**Final Project Template for Alpha .NET Students**

**ASP.NET Core MVC Final Project**

This document describes the **final project assignment** for the **ASP.NET Core MVC** course at Telerik Academy.

**Project Description**

The task is to create a simple **online payment portal** where users can **access the accounts** to which they have access rights with their account balances **and make simple payments** internally between the accounts registered in the system.

Design and implement the application using **ASP.NET Core 2.2 MVC**.

The application should have:

* **public part** (accessible without authentication)
* **private part** (available for registered users)
* **administrative part** (available for administrators only)

**Public Part**

The **public part** of your projects should be **visible without authentication**. The public part consists of just one page - **home page** with **login box** and **banner ad.**

**Private Part (Users only)**

**Registered users** should have private part in the web application accessible after **successful login**.

The private part of the application consists of:

* Account dashboard page
* Transactions page

**Administration Part**

**System administrators** should have administrative access to the system.

The admin part of the application consists of:

* Login page for administrators
* User administration page
* Banner ad administration page

**General Requirements**

Your Web application should use the following technologies, frameworks and development techniques:

* Use **ASP.NET Core 2.2 MVC** and **Visual Studio 2017**
* You should use **Razor** template engine for generating the UI
  + You may use any JavaScript library of your choice
  + Use **sections** and **partial views**
  + Use **tag helpers**
* Use **MS SQL Server** as database back-end
  + Use **Entity Framework 2.1** to access your database
  + Using **repositories and/or service layer** is a must
* Use at least **2 areas** in your project (e.g. for administration)
* Create **tables with data** with **server-side paging and sorting** for a model entity
  + You can use Kendo UI grid, jqGrid, DataTables or any other library or generate your own HTML tables
* Create beautiful and responsive UI
  + You may use **Bootstrap** or **Materialize**
  + You may change the standard theme and modify it to apply own web design and visual styles
* Use **AJAX form** communication in some parts of your application
* Use **caching** of data where it makes sense (e.g. starting page)
* Apply **error handling** and **data validation** to avoid crashes when invalid data is entered (both client-side and server-side)
* Prevent yourself from **security** holes (XSS, XSRF, Parameter Tampering, etc.)
  + Handle correctly the **special HTML characters** and tags like <script>, <br />, etc.
* Create **unit tests** for your "business" functionality following the best practices for writing unit tests (**at least 80% code coverage**)
* Use **Dependency Inversion** principle and **Dependency Injection** technique following the best practices
* Use GitHub and take advantage of the **branches** for writing your features.
* **Documentation** of the project and project architecture (as .md file, including screenshots)

# Public part

## Home page with login box and banner ad (public)

* The home page should be available publicly.
* The homepage should display a random banner ad from all banner ads registered by an administrator in the administrative module.
* The homepage should display a login section with username and password fields
* The users should be able to login when correct username/password is provided

# Private area

The private part of the website should be available only to logged-in users. If an anonymous user tries to access any of these pages, they should be redirected to the login page.

It should have:

- A top service bar on all pages that displays the following information:

* The name of the currently logged-in user.
* Logout button

- Navigation bar on all pages with following menus:

* Accounts (link to Account dashboard page)
* Transactions (link to Transactions page)

Default page in private area after login is Account dashboard page.

## Account dashboard page (private)

* Account dashboard page should present a list of all accounts to which the user has access
* For each account in the list, following information should be presented:
  + Nickname
  + Account number
  + Current balance
* For each account additional options should be available:
  + Rename account – changes the nickname of the account
  + Make payment – initiates payment from respective account (pls see Make payment functionality later in the document)
  + View transactions – Presents all transactions on the account at the Transactions page (pls see Transactions page later in the document)
* If the user has more than one registered accounts, also a new section should appear on the page, with pie chart for all accounts with pie sizes proportional to the balance of each account. (for the purposes of this task, Kendo or other library by choice of the developer can be used for chart presentation)

## Transactions page (private)

* Transactions page presents a list of all payments made or received by the user

For each transaction, following information should be presented:

* + Payer account (initiating account – the account from which the payment is made)
  + Payer client name (Name of the client to which the payer account belongs)
  + Payee account (receiving account – the account to which the payment is made)
  + Payee client name (Name of the client to which the payee account belongs)
  + Description
  + Amount
  + Timestamp (date / time)
  + Indication if this is an outgoing or incoming transaction (+/-; red/green or other by choice of the developer)
  + Status (“Sent” or “Saved”)
    - If Status is “Saved”, then following buttons/links should be presented:
      * Send – action identical to “Send” option in Make payment functionality for respective payment (pls see Make payment functionality later in the document)
      * Edit – opens respective payment for editing as described in Make payment functionality with respective fields preloaded with already saved data)
* Transactions should be sorted by timestamp in descending order (newest first)
* User should be able to filter transactions by account (with a dropdown list of accounts, for example positioned above the transactions list). If the user is redirected to Transactions page from “View transactions” functionality for an account from Account dashboard page, then respective account should be preloaded in the account selection dropdown.
* Transactions should be presented in pages of 5 (with a navigation row positioned below transactions list with page numbers allowing the user to navigate to specific page and options “next page”, “prev page”
* On Transactions page also a Button/Link should be available to initiate a new payment (pls see Make payment functionality later in the document)

Transactions list for the user should contain only:

* transactions where payer account is registered for the user, no matter of status; or
* transactions where payee account is registered for the user and the status of payment is “Sent”

## Make payment functionality (private)

* + Payer account – dropdown to select an account from which the payment will be made
  + Payer client name (Name of the client to which the payer account belongs)
  + Payee account – the account to which the payment will be made -
    - Editable field - the user manually enters account number; while typing, after some symbols typed, a lookup is initiated to search for matching accounts in the database; the user then selects from result(s) found. In case of non-existing account, appropriate error message should be displayed.
  + Payee client name (Name of the client to which the payee account belongs) - non editable, to be filled in based on the selected payee account
  + Description (limited to 35 symbols)
  + Amount

All fields are mandatory.

* Make payments functionality can be initiated either from Account dashboard or from Transactions page.
  + If initiated from Account dashboard, the Payer account should be preloaded with the account from which the payment was initiated
  + If initiated from Transactions page, the Payer account stays empty by default
* Amount should not exceed the balance amount of the account
* Two options should be available:
  + “Save” – saves the payment with status “Saved” without any other action
  + “Send” –
    - saves the payment with status “Sent”, and also
    - updates respective account balances:
      * updates the balance of the payer account (subtracts the amount of payment from current balance)
      * updates the balance of the payee account (adds the amount of payment to the balance of respective account)

After saving or sending, the payment should be visualized in transactions list on Transactions page.

# Admin area

## Login

Administrative users should have a separate administrative site.

* The homepage should display a login section with username and password fields for administrators
* The users should be able to login when correct username/password is provided

For the purposes of simplification of this project, administrators with their usernames and passwords should be manually entered as records in the database by the developer.

## User management

Admins should be able to register clients, accounts and users with following functionality

Add client, with following attributes:

* Name (limited to 35 symbols)

-> Add accounts:

For each client – add accounts, with following attributes:

* Account number – 10 digit number
* (account balance) – initializes with a default balance of 10 000 BGN (configurable in system config)
* (account nickname) – initializes by default = account number

--> Add users:

For each client – add users, with following attributes:

* Name (limited to 35 symbols)
* Username (for login) (limited to 16 symbols)
* Password (for login) (limited to 32 symbols; to be stored as hash in the database)

--> Register accounts for each user:

For each user of the client the administrator selects the subset of accounts (from all accounts of the client) to which the user will have access.

## Banner management

Admins should be able to define banners with ads to be presented to users on user site home page.

- Add banner, with following attributes:

* Banner image (image with fixed size depending on the UI design implemented by developer)
* Banner link – a link with URL to redirect when the banner is clicked on
* Banner validity period – the banner should be visualized to users only during this period
  + Start date
  + End date

- Remove banner

### Partner Evaluations and Criteria

Partner representatives should evaluate the project according to the following criteria:

1. Feature Completeness and Quality
   * Have all requirements been met? Have optional requirements been implemented?
   * Are there any issues with performance, bugs, or malfunction?
2. Code Architecture and Design Quality
   * Loose coupling
   * Modularity
   * Extensibility
3. User Experience and Creativity

The final grade of the project is determined by the following rules.

Each of these 3 criteria are awarded points from 0-5, where 0 is the lowest score and 5 is the highest score:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 |
| Missing | Poor | Fair | Good | Very Good | Excellent |

Then the final project grade is determined by the following formula:

**[Feature Completeness] \* 3 + [Code Architecture Quality] \* 2 + [User Experience and Creativity] \* 1.**

Maximum number of points is 5\*3 + 5\*2 + 5 = 30.

This final score should be awarded by partner representative on **Dec 18**