Full-Stack SMS Management System

```
/sms-management
|-- /frontend               # React frontend
|   |-- /src
|   |-- /public
|   |-- package.json
|   |-- README.md           # Frontend documentation
|-- /backend                # FastAPI backend
|   |-- /app
|       |-- main.py        # Main FastAPI application
|       |-- models.py      # Database models
|       |-- routes.py      # API routes
|       |-- utils.py       # Utility functions
|   |-- requirements.txt    # Python dependencies
|   |-- README.md           # Backend documentation
|-- docker-compose.yml       # For Docker deployment
|-- README.md               # Main project documentation
```

1. Backend Development (FastAPI)
Requirements (requirements.txt)

```
fastapi
uvicorn
pydantic
pymongo
sqlalchemy
mysql-connector-python
python-telegram-bot
python-jose
```

Main Application (main.py)

```
from fastapi import FastAPI, Depends, HTTPException
from fastapi.security import OAuth2PasswordBearer, OAuth2PasswordRequestForm
from sqlalchemy.orm import Session
from . import models, database, utils

app = FastAPI()

oauth2_scheme = OAuth2PasswordBearer(tokenUrl="token")

@app.post("/login")
async def login(form_data: OAuth2PasswordRequestForm = Depends()):
    user = utils.verify_user(form_data.username, form_data.password)
    if not user:
        raise HTTPException(status_code=400, detail="Incorrect username or
password")
    return {"access_token": user, "token_type": "bearer"}

@app.post("/sessions/start")
async def start_session(country: str, operator: str):
    session_id = f"program1_{country}_{operator}"
    # Logic to start the session
    return {"message": f"Session {session_id} started"}
```

Models (models.py)
python

```
from sqlalchemy import Column, Integer, String, Float
from .database import Base

class SMSMetrics(Base):
    __tablename__ = "sms_metrics"
```

```
    id = Column(Integer, primary_key=True, index=True)
    country = Column(String)
    operator = Column(String)
    sms_sent = Column(Integer)
    success_rate = Column(Float)
    errors = Column(Integer)

2. Frontend Development (React)

Setup Instructions

1.Create a new React app:

npx create-react-app frontend
cd frontend
npm install axios react-router-dom


Dashboard Component (Dashboard.js)

import React, { useEffect, useState } from 'react';
import axios from 'axios';

const Dashboard = () => {
    const [metrics, setMetrics] = useState({});

    useEffect(() => {
        const fetchMetrics = async () => {
            const response = await axios.get('/metrics/sms_sent');
            setMetrics(response.data);
        };
        fetchMetrics();
    }, []);

    return (
        <div>
            <h1>Dashboard</h1>
            <p>SMS Sent: {metrics.sms_sent}</p>
        </div>
    );
};

export default Dashboard;

3. Database Design
MongoDB Schema (for configurations)
Collection: countryOperators
Fields: country, operator, high_priority, session_details

MySQL Schema (for metrics)

CREATE TABLE sms_metrics (
    id INT AUTO_INCREMENT PRIMARY KEY,
    country VARCHAR(100),
    operator VARCHAR(100),
    sms_sent INT,
    success_rate DECIMAL(5,2),
    errors INT,
    timestamp DATETIME DEFAULT CURRENT_TIMESTAMP
);

4. Monitoring and Alerts
Prometheus Setup
Use a Prometheus client to expose metrics from your FastAPI app.
```

Configure Prometheus to scrape your metrics endpoint.

5.Deployment
Docker Compose (docker-compose.yml)

```
version: '3'
services:
  backend:
    build: ./backend
    ports:
      - "8000:8000"
    depends_on:
      - mongo
      - mysql
  frontend:
    build: ./frontend
    ports:
      - "3000:3000"
  mongo:
    image: mongo
  mysql:
    image: mysql
    environment:
      MYSQL_ROOT_PASSWORD: root
      MYSQL_DATABASE: sms_metrics
```

6.Documentation

Main README.md
markdown
```
# SMS Management System

## Overview
A full-stack SMS management system for controlling SMS sending programs and
monitoring metrics.

## Features
- Manage SMS program sessions.
- Real-time metrics dashboard.
- Country-operator management with CRUD operations.
- Alerts via Telegram for critical events.
- Secure JWT authentication.

## Getting Started
### Backend
1. Navigate to `backend`.
2. Install dependencies:
   ```bash
   pip install -r requirements.txt
```

Sure! Hereâ ™s a comprehensive outline for your full-stack web application assignment based on the requirements you provided. This includes backend and frontend code snippets, database schema, deployment instructions, and documentation.

Full-Stack SMS Management System
Project Structure
bash
Copy code
```
/sms-management
|-- /frontend               # React frontend
|   |-- /src
|   |-- /public
```
```

```
|   |-- package.json
|   |-- README.md           # Frontend documentation
|-- /backend                # FastAPI backend
|   |-- /app
|       |-- main.py        # Main FastAPI application
|       |-- models.py      # Database models
|       |-- routes.py      # API routes
|       |-- utils.py       # Utility functions
|   |-- requirements.txt    # Python dependencies
|   |-- README.md           # Backend documentation
|-- docker-compose.yml       # For Docker deployment
|-- README.md               # Main project documentation
```

1. Backend Development (FastAPI)

Requirements (requirements.txt)

plaintext

Copy code

```
fastapi
uvicorn
pydantic
pymongo
sqlalchemy
mysql-connector-python
python-telegram-bot
python-jose
```

Main Application (main.py)

python

Copy code

```
from fastapi import FastAPI, Depends, HTTPException
from fastapi.security import OAuth2PasswordBearer, OAuth2PasswordRequestForm
from sqlalchemy.orm import Session
from . import models, database, utils

app = FastAPI()

oauth2_scheme = OAuth2PasswordBearer(tokenUrl="token")

@app.post("/login")
async def login(form_data: OAuth2PasswordRequestForm = Depends()):
    user = utils.verify_user(form_data.username, form_data.password)
    if not user:
        raise HTTPException(status_code=400, detail="Incorrect username or
password")
    return {"access_token": user, "token_type": "bearer"}

@app.post("/sessions/start")
async def start_session(country: str, operator: str):
    session_id = f"program1_{country}_{operator}"
    # Logic to start the session
    return {"message": f"Session {session_id} started"}
```

Models (models.py)

python

Copy code

```
from sqlalchemy import Column, Integer, String, Float
from .database import Base

class SMSMetrics(Base):
    __tablename__ = "sms_metrics"
    id = Column(Integer, primary_key=True, index=True)
    country = Column(String)
    operator = Column(String)
    sms_sent = Column(Integer)
    success_rate = Column(Float)
    errors = Column(Integer)
```

2. Frontend Development (React)

Setup Instructions
Create a new React app:
bash
Copy code
```
npx create-react-app frontend
cd frontend
npm install axios react-router-dom
```
Dashboard Component (Dashboard.js)
javascript
Copy code
```
import React, { useEffect, useState } from 'react';
import axios from 'axios';

const Dashboard = () => {
    const [metrics, setMetrics] = useState({});

    useEffect(() => {
        const fetchMetrics = async () => {
            const response = await axios.get('/metrics/sms_sent');
            setMetrics(response.data);
        };
        fetchMetrics();
    }, []);

    return (
        <div>
            <h1>Dashboard</h1>
            <p>SMS Sent: {metrics.sms_sent}</p>
        </div>
    );
};

export default Dashboard;
```
3. Database Design
MongoDB Schema (for configurations)
Collection: countryOperators
Fields: country, operator, high_priority, session_details
MySQL Schema (for metrics)
sql
Copy code
```
CREATE TABLE sms_metrics (
    id INT AUTO_INCREMENT PRIMARY KEY,
    country VARCHAR(100),
    operator VARCHAR(100),
    sms_sent INT,
    success_rate DECIMAL(5,2),
    errors INT,
    timestamp DATETIME DEFAULT CURRENT_TIMESTAMP
);
```
4. Monitoring and Alerts
Prometheus Setup
Use a Prometheus client to expose metrics from your FastAPI app.
Configure Prometheus to scrape your metrics endpoint.
5. Deployment
Docker Compose (docker-compose.yml)
yaml
Copy code
```
version: '3'
services:
  backend:
    build: ./backend
    ports:
      - "8000:8000"
    depends_on:
```

```
      - mongo
      - mysql
  frontend:
    build: ./frontend
    ports:
      - "3000:3000"
  mongo:
    image: mongo
  mysql:
    image: mysql
    environment:
      MYSQL_ROOT_PASSWORD: root
      MYSQL_DATABASE: sms_metrics
```

6. Documentation

Main README.md

markdown

Copy code

```
# SMS Management System

## Overview
A full-stack SMS management system for controlling SMS sending programs and
monitoring metrics.

## Features
- Manage SMS program sessions.
- Real-time metrics dashboard.
- Country-operator management with CRUD operations.
- Alerts via Telegram for critical events.
- Secure JWT authentication.

## Getting Started
### Backend
1. Navigate to `backend`.
2. Install dependencies:
   ```bash
   pip install -r requirements.txt


3.Run the server:
bash
uvicorn app.main:app --reload

Frontend

1.Navigate to frontend.

2.Install dependencies:
bash
npm install

3.Start the React app:
bash
npm start
```
```