Assignment 1	Project Summary
Course	Full Stack Application Development with Node.js + Express.js + React.js - 2024

Project author		
	Name	FN
	Kristian Petrov	8MI0600201

Project name	Project Master

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

Proper Project Management is an essential part of every successful business. The Project Master brings the agile project management to a completely new level. It provides functionalities for account management, task management, team management and board management. **Client-Server** will be the applied architectural style. The system Frontend will be developed using **React.js**, while **Node.js + express** will serve the backend, utilizing a ***REST/JSON API***. As a database, we've chosen **MySQL**. Additional features will be **Oauth** and eventually **Sockets** for notifications.

The main **roles** are:

- **Worker** - registered user, can create and participate to projects and teams
- **Administrator** (extends Worker) - monitor the application for issues and misuse
- **Project Worker** (extends Worker) - in the context of a project, can create and modify issues
- **Project Moderator** (extends Project Worker) - can manage issues (creating a custom issue), add and remove Project Workers
- **Project Owner** (extends Project Moderator) - can delete and configure the project

2. Main Use Cases / Scenarios

Use case name	Brief Descriptions	Actors Involved
2.1. Register Worker	Worker can register using his account in Google or with email, password, and names. Then, he should provide more information like Birthday and this from point 2.3.	Worker
2.2. Login Worker	Worker can login with Google or his email and password.	*Worker*
2.3. Manage Worker Account	Worker can manage and view his Organization, Department, Position, Location and contacts.	*Worker*
2.4. Manage Project	Worker can create a project which makes him a project owner. Then he can delete and update the project information like (name, description, location, tags) He can invite and remove workers from the project. - CRUD project	*Worker*
2.5. Board Management	Every project has a board (Kanban) which shows the issues to be done. The worker which is a part of the project can view and modify them. Moderator Worker can modify the board	*Worker*
2.6. Issue Management	Every worker part of a particular project can - CRUD issues for the particular project - Make a comment to an Issue - Attach files to an Issue	*Worker*
2.7. Team Management	CRUD teams Add and remove members from a team by the Team Owner	*Worker*
2.8. Browse Issues for a Project	Project Worker can browse and filter all the issues for a project he is participating in	Worker
2.9. Browse Issues for a Team	Team Worker can browse and filter all the issues for a team he is participating in	
2.10.		
2.11.		

2.12.		

3. Main Views (React Frontend)		
View name	**Brief Descriptions**	**URI**
3.1. Start page	Presents the introductory information for the purpose of the system as well as detailed instructions on how to start using it. - Introduction text - Register Option - Login Option	/
3.2. Register Page	Presents the workers register page, requires - Email - Real Names - Display Name - Password twice - Birthday - Location - Organization - Position - Option for Oauth register	*/register*
3.3. Login page	Presents a form where the user can enter login information, components - Email input - Password input - Option for Oauth	*/login*
3.4. Home page	Presents the workers home page - Header with links (part of every view below) to boards… - Projects he is part of - Issues assigned to him - Issues he has seen	*/home*
3.5. Account Management page	User can view and change his information - Display Names - Location - Organization - Position	*/account*
3.6. Create project page	Presents a form with the following inputs: - Name - Description - Choose from project templates - for the moment we have one project template - Location - Tags - Members	/project-create

3.7. Project Details	View and change project details, accessible by the project owner - Invite and remove a member	*/project/:projectKey/details*
3.8. Project Board	Presents a view of the project boards - Can open an Issue component	*/project/:projectKey/board*
3.9. Project Issues	Presents a view with the issues and ability to filter them by name, label, assignee, status, and and paginate	*/project/:projectKey/issues*
3.10. Create Issue Page	Presents a form for issue creation, includes - Name - Description - Status - Assignee - Project - Labels - Attachments - Other custom inputs	*/issue-create*
3.11. Issue page	Presents a view of the **issue information**, including **comments** We can also update the issue	*/project/{:projectKey}/issues/{:issueKey}*
3.12. Create Team Page	Presents a form with information about the team: - Name - Description - Tags - Members - Location	*/team-create*
3.13. Team Page	Presents information about the team members, active issues and projects	*/teams/{teamKey}*

4. API Resources (Node.js Backend)		
View name	**Brief Descriptions**	**URI**
4.1. Auth	Authentication route - /register	*/api/auth*

	- /login - /logout	
4.2. Users	Get many users, accessible only by the admin, with filters and pagination	*/api/users*
4.3. User	GET and PATCH the user information - Only the direct user information	*/api/users/{:userId}*
4.4. User projects	GET all projects a user is member of POST add project to the user	*/api/users/{:userId}/projects*
4.5. User issues	GET all user issues, with filters, type and pagination	*/api/users/{:userId}/issues*
4.6. Projects	GET all projects with pagination and filtering, only by the admin POST create a new project	*/api/projects*
4.7. Project	GET, PATCH - Change the information about the project - Archive - Get the basic information about the project	*/api/projects/{:projectId}*
4.8. Invite a project member	POST - email	*/api/projects/{:projectId}/invitations*
4.9. Accept a project invitation	GET	*/api/projects/{:projectId}/invitations?token={token]*
4.10. Issues	GET with filters like assignee, project and team and pagination POST create a new issue	*/api/issues*
4.11. Issue	GET, PATCH, DELETE an Issue	*/api/issues/{:issueId}*
4.12. Teams	GET, POST	*/api/teams*
4.13. Team	GET, PATCH, DELETE	*/api/teams/{:teamId}*