**OverView of What was done so Far in Backend:**

**Here’s a concise recap of everything we’ve done so far, end-to-end, on the backend side:**

**1. Environment and Tooling**

* **RHEL 9 VM in VirtualBox, with**
  + **GCC 11, CMake, Git, OpenSSL dev**
  + **Crow (header-only C++ microframework) under include/Crow**
* **Project scaffolding under /opt/buffer\_overflow:**
  + **src/ for main.cpp**
  + **data/ for auth.csv & customers.csv**
  + **scripts/ for create\_user.sh, start.sh, stop.sh**

**2. User Creation & Authentication**

* **scripts/create\_user.sh**
  + **Prompts for username, role (admin/user), password**
  + **Enforces min-length (8 for admin, 6 for user)**
  + **Hashes password with SHA-256 (via OpenSSL) on one line**
  + **Appends username,hash,role to data/auth.csv**
* **check\_credentials() in main.cpp**
  + **Reads each line of data/auth.csv**
  + **Splits out the username, hash, and role**
  + **Compares stored hash to sha256(password)**
  + **Returns true only if both match**

**3. Server Setup (main.cpp)**

* **Logging utility log(...) to prepend timestamps to app.log and stdout**
* **SHA-256 helper sha256(string) wrapping OpenSSL’s SHA256()**
* **Data model struct Customer +**
  + **load\_customers() to parse data/customers.csv into std::vector<Customer>**
  + **save\_customers() to overwrite data/customers.csv from that vector**

**4. Crow Routes Implemented**

| **Endpoint** | **Method** | **Behavior** |
| --- | --- | --- |
| **Serve SPA shell /** | **GET** | **Streams UI/index.html (or UI/login.html) as the root page** |
| **Login /api/login** | **POST** | **Parses JSON { user, pass }, calls check\_credentials(), returns 200 OK or 401 Unauthorized plus {"status":"ok"}/{"status":"error"}** |
| **List customers /api/customers** | **GET** | **Loads all customers, returns {"customers":[ … ]} JSON array** |
| **Create customer /api/customers** | **POST** | **Reads JSON payload, assigns a new id, appends to CSV, returns 201 Created** |
| **Update customer /api/customers/:id** | **PUT** | **Finds by id, updates fields in memory, re-writes CSV, returns 200 OK or 404 Not Found** |
| **Delete customer /api/customers/:id** | **DELETE** | **Removes by id from memory, re-writes CSV, returns 200 OK or 404 Not Found** |
| **Export CSV /api/export** | **GET** | **Streams data/customers.csv with Content-Type: text/csv; Content-Disposition: attachment** |
| **Health check /health** | **GET** | **Returns plain-text “Server running”** |
| **Test JSON /api/hello** | **GET** | **Returns {"message":"Hello!"}** |

**5. Build & Link**

* **CMakeLists.txt updated to:**
  + **Include Crow and ASIO headers**
  + **find\_package(OpenSSL REQUIRED)**
  + **target\_link\_libraries(app-server PRIVATE OpenSSL::Crypto)**
* **Rebuild cycle:**
* **cd build**
* **cmake ..**
* **make -j4**
* **./scripts/stop.sh**
* **./scripts/start.sh**

**6. Smoke-Testing with curl**

1. **Login:**
2. **curl -i -X POST …/api/login -d '{"user":"jeevan","pass":"…"}'**
3. **# → 200 OK + {"status":"ok"} or 401**
4. **CRUD:**
5. **curl -i GET …/api/customers**
6. **curl -i POST …/api/customers -d '{…}'**
7. **curl -i PUT …/api/customers/1 -d '{…}'**
8. **curl -i DELETE …/api/customers/1**
9. **Export:**
10. **curl -i GET …/api/export -o customers.csv**

**All returned the expected status codes and payloads.**

**With the backend fully wired up and verified, we’re now ready to layer on the AngularJS front-end and role-based UI next.**