Installer Bug Analysis - Root Cause & Fix

Executive Summary

The clean installer (`install.sh`) stops prematurely after database setup and never completes the build, service setup, or firewall configuration steps. This forces users to run the `update_from_github.sh` script to get the application working, which defeats the purpose of a one-command installation.

Root Cause: The installer that the user ran (Version 2.0) is not the current `install.sh` in the main branch. It appears to be from a different branch or an older version that was downloaded via curl/wget.

Impact: Users cannot complete a clean installation without manual intervention.

Solution: Fix the current installer to be robust, complete all steps, and handle edge cases properly.

Investigation Timeline

1. Initial Observations

User's Experience:
1. User ran the clean installer (`install.sh`)
2. Installer stopped/failed at some point
3. User had to run `update_from_github.sh` to get the app working
4. Update script succeeded and got the app running

Current State:
- Installation directory: `/home/ubuntu/Sports-Bar-TV-Controller`
- Repository: `~/github_repos/Sports-Bar-TV-Controller`
- Branch: `main`
- App is running via `sh -c next start` (not PM2)
- PM2 is installed but not in PATH

2. Log Analysis

Installation Log: `/tmp/sportsbar-install-20251007-213521.log`

What Completed Successfully:
1. ✅ System check
2. ✅ System dependencies
3. ✅ Node.js installation
4. ✅ Ollama setup
5. ✅ Service user configuration
6. ✅ Repository cloning
7. ✅ NPM dependencies installation
8. ✅ Database schema initialization
9. ✅ Prisma Client generation

What Never Happened:
- ❌ Build application (Step 9/11)
- ❌ Setup systemd service (Step 10/11)
- ❌ Configure firewall (Step 11/11)

Last Log Entry:

```
✔ Generated Prisma Client (v6.17.0) to ./node_modules/@prisma/client in 569ms

Start by importing your Prisma Client (See: https://pris.ly/d/importing-client)
```

The log ends abruptly with no error message.

3. Key Findings

Finding #1: Different Installer Version

The log shows:

```
[2025-10-07 21:35:21] Installation started - Version 2.0
[2025-10-07 21:35:21] Installation state saved: system_check - completed
```

But the current `install.sh` on main branch:
- Does NOT have “Version 2.0” string
- Does NOT have “Installation state saved” functionality
- Has different structure and flow

Conclusion: The user ran a different installer (likely from a curl command that downloaded from a different branch or older version).

Finding #2: PM2 Not in PATH

PM2 Installation:
- PM2 is installed: `~/.npm-global/lib/node_modules/pm2@6.0.13`
- PM2 binary location: `~/.npm-global/bin/pm2`
- npm global prefix: `/home/ubuntu/.npm-global`

PATH Issue:
- `~/.npm-global/bin` is in `.bashrc` (4 duplicate entries!)
- But NOT in the system PATH for non-interactive shells
- The installer runs in non-interactive mode (via curl/wget)
- So PM2 is not accessible during installation

Current PATH:

```
/home/ubuntu/.local/bin:/opt/computersetup/.pyenv/shims:/opt/computersetup/.pyenv/
bin:/usr/local/nvm/versions/node/v22.14.0/bin:/usr/local/sbin:/usr/local/bin:/usr/
sbin:/usr/bin:/sbin:/bin
```

Notice: `~/.npm-global/bin` is NOT in this PATH.

Finding #3: Current Installer Limitations

Issues with Current `install.sh`:

1. **No PM2 Installation:** The installer doesn't install PM2 at all
2. **Interactive Prompts:** Multiple `prompt_yes_no` calls that block in non-interactive mode:
   - Line 187: "Continue anyway?" (low disk space)
   - Line 210: "Remove existing installation?"
   - Line 512: "Overwrite existing .env file?"
   - Line 618: "Set up systemd service?" (default: yes)
   - Line 660: "Start the service now?" (default: yes)
   - Line 688: "Configure firewall?" (default: yes)
3. **Systemd Service Only:** Uses systemd instead of PM2 for process management
4. **No Build Verification:** Doesn't verify the build completed successfully
5. **Silent Failures:** With `set -e`, any error causes immediate exit without proper logging

4. Why Update Script Succeeded

Key Differences in `update_from_github.sh`:

1. **Installs PM2:**
   ```
   bash
   install_pm2() {
       log "📦 Installing PM2 globally..."
       if sudo -n npm install -g pm2 2>/dev/null; then
           log_success "PM2 installed successfully"
       fi
   }
   ```
2. **Uses PM2 for Process Management:**
   ```
   bash
   pm2 start npm --name "sports-bar-tv-controller" -- start
   pm2 save
   ```
3. **No Interactive Prompts:** Runs completely automated
4. **Better Error Handling:** Doesn't use `set -e`, handles errors gracefully
5. **Comprehensive Logging:** Logs every step with timestamps
6. **Builds the Application:** Explicitly runs `npm run build`

Root Cause Summary

Primary Issue: Wrong Installer Version

The user ran a "Version 2.0" installer that:
- Is not the current `install.sh` on main branch
- Has different functionality and structure
- Stopped after database setup for unknown reasons
- Never completed the build and service setup

Secondary Issues with Current Installer

Even if the user had run the current `install.sh`, it would have issues:

1. **Missing PM2:** Doesn't install PM2, relies on systemd only
2. **Interactive Prompts:** Blocks in non-interactive curl/wget installations
3. **No PATH Configuration:** Doesn't ensure npm global bin is in PATH
4. **Incomplete Error Handling:** `set -e` causes silent exits on errors
5. **No Build Verification:** Doesn't verify build success before proceeding

---

Recommended Fixes

Fix #1: Make Installer Non-Interactive

Problem: Interactive prompts block curl/wget installations

Solution:
- Add `--non-interactive` flag support
- Default to sensible choices when running non-interactively
- Detect if running from pipe (curl/wget) and auto-enable non-interactive mode

```
# Detect if running from pipe
if [ -t 0 ]; then
    INTERACTIVE=true
else
    INTERACTIVE=false
fi

# Use in prompts
if [ "$INTERACTIVE" = true ]; then
    prompt_yes_no "Continue?" "y"
else
    # Auto-proceed with default
    return 0
fi
```

Fix #2: Install and Configure PM2

Problem: PM2 not installed, PATH not configured

Solution:
- Install PM2 globally during installation
- Configure npm global prefix properly
- Add npm global bin to PATH in profile
- Verify PM2 is accessible

```
install_pm2() {
    print_info "Installing PM2 process manager..."

    # Configure npm global prefix
    npm config set prefix "$HOME/.npm-global"

    # Install PM2
    npm install -g pm2

    # Add to PATH in .profile (for all shells)
    if ! grep -q ".npm-global/bin" "$HOME/.profile" 2>/dev/null; then
        echo 'export PATH="$HOME/.npm-global/bin:$PATH"' >> "$HOME/.profile"
    fi

    # Source for current session
    export PATH="$HOME/.npm-global/bin:$PATH"

    # Verify
    if command -v pm2 &> /dev/null; then
        print_success "PM2 installed: $(pm2 --version)"
    else
        print_error "PM2 installation failed"
        exit 1
    fi
}
```

Fix #3: Use PM2 Instead of Systemd

Problem: Systemd requires sudo and interactive prompts

Solution:
- Use PM2 as primary process manager
- Offer systemd as optional (for advanced users)
- PM2 works without sudo for user installations

```
setup_pm2_service() {
    print_info "Setting up PM2 process manager..."

    cd "$INSTALL_DIR"

    # Start with PM2
    pm2 start npm --name "sports-bar-tv-controller" -- start
    pm2 save

    # Setup PM2 startup (optional, requires sudo)
    if [ "$INTERACTIVE" = true ] && check_command sudo; then
        if prompt_yes_no "Configure PM2 to start on boot?" "y"; then
            pm2 startup
            # User will need to run the command shown
        fi
    fi

    print_success "Application started with PM2"
}
```

Fix #4: Better Error Handling

Problem: `set -e` causes silent exits

Solution:
- Keep `set -e` but add proper error traps
- Log errors before exiting
- Show helpful error messages

```
error_handler() {
    local line_number=$1
    local error_code=$2

    print_error "Installation failed at line $line_number (exit code: $error_code)"
    print_error "Check log file: $LOG_FILE"

    # Show last 20 lines of log
    if [ -f "$LOG_FILE" ]; then
        echo ""
        echo "Last 20 lines of log:"
        tail -n 20 "$LOG_FILE"
    fi

    exit 1
}

trap 'error_handler ${LINENO} $?' ERR
```

Fix #5: Build Verification

Problem: No verification that build succeeded

Solution:
- Check for `.next` directory
- Verify build artifacts exist
- Test that server can start

```
verify_build() {
    print_info "Verifying build..."

    if [ ! -d "$INSTALL_DIR/.next" ]; then
        print_error "Build directory not found"
        return 1
    fi

    if [ ! -f "$INSTALL_DIR/.next/BUILD_ID" ]; then
        print_error "Build ID not found"
        return 1
    fi

    print_success "Build verified successfully"
    return 0
}
```

Implementation Plan

Phase 1: Fix Current Installer (Immediate)

1. Add non-interactive mode detection

2. Install PM2 properly with PATH configuration
3. Use PM2 for process management
4. Improve error handling and logging
5. Add build verification

Phase 2: Testing (Before PR)

1. Test clean install on fresh system
2. Test with curl/wget (non-interactive)
3. Test with local script (interactive)
4. Verify PM2 starts on boot
5. Verify app is accessible

Phase 3: Documentation (With PR)

1. Update INSTALLATION.md
2. Add troubleshooting section
3. Document PM2 commands
4. Add FAQ for common issues

Testing Checklist

- [ ] Clean install on fresh Ubuntu system
- [ ] Install via curl (non-interactive)
- [ ] Install via local script (interactive)
- [ ] PM2 is in PATH after install
- [ ] App starts automatically
- [ ] App survives reboot
- [ ] Build completes successfully
- [ ] Database is created properly
- [ ] .env file is configured
- [ ] No interactive prompts in curl mode
- [ ] Error messages are helpful
- [ ] Log file is comprehensive

Conclusion

The installer bug is caused by:
1. User running a different installer version (Version 2.0)
2. Current installer lacking PM2 support
3. Interactive prompts blocking non-interactive installations
4. Poor error handling causing silent failures

The fix requires:
1. Making installer fully non-interactive
2. Installing and configuring PM2 properly
3. Using PM2 for process management

4. Improving error handling and logging
5. Adding build verification

Once fixed, users will be able to run a single curl command and have a fully working installation without any manual intervention.