SCOUTPULSE MASTER SPECIFICATION

Complete Product Vision & Technical Specification

Version: 2.0

Last Updated: December 8, 2025

Status: Production Development

EXECUTIVE SUMMARY

ScoutPulse is a premium baseball recruiting platform connecting high school players, college coaches, JUCO programs, and high school/showcase coaches. Built with Next.js 14, TypeScript, Supabase, and featuring a distinctive glassmorphism design with fluid animations.

Core Value Proposition:

• Players: Showcase talent, connect with college programs, track recruiting journey

• College Coaches: Discover talent, manage recruiting pipeline, streamline process

• HS/Showcase Coaches: Promote players, connect them with opportunities

• JUCO Coaches: Manage transfer portal, recruit community college talent

1. PRODUCT ARCHITECTURE

Tech Stack

Frontend:

• Next.js 14 (App Router)

• TypeScript

• Tailwind CSS

• Framer Motion (animations)

• React Spring (physics-based animations)

Backend:

• Supabase (PostgreSQL database)

• Supabase Auth (authentication)

• Supabase Storage (media files)

• Supabase Edge Functions (serverless)

APIs & Services:

• Google Places API (location/map data)

• Google Cloud Vision (video/image analysis)

• Stripe (payments)

• SendGrid (emails)

Deployment:

• Vercel (hosting)

• Supabase Cloud (database)

2. USER ROLES & PERMISSIONS

Player

Access:

• Personal dashboard

• Profile management (public & private)

• Video uploads

• Stats tracking

• Team Hub

• College Journey timeline

• Messaging with coaches

• Notification center

College Coach

Access:

• Discovery dashboard with map

• Watchlist/Pipeline management

• Recruiting Planner (diamond visualization)

• Calendar & scheduling

• Player profiles (view)

• Messaging system

• Program page management

• Analytics dashboard

High School / Showcase Coach

Access:

• Roster management

• Player tracking

• Team performance

• College coach engagement

• Event scheduling

JUCO Coach

Access:

- Transfer portal
- Player database
- Recruiting tools
- Analytics

3. DESIGN SYSTEM

Visual Identity

Glassmorphism Core

Every UI element uses glass effect:

```
/* Standard Glass Card */
backdrop-filter: blur(24px);
background: rgba(255, 255, 255, 0.05-0.15);
border: 1px solid rgba(255, 255, 255, 0.15);
box-shadow: 0 8px 32px rgba(0, 0, 0, 0.1);
border-radius: 16px;
```

Color Palette

Primary: Dark green gradients (#0a4d3c → #1a5f4a)

Accent: Bright green (#10b981)

Glass overlays: White with 5-15% opacity

Text: White primary, white/70 secondary

Borders: White/15 opacity

Typography

Font: Inter (system fallback)

Hierarchy:

• Hero: 48px bold

• H1: 32px bold

• H2: 24px semibold

• H3: 20px semibold

• Body: 16px regular

• Small: 14px regular

• Tiny: 12px regular

Spacing Scale

4, 8, 12, 16, 24, 32, 48, 64, 96px

4. ANIMATION PRINCIPLES

Core Animation Rules

NO SPINNERS - Use shimmer loading skeletons only

Spring Physics Everywhere:

```
transition: {
type: "spring",
stiffness: 300,
damping: 30
}
```

Animation Categories

1. Micro-interactions

• Hover: lift 4px + shadow increase (300ms)

• Click: scale 0.98 (100ms)

• Focus: glass ring (200ms)

• Ripple on buttons (500ms)

2. Number Animations

• Count up on first display

• Smooth easing

• Sparkles on trending metrics

3. Page Transitions

• Fade + slide (400ms)

• Stagger children (50ms delay)

• Exit before enter

4. Loading States

• Shimmer skeletons (NOT spinners)

• Pulse effect

• Wave animation

5. Button States

• Idle: glass effect

• Hover: lift + glow

• Active: scale down

• Disabled: opacity 50%

6. Drag & Drop

• Physics-based trails

• Ghost preview
• Snap to grid
• Success celebration

7. Map Interactions

• Zoom with momentum
• Pin bounce on add
• Cluster expand

8. Diamond Planner

• Drag players between bases
• Trail effects
• Position snapping
• Celebration on commit

9. Feed Updates

• Slide in from top
• Fade in list items
• Stagger by 50ms

10. Modals

• Backdrop blur in (200ms)
• Scale + fade content (300ms)
• Slide out (200ms)

5. FEATURE SPECIFICATIONS

5.1 PLAYER FEATURES

Dashboard

Purpose: Central hub for player's recruiting journey

Components:

• Hero card (glass, stats, avatar)

• Quick stats (at-bats, hits, HR, average)

• Recent activity feed

• Upcoming events calendar

• Coach interest indicators

• Profile completion meter

Design:

• Full glassmorphism

• Grid layout

• Responsive (mobile-first)

• Smooth transitions

Profile Page

Public View:

• Hero section (photo, name, position, grad year, location)

• Video highlights (grid)

• Stats table (by season)

• Biography

• Contact/recruit button

Edit View:

• Inline editing

• Upload videos (drag & drop)

• Stats management

• Privacy controls

• Preview toggle

Technical:

• Video transcoding (Supabase)

• Image optimization (Next.js)
• SEO metadata
• Social sharing

Team Hub

Purpose: Connect with teammates, track team stats

Features:

• Roster grid (with avatars)
• Team stats dashboard
• Upcoming games
• Team chat
• Shared media

College Journey

Purpose: Track recruiting timeline

Visualization:

• Vertical timeline
• Milestones (camp, visit, offer, commit)
• Date stamps
• Photos/notes per event
• Progress percentage

Video Management

Upload:

• Drag & drop
• Multiple files
• Progress indicators
• Auto-tagging (AI)

Organization:

• Folders (by season, event)
• Tags (pitch type, result)
• Search/filter
• Share links

Player:

• Embedded player
• Quality selection
• Download option
• Stats overlay

5.2 COLLEGE COACH FEATURES

Discovery Dashboard

Purpose: Find and evaluate talent

Layout:

• Map view (Google Maps)

• Player pins (clustered)

• Filters sidebar (position, grad year, location, stats)

• List view toggle

• Search bar

Map Features:

• Click pin → player card preview

• Radius search

• Heat map visualization

• Save searches

Filters:

• Position

• Graduation year

• Location (state, radius)

• Stats thresholds

• Video availability

• GPA range

Watchlist / Pipeline

Purpose: Manage recruiting prospects

Organization:

• Stages: Identified → Evaluating → Reached Out → Interested → Offered → Committed

• Drag & drop between stages

• Bulk actions

• Notes per player

• Reminders/follow-ups

Views:

• Kanban board

• Table view

• Calendar view

Features:

• Player cards (compact)

• Quick actions (message, schedule)

• Filters

• Export to CSV

Recruiting Planner (Diamond View)

Purpose: Visual pipeline management

Visualization:

• Baseball diamond layout

• Home: New prospects

• 1st Base: Initial contact

• 2nd Base: Evaluating

• 3rd Base: Strong interest

• Home Plate: Committed

Interactions:

• Drag players between bases

• Physics animations

• Confetti on commit

• Player count per base

• Quick view on hover

Calendar

Purpose: Schedule recruiting activities

Features:

• Month/week/day views

• Events: Visits, camps, games, calls

• Player associations

• Reminders

• Integration with player availability

• Export to Google Calendar

Event Types:

• Campus visit

• Game attendance

• Phone call

• Video review

• Camp/showcase

Player Profiles (Coach View)

Enhanced Data:

• Full stats history
• Video library
• Academic info
• Coach contacts
• Notes (private)
• Interest level
• Communication log

Actions:

• Message player
• Schedule visit
• Add to watchlist
• Move pipeline stage
• Share with staff
• Download recruiting package

Analytics Dashboard

Metrics:

• Pipeline conversion rates
• Geographic recruiting map
• Position breakdown
• Class composition
• Time to commit
• Source effectiveness

Visualizations:

• Charts (line, bar, pie)
• Heat maps
• Trend analysis
• Filters (date range, position)

5.3 HIGH SCHOOL / SHOWCASE COACH FEATURES

Roster Management

Purpose: Manage player roster

Features:

• Player list (grid/table)

• Add/remove players

• Import from CSV

• Player profiles (linked)

• Stats tracking

• Contact info

Actions:

• Bulk invite to platform

• Share roster with college coaches

• Generate recruiting packets

• Track player progress

Team Performance

Dashboard:

• Team stats

• Win/loss record

• Player highlights

• Upcoming schedule

College Coach Connections

Features:

• Message college coaches

• Share player profiles

• Track interest

• Schedule visits/camps

5.4 JUCO FEATURES

Transfer Portal

Purpose: Manage JUCO transfer recruiting

Features:

• Player database (JUCO players)

• Search/filters

• Player profiles

• Contact tools

• Pipeline management (similar to 4-year coaches)

6. SMART FEATURES (AI/ML)

Predictive Search

- Auto-suggest players as typing
- Smart filters based on previous searches
- "Players like this" recommendations

Filter Memory

- Remember coach preferences
- Auto-apply frequent filters
- Suggested searches

Smart Notifications

- Relevant only (no spam)
- Grouped by priority
- Intelligent timing
- Digest mode

Player Recommendations

- "You might like" based on watchlist
- Similar player suggestions
- Geographic recommendations

Comparison Tool

- Side-by-side player comparison
- Stat overlays
- Video sync playback

Notes AI

- Auto-summarize meeting notes
- Extract action items

• Suggest follow-ups

Calendar Intelligence

• Smart scheduling suggestions
• Conflict detection
• Travel time optimization
• Best time to reach out

Pipeline Automation

• Auto-move players based on actions
• Stale prospect alerts
• Follow-up reminders
• Conversion predictions

Dashboard Insights

• Weekly recruiting summary
• Trend analysis
• Performance predictions
• Action recommendations

Video Intelligence

• Auto-tagging (pitch type, result)
• Play detection
• Quality scoring
• Highlight generation

Advanced Search

• Natural language queries
• Complex filter combinations
• Saved searches with alerts

Collaboration

• Staff notes/comments
• Activity feed per player
• Assignment tracking

Smart Exports

- Custom report generation
- Recruiting package PDFs
- Data exports

7. INTERACTION PATTERNS

Progressive Disclosure

- Show essential info first
- Expand for details
- Minimize cognitive load

Optimistic UI

- Instant feedback
- Background sync
- Graceful error handling

Contextual Help

- Tooltips on hover
- Inline guidance
- Empty state instructions

Undo Everything

- Accidental delete protection
- Toast with undo button
- 5-second window

Multi-Select

- Bulk actions
- Visual feedback
- Action bar appears

Drag Everywhere

- Intuitive interactions
- Visual previews
- Snap to grid/zones

Inline Editing

- Click to edit
- Save on blur
- Validation feedback

Smart Defaults

- Pre-fill forms
- Remember preferences
- Contextual suggestions

Mobile-First

- Touch-friendly (48px targets)
- Swipe gestures
- Bottom navigation
- Pull to refresh

Delight Moments

- Confetti on commit
- Sparkles on achievements
- Smooth transitions
- Easter eggs

8. PERFORMANCE REQUIREMENTS

Perceived Performance

- Optimistic UI updates
- Skeleton loading (no spinners)
- Prefetch on hover
- Image placeholders

Real Performance

- < 2s page load (LCP)
- < 3s time to interactive (TTI)
- 60fps animations
- < 100ms input latency

Optimization Strategies

- Code splitting by route
- Lazy load components
- Image optimization (WebP, responsive)
- Virtual scrolling (long lists)
- Service worker caching
- CDN for static assets

Metrics

- Lighthouse score > 90
- Core Web Vitals green
- Bundle size < 200KB (initial)

9. ACCESSIBILITY (WCAG AA)

Keyboard Navigation

- Tab order logical
- Skip links
- Focus indicators visible
- Keyboard shortcuts (Cmd+K)

Screen Readers

- Semantic HTML
- ARIA labels
- Alt text
- Live regions for updates

Visual

- Color contrast 7:1 minimum (AAA)
- No info by color alone
- Focus indicators high contrast
- Resizable text

Motion

- Respects prefers-reduced-motion
- Disable animations option
- No auto-play videos

Forms

- Labels associated
- Error messages clear
- Validation feedback
- Help text available

10. MOBILE RESPONSIVENESS

Breakpoints

- Mobile: 375px - 767px
- Tablet: 768px - 1023px
- Desktop: 1024px+

Mobile Adaptations

- Bottom navigation
- Hamburger menu
- Touch targets 48px+
- Swipe gestures
- Pull to refresh
- Bottom sheets (instead of modals)

Touch Interactions

- Tap for select
- Long press for options
- Swipe to delete
- Pinch to zoom (maps)
- Drag to reorder

11. TECHNICAL IMPLEMENTATION

File Structure

```
scoutpulse/
■■■ app/
■ ■■■ (auth)/
■ ■ ■■■ login/
■ ■ ■■■ signup/
■ ■■■ (dashboard)/
■ ■ ■■■ player/
■ ■ ■ ■■■ dashboard/
■ ■ ■ ■■■ profile/
■ ■ ■ ■■■ team-hub/
■ ■ ■ ■■■ journey/
■ ■ ■■■ coach/
■ ■ ■ ■■■ dashboard/
■ ■ ■ ■■■ discover/
■ ■ ■ ■■■ watchlist/
■ ■ ■ ■■■ planner/
■ ■ ■ ■■■ calendar/
■ ■ ■■■ hs-coach/
■ ■ ■ ■■■ dashboard/
■ ■ ■ ■■■ roster/
■ ■ ■■■ juco/
■ ■ ■■■ dashboard/
■ ■ ■■■ portal/
■ ■■■ api/
■■■ components/
■ ■■■ ui/
■ ■ ■■■ GlassCard.tsx
■ ■ ■■■ GlassButton.tsx
■ ■ ■■■ GlassInput.tsx
■ ■ ■■■ LoadingSkeleton.tsx
■ ■ ■■■ Modal.tsx
■ ■■■ player/
■ ■■■ coach/
■ ■■■ shared/
■■■ lib/
■ ■■■ supabase/
■ ■■■ animations/
■ ■■■ utils/
■ ■■■ hooks/
■■■ public/
■■■ styles/
```

Database Schema

Users Table

```
users (
id uuid PRIMARY KEY,
role enum ('player', 'coach', 'hs_coach', 'juco'),
email text UNIQUE,
name text,
avatar_url text,
created_at timestamp,
updated_at timestamp
)
```

Players Table

```
players (
id uuid PRIMARY KEY,
user_id uuid REFERENCES users(id),
position text,
grad_year integer,
location text,
height integer,
weight integer,
bats text,
throws text,
gpa decimal,
bio text,
stats jsonb,
videos jsonb,
visibility text
)
```

Coaches Table

```
coaches (
id uuid PRIMARY KEY,
user_id uuid REFERENCES users(id),
school text,
conference text,
division text,
program_info jsonb
)
```

Watchlist Table

```
watchlist (
id uuid PRIMARY KEY,
coach_id uuid REFERENCES coaches(id),
player_id uuid REFERENCES players(id),
stage text,
notes text,
priority integer,
created_at timestamp,
updated_at timestamp
)
```

Events Table

```
events (
id uuid PRIMARY KEY,
user_id uuid REFERENCES users(id),
title text,
type text,
start_time timestamp,
end_time timestamp,
location text,
player_ids uuid[],
notes text
)
```

Messages Table

```
messages (
id uuid PRIMARY KEY,
sender_id uuid REFERENCES users(id),
recipient_id uuid REFERENCES users(id),
content text,
read boolean,
created_at timestamp
)
```

12. API ENDPOINTS

Authentication

- POST /api/auth/signup
- POST /api/auth/login
- POST /api/auth/logout
- GET /api/auth/session

Players

- GET /api/players (search/filter)
- GET /api/players/:id
- PUT /api/players/:id
- POST /api/players/:id/videos
- GET /api/players/:id/stats

Coaches

- GET /api/coaches/:id/watchlist
- POST /api/coaches/:id/watchlist
- PUT /api/coaches/:id/watchlist/:playerId
- DELETE /api/coaches/:id/watchlist/:playerId

Events

- GET /api/events
- POST /api/events
- PUT /api/events/:id
- DELETE /api/events/:id

Messages

- GET /api/messages
- POST /api/messages
- PUT /api/messages/:id/read

Search

- GET /api/search/players
- GET /api/search/coaches
- POST /api/search/save

13. SECURITY & PRIVACY

Authentication

- Supabase Auth
- Row Level Security (RLS)
- JWT tokens
- Secure cookie sessions

Data Privacy

- Player profile visibility controls
- Private notes (coach only)
- Encrypted messages
- GDPR compliant
- Data export/delete

Permissions

- Role-based access control
- Resource ownership validation
- API rate limiting
- CORS configuration

14. DEPLOYMENT & MONITORING

Hosting

- Vercel (frontend)
- Supabase Cloud (backend)
- Cloudflare CDN

CI/CD

- GitHub Actions
- Automated testing
- Preview deployments
- Production deployments

Monitoring

- Vercel Analytics
- Supabase Logs
- Error tracking (Sentry)
- Performance monitoring

Backups

- Daily database backups
- Media storage redundancy
- Disaster recovery plan

15. ROADMAP

Phase 1: MVP (Current)

- ■ User authentication
- ■ Player profiles
- ■ Coach discovery
- ■ Basic messaging
- ■ Glassmorphism design
- ■ Core animations

Phase 2: Enhanced Features

- Advanced search
- Video intelligence
- Calendar integration
- Analytics dashboard
- Mobile apps

Phase 3: AI Features

- Predictive analytics
- Smart recommendations
- Auto-tagging
- Natural language search

Phase 4: Marketplace

- Camp registration
- Showcases
- Training programs
- Equipment

Phase 5: Social Features

- Player network
- Team pages

- Achievement badges
- Leaderboards

16. SUCCESS METRICS

User Engagement

- Daily active users
- Session duration
- Feature adoption
- Retention rate

Recruiting Effectiveness

- Player-coach connections
- Messages sent
- Commits tracked
- Time to commit

Performance

- Page load times
- Error rates
- Uptime %
- API response times

Business

- User growth rate
- Premium conversions
- Revenue per user
- Customer satisfaction (NPS)

17. SUPPORT & DOCUMENTATION

User Documentation

- Getting started guides
- Feature tutorials
- Video walkthroughs
- FAQ

Developer Documentation

- API reference
- Component library
- Design system
- Deployment guide

Support Channels

- Email support
- In-app chat
- Knowledge base
- Community forum

APPENDIX A: GLASSMORPHISM CODE EXAMPLES

GlassCard Component

```
import { motion } from 'framer-motion';

export const GlassCard = ({ children, className = '' }) => (
<motion.div
className={`
backdrop-blur-2xl bg-white/10
border border-white/15
rounded-2xl shadow-xl
hover:shadow-2xl hover:-translate-y-1
transition-all duration-300
${className}
`}
initial={{ opacity: 0, y: 20 }}
animate={{ opacity: 1, y: 0 }}
transition={{ type: "spring", stiffness: 300, damping: 30 }}
>
{children}
</motion.div>
);
```

GlassButton Component

```
export const GlassButton = ({ children, onClick, variant = 'primary' }) => (
<motion.button
className={`
backdrop-blur-xl px-6 py-3 rounded-xl
border border-white/20
font-semibold transition-all
${variant === 'primary'
? 'bg-emerald-500/20 hover:bg-emerald-500/30'
: 'bg-white/5 hover:bg-white/10'}
`}
whileHover={{ scale: 1.02, y: -2 }}
whileTap={{ scale: 0.98 }}
onClick={onClick}
>
{children}
</motion.button>
);
```

APPENDIX B: ANIMATION CODE EXAMPLES

Number Count Up

```
import { useSpring, animated } from 'react-spring';

export const CountUpNumber = ({ value }) => {
const { number } = useSpring({
from: { number: 0 },
number: value,
delay: 200,
config: { mass: 1, tension: 20, friction: 10 },
});

return <animated.span>{number.to(n => n.toFixed(0))}</animated.span>;
};
```

Page Transition

```
import { motion } from 'framer-motion';

export const PageTransition = ({ children }) => (
<motion.div
initial={{ opacity: 0, y: 20 }}
animate={{ opacity: 1, y: 0 }}
exit={{ opacity: 0, y: -20 }}
transition={{ duration: 0.4 }}
>
{children}
</motion.div>
);
```

Loading Skeleton

```
export const LoadingSkeleton = () => (
<div className="animate-pulse">
<div className="h-4 bg-white/10 rounded w-3/4 mb-4"></div>
<div className="h-4 bg-white/10 rounded w-1/2"></div>
</div>
);
```

CONCLUSION

ScoutPulse is designed to be the premium baseball recruiting platform with:

• Distinctive glassmorphism design

• Fluid, physics-based animations

• Comprehensive feature set for all user types

• Production-ready code quality

• Accessibility and performance focus

• Scalable architecture

This master spec serves as the single source of truth for all development, design, and product decisions.

Last Updated: December 8, 2025

Version: 2.0

Status: Living Document