# Developer Log: Implementing Secure Game Master Authentication

## Introduction

In this project, we aimed to develop a secure and robust authentication system for Game Masters in our application. This log documents the various steps, decisions, and implementations made to achieve this goal.

## Entity Models

**StoredOTP Entity:** We started by defining the StoredOTP entity, which is responsible for storing OTPs along with their expiration timestamps. This entity ensures that each OTP is unique and time-bound, enhancing the security of the verification process.

**GameMaster Entity:** The GameMaster entity stores information about the Game Masters, including their email, name, and a boolean flag indicating whether they are a Game Master. The email field is unique, ensuring that no two Game Masters can share the same email address.

## Repository Interfaces

**GameMasterRepository:** The GameMasterRepository interface extends JpaRepository and includes custom query methods to find Game Masters by their email. This repository handles all database interactions related to the GameMaster entity.

**OtpRepository:** Similarly, the OtpRepository interface extends JpaRepository and includes methods to find StoredOTP entries by email. This repository manages the storage and retrieval of OTPs from the database.

## Service Implementations

**Code Generation Service:** The CodeGeneratorServiceImpl is responsible for generating random OTPs. It uses a secure random number generator to create OTPs of specified lengths, ensuring that the codes are difficult to predict.

**Code Storage Service:** The CodeStorageServiceImpl handles the storage and retrieval of OTPs. It checks if an OTP already exists for a user, updates it if necessary, and saves the OTP with an expiration timestamp. The service also retrieves stored OTPs and deletes expired ones, maintaining the integrity and security of the OTP system.

**Custom User Details Service:** The CustomUserDetailsService implements UserDetailsService to load user-specific data during authentication. It fetches Game Masters by email and creates a UserDetails object, which Spring Security uses for authentication and authorization.

**Email Service:** The EmailServiceImpl manages the sending of OTPs via email. It generates the OTP, stores it, and then sends an email to the Game Master with the OTP and its expiration time. This service ensures that OTPs are delivered securely and promptly.

**Verification Service:** The VerificationServiceImpl verifies the OTP entered by the user. It retrieves the stored OTP for a given email, checks if it matches the entered OTP, and verifies if the OTP is still valid. This service is crucial for ensuring that only valid OTPs are accepted for authentication.

## Controller

**GameMasterController:** The GameMasterController handles HTTP requests related to Game Master management and OTP verification. It includes endpoints for adding, finding, and deleting Game Masters, as well as generating and verifying OTPs. The controller interacts with the various services to provide a seamless and secure user experience.

## Conclusion

The implementation of a secure Game Master authentication system involved creating and integrating multiple components, including entity models, repositories, services, and controllers. This multi-layered approach ensures that the authentication process is robust, secure, and user-friendly. The use of OTPs, secure email transmission, and JWT tokens adds significant security measures to protect sensitive data and authenticate users effectively. The system's success in the project demo highlighted its effectiveness in providing a secure and reliable authentication mechanism for Game Masters.