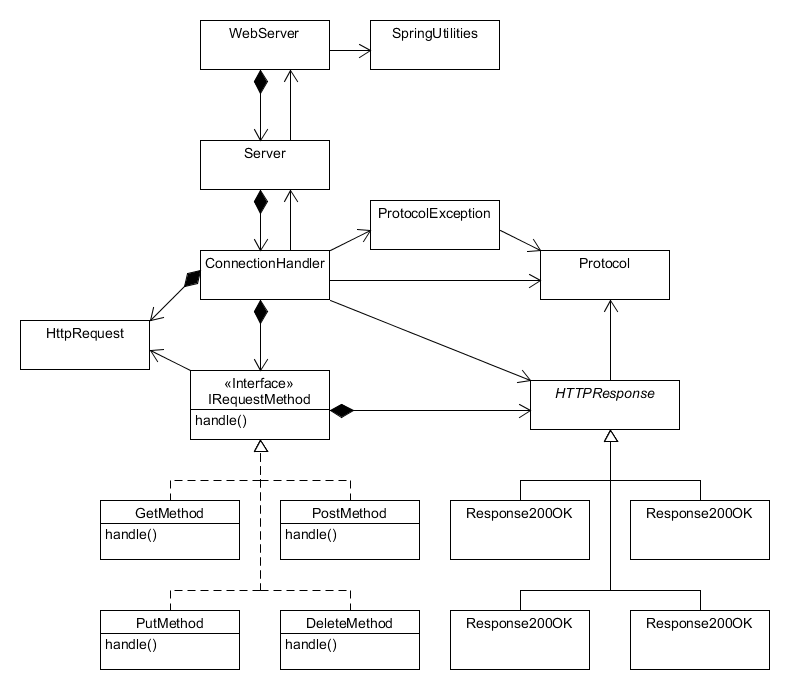
# Milestone 1 – Web Server Report

## Architecture Diagram

<TODO>

## Detailed Design



Our refactoring of the web server utilized the following design patterns:

**Strategy Pattern** – The IRequestMethod interface allows for the various implementations of request handling to be completed in unique classes. This way, additional request handling can implanted with minimal changes to the ConnectionHandler class – simply add the new request to the ConnectionHandler’s map of request methods.

**Bridge Pattern** – The abstract HTTPResponse class is used by the ConnectionHandler to write the generated response back to the client. However, the responses vary depending on the response code. Using a bridge pattern, each different response’s implementation can be handled in separate classes without the ConnectionHandler needing to have any knowledge of how it is implemented.

## Further Improvements

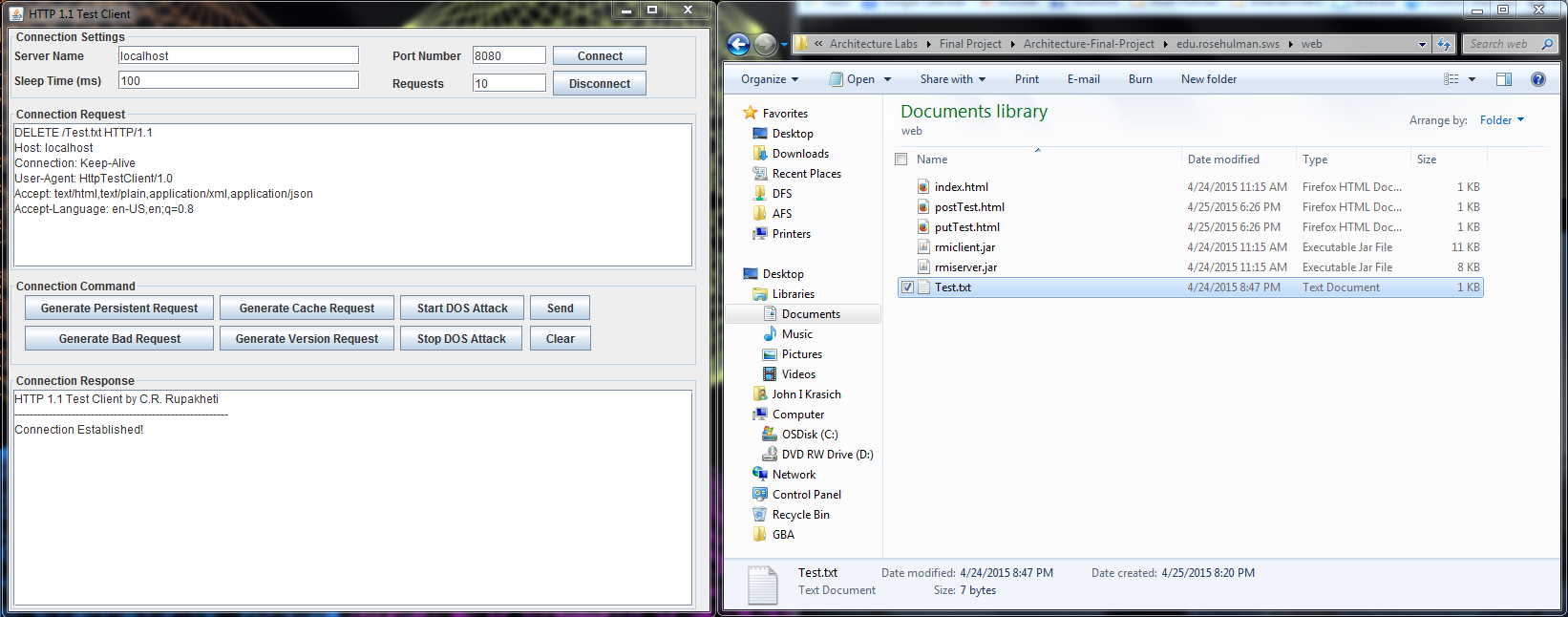
One area that can be further refactored would involve dividing the responsibilities of the ConnectionHandler class separately between requests and responses. The “run” method is rather long – breaking this up into different methods (or different classes) would make the code much more organized and easier to understand.

## Test Report

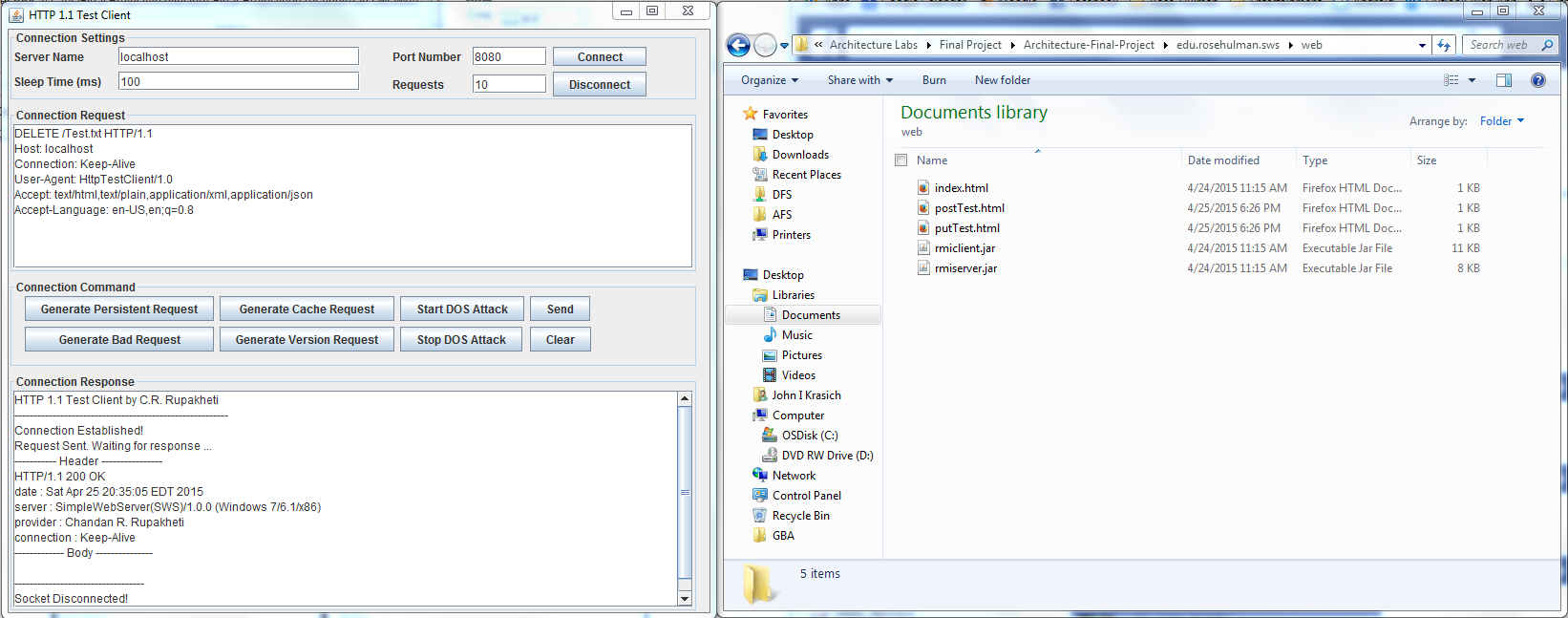
### Testing Utility

#### DELETE

Before:

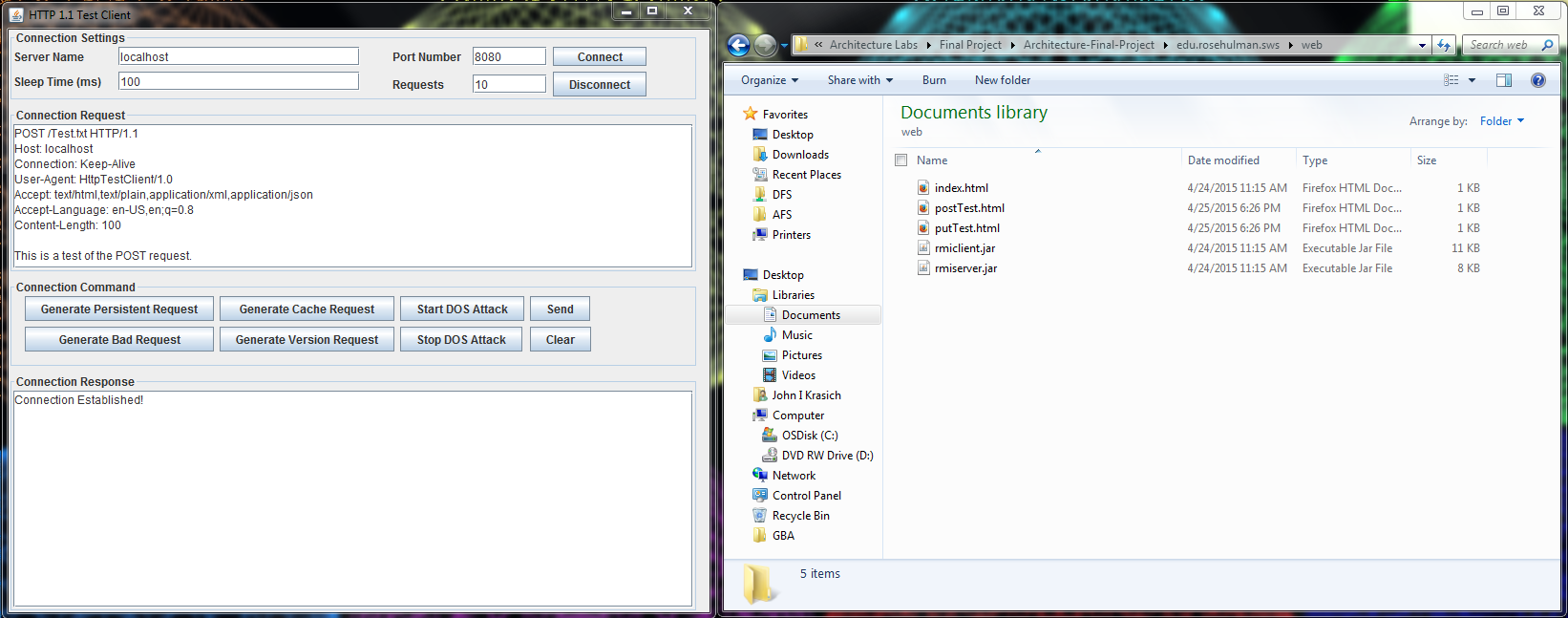


After: Response – 200 OK. File Test.txt has been deleted successfully

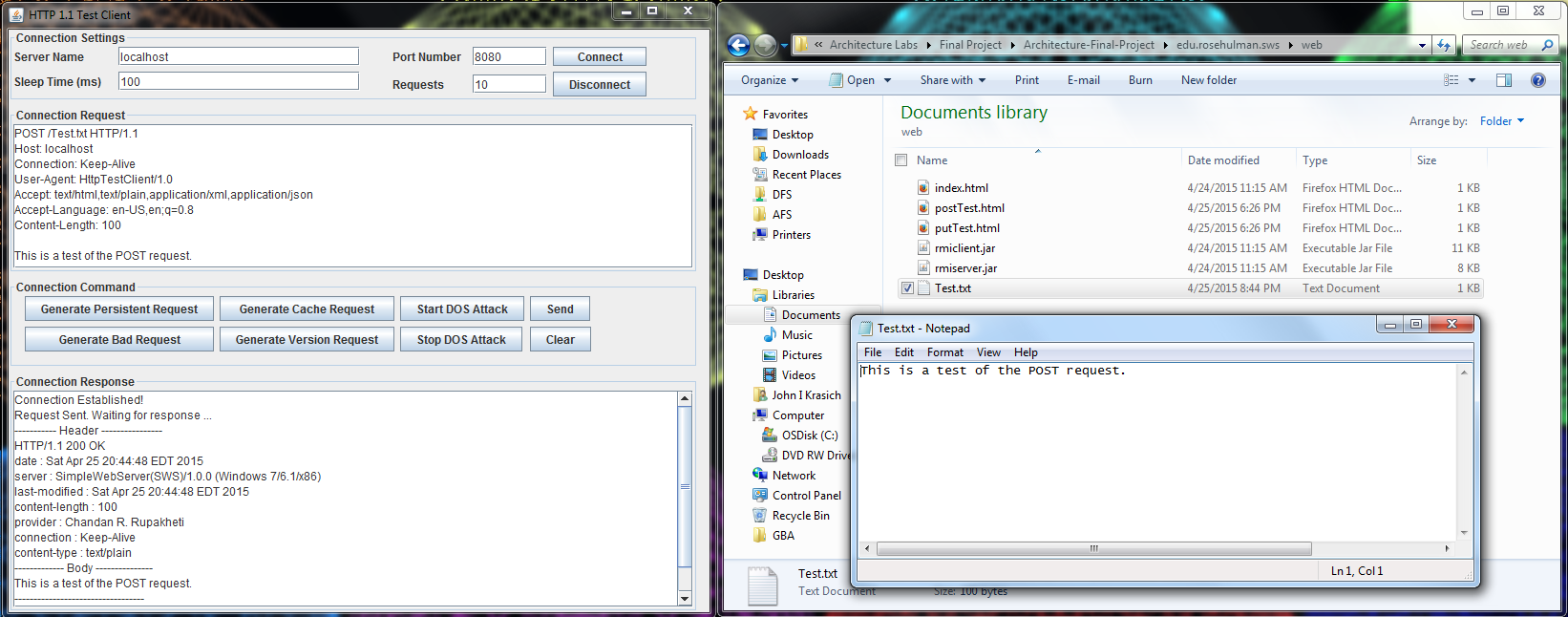


#### POST

Before:

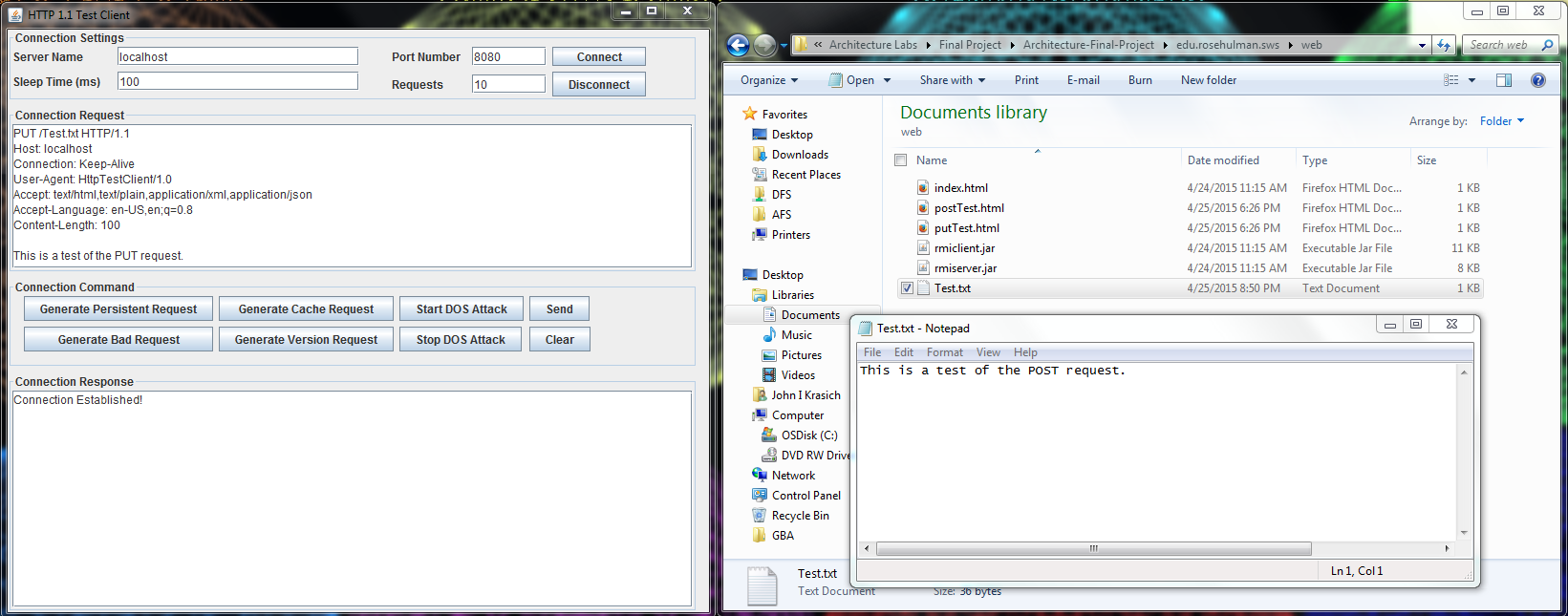


After: Response 200 OK. The file Test.txt has been created and filled with the body of the request.

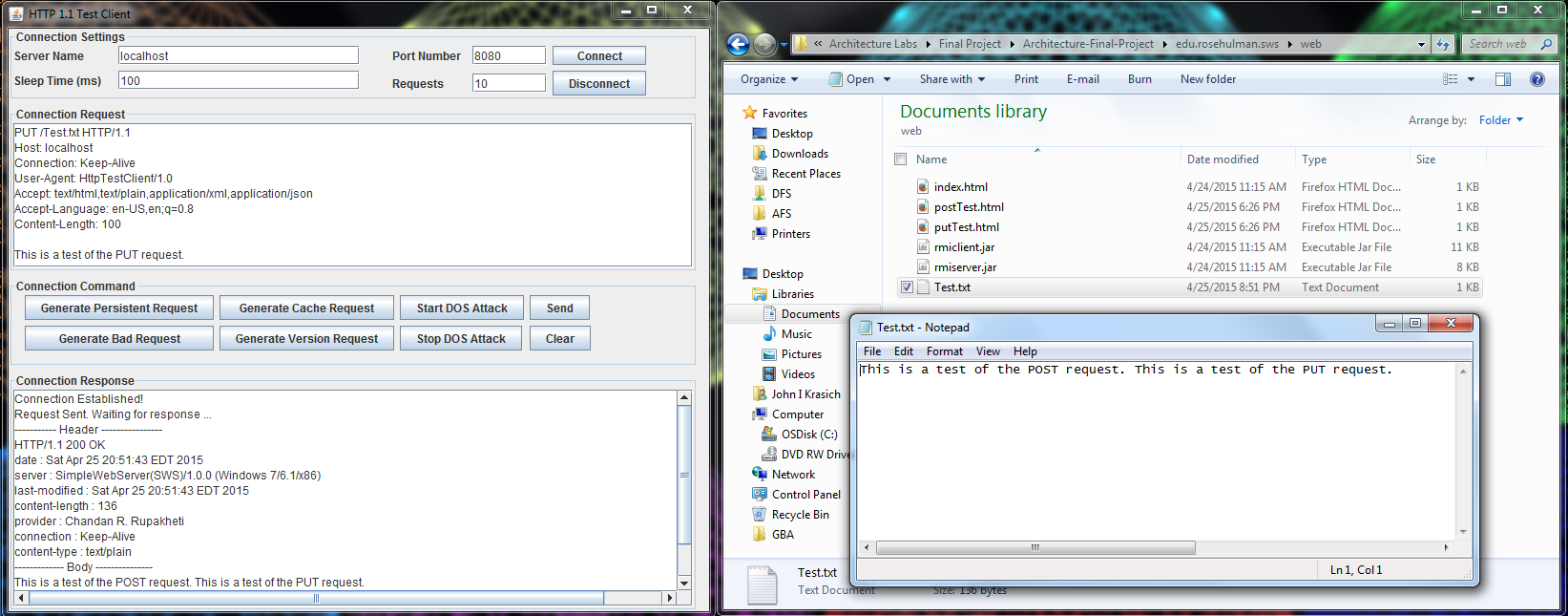


#### PUT

Before:



After: Response 200 OK. The body of the request was appended to the Test.txt file.



### Web Browser

#### DELETE

Before:

After:

#### POST

Before:

After:

#### PUT

Before:

After: