interface UserAchievement { achievementId: string; // Achievement ID userId: string; // User ID earned: boolean; // Earned flag progress?: number; // Progress towards achievement earnedAt?: number; // Timestamp when earned }

~~~
## 7. Configuration API

### 7.1 System Settings

```typescript
interface SystemConfig {
  game: GameConfig;           // Game configuration
  storage: StorageConfig;     // Storage configuration
  sync: {                     // Sync configuration
    autoSync: boolean;        // Auto sync when online
    syncInterval: number;     // Sync interval in minutes
    conflictResolution: 'manual' | 'server' | 'client'; // Conflict resolution strategy
  };
  content: {                  // Content configuration
    baseUrl?: string;         // Content server URL
    maxSize: number;          // Max content package size
    autoUpdate: boolean;      // Auto update content
  };
  ui: {                       // UI configuration
    theme: string;            // UI theme
    fontSize: number;         // Base font size
    animations: boolean;      // Enable UI animations
    accessibility: {          // Accessibility options
      highContrast: boolean;
      largeText: boolean;
      screenReader: boolean;
    };
  };
  performance: {              // Performance configuration
    targetFps: number;        // Target framerate
    quality: 'low' | 'medium' | 'high'; // Graphics quality
    powerSaving: boolean;     // Power saving mode
  };
}

interface ConfigManager {
  getConfig(): SystemConfig;  // Get current configuration
  updateConfig(updates: Partial<SystemConfig>): void; // Update configuration
  resetToDefaults(): void;    // Reset to defaults
  exportConfig(): string;     // Export configuration
  importConfig(config: string): boolean; // Import configuration
}
~~~

7.2 User Settings

typescript

```typescript
interface UserSettings {
  profile: {                  // User profile settings
    displayName: string;      // Display name
    avatar: string;           // Avatar reference
    status?: string;          // Optional status
  };
  preferences: {              // User preferences
    theme: string;            // UI theme
    notifications: boolean;   // Enable notifications
    sound: boolean;           // Enable sound
    music: boolean;           // Enable music
    volume: number;           // Volume level
  };
  controls: {                 // Control settings
    movementType: 'arrows' | 'wasd' | 'joystick'; // Movement control
    interactionKey: string;   // Interaction key
    touchControls: boolean;   // Enable touch controls
    gamepadEnabled: boolean;  // Enable gamepad
  };
  accessibility: {            // Accessibility settings
    highContrast: boolean;    // High contrast mode
    largeText: boolean;       // Large text mode
    reducedMotion: boolean;   // Reduced motion
    audioCues: boolean;       // Audio cues
    colorblindMode: 'none' | 'protanopia' | 'deuteranopia' | 'tritanopia'; // Colorblind modes
  };
}

interface UserSettingsManager {
  getUserSettings(userId: string): Promise<UserSettings>; // Get user settings
  updateSettings(userId: string, settings: Partial<UserSettings>): Promise<void>; // Update set
  resetSettings(userId: string): Promise<void>; // Reset to defaults
}
```

8. Analytics API

8.1 Usage Analytics

typescript

```typescript
interface AnalyticsEvent {
  type: string;                 // Event type
  userId: string;               // User ID
  timestamp: number;            // Event timestamp
  data: any;                    // Event data
}

interface AnalyticsManager {
  trackEvent(event: AnalyticsEvent): void; // Track analytics event
  getSessionStats(userId: string): SessionStats; // Get current session stats
  getUserStats(userId: string): Promise<UserStats>; // Get user statistics
  getResourceStats(resourceId: string): Promise<ResourceStats>; // Get resource statistics
  exportAnalytics(startDate: number, endDate: number): Promise<AnalyticsEvent[]>; // Export anc
}

interface SessionStats {
  startTime: number;            // Session start time
  duration: number;             // Current duration in seconds
  resourcesViewed: string[];    // Resources viewed this session
  quizzesTaken: string[];       // Quizzes taken this session
  achievements: string[];       // Achievements earned this session
  rooms: {                      // Room visit data
    [roomId: string]: number; // Time spent in each room
  };
}

interface UserStats {
  totalSessions: number;        // Total sessions
  totalTime: number;            // Total time spent
  resourceCompletion: number; // Resource completion percentage
  quizAverage: number;          // Average quiz score
  achievementCount: number;     // Total achievements
  lastActive: number;           // Last activity timestamp
  preferredRooms: {             // Preferred rooms
    roomId: string;             // Room ID
    timeSpent: number;          // Time spent in room
  }[];
}

interface ResourceStats {
  views: number;                // Total views
  completions: number;          // Total completions
  averageTime: number;          // Average time spent
  averageScore?: number;        // Average score (if quiz)
  popularTimes: {               // Popular access times
    hour: number;               // Hour of day
    count: number;              // Access count
    }[];
  }
```

8.2 Learning Analytics

typescript

```
interface LearningAnalytics {
  getUserLearningPath(userId: string): Promise<LearningPath>; // Get user learning path
  getResourceEffectiveness(resourceId: string): Promise<ResourceEffectiveness>; // Get resource
  getLearningGaps(userId: string): Promise<LearningGap[]>; // Get learning gaps
  getRecommendations(userId: string): Promise<Recommendation[]>; // Get personalized recommend
  generateReport(type: ReportType, parameters: any): Promise<Report>; // Generate analytics re
}

interface LearningPath {
  userId: string;              // User ID
  resources: {                 // Resources in path
    resourceId: string;        // Resource ID
    order: number;             // Order in path
    timeSpent: number;         // Time spent
    completed: boolean;        // Completion status
    score?: number;            // Score if applicable
  }[];
  clusters: {                  // Topic clusters
    name: string;              // Cluster name
    resources: string[];       // Resources in cluster
    mastery: number;           // Mastery level (0-1)
  }[];
}

interface ResourceEffectiveness {
  resourceId: string;          // Resource ID
  viewCount: number;           // Total views
  completionRate: number;      // Completion rate
  averageScore?: number;       // Average score if quiz
  timeDistribution: {          // Time spent distribution
    min: number;               // Minimum time
    max: number;               // Maximum time
    avg: number;               // Average time
    median: number;            // Median time
  };
  effectivenessScore: number; // Overall effectiveness score
}

interface LearningGap {
  userId: string;              // User ID
  topic: string;               // Topic area
  resources: string[];         // Related resources
  performance: number;         // Performance level
  recommendations: string[];  // Recommended resources
}

type ReportType = 'user_progress' | 'class_overview' | 'resource_effectiveness' | 'engagement';
```

```
interface Report {
  type: ReportType;           // Report type
  generatedAt: number;        // Generation timestamp
  parameters: any;            // Report parameters
  data: any;                  // Report data
  visualizations: {           // Report visualizations
    type: string;             // Visualization type
    data: any;                // Visualization data
    options?: any;            // Visualization options
  }[];
}
```

9. Classroom Management API

9.1 Class System

typescript

```typescript
interface Classroom {
  id: string;                    // Classroom ID
  name: string;                  // Classroom name
  description: string;           // Classroom description
  teacherId: string;             // Teacher ID
  students: string[];            // Student user IDs
  courses: string[];             // Course IDs
  created: number;               // Creation timestamp
  settings: {                    // Classroom settings
    joinCode?: string;           // Optional join code
    visibility: 'public' | 'private'; // Visibility setting
    chatEnabled: boolean;        // Enable classroom chat
    progressSharing: boolean; // Share progress between students
  };
}

interface ClassroomManager {
  createClassroom(data: Partial<Classroom>): Promise<string>; // Create classroom
  getClassroom(id: string): Promise<Classroom>; // Get classroom
  updateClassroom(id: string, updates: Partial<Classroom>): Promise<void>; // Update classroom
  deleteClassroom(id: string): Promise<void>; // Delete classroom
  addStudent(classId: string, userId: string): Promise<void>; // Add student
  removeStudent(classId: string, userId: string): Promise<void>; // Remove student
  getClassProgress(classId: string): Promise<ClassProgress>; // Get class progress
}

interface ClassProgress {
  classId: string;               // Class ID
  overallProgress: number;       // Overall progress percentage
  studentProgress: {             // Individual student progress
    [userId: string]: {
      overall: number;           // Overall completion percentage
      resources: {               // Resource progress
        [resourceId: string]: {
          completed: boolean; // Completion status
          score?: number;        // Score if applicable
        };
      };
    };
  };
  resourceStats: {               // Resource statistics
    [resourceId: string]: {
      views: number;             // View count
      completions: number;       // Completion count
      averageScore?: number;     // Average score
    };
  };
}
```

9.2 Assignment System

typescript

```
interface Assignment {
  id: string;                 // Assignment ID
  classId: string;            // Class ID
  title: string;              // Assignment title
  description: string;        // Assignment description
  resources: string[];        // Required resources
  dueDate?: number;           // Optional due date
  points: number;             // Point value
  createdAt: number;          // Creation timestamp
  settings: {                 // Assignment settings
    allowLate: boolean;       // Allow late submissions
    minScore?: number;        // Minimum passing score
    attempts?: number;        // Maximum attempts
    timeLimit?: number;       // Time limit in minutes
  };
}

interface AssignmentStatus {
  assignmentId: string;       // Assignment ID
  userId: string;             // User ID
  started: boolean;           // Started flag
  startedAt?: number;         // Start timestamp
  completed: boolean;         // Completion flag
  completedAt?: number;       // Completion timestamp
  score?: number;             // Achieved score
  attempts: number;           // Attempt count
  timeSpent: number;          // Time spent in seconds
  feedback?: string;          // Teacher feedback
}

interface AssignmentManager {
  createAssignment(assignment: Partial<Assignment>): Promise<string>; // Create assignment
  getAssignment(id: string): Promise<Assignment>; // Get assignment
  updateAssignment(id: string, updates: Partial<Assignment>): Promise<void>; // Update assignme
  deleteAssignment(id: string): Promise<void>; // Delete assignment
  getAssignmentStatus(assignmentId: string, userId: string): Promise<AssignmentStatus>; // Get
  submitAssignment(assignmentId: string, userId: string, data: any): Promise<void>; // Submit a
  gradeAssignment(assignmentId: string, userId: string, score: number, feedback?: string): Prom
}
```

10. Extension API

10.1 Plugin System

typescript

```typescript
interface Plugin {
  id: string;                    // Plugin ID
  name: string;                  // Plugin name
  version: string;               // Plugin version
  author: string;                // Plugin author
  description: string;           // Plugin description
  entryPoint: string;            // Plugin entry point
  hooks: {                       // Plugin hooks
    [hookName: string]: Function;
  };
  permissions: string[];         // Required permissions
  config?: any;                  // Plugin configuration
}

interface PluginManager {
  register(plugin: Plugin): Promise<void>; // Register plugin
  unregister(id: string): Promise<void>; // Unregister plugin
  getPlugin(id: string): Plugin; // Get plugin
  getRegisteredPlugins(): Plugin[]; // Get all plugins
  callHook(name: string, args?: any): Promise<any>; // Call plugin hook
  updatePluginConfig(id: string, config: any): Promise<void>; // Update plugin config
}
```

10.2 Customization API

typescript

```typescript
interface CustomizationManager {
  getThemes(): ThemeDefinition[]; // Get available themes
  getCurrentTheme(): string;      // Get current theme
  setTheme(themeId: string): void; // Set theme
  getCustomCSS(): string;         // Get custom CSS
  setCustomCSS(css: string): void; // Set custom CSS
  getRoomTemplates(): RoomTemplate[]; // Get room templates
  createRoom(template: string, data: any): Promise<string>; // Create custom room
  registerAssetPack(pack: AssetPack): Promise<void>; // Register asset pack
}

interface ThemeDefinition {
  id: string;                   // Theme ID
  name: string;                 // Theme name
  description: string;          // Theme description
  colors: {                     // Theme colors
    primary: string;
    secondary: string;
    background: string;
    text: string;
    accent: string;
    [key: string]: string;      // Additional colors
  };
  fonts: {                      // Theme fonts
    main: string;
    heading: string;
    ui: string;
  };
  assets?: {                    // Theme assets
    [key: string]: string;      // Asset references
  };
}

interface RoomTemplate {
  id: string;                   // Template ID
  name: string;                 // Template name
  description: string;          // Template description
  thumbnail: string;            // Template thumbnail
  mapData: any;                 // Base map data
  objects: MapObject[];         // Default objects
  customizableAreas: {          // Customizable areas
    id: string;                 // Area ID
    name: string;               // Area name
    type: string;               // Customization type
    options?: any[];            // Customization options
  }[];
}
```

```
interface AssetPack {
  id: string;                   // Pack ID
  name: string;                 // Pack name
  description: string;          // Pack description
  version: string;              // Pack version
  assets: {                     // Asset definitions
    [key: string]: {
      type: 'image' | 'spritesheet' | 'audio' | 'tileset'; // Asset type
      url: string;              // Asset URL/path
      metadata?: any;           // Asset metadata
    };
  };
}
```

11. Mobile Integration API

11.1 Notification System

typescript

```typescript
interface NotificationConfig {
  enabled: boolean;           // Enable notifications
  types: {                    // Notification types
    [type: string]: boolean;  // Enable/disable by type
  };
  quiet: {                    // Quiet hours
    enabled: boolean;         // Enable quiet hours
    start: number;            // Start hour (0-23)
    end: number;              // End hour (0-23)
  };
}

interface Notification {
  id: string;                 // Notification ID
  userId: string;             // Target user ID
  type: string;               // Notification type
  title: string;              // Notification title
  message: string;            // Notification message
  data?: any;                 // Additional data
  createdAt: number;          // Creation timestamp
  read: boolean;              // Read status
  priority: 'low' | 'normal' | 'high'; // Priority level
  actions?: {                 // Optional actions
    id: string;               // Action ID
    title: string;            // Action title
    data?: any;               // Action data
  }[];
}

interface NotificationManager {
  sendNotification(notification: Partial<Notification>): Promise<string>; // Send notification
  getNotifications(userId: string): Promise<Notification[]>; // Get user notifications
  markAsRead(notificationId: string): Promise<void>; // Mark as read
  deleteNotification(notificationId: string): Promise<void>; // Delete notification
  getConfig(userId: string): Promise<NotificationConfig>; // Get notification config
  updateConfig(userId: string, config: Partial<NotificationConfig>): Promise<void>; // Update c
}
```

11.2 Device Integration

typescript

```typescript
interface DeviceInfo {
  id: string;                    // Device ID
  platform: 'ios' | 'android' | 'web' | 'desktop'; // Platform
  model?: string;                // Device model
  osVersion?: string;            // OS version
  appVersion: string;            // App version
  screenSize: {                  // Screen dimensions
    width: number;
    height: number;
  };
  capabilities: {                // Device capabilities
    camera: boolean;             // Has camera
    microphone: boolean;         // Has microphone
    location: boolean;           // Has location
    vibration: boolean;          // Has vibration
    notification: boolean;       // Has notifications
    orientation: boolean;        // Has orientation
  };
  storage: {                     // Storage info
    total: number;               // Total storage in bytes
    available: number;           // Available storage in bytes
  };
}

interface DeviceManager {
  getDeviceInfo(): DeviceInfo; // Get device info
  registerDevice(userId: string): Promise<void>; // Register device
  unregisterDevice(): Promise<void>; // Unregister device
  requestPermission(permission: string): Promise<boolean>; // Request permission
  checkPermission(permission: string): boolean; // Check permission
  vibrate(pattern?: number | number[]): void; // Vibrate device
  getOrientation(): 'portrait' | 'landscape'; // Get orientation
  getNetworkStatus(): 'online' | 'offline' | 'limited'; // Get network status
  getBatteryLevel(): number; // Get battery level
}
```