Future Enhancements Master Plan

College Algebra Learning Platform - Phases 2-5

Document Version: 1.0
Planning Date: December 15, 2025
Project: Master College Algebra Platform
Current Version: 2.0 (Hint System Complete)
Repository: https://github.com/mltmacster/college_algebra_website

---

Executive Summary

This master plan outlines the structured development, testing, and deployment of 4 major enhancement priorities for the College Algebra Learning Platform. All phases follow established development standards with comprehensive planning, testing protocols, and deployment verification.

Enhancement Priorities Overview

Priority	Enhancement	Estimated Timeline	Complexity	Impact
P1	Advanced Analytics	6-8 weeks	High	High
P2	Content Expansion	8-10 weeks	Medium	High
P3	Gamification 2.0	4-6 weeks	Medium	Medium
P4	Accessibility	6-8 weeks	High	High

Total Development Timeline: 24-32 weeks (6-8 months)

Development Standards & Principles

1. Planning Phase Requirements

- Detailed technical specifications
- Database schema changes documented
- API endpoint specifications
- UI/UX mockups and wireframes
- User story mapping
- Acceptance criteria defined

2. Testing Phase Requirements

- Unit tests (80%+ coverage)
- Integration tests (all API endpoints)
- E2E tests (critical user flows)
- Performance testing (load/stress)
- Accessibility testing (WCAG 2.1 AA)
- Security testing (authentication/authorization)

3. Deployment Phase Requirements

- Build verification (0 TypeScript errors)
- Staging environment testing
- Production deployment checklist
- Rollback procedures documented
- Monitoring and alerting configured
- Post-deployment verification

4. Code Quality Standards

- TypeScript strict mode enabled
- ESLint passing (0 errors)
- Prettier formatting enforced
- Code review required (2 approvers)
- Documentation updated (README, API docs)
- Git commit message standards

Priority 1: Advanced Analytics System

Phase 2.1 - Enhanced Analytics Infrastructure

Timeline: 6-8 weeks
Status: Planning Phase
Complexity: High

Business Objectives

1. **Instructor Insights**: Enable data-driven teaching interventions
2. **Student Success**: Identify at-risk students early
3. **Content Optimization**: Measure effectiveness of hints, problems, and modules
4. **Platform Improvement**: A/B test new features before full rollout

Feature Requirements

1.1 Hint Effectiveness Tracking

User Stories:
- As an instructor, I want to see which hints students use most frequently
- As a content creator, I want to identify hints that aren’t helpful
- As an admin, I want to measure hint impact on problem completion rates

Technical Specifications:

```
// New Prisma Models
model HintUsage {
  id          String   @id @default(cuid())
  userId      String
  problemId   String
  hintIndex   Int
  timestamp   DateTime @default(now())
  wasHelpful  Boolean? // Student feedback
  solvedAfter Boolean  // Did student solve after hint?
  timeToSolve Int?     // Seconds from hint to solution

  user        User     @relation(fields: [userId], references: [id])

  @@index([problemId])
  @@index([userId])
  @@index([timestamp])
}

model HintFeedback {
  id          String   @id @default(cuid())
  hintUsageId String
  rating      Int      // 1-5 stars
  comment     String?
  timestamp   DateTime @default(now())

  hintUsage  HintUsage @relation(fields: [hintUsageId], references: [id])
}
```

API Endpoints:
- `POST /api/analytics/hint-usage` - Track hint usage event
- `GET /api/analytics/hint-effectiveness` - Retrieve aggregated metrics
- `POST /api/analytics/hint-feedback` - Submit hint feedback
- `GET /api/analytics/problem-difficulty` - Calculate difficulty scores

UI Components:
- Hint effectiveness dashboard (instructor view)
- Per-problem hint analytics chart
- Hint helpfulness rating widget (student view)
- Difficulty heatmap visualization

1.2 Problem Difficulty Calibration

User Stories:
- As a content creator, I want to automatically identify problems that are too hard/easy
- As an instructor, I want to see which problems students struggle with most
- As a system admin, I want to rebalance problem difficulty across modules

Technical Specifications:

```
model ProblemDifficultyMetrics {
  id                String   @id @default(cuid())
  problemId         String   @unique
  attempts          Int      @default(0)
  completions       Int      @default(0)
  avgAttempts       Float
  avgTimeToSolve    Int      // seconds
  avgHintsUsed      Float
  successRate       Float    // completions / attempts
  calculatedDiff    String   // easy/medium/hard/expert
  lastCalculated    DateTime @default(now())

  @@index([successRate])
  @@index([calculatedDiff])
}
```

Difficulty Algorithm:

```
function calculateDifficulty(metrics: ProblemDifficultyMetrics): string {
  const score = (
    (1 - metrics.successRate) * 0.4 +           // 40% weight
    (metrics.avgHintsUsed / 5) * 0.3 +          // 30% weight
    (metrics.avgAttempts / 5) * 0.2 +           // 20% weight
    (metrics.avgTimeToSolve / 600) * 0.1        // 10% weight (capped at 10 min)
  );

  if (score < 0.25) return 'easy';
  if (score < 0.5) return 'medium';
  if (score < 0.75) return 'hard';
  return 'expert';
}
```

1.3 Learning Path Optimization

User Stories:
- As a student, I want personalized module recommendations based on my performance
- As an instructor, I want to see optimal learning sequences for struggling students
- As an admin, I want to identify prerequisite knowledge gaps

Technical Specifications:

```
model LearningPathAnalysis {
  id               String   @id @default(cuid())
  userId           String
  currentModule    String
  recommendedNext String[]
  strengthAreas    Json     // { topic: score }
  weaknessAreas    Json     // { topic: score }
  confidence       Float    // 0-1
  lastUpdated      DateTime @default(now())

  user             User     @relation(fields: [userId], references: [id])

  @@unique([userId, currentModule])
}

model PrerequisiteMatrix {
  id             String @id @default(cuid())
  moduleId       String
  prerequisite   String
  strength       Float  // 0-1, how important is this prerequisite

  @@unique([moduleId, prerequisite])
}
```

Machine Learning Integration:
- Recommendation engine using student performance data
- Collaborative filtering for similar student paths
- Knowledge graph for prerequisite relationships
- Confidence scoring for recommendations

1.4 A/B Testing Framework

User Stories:
- As a product manager, I want to test new features with a subset of users
- As a developer, I want to measure feature impact before full rollout
- As an instructor, I want to compare teaching methodologies

Technical Specifications:

```
model ABTest {
  id          String   @id @default(cuid())
  name        String   @unique
  description String
  startDate   DateTime
  endDate     DateTime?
  isActive    Boolean  @default(true)
  variants    Json     // [{ id: 'A', name: 'Control', weight: 0.5 }, ...]
  targetUsers String[] // empty = all users
  metrics     Json     // { primary: 'completion_rate', secondary: [...] }
  results     Json?

  @@index([isActive])
}

model ABTestAssignment {
  id        String   @id @default(cuid())
  testId    String
  userId    String
  variant   String
  assignedAt DateTime @default(now())

  test      ABTest   @relation(fields: [testId], references: [id])
  user      User     @relation(fields: [userId], references: [id])

  @@unique([testId, userId])
}

model ABTestEvent {
  id        String   @id @default(cuid())
  testId    String
  userId    String
  variant   String
  eventType String   // 'view', 'click', 'complete', etc.
  eventData Json?
  timestamp DateTime @default(now())

  @@index([testId, variant])
  @@index([userId])
}
```

A/B Testing API:

```
// Assign user to variant
POST /api/ab-test/assign
{ testId: string, userId: string }
→ { variant: 'A' | 'B' | 'C' }

// Track event
POST /api/ab-test/event
{ testId, userId, eventType, eventData }

// Get results
GET /api/ab-test/:testId/results
→ { variants: [{ id, metrics, significance }] }
```

Database Migration Plan

Step 1: Schema Updates

```
# Create new models
npx prisma migrate dev --name add_analytics_tables

# Tables to add:
- HintUsage
- HintFeedback
- ProblemDifficultyMetrics
- LearningPathAnalysis
- PrerequisiteMatrix
- ABTest
- ABTestAssignment
- ABTestEvent
```

Step 2: Data Backfill

```
// scripts/backfill-analytics.ts
// Backfill historical data from existing tables
// Calculate initial difficulty metrics from ProblemAttempt records
```

Step 3: Index Optimization

```sql
CREATE INDEX idx_hint_usage_problem ON "HintUsage"("problemId", "timestamp");
CREATE INDEX idx_problem_metrics_success ON "ProblemDifficultyMetrics"("successRate");
CREATE INDEX idx_ab_test_active ON "ABTest"("isActive", "startDate");
```

API Development

New Endpoints (11 total)

1. **Hint Analytics**
   - `POST /api/analytics/hint-usage`
   - `GET /api/analytics/hint-effectiveness`
   - `POST /api/analytics/hint-feedback`

2. **Problem Calibration**
   - `GET /api/analytics/problem-difficulty`
   - `POST /api/analytics/recalculate-difficulty`
   - `GET /api/analytics/difficulty-heatmap`

3. **Learning Paths**
   - `GET /api/analytics/learning-path/:userId`
   - `GET /api/analytics/recommendations/:userId`

4. **A/B Testing**
   - `POST /api/ab-test/assign`
   - `POST /api/ab-test/event`
   - `GET /api/ab-test/:testId/results`

UI Components Development

1. Enhanced Instructor Dashboard

File: `/app/components/analytics/enhanced-instructor-portal.tsx`

Features:
- Real-time analytics with auto-refresh

- Hint effectiveness charts (Chart.js)
- Problem difficulty heatmap
- Student learning path visualization
- At-risk student alerts
- Exportable reports (CSV, PDF)

2. A/B Test Manager

File: `/app/components/analytics/ab-test-manager.tsx`

Features:
- Create/configure tests
- Monitor active tests
- Statistical significance calculator
- Results dashboard with confidence intervals
- Variant comparison charts

3. Student Insights Widget

File: `/app/components/analytics/student-insights.tsx`

Features:
- Personal learning path recommendations
- Strength/weakness breakdown
- Suggested next modules
- Progress comparison with peers (anonymized)

Testing Strategy

Unit Tests (Target: 85% coverage)

```typescript
// __tests__/analytics/hint-tracking.test.ts
describe('Hint Usage Tracking', () => {
  it('should record hint usage event', async () => {
    // Test hint tracking API
  });

  it('should calculate hint effectiveness', () => {
    // Test algorithm
  });

  it('should aggregate hint metrics by problem', () => {
    // Test aggregation
  });
});

// __tests__/analytics/difficulty-calibration.test.ts
describe('Problem Difficulty Calibration', () => {
  it('should calculate difficulty score correctly', () => {
    const metrics = {
      successRate: 0.6,
      avgHintsUsed: 2.5,
      avgAttempts: 1.8,
      avgTimeToSolve: 180
    };
    expect(calculateDifficulty(metrics)).toBe('medium');
  });

  it('should recalibrate when metrics change', () => {
    // Test recalibration logic
  });
});

// __tests__/analytics/ab-testing.test.ts
describe('A/B Testing Framework', () => {
  it('should assign users to variants consistently', () => {
    // Test variant assignment
  });

  it('should calculate statistical significance', () => {
    // Test chi-square test
  });
});
```

Integration Tests

```
// __tests__/integration/analytics-api.test.ts
describe('Analytics API Integration', () => {
  it('should track hint usage end-to-end', async () => {
    // Student requests hint
    // Usage event recorded
    // Metrics updated
    // Dashboard reflects change
  });

  it('should update difficulty metrics after problem completion', async () => {
    // Student completes problem
    // Difficulty metrics recalculated
    // Heatmap updated
  });
});
```

Performance Tests

```
// __tests__/performance/analytics-load.test.ts
describe('Analytics Performance', () => {
  it('should handle 1000 concurrent hint tracking requests', async () => {
    // Load test hint tracking API
    // Response time < 200ms
  });

  it('should generate dashboard data in < 1 second', async () => {
    // Test dashboard query performance
  });
});
```

Deployment Plan

Phase 2.1.1: Database Migration (Week 1)

- Create analytics tables in staging
- Backfill historical data
- Verify indexes and performance
- Deploy to production (off-peak hours)

Phase 2.1.2: Backend API (Weeks 2-3)

- Implement hint tracking endpoints
- Deploy to staging
- Integration testing
- Deploy to production (gradual rollout)

Phase 2.1.3: Analytics Dashboard (Weeks 4-5)

- Build instructor dashboard components
- User acceptance testing
- Deploy to production

Phase 2.1.4: A/B Testing Framework (Weeks 6-7)

- Implement A/B test infrastructure
- Internal testing with 10% of users

- Full rollout

Phase 2.1.5: Verification & Documentation (Week 8)

- Performance verification
- Documentation updates
- Training materials for instructors

Success Metrics

Technical Metrics:
- ✅ 0 TypeScript errors
- ✅ 85%+ test coverage
- ✅ < 200ms API response time (p95)
- ✅ < 1s dashboard load time
- ✅ 0 critical bugs in production

Business Metrics:
- ✅ 90%+ instructor adoption of analytics dashboard
- ✅ 20% improvement in at-risk student identification
- ✅ 15% reduction in problem completion time (with optimized hints)
- ✅ 3+ successful A/B tests completed

Risk Assessment

High Risk:
- **Database Performance**: Analytics queries may be slow
- Mitigation: Aggressive indexing, query optimization, caching
- **Data Privacy**: Student analytics must comply with FERPA
- Mitigation: Anonymization, role-based access, audit logging

Medium Risk:
- **A/B Test Complexity**: Statistical analysis requires expertise
- Mitigation: Use established libraries (statsig, optimizely)
- **Dashboard Complexity**: Too many metrics may overwhelm instructors
- Mitigation: Progressive disclosure, default views, training

Low Risk:
- **Hint Tracking**: Straightforward implementation
- **Difficulty Calibration**: Well-defined algorithm

Priority 2: Content Expansion

Phase 3.1 - Comprehensive Content Development

Timeline: 8-10 weeks
Status: Planning Phase
Complexity: Medium

Business Objectives

1. **Increased Practice**: Expand from 30 to 80+ practice problems
2. **Multi-Modal Learning**: Add video explanations for visual learners
3. **Interactive Tools**: Enable hands-on graphing and simulation

4. **Real-World Context**: Develop comprehensive business case studies

Feature Requirements

2.1 Additional Practice Problems (50+ new problems)

Distribution:
- Linear Equations: 5 → 12 problems (+7)
- Functions & Graphing: 5 → 15 problems (+10)
- Quadratic Functions: 5 → 12 problems (+7)
- Exponential Functions: 5 → 15 problems (+10)
- Matrix Operations: 5 → 12 problems (+7)
- Sequences & Probability: 5 → 14 problems (+9)

Total: 80 problems (50 new)

Problem Categories:
1. **Foundational** (30%): Basic skill building
2. **Applied Business** (50%): Real-world scenarios
3. **Challenge** (20%): Advanced problem-solving

Content Development Process:

```
Week 1-2: Problem ideation and business context research
Week 3-4: Problem authoring (15 problems/week target)
Week 5-6: Peer review and refinement
Week 7: Hint development (3-5 hints per problem)
Week 8: Implementation and testing
```

Technical Implementation:

```
// Extend interactive-problems.ts
export const linearEquationsProblemsExpanded: Problem[] = [
  ...linearEquationsProblems, // existing 5
  {
    id: 'linear-006',
    title: 'Subscription Service Cost Analysis',
    // ... new problem
  },
  // ... 6 more new problems
];
```

2.2 Video Explanations

Video Content Plan:
- **Total Videos**: 30-40 videos
- **Duration**: 5-15 minutes each
- **Format**: Screen recording with voiceover
- **Topics**: Module concepts, problem walkthroughs, business applications

Video Categories:
1. **Concept Explainers** (18 videos)
- One per major topic (3 per module)

- Abstract concepts made concrete
- Business context examples

1. **Problem Walkthroughs** (12 videos)
   - Selected challenge problems
   - Step-by-step solutions
   - Common pitfalls highlighted

2. **Business Case Studies** (6 videos)
   - Real company examples
   - Decision-making scenarios
   - Industry applications

3. **Quick Tips** (4 videos)
   - Calculator usage
   - Formula memorization
   - Test-taking strategies

Technical Implementation:

```
// New model
model VideoResource {
  id          String   @id @default(cuid())
  title       String
  description String
  moduleId    String?
  problemId   String?
  duration    Int      // seconds
  videoUrl    String   // S3 or YouTube
  thumbnailUrl String
  transcript  String?  // For accessibility
  views       Int      @default(0)
  createdAt   DateTime @default(now())

  module      LearningModule? @relation(fields: [moduleId], references: [id])
}

// Component
// components/video-player.tsx
export function VideoPlayer({ videoId }: { videoId: string }) {
  // Video.js or React Player integration
  // Playback tracking for analytics
  // Transcript overlay option
}
```

Video Production Workflow:
1. Script writing and review
2. Screen recording (OBS Studio)
3. Voiceover recording (professional mic)
4. Editing (DaVinci Resolve or Premiere)
5. Captioning (automated + manual review)
6. Upload to hosting (AWS S3 + CloudFront)
7. Integration testing

2.3 Interactive Graphing Tools

User Stories:
- As a student, I want to graph functions and see real-time changes
- As a student, I want to explore break-even points visually
- As an instructor, I want students to develop graphing intuition

Technical Specifications:

```
// Integration with Desmos API or custom D3.js implementation

// components/interactive-graph.tsx
import { Calculator } from '@desmos/calculator';

export function InteractiveGraph({
  equation,
  domain,
  annotations
}: GraphProps) {
  // Render graph with Desmos
  // Add business context annotations
  // Allow student to adjust parameters
  // Show impact on business metrics
}

// Example usage in Functions module
<InteractiveGraph
  equation="y = 4x"
  domain={{ x: [0, 100], y: [0, 400] }}
  annotations={[
    { point: [80, 320], label: 'Break-even: 80 cups' },
    { line: 'y = 200 + 1.5x', color: 'red', label: 'Total Cost' }
  ]}
/>
```

Features:
- Drag-and-drop points
- Real-time equation updates
- Zoom and pan
- Export as image
- Share configurations
- Mobile-responsive

Modules Using Interactive Graphs:
- Functions & Graphing (primary)
- Quadratic Functions (parabolas)
- Exponential Functions (growth curves)
- Linear Equations (break-even analysis)

2.4 Business Case Studies

Case Study Structure:

```
interface CaseStudy {
  id: string;
  title: string;
  company: string; // Real or fictional
  industry: string;
  scenario: string; // 300-500 words
  challenges: string[];
  objectives: string[];
  datasets: DataFile[]; // CSV, JSON
  questions: CaseQuestion[];
  resources: Resource[];
  estimatedTime: number; // minutes
}

interface CaseQuestion {
  id: string;
  question: string;
  type: 'calculation' | 'analysis' | 'recommendation';
  hints: string[];
  rubric: GradingRubric;
}
```

Case Studies to Develop (6 total):

1. **Coffee Chain Expansion** (Linear Equations)
   - Fixed costs, variable costs, break-even analysis
   - Multiple location scenarios
   - Profitability projections

2. **SaaS Pricing Optimization** (Functions)
   - Demand curves
   - Revenue maximization
   - Tiered pricing models

3. **Manufacturing Investment** (Quadratic Functions)
   - Cost functions with economies of scale
   - Optimal production volume
   - ROI calculations

4. **Startup Growth Modeling** (Exponential Functions)
   - User acquisition curves
   - Revenue forecasting
   - Burn rate analysis

5. **Portfolio Optimization** (Matrix Operations)
   - Asset allocation
   - Risk-return tradeoffs
   - Rebalancing strategies

6. **Inventory Management** (Sequences)
   - Seasonal demand patterns
   - Ordering strategies
   - Cost minimization

Implementation:

```
// components/case-study-viewer.tsx
export function CaseStudyViewer({ caseStudyId }: Props) {
  return (
    <div className="case-study-container">
      <CaseHeader />
      <ScenarioDescription />
      <DataExplorer /> {/* Interactive data tables */}
      <QuestionSequence /> {/* Guided questions */}
      <ResourceLibrary /> {/* Videos, articles, tools */}
      <SubmissionArea /> {/* Student work */}
      <PeerReview /> {/* Optional peer feedback */}
    </div>
  );
}
```

Database Schema Additions

```
model ProblemExpanded {
  id               String   @id @default(cuid())
  legacyId         String?  // Reference to original interactive-problems.ts
  moduleId         String
  title            String
  description      String
  businessContext  String
  problemStatement String
  type             String
  difficulty       String
  category         String   // foundational, applied, challenge
  points           Int
  hints            Json     // Array of hint objects
  solution         Json
  choices          Json?
  steps            Json?
  videoId          String?
  graphConfig      Json?    // For interactive graphs
  createdAt        DateTime @default(now())
  updatedAt        DateTime @updatedAt

  module           LearningModule @relation(fields: [moduleId], references: [id])
  video            VideoResource? @relation(fields: [videoId], references: [id])

  @@index([moduleId])
  @@index([difficulty])
  @@index([category])
}

model CaseStudy {
  id               String   @id @default(cuid())
  title            String
  company          String
  industry         String
  scenario         String   @db.Text
  challenges       Json
  objectives       Json
  datasets         Json
  questions        Json
  resources        Json
  estimatedTime    Int
  moduleIds        String[] // Can span multiple modules
  difficulty       String
  createdAt        DateTime @default(now())
  updatedAt        DateTime @updatedAt

  @@index([difficulty])
}

model CaseStudySubmission {
  id             String   @id @default(cuid())
  caseStudyId    String
  userId         String
  answers        Json
  status         String   // draft, submitted, graded
  score          Float?
  feedback       String?
  submittedAt    DateTime?
  gradedAt       DateTime?
  createdAt      DateTime @default(now())

  caseStudy      CaseStudy @relation(fields: [caseStudyId], references: [id])
  user          User      @relation(fields: [userId], references: [id])

  @@unique([caseStudyId, userId])
}
```

Testing Strategy

Content Quality Testing

```
// __tests__/content/problem-validation.test.ts
describe('New Problem Content', () => {
  it('should have correct business context', () => {
    // Verify each problem has realistic business scenario
  });

  it('should have 3-5 progressive hints', () => {
    // Verify hint count and progression
  });

  it('should have complete solutions', () => {
    // Verify solution steps and final answers
  });
});
```

Video Integration Testing

```
// __tests__/integration/video-player.test.ts
describe('Video Player Integration', () => {
  it('should load video and display correctly', async () => {
    // Test video loading
  });

  it('should track playback for analytics', () => {
    // Test tracking events
  });

  it('should display transcript when enabled', () => {
    // Test accessibility feature
  });
});
```

Interactive Graph Testing

```
// __tests__/components/interactive-graph.test.ts
describe('Interactive Graphing Tool', () => {
  it('should render equation correctly', () => {
    // Test Desmos integration
  });

  it('should update when parameters change', () => {
    // Test reactivity
  });

  it('should export graph as image', () => {
    // Test export functionality
  });
});
```

Deployment Plan

Phase 3.1.1: Problem Database Expansion (Weeks 1-2)

- Migrate from static file to database storage
- Implement problem CMS for easier management
- Deploy expanded problem set

Phase 3.1.2: Video Infrastructure (Weeks 3-4)

- Set up AWS S3 + CloudFront for video hosting
- Implement video player component
- Deploy first 10 videos

Phase 3.1.3: Interactive Graphing (Weeks 5-6)

- Integrate Desmos API
- Build custom graph components
- Deploy to Functions & Graphing module

Phase 3.1.4: Case Studies (Weeks 7-9)

- Develop case study infrastructure
- Publish first 3 case studies
- Collect student feedback

Phase 3.1.5: Final Integration (Week 10)

- Complete remaining content
- Full platform testing
- Documentation updates

Success Metrics

Content Metrics:
- ✅ 80+ total practice problems (50 new)
- ✅ 30+ video resources published
- ✅ 6 interactive graphing modules
- ✅ 6 comprehensive case studies

Engagement Metrics:
- ✅ 40% increase in practice problem attempts
- ✅ 60% of students watch at least 1 video per module
- ✅ 80%+ positive feedback on new content
- ✅ 25% improvement in module completion rates

Priority 3: Gamification 2.0

Phase 4.1 - Advanced Gamification Features

Timeline: 4-6 weeks
Status: Planning Phase
Complexity: Medium

Business Objectives

1. **Increased Engagement**: Drive daily active users through competition

2. **Community Building**: Foster peer learning and support
3. **Motivation**: Reward consistent effort and achievement
4. **Retention**: Reduce dropout rates through social features

Feature Requirements

3.1 Leaderboards

User Stories:
- As a student, I want to see how I rank compared to my classmates
- As a competitive learner, I want to chase the top spot
- As an instructor, I want to leverage healthy competition

Technical Specifications:

```
model Leaderboard {
  id          String   @id @default(cuid())
  name        String
  type        String   // global, class, module, weekly
  scope       String?  // classId or moduleId
  metric      String   // points, problems_solved, streak, etc.
  period      String   // all_time, monthly, weekly
  isActive    Boolean  @default(true)
  createdAt   DateTime @default(now())

  @@index([type, scope, period])
}

model LeaderboardEntry {
  id              String   @id @default(cuid())
  leaderboardId   String
  userId          String
  rank            Int
  score           Float
  previousRank    Int?
  updatedAt       DateTime @default(now())

  leaderboard     Leaderboard @relation(fields: [leaderboardId], references: [id])
  user            User        @relation(fields: [userId], references: [id])

  @@unique([leaderboardId, userId])
  @@index([leaderboardId, rank])
}

model UserScore {
  id                String   @id @default(cuid())
  userId            String
  totalPoints       Int      @default(0)
  problemsSolved    Int      @default(0)
  currentStreak     Int      @default(0)
  longestStreak     Int      @default(0)
  lastActivityDate  DateTime @default(now())
  level             Int      @default(1)
  experiencePoints  Int      @default(0)

  user              User     @relation(fields: [userId], references: [id])

  @@unique([userId])
  @@index([totalPoints])
  @@index([level])
}
```

Leaderboard Types:

1. **Global Leaderboard**
   - All-time top performers
   - Updated hourly
   - Top 100 displayed

2. **Class Leaderboard**
   - Per-class competition
   - Instructor can enable/disable
   - Privacy controls

3. **Module Leaderboard**
   - Per-module mastery
   - Encourages comprehensive learning

4. **Weekly Challenge**
   - Resets every Monday
   - Special rewards for top 10

UI Components:

```
// components/gamification/leaderboard-widget.tsx
export function LeaderboardWidget({ type, scope }: Props) {
  return (
    <Card>
      <CardHeader>
        <Trophy className="h-6 w-6 text-yellow-500" />
        <h3>Top Performers This Week</h3>
      </CardHeader>
      <CardContent>
        <LeaderboardTable />
        <YourRank /> {/* Highlighted */}
        <ViewFullLeaderboard />
      </CardContent>
    </Card>
  );
}
```

3.2 Team Challenges

User Stories:
- As a student, I want to collaborate with peers on challenges
- As an instructor, I want to promote teamwork and peer learning
- As a team member, I want to contribute to group success

Technical Specifications:

```
model Team {
  id          String   @id @default(cuid())
  name        String
  description String?
  avatarUrl   String?
  createdBy   String
  maxMembers  Int      @default(5)
  isActive    Boolean  @default(true)
  totalPoints Int      @default(0)
  createdAt   DateTime @default(now())

  creator     User     @relation("TeamCreator", fields: [createdBy], references: [id])
  members     TeamMember[]
  challenges  TeamChallengeProgress[]

  @@index([totalPoints])
}

model TeamMember {
  id        String   @id @default(cuid())
  teamId    String
  userId    String
  role      String   // owner, member
  points    Int      @default(0)
  joinedAt  DateTime @default(now())

  team      Team     @relation(fields: [teamId], references: [id])
  user      User     @relation(fields: [userId], references: [id])

  @@unique([teamId, userId])
}

model Challenge {
  id          String   @id @default(cuid())
  title       String
  description String
  type        String   // solo, team
  difficulty  String
  startDate   DateTime
  endDate     DateTime
  goals       Json     // [{ metric, target, points }]
  rewards     Json     // { badges, points, unlocks }
  isActive    Boolean  @default(true)

  @@index([isActive, startDate])
}

model TeamChallengeProgress {
  id            String   @id @default(cuid())
  teamId        String
  challengeId   String
  progress      Json     // { metric: current_value }
  isCompleted   Boolean  @default(false)
  completedAt   DateTime?

  team          Team     @relation(fields: [teamId], references: [id])
  challenge     Challenge @relation(fields: [challengeId], references: [id])

  @@unique([teamId, challengeId])
}
```

Challenge Types:

1. **Speed Challenges**
   - Solve 10 problems in 30 minutes
   - Team members contribute to shared goal
2. **Mastery Challenges**
   - Achieve 90%+ accuracy on module
   - All team members must complete
3. **Collaboration Challenges**
   - Share hints with teammates
   - Review each other's solutions
4. **Module Marathon**
   - Complete entire module as team
   - Bonus for synchronized completion

Team UI:

```
// pages/teams.tsx
export default function TeamsPage() {
  return (
    <>
      <TeamList />
      <CreateTeamButton />
      <ActiveChallenges />
      <TeamLeaderboard />
    </>
  );
}

// components/gamification/team-card.tsx
export function TeamCard({ team }: Props) {
  return (
    <Card>
      <TeamAvatar />
      <TeamName />
      <MemberCount />
      <TotalPoints />
      <ActiveChallenges />
      <JoinButton />
    </Card>
  );
}
```

3.3 Streak Tracking

User Stories:
- As a student, I want to build daily learning habits
- As a motivated learner, I don't want to break my streak
- As an instructor, I want to encourage consistent practice

Technical Specifications:

```
model StreakData {
  id               String   @id @default(cuid())
  userId           String   @unique
  currentStreak    Int      @default(0)
  longestStreak    Int      @default(0)
  lastActiveDate   DateTime
  streakStartDate DateTime?
  freezesUsed      Int      @default(0) // Streak protection
  freezesAvailable Int      @default(2)

  user             User     @relation(fields: [userId], references: [id])
}

model DailyActivity {
  id                 String   @id @default(cuid())
  userId             String
  date               DateTime @db.Date
  problemsSolved     Int      @default(0)
  timeSpent          Int      @default(0) // seconds
  pointsEarned       Int      @default(0)
  hintsUsed          Int      @default(0)
  isQualifyingDay    Boolean  @default(false) // Met minimum activity

  user               User     @relation(fields: [userId], references: [id])

  @@unique([userId, date])
  @@index([userId, date])
}
```

Streak Mechanics:
- **Qualifying Day**: Solve at least 3 problems OR spend 20+ minutes
- **Streak Freeze**: Use 1 freeze to protect 1 missed day (max 2 per month)
- **Milestones**: Special badges at 7, 30, 100, 365 days
- **Recovery**: 1-day grace period (streak saved if active next day)

Streak UI:

```
// components/gamification/streak-widget.tsx
export function StreakWidget() {
  return (
    <Card>
      <div className="flex items-center">
        <Flame className="h-8 w-8 text-orange-500" />
        <div>
          <h3 className="text-3xl font-bold">{currentStreak} Days</h3>
          <p className="text-sm text-gray-600">Current Streak</p>
        </div>
      </div>
      <StreakCalendar /> {/* Visual calendar */}
      <StreakMilestones />
      <FreezesAvailable count={2} />
    </Card>
  );
}
```

3.4 Achievement Sharing

User Stories:
- As a proud student, I want to share my achievements on social media

- As a motivated learner, I want recognition from my network
- As a platform, I want organic growth through social sharing

Technical Specifications:

```
model SharedAchievement {
  id              String   @id @default(cuid())
  userId          String
  achievementType String   // badge, streak, leaderboard, completion
  achievementData Json
  shareUrl        String   @unique
  imageUrl        String   // Generated share card
  views           Int      @default(0)
  createdAt       DateTime @default(now())
  expiresAt       DateTime? // Optional expiration

  user            User     @relation(fields: [userId], references: [id])

  @@index([shareUrl])
}
```

Share Card Generation:

```
// lib/share-card-generator.ts
import { createCanvas } from 'canvas';

export async function generateShareCard(
  achievement: Achievement
): Promise<string> {
  const canvas = createCanvas(1200, 630); // Open Graph size
  const ctx = canvas.getContext('2d');

  // Draw background gradient
  // Add achievement icon
  // Add user name
  // Add achievement details
  // Add platform branding

  const buffer = canvas.toBuffer('image/png');
  const imageUrl = await uploadToS3(buffer);
  return imageUrl;
}
```

Sharing Options:
- **Twitter/X**: Pre-filled tweet with image
- **LinkedIn**: Professional achievement post
- **Facebook**: Share to timeline
- **Email**: Share with instructor/advisor
- **Copy Link**: Share anywhere

Share Card Templates:
1. **Badge Unlock**: “I earned the [Badge Name] badge!”
2. **Streak Milestone**: “I maintained a [X]-day learning streak!”
3. **Leaderboard**: “I ranked #[X] on the [Leaderboard Name]!”
4. **Module Completion**: “I completed [Module Name]!”
5. **Team Victory**: “Our team conquered [Challenge Name]!”

Testing Strategy

Gamification Logic Testing

```
// __tests__/gamification/streak-tracking.test.ts
describe('Streak Tracking', () => {
  it('should increment streak on qualifying day', () => {
    // Test streak increment logic
  });

  it('should reset streak after 2 missed days', () => {
    // Test streak reset
  });

  it('should apply streak freeze correctly', () => {
    // Test freeze mechanic
  });
});

// __tests__/gamification/leaderboard.test.ts
describe('Leaderboard System', () => {
  it('should rank users by total points', () => {
    // Test ranking algorithm
  });

  it('should update ranks when scores change', () => {
    // Test real-time updates
  });

  it('should handle ties correctly', () => {
    // Test tie-breaking rules
  });
});
```

Team Functionality Testing

```
// __tests__/gamification/teams.test.ts
describe('Team Challenges', () => {
  it('should allow team creation', async () => {
    // Test team creation flow
  });

  it('should aggregate team member progress', () => {
    // Test progress aggregation
  });

  it('should complete challenge when goal met', () => {
    // Test challenge completion
  });
});
```

Deployment Plan

Phase 4.1.1: Leaderboards (Weeks 1-2)

- Implement scoring system
- Build leaderboard infrastructure
- Deploy global and class leaderboards

Phase 4.1.2: Streak Tracking (Week 3)

- Implement daily activity tracking
- Build streak calculation logic
- Deploy streak widgets

Phase 4.1.3: Team Challenges (Weeks 4-5)

- Implement team management
- Build challenge system
- Deploy first 3 challenges

Phase 4.1.4: Achievement Sharing (Week 6)

- Build share card generator
- Implement social sharing
- Deploy sharing features

Success Metrics

Engagement Metrics:
- ✅ 50%+ student participation in leaderboards
- ✅ 30%+ students maintain 7+ day streak
- ✅ 25%+ students join or create teams
- ✅ 15% of achievements shared on social media

Business Metrics:
- ✅ 30% increase in daily active users
- ✅ 20% reduction in dropout rate
- ✅ 40% increase in problems solved per week
- ✅ 10% organic growth from social sharing

Priority 4: Accessibility

Phase 5.1 - Comprehensive Accessibility Implementation

Timeline: 6-8 weeks
Status: Planning Phase
Complexity: High

Business Objectives

1. **Inclusive Education**: Ensure platform accessible to students with disabilities
2. **Legal Compliance**: Meet WCAG 2.1 AA standards, ADA compliance
3. **Market Expansion**: Reach international students with multi-language support
4. **User Experience**: Improve usability for all users

Feature Requirements

4.1 Screen Reader Optimization

User Stories:
- As a blind student, I want to navigate the platform using screen readers
- As a visually impaired student, I want meaningful descriptions of visual content
- As an instructor, I want all students to access learning materials equally

WCAG 2.1 AA Requirements:
- ✅ **1.1.1 Non-text Content**: All images have alt text
- ✅ **1.3.1 Info and Relationships**: Proper semantic HTML
- ✅ **2.1.1 Keyboard**: All functionality via keyboard
- ✅ **2.4.3 Focus Order**: Logical tab order
- ✅ **3.1.1 Language**: Page language identified
- ✅ **4.1.2 Name, Role, Value**: ARIA labels for custom components

Technical Implementation:

```
// Semantic HTML structure
<main role="main" aria-labelledby="page-title">
  <h1 id="page-title">Linear Equations Module</h1>

  <nav aria-label="Module navigation">
    <ul>
      <li><a href="#overview">Overview</a></li>
      <li><a href="#practice">Practice Problems</a></li>
    </ul>
  </nav>

  <section aria-labelledby="problem-section">
    <h2 id="problem-section">Practice Problems</h2>
    {/* Problems */}
  </section>
</main>

// ARIA labels for interactive components
<Button
  aria-label="Submit answer for problem 1"
  aria-describedby="problem-1-description"
  onClick={handleSubmit}
>
  Submit
</Button>

// Live regions for dynamic updates
<div aria-live="polite" aria-atomic="true">
  {feedback && <p>{feedback}</p>}
</div>

// Math content accessibility
import 'mathlive';

<math-field
  read-only
  aria-label="Equation: 4x equals 200 plus 1.5x"
>
  4x = 200 + 1.5x
</math-field>
```

Screen Reader Testing:
- **NVDA** (Windows): Primary testing tool
- **JAWS** (Windows): Secondary testing
- **VoiceOver** (macOS/iOS): Apple ecosystem
- **TalkBack** (Android): Mobile testing

Accessibility Audit Tools:
- axe DevTools (browser extension)
- WAVE (Web Accessibility Evaluation Tool)
- Lighthouse accessibility score
- Manual keyboard navigation testing

4.2 Keyboard Navigation

User Stories:
- As a motor-impaired student, I want to navigate without a mouse
- As a power user, I want keyboard shortcuts for efficiency
- As a student with RSI, I want to minimize mouse usage

Technical Implementation:

```
// Global keyboard shortcuts
const SHORTCUTS = {
  'mod+k': 'Open command palette',
  'mod+/': 'Show keyboard shortcuts',
  'g h': 'Go to home',
  'g m': 'Go to modules',
  'g p': 'Go to progress',
  'n': 'Next problem',
  'p': 'Previous problem',
  'h': 'Show hint',
  's': 'Submit answer',
  '?': 'Help',
};

// Keyboard shortcut hook
import { useHotkeys } from 'react-hotkeys-hook';

export function useAppShortcuts() {
  const router = useRouter();

  useHotkeys('g h', () => router.push('/'));
  useHotkeys('g m', () => router.push('/modules'));
  useHotkeys('n', () => goToNextProblem());
  // ... more shortcuts
}

// Focus management
import { useFocusTrap } from '@headlessui/react';

export function Modal({ children }: Props) {
  const ref = useFocusTrap(); // Trap focus within modal

  return (
    <div ref={ref} role="dialog" aria-modal="true">
      {children}
    </div>
  );
}

// Skip links
<a href="#main-content" className="skip-to-content">
  Skip to main content
</a>
```

Focus Indicators:

```css
/* Enhanced focus styles */
*:focus-visible {
  outline: 3px solid #3B82F6;
  outline-offset: 2px;
  border-radius: 4px;
}

/* Button focus states */
button:focus-visible {
  box-shadow: 0 0 0 3px rgba(59, 130, 246, 0.5);
}

/* Skip to content link */
.skip-to-content {
  position: absolute;
  top: -40px;
  left: 0;
  background: #000;
  color: #fff;
  padding: 8px;
  z-index: 100;
}

.skip-to-content:focus {
  top: 0;
}
```

Keyboard Navigation Map:
- **Tab**: Forward navigation
- **Shift+Tab**: Backward navigation
- **Enter/Space**: Activate buttons/links
- **Arrow Keys**: Navigate lists, radio groups
- **Escape**: Close modals/menus
- **Home/End**: Jump to start/end

4.3 High Contrast Mode

User Stories:
- As a low-vision student, I want high contrast colors
- As a student with color blindness, I want distinguishable elements
- As a user with photosensitivity, I want reduced brightness options

Technical Implementation:

```
// Color contrast ratios (WCAG AA requires 4.5:1 for normal text)
const CONTRAST_THEMES = {
  default: {
    text: '#1F2937',      // Gray-800
    background: '#FFFFFF',
    primary: '#3B82F6',   // Blue-500
    // Contrast ratios all > 4.5:1
  },
  highContrast: {
    text: '#000000',
    background: '#FFFFFF',
    primary: '#0000EE',
    success: '#008000',
    error: '#CC0000',
    // Contrast ratios all > 7:1 (AAA)
  },
  darkHighContrast: {
    text: '#FFFFFF',
    background: '#000000',
    primary: '#FFFF00',
    success: '#00FF00',
    error: '#FF0000',
  },
};

// Theme context
interface ThemeContext {
  contrastMode: 'default' | 'high' | 'dark-high';
  setContrastMode: (mode: string) => void;
}

export function ThemeProvider({ children }: Props) {
  const [contrastMode, setContrastMode] = useState('default');

  useEffect(() => {
    // Apply theme CSS variables
    const theme = CONTRAST_THEMES[contrastMode];
    Object.entries(theme).forEach(([key, value]) => {
      document.documentElement.style.setProperty(`--color-${key}`, value);
    });
  }, [contrastMode]);

  return (
    <ThemeContext.Provider value={{ contrastMode, setContrastMode }}>
      {children}
    </ThemeContext.Provider>
  );
}

// Color blindness simulator integration
import { simulate } from 'color-blind';

export function ColorBlindnessSimulator({ type }: Props) {
  // Preview site with different color blindness types
  // Protanopia, Deuteranopia, Tritanopia
}
```

Accessibility Settings Panel:

```
// components/accessibility-settings.tsx
export function AccessibilitySettings() {
  return (
    <Dialog>
      <DialogTitle>Accessibility Settings</DialogTitle>
      <DialogContent>
        <Setting label="Contrast Mode">
          <RadioGroup>
            <Radio value="default">Default</Radio>
            <Radio value="high">High Contrast</Radio>
            <Radio value="dark-high">Dark High Contrast</Radio>
          </RadioGroup>
        </Setting>

        <Setting label="Font Size">
          <Slider min={12} max={24} step={2} />
        </Setting>

        <Setting label="Reduce Motion">
          <Toggle />
        </Setting>

        <Setting label="Screen Reader Optimizations">
          <Toggle />
        </Setting>
      </DialogContent>
    </Dialog>
  );
}
```

4.4 Multi-Language Support (i18n)

User Stories:
- As an international student, I want to use the platform in my native language
- As a non-English speaker, I want math terminology in my language
- As an instructor, I want to offer courses in multiple languages

Languages to Support (Phase 1):
1. **English** (en-US) - Primary
2. **Spanish** (es-ES) - High priority
3. **Chinese Simplified** (zh-CN) - High priority
4. **French** (fr-FR) - Medium priority
5. **Portuguese** (pt-BR) - Medium priority
6. **Arabic** (ar-SA) - Medium priority (RTL support)

Technical Implementation:

```
// next-i18next configuration
// next-i18next.config.js
module.exports = {
  i18n: {
    defaultLocale: 'en',
    locales: ['en', 'es', 'zh', 'fr', 'pt', 'ar'],
    localeDetection: true,
  },
  reloadOnPrerender: process.env.NODE_ENV === 'development',
};

// Translation files structure
// locales/en/common.json
{
  "navigation": {
    "home": "Home",
    "modules": "Learning Modules",
    "progress": "Progress",
    "badges": "Badges"
  },
  "problems": {
    "submit": "Submit Answer",
    "hint": "Get Hint ({{count}} available)",
    "correct": "Correct! Great job!",
    "incorrect": "Not quite right. Try again!"
  }
}

// locales/es/common.json
{
  "navigation": {
    "home": "Inicio",
    "modules": "Módulos de Aprendizaje",
    "progress": "Progreso",
    "badges": "Insignias"
  },
  "problems": {
    "submit": "Enviar Respuesta",
    "hint": "Obtener Pista ({{count}} disponibles)",
    "correct": "¡Correcto! ¡Buen trabajo!",
    "incorrect": "No es del todo correcto. ¡Inténtalo de nuevo!"
  }
}

// Usage in components
import { useTranslation } from 'next-i18next';

export function NavigationBar() {
  const { t } = useTranslation('common');

  return (
    <nav>
      <Link href="/">{t('navigation.home')}</Link>
      <Link href="/modules">{t('navigation.modules')}</Link>
      <Link href="/progress">{t('navigation.progress')}</Link>
      <Link href="/badges">{t('navigation.badges')}</Link>
    </nav>
  );
}

// Problem content translation
interface ProblemTranslation {
  locale: string;
  title: string;
  description: string;
  businessContext: string;
  problemStatement: string;
  hints: string[];
  solution: {
    steps: string[];
    explanation: string;
    finalAnswer: string;
  };
}

model ProblemContent {
  id          String   @id @default(cuid())
  problemId   String
  locale      String
  content     Json     // ProblemTranslation

  @@unique([problemId, locale])
  @@index([locale])
}
```

RTL Support (Arabic):

```
/* RTL stylesheet */
html[dir="rtl"] {
  direction: rtl;
}

html[dir="rtl"] .text-left {
  text-align: right;
}

html[dir="rtl"] .ml-4 {
  margin-left: 0;
  margin-right: 1rem;
}

/* Automatic flipping with logical properties */
.container {
  padding-inline-start: 1rem;
  padding-inline-end: 1rem;
  margin-inline: auto;
}
```

Translation Workflow:
1. Extract translatable strings to JSON files
2. Professional translation service (or community translation)
3. Review by native speakers
4. QA testing in each language
5. Continuous translation updates

Math Terminology:

```
// locales/en/math.json
{
  "terms": {
    "equation": "Equation",
    "variable": "Variable",
    "coefficient": "Coefficient",
    "breakEven": "Break-even Point",
    "revenue": "Revenue",
    "cost": "Cost",
    "profit": "Profit"
  }
}

// locales/es/math.json
{
  "terms": {
    "equation": "Ecuación",
    "variable": "Variable",
    "coefficient": "Coeficiente",
    "breakEven": "Punto de Equilibrio",
    "revenue": "Ingresos",
    "cost": "Costo",
    "profit": "Ganancia"
  }
}
```

Testing Strategy

Accessibility Testing

```
// __tests__/accessibility/wcag.test.ts
import { axe } from 'jest-axe';

describe('WCAG 2.1 AA Compliance', () => {
  it('should have no accessibility violations on home page', async () => {
    const { container } = render(<HomePage />);
    const results = await axe(container);
    expect(results).toHaveNoViolations();
  });

  it('should have proper ARIA labels on interactive elements', () => {
    // Test ARIA attributes
  });

  it('should maintain focus order', () => {
    // Test tab order
  });
});

// __tests__/accessibility/keyboard-nav.test.ts
describe('Keyboard Navigation', () => {
  it('should navigate with Tab key', () => {
    render(<NavigationBar />);
    const firstLink = screen.getByText('Home');
    firstLink.focus();
    userEvent.tab();
    expect(screen.getByText('Modules')).toHaveFocus();
  });

  it('should trigger actions with Enter/Space', () => {
    // Test keyboard activation
  });
});

// __tests__/accessibility/screen-reader.test.ts
describe('Screen Reader Support', () => {
  it('should announce live region updates', () => {
    // Test aria-live regions
  });

  it('should provide meaningful alt text', () => {
    // Test image alt attributes
  });
});
```

Internationalization Testing

```
// __tests__/i18n/translation.test.ts
describe('Internationalization', () => {
  it('should render Spanish translations', () => {
    const { container } = render(<HomePage />, { locale: 'es' });
    expect(screen.getByText('Inicio')).toBeInTheDocument();
  });

  it('should switch languages dynamically', () => {
    // Test language switcher
  });

  it('should handle RTL layout for Arabic', () => {
    const { container } = render(<HomePage />, { locale: 'ar' });
    expect(container.firstChild).toHaveAttribute('dir', 'rtl');
  });
});
```

Visual Regression Testing

```
// __tests__/visual/contrast-modes.test.ts
import { toMatchImageSnapshot } from 'jest-image-snapshot';

describe('Contrast Modes', () => {
  it('should render correctly in high contrast mode', async () => {
    const image = await page.screenshot();
    expect(image).toMatchImageSnapshot();
  });
});
```

Deployment Plan

Phase 5.1.1: Screen Reader & Keyboard (Weeks 1-3)

- Audit all pages with axe DevTools
- Add semantic HTML and ARIA labels
- Implement keyboard shortcuts
- Screen reader testing with NVDA/VoiceOver
- Deploy accessibility improvements

Phase 5.1.2: High Contrast Mode (Week 4)

- Design high contrast themes
- Implement theme switcher
- Test with color contrast tools
- Deploy contrast modes

Phase 5.1.3: i18n Infrastructure (Week 5)

- Set up next-i18next
- Extract translatable strings
- Implement language switcher
- Deploy infrastructure

Phase 5.1.4: Translation (Weeks 6-7)

- Translate UI strings (Spanish, Chinese)

- Translate 30 practice problems
- Native speaker review
- Deploy Spanish and Chinese versions

Phase 5.1.5: RTL & Additional Languages (Week 8)

- Implement RTL support for Arabic
- Add French and Portuguese
- Final accessibility audit
- Deploy complete accessibility suite

Success Metrics

Accessibility Metrics:
- ✅ WCAG 2.1 AA compliance (Lighthouse score 95+)
- ✅ 0 critical accessibility violations (axe)
- ✅ 100% keyboard navigability
- ✅ Support for 3+ screen readers

Internationalization Metrics:
- ✅ 6 languages supported
- ✅ 95%+ translation coverage
- ✅ RTL layout functional for Arabic
- ✅ 20% increase in international users

User Impact:
- ✅ 15% increase in users with accessibility settings enabled
- ✅ 90%+ satisfaction from accessibility users
- ✅ 30% increase in non-English speaking students

Master Timeline

Quarterly Roadmap (24-32 weeks)

Q1 (Weeks 1-12): Advanced Analytics + Content Expansion Kickoff
- Weeks 1-8: Priority 1 (Advanced Analytics) - Complete
- Weeks 9-12: Priority 2 (Content Expansion) - Start

Q2 (Weeks 13-24): Content Expansion + Gamification
- Weeks 13-18: Priority 2 (Content Expansion) - Complete
- Weeks 19-24: Priority 3 (Gamification 2.0) - Complete

Q3 (Weeks 25-32): Accessibility + Polish
- Weeks 25-32: Priority 4 (Accessibility) - Complete

Gantt Chart Visualization

```
Priority 1: Advanced Analytics
███████████████ (Weeks 1-8)

Priority 2: Content Expansion
        ███████████████████████ (Weeks 9-18)

Priority 3: Gamification 2.0
                   ████████████ (Weeks 19-24)

Priority 4: Accessibility
                       ████████████████ (Weeks 25-32)
```

Resource Requirements

Development Team

Core Team:
- **Full-Stack Developer** (1): All phases
- **Frontend Developer** (1): UI/UX heavy phases (P2, P3, P4)
- **Backend Developer** (0.5): Analytics infrastructure (P1)
- **Content Creator** (1): Content expansion (P2)
- **QA Engineer** (0.5): All phases

Specialists:
- **Data Scientist** (0.25): Analytics algorithms (P1)
- **Video Producer** (0.5): Video content (P2)
- **Accessibility Expert** (0.5): WCAG compliance (P4)
- **Translator** (0.25): Internationalization (P4)

Budget Estimate

Development Costs:
- Advanced Analytics: $25,000 - $35,000
- Content Expansion: $30,000 - $40,000
- Gamification 2.0: $15,000 - $20,000
- Accessibility: $20,000 - $30,000

Total: $90,000 - $125,000

Additional Costs:
- Translation services: $5,000 - $10,000
- Video production: $8,000 - $12,000
- Accessibility audit: $3,000 - $5,000
- Testing tools/licenses: $2,000

Grand Total: $108,000 - $154,000

Risk Management

High-Risk Areas

1. **Analytics Performance at Scale**
   - Risk: Database queries slow down with large datasets
   - Mitigation: Aggressive indexing, caching, materialized views
   - Contingency: Implement data warehouse (BigQuery, Snowflake)

2. **Content Quality Control**
   - Risk: New problems have errors or unclear instructions
   - Mitigation: Peer review process, student beta testing
   - Contingency: Rollback bad content, fix and redeploy

3. **Accessibility Legal Compliance**
   - Risk: ADA/Section 508 non-compliance leads to legal issues
   - Mitigation: Professional accessibility audit, WCAG checklist
   - Contingency: Hire accessibility consultant, remediation plan

4. **Translation Accuracy**
   - Risk: Poor translations confuse international students
   - Mitigation: Native speaker review, pilot with small user group
   - Contingency: Re-translate with better service, student feedback loop

Medium-Risk Areas

- **Scope Creep**: Features expand beyond plan
- Mitigation: Strict change control, prioritization framework

- **Resource Availability**: Key team members unavailable

- Mitigation: Knowledge sharing, documentation, backup resources

- **Third-Party Dependencies**: Libraries/APIs change or deprecate

- Mitigation: Version pinning, vendor monitoring, migration plans

---

Quality Assurance Framework

Testing Requirements Per Phase

Pre-Development:
- ✅ Requirements review
- ✅ Technical design review
- ✅ Security assessment

During Development:
- ✅ Unit tests (80%+ coverage)
- ✅ Integration tests (all APIs)
- ✅ Code review (2 approvers)
- ✅ Daily builds pass

Pre-Deployment:
- ✅ E2E tests (critical flows)
- ✅ Performance tests (load/stress)
- ✅ Security scan (Snyk, npm audit)
- ✅ Accessibility audit (axe, WAVE)
- ✅ Cross-browser testing
- ✅ Mobile responsive testing
- ✅ Staging environment verification

Post-Deployment:
- ✅ Production smoke tests
- ✅ Monitoring and alerts configured
- ✅ User acceptance testing
- ✅ Analytics validation

Continuous Integration Pipeline

```yaml
# .github/workflows/enhancement-ci.yml
name: Enhancement CI/CD

on:
  push:
    branches: [main, develop]
  pull_request:
    branches: [main, develop]

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: actions/setup-node@v3
      - run: yarn install
      - run: yarn lint
      - run: yarn test --coverage
      - run: yarn test:e2e
      - run: yarn build

  accessibility:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - run: yarn install
      - run: yarn test:accessibility
      - run: yarn lighthouse-ci

  deploy-staging:
    needs: [test, accessibility]
    if: github.ref == 'refs/heads/develop'
    runs-on: ubuntu-latest
    steps:
      - run: yarn deploy:staging

  deploy-production:
    needs: [test, accessibility]
    if: github.ref == 'refs/heads/main'
    runs-on: ubuntu-latest
    steps:
      - run: yarn deploy:production
```

Documentation Requirements

Technical Documentation

1. **API Documentation**
   - OpenAPI/Swagger specs for new endpoints
   - Request/response examples
   - Authentication requirements
   - Rate limiting details

2. **Component Documentation**
   - Storybook for UI components

   - Props and usage examples
   - Accessibility guidelines

3. **Database Documentation**
   - Entity-relationship diagrams
   - Migration scripts
   - Query optimization guides

4. **Deployment Documentation**
   - Environment setup
   - Configuration management
   - Rollback procedures

User Documentation

1. **Student Guides**
   - How to use new features
   - Keyboard shortcuts reference
   - Accessibility settings guide
   - Multi-language switching

2. **Instructor Guides**
   - Analytics dashboard walkthrough
   - A/B testing how-to
   - Team challenge setup
   - Content management

3. **Admin Guides**
   - System configuration
   - User management
   - Content moderation
   - Performance monitoring

Success Criteria Summary

Priority 1: Advanced Analytics

- ✅ 85%+ test coverage
- ✅ < 200ms API response time
- ✅ 90%+ instructor adoption
- ✅ 20% improvement in at-risk student identification

Priority 2: Content Expansion

- ✅ 80+ total practice problems
- ✅ 30+ video resources
- ✅ 6 interactive graphing modules
- ✅ 40% increase in practice attempts

Priority 3: Gamification 2.0

- ✅ 50%+ leaderboard participation

- ✅ 30% increase in daily active users
- ✅ 25%+ students join teams
- ✅ 20% reduction in dropout rate

Priority 4: Accessibility

- ✅ WCAG 2.1 AA compliance
- ✅ 6 languages supported
- ✅ RTL layout functional
- ✅ 95+ Lighthouse accessibility score

Approval & Sign-Off

Phase Approval Process

Planning Review:
- ☐ Product Owner approval
- ☐ Technical Lead approval
- ☐ Stakeholder sign-off

Development Milestones:
- ☐ Sprint reviews (every 2 weeks)
- ☐ Demo sessions
- ☐ Stakeholder feedback incorporated

Pre-Deployment:
- ☐ QA approval
- ☐ Security review passed
- ☐ Performance benchmarks met
- ☐ Documentation complete

Post-Deployment:
- ☐ Production verification
- ☐ Success metrics tracked
- ☐ Retrospective completed

Revision History

Version	Date	Author	Changes
1.0	2025-12-15	DeepAgent	Initial master plan

Document Status: Planning Phase
Next Review Date: 2025-12-22
Owner: Michael (mltmacster@gmail.com)
Repository: https://github.com/mltmacster/college_algebra_website

---

This master plan provides comprehensive guidance for the next 6-8 months of platform development. All phases are designed to maintain the high standards established in v1.0-2.0 development, with thorough planning, testing, and deployment verification.