# Practical Project Assignment for the [Software Technologies Course @ SoftUni](https://softuni.bg/courses/software-technologies)

Design and implement a **simple Web-based application**, e.g. **blog** / **forum** / **photo album** / **listings site** / **other**.

## Technologies by Choice

Use are free to choose the **technology stack** for your project development, e.g.

* HTML + CSS + **JavaScript** + AJAX + REST + JSON + Kinvey mBaaS
* **PHP** + MySQL + Apache + HTML + CSS
* C# + **ASP.NET MVC** + Entity Framework + SQL Server + HTML + CSS
* Java + **hyutyuyutyutertertwertertewtertertertert**  Hibernate + MySQL + Tomcat + HTML + CSS

You are allowed to use in addition other **technologies** like Bootstrap, SASS, LESS, MongoDB, Node.js, etc. You are allowed to use **frameworks** like AngularJS, ReactJS, Laravel, Symfony, Nancy, Spark, Play, JSF, etc. You are allowed to use **development tools**, **libraries** and **resources** like Web design templates, code generators, code libraries, JavaScript libraries, PHP composer packages, .NET NuGet packages, Java Maven artefacts and others.

## Team by Choice

You are allowed to work in a **team by choice** or to work **individually**.

Requirements for the **team projects**:

* All team members should be part of the “**Software Technologies**” course at SoftUni.
* **Onsite** and **online** students can work together (mixed teams).
* Teams can have **up to 6 members**.
* All team members should use source control repository (like GitHub).
* Larger teams should develop larger projects (see below).

Requirements for **individual** projects:

* Students are allowed to **work alone** without a team.
* Individuals should use source control repository (like GitHub).

Everyone should use a **source control system**.

* Use **Git** or other source control system for your project development.
* Use **GitHub**, **Bitbucket**, **CodePlex** or other team collaboration platform.
* Your **source code** should be open-source and **public** in Internet.

## Project Scope

Your project should implement **at least** the following **functionality**:

* User **registration**, **login** and **logout**.
* **View** some content (e.g. blog articles, listings, photos, issues, publications).
* **Create** new content (e.g. post new blog article, post new listing, upload new photo, create new issue).

Your project should keep its data in a **database** or in a **backend service**:

* Use at least **2 tables** (**collections**) with a relationship, e.g. users and blog posts.
* Use a **database** (like MySQL or MongoDB) or cloud-based backend (like Kinvey, MongoLab or RedisLab).

Your project should implement **at least 4 pages** (views).

You are allowed to create a project which is **very** **similar to the “Blog System”** developed during the course. You are allowed even to take the Blog System source code and modify it for your needs.

### Requirements for the Individual Projects

Individual projects can be **small**, like the “Blog System” developed during the course.

* Minimum **4 pages** (views) and **2 database tables**.
* Implement user registration, login, view content, create content.
* **Optionally** implement more functionality.

### Requirements for the Team Projects

Larger teams should develop **larger projects**. The minimum number of application pages and database tables required for the project depend on the **count of team members**.

* Minimum (3 + **team\_members\_count**) **pages** (views).
* Minimum **team\_members\_count** **database tables** (or data collections / entities).
* Implement user registration, login, view content, create content.
* Implement **more functionality** by choice. Larger teams should implement more functionality.

## Forbidden Techniques and Tools

* Your project should be created mainly by **you and your team**.
* You are **not allowed to copy a project from Internet** and present it as your development.
* You can use external libraries, frameworks and tools, but **not to clone a project** and present it as yours.

## Commit Logs

* Each team member should have **at least 5 commits** (changes) in the project repository.
* Please **commit every day** during the project development to demonstrate your work progress.
* More commits (especially in more than the last 1-2 days) are better during the project assessment.

## Deliverables

Submit the **URL of your project source code** as deliverable, e.g. <https://github.com/nakov/TurtleGraphics.NET>. Each team member submits the same **source code URL**. Put the following assets in the project repository:

* The complete **source code** of your project (PHP, Java, C#, JS, HTML, CSS, images, scripts and other files).
* Any other project assets (optionally): documentation, design, database sample data scripts, tests, etc.
* Don’t commit the **libraries** that can be automatically downloaded by the **package manager** (npm packages, composer packages, NuGet packages, maven packages).

## Public Project Defense

Each team will have to deliver a **public defense** of its work in front of the SoftUni trainers.

The teams will have **only ~15 minutes** for the following:

* **Demonstrate** the application’s functionality (very shortly).
* Show the **source code** and explain briefly how it works.

At least **one team member** should come at the defense.

Hints for better presentation:

* Be **well prepared** for presenting maximum of your work for minimum time.
* **Open all project assets** beforehand to **save time**: open your site in the browser, login and open the user / admin panel in another browser, open your GitHub project page to show the commit logs, etc.
* Test how to connect your laptop with the **multimedia projector** before the defense to save time.

## Assessment Criteria

* **Functionality** – **0…70**
  + **What** is implemented? Does it work correctly? Does it have intuitive UI?
  + How much **effort** you have put in this project?
  + Is the functionality **enough** for the team size (larger teams should deliver more)?
  + What portion of the work is **own code written by your team** and what is ready-to-use framework?
* **Teamwork** – **0…30**
  + **Individual projects** show the **commit logs** in the source control repository.
  + **Team projects** show the **commit logs** and explain the **role of each team member**.
* **Bonus** – **0...10**
  + Bonus point are given for implementing more than expected.

## Sample Projects

The below described projects are **sample**, just to give you some **ideas**. You could work on your own project.

### Issue Tracker

Design and implement a simple **issue tracking system** (bug tracker).

**Required** functionalities:

* User **registration** (and optionally user profiles) / **login** / **logout**.
* **View** **all issues** (optionally with paging), without login.
* **View issue** details, without login.
* Create **new issue** (after login). Issues have title, description, author, state and submission date and time. States are: New, Open, Fixed and Closed.

**Optional** functionalities:

* **Edit issue** (after login). Can change only title, description and state.
* Add **new comment** for existing issue by visitors – each visitor must fill out his name and comment text.
* Implement a sidebar holding a **list of all issues states**. Clicking at issue state shows all issues matching this state.
* Functionality for **searching** by title (as substring).
* **Admin panel**: add / edit / delete issues and comments, etc.

### Blog

Design and implement a simple **blog system**.

**Required** functionalities:

* User **registration** (and optionally user profiles) / **login** / **logout**.
* **View all posts** (optionally with paging), without login.
* Create **new post** by the blog owner (after login). Optionally, each post may have **tags**.

**Optional** functionalities:

* Add **comments** for every post by visitors – each visitor must fill out his name, email (optionally) and comment text.
* Implement a sidebar holding a **list of posts** sorted by month / year / etc. and a list of the **most popular tags**.
* **Counter of visits** for each post.
* Functionality for **searching** by tags.
* **Admin panel**: add / edit / delete posts, comments, tags, etc.

### Forum

Design and implement a simple **forum** (discussion board) system.

**Required** functionalities:

* User **registration** (and optionally user profiles) / **login** / **logout**.
* **View** all questions (optionally by category, optionally with paging), without a login.
* Ask a **new question** by the forum users (after login). Optionally question may have **tags** and **category**.

**Optional** functionalities:

* Implement **tags** for the forum questions.
* Implement **categories** for the forum questions.
* Implement adding **answers** to the questions by the forum visitors – each visitor must fill out his name, email (optionally) and comment text.
* **Counter** for visits for each question.
* Functionality for **searching** by question, answer and tags.
* Implement **ranking** according to user activity.
* **Admin panel**: add /edit / delete forum posts, tags, answers, categories.

### Photo Gallery

Design and implement a simple **photo gallery** (photo album).

**Required** functionalities:

* User **registration** (and optionally user profiles) / **login** / **logout**.
* **Browse albums** and **photos** (and optionally categories, optionally with paging), without a login.
* **Upload photos** (after login, optionally validate image size and type) / **download** photos.

**Optional** functionalities:

* **Create new album** in a category.
* Add **comments** to photos and albums.
* Implement album's **ranking system** (e.g. vote from 1 to 10 or like / dislike).
* Show the most **highly ranked** albums in a special section at the main page.
* Functionality for **searching** by album name / category.
* **Admin panel**: add / edit / delete albums, photos, comments.

### Music Catalog

Design and implement a simple **online music catalog** holding songs and playlist.

**Required** functionalities:

* User **registration** (and optionally user profiles) / **login** / **logout**.
* **View** all playlists / songs (optionally with paging), without a login
* **Listen** to songs online. **Download** songs (optionally).
* **Upload songs** (optionally validate file size and type), after login.

**Optional** functionalities:

* Create a **new playlist**.
* Add **comments** to songs and playlists.
* Implements **genres** for the songs and playlists.
* Implement playlists' and songs' **ranking system**. Show the most **highly ranked** playlists in a special section at the main page.
* Functionality for **searching** / **filtering** by playlist name / song name / genre.
* **Admin panel**: add / edit / delete songs, playlists, genres, comments.

### Ads Listing Site

Design and implement a simple **ads listing site** where users publish their ads and visitors can browse and view them.

**Required** functionalities:

* User **registration** (and optionally user profiles) / **login** / **logout**.
* **View** all ads (optionally by category and town, optionally with paging), without a login.
* **Post a new ad** (optionally with a photo), after login.

**Optional** functionalities:

* Implement **towns** and **categories** for the ads. Implement browse by category / town.
* Post new ads in state “**Waiting approval**”. Administrators can see and approve them.
* Implement a **mini-photo gallery**: each ad could hold a set of **photos**. Users can upload / delete photos. Visitors can view the photos for each ad. One of the photos is designated as **primary**.
* **My ads panel**: view / edit / delete own ads (posted by the current user).
* **Admin panel**: add / edit / delete users, ads, categories, towns.

## Deadline

All projects should be submitted not later than **31-August-2016**.