QnGenius Implementation Strategy & Next Steps

Current Status Analysis

Your QnGenius application has a solid foundation but needs several critical improvements to become production-ready. Here's the prioritized implementation strategy:

Phase 1: Critical Fixes (1-2 weeks)

1.1 Database Configuration Fix

Priority: CRITICAL

Replace the hardcoded database path in DatabaseUtil.java):

```
java

// REMOVE this line:

String propertiesFilePath = "C:/db.properties";

// REPLACE with ConfigManager usage:

private static final ConfigManager config = ConfigManager.getInstance();
```

Files to create/modify:

- (src/main/resources/config/db.properties)
- Replace (DatabaseUtil.java) with (EnhancedDatabaseUtil.java)
- Add HikariCP dependency to your (pom.xml)

1.2 Role-Based Authentication

Priority: CRITICAL

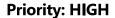
Issues in current code:

- (LoginController) doesn't check user roles
- No proper navigation based on roles
- Missing User model implementation

Files to update:

- 1. Replace empty (User.java) with the enhanced version
- 2. Update LoginController.java with role-based navigation
- 3. Make all controllers implement (UserAwareController) interface

1.3 Complete Model Classes



Your Role.java and User.java are empty. Replace them with complete implementations.

1.4 Error Handling

Priority: HIGH

Add comprehensive try-catch blocks and user-friendly error messages across all controllers.

Phase 2: Core Features Enhancement (2-3 weeks)

2.1 Service Layer Implementation

Add business logic separation:

- (QuestionService.java) (provided)
- (UserService.java)
- (BlueprintService.java)
- (ExamService.java)

2.2 Bulk Import Feature

Priority: HIGH for productivity

Implement the BulkImportController.java with:

- Excel file parsing
- Question validation
- Duplicate detection
- Batch processing

Required dependencies:

xml	

2.3 Modern Dashboard

Replace the basic (MainAppController) with the (ModernDashboardController) for better user experience.

Phase 3: UI/UX Improvements (2-3 weeks)

3.1 Modern Styling

- Apply the modern CSS stylesheet
- Update all FXML files with consistent styling
- Implement responsive design principles

3.2 Enhanced Navigation

- Add proper menu system
- Implement breadcrumbs
- Add keyboard shortcuts

3.3 Data Validation

- Client-side form validation
- Real-time feedback
- Input sanitization

Critical Issues to Address

1. Security Vulnerabilities

- SQL injection prevention (partially done with PreparedStatement)
- Input validation missing
- Password policies not enforced
- Session management missing

2. Database Performance

- No connection pooling (fixed with HikariCP)
- Missing database indexes
- No query optimization

3. Code Quality Issues

- Inconsistent error handling
- No logging framework
- Missing unit tests
- No documentation

Add these to your pom.xml:						
xml						

```
<dependencies>
  <!-- Connection Pooling -->
  <dependency>
    <groupId>com.zaxxer
    <artifactId>HikariCP</artifactId>
    <version>5.0.1</version>
  </dependency>
  <!-- Excel Processing -->
  <dependency>
    <groupId>org.apache.poi
    <artifactId>poi</artifactId>
    <version>5.2.4</version>
  </dependency>
  <dependency>
    <groupId>org.apache.poi
    <artifactId>poi-ooxml</artifactId>
    <version>5.2.4</version>
  </dependency>
  <!-- Logging -->
  <dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-api</artifactId>
    <version>2.0.9</version>
  </dependency>
  <dependency>
    <groupId>ch.qos.logback
    <artifactId>logback-classic</artifactId>
    <version>1.4.11</version>
  </dependency>
  <!-- Testing -->
 <dependency>
    <groupId>org.junit.jupiter
    <artifactId>junit-jupiter</artifactId>
    <version>5.10.0</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>org.testfx</groupId>
    <artifactId>testfx-junit5</artifactId>
    <version>4.0.17</version>
    <scope>test</scope>
```

</dependency>
</dependencies>

File Structure Improvements

src/main/java/com/qngenius/
— controller/
AdminController.java (existing - needs updates)
CoEController.java (existing - needs updates)
EnhancedLoginController.java (new)
— ModernDashboardController.java (new)
BulkImportController.java (new)
UserAwareController.java (interface)
User.java (replace empty version)
Role.java (complete implementation)
L [existing models] (keep as-is)
service/
— QuestionService.java (new)
UserService.java (new)
BlueprintService.java (new)
ExamService.java (new)
util/
ConfigManager.java (new)
EnhancedDatabaseUtil.java (replaces DatabaseUtil)
ValidationUtil.java (new)
L—— exception/
—— QnGeniusException.java (new)
ServiceException.java (new)
L—— ValidationException.java (new)
src/main/resources/

db.properties
bulk-import.fxml (new)
[existing FXML files]
L—— css/
— modern-dashboard.css (new)
L style.css (enhanced)

Implementation Order

- 1. Week 1: Fix database configuration and connection pooling
- 2. Week 1: Implement role-based authentication
- 3. Week 2: Complete User and Role models
- 4. **Week 2**: Add service layer architecture
- 5. Week 3: Implement bulk import feature
- 6. Week 3: Create modern dashboard
- 7. **Week 4**: Apply modern styling and CSS
- 8. **Week 4**: Add comprehensive error handling

Testing Strategy

- 1. **Unit Tests**: For service layer methods
- 2. **Integration Tests**: For database operations
- 3. **UI Tests**: Using TestFX for controller testing
- 4. **Manual Testing**: For user workflows

Performance Considerations

1. **Database