# Visionary Art

## 人工智能绘画分享站：项目开发里程碑 3 汇报

2051857 曾诗容——项目主管 + 前端开发人员，
2052636 陈骁——ECS 云服务器技术顾问 + 全栈开发人员，
2050250 李其桐——技术支持 + 全栈开发人员，
2054080 林奕如——需求分析师 + 前端开发人员，
2053865 刘昱彤——产品经理 + 前端开发人员，
1751118 吴达鹏——运维 + 全栈开发人员，
2053868 于采篱——项目主管 + 前端开发人员

Computer Science and Technology Department, College of Electronic and Information Engineering(CEIE), Tongji University.
同济大学 电子与信息工程学院 计算机科学与技术系

2023 年 5 月 30 日

# 项目内容简介

## 项目开发背景

- 随着人工智能技术的飞速发展，AI 绘画技术也日趋成熟。近些日子来，无数精美的 AI 绘画作品都让我们眼前一新。对于没有接触过 AI 绘画领域的小白，他们可能也想体验 AI 绘画的奇妙，但不知从何下手。对于钻研 AI 绘画领域的技术人员，他们可能想分享自己的训练成果，同时和其他从业人员沟通交流，但是缺乏相关的平台。本软件的创建便是为了解决以上问题，为小白和技术人员提供一个在线生成图片，上传分享训练参数并和他人沟通的平台。

## 项目核心功能需求

1. 用户能够在线欣赏、生成图片，下载相关模型参数
2. 用户能够上传分享自己训练的模型参数
3. 用户能够使用软件系统提供的相关社交功能，与其他用户进行交流

# 项目发布测试反馈与 issue 提取

## 相关问题

本小组基于迭代 2 项目发布系统测试的反馈，结合答辩过程中与甲方沟通的相关情况，经过迭代 3 会议讨论，提取了以下问题：

- 在高并发情景下，绘图请求切换模型过多可能导致 Stable Diffusion 服务缓存池溢出，导致服务崩溃
- 模型查询、模型下载、模型上传等功能模块的后端接口由于数据库事务设计不当，存在性能瓶颈，导致用户体验不佳
- SD 服务生成图片存在合法性问题，部分图片中出现了不合理的色块、扭曲的图像结构和我国法律法规禁止在互联网平台上传播的敏感内容
- 前端交互逻辑中存在不合理点，例如搜索界面的搜索按钮应该支持通过键盘按键触发，而不是需要用户手动点击；某些按钮功能提示缺失等
- 配置的 ECS 服务器由于硬件条件限制存在性能瓶颈，在高并发情景下的请求响应与处理可能由于硬件瓶颈失败

# 项目发布测试反馈与 issue 提取

## 解决方案

本小组针对以上问题，在迭代 3 开发会议上进行讨论，最终提出了以下解决方案：

- 优化 SD 服务缓存池策略，基于 LRU 合理设计缓存换出机制，保证缓存命中率的情况下尽可能减少缓存池溢出的可能性
- 在服务端 DAO 操作和数据库事务之间，添加 Redis 中间件缓存访问机制，采用旁路缓存策略，将数据库访问压力转移到 Redis 上，从而在涉及到大表 join 查询或者高并发事务情景下减少数据库访问压力
- 由小组的相关前端开发人员，在单元测试和集成测试中，重新从用户角度出发对于人机交互逻辑进行省察，从而以用户友好为核心导向，进一步优化前端界面交互逻辑的相关功能需求实现
- 由小组的相关 SD 服务开发与维护人员，重新精读 High-Resolution Image Synthesis with Latent Diffusion Models 一文 [1]，分别从 prompt 文本预处理方法、网络模型结构和权重文件检测的三个方向，对于 <x>2image 操作的生成图片合法性进行检测和优化

# 本次里程碑任务

- 根据上述 issue 抽取工作中的相关内容项，对于项目开发中的技术债进行偿还，进一步提升系统的运行性能和可用性
- 进一步部署并且完善项目的自动化测试和 CI 流程，通过 Apifox 管理开发工作流，部署整套的项目自动化集成与测试解决方案，并且撰写测试报告与系统文档，为项目交付上线做好准备
- 使用 docker image 打包环境，并且编写详细的测试报告、软件部署说明和软件使用说明，尽可能降低甲方的部署操作成本，准备软件系统交付工作
- 按照小组开发章程，定时开展组会进行小组内部沟通，拉通对齐组员开发进度，稳步推进开发工作，保证在项目开发后期能够按照计划完成项目开发任务，交付能够让甲方客户满意的软件系统产品

# 工作项总览与相关图表

详见华为云

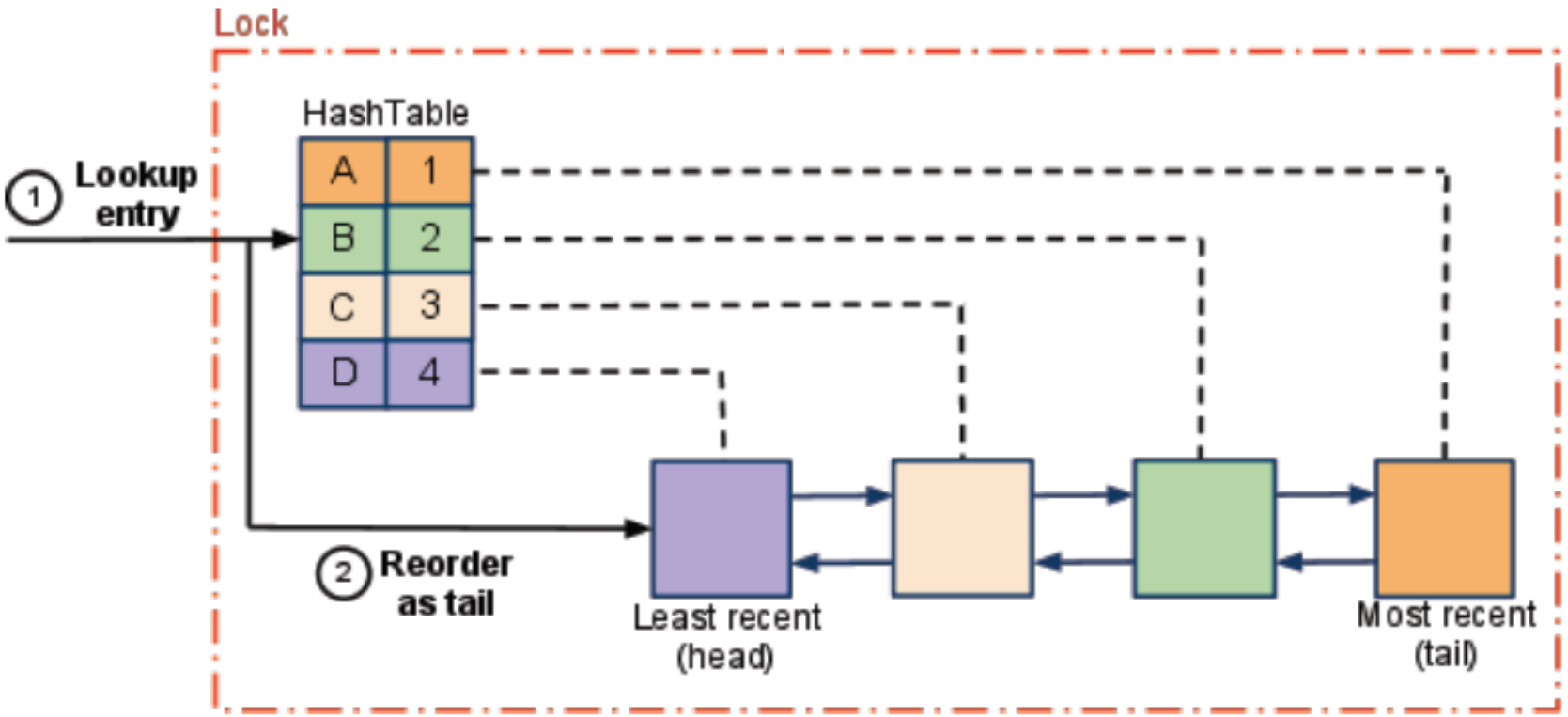
# 项目技术栈

- **主要开发语言**：Python
- **HTTP 服务器**：Tornado + FastAPI
- **前端技术选型**：Vue.js + Element UI/HTML + CSS + Js/Gradio
- **持久层框架**：SQLAlchemy
- **数据库服务**：MySQL + Redis
- **版本管理工具**：Git
- **远程代码托管平台**：华为云
- **接口管理与自动化测试工具**：Apifox + Mock.js

# 优化 SD 服务缓存池策略

- 在之前的单元测试和集成测试工作中，我们小组发现 SD 绘图的最大开销来源于从硬盘加载用户模型这一过程。因此在迭代三中，我们小组的 SD 服务维护人员重新设计了模型加载缓存池算法，采用 LRU 机制，在服务器缓存中维护若干之前用户的模型加载数据结构，并且在缓存池满且缓存未命中时，再从硬盘中将新模型加载到缓存，同时换出 LRU 队列中最近被加载频度最低的模型数据结构，从而有效提高了对于用户绘图请求的响应速度。在此基础上，我们小组的 SD 服务维护人员在经过反复的系统测试和实验后，最终将缓存池大小进行了合理设置，从而使得其既能够保证缓存池中的模型可以满足大部分的用户请求场景，同时也不会因为缓存池过大而导致内存占用过高，影响 SD 服务的运行性能。

# 优化 SD 服务缓存池策略



# Redis 中间件缓存机制实现

- 在迭代 3 中，我们小组在 Tornado 服务和 MySQL 服务之间，架设了 Redis 中间件，采用旁路缓存策略，在数据库事务到来时，首先在 Redis 中查询缓存，缓存未命中时再访问数据库，然后将数据库中的数据读出并且写入 Redis 缓存，从而有效减少了对于 MySQL 数据库的访问次数，提高了系统的运行性能。
- 同时，为了保证 MySQL 和 Redis 的数据一致性，我们小组在迭代 3 会议中经过讨论和架构设计，最终决定采用缓存延迟双删的策略，即在数据库事务执行前先删除缓存，然后在事务执行完毕后，延迟一段时间后再次删除 Redis 缓存。这样在一般的并发情境下，基本上在数据库服务中不会出现脏数据。

# 生成图片合法性保证与检测

- 在迭代 2 汇报与甲方的沟通中，甲方提出了一个对于生成图片合法性进行检验的项目需求。因此，我们小组的开发人员经过会议讨论，最终对于这一合法性检验需求，提出了采用 Trie 进行非法 prompt 字段匹配校验，从而过滤非法生成字符，并且在用户 prompt 中，引入 prompt prefix 装饰器机制，对于 C 端调用的 prompt 接口方法进行封装，从而控制 OpenCLIP 编码器产生的文本向量，在每个 step 的收敛过程中都能够保证生成的文本向量在 prompt prefix 的约束下进行的整套合法性检验解决方案，在最终的系统测试中，该方案能够有效保证生成图片的合法性，并且极大地提升了用户裸文本（raw text）输入情况下的生成图片质量。

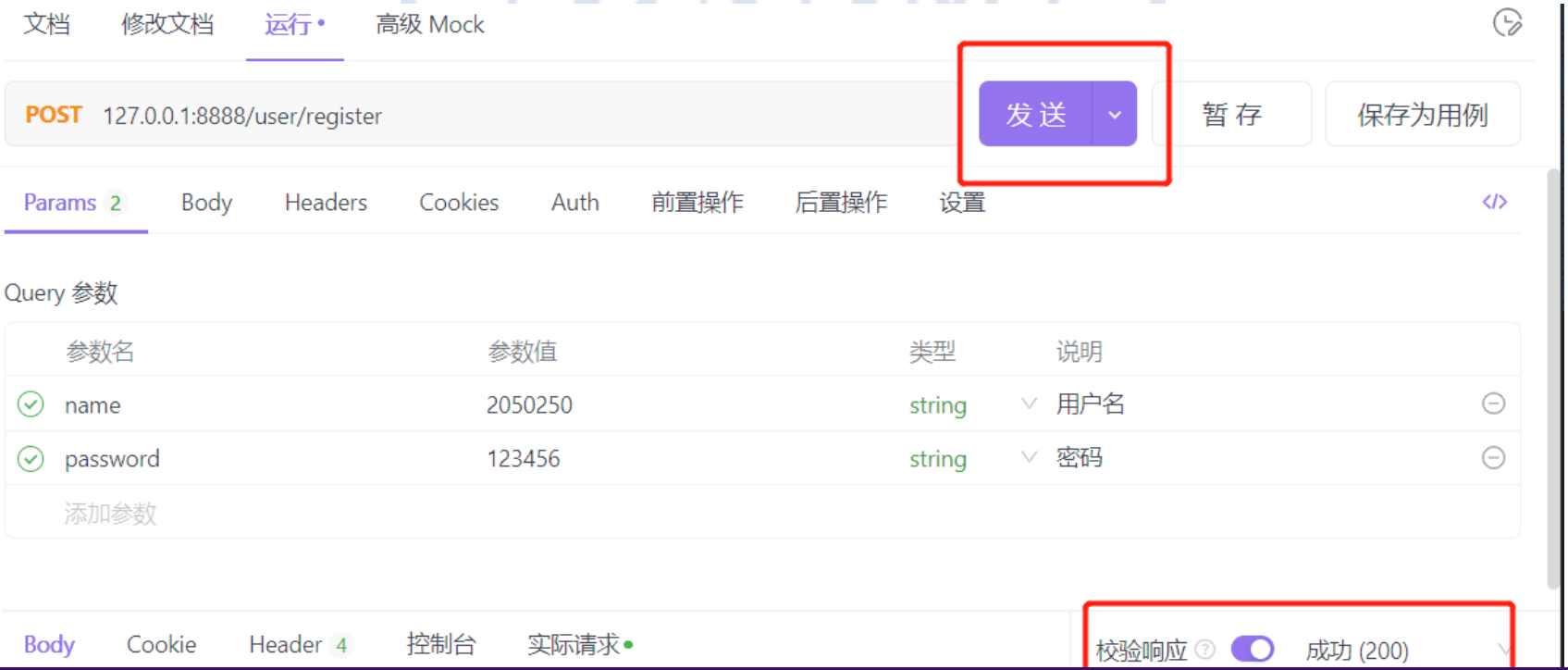
# 前端交互逻辑优化

迭代 3 开发工作中，我们小组负责前端开发的同学针对之前迭代 2 中前端交互逻辑的不足，对于前端的交互逻辑进行了优化，主要包括以下几个方面：

- 搜索界面中，支持了通过单击回车键进行搜索的交互逻辑。
- AI 画图界面中，对于 prompt 文本进行了更加详细的描述，加入了提示性文本引导用户操作，避免用户使用时因为功能和参数过于复杂导致的困扰
- 修复了登录界面中，登录栏与注册栏在浏览器页面缩放下的错位问题。
- 修复了若干界面显示样式问题，使得整个系统的前端交互逻辑更加合理。

# 单元测试

对于软件系统的单元测试，我们小组分别采用了使用 Apifox 工具和 Python requests 库请求的方式，请求每个实现接口对应的 url，并且分别验证接口返回数据格式的正确性、接口根据实际业务请求内容做出响应的正确性和接口在高并发情景下的执行正确性。首先，Apifox 接口测试流程大致如下。在每个接口定义中，填入参数并且发送，比如这里以/user/register 接口为例：



# 单元测试

## Apifox 报告

| | | |
|---|---|---|
| 测试场景 | 用户管理接口压力测试 | |
| 运行时间 | 2023-05-22 11:26:43 | |
| 运行工具 | Apifox v2.2.39 | |
| | **总数** | **失败数** |
| 循环数 | 50 | 0 |
| HTTP 接口请求数 | 300 | 0 |
| 断言数 | 0 | 0 |
| 总耗时 | 1m 52.80s | |
| 总返回数据 | 27.45KB | |
| 平均接口请求耗时 | 70ms | |
| 通过率 | 100.00% | |
| 失败率 | 0.00% | |
| 未测率 | 0.00% | |

# 集成测试

随后，在对于所有的功能接口进行单元测试正确无误后，我同样使用 Apifox 测试工具，对于系统进行集成测试。所谓集成测试，是在上文进行的单元测试基础上，将所有模块按照设计要求（如根据结构图）组装成为子系统或系统，并且以子系统为单位进行测试工作，从而验证每个模块在经过组装后，是否依然能够正确运行。这里，我同样以用户管理功能模块中的用户注册、登录和退出登录为例，对于我进行的集成测试工作具体方法和流程进行阐述。首先，在 Apifox 中设计一套模拟新用户访问系统用户管理模块的具体操作逻辑，如下所示：

# 集成测试

用户注册、登录、退出登录

删除 编辑

目录

创建者 LiQiTong

优先级 P0

创建时间 5 分钟前

修改时间 几秒前

修改者 LiQiTong

测试步骤 测试数据 测试报告

已选 5 项 移除

循环10次

POST 用户注册 (用户注册)

POST 用户登录 (用户登录)

If session_key 等于 null

POST 用户退出登录 (用户退出登录)

拖入或 添加步骤

拖入或 添加步骤

+ 添加步骤

* 运行环境 localhost:8...

测试数据 停用

循环次数 5

遇到错误时 忽略

线程数 10

间隔停顿 0 毫秒

保存请求/响应详情

全部请求

保存变量变化值

使用全局 Cookie

保存 Cookie 到全局

# 集成测试

可见这里模拟了一个典型的用户访问软件系统交互逻辑，即注册新账户后登录，如果成功则储存 session_Key，访问系统其它服务，最后登出账户，在服务端的 Redis 缓存中相应移除用户 session 标识，并且更新用户 stat 字段为离线。该测试用例运行结果如下：

# 系统测试

最后，在上述的两个测试阶段工作全部完成，并且测试结果验证正确无误后，我们小组的测试人员从软件系统客户端用户的角度，将实现的软件系统客户端打包发布，邀请同学和相关非开发人员的第三方用户来使用软件系统 C 端页面并且给出评价。同时，我们小组测试人员也对于其中 C 端的相关主要功能模块，从系统的层面进行了一个比较完整的用户交互逻辑测试，并且观察客户端的数据响应情况，从而能够有效地评估软件系统作为一个整体的性能表现。

# 目标和预期

- 围绕甲方 SRS 需求文档为核心，高质量地完成相关功能需求的开发任务
- 围绕功能用例，时刻开展自动化测试，保证项目的稳定性和可用性
- 上线项目，邀请用户体验并给出测试反馈，收集用户意见和 Bug 反馈，为第三次迭代的内容规划提供参考

# 信息和工具

- 小组中对于相关功能需求分析实现，很大程度上参考了甲方提供的 SRS 需求文档，以及甲方提供的用例图，从而使得设计与实现内容与甲方需求最大程度地做到匹配
- 技术上，我们小组借助相关网络资料以及以往的开发经验，并且和相关从业专业人士进行沟通，充分对于当前行业的流行技术生态进行调研，选择最有利于敏捷开发的技术栈，从而使得项目开发效率得到了很大的提升

# 困难与阻碍

- 我们小组开发在迭代三遇到的最大的困难是在项目部署到服务器运行并测试时，遇到项目在实际的网络环境与本地环境下运行的差异，导致项目之前的许多功能实现可能在本地运行高效、良好，但是在网络环境下则由于网络带宽、高并发场景等原因，导致项目运行效率低下。这导致我们虽然很快地实现了相关的项目需求，但是依然需要负责相关功能模块的同学花费许多的时间对于原本功能实现的底层算法与代码逻辑进行优化，尽可能地降低算法对于系统资源的消耗。这也使得我们深刻意识到了实际生产环境下的项目开发与测试的重要性，以及对于项目的高效性、可用性的重视程度，让我们对于实际可用的工业化项目的敏捷开发流程有了更加深刻的认识。

# 优势和创新

- 得益于我们小组在迭代 1 中搭建的良好项目管理架构，我们在迭代三中进一步对于整个管理流程进行了完善，高度规范化了组员从功能开发、自动化测试、代码提交、审查、合并、部署、上线等整个项目开发流程，从而使得我们小组的开发人员能够更加专注于项目的功能实现，而不是处理各种由于项目流程管理不当产生的问题，使得我们小组的开发效率得到了进一步的提升。
- 我们小组在第一次迭代的基础上，继续完善技术栈，加入了 Gradio + FastAPI 的技术选型实现 AI 画图服务，使得我们小组的项目更加完善，功能更加丰富，同时也使得我们小组的项目更加具有创新性。

# 结果和进度

- 我们小组在每周的组会上，都会各自汇报本周自主开发以及集中开发中完成任务的情况，并且由会议记录人员进行统计。最终结果表明，小组在本次迭代中，完成了甲方提出的所有功能需求，并且在自动化测试方面也取得了很大的进展，使得我们小组的项目更加稳定可靠，结果还是比较让我们感到欣喜和自豪的。
- 我们小组的开发进度推进稳健，在下一次迭代（第三次迭代）中，主要将会根据当前的相关 Bug 反馈和用户需求，对于系统的一些相关问题进行修复和优化。

# 情绪状态

- 我们小组成员在迭代开发中，由于本身都还有其它大量的学业和工作压力，因此在迭代开发过程中，有时候会出现一些情绪波动，影响打出代码的可靠性和质量，导致提交时审查门禁不通过，被发回修改。但是我们小组成员都能够很好地调整自己的情绪，保持积极向上的心态，最终完成了本次迭代的开发任务。

# 心得体会与开发意义

- 通过本次迭代任务，我们小组成员之间的配合更加融洽并且具有默契，能够作为一个团队更加高效与良好地完成相关的项目开发任务，并且更加熟悉了类似的大型项目的开发流程与管理模式，为之后的下一次迭代乃至今后学习工作中的类似项目开发情景打下了良好的基础。

# 参考文献

[1] Robin Rombach et al. *High-Resolution Image Synthesis with Latent Diffusion Models*. 2021. arXiv: 2112.10752 [cs.CV].