perplexity

Phase 1: AI Marketing Platform Setup for Windows 11

Perfect! Let me adapt the development setup specifically for Windows 11. Here's your comprehensive Phase 1 guide tailored for the Windows environment.

1.1 Technical Architecture Planning

The tech stack remains the same, but we'll use Windows-specific installation methods:

Core Stack:

- **Backend:** Node.js with Express.js
- **Frontend:** React with Next.js
- **Database:** PostgreSQL + Redis
- **Development:** Docker Desktop for Windows

1.2 Windows 11 Development Environment Setup

Step 1: Install Required Tools for Windows 11

Install Node.js:

```
# Download and install Node.js 18+ from official website
# Or use Chocolatey package manager
choco install nodejs

# Verify installation
node --version
npm --version
```

Install Docker Desktop:

```
# Download Docker Desktop for Windows from docker.com
# Or install via Chocolatey
choco install docker-desktop

# After installation, restart your computer and start Docker Desktop
```

Install Git:

```
# Download Git for Windows or use Chocolatey
choco install git

# Configure Git
git config --global user.name "Your Name"
git config --global user.email "your.email@example.com"
```

Install VS Code (Optional but Recommended):

```
# Download from code.visualstudio.com or use Chocolatey
choco install vscode

# Install useful extensions
code --install-extension ms-vscode.vscode-typescript-next
code --install-extension bradlc.vscode-tailwindcss
code --install-extension ms-vscode.vscode-docker
code --install-extension ms-vscode.PowerShell
```

Step 2: Project Structure Creation (Windows PowerShell)

```
# Create main project directory
mkdir ai-marketing-platform
cd ai-marketing-platform

# Create comprehensive folder structure
mkdir backend, frontend, database, docker-configs, docs, scripts, shared
mkdir backend\src, backend\tests, backend\config
mkdir backend\src\controllers, backend\src\models, backend\src\routes, backend\src\servic
mkdir frontend\src, frontend\public, frontend\components, frontend\pages, frontend\hooks,
mkdir database\migrations, database\seeds, database\schemas
mkdir docs\api, docs\deployment, docs\user-guide
mkdir shared\types, shared\constants, shared\validators

# Create initial configuration files
New-Item -ItemType File -Path "backend\package.json"
New-Item -ItemType File -Path "frontend\package.json"
New-Item -ItemType File -Path "docker-compose.yml"
New-Item -ItemType File -Path ".env.example"
New-Item -ItemType File -Path ".gitignore"
New-Item -ItemType File -Path "README.md"
```

Step 3: Backend Setup (Windows-Specific)

```
cd backend

# Initialize package.json
npm init -y

# Install core dependencies
npm install express cors helmet morgan dotenv bcryptjs jsonwebtoken
```

```
npm install mongoose pg redis ioredis
npm install @google/maps googleapis nodemailer stripe

# Install development dependencies
npm install -D nodemon jest supertest eslint prettier husky lint-staged
npm install -D @types/node @types/express typescript ts-node

# Create TypeScript configuration
npx tsc --init
```

Windows-Optimized Package.json Scripts:

```
{
  "name": "ai-marketing-backend",
  "version": "1.0.0",
  "main": "src/server.js",
  "scripts": {
    "dev": "nodemon src/server.js",
    "dev:win": "nodemon --exec \"node src/server.js\"",
    "start": "node src/server.js",
    "test": "jest",
    "test:win": "jest --runInBand",
    "lint": "eslint src/",
    "format": "prettier --write src/",
    "build": "tsc",
    "clean": "if exist dist rmdir /s /q dist"
  },
  "engines": {
    "node": ">=18.0.0"
  }
}
```

Step 4: Frontend Setup (Windows Command)

```
cd ..\frontend

# Create Next.js app
npx create-next-app@latest . --typescript --tailwind --eslint --app

# Install additional dependencies
npm install axios @tanstack/react-query
npm install @headlessui/react @heroicons/react
npm install react-hook-form yup @hookform/resolvers
npm install recharts lucide-react

# Windows-specific development dependencies
npm install -D cross-env
```

Step 5: Windows-Optimized Docker Configuration

Create `docker-compose.yml` (Windows paths):

```
version: '3.8'

services:
  # PostgreSQL Database
  postgres:
    image: postgres:15
    environment:
      POSTGRES_DB: ai_marketing_platform
      POSTGRES_USER: admin
      POSTGRES_PASSWORD: password123
    ports:
      - "5432:5432"
    volumes:
      - postgres_data:/var/lib/postgresql/data
      - ./database/schemas:/docker-entrypoint-initdb.d
    networks:
      - ai-marketing-network

  # Redis Cache
  redis:
    image: redis:7-alpine
    ports:
      - "6379:6379"
    volumes:
      - redis_data:/data
    networks:
      - ai-marketing-network

  # Backend API
  backend:
    build:
      context: ./backend
      dockerfile: Dockerfile.windows
    ports:
      - "3001:3001"
    environment:
      - NODE_ENV=development
      - DATABASE_URL=postgresql://admin:password123@postgres:5432/ai_marketing_platform
      - REDIS_URL=redis://redis:6379
    volumes:
      - ./backend:/app
      - /app/node_modules
    depends_on:
      - postgres
      - redis
    networks:
      - ai-marketing-network

  # Frontend
  frontend:
    build:
      context: ./frontend
```

```
      dockerfile: Dockerfile.windows
    ports:
      - "3000:3000"
    environment:
      - NEXT_PUBLIC_API_URL=http://localhost:3001
    volumes:
      - ./frontend:/app
      - /app/node_modules
    depends_on:
      - backend
    networks:
      - ai-marketing-network

volumes:
  postgres_data:
  redis_data:

networks:
  ai-marketing-network:
    driver: bridge
```

Step 6: Windows Environment Configuration

Create `.env.example` (Windows-friendly paths):

```
# Database
DATABASE_URL=postgresql://admin:password123@localhost:5432/ai_marketing_platform
REDIS_URL=redis://localhost:6379

# JWT
JWT_SECRET=your-super-secret-jwt-key-change-this-in-production
JWT_EXPIRES_IN=7d

# Google APIs
GOOGLE_MAPS_API_KEY=your-google-maps-api-key
GOOGLE_CLIENT_ID=your-google-oauth-client-id
GOOGLE_CLIENT_SECRET=your-google-oauth-client-secret

# Email Service
SENDGRID_API_KEY=your-sendgrid-api-key
EMAIL_FROM=noreply@yourdomain.com

# Stripe Payment
STRIPE_SECRET_KEY=sk_test_your-stripe-secret-key
STRIPE_WEBHOOK_SECRET=whsec_your-webhook-secret

# OpenAI
OPENAI_API_KEY=your-openai-api-key

# General
NODE_ENV=development
PORT=3001
FRONTEND_URL=http://localhost:3000

# Windows-specific
```

```
TEMP_DIR=C:\temp\ai-marketing
LOG_DIR=C:\logs\ai-marketing
```

Step 7: Create Windows Dockerfile

Create `backend/Dockerfile.windows`**:**

```
FROM node:18-alpine

WORKDIR /app

# Copy package files
COPY package*.json ./

# Install dependencies
RUN npm ci --only=production

# Copy source code
COPY . .

# Create non-root user for Windows compatibility
RUN addgroup -g 1001 -S nodejs
RUN adduser -S nextjs -u 1001

USER nextjs

EXPOSE 3001

CMD ["npm", "start"]
```

Step 8: Windows-Specific Scripts

Create `scripts/setup-windows.ps1`**:**

```
# Windows Setup Script for AI Marketing Platform

Write-Host "Setting up AI Marketing Platform on Windows 11..." -ForegroundColor Green

# Check if required tools are installed
function Test-Command($cmdname) {
    return [bool](Get-Command -Name $cmdname -ErrorAction SilentlyContinue)
}

# Verify Node.js installation
if (-not (Test-Command node)) {
    Write-Host "Node.js not found. Please install Node.js 18+" -ForegroundColor Red
    exit 1
}

# Verify Docker installation
if (-not (Test-Command docker)) {
    Write-Host "Docker not found. Please install Docker Desktop" -ForegroundColor Red
    exit 1
```

```
}

# Create .env file from example
if (-not (Test-Path ".env")) {
    Copy-Item ".env.example" ".env"
    Write-Host ".env file created. Please update with your API keys." -ForegroundColor Ye
}

# Install backend dependencies
Write-Host "Installing backend dependencies..." -ForegroundColor Blue
cd backend
npm install
cd ..

# Install frontend dependencies
Write-Host "Installing frontend dependencies..." -ForegroundColor Blue
cd frontend
npm install
cd ..

# Start Docker services
Write-Host "Starting Docker services..." -ForegroundColor Blue
docker-compose up -d postgres redis

Write-Host "Setup complete! Run 'npm run dev' in backend and frontend folders to start de
```

Step 9: Windows Development Commands

Create `scripts/dev-start.ps1`**:**

```
# Development startup script for Windows

Write-Host "Starting AI Marketing Platform Development Environment..." -ForegroundColor C

# Start Docker services
Write-Host "Starting database services..." -ForegroundColor Blue
docker-compose up -d postgres redis

# Wait for services to be ready
Start-Sleep -Seconds 5

# Start backend in new terminal
Write-Host "Starting backend server..." -ForegroundColor Blue
Start-Process powershell -ArgumentList "-NoExit", "-Command", "cd backend; npm run dev"

# Start frontend in new terminal
Write-Host "Starting frontend server..." -ForegroundColor Blue
Start-Process powershell -ArgumentList "-NoExit", "-Command", "cd frontend; npm run dev"

Write-Host "Development servers starting..." -ForegroundColor Green
Write-Host "Backend: http://localhost:3001" -ForegroundColor Yellow
Write-Host "Frontend: http://localhost:3000" -ForegroundColor Yellow
Write-Host "Database: localhost:5432" -ForegroundColor Yellow
Write-Host "Redis: localhost:6379" -ForegroundColor Yellow
```

Windows 11 Specific Setup Instructions

1. Enable WSL2 (Optional but Recommended)

```
# Run as Administrator
wsl --install

# After restart, set WSL2 as default
wsl --set-default-version 2
```

2. Configure Windows Terminal

```
// Add to Windows Terminal settings.json
{
    "profiles": {
        "list": [
            {
                "name": "AI Marketing Dev",
                "commandline": "powershell.exe -NoExit -Command \"cd C:\\your-path\\ai-ma
                "startingDirectory": "C:\\your-path\\ai-marketing-platform"
            }
        ]
    }
}
```

3. Windows Firewall Configuration

```
# Allow Node.js and Docker through Windows Firewall
New-NetFirewallRule -DisplayName "Node.js Dev Server" -Direction Inbound -Port 3000,3001
```

Quick Start Commands for Windows 11

```
# Clone and setup (run these in order)
git clone https://github.com/yourusername/ai-marketing-platform.git
cd ai-marketing-platform

# Run setup script
.\scripts\setup-windows.ps1

# Start development environment
.\scripts\dev-start.ps1
```

Troubleshooting Windows 11 Issues

Common Windows Issues:

1. **Port conflicts:** Use `netstat -ano | findstr :3000` to check port usage
2. **Docker issues:** Ensure Docker Desktop is running and virtualization is enabled in BIOS
3. **Permission errors:** Run PowerShell as Administrator when needed
4. **Path length limits:** Enable long path support in Windows

This Windows 11-optimized setup will give you a solid foundation for Phase 1 development. The next step would be implementing the authentication system and basic API endpoints.

Would you like me to help you with the next phase of development, such as setting up the authentication routes or database connections?