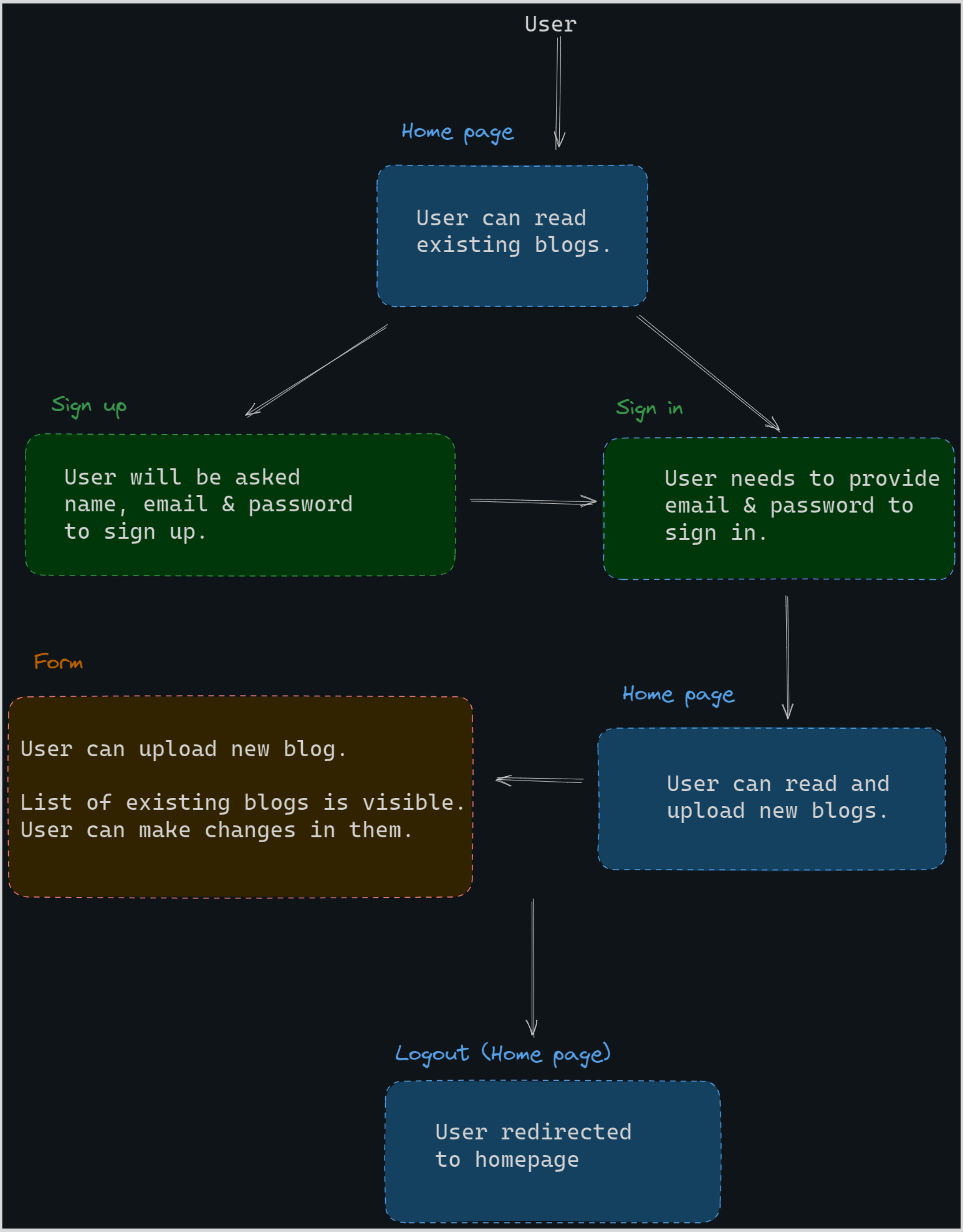
Index

Website (frontend) Architecture & User interaction flow



User
Home page
User can read existing blogs.
Sign up
User will be asked name, email & password to sign up.
Sign in
User needs to provide email & password to sign in.
Home page
User can read and upload new blogs.
Form
User can upload new blog.
List of existing blogs is visible.
User can make changes in them.
Logout (Home page)
User redirected to homepage

Modules Brief Description

Module / Page	Description
1. Home page (Blog)	- Default page. - User can see blogs’ images, titles & descriptions of blogs. - Buttons (if not logged in)- Blogs, Sign up, Sign in. - Buttons (if logged in)- Forms, Blogs, Log out.
2. Sign up	- User will be asked name, email & password to sign up. - Successful sign up redirects user to Sign in page..
3. Sign in	- User needs to provide a valid email & password to login. - Successful login redirects user to home page.
4. Forms	- Available for logged users only. - User can add new blog or make changes in existing blogs.

Instructions to run app in local

Prerequisite

- Node.js in host machine.
- VS code or any supporting editor.

- Frontend Repo: https://github.com/Anand-1432/Techdome-frontend
- Backend Repo: https://github.com/Anand-1432/Techdome-backend

Run app

- Open frontend directory in VS code.
- Open Frontend directory in VS code terminal
- Run npm install.
- Run npm start

Errors

- Incase of error with npm, remove existing ‘node_modules’ directory then retry.
- Try ‘npm audit fix’.

Access

- App should open in a browser.
- Or access localhost:3000 in a browser.

Create Docker Image

1. Dockerfile & .gitignore file

```
# Installs node
FROM node:latest


# Setting /react-app as working directory.
WORKDIR /react-app


# Copy package.json & package-lock.json to working dir.
COPY package*.json ./


# Installs dependencies - node_modules.
RUN npm install


# Copy all files to workdir.
COPY . .


# Exposing Port 3000 for app.
EXPOSE 3000


# Starts app.
CMD npm start
```

.gitignore

```
node_modules


# Dockerfile -            # Dockerfile will be saved in the work-dir for future reference.


# .dockerignore
```

2. Create docker image

```
$ docker build -t <username/image-name:version> .
```

3. Verify image creation

```
$ docker images
```

3. Upload images to docker hub

```
$ docker push <username/image-name:v1>
```

Note - Above command must be run inside a frontend dir where Dockerfile is kept.

K8s Manifests

1. Frontend - Deployment + Service YAMLs

```yaml
#Frontend Deployment

apiVersion: apps/v1
kind: Deployment

metadata:
  name: frontend-dep

spec:
  selector:
    matchLabels:
      app: react
      tier: frontend

  replicas: 1

  template:
    metadata:
      labels:
        app: react
        tier: frontend

    spec:
      containers:
        - name: con
          image: dev7495/react-frontend-ni:v1
          ports:
          - containerPort: 3000
---
```

```yaml
#Frontend Service

apiVersion: v1
kind: Service

metadata:
  name: frontend-svc

spec:
  selector:
    app: react
    tier: frontend

  ports:
  - protocol: "TCP"
    port: 3000
    targetPort: 3000
    nodePort:

  type: NodePort
...
```

2. Backend - Deployment + Service YAMLs

```
# Backend Deployment

apiVersion: apps/v1
kind: Deployment

metadata:
  name: backend-dep

spec:
  selector:
    matchLabels:
      app: react
      tier: backend

  replicas: 1

  template:
    metadata:
      labels:
        app: react
        tier: backend

    spec:
      containers:
        - name: back-con
          image: dev7495/react-backend2:v1
          ports:
          - containerPort: 3000
---
```

```
# Backend Service

apiVersion: v1
kind: Service

metadata:
  name: backend-svc

spec:
  selector:
    app: react
    tier: backend

  ports:
  - protocol: TCP
    port: 3000
    targetPort: 3000
...
```

3.1 MongoDB Deployment YAML

```
apiVersion: apps/v1
kind: Deployment

metadata:
  name: mongo-deployment
  labels:
    app: mongo

spec:
  replicas: 1
  selector:
    matchLabels:
      app: mongo
#     tier: backend

  template:
    metadata:
      labels:
        app: mongo
#       tier: backend

    spec:
      containers:
      - name: mongo
        image: mongo
        ports:
        - containerPort: 27017

        env:
        - name: MONGO_INITDB_ROOT_USERNAME
          valueFrom:
```

```
          secretKeyRef:
            name: mongo-secret
            key: mongo-user

      - name: MONGO_INITDB_ROOT_PASSWORD
        valueFrom:
          secretKeyRef:
            name: mongo-secret
            key: mongo-password
---
```

3.1 MongoDB Service YAML

```
# MongoDB Secret

apiVersion: v1
kind: Service

metadata:
  name: mongo-service

spec:
  selector:
    app: mongo

  ports:
    - protocol: TCP
      port: 27017
      targetPort: 27017
      nodePort:                # Added nodePort to verify
  type: NodePort               # if mongodb is working.
```

3.2 MongoDB Secret YAML

```
# MongoDB Secret


apiVersion: v1
kind: Secret


metadata:
  name: mongo-secret


type: Opaque


data:
  mongo-user:  bW9uZ291c2Vy
  mongo-password: bW9uZ29wYXNzd29yZA==
---
```

3.2 MongoDB Configmap YAML

```
apiVersion: v1
kind: ConfigMap


metadata:
  name: mongo-config


data:
  mongo-url: mongo-service
```