Progress Tracking System Documentation

This documentation explains how to integrate and use the redesigned Progress Tracking System.

Table of Contents

1. Overview
2. File Structure
3. Integration Guide
4. API Reference
5. UI Components
6. Data Structure
7. Troubleshooting

Overview

The Progress Tracking System is designed to track a student's progress through interactive activities, lessons, and assessments. It provides:

- Consistent storage and retrieval of progress data
- Achievement tracking and notifications
- Progress visualization tools
- Detailed statistics on completion status
- Support for multiple activity types

File Structure

The system consists of two main JavaScript files:

1. `progress-tracking.js` - Core tracking functionality
2. `progress-manager-ui.js` - UI components for displaying progress

Both files should be included in your HTML pages where progress tracking is needed.

```html
<script src="js/progress-tracking.js"></script>
<script src="js/progress-manager-ui.js"></script>
```

Integration Guide

Step 1: Initialize the Progress Tracking System

Add this code to your main JavaScript file or in a script tag at the bottom of your pages:

javascript

Copy

```javascript
// Initialize with your configuration
ProgressTracker.init({
  categories: {
    activities: {
      grammarActivities: [
        'mcq', 'gap-fill', 'matching', 'transformation',
        'drag-drop', 'error-correction', 'situations'
      ],
      practiceActivities: [
        'tenses-practice', 'conditionals-practice', 'modals-practice',
        'passive-practice', 'articles-practice', 'prepositions-practice'
      ]
    },
    lessons: {
      grammarLessons: [
        'tenses', 'conditionals', 'modals', 'passive',
        'articles', 'prepositions', 'clauses'
      ],
      specialTopics: [
        'academic-writing', 'business-english', 'pronunciation'
      ]
    },
    assessments: {
      grammarTests: [
        'tenses-test', 'conditionals-test', 'modals-test',
        'passive-test', 'articles-test', 'prepositions-test'
      ],
      comprehensiveTests: [
        'mid-course-test', 'final-test'
      ]
    }
  },
  // Optional: callback when progress changes
  onProgressChange: function(details) {
    console.log('Progress changed:', details);
    // Update UI or notify user if needed
  }
});

// Initialize the UI components (optional)
ProgressManagerUI.init({
  // Use ProgressTracker as the data provider
  progressDataProvider: ProgressTracker.getOverallProgress,
  // Use the same category definitions
```

```
  categoryDefinitions: ProgressTracker.getCategoryDefinitions(),
  // Enable animations
  animateProgressBars: true
});
```

Step 2: Track Completion of Activities

When a student completes an activity (like an MCQ quiz), mark it as completed:

javascript Copy

```
// In your MCQ module completion handler
function onQuizCompleted(score, totalQuestions) {
  ProgressTracker.markItemCompleted('activities', 'mcq-present-tense', {
    score: score,
    maxScore: totalQuestions,
    title: 'Present Tense Quiz',
    answers: {
      'q1': 2, // The student's answers
      'q2': 0,
      'q3': 1,
      // ...etc
    }
  });
}
```

Step 3: Track Lesson Views

When a student views a lesson page:

javascript Copy

```
// On lesson page load
ProgressTracker.updateItemProgress('lessons', 'present-tense', {
  lastViewed: new Date().toISOString(),
  progress: 100, // or a percentage if you're tracking partial completion
  title: 'Present Tense'
});
```

Step 4: Display Progress on Pages

Add progress displays to your pages using either HTML attributes or JavaScript:

HTML Method:

```html
<!-- Simple progress bar -->
<div class="progress-component" data-progress-type="bar"></div>

<!-- Category-specific progress bar -->
<div class="progress-component" data-progress-type="bar" data-category="activities"></div>

<!-- Achievement display -->
<div class="progress-component" data-progress-type="achievement" data-show-all="true"></div>

<!-- Full progress summary -->
<div class="progress-component" data-progress-type="summary"></div>
```

JavaScript Method:

```javascript
// Create a progress bar
ProgressManagerUI.createProgressBar('container-id', {
  category: 'activities',
  animate: true
});

// Create an achievement display
ProgressManagerUI.createAchievementDisplay('container-id', {
  showAll: true
});

// Create a full progress summary
ProgressManagerUI.createProgressSummary('container-id');
```

API Reference

ProgressTracker (Core Module)

Initialization

- `ProgressTracker.init(config)` - Initialize the progress tracking system

Progress Management

- `ProgressTracker.markItemCompleted(category, itemId, details)` - Mark an item as completed
- `ProgressTracker.updateItemProgress(category, itemId, details)` - Update an item's progress

- `ProgressTracker.isItemCompleted(category, itemId)` - Check if an item is completed
- `ProgressTracker.getItemProgress(category, itemId)` - Get details about an item's progress

Statistics and Reporting

- `ProgressTracker.getCompletedCount(category)` - Get count of completed items in a category
- `ProgressTracker.getTotalCount(category)` - Get total count of items in a category
- `ProgressTracker.getCompletionPercentage(category)` - Get completion percentage for a category
- `ProgressTracker.getOverallProgress()` - Get overall progress statistics
- `ProgressTracker.getStats()` - Get usage statistics

Achievement System

- `ProgressTracker.registerAchievement(id, definition)` - Register a new achievement
- `ProgressTracker.getUnlockedAchievements()` - Get all unlocked achievements
- `ProgressTracker.getAllAchievements()` - Get all achievements with unlocked status
- `ProgressTracker.checkAchievements()` - Check for newly unlocked achievements

Data Management

- `ProgressTracker.saveProgress()` - Manually save progress data
- `ProgressTracker.loadProgress()` - Manually load progress data
- `ProgressTracker.resetAllProgress(confirm)` - Reset all progress data
- `ProgressTracker.exportProgressData()` - Export progress data as JSON
- `ProgressTracker.importProgressData(jsonData, merge)` - Import progress data from JSON

ProgressManagerUI (UI Module)

Initialization

- `ProgressManagerUI.init(config)` - Initialize the UI module

Component Creation

- `ProgressManagerUI.createProgressBar(containerId, options)` - Create a progress bar
- `ProgressManagerUI.createAchievementDisplay(containerId, options)` - Create an achievement display
- `ProgressManagerUI.createProgressSummary(containerId)` - Create a progress summary
- `ProgressManagerUI.createActivityGrid(containerId, options)` - Create an activity grid

Updates

- `ProgressManagerUI.updateAllProgressComponents()` - Update all progress components

Data Structure

Progress Data Structure

The core data structure stored in localStorage:

javascript

Copy

```javascript
{
  // Activities stores progress for all interactive elements
  "activities": {
    "mcq-present-tense": {
      "completed": true,
      "completedAt": "2023-05-10T15:30:45.123Z",
      "score": 8,
      "maxScore": 10,
      "title": "Present Tense Quiz",
      "answers": {
        "q1": 2,
        "q2": 0,
        // ...
      }
    },
    // ... more activities
  },

  // Lessons tracks viewed/completed lesson pages
  "lessons": {
    "present-tense": {
      "lastViewed": "2023-05-10T14:20:30.123Z",
      "progress": 100,
      "title": "Present Tense"
    },
    // ... more lessons
  },

  // Assessments tracks formal tests and quizzes
  "assessments": {
    // ... assessment data
  },

  // Stats holds overall usage statistics
  "stats": {
    "totalTimeSpent": 7200,
    "lastActive": "2023-05-10T16:45:12.123Z",
    "dailyStreak": 3,
    "lastActivityDate": "2023-05-10"
  },

  // Achievements tracks unlocked achievements
  "achievements": {
    "first_activity": {
```

```
      "unlocked": true,
      "unlockedAt": "2023-05-08T10:15:22.123Z",
      "title": "First Steps",
      "description": "Complete your first activity",
      "icon": "🏆"
    },
    // ... more achievements
  }
}
```

Category Definitions Structure

javascript Copy

```
{
  "activities": {
    "grammarActivities": [
      "mcq", "gap-fill", "matching", "transformation",
      "drag-drop", "error-correction", "situations"
    ],
    "practiceActivities": [
      "tenses-practice", "conditionals-practice", "modals-practice",
      "passive-practice", "articles-practice", "prepositions-practice"
    ]
  },
  "lessons": {
    // ... lesson categories
  },
  "assessments": {
    // ... assessment categories
  }
}
```

Troubleshooting

Common Issues

1. **Progress not saving**
   - Check that localStorage is available in the browser
   - Check browser privacy settings
   - Try using the `saveProgress()` method manually
2. **Progress bar not updating**
   - Make sure you're calling `updateAllProgressComponents()` after progress changes

- Verify that your progress components have the correct data attributes

3. **Achievements not displaying**
   - Check that achievements are properly registered
   - Verify the criteria functions for achievement unlocking

Debugging

Enable debug mode for more detailed logs:

javascript Copy

```javascript
ProgressTracker.init({
  // ... other config
  debugMode: true
});

ProgressManagerUI.init({
  // ... other config
  debug: true
});
```

Integration with MCQ System

The Progress Tracking System integrates seamlessly with the MCQ system. Here's how to connect them:

javascript

Copy

```javascript
// In your MCQ module
MCQModule.init({
  containerId: 'mcq',
  answers: {
    'q1': 2,
    'q2': 1,
    'q3': 0,
    // ...etc
  },
  onComplete: function(score, total) {
    // Mark the MCQ as completed in the activities category
    ProgressTracker.markItemCompleted('activities', 'mcq-present-tense', {
      score: score,
      maxScore: total,
      completed: true,
      completedAt: new Date().toISOString(),
      title: 'Present Tense Quiz'
    });

    // Also mark the lesson as viewed/completed
    ProgressTracker.updateItemProgress('lessons', 'present-tense', {
      lastViewed: new Date().toISOString(),
      progress: 100,
      title: 'Present Tense'
    });
  }
});
```

This ensures that every time a student completes an MCQ quiz, their progress is tracked and achievements are potentially unlocked.