#!/bin/bash

set -e

# CHANGELOG: This script creates a production-ready TowService repo under your GitHub org/user

# Prereqs: gh CLI installed and authenticated, Git installed, Node.js installed, Docker optional

GITHUB\_ORG="lgtowing"            # your GitHub org/user

REPO\_NAME="TowService-starter-prod"  # production-ready repo name (you can change)

VISIBILITY="public"               # public or private

INCLUDE\_DOCKER="yes"              # yes/no

BRAND\_PRIMARY="#1F4E79"            # default primary color

BRAND\_ACCENT="#FF6B00"             # default accent color

LOGO\_PATH="shared/assets/logo.png" # logo path inside repo

echo "Creating GitHub repo ${GITHUB\_ORG}/${REPO\_NAME} (visibility: ${VISIBILITY})..."

gh repo create "${GITHUB\_ORG}/${REPO\_NAME}" --${VISIBILITY} --source=. --remote=origin

echo "Cloning newly created repo locally..."

git clone "https://github.com/${GITHUB\_ORG}/${REPO\_NAME}.git" /tmp/${REPO\_NAME}-prod

cd /tmp/${REPO\_NAME}-prod

echo "Initializing scaffold..."

rm -rf \*

# Backend scaffold (NestJS-like with Prisma)

mkdir -p backend/src/{auth,tow,driver,client,common}

cat > backend/package.json <<'JSON'

{

  "name": "towservice-backend",

  "version": "1.0.0",

  "private": true,

  "scripts": {

    "start": "node dist/main.js",

    "build": "tsc",

    "start:dev": "ts-node-dev --respawn --transpile-only src/main.ts"

  },

  "dependencies": {

    "@prisma/client": "^4.0.0",

    "bcrypt": "^5.0.1",

    "cors": "^2.8.5",

    "express": "^4.18.2",

    "jsonwebtoken": "^9.0.0",

    "socket.io": "^4.5.0",

    "typeorm": "^0.2.41",

    "pg": "^8.9.0",

    "reflect-metadata": "^0.1.13",

    "rxjs": "^7.5.0",

    "express-jwt": "^3.1.0",

    "passport": "^0.7.0",

    "passport-jwt": "^4.0.0",

    "typeorm-model-generator": "^0.4.0"

  },

  "devDependencies": {

    "ts-node": "^10.9.1",

    "typescript": "^4.9.0",

    "@types/express": "^4.17.20",

    "@types/node": "^18.11.9",

    "ts-node-dev": "^2.0.0"

  }

}

JSON

mkdir -p backend/src/main

cat > backend/src/main.ts <<'TS'

import 'reflect-metadata';

import express from 'express';

import http from 'http';

import { Server } from 'socket.io';

const app = express();

app.use(express.json());

const server = http.createServer(app);

const io = new Server(server, { cors: { origin: '\*' } });

io.on('connection', (socket) => {

  socket.on('subscribe', (towId: string) => {

    socket.join(towId);

  });

  // You can emit('update', payload) to room towId

});

app.post('/tows', (req, res) => {

  const id = `tow\_${Date.now()}`;

  const tow = { id, status: 'Dispatched', ...req.body };

  res.json(tow);

});

server.listen(3000, () => console.log('Backend listening on port 3000'));

TS

mkdir -p backend/src/tow/dtos

cat > backend/src/tow/dtos/create-tow.dto.ts <<'TS'

export class CreateTowDto {

  clientId: string;

  pickupLocation: { lat: number; lon: number; address?: string };

  dropoffLocation: { lat: number; lon: number; address?: string };

  vehicleDetails?: string;

  notes?: string;

}

TS

# Prisma-like schema placeholder (optional, can migrate to real Prisma later)

cat > backend/prisma.schema <<'SCHEMA'

datasource db {

  provider = "postgresql"

  url      = env("DATABASE\_URL")

}

generator client {

  provider = "prisma-client-js"

}

model Tow {

  id        String @id

  clientId  String

  status    String

  // ... more fields per MVP

  createdAt DateTime @default(now())

}

SCHEMA

# Mobile scaffolds (RN)

mkdir -p mobile/client/src/screens

cat > mobile/client/App.tsx <<'TS'

import React from 'react';

import { View, Text, TextInput, Button } from 'react-native';

export default function ClientApp() {

  return (

    <View style={{ padding: 20 }}>

      <Text>Tow Service - Client</Text>

      <TextInput placeholder="Pickup" style={{ borderWidth: 1, marginVertical: 6 }} />

      <TextInput placeholder="Drop-off" style={{ borderWidth: 1, marginVertical: 6 }} />

      <TextInput placeholder="Notes" style={{ borderWidth: 1, marginVertical: 6 }} />

      <Button title="Request Tow" onPress={() => {}} />

    </View>

  );

}

TS

mkdir -p mobile/driver/src/screens

cat > mobile/driver/App.tsx <<'TS'

import React from 'react';

import { View, Text } from 'react-native';

export default function DriverApp() {

  return (

    <View style={{ padding: 20 }}>

      <Text>Driver App - Assigned Tows</Text>

    </View>

  );

}

TS

# Shared branding

mkdir -p shared/assets

# Placeholder for logo; you’ll replace with actual binary in real repo

cat > shared/assets/logo.png <<'PNG'

<binary-image-placeholder>

PNG

# OpenAPI placeholder

cat > openapi.yaml <<'YAML'

openapi: 3.0.0

info:

  title: TowService API

  version: 1.0.0

paths:

  /tows:

    post:

      summary: Create tow request

      responses:

        '200':

          description: OK

YAML

# README

cat > README.md <<'MD'

TowService Production-Ready Scaffold

- Backend: NestJS-like with Prisma + PostgreSQL (scaffolded)

- Frontend: React Native client + driver (scaffold)

- Real-time updates via Socket.IO

- Docker support for local dev (PostgreSQL + Redis)

- Uses shared/assets/logo.png as branding asset

- How to run: follow steps in this repo's setup guide

MD

# Commit and push

git init

git add .

git commit -m "Prod TowService scaffold: NestJS-like backend, Prisma schema, RN client/driver, branding placeholders"

git branch -M main

git remote add origin "https://github.com/${GITHUB\_ORG}/${REPO\_NAME}.git"

git push -u origin main

# Final notes

echo "Repo created at https://github.com/${GITHUB\_ORG}/${REPO\_NAME}"

echo "Next steps: clone, run npm install in backend and mobile, flesh out real NestJS modules, migrate to Prisma, wire up RN screens, and swap in your real logo and brand colors."

echo "If you want, I can also add a GitHub Actions workflow, Dockerfiles, and a docker-compose.yml for full local dev with containers."