Abdulrehman Ismaeel

Test Report

To test the system, I manually checked all the major features: user login/logout, creating posts, replying (including nested replies), voting, admin controls, and marking accepted answers. I also tested edge cases like trying to reply without logging in or accessing admin-only features as a regular user. All API endpoints were tested using Postman, and I verified database behavior directly using MySQL. On the frontend, I checked that the UI updated correctly after each action and that all state changes reflected in real time.

Packages I used:

axios: Used to send and test HTTP requests from the frontend to the backend APIs.
• bcryptjs: Ensures passwords are securely hashed before being stored — verified that hashes worked by checking database inserts.
• cookie & cookie-parser: Handled cookies for session/token tracking; I tested login persistence and logout functionality.
• cors: Enabled cross-origin requests between React frontend and Node backend during development.
• dotenv: Managed environment variables for things like DB credentials and JWT secrets, which I tested by switching between environments.
• express: Core of the backend API – I tested all routes and middleware through Postman and browser.
• jsonwebtoken: Used for user authentication; I verified protected routes could only be accessed with a valid token.
• mysql2: Managed all DB operations; I tested inserts, deletes, and nested replies using both raw SQL and the live app.
• multer: Used for image uploads in replies — tested by uploading and rendering images successfully.
• react-quill: Rich text editor for posts and replies; I checked for correct formatting and content submission.
• wait-on: Delays app start until the MySQL server is ready — helped prevent race conditions during testing.
• nodemon: Automatically restarted the backend server on code changes, speeding up testing/debugging.
• moment: Used for formatting timestamps (like “5 minutes ago”) — verified by checking the UI display.

Biggest Issue: The biggest issue I ran into was properly rendering and managing nested replies—figuring out the right recursive logic both on the backend (for fetching the reply tree) and frontend (for displaying it cleanly) took the most time and trial-and-error. Another issue I ran into was handling authentication errors properly — sometimes expired or invalid tokens didn’t redirect users to the login page, which caused confusion until we added better error handling and redirects.