Drill Achievement Levels - Implementation Approach

Overview

This document outlines the step-by-step approach for implementing customizable achievement levels for drills in the Steve Sherman Billiards Training System. The feature will allow drill-specific achievement thresholds while maintaining backward compatibility with the current percentage-based system.

Current System Analysis

Existing Structure

- **Drills**: `wp_drills` table with `max_score` field
- **Scoring**: `wp_drill_scores` table with calculated `percentage` field
- **Current Achievements**: Hard-coded percentage thresholds (50%, 70%, perfect score)
- **UI Feedback**: Personal best, perfect score, halfway messages

Current Achievement Logic (from drill-app.html)

- **Perfect Score**: `score === drill.max_score`
- **Personal Best**: `score > previousBest`
- **Halfway**: `(score / maxScore) >= 0.5 && < 0.7`

Feature Requirements

Core Functionality

1. **Optional Achievement System**: Drills can use default percentage-based OR custom achievement levels
2. **Drill-Specific Levels**: Each drill can define its own achievement criteria
3. **Flexible Scoring Types**: Support percentage-based, count-based (balls pocketed), or custom metrics
4. **Backward Compatibility**: Existing drills continue to work with current system
5. **Achievement Display**: Show achievement levels in drill browser and detail pages
6. **Progress Tracking**: Track and display user's achievement level per drill

Database Design

New Tables

1. `wp_drill_achievement_types`

```sql
CREATE TABLE `wp_drill_achievement_types` (
  `id` int NOT NULL AUTO_INCREMENT,
  `name` varchar(50) NOT NULL,
  `display_name` varchar(100) NOT NULL,
  `description` text,
  `calculation_method` enum('percentage', 'score') DEFAULT 'percentage',
  `is_active` tinyint(1) DEFAULT '1',
  `created_at` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`),
  UNIQUE KEY `unique_achievement_type_name` (`name`)
);
```

Initial Data:

- `percentage` - "Percentage Score" - Current system (default) - Achievements based on percentage of max_score achieved
- `score` - "Absolute Score" - Achievements based on actual score values (balls pocketed, points earned, time, etc.)

2. `wp_achievement_level_names`

```sql
```

```sql
CREATE TABLE `wp_achievement_level_names` (
  `id` int NOT NULL AUTO_INCREMENT,
  `theme_name` varchar(100) NOT NULL,
  `level_number` int NOT NULL,
  `level_name` varchar(100) NOT NULL,
  `display_color` varchar(7) DEFAULT NULL,
  `display_icon` varchar(50) DEFAULT NULL,
  `description` text,
  `sort_order` int DEFAULT '0',
  `is_active` tinyint(1) DEFAULT '1',
  `created_by` int DEFAULT NULL,
  `created_at` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  `updated_at` timestamp NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`),
  UNIQUE KEY `unique_theme_level` (`theme_name`, `level_number`, `is_active`),
  KEY `idx_level_names_theme` (`theme_name`),
  KEY `idx_level_names_created_by` (`created_by`),
  CONSTRAINT `fk_level_names_created_by` FOREIGN KEY (`created_by`) REFERENCES `wp_drill_users` (`id`) ON DELETE S
);
```

Initial Data - Default Themes:

Traditional Skills Theme:

- Level 1: "Novice" (#CD7F32 - Bronze)
- Level 2: "Intermediate" (#C0C0C0 - Silver)
- Level 3: "Advanced" (#FFD700 - Gold)
- Level 4: "Expert" (#E5E4E2 - Platinum)
- Level 5: "Master" (#B9F2FF - Diamond)

Gaming Theme:

- Level 1: "Bronze" (#CD7F32)
- Level 2: "Silver" (#C0C0C0)
- Level 3: "Gold" (#FFD700)
- Level 4: "Platinum" (#E5E4E2)
- Level 5: "Diamond" (#B9F2FF)

Shooting Sports Theme:

- Level 1: "Rookie" (#CD7F32)
- Level 2: "Shooter" (#C0C0C0)
- Level 3: "Marksman" (#FFD700)
- Level 4: "Sharpshooter" (#E5E4E2)
- Level 5: "Expert Marksman" (#B9F2FF)

3. wp_drill_achievement_levels

```sql
CREATE TABLE `wp_drill_achievement_levels` (
  `id` int NOT NULL AUTO_INCREMENT,
  `drill_id` int NOT NULL,
  `achievement_type_id` int NOT NULL,
  `level_number` int NOT NULL,
  `level_name_id` int NOT NULL COMMENT 'References wp_achievement_level_names.id',
  `threshold_value` decimal(8,2) NOT NULL,
  `description` text,
  `is_active` tinyint(1) DEFAULT '1',
  `created_at` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  `updated_at` timestamp NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`),
  UNIQUE KEY `unique_drill_level` (`drill_id`, `level_number`, `is_active`),
  KEY `idx_drill_achievement_levels_drill` (`drill_id`),
  KEY `idx_drill_achievement_levels_type` (`achievement_type_id`),
  KEY `idx_drill_achievement_levels_name` (`level_name_id`),
  CONSTRAINT `fk_achievement_levels_drill` FOREIGN KEY (`drill_id`) REFERENCES `wp_drills` (`id`) ON DELETE CASCADE,
  CONSTRAINT `fk_achievement_levels_type` FOREIGN KEY (`achievement_type_id`) REFERENCES `wp_drill_achievement_
  CONSTRAINT `fk_achievement_levels_name` FOREIGN KEY (`level_name_id`) REFERENCES `wp_achievement_level_nam
);
```

3. wp_user_drill_achievements

```sql
```

```
CREATE TABLE `wp_user_drill_achievements` (
  `id` int NOT NULL AUTO_INCREMENT,
  `user_id` int NOT NULL,
  `drill_id` int NOT NULL,
  `achievement_level_id` int NOT NULL,
  `achieved_score` decimal(8,2) NOT NULL,
  `achieved_at` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  `score_entry_id` int DEFAULT NULL COMMENT 'References wp_drill_scores.id',
  PRIMARY KEY (`id`),
  UNIQUE KEY `unique_user_drill_achievement` (`user_id`, `drill_id`, `achievement_level_id`),
  KEY `idx_user_achievements_user` (`user_id`),
  KEY `idx_user_achievements_drill` (`drill_id`),
  KEY `idx_user_achievements_level` (`achievement_level_id`),
  KEY `idx_user_achievements_score_entry` (`score_entry_id`),
  CONSTRAINT `fk_user_achievements_user` FOREIGN KEY (`user_id`) REFERENCES `wp_drill_users` (`id`) ON DELETE CAS
  CONSTRAINT `fk_user_achievements_drill` FOREIGN KEY (`drill_id`) REFERENCES `wp_drills` (`id`) ON DELETE CASCADE,
  CONSTRAINT `fk_user_achievements_level` FOREIGN KEY (`achievement_level_id`) REFERENCES `wp_drill_achievement_
  CONSTRAINT `fk_user_achievements_score` FOREIGN KEY (`score_entry_id`) REFERENCES `wp_drill_scores` (`id`) ON DE
);
```

Table Modifications

Update `wp_drills`

```sql
ALTER TABLE `wp_drills`
ADD COLUMN `uses_custom_achievements` tinyint(1) DEFAULT '0' COMMENT 'Whether drill uses custom achievement
ADD COLUMN `default_achievement_type_id` int DEFAULT NULL COMMENT 'Default achievement type for this drill',
ADD KEY `idx_drills_custom_achievements` (`uses_custom_achievements`),
ADD KEY `idx_drills_achievement_type` (`default_achievement_type_id`),
ADD CONSTRAINT `fk_drills_achievement_type` FOREIGN KEY (`default_achievement_type_id`) REFERENCES `wp_drill_ac
```

Implementation Phases

Phase 1: Database Foundation (Week 1)

Goal: Establish the database structure and basic data

Step 1.1: Create Achievement Types Table

- Create `wp_drill_achievement_types` table

- Insert default achievement types
- Test table creation and data insertion

Step 1.2: Create Level Names Lookup Table

- Create `wp_achievement_level_names` table
- Insert default theme data (Traditional, Gaming, Shooting Sports)
- Test theme and level relationships
- Verify unique constraints for themes

Step 1.3: Create Achievement Levels Table

- Create `wp_drill_achievement_levels` table
- Test foreign key constraints
- Create indexes for performance

Step 1.3: Create User Achievements Table

- Create `wp_user_drill_achievements` table
- Test all foreign key relationships
- Verify unique constraints work correctly

Step 1.4: Modify Drills Table

- Add new columns to `wp_drills`
- Update existing drills to use percentage type by default
- Test backward compatibility

Deliverables:

- [ ] All tables created successfully
- [ ] Default data inserted
- [ ] Foreign keys and constraints working
- [ ] Backward compatibility verified

Phase 2: API Layer (Week 2)

Goal: Create API endpoints to manage achievement levels

Step 2.1: Achievement Types and Level Names API

Create endpoints in `drill-api.php`:

- `GET /achievement-types` - List all achievement types
- `GET /achievement-types/{id}` - Get specific type details
- `GET /level-names/themes` - List all available themes
- `GET /level-names/themes/{theme_name}` - Get all levels for a theme
- `POST /level-names/themes` - Create new theme (admin/coach only)
- `PUT /level-names/{id}` - Update level name (admin/coach only)
- `DELETE /level-names/{id}` - Remove level name (admin/coach only)

Step 2.2: Achievement Levels API

Create endpoints for managing drill-specific levels:

- `GET /drills/{id}/achievement-levels` - Get levels for a drill
- `POST /drills/{id}/achievement-levels` - Create new level (admin/coach only)
- `PUT /achievement-levels/{id}` - Update level (admin/coach only)
- `DELETE /achievement-levels/{id}` - Remove level (admin/coach only)

Step 2.3: User Achievements API

Track user progress:

- `GET /users/{id}/achievements` - Get user's achievements
- `GET /users/{id}/achievements/drill/{drill_id}` - Get achievements for specific drill
- `POST /achievements/check` - Check and award achievements (called after score submission)

Step 2.4: Enhanced Drills API

Extend existing drill endpoints:

- Add achievement levels to drill details response
- Add achievement statistics to drill list
- Include user's achievement status per drill

Deliverables:

- [ ] All API endpoints functional
- [ ] Proper authentication/authorization
- [ ] Error handling implemented
- [ ] API documentation updated

Phase 3: Core Logic Implementation (Week 3)

Goal: Implement achievement calculation and awarding logic

Step 3.1: Achievement Calculation Engine

Create `AchievementCalculator` class:

```php
class AchievementCalculator {
    public function calculateAchievementLevel($drill_id, $score, $max_score);
    public function checkForNewAchievements($user_id, $drill_id, $score, $max_score);
    public function awardAchievement($user_id, $achievement_level_id, $score, $score_entry_id);
    public function getAchievementProgress($user_id, $drill_id);

    // Determines whether to use percentage or score-based calculation
    private function getDrillAchievementType($drill_id);
}
```

Step 3.2: Default Achievement Setup

Create automatic achievement level generation:

- For drills without custom levels, generate default percentage-based levels
- Configurable default thresholds (25%, 50%, 75%, 90%, 100%)
- Automatic level names ("Novice", "Intermediate", "Advanced", "Expert", "Master")

Step 3.3: Score Submission Integration

Modify score submission process:

- After saving score, check for achievement level changes
- Award new achievements automatically
- Return achievement data with score submission response

Step 3.4: Retroactive Achievement Calculation

Create utility to award achievements for existing scores:

- Process all historical scores
- Award appropriate achievements based on current level definitions
- Batch processing for performance

Deliverables:

- [ ] Achievement calculation working correctly
- [ ] Score submission integration complete
- [ ] Retroactive processing utility ready
- [ ] Unit tests for core logic

Phase 4: Admin Interface (Week 4)

Goal: Provide tools for coaches/admins to manage achievement levels

Step 4.1: Achievement Level Management UI

Create admin interface pages:

- List all drills with achievement level status
- Drill-specific achievement level editor with theme selector
- Visual threshold editor with preview
- Bulk operations for similar drills
- **Theme Management Interface**: Create, edit, and manage level name themes

Step 4.2: Level Names and Theme Configuration

Admin tools for managing achievement themes:

- Create new level name themes (e.g., "Martial Arts", "Military Ranks", "Pool Specific")
- Edit existing theme level names and colors
- Preview themes before applying to drills
- Import/export theme configurations
- Set default themes for new drills

Step 4.3: Analytics and Reporting

Achievement insights:

- Achievement distribution per drill
- User achievement progress reports
- Popular achievement levels
- Difficulty analysis based on achievement rates

Deliverables:

- [ ] Admin interface functional
- [ ] Achievement level editor working
- [ ] Analytics dashboard complete
- [ ] User-friendly help documentation

Phase 5: Frontend Integration (Week 5-6)

Goal: Update user-facing interfaces to display achievements

Step 5.1: Drill Browser Updates

Enhance `drill-app.html`:

- Display achievement levels in drill cards
- Show user's current achievement level per drill
- Add achievement level filtering
- Visual indicators for different achievement types

Step 5.2: Drill Detail Page Updates

Enhance `drill-detail.html`:

- Show complete achievement level ladder
- Highlight user's current level and next target
- Display achievement history
- Enhanced celebration messages

Step 5.3: Score Submission Feedback

Improve scoring experience:

- Real-time achievement level preview as user enters score
- Enhanced success messages for new achievement levels
- Achievement badge/icon display
- Social sharing options for achievements

Step 5.4: User Profile/Progress

Create achievement viewing:

- User achievement showcase
- Progress tracking across all drills

- Achievement statistics and milestones
- Leaderboard integration

Deliverables:

- [ ] Drill browser shows achievements
- [ ] Drill detail page enhanced
- [ ] Score submission UX improved
- [ ] User progress viewing complete

Phase 6: Testing and Polish (Week 7)

Goal: Comprehensive testing and user experience refinement

Step 6.1: Automated Testing

- Unit tests for achievement calculation
- API endpoint testing
- Database constraint testing
- Performance testing with large datasets

Step 6.2: User Acceptance Testing

- Coach feedback on admin interface
- Student feedback on achievement display
- Mobile responsiveness testing
- Cross-browser compatibility

Step 6.3: Performance Optimization

- Database query optimization
- Frontend loading performance
- Caching strategies for achievement data
- Background processing for retroactive calculations

Step 6.4: Documentation and Training

- User documentation
- Coach training materials
- Developer documentation

- Migration guides

Deliverables:

- [ ] All tests passing
- [ ] Performance optimized
- [ ] Documentation complete
- [ ] Ready for production deployment

Technical Considerations

Performance

- **Indexing Strategy**: Proper indexes on achievement lookup tables
- **Caching**: Cache frequently accessed achievement data
- **Batch Processing**: Handle retroactive achievement calculations efficiently
- **Lazy Loading**: Load achievement data only when needed

Data Migration

- **Backward Compatibility**: Existing scores continue to work
- **Default Levels**: Auto-generate achievement levels for existing drills
- **Historical Data**: Process existing scores for achievement awards
- **Rollback Plan**: Ability to disable feature if issues arise

Security

- **Authorization**: Only coaches/admins can create/modify achievement levels
- **Validation**: Proper validation of achievement thresholds
- **SQL Injection**: Use prepared statements for all queries
- **Input Sanitization**: Clean all user inputs

Scalability

- **Database Design**: Efficient schema for large user base
- **API Rate Limiting**: Prevent abuse of achievement endpoints
- **Background Jobs**: Process intensive operations asynchronously
- **Monitoring**: Track achievement system performance

Example Configurations

Default Percentage-Based (Current System)

```json
{
  "drill_id": 15,
  "achievement_type": "percentage",
  "theme": "traditional",
  "levels": [
    {"level": 1, "level_name_id": 1, "name": "Novice", "threshold": 25.0, "color": "#CD7F32"},
    {"level": 2, "level_name_id": 2, "name": "Intermediate", "threshold": 50.0, "color": "#C0C0C0"},
    {"level": 3, "level_name_id": 3, "name": "Advanced", "threshold": 75.0, "color": "#FFD700"},
    {"level": 4, "level_name_id": 4, "name": "Expert", "threshold": 90.0, "color": "#E5E4E2"},
    {"level": 5, "level_name_id": 5, "name": "Master", "threshold": 100.0, "color": "#B9F2FF"}
  ]
}
```

Custom Score-Based (Absolute Values)

```json
{
  "drill_id": 8,
  "achievement_type": "score",
  "theme": "shooting_sports",
  "levels": [
    {"level": 1, "level_name_id": 11, "name": "Rookie", "threshold": 10, "color": "#CD7F32"},
    {"level": 2, "level_name_id": 12, "name": "Shooter", "threshold": 20, "color": "#C0C0C0"},
    {"level": 3, "level_name_id": 13, "name": "Marksman", "threshold": 30, "color": "#FFD700"},
    {"level": 4, "level_name_id": 14, "name": "Sharpshooter", "threshold": 40, "color": "#E5E4E2"},
    {"level": 5, "level_name_id": 15, "name": "Expert Marksman", "threshold": 50, "color": "#B9F2FF"}
  ]
}
```

Note: Score-based achievements work with any scoring metric - balls pocketed, points earned, time completed, consecutive successes, etc. The threshold represents the actual score value needed to achieve that level.

Additional Score-Based Examples

Time-Based Drill (Lower scores = better performance)

```json
{
  "drill_id": 25,
  "name": "Speed Shooting Challenge",
  "max_score": 300,
  "achievement_type": "score",
  "theme": "gaming",
  "levels": [
    {"level": 1, "level_name_id": 6, "name": "Bronze", "threshold": 60, "color": "#CD7F32"},
    {"level": 2, "level_name_id": 7, "name": "Silver", "threshold": 120, "color": "#C0C0C0"},
    {"level": 3, "level_name_id": 8, "name": "Gold", "threshold": 180, "color": "#FFD700"},
    {"level": 4, "level_name_id": 9, "name": "Platinum", "threshold": 240, "color": "#E5E4E2"},
    {"level": 5, "level_name_id": 10, "name": "Diamond", "threshold": 300, "color": "#B9F2FF"}
  ]
}
```

Points-Based Drill Using Custom Theme

```json
{
  "drill_id": 12,
  "name": "Five Ball Position Drill",
  "max_score": 40,
  "achievement_type": "score",
  "theme": "custom_pool_theme",
  "levels": [
    {"level": 1, "level_name_id": 16, "name": "Chalk User", "threshold": 8, "color": "#CD7F32"},
    {"level": 2, "level_name_id": 17, "name": "Rack Runner", "threshold": 16, "color": "#C0C0C0"},
    {"level": 3, "level_name_id": 18, "name": "Cue Master", "threshold": 24, "color": "#FFD700"},
    {"level": 4, "level_name_id": 19, "name": "Table Commander", "threshold": 32, "color": "#E5E4E2"},
    {"level": 5, "level_name_id": 20, "name": "Pool Legend", "threshold": 40, "color": "#B9F2FF"}
  ]
}
```

Success Metrics

Technical Metrics

- [ ] Zero breaking changes to existing functionality
- [ ] API response times under 200ms
- [ ] Database queries optimized (no N+1 issues)

- [ ] 100% test coverage for achievement logic

User Experience Metrics

- [ ] Positive coach feedback on admin interface usability
- [ ] Increased user engagement with drill practice
- [ ] Reduced support tickets related to scoring confusion
- [ ] Mobile-friendly achievement displays

Business Metrics

- [ ] Feature adoption rate by coaches
- [ ] Increased drill completion rates
- [ ] User retention improvement
- [ ] Positive user feedback scores

Risk Management

Technical Risks

- **Database Performance**: Large achievement tables could slow queries
  - *Mitigation*: Proper indexing and query optimization
- **Complex Calculations**: Achievement logic could become CPU-intensive
  - *Mitigation*: Efficient algorithms and caching
- **Data Integrity**: Inconsistent achievement states
  - *Mitigation*: Database constraints and validation

User Experience Risks

- **Feature Complexity**: Too many options could confuse users
  - *Mitigation*: Sensible defaults and progressive disclosure
- **Information Overload**: Achievement data cluttering interface
  - *Mitigation*: Clean, intuitive design and optional display
- **Backward Compatibility**: Changes affecting existing workflows
  - *Mitigation*: Careful testing and gradual rollout

Business Risks

- **Development Timeline**: Feature complexity could cause delays
  - *Mitigation*: Phased approach with clear milestones
- **Resource Requirements**: Additional server/database load

    - *Mitigation*: Performance testing and capacity planning
- **User Adoption**: Feature might not be used as expected
    - *Mitigation*: User research and iterative improvement

Next Steps

Immediate Actions (This Week)

1. **Review and Approve**: Stakeholder review of this approach document
2. **Environment Setup**: Prepare development database for new tables
3. **Team Alignment**: Ensure all developers understand the approach
4. **Timeline Confirmation**: Validate the proposed 7-week timeline

Phase 1 Kickoff (Next Week)

1. **Create Database Schema**: Implement the new tables
2. **Insert Default Data**: Set up achievement types and default levels
3. **Test Database Design**: Verify all constraints and relationships
4. **Document Changes**: Update database documentation

Conclusion

This comprehensive approach ensures a robust, scalable implementation of drill achievement levels while maintaining backward compatibility. The phased approach allows for iterative development, testing, and refinement, reducing risk and ensuring high-quality delivery.

The feature will enhance user engagement by providing more granular feedback on progress while giving coaches powerful tools to customize the learning experience for their students.