Development Environment Setup Guide: Instant Marquees Melbourne Internal Operations Software (IMMIOS)

Project Name: Instant Marquees Melbourne Internal Operations Software (IMMIOS)

Version: 1.0
Date: July 11, 2025

1. Introduction

This guide provides step-by-step instructions for setting up your local development environment for the Instant Marquees Melbourne Internal Operations Software (IMMIOS). This setup is tailored for a **Next.js (React) frontend, Node.js backend (via Next.js API Routes), Tailwind CSS, and Firestore database**, with deployment targeted for **Vercel**.

This document is designed to be easily digestible and actionable, particularly when using Cursor AI for code generation and development.

2. Prerequisites

Before you begin, ensure you have the following software installed on your development machine:

- **Node.js:** Version 18.x or later.
  - You can download it from nodejs.org.
  - Verify installation: Open your terminal/command prompt and run node -v and npm -v.
- **Git:** For version control.
  - Download from git-scm.com.
  - Verify installation: git --version.
- **Firebase CLI (Command Line Interface):** For interacting with Firestore and Firebase services.
  - Install globally via npm: npm install -g firebase-tools
  - Verify installation: firebase --version
  - Log in to Firebase: firebase login (follow the prompts in your browser).
- **Vercel CLI:** For deploying your Next.js application to Vercel.
  - Install globally via npm: npm install -g vercel
  - Verify installation: vercel --version
  - Log in to Vercel: vercel login (follow the prompts in your browser).
- **Text Editor / IDE:**
  - **Visual Studio Code (VS Code)** is highly recommended due to its excellent

JavaScript/TypeScript support, extensions, and integration with AI tools like Cursor.
    - Download from code.visualstudio.com.
- **Cursor AI:** Ensure your Cursor AI environment is set up and ready to integrate with your VS Code or preferred IDE.

3. Project Setup

Follow these steps to get the IMMIOS project running locally:

3.1. Clone the Repository

First, clone the project's Git repository to your local machine. Replace [YOUR_REPOSITORY_URL] with the actual URL of your Git repository.

```
git clone [YOUR_REPOSITORY_URL]
cd instant-marquees-melbourne-app # Or whatever your project folder is named
```

3.2. Install Dependencies

Navigate into your project directory and install all necessary Node.js dependencies:

```
npm install
```

3.3. Firebase Project Setup & Configuration

You will need a Firebase project set up for your application.

1. **Create a Firebase Project:**
    - Go to the Firebase Console.
    - Click "Add project" and follow the steps to create a new project (e.g., "Instant Marquees App").
    - Enable **Firestore Database** and **Firebase Authentication** for your project.
        - For Firestore, choose "Start in production mode" (you will configure security rules later). Select a location for your database (e.g., asia-southeast2 for Melbourne).
        - For Authentication, enable "Email/Password" provider.
2. **Get Firebase Project Configuration:**
    - In your Firebase project, go to "Project settings" (the gear icon next to "Project overview").
    - Under "Your apps," click the "Web" icon (</>) to add a web app.
    - Follow the steps, register your app, and you will be provided with a Firebase

SDK snippet. Copy the firebaseConfig object from this snippet. It will look something like this:

```
const firebaseConfig = {
  apiKey: "YOUR_API_KEY",
  authDomain: "YOUR_AUTH_DOMAIN",
  projectId: "YOUR_PROJECT_ID",
  storageBucket: "YOUR_STORAGE_BUCKET",
  messagingSenderId: "YOUR_MESSAGING_SENDER_ID",
  appId: "YOUR_APP_ID",
  measurementId: "YOUR_MEASUREMENT_ID" // Optional
};
```

3. **Configure Environment Variables:**
   - Create a file named .env.local in the root of your project directory.
   - Add your Firebase configuration variables to this file, prefixed with NEXT_PUBLIC_ for frontend access, and without for backend API routes.

```
# Firebase Configuration (for Frontend & Backend)
NEXT_PUBLIC_FIREBASE_API_KEY="YOUR_API_KEY"
NEXT_PUBLIC_FIREBASE_AUTH_DOMAIN="YOUR_AUTH_DOMAIN"
NEXT_PUBLIC_FIREBASE_PROJECT_ID="YOUR_PROJECT_ID"
NEXT_PUBLIC_FIREBASE_STORAGE_BUCKET="YOUR_STORAGE_BUCKET"
NEXT_PUBLIC_FIREBASE_MESSAGING_SENDER_ID="YOUR_MESSAGING_SENDER_ID"
NEXT_PUBLIC_FIREBASE_APP_ID="YOUR_APP_ID"

# For Server-Side Admin SDK (Firebase Admin SDK requires service account)
# This is for more secure backend operations, e.g., creating users directly.
# For MVP, we might use client-side SDK for basic user creation via Admin UI,
# but for production, a service account is recommended for server-side operations.
# If using Firebase Admin SDK, you'll download a JSON key file and reference it here.
# FIREBASE_ADMIN_SDK_PRIVATE_KEY_ID="your_private_key_id"
# FIREBASE_ADMIN_SDK_PRIVATE_KEY="-----BEGIN PRIVATE KEY-----\n...\n-----END
PRIVATE KEY-----\n"
#
FIREBASE_ADMIN_SDK_CLIENT_EMAIL="your-service-account@your-project-id.iam.g
serviceaccount.com"
# FIREBASE_ADMIN_SDK_CLIENT_ID="your_client_id"
# FIREBASE_ADMIN_SDK_AUTH_URI="https://accounts.google.com/o/oauth2/auth"
# FIREBASE_ADMIN_SDK_TOKEN_URI="https://oauth2.googleapis.com/token"
#
FIREBASE_ADMIN_SDK_AUTH_PROVIDER_X509_CERT_URL="https://www.googleapis.c
om/oauth2/v1/certs"
# FIREBASE_ADMIN_SDK_CLIENT_X509_CERT_URL="your_client_x509_cert_url"
# FIREBASE_ADMIN_SDK_UNIVERSE_DOMAIN="googleapis.com"
```

Note: For the MVP, we will primarily use the client-side Firebase SDK for authentication and Firestore interactions directly from the frontend and Next.js API routes. If more secure server-side admin operations are needed, you'll need to set up a Firebase Admin SDK service account and its environment variables.

3.4. Firestore Security Rules

IMPORTANT: By default, Firestore starts in "production mode" with very restrictive security rules. You **MUST** update these rules to allow your application to read and write data.

1. Go to the Firestore Database section in your Firebase Console.
2. Navigate to the "Rules" tab.
3. Replace the default rules with the following (or a more refined version as your application evolves). These rules allow authenticated users to read and write to their own data and public data within the artifacts collection, which is the standard for Canvas environments.

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {

    // --- Standard Canvas Rules (DO NOT MODIFY) ---
    // Public data (shared across users for this app)
    match /artifacts/{appId}/public/data/{collection=**} {
      allow read, write: if request.auth != null;
    }

    // Private data (specific to a user for this app)
    match /artifacts/{appId}/users/{userId}/{collection=**} {
      allow read, write: if request.auth != null && request.auth.uid == userId;
    }
    // --- End Standard Canvas Rules ---

    // --- Your Application-Specific Rules (Modify as needed) ---
    // Allow authenticated users to read/write to 'users' collection
    match /users/{userId} {
      allow read, write: if request.auth != null && request.auth.uid == userId;
    }

    // Allow authenticated users to read/write to 'staff', 'products', 'components', 'vehicles', 'jobs' collections
```

```
    // For 'jobs', you might want more granular rules later (e.g., only update if
assigned staff)
    match /{collection}/{document} {
      allow read, write: if request.auth != null;
    }

    // Allow authenticated users to read/write to the 'settings' document
    match /settings/app-settings {
      allow read: if request.auth != null;
      allow write: if request.auth != null &&
get(/databases/$(database)/documents/users/$(request.auth.uid)).data.role ==
'admin';
    }

    // Allow authenticated users to read/write to 'user_notifications' collection
    match /user_notifications/{notificationId} {
      allow read, write: if request.auth != null && request.auth.uid ==
resource.data.userId;
    }
  }
}
```

Note: The __app_id and __initial_auth_token variables are provided by the Canvas environment at runtime. Your local setup will use the projectId from your .env.local for appId and you'll manage authentication via Firebase's standard methods (email/password).

3.5. Run the Development Server

Once dependencies are installed and Firebase is configured, start the Next.js development server:

```
npm run dev
```

This will typically start the application at http://localhost:3000. Open this URL in your web browser.

3.6. Initial User Creation (Admin)

Since this is an internal app, you'll need to create an initial admin user. For the prototype, you can manually create an admin user directly in the Firebase

Authentication console:

1. Go to Firebase Console -> Authentication.
2. Click "Add user."
3. Enter an email and password for your admin account.
4. After creating the user, you'll need to manually add a document to the users collection in Firestore with the same uid as the newly created user, and set their role to 'admin'.
    - **Collection:** users
    - **Document ID:** (The UID of the user you just created in Firebase Auth)
    - **Fields:**
        - email: "your-admin-email@example.com"
        - role: "admin"
        - createdAt: (timestamp)
        - lastLoginAt: (timestamp)

This will allow you to log in as an admin and access the admin functionalities.

4. Using Cursor AI

With your project set up, you can now use Cursor AI within your VS Code (or integrated environment) to assist with code generation.

- **Provide Context:** When prompting Cursor, be specific. Refer to the sections and fields defined in the **Technical Design Document** and **Requirements Document**.
- **Modular Requests:** Ask Cursor to generate code for specific components, API routes, or database interactions. For example:
    - "Generate a React component for the job calendar view, using react-big-calendar and Tailwind CSS."
    - "Write the Next.js API route for /api/jobs (POST) that creates a new job document in Firestore, including initial stock availability checks."
    - "Create the Socket.IO server-side logic to emit jobUpdated:productsOrdered when a job's productsOrdered array is modified."
    - "Design the Firestore security rules for the jobs collection to allow only authenticated users to create/update jobs."
- **Iterate:** Cursor might not get it perfect on the first try. Provide feedback and refine your prompts.

5. Deployment to Vercel

Once you have working code and are ready to deploy:

1. **Link Project to Vercel:**
   vercel link

   Follow the prompts to link your local project to a new or existing Vercel project.
2. **Add Environment Variables to Vercel:**
   - Go to your Vercel project dashboard.
   - Navigate to "Settings" -> "Environment Variables."
   - Add all the NEXT_PUBLIC_FIREBASE_... variables that you have in your .env.local file. Ensure they are available for the "Production," "Preview," and "Development" environments as needed.
3. **Deploy:**
   vercel

   Vercel will build and deploy your Next.js application. It will provide you with a preview URL, and if it's your production branch, your custom domain will be updated.

This guide should provide a solid foundation for you to begin developing the IMMIOS using your preferred tools and workflow. Good luck!