快速
详细
恰到好处

系统开发即时参考

黄演胜

DataFlyer

本书是一部专为IT技术人员编撰的实用指南，内容详实，便于随时查阅，旨在协助读者迅速掌握应对日常问题的各种关键知识。不同于传统的系统开发教科书，书中聚焦企业实际运营中常见的难题，基于作者25年的一线经验，总结出一系列行之有效的处理思路与解决方案。

如何按时、按预算完成项目并满足用户期望？
开发企业级关键任务系统需要考虑哪些因素？
如何先从信息入手，再进行界面和交互设计？
有哪些技术选项可供考虑以及如何选择？
如何保护您的系统和重要数据？
如何从现有系统迁移和转换数据？
如何获取合适的资源并获得所有批准？
如何在部署前识别和解决潜在的问题？
如何安全地发布系统？

不同类型的读者也能从本书中受益。学生或应届毕业生可以在职业生涯开始之前获得第一手的工作知识；现有的从业人员可以更好地了解他们不熟悉的领域；管理人员可以了解到最新的技术发展趋势。

成千上万的实用信息，很多细节尚未被记录过，所以还未被人工智能所涵盖。简而言之，如何加快项目开发速度，避免错过不同用户的重要需求，如何减低最终产品与客户期望之间的落差，在选择技术方案时最重要的考虑因素是什么，有哪些调试与性能优化技巧？如何避免项目进度被不熟悉的技术所影响，如何更好地与各方合作，如何应对不同类型的挑战，以及最重要的，如何避免项目延误。

另外，本书还涵盖了公共和私人组织采用的标准、实践和程序，以及由此出现的问题，包括所需的全套文档、采购流程、现有数据的迁移、安全控制、隐私保护、用户验收测试管理和生产部署程序。

封面排版设计：黄演胜。

封面图片由 vectorjuice / Freepik 设计并经授权使用。

黄演胜.系统开发即时参考

简介

本书主要是给信息技术人员在日常工作中随手参考与直接使用，涵盖了系统开发不同方面的基本资讯，內容足够详细，以帮助他们解决日常遇到的问题。

由于精简，任何人都可以在短时间内快速阅读并学会当中的知识。遇到任何特殊问题，可以立即参考相应的章节，包括有什么事项要检查，步骤是什么，有哪些地方需要多加注意，以及应当怎样做。

本书建基于笔者在公共和私营部门超过二十多年的软件与系统开发经验，大多已被大公司或政府采用多年。读者会发现有些程序可能较为适合有特定规模的项目，但笔者的初衷是先让大家参考，然后再根据自己的业务或发展需要，再定制、修改甚至是扩展相关程序。

目录

引言

本书主要是给信息技术人员在日常工作中随手参考与直接使用，内容涵盖了系统开发不同方面的基本资讯，并足够详细，以协助解决日常遇到的问题。

项目规划

项目需要按时完成，满足用户期望并在预算之内。它涉及许多不同的参与方和任务。在政府这样的大型机构工作时，需要遵循一些标准和规则。这是一项真正具有挑战性的任务，如果没有适当的规划和管理，很容易出现问题或失控。

本章列出了项目的基本任务、管理措施、标准和常见问题，还会提及在不同项目中经常出现的问题，并介绍处理这些问题的一些行业实践经验。此外，本章也涵盖了需要做什么、如何做、如何预防以及遇到问题时如何应对。

互动设计

系统开发涉及静态和动态内容，需要适当的设计。用户通过界面与系统交互，并期望获得操作的帮助和反馈。

本章旨在解释如何呈现信息，如何使系统使用起来更舒适，以及如何使操作更便捷，包括文本、音频、视频、动画元素甚至人工智能设备的处理，从而使整体体验更加友好。

架构与编码

计算机系统最重要、最关键的任务是系统设计和实现。它涉及收集用户需求、通过确定基本架构和组件来制定初步的概念设计，然后参考模式、框架和一些第三方中间件，以选择可用的解决方案、合适的语言或工具，以及实现的细节。此外，测试和质量保证，对于确保最终产品能否满足用户期望也至关重要。

本章列出几乎所有主要的系统设计和开发技术以供参考和评估，包括规划、设计、实现和维护相关的。读者将认识到学校里可能还没来得及教授的开发技术、进入新行业需要了解的技术知识、哪些技术适用于哪些情况，并了解最新的行业发展趋势，以及开发建议、调试和性能调优技巧。

数据迁移和转换

大部分系统开发是对现有计算机系统的改造。许多现有数据需要迁移到新系统，并且涉及某种类型的数据转换。 由于现有系统和新系统的数据结构不同，因此需要进行数据映射。

本章的目的是列出基本任务，从确定要迁移的数据，如何在新系统中映射和重新创建它们，如何检查数据是否已正确迁移和转换。 它涵盖了做什么、怎么做和如何验证的程序。

安全与隐私保护

如今大多数计算机系统都基于网络环境，可以通过互联网或内网访问。它们很容易受到来自外部或内部的网络攻击。正因如此，安全性在系统开发中至关重要。如果没有周全的考虑和充分的控制，数据很容易被恶意篡改或窃取。当大多数交易都涉及隐私时，无论是个人信息还是金钱，其后果都可能是致命的。

本章旨在列出从规划、设计到实施和维护阶段，IT 安全和隐私保护方面的基本考虑因素。此外，本章也涵盖了系统管理和相关代码开发的实践，必须做的事情、如何检测、预防和应对。

采购计划

计算机系统需要市场上的硬件和软件才能实现，而获取这些硬件和软件则需要采购。与在零售店购买个人产品只需简单的付款即可不同，公司和政府部门需要遵循规则和操作流程。

除了需要获得多项批准才能最终购买外，公共机构还需遵守世界贸易组织的政府采购协议。

本章旨在列出常见的 IT 采购实践，包括流程和需要完成的任务，以帮助员工更好地理解整个流程并做好充分的准备。

用户验收测试计划

系统开发是在收集与理解用户的需求后、在商定并记录于系统和功能规范的基础上进行。 在大多数情况下，最终产品与用户的期望有所不同，并且由于误解、发现新情况或其他原因而出现问题或错误。

用户验收测试是为了找出开发和用户团队之间认知上的差距，并确保在生产部署之前识别与解决所有潜在问题。为此，必须仔细规划并确保所有场景都经过测试。

本书的目的是列出所有需要完成的任务，包括所有的详细步骤，以帮助工作人员了解整个过程并完成手头的工作。

系统推出程序

推出新系统并非是一件简单的事情，特别是对于具有一定规模的项目而言。

许多系统都是由各种硬件设备与不同的软件构件或服务所构建，开发人员必须与不同领域、范畴或不同级别的人员进行协调与合作，他们很可能来自外部，甚至是世界其他地区的供应商。

系统推出的过程有太多的细节需要关心、检查和验证；有时相当复杂，待办事项的清单可以非常长，所以工作人员很容易错过或遗漏一些事项与细节。本书旨在协助读者一步一步处理，并建议适当的方式以完成手上的工作。

III. 普遍现象

i. 旧系统问题

- 遇到大量漏洞或问题
- 现有系统难以使用，界面不友好
- 系统限制，因此无法灵活应对特定的日常操作
- 先前项目仓促完成，时间和资源分配不足
- 仅解决了最严重的问题，其余问题均采用临时解决方案或未解决
- 由于招标仅授予了相对较好的供应商，先前供应商缺乏足够的知识来处理这些问题
- 由于时间紧迫、压力巨大，且无法拿出令人满意的解决方案，先前员工感到沮丧甚至被解雇
- 知识传递有限，通常通过口头交流进行，无论是技术还是业务操作
- 文档质量差或信息不及时更新，缺失或不准确

ii. 常见问题

- 大型项目通常无法按原计划完成，存在诸多质量问题，且耗费额外的人力和财力。
- 项目要么工期延长，要么调整范围，最坏的情况是终止。
- 对于分阶段进行的项目，通常由于第一阶段的延期，导致后续阶段在第一阶段尚未完成的情况下启动。后续阶段任务堆积过多，导致项目延误
- 无论公司规模大小、员工毕业于哪里、性别或种族如何，情况都大体相同。
- 只有经验丰富、技术成熟且业务运作熟悉的简单项目才能获得更高的成功率。
- 投标复杂度被低估，所需任务过多或过于复杂，时间却不足。
- 招标文件为后期提出额外需求留下了太多余地。
- 由于同时采用多种方法(例如云端和本地环境、瀑布加敏捷开发)导致工作量过大。
- 时间浪费在琐碎问题和工作上，而没有专注于核心和主要工作。
- 时间浪费在太多实际用途不大的会议上，尤其是日常会议。
- 缺乏管理和控制工作以在限期前完成的意识。
- 时间浪费在互相批评上，而不是专注于共同寻找解决方案。
- 最高权力的利益相关者介入太晚，推翻了现有工作。
- 用户直到系统可视化或尝试后才知道自己真正想要什么。
- 用户需求变更过多且过于频繁。
- 用户与供应商之间就新需求是否在范围内存在分歧
- 坚持现有的工作方法，即使发现问题也拒绝更改或报告
- 遇到技术难题，解决方案需要时间摸索和尝试
- 技术采用不当，缺乏充分评估，且变革为时已晚
- 无法招聘到合适的员工，只能找到相对较好的员工
- 监控方为自身而非项目利益而夸大对供应商的批评与监管过严
- 供应商害怕负面评价而掩盖问题，即使与客户共同讨论能找到更好的解决方案
- 项目管理办公室任命的人员不合适，无法有效运作。
- 没有需求细节的记录，导致错误或缺失
- 没有架构师检查不同子系统和组件的整体系统设计
- 缺乏成功项目管理所需的软硬技能：包括知识、人际关系、协调能力、谈判能力、领导力、团队建设、乐观和毅力。
- 有新成员加入，但需要时间跟进，无法解决眼前问题

ii. 框架

用于改进流程的广泛结构或模型，无需具体的分步说明

- **能力成熟度模型集成 (CMMI)**

  由软件工程研究所 (SEI) 开发的流程改进框架，主要用于软件开发流程，为组织提供了一个可遵循的模型，其成熟度级别从 1(初始)到 5(优化)，旨在提升绩效和质量：

  第 1 级初始： 流程不可预测、控制不佳且反应迟钝
  第 2 级可重复：流程的特点是项目按照政策进行规划、执行、衡量和控制
  第 3 级已定义：流程已充分记录、标准化，并已集成到组织的标准流程中
  第 4 级已管理：流程得到量化管理，组织使用指标来管理和控制流程
  第 5 级优化： 基于对常见差异原因的量化理解，持续改进流程

- **ITIL(信息技术基础架构库)**

  它为 IT 服务管理 (ITSM) 提供了一套全面的最佳实践，以协调 IT 服务与业务需求相结合，并改进服务交付。它涵盖以下服务管理流程：

  - 服务战略： 定义服务交付方式
  - 服务设计： 设计服务和流程以支持
  - 服务转型： 在服务部署期间管理变更、风险和质量保证
  - 服务运营： 确保服务有效高效地交付
  - 持续服务改进：持续改进服务和流程。

  框架细节在五份核心出版物中提及，并附有指南和最佳实践。

- **TOGAF(开放组织架构框架)**

  广泛使用的企业架构框架，提供全面的企业信息架构设计、规划、实施和管理方法，帮助企业将 IT 战略与业务目标保持一致：

  - **架构开发方法 (ADM)** 逐步开发企业架构的方法
  - **企业连续体** 用于指导新架构开发的架构资产库
  - **架构内容框架** 提供用于开发架构可交付成果的详细模型
  - **TOGAF 参考模型** 可用作开发特定架构模板的标准化模型

...

这些框架相互补充，可以整合在一起，但各国的采用差异很大，并且会产生高昂的成本，尤其是在培训、认证、流程实施、招募认证专业人员或外部审计师方面。

iv. 其他方法

- PRINCE2(受控环境下的项目)

  结构化方法，确保项目得到周密的规划、执行和控制。

  - 明确角色和职责：确保问责制和效率
  - 持续的业务论证：确保项目可行且有益
  - 注重产品：交付符合项目目标的有形可用产品
  - 因地制宜：根据具体需求和情况调整方法
  - 分阶段管理：监控进度并控制风险
  - 例外情况管理：在规定的容差范围内做出决策，仅在出现例外情况时才要求高层管理人员参与
  - 从经验中学习：运用以往项目的经验教训进行改进
- 政府资讯科技总监办公室大型项目实施
  - 为香港特区政府大型 IT 项目的独特需求量身定制
  - 提供范围界定、规划和风险管理的详细指南，包括复杂性评估、利益相关者分析、定义项目边界和可交付成果、风险缓解和外包管理

观察发现

- 遵循标准和方法，如果使用和参考得当，可以防止项目出现常见问题，但由于每天都会出现新问题，因此无法保证一定有效。
- 统计数据显示采用敏捷方法论的有效性似因项目而异，成败似与管理者自身息息相关。
- 如果缺乏适当的管理，已知问题仍然存在的情况很常见。
- 与其单纯地遵循流程，不如专注于解决问题并实现目标。
- 项目成功可能需要调整、运用或结合不同方法。解决方案有无限可能，取决于自身创新
- 不要迷信任何参考资料，除非它真的有帮助，能够解决手头的问题。

V. 需求收集

- 尽早让所有利益相关者参与进来，特别是做出最终决策的高层管理人员
- 除了用户代表外，邀请执行日常操作的前端人员，因为他们了解每个细节
- 确保覆盖不同级别和用户组
- 会议邀请应尽早发出
- 与用户核实现有问题、错误、限制、评论、疑虑和期望
- 检查错误日志，查找需要处理的隐藏问题
- 检查迄今为止的投诉、反馈或报告，可以通过员工访谈进行
- 现场观察，查找未提及的特定问题
- 对不同组或级别的特定用户进行角色扮演，以检查是否有任何遗漏
- 进行焦点小组测试，以了解是否需要考虑任何未意识到的行为偏好
- 使用电子邮件或电话再次确认细节
- 不同用户群体的需求可能存在矛盾，需要讨论
- 原型设计（例如模拟屏幕）更易于沟通
- 对于较大的用户群体，如果时间紧迫或需要延长时间，请让他们先进行整合以节省时间
- 会议期间，请征求录音许可，仅用于再次确认细节
- 如果允许，考虑使用人工智能技术在会议期间将语音转换为文本
- 详细记录需求，并发送给用户进行双重验证和确认
- 设定需求收集截止日期，并提前提醒用户
- 除非需求至关重要或影响系统正常运行，否则不应添加可能影响系统正常运行的额外需求严重延误项目进度
- 截止日期之后提出的需求，尤其是在关键需求的情况下，可能会在实施过程中被重新考虑。
- 如果用户需求在项目范围、时间范围内且只需投入少量精力，则应积极响应。
- 向用户解释过多的需求可能会延误整个项目。

...

VI. 开发策略

- 项目授予或确认后，系统分析师或程序员可以开始研究、学习和测试可能用到的核心技术。
- 如果使用 Spring 或 Spring Boot 等标准框架，则可以开始基础编码。
- 可以先开发基本功能，如系统代码/参数、账户、日志记录、审计跟踪。
- 在实施之前进行核心代码试点测试以避免对项目性能等产生负面影响。
- 尽可能复用在其他项目中开发的特性或功能，例如微服务
- 尽可能地重复使用现有的数据库结构或配置文件
- 多个项目需要开发相同的功能，并指派相同的人员负责
- 需要架构师检查整体技术问题，并与每个团队负责人进行充分沟通
- ...

VII. 设计考虑

- 从用户的角度思考，而不是开发人员或设计师的角度
- 最大限度地减少用户的学习曲线
- 减轻用户的大脑负担
- 界面应易于理解且使用方便
- 常用操作
  - 考虑使用双因素身份验证，例如通过电子邮件发送验证码
  - 防止用户同时多次登录，保护账户安全
  - 登录后，在收件箱中显示未完成的待办任务
  - 提醒用户任务接近截止日期
  - 可自行设置任务开始、3/4、中间、1/4 和截止日期的提醒时间
  - 将用户日常使用、最常用和最重要的功能放在最显眼的位置
  - 确保所有用例都得到处理，不会遗漏
  - 考虑用户最高效、最有效的操作
  - 保持简洁，避免步骤过于复杂
  - 简化各方之间的工作流程，避免遗漏重要环节
  - ...

- 技术关注点
  - 响应不同的设备、平台和浏览器使用情况，尤其是分辨率
  - 加载、响应时间和性能
  - 最大事务数
  - 最大用户数支持
  - 并发性：防止死锁，且多个用户访问同一记录时不会覆盖原有记录
  - 多语言支持
  - ...

VIII. 实施考虑

- 架构设计，确定需要哪些组件、它们如何协同工作和通信
- 以框图列出所有网络节点及其内部的安全控制，例如防火墙或代理
- 根据存储的数据和 5 年容量预测来计算系统规模
- 确定所需的 CPU 核心、内存和存储空间
- 决定使用本地服务器、私有云/公有云还是混合云。混合云会增加项目复杂性，而公有云需要保护企业机密或敏感数据。
- 确定版本控制、存储库管理、部署和推广工具
- 使用最常用的技术，并充分利用现有的支持、解决方案和人才
- 使用稳定但非最新版本以避免出现问题在市场上无解决方案的情况
- 避免使用与特定产品、组织或员工绑定的复杂技术
- 在确定使用哪种工具之前，索取功能实现代码示例
- ...

IX. 任务分配

- 根据员工的强项和弱项分配任务
- 将任务分配给有经验、兴趣或愿意接受挑战的人员
- 将工作和任务与员工的职业目标相结合，以获得最佳效果
- 为确保项目质量，请勿将同一员工同时分配到两个处于关键阶段且工作量很大的项目
- 提供设计模式、框架或最佳实践的指导，部分指南可在网上获取
- 对照需求、系统和功能规范，确定详细需求
- 在要点或程序规范中列出重要细节
- 简要解释需求，并提醒关键领域，例如企业功能、最大并发用户数、性能和响应时间
- 提醒员工不要遗漏重要功能、不同场景和导航路径

XIV. 进度控制

- 根据交付成果的提交截止日期，按顺序列出待完成的任务
- 标记每个任务，并附上详细的子任务或工作分解，以便更好地监控和控制
- 检查所有任务的优先级和依赖关系，并在必要时重新调整顺序，以便最佳安排加快任务完成进度，即关键路径
- 应突出显示里程碑，因为将向供应商支付款项，例如 SA&D、UAT 或推出
- 列出目标开始/结束日期，然后在最后一行列出实际开始/结束日期，并在最后一行注明团队成员姓名，特别关注对项目影响最大的关键任务。
- 由项目经理或团队负责人填写(自上而下)，也可以要求团队成员根据以往类似工作经验自行填写估算(自下而上)，并由系统分析员进行调整。
- 为项目任务留出充足的时间。
- 缓冲时间涵盖申请、反馈、批准、拒绝、修改、重新提交以及防火墙设置等生效所需天数。
- 留出 1 到 2 周的用户意见征集时间，但要避免用户无限循环，导致项目延误，尤其是在实施阶段，需求收集阶段包含过多细节。
- ...

XV. 质量控制

- 尽早开始研究并获取相关技能清单
- 从用户角度思考哪些方面可以改进以及如何改进
- 确保涵盖所有用户场景和导航路径
- 考虑高效且有效的实施方案
- 参考现有的最佳实践和实施方案
- 必要时，向主管、同事、技术论坛(本地或国外)寻求帮助
- 利用市场上现有的测试工具及早发现问题
- 同行评审，尤其是具有类似经验的同行评审，可作为未来的备用资源
- 尽早测试主要技术的性能或特殊需求
- 记录系统架构、功能规范和网络图以供检查
- 制作清单或流程，以便于后续跟进，确保项目无遗漏
- 制作招标和评估指南，以确保计划的特性和功能可用
- 记录有助于新员工了解项目并进行后续跟进
- 文档提供足够的详细信息以供快速参考
- ...

XVII. 风险管理

- 不时评估可能影响项目进度、成本和质量的潜在风险
- 评估发生的可能性，即高、中、低概率
- 评估影响的严重程度，即关键、高或低
- 通过矩阵登记并报告风险，立即通知相关方，而不是等待
- 与所有责任方讨论，寻求如何使项目重回正轨的建议
    - 管理层不应仅凭自己的假设行事，而应咨询员工的意见
- 让做最终决策的高层或熟悉运营的一线员工更早地参与
- 倾向持续渐进式的变更，限制项目范围而不是进行大规模变更
- ...

III. 识别用户

用户设计时应考虑以下因素：

- 设计以用户需求和便捷性为中心，而非设计师或开发者自己
- 在线提供的用户信息可能不真实，例如年龄或性别
- 用户的理解或解读可能因背景或经验而异
- 允许用户自定义偏好，例如语言、沟通渠道或节奏
- 考虑国际用户的文化差异
- 了解目标用户群体的禁忌和习俗，例如宗教信仰
- 简化术语和操作，并为儿童和老年人提供指导或帮助
- 提高便利性，减轻用户的心理负担，无需记住命令或代码
- 用户无需学习太多知识即可使用
- 用户操作基于以往经验，可能会忽略重复的细节
- 不要假设用户有时间阅读手册或详细检查所有内容
- 不要要求用户执行系统可以替他们完成的操作
- 不同类型的用户：
  - 年龄，例如儿童、青少年、成年人、退休人士
  - 性别
  - 语言(一种或多种)
  - ...

IV. 需求收集

- 如有招标要求，请查阅
- 通过以下方式深入核实需求：
  - 主要用户访谈
  - 远程及不同地点的用户调研
  - 现场考察并观察实际工作和使用情况
  - 焦点小组访谈，尤其针对前端操作人员
  - 专门测试，了解用户对不同设计的偏好和反应
- 检查每个潜在使用场景
- 从头到尾走一遍整个操作流程，检查是否有遗漏
- 生成需求规范并与用户确认

* 注意，用户可能仍有遗漏或忘记提及需要处理的事项

V. 产品分析

- 通过评估现有产品，确定所需的特性或功能：
  - 需要保留的现有特性或功能
  - 需要解决的问题，例如：局限或不便之处
  - 需要修复的漏洞
  - 需要进一步增强的功能，以提供更多或更好的功能
  - 简化或合并现有功能、流程或操作
  - 替换或删除不必要的特性或功能
  - 引入新功能或功能以满足新需求
- 对于面向私人市场开发的产品
  - 参考市场趋势，通过产品比较来检查优势和劣势
  - ...

IX. 信息设计

- 呈现信息，向用户传达意义和体验
- 信息以目标、主题和逻辑流程呈现
- 涉及叙事架构，即构建和组织数据
- 重要注意事项
  - 信息应准确清晰，并核实来自不同可靠来源的事实
  - 呈现方式应清晰、简洁且具有视觉吸引力
  - 由简入繁，由大到小
  - 内容呈现是目的，导航是手段
  - 通过视觉提示(颜色或线条)、搜索或引导帮助查找信息
  - 通过事前简报和事后总结来强化记忆
- 流程
  i. 数据收集
  ii. 按类型和类别对数据进行分组，例如时间、地点或功能
  iii. 按重要性和使用频率对数据进行排序
  iv. 将数据映射到不同的用途并进行分组
  v. 确定呈现方式和顺序
    - 信息按顺序逐步传递给用户
    - 信息分组，让用户选择并深入探索
    - 根据收集到的或预期的偏好，仅向用户提供相关信息

  ...

- 信息艺术
  - 信息可以以文本、图形、声音和视频的形式呈现，例如标签、菜单、表格、导航平面、图标、地图、信息图或动画。
  - 信息图设计指南
    - 图形是信息呈现的常用方法。
      - 分类

        ...

- 视觉语法
  - 结构：通过模块的一致应用来强化
  - 可预测性：以相似的模式和方式响应
  - 效率：实现卓越的生产经济效益
  - 清晰的焦点：有限的模块展现空间逻辑
  - 灵活性：应对不同情况
  - 可维护性和可扩展性
  - 统一性：将元素协调一致，以实现沟通目标
  - 完整性：通过自然呈现的形式聚焦目标
  - 可读性：将内容划分为易于管理的子集
  - 控制性：预测用户感兴趣的区域，并通过构图简化导航
  - 接近性：与邻近元素紧密关联
  - 相似性：基本视觉特征共享性更强
  - 连续性：倾向于连续、不间断的轮廓，即使有中断，也应将闭合视为完整
  - ...

- 导航路径
  - 以自由探索或按顺序排列的方式排列
    线性和顺序的逐步操作可以确保不会遗漏任何步骤
  - 允许自由探索模式，即使用户无法查看特定信息顺序
  - 按不同目的或功能，分类或层级组织
  - 避免菜单导航层级过多
- 实用功能
  - 快捷键，快速访问
  - 使用通配符和筛选器搜索，更轻松快捷地查找信息
  - 站点地图和索引，快速访问特定信息
  - 书签或历史记录功能，方便返回访问位置
- 清晰指示
  - 显示当前位置，防止迷路
  - 显示所有导航选项，例如：下一个、上一个、退出和返回

图形设计

■ 设计与主题相符	1	2	3	4	5
■ 我喜欢产品使用的布局设计	1	2	3	4	5
■ 界面色彩对比鲜明	1	2	3	4	5
■ 文字显示清晰易读	1	2	3	4	5

界面元素

■ 界面美观且具有吸引力	1	2	3	4	5
■ 界面元素在外观和功能上保持一致	1	2	3	4	5
■ 导航控件与应用程序相符或相关	1	2	3	4	5
■ 导航对象清晰、明确且易于理解	1	2	3	4	5

导航与交互

■ 操作逻辑清晰易懂	1	2	3	4	5
■ 所有功能标识清晰	1	2	3	4	5
■ 提供足够的导航提示	1	2	3	4	5
■ 当前位置清晰显示	1	2	3	4	5
■ 返回主菜单的方式清晰易懂	1	2	3	4	5
■ 不同屏幕之间的导航方式可预测	1	2	3	4	5
■ 我有时不知道接下来该用这款软件做什么	1	2	3	4	5

- 可靠性
  采用松耦合设计，通过容错和故障转移来保持系统可用性，以最大限度地减少依赖性并减少对外部变化的影响
- 安全性
  防御威胁和漏洞，确保数据完整性和机密性
- 可维护性
  易于更新、更改或扩展
- 互操作性
  标准协议和格式，方便沟通和集成
- 可用性
  易于理解，操作路径最短
- 有效性
  最小投入，最大产出

- 企业架构
  将组织的业务流程、信息系统和技术基础设施与总体目标和宗旨相协调的战略框架
  - 特点
    - 性能：高可用性、快速加载和响应时间，可支持全球 24 小时关键业务运营
    - 可扩展性：能够处理随着业务增长而增加的工作负载
    - 灵活性：适应业务需求变化，例如新流程和法规
    - 安全性：保护敏感数据并确保正常运行的措施
    - 可维护性：易于增强或修复发现的问题
    - ...
  - 常见实施方案
    - 冗余：双活多区域部署，确保服务连续性
    - 负载均衡：跨地理分散服务器的流量路由和分配
    - 自动扩展：通过服务器集群或云服务进行动态资源分配
    - 容错：最小化或隔离错误，防止服务中断
    - 灾难恢复：故障转移机制，用于在系统故障后恢复服务
    - ...

II. 关键企业和人物

关注关键企业和人物，以便及时了解最新的技术发展，包括他们使用的社交媒体，例如 Facebook、Instagram、X、LinkedIn 或微博。

- 关键企业
  - 微软
    - 微软学习平台
      https://learn.microsoft.com/en-us/
    - 微软活动目录
      https://events.microsoft.com/
    - 微软研究院
      https://www.microsoft.com/en-us/research/
  - 亚马逊
    - 亚马逊网络服务
      https://aws.amazon.com/free/
    - 云计算培训与课程
      https://aws.amazon.com/training/?nc1=h_ls
    - AWS re:Invent
      https://reinvent.awsevents.com/
  - 谷歌
    - Android 开发者
      https://developer.android.google.cn/
    - Google for Developers
      https://developers.google.com/
    - ...

- 关键人物

	名字	贡献
1	Adrian Holovaty	Django
2	Andrew Ng 吴恩达	人工智能教育的领导者
3	Ben Treynor	站点可靠性工程
4	Bill Gates	Microsoft 创始人
5	Bobby Woolf	企业集成模式
6	Brian Behlendorf	Apache HTTP 服务器
7	Bruce Schneier	安全技术专家和作家
8	Craig McClanahan	Apache Struts
9	David Heinemeier Hansson	Ruby on Rails
10	David O. Sacks	PayPal COO
11	David Silver	AlphaGo
12	Demis Hassabis	AlphaGo
13	Diane Greene	Google Cloud
14	Doug Cutting	Apache Hadoop/ Lucene

...

III. 学习资源

除了关键企业和人物之外，还有其他资源可以学习最新技术：

- 大学开放课程
  - 麻省理工学院开放课程
    https://ocw.mit.edu/
  - EDX
    https://www.edx.org/
  - Coursera
    https://www.coursera.org/
- 培训或活动：
  - GitHub Universe
    https://githubuniverse.com/
  - Tutorials Point
    https://www.tutorialspoint.com/index.htm
  - Frontend Masters Live
    https://frontendmasters.com/
  - Esri MOOC 培训 (GIS)
    https://www.esri.com/training/mooc/
- 博客、论坛、社交媒体和网站
  - BrightTALK
    https://www.brighttalk.com
  - TechInsider
    https://www.facebook.com/techinsider
  - TechCrunch
    https://www.facebook.com/techcrunch

  ...

- Web 服务器
  - Nginx
    - 最受欢迎的 Web 服务器，以其高性能和高效率而闻名
    - 能够处理大量并发连接
    - 广泛用于提供静态内容和反向代理
    - 支持块服务器配置，可托管多个域或应用程序
    - 支持 SSL 终止、HTTP 基本身份验证、基于动态 IP 的访问控制、内容缓存和实时性能监控
    - OpenResty：基于 Nginx 非阻塞 I/O 模型的高性能 Web 平台，可处理高并发，并集成各种 Lua 库和第三方模块，允许开发人员构建可扩展且高效的 Web 应用程序、Web 服务和动态网关
  - Apache HTTP
    - 最古老、最可靠的 Web 服务器
    - 高度灵活，具有自动索引、会话跟踪、URL 重写、Gzip 压缩和虚拟主机功能
    - 广泛的模块支持，例如用于动态 URL 请求的 mod_rewrite

  ...

- 语言特性
  - Java Lambda 表达式
    - 简化函数式接口的实现
    - 表示无名匿名函数，允许将代码作为参数传递，从而实现函数式编程
  - 反射 Reflection
    - 在运行时检查并修改自身结构，以分析或操作类、方法、字段和构造函数。
    - 确定对象上哪些成员字段和方法可用的过程
    - 通常与自省（Introspection）结合使用，以确定对象的哪些方法可供其他对象访问，例如变量的 getter 和 setter。
    - 在运行时动态加载类。
    - 检查或修改对象属性。
    - 动态创建类实例或调用方法。
    - Spring 和 Hibernate 等框架使用反射来处理依赖注入和对象关系映射。
  - 注解 Annotations
    - 提供代码元数据，用于向编译器或运行时环境传递信息
    - 内置注解：@Override、@Deprecated、@SuppressWarnings
    …

- C#
  - 常用于开发 Windows 应用程序和游戏。
  - 受 Java 影响，并具有许多相似之处，例如语法、面向对象概念、多线程、自动垃圾收集以及用于处理文件、网络和数据结构的丰富库。
  - 支持语言集成查询 (LINQ)，它提供了强大的数据源查询功能，例如：对象、XML 和 SQL
  - 包含属性、事件和委托等功能
- Visual Basic
  - 在许多历史悠久的遗留系统中使用
  - 简单易读的语法
  - 通过拖放式设计工具快速开发基于 Windows 的事件驱动应用程序，这些工具可响应用户事件，例如按键和鼠标操作
  - 与 Visual Studio 完美集成
- Python
  - 简洁易学的语法
  - 无需显式声明变量，变量类型在运行时确定
  - 平台无关，安装解释器即可在不同操作系统上运行
  - 自动和手动内存管理
  - 丰富的标准库，用于 Web 开发、正则表达式、I/O 等
  - 丰富的第三方库生态系统：
    - Pandas：使用数据结构 Series 和 Data Frame 进行数据操作和分析
    - NumPy：支持大型多维数组的数值计算

    ...

- 无代码开发
  - 允许普通非技术用户无需编写任何代码即可构建应用程序
  - 使用模板和拖放式预置组件快速创建应用程序
  - 创建复杂的工作流以自动化流程和管理任务
  - 通常内置集成常用服务或 API，以连接其他工具和系统
  - 促进快速原型设计以验证想法并进行调整
  - 产品：Microsoft Power Automate、Webflow、Zapier(自动化)、Airtable(数据管理)、Comet 和 Makerpad
  - 避免使用对用户和开发者使用不同术语的工具来实现相同的功能，或错误地重新定义常见的技术术语，以免造成混淆和误解
  - 非技术用户有时会被使用误导性术语的工具定义为“公民开发者”

- Vibe 编码
  - 借助人工智能进行编程
  - 详细描述手头需要解决的问题，并将其作为大型语言模型 (LLM) 的提示，代码将自动生成并调试
  - 非常适合快速原型设计，甚至可以通过可共享的 URL 进行即时部署
  - 工具：Cursor、Windsurf 或在线集成开发环境 (IDE)，例如 Replit

VI. JavaScript 库

- 异步 JavaScript (Ajax)：
  - 能够处理多个任务，无需等待一个任务完成后再启动另一个任务
  - 请求完成后，使用回调函数处理返回响应
- jQuery
  - 快速、小巧且功能丰富的 JavaScript 库
  - 简化 HTML 文档遍历和操作、事件处理、动画和 Ajax
  - 使 JavaScript 更易于使用
- Dojo
  - 用于构建跨平台、基于 JavaScript/Ajax 应用程序的开 JavaScript 工具包
  - 提供丰富的 DOM 操作、事件处理等实用程序
- Bootstrap：
  - 用于开发响应式和移动优先网站的前端框架
  - 提供用于表单、按钮、导航等的 HTML、CSS 和 JavaScript 组件。

以上框架使用较少，因为其他框架库更高效、更强大、更灵活：

- Node.js
  - 由 Ryan Dahl 于 2009 年基于 V8 JavaScript 引擎开发
  - 跨平台开源运行时环境，可在服务器上运行 JavaScript 侧边栏
  - 主要特性：
    - 单线程事件循环，高性能处理并发连接
    - 异步和事件驱动：非阻塞 I/O 操作
    - Node 包管理器 (NPM)，包含开源库和包
    - 适用于构建快速可扩展的 Web 服务器、创建 RESTful 和微服务 API 服务、实时应用程序，甚至创建命令行工具 (CLI)。

- Express.js
  - 极简快速的 Node.js 框架
  - 常用于构建 API 和 Web 服务器。

VII. 设计原则

- 通用编码
  - 代码可读性
    - 使用有意义的变量和函数名：描述性强并反映目的
    - 一致的格式：一致的缩进、间距和换行
    - 注释和文档：解释复杂代码块的目的
  - 简洁性
    - 简单直接：避免不必要的复杂性和过度工程
    - 模块化设计：将代码分解为小的、易于管理的函数或模块
  - 可复用
    - 不重复自己 (DRY)：避免代码重复，但函数和模块可复用
    - 使用库和框架：节省时间，无需从头开始
  - 性能优化
    - 优化代码：编写高效且性能良好的代码
    - 分析和基准测试：使用工具识别性能瓶颈
  - 可靠性
    - 始终可用：无论用户数量多少，服务始终可访问
    - 回退：从错误或灾难中恢复或隔离
  - ...

- 以用户为中心的设计
  - 用户需求、愿望和目标
  - 最受注意和关注的领域
  - 待完成的任务
  - 现有的操作和导航习惯
  - 首选的交互方式
  - 对事物运作方式的假设
  - 提升用户使用动机
  - 尽量减少达成目标所需的努力
  - 简化导航
  - 消除障碍
  - 提供提示
  - 提供反馈
  - 在需要时提供帮助和协助
  - 清晰明确的导航路径
  - ...

- 帕累托原则
  - 确定对大多数结果贡献最大的任务或元素，以创建更高效、用户友好且易于维护的系统
  - 专注于用户 80% 时间会使用的前 20% 的功能
  - 识别并优化消耗 80% 系统资源的 20% 代码或流程
  - 简化并改进 80% 用户最常交互的用户界面元素
  - 解决导致 80% 错误和维护问题的 20% 代码
  - 将开发资源分配给影响最大的任务和功能

- 单例设计模式
  - 它将类的实例化限制为一个对象，以协调整个系统的操作。
  - 系统仅使用一个对象运行效率更高，或者将实例化限制为一定数量的对象，例如：使用单网关对象管理与旧系统的连接
  - 实现：
    - 类创建单个实例
    - 同一实例的全局方法：返回实例引用的全局类方法
    - 私有构造函数：将类构造函数声明为私有，以防止创建新实例
  - 获取实例的属性方法：
    - 仅首次调用创建实例
    - 锁定以防止其他调用创建实例
    - 其他调用返回同一实例

- 观察者设计模式
  - 观察者(视图)是向用户显示数据的对象
  - 主体(模型)是从相关领域建模的业务抽象
  - 观察者注册感兴趣的主体，并在不再感兴趣的主体时注销注册
  - ...

- 工厂设计模式
  - 处理对象实例化 new 和初始化(构造函数)，在创建者类和被创建类之间建立显式关联
  - 使用工厂方法创建对象，无需指定要创建的对象的具体类
  - ...

- 实现 EIP 的常用工具
  - Apache Camel：实现 EIP 的开源集成框架
  - Spring Integration:该框架在 Spring 生态系统中提供对 EIP 的支持
  - Mule ESB：支持 EIP 的企业服务总线
  - Red Hat Fuse：实现 EIP 的集成平台

- 面向服务架构
  - 将软件设计为松散耦合服务集合的架构方法
  - 关键概念
    - 服务：执行特定任务或业务流程的独立功能单元，松散耦合、可重用、独立部署和管理。
    - 服务提供者：创建、维护和发布服务。
    - 服务消费者：根据需求发现、调用和使用服务。
    - ...
  - 关键组件
    - Web 服务：基于简单对象访问协议 (SOAP) 和表述性状态转移 (REST) 标准的实现。
    - 服务总线：促进服务间通信的中介，通常以企业服务总线 (ESB) 的形式实现。
    - 业务流程执行语言 (BPEL)：用于编排和管理跨多个服务的业务流程。
    - ...

- 函数式编程
  - 将计算视为数学函数的求值。
  - 避免更改状态和可变数据。
  - 使用纯函数、不变性和声明式编程。使代码更可预测、更易理解、更易测试
  - 关键概念
    - 纯函数：对于相同的输入，始终产生相同的输出，且没有副作用
    - 数据不变性：函数一旦创建便不可更改，并使用新的数据结构进行修改，以防止意外的副作用
    - 一等函数：函数可以作为参数传递、从其他函数返回，或赋值给变量，从而支持高阶函数，从而获得更灵活、可复用的代码
    - 高阶函数：将其他函数作为参数或返回函数作为结果，以促进代码复用和组合，并更轻松地构建复杂的功能
    - 声明式编程：使用表达式描述“做什么”而不是“怎么做”，使代码更简洁、更贴近实际应用，从而提高可读性和可维护性
    - 递归：函数调用自身来解决相同问题的较小实例
    - 常用函数式编程语言
      - Haskell：具有强静态类型和惰性求值的纯函数式编程语言。
      - Lisp：支持函数式编程的最古老的编程语言
      - Erlang：适用于强调不变性和消息传递的并发和分布式系统。
      - Scala：结合了函数式和面向对象编程的特性，运行在 Java 虚拟机 (JVM) 上。
      - F#：运行在 .NET 平台上的函数式优先语言，同时支持函数式和面向对象编程。
      - JavaScript(包含 Underscore.js 和 Lodash 等库)：支持函数式编程概念。

- Lambda 架构
  - 通过结合批处理和实时数据处理来处理大规模数据
  - 关键概念
    - 批处理层：使用分布式存储系统(如 HDFS)和处理框架(如 Hadoop)处理大量历史数据并生成批处理视图
    - 速度层：使用流处理框架(如 Apache Storm 或 Spark Streaming)进行实时数据处理，以提供低延迟更新
    - 服务层：合并批处理层和速度层的结果，以提供统一的数据视图，并将处理后的数据提供给用户查询
  - 连续或分批生成原始数据(日志)
  - 消息队列收集原始数据并将其提供给两个层
  - 数据通常作为不可变的主数据集持久存储在分布式文件系统中
  - 计算所有数据的准确历史视图
  - 原始数据存储在容错存储中
  - 处理后的结果发送到服务层

- Micronaut
  - 用于构建具有快速启动时间、最小内存占用和原生云集成的微服务和无服务器应用程序
- Play 框架
  - 用于构建具有高性能和可扩展性的现代 Web 应用程序，具有非阻塞 I/O、热代码重载和内置 RESTful 服务支持
- Apache Kafka
  - 分布式事件流平台，每天可处理数万亿个事件
  - 用于高性能日志聚合、实时分析、事件溯源以及将不同系统与实时数据管道集成
  - 具有容错能力的分布式架构，以服务器集群的形式运行，可跨越多个数据中心
  - 允许通过向集群添加代理节点来实现水平扩展，以处理海量数据
  - 针对高吞吐量进行了优化，可以低延迟处理大量数据
  - 数据持久化在磁盘上并跨多个节点复制，以确保持久性和可靠性
  - 允许多个生产者向主题发布消息以及多个消费者订阅这些主题
  - Streams API，用于处理实时或复杂事件驱动应用程序中的数据流
  - Kafka Connect，用于连接外部系统，例如数据库、键值存储、搜索索引和文件系统。
- Apache Karaf
  - 基于 OSGi 框架构建，支持模块化和动态应用程序开发
  - 主要功能：shell 控制台、远程访问、热部署和动态配置
  - Web 管理控制台支持插件管理、调试和 OSGi 监控
- Apache Camel
  - 集成框架和面向消息的中间件，提供用于路由和中介规则的组件和模式
  - 使用声明式 Java 领域特定语言配置路由和中介规则

- Apache Druid
  - 开源分布式数据存储，专为大规模数据集的高性能实时分析而设计，非常适合时间序列数据、事件驱动型工作负载和交互式查询。
  - 特性
    - 实时数据提取：支持持续数据源更新和实时分析。
    - 列式存储：针对扫描大数据的分析查询进行了优化。
    - 分布式架构：可扩展、容错，并将工作负载分布在多个节点上，以快速查询性能处理大数据。
    - 灵活的数据模型：支持处理时间序列数据和基于事件的数据，适用于日志分析、指标监控和用户行为跟踪。
    - 交互式查询：允许用户快速深入数据并执行复杂的聚合和过滤。
    - 与其他工具集成：例如 Apache Kafka、Apache Hadoop 和 Grafana，用于数据处理和可视化。
- Apache CXF
  - 用于使用 JAX-WS 和 JAX-RS 等前端 API 开发服务。主要特性
  - 主要特性：
  - Web 服务标准：支持 SOAP、RESTful HTTP Web 服务标准
  - 协议支持：兼容 HTTP、JMS 和 CORBA 等多种协议
  - 集成：可与其他 Apache Camel 和 ServiceMix 集成
- ActiveMQ
  - 开源、多协议、基于 Java 的消息代理。支持多种消息传递协议
  - 主要特性：
    - JMS 支持：实现 Java 消息服务 (JMS) 用于消息传递
    - 高可用性：提供集群和故障转移功能
    - 协议支持：支持 AMQP、MQTT、STOMP 等协议
    - 集成：可与 Apache Camel 和 CXF 集成
- Apache ServiceMix
  - 开源集成容器将 Apache ActiveMQ、Camel、CXF 和 Karaf 的功能整合到一个强大的运行时平台中
  - 提供基于 OSGi 的完整、企业级 ESB(企业服务总线)。
  - 主要功能包括：
    - 可靠消息传递：使用 Apache ActiveMQ 进行消息传递
    - 路由和集成：利用 Apache Camel 进行路由和集成模式。
    - Web 服务：通过 Apache CXF 支持 Web 和 RESTful 服务
    - 基于 OSGi 的运行时：由 Apache Karaf 提供支持

- 阿里云
  - 在亚洲占据主导地位，提供针对电商、大数据和中国市场定制的可扩展服务市场
  - 主要服务
    - 计算：弹性计算服务 (ECS)、函数计算
    - 存储：对象存储服务 (OSS)、块存储
    - 数据库：云数据库(RDS、MongoDB 等 NoSQL 选项)
    - 人工智能/机器学习：机器学习平台 (PAI)
    - 大数据：MaxCompute(数据处理)
    - 其他：CDN、电商和物联网解决方案

- 私有云和混合云
  - 私有云的构建有时出于更高的安全性需求，通过将数据保存在专用环境中来降低未经授权的访问和数据泄露的风险。
    - 使用云管理平台管理云基础设施，例如 OpenStack。
    - 实施自动化工具来管理虚拟机、资源和配置，例如 Ansible、Puppet 或 Chef。
    - 使用容器编排工具自动执行容器化应用程序的部署、扩展和管理，例如 Containers。 Kubernetes (K8s)
  - 混合云使用公有云执行非敏感任务，将关键数据保留在私有云上

- 云原生
  - 为云环境设计和部署应用程序
  - 应用程序充分利用云的可扩展性、弹性、韧性和灵活性
  - 关键方面
    - 微服务架构
      - 应用程序被划分为更小、松耦合且独立的服务
      - 服务可以跨云单独部署和扩展
      - 可以灵活地为每个服务选择技术和编程语言
      - ...

    - 服务网格：管理服务到服务通信的基础设施层，例如 Istio 与 Kubernetes 协同工作，提供负载均衡、服务发现、身份验证和监控功能。
    - 开放服务网关倡议 (OSGi)：Java 模块化开发框架，常用于大型企业级应用程序，允许将应用程序由多个可重用组件(bundle)组成，并部署在容器中。
    - 容器: 使用 Docker 打包代码和依赖项，确保跨环境的一致性。
    - 编排
      - 使用 Kubernetes 等工具管理容器、自动化部署和扩展：
        - Pod：Kubernetes 对象模型中最小、最简单的单元，表示包含共享存储或网络资源的容器的运行进程实例。
        - 服务：定义 Pod 逻辑集合和访问策略的抽象，用于实现集群内外应用程序不同部分之间的通信。
        - 部署：管理 Pod 部署和扩展的抽象，为 Pod 和副本集提供声明式更新，以确保应用程序达到预期状态。
        - 副本集：确保在任何给定时间运行指定数量的 Pod 副本。
        - ...

- OLAP(联机分析处理)
  - 支持快速分析多维数据的数据处理
  - 用于商业智能应用程序中的复杂查询、数据挖掘和报表。
  - 多维分析支持跨维度的复杂计算和查询
  - 数据聚合：汇总不同级别的数据，以便快速访问
  - 数据立方体：以高效分析的方式组织和存储数据
- OLTP(联机事务处理)
  - 专为高交易量、实时处理和确保数据完整性而设计
  - 高交易量支持大量短时间的在线事务
  - 实时处理，确保快速处理事务
  - 使用 ACID(原子性、一致性、隔离性、持久性)机制确保数据完整性
- 对象关系映射
  - 应用程序中的对象与关系数据库中表之间的映射
  - 避免在多个位置重复且变化很小或没有变化的代码
  - 提高数据库操作的效率
- 面向对象数据库
  - 将数据存储在对象中，就像面向对象编程语言中的结构化数据一样
  - 支持类层次结构和继承，以表示对象之间的复杂关系
  - 数据和行为的封装
  - 可在多个应用程序之间重用，可扩展，适用于大型数据集

  ...

- 对象存储
  - 旨在存储大量非结构化数据，例如媒体文件、备份和日志，具有高可扩展性和持久性。它常用于云存储解决方案。
  - 将数据存储作为对象进行管理和操作的数据存储架构
  - 包含文件类型、大小、创建日期和权限等信息的元数据
  - 用于从存储池中检索对象的唯一标识符
  - 通常采用冗余存储，确保数据免受硬件故障的影响
  - ...

XIII. 国际化

- 编码标准定义了字符、符号和控制代码如何以二进制形式表示，以便于计算机处理和存储。
- Unicode
  - 通用字符编码标准，涵盖全球字符和符号。
  - 允许在单个文件中无缝处理多种语言。
  - 促进不同系统和设备之间的全球通信和数据交换。
  - 支持添加新字符和脚本。
  - 代码点：映射到特定字符的数值，例如：’A’ 的代码点为 U+0041
  - 编码格式：
    - UTF-8：最广泛使用的 Unicode 编码格式，每个字符使用 1 到 4 个字节：
    - 1 个字节：U+0000 到 U+007F 之间的 ASCII 字符
    - 2 个字节：U+0080 到 U+07FF 之间的希腊文和西里尔文字符
    - 3 个字节：U+0800 到 U+FFFF 之间的中文、日文、韩文字符
    - 4 个字节：U+010000 到 U+10FFFF 之间的表情符号和古文字
    - UTF-16：每个字符使用两个或四个字节
    - UTF-32：每个字符使用四个字节的固定长度

  ...

- Web3
  - 区块链基础
    分布式账本(例如以太坊、Solana)在数千个节点上存储数据并执行代码，以确保透明性、不可篡改性和无单点故障。
  - 智能合约
    区块链上的自执行程序可实现无需信任的交互自动化。
  - 去中心化应用程序 (DApp)
    - 具有前端(类似 Web 2.0)但后端位于区块链上的应用程序。
    - 通过 MetaMask 等钱包连接，使用加密货币进行访问或支付。
  - 身份与所有权
    - 用加密密钥钱包取代用户名，以证明资产/数据的所有权
    - 独特的 NFT 代币用于在链上证明所有权，如数字艺术品或游戏道具
  - 去中心化存储
    跨节点存储文件，以弥补区块链容量不足的不足
- Web 4.0
  - 尚未正式定义，仅为概念和推测
  - 超智能、共生的 Web，几乎与现实生活无缝衔接
  - 由 AI 驱动的 Web，具有增强的数据分析、预测和自动化能力
  - 名为 Think IoT 的智能设备提供 AI 助手
- 游戏开发
  - 类型：个人电脑、Web、移动设备和游戏主机(PlayStation/Switch/XBox)
  - 主机游戏开发需要获得批准后才能使用特定设备和工具。
  - AAA 游戏：由艺电 (EA)、育碧、动视、索尼互动娱乐和腾讯等大型公司制作的高预算、高知名度的电子游戏，具有高质量的图形、复杂的游戏玩法和沉浸式体验。
  - 游戏引擎
    - Unity：一款多平台游戏引擎，广泛用于 2D 和 3D 游戏开发，拥有丰富的库、图形资源和移动设备支持。
    - Unreal Engine：用于开发 AAA 级游戏和独立游戏，拥有高质量的图形和多功能性，并对 3D、虚拟现实和增强现实提供出色的支持。
    - Godot：一款开源游戏引擎，易于使用且灵活。
  - ...

- 图形和动画 API
  - DirectX：API 绕过 Windows 图形设备接口 (GDI) 并直接与硬件通信：Direct3D、DirectDraw、DirectMusic、DirectPlay 和 DirectSound。最新版本为 12。
  - OpenGL：用于渲染高质量图形的跨平台图形 API
  - Vulkan：用于高性能图形的低开销跨平台 API
  - Metal：Apple 为 iOS、macOS 和 tvOS 设计的图形和计算 API
  - WebGPU：为 Web 应用程序提供现代 GPU 访问的 Web 图形 API
  - WebGL：用于在 Web 浏览器上渲染 3D 和 2D 图形的 JavaScript API
- 声音和音乐
  - FMOD：用于在游戏中实现音效和音乐的中间件，具有类似 DAW(数字音频工作站)的直观界面
  - Wwise：用于动态混音、空间音频和互动音乐的中间件
  - Audacity：免费开源音频编辑工具
- 网络
  - Photon：用于创建实时多人在线游戏的网络解决方案
  - 游戏对象/实体的网络代码，Mirror：用于创建客户端-服务器网络模型、合作式或竞技性动作游戏的 Unity 库
  - UNet：Unity 的简单网络解决方案，缺乏可扩展性和性能，已弃用
- 人工智能
  - 寻路算法：类似 A*(A-star)的算法，广泛用于寻找两点之间的最短路径，尤其用于非玩家角色导航，这种导航通常涉及航点或用于在环境中导航的特定点
  - 行为树：适用于复杂的 AI 行为、决策和分层任务，这些任务使用分层结构，其中节点表示任务或动作，可以是复合的(包含其他节点)、装饰器(修改行为)或叶节点(动作或条件)。
  - 有限状态机：适用于简单且可预测的行为，由表示特定行为或动作的状态以及基于条件或事件发生的转换组成。

* 视频、动画和声音之间的同步对于创建高质量的人体动画至关重要，因为唇部动作需要与对话相匹配。

- 企业移动管理（EMMI）
  - 使企业能够为员工提供随时随地办公的灵活性
  - 用于管理和保护移动设备的一套技术、流程和策略
  - 确保公司数据安全，禁止安装任何允许的应用程序
  - 身份和访问管理（IAM）处理身份验证、授权和单点登录（SSO）功能
  - 移动设备管理（MDM）处理设备注册、配置、安全执行和远程数据擦除。
  - 移动应用管理（MAM）管理设备上移动应用的部署、更新和移除。控制应用权限，确保仅使用获准的应用。
  - 移动内容管理（MCM）确保安全访问公司内容，例如文档共享、编辑和同步，同时确保数据安全。
  - 使组织能够实施自带设备办公（BYOD）策略，而无需公司自有设备。
  - 特定的 EMM 服务可在本地或云端安装。
  - 需要付费许可证或订阅。
- 微信/WeChat
  - 使用 JavaScript、WXML 和 WXSS 开发轻量级应用(称为小程序)
  - 开发工具：使用微信开发者工具进行编码、调试和测试
  - 访问微信的 API，实现支付、社交分享和用户身份验证等功能
  - 文档：https://developers.weixin.qq.com/doc/
- 支付宝
  - 支付宝支持使用小程序进行应用开发
  - 使用支付宝开发者平台获取工具、API 和 SDK
  - 集成支付解决方案、营销工具和用户管理功能
  - 资源：https://developers.alipay.com/

- HarmonyOS Next(鸿蒙星河版)
  - 使用华为集成开发环境 DevEco Studio 创建应用
  - 支持 ArkTS(基于 TypeScript 的语言)和 ArkUI，用于构建分布式应用程序
  - HarmonyOS SDK 包含用于应用框架、系统开发、媒体、AI 等的工具
  - 使用 DevEco Testing 进行自动化测试和调试
  - 发布您的华为应用市场中的应用
  - 资源：https://developer.huawei.com/consumer/cn/develop/

XVI. 人工智能（AI）

- 人工智能（AI）
  能够执行任务的计算机系统通常需要人类智能，例如学习、推理、解决问题、感知、语言理解和决策。
- 通用人工智能（AGI）
  能够理解、学习和执行人类能够完成的任何智力任务，旨在在广泛领域复制人类的广泛认知能力。
- 生成式人工智能
  旨在创建文本、图像、音乐或视频内容的人工智能系统。
- 机器人流程自动化（RPA）
  自动执行重复性任务和流程，以提高效率并降低运营成本
- 具身人工智能
  与身体集成的智能系统，通过感官和运动功能与环境互动，注重现实世界的互动和通过经验学习
- 发展
  - 艾伦·图灵于 1950 年提出的图灵测试：衡量机器表现出与人类无异的智能行为的能力
  - 符号推理：逻辑理论家和一般问题解决者
  - 基于规则的系统：使用广泛知识库模拟人类专家决策过程的专家系统
  - 神经网络：从数据中学习的感知器
  - 机器学习（ML）
    - 使用统计方法的数据驱动方法
    - 旨在从数据中学习并随时间改进的算法
  - 深度学习
    - 多层深度神经网络
    - 处理海量数据(大数据)并执行复杂任务，例如图像和语音识别、自然语言处理(NLP)(语法分析、语义理解、语用理解和处理)语音识别和游戏，例如 AlphaGo

- 卷积神经网络 (CNN)
  - 专为处理网格状数据而设计的专用神经网络
  - 使用卷积层自适应地自动学习特征的空间层次结构
  - 使用滑动滤波器扫描和检测输入中的模式，例如边缘和形状
  - ...

- 生成对抗网络 (GAN)
  - 概述：GAN 由两个神经网络组成：生成器和鉴别器，它们相互对抗以生成真实的数据。
  - 生成器创建虚假数据，而鉴别器评估其真实性，从而生成高质量的合成数据。
  - 适用于创意 AI 生成图像、音频或文本，但训练难度高且不稳定
  - ...

- 长短期记忆网络 (LSTM)
  - 一种 RNN，可捕捉长程依赖关系并长时间保存信息。
  - 使用门控来控制信息流，解决了传统 RNN 的梯度消失问题，即控制记忆或遗忘的内容。
  - 适用于语言建模、机器翻译和语音合成

- 前馈神经网络 (FNN)
  - 最简单的神经网络，数据直接从输入流向输出，没有循环或记忆。
  - 多层神经元(输入、隐藏、输出)通过 ReLU 或 S 型函数激活，使用权重和偏差处理数据。
  - 易于构建，适用于静态数据，但未经调整的序列或图像则无用。

- AI Agent
  自主实体与环境交互，实现特定目标或执行任务，能够感知周围环境，根据观察做出决策并采取行动。
  - OpenManus
    - Manus AI Agent 的开源替代方案，由 MetaGPT 社区开发。
    - 任何具备基本技术技能的人均可在 GitHub 上免费获取。
    - 模块化框架，允许用户构建可定制的 AI 代理，能够自主处理从编码到数据分析的复杂任务。
    - ...

- Transformer
  - 神经网络模型，凭借并行处理和可扩展性，彻底改变了自然语言处理(NLP)、语音识别和计算机视觉。
  - 关键组件
    - 自注意力机制
      权衡句子中不同词语之间的相对重要性，并捕捉长程依赖关系，从而理解数据上下文关系。
    - 多头注意力机制
      通过并行捕捉数据的各个方面，并使用多个注意力头同时关注输入数据的不同部分，从而理解复杂的关系。
    - ...

- AutoGen
  以其灵活性和易用性而闻名，可用于开发用于各种应用的 AI 代理。
- 语义内核
  专注于自然语言理解和生成
- CrewAI
  强调多个 AI 代理之间的协作以解决复杂问题
- RASA
  常用于构建以自然方式与用户交互的对话式 AI 代理
- Caffe
  伯克利人工智能研究中心 (BAIR) 开发的深度学习框架，针对速度、模块化和表达架构进行了优化，特别针对卷积神经网络 (CNN) 和计算机视觉任务。

- 生成模型
  - 变分自编码器 (VAE) 通过将输入数据映射到低维潜在空间(编码器)，生成潜在变量的概率分布，然后从中重建原始数据(解码器)，从而生成模糊图像。
  - 扩散模型通过前向扩散将噪声逐步转化为有意义的输出，即将噪声逐步添加到输入数据中，直至变为纯噪声，从而生成高质量图像；以及使用神经网络进行逆向扩散以消除噪声并恢复原始数据

- 模型上下文协议 (MCP)
  - Anthropic 的开放标准，用于简化 LLM 连接到外部数据源/用于构建代理和工作流的工具的方式
  - 定义客户端和服务器之间如何交换上下文信息，例如消息框架、将请求与响应链接起来以及高级通信模式
  - MCP 服务器：通过 MCP 公开特定功能的轻量级程序
  - MCP 主机：通过 MCP 与数据交互的应用程序，例如 Claude Desktop 或 IDE
  - MCP 客户端：用于维护与服务器连接的协议客户端
- 代理到代理协议 (A2A)
  Google 开放协议，用于促进多个 AI 代理之间的通信和协调
- 大型模型系统组织 (LMSYS)
  致力于开发和研究开放、可访问可扩展的大型模型及支持系统基础设施的开放式研究组织，包括对话式人工智能模型 Vicuna、基于 Elo 排名的评估平台 Chatbot Arena、训练数据集 LMSYS-Chat-1M 以及大型语言模型人工智能聊天机器人训练、部署和评估平台 FastChat。

- AI 基准测试
  - 评估 AI 模型性能、准确性和效率的必备工具
  - 提供标准化测试和指标，用于比较不同的模型
  - 常见的 AI 基准测试：
    - AIME 2025
      美国数学邀请赛数学测试题
    - 聊天机器人竞技场 (LMArena)
      测试众包盲测 A/B 测试，用户可对模型答案进行投票
    - Geekbench AI
      使用真实机器学习任务的跨平台基准测试，用于评估设备上的 AI 能力以及 CPU、GPU 和 NPU 的工作负载性能
    - 通用 AI 助手 (GAIA)
      评估在真实任务中的表现，重点关注推理、多模态处理、网页浏览和工具使用能力
    - GLUE/SuperGLUE
      语言理解、测试情绪分析和问答的早期标准
    - GPQA(科学)
      测试物理、化学和生物学研究生水平的博士问题
    - ImageNet
      Vision 图像分类基准测试，使用 1,000 个类别和 120 万张图片
    - LiveCodeBench
      测试 2024 年 10 月至 2025 年 2 月期间的真实编程任务，测试功能性代码生成
    - ...

XVII. 物联网（IoT）

- 物联网（IoT）
  由嵌入传感器、软件和网络连接的物理设备、车辆、家电和其他物体组成的网络，使其能够收集和交换数据。
- AIoT：将人工智能（AI）与物联网（IoT）相结合。
- 用途：节省成本的自动化、收集实时数据用于决策、提升客户体验、连接和远程控制应用者以及从任何地方访问信息。
- 应用
  - 农业：指导灌溉
  - 医疗保健：患者监测和紧急情况警报
  - 工业自动化：监控机械、预测维护或优化供应链
  - 零售：跟踪库存并提升客户体验
  - 智慧城市：管理交通模式和自动驾驶
  - 智能家居：自动化照明、供暖和安防系统
- 要求
  - 传感器或其他用于收集/传输数据、跟踪位置、检测或避开障碍物的设备
  - 例如温度、湿度、运动、光照、压力、接近度和气体传感器、摄像头和GPS
  - 数据处理和分析软件及硬件，用于本地或云端分析数据
  - 用户界面，包括移动应用程序、网页仪表盘、可穿戴设备、物理按钮和开关、触摸屏、手势控制、运动检测和语音助手
  - ...

- 金融科技 (Fintech)
  - 创新运用技术提供金融服务和产品
  - 运用区块链、人工智能、大数据和移动应用等各种技术提供传统金融服务：
    - 数字支付：点对点移动支付解决方案和数字钱包
    - 借贷：点对点借贷服务和平台
    - 投资：机器人顾问、在线算法交易和加密货币交易所
    - 个人理财：预算应用程序和个人理财管理工具
    - ...

- 区块链
  - 去中心化、安全的数字账本技术，通过计算机网络记录交易，确保没有任何单一实体能够控制，数据也无法轻易篡改
  - 2008 年，中本聪首次引入加密货币比特币
  - 去中心化金融 (DeFi)：新的金融系统，无需传统银行即可实现直接点对点交易，并提供去中心化的借贷和交易服务
  - 数字身份验证：提供安全防篡改的数字身份，减少欺诈，增强金融交易的信任
  - 智能合约：自动执行的合约，其条款直接写入代码，交易在满足预定义条件时自动执行，以减少对中介机构的需求
  - 代币化资产：现实世界资产的代币化，例如非同质化代币 (NFT) 代表所有权或对独特物品或内容的真实性证明
  - ...

- 去中心化应用程序 (DApp)
  - 应用程序在去中心化网络而非中心化服务器(例如区块链)上运行
  - 利用智能合约实现流程自动化，无需中介机构
  - ...

- 大数据
  - 传统工具无法处理的庞大而复杂的数据集，其特点如下：
    - 数据量：以 TB 或 PB 为单位的海量数据
    - 数据速度：数据以实时或近实时数据流的方式生成和处理
    - 数据种类：不同类型的数据，例如文本、图像、视频或传感器收集的数据，其格式包括结构化、半结构化或非结构化

  - 实施
    - 确定用例和数据源
    - 设置并部署分布式计算框架以处理规模数据，例如 Hadoop、Spark
    - 使用数据采集工具(例如 Kafka)收集和流式传输实时数据
    - 将数据存储在数据湖(集中式存储库)或 NoSQL 数据库(例如 Apache Cassandra 或 MongoDB)中
    - 使用批处理或流处理技术分析数据
    - 使用数据可视化工具 Tableau、Power BI 构建仪表板或报告以获取洞察

- 量子计算
  - 利用量子力学原理处理信息的尖端计算
  - 使用同时存在于多种状态(叠加)并相互纠缠(纠缠)的量子比特或量子位
  - 与仅表示 0 或 1 的经典比特不同，量子比特可以同时表示 0 和 1，因此可以执行大规模并行计算
  - 即使相距甚远，一个量子比特的状态也与另一个量子比特的状态直接相关，这使得信息处理速度更快、更高效
  - 使用量子门来操纵量子比特的状态以执行计算
  - 量子计算机增强机器学习和人工智能解决复杂问题、分解大数、优化物流或在原子层面模拟分子相互作用，甚至可以破解当前的加密方法。
  - 它由超导电路、捕获离子或光子构成，冷却至接近绝对零度以保持量子比特的稳定性。
  - 由 IBM、谷歌、微软和亚马逊等公司资助开发。
  - D-Wave 专注于量子退火，而 Rigetti 正在开发全栈解决方案。
  - IBM 提供了一个名为 Qiskit 的模拟平台和一个开源软件开发工具包(SDK)，可与使用 Python 的量子计算机协同工作。
  - 微软宣布推出世界上第一个使用拓扑量子比特的量子处理器 Majorana 1，其设计旨在提高稳定性和抗错误能力。
  - 亚马逊打造的量子计算芯片 Ocelot 致力于通过使用发现的创新型猫量子比特来降低错误率。

XX. 安全

- 定期检查 OWASP 十大最关键的 Web 应用程序安全风险：
  - 访问控制失效：用户访问数据或执行他们无法访问的操作
  - 加密失败：加密实施错误导致敏感数据泄露
  - 注入：攻击者将恶意代码注入程序
  - 不安全的设计：可被利用的设计缺陷
  - 安全配置错误：错误的配置和设置使应用程序易受攻击
  - 易受攻击和过时的组件：使用存在已知漏洞的组件
  - 身份识别和身份验证失败：用户身份验证和会话管理问题
  - 软件和数据完整性故障：无法确保软件和数据的完整性
  - ...

- 运行代码扫描工具以识别常见问题，例如 Codacy 或 SonarQube
- 运行免费的 OWASP Zed Attack Proxy (ZAP) 扫描 Web 应用程序以查找安全漏洞
- ...

- 应用服务器设置以解决常见的安全问题
  - 隐藏服务器和操作系统版本，以防止服务器信息泄露
  - 禁用目录列表，以防止访问者查看内容
  - 从受信任的证书颁发机构 (CA) 获取有效证书，以配置 SSL/TLS，从而在服务器和客户端之间使用 HTTPS 通信
  - 限制请求大小，以防止拒绝服务 (DoS) 攻击
  - ...

- XSS 跨站脚本攻击
  攻击者将恶意脚本注入其他用户查看的网页
  - 问题
    - 窃取用户信息
    - 篡改网页内容
    - 传播恶意软件
  - 类型
    - 反射型 XSS：脚本从 Web 服务器反射，例如在搜索结果或错误消息中
    - 存储型 XSS：存储在服务器(例如数据库)中，在用户请求相关内容时发送给用户
    - 基于 DOM 的 XSS：在受害者浏览器中修改 DOM 环境后执行的脚本
  - 原因
    - 输入验证不足
    - 输出编码不当
  - 解决方案
    - 严格输入验证并过滤危险字符和脚本
    - 对输出数据进行适当的输出编码，例如 HTML 实体和特定上下文
      - 使用内容安全策略 (CSP) 设置 HTTP 安全标头
      - 在 Web 服务器中设置 X-XSS-Protection 响应标头
    - 安全开发实践
      - 使用安全的编程框架和库进行自动处理
      - 定期进行代码审计和安全测试
    - 通常，会转义或替换特殊字符，例如 < 或 >，以防止插入代码
    - 为确保亚洲文本正常显示，请通过检查字符代码点来绕过它们

- 跨站请求伪造 (CSRF)
  - 恶意网站诱骗用户在经过身份验证的 Web 应用程序上执行操作
  - 实现 CSRF 令牌，这些令牌在每个会话和请求中都是唯一的，并带有时间限制，以确保请求来自经过身份验证的用户，并通过内置支持的库和框架发出
  - 将 CSRF 令牌作为 Cookie 和请求参数发送，以便服务器验证它们是否匹配
  - 将 Cookie 的 SameSite 属性设置为 Strict 或 Lax
  - 要求状态更改请求使用自定义标头 X-Requested-With，以确保请求来自来自同一域名，而非恶意站点
  - 验证 Referer 头，确保请求来自可信来源
  - 关键操作需要用户通过重新输入密码或输入验证码进行双重确认
- 缓冲区溢出或超限
  - 程序向称为缓冲区的临时存储区写入的数据超出其可容纳范围，导致相邻内存被覆盖、程序异常崩溃或恶意代码执行。溢出会覆盖存储在堆栈中的关键控制数据，例如返回地址。它会告诉程序在函数调用后应该去往何处，并写入一些称为 shellcode 的汇编代码。
  - 使用内存安全语言(例如 Python、Java 或 C#)写入数据之前，检查缓冲区的边界。
  - 执行输入验证，确保其符合预期的格式和大小，以防止格式错误或恶意输入导致缓冲区溢出。
  - 注意编译器警告，使用工具检测潜在的缓冲区溢出，例如 SonarQube。
- 竞争条件
  - 两个或多个进程或线程同时访问共享资源，执行结果取决于特定的执行时间。
  - 攻击者在程序的检查和访问之间寻找机会窗口，通过将临时文件替换为指向敏感文件的符号链接来快速切换资源，从而造成意外修改。
  - 在没有适当锁定的情况下并发打开和关闭文件描述符会导致文件描述符泄漏。
  - 将检查和操作合并为单个原子操作，并使用适当的锁定机制和安全方法创建临时文件，以避免符号链接攻击。
- 单点登录 (SSO)
  - 一种身份验证方法，允许用户使用一组凭证(用户名和密码)登录一次即可访问多个应用程序和服务。
  - 基本工作流程
    1. 用户身份验证：用户使用凭证登录 SSO 系统
    2. 令牌生成：SSO 系统根据用户身份和相关信息生成令牌
    3. 令牌分发：将令牌发送到用户的浏览器或设备
    4. 访问请求：用户访问其他应用程序时，将向中央服务器请求令牌
    5. 令牌验证：应用程序向中央服务器验证令牌以确认身份
    6. 访问授权：验证通过后授予用户访问权限

- OAuth
  开放的访问委托标准，授予应用程序有限的用户信息访问权限。
  - 代表资源所有者批准交互，或第三方应用程序代表自身获取访问权限。
  - OAuth 2.0 通常使用 JSON Web Tokens (JWT) 作为访问令牌，以便在各方之间安全地传输包含以下内容的字符串信息：
    - 使用 HMAC SHA256 算法签名的令牌标头元数据
    - 有效载荷使用 base64 编码 JSON 格式的实际数据(称为声明)
    - 使用密钥或公钥/私钥对创建签名加密哈希，以证明令牌未被篡改
  - 用户登录，服务器发出 JWT，客户端在每次请求的 HTTP 标头中发送 JWT，服务器验证签名并信任有效负载，而无需访问数据库
- OpenID Connect (OIDC)
  - 基于 OAuth 2.0 协议的简单身份层
  - 通过添加 ID 令牌扩展 OAuth 2.0，使用令牌进行访问
  - 用户登录时，依赖方 (RP) 重定向到 OpenID 提供商(OP)
  - OP 返回包含您身份信息的 JWT 令牌，或可选的访问令牌(用于访问 API，类似 OAuth 风格)以及刷新令牌(用于稍后获取新令牌)

- 场景
  - 基础
    - 检查招标需求中列出的功能
    - 检查所有用户故事的需求规范
    - 确保功能规范中列出的所有功能都经过测试
    - 检查会议记录
    - 检查用户发送的有关新增或变更需求的电子邮件
  - 测试技术
    - 边界：空值、最小值和最大值
    - 类型：尝试不同的数据类型，例如：错误值
    - 用户组：尝试具有不同权限的不同用户组
    - 并发：多个用户在同一时间或时间段执行相同功能
    - 状态转换：测试所有状态变化，并判断其是否有效
    - 顺序：尝试不同的顺序、导航或路径(关键路径优先)
    - 决策表：列出所有可能组合的可追溯性矩阵
    - 检查数据库中的值是否发生变化
  - 数据
    - 使用自动化工具生成不同的数据集(通常需要更正)
    - 在非办公时间停工测试真实数据(必要时加密)

    ...

  - 设备
    - 不同的配置，例如 CPU、内存和存储
    - 安装不同的软件版本
    - 不同的网段，防火墙或代理设置不同
    - 不同的网络环境，无线和局域网
    - 台式机、平板电脑或移动设备，但尽可能避免模拟环境
    - 连接到不同的后端服务器
    - 确保客户端计算机上应用了必要的客户端软件并设置有效
  - 人员
    - 同行评审，如果在项目外部尤其有用
    - 主管双重检查
  - 工具：
    - 使用 Junit 提前检查错误
    - 使用 AI 工具自动创建多样化且全面的测试场景

XXII. 调试

- 难点
  - 没有特定的环境或产品可供检查，例如非生产平台的支付网关或不同版本的产品
  - 由于安全和隐私保护，没有确切的数据可供测试
  - 由于环境高度敏感，无法访问特定平台，例如警察局
  - 没有相关设备，例如低性能配置的旧机器
  - ...

- 策略
  - 正确检查并解释错误消息，但有时错误是由其他原因引起的
  - 记录返回的错误代码(如有)，并在网络上搜索可能会有所帮助，很多时候无法解决问题，但可以提供提示和方向，以便进一步深入研究问题。
  - 逐步检查并审查逻辑流程，找出问题或缺失的内容。
  - 首先跟踪潜在区域，然后逐步缩小范围，直到找到根源。
  - ...

- 日志记录
  - 在函数启动和退出时放置跟踪代码，记录开始和结束时间以检查耗时。
  - 记录代码中执行操作的人员、时间和操作内容。
  - 记录函数执行前后特定变量值的变化。
  - 使用日期时间格式将语句、变量值或错误消息输出到控制台或日志文件。
  - 应用日志轮换，至少每 3 个月压缩一次日志文件，以节省服务器空间。否则会影响服务器性能。
  - 在数据库中设置标志位，以打开或关闭是否显示调试日志。

  ...

- 技巧
  - 检查客户端控制台、服务器和事件日志是否存在错误。
  - 检查数据值是否已写入数据库。
  - 常见问题：拼写错误、范围错误、库错误、权限错误、连接错误、设置错误或数据损坏。
  - 如果在特定外部调用后没有返回任何结果，则进行跟踪，例如：异步调用
  - 检查代码是否陷入死循环
  - 检查并发进程是否存在死锁或值覆盖
  - 检查函数执行过程中的 CPU、内存和存储空间使用情况
  - 源代码
    - 确保错误被捕获或通过实现 try...catch 块或类似机制报告
    - 使用测试框架编写单元测试，以便及早发现错误

    ...

XXIII. 性能调优

- 往返时间 (RTT)：性能调优旨在缩短 Web 应用程序响应速度的关键性能指标，即从客户端向服务器发送请求，到服务器处理后将响应返回给客户端的整个过程。
- 分析
  - 通过以下方式跟踪并定位问题发生的位置：
    - 调试日志记录函数的开始和结束时间，以计算处理时间
    - 使用性能监控工具收集统计数据，例如 JMeter、Neoload 等

    ...
- 服务器
  - 纵向扩展(垂直)
    - CPU：添加或升级
    - 内存：添加或更换为更快的内存
    - ...
  - 横向扩展(水平)
    - 网络配置：负载均衡或集群，在服务器之间分配负载
    - 在本地添加服务器
    - 云环境下的服务器复制
- 代理服务器提供缓存或仅在更改或过期后才发出请求
- 反向代理将传入的客户端请求分发到多个后端服务器，并在发送给客户端之前压缩来自后端服务器的响应

...

- 使用集群在不同的服务器之间分配负载，例如 Oracle 真正应用集群
- 使用闪回功能，无需备份还原即可在各个级别(行、事务、表或整个数据库)撤消更改，从而快速从人为错误中恢复，例如 Oracle 的闪回功能。
- DBLink(数据库链接)允许一个数据库无需复制即可查询另一个数据库的远程数据，这有助于分散工作负载，但依赖于网络延迟和优化。
- ...

- 编程
  - 关闭未使用的对象资源以清理内存，例如手动垃圾回收
  - 使用变量绑定代替编写单个 SQL 语句以减少脚本编译
  - 显示加载消息并显示剩余等待时间
  - 应用异步以避免持续等待其他任务完成
  - 允许程序使用多个处理器或核心同时执行多个任务
    - 并行处理，将任务拆分为可同时处理的子任务
    - 在线程池中并发运行多个线程，并使用工作线程从主线程卸载耗时任务
  - 缓存：页面、对象、应用程序
  - ...

XXIV. DevOps

- 一套促进应用程序开发（Dev）团队和系统运维（Ops）团队之间更好协作的理念和实践
- 采用迭代开发、流程自动化、可编程部署和维护，确保持续交付高质量的软件
- 常见实践
  - 配置管理
    - 在整个系统生命周期内进行系统性变更管理，确保系统配置正确、可跟踪、按预期运行，并以可控的方式进行更新，避免系统崩溃
    - 关键方面
      - 识别配置项（CI），建立安全和性能基线
      - 控制规划，实施变更控制流程、评估和审批
      - 状态核算，通过自动化方式跟踪和报告配置项和变更请求的状态
      - 审计，验证配置项是否符合要求并确保系统完整性
    - 关键组件
      - 工件存储库
      - 集中存储构建工件、库和依赖项
      - 源代码存储库
      - 存储和管理代码库及其历史记录
      - 配置管理数据库（CMDB）
      - 存储信息并跟踪硬件、软件、网络和其他设备的状态资产
    - 工具
      - Ansible：开源自动化工具，采用基于 YAML 的配置，无需安装代理，可跨多个系统协调复杂的工作流，自动执行重复性任务和工作流，并安全处理凭证和敏感数据。
      - Puppet：开源配置管理工具，采用声明式语言，由 Puppet 代理执行，模块化代码，提供详细的报告、监控和工具集成功能。
      - Chef：将基础设施即代码（IaC）转化为自动化平台，以可扩展且高效的方式管理和自动化基础设施。
    - 系统配置
      - 在服务器中配置平台设置，例如： Java 命名和目录接口（JNDI）
      - 从环境变量中读取特定平台的设置，而不是从配置文件中读取

      ...

XXV. 网络

- 性能
  - 通过计算待传输数据的大小、并发用户数量以及数据传输开销(例如请求/响应行和报头)，确保有足够的带宽来处理预期的数据流量，并避免出现瓶颈。
  - 通过升级到光纤等更高速的连接、使用专用链路以及部署内容分发网络（CDN）将内容分发到更靠近用户的位置，从而最大限度地减少延迟，确保数据能够及时传输。
  - 实施负载均衡，在服务器之间均匀分配流量，防止过载。
  - 监控网络性能，以便及时发现和解决问题。
- 可靠性
  - 使用备用服务器或多条网络路径，确保出现任何故障时能够持续运行。

    ...
- 安全性
  - 使用超文本传输安全协议（HTTPS）保护客户端和服务器之间的数据通信。
  - 原始协议安全套接字层（SSL）用于数据通信加密，即使是 SSL 3.0 版本现在也被认为是不安全的，已被传输层安全性（TLS）取代。
  - 由于 TLS 1.3 的安全性和性能增强，建议在新的实施中使用，但 TLS 1.2 也被认为是安全的，并且被广泛使用。

    ...

- 代理
  - 隐藏真实 IP 地址，保护服务器或工作站的在线身份
  - 通过阻止对恶意或未经授权内容的访问来进一步增强安全性
  - 缓存频繁访问的网页，以减少减少服务器负载并加快用户访问速度
  - 反向代理：位于外部客户端和内部服务器之间，将客户端请求转发到相应的内部服务器
- 远程连接
  - 使用虚拟专用网络 (VPN) 提供安全的远程网络访问
  - 连接到政府网络需要获得批准，否则可能构成犯罪
  - 需要扫描计算机并安装监控软件
- 三层架构
  - 由 Web 服务器、应用服务器和数据库服务器组成
  - 每一层均可独立扩展，以应对不断增加的负载
  - 隔离机制使故障排除、更新和维护工作更加轻松，且不会影响其他层级
  - Web 服务器
    - 位于非军事区 (DMZ)，一个独立的网段充当公共互联网和内部网络之间的缓冲区
    - 处理来自客户端(Web 浏览器)的 HTTP/HTTPS 请求，并提供 HTML、CSS 和 JavaScript 文件等静态内容
    - 能够将传入流量分配到多个应用服务器
    - DMZ 通过将 Web 服务器与内部应用服务器和数据库服务器的直接访问隔离，从而保护内部网络
  - 应用服务器
    - 位于内部网络内部
    - 只能从 Web 服务器访问，以减少攻击面
    - 处理动态内容、处理业务逻辑并执行应用程序代码
  - 数据库服务器
    - 位于内部网络内部
    - 只能从应用服务器访问
    - 存储和管理应用程序数据
    - 无法被 Web 服务器直接访问，也不受外部威胁

- 存储区域网络 (SAN)
  - 增强存储设备对服务器的可访问性，就像本地连接一样
  - 使用磁盘阵列和磁带库添加或扩展存储以满足存储需求
  - 使用光纤通道 (FC) 提供高速数据传输速率
  - 使用主机总线适配器 (HBA) 将服务器连接到 SAN
  - 使用 SAN 交换机管理使用之间的数据流
  - 数据镜像和复制，以增强数据保护和灾难恢复能力
  - 广泛应用于数据中心，并支持虚拟化环境
  - 集中管理存储资源
  - 利用独立磁盘冗余阵列(RAID)配置将多个物理硬盘组合成一个逻辑单元进行数据存储
    - 通过在多个硬盘上存储数据的冗余副本来提供容错能力
    - 允许并行访问并提高整体性能
    - 常见的 RAID 级别
      - RAID 0(条带化)：将数据均匀分布在所有硬盘上，无冗余
      - RAID 1(镜像)：将数据复制到两个或更多硬盘上，提供容错和快速读取功能
      - RAID 5(带奇偶校验的条带化)：将数据和奇偶校验信息分布在三个或更多硬盘上
      - RAID 6(带双奇偶校验的条带化)：与 RAID 5 类似，但增加了奇偶校验信息，从而能够同时对两个硬盘故障提供容错能力
      - RAID 10(RAID 1 + 0)：结合镜像和条带化，提供高性能和容错能力，但需要更多硬盘

XXVI. 服务器管理

- 监控和性能调优
  - 系统监控：使用 Nagios、Zabbix 或 Prometheus 等工具检查系统性能，以便及早发现潜在问题。对于 Windows，检查事件、系统和应用程序报告的错误。
  - 性能调优：通过调整系统设置、管理资源和解决瓶颈问题来优化服务器性能。

- 安全管理
  - 用户管理：管理用户帐户、权限和访问控制，确保只有授权用户才能访问系统。
  - 安全补丁：定期应用安全补丁和更新，保护服务器免受漏洞和威胁。
  - 防火墙配置：配置和维护防火墙，保护服务器免受未经授权的访问和攻击。
  - 证书安装：生成证书签名请求 (CSR)，向证书颁发机构 (CA)(例如 DigiCert)申请服务器证书。验证后，CA 将颁发服务器证书和中间证书。此文件通常为 .cer、.crt、.pfx 或 .pem 格式，可安装在服务器证书管理器上。

- 软件安装与更新
  - 安装软件：根据需要安装和配置服务器软件和应用程序。
  - 更新软件：保持服务器软件和应用程序为最新版本，以确保兼容性和安全性。

XXVII. 文档

- 传统的正式项目通常需要以下文档：
  - 项目计划
    - 目标和宗旨
    - 成本效益分析
    - 团队及人员
    - 进度安排
  - 需求目录
    - 用户角色、岗位、级别、群组、姓名、部门、地址、邮箱和电话列表
    - 用例：操作、流程、手动或自动程序、规章制度

    ...

    - 功能需求
      - 功能名称、ID 和描述
      - 类型：通用、案例、报告和管理
      - 模式：在线或批量

      ...

    - 非功能需求
      - 页面加载时间
      - 用户操作响应时间
      - 并发用户数：平均/峰值

      ...

- 系统文档有三个用途：
  - 客户进行设计验证
  - 开发人员或支持人员遵循的实施指南
  - 接手或后续团队理解系统
- 提醒
  - 所有文档均遵循组织文档风格指南，通常附带封面，其中包含标题、目标组织、编写方、日期、版本、版权声明；分发列表和修订历史记录。每个内容页都有页眉、页脚和页码。
  - 文档需要由各方仔细检查、验证、修改和签署。
  - 切勿将其视为家庭作业，而应认真对待。

  ...

- 需求
  - 涵盖所有细节并与用户确认，以避免实施遗漏或不准确。
  - 讨论时，让所有利益相关者和前端运营人员参与。
  - 尽早寻求最高权威意见，以避免在最终阶段推翻设计。
  - 任何需求会议录音前，均应征求批准。
  - 以电子邮件形式抄送所有相关方，即可达到目的。
  - 收集小组代表的回复，但不要逐一传阅，以节省时间。

  ...

  - 确保在需求收集阶段涵盖以下事项：
    - 用户类型、角色、群组、权限、总数、休假资源安排
    - 注册、身份验证和授权
    - 日常运营业务流程和审批流程
    - 同一时间或时间段的最大并发用户数
    - 申请表以及需要存储的数据或信息
    - 接口

    ...

- 系统规模
  - 确定特定时间段(例如 5 年)的软件和硬件需求
  - 检查预期用户数、每秒事务处理量、数据类型/大小/容量
  - 检查平均和峰值负载、负载和响应时间
  - 计算所需的内存、存储空间和网络带宽、CPU 速度和核心数
    - 估算结果将得出最终分数，然后可以检查供应商有哪些符合基准的系统型号可供选择

第4章. 数据迁移和转换

I. 定义

i. 数据迁移
当来源和目标的数据结构相同时，要从一个或多个系统迁移数据到新系统。

ii. 数据转换
由于数据结构不同，来自一个或多个系统的数据将在新系统中重新创建。

...

II. 源数据

- **资料库记录**
  - i. 表和字段
    除系统表之外的所有表和字段，不再使用或没有数据的表
  - ii. 数据类型
    1. 文字
    2. 数值
    3. 日期时间
    4. 二进位物件，例如图像

    ...

- **脚本**

现有资料库中可重复使用或需要引用的资料库 SQL 脚本
例如，检视、触发器、函数或存储过程
...

VII. 数据分析

- 假设:
  - i. 数据正在使用中
  - ii. 准确无人为错误
  - iii. 无缺失记录

  * 需要澄清，删除那些不使用和不准确的数据，并添加回那些缺失的数据
- 数据分组
  - **静态数据**

    永不改变的数据
  - **动态数据**
    - **基本数据**
      - 对于系统初始化、启动和运行至关重要

        例如基本设置、功能表选项、系统参数或系统代码
      - 功能增强后会不定期变化

      ...

- **数据字典创建**
  - 新系统的数据结构应由开发团队准备好
  - 列出源资料库和目标资料库中的所有表和字段，最好在实体关系图中

VIII. 关系映射

- 源资料库和目标资料库架构映射
- 确定资料加工规则：
  - 哪些源表/字段映射到哪些目标表/字段
  - 旧资料库中的字段资料类型已更改为新资料库中的数据类型
  - 某些字段在转换过程中需要为新表定义预设值
  - 将源表字段中的值分隔到目标表上的多个字段或相反，例如地址
    - 由于数据结构的差异而导致的映射关系：
      - 一对一
      - 一对多
      - 多对一
      - 多对多

      ...

IX. 转换计划

需要起草数据转换，其中包含以下详细资讯：

- 源中的数据清单及其档案大小
- 数据映射和数据转换规则
- 总体方法：多轮试运行和演练、冻结期、切换日
- 提取、转换和验证程序逻辑
- 数据验证和验收规则
- 结果报告
- 过时数据的处理

...

第5章. 安全与隐私保护

I. 定义

安全是指保护计算机系统及其处理的信息免遭未经授权和蓄意的访问、修改或破坏，尤其指机密、敏感或个人信息。后者的保护是指隐私控制，通常由法律法规对个人数据的收集、使用、存储和分发进行规范。

II. 识别威胁

安全威胁可分为以下类型：

i. 数据泄露

1. 未经加密显示机密信息
2. 未经授权存取特定数据
3. 未经授权或无意地将数据传输给其他人或地点
4. 存储在电子介质(如 USB 闪存盘)中的数据的物理丢失

ii. 黑客攻击

侵入计算机系统，目的如下：

1. 访问或窃取数据
2. 修改交易数据
3. 修改信息显示，例如网页篡改
4. 修改信息传输，例如更改消息
5. 删除记录，例如日志
6. 控制计算机系统，例如启用或禁用门锁

iii. 故障

1. 禁用特定计算机系统或设备的服务
2. 加密数据以阻止企业正常运营
3. 冒充特定计算机系统或设备

III. 识别攻击

- **恶意软件**
  - 病毒
    可能损坏或删除计算机上数据的计算机程序。
  - 木马程序
    伪装成正常程序下载到计算机上造成损害的程序。
  - 间谍软件
    未经许可进入计算机、收集数据并将其发送给第三方的软件。
  - 蠕虫
    利用目标计算机的安全漏洞(弱点)进行自我复制并在计算机网络中传播，从而感染更多计算机的计算机程序。它会消耗大量内存和带宽。
  - 键盘记录器
    记录键盘输入以窃取密码或其他机密信息的软件或硬件。

- **拒绝服务 (DDoS)**
  通过对单个或多个系统进行过度请求，导致服务器或网络资源不可用。

- **零日攻击**
  未报告或未公布的软件漏洞，黑客可以利用这些漏洞，让人没时间创建补丁或寻找变通方法。

- **高级持续性威胁**
  长期持续的黑客攻击过程，利用恶意软件、网络钓鱼或带有恶意 URL 的电子邮件等系统漏洞，监控并提取特定目标的数据，收集周边基础设施的信息，破解密码并获取管理员权限，将控制权扩展到其他工作站和服务器。

- **勒索**
  通过加密计算机的主文件表或整个硬盘驱动器来锁定用户文件、应用程序或系统，直到用户支付赎金。赎金通常通过电子邮件或网页中隐藏的勒索软件链接进行，该链接可以与回调服务器建立连接，并设置唯一密钥来加密目标计算机上的数据。

IV. 实施预防措施

数据

- 识别并分类需要保护的信息：
  机密、保密、敏感或其他重要数据，例如个人身份信息
- 对印刷媒体、报告、日志、数据库等中的个人身份信息进行加密
- 对日志、数据库、数据或配置文件等中的密码进行加密
- 按用户组维护和更新数据访问控制列表

...

系统应用

- 按用户组维护和更新数据和功能的访问控制列表
- 对个人身份信息(例如印刷媒体、报告、日志、数据库等)进行哈希处理
- 个人密码或身份信息应进行哈希处理且无法解码
- 系统密码应进行加密或屏蔽(例如日志、数据库、数据或配置文件等)
- 存储或传输数据须加密
- 对称加密(例如 AES)的密钥长度至少为 128 位，非对称加密(例如 RSA)的密钥长度至少为 2048 位
- 用于加密和解密的密钥应定期更改
- 采用安全的身份验证方法
  - 多重身份验证 (MFA)：短信或电子邮件
  - 无密码身份验证：生物识别（指纹或人脸识别）、移动推送通知或身份验证器（微软或谷歌）
  - 基于令牌的身份验证：首次登录后获取临时访问令牌，常用于API和Web应用
    - OAuth 2.0：无需共享凭证即可授予资源访问权限，包含访问和刷新令牌
    - OpenID Connect：将身份验证 ID 令牌添加到 OAuth 并格式化为 JSON Web 令牌 (JWT)

    ...

v. 常见漏洞

- **身份验证和会话管理失效**
  利用保护不力的密码、密钥或令牌，冒充其他用户身份。
- **跨站请求伪造（CSRF）**
  强制已登录浏览器向存在漏洞的 Web 应用程序发送预身份验证请求，从而使浏览器能够执行恶意操作。
- **跨站脚本攻击（XSS）**
  在浏览器中执行脚本，劫持用户会话、破坏网站甚至引入蠕虫。
- **注入漏洞**
  通过插入特殊代码或精心设计的数据来执行命令。它可能会更新或删除任何数据。
- **不安全的直接对象引用**
  暴露内部引用，以便在未经授权的情况下访问数据库或文件。
- **恶意文件执行**
  恶意代码和数据的远程文件包含（RFI）

VI. 建立事件处理程序

- 遵循现有的计划和程序，例如事件响应、数据备份与恢复或操作
- 报告
  - 将事件报告给高级负责人员
  - 通知管理员
  - 最好在预计服务恢复时间通知最终用户
- 立即采取行动以限制影响：
  - 禁用公共访问，以防止离线 Web 服务器造成网页篡改
  - 断开网线，避免影响其他计算机
  - 关闭计算机电源，阻止勒索软件加密更多文件
  - 如有需要，请将用户重定向到替代网页

  * 必要时执行手动操作
- 服务器操作
  - 尽可能切换到备用站点或网络
  - 在规定时间内执行灾难恢复并启动弹性系统
- 跟踪并修复问题
  - 通过参考服务器日志和系统事件追踪威胁的根本原因
  - 检查事件发生前访问了哪些数据、文件和程序、URL 或电子邮件
  - 追踪攻击者的源 IP 地址或其他信息
  - 追踪攻击的执行方式和所做的更改
  - 通过修改代码、更改配置或服务器设置来解决问题
  - 重新构建源代码并再次部署到服务器
  - 恢复服务并公告
- 进一步跟进
  - 报警调查
  - 更改所有相关密码
  - 更新服务器补丁，并在必要时进一步更改配置或设置
  - 更新手册，例如事件响应计划、数据备份和恢复计划或操作手册，并包含以下详细信息：
    - 简单明了的步骤，包括如何检查和更改哪些内容
    - 事件发生时应通知哪些相关方
    - 列出所有主要联系人和备用联系人
    - 详细的恢复步骤
  - 详细的备份步骤

VII. 加强隐私保护

存储与传输

- 个人身份信息应进行哈希处理，以电子形式存储和传输。
- 非身份识别个人信息应在存储和传输过程中加密。
- 数据传输应在安全通道中进行。
- 对于两项可识别特定人员的信息，其中一项应加密。

. . .

打印和复印

- 禁止未经授权打印或复印个人信息
- 授权打印或复印的个人信息应妥善保管并妥善保存在安全的地方，以防止数据泄露给未经授权的人员，使用后应销毁。
- 禁止以任何形式私自存储个人信息。

数据传输

- 以电子形式传输个人信息时应加密，以实物形式传输时应置于密封的信封、袋子或储物箱中，以防止数据泄露给未经授权的人员。
- 个人信息原件的传输应仅依靠可靠的工作人员、运输工具或快递员，并提供充分的包装，以防止物理损坏。为防止数据丢失，最好由授权人员陪同。出于法律要求等重要目的，需要由保安人员提供额外保护。

数据收集

- 仅可出于正常业务验证需要获取部分个人身份信息。

. . .

测试与支持

- 如果生产数据用于测试目的，则应进行屏蔽或加密，并在测试结束后立即清除。

  . . .

数据保留

- 在授权或法定保留期之后，数据应以不可恢复且以人类无法理解的格式进行处置。需要在此期限之后保留的数据应获得高层管理人员的充分批准，并且不得违反任何法律要求。

V. 邀请报价或投标

- 收集供应商的联系名单，包括公司名称、员工姓名、职称、电子邮件和电话。
- 通常有专人负责大型企业或公共机构的账户管理。
- 公共机构可能只考虑符合特定条件的供应商。
- 可以通过电子邮件发送报价或招标邀请。应注明投标要求、回复或提交截止日期。

...

VI. 评审与授标

- 评估依据约定规则，所有供应商一视同仁
- 评分依据评分方案计算
- 功能和特性相同，且性能相同时，应将奖项授予产品或服务价格最低的供应商或投标人

...

VII. 采购订单

采购订单需在中标后准备并发送给供应商，订单内容需包含以下信息：

- 待采购商品或服务的订单详情，包括品牌、型号和数量
- 买卖双方的联系信息
- 收货和账单地址
- 合同和付款条款，包括价格、折扣或销售税

...

第7章. 用户验收测试计划

I. 目的

i. 测试系统是否满足以下条件：
    1. 在需求、系统或功能规范中指定的用戶需求
    2. 用户对系统操作和用户界面的期望
ii. 识别所有特定和潜在问题，在系统推出之前纠正并清除它们
iii. 确定系统是否已准备好推出

II. 交付

i. UAT 计划
ii. 测试案例
- 由开发团队起草，经用户修改确认
- 由测试人员在用户验收测试期间填写
- Word 或 Excel 格式
- 包括以下项目：
    1. 测试用例编号
    2. 测试场景描述
    3. 操作步骤
    4. 预期结果

VII. 问题记录

测试人员需要在 excel 或日志系统(如 Redmine)中记录 UAT 期间发现的问题，包含以下信息：

i. **类型**
   - **错误**
     函数未按预期或指定执行
   - **用户界面**
     导航、文本显示或图形问题
   - **性能**
     加载和响应时间
   - **设置**
     系统配置或设置不正确

ii. **分类**
   - **严重性**
     - **严重**
       系统功能无法执行
     - **高**
       发现漏洞或功能与预期不同
       ...
   - **紧迫性**
     - **高**
       需要立即处理
     - **中等**
       需要在一定时间内办理
       ...

XI. 行政措施

- 外部测试人员出席时可能需要签名
- UAT 完成后，测试用例文件将保存在特定文件夹中或发送给开发团队

XII. 系统关闭和恢复

- 修复错误或问题后，系统需要关闭以部署修改后的应用程序
- 开发团队会在部署前和系统恢复后发送电子邮件通知用户。

...

XIII. 评审会议

- 每轮 UAT 完成后， 开发团队和用户之间应召开评审会议，确认 UAT 是否通过，是否可以继续下一步或遵循什么。

...

XIV. 质量因素

可能影响 UAT 性能的用户要求的常见因素，从而导致高故障率或发现许多问题：

- 没有明确记录
- 文件不正确
- 没有深入的理解
- 误解
- 不同的解释

...

III. 推出计划

- 确定**主要**和**补充任务**
- 检查任务**依赖性**，并确定**优先级**
- 估计**所需资源**
- 确定**利益相关者**和每项任务的**责任方**
- 列出**验收标准**
- 建立**可衡量的目标**
- 列出**验证方法**
- 规划切换**流程和时间**
- 确定**回滚条件和步骤**
- 列出可**交付成果**
- 制定**详细工作计划**

IV. 准备工作

i. 明确联络点、角色和职责

角色	职责
用户	1. 提供业务需求 2. 进行结果验证 3. 批准和验收
项目团队	1. 监控和审查整个过程 2. 与用户和技术团队联络 3. 供应商管理
开发团队	1. 功能和技术设计 2. 代码开发和错误修复 3. 系统设置和配置 4. 应用程序部署 5. 提取、转换和加载数据 6. 数据清理 7. 文档
基础设施和网络团队	1. 服务器和云环境准备和设置 2. 网络布线 3. 防火墙和代理服务器配置 4. 文档

iv. 推出步骤

1. 应列出旧系统停止和新系统启动步骤。
2. 旧系统停止顺序应如下，每一种类的服务器可能有多个实例：
3. Web、应用程序、数据库 服务器
4. 执行数据转换脚本以提取、转换和加载生产数据到新系统
5. 如果有系统间依赖关系，则等待其他系统推出
6. 收集转换日志并编写异常报告
7. 提供推出状态、数据转换成功和失败记录，列出错误日志或遇到的其他问题，建议处理过时或不正确数据的解决方案
8. 用户验证

...

v. 回退步骤

1. 回退条件
   - 需要商定在什么情况下应恢复旧系统，即未通过健康检查并发现严重问题。
   - 一些小问题(例如显示不正确的文本)可以立即修复，不会影响推出
   - 新系统停止顺序应如下，每一种类的服务器可能有多个实例：
   - Web、应用程序、数据库服务器
   - 旧系统启动顺序应如下，每一种类的服务器可能有多个实例：

...

2. 互动设计

任务	开始日期	完成日期
需求收集	年/月/日	年/月/日
需求规范		
产品分析	年/月/日	年/月/日
功能规范		
交互设计	年/月/日	年/月/日
信息设计		
界面设计		
交互设计		
原型设计	年/月/日	年/月/日
用户验收测试	年/月/日	年/月/日
产品上线	年/月/日	年/月/日
维护	年/月/日	年/月/日

3. 架构与编码

任务	开始日期	完成日期
需求收集	年/月/日	年/月/日
系统分析	年/月/日	年/月/日
系统设计	年/月/日	年/月/日
文档编写	年/月/日	年/月/日
平台搭建	年/月/日	年/月/日
实施	年/月/日	年/月/日
编码		
单元测试		
安装部署	年/月/日	年/月/日
数据迁移与转换	年/月/日	年/月/日
测试	年/月/日	年/月/日
系统集成测试（SIT）		
用户验收测试（UAT）		
安全审计与隐私审查	年/月/日	年/月/日
系统部署	年/月/日	年/月/日
项目评审报告	年/月/日	年/月/日

参考资料

- **项目规划**

1. 101 Common Causes - Why Do Projects Fail? International Project Leadership Academy, calleam.com, 2016
2. 25 Point Implementation Plan to Reform Federal Information Technology Management, Vivek Kundra, The White House, 2010
3. A Guide to Project Management Body of Knowledge (PMBOK Guide), 6th Edition, Project Management Institute, 2017
4. A Project Manager's Book Of Forms, A Companion To The Pmbok Guide(Sixth Edition), 3rd Edition, Cynthia Snyder Dionisio, Wiley,2017
5. Agile Coaching, Rachel Davies and Liz Sedley, Pragmatic Bookshelf, 2009-10
6. Agile Practice Guide, Project Management Institute, 2018-11
7. Agile Principles, Patterns, and Practices in C#, Robert C. Martin Micah Martin, Pearson, 2006-07
8. Designing Visual Interfaces: Communication Oriented Techniques, Kevin Mullet and Darrell Sano, Prentice Hall, 1994
9. Exploring Requirements: Quality Before Design, Donald C. Gause and Gerald M. Weinberg, Dorset House, 1989-09
10. Head First Design Patterns, Eric Freeman, Bert Bates, Kathy Sierra and Elisabeth Robson, O'Reilly Media, 2004
11. Identifying and Managing Project Risk: Essential Tools for Failure-Proofing Your Project, Tom Kendrick, AMACOM, 2009
12. Improvement Measures for Delivery of IT Projects, OGCIO, 2014
13. Information Technology Project Management, Kathy Schwalbe, Thomson Learning, 2002
14. IT Project Management: Infamous Failures, Classic Mistakes, and Best Practices, Ryan Nelson, Amazon, 2020
15. IT 真相 打通 IT 與商務的通路, 楠真著, 廣東人民出版社 , 2019-01
16. Major Causes of Software Project Failures, Lorin J. May, Department of Computer Science, University of Brasilia
17. Management Capability Assessment, Dennis Wong, Bleu Publications, 2021
18. Microsoft .NET-Architecting Applications for the Enterprise, Dino Esposito & Andrea Saltarello, Microsoft Press, 2014
19. Multimedia ISO 9000 Logic Handbook, China Orient QA Co., Limited, 1994
20. Project Failure: Handling Chaos in IT Industry, 黃演胜, Amazon, 2016
21. Project Recovery: Case Studies and Techniques for Overcoming Project Failure, Harold R. Kerzner, Wiley, 2014
22. Quality Guide to ISO 9001/9002, Akhilesh N. Singh, Kaizen Books, 1996
23. Quality Software Management, Vol. 1: Systems Thinking, Gerald M Weinberg, Dorset House, 1992
24. Quality Software Management, Vol. 2: First-Order Measurement, Gerald M Weinberg, Dorset House, 1993
25. Quality Software Management, Vol. 3: Congruent Action, Gerald M Weinberg, Dorset House, 1994

24. Writing for Multimedia: Entertainment, Education, Training, Advertising, and the World Wide Web, 1st Edition, Timothy Garrand, Focal Press, 1996-12
25. レイアウトデザインのルール 一目を引くページにはワケがある, オブスキュアインク, ワークスコーポレーション, 2008-2
26. 互動設計的藝術, 常成, 清華大學出版社, 2022
27. 失敗から学ぶユーザインタフェース 世界は BADUI(バッド・ユーアイ)であふれている, 中村聡史, 技術評論社, 2015-11
28. 多媒體設計, 高新發、陳妹香, 全華圖書, 2002-10-17
29. 配色デザイン インスピレーションブック, Power Design inc., ソシム, 2018-7
30. 版式基礎：版式設計思維、技術與實踐, 紅糖美學, 北京大學出版社, 2023-6
31. 飛躍的構想力, 多湖輝, 久大文化, 1989
32. 設計方法卡牌, 羅莎, 電子工業出版社, 2017-8
33. 圖解力 面向未來的信息設計, [韓]禹錫晉, 金美利, 人民郵電出版社, 2016-11
34. 網頁設計,陳東生, 龍門書局, 2014-5

■ 架构与编码

系统设计

- 12 Essential Skills for Software Architects, Dave Hendricksen, Pearson, 2012
- 12 More Essential Skills for Software Architects, Dave Hendricksen, Pearson, 2015
- The Art of Scalability, Martin L. Abbott, Addison-Wesley Professional, 2015
- Clean Architecture: A Craftsman's Guide to Software Structure and Design, Robert C. Martin, Prentice Hall, 2017
- Designing for Behavior Change, Stephen Wendel, O'Reilly, 2013
- Enterprise Integration Patterns, Gregor Hohpe & Bobby Woolf, Pearson, 2003
- Head First Design Patterns, Eric Freeman, Bert Bates, Kathy Sierra, Elisabeth Robson, O'Reilly, 04
- Patterns of Enterprise Application Architecture, Martin Fowler, Pearson, 2003
- Refactoring: Improving the Design of Existing Code, Martin Fowler, Addison-Wesley, 2015
- SOA Design Patterns, Thomas Rischbeck & Thomas Erl, Prentice Hall, 2009
- Strategy & Product Development for Complex Systems, Edward Crawley, Bruce Cameron & Daniel Selva, Pearson, 2015
- 軟件架構設計, 温昱, 電子工業出版社, 2012-10
- 構建高性能 Web 站點 改善性能和擴展規模的具體做法,郭欣, 電子工業出版社, 2009-08
- 億級流量系統架構設計與實戰, 李琛軒, 電子工業出版社, 2024-05
- 億級流量網站架構核心技術, 張開濤, 電子工業出版社, 2017-04
- IT アーキテクト最強の指南書, 日経 IT エンジニアスクール, 日経 NETWORK, 日経 BP, 2016-06
- システム設計の謎を解く, 高安 厚思, SB クリエイティブ, 2013-05

系统开发

- Algorithms, 4th Edition, Robert Sedgewick and Kevin Wayne, Addison-Wesley Professional, 2011-03
- Effective De 漏洞 ging, Diomidis Spinellis, Pearson Education, 2017
- Effective Java, Second Edition, Joshua Bloch, Pearson Education, 2008
- Effective JavaScript, David Herman, Pearson Education, 2013
- Guide to the Certified Software Quality Analyst (CSQA) Common Body of Knowledge
- Head First EJB, Kathy Sierra, Bert Bates, O'Reilly, 2003
- Head Rush Ajax, Brett McLaughlin, O'Reilly, 2006
- Head First JQuery, Ryan Benedetti, O'Reilly, 2011
- Pojos in Action, Chris Richardson, Manning, 2006
- Java Coding Guidelines: 75 Recommendations for Reliable & Secure Programs, Fred Long, 2013
- Microsoft .NET-Architecting Applications for the Enterprise, Dino Esposito & Andrea Saltarello, Microsoft Press, 2014
- Quality Software Management Vol 1-4, Gerald M. Weinberg, Dorset House, 1991-1997
- Software Exorcism, Bill Blunden, Apress, 2003
- Software Requirement Patterns, Developer Best Practices, Stephen Withall, Microsoft Press, 2007-06
- 今晚來點 web 前端效能優化大補帖, 莫力全 Kyle Mo, 博碩, 2022
- 你所不知道的必學前端 De 漏洞技巧, 楊楚玄, 博碩, 2021
- Web 開発者のための大規模サービス技術入門, 伊藤 直也 /田中 慎司, 技術評論社, 2010-07
- システムエンジニア最強の指南書, 日経 IT エンジニアスクール, 日経 NETWORK, 日経 BP, 2016-04
- ビッグデータを支える技術, 西田圭介, 技術評論社, 2017-09

数据库和性能调优

- Effective MySQL Optimizing SQL Statements, Ronald Bradford, 2018
- Effective SQL, John L.Viescas, Addison-Wesley, 2017
- High Performance MySQL, Baron Schwartz, Peter Zaitsev, Vadim Tkachenko, 2012
- Oracle Database Problem Solving and Troubleshooting Handbook, Tariq Farooq (etc.), Addison-Wesley, 2016-04
- Oracle High Performance SQL Tuning, Donald Burleson, Oracle, 2001
- Scalability Rules: Principles for Scaling Web Sites, Martin Abbott, Michael Fisher, Pearson, 2017
- 品悟性能優化, 羅敏, 清華大學出版社, 2011-05
- 海量資料庫解決方案, (韓)李華植, 電子工業出版, 2013-05
- 圖解性能優化, 小田圭二, 樽松谷仁, 平山毅, 岡田憲昌, 翔泳社, 2014

作者简介

黄演胜拥有超过二十多年从事信息技术行业的工作经验。

作为开发经理、系统分析师与程序分析员多年，他曾经参与过许多不同类型的项目，包括有门户网站、电子商贸网站、内容管理系统、股票报价系统、电子支援系统、地理信息系统、网上教学系统与教学光碟产品、手机与社交媒体游戏，以及各种政府系统。

他曾在不同的机构工作过，包括两个政府的十六个部门、兩家主要电讯公司、银行与广告公司。从小型公司、著名的上市公司，以至跨国企业也有。

项目规模各不相同，从小型、中型到大型不等。团队由两人至一百多人，涉及的开支由数十万，数百万到上亿港币不等。

他毕业于新加坡理工学院，获得多媒体开发高级文凭；并毕业于格林威治大学，拥有电算系理学士学位。

您可以通过 nichowong@dataflyer.net 联系他。

阅读更多相关主题：

https://dataflyer.net/?page_id=562

系统开发快速参考

https://www.amazon.com/dp/B0FFZ3P3S5

项目规划

如何按时、按预算完成项目并满足用户期望？

https://www.amazon.com/dp/B0F9V5Z17H

交互设计

如何先从信息入手，再进行界面和交互？

https://www.amazon.com/dp/B0F9V74FSW

架构与编码

有哪些技术方案可供考虑以及如何选择？

https://www.amazon.com/dp/B0F9WB1DR4

安全与隐私保护

如何保护您的系统并保护重要数据？

https://www.amazon.com/dp/B0F9T87Y7F

数据迁移与转换

如何从现有系统迁移和转换数据？

https://www.amazon.com/dp/B0CYB17BS1

采购计划

如何获取合适的资源并获得所有审批？

https://www.amazon.com/dp/B0F9TC3STS

用户验收测试计划

如何在上线前识别并解决潜在问题？

https://www.amazon.com/dp/B0DRZZPDWX

系统推出程序

如何安全地发布系统？

https://www.amazon.com/dp/B0DBVC3N1R