TCF/TEF Platform - Technical Documentation

TCF/TEF Platform Team

2025-08-18

Contents

1 Technical Documentation - TCF/TEF Platform

1.1 □ Overview

Comprehensive technical documentation for the TCF/TEF preparation platform covering architecture, database design, API specifications, and deployment procedures.

1.2 □ System Architecture

1.2.1 Technology Stack

- **Frontend**: Next.js 15, React 19, TypeScript, Tailwind CSS v4
- **Backend**: Next.js API Routes, Prisma ORM, PostgreSQL
- **Authentication**: JWT, NextAuth.js
- **Deployment**: Vercel, Docker, AWS
- **Monitoring**: Sentry, Google Analytics

1.2.2 Architecture Diagram

```
User Section → Manager Section → Admin Section
     ↓               ↓               ↓
         Next.js API Gateway
              ↓
         PostgreSQL Database
```

1.3 □ Database Schema

1.3.1 Core Tables

1.3.1.1 Users

```sql
CREATE TABLE users (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  email VARCHAR(255) UNIQUE NOT NULL,
  password_hash VARCHAR(255) NOT NULL,
  first_name VARCHAR(100) NOT NULL,
  last_name VARCHAR(100) NOT NULL,
  role USER_ROLE NOT NULL DEFAULT 'student',
  status USER_STATUS NOT NULL DEFAULT 'active',
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

1.3.1.2 Courses

```sql
CREATE TABLE courses (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  title VARCHAR(255) NOT NULL,
  description TEXT,
  category COURSE_CATEGORY NOT NULL,
  difficulty DIFFICULTY_LEVEL NOT NULL,
  estimated_duration INTEGER NOT NULL,
  is_published BOOLEAN DEFAULT FALSE,
  created_by UUID REFERENCES users(id),
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

1.3.1.3 Tests

```
CREATE TABLE tests (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  title VARCHAR(255) NOT NULL,
  type TEST_TYPE NOT NULL,
  duration INTEGER NOT NULL,
  passing_score INTEGER NOT NULL,
  is_published BOOLEAN DEFAULT FALSE,
  created_by UUID REFERENCES users(id)
);
```

1.3.1.4 User Progress

```
CREATE TABLE user_progress (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id UUID REFERENCES users(id),
  content_id UUID NOT NULL,
  content_type CONTENT_TYPE NOT NULL,
  progress_percentage DECIMAL(5,2) DEFAULT 0,
  completed_at TIMESTAMP,
  score INTEGER
);
```

1.4 □ API Specifications

1.4.1 Authentication

```
POST /api/auth/login
{
  "email": "user@example.com",
  "password": "password123"
}

Response: {
  "token": "jwt_token",
  "user": { "id": "uuid", "role": "student" }
}
```

1.4.2 User Management

```
GET /api/users/profile
Authorization: Bearer <token>

Response: {
  "id": "uuid",
  "email": "user@example.com",
  "role": "student",
  "progress": { "coursesCompleted": 5 }
}
```

1.4.3 Course Management

```
GET /api/courses
Query: { category, difficulty, search, page, limit }

Response: {
  "courses": [{
    "id": "uuid",
    "title": "French Grammar",
    "category": "grammar",
    "progress": { "percentage": 75 }
  }],
  "pagination": { "page": 1, "total": 50 }
}
```

1.4.4 Test Management

```
POST /api/tests/:id/start
Authorization: Bearer <token>

Response: {
  "testId": "uuid",
  "questions": [{
    "id": "uuid",
    "question": "What is correct?",
    "options": ["le", "la", "les"]
  }]
}
```

1.5 □ Security Implementation

1.5.1 JWT Token Structure

```
interface JWTPayload {
  userId: string;
  email: string;
  role: UserRole;
  permissions: Permission[];
  iat: number;
  exp: number;
}
```

1.5.2 Role-Based Permissions

```
const permissions = {
  student: ['read:courses', 'write:test_attempts'],
  junior_manager: ['read:users', 'write:content'],
  senior_manager: ['read:users', 'write:content', 'publish:content'],
  admin: ['read:all', 'write:all', 'delete:all']
};
```

1.5.3 Middleware

```
export function withAuth(handler: NextApiHandler) {
  return async (req: NextApiRequest, res: NextApiResponse) => {
    const token = req.headers.authorization?.replace('Bearer ', '');
    if (!token) return res.status(401).json({ error: 'Unauthorized' });

    try {
      const decoded = jwt.verify(token, process.env.JWT_SECRET!);
      req.user = decoded;
      return handler(req, res);
    } catch (error) {
      return res.status(401).json({ error: 'Invalid token' });
    }
  };
}
```

1.6 Deployment

1.6.1 Environment Variables

```
DATABASE_URL="postgresql://user:password@localhost:5432/tcf_tef_db"
JWT_SECRET="your-jwt-secret"
NEXTAUTH_SECRET="your-nextauth-secret"
AWS_ACCESS_KEY_ID="your-aws-key"
AWS_SECRET_ACCESS_KEY="your-aws-secret"
```

1.6.2 Docker Configuration

```
FROM node:18-alpine
WORKDIR /app
COPY package*.json ./
RUN npm install
```

```
COPY . .
RUN npm run build
EXPOSE 3000
CMD ["npm", "start"]
```

1.6.3 CI/CD Pipeline

```yaml
name: CI/CD
on: [push, pull_request]
jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: actions/setup-node@v3
      - run: npm ci
      - run: npm test
      - run: npm run build
```

1.7 □ Monitoring

1.7.1 Error Tracking

```javascript
import * as Sentry from '@sentry/nextjs';

Sentry.init({
  dsn: process.env.SENTRY_DSN,
  environment: process.env.NODE_ENV,
});
```

1.7.2 Performance Monitoring

```javascript
import { getCLS, getFID, getFCP, getLCP } from 'web-vitals';

export function reportWebVitals() {
  getCLS(console.log);
  getFID(console.log);
  getFCP(console.log);
  getLCP(console.log);
}
```

1.8 □ Testing

1.8.1 Unit Testing

```javascript
import { render, screen } from '@testing-library/react';
import { CourseCard } from '../components/CourseCard';
```

```
test('renders course information', () => {
  render(<CourseCard course={mockCourse} />);
  expect(screen.getByText('French Grammar')).toBeInTheDocument();
});
```

1.8.2 API Testing

```
import { createMocks } from 'node-mocks-http';
import handler from '../pages/api/courses';

test('returns courses list', async () => {
  const { req, res } = createMocks({ method: 'GET' });
  await handler(req, res);
  expect(res._getStatusCode()).toBe(200);
});
```

1.9 □ Performance Optimization

1.9.1 Code Splitting

```
const CourseExplorer = dynamic(() => import('../components/CourseExplorer'), {
  loading: () => <Skeleton />
});
```

1.9.2 Database Optimization

```
const courses = await prisma.course.findMany({
  include: { userProgress: { where: { userId } } },
  where: { isPublished: true },
  take: 20
});
```

1.9.3 Caching

```
const redis = new Redis(process.env.REDIS_URL);

export async function getCachedData(key: string) {
  let data = await redis.get(key);
  if (!data) {
    data = await fetchData();
    await redis.setex(key, 3600, JSON.stringify(data));
  }
  return JSON.parse(data);
}
```

1.10 ▯ Development Guidelines

1.10.1 TypeScript Configuration

```json
{
  "compilerOptions": {
    "target": "ES2020",
    "strict": true,
    "baseUrl": ".",
    "paths": {
      "@/*": ["./*"],
      "@/components/*": ["./components/*"]
    }
  }
}
```

1.10.2 Git Workflow

```
feature/user-authentication
bugfix/login-validation
hotfix/security-patch
```

1.10.3 Commit Messages

```
feat: add user authentication
fix: resolve login issue
docs: update API docs
test: add unit tests
```

Document Version: 1.0.0 **Last Updated**: January 2025