3．总体设计

3.1 铸画游戏处理流程：

1. 玩家登录
2. 服务器根据玩家登录信息，从数据库中获取玩家账户数据
3. 玩家选择要进行的游戏，点击参与游戏，游戏将在设定时间开始，玩家初始得分为0。
4. 一次游戏4局，一局游戏包含8轮，每轮游戏时间为10秒。
5. 第一轮游戏从数据库里取两张图，每个用户可以选择一张进行投票。用户可以通过点击图片来选择，用户选择将存入数据库。
6. 10秒后，两张图片的投票人数进行比较，用户如果选的图进入下一轮那么用户得分加一。每轮结束都给用户反馈一个结果，显示用户是否选对人数多的图片。
7. 获胜的图片将输入到AI算法中，算法将生成2张图片作为下一轮的选项。
8. 游戏界面实现动态效果，每轮结束后将会动态刷新图片。
9. 重复步骤5-8，直到8轮（1局）游戏结束，生成1张NFT图像。
10. 重复步骤5-9，直到生成4张NFT图像（4局），游戏结束。
11. 用户的账号信息，游戏得分等数据将存储在数据库中。
12. 每次游戏开始前，前端将从数据库中获取所需的图片信息，并在游戏中展示。每轮结束后，后端将根据游戏结果更新用户的得分等信息，并将下一轮游戏所需的图片信息返回给前端。同时，获胜的图片将输入到AI算法中，生成下一轮游戏所需的图片信息。
13. 游戏结束后，系统会根据玩家得分排名，确定得分最高的玩家，并为其免费提供一幅 NFT 作品作为奖励。

2. 前端处理流程（待补充）：

1. 用户登录（要补充为什么登陆后才能玩游戏）游戏是一个需要保存用户数据的系统，我们需要将用户的游戏状态，投票选择以及得分记录在数据中，如果用户没有登录，这些数据将无法与其账户绑定，可能会造成数据丢失或混乱。
2. 用户进入游戏页面，游戏开始后发送请求到后端获取第一轮两张图片的url。
3. 后端将两张图片的url返回给前端，前端根据url渲染图片。
4. 前端开始每轮10秒的计时，10秒后自动停止投票，展示结果。
5. 用户选择一张图片，前端发送用户的选择信息给后端。
6. 后端计算结果并返回信息，前端根据后端返回的信息，展示用户选择的图片的投票结果、当前得分、胜负情况等信息。
7. 后端记录用户的选择，并从数据库中取出下一轮的两张图片的url返回给前端，前端再次渲染图片。
8. 重复3-7步骤，直到游戏结束。

3. 后端处理流程（待补充，用了什么方法要写清楚）：

1. Flask应用程序接收前端HTTP请求，从数据库中取出第一轮2张图片的url，返回给前端。
2. 接收前端用户选择的图片url，使用SQLAlchemy ORM工具操作数据库，将用户选择的图片url数据保存到数据库中。
3. 根据数据库中的用户选择计算每轮的胜负和得分，标记获胜的图片。
4. 将上一轮获胜的1张图片的url输入到AI算法的API中，获得生成的2张图片的url。采用Python的requests库发送HTTP请求，接收并解析API的响应数据。
5. 将生成的两张图片的url保存到数据库中，返回给前端作为下一轮游戏的图片，Flask框架处理HTTP请求，将生成的图片url数据返回给前端。

4. 数据库处理流程：

1. 用户在注册后，其信息将被保存到User表中。（空了一段登陆后的没写）这个登录跟数据库没有交互
2. 当用户进入游戏并进行选择后，用户的选择将被保存到Vote表中。
3. 当游戏轮次结束后，将选出胜者的图片ID输入到AI算法中，生成新的图片ID和URL，保存到Picture表中。
4. 当下一轮游戏开始时，从Picture表中获取新的图片ID和URL，替换掉原有的图片ID和URL，并更新数据库中的轮次状态。
5. 当游戏结束后，用户的得分将被保存到用户表中。

5. AI算法流程：

1. 接收API输入，将输入的url转化为图片。
2. 将1张图片作为输入，使用VACGAN算法生成两张新的图片。
3. 将生成的两张图片保存，以url的形式返回给API调用者。

3.2注册与登录流程

1. 用例名称：用户注册

主要参与者：潜在用户，系统管理员

前提条件：

* 潜在用户必须具有互联网访问权限。
* 潜在用户必须在应用程序中访问注册页面。

后置条件：

* 用户成功注册并获得帐户访问权限。

基本流程：

1. 用户访问应用程序的注册页面。
2. 用户输入所需的注册信息，例如用户名，电子邮件地址和密码，并确认同意应用程序的条款和条件。
3. 用户单击“注册”按钮。
4. 系统验证输入的信息是否有效，并检查该用户名是否已经存在。
5. 如果输入的信息有效且该用户名尚未注册，则系统创建用户帐户并向用户发送确认电子邮件。
6. 用户收到确认电子邮件并点击确认链接以激活其帐户。
7. 系统将用户重定向到登录页面，提示他们输入其凭据以登录其新帐户。

备选流程：

* 如果用户输入的信息无效或缺失，则系统将提示用户更正其输入并重新提交表单。
* 如果该用户名已经存在，则系统将提示用户选择另一个用户名。
* 如果用户没有收到确认电子邮件，则他们可以要求系统重新发送确认电子邮件或联系客户支持人员以解决问题。

异常流程：

* 如果系统在注册过程中遇到任何技术问题，例如服务器错误或连接问题，则系统将显示适当的错误消息并提示用户稍后重试。
* 如果用户的输入被系统视为垃圾邮件或欺诈，则系统可能会拒绝创建用户帐户并提示用户联系客户支持以解决问题。

2. 用例名称：用户登录

主要参与者：注册用户

前提条件：

* 用户必须具有互联网访问权限。
* 用户必须知道他们的正确登录凭据。

后置条件：

* 用户经过身份验证并获得访问其帐户的权限。

基本流程：

1. 用户启动应用程序并点击“登录”按钮。
2. 系统显示登录页面，提示用户输入用户名和密码。
3. 用户输入他们的用户名和密码，并点击“登录”按钮。
4. 系统验证用户的凭据并检查帐户是否处于活动状态。
5. 如果用户的凭据正确并且帐户处于活动状态，则系统登录用户并将其重定向到其帐户仪表板。
6. 如果用户的凭据不正确或帐户处于非活动状态，则系统向用户显示错误消息并提示他们重试。

备选流程：

* 如果用户忘记密码，可以在登录页面上单击“忘记密码”链接。系统会提示用户输入与其帐户相关联的电子邮件地址，然后向该电子邮件地址发送密码重置链接。用户可以使用链接重置密码。
* 如果用户没有帐户，则可以在登录页面上单击“注册”链接以创建新帐户。系统将提示用户提供必要的信息并创建其帐户。
* 如果用户在登录过程中遇到任何技术问题，例如服务器错误或连接问题，则系统将显示适当的错误消息并提示用户稍后重试。

异常流程：

* 如果用户多次输入其用户名或密码不正确，系统可能会锁定帐户以防止未经授权的访问。系统将提示用户联系客户支持解决问题。
* 如果系统在登录过程中遇到任何安全问题，例如可疑活动或未识别的登录尝试，则可能提示用户通过其他身份验证方法进行身份验证，例如双因素身份验证或生物识别身份验证。

3.3 总体结构

1. 前端技术：星画廊系统的前端采用HTML、CSS和JavaScript。HTML用于定义网页的结构，CSS用于定义网页的样式和排版，JavaScript用于实现网页的交互和动态效果。前端还包括钱包管理界面，用于管理数字钱包，包括数字凭证的购买和转移。
2. 后端技术：星画廊系统的后端采用Flask框架，使用Python编写。Flask框架是一个轻量级的Web应用程序框架，具有简单、灵活、易扩展等特点。后端我不懂，要补充星画廊系统的后端采用Flask框架，使用Python编写。Flask框架是一个轻量级的Web应用程序框架，具有简单、灵活、易扩展等特点。在Flask框架下，我们使用了SQLAlchemy ORM工具对数据进行管理和操作。SQLAlchemy提供了丰富的ORM功能，可以方便地进行数据库操作，同时还支持多种数据库后端。在数据库的操作过程中，我们使用了Flask框架的扩展库Flask-SQLAlchemy来简化ORM操作。

除了SQLAlchemy和Flask-SQLAlchemy之外，我们还使用了其他的Flask扩展库来加强星画廊系统的后端功能。例如我们使用了Flask-WTF扩展库来实现Web表单的验证和处理。扩展库是Flask生态系统中常用的组件，可以快速提升后端开发效率，并且与Flask框架良好地集成。

1. 数据库：

星画廊系统的数据库采用关系型数据库MySQL

### 表名：Round

### 用于记录游戏的轮数、每轮的选项、游戏结果，方便查询每一轮的游戏情况。

| **字段** | **类型** | **描述** |
| --- | --- | --- |
| RoundID | 主键，整型 | 轮次ID |
| GameID | 整型 | 所属游戏ID，记录某轮属于哪一局游戏 |
| Pic1ID | 整型 | 选择的第一幅图片ID |
| Pic2ID | 整型 | 选择的第二幅图片ID |
| Result | 布尔型 | 记录一轮游戏的结果 |

### 表名：Vote

### 用于记录用户的投票信息，方便统计每一轮的投票结果。

| **字段** | **类型** | **描述** |
| --- | --- | --- |
| RoundID | 主键，整型 | 轮次ID |
| UserID | 主键，整型 | 用户ID |
| Vote | 布尔型 | 记录玩家投票的结果，用于计算玩家得分。 |

### 表名：Score

### 用于记录用户得分情况，方便计算用户总分。

| **字段** | **类型** | **描述** |
| --- | --- | --- |
| RoundID | 主键，整型 | 轮次ID |
| UserID | 主键，整型 | 用户ID |
| Score | 整型 | 记录玩家投票的结果，用于计算玩家得分。 |

### 表名：User

### 用于存储用户信息，包括用户名、密码、电话号码、电子邮件地址以及用户参与过的游戏。

| **字段** | **类型** | **描述** |
| --- | --- | --- |
| UserID | 主键，整型 | 用户ID |
| UserName | 字符串 | 用户名 |
| Password | 字符串 | 密码 |
| Phone\_Number | 字符串 | 电话号码 |
| Email | 字符串 | 电子邮件地址 |
| GameID | 整型 | 记录用户参与过哪些游戏 |

### 表名：Picture

### 用于存储游戏中使用的图片地址，对应的图片在图床上，方便游戏页面显示图片。

| **字段** | **类型** | **描述** |
| --- | --- | --- |
| PicID | 主键，整型 | 图片ID |
| PicURL | 字符串 | 图片地址 |

### 表名：Game

### 用于存储游戏信息，包括游戏描述和原始图片，方便查询游戏信息、管理游戏内容。

| **字段** | **类型** | **描述** |
| --- | --- | --- |
| GameID | 主键，整型 | 本次游戏ID |
| UserID | 整型 | 参加游戏的用户ID |
| Description | 字符串 | 游戏描述 |
| OriginalPicID | 整型 | 记录游戏的原始图片 |

1. VACGAN图像生成算法：
2. 提取图像特征：
3. 输入层：输入图像的大小为224×224×3。
4. 卷积层1-2：两个卷积层都使用3×3的卷积核，步长为1，padding为1。第一个卷积层使用64个卷积核，第二个卷积层也使用64个卷积核。
5. 池化层1：使用2×2的池化核进行下采样，步长为2，生成大小为112×112×64的特征图。
6. 卷积层3-4：两个卷积层都使用3×3的卷积核，步长为1，padding为1。第一个卷积层使用128个卷积核，第二个卷积层也使用128个卷积核。
7. 池化层2：使用2×2的池化核进行下采样，步长为2，生成大小为56×56×128的特征图。
8. 卷积层5-7：三个卷积层都使用3×3的卷积核，步长为1，padding为1。第一个卷积层使用256个卷积核，第二个卷积层使用256个卷积核，第三个卷积层使用512个卷积核。
9. 池化层3：使用2×2的池化核进行下采样，步长为2，生成大小为28×28×512的特征图。
10. 卷积层8-10：三个卷积层都使用3×3的卷积核，步长为1，padding为1。第一个卷积层使用512个卷积核，第二个卷积层使用512个卷积核，第三个卷积层使用512个卷积核。
11. 池化层4：使用2×2的池化核进行下采样，步长为2，生成大小为7×7×512的特征图。
12. 全连接层1-3：三个全连接层分别包含4096个神经元。第一个全连接层的输入是一个大小为7×7×512=25088的向量，第二个和第三个全连接层都输入4096维向量。
13. 输出层：输出层有1000个神经元，对应1000个类别的概率。
14. 生成器（Generator）：VACGAN包含两个生成器，分别用于将一个域的图像转换为另一个域的图像。生成器由多个卷积层、反卷积层和残差块组成，其中每个残差块包含两个卷积层和一个跳跃连接（skip connection）。
15. 判别器（Discriminator）：VACGAN包含两个判别器，分别用于区分输入的图像是否属于其对应的域。判别器由多个卷积层组成，其中每个卷积层之后跟着一个Batch Normalization层和一个LeakyReLU激活函数。判别器的输出是输入图像属于其对应域的概率。
16. 恒等损失（Identic Loss）：通过计算生成图像与真实图像之间的差异，来指导生成器网络的优化。具体来说，在生成对抗网络（GAN）等模型中，恒等损失可以用来保持生成器网络在图像生成过程中，尽可能地保留原始图像的信息，防止生成出的图像过于模糊或失真。
17. 循环一致性损失（Cycle Consistency Loss）：VACGAN引入了循环一致性损失，以保证图像转换的一致性。具体来说，它包含两个损失项：正向循环一致性损失和反向循环一致性损失。正向循环一致性损失指的是将一个域的图像转换为另一个域的图像，再将转换后的图像转换回原来的域，应该得到原始图像。反向循环一致性损失与正向循环一致性损失类似，只是从另一个域开始。
18. 对抗损失（Adversarial Loss）：对抗损失是VACGAN的核心损失，它用于训练生成器和判别器。对于生成器，它需要生成足够逼真的图像以欺骗判别器；对于判别器，它需要识别出生成器生成的假图像，并将其与真实图像区分开来。

3.4功能分配

1. 前端功能：
   1. 用户登录和注册：实现用户注册和登录功能，并进行身份验证，保障系统安全。
   2. 首页：展示网站的基本信息，包括网站名称、网站简介、网站主题等。还可以放置一些关于自闭症的相关信息和链接，以便用户了解更多信息。
   3. 游戏介绍页面：介绍游戏的规则、奖励和背景，让用户了解游戏的基本情况。
   4. 画廊页面：展示已生成的NFT作品，并提供购买链接。可以按照不同的分类方式展示作品，如画风、作者、价格等。
   5. 自闭症科普页面：介绍自闭症的基本情况、症状和社会性困难，帮助用户了解自闭症相关知识。还可以提供相关资源和链接，如支持组织、医疗机构等。
   6. 游戏页面：包括游戏主界面、游戏规则、倒计时、投票按钮、用户分数和供选择的图片等。游戏页面需要能够根据不同的轮次动态更新，展示不同的图片和投票结果。
   7. 异常页面：显示错误信息，例如网络异常、数据库连接异常等。
2. 后端功能（待补充）：
   1. 用户认证和管理功能：提供用户注册、登录和注销等功能，并能够管理用户信息和权限。可以使用 Flask-WTF 实现表单验证，确保用户输入的信息符合规范。用户信息和权限的管理可以通过 Flask-SQLAlchemy 实现对数据库的操作，包括增、删、改、查等。
   2. 游戏数据处理功能：处理游戏中的图片、投票结果和用户得分等数据，并存储到数据库中。还需要实现游戏规则的逻辑判断，如判断每局游戏的胜者和输赢等。在 Flask 中，可以通过 Flask-SQLAlchemy 来处理数据库操作，包括数据的增删改查、事务管理等。为了实现游戏规则的逻辑判断，可以使用 Python 编程语言自带的逻辑判断语句、循环语句等来实现，例如 if、else、while、for 等语句。
   3. 数据库管理功能：设计和管理数据库，包括用户信息、游戏数据和作品等数据表的创建、维护和备份等操作。在 Flask 中，可以使用了 Flask-SQLAlchemy 来处理数据库的创建和维护。Flask-SQLAlchemy 是一个针对 SQLAlchemy 的 Flask 扩展，提供了 SQL 数据库的抽象层，可以方便地进行数据库操作。同时，为了确保数据的安全性，可以使用 Flask-Migrate 来进行数据库迁移管理。
   4. 数据传输设计和实现（前端到后端的数据传输）：后端API接收请求，对请求进行解析，根据请求的参数和路径，进行相应的数据处理。从数据库中获取所需数据，并将处理结果封装成JSON格式的数据返回给前端。前端接收到后端返回的数据，将数据展示在页面上。
   5. AI图像生成算法：输入原始图或获胜图像，AI算法加工成更具艺术美感的2张图像，在游戏下一轮中供玩家选择。Pix2pix是一种基于深度学习的图像生成算法，它采用对抗生成网络（GAN）结构，通过将输入图像与目标图像配对来训练生成器和判别器网络。生成器网络将输入图像转换为目标图像，而判别器网络则尝试将生成器产生的图像与真实目标图像区分开来。在星画廊系统中，使用的是基于Pix2pix的图像生成算法，它能够将原始图像或获胜图像加工成更具艺术美感的两张图像，供下一轮游戏中的玩家选择。
   6. API接口：通过API将图片输入算法，再将算法生成的2张图片输出到数据库。
   7. 安全和性能优化功能：保障网站的安全性和性能优化，如使用HTTPS协议保证数据传输的安全性、使用缓存和压缩技术优化网站的响应速度等。
3. 数据库功能：
   1. 用户数据管理：数据库需要存储用户的个人信息，包括用户名、密码、邮箱、手机号码等。当用户进行注册、登录、修改个人信息等操作时，数据库需要进行相关的数据处理和存储。
   2. 游戏数据管理：数据库需要存储游戏相关的数据，包括游戏轮次、每轮游戏的两张图片、用户投票记录、每个用户的得分、排名等信息。当用户进行投票、游戏结束等操作时，数据库需要进行相关的数据处理和存储。
   3. NFT数据管理：数据库需要存储生成的NFT作品信息，包括作品名称、作者、价格、描述、购买链接等。当新的NFT作品铸造后，数据库需要进行相关的数据处理和存储。
   4. 自闭症科普数据管理：数据库需要存储自闭症相关的知识和资源，包括自闭症基本情况、症状、社会性困难、支持组织、医疗机构等信息。当新的资源添加或修改时，数据库需要进行相关的数据处理和存储。
   5. 数据库安全管理：数据库需要进行用户身份认证和权限管理，以确保用户数据和系统安全。在数据库设计中需要考虑加密、防注入等安全措施。
   6. 数据库性能管理：数据库需要优化查询速度、存储空间等性能方面的问题，以保证系统的高效性。在数据库设计中需要考虑索引、分表、分区等性能优化措施。
4. VACGAN图像生成算法：

VACGAN算法用于生成游戏中的图片选项，根据玩家选择的方向将图像变得更加美观、有艺术感。其具体功能如下：

* 1. 接收前端传来的胜出图片，并将其作为输入数据。
  2. 根据输入数据，生成两张新的图片作为下一轮游戏的选项，同时保存这两张新图片到图像数据库中。
  3. 将生成的两张图片的URL地址返回给后端，以便前端进行展示和用户选择。
  4. 在项目中，VACGAN算法被封装在API中，用于和后端进行数据交互。后端将传来的胜出图片的URL地址通过API传递给算法，算法再将生成的两张新图片的URL地址通过API返回给后端，后端再将其传递给前端进行展示。