Security Requirements Generated (GEN1+GEN2)

1. Implement role-based access control to ensure students can only book exam sessions for their own courses. – Access Management

2. Ensure that course coordinators can only modify schedules for courses they are assigned to. – Secure Design

3. Restrict technical staff from viewing or modifying user accounts without proper authorization. – Web Application Security

4. Use secure cryptographic algorithms to protect stored passwords and sensitive user data. – Data Protection

5. Ensure all API endpoints use HTTPS to prevent data interception. – Secure Coding

6. Regularly update cryptographic libraries to mitigate known vulnerabilities. – Security Testing

7. Sanitize all user inputs in the class scheduling form to prevent SQL injection. – Input Validation

8. Validate and escape all messages sent between students and professors to prevent XSS. – Web Application Security - Input Handling

9. Use parameterized queries for all database interactions involving user input. – Secure Coding

10. Ensure the system architecture does not expose internal system details in error messages. – Secure Design

11. Implement rate limiting on login attempts to prevent brute force attacks. – Web Application Security - Authentication

12. Disable directory listing on the web server to prevent information disclosure. – Security Testing

13. Configure security headers such as CSP and X-Frame-Options to mitigate clickjacking and other attacks. – Web Application Security

14. Ensure all third-party libraries used in the application are up-to-date and free from known vulnerabilities. – Security Testing

15. Conduct regular security audits to identify and remediate misconfigurations. – Secure Design

16. Implement secure session management to prevent session fixation attacks. – Authentication

17. Ensure session tokens are invalidated upon logout and after a period of inactivity. – Web Application Security - Authentication

18. Use secure, HttpOnly, and SameSite attributes for all cookies. – Secure Coding

19. Enforce strong password policies for all user accounts, including minimum length and complexity requirements. – Authentication

20. Implement multi-factor authentication for professors and course coordinators. – Web Application Security - Authentication

21. Provide clear feedback to users during password reset to prevent account enumeration. – Secure Design

22. Log all access control failures and review them regularly for potential security issues. – Security Testing

23. Ensure that sensitive operations, such as deleting classes, require re-authentication. – Access Management

24. Implement checks to prevent CSRF attacks on all state-changing requests. – Web Application Security

25. Regularly scan dependencies for known vulnerabilities and update them promptly. – Security Testing

26. Use a dependency management tool to track and update third-party libraries. – Secure Coding

27. Ensure all third-party components are obtained from trusted sources. – Secure Design

28. Monitor and log all failed login attempts and unusual access patterns. – Security Testing

29. Implement automated alerts for suspicious activities, such as multiple failed login attempts from the same IP. – Web Application Security

30. Conduct regular security training for developers to recognize and mitigate logging and monitoring failures. – Secure Design

31. All sensitive data, including user credentials and personal information, must be encrypted both in transit and at rest using industry-standard encryption algorithms. – Data Protection

32. Access to sensitive data must be restricted based on user roles, ensuring that only authorized users can view or modify such data. – Data Protection

33. Regular audits must be conducted to ensure compliance with data protection policies and to identify any unauthorized access or data breaches. – Data Protection

34. All user inputs must be validated on both the client and server sides to prevent injection attacks and ensure data integrity. – Input Validation

35. Input fields must enforce strict character limits and formats to prevent buffer overflow and other input-related vulnerabilities. – Input Validation

36. Sanitize all user inputs to remove potentially harmful characters or scripts before processing or storing them in the database. – Input Validation

37. Implement secure session management to prevent session hijacking and ensure that session tokens are invalidated after logout. – Web Application Security - Input Handling

38. Use parameterized queries or prepared statements to prevent SQL injection attacks when handling database inputs. – Web Application Security - Input Handling

39. Ensure that all file uploads are scanned for malware and restricted to specific file types and sizes to prevent malicious file execution. – Web Application Security - Input Handling

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

Counting per category (total = gen1 + gen2):

- Access Management: total=2 (gen1=2, gen2=0)

- Authentication: total=2 (gen1=2, gen2=0)

- Data Protection: total=4 (gen1=1, gen2=3)
- Input Validation: total=4 (gen1=1, gen2=3)
- Secure Coding: total=4 (gen1=4, gen2=0)
- Secure Design: total=6 (gen1=6, gen2=0)
- Security Testing: total=6 (gen1=6, gen2=0)
- Web Application Security: total=4 (gen1=4, gen2=0)
- Web Application Security - Authentication: total=3 (gen1=3, gen2=0)
- Web Application Security - Input Handling: total=4 (gen1=1, gen2=3)