# COMP2913 2024/2025 学年：软件工程项目简介

## 项目描述

## 1 该产品

该项目涉及开发一款用于通勤和/或一次性出行的汽车共享应用程序。

用户必须在系统中注册才能使用该系统并预订行程。

用户可以:

- 进行预订，选择特定行程，并在行程尚未开始的情况下选择取消预订。
- 与提议行程的用户就（时间/接送地点/费用/延误和取消）进行协商。
- 根据自身需求搜索行程——接送地点、人数、日期/时间均可自选。
- 提交行程以供提供（时间/接送地点/费用/最多人数）
- 为行程提供更多选择——行李/婴儿车的车厢大小、无障碍设施。
-

创建一个账户并跟踪自己的预订情况

- 在地图上查看接送点
- 查看有关他们的预订、时长和费用的一些统计数据。
- 定期通勤可享受折扣优惠
- 一旦在拼车服务中固定下来，能够将通勤预订设置为每次更快捷的流程（例如周一/周三/周五）。
- 评价行程并确认行程已发生

就预订事宜提出支持请求。

系统管理员可以:

根据注册用户数量估算收入。

- 查看所有行程（已预订/已取消/可用）的相关数据
- 配置每次预订的费用成本
- 为常客提供折扣，例如每周乘坐 4 次。
- 若用户需要就预订事宜提出问题，请提供支持——开启对话/反馈渠道

## 2 问题领域

熟悉问题领域会对这个项目有所帮助。花些时间研究类似系统是如何运作的。如果合适的话，可以运用一些 SQIRO 调查技术。

### 3 Implementation

具体的功能需求在产品待办事项列表中逐一列出，单独提供。需要注意的一个重要（且希望显而易见）的要点是，您的解决方案不会是一个真正的系统！它需要记录客户详情、客户数据和支付信息。

尽管您的解决方案会受到以下技术限制的约束，但这仍为您在选择工具、框架、库和编程语言方面留有一定的自由度，以便您能够根据团队的兴趣和能力进行选择。

在选择技术时，要实事求是地考虑团队在可用时间内能够完成的工作量，并且要记住，只要项目组织和管理得当，即使实施规模相对较小，也能取得不错的成绩。

## 4 基本架构

### 服务器：

您的解决方案应采用客户端 - 服务器架构，并建议您使用合适的 Web 框架实现三层架构方法。要获得最高分，您的系统应能够处理来自多个客户端的同时连接。

服务器应当与一个存储特定领域详细信息的数据库进行交互：例如，注册用户、行程以及时间和日期。理想情况下，这应当是一个 SQL 或 noSQL 数据库。

将数据以 CSV、JSON 或其他某种格式存储在文件中的解决方案是可以接受的，但会获得较少的分数。

为简化开发、演示和评分工作，请使用嵌入式数据库或无需特殊权限或在系统目录中安装即可运行的数据库服务器。SQLite 是一种合适的嵌入式数据库选择。本项目没有指定使用何种编程语言，由您的团队自行选择一种大家都能熟练使用的语言。

### 客户：

该系统有两个必需的接口：

- 客户界面；
- 一个管理界面。

您有以下选项可供您的客户选择：

C1. 一个桌面、网络或移动界面，客户可通过其注册、支付服务费用以及查看自己的数据。C2. 一个桌面或网络版的管理界面，管理人员可通过其维护和修改账户、更改详情，并执行一系列其他管理任务。

如果您选择网络客户端，请注意其应提供响应式、适合移动设备且易于访问的界面，并且应适当使用合适的 JavaScript 库，而不仅仅是基于 HTML5 和 CSS3。

**测试：**

测试不应是事后才想到的事情。在开发过程中，您的解决方案应始终接受测试，并且在可能的情况下，测试应实现自动化（或许通过 DevOps 的持续集成/持续交付流程）。在这方面投入大量精力是值得的，我们期望在您的文档中看到测试策略以及多种不同类型测试的证据。

**构建与部署：**

您应当尽可能多地将构建、测试和部署流程自动化。虽然 GitHub 内置了许多支持此功能的工具，但根据您的系统性质和技术选择，可能还有其他更合适的工具。如果 GitHub 无法满足您的需求，可以尝试使用 Jenkins（https://www.jenkins.io）。

应当能够使用您所选的构建工具以最少的命令来构建、部署和运行您的解决方案。例如，使用一个命令来构建整个系统，一个命令来运行服务器，以及一个命令来运行桌面应用程序（假设您选择了后者），这会是很好的安排方式。如果使用 Github，那么请研究一下 Github Actions（https://github.com/features/actions）如何能帮助您实现这一点。

请注意，我们并不期望您将解决方案中的服务器端组件部署到云端的真实服务器上！如果您有能力且有兴趣这样做，当然可以将其作为可选功能来实现，但这并非强制要求。出于评分目的，我们要求任何解决方案都能在本地运行。

与测试一样，尽早着手并投入大量精力去做这件事是值得的。构建自动化能为您节省大量时间和精力，所以您应尽早将其设置好。

**5 方法：**

所有团队应遵循的总体方法是简化和缩小版的 Scrum 方法。

该流程由产品负责人（PO）主导，其会为每个团队提供一份产品待办事项列表中初始的、按优先级排序的需求清单（单独发布）。

主要的开发阶段被细分为三个冲刺阶段。在
在每个冲刺阶段的开始，你们应当举行一次冲刺规划会议，在会上确定要实现的产品待办事项列表中的一部分内容，并明确交付这些产品待办事项所需的任务。
列出各项任务，然后将这些任务分配给团队成员。请记住，在每个冲刺阶段结束时，您都应该有一个最小可行产品（MVP）。

在这个简化的 Scrum 版本中，产品负责人（PO）不会出席这些规划会议。因此，团队需要在每次规划会议前检查产品待办事项列表，并在必要时通过面对面交流或 Teams 软件向产品负责人寻求澄清。

在每个冲刺阶段，您应当举行两到四次状态会议。状态会议时间很短，不超过 15 分钟，每个团队成员都要简要说明自上次团队会议以来自己做了什么，接下来到下一次状态会议之前要做什么，以及可能会阻碍进展的问题。对于问题的讨论应推迟到其他会议或在线交流中进行，不一定需要整个团队参与。

您应当以一次冲刺评审会议来结束您的冲刺阶段。该会议应当在向产品负责人简要展示进度（以及您的最小可行产品）之后开始，或者紧接着开始。团队的所有成员都应参加此次会议。

在第 1、2 和 3 个冲刺阶段，冲刺结束时，整个团队要向产品负责人汇报，作为评审流程的一部分。会议将包括对已完成工作的部分演示，回顾上一个冲刺的情况以及各项任务的状态，并讨论即将到来的冲刺。会议还留出时间让产品负责人提问、澄清以及提供反馈。此外，团队成员也可以利用这个机会讨论可能存在的团队问题，以便尽早解决任何冲突。

您应当利用这些反馈来思考产品负责人给出的意见，并找出在您的开发流程中哪些方面可以在下一次冲刺中加以改进。

**Scrum 主管：**

您需要有人担任 Scrum 主管（SM）这一角色。SM 应负责组织并主持所有冲刺规划、冲刺评审和状态会议。SM 应指定另一名团队成员在规划和评审会议期间做记录。记录员负责将记录上传至项目维基（见下文）。状态会议无需正式记录。

SM 应为所有会议（包括状态会议）记录考勤情况，可在项目维基中使用一个或多个页面进行记录。SM 还应代表团队通过电子邮件或其他适当方式调查任何缺勤或任务未按时完成的情况。

请记住，Scrum 主管并非项目负责人！影响项目的决策应由整个团队共同做出。Scrum 主管是协调者，而非决策者。在我们对 Scrum 的改进版本中，Scrum 主管也是开发人员，但由于其职责所在，其开发工作量可少于团队中的其他成员。

建议 Scrum 主管（SM）的角色在团队成员之间轮换，但 Scrum 主管的变更仅允许在冲刺（sprint）之间进行，冲刺期间不允许变更。

**团队成员：**

团队成员应尽可能参加所有会议，如有缺席必须说明原因。团队成员应使用项目的任务跟踪器记录其正在执行的任务的相关信息。其他项目文档应存放在维基中。

团队成员应将 Git 版本控制用于所有编程活动，并应定期将本地所做的更改推送到远程项目仓库，以方便代码审查以及与团队中的其他成员共享新代码。

团队成员可以独自完成任务，也可以与他人合作。您应当考虑采用诸如极限编程（XP）所倡导的“结对编程”方法。这种方法尤其适用于系统中复杂或关键的部分。

## 6 项目工具

您的团队将在模块的 Github 组织内获得一个私有 Github（GH）代码库。您必须使用分配给您的 GH 代码库，并且我们期望在每个冲刺阶段都能看到每个团队成员定期向该代码库提交代码。

对于每个团队成员来说，一项至关重要的首要任务是检查您是否能够成功访问团队的代码库。请务必在第一轮冲刺开始前完成此项操作。

### 版本控制：

所有源代码都必须使用 Git 版本控制系统进行管理。每个团队都应拥有一个共享的代码库，可通过 HTTPS 或 SSH 协议进行访问。

如果您尚未设置 SSH 访问权限，我们建议您在第一轮冲刺开始前完成设置。在第一轮冲刺开始前，您还需要做出的一个重要决定是您偏好的 Git 工作流程。默认情况下，所有团队成员在 GitHub 中都已被授予管理员权限，这意味着所有团队成员都有权将提交推送到主分支。不过，最好避免这种情况，采用“功能分支”工作流程，即每个成员或每对成员在单独的分支上开发每个功能，功能完成后提交合并请求。

需要有人处理合并请求，并负责将功能合并到主分支，解决出现的任何冲突。我们建议指定一名或多名技术负责人来完成这项工作。理想情况下，技术负责人应是团队中经验最丰富的程序员/ Git 用户。

### 问题跟踪器：

每个团队的代码库都将包含一个问题跟踪器。相关文档请访问 https://docs.github.com/en/issues/tracking-your-work-with-issues/about-issues 。

您的团队应当就一些合适的议题标签达成一致，并在跟踪器中记录任何议题之前将其设置好。GitHub 提供了一组合理的默认标签，您可以根据需要添加或删除标签。如果您愿意，还可以对标签进行优先级排序。

除了设置一些标签外，您还应该定义一些里程碑。“第一轮冲刺”、“第二轮冲刺”、“第三轮冲刺”和“最终演示”是必需的，但如果您愿意，也可以定义其他里程碑。

使用追踪器记录分配给每个团队成员的任务。该团队成员应在追踪器中被记录为该问题的“负责人”。请注意，负责人需对任务负责。

在任务完成后关闭该问题。您还应使用跟踪器记录测试期间发现的错误、对功能的修改建议等。

问题使用 GitHub 自己的一种名为 Markdown 的轻量级标记语言编写。花些时间熟悉一下这种语言。更多信息请参阅

https://docs.github.com/en/get-started/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax

GitHub 还提供了项目看板，您可以在其中将问题分组到不同的列中，以看板风格展示当前状态。您可能会发现这是一种很有用的可视化项目整体状态的方式。更多信息请参阅

https://docs. github. com/en/github-ae@latest/issues/organizing-your-work-with-project-boards/managing-project-boards/about-project-boards

**维基：**

维基是您项目的网站，也是所有与项目相关材料的存放地。
未存储在代码库中。我们会为您提供有关我们期望在维基文档中看到的内容的更多信息。

https://docs.github.com/en/communities/documenting-your-project-with-wikis

维基页面使用 GitHub 风格的 Markdown 编写，所以请花些时间熟悉一下语法（见上文链接）。