Fiddyfiddy Documentation & Playbook

Version: 10.0
Last Updated: February 2026
Platform: Digital 50/50 Raffle System

Table of Contents

---

1. System Overview

What is Fiddyfiddy?

Fiddyfiddy is a digital 50/50 raffle platform that enables organizations to run fundraising raffles with Venmo payments. The system handles:

- Raffle creation and management
- Ticket sales via Venmo

- Hybrid payment verification (trust by default, verify winner)
- Random winner drawing
- Jackpot payout processing
- IRS compliance for winnings

Key Features

Feature	Description
Digital Tickets	QR code/link based ticket purchases
Venmo Payments	No transaction fees for players or organizers
Hybrid Verification	Trust by default, verify only the winner at draw time
Random Drawing	Cryptographically random winner selection
Redraw Support	Handle non-responsive winners
IRS Compliance	Jackpots capped at $600 to avoid W-2G requirements
Multi-Organizer	Each organizer manages their own raffles

Tech Stack

Component	Technology
Frontend	Next.js 14 (React)
Styling	Tailwind CSS
Backend	Next.js API Routes
Database	Knack
Authentication	JWT + bcrypt
Email	SendGrid
Payments	Venmo (manual)
Hosting	Vercel
Source Control	GitHub

2. Architecture

Directory Structure

```
fiddyfiddy/
├── app/                    # Next.js App Router
│   ├── api/                # API endpoints
│   │   ├── auth/           # Authentication (login, logout, register, hash)
│   │   ├── dashboard/      # Organizer dashboard data
│   │   ├── draw/           # Drawing operations
│   │   ├── raffles/        # Raffle CRUD + operations (including cancel)
│   │   ├── tickets/        # Ticket operations
│   │   └── admin/          # Admin operations (user management)
│   ├── about/              # About/info page
│   ├── admin/              # Admin pages
│   ├── dashboard/          # Organizer dashboard
│   ├── lobby/              # Public raffle listing
│   ├── login/              # Login page
│   ├── raffle/             # Raffle management pages
│   │   └── [id]/
│   │       ├── draw/       # Drawing interface
│   │       ├── edit/       # Edit raffle (includes cancel button)
│   │       ├── payout/     # Payout confirmation
│   │       ├── report/     # Raffle report
│   │       └── verify/     # Ticket verification
│   ├── r/                  # Player-facing raffle pages
│   │   └── [id]/
│   │       └── confirm/    # Payment confirmation
│   ├── register/           # Organizer registration (self-service)
│   └── ticket/             # Ticket lookup
├── lib/                    # Shared libraries
│   ├── auth.js             # JWT authentication
│   ├── drawing.js          # Drawing logic
│   ├── knack.js            # Knack API wrapper
│   ├── sendgrid.js         # Email templates (includes cancellation)
│   ├── utils.js            # Utility functions
│   └── venmo.js            # Venmo link generation
├── scripts/                # Utility scripts
│   └── create-user.js      # Manual user creation
└── public/                 # Static assets
```

Data Flow

```
[Player] → Lobby → Raffle Page → Venmo Payment → Confirm Page → Ticket Email
                                    ↓
[Organizer] → Dashboard → Verify Tickets → Draw Winner → Confirm Payout → Report
```

---

3. Environment Configuration

Required Variables

Create `.env.local` for local development or add to Vercel dashboard for production:

```
env

# Knack Database
KNACK_APP_ID=696fe0792dbca8488118f60c
KNACK_API_KEY=95d39c50-0c55-46b2-9b12-3407887c8b78

# SendGrid Email
SENDGRID_API_KEY=SG.xxxxxxxxxxxxx
SENDGRID_FROM_EMAIL=info@fiddyfiddy.org
SENDGRID_FROM_NAME=Fiddyfiddy

# Authentication
JWT_SECRET=YourSecureRandomString32CharsMin

# Site Configuration
NEXT_PUBLIC_SITE_URL=https://fiddyfiddy.vercel.app
NEXT_PUBLIC_SITE_NAME=Fiddyfiddy

# Platform Settings
OWNER_VENMO=@fiddyfiddy

# Optional: Google Analytics
NEXT_PUBLIC_GA_ID=G-XXXXXXXXXX
```

Variable Reference

Variable	Purpose	Where to Get
`KNACK_APP_ID`	Knack application identifier	Knack → Settings → API & Code

Variable	Purpose	Where to Get
`KNACK_API_KEY`	Knack API authentication	Knack → Settings → API & Code
`SENDGRID_API_KEY`	Email sending	SendGrid → Settings → API Keys
`SENDGRID_FROM_EMAIL`	Sender email address	Must be verified in SendGrid
`JWT_SECRET`	Token signing key	Generate: `openssl rand -base64 32`
`NEXT_PUBLIC_SITE_URL`	Base URL for links	Your Vercel/custom domain
`OWNER_VENMO`	Platform owner Venmo	Your Venmo handle
`OWNER_EMAIL`	Owner notification email	Your email (receives new organizer alerts)

4. Knack Database Schema

Objects Overview

Object	ID	Purpose
Settings	`object_4`	Platform-wide configuration
Users	`object_5`	Organizers and admins
Raffles	`object_6`	Raffle definitions
Tickets	`object_7`	Individual tickets
Draw Log	`object_8`	Drawing history
Transactions	`object_9`	Payment records

Settings Object (object_4)

Field	ID	Type	Description
owner_venmo	field_49	Text	Platform owner Venmo handle
owner_prime_default	field_50	Number	Default owner percentage
restricted_states	field_51	Text	States where raffles prohibited

Field	ID	Type	Description
refund_window_days	field_52	Number	Days allowed for refund
payout_deadline_hours	field_53	Number	Hours to claim prize
max_redraws	field_54	Number	Maximum redraw attempts
support_email	field_55	Email	Support contact
platform_name	field_56	Text	Platform display name

Users Object (object_5)

Field	ID	Type	Description
email	field_57	Email	User email (login)
password	field_58	Text	bcrypt hashed password
role	field_59	Text	Owner, Sponsor, Organizer
name	field_60	Text	Display name
venmo_handle	field_61	Text	Venmo username
phone	field_62	Phone	Contact number
status	field_63	Text	Active, Pending, Suspended

Raffles Object (object_6)

Field	ID	Type	Description
raffle_name	field_117	Text	Raffle display name
beneficiary_name	field_64	Text	Who receives funds
beneficiary_type	field_65	Text	Team, Event, Individual
beneficiary_venmo	field_118	Text	Beneficiary Venmo handle
ticket_price	field_66	Currency	Price per ticket
max_tickets	field_67	Number	Maximum tickets available

Field	ID	Type	Description
tickets_sold	field_68	Number	Current tickets sold
status	field_69	Text	Draft, Active, Drawing, Complete, Cancelled
draw_trigger	field_70	Text	Manual, Time, TicketCount
draw_time	field_71	DateTime	Scheduled draw time
draw_ticket_count	field_72	Number	Tickets to trigger auto-draw
is_public	field_73	Boolean	Show in public lobby
ticket_prefix	field_74	Text	Ticket number prefix (e.g., TIGERS)
organizer_venmo	field_75	Text	Organizer's Venmo
logo	field_76	Image	Raffle/team logo
state_restrictions	field_77	Text	Additional restricted states
owner_prime	field_78	Number	Owner percentage (default 11%)
min_tickets_enabled	field_79	Boolean	Require minimum tickets
min_tickets	field_80	Number	Minimum before draw
redraw_count	field_81	Number	Number of redraws done
drawn_at	field_82	DateTime	When drawing occurred
organizer	field_83	Connection	→ Users
winning_ticket	field_84	Connection	→ Tickets
payout_confirmed	field_85	Boolean	Winner paid
payout_confirmed_at	field_86	DateTime	When payout confirmed
jackpot_current	field_87	Equation	Calculated jackpot
tickets_remaining	field_88	Equation	Max - Sold
suggested_max	field_89	Equation	Max for $600 jackpot
redraws_remaining	field_90	Equation	Max redraws - used

Tickets Object (object_7)

Field	ID	Type	Description
raffle	field_91	Connection	→ Raffles
ticket_number	field_92	Text	Unique ticket ID (e.g., TIGERS-20260201-0001)
sequence_number	field_93	Number	Sequential number
player_email	field_94	Email	Buyer's email
player_venmo	field_95	Text	Buyer's Venmo handle
venmo_note	field_96	Text	Payment note/memo
venmo_txn_id	field_97	Text	Venmo transaction ID
status	field_98	Text	Pending, Verified, Rejected, Winner
payment_recipient	field_99	Text	Who received payment
created_at	field_100	DateTime	Purchase timestamp
verified_at	field_101	DateTime	Verification timestamp

Draw Log Object (object_8)

Field	ID	Type	Description
raffle	field_102	Connection	→ Raffles
ticket	field_103	Connection	→ Tickets
draw_number	field_104	Number	1, 2, 3... (redraw count)
result	field_105	Text	Selected, Confirmed, Redraw
reason	field_106	Text	Why redraw occurred
timestamp	field_107	DateTime	When draw happened

Transactions Object (object_9)

Field	ID	Type	Description
raffle	field_108	Connection	→ Raffles
ticket	field_109	Connection	→ Tickets
type	field_110	Text	Purchase, Payout, Refund
amount	field_111	Currency	Transaction amount
from_venmo	field_112	Text	Sender Venmo
to_venmo	field_113	Text	Recipient Venmo
status	field_114	Text	Pending, Confirmed, Failed
confirmed_at	field_115	DateTime	Confirmation timestamp
notes	field_116	Text	Additional details

5. User Roles & Permissions

Role Hierarchy

```
Owner
└── Sponsor
    └── Organizer
        └── Player (no account required)
```

Permissions Matrix

Action	Owner	Sponsor	Organizer	Player
View all raffles	✅	❌	❌	❌
Manage users	✅	❌	❌	❌
Approve organizers	✅	✅	❌	❌
Create raffles	✅	✅	✅	❌

Action	Owner	Sponsor	Organizer	Player
Manage own raffles	✅	✅	✅	❌
Verify tickets	✅	✅	✅	❌
Draw winners	✅	✅	✅*	❌
View lobby	✅	✅	✅	✅
Buy tickets	✅	✅	✅	✅

*Organizers with "Pending" status can create raffles but cannot draw winners until approved.

User Status Values

Status	Can Login	Can Create Raffles	Can Draw
Active	✅	✅	✅
Pending	✅	✅	❌
Suspended	❌	❌	❌

6. Raffle Lifecycle

Status Flow

Status Definitions

Status	Description	Allowed Actions
Draft	Created but not live	Edit, Activate, Delete
Active	Accepting ticket purchases	Verify tickets, Share, Draw, Cancel
Drawing	Winner selection in progress	Confirm winner, Redraw

Status	Description	Allowed Actions
Complete	Winner confirmed and paid	View report
Cancelled	Raffle cancelled by organizer	View only

Cancelling a Raffle

Organizers can cancel active raffles with player notification:

1. Navigate to Dashboard → Click "Details" on raffle
2. Scroll down → Click "❌ Cancel Raffle"
3. Enter cancellation reason (required)
4. Confirm cancellation

What happens:

- Raffle status changes to "Cancelled"
- All ticket holders receive email with:
  - Cancellation notice
  - Organizer's reason
  - Organizer's contact for refunds
- Raffle remains in system for audit trail

Creating a Raffle

1. Navigate to Dashboard → "New Raffle"
2. Enter raffle details:
   - **Raffle Name**: Display name (e.g., "Spring Fundraiser")
   - **Beneficiary Name**: Who receives funds
   - **Beneficiary Type**: Team, Event, or Individual
   - **Ticket Price**: $1-$50 recommended
   - **Max Tickets**: Auto-calculated to keep jackpot ≤ $600
   - **Ticket Prefix**: 1-10 characters (e.g., "TIGERS")
   - **Venmo Handle**: Where players send payment
3. Save as Draft
4. Activate when ready to sell

Activation Checklist

Before activating, verify:

- [ ] Raffle name is correct
- [ ] Ticket price is set
- [ ] Venmo handle is valid
- [ ] Max tickets calculated properly
- [ ] Beneficiary information complete

---

7. Ticket Lifecycle

Status Flow (Hybrid Verification)

```
Created → Verified (automatic) → (Winner)
            ↓
          Rejected (if organizer flags)
```

Note: As of v10, tickets are **auto-verified** on creation. This "trust by default" model reduces friction for players. Organizers verify payment only for the winning ticket at draw time.

Status Definitions

Status	Description
Verified	Default status - eligible for drawing (payment verified at draw time)
Rejected	Organizer flagged as invalid
Winner	Selected as winning ticket

Ticket Number Format

```
{PREFIX}-{YYYYMMDD}-{SEQUENCE}

Example: TIGERS-20260201-0023
         ^^^^^^ ^^^^^^^^ ^^^^
         Prefix  Date   Seq#
```

Purchase Flow (Player) - Simplified

1. Player visits raffle page (`/r/{id}`)
2. Enters email and Venmo handle
3. Clicks "Buy Ticket"
4. Ticket created and **auto-verified**
5. Redirected to Venmo to complete payment
6. Receives ticket confirmation email immediately

Note: Players no longer need to submit transaction IDs or screenshots. The system trusts that payment will be made.

Verification at Draw Time (Organizer)

1. Navigate to Drawing page
2. Draw winner
3. Check your Venmo for payment from winner's handle
4. If payment found → Confirm winner
5. If no payment → Redraw

Manual Rejection (Optional)

Organizers can still reject suspicious tickets:

1. Dashboard → Raffle → "Verify" page
2. Click on ticket row to expand
3. Click "Reject" if needed

---

8. Drawing Process

Pre-Draw Checklist

- [ ] All pending tickets verified or rejected
- [ ] Minimum ticket requirement met (if enabled)
- [ ] Organizer account is "Active" (not "Pending")

Drawing Steps

1. Navigate to Raffle → "Draw"

2. Review ticket summary
3. Click "Draw Winner"
4. System selects random verified ticket
5. Winner displayed with contact info
6. Options:
    - **Confirm Winner**: Mark as final winner
    - **Redraw**: Select new winner (if unresponsive)

Redraw Reasons

Reason	When to Use
No response	Winner doesn't respond within deadline
Invalid contact	Cannot reach winner
Declined prize	Winner refuses prize
Disqualified	Winner violates rules

Post-Draw

1. Winner notified via email
2. Organizer sends jackpot via Venmo
3. Organizer confirms payout in system
4. Raffle marked "Complete"
5. Report generated

---

9. Payment Flow

Ticket Purchase

```
Player → Venmo → Organizer/Beneficiary Venmo
```

Jackpot Payout

```
Organizer → Venmo → Winner
```

Venmo Link Generation

The system generates Venmo payment links with pre-filled:

- **Recipient**: Organizer or beneficiary Venmo handle
- **Amount**: Ticket price
- **Note**: Ticket number for reference

Example: `venmo://paycharge?txn=pay&recipients=@fiddyfiddy&amount=5.00&note=TIGERS-20260201-0023`

---

10. API Reference

Authentication Endpoints

Endpoint	Method	Description
`/api/auth/login`	POST	User login
`/api/auth/logout`	POST	User logout
`/api/auth/register`	POST	New organizer registration
`/api/auth/hash`	POST	Generate password hash (admin)

Raffle Endpoints

Endpoint	Method	Description
`/api/raffles`	GET	List raffles
`/api/raffles`	POST	Create raffle
`/api/raffles/[id]`	GET	Get raffle details
`/api/raffles/[id]`	PUT	Update raffle
`/api/raffles/[id]/activate`	POST	Activate raffle
`/api/raffles/[id]/pending-tickets`	GET	Get pending tickets
`/api/raffles/[id]/verify-tickets`	POST	Bulk verify tickets

Endpoint	Method	Description
/api/raffles/[id]/qr	GET	Get QR code
/api/raffles/[id]/report	GET	Get raffle report
/api/raffles/[id]/payout-info	GET	Get payout details
/api/raffles/[id]/confirm-payout	POST	Confirm payout sent
/api/raffles/[id]/cancel	POST	Cancel raffle (requires reason)

Ticket Endpoints

Endpoint	Method	Description
/api/tickets	POST	Create ticket
/api/tickets/[id]	GET	Get ticket details
/api/tickets/[id]/verify	POST	Verify/reject ticket
/api/tickets/by-number/[number]	GET	Lookup by ticket number

Draw Endpoints

Endpoint	Method	Description
/api/draw/[id]	POST	Execute drawing
/api/draw/[id]/status	GET	Get draw status
/api/draw/[id]/confirm	POST	Confirm winner
/api/draw/[id]/redraw	POST	Redraw winner

Admin Endpoints

Endpoint	Method	Description
/api/admin/users	GET	List all users
/api/admin/users	PUT	Update user status

11. IRS Compliance

Reporting Thresholds

Form	Threshold	Requirement
W-2G	$600+ AND 300x wager	Report to IRS, provide copy to winner
Form 945	Any withholding	Annual summary of non-payroll withholding
Schedule G	$15,000+ gaming income	Part of Form 990 for nonprofits

Fiddyfiddy Strategy

To avoid Form W-2G requirements:

- **Maximum jackpot**: $599.99
- **Auto-calculated max tickets**: Based on ticket price
- **Formula**: Max Tickets = floor($1200 / ticket_price)

Ticket Price to Max Tickets

Ticket Price	Max Tickets	Max Jackpot
$1	1,200	$600
$2	600	$600
$5	240	$600
$10	120	$600
$20	60	$600

Record Keeping Requirements

Maintain records of:

- All ticket purchases
- Winner information

- Payout confirmations
- Draw logs with timestamps

---

12. Deployment Guide

Initial Deployment

1. **Push to GitHub**

```bash
cd fiddyfiddy
git init
git add .
git commit -m "Initial commit"
git remote add origin https://github.com/username/fiddyfiddy.git
git branch -M main
git push -u origin main
```

2. **Connect to Vercel**
   - Go to vercel.com/new
   - Import GitHub repository
   - Set Root Directory: `fiddyfiddy` (if nested)
   - Add environment variables
   - Deploy
3. **Configure Environment Variables**
   - Add all variables from `.env.example`
   - Redeploy after adding variables
4. **Create First User**
   - POST to `/api/auth/hash` with password
   - Create user in Knack with hashed password
   - Set role to "Owner" and status to "Active"

Updating Deployment

```bash
```

```
# Make changes locally
git add .
git commit -m "Description of changes"
git push

# Vercel auto-deploys on push
```

Custom Domain

1. In Vercel: Settings → Domains
2. Add your domain
3. Configure DNS:
   - A record: `76.76.21.21`
   - CNAME for www: `cname.vercel-dns.com`
4. Update `NEXT_PUBLIC_SITE_URL`
5. Redeploy

---

13. Testing Checklist

Pre-Launch Testing

Authentication

- [ ] Login with valid credentials
- [ ] Login fails with wrong password
- [ ] Logout clears session
- [ ] Register new organizer (Pending status)
- [ ] Pending user can login but cannot draw

Raffle Management

- [ ] Create new raffle (Draft status)
- [ ] Edit raffle details
- [ ] Activate raffle (Draft → Active)
- [ ] Share link works
- [ ] QR code generates correctly

Ticket Purchase

- [ ] View raffle page (logged out)
- [ ] Venmo link generates correctly
- [ ] Ticket auto-verified immediately
- [ ] Receive confirmation email
- [ ] Urgency message appears when tickets low

Cancel Raffle

- [ ] Cancel button visible on active raffle edit page
- [ ] Cancel modal opens with reason field
- [ ] Cancellation requires reason
- [ ] Players receive cancellation email
- [ ] Raffle status changes to "Cancelled"

Verification (Optional)

- [ ] View all tickets on verify page
- [ ] Expand ticket row (drill-down)
- [ ] Reject suspicious ticket
- [ ] Rejected tickets ineligible for draw

Drawing

- [ ] Draw winner from verified tickets
- [ ] Winner email sent
- [ ] Check winner's Venmo payment in organizer's Venmo
- [ ] Redraw works if no payment found
- [ ] Confirm winner
- [ ] Payout confirmation

Reports

- [ ] Generate raffle report
- [ ] Export/view report data

New Organizer Flow

- [ ] Register at /register
- [ ] Auto-approved (Active status)
- [ ] Owner receives email notification
- [ ] Can login immediately
- [ ] Can create raffle immediately

Mobile Testing

- [ ] Lobby displays correctly
- [ ] Raffle page is touch-friendly
- [ ] Venmo deep link opens app
- [ ] Jackpot font size proportional
- [ ] Venmo @ symbol not overlapping

---

14. Troubleshooting

Debug Endpoint

Access `/api/debug?raffleId={id}` to see raw Knack data:

- Raw field values
- Mapped field values
- Useful for field mapping issues

Common Symptoms

"Login failed" but credentials are correct

1. Check user exists in Knack
2. Verify password is bcrypt hashed
3. Check user status is not "Suspended"
4. Verify `field_57` (email) format

Tickets not appearing in verification

1. Check ticket status is "Pending"
2. Verify raffle connection (field_91)
3. Check filters in `/api/raffles/[id]/pending-tickets`

Venmo links not working

1. Verify Venmo handle format (no @ in database)
2. Test on mobile (desktop may not have Venmo app)
3. Check URL encoding of special characters

Drawing fails

1. Verify organizer status is "Active" (not "Pending")
2. Check eligible tickets exist (Verified status)
3. Review error in browser console

Emails not sending

1. Verify SendGrid API key
2. Check sender email is verified in SendGrid
3. Review SendGrid activity log
4. Check spam folders

Log Locations

Environment	Where to Check
Local	Terminal running `npm run dev`
Vercel	Vercel Dashboard → Deployments → Logs
Knack	Knack Activity Log

---

15. Common Issues & Solutions

Issue: Email field returns HTML

Symptom: `player_email` shows `<a href="mailto:...">...</a>`

Solution: Already handled in code with `stripHtml()` function:

```
javascript

const stripHtml = (str) => str?.replace?.(/<[^>]*>/g, '') || str;
```

Issue: Currency fields return formatted strings

Symptom: `ticket_price` shows `"$5.00"` instead of `5`

Solution: Use `parseCurrency()` helper:

```javascript
function parseCurrency(value) {
  if (!value) return 0;
  if (typeof value === 'number') return value;
  const cleaned = String(value).replace(/[$,]/g, '');
  return parseFloat(cleaned) || 0;
}
```

Issue: Connection fields return nested objects

Symptom: `organizer` shows `[{id: "xxx", identifier: "..."}]`

Solution: Extract ID from raw field:

```javascript
organizer: record.field_83_raw?.[0]?.id || record.field_83
```

Issue: Boolean fields inconsistent

Symptom: `is_public` sometimes `true`, sometimes `"Yes"`

Solution: Handle all cases:

```javascript
is_public: record.field_73 === true ||
           record.field_73 === 'Yes' ||
           record.field_73 === 'yes'
```

Issue: Vercel deployment fails

Symptom: Build error on Vercel

Solutions:

1. Check Root Directory setting
2. Verify all environment variables set
3. Review build logs for specific error
4. Test local build: `npm run build`

Issue: "Cannot read property of undefined"

Symptom: API returns 500 error

Solutions:

1. Check Knack API credentials
2. Verify object/field IDs match schema
3. Add null checks in field mapping
4. Review Knack record exists

---

16. UX & Engagement Features

Urgency Messaging

The player raffle page displays dynamic urgency messages based on ticket availability:

Tickets Remaining	Message
≤ 20	🔥 "Only X tickets left!" (amber, pulsing)
21-50	⚡ "Selling fast — X remaining" (cyan)
> 50	No message

Share Prompts

Footer includes sharing options to drive virality:

- "Know someone who'd want to play?"
- 💬 "Text a Friend" button (opens SMS with link)
- 📎 "Copy Link" button

Self-Service Organizer Registration

- Public registration at `/register`
- Organizers auto-approved with "Active" status
- Platform owner receives email notification on each signup
- New organizers can immediately create and run raffles

About Page

Public information page at `/about` includes:

- What is Fiddyfiddy explanation
- How it works (3-step process)
- Benefits for organizers
- FAQ section
- Contact information
- CTA buttons for registration and browsing raffles

Future Enhancement Ideas

Enhancement	Benefit
Countdown timer	Creates urgency for timed draws
Recent activity feed	"John just bought 3 tickets" — social proof
Share-to-unlock	"Share to get 1 bonus entry" — virality
Lucky number picker	Let players pick their number — ownership feeling
Winner showcase	Past winners build trust and excitement
Progress milestones	"50% to goal! 🎉" — gamification
Push notifications	"Drawing in 1 hour!" — re-engagement
Referral tracking	Track who shared, reward top sharers

Appendix A: Quick Reference

URLs

Environment	URL
Production	https://fiddyfiddy.vercel.app
Knack Builder	https://builder.knack.com
SendGrid	https://app.sendgrid.com
Vercel	https://vercel.com/dashboard

Environment	URL
GitHub	https://github.com/mjmasone/fiddyfiddy

Key Commands

```bash
# Local development
npm run dev

# Build for production
npm run build

# Start production server
npm start

# Git: Push updates
git add .
git commit -m "message"
git push
```

Support Contacts

- **Technical Issues**: Review this documentation
- **Knack Support**: support@knack.com
- **SendGrid Support**: support@sendgrid.com
- **Vercel Support**: support@vercel.com

Appendix B: Version History

Version	Date	Changes
10.0	Feb 2026	**Major Update:** Hybrid verification (auto-verify tickets), cancel raffle with player notifications, about page, self-service organizer signups with owner email alerts, UI fixes (jackpot font, Venmo @ overlap), urgency messaging, share prompts
9.0	Feb 2026	Added verify table drill-down

Version	Date	Changes
8.0	Jan 2026	Production deployment
7.0	Jan 2026	Field mapping fixes
6.0	Jan 2026	Auto-approve organizers
5.0	Jan 2026	Dashboard + sharing
4.0	Jan 2026	Drawing + redraws
3.0	Jan 2026	Ticket verification
2.0	Jan 2026	Full Next.js rewrite
1.0	Dec 2025	Initial prototype

This documentation is maintained alongside the Fiddyfiddy codebase. For the latest version, check the repository.