Inspection & evidence control system

By

Meet Patel (16MCA012)

Asha Patel (15IT073)

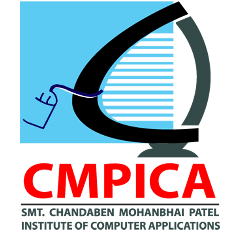
Under Guidance

of

Internal Guide

Dr. Pranav Vyas

Submitted to



Smt. Chandaben Mohanbhai Patel Institute of Computer Applications

CHARUSAT

Changa

April 2019

charusatlogo

CHAROTAR UNIVERISTY OF SCIENCE & TECHNOLOGY

Changa

**Acknowledgement**

Knowledge in itself is a continuous process. At this moment of our substantial enhancement, we rarely find words to express our gratitude towards those who were constantly involved with us.

The completion of any inter disciplinary project depends upon coordination, cooperation and combined efforts of several resources of knowledge, creativity, skill, energy and time. The work being accomplished now, we feel our most sincere urge to recall and knowledge through these lines, trying our best to give full credit wherever it deserves.

We would like to thank our project guide **Dr. Pranav Vyas** and Dean & Principal **Dr. Atul Patel** who advised and gave us moral support through the duration of our project. Without their constant encouragement we could not have been able to achieve what we have.

It’s our good fortune that we had support and well wishes of many. We are thankful to all and those names which have been forgotten to acknowledge here but contributions have not gone unnoticed.

With Sincere Regards,

**Meet Patel – 16MCA012**

**Asha Patel - 15IT073**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. No** |  | **Subject** | **Page Number** |
| **1** |  | **Project Profile** |  |
| **2** |  | **Company Profile** |  |
| **3** |  | **Introduction to tools** |  |
| **4** |  | **System Study** |  |
|  |  | Existing System |  |
|  |  | Proposed System |  |
|  |  | Scope of the Proposed System |  |
|  |  | Aim and Objective of the Proposed System |  |
|  |  | Feasibility Study |  |
|  | **r** | Operational Feasibility |  |
|  |  | Technical Feasibility |  |
|  |  | Economic Feasibility |  |
| **5** |  | **System Analysis** |  |
|  |  | Requirements Specification |  |
|  |  | System Modules |  |
|  |  | Use Case Diagram |  |
|  |  | Activity Diagram |  |
|  |  | Class Diagram |  |
|  |  | Sequence Diagram |  |
| **6** |  | **System Design** |  |
|  |  | Data Dictionary |  |
|  |  | Screen Layouts |  |
| **7** |  | **System Testing** |  |
|  |  | Testing Strategies |  |
| **8** |  | Future Enhancement |  |
| **9** |  | Bibliography/References |  |

**Table of Contents**

Project Profile

* **Project Profile**

**Project Name:** Eviware (Inspection and Evidence control system)

**Type of Application:** Web Application

**Project Description:**

* + Inspection and evidence system specializing in management and controlling the proper flow of documents and task from point of origin to point of distribution
  + Main purpose of this system is to eliminate the paper work to handle all documents. Management of proper service flow with less time and money by automating manual processes with the help of different technologies.
  + The proposed system will feature different modules that ensures the proper flow of documents and related task to all those who are concerned with a job.
  + A system that would be able to manage users with authentication, their documents, verification of team members, forms for task, flow of entities from source to destination with proper validation, assets that are required to perform different task etc.
* **Team Size: 2**
* **Front End: Angular 6**
* **Back End: ASP.NET Core**
* **Database: PostgreSQL**
* **Tools used: Visual Studio 2017, Visual Studio Code**

Company Profile

* 
* Promact is an excellence driven company with a passion for technology where people love what they do. We focus on delivering values, creating impact & producing exciting experiences.
* A blend of a good idea, creativity, simplicity, robust technical implementation and an exceptional team is nothing less than an innovation. We pride ourselves for that.
* Fun workplace, technology leader, target excellence - these three pillars of our vision are the key foundation of Promact. Our vision makes us what we are, an exciting workplace, a technologically leading company & an exceptional service provider. We love start-ups! Our team includes co-founders from other start-ups. We strive and encourage the culture of start-ups. We have helped start-ups reach from 100 users to million users.

At Promact we provide all technology solutions including Web Application, Mobile Applications & Devops. We constantly invest in new and upcoming technologies to provide our clients as well as to employees, an edge in the ever so fast-moving market. We use latest technologies such as Android, iOS, Ionic, Phonegap fort mobile application development; Angular, NodeJS, ASP.NET, PHP, Python for delivering intuitive web applications. We also use Docker, Amazon AWS, Microsoft Azure, Apache for Devops.

Introduction to tools

* **Introduction to Tools**
* **Front End Tool: Angular 6**

Angular is a JavaScript-based open-source front-end web framework mainly maintained by Google and by a community of individuals and corporations to address many of the challenges encountered in developing single-page applications.

Angular is a platform that makes it easy to build applications with the web. Angular combines declarative templates, dependency injection, end to end tooling, and integrated best practices to solve development challenges. Angular empowers developers to build applications that live on the web, mobile, or the desktop.

Angular 6 is a JavaScript framework for building web applications and apps in JavaScript, html, and TypeScript, which is a superset of JavaScript. Angular provides built-in features for animation, http service, and materials which in turn has features such as auto-complete, navigation, toolbar, menus, etc. The code is written in TypeScript, which compiles to JavaScript and displays the same in the browser.

* **Back End Tool: ASP.NET Core**

ASP.NET is a web development platform, which provides a programming model, a comprehensive software infrastructure and various services required to build up robust web applications for PC, as well as mobile devices.

ASP.NET works on top of the HTTP protocol, and uses the HTTP commands and policies to set a browser-to-server bilateral communication and cooperation.

ASP.NET is a part of Microsoft .Net platform. ASP.NET applications are compiled codes, written using the extensible and reusable components or objects present in .Net framework. These codes can use the entire hierarchy of classes in .Net framework.

The ASP.NET application codes can be written in any of the following languages:

* C#
* Visual Basic.Net
* Jscript
* J#

ASP.NET is used to produce interactive, data-driven web applications over the internet. It consists of a large number of controls such as text boxes, buttons, and labels for assembling, configuring, and manipulating code to create HTML pages. With ASP.NET Core, you can:

* Build web apps and services, [IoT](https://www.microsoft.com/internet-of-things/) apps, and mobile backends.
* Use your favorite development tools on Windows, macOS, and Linux.
* Deploy to the cloud or on-premises.
* Run on [.NET Core or .NET Framework](https://docs.microsoft.com/en-us/dotnet/articles/standard/choosing-core-framework-server).
* **Database: PostgreSQL**
* PostgreSQL is a powerful, open source object-relational database system that uses and extends the SQL language combined with many features that safely store and scale the most complicated data workloads. The origins of PostgreSQL date back to 1986 as part of the [POSTGRES](https://www.postgresql.org/docs/current/history.html) project at the University of California at Berkeley and has more than 30 years of active development on the core platform.
* PostgreSQL has earned a strong reputation for its proven architecture, reliability, data integrity, robust feature set, extensibility, and the dedication of the open source community behind the software to consistently deliver performant and innovative solutions. PostgreSQL runs on [all major operating systems](https://www.postgresql.org/download/), has been [ACID](https://en.wikipedia.org/wiki/ACID)-compliant since 2001, and has powerful add-ons such as the popular [PostGIS](https://postgis.net/) geospatial database extender. It is no surprise that PostgreSQL has become the open source relational database of choice for many people and organizations.
* PostgreSQL comes with [many features](https://www.postgresql.org/about/featurematrix/) aimed to help developers build applications, administrators to protect data integrity and build fault-tolerant environments, and help you manage your data no matter how big or small the dataset. In addition to being [free and open source](https://www.postgresql.org/about/license/), PostgreSQL is highly extensible. For example, you can define your own data types, build out custom functions, even write code from [different programming languages](https://www.postgresql.org/docs/current/xplang.html) without recompiling your database!
* **Visual Studio 2017**

Microsoft Visual Studio is an [integrated development environment](https://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) from [Microsoft](https://en.wikipedia.org/wiki/Microsoft). It is used to develop [computer programs](https://en.wikipedia.org/wiki/Computer_program), as well as [websites](https://en.wikipedia.org/wiki/Web_site), [web apps](https://en.wikipedia.org/wiki/Web_app), [web services](https://en.wikipedia.org/wiki/Web_service) and [mobile apps](https://en.wikipedia.org/wiki/Mobile_app). Visual Studio uses Microsoft software development platforms such as [Windows API](https://en.wikipedia.org/wiki/Windows_API), [Windows Forms](https://en.wikipedia.org/wiki/Windows_Forms), [Windows Presentation Foundation](https://en.wikipedia.org/wiki/Windows_Presentation_Foundation), [Windows Store](https://en.wikipedia.org/wiki/Windows_Store) and [Microsoft Silverlight](https://en.wikipedia.org/wiki/Microsoft_Silverlight). It can produce both [native code](https://en.wikipedia.org/wiki/Machine_code) and [managed code](https://en.wikipedia.org/wiki/Managed_code).

Visual Studio includes a [code editor](https://en.wikipedia.org/wiki/Code_editor) supporting [IntelliSense](https://en.wikipedia.org/wiki/IntelliSense) (the [code completion](https://en.wikipedia.org/wiki/Code_completion) component) as well as [code refactoring](https://en.wikipedia.org/wiki/Code_refactoring). [The integrated debugger](https://en.wikipedia.org/wiki/Microsoft_Visual_Studio_Debugger) works both as a source-level debugger and a machine-level debugger. Other built-in tools include a [code profiler](https://en.wikipedia.org/wiki/Profiling_(computer_programming)), forms designer for building [GUI](https://en.wikipedia.org/wiki/GUI) applications, [web designer](https://en.wikipedia.org/wiki/Web_designer), [class](https://en.wikipedia.org/wiki/Class_(computing)) designer, and [database schema](https://en.wikipedia.org/wiki/Database_schema) designer. It accepts plug-ins that enhance the functionality at almost every level—including adding support for [source control](https://en.wikipedia.org/wiki/Source_control) systems (like [Subversion](https://en.wikipedia.org/wiki/Subversion_(software)) and [Git](https://en.wikipedia.org/wiki/Git)) and adding new toolsets like editors and visual designers for [domain-specific languages](https://en.wikipedia.org/wiki/Domain-specific_language) or toolsets for other aspects of the [software development lifecycle](https://en.wikipedia.org/wiki/Software_development_lifecycle) (like the [Team Foundation Server](https://en.wikipedia.org/wiki/Team_Foundation_Server) client: Team Explorer).

Visual Studio supports 36 different [programming languages](https://en.wikipedia.org/wiki/Programming_language) and allows the code editor and debugger to support (to varying degrees) nearly any programming language, provided a language-specific service exists.

* **Visual Studio Code**

Visual Studio Code is a [source-code editor](https://en.wikipedia.org/wiki/Source_code_editor) developed by [Microsoft](https://en.wikipedia.org/wiki/Microsoft) for [Windows](https://en.wikipedia.org/wiki/Windows), [Linux](https://en.wikipedia.org/wiki/Linux) and [macOS](https://en.wikipedia.org/wiki/MacOS). It includes support for [debugging](https://en.wikipedia.org/wiki/Debugging), embedded [Git](https://en.wikipedia.org/wiki/Git) control, [syntax highlighting](https://en.wikipedia.org/wiki/Syntax_highlighting), [intelligent code completion](https://en.wikipedia.org/wiki/Intelligent_code_completion), [snippets](https://en.wikipedia.org/wiki/Snippet_(programming)), and [code refactoring](https://en.wikipedia.org/wiki/Code_refactoring). It is also customizable, so users can change the editor's [theme](https://en.wikipedia.org/wiki/Theme_(computing)), [keyboard shortcuts](https://en.wikipedia.org/wiki/Keyboard_shortcut), and preferences. The source code is [open source](https://en.wikipedia.org/wiki/Free_and_open_source) and released under the permissive [MIT License](https://en.wikipedia.org/wiki/MIT_License). The compiled binaries are [freeware](https://en.wikipedia.org/wiki/Freeware) and free for private or commercial use.

Visual Studio Code combines the simplicity of a source code editor with powerful developer tooling, like IntelliSense code completion and debugging.

First and foremost, it is an editor that gets out of your way. The delightfully frictionless edit-build-debug cycle means less time fiddling with your environment, and more time executing on your ideas. Visual Studio Code supports macOS, Linux, and Windows - so you can hit the ground running, no matter the platform.

System Study

1. **An Existing System**

* The existing system is comprised of maintaining physical documentation & manual transfer of those documents without proper sharing mechanism in order to achieve evidence control & inspection service.
* It is possible to have multiple users or task tracker in a particular job. To maintain the flow of entities, inspection, and evidence one has to manage it manually by maintaining physical documentation of each and every process in existing system for evidence control and inspection management.
* Most of the time it would be very difficult to do paperwork if tasks or documents are too large with respect to time or complexity.

1. **Need of Proposed System**

This system is being built on a web platform that will manage personal documents and business-related financial documents on a single platform without having any manual paperwork.

It manages Supply chain management from source to destination with the proper flow of goods by providing quality assurance and quality control.

This centralized platform will use different machine learning techniques to verify the proper quality and quantity of goods. This makes it easy to follow product and process chain of supply by simply reviewing reports which answer who, what, where and when for every element you see.

1. **Scope of the Proposed System**

* This system will allow users to use all the modules provided in it after success full login followed by registration.
* After proper verification of details such as username/email and password, functionality would be accessible to the user.
* As per the requirement user can create forms for job and assets, view and update profile, upload and download data sheets, create Jobs with required fields. view job with different format, add team members who are responsible to get the task done, generate report, add attachments (images, files, audio, video) etc.

1. **Aim and Objective of the Proposed System**

* Inspection and evidence system specializing in management and controlling the proper flow of documents and task from point of origin to point of distribution
* Main purpose of this system is to eliminate the paper work to handle all documents. Management of proper service flow with less time and money by automating manual processes with the help of different technologies.
* The proposed system will feature different modules that ensures the proper flow of documents and related task to all those who are concerned with a job.
* A system that would be able to manage users with authentication, their documents, verification of team members, forms for task, flow of entities from source to destination with proper validation, assets that are required to perform different task etc.

1. **Feasibility Study**

A feasibility study is a short, focused study, which aims to answer a number of questions:

* Does the system contribute to the overall objectives of the organizations?
* Can the system be implemented using current technology and within given cost and schedule constraints?
* Can the system be integrated with systems which are already in place?
* Operational Feasibility
  + The System efficiently Operates and Reduce Manual Computation and Time of Processing, Reducing Cost of Paper works and Human Errors. The application working is quite easy to use and learn. User requires no special training.
* Technical Feasibility
  + The System can be implemented using basic Computer Software and Hardware, it does not use any special additional Hardware or software.
* Economic Feasibility
  + The System implementation is quite economical if either for a single individual as well as for large enterprises. The system being developed with respect to client’s point of view. It is cost effective in the sense that it eliminated existing system’s error.

System Analysis

**Requirements Specification**

* **Hardware Specification**
* Processor : Intel x86/ AMD/ ARM based processor @ 1GHz
* RAM : Minimum 1 GB
* Hard Disk : Minimum 2 GB
* System Type : 32/64 Bit
* **Software Specification**
* Operating System : Any type of OS that supports web browser.

* Web Browser : Firefox, IE 6, Google Chrome
* Front End Tool : Angular 6
* Back End Tool : ASP.NET Core, PostgreSQL

**System Modules**

* **DASHBOARD**
  + It is provided to the users to view summary like structure of activities.
  + This sub module will display following details.

Number of jobs created by user.

Total added location

Number of reports generated by user

Space/Storage occupy by the documents stored by the user.

Information about jobs such as name, author, status (Pending, in progress, Closed).

* **PROFILE**
  + User information will be display in this sub module.
  + It includes following details.

User Name

Email

Phone

Profile photo

Delete profile, Edit profile functionality

Change password functionality.

* **USER DOCUMENTS**
* If any documents are uploaded by user, then the details such as job name in which documents are uploaded, description, total size, last modified by, and updated time and date would be display in this sub module.
* **JOB**
  + Whenever this type of system will be introduced to user, it is necessary to have one most important feature that is able to manage or prepare single Job/Task.
  + In Job tab, front end part will display List of Jobs with below necessary details.

Different working functionalities such as create job, edit job, delete job, share job, view job is given below.

* After implementation of the solution, user will be able to create Job with job details (Title, Description, location, overview, and conclusion).
* In overview and conclusion tab, user will be able to use existing forms or can create form as per the requirements.
* System will provide dynamic form building functionality to satisfy their needs. Different attachment options would be there to attach images or documents in different format.
* Create job functionality will also allow to add task that has to be done, agents who will manage task, assets require to complete job etc.
* User will be able to share job with others who are responsible to actively participate in job.

**Report Generation.**

* After job is created successfully, user can generate report as per his format as there would be different formatting options available mentioned below.
* Basic options:

Print quality (Grayscale, Colorful)

Image options (All images, Select Images)

Layout options (Full, Compact)

Thumbnail size (Normal, 2x)

Additional details (Full size media, print attachments, Download links)

* Advanced options:

Choose sections (Check In, Check List, Images, Audio, Video, Job, Attachment, Email details, Conclusion history)

Grid line options (Horizontal, Vertical)

Table header options (Left-to-right, Top-to-bottom)

* Mail will be sent to user by which he can easily download report.
* **DATA IMPORT**
* This sub module will provide functionality to import or export data sheets (CSV, XLS, XLSX) having important data require to add or download for the task that has to be done.
* **INSTACOUNT**
* As the name suggest, this is the part of project in which instant count would be done.
* During the task, it is not feasible to count entity/objects manually. So, by uploading an image system would be able to give total count of the object such as pipes of different shapes.
* **FORMS**
* Different type of forms would be created here.
* Forms created in this sub module could be use into other Job Module also.
* It will allow creating three basic types of forms given below.

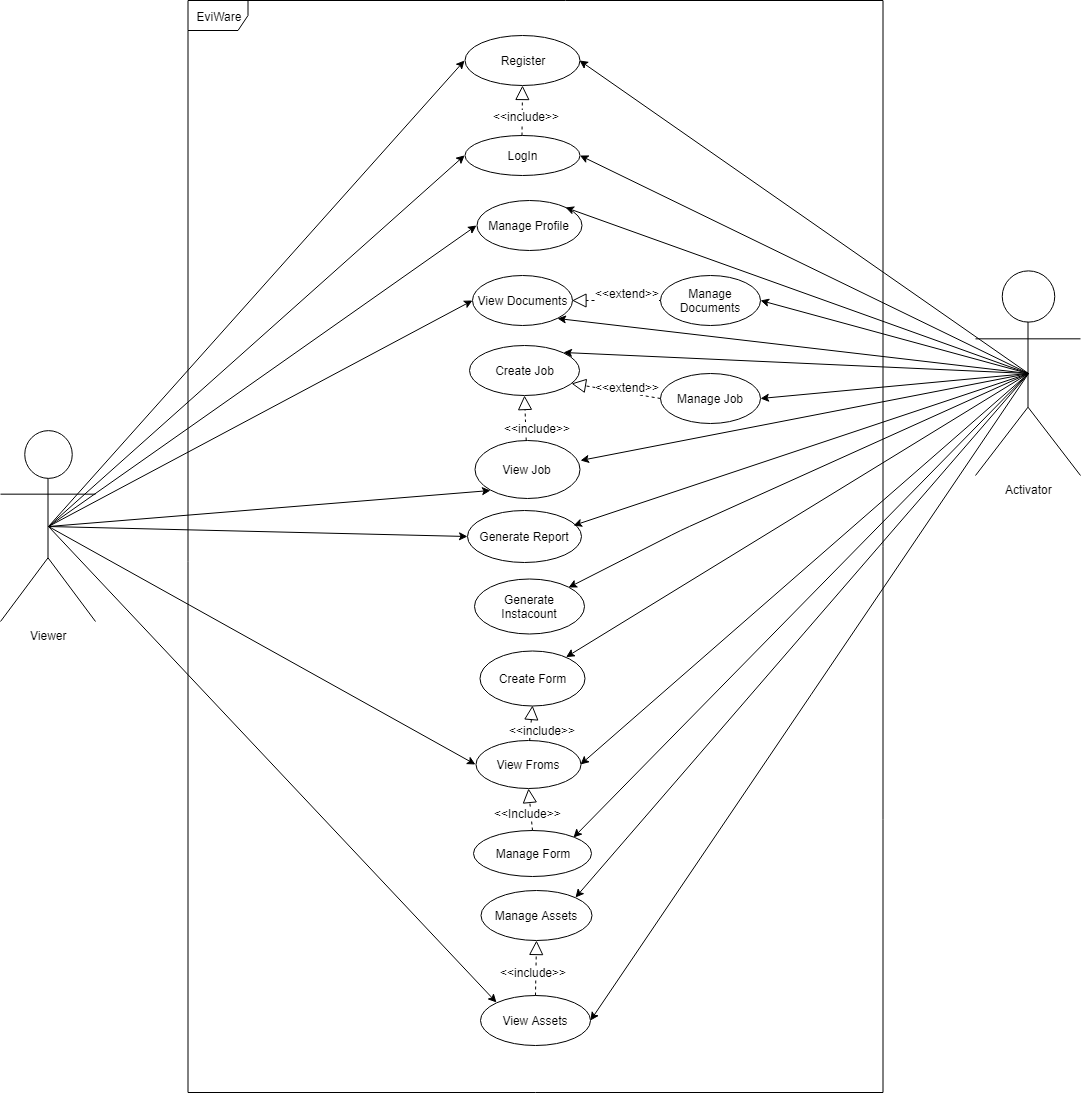
General Form

Overview Form

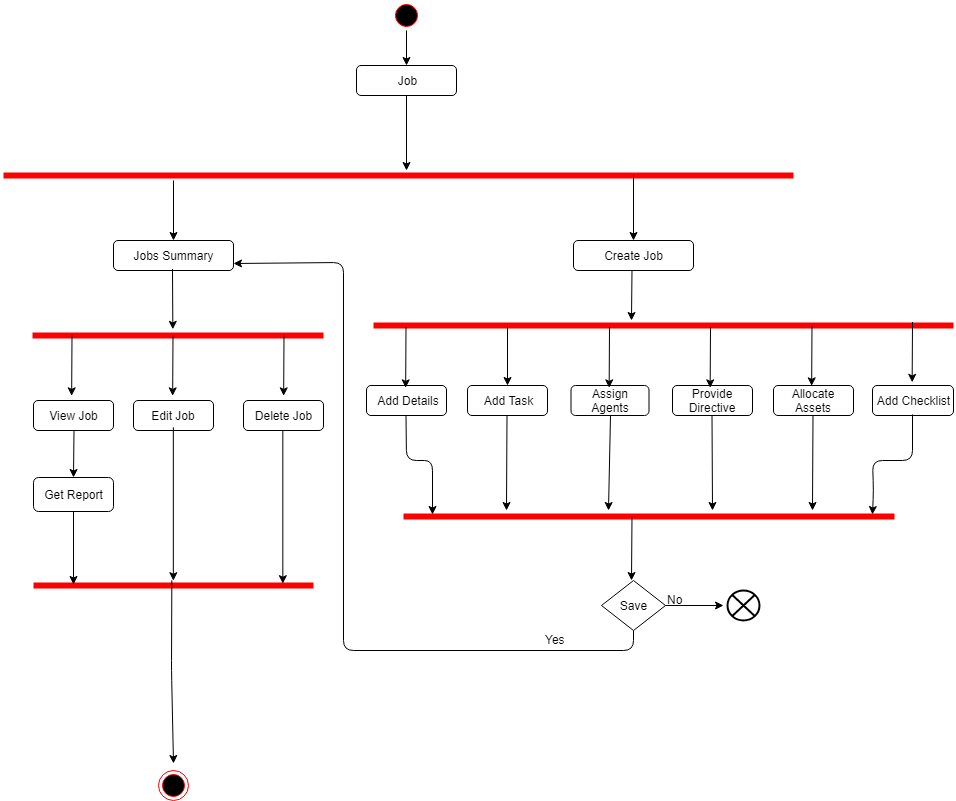
Conclusion Form.

* **ASSETS**
  + - List of assets require to complete the task would be added from here.
    - It will allow creating forms to add assets by dynamic form builder functionality.
    - These assets could be import to the Job module while creating job.
* **CONTACTS**
* This is the normal functionality that will display registered contact numbers by user.
* **UNIVERSAL SEARCH**
* This system enables functionality to go through all the details in the system in a very easy to use manner.
* As this project contains too many modules and functionality, it would be very crucial to display search result in single tab.
* This may create confusion to user to identify which result is pointing to which module.
* So, the task would be to create the search functionality that is able to display result with different tabs (all, job, data entered, mail conversations, assets, images, audio, video) and the link will redirect to the module in which search result is found.

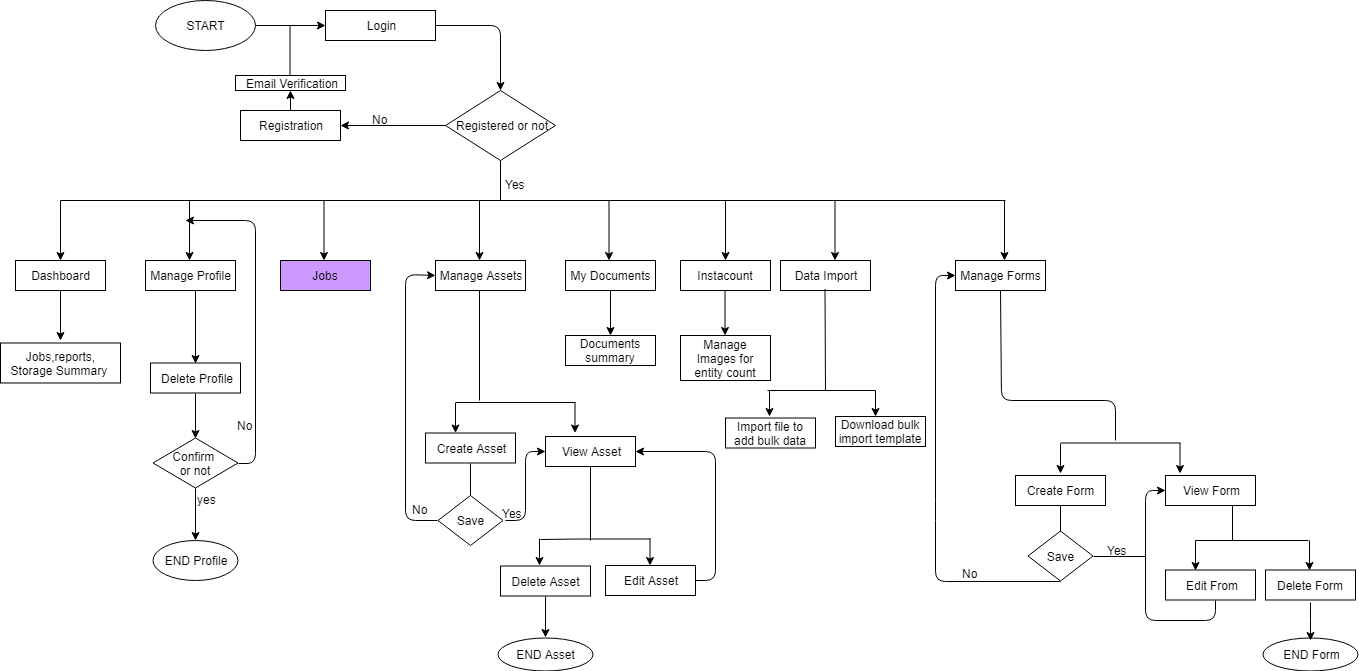
**Use Case Diagram**



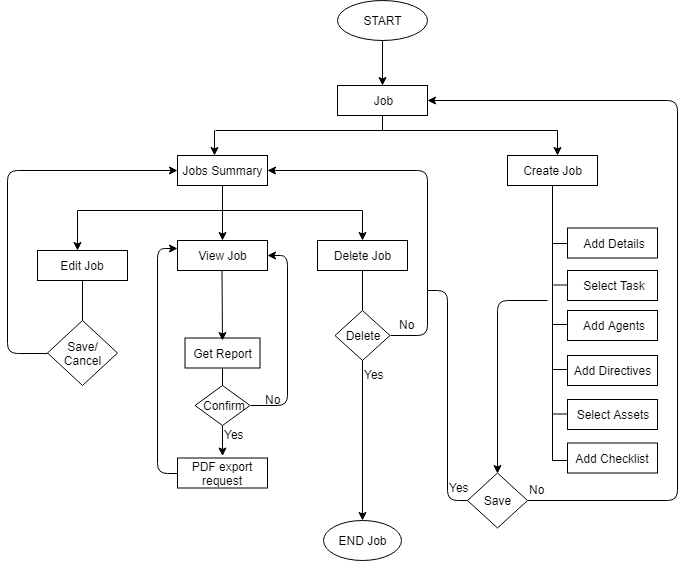
**Activity Diagram**



**Flow Chart**



**Job Module Flow Chart**



**Class Diagram**

System Design

**Data Dictionary**

* + **User:**
    - Contains the details of the user.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Fieldname** | **Datatype** | **Size** | **Constraints** | **Description** |
| id | Varchar | 50 | Primary key | Id of user |
| name | Varchar | 100 | Not null | Name of user |
| email | Varchar | 100 | Not null | Email of user |
| phone | Numeric | 15 | - | Phone no. of user |
| password | Varchar | 25 | Not null | Password of user |
| dp\_url | Text | - | - | Profile picture url |
| plan\_type | Numeric | 1 | Must be 1/2 | 1-Free, 2-Paid |
| plan\_expires\_on | Timestamp | - | Not Null | User plan expiration date |

* + **Form:**
    - Contains the details of the form.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Fieldname** | **Datatype** | **Size** | **Constraints** | **Description** |
| id | Varchar | 50 | Primary key | Id of form |
| user\_id | Varchar | 50 | Foreign key | User’s id who created the form |
| name | Varchar | 100 | Not Null | Title of the form |
| description | Text | - | - | Description of the form |
| file\_url | Text | - | Not null | Form structure file url |
| created\_on | Timestamp | - | Not null | Time when form is created |
| form\_type | Numeric | 1 | Must be 1/2/3 | 1-General, 2-Overview,3-Conclusion |
| Icon\_url | Text | - | - | Form icon’s url |

* + **Job:**
    - Contains the details of the job.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Fieldname** | **Datatype** | **Size** | **Constraints** | **Description** |
| id | Varchar | 50 | Primary key | Id of the job |
| user\_id | Varchar | 50 | Foreign key | User’s id who created the job |
| title | Varchar | 50 | Not null | Title of the job |
| description | Text | - | - | Description for the job |
| overview\_form | Varchar | 50 | Foreign key | References overview form’s id |
| conclusion\_form | Varchar | 50 | Foreign key | References conclusion form’s id |
| created\_on | Timestamp | - | Not null | Time when the form is created |
| status | Numeric | 1 | Must be 1/2/3 | 1-Pending, 2-Open, 3-Close |
| location\_text | Varchar | 100 | Not null | Location of job creation in text |
| location\_longitude | Varchar | 100 | Not null | Location in longitude |
| location\_latitude | Varchar | 100 | Not null | Location in latitude |
| special\_instruction | Text | - | - | Special instruction for the job |

* + **Job Attachments:**
    - Contains the details of the job attachment.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Fieldname** | **Datatype** | **Size** | **Constraints** | **Description** |
| id | Varchar | 50 | Primary key | Job attachment’s id |
| job\_id | Varchar | 50 | Foreign key | References job’s id |
| user\_id | Varchar | 50 | Foreign key | References user’s id |
| name | Varchar | 100 | Not Null | Name of the attached file |
| file\_url | Text | - | Not null | Attached file’s url |
| created\_on | Timestamp | - | Not null | Time when the file is attached |

* + **Job Tasks:**
    - Contains the details of the tasks related to the job.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Fieldname** | **Datatype** | **Size** | **Constraints** | **Description** |
| job\_id | Varchar | 50 | Foreign key | References job’s id |
| form\_id | Varchar | 50 | Foreign key | References form’s id |

* + **Job Agents:**
    - Contains the details of the agents to the jobs.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Fieldname** | **Datatype** | **Size** | **Constraints** | **Description** |
| job\_id | Varchar | 50 | Foreign key | References job’s id |
| user\_id | Varchar | 50 | Foreign key | References user’s id |
| agent\_type | Numeric | 1 | Must be 1/2 | 1-Activator, 2-Viewer |

* + **Job Checklist:**
    - Contains the details of the checklist associated with a job.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Fieldname** | **Datatype** | **Size** | **Constraints** | **Description** |
| job\_id | Varchar | 50 | Foreign key | References job’s id |
| checklist\_item | Varchar | 100 | Not null | Checklist item in text |

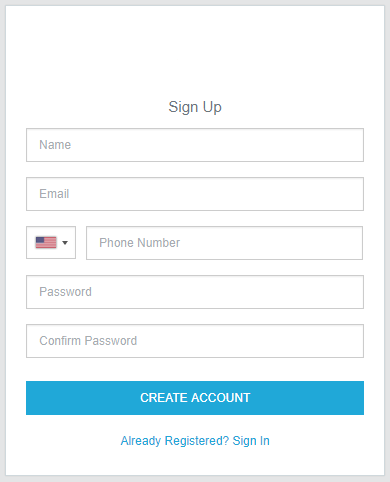
* + **Job Assets:**
    - Contains the details of the assets of a job.

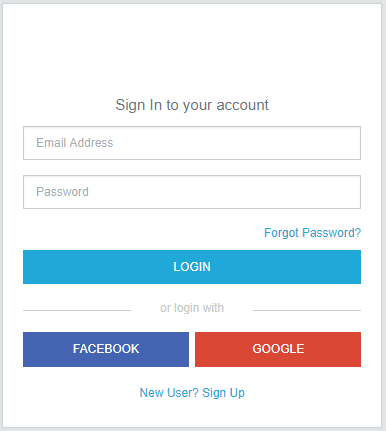
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Fieldname** | **Datatype** | **Size** | **Constraints** | **Description** |
| job\_id | Varchar | 50 | Foreign key | References job’s id |
| asset\_url | Text | - | Not null | Allocated asset’s url |

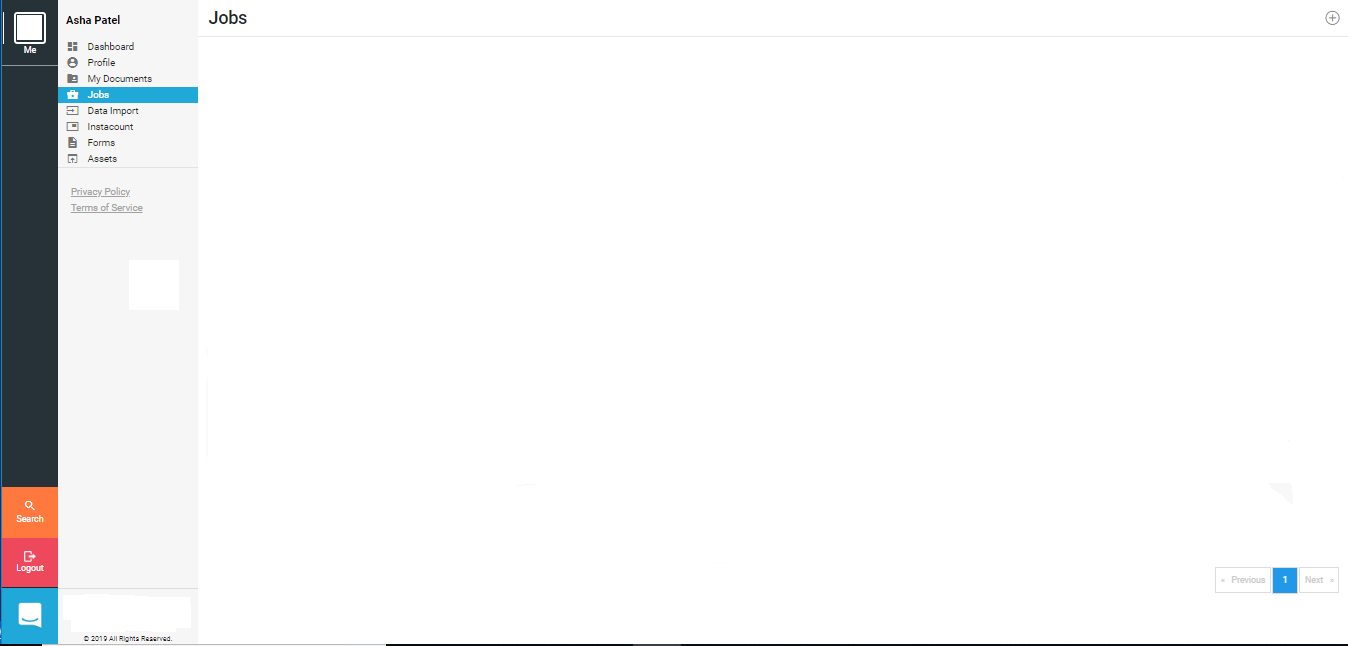
* + **Job Media:**
    - Contains the media files attached to the job.

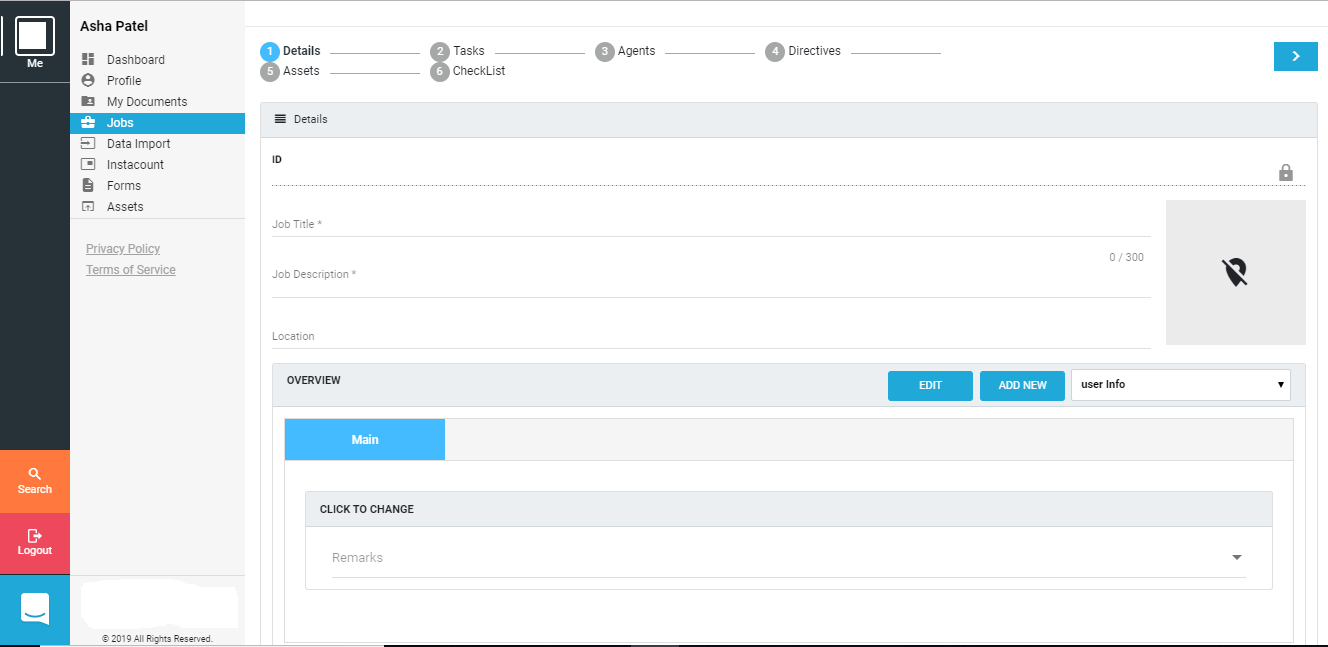
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Fieldname** | **Datatype** | **Size** | **Constraints** | **Description** |
| job\_id | Varchar | 50 | Foreign key | References job’s id |
| media\_url | Text | - | Not null | Media file’s url |

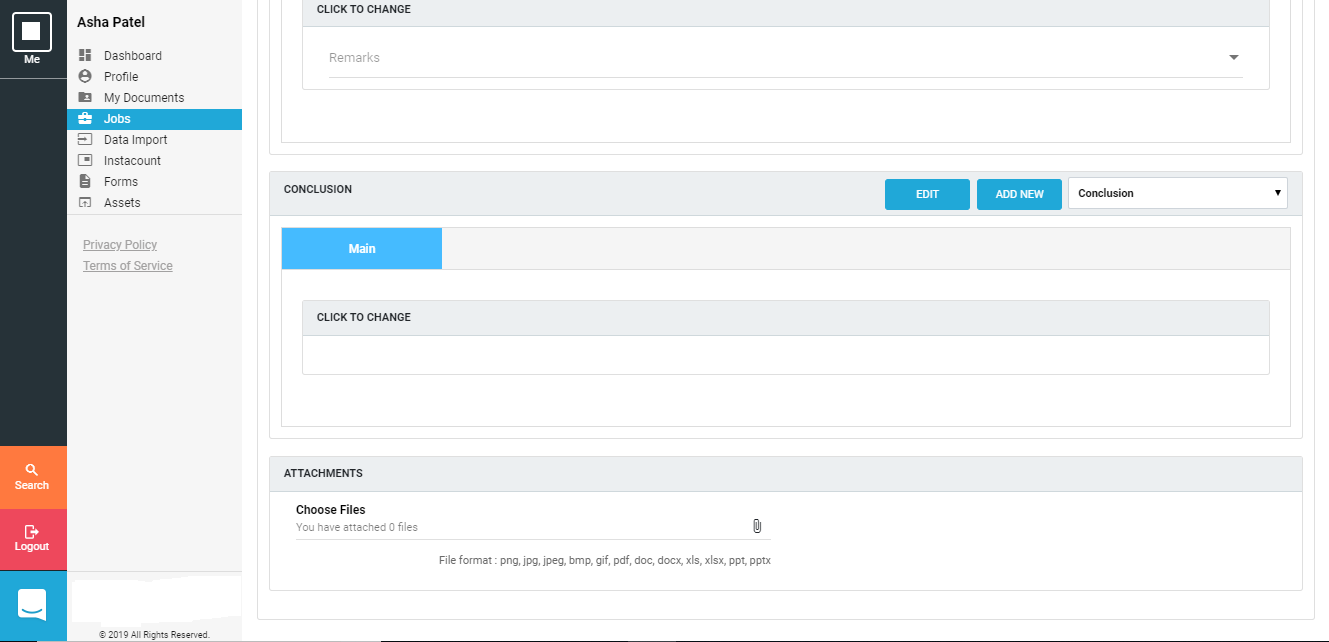
* **Screen Layout**

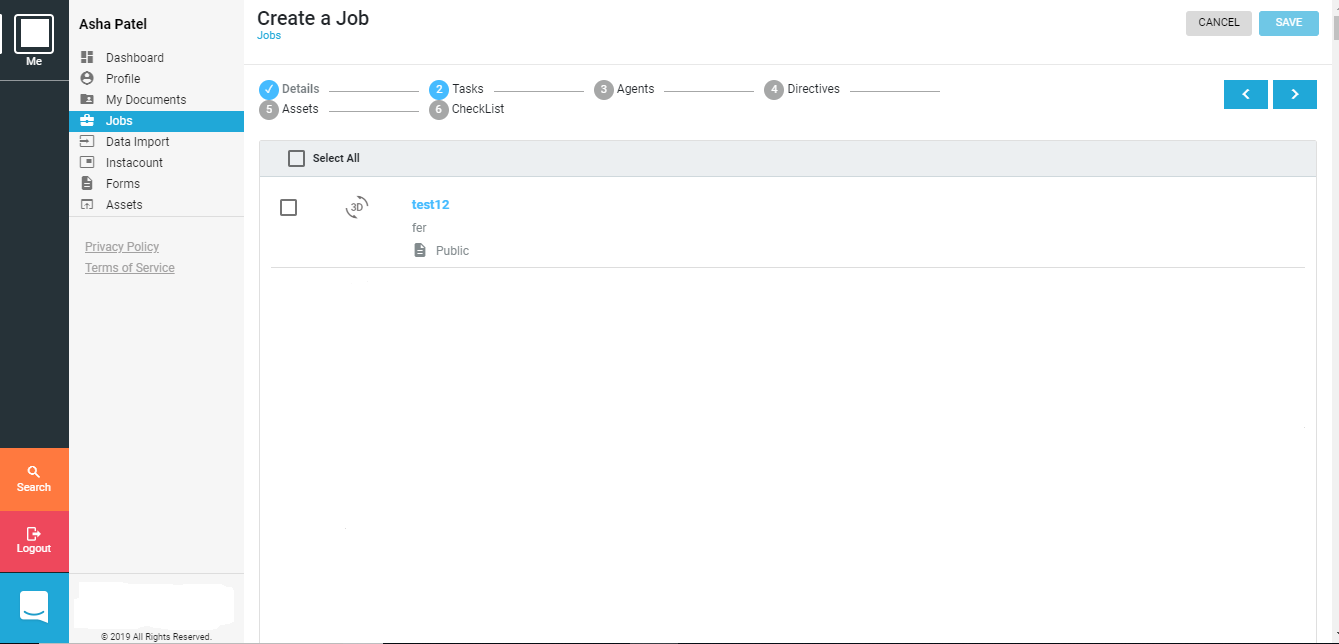
****

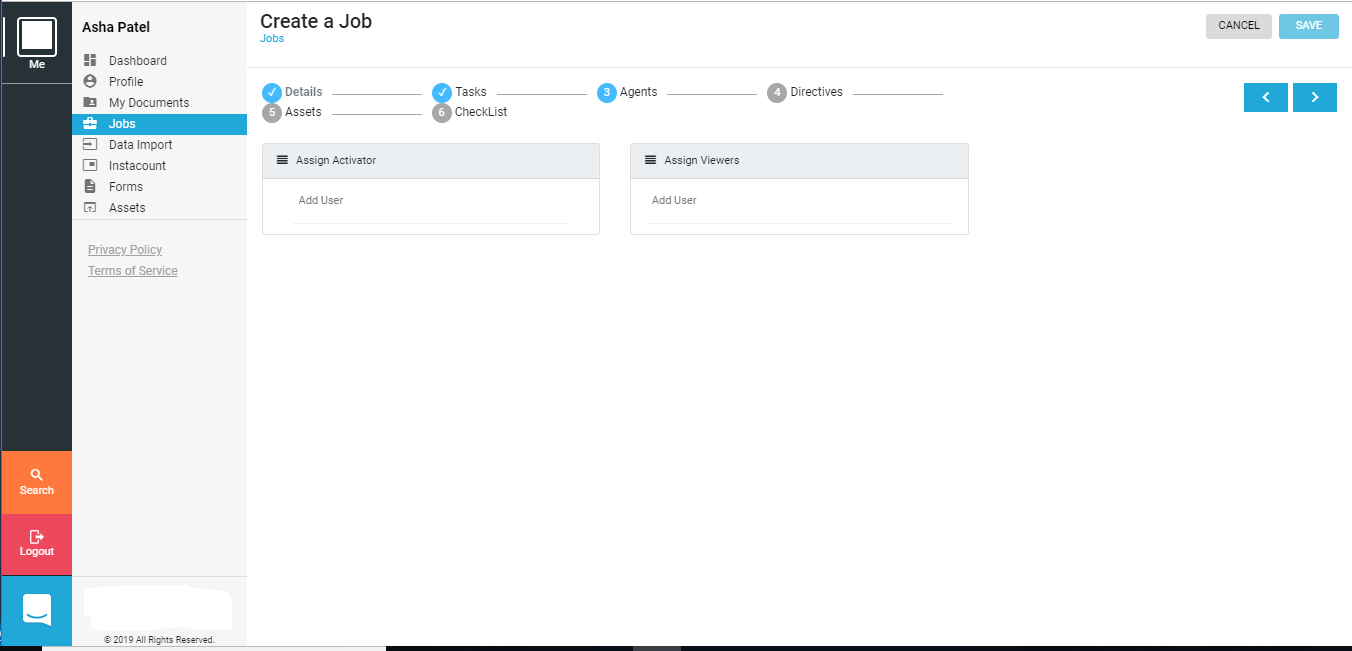


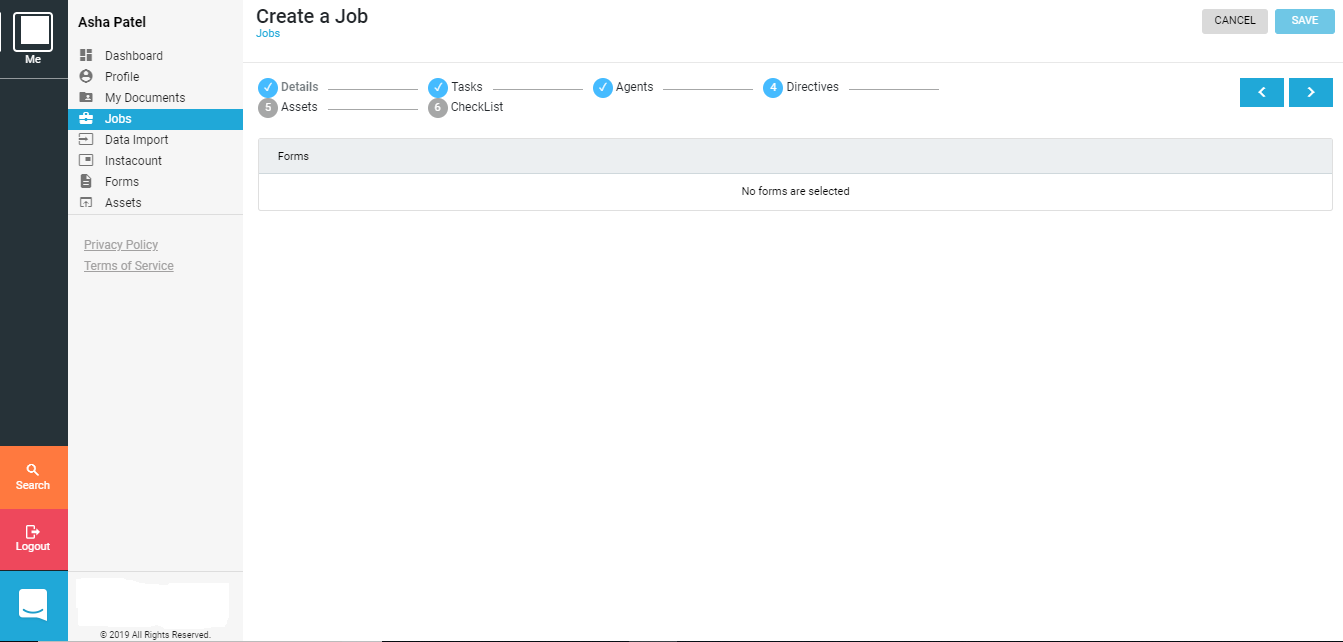


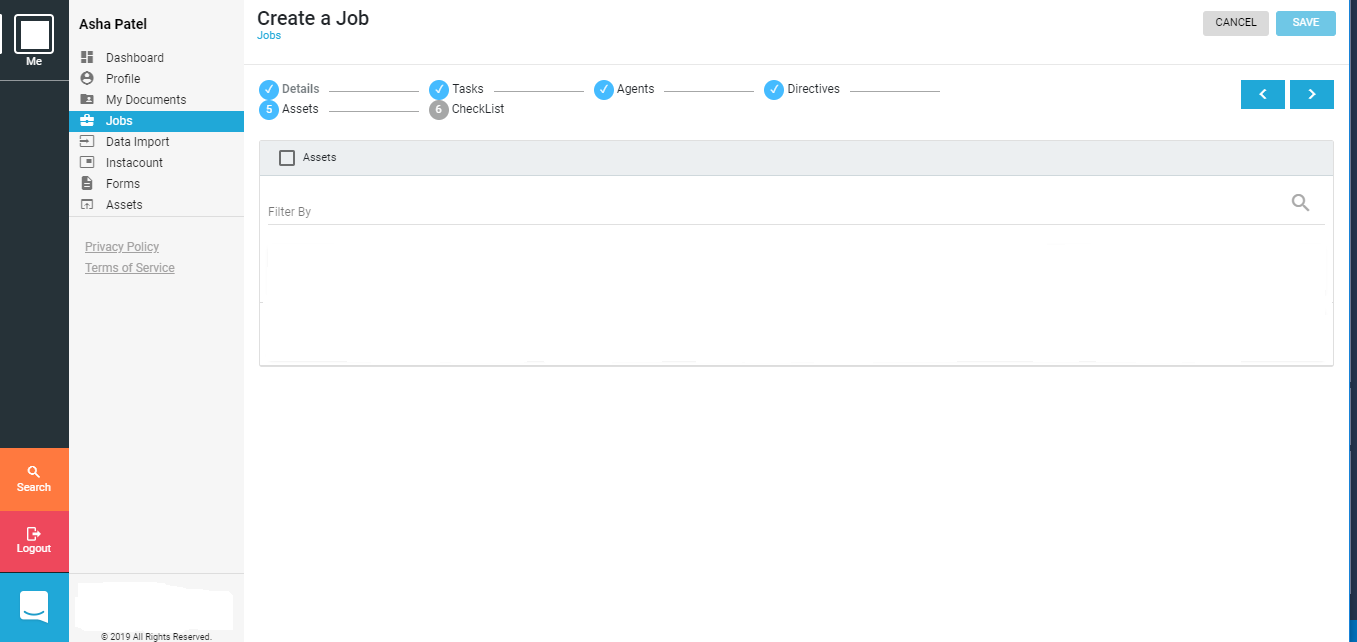


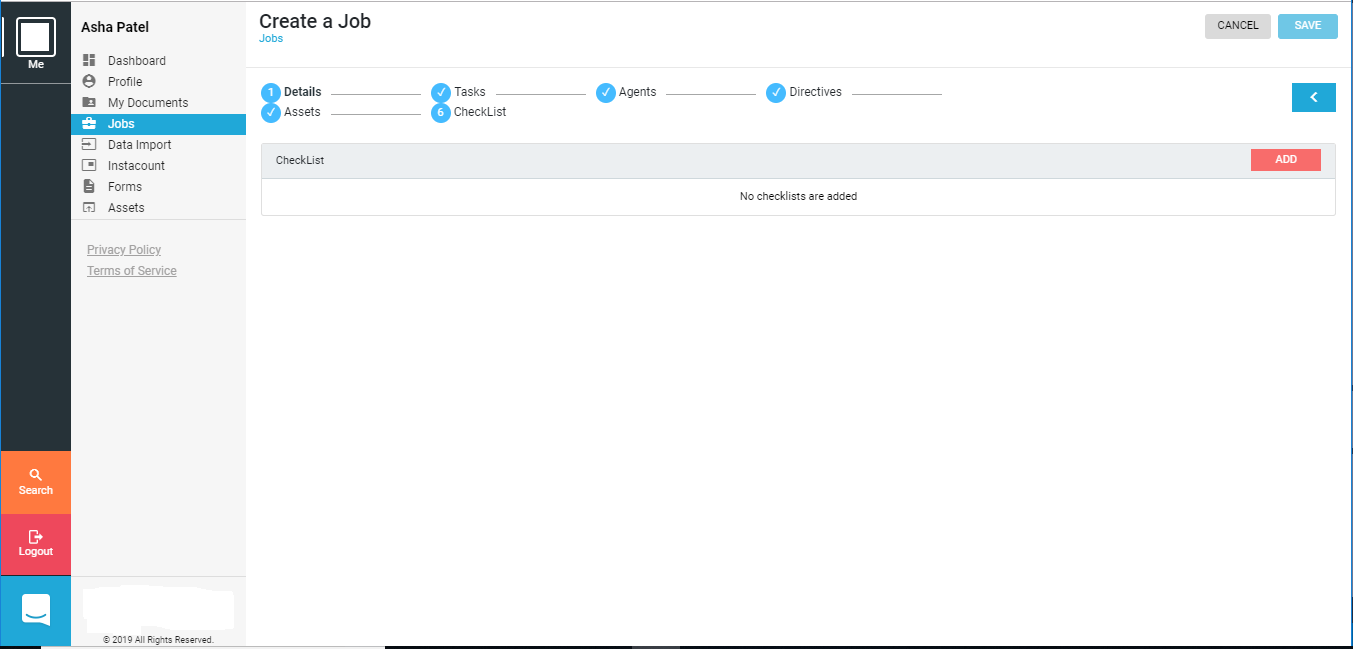


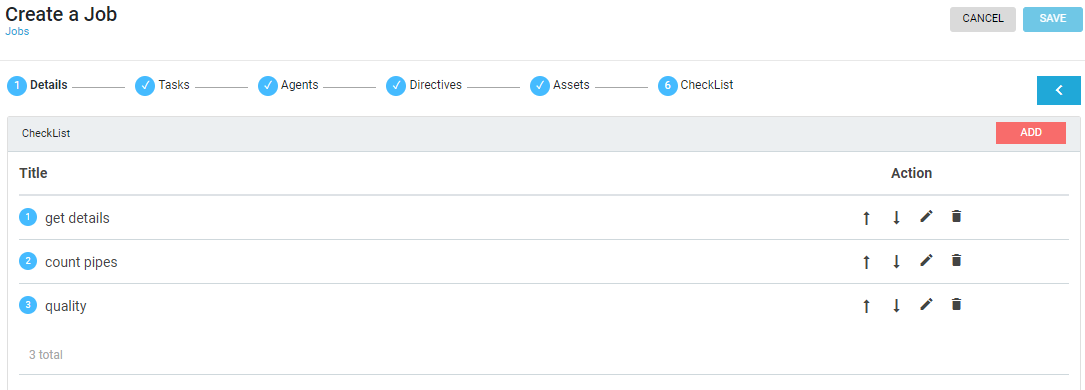
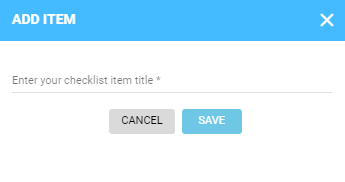


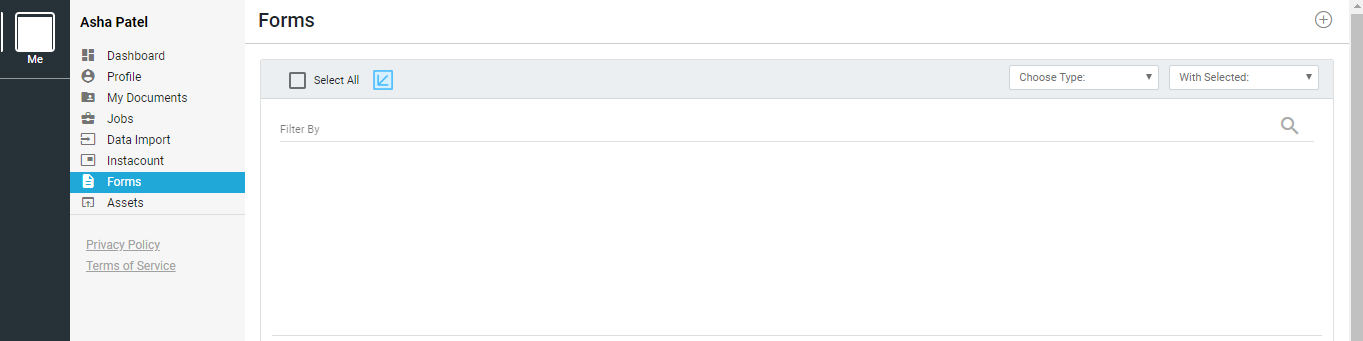


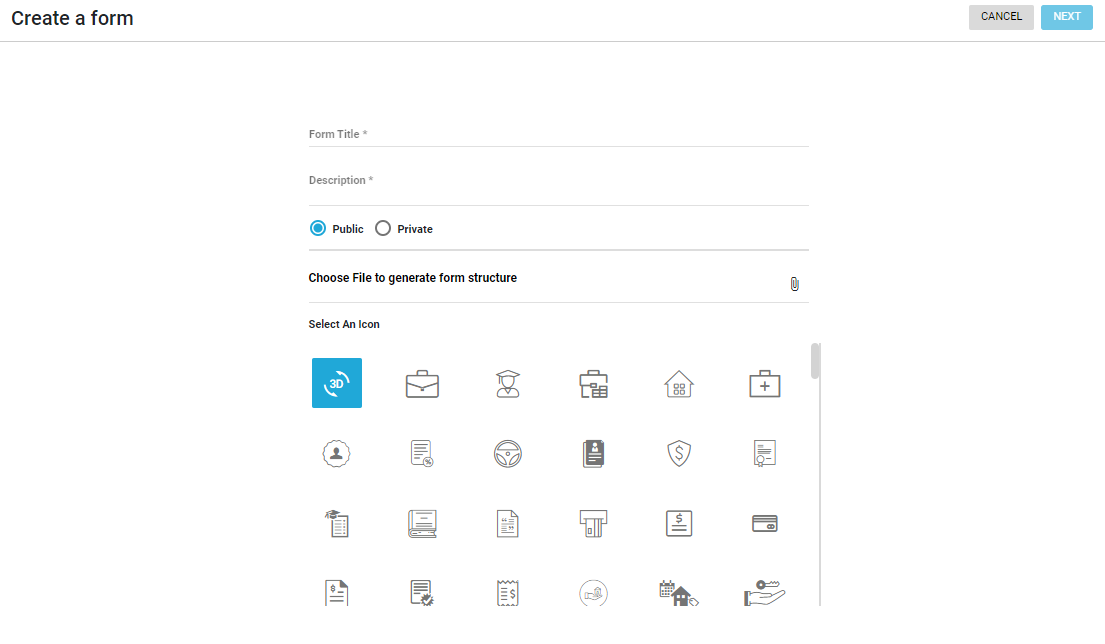
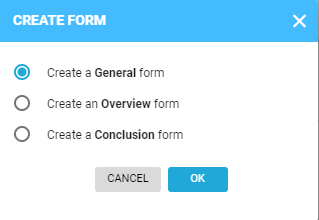












System Testing

* **Testing Strategies**

The software engineering process can be viewed as a spiral. Initially system engineering defines the role of software and leads to software requirement analysis where the information domain, function, behavior, performance, constraints and validation criteria for software are established. Moving inward along the spiral in along streamlines the level of abstraction on each turn.

A strategy for software testing may also be viewed in the context of the spiral. Unit testing begins at the vertex of the spiral and concentrates on each unit of the software as implemented in sourcecode**.**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case Id** | **Test Scenario** | **Steps of execution** | **Test Data** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| Tc\_01 |  |  |  |  |  |  |
| Tc\_02 |  |  |  |  |  |  |
| Tc\_03 |  |  |  |  |  |  |

Future enhancement

**Future Enhancement**

* In the Future a new functionality called Insta+ count will be added to determine number of products using machine learning techniques.
* A filter mechanism will be implemented to filter jobs as per certain parameters.
* Directives module will be implemented so that users will be able to provide directives to a job.
* Job reports functionality will be added so that users can generate Job reports with different options.

bibliography/

references