Security Requirements Generated (GEN1+GEN2)

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

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

3. Restrict technical staff from accessing student messages without proper authorization. – Data Protection

4. Use secure cryptographic algorithms to store passwords and sensitive data in the database. – Authentication

5. Ensure all API endpoints enforce HTTPS and use secure headers. – Secure Coding

6. Regularly update dependencies to mitigate known vulnerabilities. – Software Maintenance Security

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

8. Validate and escape all dynamic content displayed in student and professor messages. – Client-Side Security

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

10. Implement rate limiting on login attempts to prevent brute force attacks. – Operational Security

11. Ensure all sensitive data transmitted between client and server is encrypted. – Privacy Security

12. Disable unnecessary HTTP methods on the server. – Web Application Security

13. Configure security headers such as CSP and X-Frame-Options to mitigate client-side attacks. – Client-Side Security

14. Ensure session tokens are invalidated after logout or inactivity. – Authentication

15. Regularly audit and update security configurations. – Software Maintenance Security

16. Ensure all third-party libraries used for maps and scheduling are up-to-date and free of known vulnerabilities. – Secure Coding

17. Conduct regular security reviews of the codebase to identify outdated components. – Software Maintenance Security

18. Monitor and log all third-party API interactions for anomalies. – Operational Security

19. Implement multi-factor authentication for professors and course coordinators. – Authentication

20. Ensure password policies enforce strong, complex passwords. – Data Protection

21. Provide clear feedback for failed login attempts without revealing sensitive information. – Privacy Security

22. Log all access to sensitive data, such as exam bookings and schedule changes. – Operational Security

23. Ensure logs do not contain sensitive information like passwords or personal data. – Privacy Security

24. Regularly review logs for suspicious activity. – Software Maintenance Security

25. Implement CSRF tokens for all state-changing actions, such as booking exam sessions. – Web Application Security

26. Ensure all forms and APIs validate the origin of requests. – Input Validation

27. Educate users on recognizing and reporting phishing attempts. – Privacy Security

28. Ensure error messages do not reveal system details or sensitive information. – Secure Design

29. Implement proper error handling to prevent application crashes from malicious input. – Secure Coding

30. Regularly test the application for unhandled exceptions and edge cases. – Software Maintenance Security

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

Counting per category (total = gen1 + gen2):

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

- Client-Side Security: total=2 (gen1=2, gen2=0)

- Data Protection: total=2 (gen1=2, gen2=0)

- Input Validation: total=2 (gen1=2, gen2=0)

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

- Privacy Security: total=4 (gen1=4, gen2=0)

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

- Secure Design: total=2 (gen1=2, gen2=0)

- Software Maintenance Security: total=5 (gen1=5, gen2=0)

- Web Application Security: total=3 (gen1=3, gen2=0)