AI Assistant Deployment Checklist

Date: January 26, 2025
Purpose: General deployment checklist for AI Assistant feature
Status: Production Ready

Pre-Deployment

1. Code Review

- [ ] All files reviewed and tested locally
- [ ] TypeScript compilation passes (`npm run build`)
- [ ] No console errors
- [ ] No linting errors
- [ ] All tests pass (if applicable)

2. Environment Variables

Required:

- [ ] `ANTHROPIC_API_KEY` - Claude API key set
- [ ] `NEXT_PUBLIC_SUPABASE_URL` - Supabase project URL
- [ ] `NEXT_PUBLIC_SUPABASE_ANON_KEY` - Supabase anon key

Optional:

- [ ] `SENTRY_DSN` - Error monitoring (recommended)
- [ ] `NEXT_PUBLIC_GA_MEASUREMENT_ID` - Analytics (if used)

Verification:

```
# Check environment variables are set
echo $ANTHROPIC_API_KEY
echo $NEXT_PUBLIC_SUPABASE_URL
```

3. Database Setup

- [ ] Migration file exists: `supabase/migrations/20250125_create_ai_assistant_tables.sql`
- [ ] Migration applied to Supabase
- [ ] Tables created: `ai_assistant_conversations` , `ai_assistant_messages`
- [ ] RLS policies enabled
- [ ] Indexes created

Verify in Supabase Dashboard:

- Tables visible in Table Editor
- RLS policies active
- Test query works

4. API Access

- [ ] Anthropic API key valid
- [ ] Claude 3 Haiku model accessible
- [ ] API credits available
- [ ] Rate limits understood (10 req/min)

Test API Access:

```
curl https://api.anthropic.com/v1/messages \
  -H "x-api-key: $ANTHROPIC_API_KEY" \
  -H "anthropic-version: 2023-06-01" \
  -H "content-type: application/json" \
  -d '{"model":"claude-3-haiku-20240307","max_tokens":10,"messages":[{"role":"user","content":"test"}]}'
```

Deployment Steps

Step 1: Local Testing

```
# Build project
npm run build

# Start dev server
npm run dev

# Test widget
# 1. Visit http://localhost:3000
# 2. Look for chat widget in bottom-right
# 3. Send test message
# 4. Verify response streams
# 5. Check Supabase for stored messages
```

Checklist:

- [ ] Widget appears on homepage
- [ ] Widget opens when clicked
- [ ] Message sends successfully
- [ ] Response streams in real-time
- [ ] Conversation stored in Supabase
- [ ] No console errors
- [ ] Keyboard navigation works

Step 2: Git Commit

```
# Review changes
git status
git diff

# Stage files
git add components/AIAssistantWidget.tsx
git add app/api/ai-assistant/route.ts
git add lib/claude.ts
git add lib/knowledge-base.ts
git add lib/rate-limit.ts
git add types/ai-assistant.ts
git add supabase/migrations/20250125_create_ai_assistant_tables.sql
git add docs/

# Commit
git commit -m "feat(ai-assistant): deploy AI chat widget
```

```
- Add floating chat widget component
- Implement Claude 3 Haiku API integration
- Add rate limiting (10 req/min)
- Store conversations in Supabase
- Add error monitoring with Sentry
- WCAG 2.1 AA accessible"
```

Step 3: Push to Repository

```
git push origin main
# or
git push origin [your-branch-name]
```

Step 4: Vercel Deployment

Automatic (if connected):

- [ ] Push triggers deployment
- [ ] Build completes successfully
- [ ] No build errors

Manual (if needed):

- [ ] Go to Vercel dashboard
- [ ] Trigger deployment
- [ ] Monitor build logs
- [ ] Verify deployment successful

Step 5: Environment Variables in Vercel

- [ ] `ANTHROPIC_API_KEY` set in Vercel
- [ ] `NEXT_PUBLIC_SUPABASE_URL` set in Vercel
- [ ] `NEXT_PUBLIC_SUPABASE_ANON_KEY` set in Vercel
- [ ] `SENTRY_DSN` set (if using Sentry)
- [ ] All variables match local `.env.local`

Verify:

- Go to Vercel Dashboard → Project → Settings → Environment Variables
- Check all required variables are set
- Verify values are correct

Post-Deployment Verification

1. Visual Inspection

- [ ] Visit live site: https://rockywebstudio.com.au
- [ ] Chat widget visible in bottom-right corner
- [ ] Widget opens when clicked
- [ ] UI looks correct (styling, layout)
- [ ] Responsive on mobile devices

2. Functional Testing

Basic Functionality:

- [ ] Send a test message
- [ ] Receive AI response
- [ ] Response streams in real-time
- [ ] Conversation history persists
- [ ] Widget can be minimized/closed
- [ ] Keyboard navigation works

Error Handling:

- [ ] Network error handled gracefully
- [ ] Rate limit message appears (if triggered)
- [ ] Invalid input rejected
- [ ] Error messages user-friendly

Accessibility:

- [ ] Keyboard navigation (Tab, Enter, Escape)
- [ ] Focus indicators visible
- [ ] Screen reader compatible
- [ ] WCAG 2.1 AA compliant

3. Database Verification

In Supabase Dashboard:

- [ ] New conversation created in `ai_assistant_conversations`
- [ ] Messages stored in `ai_assistant_messages`
- [ ] Timestamps correct
- [ ] Data structure matches schema

Query Test:

```
SELECT COUNT(*) FROM ai_assistant_conversations;
SELECT COUNT(*) FROM ai_assistant_messages;
```

4. Performance Check

- [ ] Page load time acceptable
- [ ] Widget doesn't block page rendering
- [ ] Response time reasonable (< 2 seconds)
- [ ] No performance regressions

Lighthouse Audit:

- [ ] Performance score maintained (> 90)
- [ ] Accessibility score maintained (> 90)
- [ ] No new issues introduced

5. Monitoring

Sentry (if configured):

- [ ] Errors tracked
- [ ] No critical errors

- [ ] Alerts configured (if needed)

Supabase:

- [ ] Database queries performant
- [ ] No connection errors
- [ ] Storage usage acceptable

Claude API:

- [ ] API calls successful
- [ ] Token usage tracked
- [ ] Cost within budget

Rollback Plan

If Issues Found

Option 1: Quick Fix

```
# Make targeted fix
git commit -m "fix(ai-assistant): [description]"
git push
```

Option 2: Revert Deployment

```
# Revert last commit
git revert HEAD
git push
```

Option 3: Disable Widget

```
// In app/layout.tsx, comment out:
// <AIAssistantWidget />
```

Option 4: Database Rollback

```
-- If needed, drop tables (CAUTION: loses data)
DROP TABLE IF EXISTS ai_assistant_messages;
DROP TABLE IF EXISTS ai_assistant_conversations;
```

Success Criteria

✅ **Deployment Successful**

- Widget appears on live site
- Messages send and receive correctly
- No console errors
- No build errors

✅ **Functionality Verified**

- Streaming responses work
- Conversation history persists

- Rate limiting active
- Error handling works

✅ **Performance Maintained**

- Page load time acceptable
- Lighthouse scores maintained
- No performance regressions

✅ **Monitoring Active**

- Errors tracked (Sentry)
- Database queries performant
- API calls successful

Post-Deployment Tasks

Immediate (Today)

- [ ] Monitor error logs
- [ ] Check user feedback
- [ ] Verify analytics (if configured)
- [ ] Document any issues

Week 1

- [ ] Review conversation quality
- [ ] Analyze user engagement
- [ ] Check API costs
- [ ] Optimize if needed

Ongoing

- [ ] Monitor error rates
- [ ] Review conversation logs
- [ ] Update knowledge base as needed
- [ ] Consider model upgrade (Sonnet) if needed

Troubleshooting

Widget Not Appearing

Check:

1. `AIAssistantWidget` imported in `app/layout.tsx`
2. Component renders without errors
3. No CSS hiding widget
4. JavaScript enabled in browser

Messages Not Sending

Check:

1. API route accessible: `/api/ai-assistant`
2. Environment variables set correctly
3. Claude API key valid
4. Rate limit not exceeded

No Responses

Check:

1. Claude API key valid
2. API credits available
3. Network connection stable
4. Error logs in Sentry

Database Issues

Check:

1. Supabase connection string correct
2. Tables exist
3. RLS policies allow service role
4. Migration applied

Notes

- **Deployment Time:** ~10-15 minutes
- **Verification Time:** ~30-60 minutes
- **Risk Level:** Low (feature addition, no breaking changes)
- **Rollback Time:** <5 minutes if needed

Created: January 26, 2025
Status: Ready for Deployment
Last Updated: January 26, 2025