1. Overview

This document outlines the security measures implemented in the Tutor Scheduling Web Application. Based on the OWASP Web Security Testing Guide (WSTG), this document addresses critical security areas, including user management, authentication, authorization, data confidentiality, integrity, accountability, session management, and transport security.

This document also covers our functional testing strategies and examples of the documentation and business requirements to create the test cases.

2. Security Areas

The following table provides the security assessment checklist.

Test ID	Test Name	Test File	Status
IDNT	Identity Management Testing		
IDNT-TC-01	Test Role Definitions	apptMgr-RBAC.test.js	PASS
IDNT-TC-02	Test User Registration	registration.test.js	PASS
IDNT-TC-03	Testing for Weak or Unenforced Username Policy	registration.test.js	PASS
IDNT-TC-04	Testing of User Enumeration	See Error Handling	PASS
ATHN-	Authentication Testing		
ATHN-TC-01	Testing for Credentials Transported over an Encrypted Channel	End-to-end Testing	TBD
ATHN-TC-02	Testing for Default Credentials	registration.test.js	PASS
ATHN-TC-03	Testing for Weak Password Policy	registration.test.js	PASS
ATHN-TC-04	Testing for Password Change or Reset Functionalities	TBD	TBD
ATHZ-	Authorization Testing		
ATHZ-TC-01	Testing for Privilege Escalation	authRole.test.js	PASS
SESS-	Session Management Testing		
SESS-TC-01	Testing for Cross Site Request Forgery	TBD	TBD
SESS-TC-02	Manual Testing for Session Timeout	N/a	PASS
SESS-TC-03	Testing Cookies Attribute	End-to-end Testing	TBD
INPV	Input Validation Testing		
INPV-TC-01	Testing for SQL injection	registration.test.js	PASS
ERRH	Error Handling		
ERRH-TC-01	Testing for Proper Error Handling – Appointment Manager	appointmentManager.test.js	PASS
ERRH-TC-02	Testing for Proper Error Handling – Availability Manager	availabilityManager.test.js	PASS

ERRH-TC-03	Testing for Proper Error Handling – User Routes	userRoutes.test.js	PASS
CRYP	Cryptography		
CRYP-TC-01	Testing for Weak Transport Security	HTTPS	PASS
CRYP-TC-02	Testing for Weak Encryption	userLogin.test.js	PASS
CLNT	Client-Side Testing		
CLNT-TC-01	Testing for DOM Based Cross Site Scripting	TBD	TBD
CLNT-TC-02	Testing for HTML injection	TBD	TBD
CLNT-TC-03	Testing for CSS injection	TBD	TBD

2.1 User Management

- **Secure Registration**: Enforces new users register with secure credentials. Validation of user inputs (**sanitization**) to prevent injection attacks were tested using express validator.
- **Password Policies**: Enforces strong passwords:
  - Password length of 8-12,
  - At least one Uppercase letter,
  - At least one lowercase letter,
  - At least one alpha-numeric digit,
  - At least one character.
- **Account Lockout**: Using **express-rate-limit** the project implements an account lockout mechanism **5** failed login attempts to prevent brute-force attacks.
- **User Enumeration Prevention**: Security controls provide generic error messages during registration and login to prevent revealing whether a username or email exists.

2.2 Authentication

- **Password Hashing:** Stores passwords securely using hashing – **bcrypt**.
- **Secure Token Handling**: Uses secure cookies with HttpOnly and Secure flags for session tokens. Periodically refresh tokens to mitigate session hijacking risks.
- **Account Recovery Mechanisms**: The next sprint will ensure secure password reset functionality by using temporary, expiring links sent via email.
- **Multi-Factor Authentication (MFA)**: For the next sprint we are researching the addition of MFA to accounts to increase security.
- **Credential Stuffing**: The next sprint will research CAPTCHA to block bots.

2.3 Authorization

- **Role-Based Access Control (RBAC)**: Implements RBAC to manage permissions, assigning roles such as "student" and "tutor" with specific access rights.
- **Least Privilege Principle**: Enum roles are attached to session data that are checked in routing. Limits users to the minimum permissions required for their tasks, reducing the risk of unauthorized access.
    - **No role**: Non-registered users may view available tutors and GET requests to non-form webpages
    - **Student role**: Students may only login, logout, and make appointments.
    - **Tutor role**: Tutors may login, log out, make appointments, schedule an availability schedule, and update appointment statuses.
    - **Admin role**: Has all the privileges of tutors and students.
- **Access Control Checks**: In the next sprint, end-to-end testing shall ensure that each endpoint includes server-side checks to confirm that users have permission to access the requested resources.

2.4 Data Confidentiality

- **Data Encryption**: Enforces HTTPS with TLS protocol to secure data in transit. Encrypt sensitive fields, such as passwords and session tokens, with robust encryption algorithms.
- **Sensitive Data Masking**: Hashing – bcrypt - avoids exposing sensitive information like user IDs or session IDs in URLs and in front-end code.

2.5 Integrity

- **Input Validation**: This project uses **express-validator** to validate user input and prevent injection attacks.
- **Anti-CSRF Tokens**: Uses CSRF tokens with **csurf** for all state-changing requests to protect against Cross-Site Request Forgery.
- **Data Integrity Checks**: Regularly perform audits on the database to detect and prevent unauthorized data manipulation.

2.6 Accountability

- **Logging & Monitoring**: Uses **Pino** to log critical user actions, such as login attempts, new user accounts, password changes, and access to sensitive data.
- **Audit Trails**: MongoDB stores an audit trail for administrator-level actions and sensitive data changes to enable post-incident analysis.

2.7 Session Management

- **Session Routing**: Redirects users' post-registration, setting up for controlled sessions upon login.
- **Session Timeout**: Set a session timeout (15 minutes of inactivity) to minimize risks associated with unattended sessions.
- **Token Revocation on Logout**: Terminates sessions destroys cookie data on logout by invalidating session tokens to prevent reuse.
- **Secure Session Cookies**: Use cookies with Secure, HttpOnly, and SameSite attributes to prevent unauthorized access via JavaScript or cross-site requests.

2.8 Transport Security

- **Enforce HTTPS**: Ensure all communication occurs over HTTPS (HTTP Strict Transport Security)

3. Testing Strategies and Documentation

Our testing strategy includes a combination of unit and integration testing. The test cases are created from business requirements that describe the required features of the application. The documentation for our test suites includes a table for our business requirements. The example tables listed are not an exhaustive list of test and business cases that have been created and documented. Following is a sample of the business requirements laid out in our documentation:

Requirement ID	Description	Priority	Status
BR-001	Users are able to create an account.	High	Approved
BR-002	Registered users can access their account.	High	Approved
BR-003	Unauthorized cannot successfully log in.	High	Approved
BR-004	Students need to be able to schedule tutoring appointments with tutors	High	Approved
BR-005	Students need to be able to view appointments that they have made.	High	Approved

These business requirements are then matched to our functional test cases, which are also laid out in a table. Our document currently logs over twenty-nine test cases including unit and integration test cases:

				the user has been saved	
TC-002	BR-002	Verify the user can successfully login with credentials.	The user is registered in the database.	1. Send a login request with a valid username and password. 2. Verify that the response redirects the user to their profile page. 3. Ensure that session variables are set.	The user is logged in successfully and is redirected to the profile page.
TC-003	BR-003	Verify that the system responds appropriately if the user does not exist.	The user does not exist in the database.	1. Send a login request with a non-existing username 2. Verify that the login	The login view is rendered with the message “User not found”.

These test cases are then matched to our business requirements in a requirements traceability matrix:

Requirement ID	**Requirement Description**	**Test Case ID**
BR-001	Users can register with a username and password.	TC-001
BR-002	Registered users can log in with valid credentials.	TC-002
BR-003	The system returns an error if the user does not exist.	TC-003
BR-003	The system returns an error if the password is incorrect.	TC-004
BR-004	The system successfully creates an appointment between a user and a tutor.	TC-005, TC-006
BR-005	Students should be able to view appointments they have made with tutors.	TC-007, TC-008, TC-009, TC-010
BR-006	Appointments need to be able to be	TC-011, TC-012, TC-013, TC-014, TC-015,

We are also keeping track of the state of our test cases in a test case status matrix:

Test Case ID	Test Description	Status	Defects/Issues	Comments
TC-001	Verify that a new user can be saved to the database.	Pass	None	
TC-002	Verify successful login with proper credentials.	Pass	None	
TC-003	Verify system response when user does not exist.	Pass	None	
TC-004	Verify system response with incorrect password	Pass	None	
TC-005	Verify that an appointment can be	Pass	None	

The current documentation and tests that have been created cover all of our business requirements matching our features to our tests. Our tests ensure that our features function as expected and meet the expectations of the required product that we are developing.