These are the REST API for the ATM Monitoring Application using **Java Spring Boot**. For **Authorization**, **transaction breakdown**, **failure logs**, and **camera footage download**.

Here I have assumed that you already have Spring Boot set up and the necessary dependencies (like Spring Security, JWT for authentication, etc.) in place.

**Project Setup**

You should have the following dependencies in your pom.xml:

<dependencies>

<!-- Spring Boot Dependencies -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-security</artifactId>

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt</artifactId>

<version>0.9.1</version>

</dependency>

<!-- Lombok for simplified Java classes -->

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<scope>provided</scope>

</dependency>

</dependencies>

**1. Authorization**

First, create a JWT-based authentication system to authorize API calls.

**JWT Token Generation (AuthService.java)**

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import org.springframework.security.core.Authentication;

import org.springframework.stereotype.Service;

import java.util.Date;

@Service

public class AuthService {

private final String SECRET\_KEY = "my\_secret\_key";

private final int EXPIRATION\_TIME = 3600000; // 1 hour

public String generateToken(Authentication authentication) {

String username = authentication.getName();

return Jwts.builder()

.setSubject(username)

.setIssuedAt(new Date())

.setExpiration(new Date(System.currentTimeMillis() + EXPIRATION\_TIME))

.signWith(SignatureAlgorithm.HS512, SECRET\_KEY)

.compact();

}

}

**AuthenticationController.java**

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

import org.springframework.security.core.Authentication;

import org.springframework.web.bind.annotation.\*;

@RestController

@RequestMapping("/api/auth")

public class AuthenticationController {

private final AuthenticationManager authenticationManager;

private final AuthService authService;

public AuthenticationController(AuthenticationManager authenticationManager, AuthService authService) {

this.authenticationManager = authenticationManager;

this.authService = authService;

}

@PostMapping("/token")

public String login(@RequestBody LoginRequest loginRequest) {

Authentication authentication = authenticationManager.authenticate(

new UsernamePasswordAuthenticationToken(loginRequest.getUsername(), loginRequest.getPassword()));

return authService.generateToken(authentication);

}

}

**LoginRequest.java**

public class LoginRequest {

private String username;

private String password;

// Getters and setters

}

**JWT Filter for Token Validation**

Create a JWT filter to verify tokens for incoming requests.

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import org.springframework.security.core.context.SecurityContextHolder;

import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.security.core.userdetails.UserDetailsService;

import org.springframework.stereotype.Component;

import org.springframework.web.filter.OncePerRequestFilter;

import javax.servlet.FilterChain;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import java.io.IOException;

@Component

public class JwtFilter extends OncePerRequestFilter {

private final String SECRET\_KEY = "my\_secret\_key";

private final UserDetailsService userDetailsService;

public JwtFilter(UserDetailsService userDetailsService) {

this.userDetailsService = userDetailsService;

}

@Override

protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain filterChain) throws IOException {

String token = request.getHeader("Authorization");

if (token != null && token.startsWith("Bearer ")) {

String jwt = token.substring(7);

String username = Jwts.parser()

.setSigningKey(SECRET\_KEY)

.parseClaimsJws(jwt)

.getBody()

.getSubject();

UserDetails userDetails = userDetailsService.loadUserByUsername(username);

SecurityContextHolder.getContext().setAuthentication(

new UsernamePasswordAuthenticationToken(userDetails, null, userDetails.getAuthorities())

);

}

filterChain.doFilter(request, response);

}

}

**2. Transaction Breakdown API**

**ATMTransactionController.java**

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.RestController;

@RestController

@RequestMapping("/api/atm")

public class ATMTransactionController {

@GetMapping("/transactions/breakdown")

public TransactionBreakdown getTransactionBreakdown(@RequestParam String from, @RequestParam String to) {

// Assuming from and to are date ranges in String format

// Query the database or logs to get the transactions summary

TransactionBreakdown breakdown = new TransactionBreakdown();

breakdown.setDepositCount(30); // Mock value

breakdown.setCashWithdrawalCount(120); // Mock value

breakdown.setBalanceInquiryCount(40); // Mock value

return breakdown;

}

}

**TransactionBreakdown.java**

public class TransactionBreakdown {

private int depositCount;

private int cashWithdrawalCount;

private int balanceInquiryCount;

// Getters and setters

}

**3. Failure Logs API**

**ATMFailureController.java**

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.RestController;

import java.util.List;

@RestController

@RequestMapping("/api/atm")

public class ATMFailureController {

@GetMapping("/failures")

public List<FailureLog> getFailures(@RequestParam String from, @RequestParam String to) {

// Query the database or logs to fetch the failures between the time ranges

return List.of(

new FailureLog("FAIL001", "Card Reader Error", "Cash Withdrawal", "Card reader failed to read card"),

new FailureLog("FAIL002", "Cash Dispenser Jam", "Deposit", "Cash dispenser jammed during transaction")

);

}

}

**FailureLog.java**

public class FailureLog {

private String failureId;

private String failureType;

private String transactionType;

private String context;

public FailureLog(String failureId, String failureType, String transactionType, String context) {

this.failureId = failureId;

this.failureType = failureType;

this.transactionType = transactionType;

this.context = context;

}

// Getters and setters

}

**4. Download Camera Images/Videos API**

**ATMCameraController.java**

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.RestController;

@RestController

@RequestMapping("/api/atm")

public class ATMCameraController {

@GetMapping("/camera-logs/download")

public CameraLogResponse downloadCameraLogs(@RequestParam String from, @RequestParam String to) {

// Query the database or logs to fetch the video footage URLs between the time ranges

return new CameraLogResponse("https://bank-system.com/downloads/footage/2024-10-22\_10-12.mp4");

}

}

**CameraLogResponse.java**

public class CameraLogResponse {

private String downloadUrl;

public CameraLogResponse(String downloadUrl) {

this.downloadUrl = downloadUrl;

}

// Getter and Setter

}

**Security Configuration**

You need to configure Spring Security to apply JWT filtering and handle authentication.

**SecurityConfig.java**

import org.springframework.context.annotation.Bean;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.config.annotation.authentication.configuration.AuthenticationConfiguration;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

import org.springframework.security.web.SecurityFilterChain;

import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;

@EnableWebSecurity

public class SecurityConfig {

private final JwtFilter jwtFilter;

public SecurityConfig(JwtFilter jwtFilter) {

this.jwtFilter = jwtFilter;

}

@Bean

public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

http.csrf().disable()

.authorizeRequests()

.antMatchers("/api/auth/\*\*").permitAll()

.anyRequest().authenticated()

.and()

.addFilterBefore(jwtFilter, UsernamePasswordAuthenticationFilter.class);

return http.build();

}

@Bean

public AuthenticationManager authenticationManager(AuthenticationConfiguration authenticationConfiguration) throws Exception {

return authenticationConfiguration.getAuthenticationManager();

}

}

**Summary of Endpoints**

* POST /api/auth/token: Generate a JWT token for authenticated requests.
* GET /api/atm/customers/last-24-hours: Get total customers in the last 24 hours.
* GET /api/atm/transactions/breakdown: Get transaction breakdown by type.
* GET /api/atm/failures: Get a list of ATM failures and system/device errors.
* GET /api/atm/camera-logs/download: Download camera logs by time range.