

Java HTTP Server Project — Team Split (3 People)

Person 1 — Core Server & HTTP Engine

- Build the non-blocking server using Java NIO (Selector, ServerSocketChannel).
- Handle multiple ports with one process and one thread.
- Manage connection states, buffers, and timeouts.
- Parse HTTP/1.1 requests (headers, Content-Length, chunked encoding).
- Build correct HTTP responses with proper status codes.
- Ensure server stability (never crash, safe resource cleanup).
- Deliverables: Server.java, ConnectionContext.java, HttpParser.java, HttpRequest.java, HttpResponse.java.
- Tests: request parsing, chunked handling, timeouts, basic stress tests.

Person 2 — Configuration, Routing & Static Content

- Parse and validate server configuration file (config.json).
- Support multiple ports, default server, body size limits, and custom error pages.
- Implement routing system (paths, allowed methods, redirects).
- Serve static files securely with proper Content-Type.
- Handle directories (index file, listing on/off, forbidden access).
- Generate default and custom error pages (400, 403, 404, 405, 413, 500).
- Deliverables: ConfigLoader.java, Config.java, Router.java, Route.java, StaticFileHandler.java, ErrorHandler.java.
- Tests: routing rules, redirects, method restrictions, error pages.

Person 3 — Uploads, DELETE, Sessions & CGI

- Handle POST file uploads (multipart/form-data).
- Enforce client body size limits and upload directories.
- Implement DELETE method with secure path validation.
- Manage cookies and sessions (session ID, expiration, cleanup).
- Execute CGI scripts (e.g. Python) using ProcessBuilder.
- Pass PATH_INFO, stdin data, and capture stdout as HTTP response.
- Handle CGI timeouts and errors safely.
- Deliverables: UploadHandler.java, DeleteHandler.java, CGIHandler.java, Session.java, Cookie.java.
- Tests: uploads, delete permissions, session persistence, CGI execution.

Integration & Workflow

- Person 1 exposes a clean request → response interface.
- Person 2 routes requests to the correct handler.
- Person 3 implements advanced handlers (upload, delete, CGI).
- Clear separation: only Person 1 touches low-level NIO.
- Milestones: MVP → Routing → POST/DELETE → CGI/Sessions → Stress testing.