- **Client Server Connection**
  - The connection from the client was accepted by the server using bind, listen, and accept. A thread is created by the server using pthread_create everytime there is a connection. The connection to the server is done in WTFserver.c. All the commands are handled in header.c
- **Pthreads and Mutexes**
  - Every project was locked before any data was transferred. The project name was put into a linked list and its address becomes locked so operations on it are safe. Once the command is done, the mutex is unlocked.
- **Configure**
  - Configure takes in a Ip address and port from the user, which it then stores into a configure file so that the program can read from it in order to create connections for later commands.
- **Checkout**
  - Sends over all of the files to the client for its current version. The server first loops through a list of all the files in the .Manifest using a linked list. The server sends the file name, then sends the file size, and then sends the contents of the file to the client. Finally, the .Manifest is sent to the client. The name of the file and its contents are sent when the client sends a connection handler to the client. The server sends a connection handler to the client to let it know that checkout is complete.
- **Update**
  - Compares the server's and client's .Manifest to see if any changes have to be made. A connection handler is sent to the client if the server and client have equal .Manifest version numbers. The two files are compared by traversing the client .Manifest with linked lists to see if there is an add, modify, or conflict. All of these actions are written into a .Update file or .Conflict file when needed. Then the two files are compared again by traversing the server .Manifest to see if there is a delete.
- **Upgrade**
  - The server sends the .Update file or .Conflict file if there is one and the client applies the changes listed. This is done by sending every file name and its contents to the client and the client stops accepting files when a connection handler is sent by the client. For example, if an Update file contains A D and M, upgrade creates files in the project on client side and an entry in manifest, while deleting files in client if client side contains files that server does not have. In addition, changes content in file on client side to match content on server side.
- **Commit**
  - Client creates a commit file and sends it to the server. The server sends all the contents of its .Manifest to the client, where the client compares the entries to see what update is needed on the server side. The file contains A if server side does not contain files that client has; D if server contains files that client does not have, and both sides have file in manifest with same hash, and M if both sides have file in manifest with same hash, but the live hash of client is different from its stored hash

- **Push**
    - The server updates the new projects directory if the .Commit file that the client sends is the same as the .Commit file that the server has in its directory. If this is the case, all expiring commits are expired by removing the unneeded .Commit files. The server checks if the .Commit files are the same by comparing hashes. The server then traverses the .Commit file and does the appropriate adds, modifies, or deletes. A connection handler is sent if the commit is successful to the client. All the .Commit files are then removed.
- **Create**
    - The server creates a project based on the project name given. If the repository was not already made, it is created. The client sends the project name to the server and then it creates the project and a .Manifest and .History file for it with a version number of one. The project directory and the repository are created using mkdir.
- **Destroy**
    - The server traverses through the project given and destroys it and all its contents. All the files and subdirectories are deleted recursively. The mutex is destroyed when destroy is called.
- **Add**
    - This function adds an entry for the selected file into .Manifest, which contains filename with file version and hash of files contents if name does not already exist. In addition, the function increments the file version and updates the stored hash if the file's contents already exist in the Manifest.
- **Remove**
    - Removes the selected file's entry from the .Manifest.
- **Currentversion**
    - The server simply sends the current version of the project to the client and the version numbers of all the files in the .Manifest. This is done by traversing a linked list that consists of a list of the files and their version numbers. The name and version number is sent only when the client sends a connection handler to the server.
- **History**
    - The server sends the .History file of the project to the client. The contents of the .History file are sent when a connection handler is read in by the server by the client.
- **Rollback**
    - The server reverts to a version that the client provides if valid. This is done by traversing all the previous projects and destroying them until the version that the client wanted is reached. A connection handler is sent to the client letting it know that rollback has been completed.