Assignment 1	Project Summary
Course	**Web Application & Service Development with Spring Framework v5 - 2020**

Project author		
№	Pseudonym	Face-to-face/ online
1	BKTS	online

Project name	Jyra Project Management Tool (JPMT)

1. Short project description (Business needs and system features)

Nowadays as we see a rapid growth in software development, the organizations should be provided with flexible software to manage their teams and projects. The ***Jyra Project Management Tool*** provides the ability for users to define and manage new Projects, Sprints and Tasks, as well as to report their progress. In addition to that it allows users to register, and administrators to manage them. The system will be developed using ***Spring 5 Application Development Framework***. It will implement a web-based front-end client using ***Thymeleaf*** templates and ***jQuery***. Each page will have a distinct URL, and the routing between pages will be done server side using ***SpringMVC***. The backend will be implemented as a ***REST*** using JSON data serialization. The main user roles (actors in UML) are:

- *Anonymous Users* – can only view the information page, login and register.
- *Developers* (extends *User*) – can add new Sprints and Tasks, as well as to verify/report completion of TaskResults and SprintResults for different Tasks and Sprints.

- *Product Owner* (extends *User*) – can add new Projects and Sprints and Tasks, as well as to verify/report completion of TaskResults, SprintResults and ProjectResults for different Tasks, Sprints and Projects.
- *Administrator* (extends *User*) – can add new Projects. In addition can manage all User's data, except their passwords.

2. Main Use Cases / Scenarios

Use case name	Brief Descriptions	Actors Involved
2.1. Browse Projects, Sprints and Tasks	The *User* can browse the information views (Projects, Tasks, Sprints, results for the previously listed and About) in *JPMT*.	All registered users
2.2. Register	*Anonymous User* can register in the system by providing a valid e-mail address, first and last name, and choosing a password. Admins can not be registered.	*Anonymous User*
2.3. Change User Data	*Registered Users* can manage their own personal data and change their passwords. *Administrators should be able to manage all User's data, except their passwords* and assign them *Roles*: *Developer, Product Owner*, or *Administrator*.	*Registered User, Administrator*
2.4. Manage Users	*Administrator* can browse and choose a *User* to manage, and can manage the chosen User - edit or delete.	*Administrator*
2.5. Manage Projects	All users can browse projects.	*Administrator, ProductOwner*

	Administrators can create projects. *ProductOwners* can create and complete projects.	
2.6. Manage Sprints and Tasks	*ProductOwners* and *Developers* can add new Sprints and Tasks to existing Projects, as well as to verify/report completion of SprintResults for different Sprints. In addition, *Developers* can create TaskResults for completed tasks.	*ProductOwners, Developers*
2.7. Add/Edit Sprint	*ProductOwners* and *Developers* specify/edit *Sprint* meta-data such as: start date, end date, owner, developers, tasks, and completedTaskResults.	*ProductOwners, Developers*
2.8. Add/Edit Task	*ProductOwners* and *Developers* specify/edit *taks's kind, title, task's creator, estimated effort points, status, sprint which the task belongs to, assigned developers, description, tags.*	*ProductOwners, Developers*
2.9. Complete Task	*Developers* can change the status to Done.	*Developers*
2.10. Complete Sprint	Sprints can be completed by *ProductOwners* and *Developers.*	*ProductOwners*, *Developers*

3. Main Views (Frontend)		
View name	**Brief Descriptions**	**URI**
3.1. About	Presents the main information for the purpose of the system. Prominently offers ability to register.	/index
3.2. User Registration	Presents a view allowing the *Anonymous Users* to register in *JPMT*.	*/register*

3.3. Login	Presents a view allowing the users to login.	*/login*
3.4. Tasks	Presents tasks available. Offers abilities to browse, search and view.	*/tasks*
3.5. View Task	Presents information about a specific task such as title, description, type, status, assignee, reporter, sprint, task estimate and result.	*/tasks/{id}*
3.6. Backlog	The main view for the backlog. Shows which tasks are included in the project, as well options to create and edit tasks and create sprints.	*/projects/{projectId}/backlog*
3.7. User Data	Provides ability to view the user's profile. It includes information about username, names, email, status, and list of assigned and completed tasks. Each user can manage his/her own profile.	*/users/{userId}*
3.8. Board	The board presents the main information for the current sprint such as sprint name, remaining time, participants in the sprint, owner, the main four statuses for each task: To Do, In Progress, In review, Done and list of tasks in every status.	*/projects/{projectId}/board*
3.9. Users	Presents list of all Users. *Administrators* can edit User's data and delete the users.	*/users*
3.10 Project	Presents the main information about the project such as title, project owner, description, start date and participating developers.	*/projects/{projectId}*
3.11 TaskResult	Shows information regarding the result of the task such as task name, reporter, actual effort and result description.	*taskresults/{taskId}/task-result*

3.12 SprintResult	Shows information regarding the result of the sprint such as sprint name, team velocity, description and list of the completed task in the sprint.	*sprintresults/{sprintId}/sprint-result*
3.13 ProjectResult	Shows information regarding the result of the project such as project title, end date, duration, description and list of sprint results.	*projectresults/{projectId}/project-result*

4. API Resources (Backend)		
View name	**Brief Descriptions**	**URI**
4.1 Users	GET *User Data* for all users, and POST new *User Data* (Id is auto-filled by *JPMT* and modified entity is returned as result from POST request).	*/api/users*
4.2 User	GET, PUT, DELETE *User Data* for *User* with specified *userId*.	*/api/users/{userId}*
4.3 Projects	GET all *Projects*, and POST new *Project* (Id is auto-filled by *JPMT* and modified entity is returned as result from POST request).	*/api/projects*
4.4 Project	GET, PUT information for the project with the specified id. DELETE project with the specified id.	*/api/projects/{projectId}*
4.5 Boards	GET all *Boards*, and POST new *Board* (Id is auto-filled by *JPMT* and modified entity is returned as result from POST request).	*/api/boards*
4.6 Board	GET, PUT information for the board with the specified id. DELETE board with the specified id.	*/api/boards/{boardId}*
4.7 Tasks	GET tasks and POST new *Task*(Id is auto-filled by *JPMT* and modified entity is returned as result from POST request).	*/api/tasks*

4.8 Task	GET, PUT, DELETE *Task Data* for *Task* with specified *taskId*.	*/api/tasks/{taskId}*
4.9 Task Results	GET all *Task Results.*	*/api/tasks/task-results*
4.9 Task Result	GET *Task Results* for *Task* with specified *taskId*, and POST new *Task Result* for the specified *taskId*(Id is auto-filled by *JPMT* and modified entity is returned as result from POST request). PUT new information about the *Task Result* (modified entity is returned as result from PUT request). DELETE *Task Result*.	*/api/tasks/{taskId}/task-result*
4.10 Sprint Results	GET all *Task Results.*	*/api/sprints/sprint-results*
4.11 Sprint Result	GET *Sprint Result* for the Sprint with specified *sprintId*, and POST new *Sprint Result* for the specified *sprintId* (Id is auto-filled by *JPMT* and modified entity is returned as result from POST request). PUT new information about the *Sprint Result* (modified entity is returned as result from PUT request). DELETE *Sprint Result*.	*/api/sprints/{sprintId}/sprint-result*
4.12 Project Results	GET all *Project Results.*	*/api/project/project-results*
4.13 Project Result	GET *Project Results* for *Project* with specified *projectId*, and POST new *Project Result* (Id is auto-filled by *JPMT* and modified entity is returned as result from POST request). PUT new information about the *Project Result* (modified entity is returned as result from PUT request). DELETE *Project Result*.	*/api/projects/{projectId}/project-result*

5. Frameworks and libraries	
Name	**Brief Descriptions**
5.1. jQuery	jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API
5.2. Bootstrap	Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains CSS and JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components.
5.3. Lombok	Project Lombok is a java library that automatically plugs into your editor and build tools, spicing up your java. Never write another getter or equals method again, with one annotation your class has a fully featured builder, Automate your logging variables, and much more.
5.4. CommonMark	Core of commonmark-java (implementation of CommonMark for parsing markdown and rendering to HTML)
5.5. Swagger	Swagger is an Interface Description Language for describing RESTful APIs expressed using JSON. Swagger is used together with a set of open-source software tools to design, build, document, and use RESTful web services.
5.6. Select2	Select2 gives you a customizable select box with support for searching, tagging, remote data sets, infinite scrolling, and many other highly used options.

6. Architecture

Jyra PMT follows a layered architecture in which each layer communicates with the layer directly below or above (hierarchical structure) it. There are three layers in Jyra PMT are as follows:

- Presentation Layer - In short, it consists of views i.e., frontend part.
- Business Layer - It consists of service classes and uses services provided by data access layers.
- Database Layer - In the database layer, CRUD (create, retrieve, update, delete) operations are performed.