🚀 Executive Dashboard - Complete Antigravity Setup Guide

Copy this entire prompt into Antigravity to set up your Executive Dashboard with full guidance on obtaining all required credentials.

📋 Overview

This is a production-ready Executive Dashboard with:
- **Dashboard** - Real-time metrics and overview
- **Projects Management** - Syncs with ClickUp
- **Calendar Integration** - Syncs with Google Calendar
- **Invoice Tracking** - Automated email reminders
- **AI Assistant** - Query your data using Ollama (local/free) or OpenAI (cloud/paid)

Tech Stack: Next.js 14, TypeScript, Tailwind CSS, Supabase, ClickUp API, Google Calendar API, Gmail API

⚡ Quick Setup Steps

1. **Install Dependencies**
2. **Get API Keys & Credentials** (detailed guide below)
3. **Configure Environment Variables**
4. **Setup Supabase Database**
5. **Start AI Service** (Ollama or OpenAI)
6. **Run Development Server**
7. **Sync Your Data**

📦 Step 1: Install Dependencies

```bash
npm install
```

This installs all required packages (~3-5 minutes).

🔑 Step 2: Get All API Keys & Credentials

2.1 Supabase (Database) - **REQUIRED**

What it's for: Stores all your data (projects, invoices, calendar events, clients)

How to get it:

1. Go to https://supabase.com
2. Sign up/Login (free tier works perfectly)
3. Click "New Project"
   - Name: `executive-dashboard`
   - Database Password: Choose a strong password (save it!)
   - Region: Choose closest to you
   - Click "Create new project" (takes ~2 minutes)
4. Once created, go to **Settings** → **API**
5. Copy these 3 values:
   - **Project URL** (looks like: `https://xxxxx.supabase.co`)
   - **anon public** key (under "Project API keys")
   - **service_role** key (under "Project API keys")

Save these for Step 3!

2.2 ClickUp (Project Management) - **REQUIRED**

What it's for: Syncs your ClickUp tasks into the dashboard as projects

How to get it:

1. Go to https://app.clickup.com/settings/apps
2. Login to your ClickUp account
3. Scroll to **API Token** section
4. Click "Generate" or "Create an App"
5. Click "Generate" to create a Personal API Token
6. Copy the token (starts with `pk_`)
7. **Get your Space ID:**
   - Go to your ClickUp workspace
   - Click on a Space
   - Look at the URL: `app.clickup.com/{TEAM_ID}/v/li/{SPACE_ID}`
   - Copy the `{SPACE_ID}` number (usually 10 digits)

Example URL: `app.clickup.com/12345/v/li/9016996890`
- Space ID = `9016996890`

Save these for Step 3!

2.3 Google Service Account (Calendar & Gmail) - **REQUIRED**

What it's for: Syncs Google Calendar events and sends automated emails

How to get it:

1. Go to https://console.cloud.google.com
2. Sign in with your Google account

3. Click "Select a project" → "New Project"
   - Name: `executive-dashboard`
   - Click "Create"
4. Wait for project creation (~30 seconds)
5. **Enable APIs:**
   - In the search bar, type "Google Calendar API"
   - Click on it → Click "Enable"
   - Go back, search "Gmail API"
   - Click on it → Click "Enable"
6. **Create Service Account:**
   - In left menu: "IAM & Admin" → "Service Accounts"
   - Click "Create Service Account"
   - Name: `dashboard-bot`
   - Description: `Service account for executive dashboard`
   - Click "Create and Continue"
   - Skip role assignment (click "Continue")
   - Click "Done"
7. **Create Key:**
   - Click on the service account you just created
   - Go to "Keys" tab
   - Click "Add Key" → "Create new key"
   - Select "JSON"
   - Click "Create"
   - A JSON file downloads automatically
8. **Extract values from JSON:**
   - Open the downloaded JSON file
   - Copy these two values:
     - `client_email` (looks like: `dashboard-bot@xxxxx.iam.gserviceaccount.com`)
     - `private_key` (long text starting with `-----BEGIN PRIVATE KEY-----`)

Important: The `private_key` contains `\n` characters. Keep them as-is.

Calendar ID:
- Use `primary` or your Gmail address (e.g., `youremail@gmail.com`)

Save these for Step 3!

2.4 Google OAuth (Optional - For User Login)

What it's for: Allows users to login with Google (optional feature)

How to get it:

1. Same Google Cloud Console (console.cloud.google.com)
2. Go to "APIs & Services" → "Credentials"
3. Click "Create Credentials" → "OAuth client ID"
4. If prompted, configure OAuth consent screen:
   - User Type: "External"
   - App name: `Executive Dashboard`
   - User support email: Your email
   - Developer contact: Your email
   - Click "Save and Continue" through all screens
5. Back to "Create OAuth client ID":
   - Application type: "Web application"
   - Name: `Executive Dashboard Web`
   - Authorized redirect URIs: `http://localhost:3000/api/auth/google/callback`
   - Click "Create"
6. Copy:
   - **Client ID** (ends with `.apps.googleusercontent.com`)
   - **Client Secret** (starts with `GOCSPX-`)

Save these for Step 3!

2.5 AI Assistant - Choose ONE Option

Option A: Ollama (Local/Free) - Recommended for Development

What it's for: AI assistant that runs on your computer (free, private)

How to setup:

- Just use Docker (included in project)
- Will be set up automatically in Step 6

Option B: OpenAI (Cloud/Paid) - Recommended for Production

What it's for: AI assistant using ChatGPT (fast, reliable, paid)

How to get it:

1. Go to https://platform.openai.com/signup
2. Sign up/Login
3. Add payment method (pay-as-you-go, ~$0.01 per query)
4. Go to https://platform.openai.com/api-keys
5. Click "Create new secret key"
6. Name: `executive-dashboard`
7. Copy the key (starts with `sk-`)

Save for Step 3!

🔧 Step 3: Configure Environment Variables

Create a file named `.env.local` in the project root:

```bash
# Copy .env.example to .env.local
cp .env.example .env.local
```

Now edit `.env.local` and paste your credentials:

```env
```

```
# ==========================================
# EXECUTIVE DASHBOARD - ENVIRONMENT VARIABLES
# ==========================================


# SUPABASE (Required)
NEXT_PUBLIC_SUPABASE_URL=https://your-project-id.supabase.co
NEXT_PUBLIC_SUPABASE_ANON_KEY=your-anon-key-here
SUPABASE_SERVICE_ROLE_KEY=your-service-role-key-here


# CLICKUP (Required)
CLICKUP_API_TOKEN=pk_your_clickup_api_token_here
CLICKUP_SPACE_ID=your_space_id_here


# GOOGLE SERVICE ACCOUNT (Required)
GOOGLE_CLIENT_EMAIL=dashboard-bot@your-project.iam.gserviceaccount.com
GOOGLE_PRIVATE_KEY="-----BEGIN PRIVATE
KEY-----\nYOUR_PRIVATE_KEY_HERE\n-----END PRIVATE KEY-----\n"
GOOGLE_CALENDAR_ID=primary


# GMAIL SENDER (Required)
GMAIL_SENDER_EMAIL=your-email@gmail.com


# AI ASSISTANT - Choose Ollama OR OpenAI


# Option A: Ollama (Local/Free)
OLLAMA_BASE_URL=http://localhost:11434
OLLAMA_MODEL=llama3.1:8b


# Option B: OpenAI (Cloud/Paid)
OPENAI_API_KEY=sk-your-openai-api-key-here
OPENAI_MODEL=gpt-4o


# GOOGLE OAUTH (Optional)
GOOGLE_OAUTH_CLIENT_ID=your-client-id.apps.googleusercontent.com
GOOGLE_OAUTH_CLIENT_SECRET=GOCSPX-your-client-secret
```

```
# APP CONFIG
NEXT_PUBLIC_APP_URL=http://localhost:3000
```

⚠️ Important Notes:
- Replace ALL placeholder values with your actual credentials
- Keep the quotes around `GOOGLE_PRIVATE_KEY`
- If using Ollama, you can skip the OpenAI key
- If using OpenAI, you can skip the Ollama setup

🗄️ Step 4: Setup Supabase Database

1. Go to your Supabase Dashboard: https://supabase.com/dashboard
2. Select your project
3. Click **SQL Editor** in the left sidebar
4. Click "New query"
5. Open the file `supabase-schema.sql` in this project
6. Copy ALL the contents (Ctrl+A, Ctrl+C)
7. Paste into the Supabase SQL Editor
8. Click **RUN** (or press Ctrl+Enter)
9. Wait for success message (~5 seconds)

What this does:
- Creates 6 tables: clients, projects, invoices, calendar_events, roi_metrics, automation_logs
- Sets up indexes for performance
- Creates database views for dashboard stats
- Inserts sample data so you can see the dashboard working immediately

---

🤖 Step 5: Start AI Service

If using Ollama (Local/Free):

```bash
# Start Ollama container
docker-compose up -d


# Wait 30 seconds for container to start
sleep 30


# Pull the AI model (this takes 30-60 minutes, ~4.7GB download)
docker exec executive-ollama ollama pull llama3.1:8b
```


Note: The model download runs in background. The dashboard will work with sample responses until it completes.

If using OpenAI (Cloud/Paid):

Nothing to do! Just make sure you added your `OPENAI_API_KEY` in `.env.local`.

🚀 Step 6: Run Development Server


```bash
npm run dev
```


Expected output:
```
▲ Next.js 14.2.5
- Local:        http://localhost:3000
- Environments: .env.local
```

✓ Ready in 2s
```

**Open your browser:** http://localhost:3000

You should see the Executive Dashboard! 🎉

---

## 🔄 Step 7: Sync Your Real Data

The dashboard starts with sample data. Now let's sync your real data:

### Sync ClickUp Tasks:

1. Click the **"Sync All"** button in the top navigation bar
2. Or use the API:
```bash
curl -X POST http://localhost:3000/api/clickup/sync
```

This imports all your ClickUp tasks as projects.

Sync Google Calendar Events:

1. Click the **"Sync All"** button (syncs both)
2. Or use the API:
```bash
curl -X POST http://localhost:3000/api/calendar/sync
```

This imports your calendar events.
```

🎯 Features Guide

Dashboard Page (`/dashboard`)
- Pipeline value from leads
- Active projects count
- Overdue invoices total
- Upcoming meetings
- AI Assistant chat
- ROI metrics

Projects Page (`/projects`)
- View all projects
- Filter by status (active, planning, on_hold, completed)
- See project health scores
- Budget tracking
- ClickUp sync status

Calendar Page (`/calendar`)
- Monthly calendar view
- Google Calendar sync
- Event details
- Upcoming events list

Invoices Page (`/invoices`)
- All invoices list
- Filter by status (draft, sent, paid, overdue)
- Send reminder emails
- Track payments

Settings Page (`/settings`)
- Configure integrations
- Manage API connections
- View sync history

🤖 Using the AI Assistant

The AI Assistant can answer questions about your data:

Example questions:
- "Show me all overdue invoices"
- "What projects are due this week?"
- "How many meetings do I have today?"
- "What's my total pipeline value?"
- "List active projects with low health scores"
- "Send reminder for invoice INV-001"

The AI queries your Supabase database and provides accurate answers based on your real data.

🔧 Troubleshooting

Issue: "Cannot find module" errors
Solution:
```bash
rm -rf node_modules package-lock.json
npm install
```

Issue: CSS errors about "border-border class"
Solution: Make sure `tailwind.config.ts` includes:
```typescript
colors: {
  border: 'hsl(var(--border))',
  input: 'hsl(var(--input))',
  // ... other color variables
}
```

```

### Issue: Port 3000 already in use
**Solution:**
```bash
# Windows
netstat -ano | findstr :3000
taskkill /F /PID <PID_NUMBER>

# Mac/Linux
lsof -ti:3000 | xargs kill -9
```

Issue: Supabase connection errors
Solution:
- Verify `.env.local` has correct Supabase URL and keys
- Check Supabase project is active
- Verify schema was run successfully

Issue: ClickUp sync not working
Solution:
- Verify ClickUp API token is valid
- Check Space ID is correct (from ClickUp URL)
- Make sure you have tasks in that Space

Issue: Google Calendar not syncing
Solution:
- Verify Service Account credentials
- Make sure Calendar API is enabled in Google Cloud Console
- Check `GOOGLE_CALENDAR_ID` is correct

Issue: AI Assistant not responding
For Ollama:
- Check Docker is running: `docker ps`
- Verify container is up: `docker logs executive-ollama`
- Wait for model download to complete
```

For OpenAI:
- Verify API key is valid
- Check you have credits in your OpenAI account
- Test at https://platform.openai.com/playground

📊 Testing the Setup

1. Check Dashboard Loads
- Go to http://localhost:3000
- Should see sample data

2. Test ClickUp Sync
```bash
curl -X POST http://localhost:3000/api/clickup/sync
```
Should return: `{"success": true, "synced": X, "message": "..."}`

3. Test Calendar Sync
```bash
curl -X POST http://localhost:3000/api/calendar/sync
```
Should return: `{"success": true, "synced": X}`

4. Test AI Assistant
- Type "hi" in the AI chat
- Should get a response

5. Check All Pages
- Dashboard: http://localhost:3000/dashboard
- Projects: http://localhost:3000/projects
- Calendar: http://localhost:3000/calendar
- Invoices: http://localhost:3000/invoices

🚢 Production Deployment

When ready to deploy:

Build for Production:
```bash
npm run build
npm start
```

Deploy to Vercel (Recommended):
1. Push code to GitHub
2. Go to https://vercel.com
3. Import your repository
4. Add all environment variables from `.env.local`
5. Deploy!

Deploy to Other Platforms:
- Set all environment variables in your hosting dashboard
- Make sure Node.js 18+ is available
- Run `npm run build` then `npm start`

📚 Additional Resources

API Endpoints

Endpoint	Method	Description
`/api/projects`	GET	Get all projects
`/api/projects`	POST	Create new project
`/api/clickup/sync`	POST	Sync ClickUp tasks
`/api/calendar/events`	GET	Get calendar events

`/api/calendar/sync`	POST	Sync Google Calendar
`/api/invoices`	GET	Get all invoices
`/api/gmail/send`	POST	Send email
`/api/ai/chat`	POST	Chat with AI assistant
`/api/health`	GET	Check system health

Database Tables

- `clients` - Client information
- `projects` - Projects (synced from ClickUp)
- `invoices` - Invoice tracking
- `calendar_events` - Calendar events (synced from Google)
- `roi_metrics` - ROI tracking metrics
- `automation_logs` - Automation activity logs

✅ Setup Complete Checklist

Before considering setup complete:

- [ ] Dependencies installed (`npm install`)
- [ ] `.env.local` created with all credentials
- [ ] Supabase schema executed successfully
- [ ] Development server running (`npm run dev`)
- [ ] Dashboard loads at http://localhost:3000
- [ ] ClickUp sync tested and working
- [ ] Google Calendar sync tested and working
- [ ] AI Assistant responds to messages
- [ ] All pages accessible (Dashboard, Projects, Calendar, Invoices)

🎉 Success!

Your Executive Dashboard is now fully set up and running!

What's Next?
1. Customize the dashboard to your needs
2. Add more projects and clients
3. Set up automated sync (cron jobs)
4. Invite team members
5. Deploy to production

Need Help?
- Check browser console (F12) for errors
- Check server logs in terminal
- Verify all API keys are correct
- Make sure Supabase schema ran successfully

Enjoy your new Executive Dashboard! 🚀