TCF/TEF Platform - Manager Section Documentation

TCF/TEF Platform Team

2025-08-18

Contents

1 Manager Section - Comprehensive Documentation

1.1 □ Overview

The Manager Section serves as the operational management interface for the TCF/TEF platform, providing content management, user oversight, analytics, and operational control. This section bridges the gap between administrative oversight and user experience, enabling managers to ensure quality content delivery and optimal user experience.

1.2 □ Target Users

- **Junior Managers**: Content creators and basic user management
- **Senior Managers**: Advanced content management, analytics, and team oversight
- **General Managers**: Comprehensive platform management and strategic oversight

1.3 □ Architecture Overview

1.3.1 Role-Based Access Control (RBAC)

- **Junior Manager**: Limited content creation and basic user oversight
- **Senior Manager**: Advanced content management, analytics, and team management
- **General Manager**: Full platform management capabilities

1.3.2 Technology Stack

- **Frontend**: Next.js 15, React 19, TypeScript
- **Styling**: Tailwind CSS v4, shadcn/ui components
- **State Management**: React hooks, Context API, Zustand
- **Authentication**: JWT-based with role-based permissions
- **Real-time**: WebSocket for live updates and notifications
- **Analytics**: Custom analytics dashboard with data visualization

1.4 □ Page-by-Page Documentation

1.4.1 1. Manager Dashboard (`/manager`)

1.4.1.1 Purpose Central command center providing overview of platform operations, key metrics, and quick access to essential functions.

1.4.1.2 Functional Requirements

- **Performance Metrics**: Real-time KPIs and performance indicators
- **Quick Actions**: Common tasks and shortcuts
- **Recent Activity**: Latest platform activities and updates

- **Alert System**: Important notifications and system alerts
- **Navigation Hub**: Easy access to all manager functions

1.4.1.3 User Stories

```
As a manager
I want to see an overview of platform performance
So that I can quickly assess the current state of operations

As a manager
I want quick access to common tasks
So that I can efficiently manage daily operations
```

1.4.1.4 Technical Specifications

```
interface ManagerDashboard {
  metrics: DashboardMetrics;
  recentActivity: ActivityLog[];
  alerts: Alert[];
  quickActions: QuickAction[];
  performanceTrends: TrendData[];
}

interface DashboardMetrics {
  totalUsers: number;
  activeUsers: number;
  contentPublished: number;
  liveSessions: number;
  revenue: number;
  userSatisfaction: number;
}
```

1.4.1.5 API Endpoints

```
// Get dashboard data
GET /api/manager/dashboard
Query Parameters: timeframe, team

// Get performance metrics
GET /api/manager/metrics
Query Parameters: period, category

// Get recent activity
GET /api/manager/activity
Query Parameters: limit, type
```

1.4.2 2. Analytics Overview (`/manager/analytics`)

1.4.2.1 Purpose

Comprehensive analytics and reporting interface for data-driven decision making.

1.4.2.2 Functional Requirements

- **Data Visualization**: Charts, graphs, and interactive dashboards
- **Custom Reports**: Generate and export custom reports
- **Real-time Data**: Live data updates and monitoring
- **Filtering Options**: Advanced filtering and data segmentation
- **Export Capabilities**: Multiple export formats (PDF, CSV, Excel)

1.4.2.3 User Stories

```
As a manager
I want to analyze user engagement patterns
So that I can optimize content delivery strategies

As a manager
I want to generate custom reports
So that I can provide insights to stakeholders
```

1.4.2.4 Technical Specifications

```
interface AnalyticsData {
  userEngagement: EngagementMetrics;
  contentPerformance: ContentMetrics;
  revenueAnalytics: RevenueData;
  learningOutcomes: OutcomeMetrics;
  trends: TrendAnalysis;
}

interface ReportConfig {
  type: ReportType;
  timeframe: TimeRange;
  filters: FilterCriteria;
  format: ExportFormat;
  schedule?: ScheduleConfig;
}
```

1.4.2.5 API Endpoints

```
// Get analytics data
GET /api/manager/analytics
Query Parameters: timeframe, category, filters

// Generate custom report
```

```
POST /api/manager/analytics/reports

// Export data
GET /api/manager/analytics/export
Query Parameters: format, filters
```

1.4.3 3. Content Management (`/manager/content`)

1.4.3.1 Purpose Comprehensive content lifecycle management system for creating, editing, and publishing learning materials.

1.4.3.2 Functional Requirements

- **Content Library**: Centralized content repository
- **Version Control**: Content versioning and history tracking
- **Workflow Management**: Approval and publishing workflows
- **Media Management**: File upload and media handling
- **Quality Control**: Content validation and review processes

1.4.3.3 User Stories

```
As a manager
I want to manage all platform content in one place
So that I can ensure quality and consistency

As a manager
I want to track content performance
So that I can optimize learning materials
```

1.4.3.4 Technical Specifications

```
interface ContentItem {
  id: string;
  title: string;
  description: string;
  type: ContentType;
  status: ContentStatus;
  version: number;
  createdBy: string;
  createdAt: Date;
  updatedAt: Date;
  metadata: ContentMetadata;
  performance: PerformanceMetrics;
}

interface ContentWorkflow {
  id: string;
  contentId: string;
  status: WorkflowStatus;
  assignedTo: string;
  dueDate: Date;
  comments: Comment[];
  history: WorkflowHistory[];
}
```

1.4.3.5 API Endpoints

```
// Get content library
GET /api/manager/content
Query Parameters: type, status, author, date

// Create content
POST /api/manager/content

// Update content
PUT /api/manager/content/:id

// Publish content
POST /api/manager/content/:id/publish

// Get content analytics
GET /api/manager/content/:id/analytics
```

1.4.4 4. Content Creation (`/manager/content/create`)

1.4.4.1 Purpose Advanced content creation interface with rich editing capabilities and multimedia support.

1.4.4.2 Functional Requirements

- **Rich Text Editor**: Advanced text editing with formatting options
- **Media Integration**: Image, video, and audio embedding
- **Template System**: Pre-built content templates
- **Collaboration Tools**: Multi-user editing and commenting
- **Preview System**: Real-time content preview

1.4.4.3 User Stories

```
As a manager
I want to create rich, engaging content
So that students have high-quality learning materials

As a manager
I want to collaborate with team members
So that we can create better content together
```

1.4.4.4 Technical Specifications

```
interface ContentEditor {
  content: ContentData;
  template: Template;
  collaborators: Collaborator[];
  version: number;
  autoSave: boolean;
  preview: PreviewData;
}

interface ContentTemplate {
  id: string;
  name: string;
  category: TemplateCategory;
  structure: TemplateStructure;
  variables: TemplateVariable[];
}
```

1.4.4.5 API Endpoints

```
// Get content templates
GET /api/manager/content/templates

// Create content draft
POST /api/manager/content/draft

// Save content
PUT /api/manager/content/:id/save

// Preview content
POST /api/manager/content/:id/preview
```

1.4.5 5. User Management (`/manager/students`)

1.4.5.1 Purpose Comprehensive user management system for overseeing student accounts, progress, and engagement.

1.4.5.2 Functional Requirements

- **User Directory**: Complete user listing with search and filters
- **Progress Tracking**: Individual and group progress monitoring
- **Communication Tools**: Direct messaging and notification system
- **Account Management**: User account administration
- **Performance Analytics**: User performance insights

1.4.5.3 User Stories

```
As a manager
I want to monitor student progress
So that I can provide targeted support

As a manager
I want to communicate with students
So that I can address their needs effectively
```

1.4.5.4 Technical Specifications

```
interface UserManagement {
  users: User[];
  filters: UserFilters;
  bulkActions: BulkAction[];
  communication: CommunicationTools;
  analytics: UserAnalytics;
}

interface User {
  id: string;
  profile: UserProfile;
  progress: ProgressData;
  engagement: EngagementMetrics;
  subscription: SubscriptionData;
  status: UserStatus;
}
```

1.4.5.5 API Endpoints

```
// Get user list
GET /api/manager/users
Query Parameters: filters, search, page, limit

// Get user details
GET /api/manager/users/:id

// Update user
PUT /api/manager/users/:id

// Send message to user
POST /api/manager/users/:id/message

// Get user analytics
GET /api/manager/users/:id/analytics
```

1.4.6 6. Live Sessions Management (`/manager/sessions`)

1.4.6.1 Purpose Comprehensive management of live learning sessions, including scheduling, hosting, and monitoring.

1.4.6.2 Functional Requirements

- **Session Scheduling**: Calendar-based session planning
- **Host Management**: Instructor assignment and management
- **Participant Tracking**: Real-time participant monitoring
- **Quality Control**: Session quality monitoring and feedback
- **Recording Management**: Session recording and distribution

1.4.6.3 User Stories

```
As a manager
I want to schedule and manage live sessions
So that students have regular learning opportunities

As a manager
I want to monitor session quality
So that I can ensure high-quality learning experiences
```

1.4.6.4 Technical Specifications

```
interface LiveSession {
  id: string;
  title: string;
  description: string;
  instructor: Instructor;
  scheduledTime: Date;
  duration: number;
  maxParticipants: number;
  currentParticipants: number;
  status: SessionStatus;
  quality: QualityMetrics;
}

interface SessionManagement {
  sessions: LiveSession[];
  calendar: CalendarView;
  instructors: Instructor[];
  analytics: SessionAnalytics;
  recordings: Recording[];
}
```

1.4.6.5 API Endpoints

```
// Get live sessions
GET /api/manager/sessions
Query Parameters: date, instructor, status

// Create session
POST /api/manager/sessions

// Update session
PUT /api/manager/sessions/:id

// Get session analytics
GET /api/manager/sessions/:id/analytics

// Manage recordings
GET /api/manager/sessions/:id/recordings
```

1.4.7 7. Feed Management (`/manager/feed`)

1.4.7.1 Purpose Content feed curation and user experience optimization through intelligent content recommendation.

1.4.7.2 Functional Requirements

- **Content Curation**: Manual and automated content selection
- **Recommendation Engine**: AI-powered content recommendations
- **A/B Testing**: Content performance testing
- **Engagement Analytics**: Feed performance monitoring
- **Personalization**: User-specific content feeds

1.4.7.3 User Stories

```
As a manager
I want to curate content feeds
So that users see the most relevant content

As a manager
I want to optimize content recommendations
So that user engagement increases
```

1.4.7.4 Technical Specifications

```
interface FeedManagement {
  feeds: ContentFeed[];
  recommendations: RecommendationEngine;
  testing: ABTesting;
  analytics: FeedAnalytics;
  personalization: PersonalizationRules;
}
```

```
interface ContentFeed {
  id: string;
  name: string;
  type: FeedType;
  content: ContentItem[];
  algorithm: AlgorithmConfig;
  performance: PerformanceMetrics;
}
```

1.4.7.5 API Endpoints

```
// Get content feeds
GET /api/manager/feeds

// Update feed configuration
PUT /api/manager/feeds/:id

// Test feed performance
POST /api/manager/feeds/:id/test

// Get feed analytics
GET /api/manager/feeds/:id/analytics
```

1.4.8 8. Posts Management (`/manager/posts`)

1.4.8.1 Purpose Social learning content management including announcements, discussions, and community engagement.

1.4.8.2 Functional Requirements

- **Post Creation**: Rich post creation with multimedia support
- **Moderation Tools**: Content moderation and approval workflows
- **Engagement Tracking**: Post performance and engagement metrics
- **Community Management**: User interaction and community building
- **Content Scheduling**: Automated post scheduling

1.4.8.3 User Stories

```
As a manager
I want to create engaging posts
So that I can build an active learning community

As a manager
I want to moderate user-generated content
So that I can maintain quality standards
```

1.4.8.4 Technical Specifications

```
interface Post {
  id: string;
  title: string;
  content: string;
  author: string;
  type: PostType;
  status: PostStatus;
  engagement: EngagementMetrics;
  scheduledFor?: Date;
  tags: string[];
}

interface PostManagement {
  posts: Post[];
  moderation: ModerationTools;
  analytics: PostAnalytics;
  scheduling: SchedulingTools;
  community: CommunityMetrics;
}
```

1.4.8.5 API Endpoints

```
// Get posts
GET /api/manager/posts
Query Parameters: type, status, author, date

// Create post
POST /api/manager/posts

// Update post
PUT /api/manager/posts/:id

// Moderate post
POST /api/manager/posts/:id/moderate

// Get post analytics
GET /api/manager/posts/:id/analytics
```

1.4.9 9. Settings Management (`/manager/settings`)

1.4.9.1 Purpose Platform configuration and customization for optimal operational management.

1.4.9.2 Functional Requirements

- **System Configuration**: Platform-wide settings management

- **Team Management**: Team structure and permission management
- **Notification Settings**: Communication and alert configuration
- **Integration Settings**: Third-party service configuration
- **Backup and Recovery**: Data backup and system recovery

1.4.9.3 User Stories

```
As a manager
I want to configure platform settings
So that the system operates optimally

As a manager
I want to manage team permissions
So that team members have appropriate access
```

1.4.9.4 Technical Specifications

```typescript
interface Settings {
  system: SystemConfig;
  team: TeamConfig;
  notifications: NotificationConfig;
  integrations: IntegrationConfig;
  security: SecurityConfig;
}

interface TeamConfig {
  members: TeamMember[];
  roles: Role[];
  permissions: Permission[];
  hierarchy: Hierarchy;
}
```

1.4.9.5 API Endpoints

```
// Get settings
GET /api/manager/settings

// Update settings
PUT /api/manager/settings

// Get team configuration
GET /api/manager/settings/team

// Update team configuration
PUT /api/manager/settings/team
```

1.4.10 10. Profile Management (`/manager/profile`)

1.4.10.1 Purpose

Personal profile and account management for managers.

1.4.10.2 Functional Requirements

- **Profile Information**: Personal details and preferences
- **Account Security**: Password and security settings
- **Notification Preferences**: Personal notification configuration
- **Activity History**: Personal activity tracking
- **Preferences**: Personal interface and workflow preferences

1.4.10.3 User Stories

```
As a manager
I want to manage my profile information
So that my account is up to date

As a manager
I want to control my notification preferences
So that I receive relevant updates
```

1.4.10.4 Technical Specifications

```
interface ManagerProfile {
  id: string;
  personalInfo: PersonalInfo;
  preferences: ManagerPreferences;
  security: SecuritySettings;
  activity: ActivityHistory;
  notifications: NotificationPreferences;
}

interface ManagerPreferences {
  interface: InterfacePreferences;
  workflow: WorkflowPreferences;
  analytics: AnalyticsPreferences;
  communication: CommunicationPreferences;
}
```

1.4.10.5 API Endpoints

```
// Get profile
GET /api/manager/profile

// Update profile
PUT /api/manager/profile
```

```
// Update preferences
PUT /api/manager/profile/preferences

// Update security settings
PUT /api/manager/profile/security
```

1.5 □ Security & Access Control

1.5.1 Role-Based Permissions

```
interface ManagerPermissions {
  junior: {
    content: ['read', 'create_basic'],
    users: ['read_limited'],
    analytics: ['read_basic'],
    settings: ['read_own']
  };
  senior: {
    content: ['read', 'create', 'edit', 'publish'],
    users: ['read', 'manage'],
    analytics: ['read', 'export'],
    settings: ['read', 'edit_team']
  };
  general: {
    content: ['read', 'create', 'edit', 'publish', 'delete'],
    users: ['read', 'manage', 'delete'],
    analytics: ['read', 'export', 'configure'],
    settings: ['read', 'edit', 'configure']
  };
}
```

1.5.2 Security Measures

- **Authentication**: Multi-factor authentication
- **Authorization**: Role-based access control
- **Audit Logging**: Comprehensive activity tracking
- **Data Encryption**: End-to-end encryption
- **Session Management**: Secure session handling

1.6 □ Analytics & Reporting

1.6.1 Key Metrics

- **User Engagement**: Active users, session duration, content consumption
- **Content Performance**: Views, completion rates, user feedback
- **Operational Metrics**: System uptime, response times, error rates
- **Business Metrics**: Revenue, conversion rates, retention

1.6.2 Reporting Capabilities

- **Real-time Dashboards**: Live data visualization
- **Custom Reports**: Configurable report generation
- **Scheduled Reports**: Automated report delivery
- **Export Options**: Multiple format support

1.7 □ Testing & Quality Assurance

1.7.1 Testing Strategy

- **Unit Testing**: Component and function testing
- **Integration Testing**: API and workflow testing
- **User Acceptance Testing**: End-to-end user scenarios
- **Performance Testing**: Load and stress testing

1.7.2 Quality Assurance

- **Code Review**: Peer review processes
- **Automated Testing**: CI/CD pipeline integration
- **Manual Testing**: User experience validation
- **Security Testing**: Vulnerability assessment

1.8 □ Deployment & Operations

1.8.1 Development Workflow

- **Version Control**: Git-based development
- **Code Review**: Pull request workflows
- **Testing**: Automated test suites
- **Deployment**: Staged deployment process

1.8.2 Production Operations

- **Monitoring**: Real-time system monitoring
- **Logging**: Comprehensive log management
- **Backup**: Automated backup procedures
- **Recovery**: Disaster recovery planning

1.9 □ Future Enhancements

1.9.1 Planned Features

- **Advanced Analytics**: Machine learning-powered insights
- **Automation**: Workflow automation tools
- **Integration**: Enhanced third-party integrations
- **Mobile Management**: Mobile app for managers

1.9.2 Technical Improvements

- **Performance**: Optimization and scaling
- **Security**: Enhanced security measures
- **User Experience**: Improved interface design
- **Accessibility**: Better accessibility support

Document Version: 1.0.0 **Last Updated**: January 2025 **Next Review**: March 2025