System and Software Architecture Description (SSAD)

Software Quality Analysis as a Service (SQUAAD)

Team No.1

Implementers:
Aleksandr Chernousov
Chris Harman
Supicha Phadungslip

Testers:
Kavneet Kaur
Reza Khazali
George Llames
Sahar Pure

Version History

Date	Author	Version	Changes made	Rationale
10/15/17	RA	1.0		● Initial draft for use with SQUAAD v1.0
12/1/17	RA	1.1		● Initial draft for use with SQUAAD v1.1
02/16/18	SP	2.0	Updated section 2.1.3, 3.1.3	● Added backend communication ● Updated sequence diagrams ● Updated system behavior
04/26/18	SP	3.0	Updated section 1.2, 3, 4	● Updated SSAD status ● Updates Class Diagram and component diagram

Table of Contents

Table of Tables

Table of Figures

1. Introduction

1.1. Purpose of the SSAD

The purpose of this document is to present a high-level schematic of the design of the SQUAAD software and hardware system. This document provides tables and figures that show the components that make up the system, how they interact with each other, and what infrastructure is required for implementation.

For the SSAD of the project, we provide the following:

- Design physical and logical Architecture of the system
- Design Use Case Diagrams
- Use Case Diagrams Descriptions

1.2. Status of the SSAD

System Analysis Overview, System Analysis Rationale, Design Overview and Design Rationale has been added. This is the final SSAD information for April 2018 release.

2. System Analysis

2.1. System Analysis Overview

The primary purpose of the Software Quality as a Service System is to track the quality of the software projects based on various quality metrics. With this system, users can register, become the contributor of the project, add other users as the contributor, send request for analyzing their project, receive notification when their request is done. Also, the system has an admin, who is responsible for managing users, user's permissions, project existing in the system and are sent for getting analyzed.

2.1.1. System Context

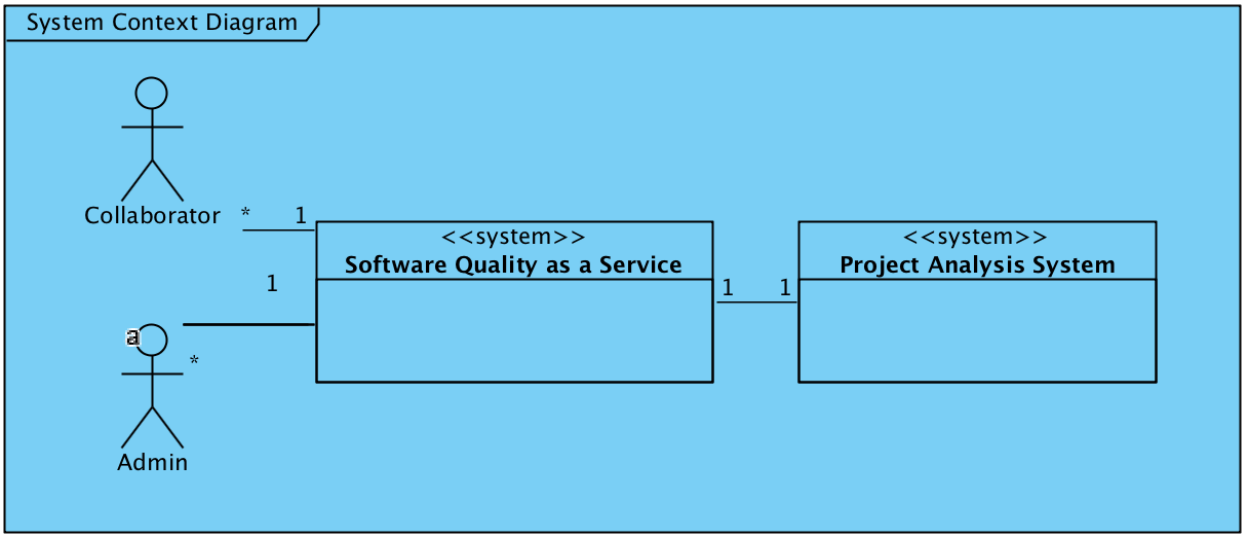

Figure 1: System Context Diagram

Actor	Description	Responsibilities
Collaborator	User who uses the system to analyze his project.	• Using the product for analyzing their software based on related software quality metric. • Submit project to the Software Quality as a Service system.

Admin	Admin is an internal user who is in charge of controlling software projects adding to the system, users adding to the projects, and projects sent to the system for analysis.	• Grant and revoke user's permission • Approve or reject add project request.

Table 1: Actors Summary

2.1.2. Artifacts & Information

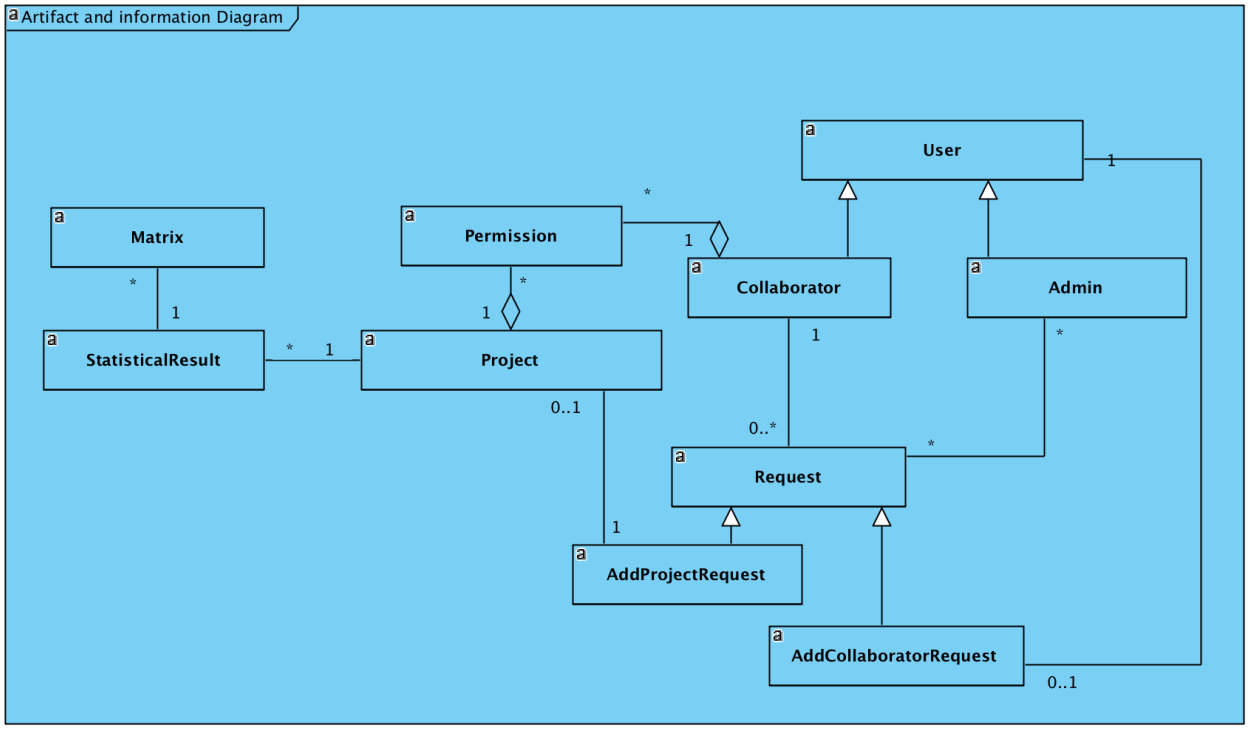

Figure 2: Artifacts and Information Diagram

Artifact	Purpose
User	Contain user's information such as first name, last name, username, password and role
Collaborator	Contain collaborator's operation. This artifact has relation with permission and add collaborator request.
Admin	Contain admin's operation
Request	Contain request's information such as created time, creator, status and approve detail.

Add Project Request	Contain project detail that user would like to add.
Add Collaborator Request	Contain collaborator and project detail that user would like to add.
Permission	Contain list of user permission. If user can access project, it must has entity contain the user and project in this artifact.
Project	Contain project's information such as project name, project URL.
StatisticalResult	Contain statistical data represent a project analysis result.
Matrix	Contain list of matrix that our system support.

Table 2: Artifacts and Information Summary

2.1.3. Behavior

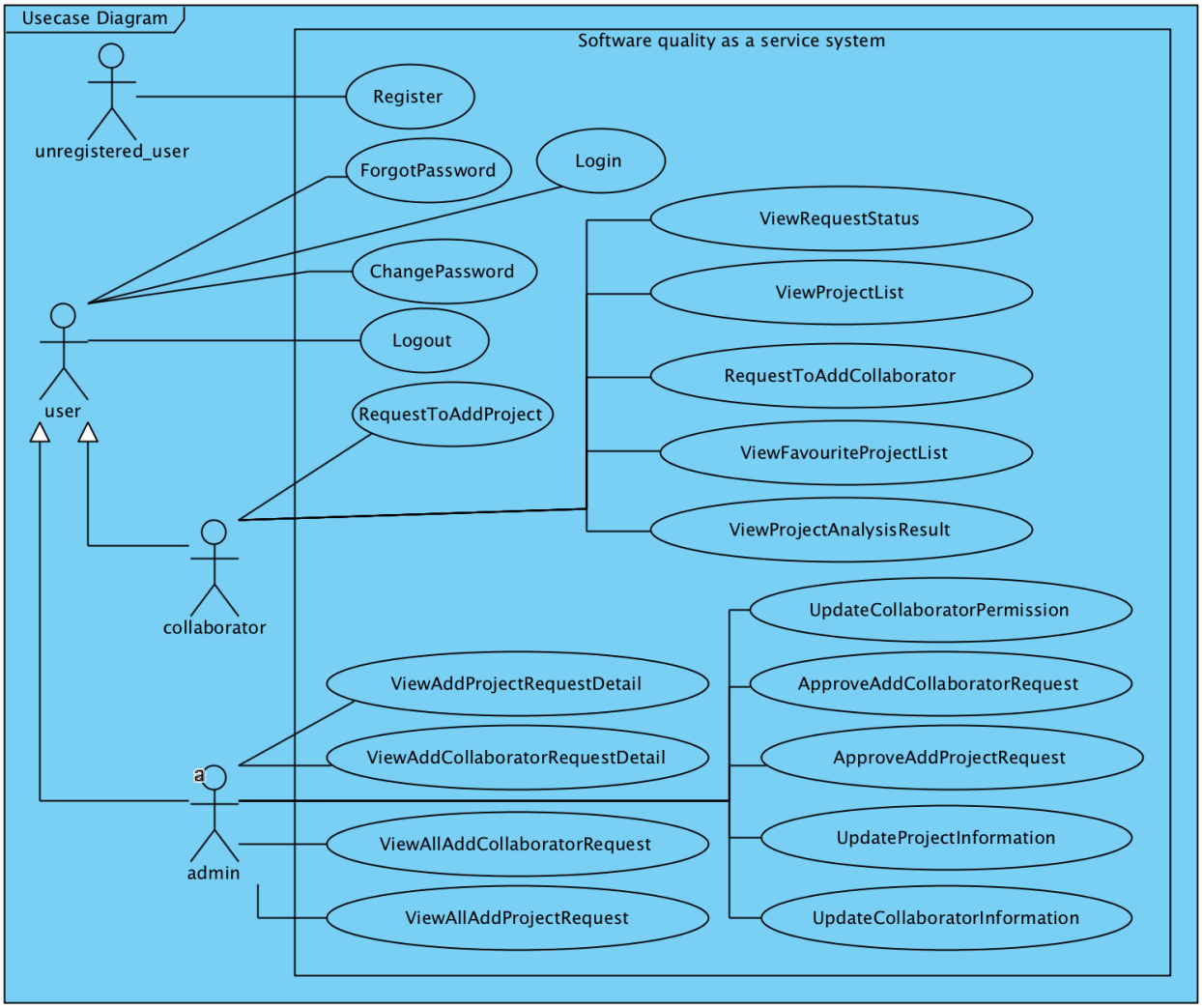

Figure 3:Process Diagram

Use case Id	Use Case
UC-1	Register
UC-2	Login
UC-3	Logout
UC-4	ForgotPassword
UC-5	ChangePassword
UC-6	RequestToAddCollaborator
UC-7	RequestToAddProject

UC-8	ViewRequestStatus
UC-9	ViewProjectList
UC-10	ViewProjectAnalysisResult
UC-11	ViewFavouriteList
UC-12	ViewAddProjectRequestDetail
UC-13	ViewAddCollaboratorRequestDetail
UC-14	ViewAllAddProjectRequest
UC-15	ViewAllAddCollaboratorRequest
UC-16	ApproveAddCollaboratorRequest
UC-17	ApproveAddProjectRequest
UC-18	UpdateProjectInformation
UC-19	UpdateCollaboratorInformation
UC-20	UpdateCollaboratorPermission

Table 3: Use case list

2.1.3.1. Capability : Register, login, logout and forgot password

2.1.3.1.1. Process: register

Identifier	UC-1: Register
Purpose	Create new account
Requirements	
Development Risks	None
Pre-conditions	None
Post-conditions	1. User profile will be created. 2. User can login to the system.

Table 4: Process Description: Register

Seq#	Actor's Action	System's Response
1	Unregistered user goes to the web app	

2		Show login page
3	Unregistered user clicks sign up button	
4		Redirect to sign up page and show sign up form.
5	Unregistered user fill form and submit	
6		System receive user information and create new user.
7		System alert “Success to create user”

Table 5: Typical Course of Action: Register - success

Seq#	Actor’s Action	System’s Response
1	Unregistered user goes to the web app	
2		Show login page
3	Unregistered user clicks sign up button	
4		Redirect to sign up page and show sign up form.
5	Unregistered user fill form and submit	
6		System receive user information and create new user but fail.
7		System alert fail to create user with a proper reason.

Table 6: Typical Course of Action: Register – fail

2.1.3.1.2. Process: login

Identifier	UC-2: Login
Purpose	Identify

Requirements	
Development Risks	User should only access the system with HTTPS protocol.
Pre-conditions	User should be registered into the system.
Post-conditions	User can access system function that he has permission

Table 7: Process Description : Login

Seq#	Actor's Action	System's Response
1	Collaborator opens the web app.	
2		Display login page
3	Collaborator fills login form with valid collaborator username and password.	
4	Collaborator clicks submit button.	
5		System verifies username and password
6		System redirects to dashboard page.

Table 8: Typical Course of Action: Collaborator Login success

Seq#	Actor's Action	System's Response
1	Admin opens the web app.	
2		Display login page
3	Admin fills login form with valid admin username and password.	
4	Admin clicks submit button.	
5		System verifies username and password
6		System redirects to admin control panel.

Table 9: Typical Course of Action: Admin Login success

Seq#	Actor's Action	System's Response
1	User opens to the web app	
2		Display login page
3	User fills login form with invalid username or password and submit	
4		System verifies username and password.
5		System alerts "Invalid username or password."

Table 10: Typical Course of Action: User Login fail

2.1.3.1.3. Process: logout

Identifier	UC-3: Logout
Purpose	User exits the system; user's profile will be secured (not accessed by anyone).
Requirements	
Development Risks	None
Pre-conditions	User must be logged in to the system.
Post-conditions	User will not have further access to the system.

Table 11:Process Description: Logout

Seq#	Actor's Action	System's Response
1	User click logout button	
2		System clears user session

Table 12:Typical Course of Action: User Login fail

2.1.3.1.4. Process: forgot password

Identifier	UC-4: ForgotPassword

Purpose	Request for new password
Requirements	**WC_4425**,
Development Risks	Not everyone should be able to initiate the link to reset the password, means that the link to retrieve a new password should be confidential, has a unique id or a UUID, and shall expire after first time opening and also after a short amount of time.
Pre-conditions	User has an account.
Post-conditions	User receives email with resetting password link.

Table 13: Process Description: forgot password

Seq#	Actor's Action	System's Response
1	User goes to the web app	
2		Show login page
3	User click forgot password button	
4		Redirect to forgot password page
5	User fill form and submit	
6		System sends email with resetting password link to user's email.

Table 14:Typical Course of Action: forgot password

2.1.3.1.5. Process: change password

Identifier	UC-5: ChangePassword
Purpose	User change password
Requirements	NA
Development Risks	None
Pre-conditions	User has logined to the system
Post-conditions	User password is changed.

Table 15: Process Description: change password

Seq#	Actor's Action	System's Response
1	User open change password page	
2		Redirect to change password page
3	User fill form and submit	
4		System update user password.
		System alert "Success change password"

Table 16: Typical Course of Action: change password - success

Seq#	Actor's Action	System's Response
1	User open change password page	
2		Redirect to change password page
3	User fill form and submit	
4		System update user password but fail
5		System alerts "Invalid password"

Table 17: Typical Course of Action: change password - fail

2.1.3.2. Capability : Create request and view request status

2.1.3.2.1. Process: Request to add collaborator

Identifier	UC-6 : RequestToAddCollaborator
Purpose	Request to add new collaborator to private project that he has a permission.
Requirements	**WC_4424**
Development Risks	None
Pre-conditions	Collaborator login to the system
Post-conditions	System creates add collaborator request and send notify email to admin and collaborator

Table 18: Process Description: request to add collaborator

Seq#	Actor's Action	System's Response
1	Collaborator clicks add collaborator button	
2		Show add collaborator page and form
3	Collaborator fills add collaborator form	
4		System creates request and send notify email to admin and collaborator.
		System alerts "Success to create request."

Table 19: Typical Course of Action: request to add collaborator - success

Seq#	Actor's Action	System's Response
1	Collaborator clicks add collaborator button	
2		Show add collaborator page and form
3	Collaborator fills add collaborator form	
4		System creates request and found that the user already had permission to access the project.
		System alerts "This user already had permission."

Table 20: Typical Course of Action: request to add collaborator – fail: already had permission

Seq#	Actor's Action	System's Response
1	Collaborator clicks add collaborator button	
2		Show add collaborator page and form
3	Collaborator fills add collaborator form	

4		System creates request and found that this is a duplicate request.
		System alerts “fail to create request, duplicate request”

Table 21: Typical Course of Action: request to add collaborator – fail: duplicate request

2.1.3.2.2. Process: Request to add project

Identifier	UC-7: RequestToAddProject
Purpose	Request to add project
Requirements	**WC_4449**
Development Risks	None
Pre-conditions	Collaborator login to the system
Post-conditions	System creates add project request and send notify email to admin and collaborator

Table 22: Process Description : request to add project

Seq#	Actor’s Action	System’s Response
1	Collaborator clicks add project button	
2		Show add project page and form
3	Collaborator fills add project form	
4		System creates request and send notify email to admin and collaborator.

Table 23: Typical Course of Action: request to add project - success

Seq#	Actor’s Action	System’s Response
1	Collaborator clicks add project button	
2		Show add project page and form
3	Collaborator fills add project form	

4		System creates request and found that this project already added to system.
5		System alerts “fail to create request, this project already added to the system”

Table 24: Typical Course of Action: request to add project – fail: already had project in system

Seq#	Actor’s Action	System’s Response
1	Collaborator clicks add project button	
2		Show add project page and form
3	Collaborator fills add project form	
4		System creates request and found that this is duplicate request
5		System alerts fail to create request, duplicate request

Table 25: Typical Course of Action: request to add project – fail: duplicate request

2.1.3.2.3. Process: View request status

Identifier	UC-8: ViewRequestStatus
Purpose	Collaborator can view the status of request that he creates.
Requirements	**WC_4508, WC_4609**
Development Risks	None
Pre-conditions	Collaborator login to the system
Post-conditions	None

Table 26:Process Description : view request status

Seq#	Actor’s Action	System’s Response
1	Collaborator clicks view request status button	

2		Display a request status summary page

Table 27: Typical Course of Action: view request status

2.1.3.3. Capability : access project

2.1.3.3.1. Process: View project list

Identifier	UC-9: ViewProjectList
Purpose	Collaborator can view list of the project that he has permission.
Requirements	**WC_4509, WC_4338**
Development Risks	None
Pre-conditions	Collaborator login to the system
Post-conditions	None

Table 28: Process Description : view project list

Seq#	Actor's Action	System's Response
1	Collaborator opens dashboard page	
2		Display a project list on dashboard page.

Table 29: Typical Course of Action: view project list

2.1.3.3.2. Process: View project analysis result

Identifier	UC-10: ViewProjectAnalysisResult
Purpose	Collaborator can view project analysis result that he has a permission.
Requirements	**WC_4453**
Development Risks	None
Pre-conditions	Collaborator login to the system
Post-conditions	None

Table 30: Process Description : View project analysis result

Seq#	Actor's Action	System's Response
1	Collaborator opens dashboard page	
2		Display a project list table on dashboard page.
3	Collaborator clicks row in project list table.	
4		System redirect to statistical page and display statistical data.

Table 31: Typical Course of Action: view project analysis result

2.1.3.3.3. Process: View favorite project list

Identifier	UC-11: View Favorite Project List
Purpose	User can view the project that he marks as favorite project in favorite project list.
Requirements	NA
Development Risks	None
Pre-conditions	Collaborator login to the system
Post-conditions	None

Table 32: Process Description : view favorite project list

Seq#	Actor's Action	System's Response
1	Collaborator opens dashboard page	
2		Display a **favorite** project list table on dashboard page.

Table 33: Typical Course of Action: view favorite project list – has favorite project

Seq#	Actor's Action	System's Response
1	Collaborator opens dashboard page	

2		Display an empty **favorite** project list table on dashboard page.

Table 34: Typical Course of Action: view favorite project list – doesn't has favorite project

2.1.3.4. Capability : view and approve request

2.1.3.4.1. Process: view add project request detail

Identifier	UC-12: ViewAddProjectRequestDetail
Purpose	Admin can view project request detail
Requirements	NA
Development Risks	None
Pre-conditions	Admin login to the system
Post-conditions	None

Table 35: Process Description : view add project request detail

Seq#	Actor's Action	System's Response
1	Admin opens admin panel.	
2		Display admin panel page
3	Admin clicks "request" button on the request menu.	
4		Display request page
5	Admin clicks "add project" tab	
6		Display add project request table which contain request detail.

Table 36: Typical Course of Action: view add project request detail

2.1.3.4.2. Process: view add collaborator request detail

Identifier	UC-13: view add collaborator request detail
Purpose	
Requirements	**WC_4452**

Development Risks	None
Pre-conditions	Admin login to the system
Post-conditions	None

Table 37: Process Description : view add collaborator request detail

Seq#	Actor's Action	System's Response
1	Admin opens admin panel.	
2		Display admin panel page
3	Admin clicks "request" button on the request menu.	
4		Display request page
5	Admin clicks "add collaborator" tab	
6		Display add collaborator request table which contain request detail.

Table 38: Typical Course of Action: view add collaborator request detail

2.1.3.4.3. Process: view all add project request

Identifier	UC-14: view all add project request
Purpose	View all add project request that a collaborator created
Requirements	NA
Development Risks	None
Pre-conditions	Admin login to the system
Post-conditions	None

Table 39: Process Description : view all add project request

Seq#	Actor's Action	System's Response
1	Admin opens admin panel.	

2		Display admin panel page
3	Admin clicks “request” button on the request menu.	
4		Display request page
5	Admin clicks “add project” tab	
6		Display add project request table.

Table 40: Typical Course of Action: view all add project request

2.1.3.4.4. Process: view all add collaborator request

Identifier	UC-15: ViewAllAddCollaboratorRequest
Purpose	View all add collaborator request that a collaborator created
Requirements	NA
Development Risks	None
Pre-conditions	Admin login to the system
Post-conditions	None

Table 41: Process Description : view all add collaborator request

Seq#	Actor’s Action	System’s Response
1	Admin opens admin panel.	
2		Display admin panel page
3	Admin clicks “request” button on the request menu.	
4		Display request page
5	Admin clicks “add collaborator” tab	

6		Display add collaborator request table.

Table 42: Typical Course of Action: view all add collaborator request

2.1.3.4.5. Process: approve add collaborator request

Identifier	UC-16: approve add collaborator request
Purpose	approve add collaborator request
Requirements	**WC_4452**
Development Risks	None
Pre-conditions	Admin login to the system
Post-conditions	System adds permission to the collaborator

Table 43: Process Description : approve add collaborator request

Seq#	Actor's Action	System's Response
1	Admin opens admin panel.	
2		Display admin panel page
3	Admin clicks "request" button on the request menu.	
4		Display request page
5	Admin clicks "add collaborator" tab	
6		Display add collaborator request table.
7	Admin clicks "approve" button that display in collaborator request row.	
8		System adds permission to the collaborator

Table 44: Typical Course of Action: approve add collaborator request

2.1.3.4.6. Process: approve add project request

Identifier	UC-17: approve add project request
Purpose	User can view the project that he marks as favorite project in favorite project list.
Requirements	NA
Development Risks	None
Pre-conditions	Admin login to the system
Post-conditions	System changes project status to active

Table 45: Process Description : approve add project request

Seq#	Actor's Action	System's Response
1	Admin opens admin panel.	
2		Display admin panel page
3	Admin clicks "request" button on the request menu.	
4		Display request page
5	Admin clicks "add project" tab	
6		Display add project request table.
7	Admin clicks "approve" button that display in project request row.	
8		System changes project status to active

Table 46: Typical Course of Action: approve add project request

2.1.3.5. Capability : update project information, collaborator information and collaborator permission.

2.1.3.5.1. Process: update project information

Identifier	UC-18: update project information
Purpose	update project information such as project name, project URL.
Requirements	NA
Development Risks	None
Pre-conditions	Admin login to the system
Post-conditions	System updates the project detail.

Table 47: Process Description : update project information

Seq#	Actor's Action	System's Response
1	Admin opens admin panel.	
2		Display admin panel page
3	Admin clicks "project" button on the request menu.	
4		Display project page that contains a project list table.
5	Admin clicks project that display in table row.	
6		System display project detail form.
7	Admin change data in project detail form and submit form.	
8		System updates the project detail.

Table 48: Typical Course of Action: update project information

2.1.3.5.2. Process: update collaborator information

Identifier	UC-19: update collaborator information
Purpose	User can view the project that he marks as favorite project in favorite project list.
Requirements	**WC_4503**
Development Risks	update collaborator information such as collaborator email, collaborator name.
Pre-conditions	Admin login to the system
Post-conditions	System updates the collaborator information.

Table 49: Process Description : update collaborator information

Seq#	Actor's Action	System's Response
1	Admin opens admin panel.	
2		Display admin panel page
3	Admin clicks "user" button on the request menu.	
4		Display user page that contains a user list table.
5	Admin clicks user that display in table row.	
6		System display user detail form.
7	Admin change data in user detail form and submit form.	
8		System updates the user detail.

Table 50: Typical Course of Action: update collaborator information

2.1.3.5.3. Process: update collaborator permission

Identifier	UC-20: update collaborator permission
Purpose	grant and revoke user permission

Requirements	
Development Risks	None
Pre-conditions	Admin login to the system
Post-conditions	System updates user permission

Table 51: Process Description : update collaborator permission

Seq#	Actor's Action	System's Response
1	Admin opens admin panel.	
2		Display admin panel page
3	Admin clicks "access" button on the request menu.	
4		Display access page that contains a access list table.
5	Admin clicks row that display in table.	
6		System display access detail form.
7	Admin change data in access detail form and submit form.	
8		System updates the user permission.

Table 52: Typical Course of Action: update collaborator permission

2.1.4. Modes of Operation

Admin Mode: admin sees requests for analysis, managing user permissions, accept or deny adding contributor to the projects.

User Mode: does not see admin page, user sees registering, login, user profile, sending a request for project analysis, see their favorite projects, see their own projects that are contributor in, see their projects software quality.

2.2. System Analysis Rationale

Difference between adding a project and adding a request for analysis.

- Well everyone would think that first we add the project to the system, means we first send a request for that, and next, send another request for analysis the project, but they will get done at the same step. The moment that contributor sends a project for analysis, the project will be added to the system and then it would be sent for analysis.
- There is no concept of project owner, everyone (every contributor) can send project for analysis, everyone (contributor) can send request for adding another contributor to the project.
- We don't have anything to do with fetching the code from GitHub repository of the project for analyzing. Pooyan (Client) has tools that will get the code from the repository for doing the software quality analysis.

3. Technology-Independent Model

3.1. Design Overview

3.1.1. System Structure

System structure will be described in section 4.1.1 System Structure

3.1.2. Design Classes

Design Classes will be described in section 4.2.1 Design Classes

3.1.3. Process Realization

Process Realization will be described in section 4.3.1 Process Realization.

3.2. Design Rationale

We have figured out the front end and back end to stay on the MVC pattern and also have the PostgreSQL ORDBMS to have connection with the server, all wrapped up by the Docker containers, so that there will be no further need to do building and compiling every time migrating to other platform, just with having Docker installed, we will be able to run the application on that platform. Also, we have the NGINX proxy settings all set, so that with just having the certifications we will be able to have the https connection. Also for the design of the software itself, the user interface and back end are separated, well packaged and the difference between the class of backend and frontend is because of the design of the UI with ReactJS. For the ReactJs all of the forms have on-the-fly validation and also all of the interactions for validation on the backend are done JSON object with AJAX request.

4. Technology-Specific System Design

4.1. Design Overview

4.1.1. System Structure

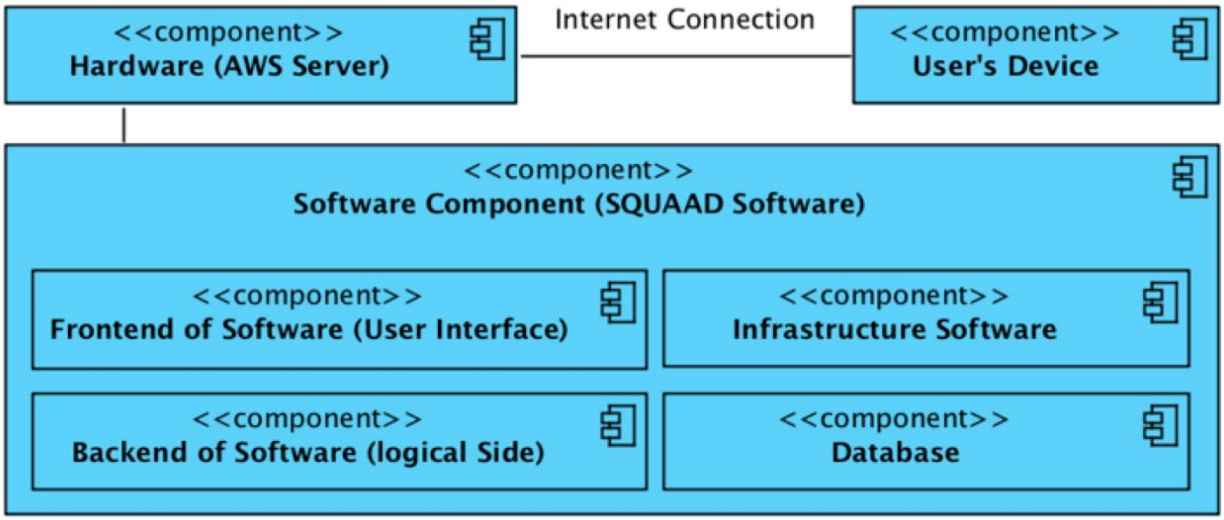

Figure 4: Conceptual Domain Model

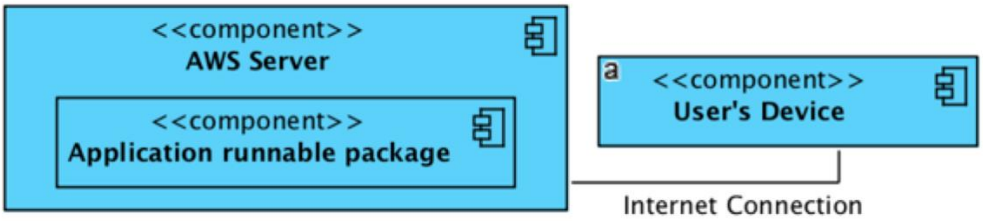

Figure 5: Hardware Component Diagram

For the Hardware Component Class Diagram, it is worthy to mention that our design for technology independent model will be like figure 4. And disregarding the technology, the Hardware component class diagram will be the conceptual domain model, and mostly their wrappers, which is Hardware (Application Server) and Personal Computer, Laptop (User's Device) Interacting with each other through internet connect.

Hardware Component	Description
User's device	This is the user's device which they use to connect to the software system (The Web Application). It is meant to be a computer with Chrome Web Browser)

Application Web Server	Server used to host the web application
Deployed Software (Physical Software)	Physical Resource allocated to the software by the Docker container, Docker runs it on the Linux kernel.

Table 53: Hardware component

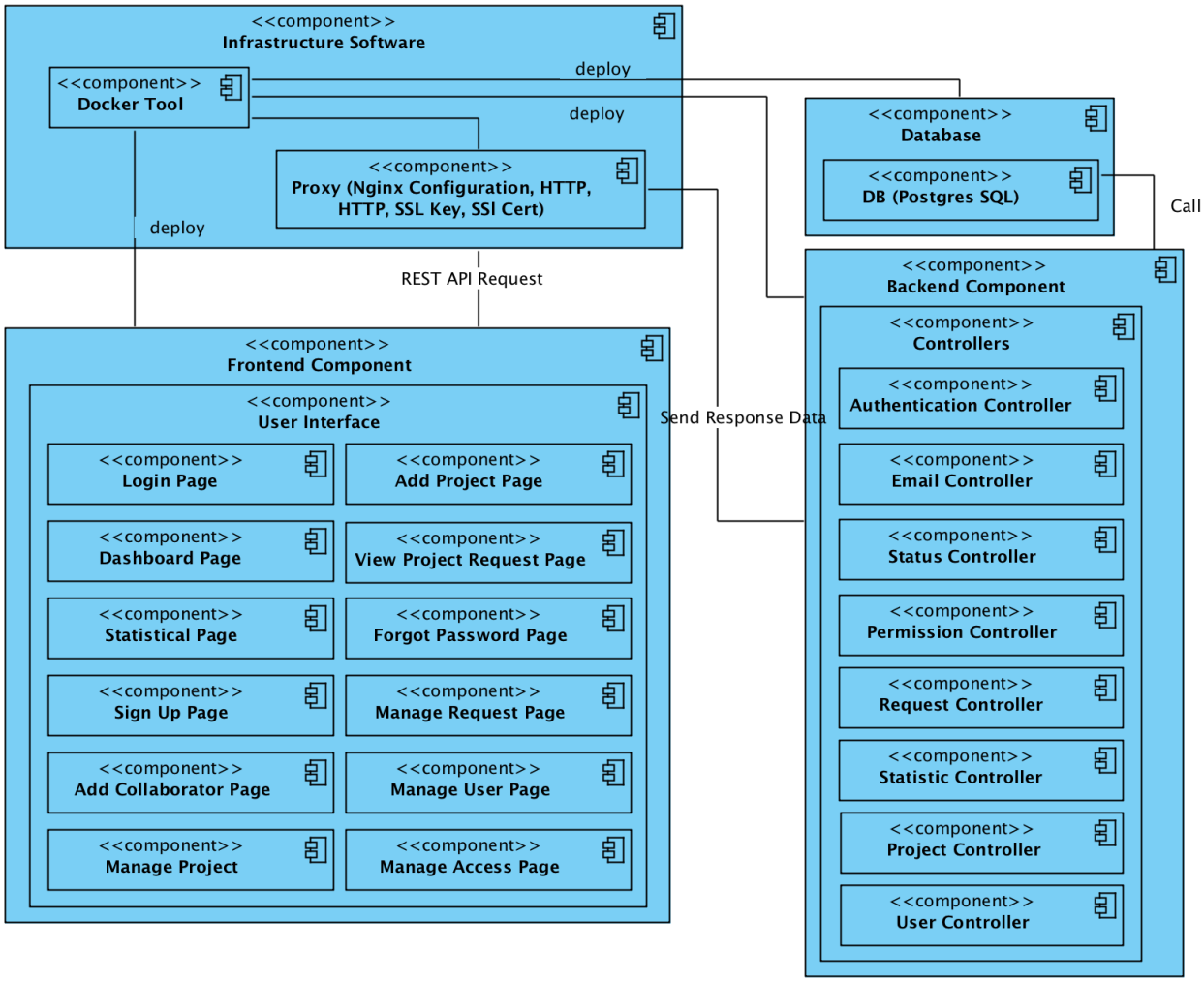

Figure 6: Software Component Diagram

The deployment diagram in here is the same as the deployment diagram in part 4, technology specific system design, with this only difference that in there is no speak of the technology, so that the deployment diagram would look like Figure 5 in here. There should be an Application Web Server, which is the web host and has a domain, IP, etc. And the application will sit on the Application Web Server.

Software Component	Description
Infrastructure Software	This component consists the configuration of the web server

	which will handle the HTTP/HTTPS requests and hand over them to the software backend.
Database	SQUAAD using Postgres SQL.
Frontend Component	Develop using ReactJS, Bootstrap, Axios
Backend Component	Provide REST API
Docker Tool	Docker tool is for easier create, deploy, and run the application. It's a semi virtual machine with this difference that it is not an actual virtual machine, i.e. it does not install like the way virtual machine does. It allows the developer to package up an application with all of libraries and other dependencies and ship it all as one package. Docker has a so much better performance than the way virtual machines get installed. https://opensource.com/resources/what-docker
Nginx	Proxy Server
Postgres SQL	This Component is the ORDBMS server which will have connection and communication with the software backend.
Login Page	Login page contains login form, form validator. This component will call backend API to login to the system.
Dashboard Page	Call backend API to get project list, favorite project. And, display those data.
Statistical Page	This part handles the frontend of the visualization of the charts, diagrams, tables showing the quality of the software based on various metrics.
Sign Up Page	Display sign up form, validate. This page will call backend API to create user.
Add Collaborator Page	The frontend functionality checking whether the email is valid, that email exists in the DB table of the all of the emails, whether that email is already in part of that project or not and if it is already not adding it anymore, and whether the request is already in the system.
Add Project Page	Display add project form. This page calls GitHub API to get project path. And, calls backend server to create 'add project' request.
Add Collaborator Page	Display add collaborator form. It calls backend API to create 'add collaborator' request.
View Project Request Page	This part is the backend and logic side for adding a new collaborator to the project.

Forgot Password Page	This part is to fetch the information about a project, do the needed computation based on given formulas, and send the needed results to the front end, to depict the diagrams in the frontend or user interface.
Manage Request Page	Admin page for managing request. It calls backend API to update ‘add project’ and ‘add collaborator’ request’s status.
Manage Project	Admin page for managing project detail such as project URL, isPublic, etc.
Manage User Page	Admin page for managing user detail such as user’s status, etc.
Manage Access Page	Admin page for managing permission to access project.
Authentication Controller	Contains Backend API working on authentication.
Email Controller	Contains Backend API working on sending email.
Status Controller	Contains Backend API providing server status and login status.
Permission Controller	Contains Backend API working on user’s permission.
Request Controller	Contains Backend API working on managing request.
Statistic Controller	Contains Backend API providing statistical data.
Project Controller	Contains Backend API working on managing project.
User Controller	Contains Backend API working on managing user.

Table 54: Software component

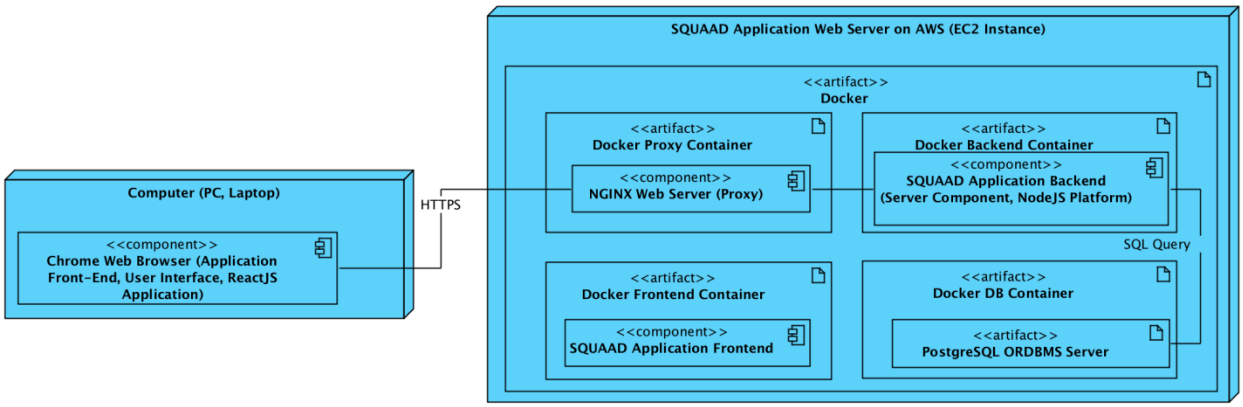

Figure 7: Deployment Diagram

4.1.2. Design Classes

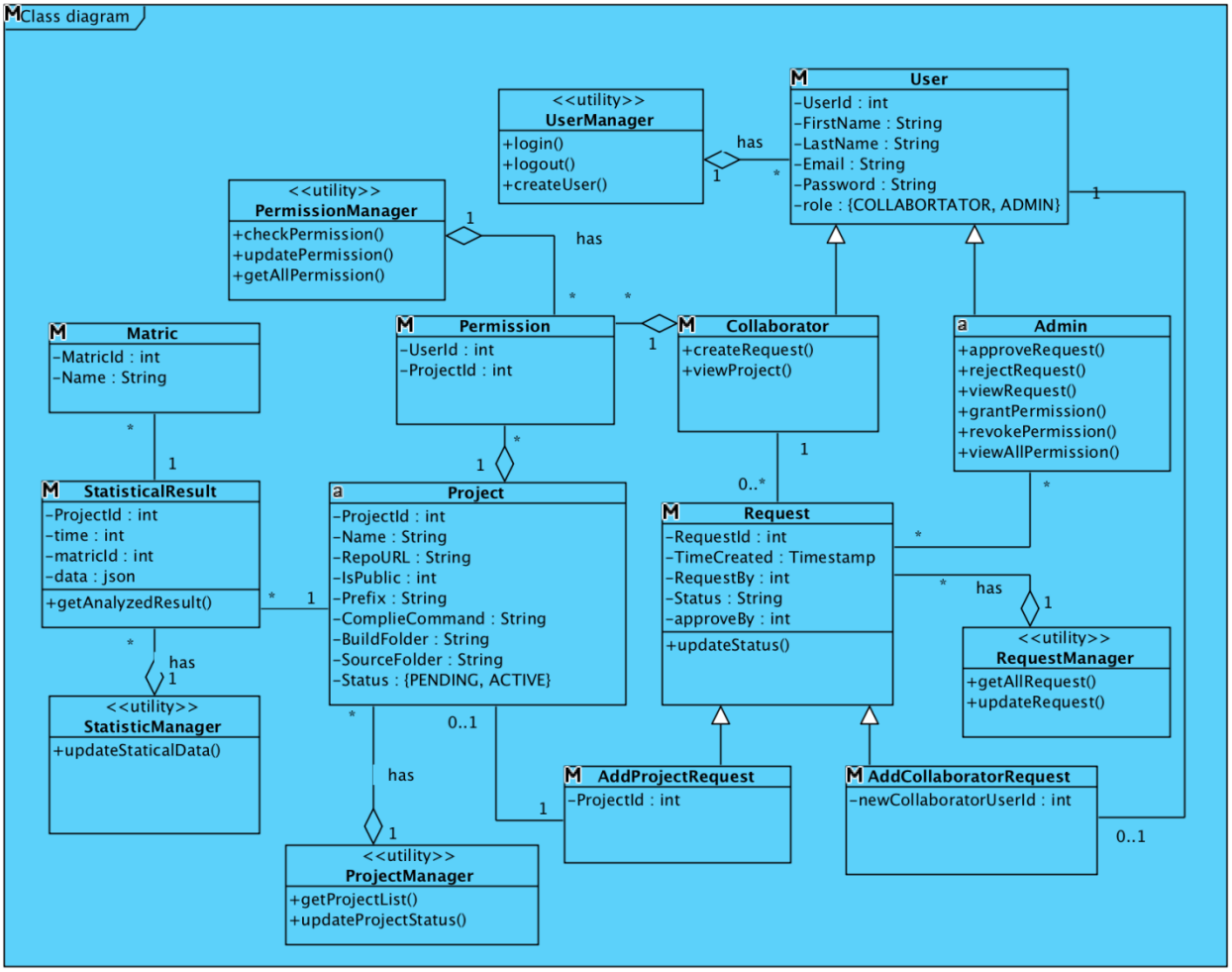

Table 55: Supporting Software Component Class Diagram

Class	Type	Description
User	Entity	Describe and store user information
Collaborator	Entity	Describe a user that can access project data
Admin	Entity	Describe a user who has ability to manage project and user permission.
Request	Entity	Describe and store request information
AddProjectRequest	Entity	Describe and store add project request information. This class has relation with project and user.

AddCollaboratorRequest	Entity	Describe and store add collaborator request information. This class has a relation with user.
Project	Entity	Describe and store project information
Permission	Entity	Describe and store list of user that has permission to access project and which project he can access.
Matric	Entity	Describe and store matric information
StatisticalResult	Entity	Describe and store statistical data
ProjectManager	Entity	Use for project management.
StatisticManager	Utility	Use for statistical data management such as insert or update statistical data
PermissionManager	Utility	Use for managing user permission
UserManager	Utility	Use for managing user of the system
RequestManager	Utility	Use for managing request that user submit

Table 56: Component in class diagram

4.1.3. Process Realization

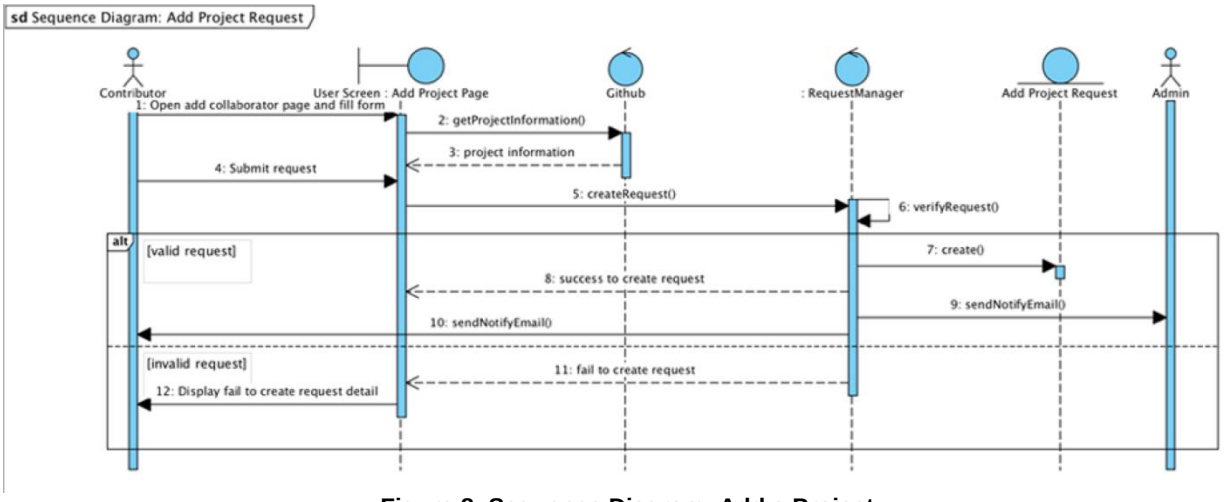

Figure 8: Sequence Diagram: Add a Project

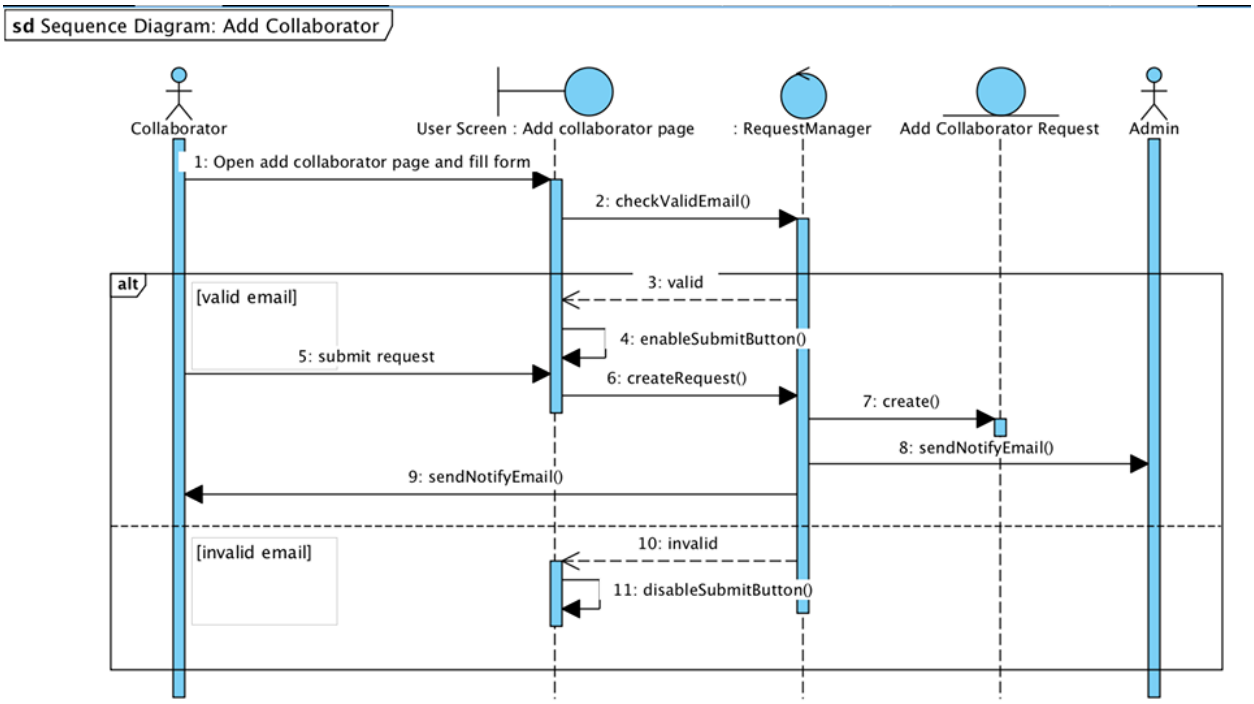

Figure 9: Sequence Diagram: Add a Collaborator

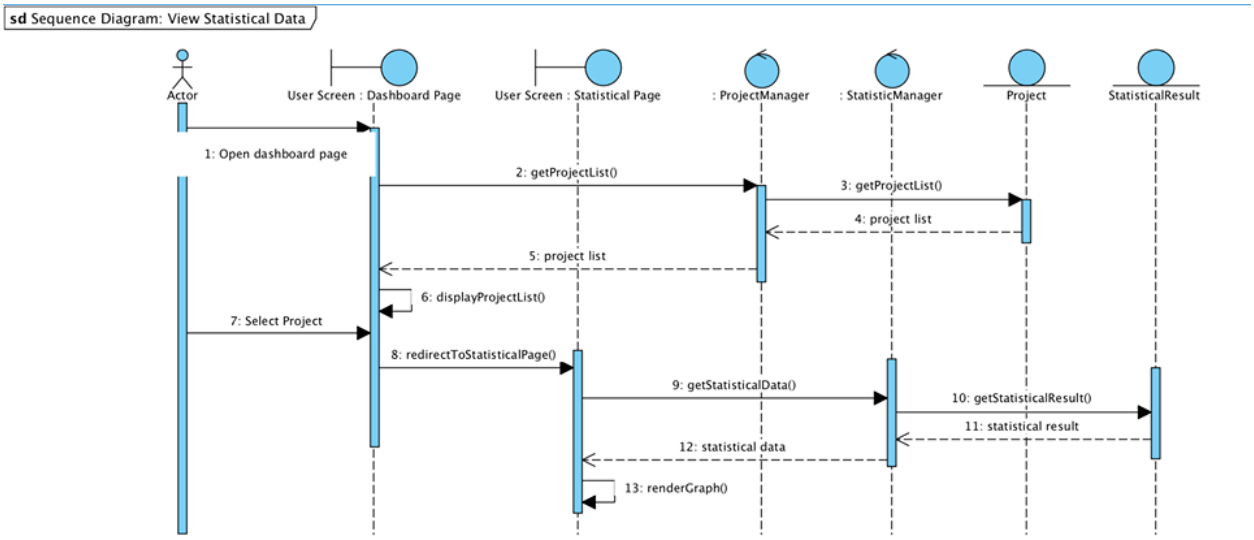

Figure 10: Sequence Diagram: View Statistical Data

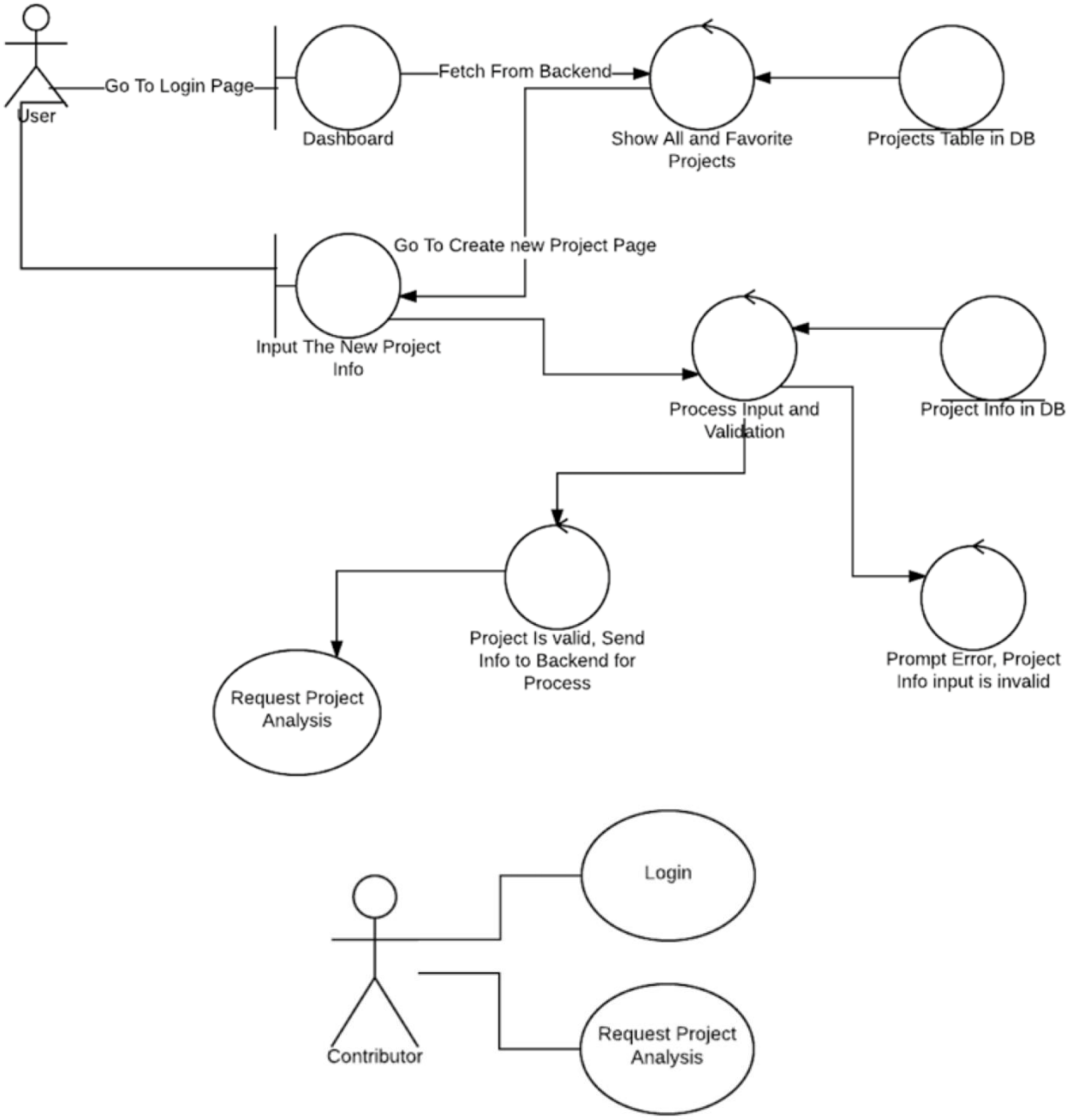

Figure 11: Robustness Diagram, Login and Request Project Analysis Use Cases

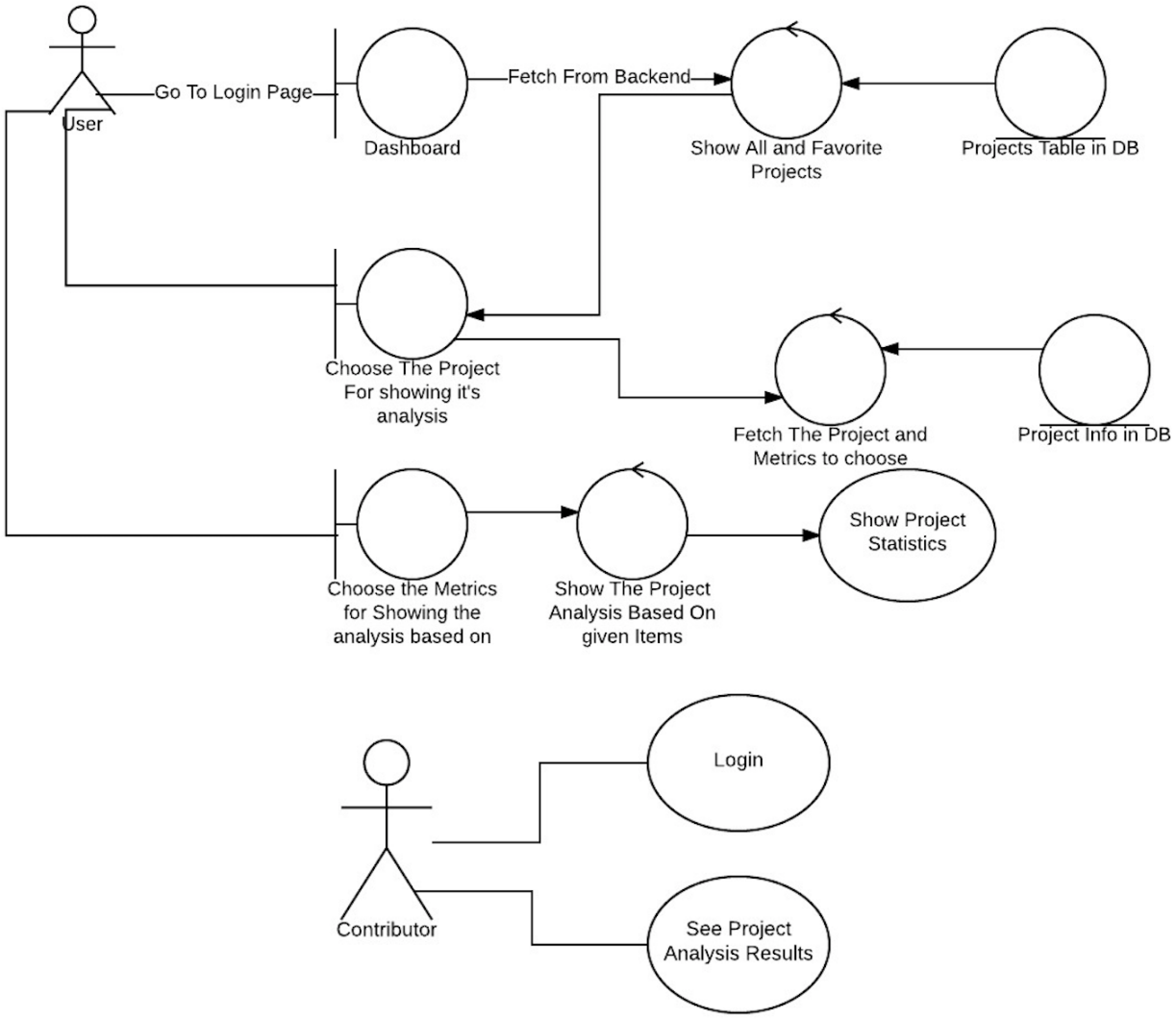

Figure 12: Robustness Diagram, Login and Project Statistics Use Cases

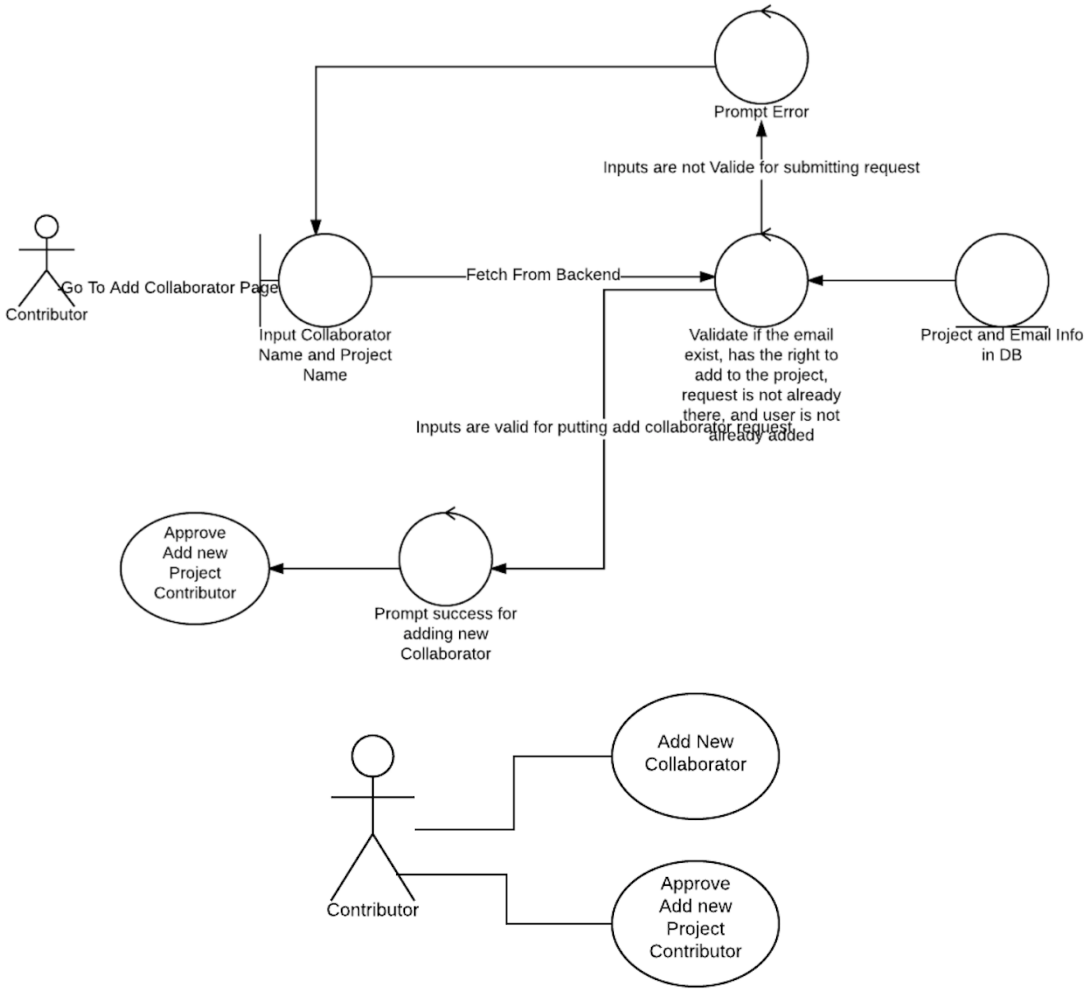

Figure 13: Robustness Diagram, Add New Collaborator and Approve Add new Project Contributor Use Cases

4.2. Design Rationale

There is still more API needed to talk about, and the major one is the API to fetch and receive the analyzed project from the other Software for analyzing software progress based on various metrics.

The purpose of the above deign were to make it better understand what is used in the overall design and architecture of the system, what more components should be considered, in what level of abstraction the components are working and how they have communication and interaction with each other.

5. Architectural Styles, Patterns and Frameworks

Name	Description	Benefits, Costs, and Limitations
Three-Tier	Having Major 3 parts of User Interface Tier(ReactJS), Logic Tier (Server, or Backend with Node Js, and ExpressJs), and Data Tier (Postgresql ORDBMS)	●Changes in any part is independent from the other parts, no impact on other Tiers. ●This architecture has no cost, for using explicitly this architecture, we do not pay for other 3rd party system (JavaScript library and frameworks are open-source, likewise for the PostgreSQL)
Client Server	Having this style in 2 parts: ●Sending Request from computer browser to the application server ●Docker Tool communicating with Containers	●Modifiability, maintenance, evolvability, are the most important benefits to our project. (Client Pooyan even can later use our backend for mobile or another platform development) ●Docker itself gives us this opportunity to automatically and easier do the build and deployment, without the concern of the destination platform structure. In fact, it makes developers and operators free of the build and compile challenges.

REST	Using React and Express Js Axios Library, the flow of data is always one-way (Redux Concept), also there is no state kept on transition between back-end and front-end, i.e. using Axios library, easy to send HTTP request to REST endpoints	●REST, the abbreviation of Representational State Transfer, having multiple principles to separation of the client and server side, visibility, reliability, scalability. Most important principles are the stateless, Uniformed Interface, and Layered System that we are using here. [https://restfulapi.net 2017/11/27] ●REST Architecture is independent of the type of platforms and languages. (Could work with HTML, XML, JSON; In this project, we were working with JSON)
Express JS	is a web application framework for Node.js, released as free and open-source software under the MIT License. It is designed for building web applications and APIs.	● provides a thin layer of fundamental web application features, without obscuring Node.js features. ● provides robust set of features for web and mobile applications.
ReactJS	React JS Software Pattern: A JS Library for building fast, simple, and scalable to create Web App	● Declarative Structure: Expressing logic of the program without describing the control workflow ● With this structure, react code is more predictable, easier to debug.

Table 57: Architectural Styles, Patterns, and Frameworks