Computer Networks Project 2 Report

1. Introduction:

Project Overview: This project involved securing a web application by integrating authentication, implementing security enhancements, optimizing performance, and deploying on a serverless architecture using Netlify and Firebase.

Technology Stack:

- Frontend: HTML, CSS, JavaScript, Netlify
- Backend: Firebase Authentication, Firestore Database, Netlify Functions
- Security: CSRF protection, HTTPS, authentication security
- Performance Optimization: Netlify CDN

Team Responsibilities:

- Rian Merritt: Database integration and backend communication
- Sebastian Geer: Frontend Development and login UI
- Zeba Karobi: Final report and network analysis
- BhaskarTeja Pidugu: Edit final report and network analysis

2. Security Enhancements:

Authentication & Database Security

- Firebase Authentication was used to handle user authentication, allowing users to sign in using email/password authentication.
- Firebase automatically handles password hashing and storage using industry-standard security protocols, ensuring user credentials are protected.

Netlify HTTPS & Security Features

- Netlify enforces HTTPS by default, ensuring all data is encrypted using TLS.
- Enabling automatic certificate renewals prevents expired certificates from compromising security.

CSRF & Session Management

- Firebase uses authentication tokens to ensure that user requests are valid.
- Authentication tokens are generated securely and validated before processing any request.
- Implemented token-based authentication in API calls to mitigate CSRF attacks
- **CSRF Protection Strategies:**
    - Implemented Firebase authentication tokens that expire after a session.
    - Restricted authentication operations to only authorized domains.
    - Used security rules to prevent unauthorized database access.

3. Database Integration:

Firebase Firestore for Data Storage

- User authentication data is stored in Firestore under /users collections.
- Each authenticated user has a document containing user profile details, timestamps, and authentication history.

Using Firebase Authentication in the UI

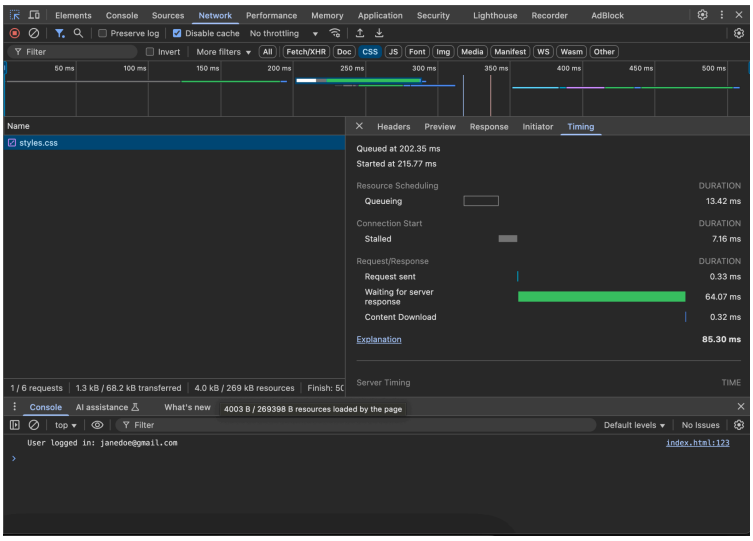
- Below is a code snippet for integrating Firebase Authentication into the frontend:

```
import { app } from "../../firebaseConfig.js";
import { getAuth, signInWithEmailAndPassword } from "firebase/auth";

const auth = getAuth(app);

document.getElementById("login-form").addEventListener("submit", (e) => {
  e.preventDefault();
  const email = document.getElementById("email").value;
  const password = document.getElementById("password").value;

  signInWithEmailAndPassword(auth, email, password)
    .then((userCredential) => {
      // Successful login
      alert("Login Successful");
      // Redirect to homepage
      window.location.href = "index.html";
    })
    .catch((error) => {
      alert("Login error: " + error.message);
    });
});
```

4. Performance Optimization and Scalability:

Built-in Netlify CDN

- Netlify's Content Delivery Network (CDN) ensures fast website performance by caching static assets and serving them from edge locations closest to users.



5. Deployment and infrastructure:

Hosting on Netlify

- Steps taken to deploy the frontend:

    - Pushed the code to GitHub
    - Connected the repository to Netlify
    - Configured build settings and deployed the site
- Firebase authentication and Firestore database were linked to Netlify using Firebase Admin SDK.

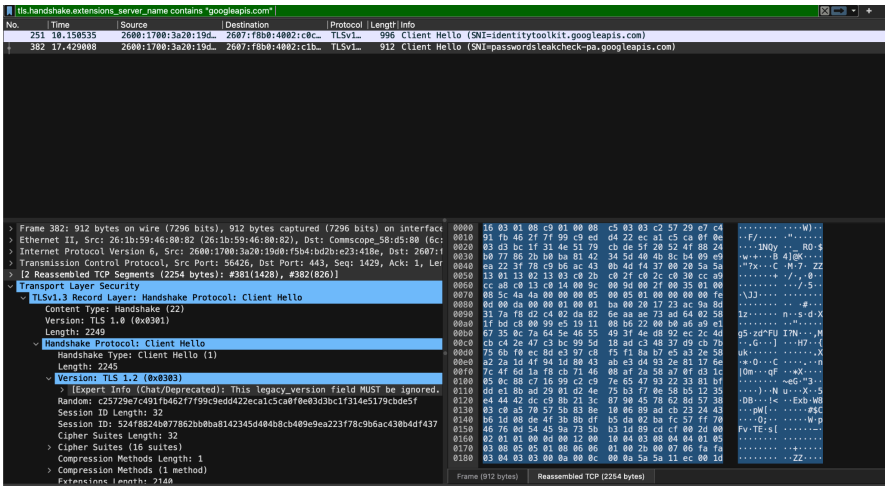
Serverless Backend Implementation

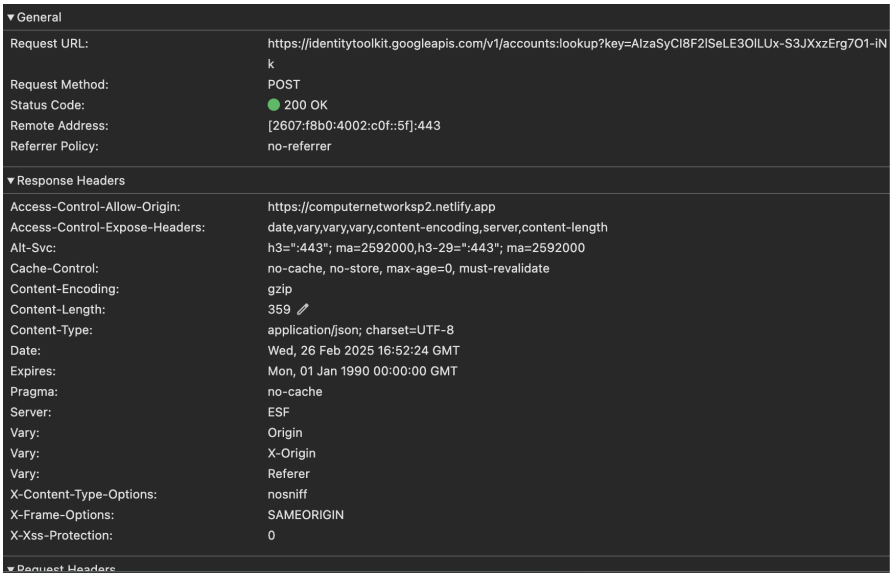
- Instead of running a dedicated backend server, Netlify Functions process requests dynamically.
- This ensures scalability while keeping infrastructure costs low.

6. Network Monitoring and Security Analysis:

Wireshark Network Analysis Results

- screenshots of wireshark network logs (filtered out for https requests to googleapis.com) and dev tool logs of firebase api requests

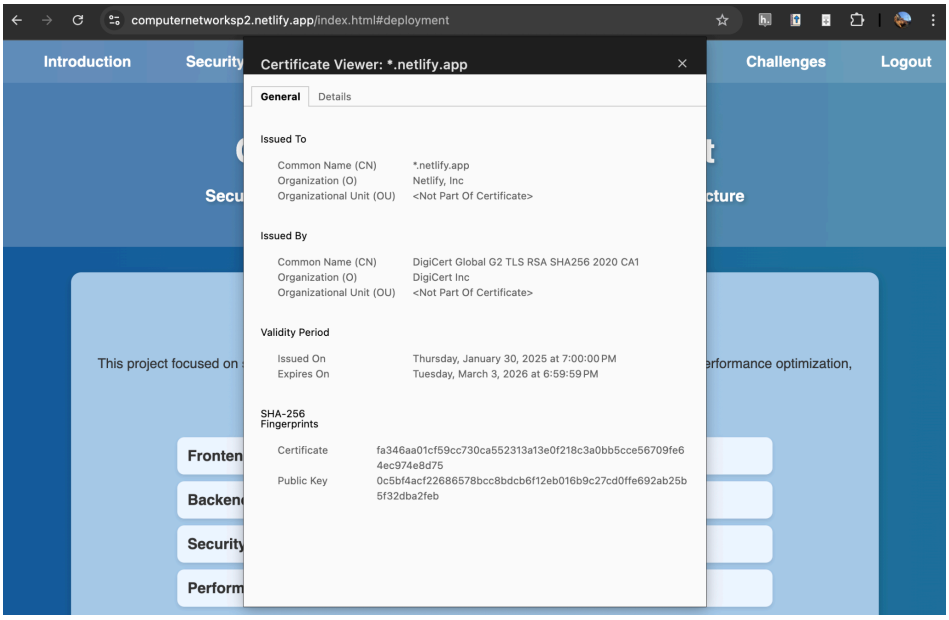




- What is monitored?
    - Firebase requests, authentication tokens, Netlify API calls

Testing HTTPS & Authentication Security

- Screenshot of SSL certificate details



7. Challenges:

Challenge #1: Firebase Authentication Issues
Solution: Ensured proper Firebase project configuration and updated the authentication rules.

Challenge #2: Capturing Firebase traffic with wireshark
Solution: Disabled QUIC in Chrome to force Firebase traffic over TLS (which Wireshark can analyze).