四 川 大 学

网络空间安全学院

身份鉴别系统实践报告

|  |  |
| --- | --- |
| 姓 名： |  |
| 学 号： |  |
| 课程名称： | 网络空间安全技术 |
| 任课教师： | 方勇 |

二〇二四年 十二月 九日

**《网络空间安全技术》**

**课程实践报告撰写评分标准**

|  |  |  |  |
| --- | --- | --- | --- |
| **一级大纲** | **关键点** | **评价要素** | **评分** |
| **引言与背景（10分）** | **问题描述**  **（6分）** | 是否准确描述了身份鉴别系统的相关背景和必要性。 |  |
| **目标明确**  **（4分）** | 报告是否清晰阐述了实践的目的及预期结果。 |  |
| **设计与实现**  **（30分）** | **系统设计**  **（15分）** | 是否详细描述了身份鉴别系统的设计方案，包括流程图、架构图等。设计是否包含了多种鉴别因素且具有合理性和可扩展性。 |  |
| **技术选型**  **（5分）** | 是否清楚解释了所选技术、框架或工具的原因。 |  |
| **实现过程**  **（10分）** | 是否全面、准确描述了系统实现的过程，步骤清晰且有逻辑。 |  |
| **安全性分析**  **（15分）** | **-** | 是否清晰阐述了所采用的身份鉴别机制的原理及其安全性优势。 |  |
| **实验与测试（20分）** | **测试方案**  **(10分)** | 是否提供了详细的测试方案，包括测试用例的覆盖率和结果分析。 |  |
| **结果分析**  **(10分)** | 测试数据是否完整，并对测试结果进行了充分的分析，验证了系统的安全性和功能性。 |  |
| **总结与展望**  **(10分)** | **总结**  **(5分)** | 对实践过程中的发现与收获进行了总结。 |  |
| **改进与展望**  **(5分)** | 是否对未来的改进提出了合理的建议，并对系统的发展方向进行了展望。 |  |
| **文档规范性**  **(15分)** | **写作能力**  **（5分）** | 报告是否结构清晰、逻辑严谨。语言是否流畅、专业，是否符合学术报告规范。 |  |
| **学术不端检测**  **（10分）** | 报告提交前需先进行学术不端检测，检测报告同实践报告一并提交，且查重需低于15%。 |  |
| **总 分** | | |  |

**一、引言**

在当今网络安全日益重要的背景下，身份鉴别系统在保护用户隐私和防止未授权访问方面起着关键作用。本项目旨在实现一个基于Flask框架的身份鉴别系统，支持用户注册、登录、二次验证、重置密码以及安全管理。系统主要采用Python作为开发语言，通过整合TOTP（Time-based One-Time Password）技术，加强了系统的安全性。

**二、项目目标**

项目的目标是设计并实现一个能够进行用户身份验证、数据保护和防止欺诈行为的身份鉴别系统。具体目标包括：

* 提供基本的用户注册、登录和重置密码功能。
* 实现基于TOTP的二次身份验证。
* 支持多层安全策略，确保系统的安全性。

**三、技术栈**

项目所使用的主要技术和工具包括：

* Flask：一个轻量级的Web框架，用于构建系统的核心架构。
* SQLAlchemy：Flask中的ORM（对象关系映射）工具，用于数据库交互。
* bcrypt：用于密码加密存储。
* PyOTP：用于生成和验证TOTP验证码。
* SQLite：作为数据库管理系统，存储用户信息。
* Docker：容器化技术，用于简化部署和环境配置。

**四、系统架构**

系统架构包括前端、后端和数据库三个主要部分。后端部分使用Flask框架提供API接口，前端部分负责用户交互。数据库部分存储用户信息，并使用SQLAlchemy进行数据操作。

前端 (HTML/CSS/JS)

|

后端 (Flask)

|

数据库 (SQLite)

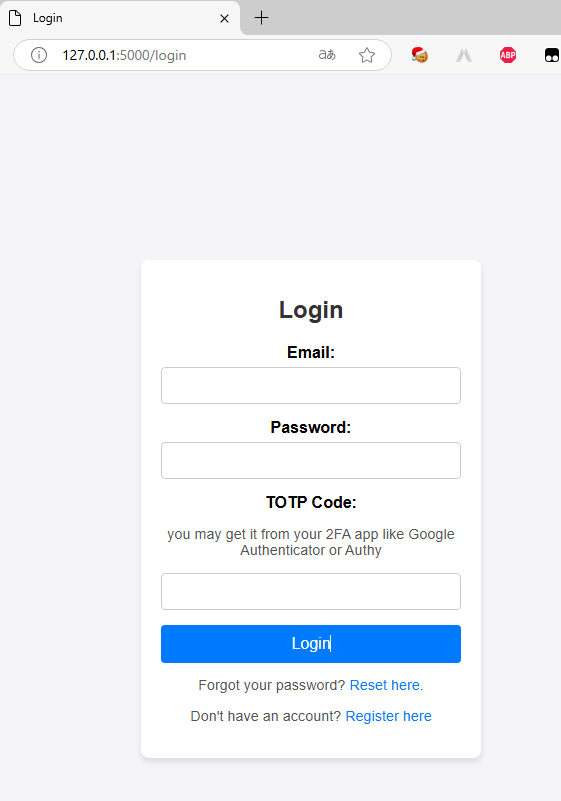
**五、开发过程**

（一）环境搭建

首先，我们在本地环境中创建了一个虚拟环境，并安装了所需的依赖。然后配置了Flask应用的基础结构，包括路由、视图函数和数据库模型。

（二）用户注册与登录模块

在用户注册过程中，我们要求用户提供用户名、电子邮件和密码，密码通过bcrypt进行加密存储。在用户登录时，系统通过比对存储的加密密码来验证用户的身份。



（三）安全机制与二次验证

为了进一步增强安全性，我们实现了基于时间的一次性密码（TOTP）。用户在登录后需要扫描二维码，通过手机应用（如Google Authenticator）生成TOTP验证码，并输入该验证码进行验证。若验证通过，用户即完成登录。

QR 代码

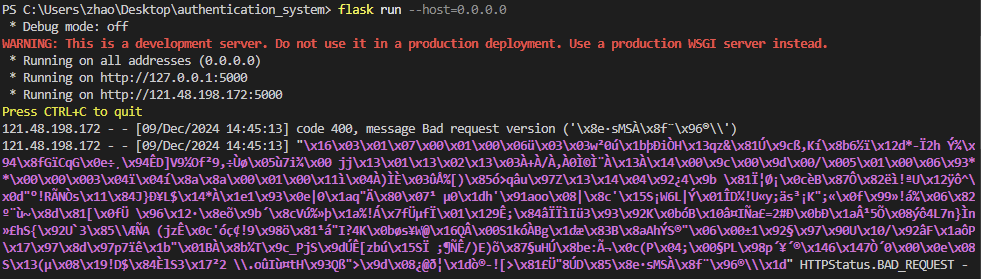
描述已自动生成

**六、测试与评估**

测试过程中，我们通过单元测试和集成测试来验证各个模块的正确性。特别是安全性测试，确保密码存储的加密方式安全可靠，TOTP验证码的生成和验证功能正常。

1. **遇到的问题与解决方案**

数据库表结构问题：在项目初期，遇到数据库表未正确创建的问题。通过使用db.create\_all()重新初始化数据库解决了这一问题。



TOTP配置错误：在生成TOTP时，因缺少正确的密钥导致验证码无法生成。通过查阅PyOTP文档，正确配置密钥后解决了此问题。

**八、总结与反思**

本项目成功实现了一个基础的身份鉴别系统，具备了用户注册、登录、TOTP二次验证等功能。通过这个实践，我们不仅加深了对Web安全技术的理解，还提高了对Flask框架、数据库操作和安全加密技术的应用能力。

**九、未来改进建议和系统发展方向**

（一）用户体验优化

虽然当前系统已经有了基础的功能，但在实际使用中，用户体验（UX）依然可以得到进一步优化。

1.优化注册流程

用户注册流程可能需要进一步简化和优化。比如：

* 自动化错误提示：对于用户输入的电子邮件或用户名，如果已存在，可以即时反馈，避免用户提交表单后再看到错误信息。
* 分步注册：将注册流程分为多个步骤，逐步引导用户完成注册，减少一次性表单的复杂度，提升用户的注册体验。

2.界面友好性改进

* 响应式设计：目前的界面可以在桌面端正常工作，但对于移动端设备，可能存在布局不佳的问题。可以通过CSS媒体查询进行优化，使其适应不同屏幕尺寸。
* 二维码生成优化：二维码的显示方式可以进一步优化，比如增加适应不同设备的二维码大小、提示文本等，使用户更容易扫描并成功验证。

3. 自定义主题和个性化设置

* 主题选择：让用户可以选择不同的主题（浅色/深色模式等），提高个性化体验。
* 语言支持：多语言支持可以吸引更多不同地区的用户，特别是对于全球化产品而言。

（二）系统性能优化

随着系统用户数的增多，可能会遇到性能瓶颈。以下是一些性能优化的方向：

1.数据库优化

* 数据库索引：对于常用的查询，如用户的电子邮件、用户名等，可以在数据库中为这些字段添加索引，加快查询速度。
* 数据库分区：当用户数量极大时，数据库可能需要做分区处理，将数据分布到多个表或数据库实例中，避免单一数据库负载过高。

2. 缓存机制

* 验证码缓存：生成TOTP验证码时，可以使用Redis等缓存机制来存储验证码，避免每次请求都对数据库进行大量访问，减少延迟。
* 常见查询缓存：对于频繁访问的数据（如用户基本信息），可以利用缓存机制减轻数据库负担，提高响应速度。

3.负载均衡

当系统的流量激增时，可以通过负载均衡技术，将请求分配到多个服务器上，以提高系统的可用性和性能。

参考文献

[1]Flask Documentation. [OL] https://flask.palletsprojects.com/ [2024-12-09].

[2]SQLAlchemy Documentation. [OL] https://docs.sqlalchemy.org/ [2024-12-09]

[3]PyOTP Documentation. [OL] https://pypi.org/project/PyOTP/ [2024-12-09]