1. Overall Testing Approach

* Backend Testing:
* Verify Node.js endpoints, CouchDB integration, and document structure.

initPostDB() – early version is returning “error happened in your connection. Reason: getaddrinfo ENOTFOUND couchdb” 🡪 seems to be a location or authentication issue.

* Solution was to target the container rather than localhost. http://<u>:<p>@<container>:5984 successfully allowed connection.

app.post(“/postmessage”) – initial version returning error: bad special document member: \_type

Endpoint testing:

* Confirm new documents are created, read, and updated correctly.
* Frontend Testing:
* Check creation, display, and updating of posts and nested responses.
* Confirm the threaded or hierarchical structure is rendered correctly.

2. Functional Test Cases

* Coverage: Include normal and edge cases (e.g., missing fields, invalid parentId).
* Documentation: Show expected vs. actual results with screenshots or console logs.

3. Load Testing with npm loadtest   
● Setup: npm install -g loadtest or npx loadtest   
● Execution:

○ Example: loadtest --concurrency=10 --requests=1000   
http://localhost:3000/postmessage   
● Report:   
○ Provide requests per second, latency, error counts, etc.   
○ Briefly discuss your server performance under load.   
4. Challenges and Solutions   
● Document any major issues (e.g., CouchDB credentials, Docker networking, or nesting   
logic) and how you addressed them.