MUSIC PORTAL

SECURE SOFTWARE PLAN

Version *1.0*

08/01/2024

VERSION HISTORY

Version #	Implemented By	Revision Date	Approved By	Approval Date	Reason
1.0	*Zhivko Georgiev*	*08/01/2024*			Initial Design Definition draft

TABLE OF CONTENTS

1 PROJECT SUMMARY

1.1 PROJECT GOALS

Develop a secure and user-friendly MusicPortal web application that allows users to browse, purchase, and download albums. The application should provide a seamless and enjoyable experience for both registered and unregistered users, ensuring the confidentiality and integrity of user data and transactions.

1.2 PROJECT SCOPE

In Scope:

- Secure user authentication and authorization.
- Album management, including storage, retrieval, and searching.
- User album purchase with credit accounts and download functionality.
- User account management with credit recharge.
- Implementation of security measures: secure coding, HTTPS, encryption.
- User-friendly interface for album browsing and representation.

Out of Scope:

- External payment gateway integration.
- User profile customization.
- Social features and interactions.
- Advanced search functionality.

2 TESTING OBJECTIVES

The primary objective of security testing for the MusicPortal application is to ensure the robustness and resilience of the system against potential security threats. The key objectives include security testing of common security vulnerabilities such as SQL injection, Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), and others, working authorization and authentitacation.

3 SECURE REQUIREMENTS

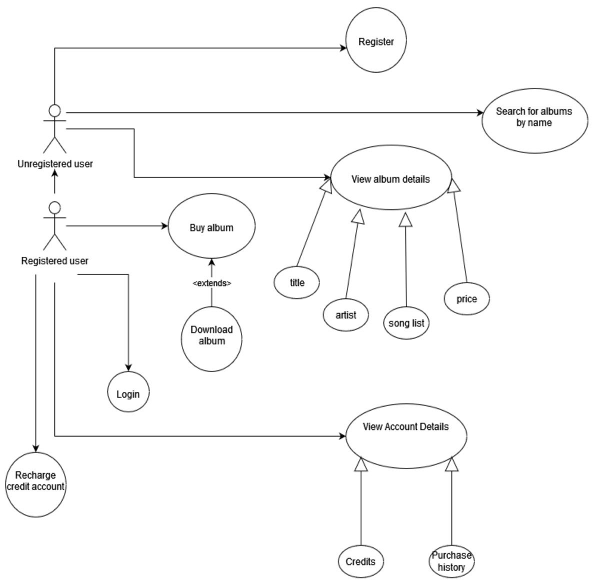
This section outlines the system functional and nonfunctional requirements.

3.1 USE-CASES

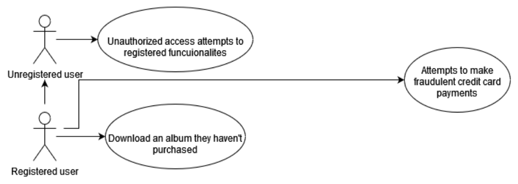
3.2 ABUSE CASES

3.3 MISUSE CASES

Use-Case

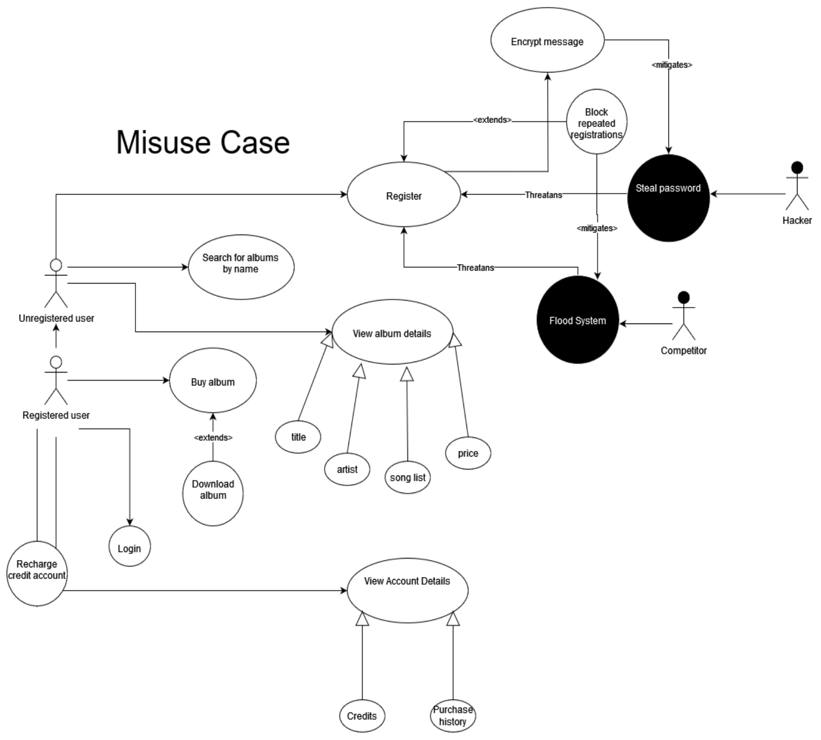


Abuse Cases



Unauthorized access attempts to registered funcuionalites
Unregistered user
Attempts to make fraudulent credit card payments
Download an album they haven't purchased
Registered user

Misuse Case



THREAT ASSESSMENT

3.4 ASSETS

- **Album Details**
- **User Account**
- **Credit Account**
- **Search Functionality**
- **Download/Buy Links**

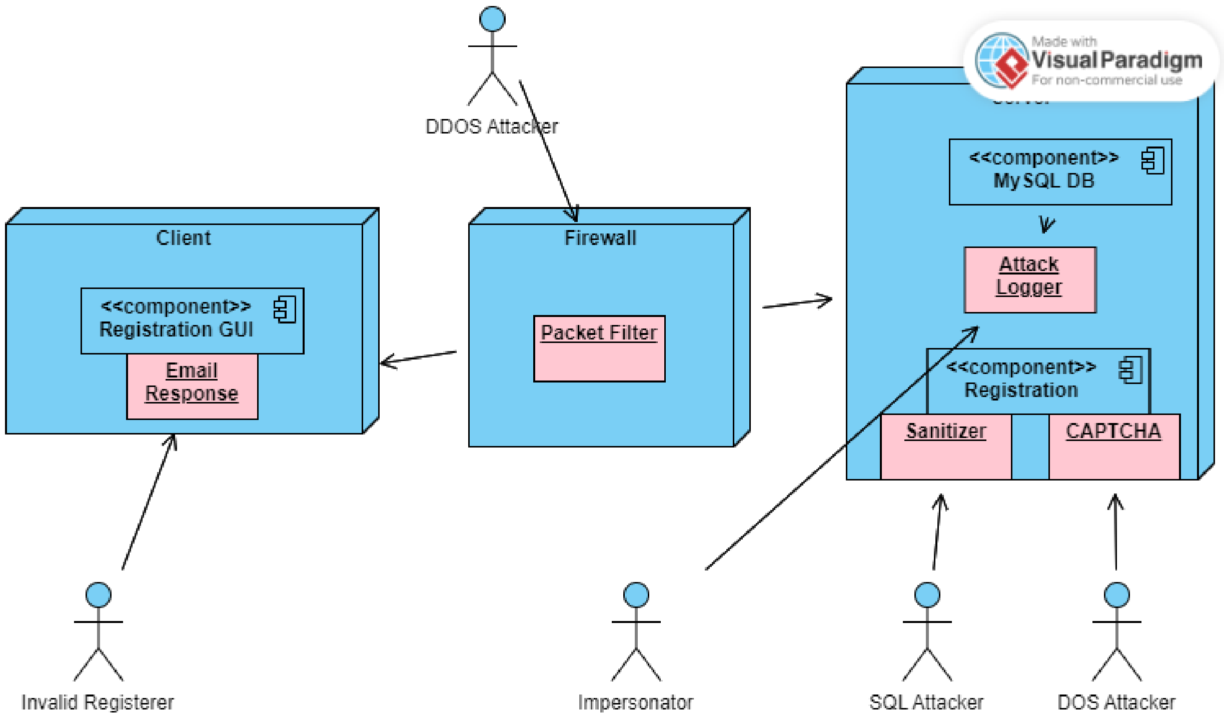
3.5 THREATS MODELING

1. **Unauthorized Access**: Malicious users gaining unauthorized access to user accounts, compromising sensitive information. Impact: Unauthorized viewing, modification, or deletion of user data. Countermeasures: Strong authentication mechanisms. Regularly audit and monitor user account activities for anomalies.
2. **SQL Injection**: Injection of malicious SQL queries to manipulate or extract data from the database. Impact: Data breaches, unauthorized access to database contents. Countermeasures: Use parameterized queries and prepared statements. Input validation and sanitization.
3. **Cross-Site Scripting (XSS):** Injection of malicious scripts into web pages viewed by other users. Impact: Session hijacking, defacement of web pages. Countermeasures: Input validation for user-generated content. Content Security Policy (CSP) implementation.
4. **Cross-Site Request Forgery (CSRF):** Forcing users to perform actions without their consent. Impact: Unauthorized actions on behalf of authenticated users. Countermeasures: Anti-CSRF tokens in forms. Strict enforcement of same-origin policy.

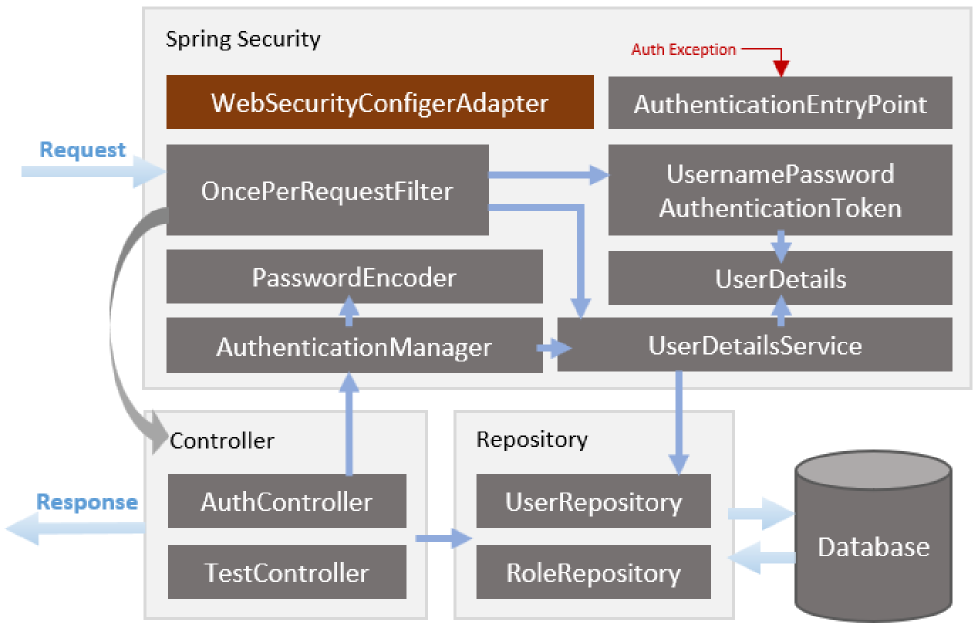
4 SECURE DESIGN

This section outlines the system and hardware architecture design of the system that is being built.

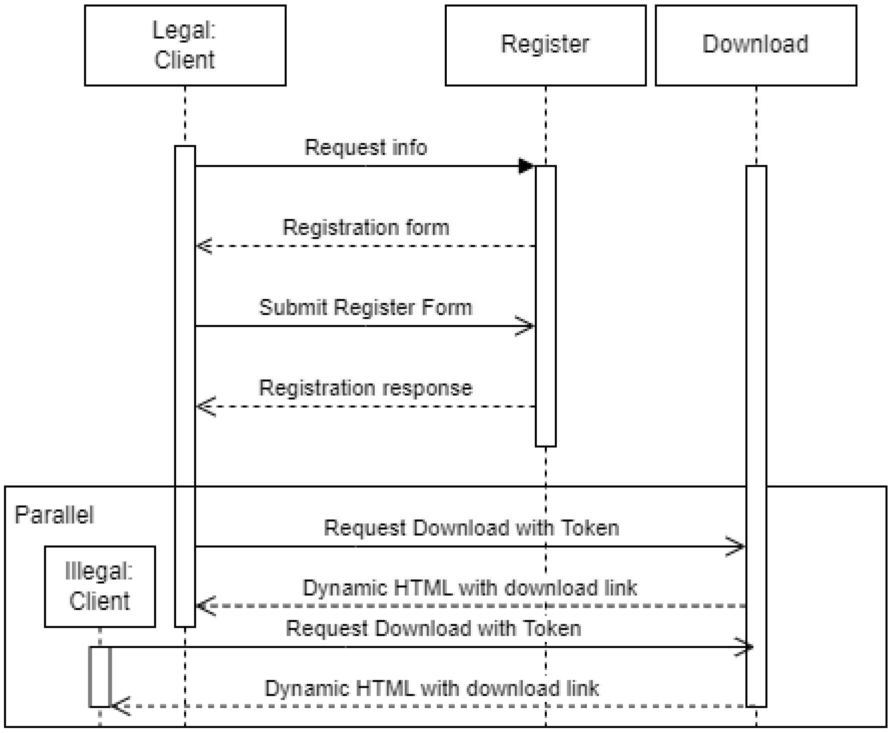
4.1 PHYSICAL SECURITY ARCHITECTURE



4.2 COMPONENT / SERVICE SECURITY ARCHITECTURE



4.3 LOGICAL SECURITY ARCHITECTURE



5 SECURITY DEVELOPMENT AND TESTING

5.1 STATIC ANALYSIS

Utilize automated source code analysis tools listed in Appendix C to perform a comprehensive analysis of the MusicPortal source code. Tools such as Find-Bugs will be employed to identify inconsistencies, errors, and potential vulnerabilities. Code review will allow for an in-depth investigation of risky code sections and ensures a holistic assessment of the source code. SAST instruments assess the code and identify issues and give thorough reports with recommendations.

5.2 DYNAMIC TESTING

DAST instruments scan the code during its execution. Its goal is to find issues in the work of the applications by employing different types of attacks. DAST usually uses fuzzing, which is an automated method for testing, sending invalid, unexpected data to the application.

5.3 PENETRATION TESTING

Penetration testing or Ethical Hacking is a process for simulating real attacks towards the application and its data. It helps measure the security of the application in a controlled environment. No vulnerabilities were found during testing.

6 TEST RESULTS

Main functionalities of the business logic have been tested with unit tests and manual testing. A high percentage of code coverage has been achieved, which is satisfactory. The application uses the latest versions of spring security and logging which removes the possibility of dependency vulnerability.

7 APPENDIX C. TOOLSET

Tool	Open source
Source-code analyzers	Rough Audi t ing Tool for Security, Find-Bugs
Web application scanners	WebScarab, Burp Suite
Database scanners	Scuba, SQLrecon
Binary analysis tools	Fx-Cop, BugScam
Runtime analysis tools	CLR Profiler, .NETMon
Configuration analysis tools	SSLDigger, Security Configuration Tool set
Proxy servers tools	Paros, Fiddler
Miscellaneous tools	FireFox toolbar, SiteDigger, JUnit