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. – Web Application Security

3. Restrict unregistered users from accessing any user-specific data beyond public schedules. – Secure Design

4. Use parameterized queries to prevent SQL injection when searching for classroom availability. – Secure Coding

5. Sanitize all user inputs in the class scheduling form to prevent XSS attacks. – Input Validation

6. Validate and encode all outputs displayed on the user interface to prevent client-side script injection. – Client-Side Security

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

8. Ensure that passwords are hashed using a strong cryptographic algorithm before storage. – Authentication

9. Implement secure session management to prevent session hijacking. – Web Application Security

10. Conduct regular security audits to identify and remediate insecure dependencies. – Security Testing

11. Ensure all third-party libraries used for maps and scheduling are up-to-date and free from known vulnerabilities. – Secure Design

12. Monitor and log all access to sensitive APIs to detect and prevent abuse. – Privacy Security

13. Configure security headers such as CSP and HSTS to protect against common web vulnerabilities. – Web Application Security

14. Disable unnecessary HTTP methods and endpoints to reduce the attack surface. – Secure Design

15. Implement rate limiting on authentication endpoints to prevent brute force attacks. – Authentication

16. Ensure that all error messages are generic and do not reveal sensitive system information. – Secure Coding

17. Log all security-relevant events, such as failed login attempts, for auditing purposes. – Security Testing

18. Implement proper session timeout mechanisms to reduce the risk of session fixation. – Web Application Security

19. Enforce multi-factor authentication for all administrative staff accessing sensitive functions. – Authentication

20. Implement password complexity requirements and regular password rotation policies. – Access Management

21. Provide clear and secure password recovery mechanisms without compromising security. – Privacy Security

22. Ensure that all data exchanged between students and professors is encrypted end-to-end. – Data Protection

23. Implement secure file upload validation to prevent malicious file uploads. – Input Validation

24. Regularly review and update cryptographic protocols to adhere to current best practices. – Secure Coding

25. Limit the number of login attempts to prevent brute force attacks on user accounts. – Authentication

26. Implement CAPTCHA for high-risk actions such as account creation or password reset. – Client-Side Security

27. Monitor and alert on suspicious login patterns, such as multiple failed attempts from different locations. – Security Testing

28. Ensure that all APIs are properly documented and include security considerations. – Web Application Security

29. Validate and sanitize all API inputs to prevent injection attacks. – Input Validation

30. Implement proper authentication and authorization checks for all API endpoints. – Access Management

Total: GEN1=30, GEN2=0, Total=30

Counting per category (total = gen1 + gen2):

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