

Git Merge Conflict Fix - Complete Analysis

Problem Summary

You were encountering a git merge conflict when running the update script:

```
error: The following untracked working tree files would be overwritten by merge:
    data/sports_bar.db
Please move or remove them before you merge.
Aborting
```

Why `git stash` didn't work: The `git stash` command only saves **tracked** files with uncommitted changes. The problematic files were **untracked** (not in git's index), so stashing had no effect on them.

Root Cause Analysis

1. Missing `.gitignore` Entries

The `.gitignore` file was missing several important patterns:

- `data/*.db` - Database files in the data directory
- `data/*.db-journal` - SQLite journal files
- `benchmark-reports/` - Benchmark report artifacts
- `next/` - Next.js build directory
- `*.tgz` - NPM package archives

Result: These files were being treated as untracked files instead of being ignored, causing conflicts when git tried to pull files with the same names from the repository.

2. No Conflict Prevention in Update Script

The `update_from_github.sh` script performed `git pull` without checking for untracked files that would conflict with incoming changes.

Result: When the repository contained files like `data/sports_bar.db` and you had a local untracked file with the same name, git would abort the merge to prevent data loss.

Solution Implemented

1. Updated `.gitignore` (Lines 60-101)

Added:

```
# Database
prisma/*.db
prisma/*.db-journal
data/*.db          # NEW: Ignore database files in data/
data/*.db-journal  # NEW: Ignore journal files in data/

# Build artifacts and temporary files
benchmark-reports/ # NEW: Benchmark reports
next/              # NEW: Next.js build artifacts
*.tgz              # NEW: NPM package files
*.tar.gz           # NEW: Compressed archives
!**/backups/*.tar.gz # EXCEPT: Keep backup archives
```

Why this works:

- Git now ignores these files completely
- They won't show up as untracked files
- No conflicts during `git pull`
- Your local files are preserved

2. Enhanced Update Script (Lines 981-1085)

Added three new sections:

A. Untracked File Conflict Detection (Lines 981-1024)

```
# Before git pull:
1. Fetch latest changes from origin/main
2. Compare incoming changes with local untracked files
3. Identify files that would conflict
```

Logic:

- Runs `git fetch origin main` to get latest changes
- Compares `git diff --name-only HEAD origin/main` with local untracked files
- Creates list of files that exist locally but aren't tracked by git

B. Automatic Backup (Lines 996-1024)

```
# If conflicts found:
1. Create timestamped backup directory
2. Move conflicting files to backup (preserving structure)
3. Log each backed up file
```

Backup location:

```
~/sports-bar-backups/untracked-conflicts-YYYYMMDD-HHMMSS/
```

Example:

```
~/sports-bar-backups/untracked-conflicts-20251007-193000/
├── data/
│   └── sports_bar.db
├── benchmark-reports/
│   └── report.md
└── next/
    └── build.js
```

C. Smart Restore (Lines 1059-1085)

```
# After successful git pull:
1. Check each backed up file
2. If file is now tracked by git → keep new version from repo
3. If file is still untracked → restore from backup
4. Keep backup directory for reference
```

Why this is smart:

- If the repo added `data/sports_bar.db` as a tracked file, it uses the repo version
- If `data/sports_bar.db` is still meant to be local-only, it restores your version
- You never lose data



Testing Performed

Created test scenario with conflicting files:

```
# Created test files
data/sports_bar.db
benchmark-reports/test-report.md
next/test.js
sports-bar-ai-assistant@0.1.0.tgz

# Verified .gitignore
✓ All files properly ignored by git
✓ git check-ignore confirms rules working
✓ git status shows clean working tree
```



How It Works - Step by Step

Before This Fix:

1. User **runs**: `./update_from_github.sh`
2. Script **runs**: `git pull origin main`
3. Git finds: **data**/sports_bar.db (untracked local file)
4. Git sees: **data**/sports_bar.db (incoming from repo)
5. Git aborts: **"untracked working tree files would be overwritten"**
6. User stuck: Can't update without manual **intervention**

After This Fix:

1. User **runs**: `./update_from_github.sh`
2. Script **runs**: `git fetch origin main`
3. Script checks: Compare incoming changes vs local untracked files
4. Script finds: **data/sports_bar.db** conflicts
5. Script backs up: Move **to** `/sports-bar-backups/untracked-conflicts-TIMESTAMP/`
6. Script **runs**: `git pull origin main` (succeeds!)
7. Script checks: Is **data/sports_bar.db** now tracked?
 - **If YES**: Keep repo version (it's now part of the project)
 - **If NO**: **Restore** from backup (it's still local-only)
8. User happy: Update completed automatically



Benefits

For Users:

- ✓ **No more manual conflict resolution** - Script handles it automatically
- ✓ **No data loss** - All conflicting files are backed up
- ✓ **Smart handling** - Distinguishes between repo files and local files
- ✓ **Clear logging** - Know exactly what happened and where files are
- ✓ **Safe updates** - Can run update script without fear

For Developers:

- ✓ **Better .gitignore** - Prevents future conflicts
- ✓ **Robust update process** - Handles edge cases
- ✓ **Maintainable code** - Clear, well-documented logic
- ✓ **Timestamped backups** - Easy to track and debug



Pull Request Created

PR #121: Fix: Handle untracked file conflicts in update script

URL: <https://github.com/dfultonthebar/Sports-Bar-TV-Controller/pull/121>

Status: ✓ Open and ready for review



Next Steps for You

1. Review the Pull Request

Visit: <https://github.com/dfultonthebar/Sports-Bar-TV-Controller/pull/121>

Review the changes to ensure they meet your requirements.

2. Complete Your Update (After PR is Merged)

Once the PR is merged to main, you can complete your update:

```
# Navigate to your project
cd ~/Sports-Bar-TV-Controller

# Fetch the latest changes
git fetch origin main

# Reset to the latest main (this will get the fix)
git reset --hard origin/main

# Now run the update script - it will handle conflicts automatically!
./update_from_github.sh
```

3. If You Have Untracked Files Now

If you currently have conflicting untracked files:

Option A: Let the new script handle it (RECOMMENDED)

```
# After PR is merged and you've pulled the fix:
./update_from_github.sh
# The script will automatically backup and handle conflicts
```

Option B: Manual cleanup (if you want to do it now)

```
# Backup your files manually
mkdir -p ~/manual-backup
cp data/sports_bar.db ~/manual-backup/ 2>/dev/null || true
cp -r benchmark-reports ~/manual-backup/ 2>/dev/null || true
cp -r next ~/manual-backup/ 2>/dev/null || true
cp *.tgz ~/manual-backup/ 2>/dev/null || true

# Remove the conflicting files
rm -f data/sports_bar.db
rm -rf benchmark-reports next
rm -f *.tgz

# Now pull the fix
git pull origin main

# Run the update script
./update_from_github.sh
```



Technical Details






Files Modified:

1. `.gitignore` - Added 7 new ignore patterns
2. `update_from_github.sh` - Added 104 lines of conflict handling logic

Key Functions Added:

- **Conflict Detection:** Compares incoming changes with local untracked files
- **Automatic Backup:** Creates timestamped backup directories
- **Smart Restore:** Intelligently restores non-conflicting files
- **Enhanced Logging:** Clear messages about what's happening

Safety Features:

-  Preserves directory structure in backups
-  Never overwrites existing backups (uses timestamps)
-  Keeps backup directory for user reference
-  Logs every action for transparency
-  Fails safely if backup fails



Additional Resources

Understanding Git Untracked Files:

- **Tracked files:** Files git knows about (in the index)
- **Untracked files:** Files git doesn't know about (not in index)
- **Ignored files:** Files git is told to ignore (.gitignore)

Why This Matters:

```
git stash      # Only saves tracked files with changes
git clean -fd  # Removes untracked files (dangerous!)
.gitignore     # Tells git to ignore certain files
```

Backup Locations:

```
~/sports-bar-backups/
├── config-backup-TIMESTAMP.tar.gz      # Regular config backups
├── database-backups/
│   ├── dev-db-TIMESTAMP.sql.gz        # Database SQL dumps
│   └── untracked-conflicts-TIMESTAMP/  # NEW: Conflict backups
│       └── [conflicting files preserved here]
```







Lessons Learned

Why This Happened:

1. **Database location changed:** The database moved from `prisma/dev.db` to `data/sports_bar.db`
2. **Build artifacts accumulated:** Benchmark reports and build files weren't ignored
3. **No conflict prevention:** Update script didn't anticipate this scenario

Prevention for Future:

1.  Comprehensive .gitignore patterns
2.  Proactive conflict detection in update script
3.  Automatic backup and restore logic
4.  Clear documentation and logging



Pro Tips

For Running Updates:

```
# Always check status before updating
git status

# Run update with AI checks (recommended)
./update_from_github.sh

# Run update without AI checks (faster)
./update_from_github.sh --skip-ai

# Run update with benchmark
./update_from_github.sh --benchmark
```

For Checking Backups:

```
# List all backups
ls -lh ~/sports-bar-backups/

# View latest conflict backup
ls -lh ~/sports-bar-backups/untracked-conflicts-*/

# Restore a specific file from backup
cp ~/sports-bar-backups/untracked-conflicts-TIMESTAMP/path/to/file ./path/to/file
```



Summary

Problem: Git merge conflicts from untracked files blocking updates

Root Cause: Missing .gitignore entries + no conflict handling in update script

Solution: Enhanced .gitignore + automatic conflict detection and backup

Result: Seamless updates with automatic conflict resolution

Status:  Fixed and ready for review in PR #121

Need Help?

- Review the PR: <https://github.com/dfultonthebar/Sports-Bar-TV-Controller/pull/121>
- Check backups: `ls ~/sports-bar-backups/`
- View logs: `cat ~/Sports-Bar-TV-Controller/update.log`

Important Note: Remember to give the GitHub App access to your private repositories if needed: [GitHub App Permissions](https://github.com/apps/abacusai/installations/select_target) (https://github.com/apps/abacusai/installations/select_target)