Security Requirements Generated (GEN1+GEN2)

1. Implement role-based access control to ensure that only authorized users can modify class schedules or exam bookings. – Access Management

2. Enforce strict validation on user inputs to prevent unauthorized access to sensitive data such as exam schedules or student messages. – Input Validation

3. Ensure that all API endpoints handling class scheduling or exam bookings validate the user's permissions before processing requests. – Web Application Security

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

5. Implement secure session management to prevent session hijacking or fixation attacks. – Authentication

6. Ensure that all authentication tokens are securely generated and invalidated upon logout. – Web Application Security - Authentication

7. Sanitize all user inputs to prevent SQL injection when querying class schedules or exam bookings. – Secure Coding

8. Use parameterized queries for all database operations involving user inputs. – Input Validation

9. Implement input validation to prevent cross-site scripting (XSS) attacks in student-professor communication. – Client-Side Security

10. Design the system to handle high traffic loads during peak times, such as exam booking periods. – Availability

11. Ensure that the system can recover quickly from failures to maintain continuous access to class schedules. – Secure Design

12. Implement load balancing to distribute traffic evenly across servers. – Web Application Security

13. Configure security headers to prevent clickjacking and other client-side attacks. – Client-Side Security

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

15. Implement content security policy (CSP) to mitigate the risk of XSS attacks. – Web Application Security

16. Regularly audit and update dependencies to avoid vulnerabilities in third-party components. – Secure Coding

17. Ensure that all libraries and frameworks used are from trusted sources and regularly maintained. – Secure Design

18. Monitor for and apply security patches for all dependencies in a timely manner. – Web Application Security

19. Implement multi-factor authentication for professors and course coordinators to enhance account security. – Authentication

20. Enforce strong password policies for all user accounts, including students and faculty. – Web Application Security - Authentication

21. Provide secure password recovery mechanisms without compromising account security. – Data Protection

22. Log all access attempts to sensitive data, such as exam bookings or class schedule changes. – Access Management

23. Ensure that logs are stored securely and are tamper-evident. – Data Protection

24. Implement monitoring to detect and alert on suspicious activities, such as multiple failed login attempts. – Web Application Security

25. Ensure that all APIs used for student-professor communication are protected against CSRF attacks. – Client-Side Security

26. Implement anti-CSRF tokens for all forms handling class scheduling or exam bookings. – Secure Coding

27. Validate the origin of requests to prevent unauthorized actions from being executed. – Input Validation

28. Regularly review and update server configurations to prevent security misconfigurations. – Secure Design

29. Ensure that default accounts and passwords are disabled or changed during deployment. – Authentication

30. Conduct periodic security audits to identify and rectify any misconfigurations. – Web Application Security

31. The system must ensure 99.9% uptime during peak usage hours to guarantee continuous access to class schedules and classroom availability. – Availability

32. Implement load balancing and failover mechanisms to prevent downtime during high traffic periods. – Availability

33. Regularly monitor and test the system's resilience to ensure it can recover quickly from failures without significant downtime. – Availability

Total: GEN1=30, GEN2=3, Total=33

Counting per category (total = gen1 + gen2):

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

- Authentication: total=3 (gen1=3, gen2=0)

- Availability: total=4 (gen1=1, gen2=3)

- Client-Side Security: total=3 (gen1=3, gen2=0)

- Data Protection: total=3 (gen1=3, gen2=0)

- Input Validation: total=3 (gen1=3, gen2=0)

- Secure Coding: total=4 (gen1=4, gen2=0)

- Secure Design: total=3 (gen1=3, gen2=0)

- Web Application Security: total=6 (gen1=6, gen2=0)

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