### 3. 进度计划

#### 3.1实现架构

项目的实现架构将分为前端、后端、数据库、第三方服务四个主要部分：

· **前端架构**

Web端：使用React.js或Vue.js作为前端框架，结合Tailwind CSS或Bootstrap进行响应式设计，确保Web端界面在PC端和移动端均有良好的用户体验。

微信小程序：采用微信小程序原生框架进行开发，配合WeUI组件库，确保界面符合微信小程序的设计规范，优化用户体验。

此框架将提供用户友好、响应迅速的界面，允许学生快速发布和查找任务。

功能模块：

* + 用户注册与登录界面（Web端使用OAuth2或JWT，小程序端使用微信登录）
  + 任务发布与接单界面
  + 任务详情展示与路径规划（小程序端使用微信地图API，Web端使用高德地图API）
  + 实时消息推送界面（通过Socket.IO或WebSocket）
  + 支付接口与用户钱包功能界面（小程序端使用微信支付，Web端集成微信支付和支付宝）
  + 用户评价系统

· **后端架构**

使用Node.js（Express）或Python（Django/Flask）作为后端服务框架，负责任务发布、接单、支付及通知管理。

功能模块：

* + 任务管理：任务的发布、接单、取消等逻辑处理
  + 实时消息推送：集成Socket.IO或WebSocket实现实时通信
  + 用户管理：支持Web端OAuth登录和微信小程序登录
  + 信用体系：包含信用评价、任务记录
  + 支付系统：集成微信支付或支付宝，处理在线支付与资金结算
  + 安全管理：确保数据传输与支付安全
  + 任务推荐功能：基于用户位置和行程的任务推荐，提升匹配效率

· **数据库架构**

数据库：采用MongoDB或MySQL存储用户数据、任务信息、接单记录、支付记录等。需要设计任务的快速查询与筛选，支持地理位置匹配。

数据库模块：

* + 用户表：存储用户基本信息与信用评分
  + 任务表：存储任务内容、发布者与接单者信息、路线等
  + 支付记录表：记录任务酬劳的支付与接收信息
  + 评价表：保存用户互评信息

· **第三方服务**

地图服务：Web端集成高德地图API，小程序端使用微信地图API，提供地点选择、路线规划、任务地点匹配功能

支付服务：集成微信支付或支付宝API，完成支付流程的集成

#### 3.2技术路线

· **代码管理与协作：**使用Git和GitHub进行代码管理与协作

· **前端**: React/Vue.js进行UI开发，Bootstrap/Tailwind CSS进行响应式设计。

· **后端**: Node.js/Express 或 Python/Django/Flask构建服务端，处理任务发布与接单。

· **实时消息推送**: 使用Socket.IO/WebSocket实现任务通知和实时更新。

· **数据库管理：MongoDB或MySQL用于存储数据，设计支持高效查询和任务筛选。**

· **支付系统**: 支付宝API或微信支付API，处理在线支付。

· **地图与路径规划**: 集成高德地图API/微信地图API，实现地图展示与路径规划。

· **任务推荐系统：根据地理位置和用户行程，提供任务推荐**

任务推荐系统：初步设想是基于用户当前的位置和行程来推荐合适的任务。若用户未设置行程，系统会根据用户当前所在的位置，推荐10分钟内可完成的任务，以提升时效性。如果用户设置了行程，比如从教学楼到宿舍的路径，系统会优先推荐在用户行程沿线的任务，确保用户不需要绕路。此外，任务的优先级还会根据发布时间、任务紧急度等多个因素综合排序。通过这样的匹配机制，最大化顺路任务的完成率，提升平台的使用效率。

#### 3.3人员计划

· **项目统筹与原型设计**: 1人，负责统筹项目总体规划与原型设计。

· **前端开发**: 1人，负责界面设计与前端功能实现。

· **后端开发**: 2人，负责后台架构、数据库管理及API开发。

· **算法设计**: 1人，负责任务推荐算法设计与实现。

·  **文档编写：**团队完成，项目实施方案编写。

· **PPT制作：**团体完成，项目展示PPT。

#### 3.4开发计划

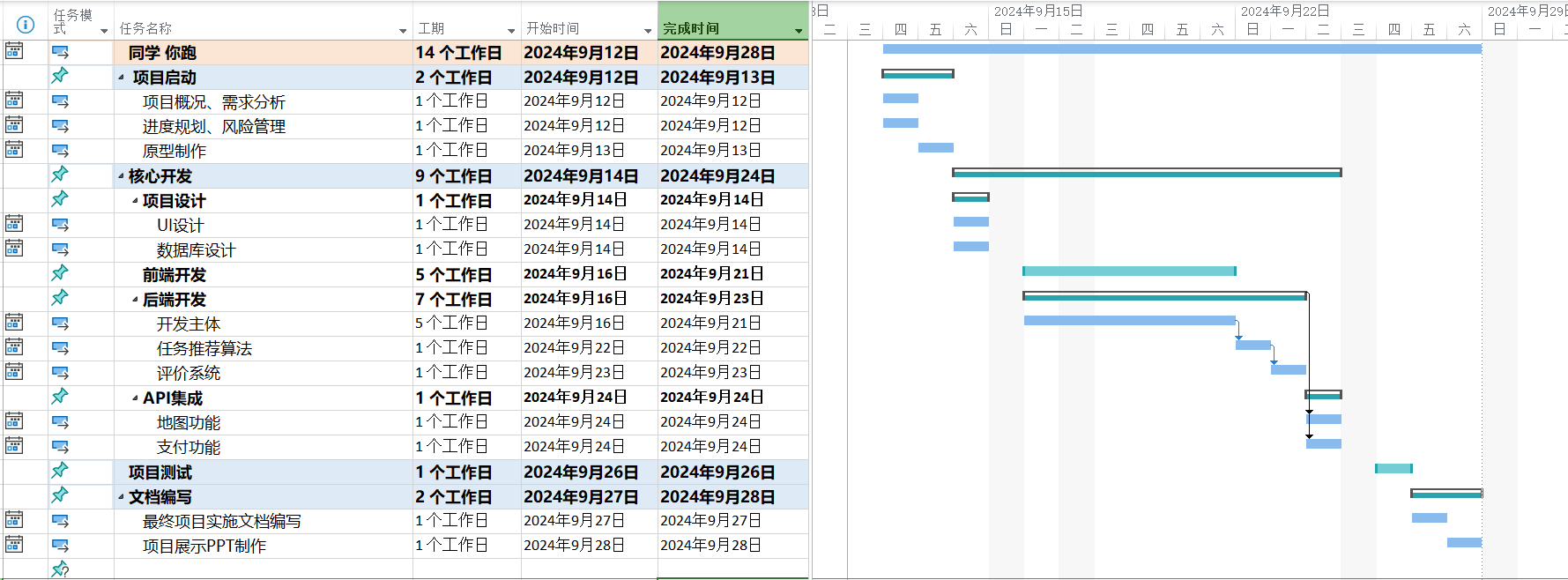
1. 项目启动阶段：明确产品需求，完善需求文档。

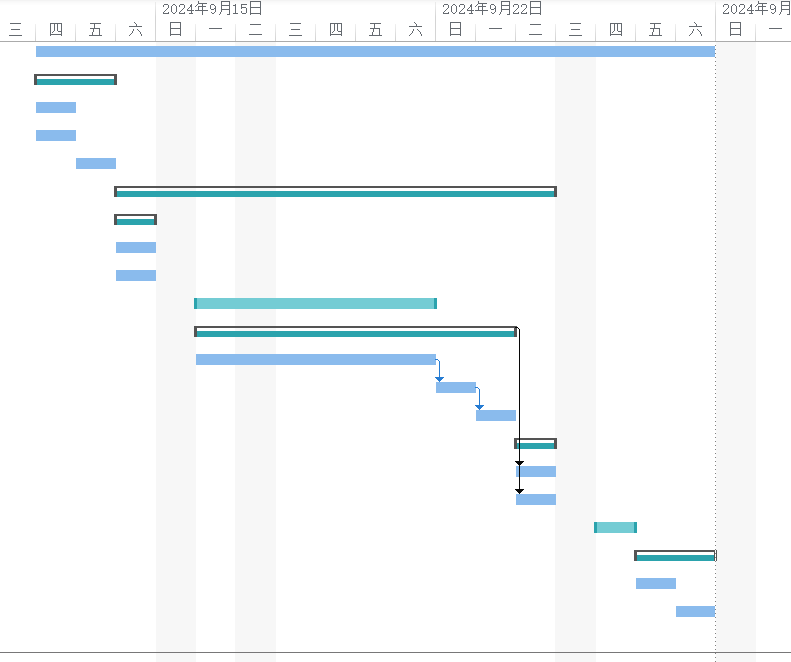
2. 核心开发阶段：并行开发前端和后端，完成界面设计、任务发布、接单、消息推送、支付集成等功能模块，开发任务匹配算法。

3. 测试阶段：进行功能测试、优化与BUG修复，确保系统稳定运行。

4. 验收阶段：完成项目实施方案与PPT，进行项目展示和最终验收。







### 风险管理

4.1 风险来源

4.1.1需求风险

①需求变动风险：需求在开发过程中可能发生变化，导致开发周期延长。

②需求不明确: 初期需求不完全清晰，可能导致部分功能开发后不符合用户预期。

③需求过度扩展：在开发过程中，可能会出现功能需求过度扩展，影响项目进度。

④市场需求变化: 类似产品快速出现，用户需求可能转移，竞争压力加大。

4.1.2开发环境风险

①技术选型不当: 技术选型可能在实际开发中发现不适合或存在问题，影响开发进度。 ②第三方服务依赖: 地图API、支付API等第三方服务如果出现故障，会直接影响平台功能的正常运作。

③工具兼容性问题: 前后端工具及框架可能存在兼容性问题，影响开发与部署效率。

④开发工具故障：开发工具或环境可能出现问题，导致开发进度中断。

4.1.3人员风险

①对接不恰当：团队成员作为先决条件的任务不能按时完成，影响后续任务进行。

**②技能不足**: 团队成员技术能力不足，需要更多的时间熟悉软件工具和环境，无法按时完成复杂模块的开发。

**③沟通不畅**: 团队内部沟通不畅，可能导致信息不对称，不同开发阶段的协作效率低下，影响项目进展。

**④人员流动**: 项目关键人员的离职或调动，可能导致项目开发进度滞后。

4.1.4组织管理风险

**①项目进度失控**: 缺乏有效的项目管理与进度跟踪，可能导致任务延期。

**②资源分配不当**: 开发过程中资源分配不足或过度，导致某些功能开发滞后或资源浪费。

③管理决策延误：管理层的决策不及时可能会导致项目无法在关键节点上作出调整。

④预算与时间不足：预算不足和由于开发过程耗时增加导致的项目时间超期。

4.2 风险应对策略

针对不同类型的风险，采取相应的应对措施，提前制定计划，确保项目顺利进行：

##### 4.2.1 需求风险应对

* **需求变更控制**：设立需求变更控制流程，确保每次变更经过评估后才进入开发。建立明确的需求边界，减少不必要的变动。
* **需求澄清**：在项目启动时进行充分的需求调研与沟通，确保需求的清晰度。定期与用户代表或需求方讨论，确认需求是否满足。
* **需求扩展管理**：在项目执行过程中，严格控制新需求的引入，优先完成已明确的核心功能。避免在开发后期进行大规模需求扩展。

##### 4.2.2 技术风险应对

* **技术选型验证**：在项目初期通过技术调研和小规模原型验证技术栈的可行性，确保技术选型适用于项目开发需求。
* **第三方服务备选方案**：为重要的第三方服务（如地图、支付）设定备选解决方案，以应对可能的服务中断或更改。
* **兼容性测试**：在开发早期进行多模块的兼容性测试，确保前后端、数据库和第三方服务间的顺利对接，减少集成问题。

##### 4.2.3 人员风险应对

* **技能提升**：在项目启动前或开发早期提供团队培训，确保团队成员掌握所需技术。设立明确的任务分工，便于成员在熟悉领域内高效工作。
* **沟通机制优化**：使用高效的沟通工具（如Slack、Trello）和管理工具（如Jira）加强团队之间的沟通，定期召开会议确保项目进展透明。
* **人员流失预防与应对**：为关键岗位设定备份方案，确保即使人员流动，项目进度仍能保持稳定。制定知识转移计划，确保新成员能快速接手工作。

##### 4.2.4 组织与管理风险应对

* **项目进度控制**：采用敏捷开发模式，定期进行项目评估，确保每个迭代周期内按时完成任务。设立明确的进度跟踪机制，及时发现问题并调整计划。
* **资源管理优化**：定期评估资源分配，确保各模块开发的资源充足且不过度浪费。根据实际开发进展动态调整资源配置，提升效率。
* **快速决策机制**：为项目建立快速决策流程，尤其是关键技术方向和功能实现上，确保能及时做出决策，不影响整体开发。

#### 4.3 风险评估与监控

* **风险评估**：在项目启动前，进行全面的风险识别和评估，量化每类风险的影响程度和可能性，制定相应的应对措施。
* **持续监控**：在项目开发过程中，定期评估项目进展和潜在的风险变动，及时更新风险应对计划。
* **风险调整**：根据项目实际情况和环境变化，动态调整风险应对策略，确保项目保持在可控范围内。