TCF/TEF Platform - User Section Documentation

TCF/TEF Platform Team

2025-08-18

Contents

1 User Section - Comprehensive Documentation

1.1 □ Overview

The User Section serves as the primary interface for students preparing for TCF/TEF French language proficiency tests. This section provides a comprehensive learning experience with personalized content, interactive tests, live sessions, and progress tracking.

1.2 □ Target Users

- **Primary**: Students preparing for TCF/TEF exams
- **Secondary**: French language learners at all levels
- **Tertiary**: Educational institutions and language schools

1.3 □ Architecture Overview

1.3.1 Technology Stack

- **Frontend**: Next.js 15, React 19, TypeScript
- **Styling**: Tailwind CSS v4, shadcn/ui components
- **State Management**: React hooks, Context API
- **Authentication**: JWT-based authentication
- **Real-time**: WebSocket for live sessions
- **AI Integration**: OpenAI API for personalized assistance

1.3.2 Key Features

- Responsive design for all devices
- Bilingual interface (French/English)
- Progressive Web App capabilities
- Offline content access
- Real-time progress tracking
- AI-powered learning assistance

1.4 □ Page-by-Page Documentation

1.4.1 1. Home/Landing Page (/)

1.4.1.1 Purpose Primary entry point and marketing landing page showcasing platform capabilities and encouraging user engagement.

1.4.1.2 Functional Requirements

- **Hero Section**: Compelling call-to-action with value proposition
- **Course Explorer**: Visual representation of available learning paths
- **Progress Snapshot**: Real-time learning progress display
- **Live Session Banner**: Prominent display of upcoming live sessions

- **AI Assistant**: Floating chat interface for immediate assistance

1.4.1.3 User Stories

```
As a new student
I want to understand what the platform offers
So that I can decide if it meets my learning needs

As a returning student
I want to quickly see my progress and next steps
So that I can continue my learning journey efficiently
```

1.4.1.4 Technical Specifications

```
interface HomePageData {
  userProgress: ProgressSnapshot;
  featuredCourses: Course[];
  liveSessions: LiveSession[];
  achievements: Achievement[];
  recommendations: ContentRecommendation[];
}
```

1.4.1.5 UI/UX Guidelines

- **Hero Section**: Large, engaging imagery with clear value proposition
- **Navigation**: Intuitive menu structure with clear labeling
- **Call-to-Action**: Prominent buttons with contrasting colors
- **Progress Display**: Visual progress indicators with clear metrics
- **Responsive Design**: Optimized for mobile, tablet, and desktop

1.4.2 2. Courses Page (`/cours`)

1.4.2.1 Purpose Comprehensive course catalog allowing users to browse, filter, and enroll in learning content.

1.4.2.2 Functional Requirements

- **Course Catalog**: Grid/list view of available courses
- **Filtering System**: By category, difficulty, duration, and progress
- **Search Functionality**: Full-text search across course content
- **Progress Tracking**: Visual indicators of course completion
- **Enrollment System**: One-click course enrollment

1.4.2.3 User Stories

```
As a student
I want to browse available courses by category
So that I can find content relevant to my learning goals
```

```
As a student
I want to see my progress in each course
So that I can track my learning journey
```

1.4.2.4 Technical Specifications

```
interface Course {
  id: string;
  title: string;
  description: string;
  category: CourseCategory;
  difficulty: DifficultyLevel;
  estimatedDuration: number;
  lessons: Lesson[];
  progress: ProgressData;
  isEnrolled: boolean;
}

interface CourseFilters {
  category?: CourseCategory[];
  difficulty?: DifficultyLevel[];
  duration?: DurationRange;
  progress?: ProgressStatus[];
}
```

1.4.2.5 API Endpoints

```
// Get course catalog
GET /api/courses
Query Parameters: category, difficulty, duration, progress, search

// Get specific course
GET /api/courses/:id

// Enroll in course
POST /api/courses/:id/enroll

// Get course progress
GET /api/courses/:id/progress
```

1.4.3 3. Tests Section (/tests)

1.4.3.1 Purpose Comprehensive test simulation environment for TCF/TEF exam preparation with various test types and difficulty levels.

1.4.3.2 Functional Requirements

- **Test Types**: TCF, TEF, practice tests, diagnostic assessments
- **Duration Options**: 5, 10, 20-minute quick tests
- **Question Types**: Multiple choice, fill-in-blank, listening, writing
- **Real-time Scoring**: Immediate feedback and performance analysis
- **Progress Tracking**: Historical performance and improvement metrics

1.4.3.3 User Stories

```
As a student
I want to take practice tests that simulate real exam conditions
So that I can prepare effectively for the actual test

As a student
I want to see detailed feedback on my test performance
So that I can identify areas for improvement
```

1.4.3.4 Technical Specifications

```
interface Test {
  id: string;
  title: string;
  type: TestType;
  duration: number;
  questions: Question[];
  passingScore: number;
  difficulty: DifficultyLevel;
}

interface TestAttempt {
  id: string;
  testId: string;
  userId: string;
  startTime: Date;
  endTime?: Date;
  answers: Answer[];
  score?: number;
  status: TestStatus;
}
```

1.4.3.5 API Endpoints

```
// Get available tests
GET /api/tests
Query Parameters: type, duration, category

// Start test
POST /api/tests/:id/start
```

```
// Submit test
POST /api/tests/:id/submit

// Get test results
GET /api/tests/:id/results
```

1.4.4 4. Live Sessions (/live)

1.4.4.1 Purpose
Real-time interactive learning sessions with certified instructors and peer collaboration.

1.4.4.2 Functional Requirements

- **Session Scheduling**: Calendar view of upcoming sessions
- **Real-time Video**: High-quality video streaming
- **Interactive Chat**: Text and voice communication
- **Screen Sharing**: Instructor and student screen sharing
- **Recording**: Session recording for later review
- **Participation Tracking**: Attendance and engagement metrics

1.4.4.3 User Stories

```
As a student
I want to join live sessions with native French speakers
So that I can improve my speaking and listening skills

As a student
I want to interact with other learners in real-time
So that I can practice conversation and build confidence
```

1.4.4.4 Technical Specifications

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

interface SessionParticipant {
  userId: string;
  sessionId: string;
  joinTime: Date;
  leaveTime?: Date;
  engagementScore: number;
}
```

1.4.4.5 API Endpoints

```
// Get live sessions
GET /api/live-sessions
Query Parameters: date, instructor, category

// Join session
POST /api/live-sessions/:id/join

// Leave session
POST /api/live-sessions/:id/leave

// Get session recording
GET /api/live-sessions/:id/recording
```

1.4.5 5. Favorites (/favoris)

1.4.5.1 Purpose Personal bookmarking system for users to save and organize preferred content.

1.4.5.2 Functional Requirements

- **Content Bookmarking**: Save courses, tests, and sessions
- **Organization**: Create folders and categories
- **Quick Access**: Easy navigation to saved content
- **Sync**: Cross-device synchronization
- **Sharing**: Share favorite content with other users

1.4.5.3 User Stories

```
As a student
I want to save interesting courses for later review
So that I can build a personalized learning library

As a student
I want to organize my saved content by topic
So that I can easily find what I need when studying
```

1.4.5.4 Technical Specifications

```
interface Favorite {
  id: string;
  userId: string;
  contentId: string;
  contentType: ContentType;
  addedAt: Date;
  folder?: string;
  tags: string[];
}

interface FavoriteFolder {
  id: string;
  userId: string;
  name: string;
  description?: string;
  color: string;
  createdAt: Date;
}
```

1.4.5.5 API Endpoints

```
// Get user favorites
GET /api/favorites
Query Parameters: type, folder, tags

// Add to favorites
POST /api/favorites

// Remove from favorites
DELETE /api/favorites/:id

// Organize favorites
PUT /api/favorites/:id/organize
```

1.4.6 6. Achievements (/achievements)

1.4.6.1 Purpose Gamification system to motivate learning through badges, points, and progress recognition.

1.4.6.2 Functional Requirements

- **Achievement System**: Badges for various accomplishments
- **Progress Tracking**: Visual progress indicators
- **Leaderboards**: Competitive ranking system
- **Rewards**: Unlockable content and features
- **Social Sharing**: Share achievements on social media

1.4.6.3 User Stories

```
As a student
I want to earn badges for completing learning milestones
So that I feel motivated to continue my studies

As a student
I want to see how I rank compared to other learners
So that I can set competitive goals for myself
```

1.4.6.4 Technical Specifications

```
interface Achievement {
  id: string;
  name: string;
  description: string;
  icon: string;
  criteria: AchievementCriteria;
  points: number;
  rarity: AchievementRarity;
  unlockedAt?: Date;
}

interface Leaderboard {
  userId: string;
  username: string;
  totalPoints: number;
  rank: number;
  achievements: Achievement[];
}
```

1.4.6.5 API Endpoints

```
// Get user achievements
GET /api/achievements

// Get leaderboard
GET /api/leaderboard
Query Parameters: timeframe, category

// Check achievement progress
GET /api/achievements/progress
```

1.4.7 7. Notifications (/notifications)

1.4.7.1 Purpose Centralized communication system for platform updates, reminders, and personalized messages.

1.4.7.2 Functional Requirements

- **Real-time Notifications**: Instant delivery of important messages
- **Categorization**: Different types of notifications
- **Read/Unread Status**: Track notification engagement
- **Action Buttons**: Direct actions from notifications
- **Preferences**: Customizable notification settings

1.4.7.3 User Stories

```
As a student
I want to receive reminders about upcoming live sessions
So that I don't miss important learning opportunities

As a student
I want to control which notifications I receive
So that I'm not overwhelmed with irrelevant messages
```

1.4.7.4 Technical Specifications

```
interface Notification {
  id: string;
  userId: string;
  title: string;
  message: string;
  type: NotificationType;
  priority: NotificationPriority;
  isRead: boolean;
  actionUrl?: string;
  actionText?: string;
  createdAt: Date;
  expiresAt?: Date;
}

interface NotificationPreferences {
  userId: string;
  email: boolean;
  push: boolean;
  inApp: boolean;
  categories: NotificationCategory[];
}
```

1.4.7.5 API Endpoints

```
// Get notifications
GET /api/notifications
Query Parameters: type, read, limit

// Mark as read
PUT /api/notifications/:id/read
```

```
// Update preferences
PUT /api/notifications/preferences

// Delete notification
DELETE /api/notifications/:id
```

1.4.8 8. Download (/download)

1.4.8.1 Purpose Offline content access system allowing users to download materials for offline study.

1.4.8.2 Functional Requirements

- **Content Download**: Download courses, tests, and materials
- **Offline Access**: View downloaded content without internet
- **Sync**: Synchronize progress when back online
- **Storage Management**: Monitor and manage download storage
- **Quality Options**: Different quality levels for downloads

1.4.8.3 User Stories

```
As a student
I want to download course materials for offline study
So that I can learn even without internet access

As a student
I want to manage my downloaded content
So that I don't run out of storage space
```

1.4.8.4 Technical Specifications

```
interface Download {
  id: string;
  userId: string;
  contentId: string;
  contentType: ContentType;
  status: DownloadStatus;
  progress: number;
  fileSize: number;
  quality: DownloadQuality;
  downloadedAt: Date;
  expiresAt?: Date;
}

interface DownloadPreferences {
  userId: string;
  autoDownload: boolean;
  wifiOnly: boolean;
  quality: DownloadQuality;
  maxStorage: number;
}
```

1.4.8.5 API Endpoints

```
// Get downloads
GET /api/downloads
Query Parameters: status, type

// Start download
POST /api/downloads/:id

// Cancel download
DELETE /api/downloads/:id

// Get download progress
GET /api/downloads/:id/progress
```

1.4.9 9. Subscription (/abonnement)

1.4.9.1 Purpose Subscription management system for premium features and content access.

1.4.9.2 Functional Requirements

- **Plan Comparison**: Clear comparison of different subscription tiers
- **Payment Processing**: Secure payment handling
- **Billing Management**: Invoice and payment history
- **Feature Access**: Premium feature unlocking
- **Cancellation**: Easy subscription management

1.4.9.3 User Stories

```
As a student
I want to compare different subscription plans
So that I can choose the best option for my needs

As a student
I want to easily manage my subscription
So that I can upgrade, downgrade, or cancel as needed
```

1.4.9.4 Technical Specifications

```
interface Subscription {
  id: string;
  userId: string;
  planId: string;
  status: SubscriptionStatus;
  startDate: Date;
  endDate: Date;
  autoRenew: boolean;
  paymentMethod: PaymentMethod;
  billingCycle: BillingCycle;
}

interface SubscriptionPlan {
  id: string;
  name: string;
  description: string;
  price: number;
  billingCycle: BillingCycle;
  features: Feature[];
  limitations: Limitation[];
}
```

1.4.9.5 API Endpoints

```
// Get subscription plans
GET /api/subscriptions/plans

// Get current subscription
GET /api/subscriptions/current

// Subscribe to plan
POST /api/subscriptions

// Cancel subscription
PUT /api/subscriptions/cancel

// Update payment method
PUT /api/subscriptions/payment-method
```

1.4.10 10. Profile (`/profil`)

1.4.10.1 Purpose User profile management system for personal information, preferences, and account settings.

1.4.10.2 Functional Requirements

- **Profile Information**: Personal details and preferences
- **Avatar Management**: Profile picture upload and management
- **Privacy Settings**: Control over data visibility
- **Account Security**: Password and security settings
- **Learning Preferences**: Customize learning experience

1.4.10.3 User Stories

```
As a student
I want to update my profile information
So that my learning experience is personalized

As a student
I want to control my privacy settings
So that I feel comfortable with my data usage
```

1.4.10.4 Technical Specifications

```
interface UserProfile {
  id: string;
  email: string;
  firstName: string;
  lastName: string;
  avatar?: string;
  language: Language;
  timezone: string;
  preferences: UserPreferences;
  privacy: PrivacySettings;
  createdAt: Date;
  updatedAt: Date;
}

interface UserPreferences {
  theme: Theme;
  notifications: NotificationPreferences;
  learning: LearningPreferences;
  accessibility: AccessibilitySettings;
}
```

1.4.10.5 API Endpoints

```
// Get user profile
GET /api/profile

// Update profile
PUT /api/profile

// Upload avatar
POST /api/profile/avatar

// Update preferences
PUT /api/profile/preferences

// Update privacy settings
```

```
PUT /api/profile/privacy
```

1.5 □ Security & Privacy

1.5.1 Authentication

- JWT-based authentication with refresh tokens
- Multi-factor authentication support
- Session management with automatic logout
- Secure password requirements

1.5.2 Data Protection

- GDPR compliance for user data
- Encrypted data transmission (HTTPS)
- Secure data storage with encryption
- Regular security audits

1.5.3 Privacy Controls

- User consent management
- Data retention policies
- Right to data deletion
- Transparent data usage policies

1.6 □ Analytics & Performance

1.6.1 User Analytics

- Learning progress tracking
- Engagement metrics
- Performance analytics
- A/B testing capabilities

1.6.2 Performance Monitoring

- Page load times
- API response times
- Error tracking
- User experience metrics

1.7 □ Testing Strategy

1.7.1 Unit Testing

- Component testing with React Testing Library
- API endpoint testing
- Utility function testing

1.7.2 Integration Testing

- User workflow testing
- Cross-browser compatibility
- Mobile responsiveness testing

1.7.3 Performance Testing

- Load testing for concurrent users
- Stress testing for peak usage
- Memory and CPU usage monitoring

1.8 □ Deployment

1.8.1 Development Environment

- Local development setup
- Hot reloading
- Environment configuration
- Debug tools

1.8.2 Production Deployment

- CI/CD pipeline
- Environment management
- Monitoring and logging
- Backup and recovery

1.9 □ Future Enhancements

1.9.1 Planned Features

- Advanced AI tutoring
- Virtual reality learning experiences
- Social learning features
- Advanced analytics dashboard
- Mobile app development

1.9.2 Technical Improvements

- Performance optimization
- Enhanced security measures
- Improved accessibility
- Internationalization support

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