PsychSync - Complete Project Documentation

Table of Contents

Product Requirements Document

Vision Statement

PsychSync empowers organizations to build exceptional teams through data-driven personality insights, team optimization algorithms, and predictive analytics, ultimately increasing productivity and reducing team conflicts.

Target Users

Primary Users

1. **HR Managers** - Team formation and organizational development
2. **Team Leaders** - Team optimization and performance improvement
3. **Project Managers** - Resource allocation and team planning
4. **C-Level Executives** - Strategic workforce planning

Secondary Users

1. **Team Members** - Self-awareness and professional development
2. **Consultants** - Team building and organizational consulting
3. **Coaches** - Leadership and team development

User Personas

Sarah Chen - HR Director

- **Age**: 35-45
- **Experience**: 10+ years in HR

- **Goals**: Improve team formation efficiency, reduce turnover
- **Pain Points**: Time-consuming team building, subjective decision making
- **Tech Comfort**: High

Mike Rodriguez - Engineering Manager

- **Age**: 30-40
- **Experience**: 8+ years in tech leadership
- **Goals**: Build high-performing development teams
- **Pain Points**: Team conflicts, poor collaboration
- **Tech Comfort**: Very High

Lisa Thompson - Team Lead

- **Age**: 28-38
- **Experience**: 5+ years in management
- **Goals**: Understand team dynamics, improve communication
- **Pain Points**: Personality clashes, unclear team roles
- **Tech Comfort**: Medium-High

Functional Requirements

Core Features

1. User Management

- **User Registration & Authentication**
  - Email/password registration
  - SSO integration (Google, Microsoft, SAML)
  - Multi-factor authentication
  - Password reset functionality
- **User Profiles**
  - Personal information management
  - Professional background
  - Assessment history
  - Privacy settings
- **Organization Management**
  - Multi-tenant architecture
  - User role management

    - Organization settings
    - Billing and subscription management

2. Assessment System

- **Assessment Frameworks**
    - MBTI (Myers-Briggs Type Indicator)
    - Big Five (OCEAN) personality traits
    - DISC behavioral assessment
    - Enneagram types
    - StrengthsFinder themes
    - Custom assessment creation
- **Assessment Taking**
    - Adaptive questioning
    - Progress saving and resumption
    - Time tracking
    - Mobile-optimized interface
- **Results Processing**
    - Real-time scoring
    - Detailed personality profiles
    - Comparative analysis
    - Historical tracking

3. Team Management

- **Team Creation & Configuration**
    - Team setup wizard
    - Member invitation system
    - Role and responsibility assignment
    - Team goal setting
- **Team Composition Analysis**
    - Personality distribution visualization
    - Compatibility scoring
    - Potential conflict identification
    - Collaboration strength assessment
- **Team Optimization**

- AI-powered team recommendations
- Role-based optimization
- Skill gap analysis
- Performance prediction

4. Analytics & Insights

- **Individual Analytics**
  - Personality profile dashboard
  - Growth tracking
  - Skill development recommendations
  - Career path suggestions
- **Team Analytics**
  - Team dynamics visualization
  - Performance metrics tracking
  - Communication pattern analysis
  - Productivity correlation analysis
- **Organizational Analytics**
  - Company-wide personality trends
  - Team performance benchmarking
  - Recruitment insights
  - Culture analysis

5. Reporting & Visualization

- **Interactive Dashboards**
  - Real-time metrics display
  - Customizable widgets
  - Drill-down capabilities
  - Export functionality
- **Automated Reports**
  - Scheduled report generation
  - Custom report builder
  - Email delivery
  - PDF/Excel export

Non-Functional Requirements

Performance

- **Response Time**: API endpoints < 200ms (95th percentile)
- **Page Load**: Initial load < 2 seconds
- **Concurrent Users**: Support 10,000+ simultaneous users
- **Throughput**: Handle 1,000+ requests per second

Scalability

- **Horizontal Scaling**: Auto-scaling based on demand
- **Database Scaling**: Read replicas and query optimization
- **CDN Integration**: Global content delivery
- **Microservices**: Service isolation and independent scaling

Security

- **Data Encryption**: TLS 1.3 in transit, AES-256 at rest
- **Authentication**: JWT with refresh tokens
- **Authorization**: Role-based access control (RBAC)
- **Compliance**: GDPR, SOC 2, HIPAA ready

Reliability

- **Uptime**: 99.9% availability SLA
- **Backup**: Daily automated backups with point-in-time recovery
- **Disaster Recovery**: Multi-region deployment capability
- **Monitoring**: Comprehensive logging and alerting

Technical Specifications

System Requirements

Minimum Hardware Requirements

```yaml
```

```
Development Environment:
  CPU: 4 cores, 2.5GHz
  RAM: 16GB
  Storage: 500GB SSD
  Network: Broadband internet

Production Environment:
  API Servers: 8 cores, 3.0GHz, 32GB RAM
  Database: 16 cores, 2.8GHz, 64GB RAM, 1TB NVMe
  Cache: 4 cores, 2.5GHz, 16GB RAM
```

Software Dependencies

yaml

```
Runtime:
  Node.js: >= 18.x
  Python: >= 3.9
  PostgreSQL: >= 14.x
  Redis: >= 6.x
  Elasticsearch: >= 7.x

Development:
  Docker: >= 20.x
  Kubernetes: >= 1.24
  Terraform: >= 1.0
  Git: >= 2.30
```

Data Models

Core Entities

typescript

```typescript
// User Entity
interface User {
  id: string;
  organizationId: string;
  email: string;
  firstName: string;
  lastName: string;
  role: UserRole;
  isActive: boolean;
  lastLogin?: Date;
  preferences: UserPreferences;
  createdAt: Date;
  updatedAt: Date;
}

// Team Entity
interface Team {
  id: string;
  organizationId: string;
  name: string;
  description?: string;
  teamType: TeamType;
  status: TeamStatus;
  createdBy: string;
  members: TeamMember[];
  analytics: TeamAnalytics;
  createdAt: Date;
  updatedAt: Date;
}

// Assessment Entity
interface Assessment {
  id: string;
  userId: string;
  frameworkId: string;
  status: AssessmentStatus;
  startedAt: Date;
  completedAt?: Date;
  results: AssessmentResults;
  rawScores: Record<string, number>;
  responses: AssessmentResponse[];
}

// Assessment Framework
interface AssessmentFramework {
  id: string;
```

```
  name: string;
  description: string;
  version: string;
  questionCount: number;
  estimatedDuration: number; // minutes
  categories: string[];
  scoringMethod: ScoringMethod;
  questions: AssessmentQuestion[];
}
```

Algorithms & Calculations

Team Compatibility Algorithm

```python
def calculate_team_compatibility(team_members: List[PersonalityProfile]) -> float:
    """
    Calculate overall team compatibility score based on personality profiles

    Factors considered:
    - Personality trait balance (0.3 weight)
    - Communication style compatibility (0.25 weight)
    - Work style alignment (0.25 weight)
    - Conflict potential (0.2 weight)

    Returns: Compatibility score between 0.0 and 1.0
    """
    personality_balance = calculate_personality_balance(team_members)
    communication_compatibility = calculate_communication_compatibility(team_members)
    work_style_alignment = calculate_work_style_alignment(team_members)
    conflict_potential = 1.0 - calculate_conflict_potential(team_members)

    compatibility_score = (
        personality_balance * 0.3 +
        communication_compatibility * 0.25 +
        work_style_alignment * 0.25 +
        conflict_potential * 0.2
    )

    return round(compatibility_score, 3)
```

Velocity Prediction Algorithm

```python
```

```
def predict_team_velocity(
    team_profile: TeamProfile,
    historical_data: List[TeamPerformance]
) -> PredictionResult:
    """
    Predict team velocity based on personality composition and historical data

    Uses machine learning model trained on:
    - Team personality distributions
    - Historical velocity data
    - Project complexity factors
    - Team size and experience
    """
    features = extract_team_features(team_profile)
    model = load_velocity_prediction_model()

    predicted_velocity = model.predict(features)
    confidence_interval = calculate_confidence_interval(features, historical_data)

    return PredictionResult(
        predicted_velocity=predicted_velocity,
        confidence_score=confidence_interval.confidence,
        factors=analyze_prediction_factors(features),
        recommendations=generate_velocity_recommendations(team_profile)
    )
```

User Stories & Acceptance Criteria

Epic: User Authentication

Story 1: User Registration

As a new user
I want to create an account with email and password
So that I can access the PsychSync platform

Acceptance Criteria:

- [ ] User can register with email and password
- [ ] Email validation is performed
- [ ] Password strength requirements are enforced
- [ ] Confirmation email is sent
- [ ] User account is activated upon email confirmation
- [ ] Error messages are clear and helpful

Story 2: User Login

As a registered user
I want to log in to my account
So that I can access my teams and assessments

Acceptance Criteria:

- [ ] User can log in with email and password
- [ ] Invalid credentials show appropriate error
- [ ] Successful login redirects to dashboard
- [ ] Session is maintained across browser tabs
- [ ] Remember me option available

Epic: Team Management

Story 3: Create Team

As a team leader
I want to create a new team
So that I can organize my team members and assess team dynamics

Acceptance Criteria:

- [ ] User can create team with name and description
- [ ] Team type can be selected (Development, Marketing, etc.)
- [ ] Team goals can be defined
- [ ] User is automatically assigned as team lead
- [ ] Team appears in team list immediately

Story 4: Add Team Members

As a team leader
I want to add members to my team
So that we can complete assessments and analyze team compatibility

Acceptance Criteria:

- [ ] Members can be added by email invitation
- [ ] Members can be assigned specific roles
- [ ] Invitation emails are sent automatically
- [ ] Pending invitations are tracked
- [ ] Members appear in team roster upon acceptance

Epic: Assessment Taking

Story 5: Take Personality Assessment

As a team member
I want to complete a personality assessment
So that my team can understand my work style and preferences

Acceptance Criteria:

- [ ] User can select from available assessment frameworks
- [ ] Questions are presented one at a time
- [ ] Progress is saved automatically
- [ ] Assessment can be paused and resumed
- [ ] Results are calculated upon completion
- [ ] Results are immediately available

Story 6: View Assessment Results

As a user
I want to view my assessment results
So that I can understand my personality profile and how it affects my work

Acceptance Criteria:

- [ ] Results display personality type/profile
- [ ] Detailed trait explanations are provided
- [ ] Visual charts show trait distributions
- [ ] Results can be shared with team
- [ ] Historical results can be compared

Epic: Team Optimization

Story 7: Analyze Team Compatibility

As a team leader
I want to see team compatibility analysis
So that I can understand potential strengths and challenges

Acceptance Criteria:

- [ ] Compatibility score is calculated and displayed
- [ ] Strengths and potential conflicts are identified
- [ ] Visual representation shows team balance
- [ ] Recommendations for improvement are provided
- [ ] Analysis updates when team composition changes

Story 8: Get Team Recommendations

As a team leader
I want to receive recommendations for team optimization
So that I can improve team performance and reduce conflicts

Acceptance Criteria:

- [ ] AI provides specific recommendations
- [ ] Recommendations include rationale
- [ ] Role assignment suggestions are provided
- [ ] Alternative team compositions are shown
- [ ] Impact predictions are included

Epic: Analytics Dashboard

Story 9: View Team Analytics

As a team leader
I want to see analytics about my team's performance
So that I can track progress and identify improvement areas

Acceptance Criteria:

- [ ] Dashboard shows key performance metrics
- [ ] Charts display trends over time
- [ ] Metrics can be filtered by time period
- [ ] Comparisons with other teams available
- [ ] Drill-down capabilities for detailed analysis

Story 10: Export Reports

As a manager
I want to export team reports
So that I can share insights with stakeholders

Acceptance Criteria:

- [ ] Reports can be exported as PDF
- [ ] Excel format is available
- [ ] Custom report templates can be created
- [ ] Scheduled report delivery is possible
- [ ] Reports include visual charts and graphs

API Documentation

Authentication Endpoints

POST /auth/login

Description: Authenticate user and return access token

Request Body:

```json
{
  "email": "user@example.com",
  "password": "securePassword123"
}
```

Response:

```json
{
  "success": true,
  "data": {
    "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "refresh_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "user": {
      "id": "user_123",
      "email": "user@example.com",
      "firstName": "John",
      "lastName": "Doe",
      "role": "team_lead"
    }
  }
}
```

POST /auth/register

Description: Register new user account

Request Body:

```json
{
  "email": "newuser@example.com",
  "password": "securePassword123",
  "firstName": "Jane",
  "lastName": "Smith",
  "organizationName": "Acme Corp"
}
```

Team Management Endpoints

GET /teams

Description: Retrieve all teams for authenticated user

Query Parameters:

- `page` (optional): Page number for pagination
- `limit` (optional): Number of teams per page
- `status` (optional): Filter by team status
- `search` (optional): Search term for team name/description

Response:

```json
{
  "success": true,
  "data": {
    "teams": [
      {
        "id": "team_123",
        "name": "Frontend Development Team",
        "description": "React and TypeScript specialists",
        "memberCount": 5,
        "status": "active",
        "compatibility": 0.87,
        "createdAt": "2024-01-15T10:00:00Z"
      }
    ],
    "pagination": {
      "page": 1,
      "limit": 20,
      "total": 45,
      "totalPages": 3
    }
  }
}
```

POST /teams

Description: Create new team

Request Body:

```json
{
  "name": "New Development Team",
  "description": "Full-stack development team for mobile app",
  "teamType": "development",
  "goals": ["Increase velocity", "Improve code quality"],
  "members": [
    {
      "userId": "user_456",
      "role": "developer"
    }
  ]
}
```

Assessment Endpoints

GET /assessments/frameworks

Description: Get available assessment frameworks

Response:

```json
```

```
{
  "success": true,
  "data": {
    "frameworks": [
      {
        "id": "mbti_v1",
        "name": "Myers-Briggs Type Indicator",
        "description": "Psychological preferences in how people perceive the world",
        "questionCount": 93,
        "estimatedDuration": 25,
        "categories": ["Extraversion", "Sensing", "Thinking", "Judging"],
        "isActive": true
      },
      {
        "id": "big_five_v2",
        "name": "Big Five Personality Traits",
        "description": "Five-factor model of personality dimensions",
        "questionCount": 120,
        "estimatedDuration": 30,
        "categories": ["Openness", "Conscientiousness", "Extraversion", "Agreeableness", "Neuroticism"],
        "isActive": true
      }
    ]
  }
}
```


POST /assessments

Description: Create new assessment session

Request Body:

```
json

{
  "frameworkId": "mbti_v1",
  "userId": "user_123"
}
```

Response:

```
json
```

```
{
  "success": true,
  "data": {
    "assessmentId": "assessment_789",
    "frameworkId": "mbti_v1",
    "status": "in_progress",
    "questionCount": 93,
    "currentQuestion": 1,
    "startedAt": "2024-01-15T14:30:00Z"
  }
}
```


POST /assessments/{assessmentId}/responses

Description: Submit answer to assessment question

Request Body:


```
json

{
  "questionId": "q_001",
  "responseValue": "4",
  "responseTime": 5.2
}
```


Analytics Endpoints

GET /analytics/dashboard

Description: Get dashboard analytics data

Query Parameters:

- `timeframe` (optional): 7d, 30d, 90d, 1y (default: 30d)
- `teamId` (optional): Filter by specific team

Response:


```
json
```

```
{
  "success": true,
  "data": {
    "summary": {
      "totalTeams": 12,
      "totalAssessments": 156,
      "avgCompatibility": 0.84,
      "activeUsers": 89
    },
    "trends": {
      "assessmentsOverTime": [
        {"date": "2024-01-01", "count": 5},
        {"date": "2024-01-02", "count": 8}
      ],
      "compatibilityTrend": [
        {"date": "2024-01-01", "score": 0.82},
        {"date": "2024-01-02", "score": 0.84}
      ]
    },
    "insights": [
      {
        "type": "team_performance",
        "title": "High-performing teams show 23% better compatibility",
        "description": "Teams with compatibility scores above 0.85 consistently deliver projects faster",
        "confidenceScore": 0.92
      }
    ]
  }
}
```


Database Schema

Complete Schema with Relationships

```sql
```

```sql
-- Core Tables
CREATE TABLE organizations (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    name VARCHAR(255) NOT NULL,
    domain VARCHAR(100),
    subscription_plan VARCHAR(50) DEFAULT 'free',
    settings JSONB DEFAULT '{}',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE users (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    organization_id UUID REFERENCES organizations(id) ON DELETE CASCADE,
    email VARCHAR(255) UNIQUE NOT NULL,
    password_hash VARCHAR(255),
    first_name VARCHAR(100),
    last_name VARCHAR(100),
    role VARCHAR(50) DEFAULT 'member',
    is_active BOOLEAN DEFAULT true,
    email_verified BOOLEAN DEFAULT false,
    last_login TIMESTAMP,
    preferences JSONB DEFAULT '{}',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Teams and Membership
CREATE TABLE teams (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    organization_id UUID REFERENCES organizations(id) ON DELETE CASCADE,
    name VARCHAR(255) NOT NULL,
    description TEXT,
    team_type VARCHAR(50) DEFAULT 'general',
    status VARCHAR(50) DEFAULT 'active',
    goals TEXT[],
    created_by UUID REFERENCES users(id),
    settings JSONB DEFAULT '{}',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE team_members (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    team_id UUID REFERENCES teams(id) ON DELETE CASCADE,
    user_id UUID REFERENCES users(id) ON DELETE CASCADE,
```

```
    role VARCHAR(50) DEFAULT 'member',
    permissions TEXT[],
    joined_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    left_at TIMESTAMP,
    UNIQUE(team_id, user_id)
);

-- Assessment Framework
CREATE TABLE assessment_frameworks (
    id VARCHAR(50) PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    description TEXT,
    version VARCHAR(20),
    question_count INTEGER,
    estimated_duration INTEGER,
    categories TEXT[],
    scoring_method VARCHAR(50),
    configuration JSONB DEFAULT '{}',
    is_active BOOLEAN DEFAULT true,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE assessment_questions (
    id VARCHAR(50) PRIMARY KEY,
    framework_id VARCHAR(50) REFERENCES assessment_frameworks(id),
    question_text TEXT NOT NULL,
    question_type VARCHAR(50),
    options JSONB,
    category VARCHAR(100),
    subcategory VARCHAR(100),
    order_index INTEGER,
    scoring_key VARCHAR(50),
    reverse_scored BOOLEAN DEFAULT false,
    is_active BOOLEAN DEFAULT true
);

-- User Assessments
CREATE TABLE assessments (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID REFERENCES users(id) ON DELETE CASCADE,
    framework_id VARCHAR(50) REFERENCES assessment_frameworks(id),
    status VARCHAR(50) DEFAULT 'in_progress',
    current_question INTEGER DEFAULT 1,
    started_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    completed_at TIMESTAMP,
    results JSONB,
```

```
    raw_scores JSONB,
    processing_metadata JSONB DEFAULT '{}'
);

CREATE TABLE assessment_responses (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    assessment_id UUID REFERENCES assessments(id) ON DELETE CASCADE,
    question_id VARCHAR(50) REFERENCES assessment_questions(id),
    response_value TEXT,
    response_score DECIMAL(5,2),
    response_time DECIMAL(8,2),
    answered_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Analytics and Insights
CREATE TABLE team_analytics (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    team_id UUID REFERENCES teams(id) ON DELETE CASCADE,
    metric_type VARCHAR(100),
    metric_value DECIMAL(10,4),
    metric_metadata JSONB DEFAULT '{}',
    calculation_date DATE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE team_insights (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    team_id UUID REFERENCES teams(id) ON DELETE CASCADE,
    insight_type VARCHAR(100),
    insight_title VARCHAR(255),
    insight_description TEXT,
    insight_data JSONB,
    confidence_score DECIMAL(3,2),
    generated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    expires_at TIMESTAMP
);

CREATE TABLE optimization_sessions (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    team_id UUID REFERENCES teams(id) ON DELETE CASCADE,
    requested_by UUID REFERENCES users(id),
    optimization_type VARCHAR(50),
    input_parameters JSONB,
    recommendations JSONB,
    status VARCHAR(50) DEFAULT 'processing',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    completed_at TIMESTAMP
```

```
);

-- Audit and Logging
CREATE TABLE audit_logs (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID REFERENCES users(id),
    action VARCHAR(100),
    resource_type VARCHAR(50),
    resource_id VARCHAR(100),
    old_values JSONB,
    new_values JSONB,
    ip_address INET,
    user_agent TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Indexes for Performance
CREATE INDEX idx_users_org_id ON users(organization_id);
CREATE INDEX idx_users_email ON users(email);
CREATE INDEX idx_users_active ON users(is_active);

CREATE INDEX idx_teams_org_id ON teams(organization_id);
CREATE INDEX idx_teams_status ON teams(status);
CREATE INDEX idx_teams_created_by ON teams(created_by);

CREATE INDEX idx_team_members_team_id ON team_members(team_id);
CREATE INDEX idx_team_members_user_id ON team_members(user_id);
CREATE INDEX idx_team_members_active ON team_members(team_id, user_id) WHERE left_at IS NULL;

CREATE INDEX idx_assessments_user_id ON assessments(user_id);
CREATE INDEX idx_assessments_framework_id ON assessments(framework_id);
CREATE INDEX idx_assessments_status ON assessments(status);
CREATE INDEX idx_assessments_completed ON assessments(completed_at) WHERE completed_at IS NOT NUL

CREATE INDEX idx_assessment_responses_assessment_id ON assessment_responses(assessment_id);
CREATE INDEX idx_assessment_responses_question_id ON assessment_responses(question_id);

CREATE INDEX idx_team_analytics_team_date ON team_analytics(team_id, calculation_date);
CREATE INDEX idx_team_analytics_metric_type ON team_analytics(metric_type);

CREATE INDEX idx_team_insights_team_id ON team_insights(team_id);
CREATE INDEX idx_team_insights_type ON team_insights(insight_type);
CREATE INDEX idx_team_insights_expires ON team_insights(expires_at) WHERE expires_at IS NOT NULL;

CREATE INDEX idx_audit_logs_user_id ON audit_logs(user_id);
```

```
CREATE INDEX idx_audit_logs_resource ON audit_logs(resource_type, resource_id);
CREATE INDEX idx_audit_logs_created_at ON audit_logs(created_at);
```

Frontend Component Library

Design System

Color Palette

```css
:root {
  /* Primary Colors */
  --color-primary-50: #eff6ff;
  --color-primary-100: #dbeafe;
  --color-primary-500: #3b82f6;
  --color-primary-600: #2563eb;
  --color-primary-700: #1d4ed8;

  /* Secondary Colors */
  --color-secondary-50: #f8fafc;
  --color-secondary-100: #f1f5f9;
  --color-secondary-500: #64748b;
  --color-secondary-600: #475569;
  --color-secondary-700: #334155;

  /* Success Colors */
  --color-success-50: #f0fdf4;
  --color-success-500: #22c55e;
  --color-success-600: #16a34a;

  /* Error Colors */
  --color-error-50: #fef2f2;
  --color-error-500: #ef4444;
  --color-error-600: #dc2626;

  /* Warning Colors */
  --color-warning-50: #fffbeb;
  --color-warning-500: #f59e0b;
  --color-warning-600: #d97706;
}
```

Typography Scale

```css
```

```
.text-xs { font-size: 0.75rem; line-height: 1rem; }
.text-sm { font-size: 0.875rem; line-height: 1.25rem; }
.text-base { font-size: 1rem; line-height: 1.5rem; }
.text-lg { font-size: 1.125rem; line-height: 1.75rem; }
.text-xl { font-size: 1.25rem; line-height: 1.75rem; }
.text-2xl { font-size: 1.5rem; line-height: 2rem; }
.text-3xl { font-size: 1.875rem; line-height: 2.25rem; }
```

Core Components

Button Component

typescript

```tsx
import React from 'react';
import { cva, type VariantProps } from 'class-variance-authority';

const buttonVariants = cva(
  "inline-flex items-center justify-center rounded-md font-medium transition-colors focus:outline-none focus:ring
  {
    variants: {
      variant: {
        primary: "bg-blue-600 text-white hover:bg-blue-700 focus:ring-blue-500",
        secondary: "bg-gray-200 text-gray-900 hover:bg-gray-300 focus:ring-gray-500",
        outline: "border border-gray-300 bg-white text-gray-700 hover:bg-gray-50 focus:ring-gray-500",
        ghost: "text-gray-700 hover:bg-gray-100 focus:ring-gray-500",
        destructive: "bg-red-600 text-white hover:bg-red-700 focus:ring-red-500"
      },
      size: {
        sm: "h-8 px-3 text-sm",
        default: "h-10 px-4 py-2",
        lg: "h-12 px-6 text-lg",
        icon: "h-10 w-10"
      }
    },
    defaultVariants: {
      variant: "primary",
      size: "default"
    }
  }
);

interface ButtonProps
  extends React.ButtonHTMLAttributes<HTMLButtonElement>,
    VariantProps<typeof buttonVariants> {
  loading?: boolean;
  leftIcon?: React.ReactNode;
  rightIcon?: React.ReactNode;
}

export const Button = React.forwardRef<HTMLButtonElement, ButtonProps>(
  ({ className, variant, size, loading, leftIcon, rightIcon, children, disabled, ...props }, ref) => {
    return (
      <button
        className={buttonVariants({ variant, size, className })}
        ref={ref}
        disabled={disabled || loading}
        {...props}
      >
        {loading && <LoadingSpinner size="sm" className="mr-2" />}
```

```
      {leftIcon && !loading && <span className="mr-2">{leftIcon}</span>}
      {children}
      {rightIcon && <span className="ml-2">{rightIcon}</span>}
    </button>
  );
 }
);
```

Input Component

typescript

```tsx
import React from 'react';
import { cva, type VariantProps } from 'class-variance-authority';

const inputVariants = cva(
  "flex w-full rounded-md border px-3 py-2 text-sm transition-colors file:border-0 file:bg-transparent file:text-sm
  {
    variants: {
      variant: {
        default: "border-gray-300 bg-white focus:border-blue-500 focus:ring-blue-500",
        error: "border-red-500 bg-red-50 focus:border-red-500 focus:ring-red-500",
        success: "border-green-500 bg-green-50 focus:border-green-500 focus:ring-green-500"
      }
    },
    defaultVariants: {
      variant: "default"
    }
  }
);

interface InputProps
  extends React.InputHTMLAttributes<HTMLInputElement>,
    VariantProps<typeof inputVariants> {
  label?: string;
  error?: string;
  helperText?: string;
}

export const Input = React.forwardRef<HTMLInputElement, InputProps>(
  ({ className, variant, label, error, helperText, id, ...props }, ref) => {
    const inputId = id || `input-${Math.random().toString(36).substr(2, 9)}`;
    const inputVariant = error ? 'error' : variant;

    return (
      <div className="space-y-2">
        {label && (
          <label
            htmlFor={inputId}
            className="text-sm font-medium leading-none peer-disabled:cursor-not-allowed peer-disabled:opacity-
          >
            {label}
          </label>
        )}
        <input
          className={inputVariants({ variant: inputVariant, className })}
          ref={ref}
          id={inputId}
```

```
        {...props}
      />
      {(error || helperText) && (
        <p className={`text-sm ${error ? 'text-red-600' : 'text-gray-500'}`}>
          {error || helperText}
        </p>
      )}
    </div>
  );
 }
);
```

Modal Component

typescript

```tsx
import React from 'react';
import { createPortal } from 'react-dom';
import { XMarkIcon } from '@heroicons/react/24/outline';

interface ModalProps {
  isOpen: boolean;
  onClose: () => void;
  title?: string;
  children: React.ReactNode;
  size?: 'sm' | 'md' | 'lg' | 'xl';
  showCloseButton?: boolean;
}

const sizeClasses = {
  sm: 'max-w-sm',
  md: 'max-w-md',
  lg: 'max-w-lg',
  xl: 'max-w-4xl'
};

export const Modal: React.FC<ModalProps> = ({
  isOpen,
  onClose,
  title,
  children,
  size = 'md',
  showCloseButton = true
}) => {
  if (!isOpen) return null;

  const modalContent = (
    <div className="fixed inset-0 z-50 overflow-y-auto">
      <div className="flex min-h-screen items-center justify-center p-4">
        {/* Backdrop */}
        <div
          className="fixed inset-0 bg-black bg-opacity-50 transition-opacity"
          onClick={onClose}
        />

        {/* Modal */}
        <div className={`relative bg-white rounded-lg shadow-xl w-full ${sizeClasses[size]} transform transition-al
          {/* Header */}
          {(title || showCloseButton) && (
            <div className="flex items-center justify-between p-6 border-b">
              {title && <h3 className="text-lg font-semibold text-gray-900">{title}</h3>}
              {showCloseButton && (
```

```
            <button
              onClick={onClose}
              className="p-1 text-gray-400 hover:text-gray-600 transition-colors"
            >
              <XMarkIcon className="h-6 w-6" />
            </button>
          )}
        </div>
      )}

      {/* Content */}
      <div className="p-6">
        {children}
      </div>
    </div>
  </div>
 );

 return createPortal(modalContent, document.body);
};
```

Assessment Components

AssessmentCard Component

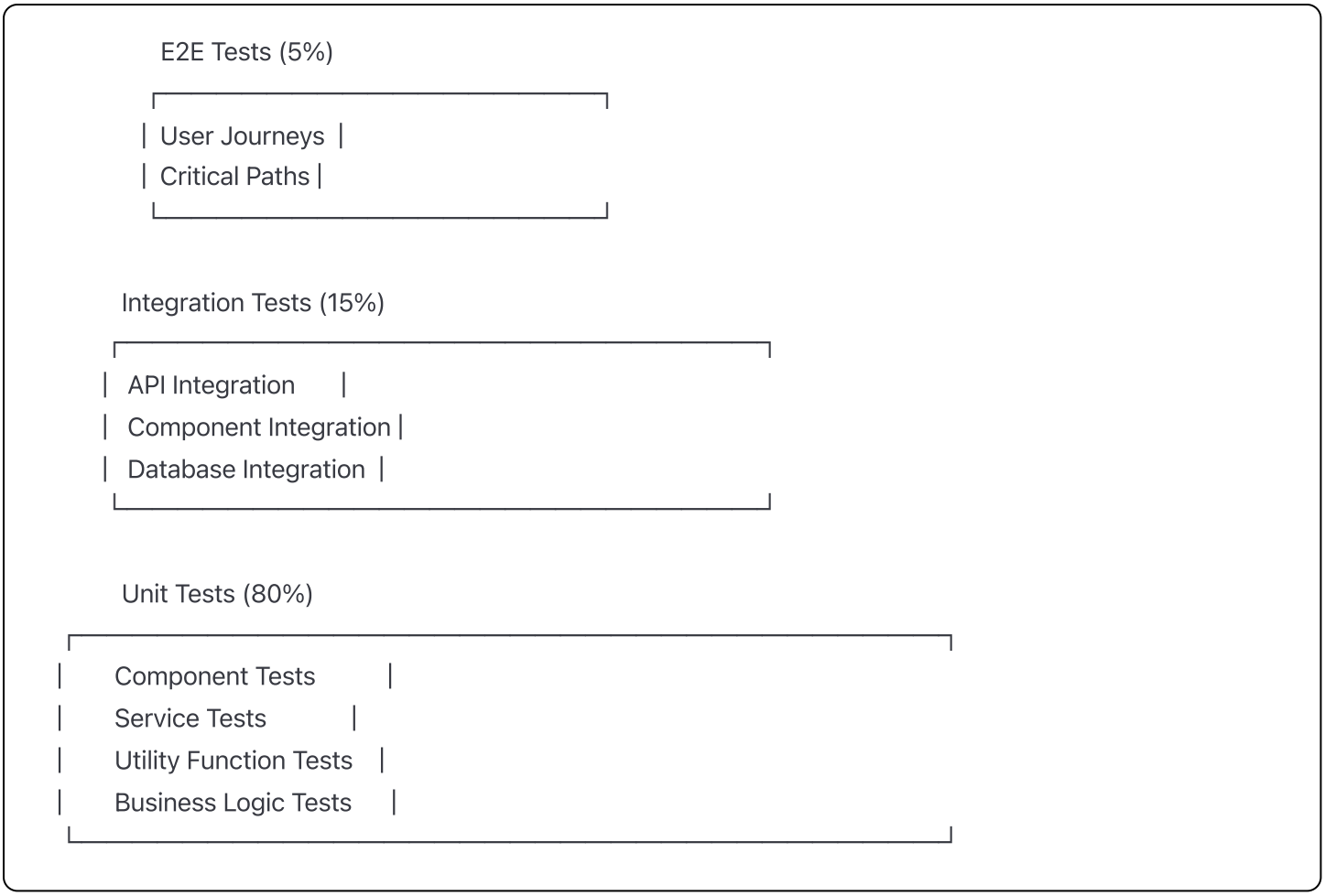
typescript

```tsx
import React from 'react';
import { Button } from '../ui/Button';
import { Badge } from '../ui/Badge';

interface AssessmentFramework {
  id: string;
  name: string;
  description: string;
  questionCount: number;
  estimatedDuration: number;
  isCompleted?: boolean;
  completedDate?: string;
}

interface AssessmentCardProps {
  framework: AssessmentFramework;
  onStartAssessment: (frameworkId: string) => void;
  onViewResults: (frameworkId: string) => void;
}

export const AssessmentCard: React.FC<AssessmentCardProps> = ({
  framework,
  onStartAssessment,
  onViewResults
}) => {
  return (
    <div className="bg-white border border-gray-200 rounded-lg p-6 hover:shadow-md transition-shadow">
      <div className="flex items-start justify-between mb-4">
        <h3 className="text-lg font-semibold text-gray-900">{framework.name}</h3>
        <Badge variant={framework.isCompleted ? 'success' : 'secondary'}>
          {framework.isCompleted ? 'Completed' : 'Available'}
        </Badge>
      </div>

      <p className="text-gray-600 mb-4">{framework.description}</p>

      <div className="flex items-center text-sm text-gray-500 mb-6 space-x-4">
        <span>{framework.questionCount} questions</span>
        <span>{framework.estimatedDuration} minutes</span>
      </div>

      <div className="flex space-x-3">
        {framework.isCompleted ? (
          <>
            <Button
              variant="outline"
```

```
            onClick={() => onViewResults(framework.id)}
          >
            View Results
          </Button>
          <Button
            variant="ghost"
            onClick={() => onStartAssessment(framework.id)}
          >
            Retake
          </Button>
        </>
      ) : (
        <Button onClick={() => onStartAssessment(framework.id)}>
          Start Assessment
        </Button>
      )}
    </div>

    {framework.isCompleted && framework.completedDate && (
      <p className="text-xs text-gray-500 mt-4">
        Completed on {new Date(framework.completedDate).toLocaleDateString()}
      </p>
    )}
  </div>
 );
};
```

Testing Strategy

Testing Pyramid



Unit Testing

Component Testing Example

typescript

```
import { render, screen, fireEvent, waitFor } from '@testing-library/react';
import { AssessmentCard } from '../AssessmentCard';

const mockFramework = {
  id: 'mbti_v1',
  name: 'Myers-Briggs Type Indicator',
  description: 'Psychological preferences assessment',
  questionCount: 93,
  estimatedDuration: 25,
  isCompleted: false
};

describe('AssessmentCard', () => {
  const mockOnStartAssessment = jest.fn();
  const mockOnViewResults = jest.fn();

  beforeEach(() => {
    jest.clearAllMocks();
  });

  it('renders framework information correctly', () => {
    render(
      <AssessmentCard
        framework={mockFramework}
        onStartAssessment={mockOnStartAssessment}
        onViewResults={mockOnViewResults}
      />
    );

    expect(screen.getByText('Myers-Briggs Type Indicator')).toBeInTheDocument();
    expect(screen.getByText('Psychological preferences assessment')).toBeInTheDocument();
    expect(screen.getByText('93 questions')).toBeInTheDocument();
    expect(screen.getByText('25 minutes')).toBeInTheDocument();
  });

  it('calls onStartAssessment when start button is clicked', () => {
    render(
      <AssessmentCard
        framework={mockFramework}
        onStartAssessment={mockOnStartAssessment}
        onViewResults={mockOnViewResults}
      />
    );

    fireEvent.click(screen.getByText('Start Assessment'));
    expect(mockOnStartAssessment).toHaveBeenCalledWith('mbti_v1');
```

```
  });

  it('shows completed state for completed assessments', () => {
    const completedFramework = {
      ...mockFramework,
      isCompleted: true,
      completedDate: '2024-01-15T10:00:00Z'
    };

    render(
      <AssessmentCard
        framework={completedFramework}
        onStartAssessment={mockOnStartAssessment}
        onViewResults={mockOnViewResults}
      />
    );

    expect(screen.getByText('Completed')).toBeInTheDocument();
    expect(screen.getByText('View Results')).toBeInTheDocument();
    expect(screen.getByText('Retake')).toBeInTheDocument();
  });
});
```

Service Testing Example

typescript

```
import { TeamService } from '../TeamService';
import { ApiClient } from '../ApiClient';

jest.mock('../ApiClient');

describe('TeamService', () => {
  let teamService: TeamService;
  let mockApiClient: jest.Mocked<ApiClient>;

  beforeEach(() => {
    mockApiClient = new ApiClient() as jest.Mocked<ApiClient>;
    teamService = new TeamService(mockApiClient);
  });

  describe('createTeam', () => {
    it('should create team successfully', async () => {
      const teamData = {
        name: 'Test Team',
        description: 'A test team',
        teamType: 'development'
      };

      const expectedResponse = {
        id: 'team_123',
        ...teamData,
        createdAt: '2024-01-15T10:00:00Z'
      };

      mockApiClient.post.mockResolvedValue({
        success: true,
        data: expectedResponse
      });

      const result = await teamService.createTeam(teamData);

      expect(mockApiClient.post).toHaveBeenCalledWith('/teams', teamData);
      expect(result).toEqual(expectedResponse);
    });

    it('should handle API errors', async () => {
      const teamData = {
        name: 'Test Team',
        description: 'A test team',
        teamType: 'development'
      };
```

```
    mockApiClient.post.mockRejectedValue(new Error('API Error'));

    await expect(teamService.createTeam(teamData)).rejects.toThrow('API Error');
  });
 });
});
```

Integration Testing

API Integration Test

typescript

```
import request from 'supertest';
import { app } from '../../../app';
import { setupTestDb, teardownTestDb, getTestUser } from '../../helpers/database';

describe('Team API Integration', () => {
  let authToken: string;
  let userId: string;

  beforeAll(async () => {
    await setupTestDb();
    const testUser = await getTestUser();
    authToken = testUser.token;
    userId = testUser.id;
  });

  afterAll(async () => {
    await teardownTestDb();
  });

  describe('POST /api/v1/teams', () => {
    it('should create a new team', async () => {
      const teamData = {
        name: 'Integration Test Team',
        description: 'A team created during integration testing',
        teamType: 'development'
      };

      const response = await request(app)
        .post('/api/v1/teams')
        .set('Authorization', `Bearer ${authToken}`)
        .send(teamData)
        .expect(201);

      expect(response.body.success).toBe(true);
      expect(response.body.data.name).toBe(teamData.name);
      expect(response.body.data.id).toBeDefined();
    });

    it('should return 401 without authentication', async () => {
      const teamData = {
        name: 'Unauthorized Team',
        teamType: 'development'
      };

      await request(app)
        .post('/api/v1/teams')
```

```
      .send(teamData)
      .expect(401);
  });
 });
});
```

End-to-End Testing

Cypress E2E Test

typescript

```
describe('Team Creation Flow', () => {
  beforeEach(() => {
    cy.login('testuser@example.com', 'password123');
    cy.visit('/teams');
  });

  it('should create a new team successfully', () => {
    // Click create team button
    cy.get('[data-testid="create-team-button"]').click();

    // Fill out team creation form
    cy.get('[data-testid="team-name-input"]').type('E2E Test Team');
    cy.get('[data-testid="team-description-input"]').type('Team created during E2E testing');
    cy.get('[data-testid="team-type-select"]').select('development');

    // Submit form
    cy.get('[data-testid="create-team-submit"]').click();

    // Verify team was created
    cy.get('[data-testid="success-notification"]')
      .should('be.visible')
      .and('contain', 'Team created successfully');

    // Verify team appears in list
    cy.get('[data-testid="team-list"]')
      .should('contain', 'E2E Test Team');
  });

  it('should validate required fields', () => {
    cy.get('[data-testid="create-team-button"]').click();
    cy.get('[data-testid="create-team-submit"]').click();

    cy.get('[data-testid="team-name-error"]')
      .should('be.visible')
      .and('contain', 'Team name is required');
  });
});
```

Deployment Guide

Environment Setup

Development Environment

```
bash
```

```
# Prerequisites
node --version  # >= 18.x
python --version  # >= 3.9
docker --version  # >= 20.x
git --version  # >= 2.30

# Clone repository
git clone https://github.com/yourorg/psychsync.git
cd psychsync

# Setup backend
cd backend
python -m venv venv
source venv/bin/activate  # Windows: venv|Scripts|activate
pip install -r requirements.txt

# Setup database
docker-compose up -d postgres redis
python manage.py migrate

# Setup frontend
cd ../frontend
npm install
npm run dev

# Start backend server
cd ../backend
python manage.py runserver
```

Production Deployment

Docker Compose Configuration

yaml

```yaml
version: '3.8'

services:
  postgres:
    image: postgres:14-alpine
    environment:
      POSTGRES_DB: psychsync_prod
      POSTGRES_USER: psychsync
      POSTGRES_PASSWORD: ${DB_PASSWORD}
    volumes:
      - postgres_data:/var/lib/postgresql/data
    ports:
      - "5432:5432"

  redis:
    image: redis:6-alpine
    ports:
      - "6379:6379"

  api:
    build: ./backend
    environment:
      DATABASE_URL: postgresql://psychsync:${DB_PASSWORD}@postgres:5432/psychsync_prod
      REDIS_URL: redis://redis:6379
      JWT_SECRET: ${JWT_SECRET}
    depends_on:
      - postgres
      - redis
    ports:
      - "8000:8000"
```