University of Glasgow | School of Computing Science

Team Project Report

C6 – SCMS Worksite E-Form System

Team 22

Kenneth Ng

Ong Yuan Lin

Chan Yu Chyi

Johnathan Yong

Tang Kin Leung

31 March 2022

Abstract

The cost of construction site projects is rising due to the Covid-19 pandemic. One of the main reasons that lead to rising costs is that construction sites' administrative tasks are being done manually. In order to tackle the problem of rising costs in construction site projects, it is necessary to implement a smart construction site system such as the E-Form system. The E-Form system aims to digitalize paper forms that can reduce printing costs, storage costs and eliminate manual data entry. This report focuses on how the team tackled a real-life project with our industry partner over the course of 7 months. The report is broken down to how the team carefully approaches the project by choosing the appropriate architecture and design, development process, and implementation of the Worksite E-Form system and practices adopted to manage the project. The report will then end with key takeaways, reflection and challenges the team faced.

Education Use Consent

We hereby give our permission for this project to be shown to other University of Glasgow students and to be distributed in an electronic format.

Project Overview

1.1 Introduction

This document report aims to provide a detailed explanation of our team project and the system we have implemented. The team will be providing a detailed explanation of the project by applying what was learnt in professional software development and team projects. Additionally, images, graphs, and diagrams are included to help better understand our projects and learning.

Construction worksites have used paper forms in Singapore since the beginning of time. Paper forms have created unnecessary productive loss spent on completing repetitive administrative tasks that can be easily avoided by adopting electronic forms (E-forms). With technology constantly progressing, E-forms are now capable of facilitating real-time data transfer and instant access to data and tracking. Therefore, an E-form system will be implemented to replace paper forms to streamline the workflow in construction worksites and provide convenience to both workers and managers.

This report will document the team's learning journey in professional software development and the team's journey in implementing the Worksite E-Form System with our industry partner - CloudPlus. The report will be broken down into several sections, with each section containing detailed explanations. The sections consist of a project overview, software product, software process, reflective essay, and a conclusion.

The software products section will cover the ambitions and achievements done by the team, which includes a discussion on what was achieved in the project. Other than ambitions and achievements, the section also covers high-level design documentation, appropriate architectural and design patterns, explicit identifiers for modules, methods and variables during implementation, including code documentation, and software organization, where it covers the installation instructions and user documentation.

The software process section will cover change management, project planning, quality assurance, and process improvement. To break it down, it consists of evidence of good change management by measuring the frequency of commits with informative log messages that explains the rationale or the change in detail, evidence of good practices in the issue management system with milestone exists for each iteration of the project as well as frequently updating of the tickets, every commit to the master branch is tested in a continuous integration environment by automated regression test suite as well as performed code reviews during a merge request, software process issues during the last retrospective are elicited and documented by the team with software process problems been thoroughly analyzed by the team using five whys, and the identified process improvement actions are well documented.

The reflection section covers the team's experiences working together for the past seven months, including what we learned and the challenges we faced and overcame as a team.

At the end of the report, the conclusion section covers the general and broader lessons we learned from the team's project.

1.2 Case Study Background

CloudPlus is a leading provider of Private Cloud Solutions and is a forward-thinking Singapore-based firm. In order to maximize productivity and efficiency for the everyday operating needs, they are dedicated to delivering cutting-edge technological solutions based on Artificial Intelligence of Things (AIoT) and Robotics. [1]

CloudPlus provide a starting point to deliver exceptionally integrated solutions to their partner and clients with some of the infrastructure that they own and operate which consists of Next-Generation Networks, Security and Cloud Services. CloudPlus not only make use of all the latest available technologies out there in the Information Technologies Industry to construct innovative, high-tech applications and software's that customizes to their client's needs, they also provide recommendations and improved solution to achieve an outcome that will blows the client's expectations with the experts in their company from different industry segments. [1]

It is known to all that CloudPlus was founded to help improve operations and processes with the help of artificial intelligence and Internet of Things (IOT) to create a better world. Furthermore, CloudPlus has worked with a lot of big clients, some of these big clients are Mass Rapid Transit Corporation, Land Transport Authority, ST Engineering and ST Logistic. [1]

CloudPlus offers a series of services which includes Real-Time Location System, Artificial Intelligence and Customized Solutions. Real-Time Location System uses an item called wireless Real-Time Location System tags that will be attached to a person or an object to identify and pinpoint the location. The objective of Artificial Intelligence is to complete specific end-goals and tasks through a flexible adaptation by accurately deciphering the external data that the system will understand and learn from with the help of those learnings. Lastly, Customized Solutions refers to custom-made software that is created to cater to the client's particular preferences and expectations for the company. [1]

All their services are backed up by CloudPlus' 100% Committed support and the majority are covered by a 100% Uptime Guarantee backed by refunds. [1]

1.3 Objectives

The construction sites in Singapore are currently heavily reliant on human methods, and everything is done manually. The cost of a construction site project will rise due to Covid 19. One of the essential considerations for construction organizations will be how to employ technology to ensure workplace safety and monitor the process.

The overall objective of CloudPlus is to implement a smart construction site system that consists of 9 projects for nine teams. The system will give the client a better picture of what is going on in the construction worksite processes. In addition, CloudPlus aims to develop an effective and intelligent integration solution with a combination of workflow management, workplace safety monitoring, worker management, worksite e-form system, worksite dashboard, and worker mobile application.

The system will allow numerous operations to be completed electronically and streamline the entire worksite management cycle. In addition, the user experience, portability, and safety are prioritized in the transformation and development of the overall nature of construction site management by placing the user at the center.

The team is assigned the worksite e-form system and has worked together to identify the objectives. The first objective is to design a user friendly and easy to use user interface so that users who are not tech-savvy will be able to navigate and access the website with ease, increasing the speed for workers to complete the task of filling up and submitting the form by 50%.

The second objective is to help project managers and workers to organize better the forms where the forms can be categorized and have a centralized location based on the project it is created, allowing workers to find the forms and project managers to assess the form quickly, overall increasing the processing speed and efficiency by 200%.

Software Product

2.1 Achievements and Ambitions

2.1.1 Achievements

The following are the agreed agreements based on project objectives given during the 4 customers meeting:

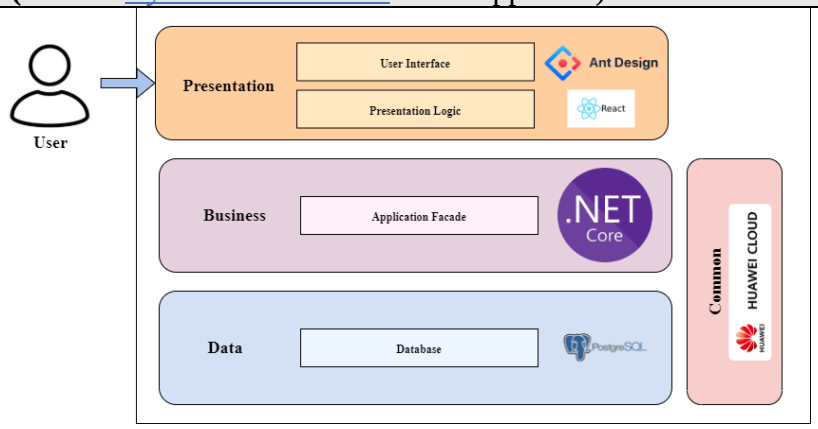
Project Manager	
Project	• Only project managers are allowed to create projects. • Manager can create multiple forms within a project.
Form	• Only project managers are allowed to create forms. • Forms must be customizable, with the flexibility to add any elements (e.g., radio button, text box etc.) and rearrangement of the questions. Similar feature as what is seen on Google forms. • Manager can save the form created as a template for future usage. • Forms must have expiry dates • Created forms can be edited. • Managers can approve or reject forms submitted by workers
Worker	
Project	• Workers can only view the project.
Form	• Workers can resubmit the form. • Workers are not allowed to edit the form after submission. • Workers can only view the status (approved, rejected or pending) of the form after submission.
Technical Stacks (Refer to *System Architecture* in the appendix)	
	
Frontend	React.js (Node.JS and UMI.JS), Ant Design Pro

Figure 1 System Architecture

Infrastructure	Huawei Cloud
Backend	.NET Core 5
Database	PostgreSQL

2.1.1 Ambitions

In addition to the above requirements, the team also proposed additional features to enhance the E-Form system. (Refer to *Archived Section* in appendix for clearer image)

2.1.1.1 Archived Section

Figure 2 Archived section in E-Form system

Forms and projects that **are not required for usage** will be **archived** in this section. Thus, **reducing cluttering** in the main section. Archived forms and projects can **no longer be restored** to their respective sections, but managers are still allowed to view them.

2.1.1.2 Search Function

Figure 3 Search feature in E-Form system

The search function enables users to **locate the required form faster**. Users can either search by form name, description, or expiry date. This applies to form management as well. The search function is also implemented for the projects.

2.1.1.3 Reason for Rejection

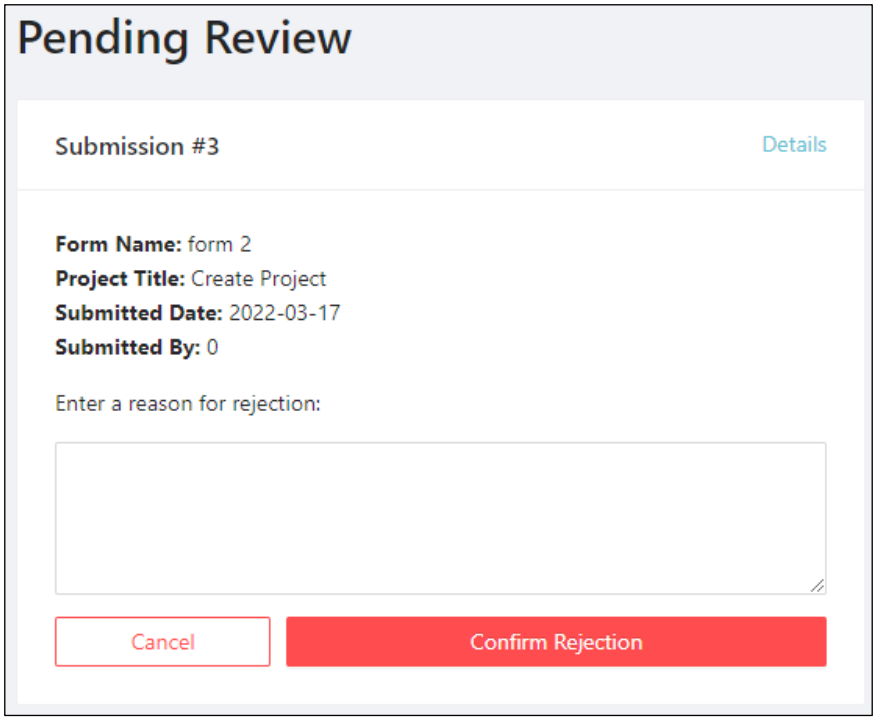


Figure 4 Reason for rejection in E-Form system

When rejecting form submission, the manager would have to write the reason why a particular form is being rejected. This way, the worker would have **better understood** why it is rejected and not make the same mistake.

2.2 Design

2.2.1 Software Architecture Pattern

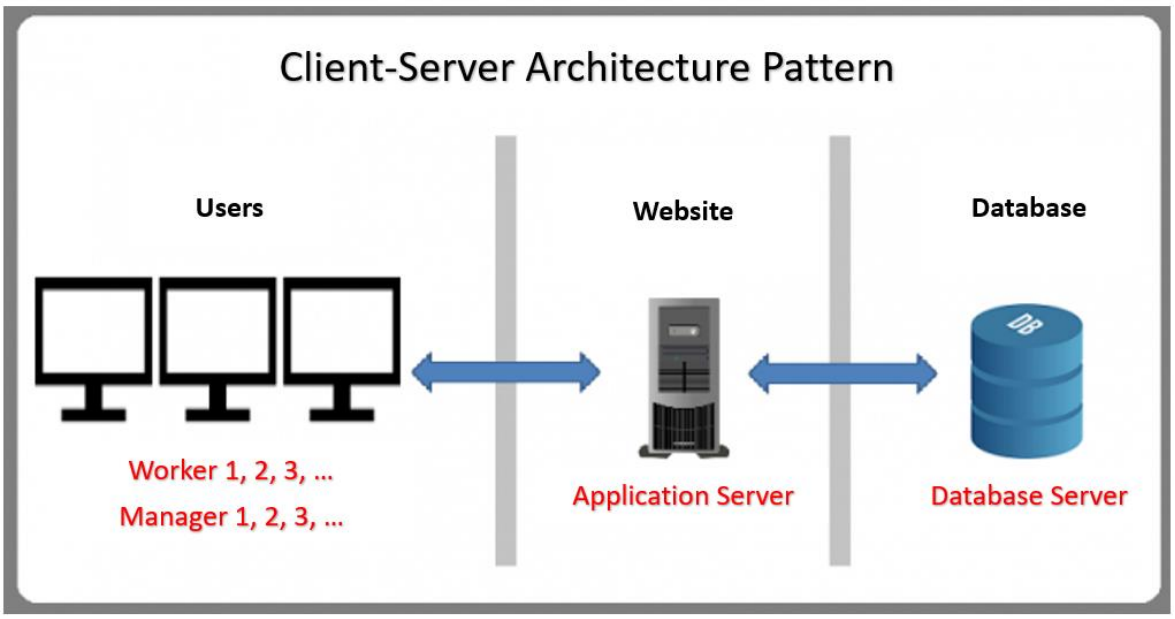


Figure 5 Client server architecture pattern for E-Form system

The team adopted a client-server architecture pattern for the E-Form system. The team chose this architecture pattern to allow data to be accessed from different locations by users from a shared database.

Each user will have access to the website on their respective devices, which will act as a client to communicate with the database server through the internet. The features of the E-Form systems act as a service that will require a connection to the database to insert, retrieve and update information. For example, the manager creates a new form to be inserted into the database, and the worker will then need to retrieve the form and fill up the form before updating the database.

2.2.1 Unified Modeling Language Diagrams

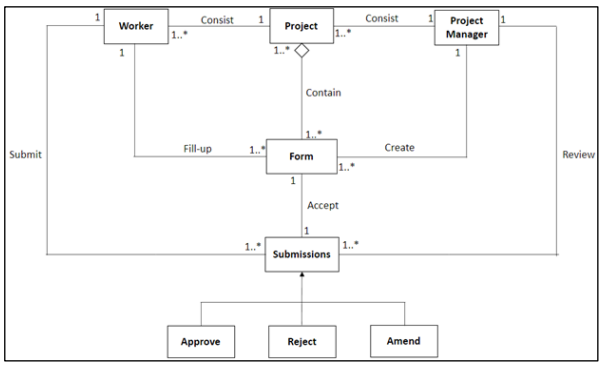


Figure 6 Class diagram of E-Form system

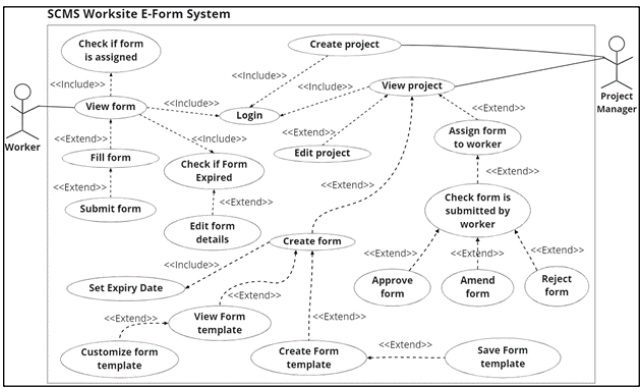


Figure 7 Use case diagram of E-Form system

(Refer to *Unified Modeling Language Diagram* in the appendix for clearer image.)

Based on the customer's requirement for the E-Form system and the team's continuous communication, the customer confirmed that the minimum viable product for the E-Form system only requires a form builder to create customized forms and allow workers to fill up the customized forms.

The team actively brainstormed for innovative ideas and proposes them to the customer, implementing additional features to the E-Form system. For example, additional features approved by the customer and implemented into the E-Form system include the Archived section, search functions, form status and reason for rejection. These additional features aim to improve the E-Form system by creating a user-friendly interface, enhancing its ease of use, and increasing its robustness with quality-of-life changes.

Figure 6 models the following structural properties in the SCMS Worksite E-Form System. It represents that one project consists of precisely one project manager and one or multiple workers. One or many projects may contain one or many forms. A manager may create one or many forms. One worker may fill up one or multiple forms. A manager may review one or many submissions, and one worker may submit one or many submissions. One submission can only accept one form. Submissions include approving, rejecting, and amendment.

Figure 7 models the use case of the SCMS Worksite E-Form System. It describes the high-level functions and the scope of our E-Form system. It also shows the interactions between the system, manager and worker. For example, both the worker and the project manager must log in to access the system. For workers to view their forms, they need to ensure that the form is assigned to them, and the forms are not expired. When the worker views a form, they will have the option to fill up the form and submit it to the managers. The worker will also be able to submit the form again if the form is not expired. Only the project manager can create the project. After the project manager views a project, they can choose to edit the project or create a form. When the project manager creates a form, they can choose to save the form template after creating a form template or customize a form template after viewing the form template. The expiry date must be set on a form when it is created. The manager can also review form submissions. If there are forms submitted by workers, the manager can approve, amend or reject the form. Rejecting the form will require the user to enter a reason before confirming the rejection. The team proposed this quality-of-life feature to ensure that the worker will not repeat the same mistake in their next form submission.

2.3 Implementation

2.3.1 Projects Creation and Customisation

(Refer to *Project Creation and Customization* in appendix for a clearer image)

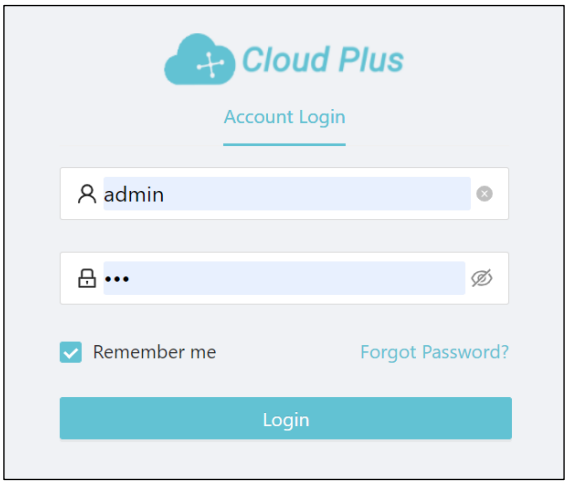


Figure 8 Login page UI

```
POST /api/login/account': async (req, res) => {
 const { password, username, type } = req.body;
 // await waitTime(2000);

 if (password === '123' && username === 'admin') {
   res.send({
     status: 'ok',
     type,
     currentAuthority: 'admin',
   });
   access = 'admin';
   return;
 }
```

Figure 9 Snippet of login access

Before the project integration with other teams, to access the E-Form system dashboard, the user must log in using their account details. During the project development, the team created a local dummy account with admin access for testing and was thus able to proceed to develop the other pages first.

After project integration, the username and password required to log in will be different, and the login credentials will be stored in a database.

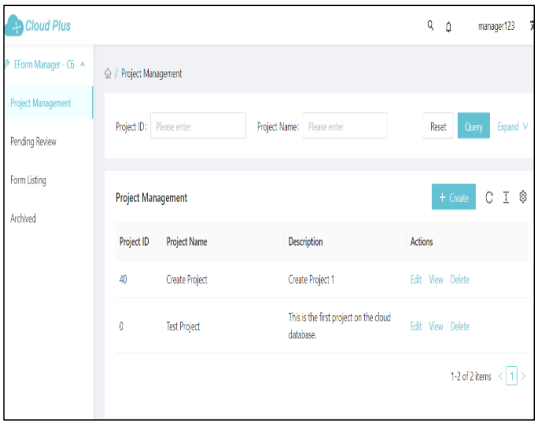


Figure 10 Project management UI

```
const submitCreateProject = () => {
  if (validateForm()) {
    const data = {
      owner: 0, // todo: update user based on logged in owner id
      title: inputProjectTitle,
      desc: inputProjectDescription,
    }

    const params = (new URLSearchParams(data)).toString();

    axios.post('http://localhost:5000/api/Project?' + params).then(res => {

      try {
        const redirect_id = res.data[0].project_id
        message.success("Project Created Successfully");
        history.push("/manager/projects/manage/" + redirect_id.toString())
      } catch (error) {
        throw new Error("Project Creation Failed")
      }

    }).catch(err => {
      message.error("Project Could Not Be Created");
    })
    console.log("data posted")
```

Figure 11 Snippet of create project code

If the user role is a manager, they will be redirected to the Project Management page. In project management, the manager can create new projects or make changes to existing projects.

Figure 12 Code snippet of project management's actions (edit and view)

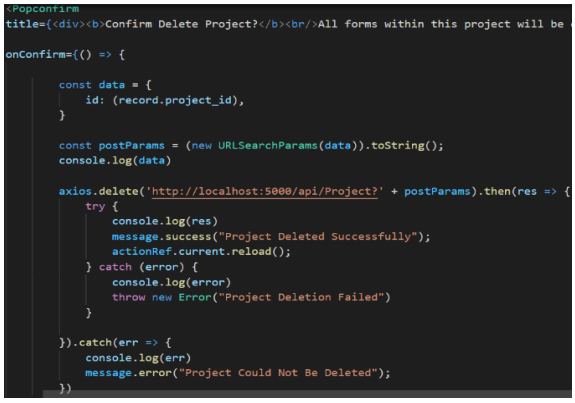


Figure 13 Code snippet of archiving project

On the project management page, the manager can create new projects or make changes to existing projects.

The list of actions that can be done to existing projects are:

1. Edit
    a. Edit information of project, such as title and description
2. View
    a. Display information of project in full view, including forms that are associated with the project
3. Delete
    a. Move specific project and its related forms into Archived section

2.3.2 Form Creation and Customization

(Refer to *Form Creation and Customization* in appendix for a clearer image)

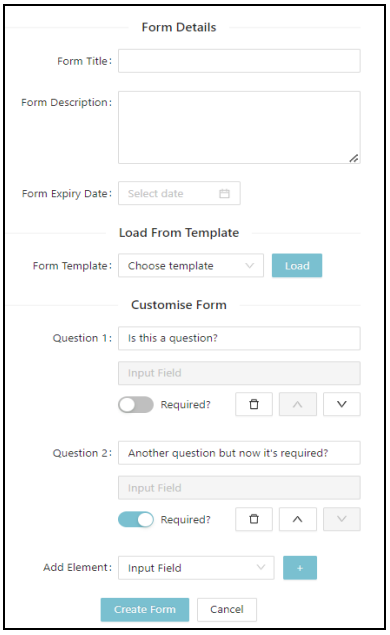


Figure 14 Initialize empty for UI

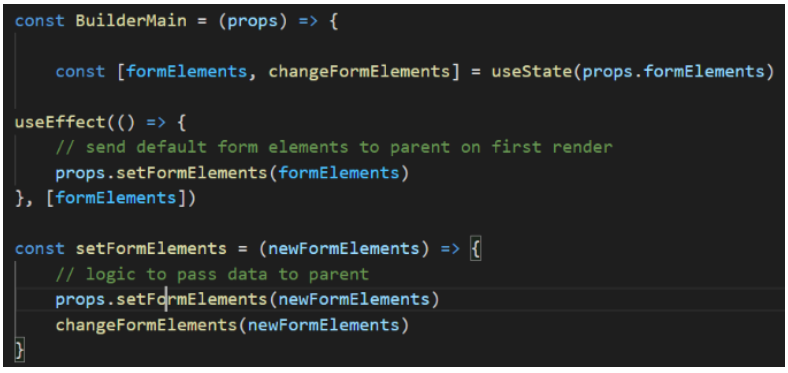


Figure 15 Code snippet of initializing empty form

Creation of forms is a feature only accessible to managers, it is handled by form builder. To create a form, managers must enter the form's title, description, and pick an expiry date for the form. Once a form has expired, workers will not be able to view the form in their 'Actions Required' page.

Figure 15 shows the code snippet of how the E-Form initialize an empty form for users to customize.

```
const addElement = (type) => {
    let newInput = {}
    let newPos = formElements[formElements.length-1].position + 1
    if (type == "input") {
        newInput = {
            position: newPos,
            key: newPos,
            type: "input",
            question: "",
            required: false
        }
        setFormElements([...formElements, newInput])
    } else if (type == "checkbox") {
        newInput = {
            position: newPos,
            key: newPos,
            type: "checkbox",
            question: "",
            checkboxOptions: ["Option 1", "Option 2"],
            required: false
        }
        setFormElements([...formElements, newInput])
    } else if (type == "radio") {
        newInput = {
```

Figure 16 Code snippet of adding elements to the form

Figure 16 shows how the E-Form system adds new elements to the empty form.

The list of elements available for users to customize the forms are:

1. Text input field
2. Checkbox
3. Radio button
4. Numeric input field
5. Text area

When users add new elements to the forms for customization, the elements are built with different attributes. In this snippet, we are using text input element as an example. It shows what are the attributes used to create the text input.

```
function BuilderInputElement(props) {
    const [inputText, setInputText] = useState(props.question)
    return (
    <>
        <Form.Item label={"Question " + (props.position+1)}>
            <Input
            placeholder="Enter Question"
            value = {inputText}
            onChange={(e) => {
                setInputText(e.target.value)
                props.onChangeQuestion(props.position, e.target.value)
                // probably would be more optimal to only save the text to object states when move/submi
            }}
            style={{
                marginBottom: 15
            }}
            />
            <Input disabled placeholder="Input Field" />
            <QuestionModifier
            type="Input Field"
            questionListSize={props.questionListSize}
            required={props.required}
            position={props.position}
            onChangeRequired={(e) => { props.onChangeRequired(props.position, e)}}
            onTriggerPositionChange={(e) => { props.onTriggerPositionChange(props.position, e)}} // +1
```

Figure 17 Code Snippet of element attributes

2.3.3 Approval and Rejection of Forms

(Refer to *Approval and Rejection of Forms* in appendix for a clearer image)

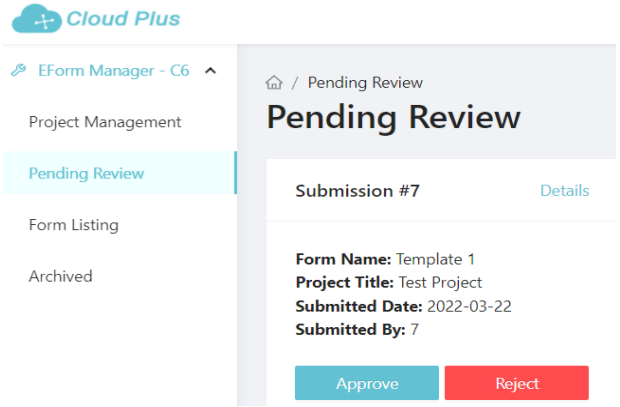


Figure 18Pending review UI

```
const approveSubmission = () => {
    setIsLoading(true)
    const data = {
        submission_id: submissionId,
    }

    const params = (new URLSearchParams(data)).toString();
    axios.post('http://localhost:5000/api/formSubmissions/manager/approve?' + params).then(res => {

        try {
            const redirect_id = res.data[0].project_id
            message.success("Submission Approved");
            setIsLoading(false)
            props.refreshData()
        } catch (error) {
            throw new Error("Approving Submission Failed")
            setIsLoading(false)
        }

    }).catch(err => {
        message.error("Submission Could Not Be Approved");
        setIsLoading(false)
    })
```

Figure 19 Code snippet of approving submission

When workers submit their forms, it will appear in Pending Review for the manager to approve or reject their submission.

Figure 20 shows how the team implement both approval of form submission and rejection of form submission.

For rejection of form submission, the manager is required to enter a reason why they reject the form submission. This will allow the worker to know where the issues lie in and thus prevent the worker from making the same mistake when they submit another form.

```
const rejectSubmission = () => {
    if (rejectReason == "") {
        message.warning("Please enter a reason for rejection")
    } else {
        setIsLoading(true)
        const data = {
            submission_id: submissionId,
            reject_reason: rejectReason,
        }

        const params = (new URLSearchParams(data)).toString();
        axios.post('http://localhost:5000/api/formSubmissions/manager/reject?' + params).then(res => {
            try {
                const redirect_id = res.data[0].project_id
                message.success("Submission Rejected");
                setIsRejecting(false)
                setIsLoading(false)
                setRejectReason("")
                props.refreshData()
            } catch (error) {
                setIsRejecting(false)
                setIsLoading(false)
                setRejectReason("")
                throw new Error("Rejecting Submission Failed")
            }
```

Figure 20 Code snippet of rejecting submissions

2.3.4 Worker Interface

(Refer to *Worker Interface* in appendix for a clearer image)

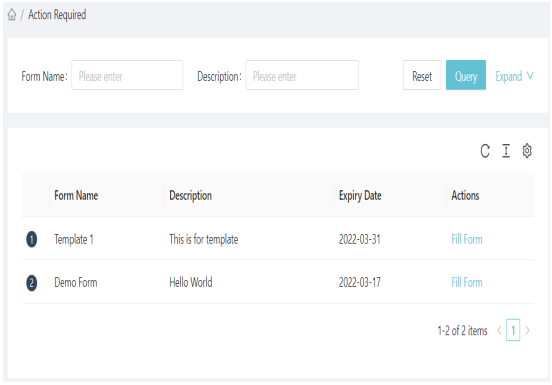


Figure 21 Actions required

```
{
    title: 'Actions',
    valueType: 'option',
    render: (text, record, _, action) => [
        <a key="editable" onClick={() => {
            history.push(`/worker/forms/fill/${record.form_id}`)
        }}>
            Fill Form
        </a>
        ],
    },
```

Figure 22 Code snippet of action required

When user is logged in as a worker, they will be redirected to the 'Action Required' page - the home page for workers. The interface is similar to the Manager's Project Management. However, 'Action Required' only display available forms to worker. Worker may fill up forms through this 'Action Required' page. In figure 22, the code snippet shows how the Fill Form module is implemented in E-Form system.

```
// proceed to process form submission
let created_Date = moment().format("YYYY-MM-DD");
const data = {
    submitted_by: 0, // todo: update user based on logged in owner id
    form_id: (params.id).toString(),
    submission_data: JSON.stringify(formData).toString(),
    created_date: created_Date.toString(),
}

const postParams = (new URLSearchParams(data)).toString();

axios.post('http://localhost:5000/api/FormSubmissions/worker/?' + postParams).then(res =>
    try {
        message.success("Form Submitted Successfully");
        history.push("/worker/status")
    } catch (error) {
        console.log(error)
        throw new Error("Form Submission Failed")
    }

}).catch(err => {
    console.log(err)
    message.error("Form Could Not Be Submitted");
```

Figure 23 code snippet of form submission

One of the key components for workers is to fill up forms that are created by managers. Workers will be required to submit their form for the manager's approval after filling up the form.

In figure 23, the code snippet shows how form submission is implemented in the E-Form system.

```
<div style={{ height:'100%' }}>
    <p style={{textAlign:"left"}}>
        <b>Form Name:</b> {submission.form_name}
        <br/>
        <b>Project Title:</b> {submission.project_title}
        <br/>
        <b>Submitted Date:</b> {submission.created_date.substring(0,10)}
        <br/>
            <Tag color={colors[submission.form_status]} key={submission.form_status}>
            <b>
                Status:
            </b>  {status[submission.form_status]}
            </Tag>

            { submission.form_status == 2 ?
            <><br/>
            <Tag style={{ marginTop:"1em"}} color={'orange'} key={submission.form_status}>
                <b>Rejection Reason:</b> {submission.reject_reason}
            </Tag></>
            :null}
        {/* todo: confirm if name should be used instead of id  */}
        <br/>
    </p>
</div>
```

Figure 24 Code snippet of form status

After form submission, workers may go to Form Management to check the status of their form submission. When their form submission is rejected, they can see the reason their form submission is rejected.

In figure 24, the code snippet shows how form status is implemented and how E-Form system shows the information of their form submission, with a rejected form as an example.

2.4 Software Organization

Figure 25 Main folders of the E-Form system

Our E-form system contains these 2 main folders, *antd-pro* for the frontend and *EFormApi* for the backend.

2.4.1 Frontend

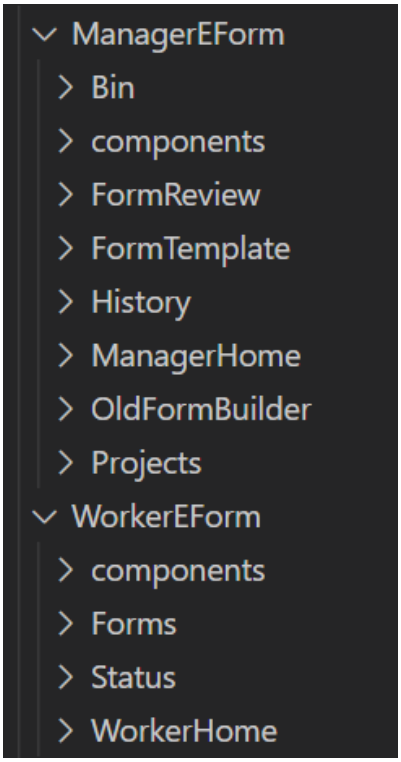


Figure 26 Snippet of frontend code structure

Under the *antd-pro* folder, the folders are further broken down based on the 2 main roles in the E-form system: **Workers** and **Managers**.

Files are organised into their respective folders based on the pages created on the website.

***antd-pro* Folder**	**Website Application**
Manager	
Projects	Project Management
FormReview	Pending Review
History	Form Listing
Bin	Archived
Worker	
Forms	Action Required
Status	Form Management

Naming convention of the pages were adjusted after consulting with the customer their preferred names for the different features in the E-Form System.

2.4.2 Backend

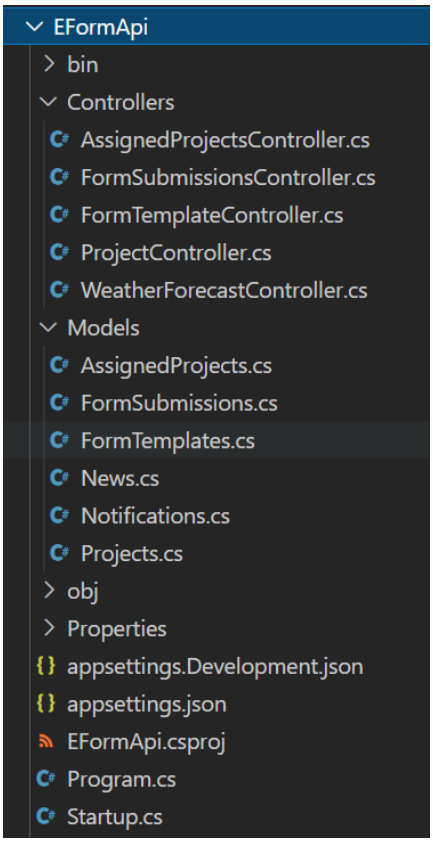


Figure 27 Snippet of backend code structure

As for the backend, files are created based on the entity relationship diagram created for the database.

Model classes defined the properties of the data to be stored in the database.

Controllers are action methods to handle the HTTP request and prepare the response to be sent to the clients.

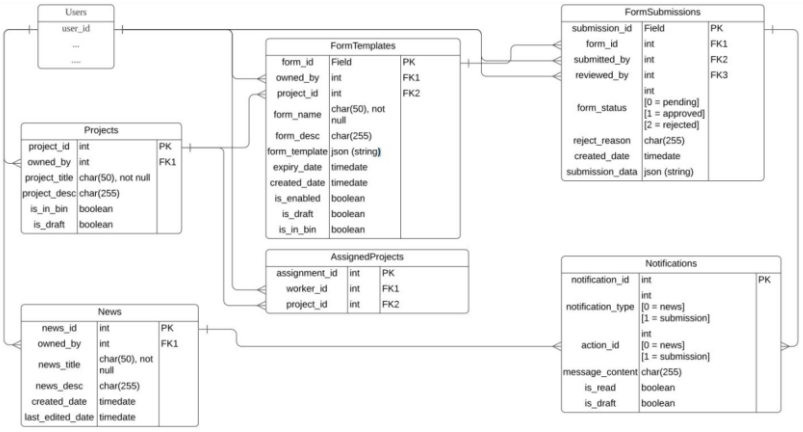


Figure 28 Entity relationship diagram of the E-Form system

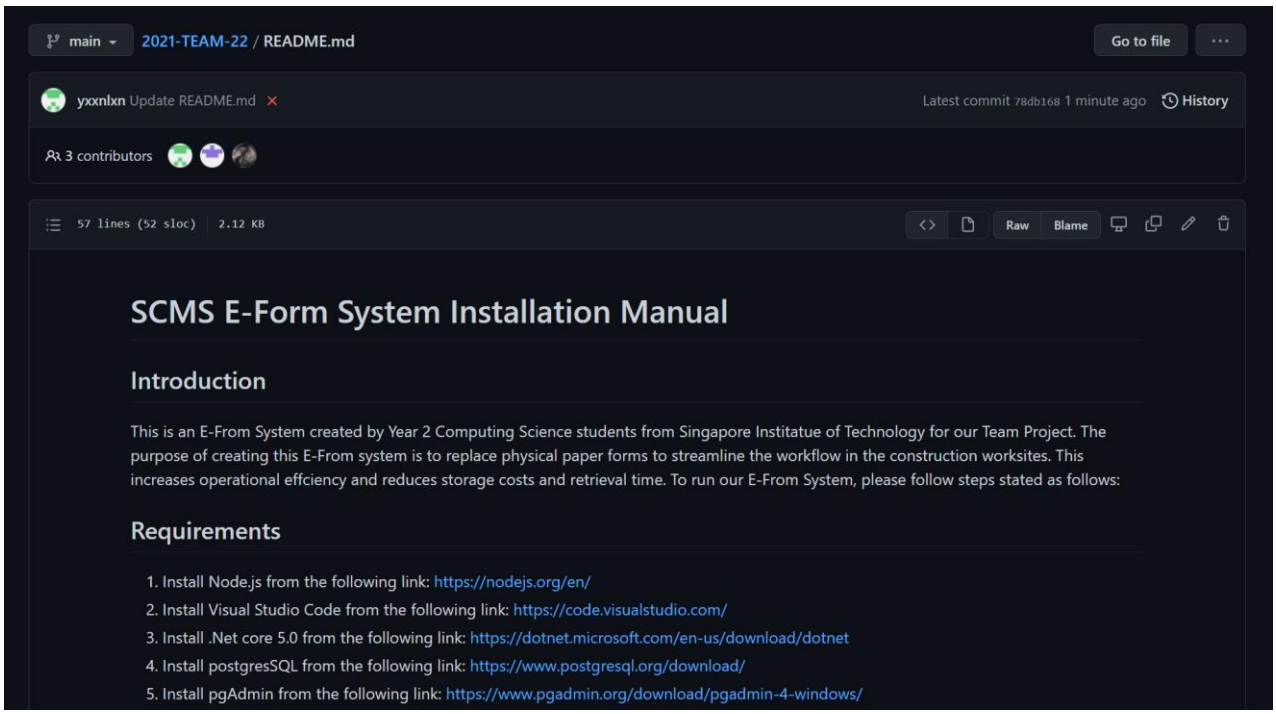


Figure 29 Snippet of installation guide in ReadME.md on GitHub

Installation guide to the E-Form system is created on the GitHub README.md to guide users on how to install the website application. For the full installation guide, you may refer to *Installation Guide* in the appendix or our GitHub.

User documentation is also created to aid the customer to get acquainted with the E-Form System. For the full user manual guide, you may refer to the *User Document* in the appendix.

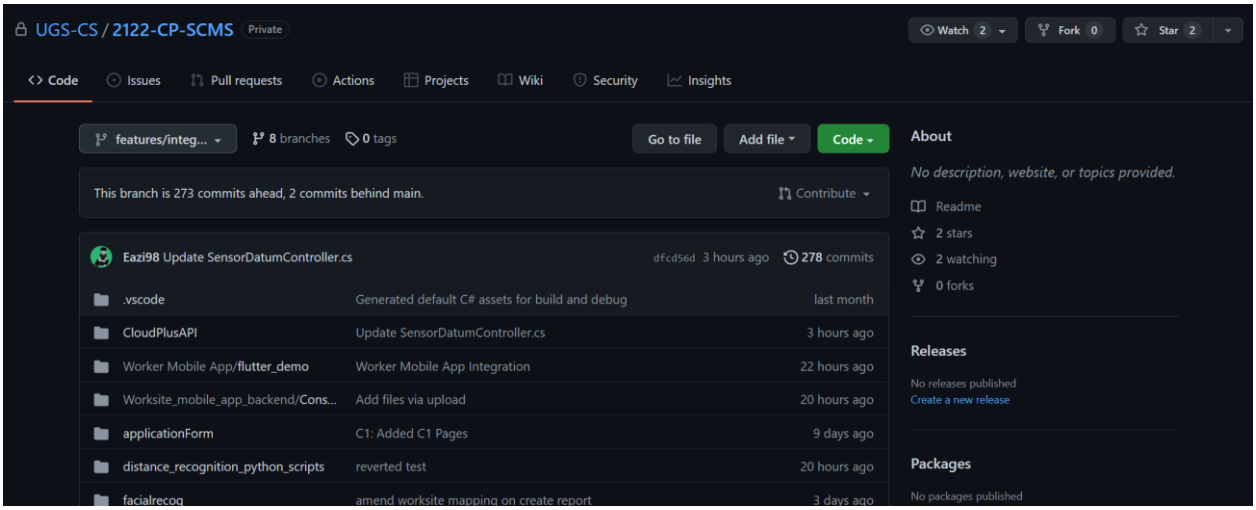


Figure 30 Snippet of the repository for integration

Figure 30 shows the repository used for integration among all the different modules in the website application. This product will be handed over to the customer together with the Installation Guide and User Documentation.

Software Process

3.1 Change Management

Change management is an important process in software development. It transits from an existing state of a software product to another improved state of the product. Change management aims to ensure that changes made in the code follow a sequential process to reduce conflicts and improve performance and implementation of new requirements and features.

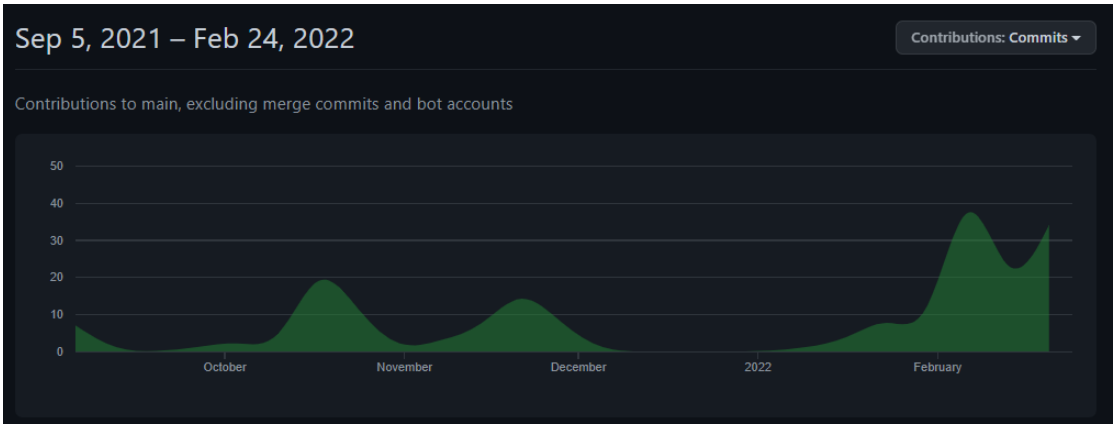


Figure 31 Snippet of contribution made to main from GitHub

As shown in figure 31, there is a consistent number of contributions made by the team. There were still concerns and uncertainties regarding the project requirements at the initial stages. Frequent meetings with the customers were made through Microsoft Teams to clear any existing doubts at hand before proceeding with implementation. Due to the school trimester break, there were hardly any commits from mid-December 2021 to January 2022. The team took this time to create an entity-relationship diagram for customers to visualize better how the database will look and set up the database on PostgreSQL before deploying it to Huawei Cloud Server. Towards the end, requirements are precise, and the team has a clearer direction of what to do, resulting in a spike of commits.

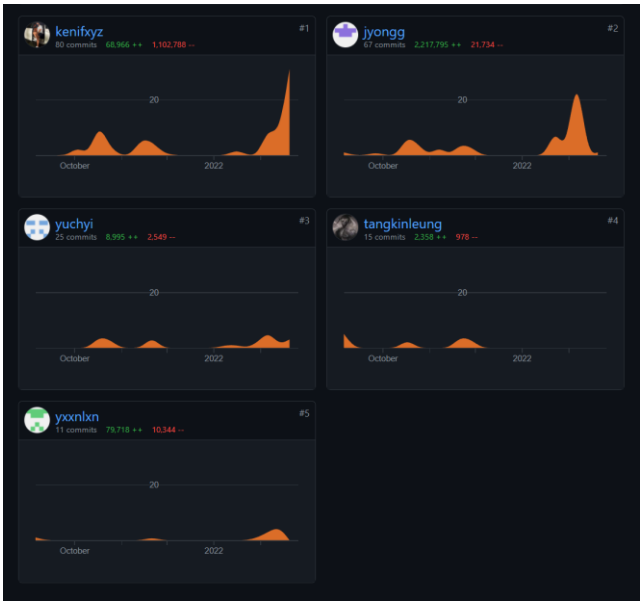


Figure 32 Snippet of contribution made per individual to main from GitHub

The contributions made were not equal since the team consisted of individuals with different strengths and weaknesses. As shown in figure 32, everyone made an effort to contribute to the main branch. Instead of having every team member focus on the form builder module, our scrum master understood the individual strengths and weaknesses early. They allocated tasks such as helping actively in planning, analysis, design, and other miscellaneous tasks such as progress report writing and customer day preparation to members who are weaker in coding. The appropriate tasks delegation results in equal team contribution and increased efficiency in the progress of this project. Hence, figure 32 cannot truly represent fair task allocation and contributions among the team members for the project.

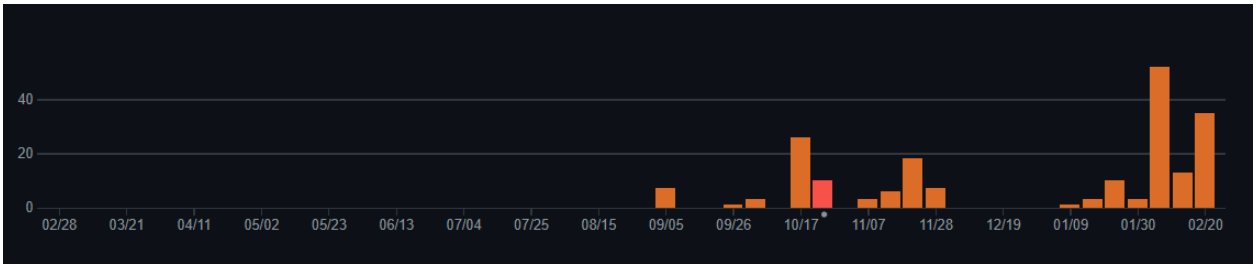
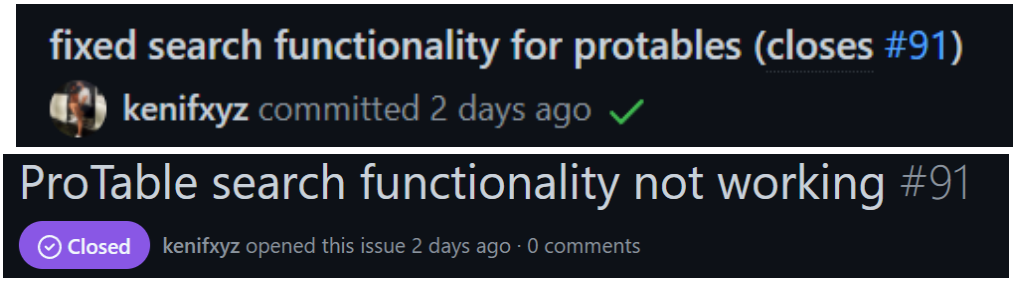


Figure 33 Snippet of commits made to main from GitHub

As shown in figure 33, the team made an effort to have regular code commits throughout the whole project, which aids in having better change management. Having frequent commits allows the team to work on the project steadily and identify build issues earlier and faster. This practice the team adopted would allow us to have an easier time debugging errors. Additionally, the code reversion process will be simplified as every reversion will only affect a small part of the project.

Log Messages with respective Issue Tracker



The team performs good practices in committing log messages by having meaningful commit messages whenever a commit is done as shown above. Log message starts off with action words followed by the main purpose of each commit. Some commits are also linked to issue trackers of the project. This allows easy identification of issues related to respective commits for easy version control. The commits are also concise, containing only one purpose per commit.

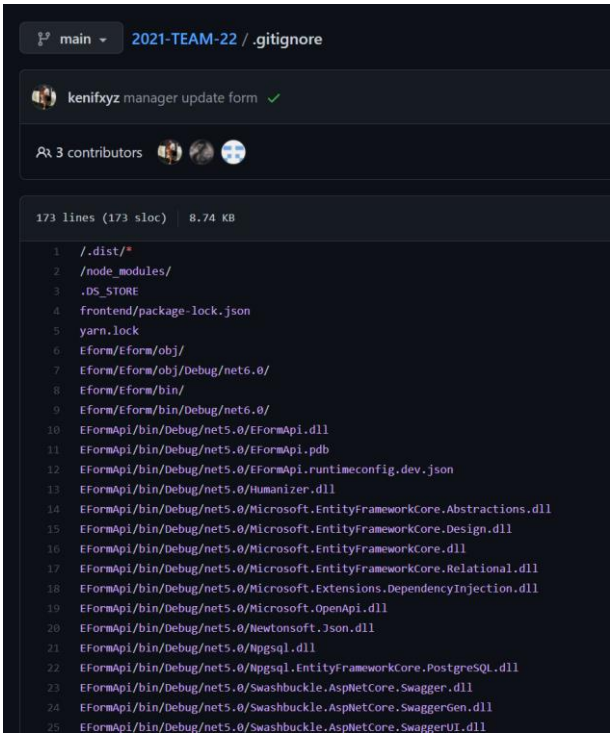


Figure 34 Snippet of .gitignore from GitHub

.gitignore folder is being utilised on files like .class, .exe, .bin etc as shown in figure 33. These build files are being generated during compilation. They should be ignored by the version control system as they will be different for different developer machine. These files generated will unnecessarily increase the size of the repository. Hence, it is best to ignore these files, thereby letting the team better focus on new and noticeably untracked files.

The team has also employed a strategy for concurrent development, as shown in figure 35. In addition, feature branching was created as shown to allow team members greater freedom to develop new features in isolation.

The team uses a hierarchal naming that follows a naming convention of <branch-type>/<new-feature> to better understand the new branches created.

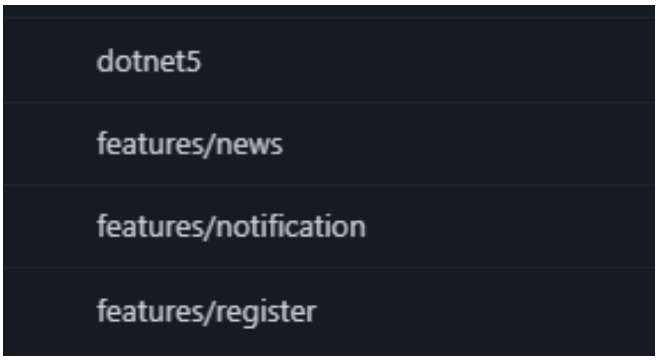


Figure 35 Snippet of feature branching

3.2 Project Planning

Project planning is a procedure undertaken after the requirement gathering with the customer. It aids the team in defining the scope of the project and facilitates software production.

3.2.1 GitHub Wiki Setup

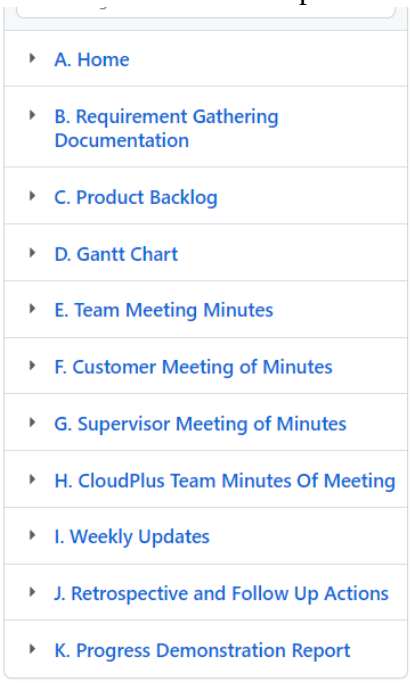


Figure 36 Snippet of GitHub wiki

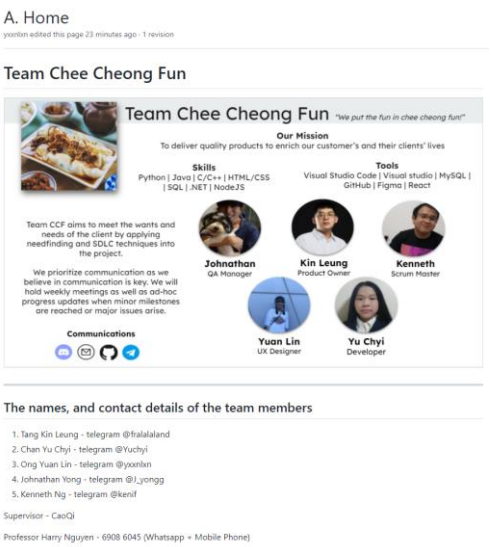


Figure 37 Snippet of wiki homepage on GitHub

Figure 36 shows how the team set up the wiki pages on GitHub. It contains all the essential information documented for the team's future references.

The **Home** page consists of team members' contact details and responsibilities for the project development. This gives a clear definition of what is everyone's role in the project development.

Requirement Gathering Documentation page contains the summary of the project details gathered after requirement day meeting. **Minutes of meetings** are all documented on the subsequent pages to note down what ensues from the meeting.

To manage project planning, changes made every sprint are also documented under **Progress Demonstration Report** page. In addition, at the end of every sprint, updated functional and non-functional requirements will be updated here together with new changes made.

3.2.2 Scope Management

After conducting requirement gathering with the customer, the team created a product backlog, consisting of a prioritized list of work for the team to follow.

The user stories and use case scenarios are created to articulate how a piece of work will deliver a particular value back to the user. By using this technique, it facilitates the team to understand the E-Form System from the perspective of the end-user.

Example of a use case scenario and user stories:

Use Case: FILL UP FORMS IN PROJECT
Goal: Fill up a form for a specific project
Actor: Worker
Preconditions:

- User logged in as worker
- There is an existing form to fill up

Trigger: Clicking of 'Fill Form' button
Scenarios/Events:

1 Users click on 'Fill Form' button.
2 System displays form window of existing form.
3 User fills up the form with details.
4 Users click on 'Submit Form' button.
5 System notifies user of the confirmation of form submission.

As a Manager, I want to view projects, So that I know what projects are created and available in the E-Form system.	As a Manager, I want to create projects, So that I can make new projects that are not in the E-Form system.	As a Manager, I want to manage projects, So that I can make changes to it.	As a Manager, I want to move projects/forms to trash bin, So that I can remove those that are completed or no longer relevant from my view.
As a Manager, I want to search projects, So that I can find a project quickly in the E-Form system.	As a Manager, I want to create new forms, So that my workers can fill in the form as required from them.	As a Manager, I want to save form as template, So that I can create a form more quickly and efficiently.	As a Manager, I want to review forms, So that I can approve or reject my workers' forms.
As a Manager, I want to fill in the reason why I rejected submitted form, So that my workers understand why their forms are rejected and correct their mistakes.	As a Manager, I want to remove projects/forms permanently from trash bin, So that I can delete those that are completed or no longer relevant.	As a Manager, I want to retrieve projects/forms from trash bin, So that I do not have to make a new one if I moved them to the trash bin by accident.	As a Manager, I want to create new announcements in 'Latest News', So that my announcements can reach my colleagues or workers quickly.
As a Worker, I want to view projects, So that I can find what project I am assigned to, and what forms I am required to fill in quickly.	As a Worker, I want to fill in forms, So that I can submit them to my manager as required from me.	As a Worker, I want to check form status, So that I can see if my forms are approved, pending or rejected.	As a Worker, I want to view my form's reason of rejection, So that I can correct my mistakes and submit a new form.
As a Worker, I want to view what is on latest news, So that I can view my managers' latest announcements to us.	As a Worker, I want to view notifications, So that I can find what are the new changes that I might have missed, such as updates of latest news and form status.		

Figure 38 User Stories

ID	As a...	I want to be able to...	So that...	Type	Effort	Priority	Status
			SPRINT 1 (23 September 2021 - 28 September 2021)				
1	Manager	See the flowchart of the system	I can have a better idea on how it works on my end	Flowchart	5	Must	Done
2	Worker	See the flowchart of the system	I can have a better idea on how it works on my end	Flowchart	5	Must	Done
3	Manager	See the prototype of the system	I can have a visualization of how the system works on my end	Figma Prototype	8	Must	Done
4	Worker	See the prototype of the system	I can have a visualization of how the system works on my end	Figma Prototype	8	Must	Done
5	Manager	Log In	I can access the system	Ant Design	21	Must	Done
6	Worker	Log In	I can access the system	Ant Design	21	Must	Done
7	Manager	View Existing Projects	I can access to whichever project I need to view	Ant Design	21	Must	Done
8	Manager	View Projects	I can see existing projects	Ant Design	21	Must	Done
9	Manager	View Form Review Page	I can see what forms are there for me to review	Ant Design	21	Must	Done
11	Worker	View Forms page	I can see the forms I need to fill up	Ant Design	21	Must	Done
14	Manager	Create Project	My workers can find the project's respective forms	Ant Design	21	Must	Done
10	Manager	View History Page	I can see my previously reviewd form and ammend is required	Ant Design	21	Should	Done
12	Worker	View Form Status page	I can see the status of the form submitted	Ant Design	21	Should	Done
			SPRINT 2 (29 September 2021 - 2 December 2021)				
13	Manager	Create Customized Form	My workers can fill in the forms	Ant Design	21	Must	Done
15	Worker	View available forms	I can fill in forms that are required from me	Ant Design	21	Must	Done
16	Manager	Approve a Form Submission	The worker that submitted it can proceed	Ant Design	21	Must	Done
17	Manager	Reject a Form Submission	The worker that submitted it can resubmit accordingly	Ant Design	21	Must	Done
18	Manager	Delete forms (both templates / customized)	Forms that are expired can be removed	Ant Design	21	Must	Done
19	Manager	Delete projects	Projects that are completed can be removed	Ant Design	21	Must	Done
20	Manager	Delete forms (both templates / customized)	Forms that are expired can be removed	Ant Design	21	Must	Done
21	Manager	Delete projects	Projects that are completed can be removed	Ant Design	21	Must	Done
22	Manager	Set expiry date to forms	Workers will not be able to view or fill the form	Ant Design	21	Must	Done
23	Manager	View history of approved or rejected forms	I can recall the actions I did previously	Ant Design	21	Must	Done
24	Worker	Fill in forms	I can work at the worksite without problems	Ant Design	21	Must	Done
25	Manager	Bin Page	I can see forms/projects I have deleted and retrieve them	Ant Design	21	Should	Done

Figure 39 Snippet of product backlog from google sheets

As shown in figure 39, the product backlog consists of all the required tasks to do, planned by the team. Product backlogs consist of all the user stories on the left-hand side. The right side describes the type of the task as well as the effort required to complete the task using Fibonacci agile estimation. Efforts are determined by the team members using the planning poker technique. MoSCoW prioritization method is also utilized to determine the priority of the task. Status indicates whether the task is completed, in progress or not yet started.

Once the tasks are planned out and organized on google sheets, they will be transferred to GitHub every sprint.

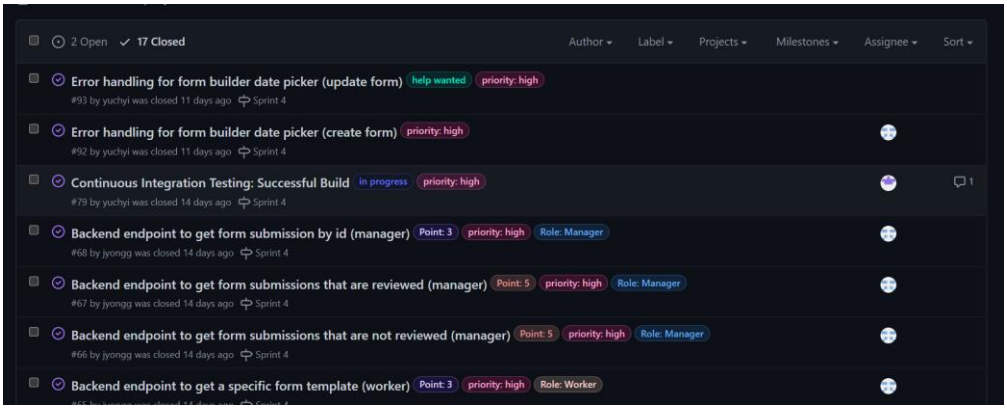


Figure 40 Snippet of issues opened for sprint 3 on GitHub

Issues are only opened at the start of every sprint with only updated tasks based on feedback, agreed deliverables during customer days and team's progress on the E-Form System.

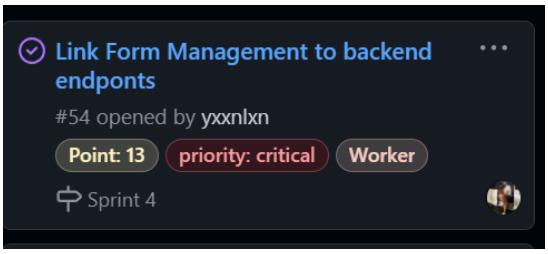


Figure 41 Example of an issue created

Issues created come with labels to further describe the task and team members could be assigned to specific tasks. This will provide a clearer idea regarding the priority of the tasks as well as which tasks the members are working on. More detailed explanation of the labels can be found at *GitHub Label Table* in appendix.

3.2.3 Project Estimation

Accurate estimation of various measures is necessary to ensure effective management. Hence, the team considered the following factors to estimate a realistic outcome to be presented.

The team was given approximately seven months to complete the project, from mid-September 2021 to March 2022. After listing down the tasks required to be completed, the team starts estimating the effort required per task. Gantt chart is also created to assist in planning and scheduling the time required to finish per task.

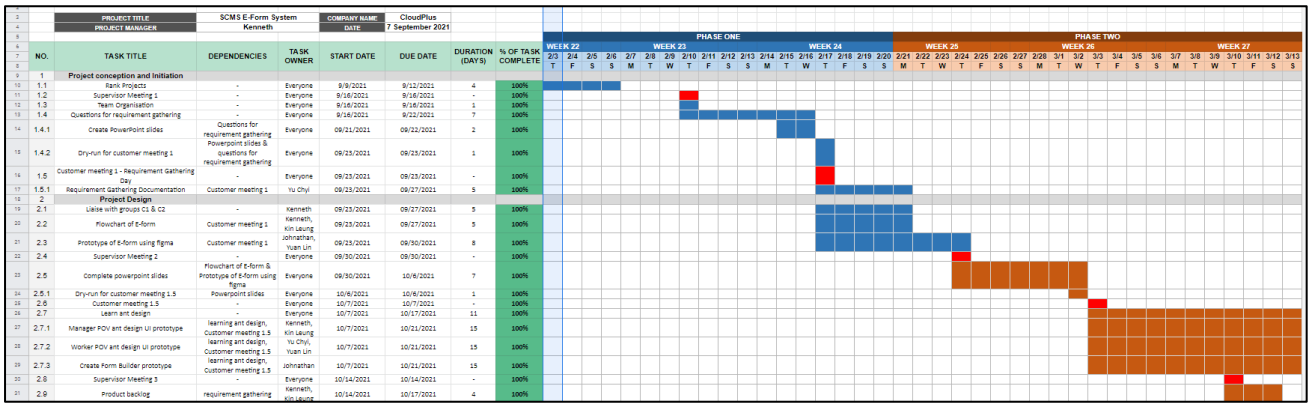


Figure 42Gantt Chart

Before every customer day, the team conducts a sprint review meeting to review the past sprint and discuss what is expected of the next sprint. At every meeting, the team will run through the E-Form

system with latest updates and determine what has been done and what has not been done. Then, the team proceed to prepare the documents for the customer day. During discussions, the team will plan questions to ask the customer and set aside time for the customer's feedback.

There were times when the team underestimated the time required for specific tasks. For example, during the first customer day meeting, the team foresaw that the discussion would take a long time due to its initial product development stage. Therefore, the team delegated about 15 minutes for the customer to provide feedback. However, the team was taken aback when the customer was pleased with the project flow and had not much feedback for the product. The team soon realised that the time could be better utilised if the team came prepared with additional questions for discussion. For the subsequent meeting, the team prepares additional questions for the customer to maximise the 30 minutes. (You may refer to *Example of Additional Questions for the Customer* in the appendix for example of questions prepared)

3.2.4 Project Scheduling

Project scheduling refers to the roadmap of all activities in a specific order and within the allotted time slot for each activity. The product owner will update the product backlog, and the team will discuss the allocation of tasks for the upcoming sprint. After task allocation, issues are created on GitHub.

An example of an issue created on GitHub is shown in figure 43. Each issue comes with labels for the assignee to better understand the assigned task. Labels are utilized to prevent long-winded explanations on a task and allow easy categorization. Searching for issues is also made easier by employing the usage of labels. Labels created can be found in the *GitHub Label Table* appendix.

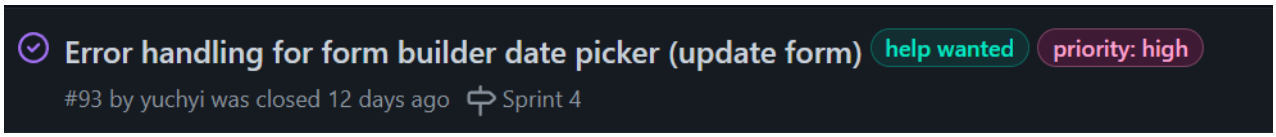


Figure 43 Snippet of issue for error handling

Other than opening issues at the start of the sprint, issues are also open for error handling amid the sprint as shown in figure 43.

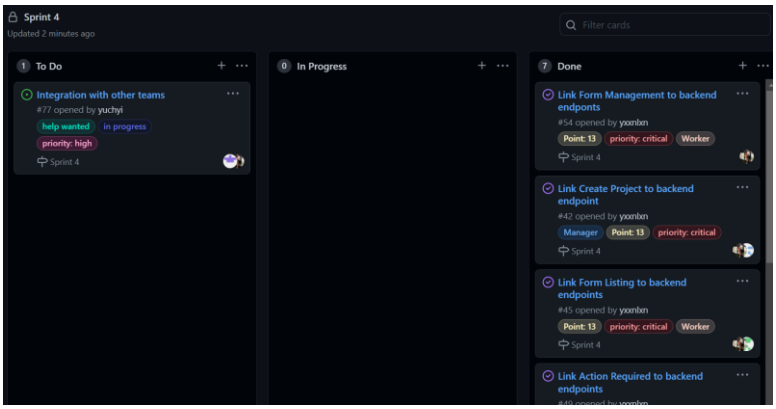


Figure 44 Sippet of project tab in GitHub

The team utilises the Kanban board on GitHub to assist in visualising work, limit work in progress and maximise efficiency. Team members can have a clearer view of team progress with one look at the Kanban.

These are the techniques adopted by the team for project planning to ensure smooth tracking of the project development to ensure maximum efficiency.

3.3 Quality Assurance

3.3.1 Automated Testing and Continuous Integration

The team employs Cypress for automated testing. Cypress is a JavaScript test automation used for web applications. Please refer to the following link to view the automated testing: https://youtu.be/fUha4Huk0H4

```yaml
name: Cypress Tests

on: [push]

jobs:
  cypress-run:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout
        uses: actions/checkout@v2
      # Install NPM dependencies, cache them correctly
      # and run all Cypress tests
      - name: Build and Start
        uses: cypress-io/github-action@v2
        with:
          working-directory: ./antd-pro
          build: yarn build
          start: yarn start

#       - name: Database Testing
#         uses: cypress-io/github-action@v2
#         with:
#           working-directory: ./EFormApi
#           run: dotnet run

#       - name: Automate Testing
#         uses: cypress-io/github-action@v2
#         with:
#           working-directory: ./antd-pro
#           start: yarn start
#         env:
#           CYPRESS_RECORD_KEY: ${{ secrets.CYPRESS_RECORD_KEY }}
#           DEBUG: "@cypress/github-action"
```

Figure 45 Cypress used for automated testing in continuous integration

Figure 45 shows where the code is used for continuous integration on GitHub.

Figure 46 Snippet of code committed undergoing continuous integration

Whenever a code is being committed to the repository, the code will undergo continuous integration and it shows that the software test suite is effective as the build/test cycle executes in less than 4 minutes as shown in figure 46.

By employing continuous integration, the team can detect bugs quicker so that we can improve the software quality.

3.3.2 Code Review

The purpose of conducting code review is to detect defects in the software and to identify refactoring opportunities for poorly structured code. Code reviews are opened as issues on GitHub as shown in figure 47.

Figure 47 Code review opened as issues on GitHub

3.3.2.1 Creating

The author creates a new issue for code review as shown in figure 48.

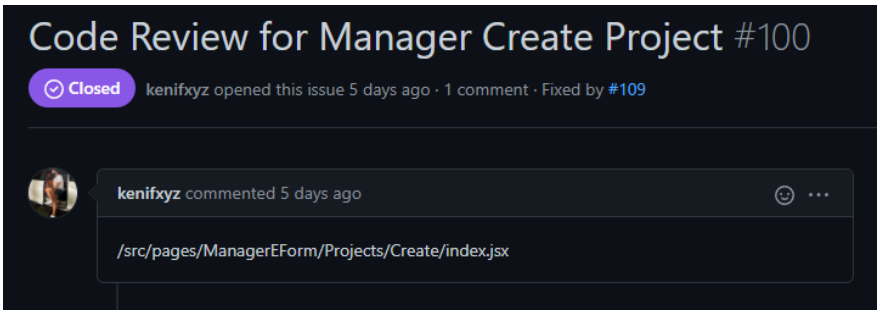


Figure 48 Creating a code review request

To make it simpler for the reviewer who is going to review the code, the author also provides the file path to the code in the comment section.

3.3.2.2 Previewing

After the author analyses his own code, he will assign to one or more of the group mates to be a reviewer, as shown in figure 49.

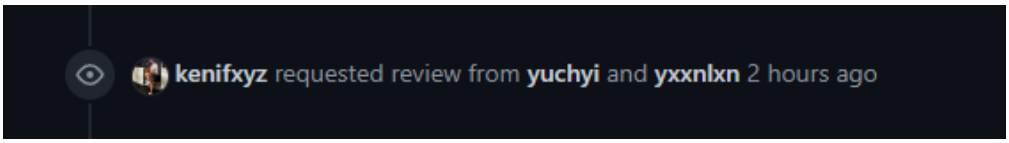


Figure 49 Snippet of the code being assigned to a reviewer for code review

3.3.2.3 Commenting

Figure 50 Snippet of reviewer commenting on the code for code review

The reviewer will analyze the code and then leave feedback for the author to make changes to it. This can be seen in figure 50.

3.3.2.4 Addressing Feedback

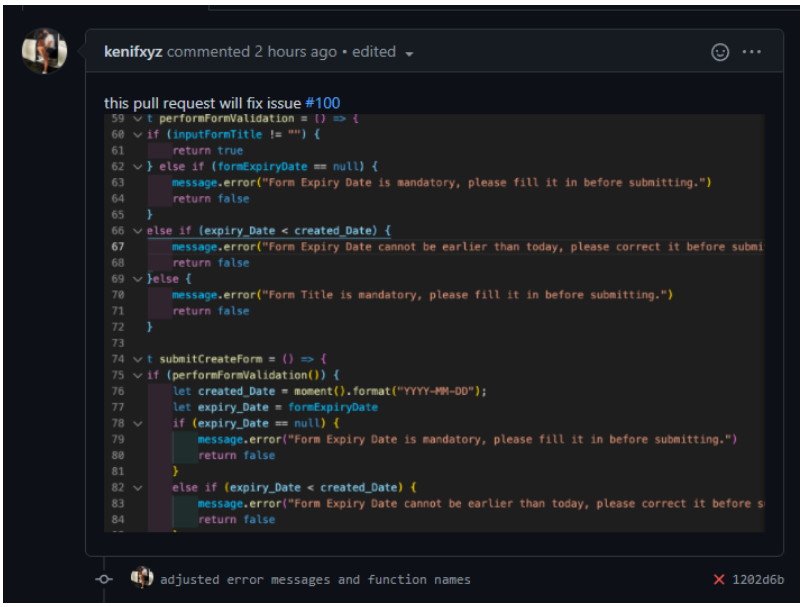


Figure 51 Snippet of author making changes for code review

The author will then make changes based on the feedback given by the reviewer and then attach it as a comment as shown in figure 51.

3.3.2.5 Approving

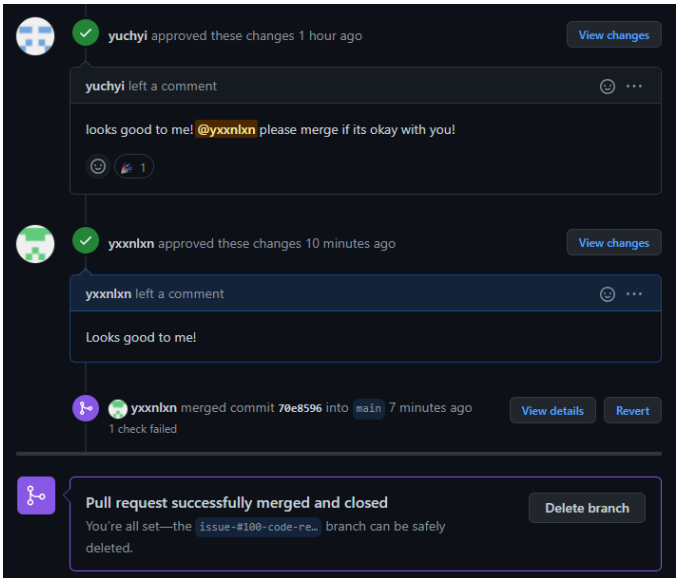


Figure 52 Snippet of reviewer closing the issue after appropriate changes are made for code review

The reviewers will then analyze the code again to make sure changes are made appropriately and then close the issue as shown in figure 52.

3.4 Process Improvement

The purpose of process improvement is to ensure there is an ongoing effort to improve the efficiency of the team's project development strategy.

At the end of each sprint, the team gathers for retrospective meetings to evaluate the past working cycle. During each meeting, the team utilises a '**Start, Stop, Continue**' technique to come up with practical ideas for team-based improvement.

Figure 53 Start, Stop, Continue

The team also came up with 9 different emojis to describe what the team feels for every sprint.

Figure 54 One Word

'**5 Whys**' technique is also implemented to find the exact root cause of the issue the team faced.

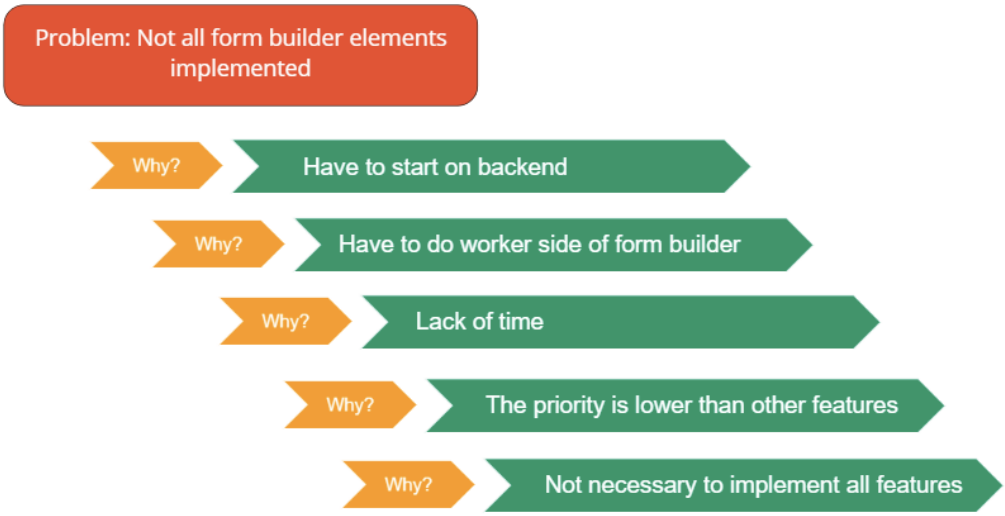


Figure 55 5 Whys

Utilizing these techniques helps the team evaluate the feedback given by all members to ensure continual improvement. The team can find potential issues that cause inefficiency and rectify them as soon as possible from the feedback.

Reflective Essay

It had been seven months since this Team Project (TP) started. This Team Project has been a fun and memorable experience that allowed us to step out of our comfort zone and work with people we are unfamiliar with - from the awkward introduction of each team member to working well as a team and overcoming various challenges together.

Team Projects allowed us to work on a real-world project before stepping out into the working industry. From this E-Form System project, we learnt how to analyze project requirements and derive the various software modelling such as use case diagram, class diagram and technical skills such as using .NET Core, react, database (PostgreSQL) and Huawei cloud. We also learned about the importance of communication skills, be it in terms of communication within the team or communication with the clients.

4.1 Key Takeaways

One key takeaway from TP would be that communication is the key for a team to work well. Initially, as a newly formed team, it was hard for everyone to speak their mind freely, which led the project to have a rocky start where the team members rarely update each other on what they were doing individually. However, after the team started implementing a weekly stand-up meeting that serves as a platform for all team members to speak up, the team started to warm up to each other and communication within the team was done more frequently. After the ice breaker, there is an evident spike in the amount of work done by the team as everyone has a better understanding of the project and no longer hesitates to speak up when unsure about any parts. As a team, everyone will inevitably have different perspectives. From this project, we have learned how to put ourselves into the other's shoes, look at things from the other perspective, and talk things out before things get out of hand.

An example would be the team members having conflicting views on whether the team should plan for integration early. The team was initially divided as some felt we should standardize the website with the other teams before proceeding to simplify the final integration. However, there are also teammates that we should focus on our project first and only plan for integration after completing our project. To resolve this conflicting view, the team had a meeting to discuss how we should proceed and discuss the concerns and viewpoints of everyone. After a lengthy discussion, the team settled on focusing on our project first. It is hard to get all teams on board to do the standardization with all teams under CloudPlus, especially since each team does not have any concrete plans yet and the project requirements are ever-changing. However, thanks to excellent communication within the team, the conflicts within the team did not escalate, which may have led to a total fallout within the team.

The second takeaway would be the importance of communicating with the client. In this TP project, agile practices were used. Thus, the team followed the agile manifesto - individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation and responding to change following a plan. At the very start of the project, the team had a project requirement gathering meeting with CloudPlus to understand the project requirement better. In addition to the customer meeting day arranged by the school, the team had requested additional meetings with the client. During these privately arranged meetings, the team double-checked if what the team had planned was in line with the customers' expectations before implementing the features. For instance, creating a high-fidelity prototype and checking if the design is acceptable to CloudPlus before coding. Other than having meetings with CloudPlus, the team used email as the primary form of communication, whereby the team would clarify questions that were not clarified during the meetings. Lastly, the team had also sent CloudPlus the meeting minutes after every meeting so that CloudPlus would have a record of what was discussed during the meeting, which they could refer to in case they forgot what was previously agreed. We believe the team was able to meet the customers' expectations for every customer day meeting is attributed to our constant communication with them. Hence, allowing our project to be able to proceed smoothly.

The third takeaway would be the experience of working on a real-world project. We get to experience how agile methodology is carried out and can interact with companies such as CloudPlus and take on the various roles of scrum master, product owner and developers. To ensure that the workload is distributed equally and to maximize the learning opportunity for each team member, the roles are rotated throughout the whole project. The roles rotation allows all members to be more involved in the project and leads to better teamwork and cohesion as everyone would have a similar experience by the end of the project. Additionally, TP allowed us to better relate to the knowledge taught in Professional Software Development (PSD) 1 and PSD 2. We got to personally experience how Agile software development is carried out and apply the knowledge taught, such as the various software modelling diagram – use case diagram, activity diagram. We believe that the experience gained from TP will be greatly beneficial to us. The experience allows us to have a peek at the actual working industry and thus better prepare us for stepping out into the workforce – involved in software development with foreign teammates, learning new programming languages required for a project, interactions with clients.

4.2 Challenges Faced

Through this project, the team had faced some challenges and overcame them together.

The first challenge is the deep learning curve of our project. As this is the first time the team has been exposed to a real-world project, the team lacked experience in this aspect and not all team members are well-versed in the programming knowledge needed for this project. Before the commencement of the project, the team had to spend some time reading up on .NET core, PostgreSQL, and ReactJS to learn about how they work so we would have a clearer idea of how to do the project before we start working on it. To overcome this challenge, we had meetings to discuss the individuals' strengths and weaknesses. We split our job scope according to everyone's strengths and weaknesses to maximize our efficiency. Whenever someone has difficulties in understanding or doing part of the project, the teammate that is more proficient in that aspect will guide to ensure that everyone learns together, and the team progresses as one.

The second challenge would be project integration. Even though everyone had experiences when it came to working on a project, it was a new experience for the team to work with other teams as there were many people involved, which made the project integration challenging. The challenges in project integration are that every team has different progress on their project, and the standardization of project design between teams is very time-consuming. While the teams tried to be on the same page on project designs for ease of integration, some of the project designs might be incompatible for project integration. In addition, bug-fixes, revisions, and quality control are common in projects, and the time it takes to complete varies between teams. This leads to waiting times, and the teams have less time for project integration. To overcome these issues, continuous communication was done between teams to standardize their project and integrate them. During integration discussions, only representatives from each group will discuss the standardizations, then relay the information to their respective team members. This arrangement allows minimal coordinating efforts and constructive interactions, increasing productivity for integration progress. Additionally, meetings were held weekly to ensure all teams were on the same page for integration for project integration to proceed smoothly.

The third challenge would be time management. Throughout the project, the team needs to be prepared and be responsive to changes as requirements are updated after every customer meeting day - improvements and changes must be made to meet the customers' expectations. These changes and improvements undoubtedly increase the complexity of the team project, which affects the time to complete the project. Moreover, while the team members would have liked to dedicate more time and effort to the team project, other school commitments such as tests, assignments, and reports required our attention. It was incredibly challenging to juggle these commitments and work on the team project, especially when every member was in a different group for the other modules, as different groups would have meetings that would start and end at different timings. To make things worse, everyone in the team has different working and sleeping hours, making it even more challenging to set up a meeting together at the same time.

The team had foreseen these issues and came up with potential solutions to overcome this challenge.

As we had identified the issue of time constraints early on, we decided to focus on ensuring that we will be able to produce a minimum viable product for CloudPlus by the end of the project and create a timeline based on it. The first solution would be to make a priority list for features in the project so that we can identify and keep track of the features to be implemented. The team made a priority list, ranking each feature based on its impact and then working on the features that ranked top in the list. For example, one of the key features was the form builder, a compulsory component of the E-Form

system. The team worked on the form builder as soon as possible and refined it before focusing on additional form elements. The second solution the team came up with was managing and communicating expectations. We actively communicate our ideas and clarify our doubts with CloudPlus frequently, understanding their needs and aiming to create the minimum viable product early. In addition to allowing the team to stay on track and not waste time implementing incorrect features/functions, this decision allows us to be involved with our stakeholders and spend more time refining the product together. Lastly, the team defined goals, plans, responsibilities, and expectations initially. In the early phase of the project, the team members got to know each other well and understand each other's strengths and working styles. This allowed us to delegate work to members best suited quickly. We have also created a Gantt chart and product backlog to help ourselves stay on track and complete our designated tasks before deadlines, updating each other's progress or asking for help through group chats. Additionally, to avoid potential meeting conflicts with other module meetings, we agreed on having a weekly meeting every Wednesday to ensure that Wednesday is reserved for TP and meetings for other modules should not be held on that day.

The fourth challenge would be the technical issues of linking the backend of .NET with the frontend of our project. Requests such as GET/POST/PUT/DELETE could be made from the frontend successfully, but we could not pass additional parameters like submitting form information to the backend at first. To resolve this technical challenge, the team made a separate internal lightweight web application to quickly make interactions between the frontend and backend and test the endpoints. We also used applications such as Postman to help us debug if there were any issues with a given endpoint. Subsequently, when the endpoints were working on the lightweight web app, we migrated the snippets of code that made backend requests to the main project.

The fifth challenge would be initially unable to solve the state management for form builder elements. Each question to be added to the form builder is a different type of component on the form builder page. For example, the input field and checkbox field are two different types of components, and they are the children of the form builder page component, which is the parent. When the information in the component is updated, there is difficulty passing back the updated state to the parent component, the form builder page. For example, the name of the question is entered in by the user, or the position of the question is changed. This was especially difficult for questions with nested states, such as checkboxes and radio buttons, which allowed the user to add or delete options. Our solution that we came up with was to build helper functions within the form builder page component that would update the state for each type of question component. These helper functions would have the context of the type of question and the data that is expected to be updated. For example, if the question component is for a checkbox, the helper function would expect a list of checkboxes options and update the state of the relevant element according to the list of questions.

4.3 Removed Features

Due to several factors such as time constraint and integration issues, features such as the news module and drag and drop form builder were not implemented.

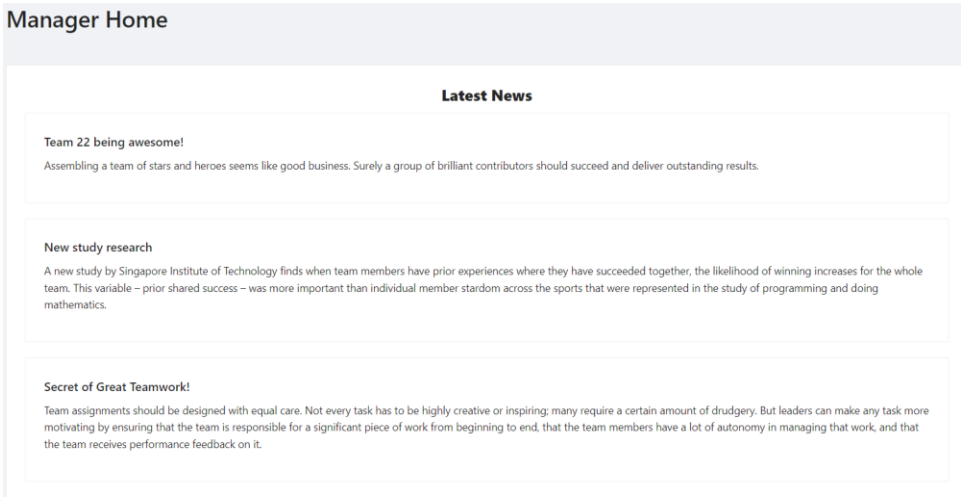


Figure 56 Snippet of news module for E-Form system

The news module was proposed as an additional feature to allow easy deployment of announcement and information to all workers and managers at their respective homepage. However, due to conflicts during integration, this feature seems redundant and brings no beneficial aid to the E-Form system and thus has been scrapped from the final product.

Another feature which was not implemented is the project manager's ability to assign specific forms to a specific worker. This was scrapped as well due to the team lack of control on the website application's user management module which was handled by another group. Instead, the team decided to implement a search feature so that the workers can have an easier time finding the forms that they need to fill in. Figure 57 below shows a snippet of how the search function looks like on the website.

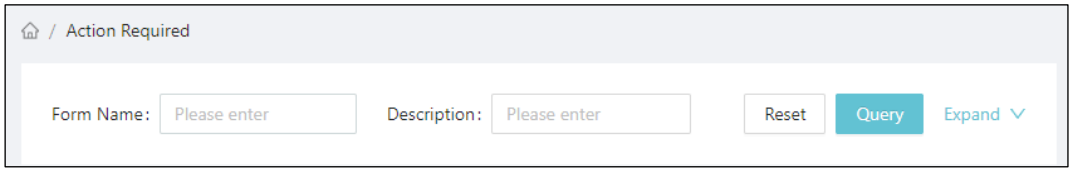


Figure 57 Snippet of search function

The current implementation of the form builder is similar to Google forms. However, the team propose that this form builder can be further improved with the elements having a drag and drop functionality. By doing so, the form elements would be much more intuitive and convenient for the users to use. An example of a drag and drop form builder is shown in figure 58. This feature was only discovered amid the project with the form builder being close to completion. Hence, it is inexpedient to change to a new feature.

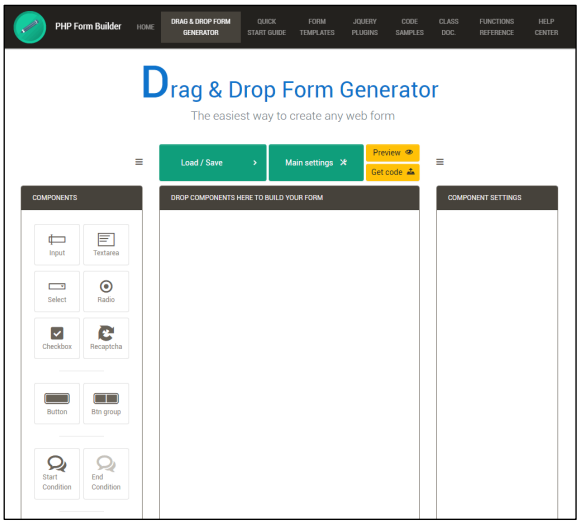


Figure 58 Snippet of drag and drop function in E-Form system

Conclusion

With the rising cost of construction site projects, there is a need to replace physical paper forms, leading to productivity loss and lower operational efficiency for employees. The team project – E-Form System, aims to be the solution that can reduce storage costs and streamline the workflow in construction worksites, leading to an increase in productivity and efficiency in employees.

The team project is an excellent opportunity for us as students to apply our practical knowledge, gain exposure to real-world projects and understand the expectations from projects that aim to resolve real-world issues. In addition, the team project teaches us problem-solving, critical thinking, and time management skills.

During the development phases of the E-Form system, the team constantly applies what we learned in CSC2002 Team Project, CSC2011/2012 Professional Software Development, and incorporates the knowledge into our development cycles. It helped the team understand each other's strengths and weaknesses early on, which sped up our workload completion and significantly helped with our time management.

Despite the deep learning curve and multiple challenges such as time management and project integration, the team brainstormed ideas to create a satisfactory product that tackled real-world issues. The team members agreed whole-heartedly that the team project is an excellent learning opportunity that is beneficial in our growth as future programmers.

We deeply feel that the product we have created will be helpful and impactful in the lives of real workers and managers at construction sites across Singapore if deployed to production by CloudPlus. This is largely because of the convenience and time savings the system brings, as well as how well designed it was – systematically and aesthetically. The team is extremely thankful and proud to have been able to be given the opportunity to work on this project.

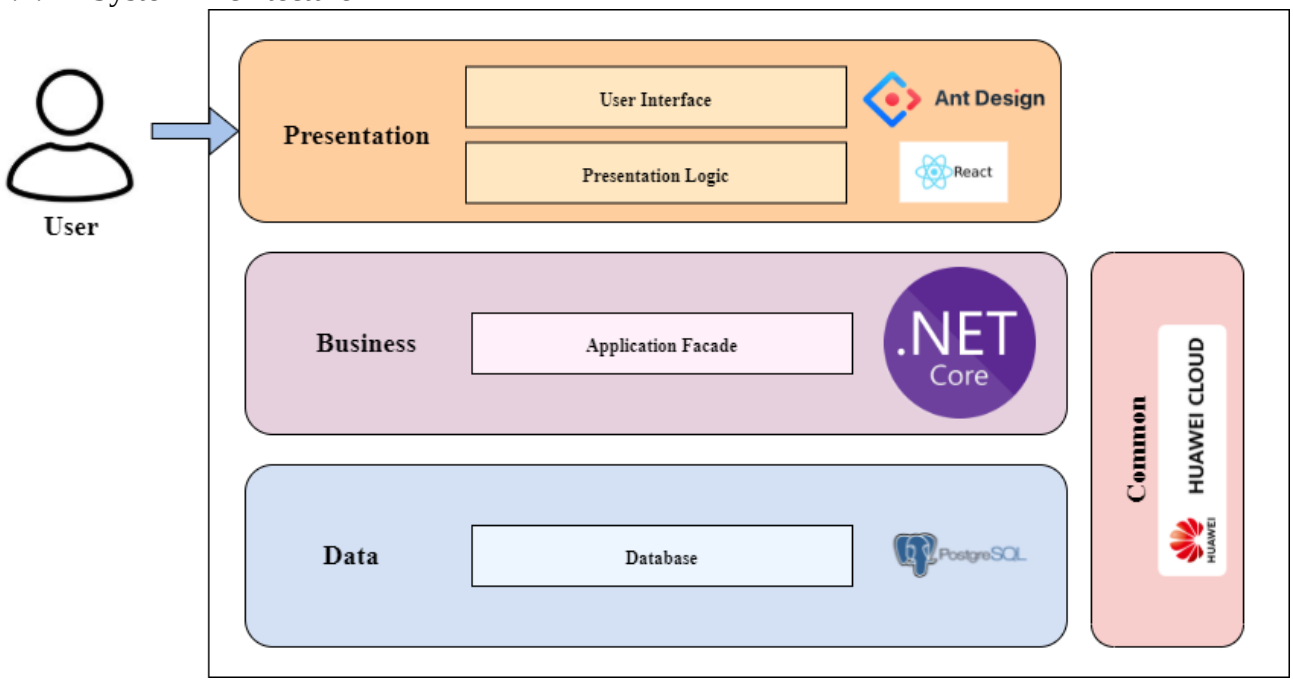
Reference

1. *We are your one stop AIoT solution provider*. Cloud Plus. (n.d.). Retrieved from https://www.cloudplus.com.sg/

Appendix

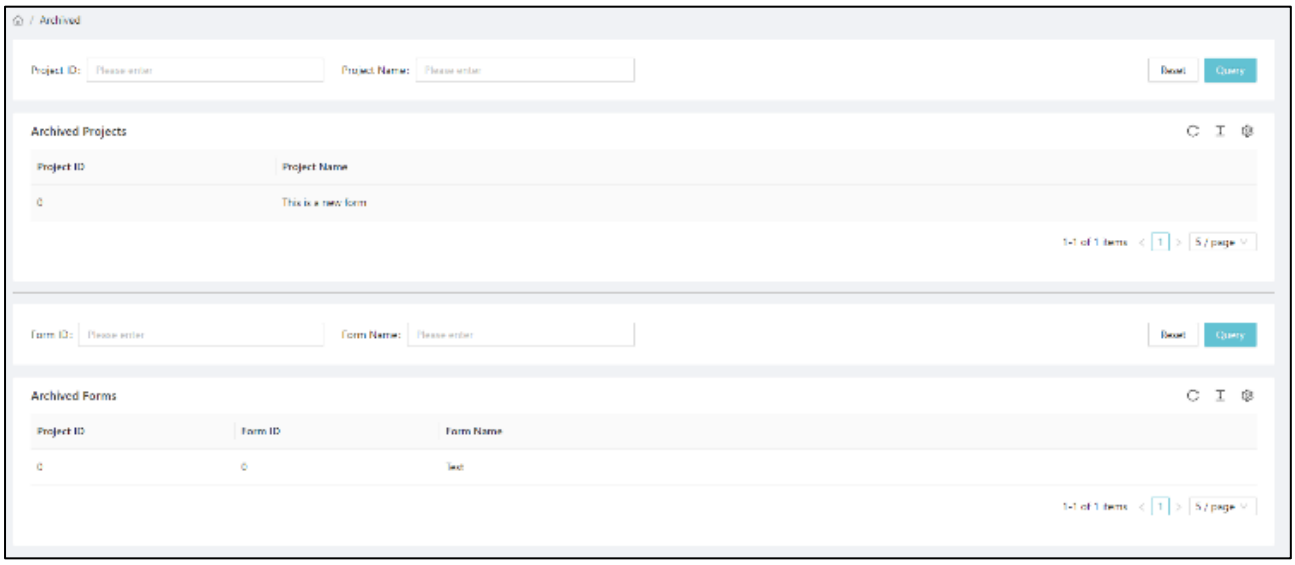
7.1 Software Product

7.1.1 System Architecture



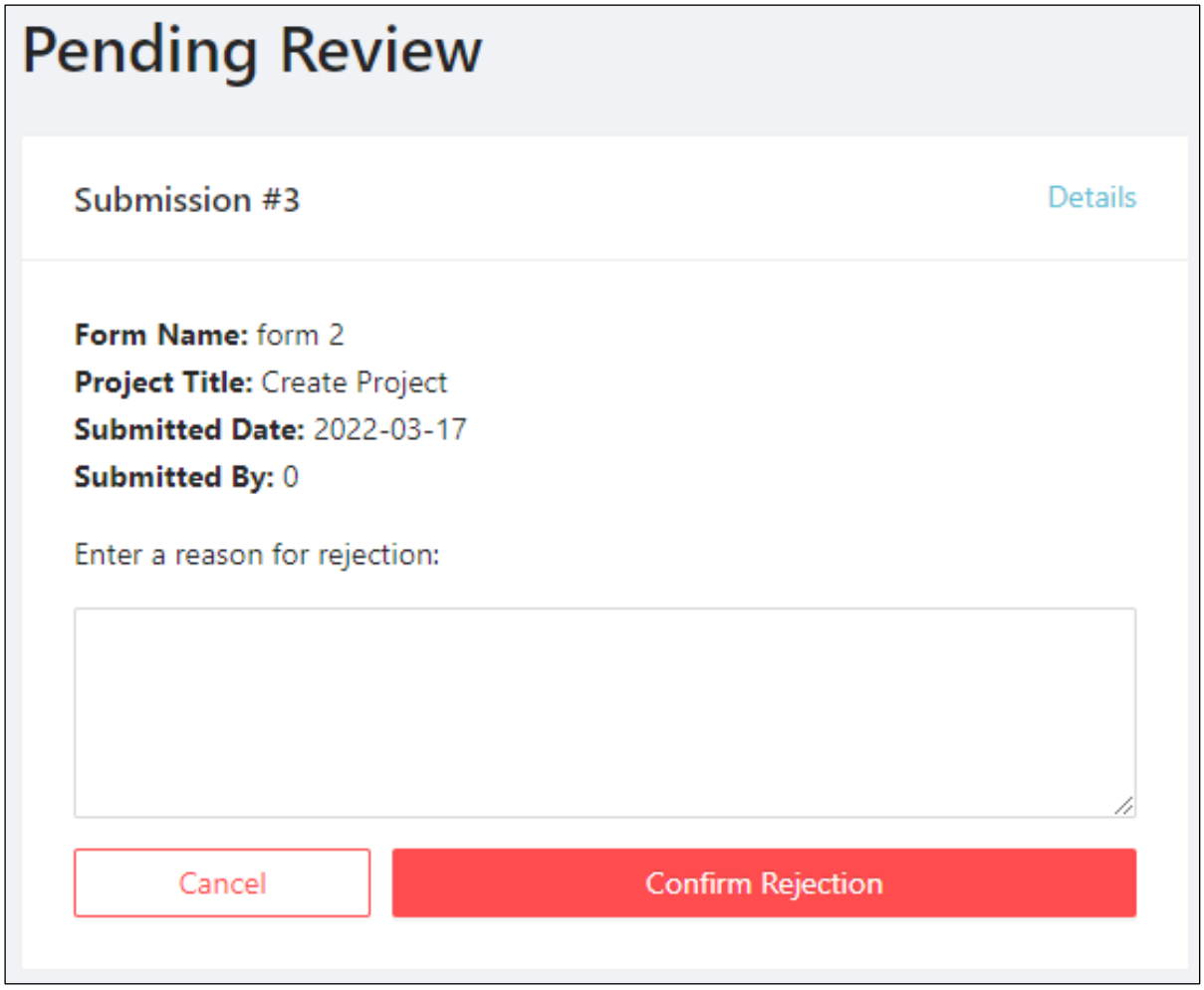
7.1.2 E-Form System

7.1.2.1 Archived Section



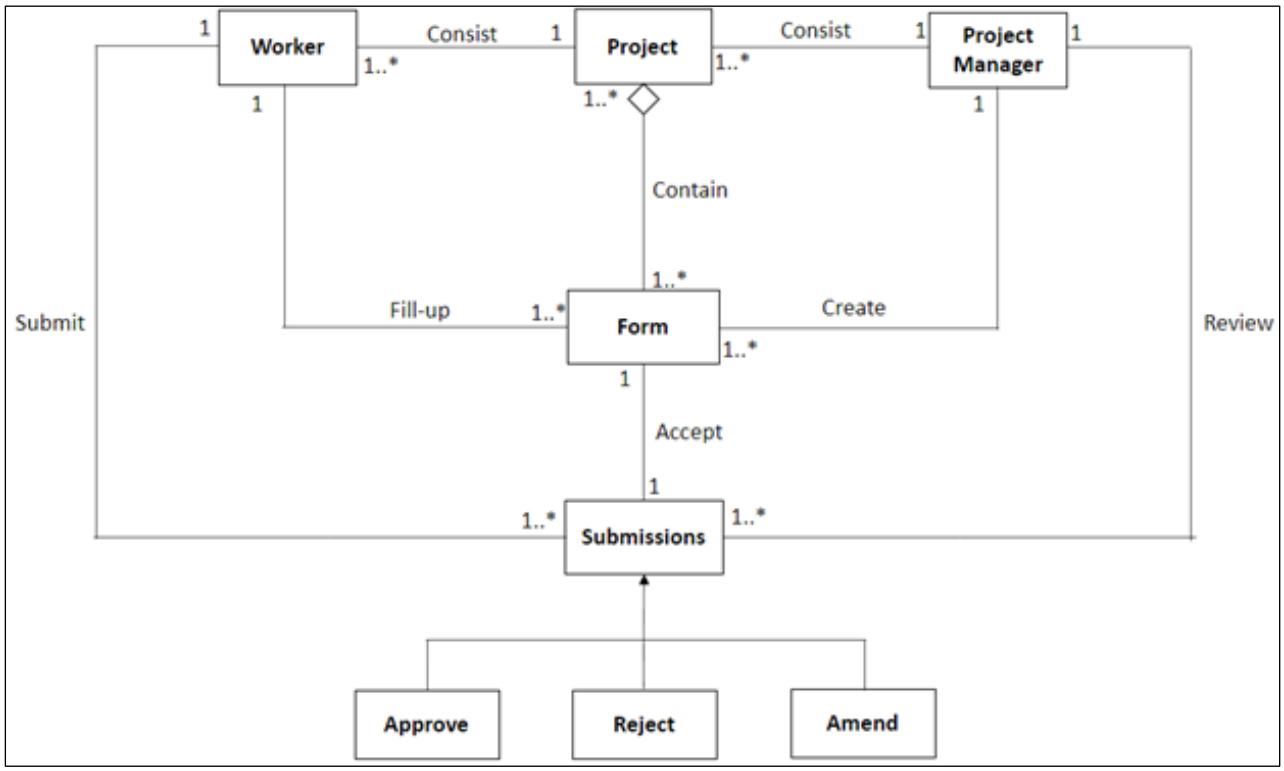
7.1.2.2 Search Function

7.1.2.3 Reason for Rejection

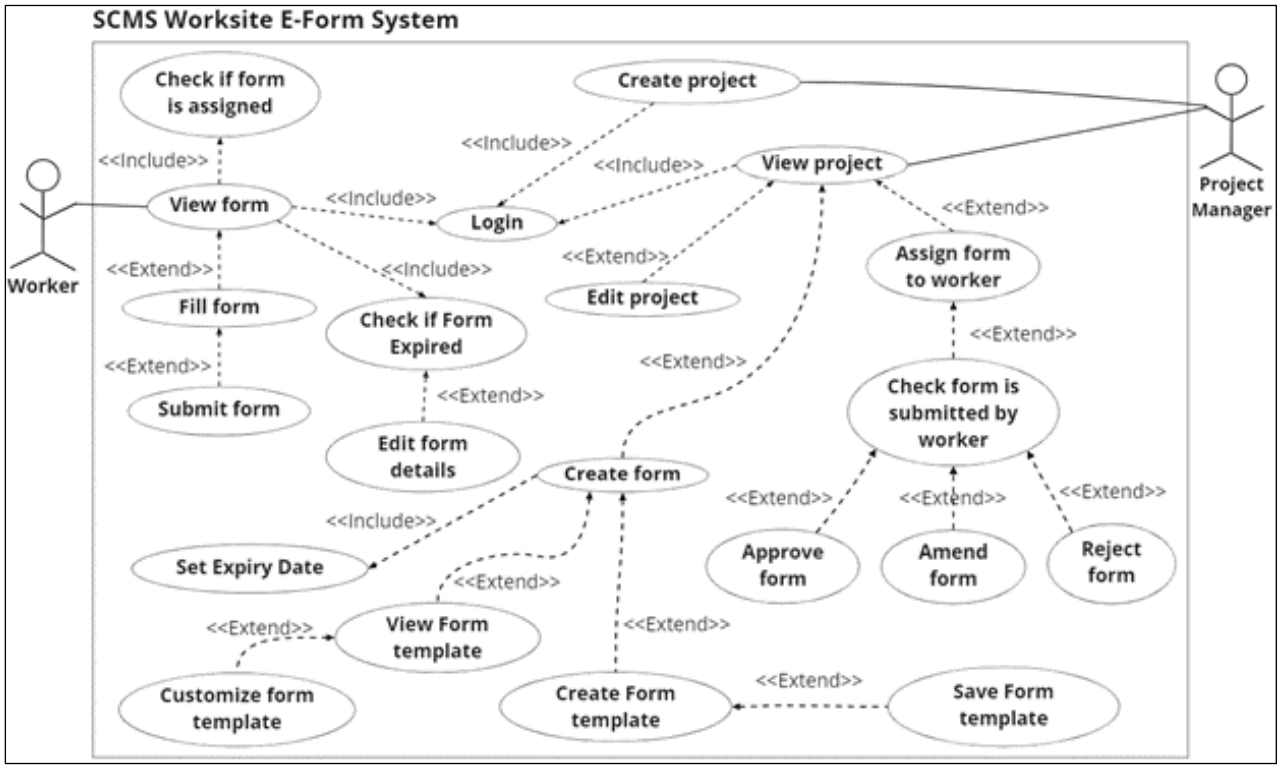


7.1.3 Unified Modeling Language Diagram

7.1.3.1 Class Diagram of E-Form System

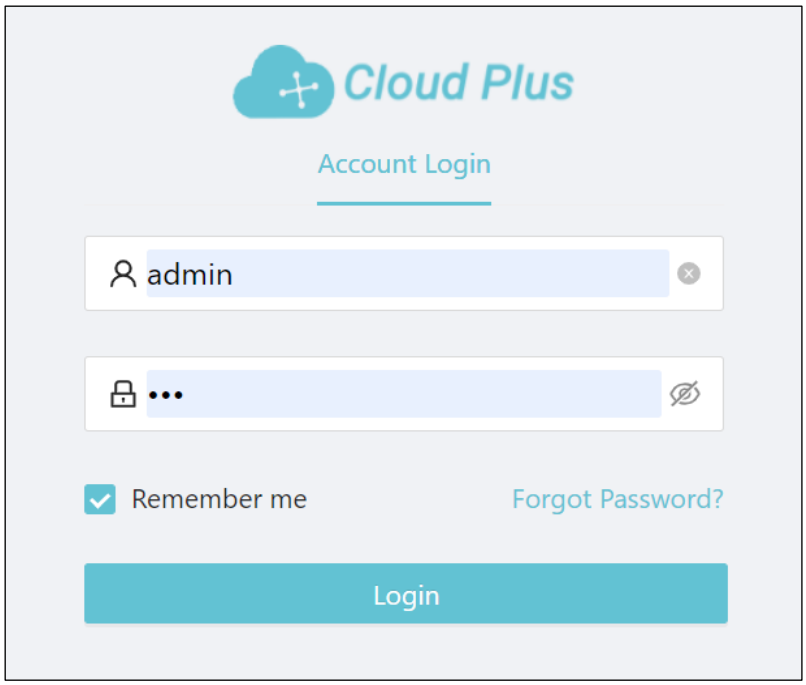


7.1.3.2 Use Case Diagram of E-Form System

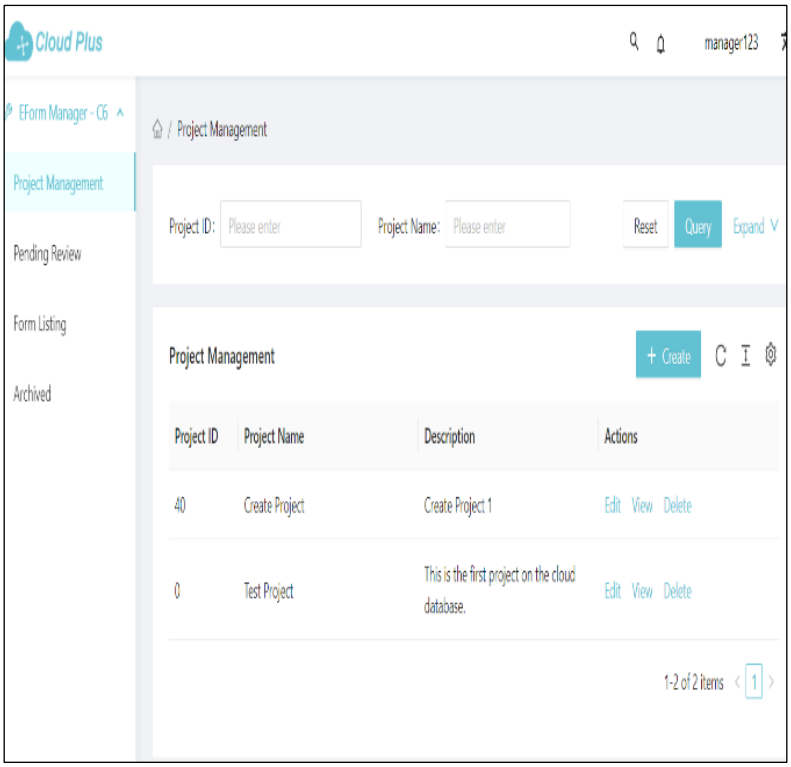


7.1.4 Implementation

7.1.4.1 Project Creation and Customization




```
'POST /api/login/account': async (req, res) => {
  const { password, username, type } = req.body;
  // await waitTime(2000);

  if (password === '123' && username === 'admin') {
    res.send({
      status: 'ok',
      type,
      currentAuthority: 'admin',
    });
    access = 'admin';
    return;
  }
```

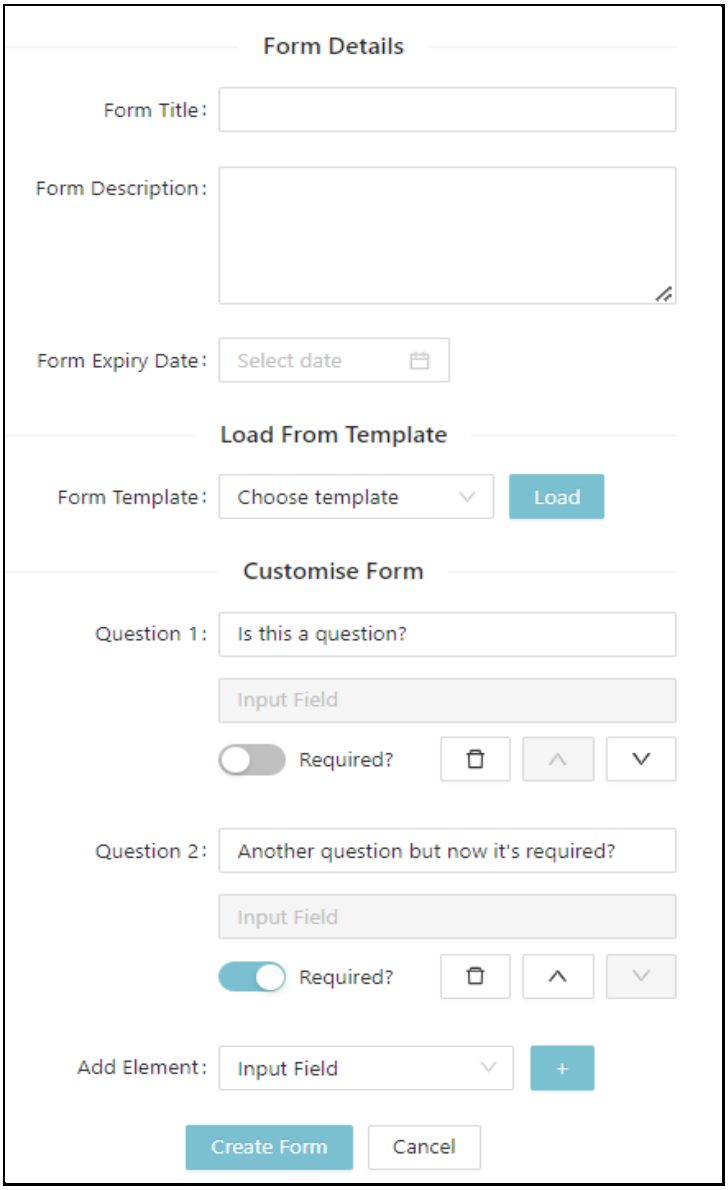



```
onst submitCreateProject = () => {
    if (validateForm()) {
        const data = {
            owner: 0, // todo: update user based on logged in owner id
            title: inputProjectTitle,
            desc: inputProjectDescription,
        }

        const params = (new URLSearchParams(data)).toString();


        axios.post('http://localhost:5000/api/Project?' + params).then(res => {

            try {
                const redirect_id = res.data[0].project_id
                message.success("Project Created Successfully");
                history.push("/manager/projects/manage/" + redirect_id.toString())
            } catch (error) {
                throw new Error("Project Creation Failed")
            }

        }).catch(err => {
            message.error("Project Could Not Be Created");
        })
        console.log("data posted")
```

```
ender: (text, record, _, action) => [
    <a key="editable" onClick={() => {
        history.push("/manager/projects/edit/" + record.project_id)
    }}>
        Edit
    </a>,
    <a
        onClick={() => {
            history.push("/manager/projects/manage/" + record.project_id)
        }}
    >
        View
    </a>,
```

```
<Popconfirm
title={<div><b>Confirm Delete Project?</b><br/>All forms within this project will be

onConfirm={() => {

        const data = {
            id: (record.project_id),
        }

        const postParams = (new URLSearchParams(data)).toString();
        console.log(data)

        axios.delete('http://localhost:5000/api/Project?' + postParams).then(res => {
            try {
                console.log(res)
                message.success("Project Deleted Successfully");
                actionRef.current.reload();
            } catch (error) {
                console.log(error)
                throw new Error("Project Deletion Failed")
            }

        }).catch(err => {
            console.log(err)
            message.error("Project Could Not Be Deleted");
        })
```

7.1.4.2 Form Creation and Customization

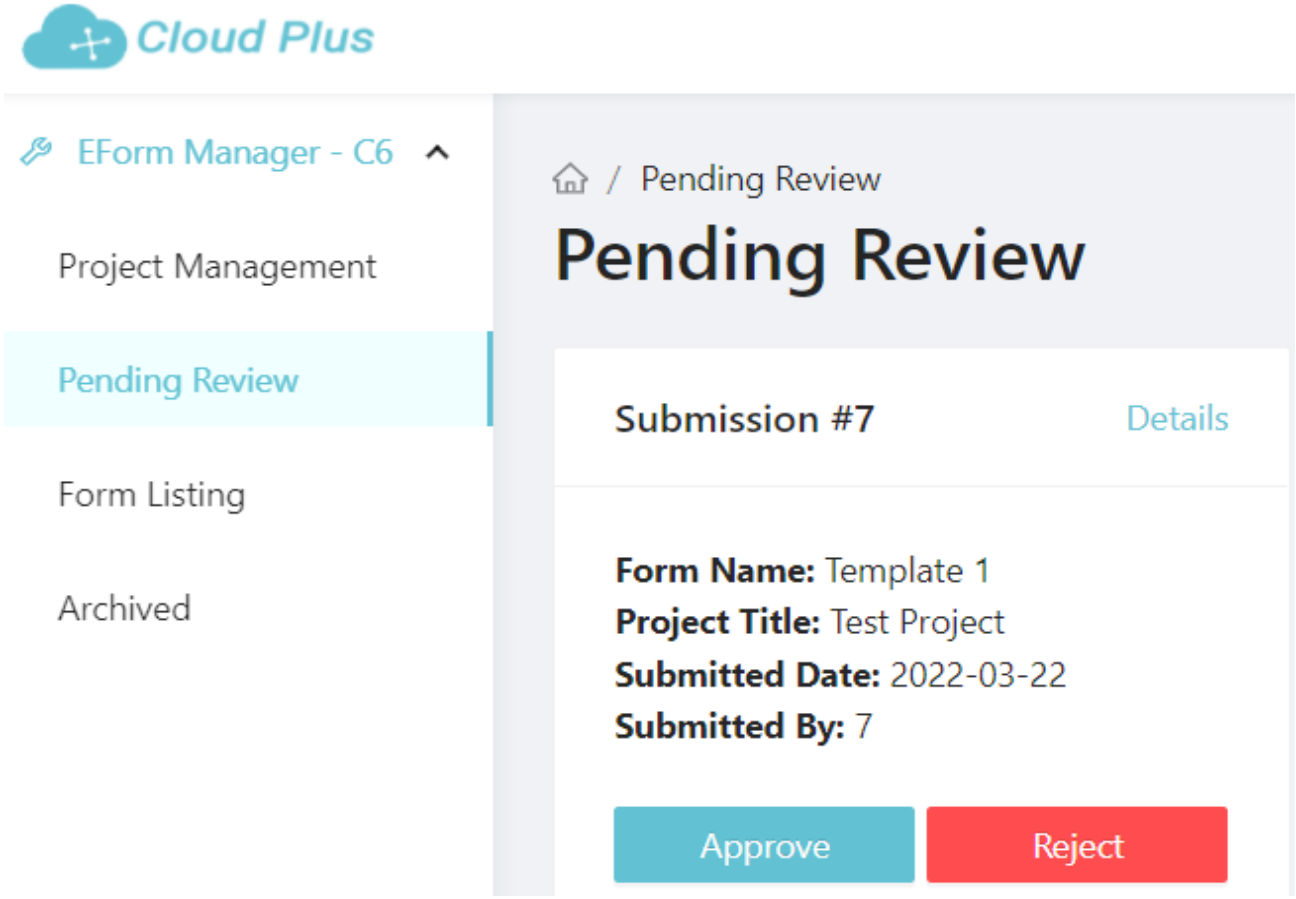



```
const BuilderMain = (props) => {

    const [formElements, changeFormElements] = useState(props.formElements)

useEffect(() => {
    // send default form elements to parent on first render
    props.setFormElements(formElements)
}, [formElements])

const setFormElements = (newFormElements) => {
    // logic to pass data to parent
    props.setFormElements(newFormElements)
    changeFormElements(newFormElements)
}
```

```
const addElement = (type) => {
    let newInput = {}
    let newPos = formElements[formElements.length-1].position + 1
    if (type == "input") {
        newInput = {
            position: newPos,
            key: newPos,
            type: "input",
            question: "",
            required: false
        }
        setFormElements([...formElements, newInput])
    } else if (type == "checkbox") {
        newInput = {
            position: newPos,
            key: newPos,
            type: "checkbox",
            question: "",
            checkboxOptions: ["Option 1", "Option 2"],
            required: false
        }
        setFormElements([...formElements, newInput])
    } else if (type == "radio") {
        newInput = {
```

```
function BuilderInputElement(props) {
    const [inputText, setInputText] = useState(props.question)
    return (
    <>
        <Form.Item label={"Question " + (props.position+1)}>
            <Input
            placeholder="Enter Question"
            value = {inputText}
            onChange={(e) => {
                setInputText(e.target.value)
                props.onChangeQuestion(props.position, e.target.value)
                // probably would be more optimal to only save the text to object states when move/submi
            }}
            style={{
                marginBottom: 15
            }}
            />
            <Input disabled placeholder="Input Field" />
            <QuestionModifier
            type="Input Field"
            questionListSize={props.questionListSize}
            required={props.required}
            position={props.position}
            onChangeRequired={(e) => { props.onChangeRequired(props.position, e)}}
            onTriggerPositionChange={(e) => { props.onTriggerPositionChange(props.position, e)}} // +1
```

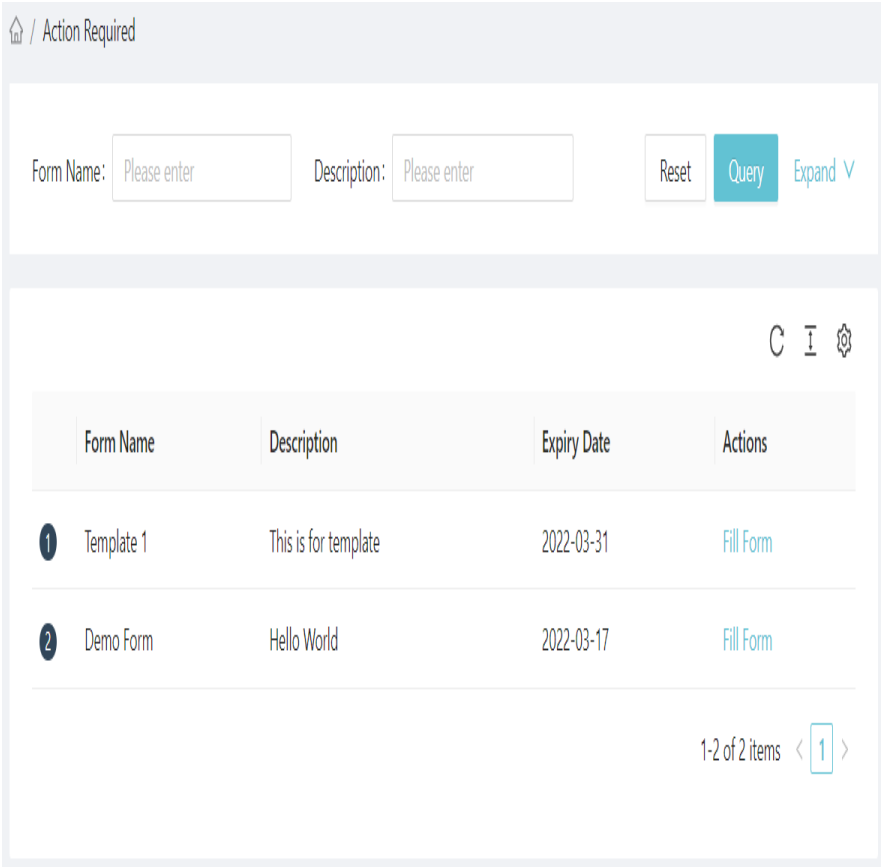
7.1.4.3 Approval and Rejection of Forms




```
const approveSubmission = () => {
    setIsLoading(true)
    const data = {
        submission_id: submissionId,
    }

    const params = (new URLSearchParams(data)).toString();
    axios.post('http://localhost:5000/api/formSubmissions/manager/approve?' + params).then(res => {

        try {
            const redirect_id = res.data[0].project_id
            message.success("Submission Approved");
            setIsLoading(false)
            props.refreshData()
        } catch (error) {
            throw new Error("Approving Submission Failed")
            setIsLoading(false)
        }

    }).catch(err => {
        message.error("Submission Could Not Be Approved");
        setIsLoading(false)
    })
```

```
const rejectSubmission = () => {
    if (rejectReason == "") {
        message.warning("Please enter a reason for rejection")
    } else {
        setIsLoading(true)
        const data = {
            submission_id: submissionId,
            reject_reason: rejectReason,
        }

        const params = (new URLSearchParams(data)).toString();
        axios.post('http://localhost:5000/api/formSubmissions/manager/reject?' + params).then(res => {
            try {
                const redirect_id = res.data[0].project_id
                message.success("Submission Rejected");
                setIsRejecting(false)
                setIsLoading(false)
                setRejectReason("")
                props.refreshData()
            } catch (error) {
                setIsRejecting(false)
                setIsLoading(false)
                setRejectReason("")
                throw new Error("Rejecting Submission Failed")
            }
```

7.1.4.4 Worker Interface



```
{
    title: 'Actions',
    valueType: 'option',
    render: (text, record, _, action) => [
        <a key="editable" onClick={() => {
            history.push(`/worker/forms/fill/${record.form_id}`)
        }}>
            Fill Form
        </a>
        ],
    },
```

```
// proceed to process form submission
let created_Date = moment().format("YYYY-MM-DD");
const data = {
    submitted_by: 0, // todo: update user based on logged in owner id
    form_id: (params.id).toString(),
    submission_data: JSON.stringify(formData).toString(),
    created_date: created_Date.toString(),
}

const postParams = (new URLSearchParams(data)).toString();

axios.post('http://localhost:5000/api/FormSubmissions/worker/?' + postParams).then(res =>
    try {
        message.success("Form Submitted Successfully");
        history.push("/worker/status")
    } catch (error) {
        console.log(error)
        throw new Error("Form Submission Failed")
    }

}).catch(err => {
    console.log(err)
    message.error("Form Could Not Be Submitted");
```

```
<div style={{ height:'100%' }}>
    <p style={{textAlign:"left"}}>
        <b>Form Name:</b> {submission.form_name}
        <br/>
        <b>Project Title:</b> {submission.project_title}
        <br/>
        <b>Submitted Date:</b> {submission.created_date.substring(0,10)}
        <br/>
            <Tag color={colors[submission.form_status]} key={submission.form_status}>
            <b>
                Status:
            </b>  {status[submission.form_status]}
            </Tag>

            { submission.form_status == 2 ?
            <><br/>
            <Tag style={{ marginTop:"1em"}} color={'orange'} key={submission.form_status}>
                <b>Rejection Reason:</b> {submission.reject_reason}
            </Tag></>
            :null}
        {/* todo: confirm if name should be used instead of id  */}
        <br/>
    </p>
</div>
```

7.1.5 Software Organization

7.1.5.1 Installation Guide

[This installation guide is only for local usage, to view the website directly on Huawei cloud, refer to this link: http://159.138.89.74:8000/ and skip to the *User Document* in the appendix if assistant is needed to navigate through the website]

SCMS E-Form System Installation

Introduction

This is an E-From System created by Year 2 Computing Science students from Singapore Institute of Technology for our Team Project.

The purpose of creating this E-From system is to replace physical paper forms to streamline the workflow in the construction worksites. This increases operational efficiency and reduces storage costs and retrieval time.

To run our E-From System, please follow the steps stated as follows:

Requirements

1. Install Node.js from the following link: https://nodejs.org/en/

2. Install Visual Studio Code from the following link: https://code.visualstudio.com/

3. Install .NET core 5.0 from the following link: https://dotnet.microsoft.com/en-us/download/dotnet

4. Install PostgreSQL from the following link: https://www.postgresql.org/download/

5. Install pgAdmin from the following link: https://www.pgadmin.org/download/pgadmin-4-windows/

Steps

1. Clone the repository to your local desktop

Frontend

1. Under command prompt key in the following commands:

To open frontend folder

cd antd-pro

Installation of required package

npm install

Start the website application

npm start

Backend

1. Under command prompt key in the following commands:

To open backend folder

cd EFormApi

Install these packages

dotnet add package Npgsql.EntityFrameworkCore.PostgreSQL --version 5.0.2

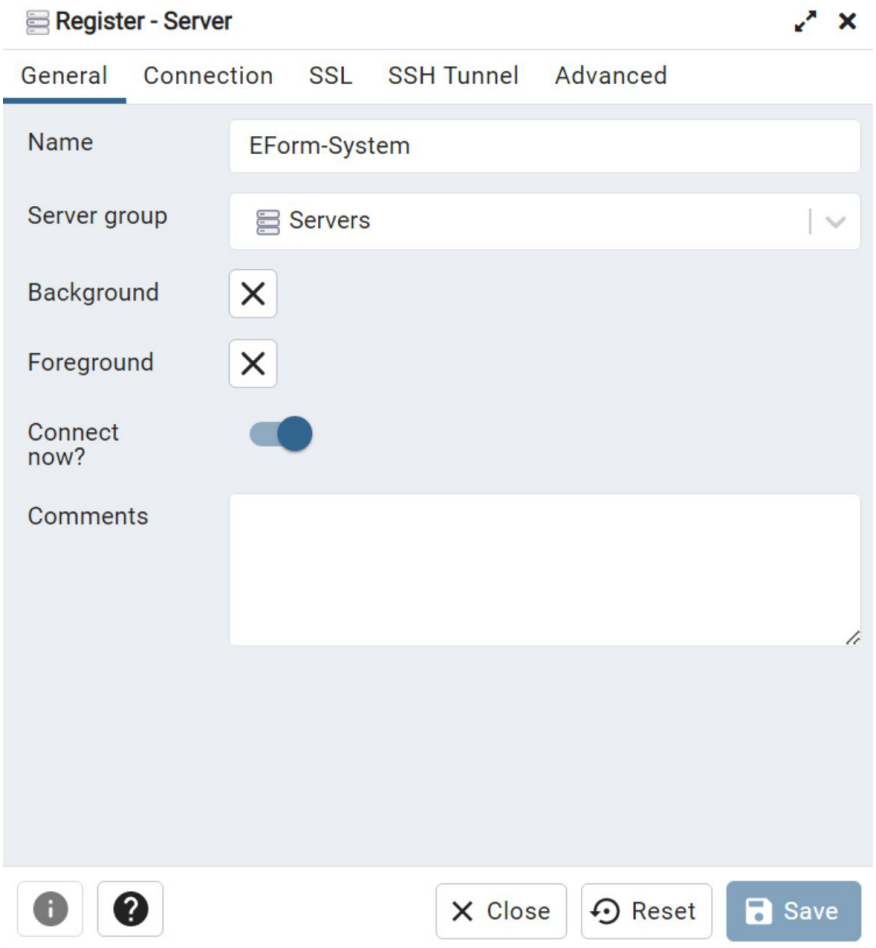
dotnet add package Microsoft.EntityFrameworkCore.Design --version 5.0.5

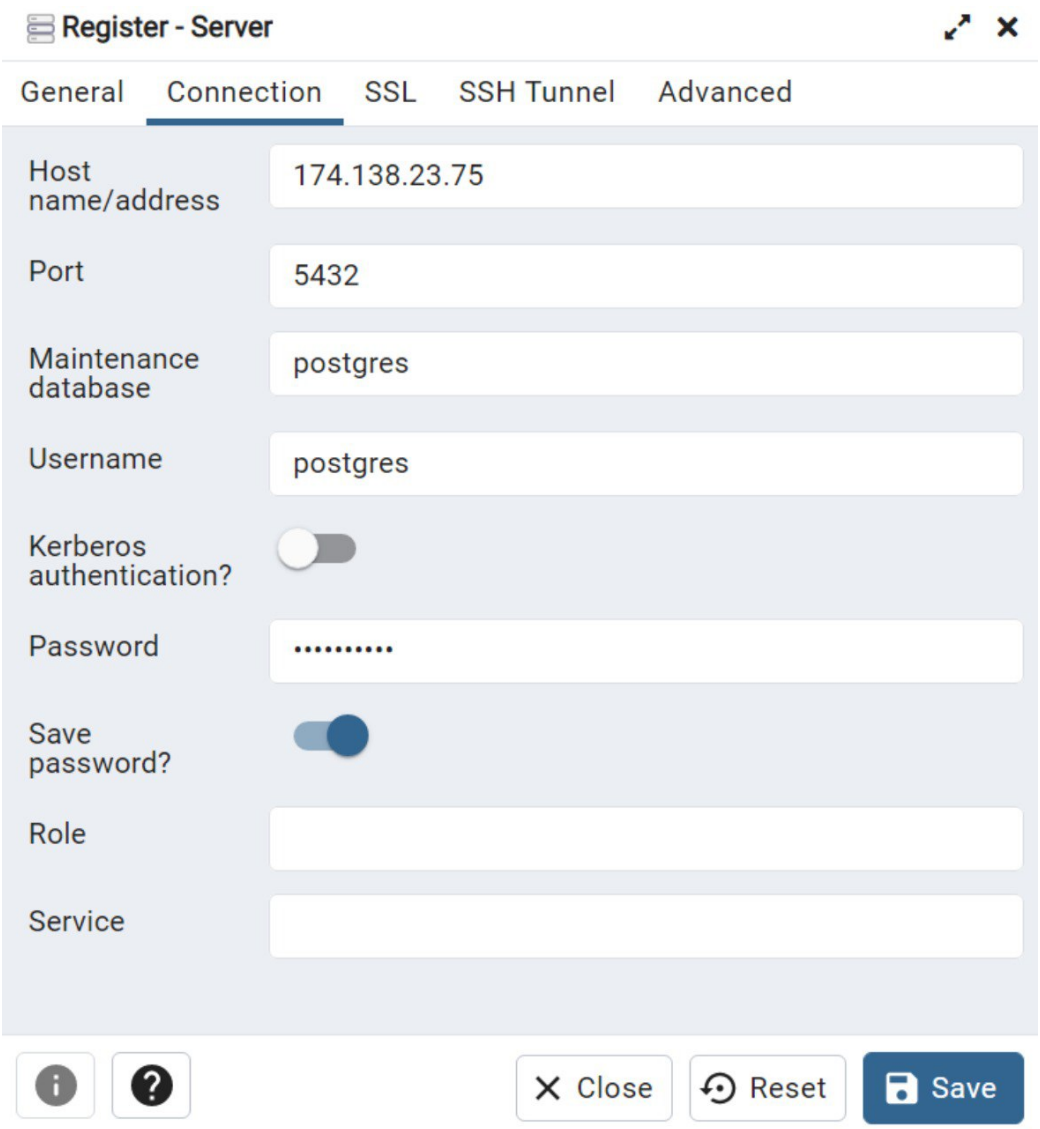
dotnet add package Newtonsoft.Json

2. Search pgAdmin at the start menu and open the application

3. Add a new server

4. Key in the following details:



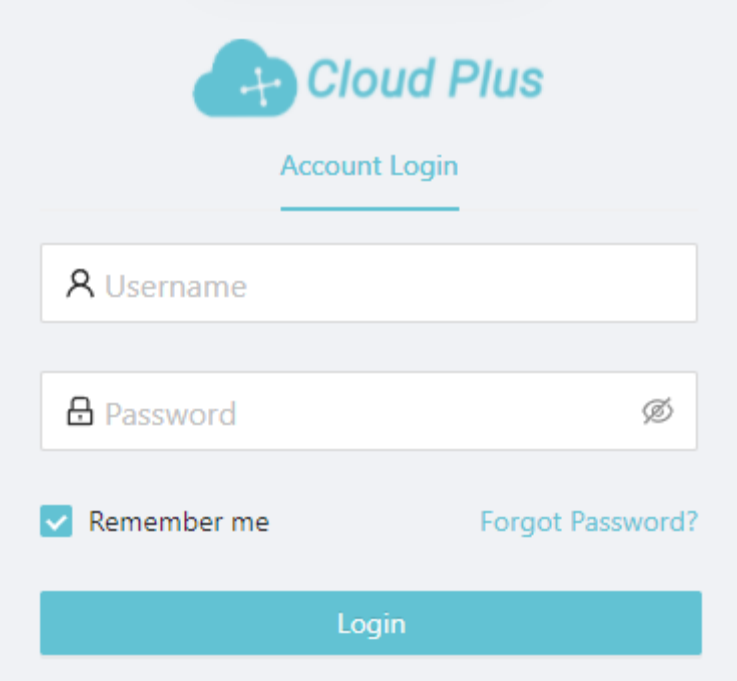


5. Key the following into the command prompt:

   dotnet build

   dotnet run

At the website, you should be able to do something like the following:



Refer to the user manual for more information the details to login, have fun! :)

7.1.5.2 User Document

SCMS E-Form System User Manual

A. User Manual for Managers

[Login]

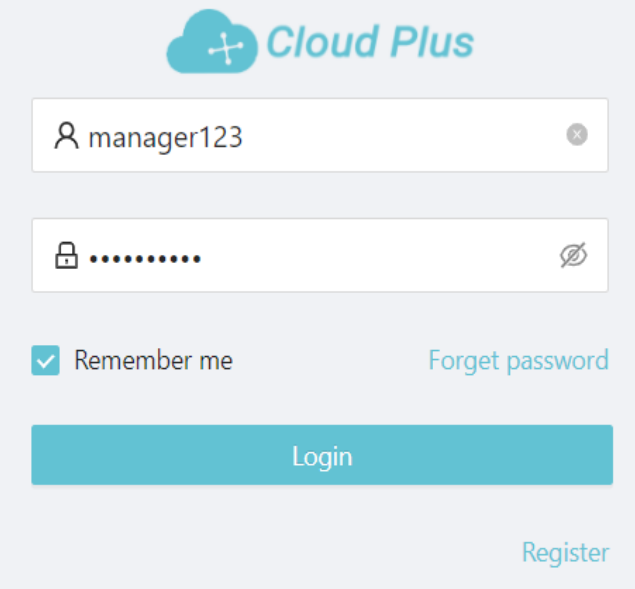
Below shows a series of instructions on how to login.

Enter the username and password.

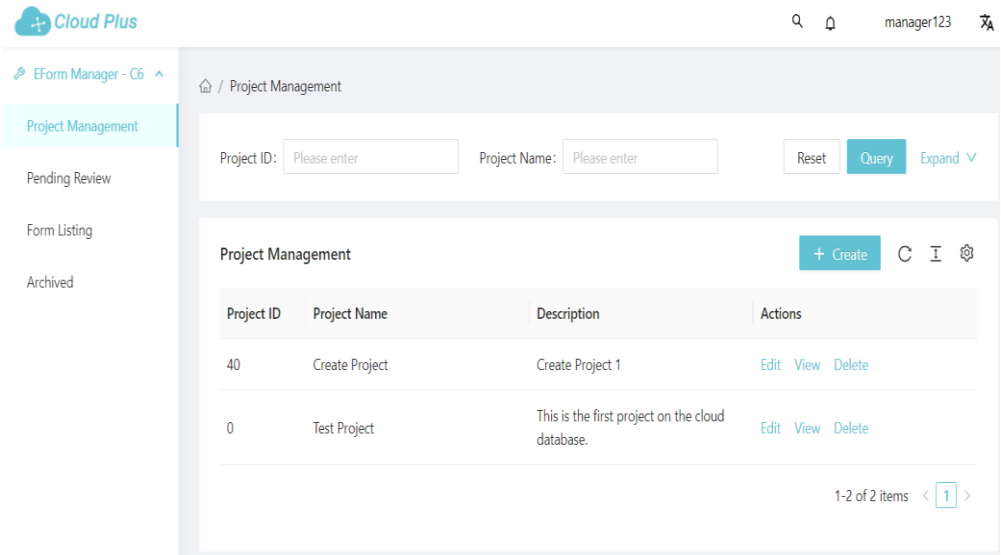
Username: manager123

Password: ant.design

Click on the Login button to login.



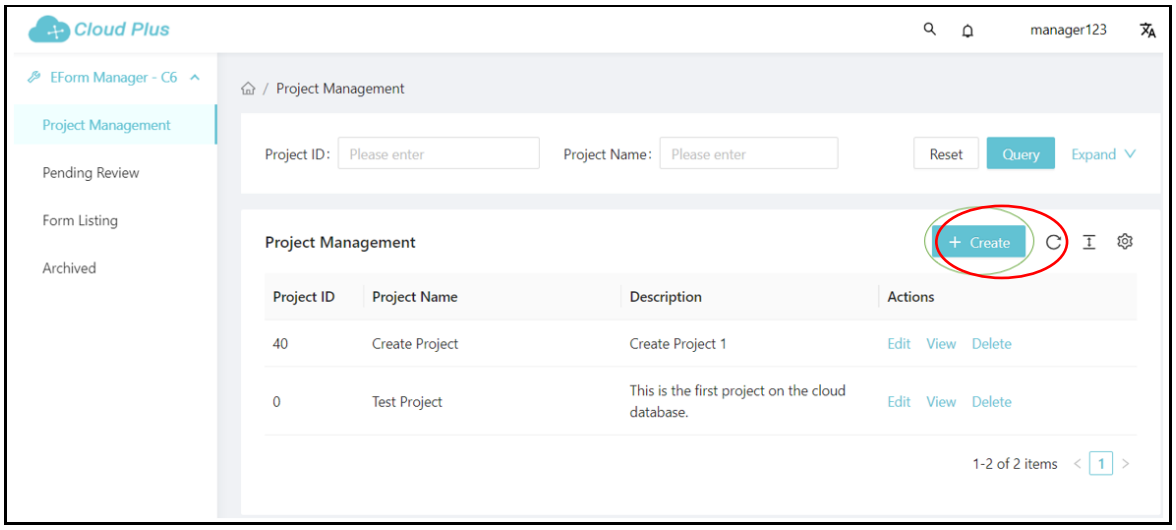
Users will be redirected to the **Project Management** page below upon successful login.



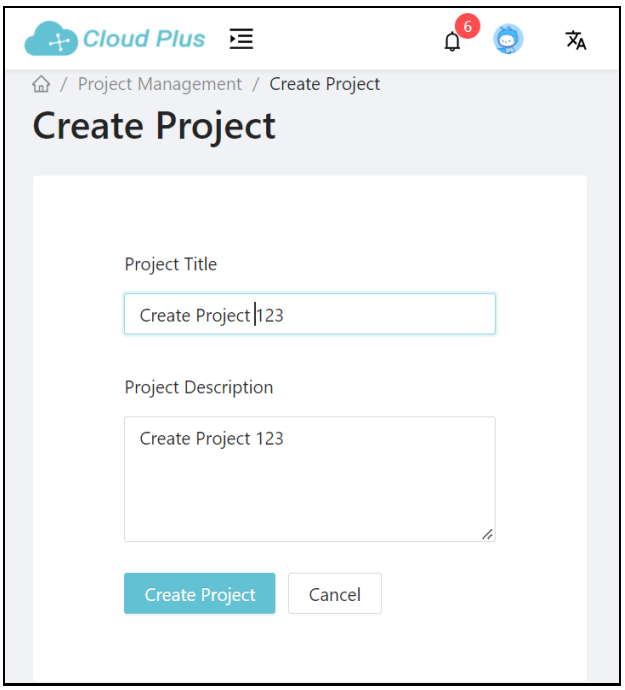
[Create Project]

Below shows a series of instructions on how to create a new project.

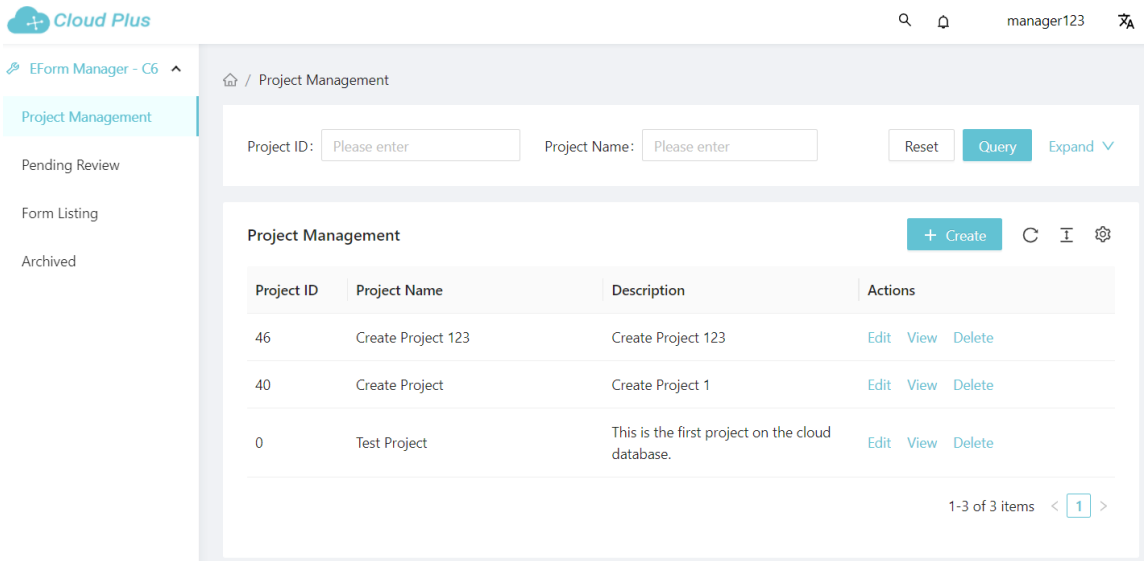
At the **Project Management** page, user can create a new project by clicking the '**+ Create**' button.



Upon clicking the '**+ Create**' button, the user will be redirected to the **Create Project** page and can enter the new project's title and description. Click on '**Create Project**' to confirm the creation of the new project.



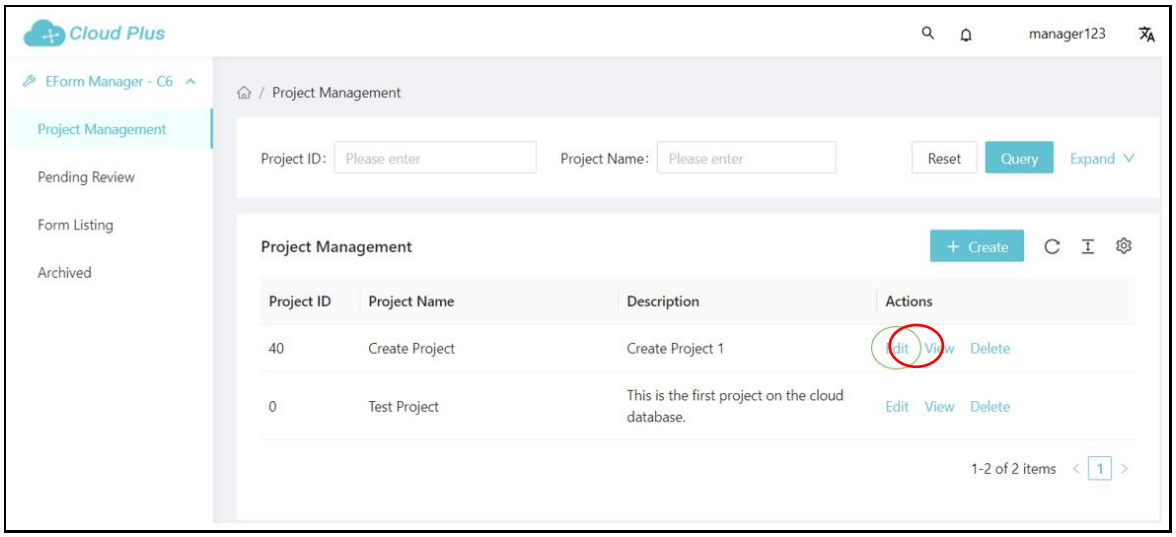
Upon clicking '**Create Project**', the project will be created and appear on the **Project Management** section.



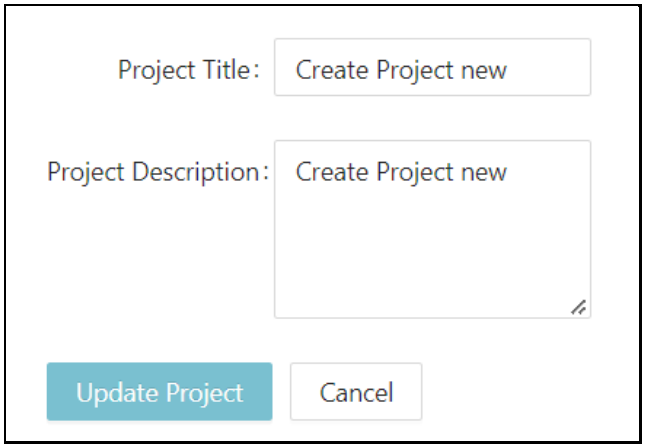
[Edit Project]

Below shows a series of instructions on how to create a new project.

At the **Project Management** section, users can edit and view the details of projects. Clicking the '**Edit**' button will allow the user to edit the project details.



Users may edit the project details, such as title and description accordingly.

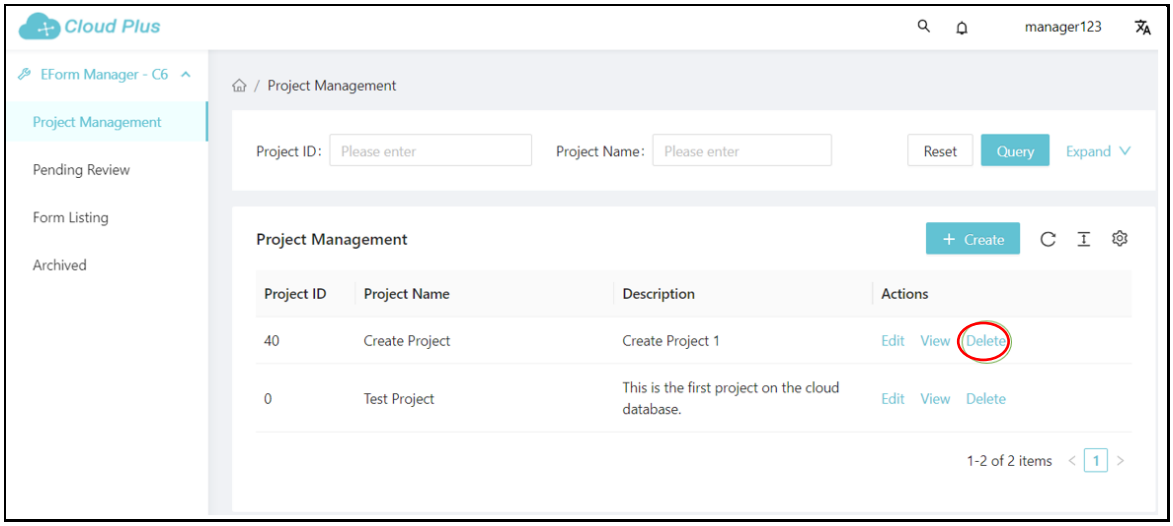


Clicking on the '**Update Project**' button will update the changes made and the user will be redirected to the **View Projects** page.

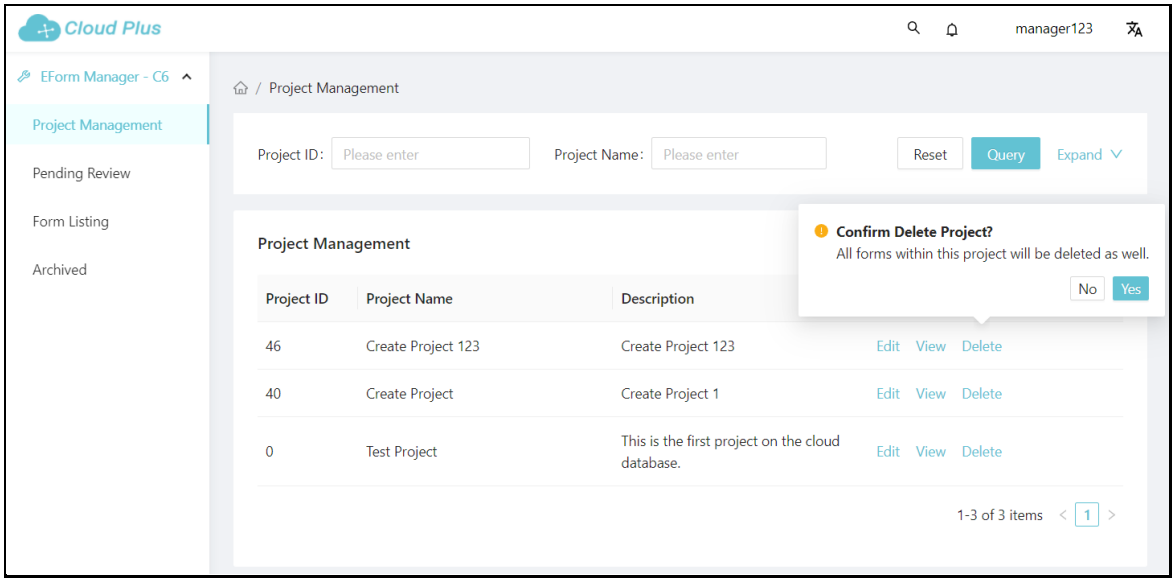
[Delete Project]

Below shows a series of instructions on how to delete projects.

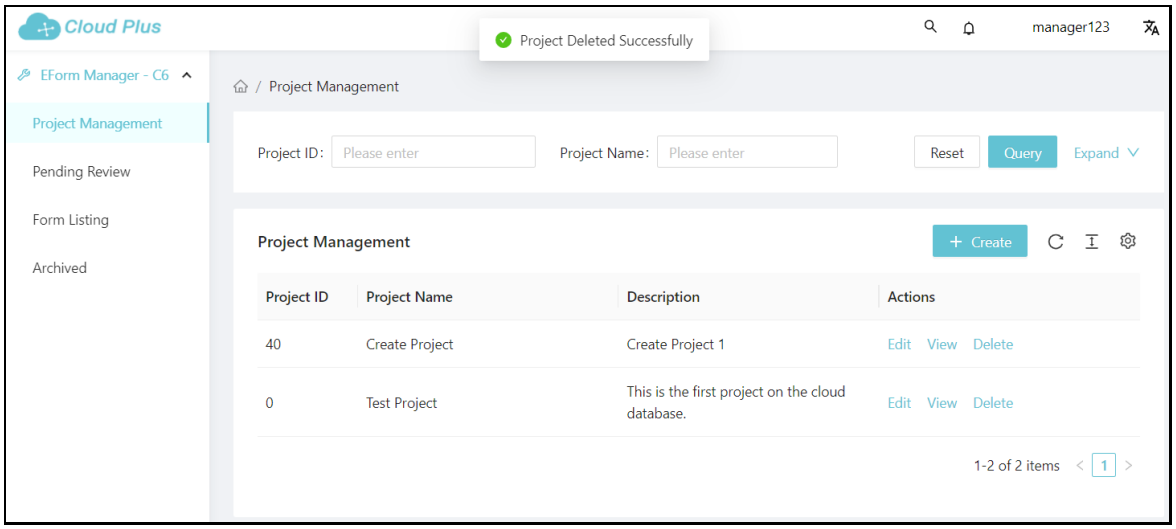
At the **Project Management** section, users can delete a project of their choosing by clicking the '**Delete**' button.



Clicking '**Delete**' will prompt a confirmation for the deletion of the project.



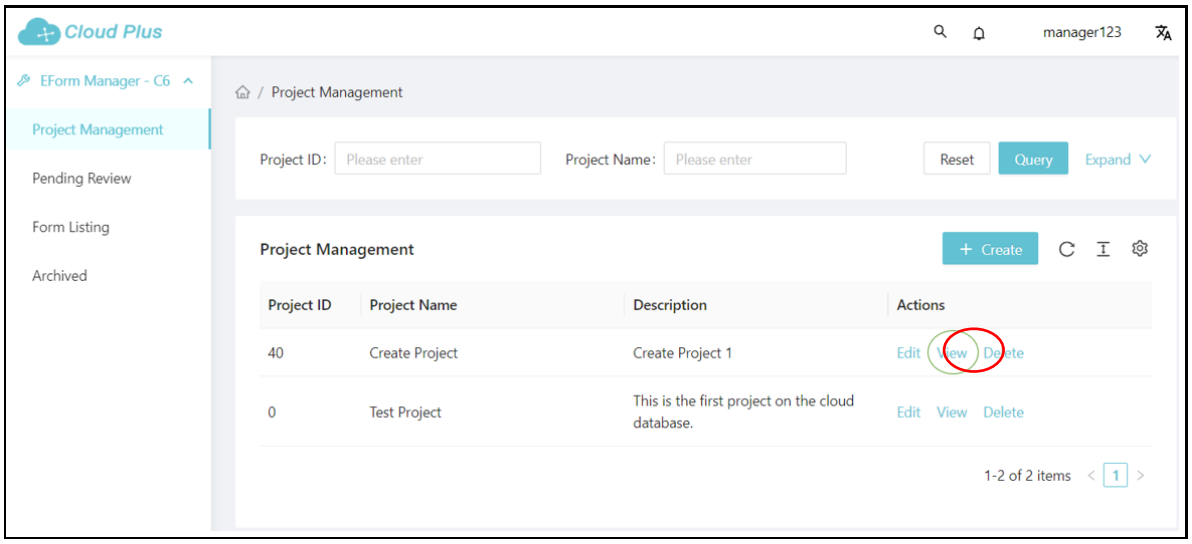
Upon clicking yes, the project and the forms in it will be deleted. An alert '**Project Deleted Successfully**' will appear as confirmation.



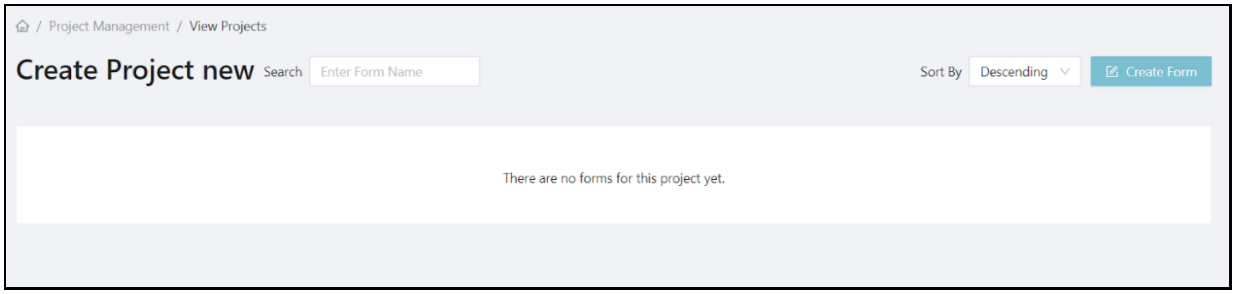
[Create Form]

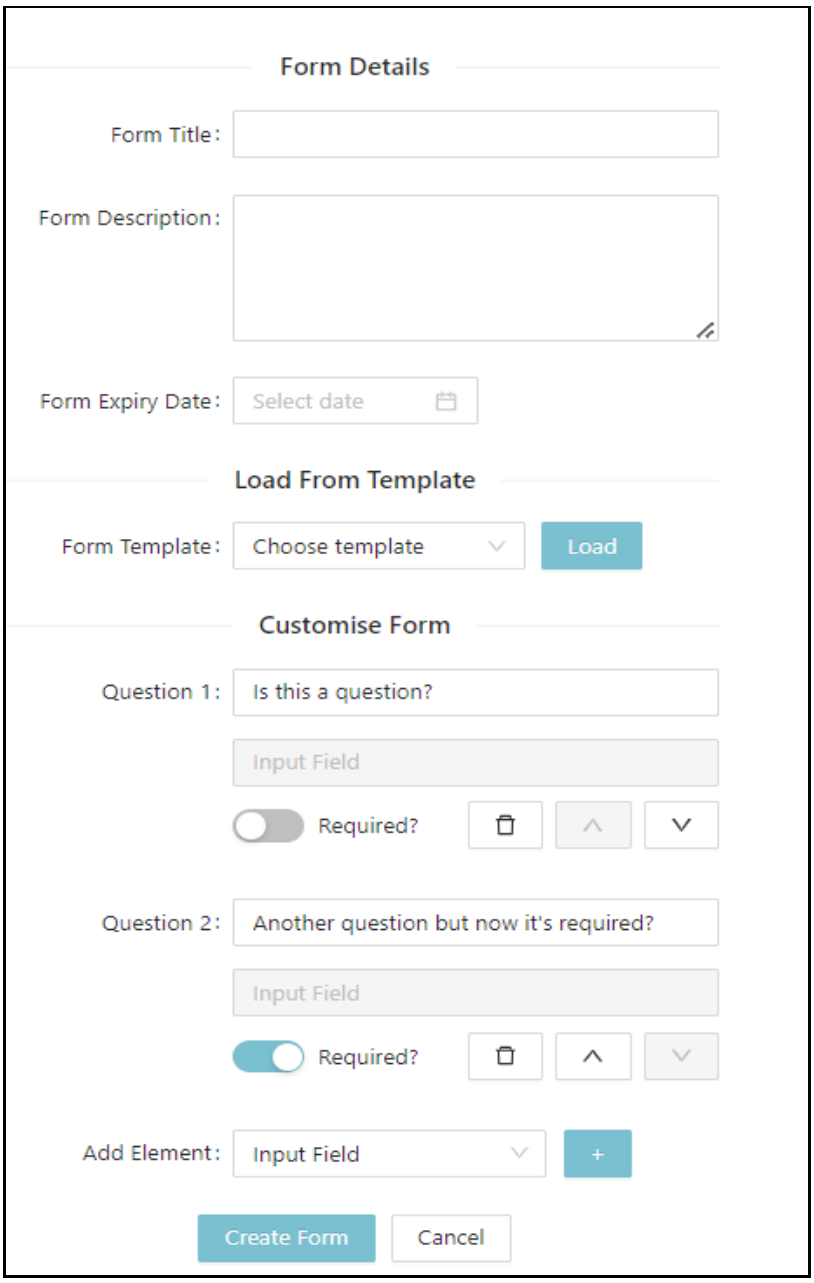
Below shows a series of instructions on how to create forms in a project.

To create forms for a project, users are required to navigate to the project's **View Projects** page by clicking the '**View**' button.

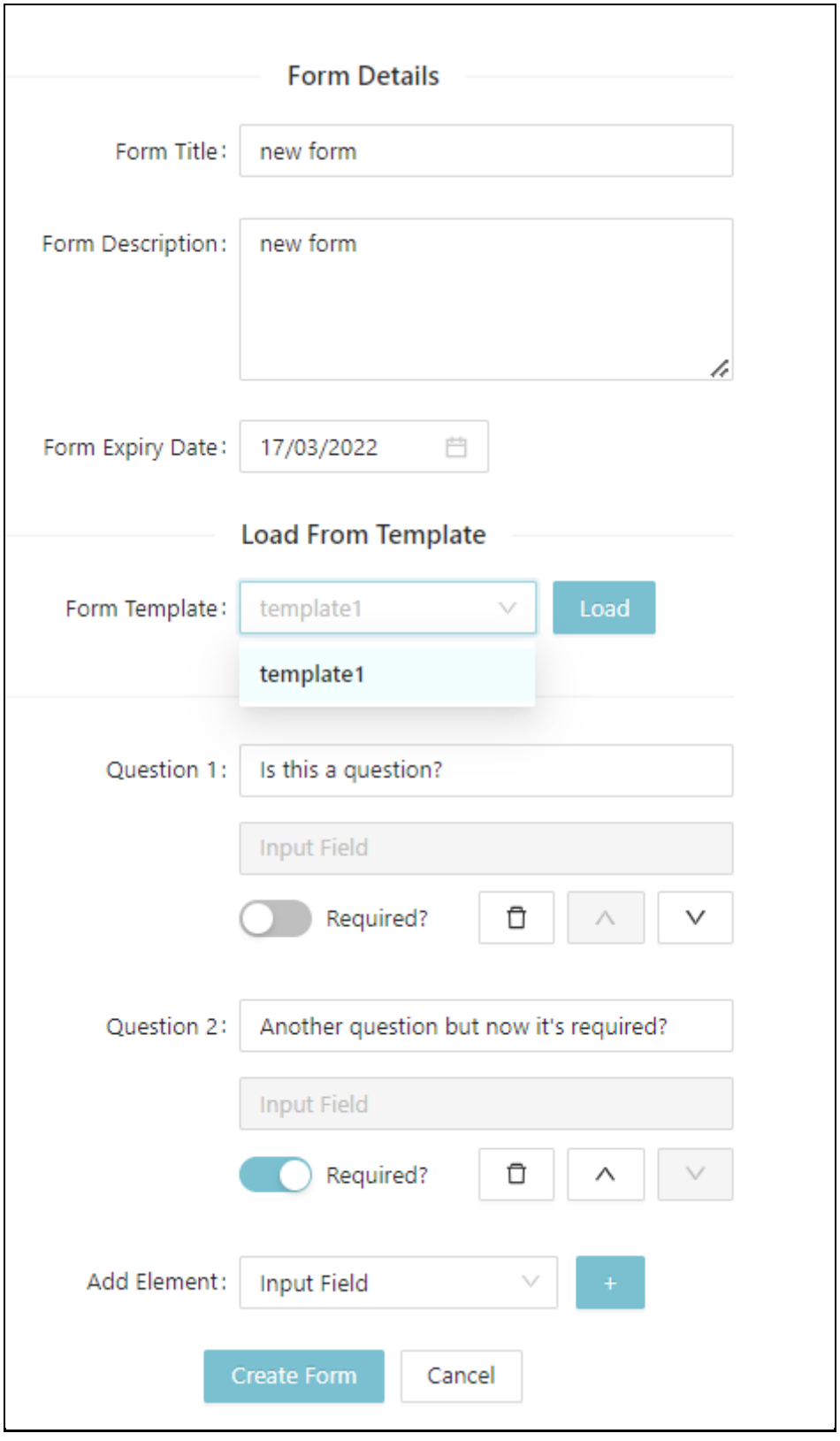


Clicking on the '**Create Forms**' button will redirect the manager to create a new form for the project.



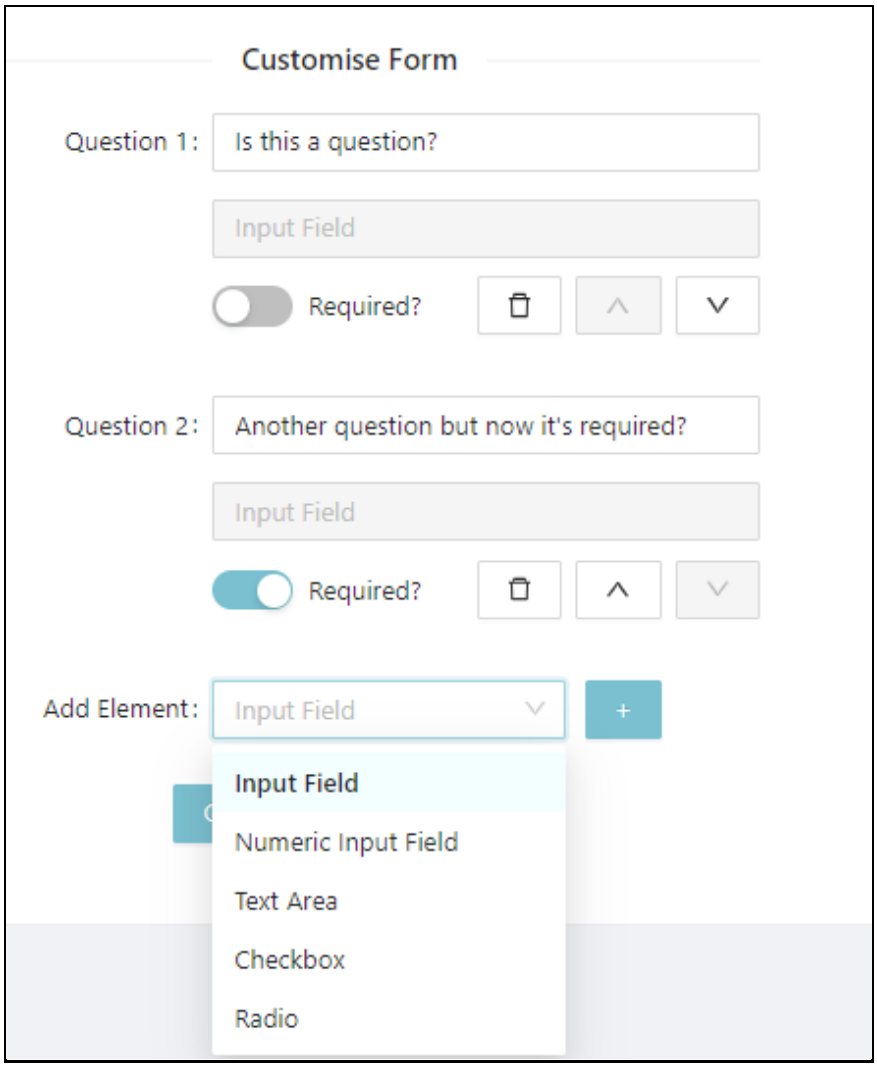


The user can create a new form by filling in the form details before selecting a form template and clicking the '**Load**' button.

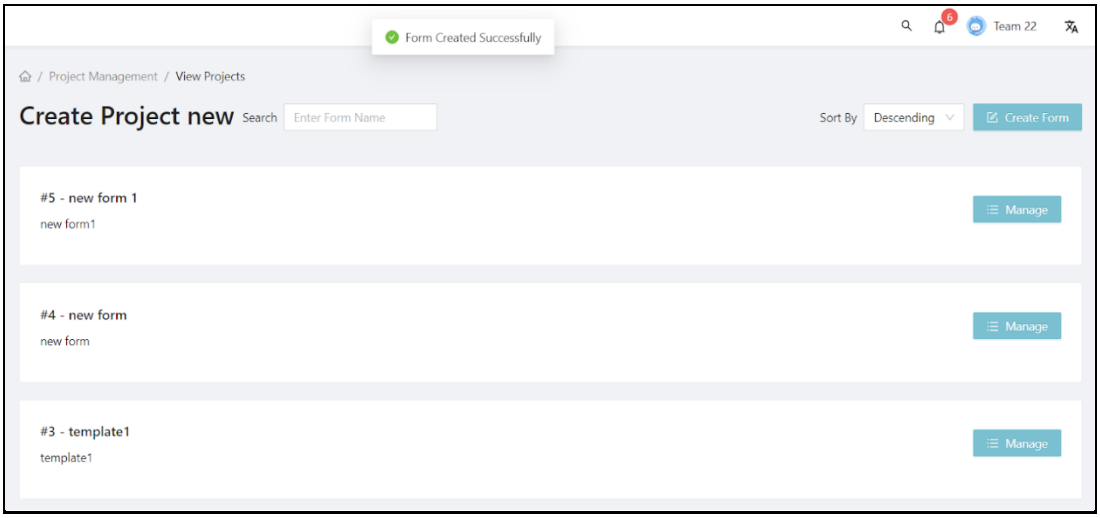


Upon loading the templates successfully, the template will be applied to the form and an alert of '**Template Loaded**' will appear as confirmation.

Alternatively, users can create a new form by making use of the form builder and select the element to be added and click on the '**+**' button.



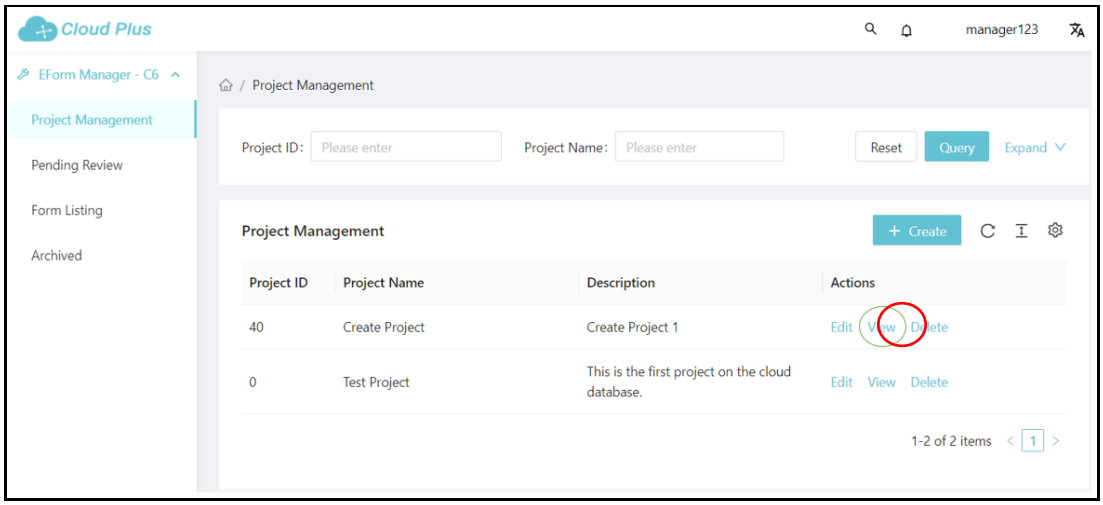
Clicking on the '**Create Form**' button will create the new form. An alert of '**Form Created Successfully**' will appear as confirmation.



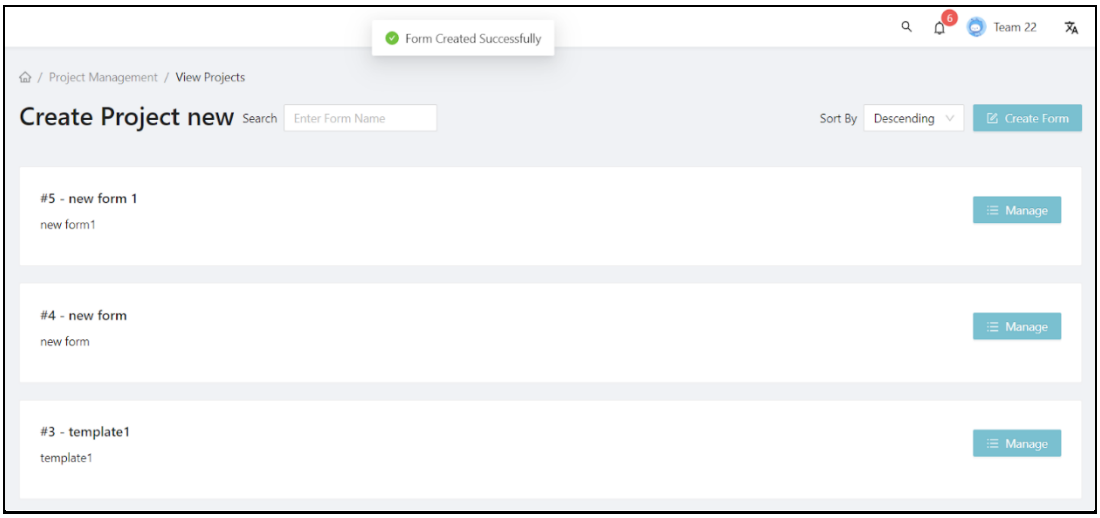
[Manager Forms]

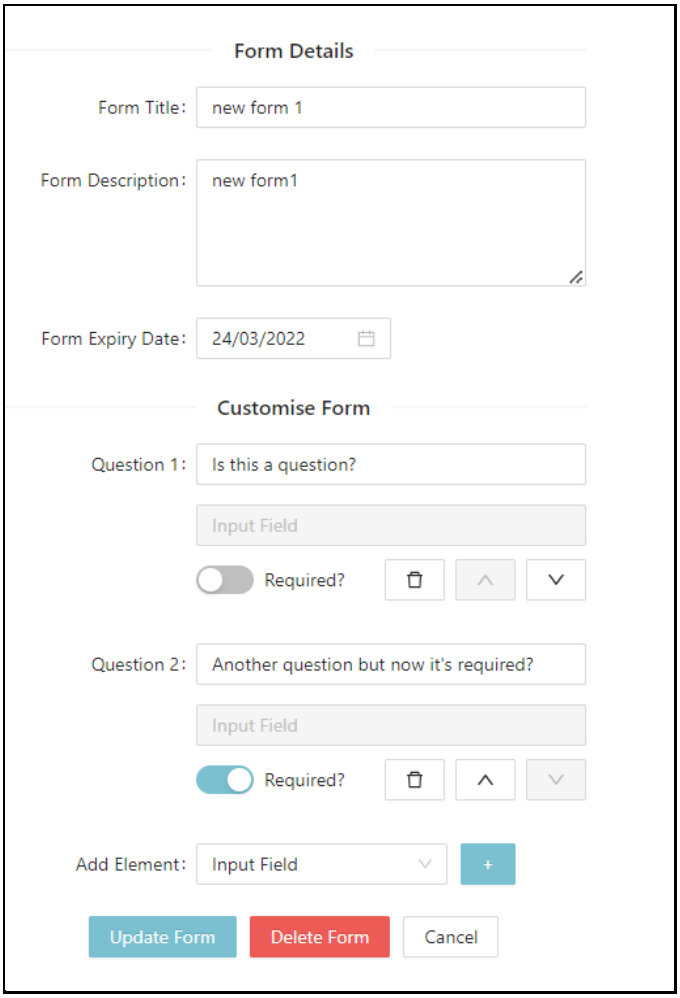
Below shows a series of instructions on how to edit forms in a project.

To edit forms for a project, users are required to navigate to the project's **View Projects** page by clicking the '**View**' button.

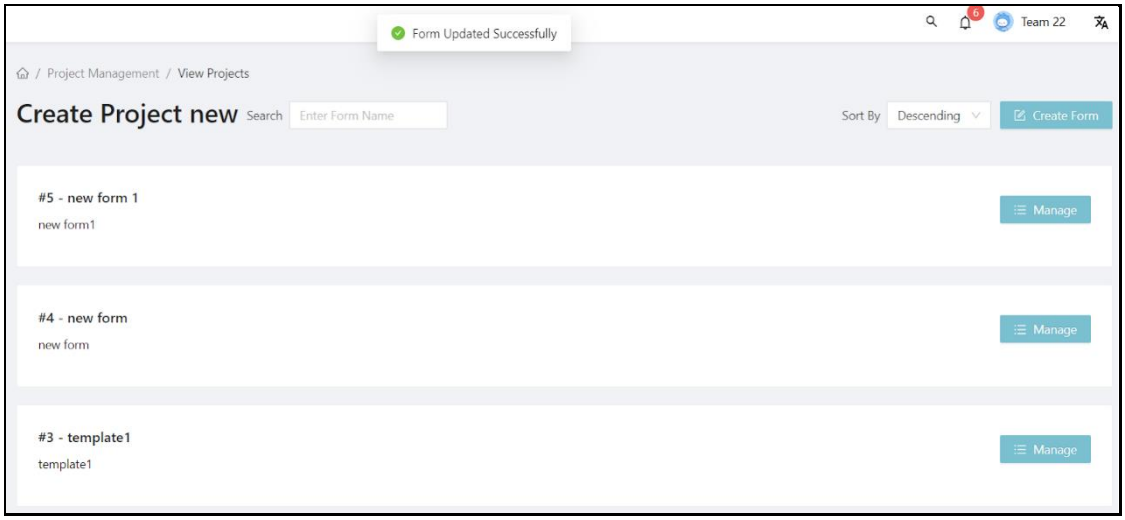


Click on the '**Manage**' button will allow the manager to edit the form.

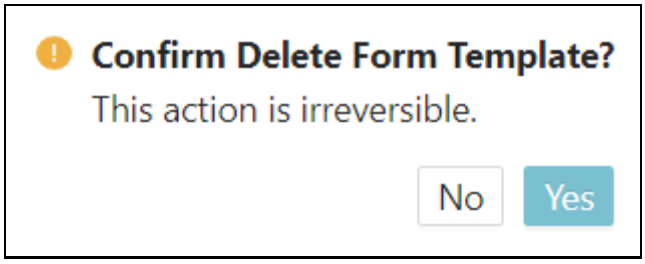




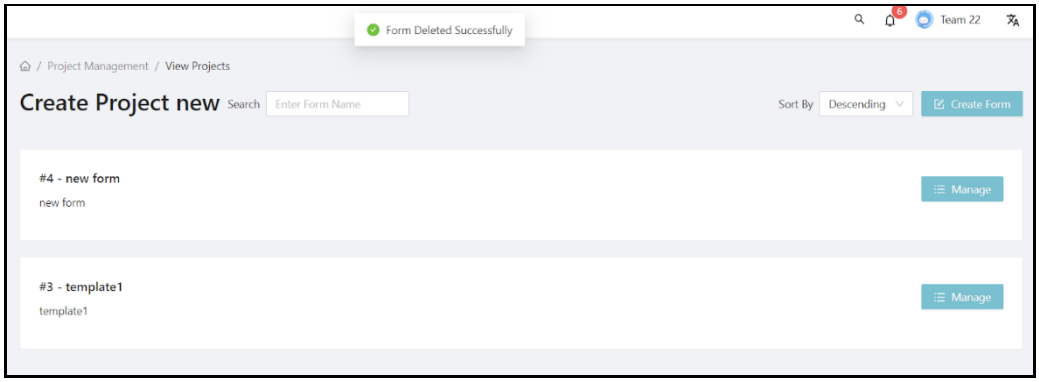
After making changes to the form, clicking on the '**Update Form**' button will update the form. An alert of '**Form Updated Successfully**' will appear as confirmation.



Alternatively, users may delete the form by clicking the '**Delete Form**' button. A confirmation prompt will appear.



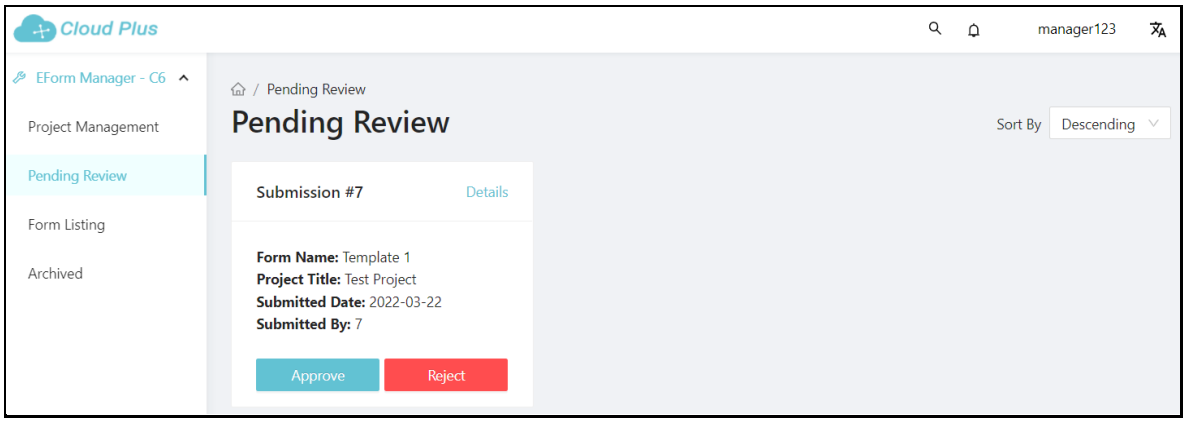
Clicking the '**Yes**' button will delete the form. An alert of '**Form Deleted Successfully**' will appear as confirmation.



[Form Review]

Below shows a series of instructions on how to review form submissions from workers.

To review form submissions, the user will be required to navigate to the **Pending Review** page by clicking the '**Pending Review**' button on the navigation bar. The **Pending Review** page shows all the forms submitted by the workers that are still pending in status.



To approve a form submission, click on the '**Approve**' button and an alert of '**Submission Approved**' will appear as confirmation

To reject a form submission, click on the '**Reject**' button.

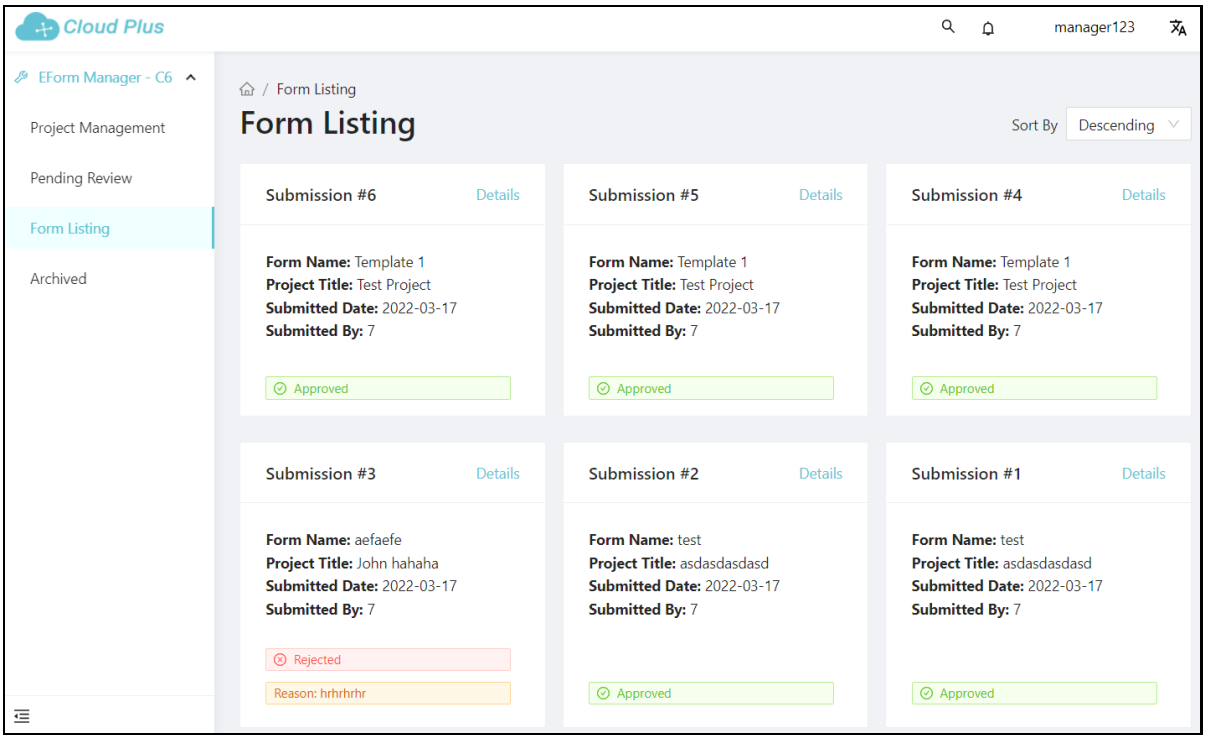
The user will be required to enter the reason for rejection.

After entering a reason, clicking **Confirm Rejection** will reject the form and an alert of '**Submission Rejected**' will appear as confirmation.

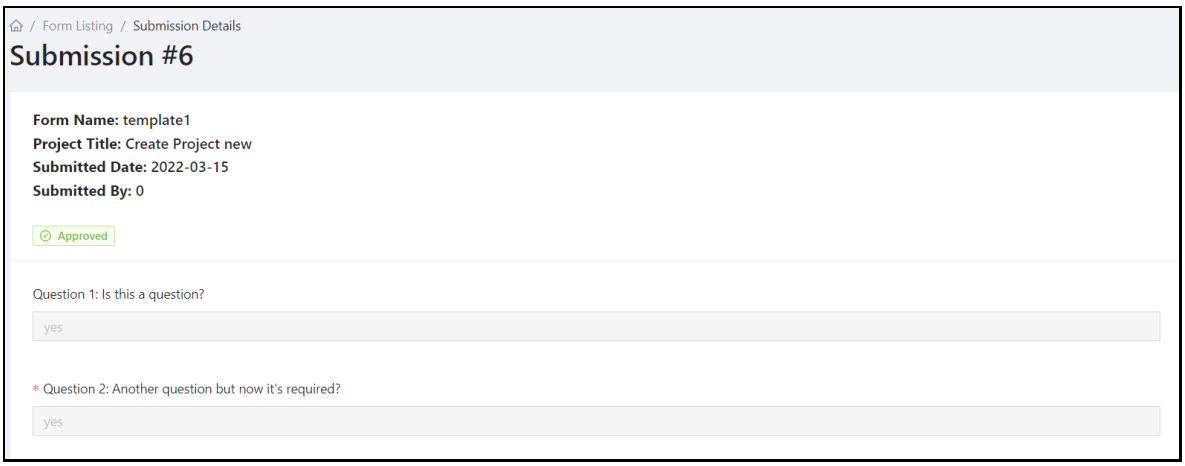
[Form Listing]

Below shows a series of instructions on how to view the details of every form approved or rejected by managers.

To view all the forms that the managers had approved or rejected, the user will be required to navigate to the **Form Listing** page by clicking '**Form Listing**' on the navigation bar.



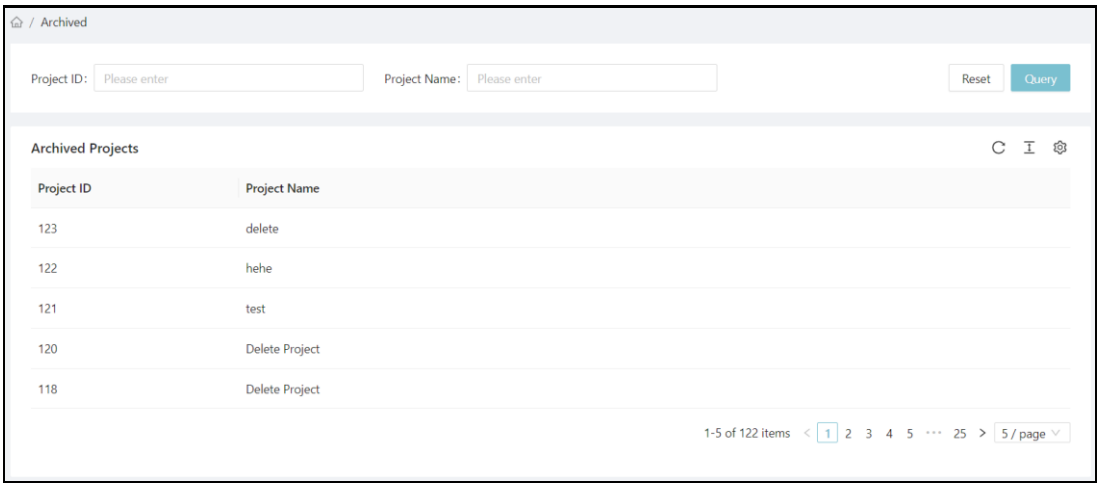
Clicking '**Details**' will show the details of the form.

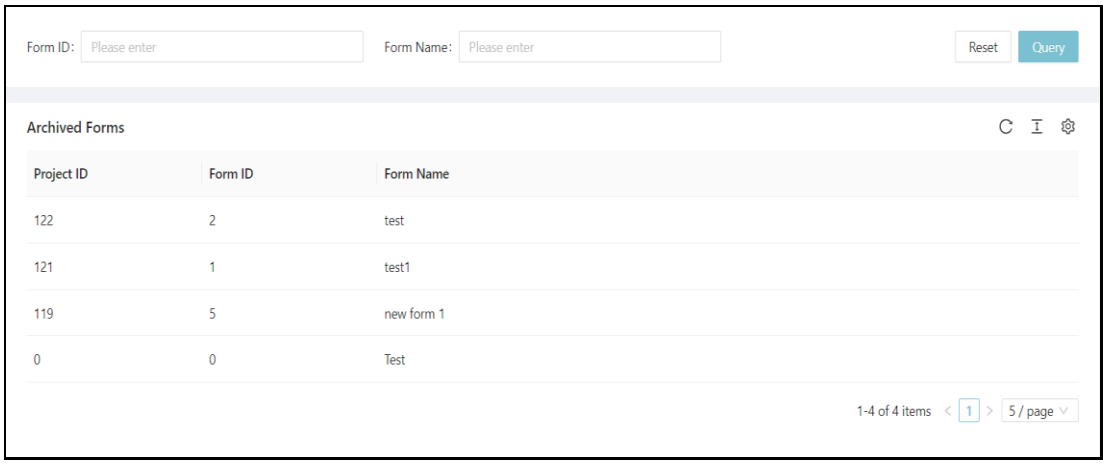


[Archived Section]

Below shows a series of instructions on how to view the deleted projects and forms.

To view all the deleted projects and forms, the user will be required to navigate to the **Archived** section by clicking '**Archived**' from the navigation bar.





B. User Manual for Workers

[Login]

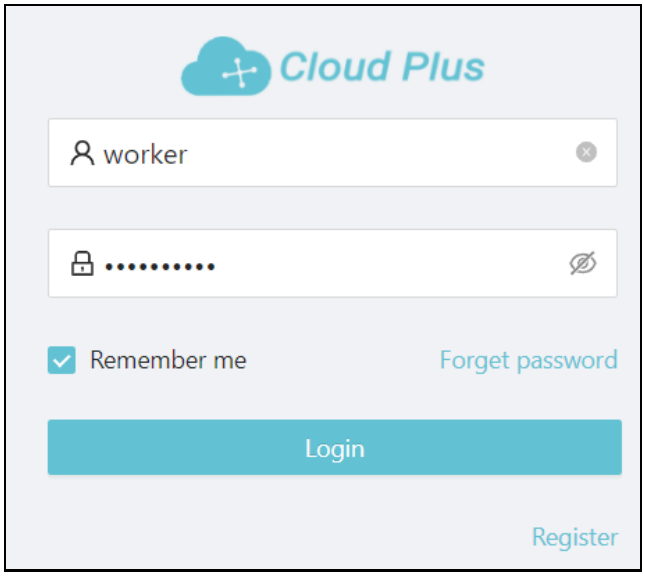
Below shows a series of instructions on how to login.

Enter the username and password.

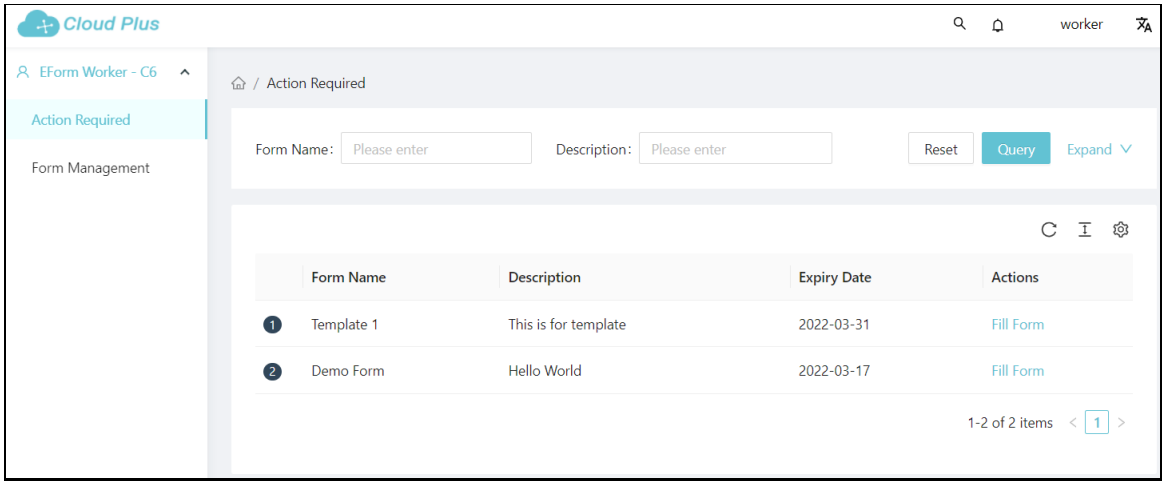
Username: worker

Password: ant.design

Click on the Login button to login.



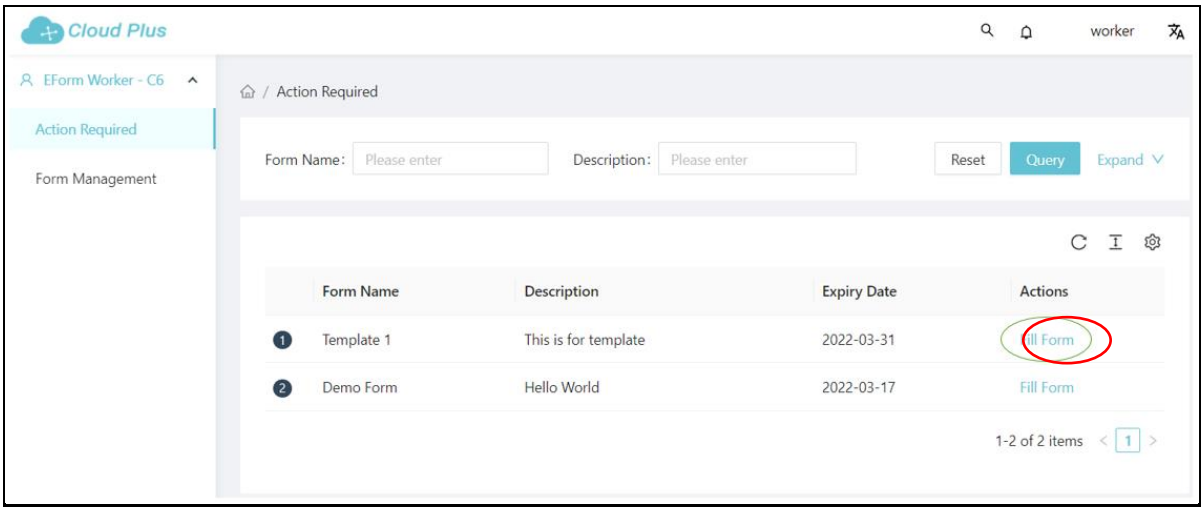
Users will be redirected to the **Action Required** page below upon successful login.



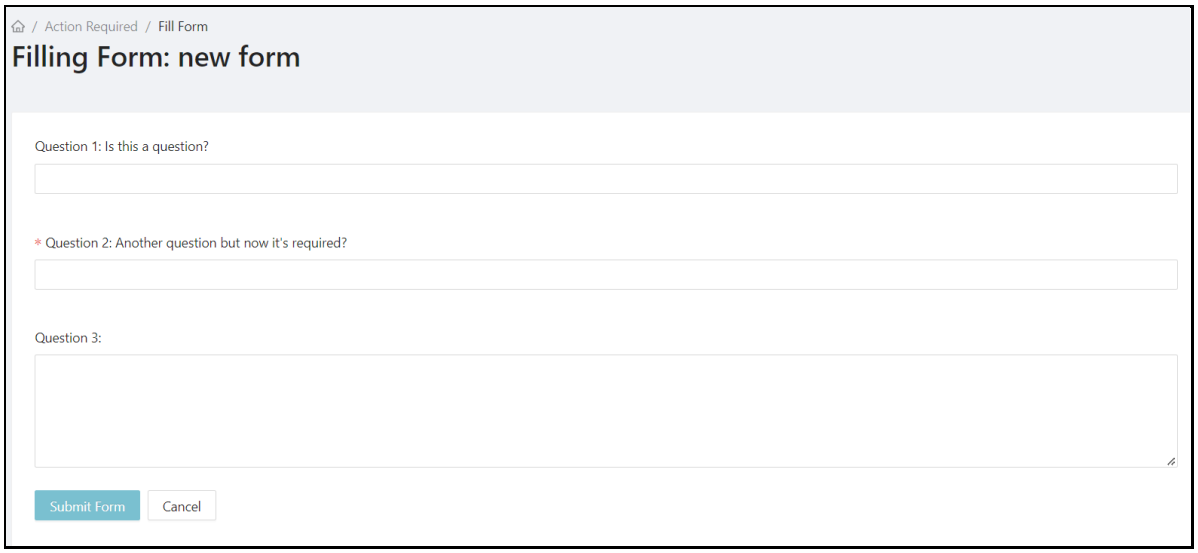
[Fill Form]

Below shows a series of instructions on how to fill up forms as workers.

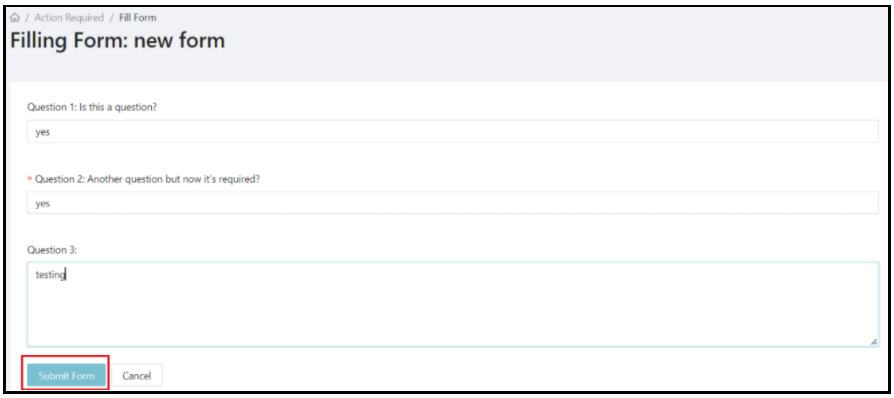
To start filling a form, users can click the '**Fill Form**' button.



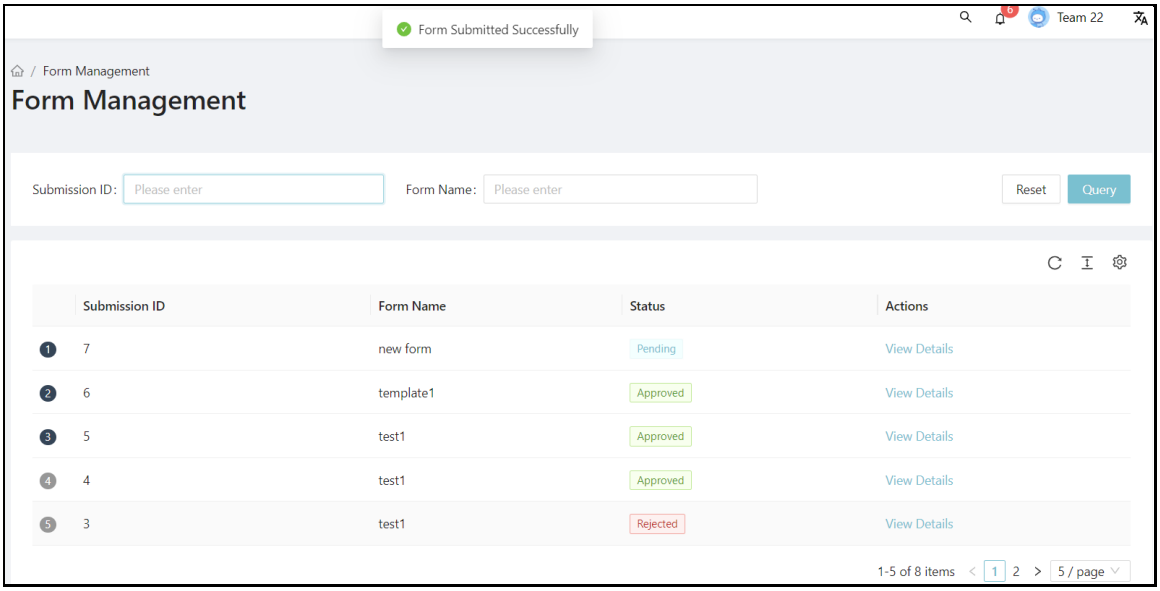
The user will be redirected to fill up the form.



After filling up the form, click on '**Submit Form**' to submit the form.



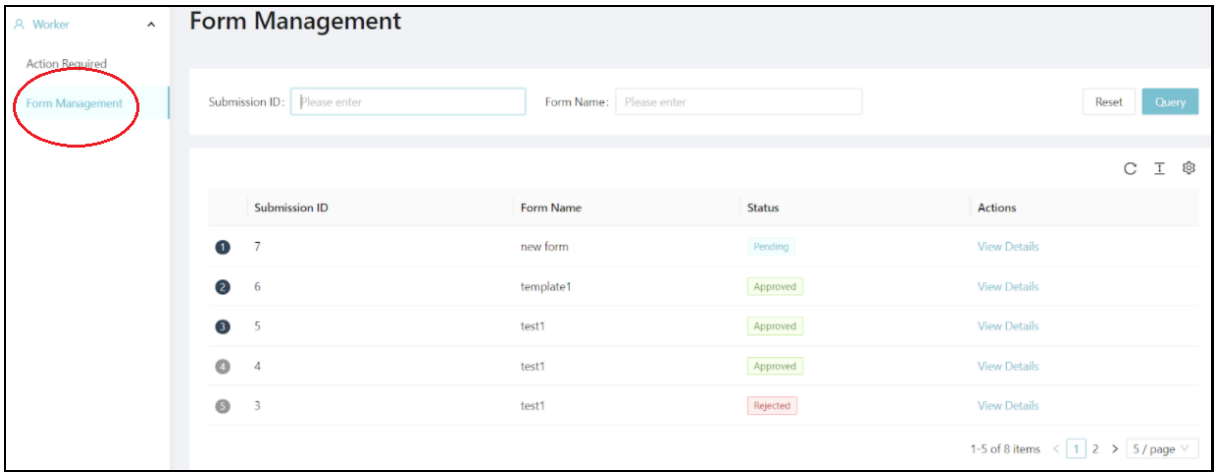
After the form submission, the user will be redirected to the **Form Management** page. An alert of '**Form Submitted Successfully**' will appear as confirmation.



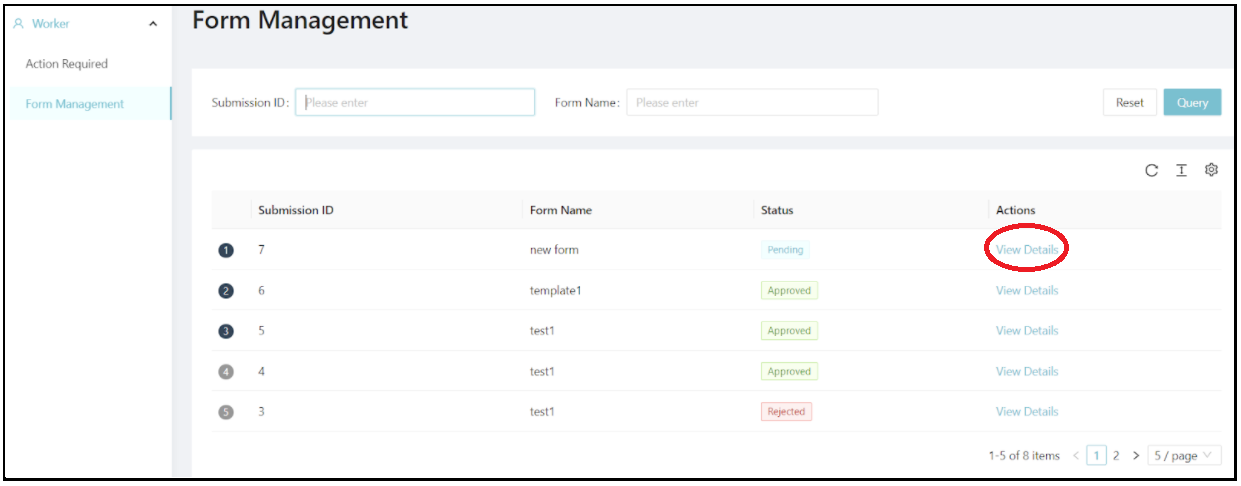
[View Status of Submitted Forms]

Below shows a series of instructions on how to view submitted forms as workers.

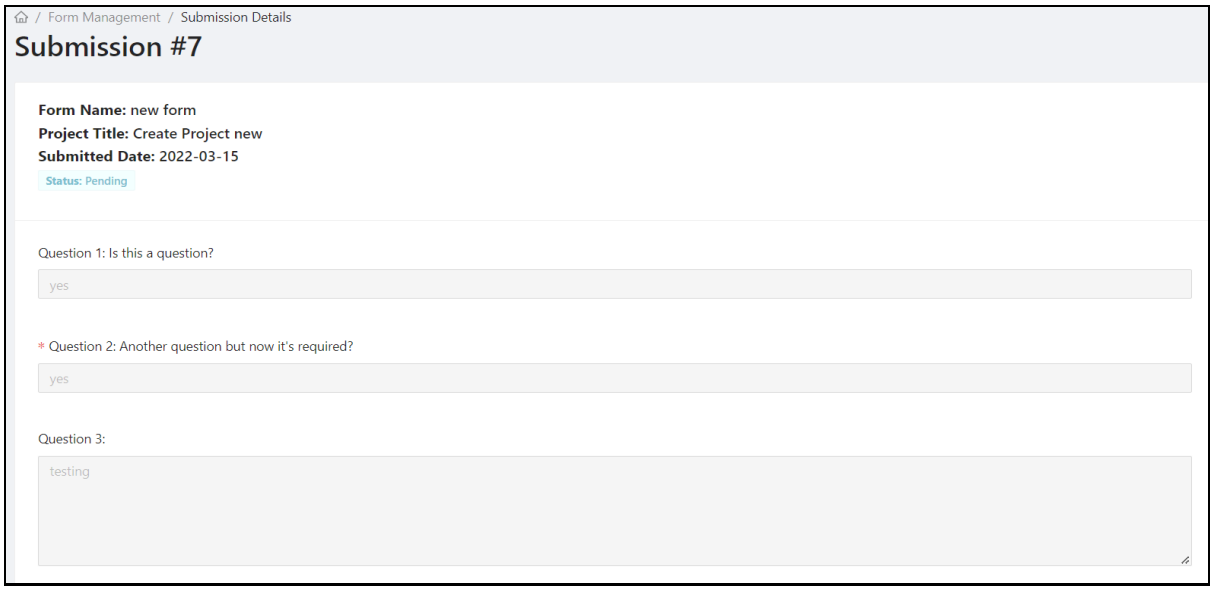
To view the status of their forms, the user can navigate to the **Form Management** page from the navigation bar.



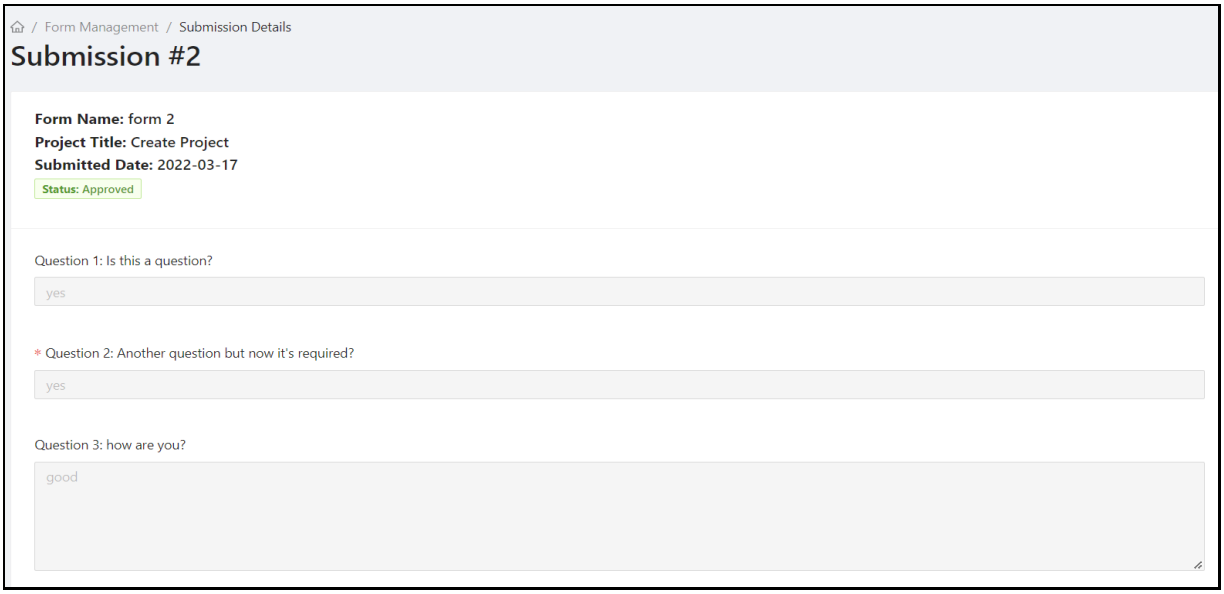
To view the details of the form submission, click on **View Details**.



An example of a form with the status '**Pending**'.

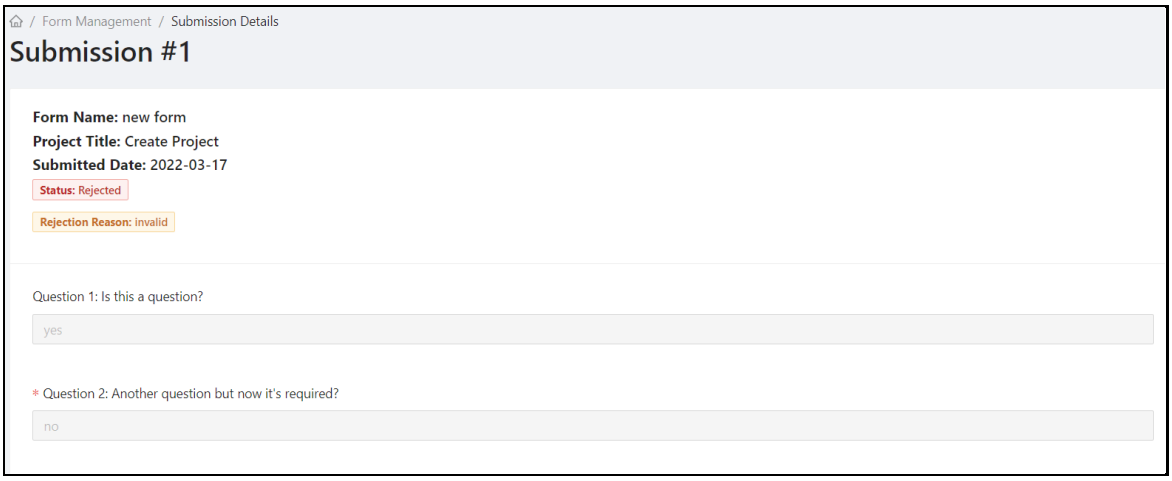


An example of a form with the status of '**Approved**'.



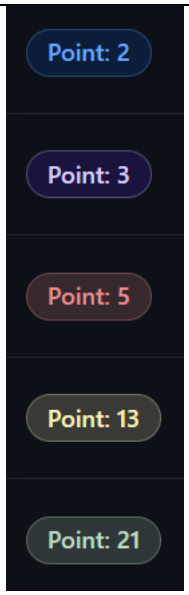
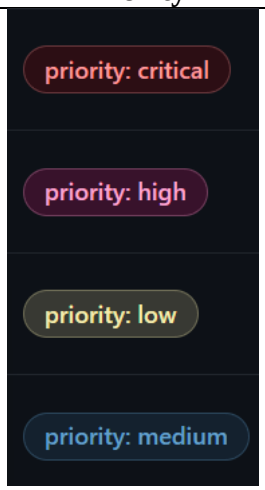
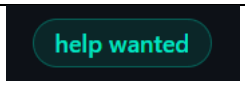
An example of a form with the status of '**Rejected**'.

The reason for rejection status will be shown to the user.



7.2 Software Process

7.2.1 GitHub Label Table

Points	Priority	Role	Miscellaneous
			
Referring to efforts required to complete the task using Fibonacci agile estimation (2,3,5,8…)	It defines the order of issues which should be resolved first, assisting the team to have a better view of which tasks are important.	Role defines the issue in which the feature is associated to the role in the E-Form system. This label makes it clearer for the team member to identify what issue is it referring to.	This label indicates when additional manpower is required to tackle the issue.

7.2.2 Example of Additional Questions for the Customer

Discussion (DO NOT SHOW THIS SLIDE)

Frontend

1. Are there any feedback regarding the looks of the E-Form system? Are there any changes you would like to see? Like font size ok, need bigger?
2. Is this what you meant by the table listing (pro component) when you suggest it in the previous meeting?
3. How do you feel about the flow of the pages? Are you okay with the current structure and naming - as confirmation we will send email to see if any pages name you want to change
4. Form template: How many form templates can be created? Is there a limit? Because we would like to set maximum amount of form templates users can create. // Ask for feedback on form templates(describe to them how it works), suggest a form template management module (+ new template, within this create page will have form builder but to build the template)
5. Form Builder: What is the difference between Checkbox and 'MultipleChoice'
6. Pro Table add to all pages? (form review/history, in worker view forms are all using card etc)

Backend

1. Kenneth: Testing of Integration: Is it possible to have a cloud server set up earlier on for us to have a centralised database to test our backend against?