**COEN 317: Distributed Systems**

**Fall 2023**

**Programming Assignment 1**

Name - Anand Santosh Kulkarni

SCU ID - W1638929

1) **Assignment Description:**

a. The source contains a simple web server implementation in Java that serves content over HTTP/1.0 and HTTP/1.1. We can configure the port on this server listens and document\_root directory which contains the webcontent. These are passed as parameters to the ServerApp.java.

b. The code can host static files, such as HTML, CSS, JPEG, PNG, Images, etc.   
All the above-mentioned files must be in webcontent directory which is to be passed as document\_root while running

c. The ServerApp handles HTTP GET requests from several clients at a given time as it handles multiple client connections via implementing multithreading to process the requests.   
Each client connection spawns a new thread

d. The server supports 200, 400, 403, 404 status codes  
  
  
The program listens for incoming connections on the specified port and manages these connections via multithreading

* In the multi-threaded approach, each client connection is handled in a separate thread, allowing concurrent processing of multiple client requests.
* The program parses incoming HTTP requests, retrieves requested files, determines content types, and constructs appropriate HTTP responses.
* Active connections are periodically checked for idle status, and connections that have been inactive for a specified period are closed.

2. **Files**

Following files are attached with the zip

1. ServerApp (Main file to run)
2. Webcontent (has all the files)
3. ReadMe
4. Snippets of logs and web browser accessing web server
5. IDE project files

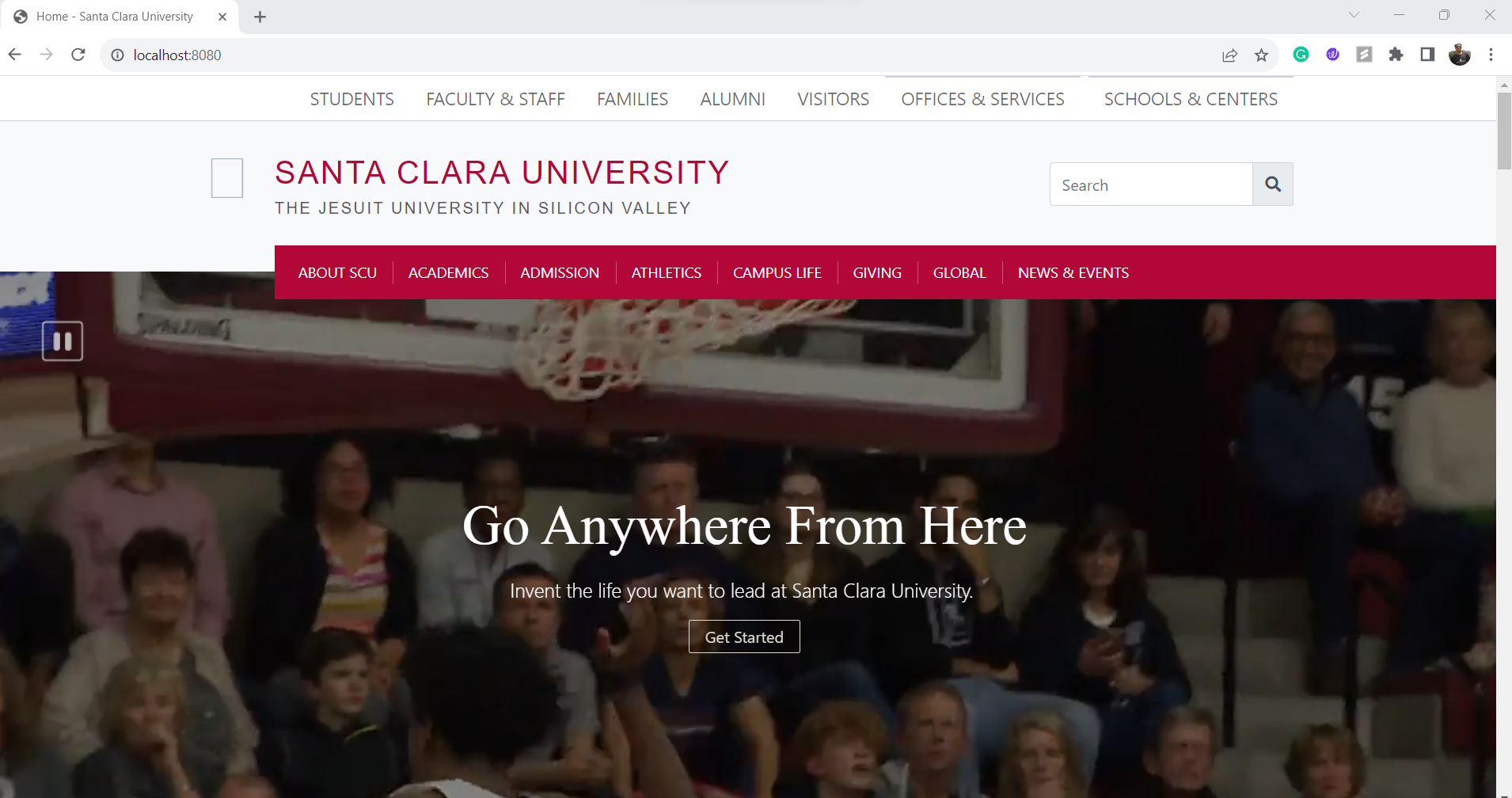
3) **Instructions to run the program on terminal:**

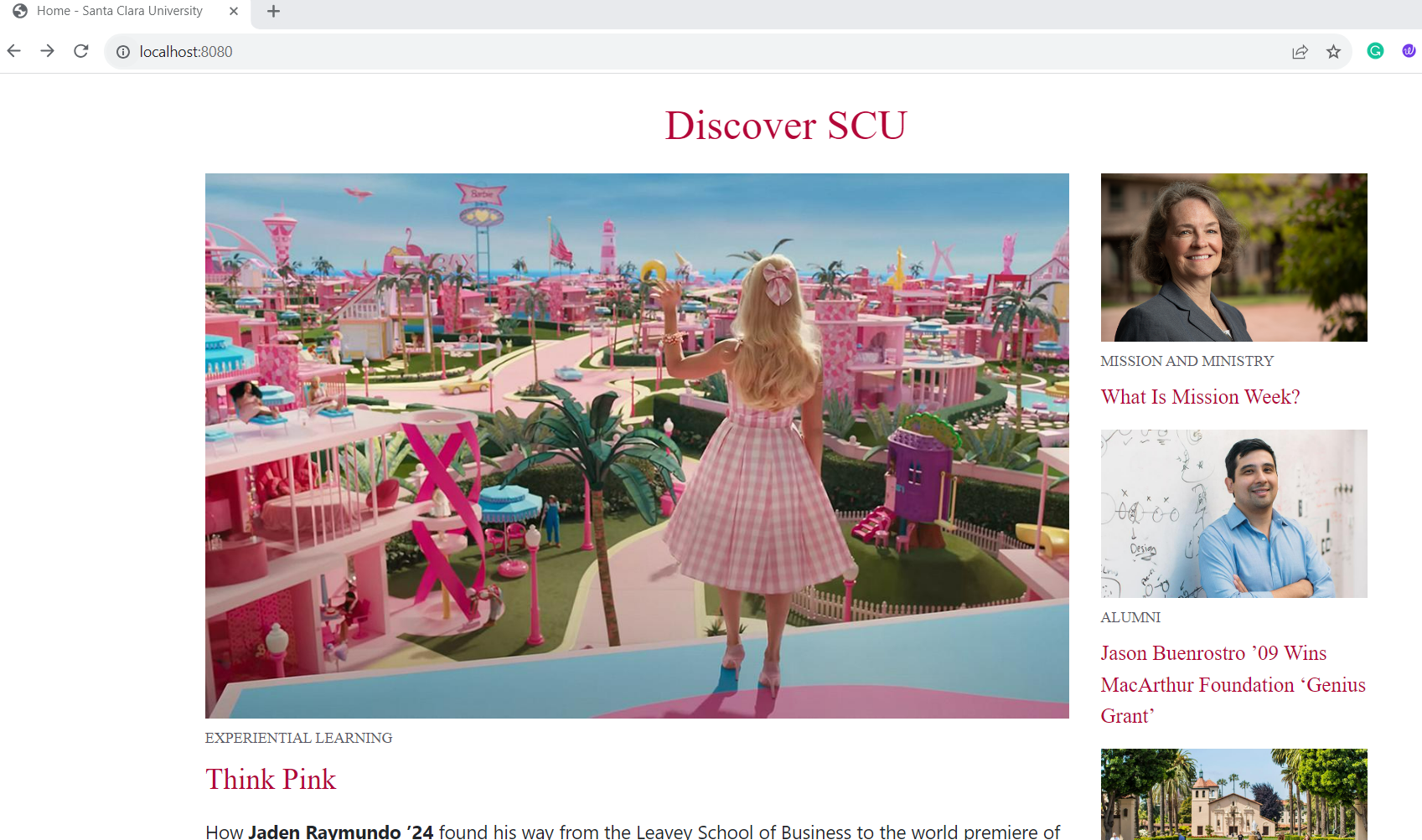
Compile using: javac ServerApp.java

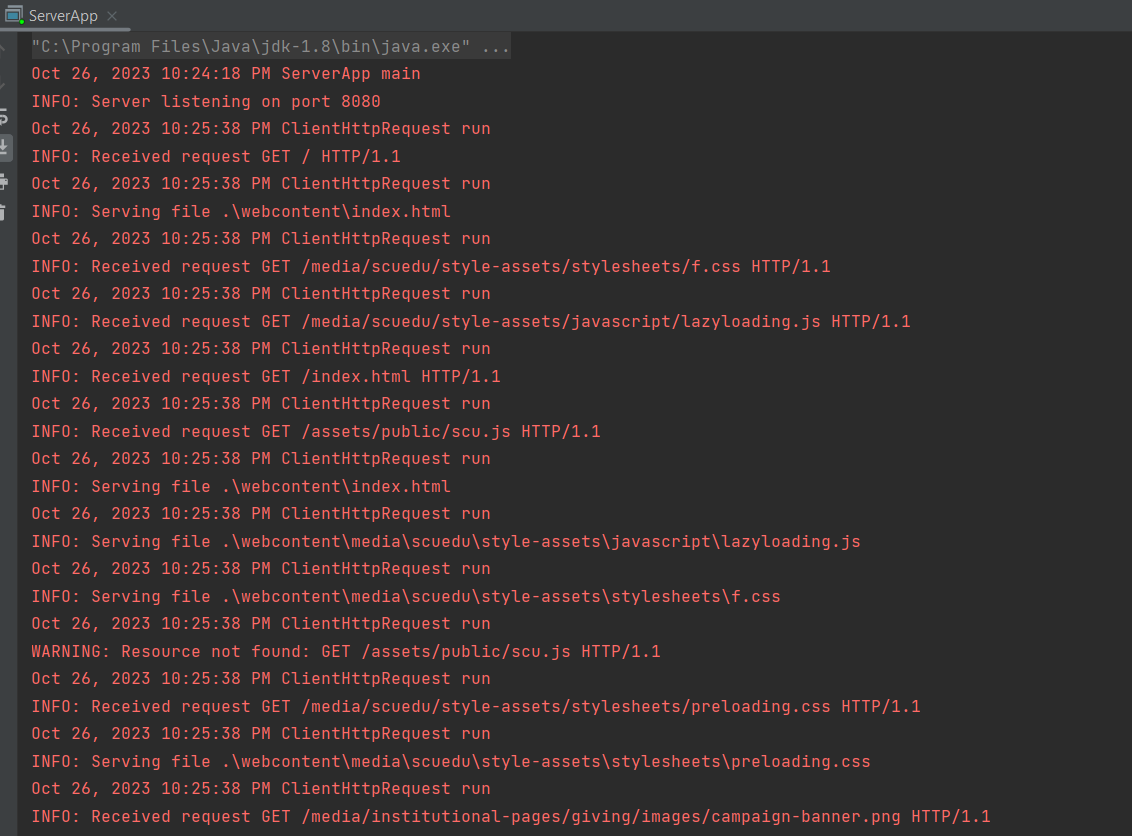
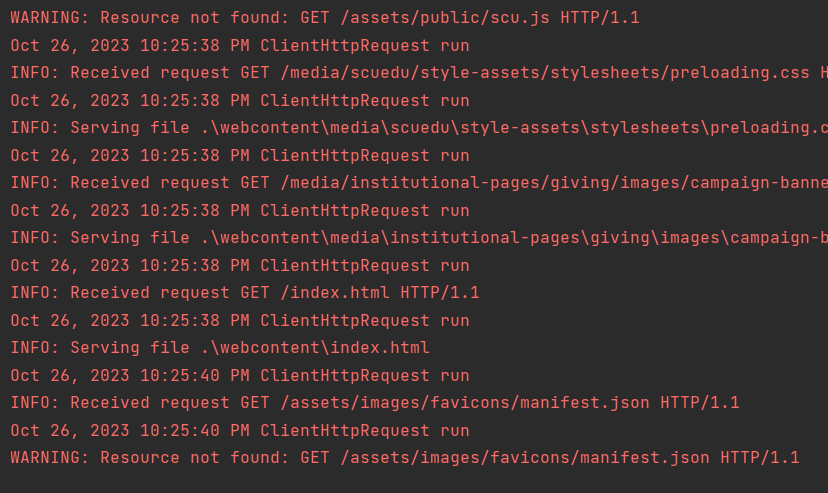
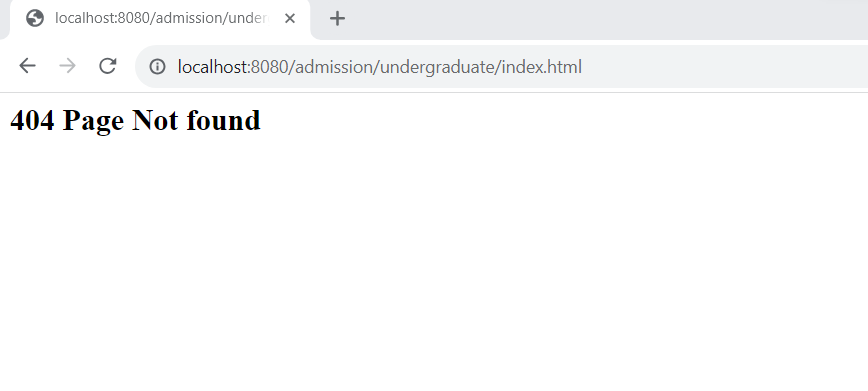
Run using: java ServerApp -document\_root “./webcontent” -port 8080

4) **Snapshots**

**Default WebPage:**



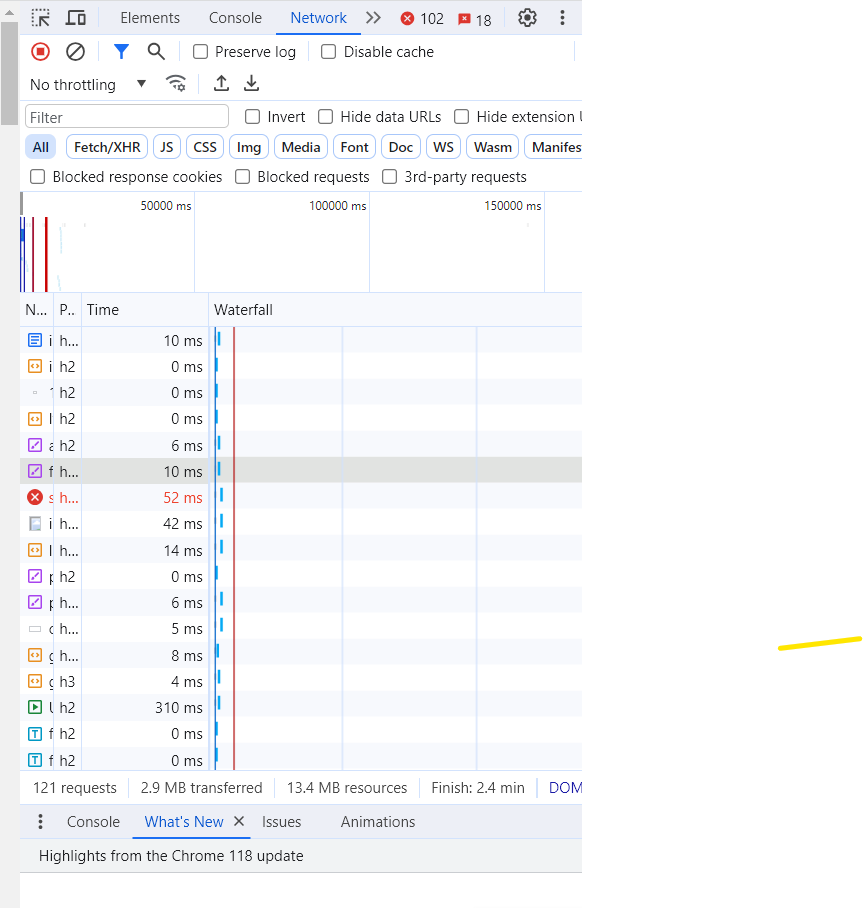
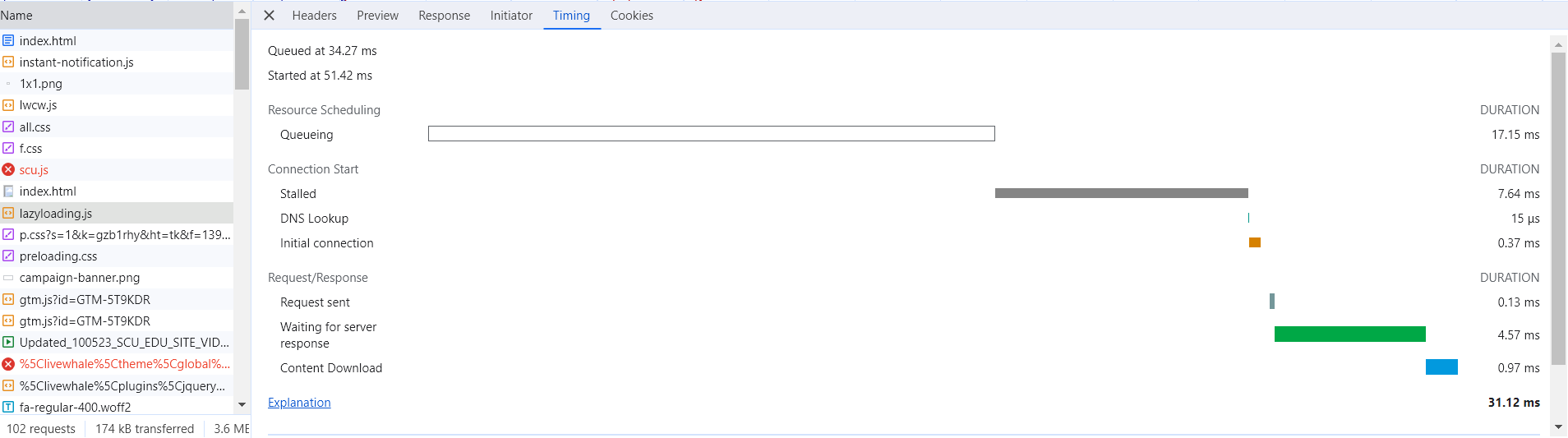
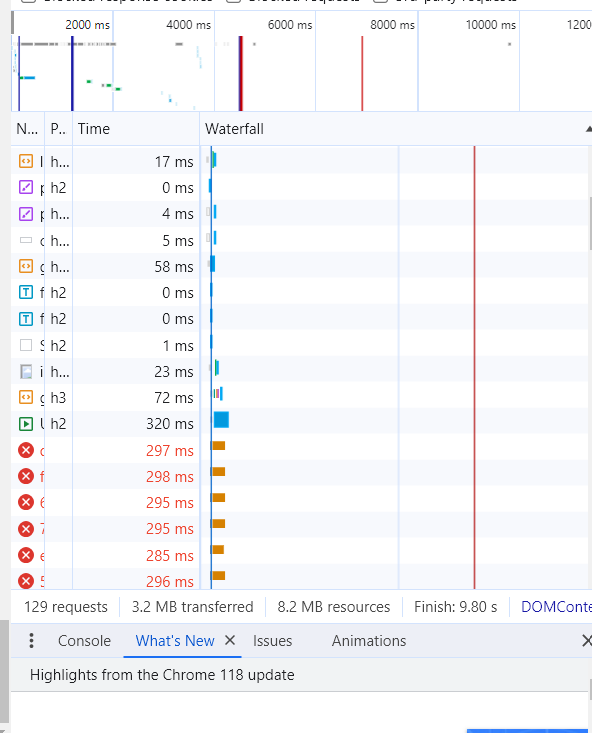
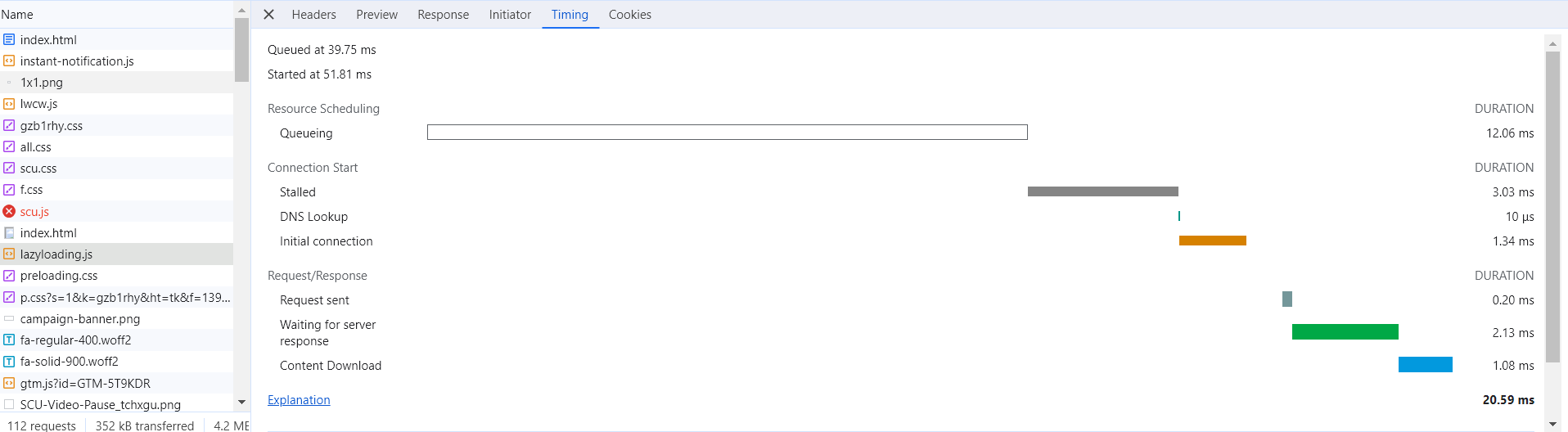
Images  


Logs  
  
  
  
  
  
  
  
Page not Founds  
  
  


Extra Credit

The ServerApp implements an idle connection timeout mechanism that efficiently manages the thread pool by periodically checking and closing idle connections  
  
The timeout has been implemented using 2 mechanisms using concurrent hashmap to close the connections and using Thread interrupts to close the connection  
  
This has following advantages and implementations:

1. More control and efficient use of thread pool
2. Efficient resource utilization
3. We can use active connections to keep track of alive connections
4. Here I have created a thread pool using java executor service which gives more control over thread handling and management.
5. I have also created a thread which checks for idle connections every 60 seconds and closes the connection if a thread is interrupted
6. No stale or closed connections (Thread pool contains 20 (changed to 13 via experimenting)) threads which are always active

Using Multi-threading without active connections and idle timeout  
  
  
  
Using Multi-threading with active connections and idle timeout  
  
  
  
  
Less resources and less finish time