🚀 GitHub Repository Setup Instructions

Complete Guide to Setting Up Your Public Repository with CI/CD

📋 Prerequisites

- GitHub account (create one at https://github.com)
- Git installed on your system
- Repository files ready (already prepared)
- Database credentials for production

1️⃣ Create GitHub Repository

Step 1: Create New Repository

1. **Go to GitHub**: https://github.com
2. **Click** “New” or “+” → “New repository”
3. **Fill in details**:
   - **Repository name**: `college-algebra-platform` (or your choice)
   - **Description**: “Interactive College Algebra Learning Platform for Business Students”
   - **Visibility**: ✅ **Public** (as requested)
   - **Initialize**: ❌ Do NOT initialize with README (we already have one)
4. **Click**: “Create repository”

Step 2: Note Your Repository URL

You’ll see something like:

```
https://github.com/YOUR_USERNAME/college-algebra-platform.git
```

Save this URL - you’ll need it shortly!

2️⃣ Configure Local Repository

Step 1: Set Up Git Remote

Run these commands in your terminal:

```
# Navigate to project
cd /home/ubuntu/college_algebra_website

# Check current git status
git status

# Add GitHub as remote origin (replace YOUR_USERNAME)
git remote add origin https://github.com/YOUR_USERNAME/college-algebra-platform.git

# Verify remote was added
git remote -v
```

Step 2: Prepare for Push

```
# Check which branch you're on
git branch

# If not on 'main', create and switch to it
git checkout -b main

# Stage all files
git add .

# Create initial commit
git commit -m "Initial commit: College Algebra Learning Platform v1.0"

# Push to GitHub
git push -u origin main
```

Step 3: Verify Files on GitHub

1. Go to your repository on GitHub
2. Verify you see all project files
3. Check that `.github/workflows/` directory is present
4. Verify README.md displays correctly

3️⃣ Configure GitHub Secrets (CI/CD Requirements)

Why Secrets?

Your CI/CD pipeline needs access to sensitive information (database URLs, API keys) without exposing them in your code.

Step 1: Navigate to Secrets

1. **Go to** your repository on GitHub
2. **Click** “Settings” tab
3. **Click** “Secrets and variables” → “Actions” (left sidebar)
4. **Click** “New repository secret”

Step 2: Add Required Secrets

Add each of these secrets one by one:

Secret 1: DATABASE_URL

```
Name: DATABASE_URL
Value: [Your production database URL]
Example: postgresql://user:password@host:5432/dbname
```

Secret 2: NEXTAUTH_SECRET

```
Name: NEXTAUTH_SECRET
Value: [Your NextAuth secret - generate one below]
```

Generate NEXTAUTH_SECRET:

```
openssl rand -base64 32
# Or
node -e "console.log(require('crypto').randomBytes(32).toString('base64'))"
```

Secret 3: NEXTAUTH_URL

```
Name: NEXTAUTH_URL
Value: [Your production URL]
Example: https://your-app-domain.com
```

Secret 4: NEXT_PUBLIC_APP_URL

```
Name: NEXT_PUBLIC_APP_URL
Value: [Same as NEXTAUTH_URL]
Example: https://your-app-domain.com
```

Secret 5: DEPLOYMENT_TOKEN (Optional)

```
Name: DEPLOYMENT_TOKEN
Value: [Your deployment platform token]
Note: Only needed if using automated deployment
```

Step 3: Verify Secrets

1. Go to “Settings” → “Secrets and variables” → “Actions”
2. You should see all 5 secrets listed
3. Secrets are encrypted and cannot be viewed once saved

4️⃣ Enable GitHub Actions

Step 1: Verify Actions Are Enabled

1. Go to your repository
2. Click “Actions” tab
3. If prompted, click “I understand my workflows, go ahead and enable them”

Step 2: Test CI/CD Pipeline

```
# Make a small change to test CI/CD
echo "# Test CI/CD" >> README.md

# Commit and push
git add README.md
git commit -m "test: Verify CI/CD pipeline"
git push origin main
```

Step 3: Monitor Workflow

1. Go to “Actions” tab on GitHub
2. You should see “CI/CD Pipeline” workflow running
3. Click on the workflow to see details
4. Wait for all jobs to complete (✅ green checkmarks)

Expected Jobs:
- ✅ Build & Test
- ✅ Security Scan
- ✅ Deploy to Production (if on main branch)

5️⃣ Set Up Branch Protection (Recommended)

Why Branch Protection?

Prevents direct pushes to main branch and requires code review.

Steps:

1. Go to “Settings” → “Branches”
2. Click “Add branch protection rule”
3. **Branch name pattern**: `main`
4. **Enable these options**:
   - ✅ Require a pull request before merging
   - ✅ Require approvals: 1
   - ✅ Require status checks to pass before merging
   - ✅ Require branches to be up to date before merging
   - Select: `build-and-test` and `security-scan`
   - ✅ Require conversation resolution before merging
   - ✅ Include administrators
5. Click “Create” or “Save changes”

6️⃣ Update README with Your Info

Edit README.md

```
# Open README.md in editor
nano app/README.md  # or use your preferred editor
```

Replace these placeholders:

1. `YOUR_USERNAME` → Your actual GitHub username (appears in 3 places)
2. Update badges URLs at the top
3. Add your actual support email
4. Update Discord link (if applicable)

Example changes:

```
# Before
![Build Status](https://i.ytimg.com/vi/GlqQGLz6hfs/sddefault.jpg)

# After
![Build Status](https://i.ytimg.com/vi/fx1Jttnj2vc/hq720.jpg?sqp=-oaymwEhCK4-
FEIIDSFryq4qpAxMIARUAAAAAGAElAADIQj0AgKJD&rs=AOn4CLC-daVhNFw6m7Zw_qYS7tRXKjU1KA)
```

Commit changes:

```
git add README.md
git commit -m "docs: Update README with repository information"
git push origin main
```

7️⃣ Development Workflow

For New Features

```
# 1. Create feature branch
git checkout -b feature/your-feature-name

# 2. Make changes to code
# ... edit files ...

# 3. Test locally
cd app
yarn dev

# 4. Commit changes
git add .
git commit -m "feat: add your feature description"

# 5. Push branch
git push origin feature/your-feature-name

# 6. Create Pull Request on GitHub
# Go to repository → "Pull requests" → "New pull request"
# Select your feature branch → Create PR

# 7. Wait for CI/CD checks to pass
# Review → Merge when ready
```

For Bug Fixes

```
# 1. Create fix branch
git checkout -b fix/bug-description

# 2. Fix the bug
# ... edit files ...

# 3. Test fix
cd app
yarn dev
# Test the specific bug fix

# 4. Commit and push
git add .
git commit -m "fix: resolve bug description"
git push origin fix/bug-description

# 5. Create PR and merge
```

8️⃣ CI/CD Pipeline Details

What Happens on Push?

1. **Build & Test Job**:
   - ✅ Installs dependencies
   - ✅ Generates Prisma Client

   - ✅ Runs TypeScript type checking
   - ✅ Runs ESLint
   - ✅ Builds production bundle
   - ✅ Uploads build artifacts

2. **Security Scan Job**:
   - ✅ Runs yarn audit
   - ✅ Checks for vulnerabilities
   - ⚠️ Warns about moderate/high issues

3. **Deploy Job** (only on main branch):
   - ✅ Triggers deployment
   - ✅ Reports status
   - ✅ Sends notifications

Build Status Badge

Add this to your README to show build status:

```
![Build Status](https://i.ytimg.com/vi/fx1Jttnj2vc/maxresdefault.jpg)
```

9️⃣ Troubleshooting

Issue: Workflow Fails on First Run

Cause: Missing secrets

Solution:
1. Check “Actions” tab → Click failed workflow
2. Look for error messages about missing secrets
3. Add missing secrets in “Settings” → “Secrets”
4. Re-run workflow: “Re-run all jobs”

Issue: Build Fails on TypeScript Errors

Cause: Type errors in code

Solution:

```
# Run locally to see errors
cd app
yarn tsc --noEmit

# Fix errors, then commit
git add .
git commit -m "fix: resolve TypeScript errors"
git push
```

Issue: Cannot Push to Main Branch

Cause: Branch protection enabled

Solution:
1. Create feature branch instead
2. Push to feature branch
3. Create Pull Request
4. Merge after approval

Issue: Secrets Not Working

Cause: Incorrect secret values or names

Solution:
1. Verify secret names match exactly (case-sensitive)
2. Delete and re-add secrets
3. Ensure no extra spaces in secret values
4. Re-run workflow

🔟 Next Steps

After Setup is Complete

1. ✅ **Verify CI/CD**: Make a test commit and ensure workflows pass
2. ✅ **Deploy**: Follow deployment instructions in README
3. ✅ **Monitor**: Check GitHub Actions regularly
4. ✅ **Document**: Update README with deployment URLs
5. ✅ **Share**: Share repository with team members

Invite Collaborators

1. Go to “Settings” → “Collaborators”
2. Click “Add people”
3. Enter GitHub username or email
4. Select permission level:
   - **Read**: View only
   - **Write**: Can push code
   - **Admin**: Full access

Set Up Notifications

1. Go to “Settings” → “Notifications”
2. Configure:
   - Email notifications for failed workflows
   - Slack/Discord webhooks (optional)
   - Issue/PR notifications

📞 Support

If you encounter issues:

1. **Check GitHub Actions logs**: Detailed error messages
2. **Review secrets**: Ensure all required secrets are set

3. **Test locally**: Run `yarn build` in app directory
4. **Check documentation**: https://docs.github.com/en/actions

---

✅ Verification Checklist

Before considering setup complete:

- [ ] Repository created and public
- [ ] All files pushed to GitHub
- [ ] `.github/workflows/` directory present
- [ ] All 5 secrets configured
- [ ] GitHub Actions enabled
- [ ] At least one successful workflow run
- [ ] README.md updated with your info
- [ ] Branch protection configured (optional)
- [ ] Build status badge working
- [ ] Collaborators invited (if applicable)

🎉 Congratulations!

Your repository is now set up with:

✅ Public visibility
✅ Automated CI/CD pipeline
✅ Security scanning
✅ Branch protection
✅ Professional documentation

You're ready to develop and deploy! 🚀

Document Version: 1.0
Last Updated: December 15, 2025
Next Review: After first deployment