HTTP SERVER

DESIGN DOCUMENTATION

### Team 3

### Robbie

### Caleb Post

### Grant Smith

# Chosen Language

Our team chose Java as our programming language. It provides us with a simple language that all three members of our team have worked with before and have some familiarity with. This enables us to focus less on details in the language and more on the implementation of the ideas behind a web server itself.

# Socket Infrastructure

Java provides us with two different sockets. It provides us with the Socket class, an implementation of the socket architecture which supports reading and writing from data buffers which the socket automatically handles transfer of packets from, and the ServerSocket class which allows us to bind new instances of Socket once an incoming connection to the ServerSocket is detected. This means that we don’t have to create one socket which when it receives a connection request we then pass connection to a different socket, instead we have to only accept the incoming socket that ServerSocket generates for us. This greatly simplifies the code.

# Stateful Connections

Since HTTP as a stateful connection isn’t a truly functioning system, http is designed to run without state, and our product is a web server and not a website this becomes a more difficult question. We need to implement some form of state in our http connection itself and not in the website layer. A standard HTTP session assumes that the session state information is always embedded in the form of HTTP cookies and therefore the connection itself is always stateless. However, state can be created for an HTTP session if we create a particular user identity that cannot be shared among users and reused by the same user only.

We choose to implement this form of stateful connection in our project. Similar to the whitelist described below, we will have a set of tokens that can be provided by the particular user to change their access level and log their actions on the server. Tokens will be unique to users and will be issued to users based on their security level. This will let us have http connections with state.

# Security Component

Create a whitelist of allowed IP addresses, similar to an “.htaccess” file.

When a client requests access to the server, their IP is cross-referenced against the whitelisted IPs and, if those IPs do not include the client’s IP, the connection is refused. Otherwise, the connection is accepted as usual. Also, the server will deny any ‘GET’ requests for the whitelist file itself.