# General requirements

This section describes requirements to all projects.

## Source code

* Every project should use version control system and source code management, best choise would be GIT or TFS.
* Every project should use <https://bitbucket.org/> or https://github.com/as repository (can use similar system with the mentors agreement, for example TFS). Each team member should have own login credentials. List of login names should be provided to the mentor to be able to identify each member of a team. Team should provide access to the bitbucket GIT repository to mentor, in case of lack of free account one of the team members should provide own credentials. Not private repositories can be used.
* Teams should use github’s or bitbucket’s task tracking systems for managing tasks or bugs. Either any other free Agile board can be used.
* Static analysis tools for code quality should be used on every project. Every project should be developed with no warnings from the analysis tools
  + FindBugs and checkstyle should be used for the Java projects.
  + ReSharper and Style Cop for the .Net projects.
* Optional: Continuous integration tools can be used such as TeamCity or Jenkins
* Optional: Sonar can be used as static analysis tool for any type of projects but it should be agreed with mentor.

## Workflow

Each task or bug-fix should follow correct workflow. The final workflow should be agreed with mentor. If not agreed it should be next:

For tasks

* Task created in tracker (New)
* Task assigned to developer to develop (Assigned)
* Task ready for review (For review)
* Task assigned to developer for code review (In review) – from this State task can be moved either to next state or be assigned back to developer responsible for development.
* Task ready for functional testing (Ready to test)
* Task in test (In test) – can be moved back to (Assigned) state or to (Done)
* Task closed (Done)

Task cannot be developer reviewed and tested by the same developer, all these activities should be done by separate persons. In case of only two developers on a project, code review and functional test duties can be joined.

For bugs:

* New or Reopened or Cloned
* Assigned
* In review
* Ready
* In test
* Done

## Quality

* Each final product should be tested. Testing is done on a pear to pear basis, this means each developer should test features developed by colleague.
* Each developed feature should be tested.
* Definition of done: feature is developed, code is reviewed, functionality is tested and satisfying to requirements.
* Test Cases for every feature and for every found bug should be created. Test cases are the part of final delivery and feature demonstration. Test Cases should be stored in the separate project’s folder.
* Each developed feature should be code reviewed by the team member not participated in the development of this feature.
* No bugs with the severity normal or higher should be present in the delivered features. (Delivered feature is a feature demonstrated on the demo meeting).
* Each team member should participate in testing activities.
* Not less than 60% of code should be covered by unit tests.

## Reporting and demonstration

* Detailed User Stories for projects should be described by teams and agreed with mentors.
* Full backlog should be sored together with the projects files on GIT repository.
* Time of demonstration meetings should be agreed with mentor. It is recommended to demonstrate weekly.
* Weekly reports should be sent to mentors with next data:
  + Status (green/yellow/red) with the small description
  + Current features in development (list)
  + Already developed features (list)
  + Entire number of Test Cases done till the reporting time
  + Entire number of Unit tests done till the reporting time

# Tasks for Teams

|  |  |  |
| --- | --- | --- |
| # | Title | Description |
| 1 | Single-Page Application | **Overview**  Create a single page web application. The application must use KnockoutJS and RequireJS on the client-side, Ninject and Entity Framework Code-First on the server-side.  **Requirements**   * Create a single page web application. The application must use KnockoutJS and RequireJS on the client-side, Ninject and Entity Framework Code-First on the server-side. * The application shall have two screens; the user can switch between these screens without reloading the whole page. * On Screen A, the user can see the list of the products (name, price, description), can add or remove products, or modify products. They must be stored in the database. Changing an existing product, creating a new product or deleting a product must not cause the whole page to be reloaded, but the item must be sent to the server and stored in the database. Every operation must be logged. Log entries must contain a timestamp, and a message: create, update or delete product. * On Screen B, the user can see the log entries. They must be ordered by the timestamp (latest first), and they must contain the date (formatted according to the users current locale settings), and the log message. * Switching between these screens must re-populate the values from the server (refresh the lists), but without reloading the whole page. * The logging and the products database schemas must be defined in separate Database Contexts and separate Visual Studio projects. * The tool can have a startup project (Asp.NET MVC5 project), the domain projects (Logging and Products), and an Extensibility project (class library to define the interfaces). * The startup project must not have direct reference to the domain projects! It can only reference the Extensibility project. However, it is allowed to add the domain projects to the build dependency of the startup project. Extensibility project cannot have reference to the startup project, nor the domain projects. It can only reference NuGet packages and built-in namespaces. * Domain projects must use the repository pattern to manipulate data. The repository interface definitions must take place in the Extensibility project, and the implementation shall be in the corresponding domain project. Each domain project should bind these interfaces in their own Ninject module. The startup project can inject these interfaces in the corresponding Controllers. It is allowed to have DTO (Data Transfer Object) classes for products and log entries in the Extensibility project. The repositories must not return entity objects, but DTO objects. * Client-server data manipulation should be implemented in WebApi2 controllers.   **Optional**  NA  **Tips**  NA |
| 2 | UML Class diagram editor (Web-based) | **Overview**  The idea is to implement visual diagram editor that allows to create UML class diagrams and generate source code out of them.  **Requirements**   * Diagram editor should support all features of UML2 standard class diagram. * Editor is able to create new class diagram. * Open existing diagram. * Edit existing diagram. * Save diagram to file. * Editor should be able to generate created classes and dependencies to Java source code. * Optional: Editor should be able to export diagrams into image format. |
| 3 | Mind map editor (Web-based) | **Overview**  The idea is to implement visual editor that allows creating mind maps online.  **Requirements**   * Editor is able to create new mind map. * Open existing mind map. * Edit existing mind map. The following items should be supported   + Nodes/subnodes: add sibling/child   + Add comment   + Add marker * Save mind map * Export mind map to file (preferable using one of the standard mind map formats). * Optional: Editor should be able to export diagrams into image format. * Optional: Support omega-mapping (when map is drawn from both ends) * Optional: Apply different styles to the map |
| 4 | Course Management System | **Overview**  The task is to develop Web-based application that is dedicated to provide online courses to the students. There are two main roles in the system – Lecturer and Student. Lecturer is able to create new courses (courses consist of lessons with educational materials and questionnaires followed by an exam), keep track of the students’ progress, evaluate examination results and confirm that the course is passed or not. Student is able to subscribe to a particular course and proceed through the education.  **Requirements**   * Lecturer should be able to create new courses, lessons and questionnaires and populate them with the actual data, attach related materials, etc. * There should be an email notification mechanism to notify about system events (subscriptions, progressing, etc). * Student should be able to subscribe/unsubscribe from a particular course, proceed through the lessons, and pass tests based on the questionnaires. * Optional: There should be a mark calculation system based on questionnaires results. * Optional: Lecturer should be able to keep track of the students’ progress, reject or accept questionnaires results. * Optional: There should be a possibility for the participants to communicate, provide and receive feedback, comments, etc.   **Optional**   * Application responses to any request within 2 seconds and constant load of 500 users |
| 5 | Lecture Scheduling Tool (Web-based)  Hard task with a lot of research needed. | **Overview**  The task is to create web-based tool that will help school or university to schedule courses and lessons in an optimized way. User should be able to load data such as availability of resources (lecture halls, laptops, technics), lecturers with some constraints in a schedule (for example some lecturers can train during all time, and some have only mornings or evenings or some time when are busy), and student groups with a numbers of needed resources.  **Requirements**   * Algorithm should use one of the newest Artificial Intelligence algorithms. * The tool should be able to create and schedule lessons at the specific time in the specific classroom for the concrete group of students in a non-conflict way. * The tool should be able to manage student classes (groups) and curriculums and generate corresponding schedule for a particular group, student or lecturer on demand. * The tool should provide some means of classroom management, availability and capacity of classrooms, keep track of classrooms assignments, etc. * Tool should be able to print to formatted file Lecture Schedule containing: room number, lecturer name, students group, time   **Optional**   * Scheduling and data-retrieval functions should be available via web-service * Web-service responses to any request within 2 seconds and constant load of 500 users |
| 6 | Extended travelling salesman problem.  (TSP). Hard task with a lot of research needed. | **Overview**  TSPis an actual task to find shortest and cheapest possible route for the special bus for disabled persons. Company has some number of special busses of next 3 types (8 places for disabled persons, 4 for disabled + 4 for ordinary, 6 for disabled +2 for ordinary). Company is providing services for disabled persons and their companions such as scheduled trips. Client of this company can reserve some trip to tomorrow or to some number days ahead. Application should be able to get files with number of cars available, and list of passengers with routes. As an output it should produce schedule for each car. As fast as possible.  **Requirements**   * Calculate the shortest and cheapest distance, number of busses needed for the company working as a buss owner. Also the roadmap should be present. (Based on a Artificial intelligence algorithms) * Route should be shown on some online map. * Route should be real one. * Calculated roués should be stored in the internal DB * Company has few different school busses. * Passenger should not wait more than 5 minutes on a buss stop * Boarding time is 5 minutes. * Arrival time should be 15 minutes earlier than actual requested time. * Passengers are given in the .xls file with next format:  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | id | address | date | name/surname | disabled | Arrive time |  | |  |  | |  |  |  |  | |
| 7 | Online animation editor (css3 and html5) | **Overview**  Creating the online animation editor without the Flash technology based on the css3 and html 5 technologies.  **Requirements**   * Software should be able to work online * User should be able to create animated scenes and banners * Editor should work with vector graphics * User should have a reach palette of vector elements * User should be able to load images to the animation * User should be able to import animation to gif or avi (with some common codecs) |
| 8 | Cloud-based fb2 reader | **Overview**  Book reader for web and mobile devices with ability to store book collection in a cloud and synchronize reading stats.  **Requirements**   * Loading books from fb2 files * Working with OPDS streams to connect to electronic libraries * Connecting to cloud storage for storing books (e.g. dropbox) * Web and mobile clients with synchronization (so you can read the book on mobile while you’re traveling, then switch to a web interface from home and continue reading from the same place you’ve stopped on mobile) * Changing the test size by simple gestures (user should stay on the same page) * Moving from page to page with simple gestures * Optional: change highlight by simple gestures |
| 9 | Photo albums sharing on Google maps | **Overview**  The main goal of the project is to develop a prototype of the social network allow publishing a photo albums on the Google maps.  **Requirements**   * System should support next roles:   + Guest (non authorized user, can only view static pages like description of the system and rules)   + User (view and create albums on a map, edit and delete own albums, comments to published albums (own or friends), following of friends subscriptions, profile configuration)   + Administrator (full access to the system, can manage users) * User should be able to register the account with the unique e-mail address. E-mail should be confirmed. Users with not confirmed e-mails are guests. * User can publish an album and map it to the place on the Google map. * Users can became friends. * User can subscribe to the friends and get all friends updates by mail. * User can unsubscribe from the friend’s updates. * User can comment on the own and friends albums. * User can remove/delete own albums and comments. * User can view his and friend’s maps with markers showing the albums. * User can we his or friends albums. * Administrator can manage all albums, comments, users and maps.   Albums should be stored in the DB |
| 10 | The Video Rental Store | **Overview**  Implement the administration system for The Video Rental Store:  **Requirements**  **Primary Features** (must be implemented)   * Support the pricing and bonus program advertised * Produce rental and return receipts detailing films, rental periods and amounts (both money and bonus points) involved * Manage films (add/remove and change category) * Manage customers (add/remove and debit/credit bonus points)   **Secondary Features** (if time allows)   * Inventory lists (i.e. all films, films available and films currently rented) * Customer lists (i.e. all customers with points, customers with current rentals and rented films)   **Tertiary Features** (for extra credit)   * Support individual rentals and returns (i.e. a customer can rent two films and return them on different days) * Support simultaneous rentals (i.e. films can be rented by different customers at the same time) * Support multiple instances of films (i.e. 2 copies of Bladerunner available) |
| 11 | Application for finding people on a photo | **Overview**  Create an application able to analyze photo using Machine Learning algorithms and available libraries to find is there person on a photo or not.  **Requirements**  The page should have the following functionality:   * Application should be configurable to check some image formats in a configurable folder; * Mark photo by yes or no depending on a presence or absence of human on a photo.   **Bonuses:**   * Caffe can be used; |
|  |  |  |
|  |  |  |
|  |  |  |