University Course Management System

Technical Documentation

---

1. Overview

The University Course Management System is a web application for managing courses, enrollments, grades, and schedules. It uses a layered architecture with Spring Boot, Spring MVC, Spring Security, Hibernate/Spring Data JPA, JdbcTemplate, Thymeleaf, and a relational database. The system supports two authentication modes: classic web login with sessions and a token-based REST API for external or SPA clients.

Main Roles

- **Student**: Enrolls in courses, views grades, sees schedules
- **Professor**: Manages course content, grades, and schedules
- **Administrator**: Manages users, courses, schedules, and reporting

Non-Functional Requirements

- Responsive UI with Thymeleaf templates
- Secure role-based access control (both MVC and REST API)
- Internationalization support (English and Croatian)
- Scheduled background tasks for notifications
- At least 80% automated test coverage

---

2. Regular Login (Spring MVC + Thymeleaf)

2.1 Functional Description

The regular login mechanism is designed for browser users who interact with server-rendered pages. Users authenticate via a Thymeleaf login form. On successful authentication, a server-side HTTP session is created and Spring Security stores the authenticated principal in the security context. Navigation and access to pages are controlled based on the user's role.

2.2 Login Flow

1. User navigates to the login page and submits credentials
2. Spring Security authenticates against the user data in the database
3. On success, a session is created and the user is redirected to a role-specific dashboard
4. On failure, an error message is shown on the same login page

2.3 Role-Based Dashboards

After login, users are redirected to different areas depending on role:

- **Student Dashboard** (/student/dashboard): Shows enrolled courses, upcoming classes, and grades
- **Professor Dashboard** (/professor/dashboard): Enables course management, assignment creation, and grade entry
- **Admin Dashboard** (/admin/dashboard): Provides administration for users, courses, and schedules

2.4 Key Endpoints

Method	Endpoint	Access	Description
GET	/login	Public	Display login form
POST	/login	Public	Process login (Spring Security)
GET	/logout	Authenticated	Destroy session, logout
GET	/register	Public	Registration form
POST	/register	Public	Create new user account

2.5 Security Features

- **Password Hashing**: BCrypt with configurable strength (10 rounds)
- **CSRF Protection**: Hidden CSRF token in every form
- **Session Management**: 30-minute timeout by default
- **HttpOnly Cookies**: Prevents JavaScript access to session ID
- **Secure Flag**: HTTPS-only in production
- **Role-Based Redirects**: Automatic dashboard routing per role

---

3. REST API Security (JWT Authentication)

3.1 Purpose and Scope

The REST API is intended for SPAs and external clients (e.g., mobile apps, Postman, other systems). It is stateless and secured using JSON Web Tokens (JWT). Two types of tokens are used:

- **Access Token**: Short-lived (15 minutes), used on every protected API call
- **Refresh Token**: Long-lived (7 days), used to obtain new access tokens without re-entering credentials

3.2 JWT Token Structure

A JWT token consists of three parts separated by periods:

- **Header**: Contains algorithm (HS256) and token type
- **Payload**: Contains user ID, username, roles, expiration time, and issued time
- **Signature**: Cryptographic signature generated with a secret key

3.3 Authentication Flow

1. Client sends credentials to the authentication endpoint
2. If valid, the server returns an access token and a refresh token
3. The client includes the access token in the Authorization header on all API requests
4. When the access token expires, the client calls a refresh endpoint with the refresh token
5. If both tokens are invalid, the client must re-authenticate

3.4 API Endpoints

Method	Endpoint	Purpose	Returns
POST	`/api/auth/login`	Authenticate user	Access token, refresh token, user info
POST	`/api/auth/refresh`	Get new access token	New access token with updated expiry
POST	`/api/auth/logout`	Invalidate tokens	Success confirmation
GET	`/api/auth/me`	Get current user info	Logged-in user details

3.5 Authorization and Role Mapping

All protected REST endpoints are guarded by security configuration and method-level annotations. The JWT contains the user's roles, which are mapped to application authorities. Examples:

- **Student APIs**: Personal enrollments, grades, and schedule
- **Professor APIs**: Assigned course management, grading, schedule management
- **Admin APIs**: User management, reports, bulk operations

Unauthorized requests result in standard HTTP error codes (401 for missing/invalid token, 403 for insufficient permissions).

---

4. Reports and Bulk Operations (JdbcTemplate)

4.1 Purpose

JdbcTemplate is used for grade export functionality to CSV format. Professors can download course grades as CSV files from their professor dashboard.

4.2 Grade Export Feature

Professors access the grade export feature per course:

- **Endpoint**: GET /professor/courses/{courseId}/grades/export
- **Returns**: CSV file containing student names and their grades
- **Implementation**: Uses GradeReportRepository with custom SQL query to fetch enrollment and grade data
- **Format**: Student name, enrollment status, grade value, date graded

---

5. Scheduler and Notifications

5.1 Scheduling Overview

The system uses a scheduler to execute background tasks at fixed times or intervals. Tasks are designed to be idempotent and safe to run repeatedly.

5.2 Typical Scheduled Tasks

- **Class Reminders**: Sends notifications for upcoming classes on weekday mornings
- **Grade Notifications**: Alerts students when new grades are published

5.3 Notification Channels

Email is the primary notification channel. The scheduler reads necessary data (e.g., upcoming classes) from the database and dispatches messages based on users' roles and enrollments.

5.4 Configuration

Scheduled tasks are configured centrally with:

- Enable/disable flags for each task type
- Cron expressions defining execution times
- Retry logic and error handling
- Thread pool size for concurrent task execution

---

6. Internationalization (i18n)

6.1 Supported Languages

The application supports multiple languages:

- **English (en)**: Primary language, all features available
- **Croatian (hr)**: Secondary language, all UI text localized

6.2 Language Selection

Users can switch language via a language selector (e.g., a link or dropdown). The selected locale is stored in session or cookie and applied to all subsequent views.

6.3 Localized Content

All user-facing texts are externalized into message bundles:

- Login page text
- Dashboard labels and messages
- Error messages and validation feedback
- Email notifications

6.4 REST API Localization

REST responses and error messages can be localized by inspecting the request's locale header or user preferences.

---

7. Testing and Quality Assurance

7.1 Test Coverage Goals

The project includes automated tests across multiple layers targeting 80%+ coverage through unit and integration testing.

7.2 Test Types

Unit Tests: Test individual methods and components in isolation using mocks

Integration Tests: Test interactions between components (e.g., controller with service)

---

8. Core Features Summary

8.1 Course Management

Administrators can create, update, and delete courses. Courses are associated with a professor and contain schedule entries. Students can view course details including descriptions, credits, and associated professor.

8.2 Enrollment Management

Students can enroll in available courses. Professors can view their course enrollments. Administrators can manage all enrollments and adjust enrollment status.

8.3 Grade Management

Professors assign grades to students within their courses. Grades trigger automatic notifications to students. CSV export available for grade reports via JdbcTemplate.

8.4 Schedule Management

Administrators create course schedules specifying day of week, time, and room. Students and professors can view schedules for their courses. Schedule information is included in email notifications.

---

9. Security and Access Control Summary

9.1 Authentication Modes

- **Session-Based (MVC)**: For browser users, using HTTP sessions with Spring Security
- **Token-Based (REST API)**: For external clients, using JWT tokens with automatic refresh

9.2 Authorization

- Role-based access control at controller and method levels
- `@PreAuthorize` annotations enforce role requirements
- Thymeleaf templates conditionally display elements based on user roles

9.3 Security Practices

- Passwords stored securely using BCrypt hashing
- CSRF protection enabled for all form submissions
- HTTPS required in production
- SQL injection prevented through parameterized queries
- XSS protection via automatic HTML escaping
- Input validation at controller and service layers
- Security headers configured in production

---

10. API Reference Summary

10.1 Authentication

- POST `/api/auth/login`: Authenticate and receive tokens
- POST `/api/auth/refresh`: Refresh access token
- POST `/api/auth/logout`: Logout and invalidate tokens
- GET `/api/auth/me`: Get current user info

10.2 Courses

- GET /api/courses: List all courses
- POST /api/courses: Create course [ADMIN]
- GET /api/courses/{id}: Get course details
- PUT /api/courses/{id}: Update course [ADMIN]
- DELETE /api/courses/{id}: Delete course [ADMIN]

10.3 Enrollments

- POST /api/enrollments/courses/{courseId}: Enroll in course [STUDENT]
- GET /api/enrollments/my-courses: List my enrollments [STUDENT]
- DELETE /api/enrollments/{id}: Drop course [STUDENT]
- GET /api/enrollments: List all enrollments [ADMIN]

10.4 Grades

- POST /api/grades: Assign grade [PROFESSOR, ADMIN]
- GET /api/grades: View grades [All]
- GET /api/grades/course/{courseId}: Course grades [PROFESSOR]
- PUT /api/grades/{id}: Update grade [PROFESSOR, ADMIN]

10.5 Schedules

- GET /api/schedules: List all schedules [ADMIN]
- POST /api/schedules: Create schedule [ADMIN]
- GET /api/courses/{id}/schedule: Course schedule [All]
- PUT /api/schedules/{id}: Update schedule [ADMIN]
- DELETE /api/schedules/{id}: Delete schedule [ADMIN]

10.6 Reports

- GET /professor/courses/{courseId}/grades/export: CSV grade export [PROFESSOR]

---

11. Deployment and Configuration

11.1 Environment Profiles

The application supports multiple profiles:

- **development**: H2 in-memory database, debug logging enabled
- **production**: External database (PostgreSQL/MySQL), security hardened

11.2 Configuration Management

Configuration is managed through:

- application.yml: Shared settings
- application-dev.yml: Development overrides
- application-prod.yml: Production overrides
- Environment variables for secrets (database password, JWT secret)

11.3 Running the Application

Development mode uses Spring Boot default settings. Production deployment requires environment variables for database credentials and JWT secret. The application can be deployed as a JAR file or in a containerized environment (Docker).

---

12. Troubleshooting Guide

Q: JWT Token Expired

Issue: Getting 401 Unauthorized error

Solution: Use the refresh_token endpoint to obtain a new access_token. Call `POST /api/auth/refresh` with your refresh token. The new access token will be valid for 15 minutes.

Q: CORS Error Accessing API from SPA

Issue: Browser blocks requests from SPA to API

Solution: CORS is configured to allow requests from authorized origins. Verify your SPA origin is whitelisted in the security configuration and use appropriate CORS headers in requests.

Q: Can't Login, Invalid Credentials Error

Issue: Login fails even with correct username/password

Solution: Verify username and password are correct. Passwords are case-sensitive. Check that the user account exists and is marked as active in the database.

Q: Schedule Notifications Not Sending

Issue: Scheduled tasks not executing or emails not received

Solution: Verify scheduler is enabled in configuration. Check that email service is configured with valid SMTP credentials. Review logs for scheduler execution and email sending errors.

---

13. Performance Tips

- Use pagination for large course and enrollment lists
- Add database indexes on frequently-queried columns
- Cache frequently-accessed courses (Redis optional)
- Monitor query performance with Hibernate SQL logging
- Use JdbcTemplate for CSV export operations

---

14. Conclusion

The University Course Management System delivers a production-ready platform for academic institution management. With dual authentication modes (session-based and JWT), comprehensive role-based access control, CSV grade export via JdbcTemplate, and scheduled notifications, the system meets all functional and non-functional requirements.

Key Achievements:

- 105-hour implementation completed
- Secure authentication with two parallel modes
- Responsive UI with internationalization
- Grade CSV export functionality
- Scheduled notifications for key events
- Production-ready deployment configuration

Repository: https://github.com/MatejStrlek/uni_course_management