Test Plan

Authentication

Test Description	Test Input	Expected Result	Actual Result	Pass/Fail
Login with incorrect credentials	User: JDeven200 Password: incorrectpass	Access denied, redirects to login screen again	Access is denied and access to other pages is redirected to the login page.	Pass
Login with correct credentials	User: JDeven200 Password: Re@11yS3cuR3M3m3P@ssw0rd	Access granted	Home page with created/assigned tickets is loaded. The user is also given a cookie and login session.	Pass
Login with attempted XSS	User: admin Password: <script> alert("app is vulnerable!");</script>	Script tags filtered, access denied	Since the input parameters are parameterised upon receipt, the script tags do not work.	Pass
Login with attempted SQL injection	User/Password: 'SELECT * FROM USER_CREDENTIALS--	Special characters filtered, 'password' field parametrised, access denied	Parameterising both user name and password means that no SQL injection occurs. Access is denied since the user/password is wrong	Pass
Ensure that passwords are stored as encrypted strings in authentication server	SELECT * FROM AUTH_DB	Set of encrypted user names and passwords e.g. USER: 8ef22ab PASSWORD: 21bbf5c	User is stored as an ID number e.g. 'JDeven200: 1' And the password is hashed e.g. '$2a$12$oGwzD5gyn5XdsgZVjNyodON5rtGo4pMv8JgEcUf.hp4wRGVA98jJi'	Pass
Ensure application blocks user/IP after 8 login attempts	User: admin Password: incorrectpass (x8)	User banned for 24 hours, access denied	Unimplemented, no ban occurs	Fail
Time user out after 30 minutes	User logs in correctly and then remains idle for 30 minutes	User should be logged out and redirected to login screen after 30 minutes.	After 30 minutes, the cookies are removed, and the session is invalidated. At this point, the user is booted back to the login screen.	Pass
Sign up with a password which is too short/long, is not alphanumeric, does not contain a special character and contains an illegal character	Enter username: 'bad'/'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa' Enter password: 'bad<>'/'<script>alert("this is far too long and also illegal")</script>'	The sign up process should fail in each case and an error message should be shown to let the user know what they did wrong.	The sign up page fails, an error for each inaccuracy is shown.	Pass

Sign up with username/password which meets the requirements (username between 5-32 characters, password between 8-16 characters, is alphanumeric and contains a special character. No illegal characters for either).	Enter username: 'gooduser' Enter password: 'g00dp@ssw0rd'	The sign up process should be successful and the user should be directed to the login page.	The sign up works, user is redirected to login page	Pass

Authorisation

Test Description	Test Input	Expected Result	Actual Result	Pass/Fail
Attempt accessing a ticket which does not belong to the currently logged in user	User "user1" attempts access of Ticket "ticket2". This ticket belongs to User "user2"	Ticket does not load. Application alerts user that they do not have the access rights to view this ticket.	Ticket does not load, no error message is loaded, but the user is booted back to their home page.	Pass
A tester attempts to make a "development" type ticket	User "user1" of type "Tester" tries to create Ticket "ticket1" of type "development"	Ticket is not created, application alerts user that they can only create tickets relevant to their role.	Ticket is not created and application sends user to a generic error page (due to time constraints). Works from a security perspective.	Pass
An Admin attempts to create a ticket of any type	Log in as 'Admin' and create a ticket of type 'development', 'testing' and 'prodcution'	Ticket is made in each case since the admin has elevated access rights	Tickets are all created	Pass

Confidentiality

Test Description	Test Input	Expected Result	Actual Result	Pass/Fail
Use burp suite to listen in on user requests/responses and attempt to view sent/received data	Use the interceptor in burp suite to view user requests and responses. Then, look for exploitable data within the parameters and request/response bodies.	Data should be encrypted through the HTTPS protocol and not viewable through this method	Data which comes through POST requests are not viewable. HTTPS was not implemented in time. A user cookie is	Fail

			viewable which could be a security flaw.	

Integrity

Test Description	Test Input	Expected Result	Actual Result	Pass/Fail
Attempt SQL injection in input box	‘SELECT * FROM USER—in input box	Special characters should be filtered out, even if they are not parameterisation should prevent SQL from executing	Special characters filtered, input parameterised. No SQL injection occurs	Pass
Attempt SQL injection in URL	‘SELECT * FROM USER—in URL	Special characters should be filtered out, even if they are not parameterisation should prevent SQL from executing	Parameterisation prevents SQL injection from working in the URL, nothing is returned on page, or in console.	Pass
Attempt SQL injection variations in input box	%27SELECT * FROM USER%20%20 ‘’SELECT * FROM USER--- ‘SeLeCt * FrOm UsEr WheRe 1=1— All in input box	Special characters should be filtered out, even if they are not parameterisation should prevent SQL from executing	Special characters filtered, input parameterised, no SQL injection occurs	Pass
Attempt SQL injection variations in URL	%27SELECT * FROM USER%20%20 ‘’SELECT * FROM USER--- ‘SeLeCt * FrOm UsEr WheRe 1=1— All in input box	Special characters should be filtered out, even if they are not parameterisation should prevent	Special characters filtered, input parameterised, no SQL injection occurs	Pass

		SQL from executing		
Attempt reflected XSS	<script>alert(1)</script> in URL	Special characters should be filtered out, script should not execute	Special characters filtered, input parameterised, script does not execute	Pass
Attempt stored XSS	Write <script>alert(1)</script> in a comment, then open the comment in app after saving to DB	Whole comment should be parameterised and treated as a single object of text. Script should not execute.	Special characters filtered, input parameterised, script does not execute	Pass
Attempt DOM-based XSS	<script>alert(document.cookie)</script> In URL	Special characters should be filtered out, script should not execute	Special characters filtered, input parameterised, script does not execute	Pass
Attempt XSS variations	<ScRiPt>alert(1)</ScRiPt> %3cscript%3ealert(1)%3c/script%3e 	Special characters should be filtered out, script should not execute	Special characters filtered, input parameterised, script does not execute	Pass