Team information:
Kristie Hunter c3huntes
Yuan Meng g4mengyu
Artem Tselikov g4tselik
Mengqi Zhang g4zhangm

Description:
CONNECT is an online platform that connects the freelancers with the employers.
It allows the freelancers to search for job offers and the employers to search for available freelancers.
CONNECT has three main actors: the employer who posts the job offer, the freelancer who is interested in the offer, and a platform that brings the parties together.

It is expected to perform the following list of HTTP Requests:

If the user is not logged in:

GET /	— Redirect to the index page for login/signup options
GET /home	— Redirect to the index page for login/signup options
GET /signup	— Redirect to the signup page
GET /login	— Redirect to the login page
POST /signup	— Send the user object to the server
POST /login	— Send the user's inputs (email and password) to the server

If a regular user(non admin) is logged in:

GET /home	— Redirect to the user's home page
GET /users	— Retrieve all known users
GET /users:email	— Retrieve the user information of the user with the given user email
GET /freelancers	— Retrieve all known freelancers
GET /freelancers:email	— Retrieve the detailed profile information of the freelancer with the given email
GET /edit	— Redirect to the edit profile page
POST /edit	— Save the user's profile information
GET /inbox	— Redirect to the user's inbox page
GET /message	— Redirect to the message page where the user can create and send a message
GET /messages:id	— Retrieve the message with the given message id
POST /messages	— Send the new message data to the server
GET /rate:email	— Redirect to the rating page of the user with the given email
POST /rate	— Send the new rating to the server
GET /admin	— Triggers GET /home, since the user is not an admin
GET /todo	— Retrieve all known unfinished jobs (jobs in progress)
GET /todo:email	— Retrieve the unfinished job (job in progress) of user with the given email
GET /offers	— Retrieve all known offers
GET /offers:id	— Retrieve the job offer information with the given id

GET /editoffer:id — Redirect to the edit page of the offer with the given id
POST /editoffer:id — Send the user's inputs of the edit offer page to the server
GET /newoffer — Redirect to the page to create a new job offer
POST /newoffer — Send the new offer information to the server

If a admin is logged in (admin has all the above functionalities plus the following):
GET /admin — Entry point for an Admin user
GET /adminHome — Redirect to the Admin Home Page
POST /edit/userEmail — Edit an Employer or Freelancer Profile
DELETE /user — Delete a User (Freelancer, Employer or Admin)

Security:

1. Authentication
    a. In our system, a user must log in before being allowed to access any page.
    b. The router checks to see if there is a user logged in the storage for the browser, and if not, redirects to the main login page where they can log in using their email and password or the Facebook 3rd Party Login.

2. Authorization
    a. A Freelancer or Employer has no access to admin functions. Anything accessible at /adminHome requires that the logged in user have an Admin status, otherwise they are redirected to the normal / home page.
    b. A Freelancer or Employer cannot edit another user's profile. If a user of that level attempts to access another profile by /edit/userEmail, they are redirected to the Freelance or User Listing page.
    c. An Admin cannot edit another admin's profile. If an Admin attempts to access another admin's profile by /edit/AdminEmail, they are redirected to User Listing page.
    d. A Freelancer cannot post an offer. If they attempt to post an offer by /newoffer, they get redirected.
    e. An Employer cannot accept an offer. Where a Freelancer can accept an offer via /offer/offerName, the Employer can only edit the offer when they go to /offer/offerName.

3. Confidentiality
    a. Passwords are encrypted on sign-up when they are stored in the database.
    b. Despite the handout saying that Admin can change a user's password, all documentation suggests that is a gross breach of confidentiality. Therefore an Admin may access and change the contents of a user's profile via /edit/userEmail, but they may not change their password.
    c. The contents of any sent message are only accessible by the recipient.
    d. The contents of an inbox is only accessible by the logged in user.
    e. The contents of an offer is only accessible by the Employer who made it or an Admin.

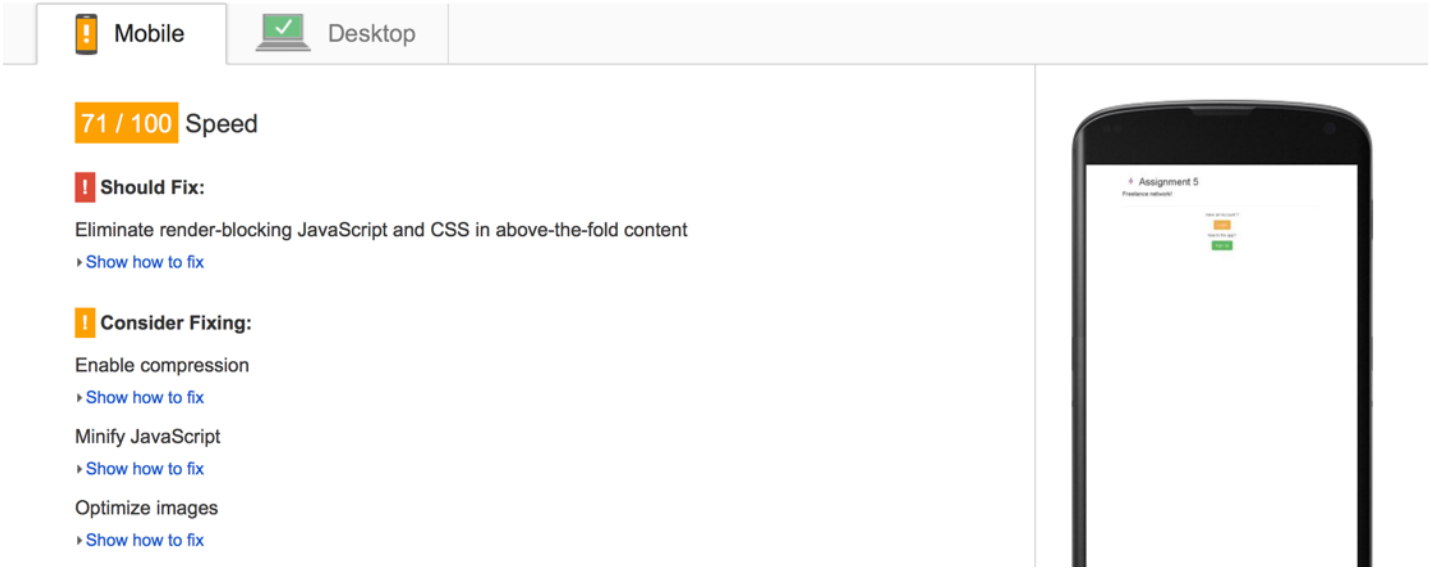
4. Data/Message Integrity
    a. Messages are stored using a Universally Unique Identifier. Therefore, as soon as a user sends a message to another user (posts it to the database), the id by which to access it is generated using uuid, which makes the probability of figuring it out and tampering with the message contents next to none.
    b. When an Employer posts an offer and once a Freelancer accepts the job, the ability to delete the offer is removed. This means that there is a permanent record of an Freelancer accepting a job (the Employer then having the ability to Accept or Reject the Freelancer, but not delete the Offer).

5. Accountability
    a. Each user is tracked by IP address when they log in. This means that if there is an attack on the system, and the IP address is traced the attacker can be determined by matching the last logged in IP.
    b. Messages, Offers and Rate/Comments are time stamped. This means that any question about when an interaction between two users occurred is a matter of permanent record in the database.

Performance:

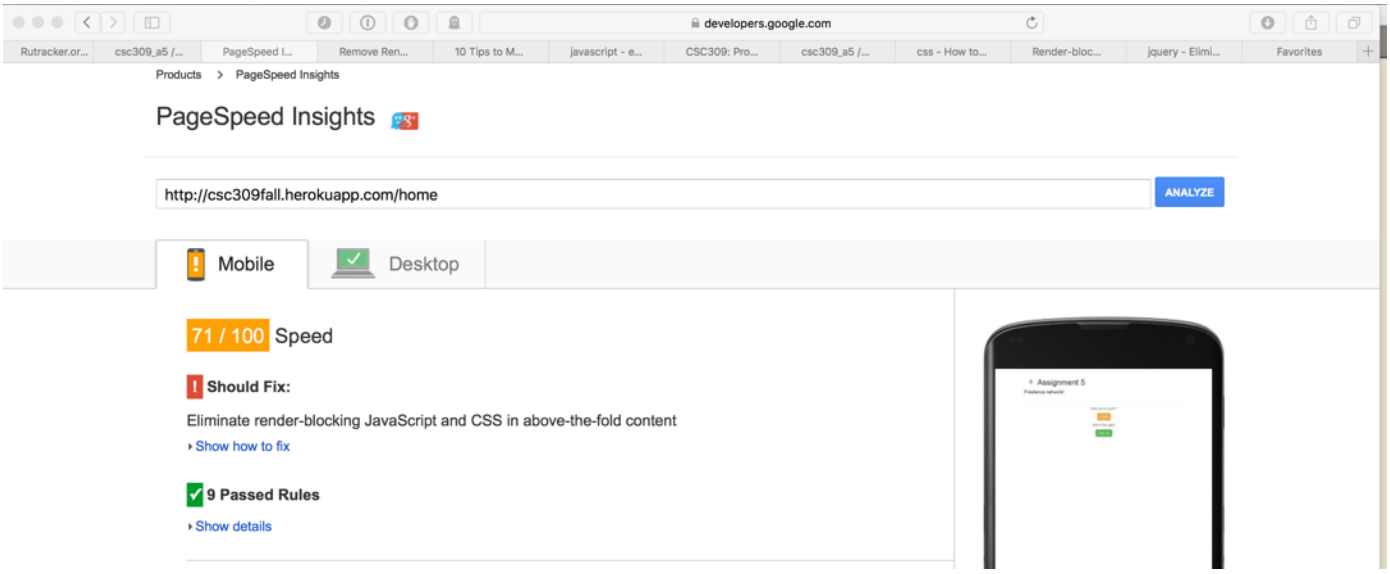
In order to measure application performance we used google page speed insights.
https://developers.google.com/speed/pagespeed/insights/

On the first try we got the following result:

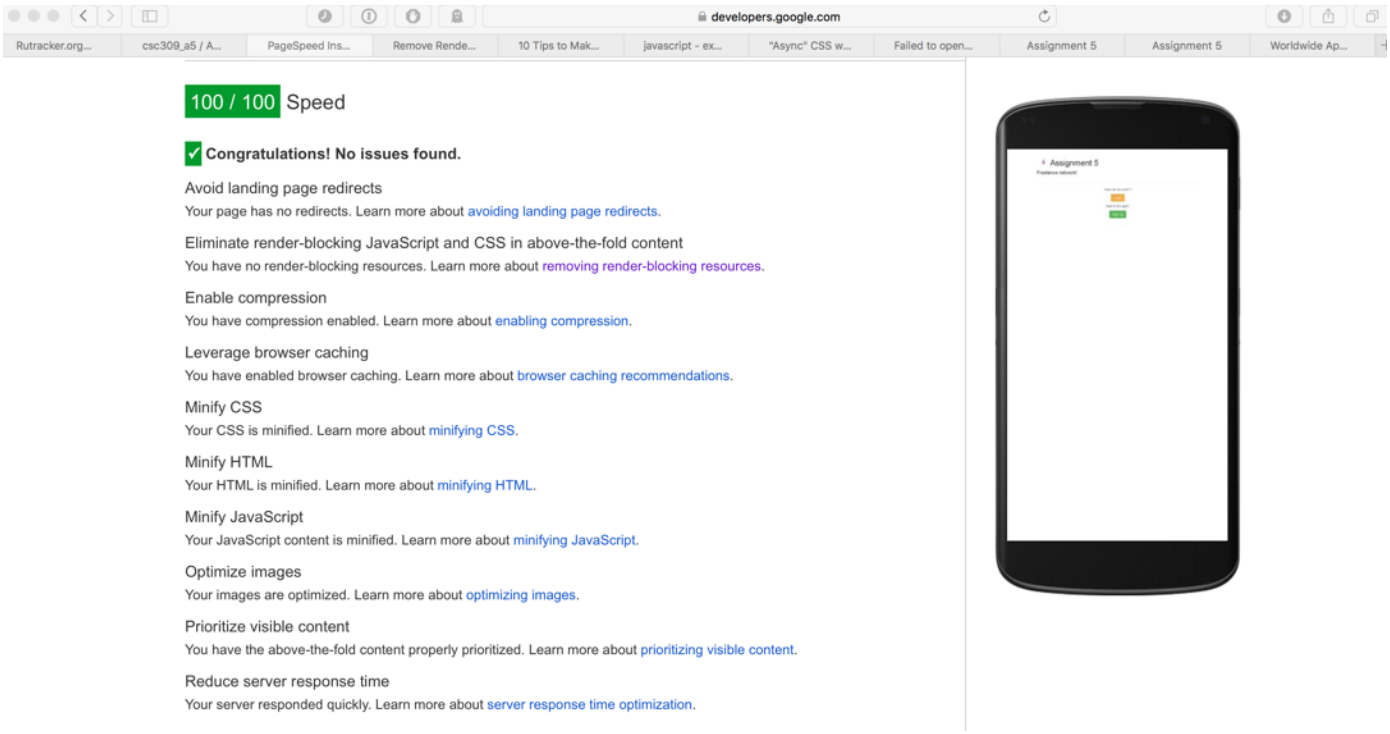


Then we added compression module to the app, which enabled gzip so that all scripts were compressed.
Also we optimized images using these recommendations https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/image-optimization

But result remained same.



Then we made all javascipts files load asynchronously and placed css at the end of body tag.
When css was located in head tag it blocked the loading of page, until browser downloaded external resources.



Youtube link:

Collaboration:
https://bitbucket.org/csc309_a5/a5
The repository will become publicly accessible after the deadline