Security Requirements Generated (GEN1+GEN2)

1. Implement role-based access control to ensure students, professors, course coordinators, and technical staff can only access their respective functionalities. – Access Management

2. Enforce strict session management to prevent session hijacking or fixation attacks. – Web Application Security

3. Ensure all API endpoints validate user permissions before processing requests. – Secure Coding

4. Use parameterized queries or ORM frameworks to prevent SQL injection in all database interactions. – Input Validation

5. Sanitize all user inputs to prevent cross-site scripting (XSS) attacks in messages and notes. – Client-Side Security

6. Implement content security policies (CSP) to mitigate injection risks in dynamic content. – Secure Design

7. Encrypt all sensitive data, such as student and professor messages, both in transit and at rest. – Data Protection

8. Ensure that classroom booking and exam session data is stored securely and cannot be tampered with. – Secure Coding

9. Use secure protocols (e.g., HTTPS) for all communications between the client and server. – Web Application Security

10. Conduct regular security audits to identify and fix insecure design flaws in the system. – Secure Design

11. Implement secure defaults for all configurations, such as disabling unnecessary services. – Secure Coding

12. Ensure that the system architecture follows the principle of least privilege. – Access Management

13. Implement multi-factor authentication (MFA) for all administrative accounts. – Authentication

14. Enforce strong password policies for all user accounts, including complexity and expiration requirements. – Web Application Security - Authentication

15. Monitor and log all authentication attempts to detect and respond to brute force attacks. – Data Protection

16. Ensure all third-party libraries and frameworks used in the application are up-to-date and free from known vulnerabilities. – Secure Coding

17. Implement automated dependency scanning to detect and mitigate outdated or vulnerable components. – Secure Design

18. Maintain an inventory of all software components and their versions for quick vulnerability assessment. – Data Protection

19. Implement secure session management to prevent unauthorized access to user accounts. – Web Application Security

20. Ensure all authentication tokens are securely generated, stored, and invalidated after logout. – Authentication

21. Use secure cookies with HttpOnly and Secure flags to protect session data. – Client-Side Security

22. Implement data integrity checks to ensure that class schedules and bookings cannot be altered maliciously. – Data Protection

23. Log all changes to critical data, such as class schedules and exam bookings, for audit purposes. – Secure Coding

24. Ensure that all data backups are encrypted and stored securely to prevent tampering. – Secure Design

25. Monitor and log all security events, such as failed login attempts and access control violations. – Data Protection

26. Implement rate limiting to prevent brute force attacks on authentication endpoints. – Web Application Security

27. Ensure that all logs are stored securely and are accessible only to authorized personnel. – Access Management

28. Provide clear and actionable error messages to users without exposing sensitive system information. – Client-Side Security

29. Implement proper exception handling to prevent information leakage in case of system failures. – Secure Coding

30. Ensure that all APIs return appropriate HTTP status codes and error messages. – Web Application Security

31. The system must ensure uptime of at least 99.9% during academic hours (8 AM to 8 PM) to guarantee continuous access to schedules and classroom information. – Availability

32. The platform must implement load balancing to handle peak traffic during exam registration periods without degradation in performance. – Availability

33. The system must automatically recover from failures within 5 minutes to minimize disruption to users accessing critical features like class scheduling. – Availability

34. All user inputs, including exam booking notes and messages to professors, must be sanitized to prevent SQL injection attacks. – Input Validation

35. Form inputs for class scheduling by course coordinators must be validated to reject malformed data (e.g., invalid time formats or non-existent classrooms). – Input Validation

36. User-generated content, such as messages between students and professors, must be filtered to prevent cross-site scripting (XSS) attacks. – Input Validation

37. Implement multi-factor authentication (MFA) for professors and course coordinators to protect sensitive actions like modifying class schedules. – Web Application Security - Authentication

38. Session tokens must expire after 30 minutes of inactivity to prevent unauthorized access to student or professor accounts. – Web Application Security - Authentication

39. Rate limiting must be enforced on login attempts to prevent brute-force attacks against student and faculty accounts. – Web Application Security - Authentication

Total: GEN1=30, GEN2=9, Total=39

Counting per category (total = gen1 + gen2):

- Access Management: total=3 (gen1=3, gen2=0)
- Authentication: total=2 (gen1=2, gen2=0)
- Availability: total=3 (gen1=0, gen2=3)
- Client-Side Security: total=3 (gen1=3, gen2=0)
- Data Protection: total=5 (gen1=5, gen2=0)
- Input Validation: total=4 (gen1=1, gen2=3)
- Secure Coding: total=6 (gen1=6, gen2=0)
- Secure Design: total=4 (gen1=4, gen2=0)
- Web Application Security: total=5 (gen1=5, gen2=0)
- Web Application Security - Authentication: total=4 (gen1=1, gen2=3)