A Maksimirska 58a, HR-10000 Zagreb
T (01) 2332 861
E info@algebra.hr
www.algebra.hr

ALGEBRA

Development of Web Applications
Project specification

1. General Information

- The defense of the project takes place during the exam periods.
- Student applies for the defense the same as for written exams.
- Students who have applied for a deadline will be contacted by a lecturer or assistant to arrange a term for the defense of the project assignment.

2. Project Task

Create a web application to manage the video content service (streaming).
The application has three modules – a third party integration module, an administrative module and a public module:

1. **Third party integration module**
   In the following text also referenced as „integration module".
   It is used for automation, for example the video content entry.

2. **Administrative module**
   It is accessed only by an administrator who, for example, changes the list of video content.

3. **Public module**
   It can be viewed by registered users. For example, the registered user selects and starts the video streaming from the list of video content.

Learning Outcome 1 (Minimum/Desired): 10/10
Learning Outcome 2 (Minimum/Desired): 10/10
Learning Outcome 3 (Minimum/Desired): 10/10
Learning Outcome 4 (Minimum/Desired): 10/10
Learning Outcome 5 (Minimum/Desired): 10/10

Algebra d.o.o.
Upisano kod trgovačkog suda u Zagrebu
MBS 080220316
OIB 24919984448

Temeljni kapital: 800.100,00 kn
direktor: Tomislav Dominković
Poslovna banka: Zagrebačka banka d.d.,
Trg bana Josipa Jelačića 10, HR-10000 Zagreb

IBAN HR0223600001101285178
MB 1361031

3. Integration module

Solve the integration module using ASP.NET Core Web API technology (Learning Outcomes 1+2).

Minimum outcomes

For the integration module, implement the following:

1. Create RESTful endpoint (CRUD) for retrieving and editing all video content.
   Properties of video content are *name*, *description*, *image*, *total time*, *streaming url*, *genre* and *tags*. Support paging, filtering by *name* or part of name and ordering by *id*, *name* and *total time*.
   The endpoint URL is api/videos.

2. Create RESTful endpoint (CRUD) for managing genres and tags.
   Properties of genre are *name* and *description*. Property of tag is *name*.
   The endpoint URLs are api/genres and api/tags.

3. Use Swagger interface and HTTPS, in both production and development environments.

Desired outcomes

For the integration module, implement the following:

1. Implement JWT token authentication with user registration and e-mail validation. When user is created, add a notification with url that needs to be followed to validate the user. Protect endpoint for videos.
   The endpoint URL is api/users.

2. Create RESTful endpoint (CRUD) for managing notifications.
   Properties of notification are *receiver email*, *subject* and *body*. The endpoint also contains an action to send all the notifications that have not been sent yet. Notifications are sent to the user's e-mail by using a configurable SMTP client.
   The endpoint URL is api/notifications.

3. Create a static page that can:
   - retrieve unsent notifications count (add needed endpoint)
   - invoke action that sends notifications

   Static page saves and displays notification data to/from localstorage.

4. Administrative module

Solve the administrative module using ASP.NET Core MVC technology (learning outcomes 3+4+5).

Minimum outcomes

For the administrative module, implement the following:

1. Create video content management pages (CRUD).
   - The page where the list is displayed contains paging and filtering by video name and genre name.
   - The page where the list is displayed contains a button for adding video that leads to the page for adding the video properties
   - The page where information about individual video is displayed has also the ability to change and save its properties and the button to delete it.

2. Create page for viewing countries.
   List of countries should support paging.

3. Create tag management page (CRUD).

4. Create genre management page (CRUD) with AJAX support.

Desired outcomes

For the administrative module, implement the following:

1. Use the AJAX technique for paging the list of videos.

2. Remember the filter on client when filtering the list of videos.

3. Create pages for managing registered users.
   - The page that displays the list of users contains filtering by first name, last name, username, and country of origin.
   - Filtering is remembered on client.
   - The page where the list is displayed contains an add user button that leads to the add user page.
   - The page on which the information about the individual user is displayed contains the capability of changing the user data and the capability of deactivating the user (soft-delete).

4. Use mapping between data access layer and business layer for this module.

5. Public module

Solve the public module using ASP.NET Core MVC technology (learning outcomes 3+4+5).

Minimum outcomes

For the public module, implement the following:

1. Create user self-register page (without e-mail verification and change password functionality).

2. Create login page.
   After successful login, a video selection page opens.

3. Create video selection page.
   Page contains cards. The card title is video name. Card contains video image and video description. Cards are filtered - there should be a textbox to enter the name or part of the name of the video content to filter by. Clicking on the card leads to page of the particular video.

4. Create video page.
   Shows the video content with all its properties. Contains a button to start playing the video. Clicking on this button redirects the page to the playback URL.

5. Show the username of the logged-in user on each page, and implement an option to log out.

Desired outcomes

For the public module, implement the following:

1. Add e-mail verification option to self-register functionality.

2. Create user profile page.
   It should contain all the basic information about the user, and a link to change password.

3. Implement paging for video selection page.
   Use the AJAX technique for paging.

4. Use mapping between data access layer and business layer for this module.

6. Important notes

- Use *the RwaMovies* database structure.
- Ensure a good user experience without showing uncaught exceptions.
- Ensure data integrity: all data should be validated. Examples are required fields, correct URLs and correct e-mail addresses.
- Distribute the application interface elements meaningfully and visually polish as good as you can.
- Stick to good coding practices.