档，接口采用 Swagger/OpenAPI 规范管理，所有文档集中托管于 GitHub 仓库，便于版本控制与团队协作。
- **会议频率**：设定项目常规沟通机制：
  - 日会：每日 15 分钟站立会议，快速同步昨日进展、当日计划与当前阻碍；
  - 周会：每周五晚上进行小组会议，包含进展汇报、问题复盘、头脑风暴与下周计划；
  - 里程碑评审：在每一阶段交付前组织全体评审会议，确认交付物质量与进度是否达标。

6 沟通与报告

为保证高效的团队协作与信息同步，制定如下沟通与报告机制：

- **沟通工具**：

  - 微信群：日常事务、快速同步与紧急通知；
  - GitHub Issues：记录任务 Bug 与功能迭代建议，确保技术沟通结构化与可追踪；
  - Jira：任务分配、状态流转（To Do / In Progress / Done）与燃尽图展示，用于敏捷开发过程的全程可视化管理。

- **项目进展报告**：
  - 每周汇总：小组成员提交每周工作成果与困难，通过 · GitHub issue 跟踪；
  - 里程碑报告：每一里程碑前一日，完成成果整理并上传至 GitHub 仓库供全体评审。

7 附件

- 项目计划甘特图（见本文档 §2.3）
- 团队组织结构图（见本文档 §3）
- GitHub 仓库：https://github.com/sysu-orz/yatmosphere

至此，项目管理文档编制完毕，请各位成员对照执行，有问题及时在项目群或 *Jira* 中反馈。祝项目圆满成功！