

Foundations of Computer Security

Saurav Haldar (2022464)

Prince Kumar (2022378)

Nikhil Rathore (2022321)

Aditi Sharma (2022025)

Project Description

Tech Stack:

- Frontend : Js & React
- Backend : Django
- Database : SQLite
- Web Server : Nginx & Gunicorn

Dependencies:

- Internal Dependencies
 - backend/api/models.py - defines db models
 - backend/api/views.py - implements API business logic
 - backend/api/urls.py - routes API requests to the correct views
 - backend/api/serializer.py - converts db objects into JSON format
 - backend/api/admin.py - registers models to Django Admin
 - backend/api/tests.py - used for unit testing the API (not implemented)
 - backend/settings.py - core django settings for db, middleware ,security, etc
 - backend/wsgi.py - WSGI application entry point for Gunicorn
 - frontend/src/components - React UI components
 - frontend/src/pages - react pages
 - frontend/src/api.js - manages frontend API calls to Django backend.
 - frontend/src/styles - css files for pages.
- External Dependencies
 - Python Backend
 - Django (5.1.6) - Core Django framework
 - djangorestframework (3.15.2) - API handling for Django

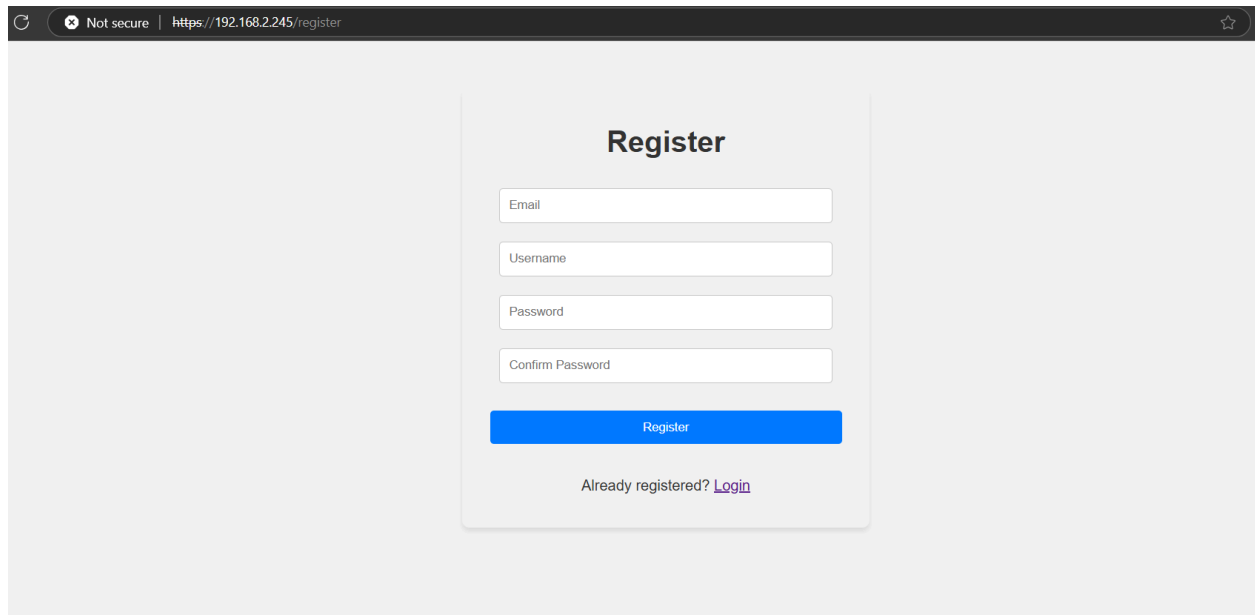
- djangorestframework_simplejwt (5.4.0) - JWT authentication for APIs
- django-cors-headers (4.7.0) - Handles CORS between frontend & backend
- pillow (11.1.0) - Handles image uploads in Django
- bcrypt (3.2.0) - Secure password hashing
- PyJWT (2.3.0) - Handles JWT authentication
- pyOpenSSL (21.0.0) - Used for Self-signed SSL/TLS certification
- python-dotenv (1.0.1) - Loads environment variables from `.env` file
- requests (2.25.1) - Handles API requests.
- unattended-upgrades (0.1) - Manages automatic system updates
- Frontend
 - react (19.0.0) - Core React library
 - react-dom (19.0.0) - React DOM rendering
 - react-router-dom (7.2.0) - Handles frontend routing
 - axios (1.8.1) - Handles API request to backend
 - dayjs (1.11.13) - Handles date formatting
 - moment (2.30.1) - Alternative date/time library
 - jwt-decode (4.0.0) - Decodes JWT tokens in frontend
 - vite (6.2.0) - Development server & build tool for react
 - vitejs/plugin-react (4.3.4) - Optimizes React for Vite

Login/Register/Home page showing HTTPS in use.

Login Page

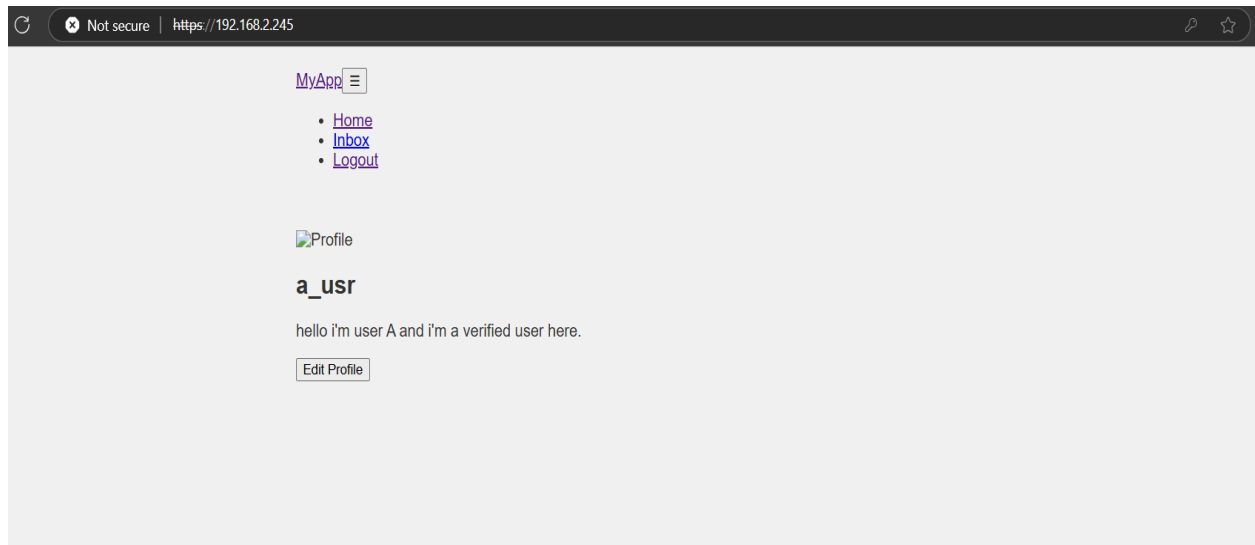
The screenshot shows a web browser window with the address bar displaying 'https://192.168.2.245/login'. The page content is a login form with the title 'Login'. It contains two input fields: 'Email' and 'Password'. Below these fields is a blue button labeled 'Login'. At the bottom of the form, there is a link that says 'New user? Register'.

Register Page



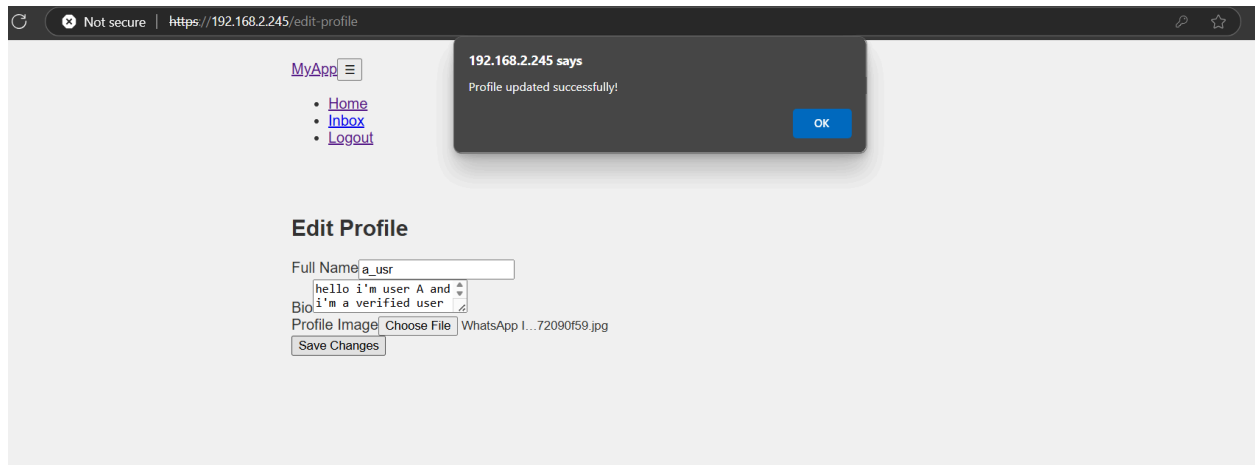
A screenshot of a web browser showing a registration page. The browser's address bar displays "https://192.168.2.245/register" with a "Not secure" warning. The page has a light gray background. In the center, there is a white rounded rectangle containing the title "Register" in bold. Below the title are four input fields labeled "Email", "Username", "Password", and "Confirm Password". A blue "Register" button is positioned below these fields. At the bottom of the white box, it says "Already registered? [Login](#)".

Home Page



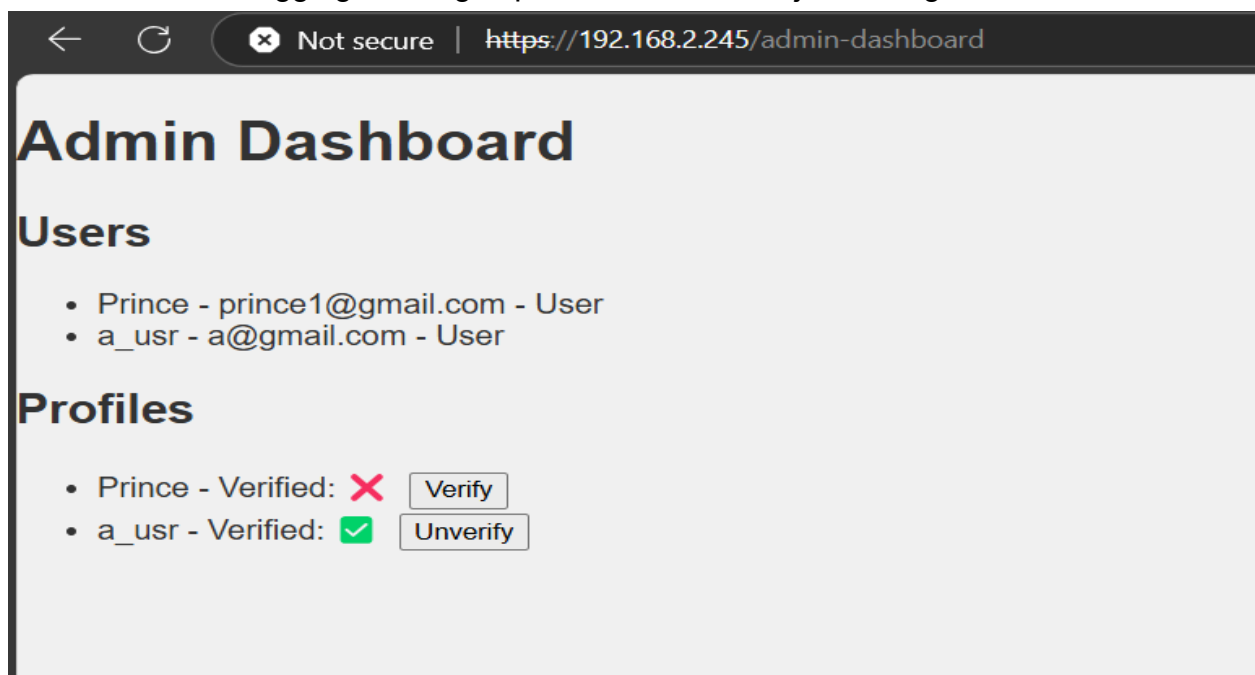
A screenshot of a web browser showing a home page. The browser's address bar displays "https://192.168.2.245". The page has a light gray background. On the left side, there is a "MyApp" header with a hamburger menu icon. Below it is a list of links: "Home", "Inbox", and "Logout". Further down, there is a "Profile" section with a small profile picture icon, the username "a_usr", and a message: "hello i'm user A and i'm a verified user here." Below the message is an "Edit Profile" button.

Edit-Profile Page



Admin Dashboard Screenshot:

To get access to the admin dashboard you have to just login like a normal user but the credentials should be of super user then only you can have the access to the admin dashboard. After logging in using superuser credentials just change the route.



Database Schema Screenshots:

```

Schema for table: api_chatmessage
- id (INTEGER)
- message (varchar(1000000000))
- is_read (bool)
- date (datetime)
- receiver_id (bigint)
- sender_id (bigint)
- user_id (bigint)
Schema for table: api_profile
- id (INTEGER)
- full_name (varchar(500))
- bio (varchar(1000))
- verified (bool)
- user_id (bigint)
Schema for table: api_user
- id (INTEGER)
- password (varchar(128))
- last_login (datetime)
- is_superuser (bool)
- first_name (varchar(150))
- last_name (varchar(150))
- is_staff (bool)
- is_active (bool)
- date_joined (datetime)
- username (varchar(100))
- email (varchar(254))
Schema for table: api_user_groups
- id (INTEGER)
- user_id (bigint)
- group_id (INTEGER)
Schema for table: api_user_user_permissions
- id (INTEGER)
- user_id (bigint)
- permission_id (INTEGER)

```

```
Schema for table: auth_group
- id (INTEGER)
- name (varchar(150))
Schema for table: auth_group_permissions
- id (INTEGER)
- group_id (INTEGER)
- permission_id (INTEGER)
Schema for table: auth_permission
- id (INTEGER)
- content_type_id (INTEGER)
- codename (varchar(100))
- name (varchar(255))
Schema for table: django_admin_log
- id (INTEGER)
- object_id (TEXT)
- object_repr (varchar(200))
- action_flag (smallint unsigned)
- change_message (TEXT)
- content_type_id (INTEGER)
- user_id (bigint)
- action_time (datetime)
Schema for table: django_content_type
- id (INTEGER)
- app_label (varchar(100))
- model (varchar(100))
```

```
Schema for table: django_migrations
- id (INTEGER)
- app (varchar(255))
- name (varchar(255))
- applied (datetime)
Schema for table: django_session
- session_key (varchar(40))
- session_data (TEXT)
- expire_date (datetime)
Schema for table: token_blacklist_blacklistedtoken
- blacklisted_at (datetime)
- token_id (bigint)
- id (INTEGER)
Schema for table: token_blacklist_outstandingtoken
- token (TEXT)
- created_at (datetime)
- expires_at (datetime)
- user_id (bigint)
- jti (varchar(255))
- id (INTEGER)
```

Commands used for setup the server and hosting of the project.
A total of approx 1800+ commands were used to setup the entire server.

As our whole project is on github so we cloned our project from the github
> ``git checkout -b master clone https://github.com/Clever-Anomaly/FCS.git``

After cloning the project we need to set up the Virtual Environment.

> ``python3 -m venv env``

Note: *Our project on the github has already has the virtual environment created so we just need to activate it as all the requirements and dependencies are already in that env*

> ``cd FCS``

For virtual environment activation

> ``source env/Scripts/activate``

Now if any requirements or dependencies are required you can now just install that.

> ``cd backend``

We'll now remove the existing SQLite database and then runs django migrations to set up the tables and after that we'll create a superuser for admin login.

> ``rm -f backend/db.sqlite3``

> ``python3 manage.py makemigrations``

> ``python3 manage.py migrate``

> ``python3 manage.py createsuperuser``

Note: if you encounter any issue regarding migration. Just run the below commands to fix that. These commands are just for removing and recreating the migrations.

> ``rm -rf backend/api/migrations/*``

> ``find . -path "__pycache__" -delete``

> ``mkdir backend/api/migrations``

> ``touch backend/api/migrations/__init__.py``

> ``sudo apt install python3-pip``

> ``sudo apt install python3-django``

> ``sudo apt install python3-djangorestframework``

> ``sudo apt install python3-djangorestframework-simplejwt``

> ``sudo apt install python3-django-cors-headers``

> ``python3 manage.py makemigrations api``

> ``python3 manage.py migrate``

> ``python3 manage.py showmigrations``

Now we'll run the backend manually by starting the django development server manually and ensures that backend is running before gunicorn+nginx setup

> ``python3 manage.py runserver 0.0.0.0:8000``

Gunicorn Setup & Service Configuration:

```
> `sudo apt install python3-gunicorn`  
> `sudo nano /etc/systemd/system/gunicorn.service`
```

Paste the below snippet

```
`[Unit]  
Description=Gunicorn daemon for Django FCS  
After=network.target  
  
[Service]  
User=iiitd  
Group=www-data  
WorkingDirectory=/home/iiitd/FCS/backend  
ExecStart=/usr/bin/gunicorn --workers 3 --bind  
unix:/home/iiitd/FCS/backend/gunicorn.sock backend.wsgi:application  
  
[Install]  
WantedBy=multi-user.target`
```

```
> `sudo systemctl daemon-reload`  
> `sudo systemctl start gunicorn`  
> `sudo systemctl enable gunicorn`  
> `sudo systemctl restart gunicorn`
```

To confirm gunicorn is installed and running

```
> `sudo systemctl status gunicorn`  
> `which gunicorn`
```

Frontend Deployment:

Here we'll install some frontend dependencies and also build the react app for production

```
> `cd /home/iiitd/FCS/frontend`  
> `npm install`  
> `npm run build`
```

Nginx Configuration:

```
> `sudo apt install nginx -y`  
> `sudo systemctl start nginx`  
> `sudo systemctl enable nginx`  
> `sudo nano /etc/nginx/sites-available/fcs`
```

Paste the below snippet

```
`server {
```



```

listen 80;
server_name 192.168.2.245;

root /home/iiitd/FCS/frontend/dist;
index index.html;

location / {
    try_files $uri /index.html;
}

location /api/ {
    proxy_pass http://unix:/home/iiitd/FCS/backend/gunicorn.sock;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
}

```

```

> `sudo ln -sf /etc/nginx/sites-available/fcs /etc/nginx/sites-enabled/`
> `sudo nginx -t`
> `sudo systemctl restart nginx`

```

To confirm nginx is running

```
> `sudo systemctl status nginx`
```

>`curl --unix-socket /home/prince/FCS/backend/gunicorn.sock <http://127.0.0.1/api/>` for testing the backend

Giving permissions to the nginx for frontend

```

> `sudo chmod -R 755 /home/prince/FCS/frontend/dist`
> `sudo chown -R www-data:www-data /home/prince/FCS/frontend/dist`

```

```
> `sudo systemctl restart nginx`
```

```

> `sudo chmod -R 755 /home/prince/FCS/frontend/`
> `sudo chmod -R 755 /home/prince/FCS/`

```

```
> `/usr/bin/gunicorn --workers 3 --bind unix:/home/prince/FCS/backend/gunicorn.sock backend.wsgi:application`
```

```
> `sudo chown www-data:www-data /home/prince/FCS/backend/db.sqlite3`  
> `sudo chown www-data:www-data /home/prince/FCS/backend`  
  
> `sudo systemctl restart gunicorn`  
> `sudo chmod 664 /home/prince/FCS/backend/db.sqlite3`  
> `sudo chmod 775 /home/prince/FCS/backend`  
> `sudo systemctl restart gunicorn`
```

Now Setting up the SSL/TLS with OpenSSL

```
> `sudo apt update && sudo apt install openssl -y`  
> `sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout  
/etc/ssl/private/fcs.key -out /etc/ssl/certs/fcs.crt`
```

Now enter the necessary details and proceed

Now add the SSL/TLS certificate to the server and also we add the admin route.

```
> `sudo nano /etc/nginx/sites-available/fcs`
```

Note: Remove the previous all things and Paste the below snippet.(It is update and contains all the previous routes, etc.)

```
`server {  
    listen 80;  
    server_name 192.168.2.245;  
    return 301 https://$host$request_uri;  
}
```

```
server {  
    listen 443 ssl;  
    server_name 192.168.2.245;  
  
    ssl_certificate /etc/ssl/certs/fcs.crt;  
    ssl_certificate_key /etc/ssl/private/fcs.key;
```

```
    ssl_protocols TLSv1.2 TLSv1.3;  
    ssl_ciphers HIGH:!aNULL:!MD5;  
    ssl_prefer_server_ciphers on;
```

```
location /admin/ {  
    proxy_pass http://unix:/home/iiiitd/FCS/backend/gunicorn.sock;  
    proxy_set_header Host $host;  
    proxy_set_header X-Real-IP $remote_addr;  
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
```

```

        proxy_set_header X-Forwarded-Proto $scheme;
    }

    location / {
        root /home/iiitd/FCS/frontend/dist;
        index index.html;
        try_files $uri /index.html;
    }

    location /api/ {
        proxy_pass http://unix:/home/iiitd/FCS/backend/gunicorn.sock;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}

```

> `sudo systemctl restart gunicorn`

> `sudo systemctl restart nginx`

Self-signed certificate

```

iiitd@fcs13:/etc/ssl/certs$ cat fcs.crt
-----BEGIN CERTIFICATE-----
MIIEDTCCAvWgAwIBAgIUHFcEMz4ynR4ktmh+a154pXNnIIswDQYJKoZIhvcNAQEL
BQAwZUxhCzAJBgNVBAYTAkOMQ4wDAYDVQQIDAVEZWxoTEWMBQGA1UEBwwNT2to
bGEgUGhlc2UzMzEOMAwGA1UECgwFSUJVEQxMjg0Yjg0Yjg0Yjg0Yjg0Yjg0Yjg0
VQQDDA0xOTIuMTY4LjIuMjg0Yjg0Yjg0Yjg0Yjg0Yjg0Yjg0Yjg0Yjg0Yjg0Yjg0
aWl0ZC5hYy5pbjAeFw0yNTAyMjg0Yjg0Yjg0Yjg0Yjg0Yjg0Yjg0Yjg0Yjg0Yjg0
CQYDVQQGEwJTTjEOMAwGA1UECAwFRGVSaGkxMjg0Yjg0Yjg0Yjg0Yjg0Yjg0Yjg0
LTMxMjg0Yjg0Yjg0Yjg0Yjg0Yjg0Yjg0Yjg0Yjg0Yjg0Yjg0Yjg0Yjg0Yjg0Yjg0
LjE2OC4yLjI0NTEuMCQGCsGCSqGSIb3DQEJARYXCHJpbmNlMjg0Yjg0Yjg0Yjg0Yjg0
aW4wgaggEiMA0GCSqGSIb3DQEBBQAA4IBDwAwggEKAoIBAQAQDhRAK4XkmPrAsAk0wz
ESv6MLtkpNOGlebTPRK0+Ie7RNBA1gZ0o3eGiDrD4rs640iE/wNszf0bnsWagLZL
Fnjwg9EirIG4Q2rUnEw4r06r099un14m+rJgXFbuzSVuwiH5p14sS64c0ae49IVH
5mhXNsdb2eVyHlKpO5E1YHMLVj2lh/z2F00jjcUVniVZCnj/cgIZ70K1GmOjeJvc
/LLCWV5ExTJgLPxHYCw6f0tiEYdbLztqp4HwJLCzznTkxjKradXP8zVct9YhsPMK
l7kkCtLaV8+n/zxiPq51ecB0oNeCDqt61RmEhyzzfq3UwNiKfS7WstEHQ+a+qLd/
/g3vAgMBAAAGjUzBRMB0GA1UdDgQWBBSJJ9A8uPH1ESM1JBdL6WTTf4hDWzAFBgNV
HSMEGDAWgBSJJ9A8uPH1ESM1JBdL6WTTf4hDWzAPBgNVHRMBAf8EBTADAQH/MA0G
CSqGSIb3DQEBBQAA4IBAQLBK6pqswEXPgK2ifWTK30kf7BAwr78xuZB2Ujx0Y7
TNn22+CeVFSmJomHusZ7u+57TSXhektWqsthxR70Ja09TvgRcZuFCe92z+wliS+E
pN21rSpWFTyLrZDnqnbkRErnrpTdmScv9BaBx46h5DbuWmGhZw793ayE0869DvY
9HwpDL7SeNTfZolwynmu5PdnNElf5vQ9Py2LL+M0qykAGw3hQ0jXuYj5XQoH/FuT
Bixg8812iICHWQ29xKEi40mUY01cDhWR1a6h/MUsvsaXQVWNNFoofdGSKeHqMBLB
o/qPjMTbyUsL4/3F+x01xiDvdrYSiH0pW/VTNc64t+vu
-----END CERTIFICATE-----

```

the private key

```

iiitd@fcs13:/etc/ssl$ sudo cat /etc/ssl/private/fcs.key
-----BEGIN PRIVATE KEY-----
MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBAwggSkAgEAAoIBAQAQDhRAK4XkmPrAsA
k0wzESv6MLtkpNOGlebTPRK0+Ie7RNBA1gZ0o3eGiDrD4rs640iE/wNszf0bnsWag
LZLFnjwg9EirIG4Q2rUnEw4r06r099un14m+rJgXFbuzSVuwiH5p14sS64c0ae4
9IVH5mhXNsdb2eVyHlKpO5E1YHMLVj2lh/z2F00jjcUVniVZCnj/cgIZ70K1GmOje
Jvc/LLCWV5ExTJgLPxHYCw6f0tiEYdbLztqp4HwJLCzznTkxjKradXP8zVct9Yh
sPMKl7kkCtLaV8+n/zxiPq51ecB0oNeCDqt61RmEhyzzfq3UwNiKfS7WstEHQ+a+
qLd//g3vAgMBAAECggEABkyeLC/0IqkO+F8mHFojPrB/glr8R0vd0+kQocrgEM6Q
fEpVntG/apWt/wL8yrK2Ddy2aHkskhzTXFJqE1jM3Pzyq444umxi37ssg9FIg+mS
2pxo8kdvFt6+L04JA/CN1AC9fhjyWiZf7ajOKVU6HxDLwPMg8Btae4s7jks6LRW6
M/ATzLDJUWHAIImESj3oLJk5EBZDm8I6kerq6uqveUrUe4hrWYb7aw/V52Q0jek8L
PtaBeN2Th+fTe+Syfzqo5rtAazmpRSMa5/0JK1VX03PRRGnf1R2RaB8nwMU0aUk
kZh0263efeLI0rFvfa/vGsoiijuzukCdQ06xoHTfIQKBgQDu807XhIexBYJhixFY
I70w2uwhFryembgjYyHYkm8jeeTAjR7b++7bhbdEMvFhljrokfx40mKwGCed+/N5
9GNvjXkbEY5QXHsgy3LPCct91se8ymUf9T0+pWMq/AArpC2H5DXse0Tm5smwYLtx
eFHpkwsfrnxTeXe8NEoAgAggqQKBgQDxVrxNeRseYsLRsJAF2C+UpHcGwGQ0tk8D
/oRZ4h+PFzLCeF0qonhNKwsWPnoCS5ySlypoibAjHvJDRn9KelQHvehH2rFfnPHx
ZE8fLAE0jzct32iA05SaaGFkcnSbRzCakm9zpDMMj/c307pFJOsNIY0Q4AHhY9fp
gmereLmg1wKBgQDffq91oT4Rcue60IrmD7cTJRDK+is950V4I9aabIyLEA3Hk7cT
ZxbIOM9eIKRFiTPNwyALbekoVMcz3Wx0cXxplpSghkXbvEtsYRosCBGPMFQAFJ
3NyjKqoQoFfAefUKL23IXJvbpG1tDW0E+tcSszgqT9d1dB1r6T1R2N0pi0QKBgHLu
JVqXs+xyh+ioNs4xt+cHs49Ji+72awax7IGJiBHys1gOTgABw1ysh0kg86WoMHY2
+wExoOn40NJJs/i2Pt/i/3aBqa1HoCAR3Cp8Drs2bMmHsJK5FFF28G7VjcVS6QFE8
L0tZBN6UqNyzPd+zxyRHuBAzPXICx31xj0JVSmdAoGBAJ8DKPdZn8dRe+dnUwd7
aIykKnqYPsQn/QJz0GsExcRYXsulQXghb0ihrrJEuqbqJbDM7zpwN3/fiwxdb2WZ
Ia85qUK63F6iQcrpLnwib1Xc1XvJWn0WAUzeWoxzPL/I6kvme45shB97exJBLRR9
yep8k9n9rjMvOIWXnaallF5v
-----END PRIVATE KEY-----

```