🌾 HarvestGuard - Complete VS Code Setup Guide

📋 Prerequisites (Install First)

Before starting, install these on your computer:

1. **Node.js** (v18 or higher)
   - Download: https://nodejs.org
   - Verify: Open terminal → `node --version`
2. **PostgreSQL** (Database)
   - Download: https://www.postgresql.org/download/
   - Or use Supabase (cloud, easier): https://supabase.com
3. **VS Code**
   - Download: https://code.visualstudio.com
4. **Git** (optional, for version control)
   - Download: https://git-scm.com

🚀 PART 1: PROJECT SETUP (10 minutes)

Step 1: Create Main Project Folder

```bash
# Open terminal (Terminal → New Terminal in VS Code)
mkdir HarvestGuard
cd HarvestGuard
```

Step 2: Open in VS Code

```bash
code .
```

This opens VS Code in the HarvestGuard folder.

📁 PART 2: FRONTEND SETUP (15 minutes)

Step 1: Create Frontend with Vite

In VS Code terminal:

```bash
npm create vite@latest frontend -- --template react
cd frontend
npm install
```

Step 2: Install Dependencies

```bash
npm install tailwindcss@latest postcss autoprefixer
npm install react-i18next i18next
npm install framer-motion
npm install recharts
npm install axios
```

Step 3: Setup Tailwind CSS

```bash
npx tailwindcss init -p
```

Edit `tailwind.config.js`:

```javascript
/** @type {import('tailwindcss').Config} */
export default {
  content: [
    "./index.html",
    "./src/**/*.{js,ts,jsx,tsx}",
  ],
  theme: {
    extend: {},
  },
  plugins: [],
}
```

Edit `src/index.css`:

```css
@tailwind base;
@tailwind components;
@tailwind utilities;
```

Step 4: Create Folder Structure

In VS Code, create these folders inside `frontend/src/`:

```
src/
  components/
  pages/
  services/
  utils/
```

Step 5: Create Landing Page

File: `src/pages/Landing.jsx`

Copy the Landing Page code from the artifact I created earlier.

Step 6: Create Dashboard

File: `src/pages/Dashboard.jsx`

Copy the Dashboard code from the artifact.

Step 7: Create Scanner

File: `src/pages/Scanner.jsx`

Copy the Scanner code from the artifact.

Step 8: Create API Service

File: `src/services/api.js`

```javascript
```

```
import axios from 'axios';

const API_URL = import.meta.env.VITE_API_URL || 'http://localhost:5000/api';

const api = axios.create({
  baseURL: API_URL,
  headers: {
    'Content-Type': 'application/json',
  },
});

// Add token to requests
api.interceptors.request.use((config) => {
  const token = localStorage.getItem('token');
  if (token) {
    config.headers.Authorization = `Bearer ${token}`;
  }
  return config;
});

export const auth = {
  register: (data) => api.post('/auth/register', data),
  login: (data) => api.post('/auth/login', data),
};

export const batches = {
  create: (data) => api.post('/batches', data),
  getAll: () => api.get('/batches'),
  getById: (id) => api.get(`/batches/${id}`),
};

export const weather = {
  get: (district) => api.get(`/weather?district=${district}`),
};

export const market = {
  getLatest: (district, commodity) =>
    api.get(`/market/latest?district=${district}&commodity=${commodity}`),
};

export default api;
```

Step 9: Update App.jsx

File: `src/App.jsx`

javascript

```
import React, { useState } from 'react';
import Landing from './pages/Landing';
import Dashboard from './pages/Dashboard';
import Scanner from './pages/Scanner';

function App() {
  const [currentPage, setCurrentPage] = useState('landing');
  const [isLoggedIn, setIsLoggedIn] = useState(false);

  const renderPage = () => {
    if (!isLoggedIn && currentPage !== 'landing') {
      return <Landing onLogin={() => setIsLoggedIn(true)} />;
    }

    switch (currentPage) {
      case 'landing':
        return <Landing onLogin={() => setIsLoggedIn(true)} />;
      case 'dashboard':
        return <Dashboard />;
      case 'scanner':
        return <Scanner />;
      default:
        return <Landing />;
    }
  };

  return <div>{renderPage()}</div>;
}

export default App;
```

Step 10: Create Environment File

File: `frontend/.env`

```
VITE_API_URL=http://localhost:5000/api
VITE_WEATHER_API_KEY=your-key-here
```

Step 11: Test Frontend

```bash
npm run dev
```

Open browser: http://localhost:5173

🔧 PART 3: BACKEND SETUP (20 minutes)

Step 1: Create Backend Folder

In terminal (go back to main HarvestGuard folder):

```bash
cd ..
mkdir backend
cd backend
```

Step 2: Initialize Node Project

```bash
npm init -y
```

Step 3: Install Dependencies

```bash
npm install express cors dotenv bcryptjs jsonwebtoken pg axios
npm install --save-dev nodemon
```

Step 4: Create Folder Structure

In VS Code, create these folders inside `backend/`:

```
backend/
  routes/
  controllers/
  services/
  middleware/
  jobs/
  db/
  utils/
```

Step 5: Create All Backend Files

Copy each file from the backend artifact I created:

File: `backend/server.js`

Copy the server.js code

File: `backend/routes/auth.js`

Copy the auth route code

File: `backend/routes/batches.js`

Copy the batches route code

File: `backend/routes/weather.js`

Copy the weather route code

File: `backend/routes/market.js`

Copy the market route code

File: `backend/routes/jobs.js`

Copy the jobs route code

File: `backend/controllers/authController.js`

Copy the auth controller code

File: `backend/controllers/batchController.js`

Copy the batch controller code

File: `backend/controllers/weatherController.js`

Copy the weather controller code

File: `backend/controllers/marketController.js`

Copy the market controller code

File: `backend/services/weatherService.js`

Copy the weather service code

File: `backend/services/etclService.js`

Copy the ETCL service code

File: `backend/services/marketService.js`

Copy the market service code

File: `backend/middleware/auth.js`

Copy the auth middleware code

File: `backend/jobs/marketScraper.js`

Copy the market scraper code

File: `backend/db/connect.js`

Copy the database connection code

File: `backend/utils/helpers.js`

Copy the helper functions code

Step 6: Update package.json Scripts

File: `backend/package.json`

Add these scripts:

```json
{
  "scripts": {
    "start": "node server.js",
    "dev": "nodemon server.js",
    "sync-market": "node jobs/marketScraper.js"
  }
}
```

Step 7: Create Environment File

File: `backend/.env`

```
PORT=5000
DATABASE_URL=postgresql://postgres:password@localhost:5432/harvestguard
JWT_SECRET=your-super-secret-key-change-this-in-production
WEATHER_API_KEY=get-from-openweathermap.org
NODE_ENV=development
```

PART 4: DATABASE SETUP (10 minutes)

Option A: Local PostgreSQL

Step 1: Create Database

Open terminal:

bash

```
# Login to PostgreSQL
psql -U postgres

# Create database
CREATE DATABASE harvestguard;

# Exit
\q
```

Step 2: Create Schema

File: `backend/db/schema.sql`

sql

```
CREATE TABLE farmers (
  id SERIAL PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  email VARCHAR(255) UNIQUE NOT NULL,
  password VARCHAR(255) NOT NULL,
  phone VARCHAR(20),
  district VARCHAR(100),
  preferred_lang VARCHAR(10) DEFAULT 'bn',
  created_at TIMESTAMP DEFAULT NOW()
);

CREATE TABLE crop_batches (
  id SERIAL PRIMARY KEY,
  farmer_id INTEGER REFERENCES farmers(id),
  crop_type VARCHAR(100) NOT NULL,
  weight DECIMAL(10,2) NOT NULL,
  storage_type VARCHAR(50),
  district VARCHAR(100),
  status VARCHAR(50) DEFAULT 'active',
  created_at TIMESTAMP DEFAULT NOW()
);

CREATE TABLE market_prices (
  id SERIAL PRIMARY KEY,
  district VARCHAR(100) NOT NULL,
  commodity VARCHAR(100) NOT NULL,
  price DECIMAL(10,2) NOT NULL,
  market_name VARCHAR(255),
  date DATE NOT NULL,
  created_at TIMESTAMP DEFAULT NOW()
);

CREATE INDEX idx_market_prices_lookup ON market_prices(district, commodity, date);
CREATE INDEX idx_batches_farmer ON crop_batches(farmer_id);
```

Step 3: Run Schema

bash

```
psql -U postgres -d harvestguard -f backend/db/schema.sql
```

Option B: Supabase (Easier, Cloud-based)

1. Go to https://supabase.com
2. Create free account
3. Create new project
4. Go to SQL Editor
5. Paste the schema SQL above
6. Run it
7. Go to Settings → Database → Copy connection string
8. Paste in `backend/.env` as `DATABASE_URL`

🎯 PART 5: GET API KEYS (5 minutes)

Weather API Key

1. Go to https://openweathermap.org
2. Sign up (free)
3. Go to API Keys
4. Copy your key
5. Paste in `backend/.env` as `WEATHER_API_KEY`

▶️ PART 6: RUN THE PROJECT (2 minutes)

Step 1: Start Backend

Open terminal in VS Code:

```bash
cd backend
npm run dev
```

You should see: `✅ Server running on port 5000`

Step 2: Start Frontend (New Terminal)

Open new terminal (click + icon):

```bash
cd frontend
npm run dev
```

You should see: `Local: http://localhost:5173`

Step 3: Open Browser

Go to: http://localhost:5173

✅ PART 7: TEST EVERYTHING (5 minutes)

Test 1: Landing Page

- Should load with Bangla text
- Toggle language should work
- Animations should play

Test 2: Register User

Open new terminal:

```bash
curl -X POST http://localhost:5000/api/auth/register \
  -H "Content-Type: application/json" \
  -d '{
    "name": "Test Farmer",
    "email": "test@example.com",
    "password": "password123",
    "phone": "01712345678",
    "district": "Gazipur",
    "preferredLang": "bn"
  }'
```

Should return token.

Test 3: Login

```bash
curl -X POST http://localhost:5000/api/auth/login \
  -H "Content-Type: application/json" \
  -d '{
    "email": "test@example.com",
    "password": "password123"
  }'
```

Test 4: Weather

```bash
curl "http://localhost:5000/api/weather?district=Gazipur"
```

Should return weather data.

📁 FINAL FOLDER STRUCTURE

```
HarvestGuard/
├── frontend/
│   ├── public/
│   ├── src/
│   │   ├── components/
│   │   ├── pages/
│   │   │   ├── Landing.jsx
│   │   │   ├── Dashboard.jsx
│   │   │   └── Scanner.jsx
│   │   ├── services/
│   │   │   └── api.js
│   │   ├── utils/
│   │   ├── App.jsx
│   │   ├── main.jsx
│   │   └── index.css
│   ├── .env
│   ├── package.json
│   ├── tailwind.config.js
│   └── vite.config.js
│
```

```
├── backend/
│   ├── routes/
│   │   ├── auth.js
│   │   ├── batches.js
│   │   ├── weather.js
│   │   ├── market.js
│   │   └── jobs.js
│   ├── controllers/
│   │   ├── authController.js
│   │   ├── batchController.js
│   │   ├── weatherController.js
│   │   └── marketController.js
│   ├── services/
│   │   ├── weatherService.js
│   │   ├── etclService.js
│   │   └── marketService.js
│   ├── middleware/
│   │   └── auth.js
│   ├── jobs/
│   │   └── marketScraper.js
│   ├── db/
│   │   ├── connect.js
│   │   └── schema.sql
│   ├── utils/
│   │   └── helpers.js
│   ├── .env
│   ├── server.js
│   └── package.json
│
└── README.md
```

🐛 TROUBLESHOOTING

Issue: "Cannot find module 'express'"

Fix: Run `npm install` in backend folder

Issue: "Port 5000 already in use"

Fix: Change PORT in backend/.env to 5001

Issue: "Database connection failed"

Fix: Check DATABASE_URL in backend/.env

Issue: "CORS error"

Fix: Backend server.js already has CORS enabled

Issue: "Weather API not working"

Fix: Get free API key from openweathermap.org

🚀 NEXT STEPS

For Hackathon Demo:

1. **Create sample data:**

```bash
npm run sync-market
```

2. **Practice demo flow:**
   - Show landing page
   - Register/login
   - Create batch
   - Show weather alerts
   - Show market recommendations
   - Demo AI scanner
3. **Prepare talking points:**
   - Problem: Post-harvest loss in Bangladesh
   - Solution: AI + weather + market intelligence
   - Impact: 40% loss reduction, 25% more profit

For Production:

1. Deploy frontend to Vercel

2. Deploy backend to Render/Railway
3. Use Supabase for database
4. Add SMS notifications (Twilio)
5. Train custom AI model

💡 VS Code Tips

Recommended Extensions:

1. **ES7+ React/Redux/React-Native snippets**
2. **Tailwind CSS IntelliSense**
3. **ESLint**
4. **Prettier**
5. **Thunder Client** (API testing)

Keyboard Shortcuts:

- `Ctrl + `` - Toggle terminal
- Ctrl + P - Quick file open
- Ctrl + Shift + P - Command palette
- Ctrl + B - Toggle sidebar
- F5 - Start debugging

🎉 YOU'RE DONE!

Your HarvestGuard project is now fully set up and running!

Test checklist:

- ✅ Frontend loads at http://localhost:5173
- ✅ Backend runs at http://localhost:5000
- ✅ Database connected
- ✅ API endpoints work

- ✅ Weather API integrated
- ✅ Bangla language works

Ready for hackathon! 🚀