**GitHub Backup Guide for Mingus Project**

**Step 1: Create GitHub Repository**

**Option A: Via GitHub Website (Recommended for beginners)**

1. **Go to GitHub.com** and log into your account
2. **Click the "+" icon** in top right corner → "New repository"
3. **Repository settings:**
   * Repository name: mingus-finance-app
   * Description: Personal finance assistant connecting wellness to financial decisions
   * **Make it Private** (recommended for your business)
   * **DO NOT** initialize with README, .gitignore, or license (since you have existing code)
4. **Click "Create repository"**

**Option B: Via GitHub CLI (if you have it installed)**

gh repo create mingus-finance-app --private --description "Personal finance assistant connecting wellness to financial decisions"

**Step 2: Prepare Your Local Project**

**Navigate to your project directory**

cd /path/to/your/mingus-project

**Check if Git is already initialized**

git status

**If you see "fatal: not a git repository":**

git init

**If Git is already initialized, check current status:**

git status

git remote -v

**Step 3: Create Essential Files**

**Create .gitignore file (CRITICAL for security)**

Create a file named .gitignore in your root directory:

# Environment variables (NEVER commit these)

.env

.env.local

.env.production

.env.staging

# API Keys and Secrets

config/secrets.py

secrets.json

credentials.json

# Database

\*.db

\*.sqlite

\*.sqlite3

# Python

\_\_pycache\_\_/

\*.py[cod]

\*$py.class

\*.so

.Python

build/

develop-eggs/

dist/

downloads/

eggs/

.eggs/

lib/

lib64/

parts/

sdist/

var/

wheels/

\*.egg-info/

.installed.cfg

\*.egg

# Virtual environments

venv/

env/

ENV/

env.bak/

venv.bak/

.venv/

# IDEs

.vscode/

.idea/

\*.swp

\*.swo

\*~

# OS generated files

.DS\_Store

.DS\_Store?

.\_\*

.Spotlight-V100

.Trashes

ehthumbs.db

Thumbs.db

# Node modules (if using any frontend)

node\_modules/

npm-debug.log\*

yarn-debug.log\*

yarn-error.log\*

# Logs

\*.log

logs/

# Temporary files

\*.tmp

\*.temp

**Create README.md file**

# Mingus - Personal Finance Assistant

A personal finance app that connects physical, mental, and relational health to financial decisions for African American professionals.

## Features

- Cash flow forecasting

- Wellness-finance correlation tracking

- Budget planning with milestone projections

- Weekly wellness check-ins

## Target Audience

African American professionals (25-35) earning $40k-$100k annually

## Tech Stack

- Backend: Python/Flask

- Database: Supabase

- Frontend: React/React Native

- Analytics: Pandas, NumPy

## Installation

[Installation instructions to be added]

## Usage

[Usage instructions to be added]

## Contributing

This is a private project by Johnnie Watson.

## License

Private - All rights reserved

**Step 4: Stage and Commit Your Files**

**Check what files will be added**

git status

**Add files to staging (be selective!)**

# Add specific files/folders (RECOMMENDED)

git add README.md

git add .gitignore

git add src/

git add requirements.txt

git add package.json

# OR add everything (CAREFUL - make sure .env files are excluded)

git add .

**Check staged files before committing**

git status

**⚠️ CRITICAL CHECK:** Make sure you don't see any of these files:

* .env files
* Database files
* Files with API keys
* Personal credentials

**Create initial commit**

git commit -m "Initial commit: Mingus personal finance app

- Added core application structure

- Implemented questionnaire system for health/relationship/career tracking

- Set up Supabase integration for data storage

- Created correlation analysis for wellness-finance insights"

**Step 5: Connect to GitHub and Push**

**Add GitHub repository as remote origin**

# Replace YOUR\_USERNAME with your actual GitHub username

git remote add origin https://github.com/YOUR\_USERNAME/mingus-finance-app.git

**Verify remote connection**

git remote -v

**Push to GitHub**

# First push requires setting upstream

git push -u origin main

# If you get an error about 'master' vs 'main':

git branch -M main

git push -u origin main

**Step 6: Verify Success**

1. **Go to your GitHub repository** in browser
2. **Check that your files are there** (but no sensitive data!)
3. **Verify README.md displays properly**
4. **Check that .gitignore is working** (no .env files should be visible)

**Step 7: Set Up Ongoing Backup Workflow**

**Daily/Weekly commits**

# Check what's changed

git status

# Add changes

git add .

# Commit with descriptive message

git commit -m "Add user authentication system and improve questionnaire UI"

# Push to GitHub

git push

**Good commit message examples:**

git commit -m "Add correlation analysis for stress vs spending patterns"

git commit -m "Fix bug in cash balance calculation"

git commit -m "Implement weekly insight email generation"

git commit -m "Update UI for mobile responsiveness"

**Common Issues & Solutions**

**Issue 1: "Permission denied (publickey)"**

**Solution:**

# Use HTTPS instead of SSH for easier setup

git remote set-url origin https://github.com/YOUR\_USERNAME/mingus-finance-app.git

**Issue 2: "Repository not found"**

**Check:**

* Repository name is correct
* Repository exists on GitHub
* You have access to the repository

**Issue 3: "Updates were rejected"**

**Solution:**

# Pull latest changes first

git pull origin main

# Then push

git push origin main

**Issue 4: Accidentally committed sensitive files**

**Solution:**

# Remove from Git but keep local file

git rm --cached .env

# Add to .gitignore

echo ".env" >> .gitignore

# Commit the removal

git add .gitignore

git commit -m "Remove sensitive files and update .gitignore"

git push

**⚠️ Important:** If you already pushed sensitive data, you may need to:

1. Change any exposed API keys immediately
2. Consider repository history cleanup (advanced topic)

**Cursor-Specific Issues**

**Why Cursor might fail:**

1. **Git not properly configured** in Cursor settings
2. **Authentication issues** - Cursor might not have GitHub access
3. **File conflicts** - Cursor trying to commit files that shouldn't be committed

**Using Cursor successfully:**

1. **Use terminal in Cursor** instead of Git UI when possible
2. **Configure Git credentials** in Cursor settings
3. **Always review** what Cursor wants to commit before confirming

**Security Checklist**

Before each push, verify:

* [ ] No .env files committed
* [ ] No API keys in code
* [ ] No database credentials
* [ ] No personal user data
* [ ] No Supabase service keys (only anon keys if needed)

**Branch Strategy (For Future)**

**For ongoing development:**

# Create feature branch

git checkout -b feature/new-questionnaire-logic

# Work on feature, commit changes

git add .

git commit -m "Implement new questionnaire logic"

# Push feature branch

git push origin feature/new-questionnaire-logic

# Create Pull Request on GitHub, then merge

**Backup Schedule Recommendation**

* **Daily:** Quick commits for any changes
* **Weekly:** Major feature commits with detailed messages
* **Monthly:** Review and clean up commit history
* **Before major changes:** Always commit current working state

**Next Steps**

1. **Set up GitHub Actions** for automated testing (future)
2. **Create separate repositories** for frontend/backend if they grow large
3. **Set up issue tracking** on GitHub for feature requests
4. **Consider GitHub Copilot** for AI coding assistance