DESIGN DOCUMENT FOR ASGN2

My program has 5 modules:
Main()

```
Main():
/* I will be skipping over the logistics of the server
      because they are the same as asgn1. I will focus on the multi-
threading.
*/
Once request is recieved it:
First, parses the request(either PUT or GET)
      if(PUT){
                  //parse filename and content length
                  //check filename for syntax errors
                  errchk = filename_check(filename);
                  //check if filename is in cache
                  errchk = in_cache(filename);
                  write_to_cache(filename,length,errchk,new_socket);

      }
      else if(GET){
                  //parse the filename
                  //check if in cache
                  if in cache then read from it
                  else read from disk

      }
      else{
                  send back an error
      }
```

The main functions only job is to check which request was sent and to check if the filename is in the cache.

```
void readfromfile(char*filename, int new_socket)
--------------------------------------------
```

This function performs the same details as the asgn1
Its job is to simply read from disk since file was not in cache
Some psuedocode:

```
get file descriptor(fd) for file by using open()

if(fd >0)
{
      if(fstat doesnt return error)
      {
                  read data of file to a buffer
                  close file
                  send ok response
                  send content length
                  send buffer of the file content
                  close socket
      }

}


}
else{
      send appropriate err msg
      close socket
      return;
}
return;
}

void write_to_hd(char*filename,int length,int flag,int new_socket)
-------------------------------------------
this module writes the file to disk


psuedocode:

//open the file and retrieve the file descriptor(fd)

if(open was succesful){
      //valreadfile will hold number of bytes read
      valreadile = read(in from the socket pointed by the i ptr)
      write(write the valreadfile number of bytes to the file);
      close file
      send the creat resp
      close socket
      return;
}
else{
      send appropriate err msg
      close socket
      return;
}

int in_cache(char* test_file)
----------------------------------------------
Thi modeule simply checks if the filename is in the cache
if(it is in the cache)
it will return its position in the deque
else if(the deque is not full)
return -1;//so that the program knows to simply push_back into the deque
else
return -2;//this will tell the program to push_back() and pop off the
front


write_to_cache(char*filename,int length,int flag,int new_socket)
------------------------------------------------------------------------
```

This function does the operation signaled to by the in_cache method.
It uses the return value from it that us listed as flag to determine what operation to do.
When dirty bit is set then it first updates the disk and then updates the cache.

```
      //if flag >= 0
      //then replace whatever is in that array element
      //else pop off the next element
      //if the dirty bit is 1 then call writetofile() then update cache
      //else if db == 0 then just update cache and call writetofile()
```