TMO Task: Step-by-Step Guide

Initial Setup & Backend Development

1. Create .NET Core Backend

```
dotnet new webapi -n Backend
```

- Used .NET Core 8.0
- Added CsvHelper package for data processing
- Created Models folder for data structures

2. Data Structure Setup

- Created `orders.csv` with columns:

```
Seller,Product,Price,OrderDate,Branch
```

- Located in `Backend/Data/` directory
- Sample data includes multiple branches and sellers

3. Backend Components

- **Models**:

```
Order.cs
SellerSummary.cs
```

- **Services**:

```
OrderService.cs - Handles CSV reading and data processing
IOrderService.cs - Service interface
```

- **Controllers**:

```
api/branches - Gets unique branch list
api/sellers?branch={branchName} - Gets seller data by branch
```

4. CORS Configuration

```
builder.Services.AddCors(options => {
    options.AddPolicy("CorsPolicy", policy => {
        policy.WithOrigins(
            "http://localhost:5173",
            "http://tmotask-frontend.s3-website-us-west-2.amazonaws.com"
        )
        .AllowAnyHeader()
        .AllowAnyMethod();
    });
});
```

Frontend Development

1. Create React App

```
npm create vite@latest Frontend -- --template react-ts
```

- Used Vite for fast development
- TypeScript for type safety
- React for UI components

2. Install Dependencies

```
npm install
npm install lucide-react # for icons
```

3. Component Structure

- **Main Components**:

```
src/App.tsx - Main dashboard
src/api.ts - API integration
```

- **Styling**:

```
App.module.scss - CSS modules for styling
```

4. API Integration

```
const API_BASE_URL = process.env.NODE_ENV === 'development'
    ? 'http://localhost:5090'
    : 'http://TmoTask-API-Prod.eba-xmbtpppu.us-west-
2.elasticbeanstalk.com';
```

AWS Deployment

1. Backend Deployment (Elastic Beanstalk)

```
eb init # Initialize EB application
eb create # Create environment
```

- Created environment: TmoTask-API-Prod
- Platform: .NET Core on Linux
- Region: us-west-2
- Self-contained deployment

2. Frontend Deployment (S3)

```
aws s3 mb s3://tmotask-frontend # Create bucket
aws s3 website s3://tmotask-frontend --index-document index.html
```

- Bucket: tmotask-frontend
- Configured for static website hosting
- Public access enabled

3. Deploy Frontend

```
npm run build
aws s3 sync dist/ s3://tmotask-frontend --delete
```

Key Features & Libraries Used

Backend Libraries

- **CsvHelper**: CSV file processing
- **.NET Core**: Web API framework
- **System.Linq**: Data processing

Frontend Libraries

- **React**: UI framework
- **TypeScript**: Type safety
- **Vite**: Build tool
- **Lucide React**: Icons
- **SCSS Modules**: Styling

AWS Services

- **Elastic Beanstalk**: Backend hosting
  - URL: http://TmoTask-API-Prod.eba-xmbtpppu.us-west-2.elasticbeanstalk.com
- **S3**: Frontend hosting
  - URL: http://tmotask-frontend.s3-website-us-west-2.amazonaws.com

Data Flow

1. CSV data loaded at backend startup
2. Frontend makes API calls on:
   - Initial load
   - Branch selection change
3. Backend processes data:
   - Filters by branch
   - Aggregates by month
   - Calculates totals
4. Frontend displays:
   - Branch selector
   - Monthly performance table
   - Loading/error states

Maintenance & Updates

1. To add new data:
   - Update `orders.csv`
   - Redeploy backend or restart service
2. To update frontend:
   - Make changes
   - Run `npm run build`

- Sync with S3

3. To update backend:
   - Make changes
   - Run `eb deploy`

Development Commands

Backend

```
dotnet run # Local development
eb deploy # Deploy to AWS
```

Frontend

```
npm run dev # Local development
npm run build # Build for production
aws s3 sync dist/ s3://tmotask-frontend --delete # Deploy to S3
```

Third-Party Libraries & Dependencies

Backend Libraries

1. CsvHelper

```
<PackageReference Include="CsvHelper" Version="30.0.1" />
```

Why Used:

- Makes CSV operations type-safe and maintainable

2. xUnit (Testing)

```
<PackageReference Include="xUnit" Version="2.4.2" />
```

Why Used:

- Industry standard for .NET testing
- Great support for async testing

3. Microsoft.NET.Test.Sdk

```
<PackageReference Include="Microsoft.NET.Test.Sdk" Version="17.14.1" />
```

Why Used:

- Required for running tests

Frontend Libraries

1. Lucide React

```
"lucide-react": "^0.358.0"
```

Why Used:

- Modern, clean icon set
- Lightweight package size
- Easy integration with React
- Customizable icons
- Consistent design language

2. SASS/SCSS

```
"sass": "^1.71.0"
```

Why Used:

- Advanced CSS features
- Better organization with nesting
- Modular CSS approach
- Reduces code duplication

3. Vite

```
"vite": "^5.0.8"
```

Why Used:

- Extremely fast development server
- Quick build times
- Built-in TypeScript support
- Modern module system

AWS SDK & CLI Tools

1. AWS CLI

```
aws-cli/2.15.0
```

Why Used:

- S3 bucket management

- Easy deployment scripts
- AWS resource management
- Configuration management
- Automation capabilities

2. Elastic Beanstalk CLI

```
eb-cli
```

Why Used:

- Simplified deployment process
- Environment management
- Application versioning
- Health monitoring
- Easy rollback capabilities

Development Tools

1. dotnet CLI

```
dotnet-sdk-8.0
```

Why Used:

- Project creation and management
- Package management
- Building and testing
- Cross-platform support
- Deployment preparation

2. npm

```
npm/10.2.4
```

Why Used:

- Package management
- Script running
- Dependency resolution
- Build process management
- Version control

Each library was carefully chosen to provide specific functionality while maintaining:

- Performance optimization
- Code maintainability
- Development efficiency
- Production reliability
- Modern development practices

These choices allow us to focus on business logic while leveraging battle-tested tools for common functionalities.