

Take Home Assignment

This project is a simple web application developed as part of the take-home assignment for the junior software engineer position. The application is built using React.js for the frontend and Django for the backend. It is deployed on AWS using S3, EC2, VPC, Subnets, CloudFront and IAM.

Demo Link:

https://drive.google.com/file/d/161daVAKOoihpB7cVjylH-ynNsxLj2SzT/view?usp=drive_link

GitHub Link:

<https://github.com/Darshanum-27/TakeHomeAssignment>

Tech Stack Used

Frontend: React.js

Backend: Django Framework

Cloud Provider: Amazon Web Services

Installation

1) Backend Setup (Django)

Create a virtual environment:

```
python -m venv env
```

Activate the virtual environment:

```
On Windows: env\Scripts\activate  
On macOS and Linux: source env/bin/activate
```

2) Install Django:

```
pip install Django
```

3) Create a new Django project:

```
django-admin startproject backend
cd backend
```

- 4) Create a new Django app:

```
python manage.py startapp api
```

- 5) Add api to INSTALLED_APPS in backend/settings.py

```
Run migrations:
python manage.py migrate
```

- 6) Start the Django development server:

```
python manage.py runserver
```

Frontend Setup (React.js)

- 1) Create a new React app:

```
npx create-react-app frontend
cd frontend
```

- 2) Install additional dependencies:

```
npm install axios
```

- 3) Start the React development server:

```
npm start
```

Connecting Frontend and Backend

- 1) In your React components, use Axios to make API calls to your Django backend:

```
javascriptCopyimport axios from 'axios';
```

```
// Example API call
axios.get('http://localhost:8000/api/endpoint')
  .then(response => {
    console.log(response.data);
  })
  .catch(error => {
    console.error('Error:', error);
  });
}
```

- 2) In your Django settings.py, add CORS settings to allow requests from the React frontend:

```
pythonCopyINSTALLED_APPS = [
    ...
    'corsheaders',
]

MIDDLEWARE = [
    'corsheaders.middleware.CorsMiddleware',
    ...
]

CORS_ALLOW_ALL_ORIGINS = True # For development only
Note: Install django-cors-headers with pip install django-cors-headers
```

Running the Project

- 1) Start the Django server:

```
cd backend
python manage.py runserver
```

- 2) In a new terminal, start the React development server:

```
cd frontend
npm start
```

Your Django backend will be running on <http://localhost:8000> and your React frontend on <http://localhost:3000>.

Output

1) Virtual Private Cloud Screenshot

The screenshot shows the AWS VPC dashboard with the following details:

Top Bar: Services, Search, [Option+S], Actions, Create VPC, N. Virginia, darshanum27.

Left Sidebar: VPC dashboard, EC2 Global View, Filter by VPC, Virtual private cloud (Your VPCs, Subnets, Route tables, Internet gateways, Egress-only internet gateways, Carrier gateways, DHCP option sets, Elastic IPs, Managed prefix lists, Endpoints, Endpoint services, NAT gateways, Peering connections), Security (Network ACLs, Security groups), DNS firewall.

Main Content: Your VPCs (1/1) Info table with one row:

Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR	DHCP opt
-	vpc-0d1810edc62ff95c3	Available	172.31.0.0/16	-	dopt-0b25aa64e81983f

Details View: For VPC vpc-0d1810edc62ff95c3, showing the following details:

Details			
VPC ID vpc-0d1810edc62ff95c3	State Available	DNS hostnames Enabled	DNS resolution Enabled
Tenancy Default	DHCP option set dopt-0b25aa64e81983f	Main route table rtb-018fb04160aec17b	Main network ACL acl-0652766033d93c7b0
Default VPC Yes	IPv4 CIDR 172.31.0.0/16	IPv6 pool -	IPv6 CIDR (Network border group) -
Network Address Usage metrics Disabled	Route 53 Resolver DNS Firewall rule groups -	Owner ID 992382449250	

2) Subnet Screenshot:

The screenshot shows the AWS Subnet dashboard with the following details:

Top Bar: Services, Search, [Option+S], Actions, Create subnet, N. Virginia, darshanum27.

Left Sidebar: VPC dashboard, EC2 Global View, Filter by VPC, Virtual private cloud (Your VPCs, Subnets, Route tables, Internet gateways, Egress-only internet gateways, Carrier gateways, DHCP option sets, Elastic IPs, Managed prefix lists, Endpoints, Endpoint services, NAT gateways, Peering connections), Security (Network ACLs, Security groups), DNS firewall.

Main Content: Subnets (1/1) Info table with one row:

Name	Subnet ID	State	VPC	IPv4 CIDR
-	subnet-0d94e4180deb1bb29	Available	vpc-0d1810edc62ff95c3	172.31.1.0/24

Details View: For Subnet subnet-0d94e4180deb1bb29, showing the following details:

Details			
Subnet ID subnet-0d94e4180deb1bb29	Subnet ARN arn:aws:ec2:us-east-1:992382449250:subnet/subnet-0d94e4180deb1bb29	State Available	IPv4 CIDR 172.31.1.0/24
Available IPv4 addresses 250	Availability Zone us-east-1e	Availability Zone ID use1-az3	Auto-assign customer-owned IPv4 address No
Network border group us-east-1	IPv6 CIDR -	Route table rtb-018fb04160aec17b	Auto-assign IPv6 address No
Default subnet No	VPC vpc-0d1810edc62ff95c3	IPv4 CIDR reservations -	IPv6 CIDR reservations -
Customer-owned IPv4 pool -	Auto-assign public IPv4 address No	Resource name DNS A record Disabled	Resource name DNS AAAA record
IPv6-only No	Outpost ID -		

3) S3 Bucket creation Screenshot:

The screenshot shows the AWS S3 console with the 'General purpose buckets' tab selected. There are two buckets listed:

Name	AWS Region	IAM Access Analyzer	Creation date
elasticbeanstalk-us-east-1-992382449250	US East (N. Virginia) us-east-1	View analyzer for us-east-1	May 28, 2024, 11:16:15 (UTC-05:00)
react-app-bucket-check	US East (N. Virginia) us-east-1	View analyzer for us-east-1	July 19, 2024, 11:08:51 (UTC-05:00)

4) S3 Bucket React.js build files upload:

The screenshot shows the AWS S3 console with the 'react-app-bucket-check' bucket selected. The 'Objects' tab is active, displaying eight objects:

Name	Type	Last modified	Size	Storage class
asset-manifest.json	json	July 19, 2024, 14:31:38 (UTC-05:00)	517.0 B	Standard
favicon.ico	ico	July 19, 2024, 14:31:38 (UTC-05:00)	3.8 KB	Standard
index.html	html	July 19, 2024, 14:31:38 (UTC-05:00)	644.0 B	Standard
logo192.png	png	July 19, 2024, 14:31:39 (UTC-05:00)	5.2 KB	Standard
logo512.png	png	July 19, 2024, 14:31:39 (UTC-05:00)	9.4 KB	Standard
manifest.json	json	July 19, 2024, 14:31:39 (UTC-05:00)	492.0 B	Standard
robots.txt	txt	July 19, 2024, 14:31:40 (UTC-05:00)	67.0 B	Standard
static/	Folder	-	-	-

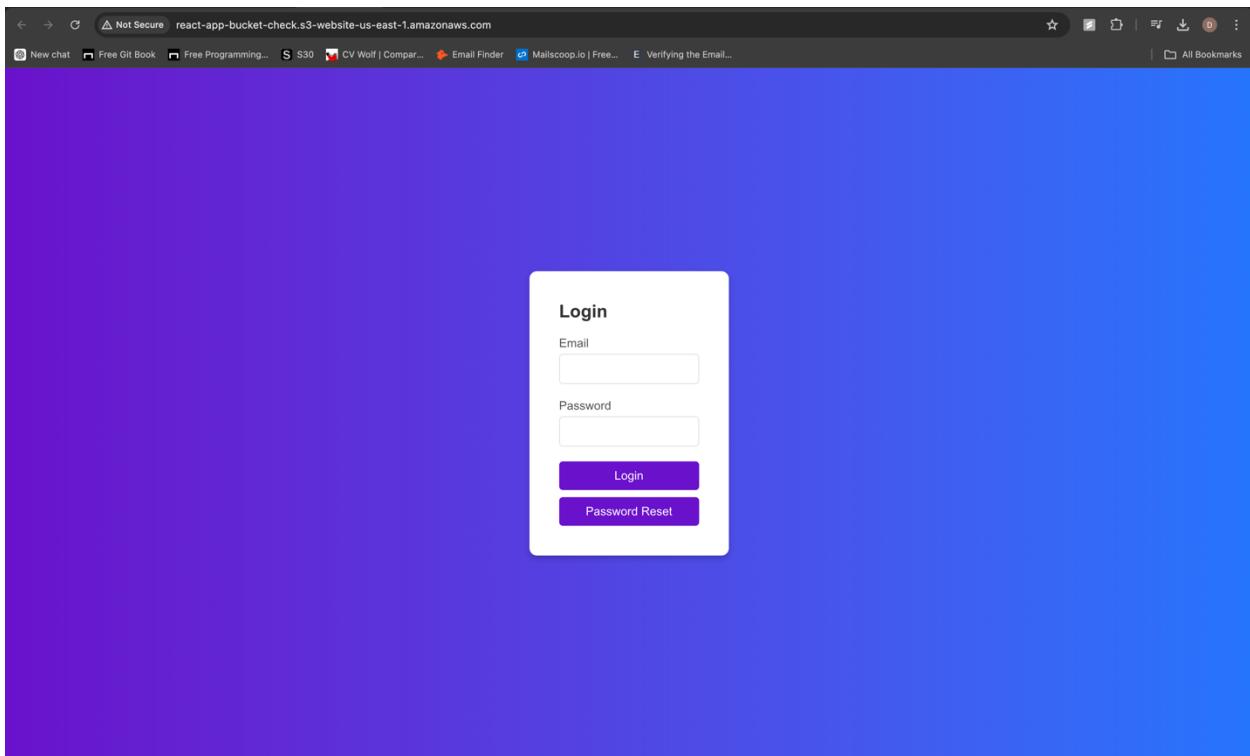
5) s3 Bucket static Link:

The screenshot shows the AWS S3 console for a bucket named 'react-app-bucket-check'. The left sidebar includes sections for Buckets, Storage Lens, and Feature spotlight. The main content area displays the 'Object Lock' settings, which are currently disabled. It also shows the 'Requester pays' setting, which is also disabled. Under 'Static website hosting', the hosting type is set to 'Bucket hosting' and the endpoint is listed as <http://react-app-bucket-check.s3-website-us-east-1.amazonaws.com>.

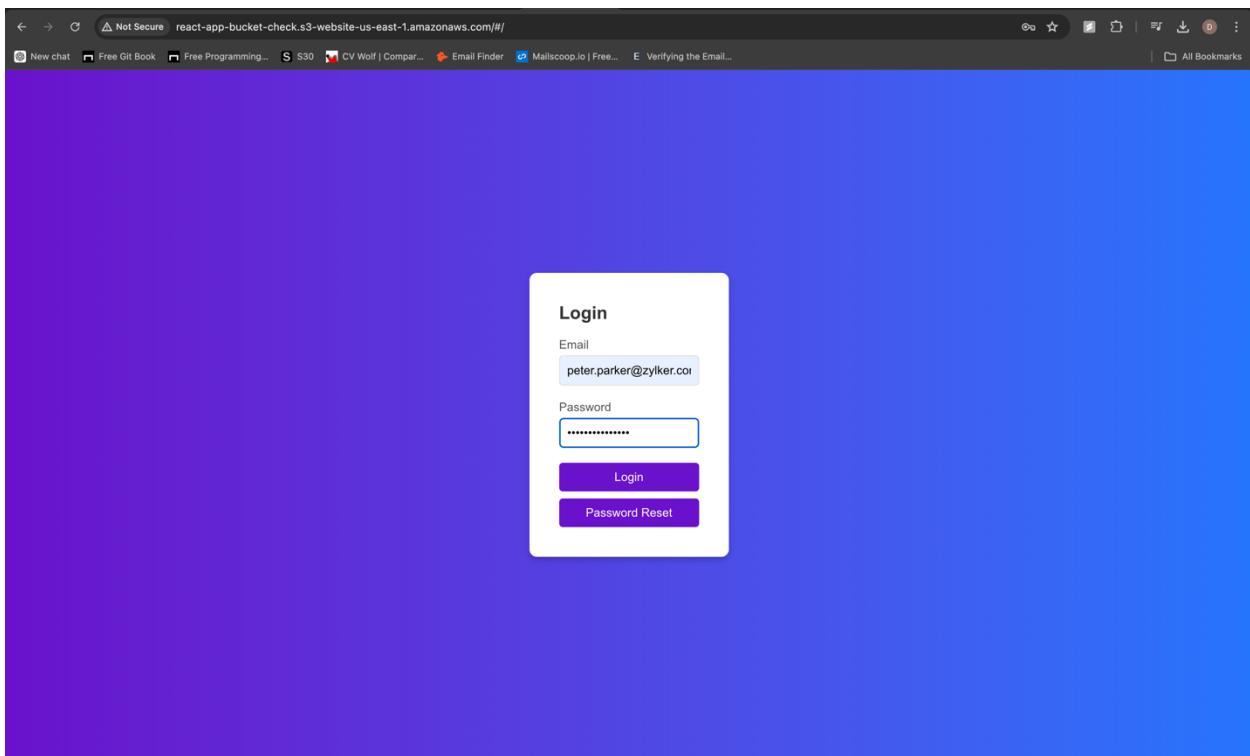
6) Ec2 instance running:

The screenshot shows the AWS EC2 Instances page with one instance listed: 'Server1' (i-09829cebd4a0fcfe7). The instance is running, t2.micro, and has passed 2/2 checks. It is located in the us-east-1e Availability Zone. The Details tab is selected, showing the instance summary, which includes its ID, IP addresses, instance type, and VPC information.

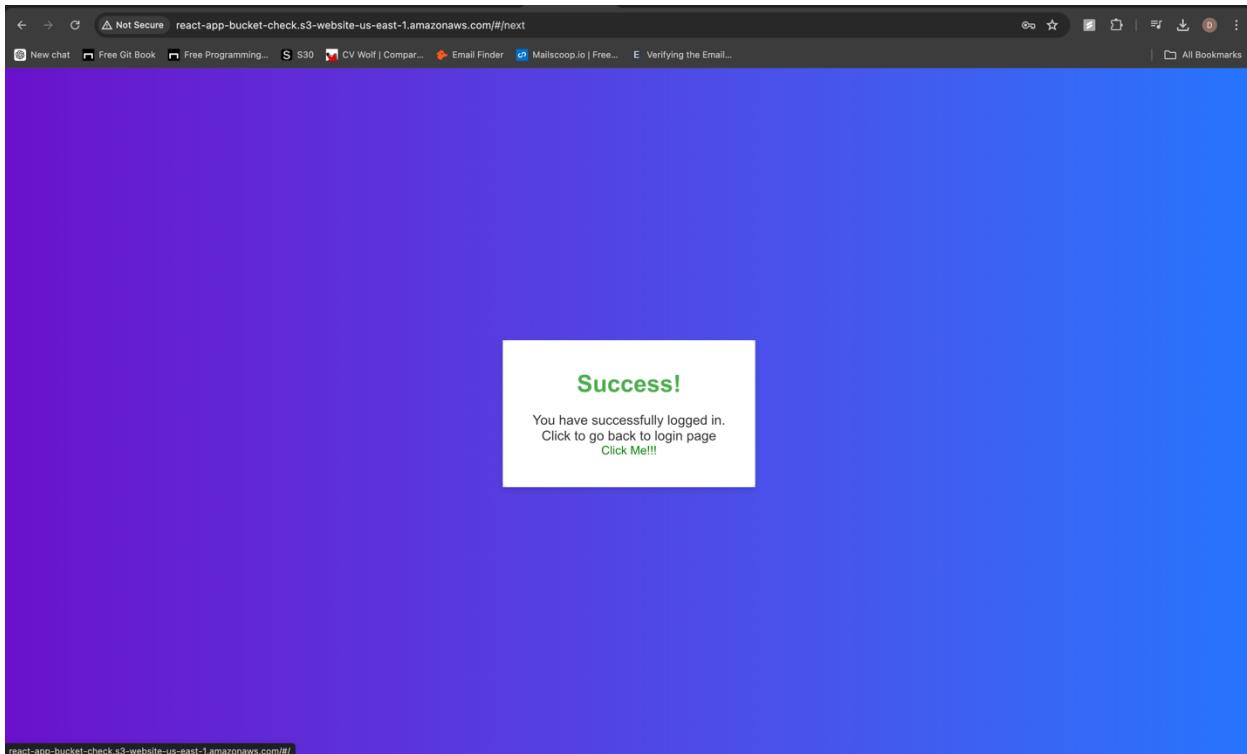
7) Landing Page:



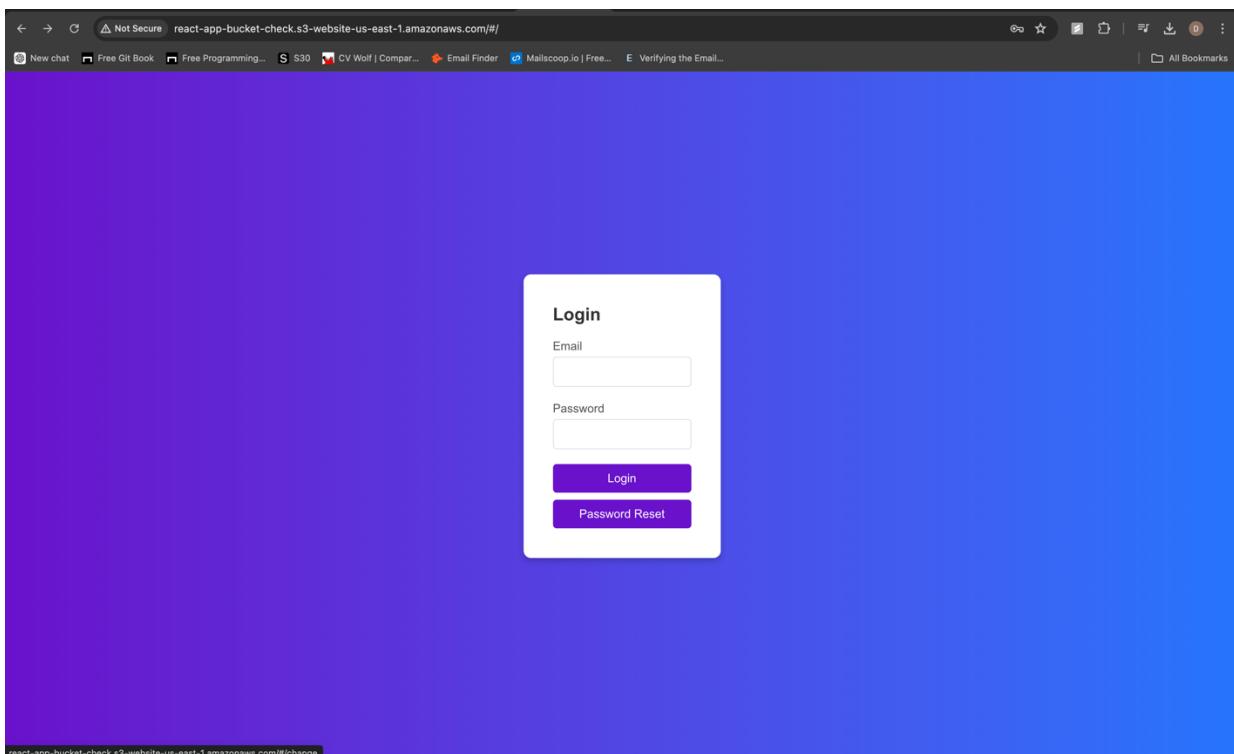
8) Default Value Enter:



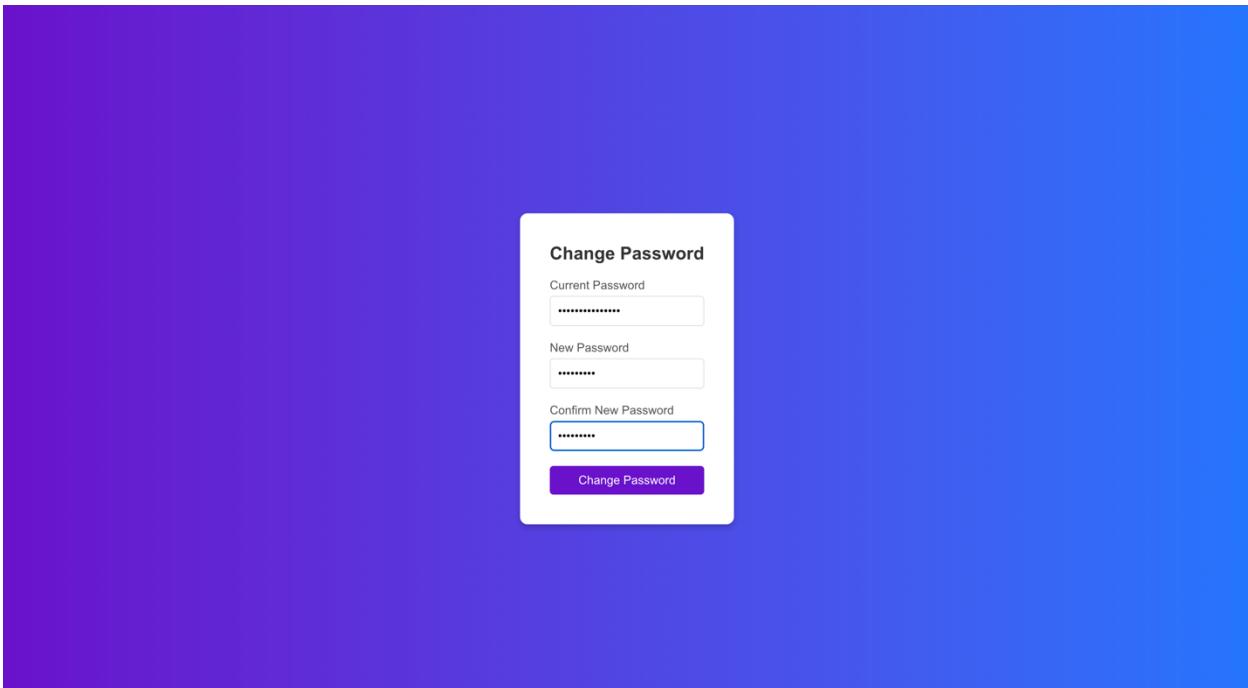
9) Login Successful:



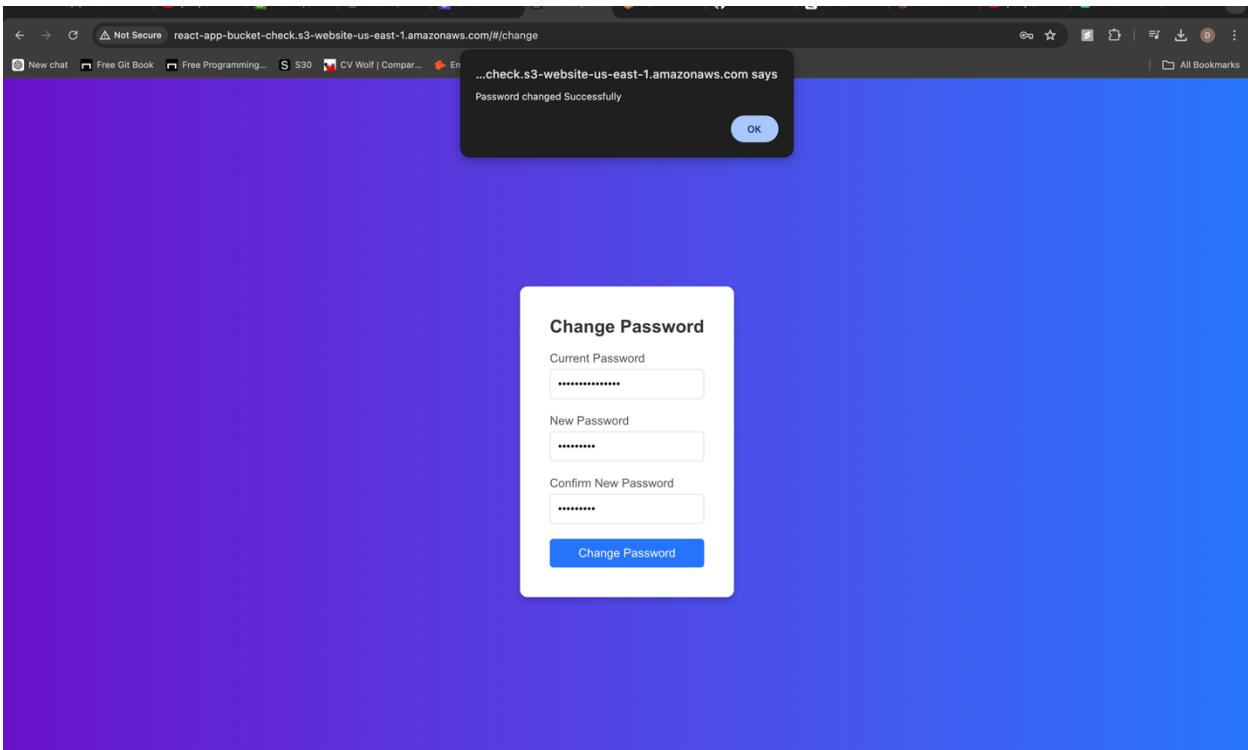
10) Back to Landing page from Login successful page:



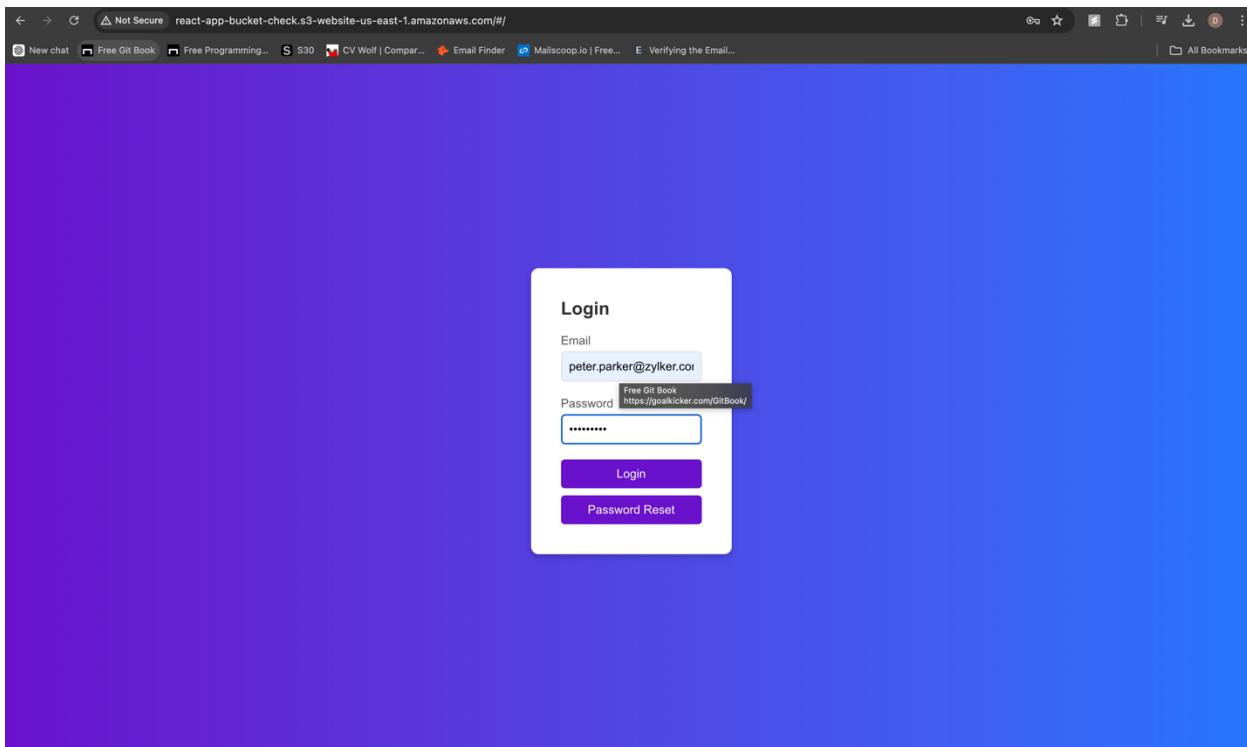
11) Change Default Password:



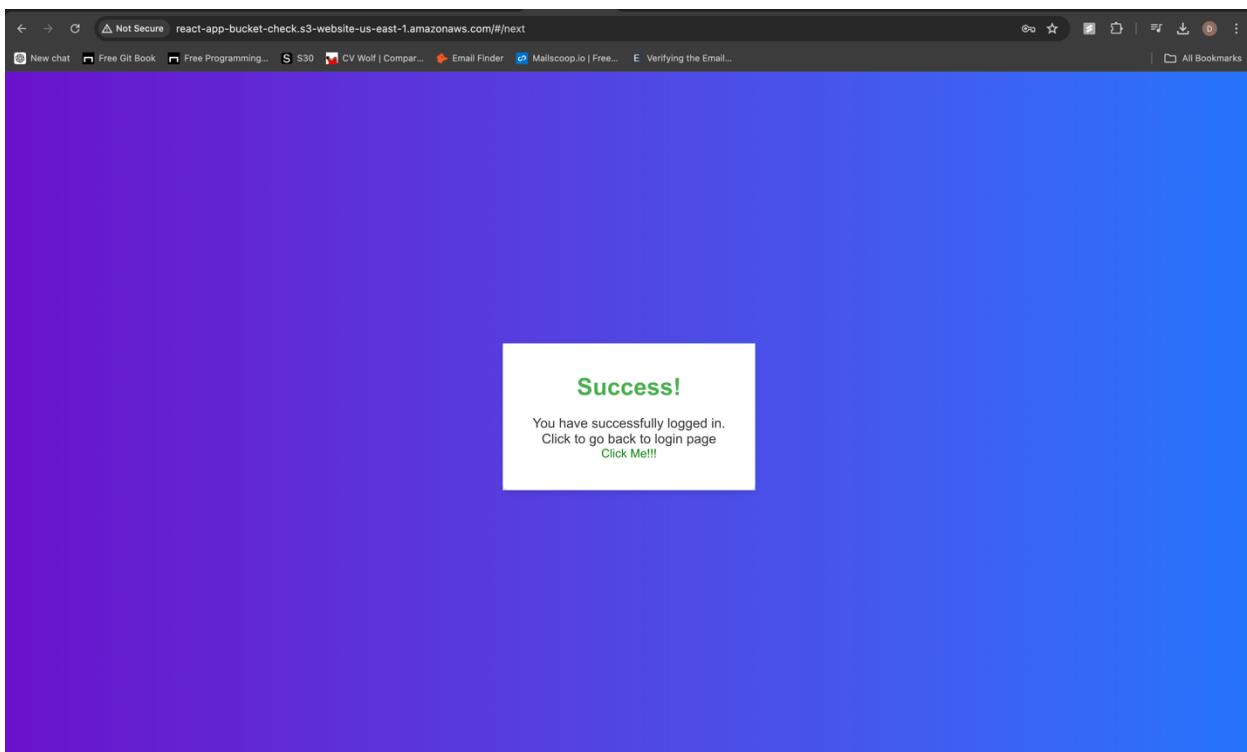
12) Default password changed to new password Successfully



13) Login with new Password:



14) Logged in Successfully:



15) Wrong Password Message:

