# HTTP/HTTPS Server Implementation Report

## 1. Design Decisions

• \*\*Core Modules\*\*: Used Node.js built-in `http` and `https` modules to meet requirements without external frameworks, ensuring full control over request/response handling. A self-signed certificate (key.pem & cert.pem) is loaded for HTTPS.

• \*\*Routing System\*\*: Implemented a custom `Router` class supporting method-based routes with parameter extraction (e.g., `/api/users/:id`) and middleware chaining.

• \*\*Request Parsing\*\*: Utilized `formidable` for multipart parsing alongside built-in JSON and URL-encoded parsers.

• \*\*Data Storage\*\*: In-memory array for user endpoints; JSON file (`data/products.json`) for product persistence.

## 2. Implementation Approach

1. \*\*Server Setup\*\*: Two servers—HTTP (3000) & HTTPS (3443)—share routing logic via `Router.handle()`.

2. \*\*User API\*\*: CRUD on `/api/users`, with error handling for invalid input & missing resources.

3. \*\*Form & File Upload\*\*: Served an HTML form at `/form` to demonstrate handling of various content types.

4. \*\*Product API with API Key\*\*: Secured CRUD under `/api/products` persisting to `data/products.json`; authentication via `x-api-key` header.

5. \*\*Redirects & Middleware\*\*: Implemented HTTP redirects and middleware for logging & auth.