University of Mumbai
2020-2021

M.H. SABOO SIDDIK COLLEGE OF ENGINEERING 8,
Saboo Siddik Road, Byculla, Mumbai - 400 008
Department of Computer Engineering

Project Report

On

“Information Technology Service Management System”

By

Abuzar Shaikh-3117048
Shivam Tiwari-5117060
Needa Shaikh-3117057

Guided By
Dr. Zainab Pirani

KASMASOFT
The Long Lodge, 265-269 Kingston Road, London, SW19 3NW

CERTIFICATE OF EVALUATION

To whomsoever it may concern

This is to certify that **Mr. Abuzar Shaikh, Ms. Needa Shaikh** and **Mr. Shivam Tiwari** have jointly undertaken their final year outhouse project entitled "**IT Service Management System (ITSM)**" for the academic year 2020-21 as per guidelines stated by Mumbai University.

In brief, ITSM is a cloud-based web app designed to ensure smooth workflow of IT projects undertaken by a company; thereby adhering to the policies, guidelines and aim of the organisation.

I hereby confirm that they have completed all the assigned tasks/backlog items. However, the final code delivery and project documentation are yet to be presented.
The project was executed under the guidance of the undersigned.

We wish them all the best in their future endeavours.

Irfan Rashid Kasmani
Managing Director
For KASMASOFT LTD Limited

Place: Mumbai
Date: 19th May 2021

M.H. SABOO SIDDIK COLLEGE OF ENGINEERING

8, Saboo Siddik Road, Byculla, Mumbai - 400 008.

This is to certify that,

Roll No	Name
3117048	Abuzar Shaikh
5117060	Shivam Tiwari
3117057	Needa Shaikh

Of Final Year/Second year (B.E. Semester VIII) degree course in Computer Engineering, have completed the specified project report on.

“Information Technology Service Management System”

As a partial fulfillment of the project work in a satisfactory manner as per the rules of the curriculum laid by the University of Mumbai, during the Academic Year July 2020 - June 2021.

Dr.Zainab Pirani
Internal Guide

Irfan Kasmani
External Guide

_____
Internal Examiner

_____
External Examiner

_____
Head Of Department

_____
Principle

This project report entitled **“IT Service Management System”** by **Abuzar Shaikh, Shivam Tiwari, Needa Shaikh** is approved for the degree of Computer Engineering.

EXAMINERS

1._____

2._____

SUPERVISORS

1. _____

2._____

Date:

Place: Mumbai

ACKNOWLEDGEMENT

We would like to express our gratitude and appreciation to our parents for motivating and encouraging us throughout the career.

We wish to express our sincere thanks to our principal Dr. Ganesh Kame, M.H. Saboo Siddik College of Engineering for providing us all the facilities, support and wonderful environment to meet our project requirements.

We would also take the opportunity to express our humble gratitude to our Head of Department of Computer Engineering Dr. Zainab Pirani for supporting us in all aspects and for encouraging us with her valuable suggestions to make our project successful.

We our highly thankful to our internal project guide Dr. Zainab Pirani whose valuable guidance helped us understand the project better, their constant guidance and willingness to share their vast knowledge made us understand this project and its manifestations in great depths and helped us to complete the project successfully.

We would also like to acknowledge with much appreciation the role of the staff of the Computer Department, especially the Laboratory staff, who gave the permission to use the labs when needed and the necessary material to complete the project.

We would like to express our gratitude and appreciate the guidance given by other supervisors and project guides, their comments and tips helped us in improving our presentation skills. Although there may be many who remain unacknowledged in this humble note of appreciation but there are none who remain unappreciated.

Abuzar Shaikh

Shivam Tiwari

Needa Shaikh

Table of Content

List of Figures
[Indicate the figures wherever used]

Abstract

ITSM (Information Technology Service Management System) is a Cloud-Based Web App that is designed to handle the workflow of various IT projects that are undertaken in an organization. The workflow includes the commencement of the idea of the project from the Business Department to the Central Technical Surveillance Department, which monitors the entire workflow, and the IT Department which is responsible for the development of the project. There are various stages from the Initialization of the project, Documentation, and Validation. The Validation stages are further divided into various sub-stages where the idea is presented to each department for assessment and for providing a rough estimation of the cost that would be incurred while working on the project. The members of various Departments may hold meetings to discuss and to infer upon the changes or modifications that are necessary to incorporate in the project. The Web App aims to serve as a platform where the information can be stored, and shared centrally, to ensure a smooth workflow.

Chapter 1

Introduction and Motivation

1.1 Introduction

"The Airline Group" or "TAG" is a consortium owning a group of airlines in Europe. The company has around 50,000 employees across the airline is up. TAG has it's own IT division with its branding – "TAG tech". "TAG tech" handles all the IT operations across the airline group.

TAG tech has 3 major functions:

- ➔ Managing Business-As-Usual (BAU) operations of the various applications running on its infrastructure
- ➔ Track and fix issues and bugs
- ➔ Programs / Projects to align with the companies strategy/vision

Managing Business – As – Usual (BAU) operations

The functioning of business applications that support day to day operations is critical to the very existence of the organization. They are also called business-critical applications. The IT infrastructure (servers, networks, support, etc) are the backbone of these applications. This function tools/agents the existing IT infrastructure can support the optimal running of these applications. This function can include application end users, helpdesk support teams, application maintenance teams, monitoring tools/agents.

Track and fix issues/bugs

IT systems (hardware/software) need to have efficient issue tracking and fixing mechanisms to ensure business continuity. This function continually monitors bugs/issues reported by the users or system monitoring tools and tracks them to completion/issue resolution. The organization has employed ample resources to ensure any issues in its business-critical systems are promptly investigated and fixed.

Deliver Programs / Projects to align with the companies strategy/vision

Every organization needs to keep up with the changing market conditions and rapid technological advancement. To cater to this, TAG tech has this major function of initiating, managing, and delivering software projects that would keep the organization aligned to its mission statement as well as keep up with the market trends and latest technologies. TAG tech implements the PRINCE2 & Agile project management methodology to manage its portfolio.

The **ITSM (Information Technology Service Management) system** ensures that it caters to all the needs that are required by “TAG - Tech”.

1.2 Aim:

An IT company handles thousands of projects throughout a year, and handling these projects can be tedious and rather exhausting if each and every file is managed manually, the only advantage of managing these files digitally is that the physical storage required is less or negligible (excluding the storage of the computer) but then managing those data files manually is rather tedious of a task and can be extremely frustrating. And to top it off the lack of version control and generation of redundant files can lead to a lot of human prone errors which may end up causing a significant monetary loss for the company. Now to simplify the entire manual process, with the ability of version control and an inhouse deployed product we decided to develop an ITSM (IT Service Management System) that saves the manual work managing the files from the ideation process till the development. This product aims to reduce the excess and redundant effort taken by the employees to manage the files and systems while being device independent.

1.3 Objectives

- ➔ To ensure that all stakeholder interactions within the project process workflow happen smoothly.
- ➔ To efficiently store data and give insights based on the stored data to improve performance.
- ➔ Centralize the entire workflow.
- ➔ Avoid duplication of the data.
- ➔ Minimize the probability of human error that existed in the previous system.

1.4 Motivation

An IT Service Management System (ITSM) serves as the backbone of any IT firm. It becomes a crucial part of an IT-based organization to ensure that it has a streamlined and smooth process. It is also the sole purpose of an ITSM to ensure that all the stakeholders get access to the entire workflow. It facilitates effective collaboration for the smooth functioning of the organization/company. ITSM makes the flow of the project transparent to the respective personals and much more streamlined and easy to manage. This project aims to develop an ITSM which focuses on the functioning of 5 different child organizations which have different levels of workflows with a similar approach towards managing product development. This project focuses on developing the application by using the latest technologies which are used in the industry and the technologies that are flexible enough to be upgraded as a micro service in the future, the system will be built on a robust and a unstructured databases

for the ability of the database to be used for analytics and for managing the huge amount of data that is generated. Since the application will be deployed on inhouse premises the concerns for security will be minimized significantly and it can be further minimized down by implementing firewall and various security rules. As the system is being made for a limited number of audiences the number of users will be significantly low and the need to implement load balancing will be next to negligible. Since the project will be made as a web app it will be accessible over all types of device and will be OS independent.

1.5 Scope

Project scope is the part of project planning that involves determining and documenting a list of specific project goals, deliverables, features, functions, tasks, deadlines, and ultimately costs. In other words, it is what needs to be achieved and the work that must be done to deliver a project. This ITSM application can be adopted and implemented by different kinds of technology-based companies providing services in any kind of sector. It is useful for industry users to use this application conveniently anywhere and at any time through the company's may be situated necessarily not in the same country/state. The issues faced by any user at any time can be addressed by the concerned authority through the feature of messaging within the application.

1.5.1 Local Scope

The product "ITSM" is designed for catering the needs of TAG group and the airlines belonging to the consortium. The tasks to be accomplished under the local scopes are:

- Make the workflow easier and smooth.
- Digitize the entire workflow.
- Make monitoring and change management easier.
- Help increase productivity and performance.
- Save time and increase efficiency.
- Facilitate transparency

1.5.2 Global Scope

The global scope of the project will deal with a broader spectrum of requirements. The tasks viewed to be accomplished under global scope are:

- Integrate ITSM with the individual airline systems.
- Involve different and new project management strategies to create a single unified platform for handling, monitoring, and delivering projects.

Chapter 2

Problem Statement

TAG Tech has a change management process to initiate a Project and engage Business, IT / Infra & Project Management teams. This is coordinated by the CMO (Change Management Organisation) team. TAG tech has been facing a lot of issues with the current workflow in the past and has identified a lot of potential problems that can occur which will have a major impact on the business. The current workflow involves a lot of interaction between stakeholders that happens outside the project process. The current workflow is based on excel sheets. This makes data handling, storage, analysis a big overhead for the organization. As there is no central system managing this document workflow, email is used to communicate and send documents. This makes the system prone to human errors and unnecessary duplication of data. The organization requires an improved workflow and software that will streamline and digitize the process, make it transparent to stakeholders, and store key data that can be used for performance analytics and process improvements.

Chapter 3

Requirement Analysis

3.1 Review of Literature

The purpose of the literature survey is to identify regularly models, and papers in our proposed research area in an attempt to appreciate, make use of, as well as bridge a missing gap, if any, between different research.

BUSINESS CASE

What is a Business Case:

A PRINCE2 business case is a component of the Project Initiation Document (PID) and lies at the heart of every PRINCE2 project. It is the key driver of all projects undertaken using the PRINCE2 method. During the project, a PRINCE2 business case is a major control document that is referenced on a regular basis to ensure and confirm that the project remains viable. PRINCE2 business cases will contain justifications for a project, such as value for money for what is to be done and why it should be done now. The Business Case's key goal is to test the viability of the project.

How to create a Business Case

The format of a PRINCE2 business case is not prescribed in the PRINCE2 methodology and will therefore vary from company to company. However, as a guideline, the following questions should be considered:

- Why do we need to undertake this project?
- What are the business benefits?
- What are the risks?
- What are the potential costs?
- How long will the project take?

The following table describes the various review papers and the motivation behind reading them:

Purpose	Paper	Year of Publication	Description
Understanding the ITIL standards	Hendro Gunawan : Strategic Management for IT Services Using the Information Technology Infrastructure Library (ITIL) Framework Antti Lahtela, Marko Jantti, Jukka Kaukola Tieto : Implementing an ITIL-based IT Service Management Measurement System.	2019,2010	This paper focuses on ITIL, its primitives and specifications to be used while designing an ITSM.
Best practices for designing an ITSM	Vipul Jain, O. P. Wali and V. Raveendra Saradhi : Information Technology Service Management [ITSM] Research: A Literature Review of Practices, Solutions and Measurement	2018	Consists of standard practiced and tested ITSM schemas, designing approach,
Implementation essentials	Jon Idena, Tom Roar Eikebrokk : IT Service Management Implementation.	2013	Focuses mostly on the pre requisites, information gathering, system analysis
Enhance Performance and capabilities	Abdulazeez Ftahi, Abdul Hafeez-Baig, Raj Gururajan : Towards Effective Knowledge Application Capability in ITSM through Socialisation, Externalisation, Internalisation and Combination.	2019	Methods to increase performance measures, scalability, availability of your ITSM
Assessment and testing	Anup Shrestha Aileen Cater-Steel Mark Toleman	2016	Discusses about the testing endeavours for a good ITSM

3.2 Existing System

The existing workflow uses a manual document-based system where each committee that wishes to modify the document has to follow a series of steps which involves manual documentation for every step which is involved in the development of a product, throughout its lifecycle.
Even though the existing system has multiple layers of stages and guidelines the communication through each stage follows a more traditional approach of contacting via emails and for decision-making events, a panel of meetings is called, and then a collective decision is made.
The updates for every iteration of delivering the product is usually communicated via meetings and/or by email and spreadsheets. The estimation of the cost, assigning of Project Manager, Proposal Lead, individual tower head, subject matter experts are done only through the spreadsheet, and for every new revision, a new sheet instance is used as it helps in maintaining time-based data. For every update in cost of expenditure that would include CAPEX, OPEX and a total of both of these will be updated and a new revised instance of the cost sheet will be created (the term cost sheet refers to a record that is maintained digitally and/or traditionally via the concerned authority). The record for FDR (Front Door Request) also creates a chain for updates in all the corresponding documents so that the records are saved, for all these steps the necessary and respective authorities are informed and are requested to update their records as well. To keep track of all the meetings for every step/iteration/epoch of the development cycle there are specific action trackers maintained by the organization, these trackers are known as:

1. Triage Tracker
2. Solution Sketch Tracker
3. Statement of Work Tracker

Each of these trackers maintains the records for their respective category and are later used for the analysis of the development to gain various insights such as the efficiency of the development, improvement over the previous work, how well the particular team performed, the number of meetings conducted, the performance of every individual team leader i.e. Proposal Lead and Project Managers, and then these factors along with or without others can be used to improve the overall development of future applications.As the initially mentioned TAG is referred to as The Airlines Group which specifies a cluster of companies under the parent name of TAG improvement of the company as a whole is an important aspect which in turn leads to providing the employees to work along with cross-company, though the process remains same it gives an employee of one child company a chance to understand the work culture, ethics, and environment of other company.

3.3 Drawbacks of existing system

- ➔ The current workflow involves a lot of interaction between stakeholders that happens outside the project process.
- ➔ The current workflow is based on excel sheets. This makes data handling, storage, analysis a big overhead for the organization.
- ➔ As there is no central system managing this document workflow, email is used to communicate and send documents. This makes the system prone to human errors and unnecessary duplication of data.

Chapter 4

DesignDetails

4.1 Design of the system

The existing approach is disadvantageous as it follows a traditional approach as mentioned in drawbacks of existing systems. Therefore, an advanced and automated system is required which overcomes the drawback of the existing approach.

The proposed system aims to make use of principles of the traditional approach and provide a more simple, central, and automated approach. Thus reducing various factors such as data redundancy, inefficient communication and human based error. For understanding the system we can divide it into various modules and understand them:

- **Databases:** The database is the most fundamental part of the application as the entire record maintenance will be on the cloud, more precisely a centralised cloud. The database architecture for each operating company is separate with some exceptions where an employee works in more than one organisation. There will be a centralized database for FDRs, costing sheet, action trackers corresponding to the respected employees working on it. The database for users will be centralized for reasons such as reducing latency while logging in, this also helps the switching of operating companies and/or domains easily. Now from the above description one can easily conclude one or both of the two things:
    a. The generated database would be large
    b. And, how will you perform analysis

  considering both the points the database technology that will be used is MongoDB or simply non relational databases, or NoSQL databases, as these store the data in the form of documents storing and analysis both becomes relatively easy and simple. Along with giving us the benefits we need it easy to scale and relatively easy to work with once the engine is properly configured.

- **The Web Stack:** Since the entire application will be automated one major concern would be to make the system responsive, this provides the flexibility to the user to make the necessary changes on the go from a smartphone, a laptop, tablet, a computer, this improves the overall efficiency of working for an organisation, consider an example, if there are 10 FDRs under a single business user, in a traditional system you have to browse thrice the number of revisions to go through them and check the status of the development, when it was last updated, what is the current status of the same and if the latest update is missing the user will have to contact the proposal lead of each projects to know their status, but our system makes all of these as simple as browsing to the dashboard and then selecting the required FDR and all the details are visible, at the same time if the business user wishes to add the comment he/she does not needs to contact the respected authority, all he/she has to do is just add a comment for that.

So now the question arises is it possible to scale the entire application to be able to work with the entire group of companies individually while being centralized at its core, and how well can the team deal with the latency issues. Answering them, yes it is true to scale the entire application using the right tools for it, and in our case we are using the Django framework which is built for large scale applications such as ours and is secure because of its strictness, as for being centralized the concept of subdomains, splitting of the databases (as in our case) can be used. Now coming towards latency issues the technique to deal with it is on both the hardware and software front, the application should be optimized to use the process, fields that are necessary, while using caching for storing less secure details on the users system such as the styling and UI components, as for the hardware front a decently configured server along with the 10 bit gigabit ethernet for data transfer, the latency issues would be resolved.

- **The Workflow on the ITSM:** The workflow will be completely automated and the approach will remain the same as that of the traditional system that is being followed, so the users will have less to no efforts getting accustomed to the application. The application will work in the following ways:
    - Every user will have a dedicated dashboard for them with the normal permissions and rights that a user gets with his/her designation. Eg:
        - The business user will have the option to create the FDR and monitor it
        - Proposal lead will have the option to update the FDR, corresponding costing sheet, action trackers along with the option for updating the details of the mentioned documents if needed.
        - The project manager will have the ability view and update the FDRs, and the Triage Action Tracker in the read only mode.
        - Each SME will have the option to update the costing sheet, FDRs as necessary for their respective towers.
    - The costing sheets, FDRs being centralized can be viewed and updated by only the necessary user and then forwarded for further actions, and then the further actions can be taken on the same object (document) thus reducing the redundancy of the documentation. While at the same time the timestamp of each of these actions being recorded for future analysis.
    - The application will provide a simple drop down for the users who belong to multiple companies for them to switch between the organizations with no effort.
    - The status update of the FDR fields, and the individual status will be automatically updated by the application thus removing the possibility of the human error of closing a particular FDR later to its actual date. And these actions will be recorded as well. Each FDR will have the option to upload the images, pdfs, docs for the solution sketch and statement of work.
    - Coming to the communication between the users, it will be accomplished by either/both medium of commenting on the documents and by using an inbuilt mailing service.

- The application will have inbuilt analysis and reporting service thus removing or reducing the need to depend on external tools, while giving the ability to perform organization specific analysis with ease and at the same time being available within a common application.
- There will be a superuser or admin who will be granted the ability to create the users, as the application is organization specific signing up can only be provided to the user by superuser or admin.

- **Security:** Coming from all the above mentioned points it is an obvious question how secure is the application. And to answer that question there are few things that are required to be implemented (or will be implemented if absent), those are a good firewall, SSL integration and secure hosting platform. Other than these the application provides domain specific logins, load balancing for simultaneous logins, access only to the necessary ports, communication within the application to avoid any data leak/breach that can happen, records of every email sent or received to trace the origin of the issue if any happens, and etc. depending upon the requirements.

4.1.3 GUI for Users

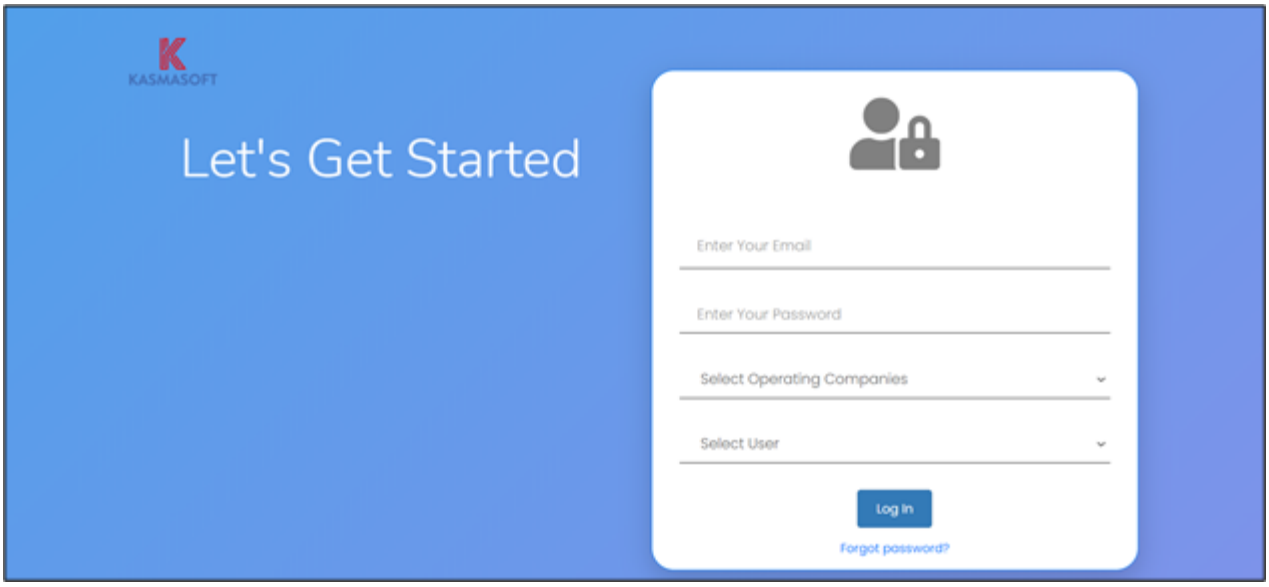


Figure 4.1.3.1: Login Page

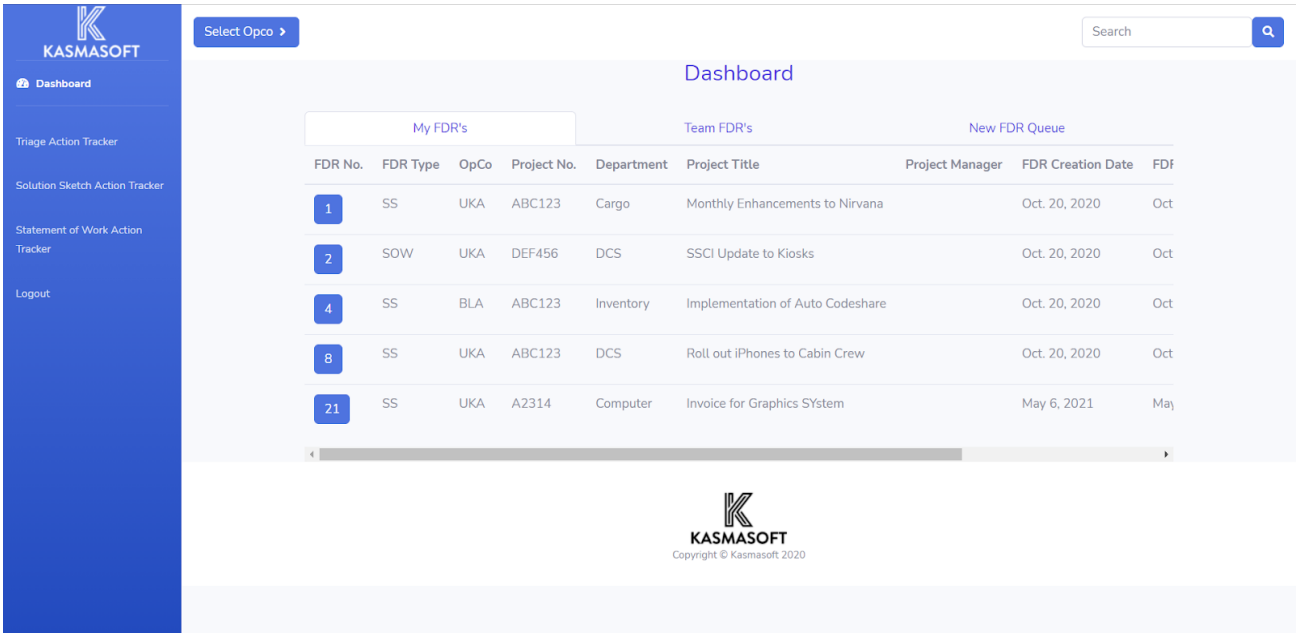


Figure 4.1.3.2: User Dashboard

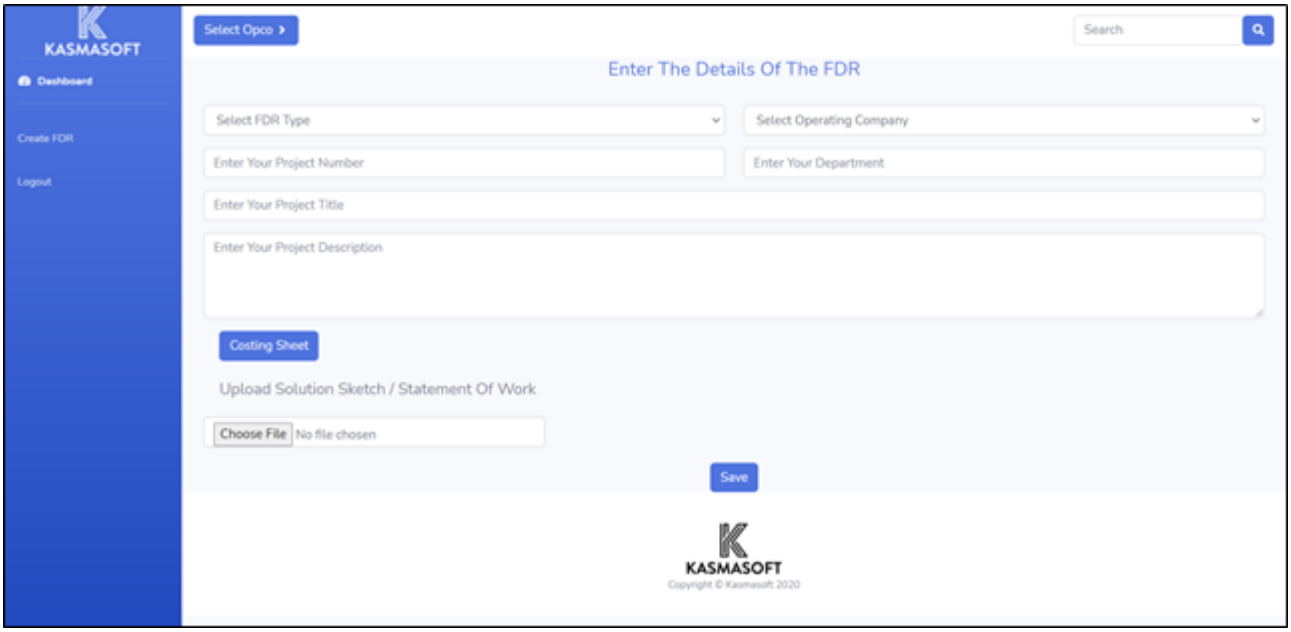


Figure 4.1.3.3: Creation of FDR

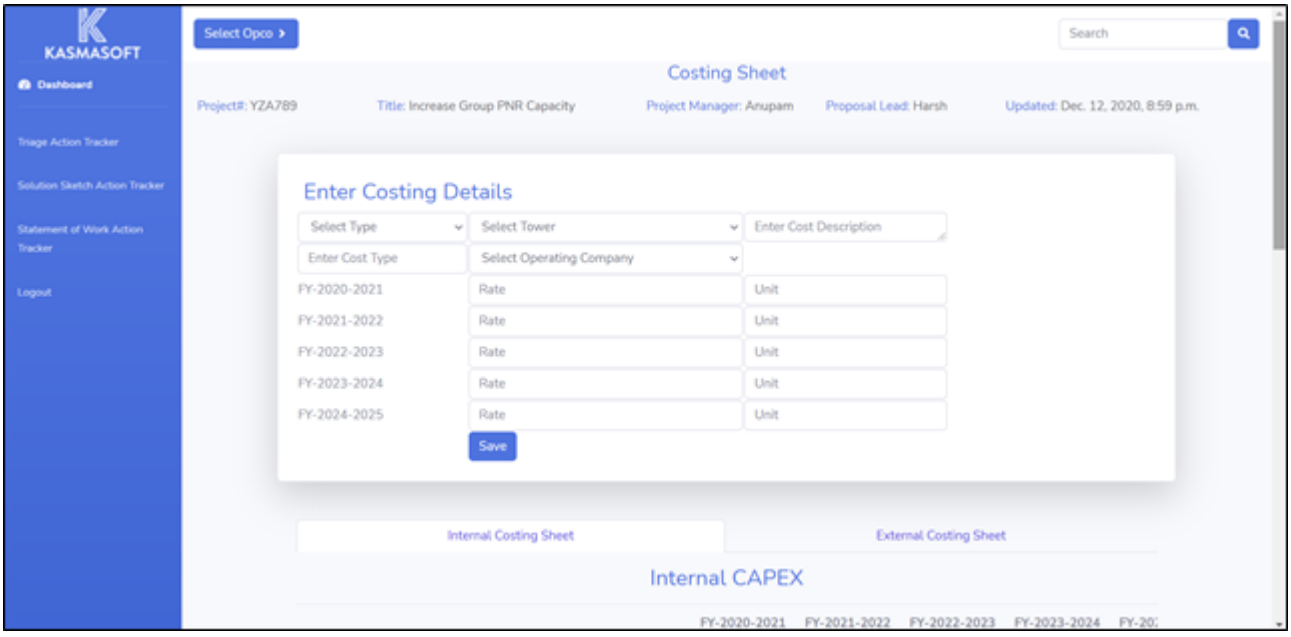


Figure 4.1.3.4: Costing Sheet

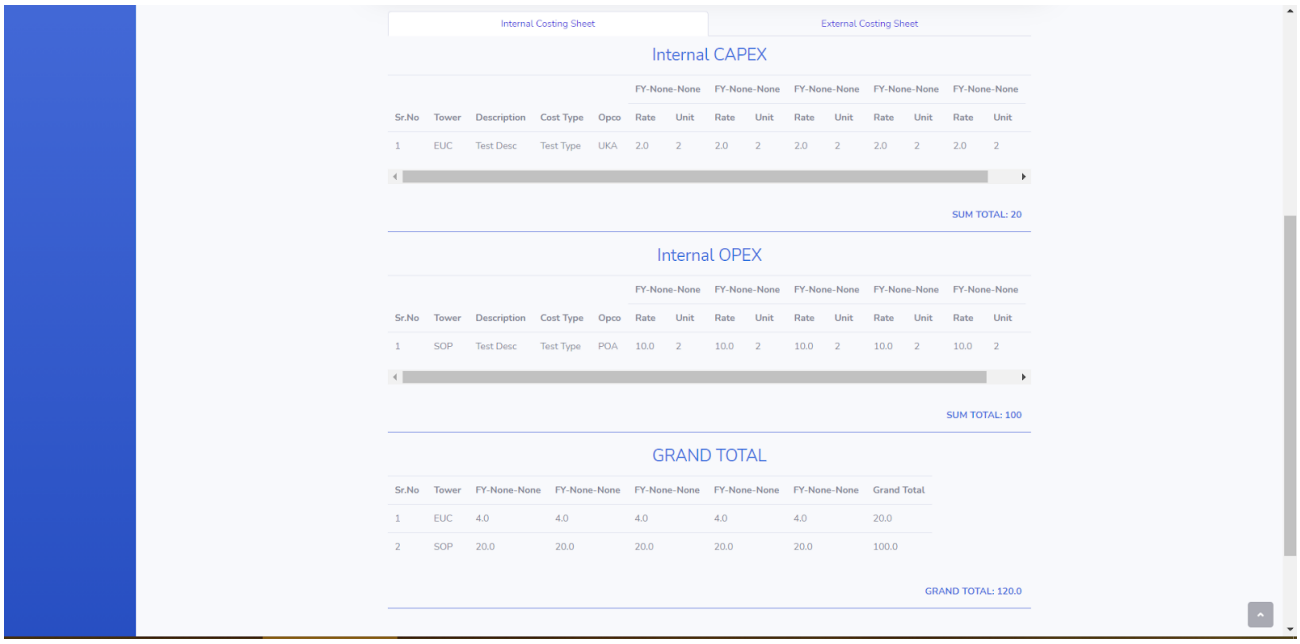


Figure 4.1.3.5: Internal Capex

Figure 4.1.3.6: External Capex

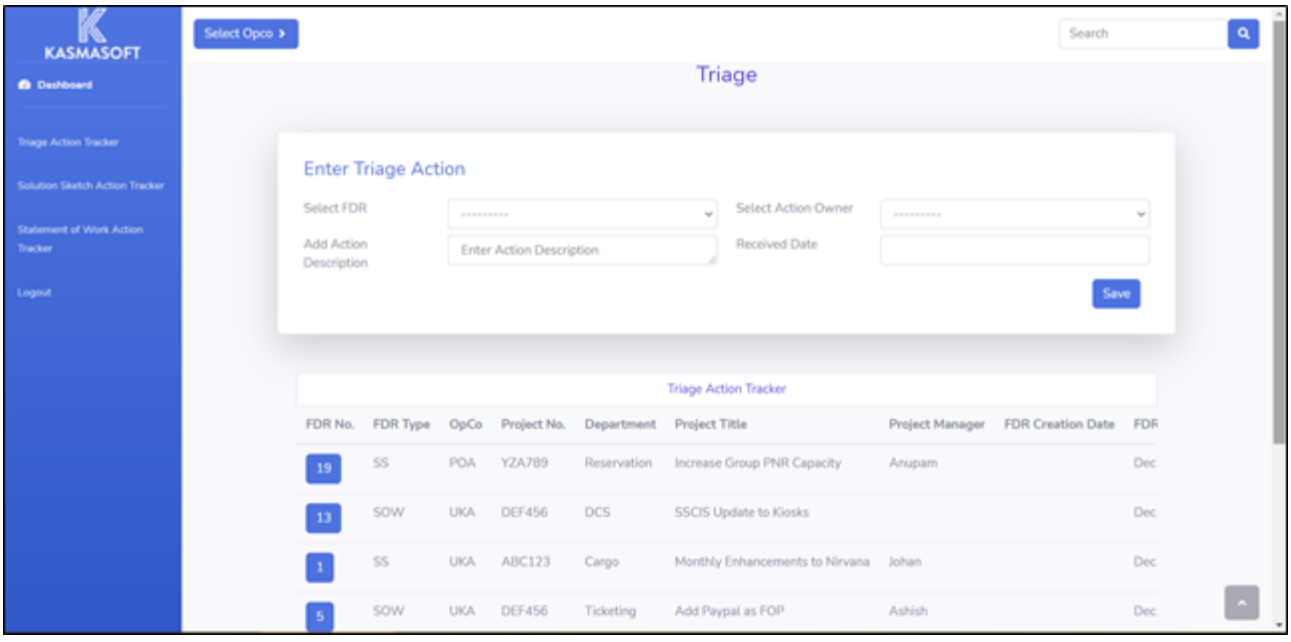


Figure 4.1.3.7: Triage Action Tracker

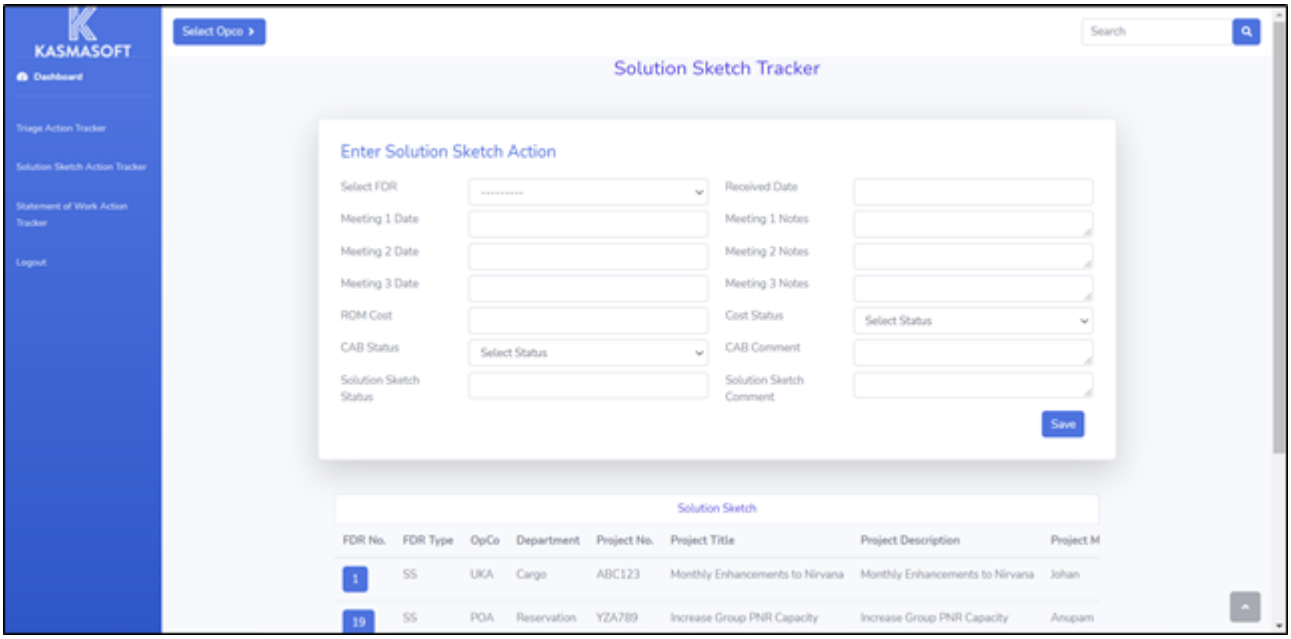


Figure : 4.1.3.8 Solution Sketch Tracker

4.2 Flow Chart

Fig 4.2.1.High Level Workflow

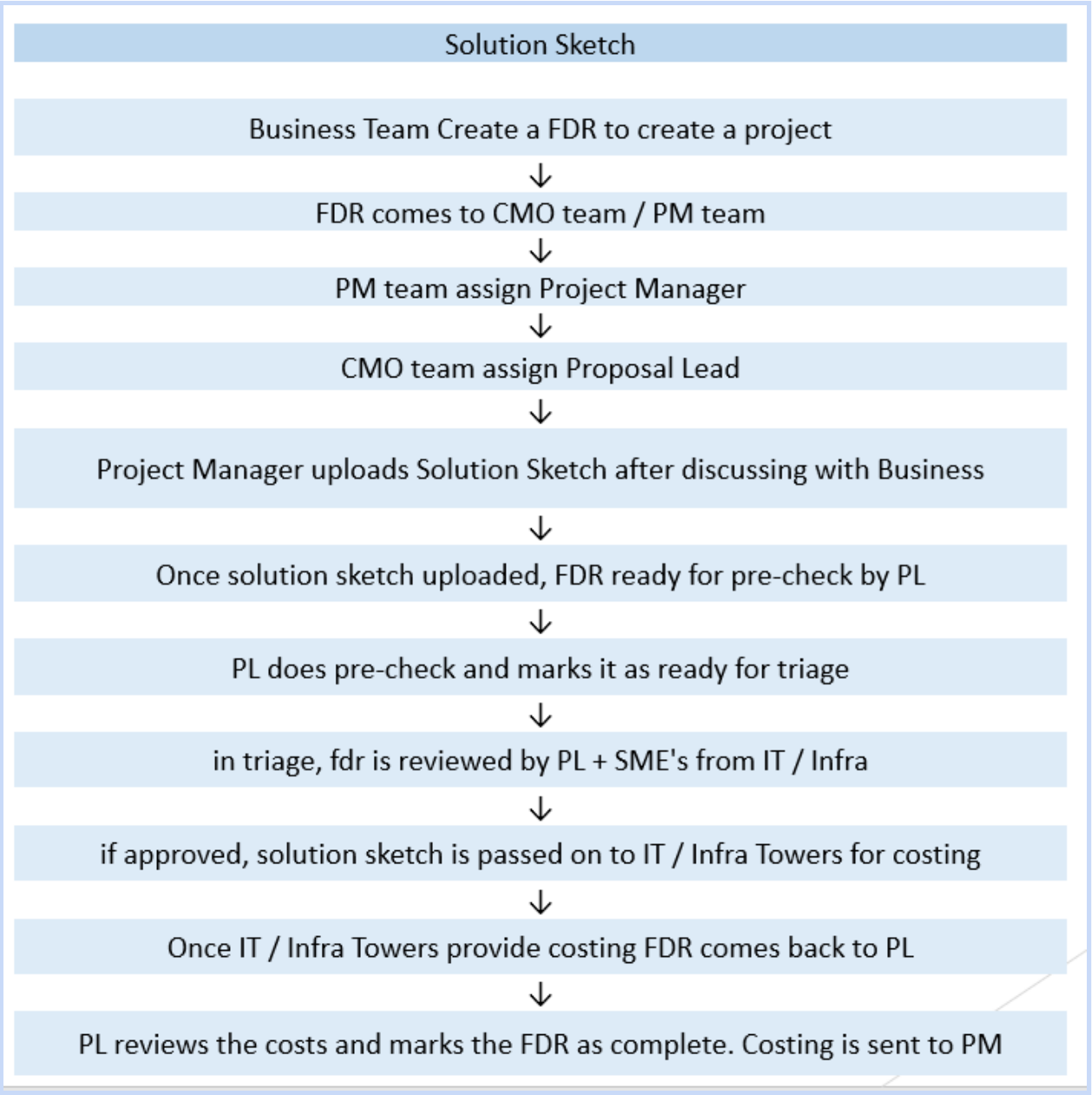


Fig 4.2.2.Solution Sketch

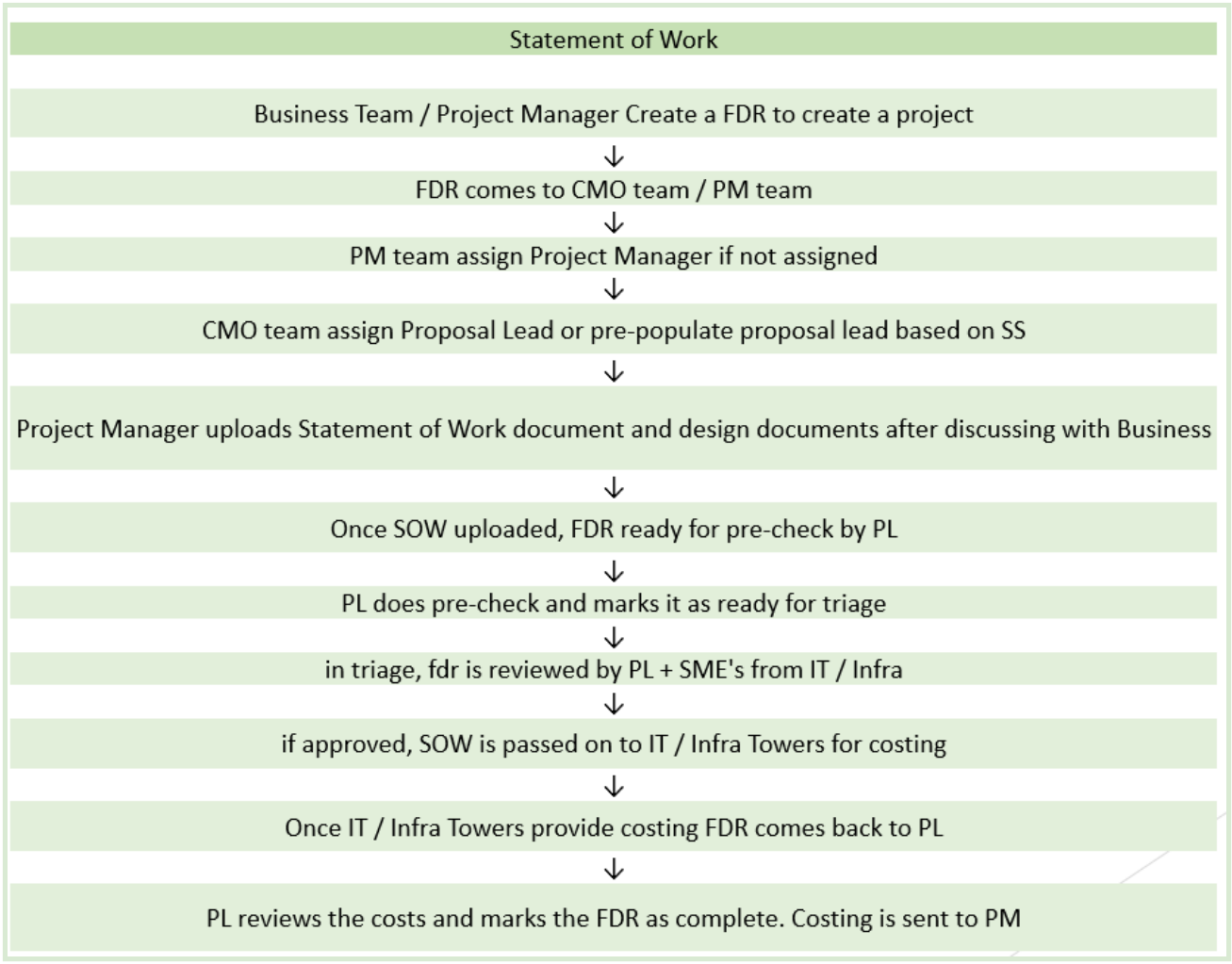
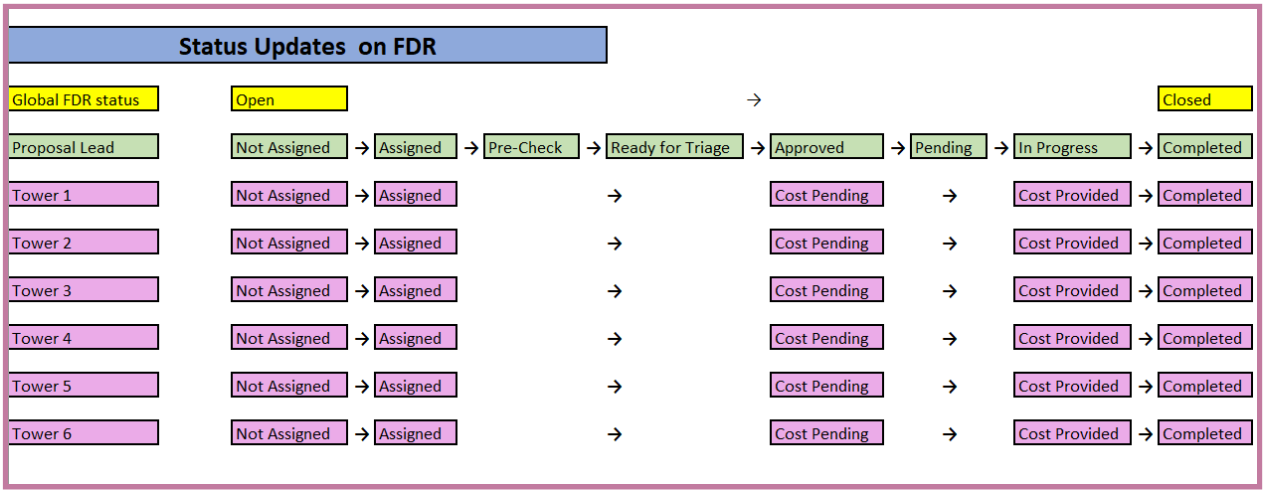


Fig 4.2.3 Statement of Work

4.3 Block Diagram



4.3 Block Diagram

4.4 Data Flow Diagram (DFD)

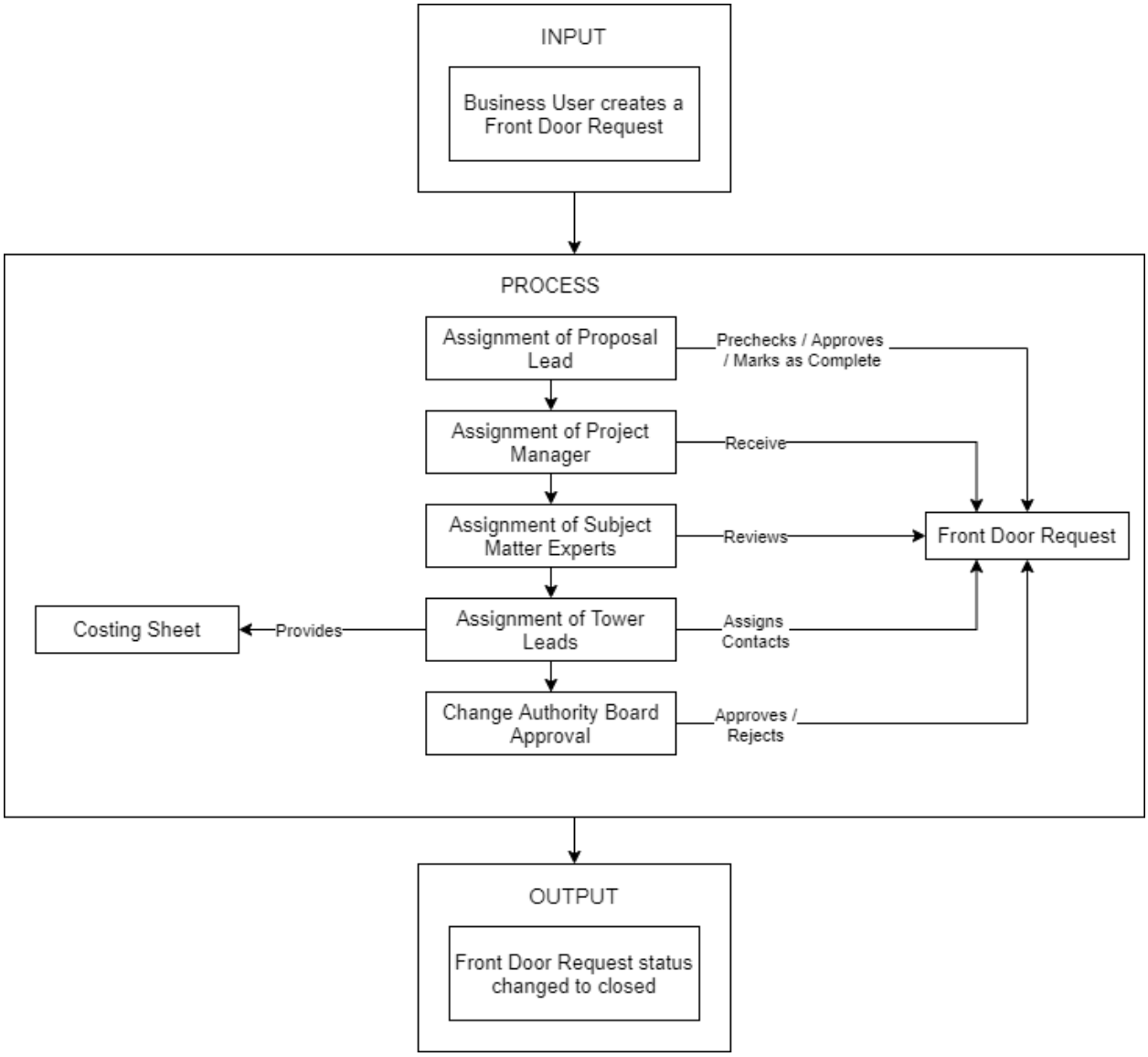
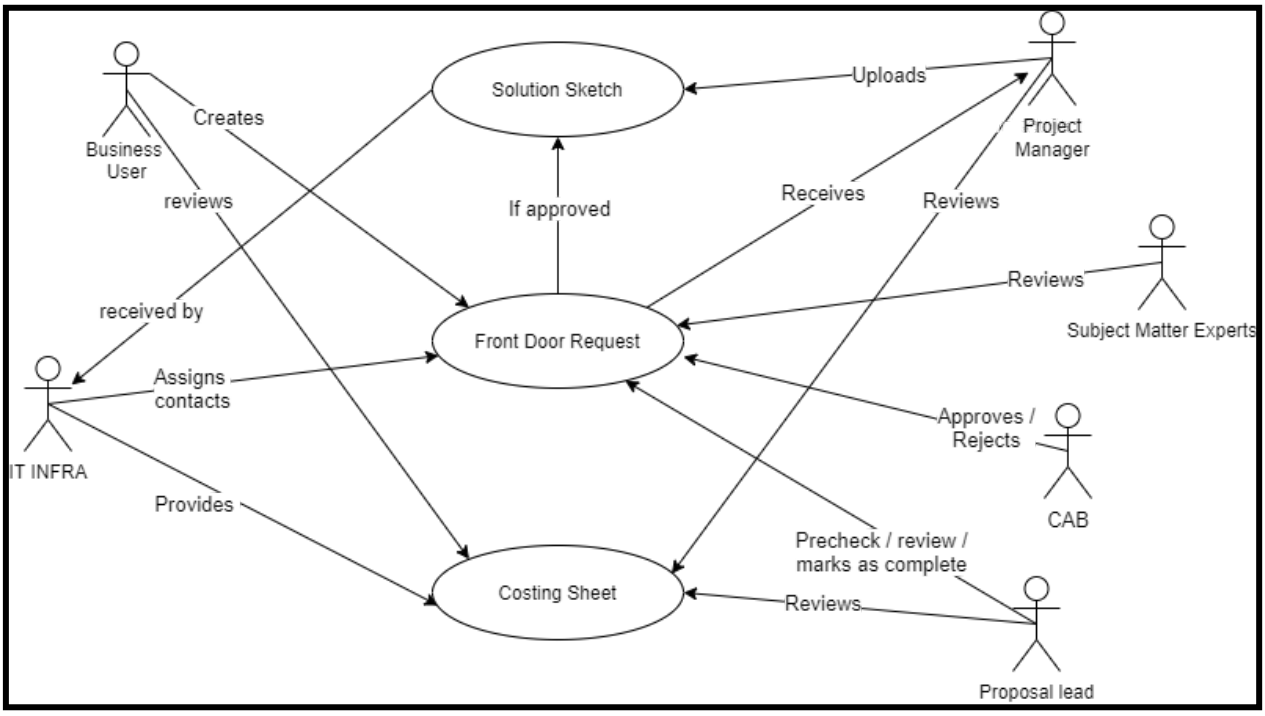


Fig 4.4. Data Flow Diagram

4.5 Use Case Diagram

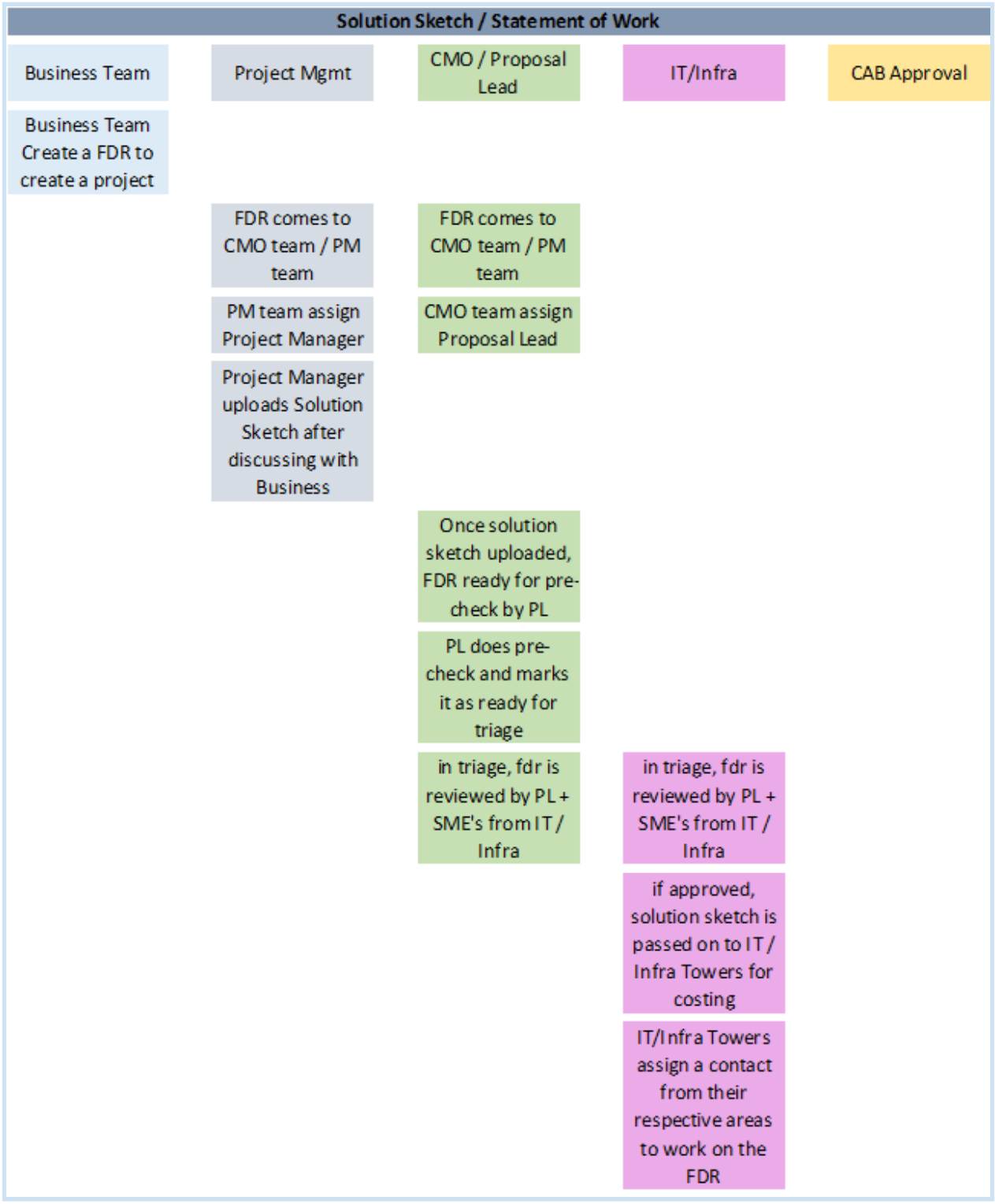


4.5. Use Case Diagram

Chapter 5

Implementation Details

5.1 Process Model



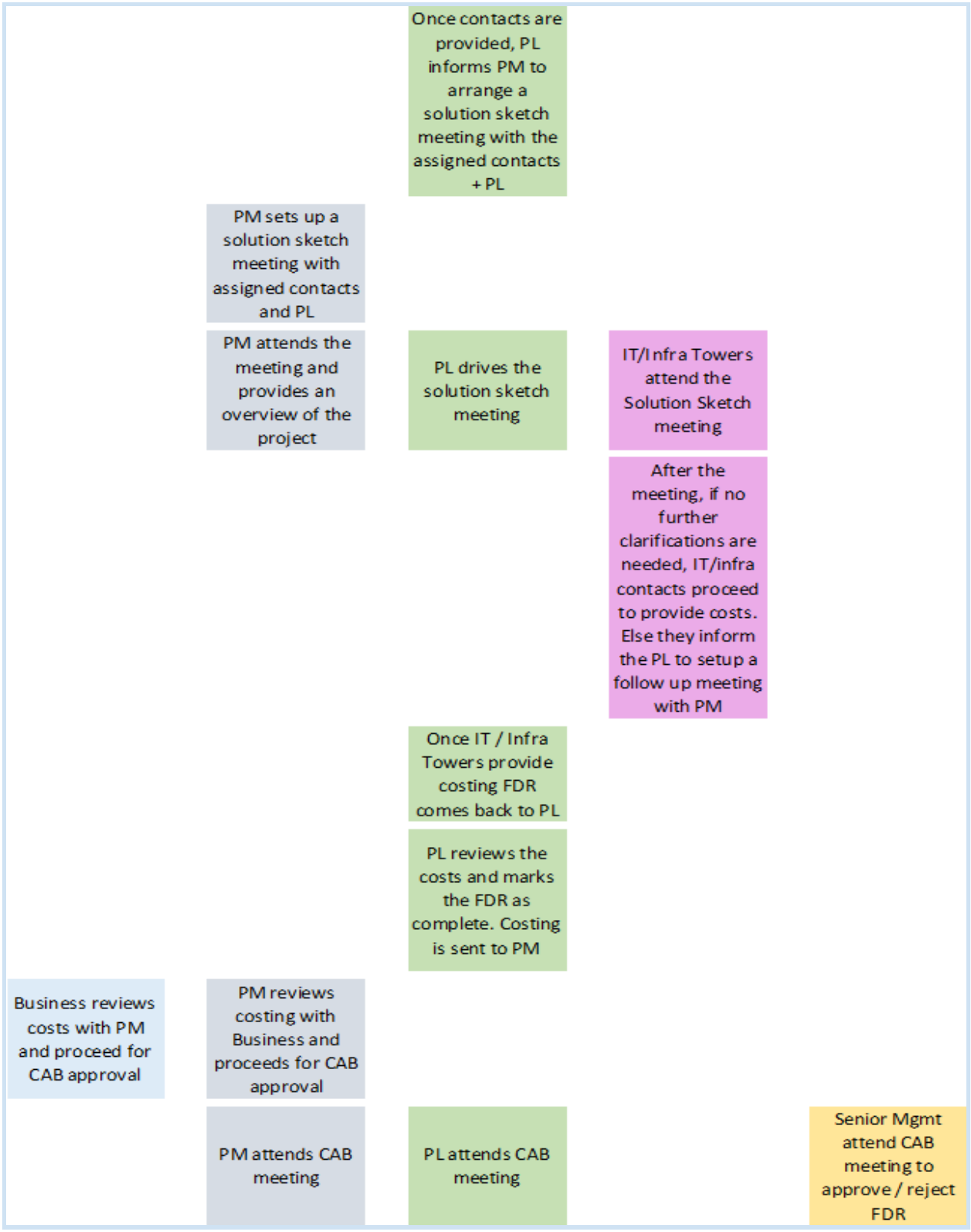


Fig 5.1 Process Model

5.2 Methodology

Introduction - Agile Methodology

Agile is the ability to create and respond to change. It is a way of dealing with, and ultimately succeeding in, an uncertain and turbulent environment.

A prime feature of this methodology is that it allows extreme agility in project requirements.

Key feature of this approach are –

1. Short term delivery cycles
2. Agile requirements
3. Less restrictive project control
4. Dynamic culture and real time communication.

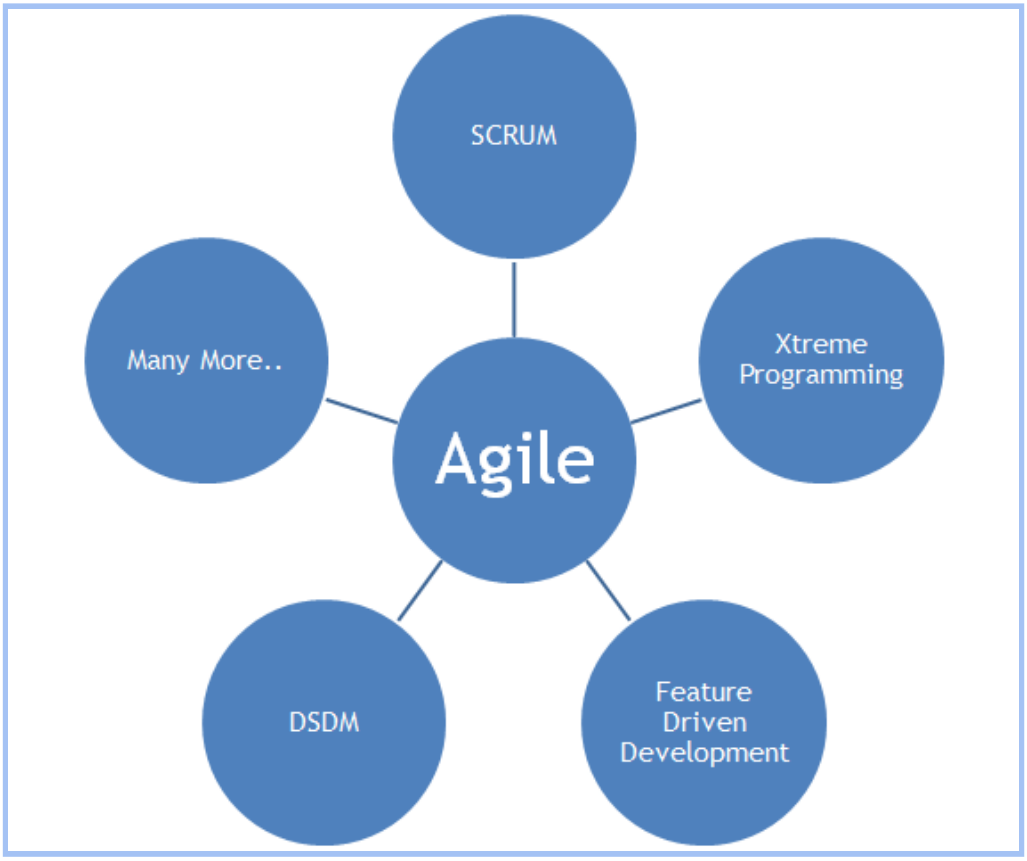


FIG 5.2.1 : AGILE FRAMEWORKS

SCRUM

- Increased uncertainty in projects can be due to
- Changing market conditions impacting requirements
- Advancement in technology
- Impossible to predict how the product should be developed

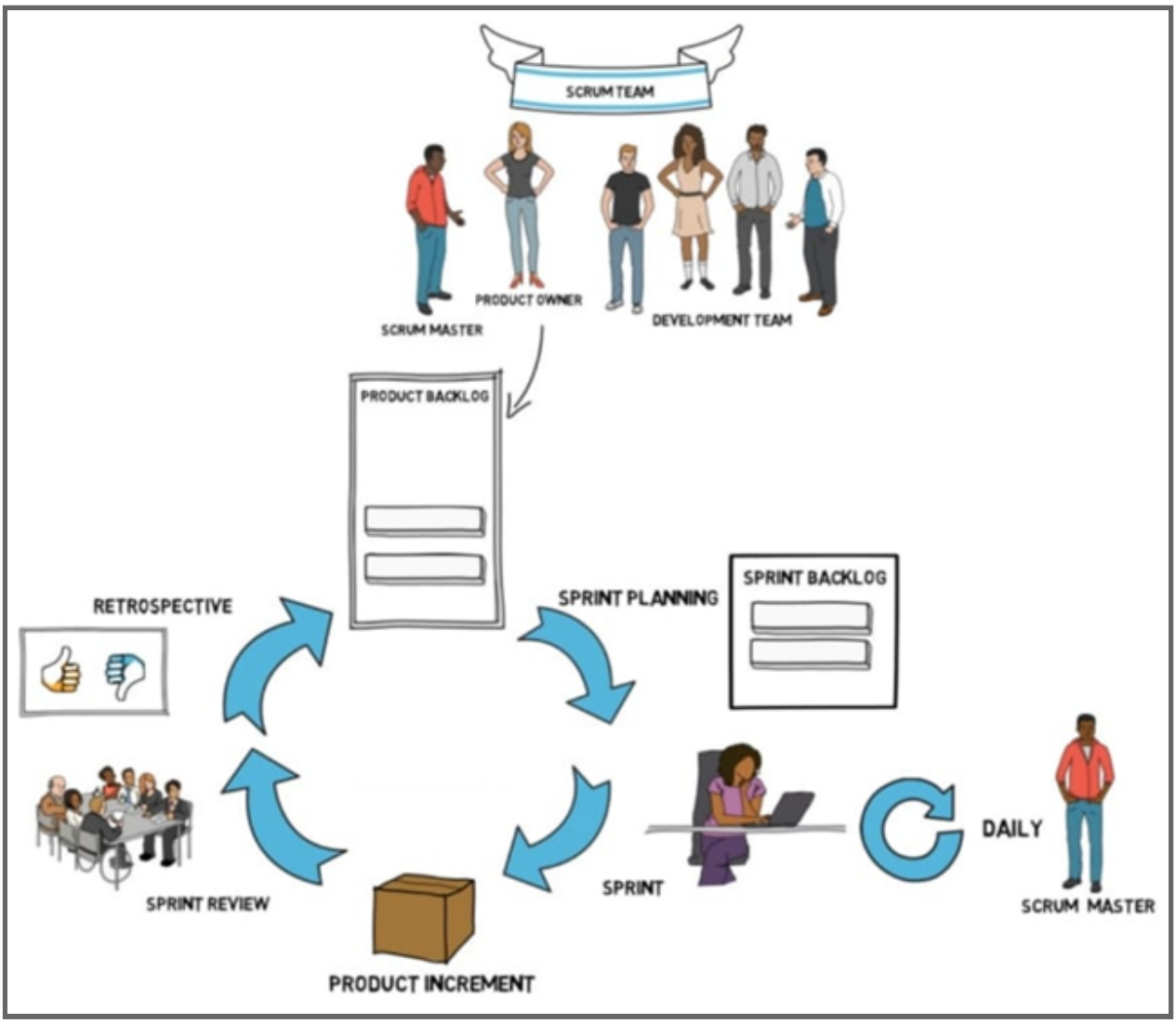


FIG 5.2.2 : OVERVIEW OF SCRUM

Chapter 6

Technology Used

6.1 Hardware Specification

1. Processor: Intel Core i7 10875H or AMD Ryzen 7 5800x or higher
2. System Memory: 16 gigabytes or higher
3. Disk Requirement: 256Gb of SSD for storing application
4. Database: A cloud-based database which can be scaled depending upon requirements

6.2 Software Requirements

1. Modern Web Browsers (Google Chrome, Microsoft Edge, Mozilla Firefox)
2. An operating system that supports modern web browsers
3. A web server for hosting the application

6.3 Programming Language Used

1. Python
2. MongoDB
3. HTML5
4. CSS3
5. JavaScript ES6
6. Jquery
7. Bootstrap

Chapter 7

Test Case Design

7.1 Unit testing

Unit testing emphasizes the verification effort on the smallest unit of software design i.e.; a software component or module. Unit testing is a dynamic method for verification, where the program is actually compiled and executed. Unit testing is performed in parallel with the coding phase. Unit testing tests units or modules not the whole software. We have tested each view/module of the application individually. As the modules were built up, testing was carried out simultaneously, tracking out each and every kind of input and checking the corresponding output until the module was working correctly.

Following is a sample of the test case :

```
from django.test import TestCase
from account.models import Account

class AccountTestCase(TestCase):
    def create_user(self):
         Account.objects.create(email="xyz@gmail.com", first_name="Asma",
last_name="Sitabkhan",

user_operating_company="IRA",user_designstion="BU",is_user_IRAIR="True",
     password="Asma")
                              Account.objects.create(email="xyz@gmail.com",
first_name="Furqaan", last_name="Mishra",

user_operating_company="SPA",user_designstion="BU",is_user_IRAIR="True",
     password="Asma")
```

```
from django.test import TestCase
from dashboard.models import TriageActionTracker

class TriageActionTrackerTestCase(TestCase):
        TriageActionTracker.objects.create(fdr_no=01,sr_no=7,
      action_tracker="Change fields",owners="Lee weatherly",
       recieved_date = "22-02-2019")
```

7.2 Integration testing

In integration testing a system consisting of different modules is tested for problems arising from component interaction. Integration testing should be developed from the system specification. Firstly, a minimum configuration must be integrated and tested. In our project we have done integration testing in a bottom up fashion i.e. in this project we have started construction and testing with atomic modules. After unit testing the modules are integrated one by one and then tested the system for problems arising from component interaction.

7.3 White Box Testing

White Box Testing (also known as Clear Box Testing, Open Box Testing, Glass Box Testing, Transparent Box Testing, Code-Based Testing or Structural Testing) is a software testing method in which the internal structure/design/implementation of the item being tested is known to the tester. The tester chooses inputs to exercise paths through the code and determines the appropriate outputs. Programming know-how and the implementation knowledge is essential. White box testing is testing beyond the user interface and into the nitty-gritty of a system. It is a Procedure to derive and/or select test cases based on an analysis of the internal structure of a component or system.This method is named so because the software program, in the eyes of the tester, is like a white/transparent box; inside which one clearly sees. For example a tester, usually a developer as well, studies the implementation code of a certain field on a webpage, determines all legal (valid and invalid) and illegal inputs and verifies the outputs against the expected outcomes, which is also determined by studying the implementation code.

White Box Testing method is applicable to the following levels of software testing:
- Unit Testing: For testing paths within a unit.
- Integration Testing: For testing paths between units.
- System Testing: For testing paths between subsystems.

Advantages of White Box Testing:
- Testing can be commenced at an earlier stage. One need not wait for the GUI to be available.
- Testing is more thorough, with the possibility of covering most paths.

7.4 Black Box Testing

Black Box Testing, also known as Behavioral Testing, is a software testing method in which the internal structure/design/implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usually functional. This method is named so because the software program, in the eyes of the tester, is like a black box; inside which one cannot see. It is a procedure to derive and/or select test cases based

on an analysis of the specification, either functional or non-functional, of a component or system without reference to its internal structure.

This method of attempts to find errors in the following categories:

- Incorrect or missing functions
- Interface errors
- Errors in data structures or external database access
- Behavior or performance errors
- Initialization and termination errors

Black Box testing method is applicable to all levels of the software testing process: Unit Testing, Integration Testing, System Testing and Acceptance Testing. The higher the level, and hence the bigger and more complex the box, the more black box testing method comes into use.

Tests are designed to answer the following questions:

- How is functional validity tested?
- How is system behavior and performance tested?
- What classes of input will make good test cases?
- Is the system particularly sensitive to certain input values?
- How are the boundaries of a data class isolated?
- What data rates and data volume can the system tolerate?
- What effect will specific combinations of data have on system operation?

By applying black-box techniques, we derive a set of test cases that satisfy the following Criteria

(1) test cases that reduce, by a count that is greater than one, the number of additional test cases that must be designed to achieve reasonable testing and

(2) test cases that tell us something about the presence or absence of classes of errors, rather than an error associated only with the specific test at hand.

7.4.1 Black Box Testing Techniques

7.4.1.1 Equivalence Partitioning

Equivalence partitioning is a black-box testing method that divides the input domain of a program into classes of data from which test cases can be derived. An ideal test case singlehandedly uncovers a class of errors (e.g., incorrect processing of all character data) that might otherwise require many cases to be executed before the general error is observed. Equivalence partitioning strives to define a test case that uncovers classes of errors, thereby reducing the total number of test cases that must be developed.

Test case design for equivalence partitioning is based on an evaluation of equivalence classes for an input condition. Using concepts introduced in the preceding section, if a set of objects can be linked by relationships that are symmetric, transitive, and reflexive, an equivalence class is present. An equivalence class represents a set of valid or invalid states for input conditions. Typically, an input condition is either a specific numeric value, a range of values, a set of related values, or a Boolean condition.

Equivalence classes may be defined according to the following guidelines:

1. If an input condition specifies a range, one valid and two invalid equivalence classes are defined.
2. If an input condition requires a specific value, one valid and two invalid equivalence classes are defined.
3. If an input condition specifies a member of a set, one valid and one invalid equivalence class is defined.
4. If an input condition is Boolean, one valid and one invalid class are defined.

As an example, consider data maintained as part of an automated banking application. The user can access the bank using a personal computer, provide a six-digit password, and follow with a series of typed commands that trigger various banking functions.

7.3.1.2 Boundary Value Analysis

Boundary value analysis leads to a selection of test cases that exercise bounding values. Boundary value analysis is a test case design technique that complements equivalence partitioning. Rather than selecting any element of an equivalence class, BVA leads to the selection of test cases at the "edges" of the class.

Guidelines for BVA are similar in many respects to those provided for equivalence partitioning:

1. If an input condition specifies a range bounded by values ‘a’ and ‘b’, test cases should be designed with values ‘a’ and ‘b’ and just above and just below and b.
2. If an input condition specifies a number of values, test cases should be developed that exercise the minimum and maximum numbers. Values just above and below minimum and maximum are also tested.
3. Apply guidelines 1 and 2 to output conditions. For example, assume that a temperature vs. pressure table is required as output from an engineering analysis program. Test cases should be designed to create an output report that produces the maximum (and minimum) allowable number of table entries.

Black Box Testing Advantages:

- Tests are done from a user’s point of view and will help in exposing discrepancies in the specifications.
- Testers need not know programming languages or how the software has been implemented.
- Tests can be conducted by a body independent from the developers, allowing for an objective perspective and the avoidance of developer-bias. Test cases can be designed as soon as the specifications are completed.

Test Case ID	Case Description	Steps to reproduce	Actual Result	Pass / Fail
TC_001	Signup / Login	-Open localhost making sure that the django server is on before running. -Use a valid and authentic username and password -Choose a valid operating company associated with the entered credentials. -Click on Login	-On successful authentication, redirection to the Dashboard of the respective user. -On putting Invalid Fields the user should be prompted with the message “Invalid Field”.	Pass
TC_002	Create an FDR	-Open Localhost . -Login with correct credentials of the Business User. - From the navigation menu “Create FDR”.	-On successful clicking you should be redirected to the FDR form.	Pass
TC_003	Update an FDR	-Open Localhost . -Login with correct credentials of the Business User. -Choose the desired FDR -Click on the FDR number	-On successful clicking you should be redirected to the FDR form.	Pass
TC_004	Create a Triage Element in the Triage Action Tracker	-Open Localhost . -Login with correct credentials of the Proposal Lead. -Click on Triage Action Tracker -Choose the desired FDR from the dropdown. -Select Appropriate dates -Add specific description. -Click Save	-Successfully added Triage element would be shown on the screen.	Pass
TC_005	Create a Change element for Solution Action Tracker	-Open Localhost . -Login with correct credentials of the Proposal Lead. -Click on Solution Sketch Action Tracker. -Choose the desired FDR from the dropdown. -Select Appropriate dates -Add specific description. -Click Save	-Successfully added Solution Sketch Action element would be shown on the screen.	Pass

TC_006	Login on Admin	-Open localhost making sure that the django server is on before running with admin extension. -Use a valid and authentic username and password -Choose a valid operating company associated with the entered credentials. -Click on Login	-On successful authentication, redirection to the Dashboard of the respective user. -On putting Invalid Fields the user should be prompted with the message “Invalid Field”	Pass
TC_007	Change Password	-Localhost login -Select forget password option present on the login screen. -Add a valid email id associated with that email ID. -Click send email	You will receive a password reset link on your registered email ID on which you can click and successfully add your new password	Pass
TC_008	Change Status of FDR	-Open Localhost . -Login with correct credentials of the Business User. - From the dashboard choose the desired FDR. -Click on the status dropdown -Click Save	From the present elements in the dropdown choose the status you wish to keep. After saving it the status should also change on the dashboard home screen	Pass
TC_009	Assign PM, PL	-Open Localhost . -Login with correct credentials of the PM/ PL - From the dashboard choose the desired FDR. -Click on the PM/PL field dropdown. -Choose the desired party -Click Save	To assign/ change PM / PL to the FDR that would be connected to FDR for the entire life cycle. After clicking save the PM/ PL name would be successfully displayed on the dashboard associated with FDR.	Pass
TC_010	Assign IT	-Open Localhost . -Login with correct credentials of the IT/ Infra SME’s - From the dashboard choose the desired FDR. -Click on the tower field dropdown. -Choose the desired party -Click Save	After clicking save the IT/ Infra representative name would be successfully displayed on the dashboard associated with FDR.	Pass

TC_011	Upload documents	-Open Localhost . -Login with correct credentials of the Business User. - From the dashboard choose the desired FDR. -Click on the upload solution sketch button - Add the desired file from your system -Click Ok -File uploaded and name would be visible -Click Save	The uploaded document would be added successfully. Verify it by clicking on the FDR again.	Pass

Chapter

8 Results

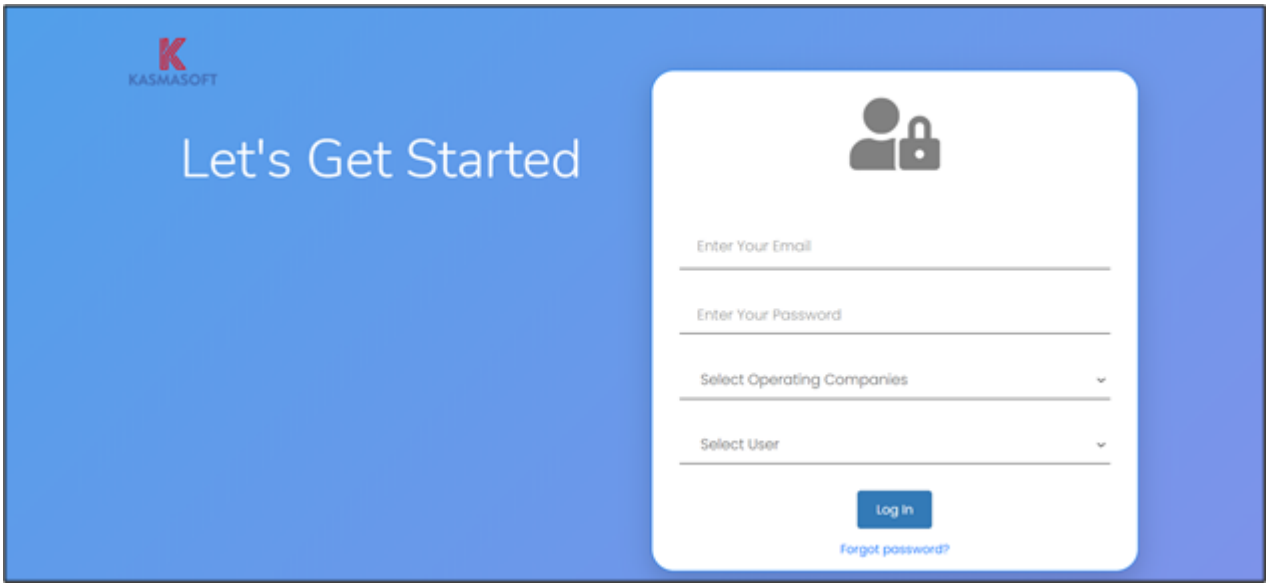


Figure 8.1: Login Page

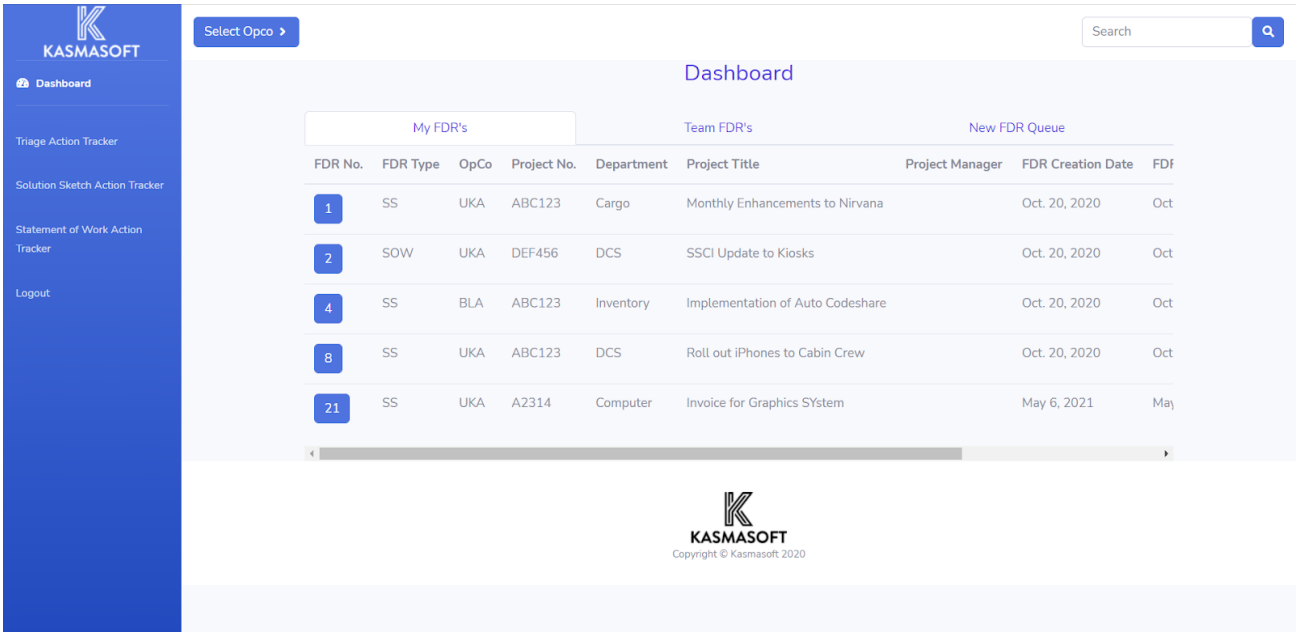


Figure 8.2: Dashboard

Figure 8.3: Creation of FDR

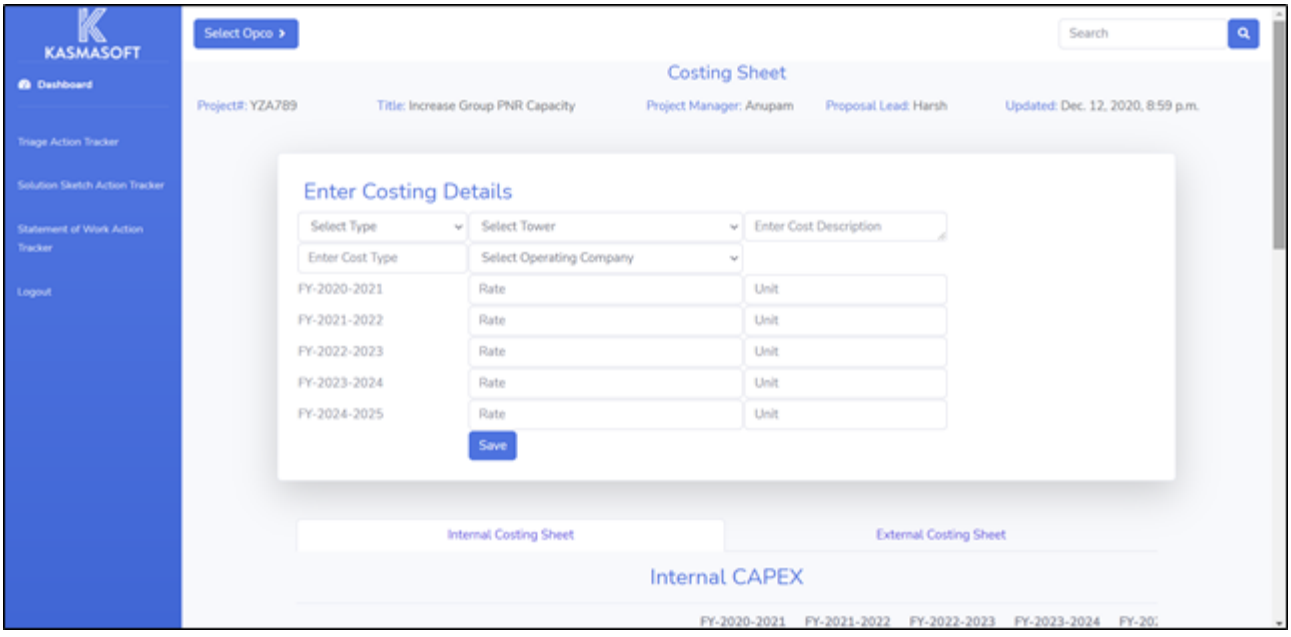


Figure 8.4: Costing Sheet

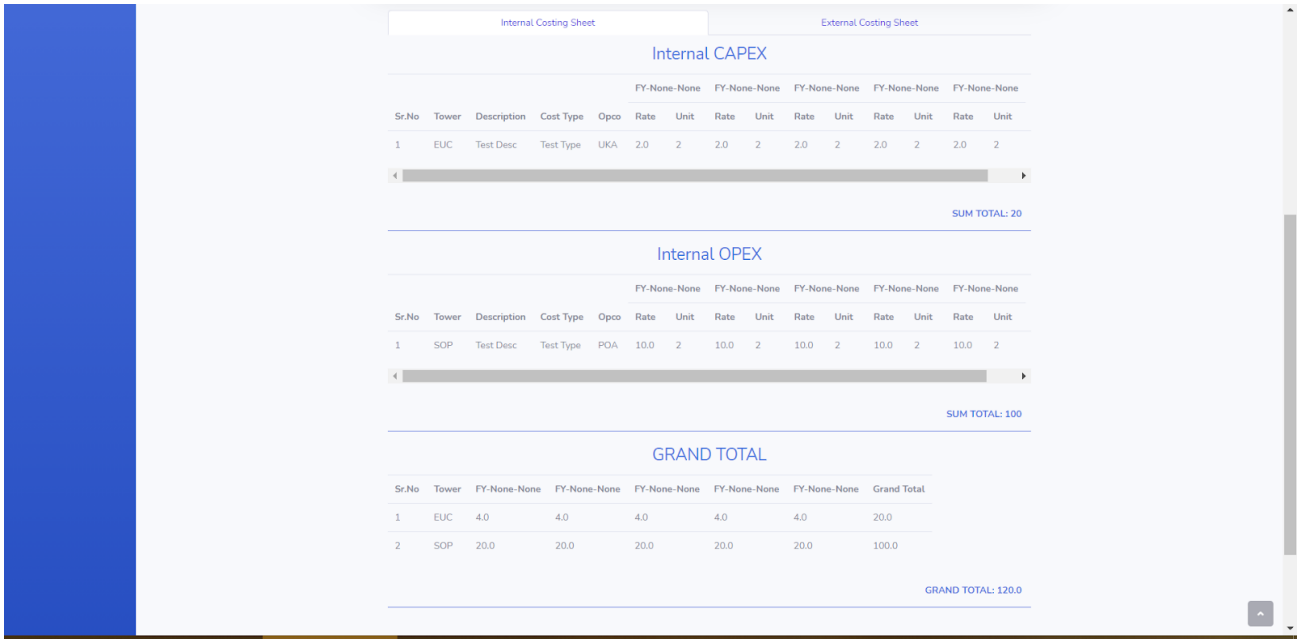


Figure 8.5: Internal Capex

Figure 8.6: External Capex

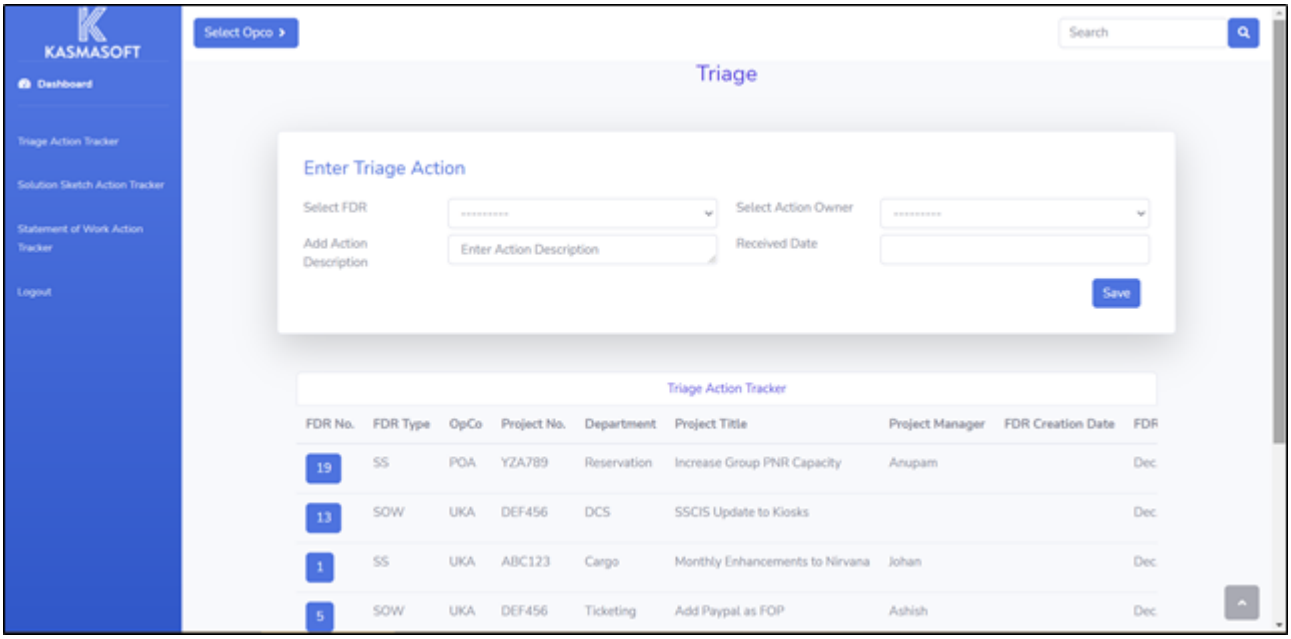


Figure 8.7: Triage Action Tracker

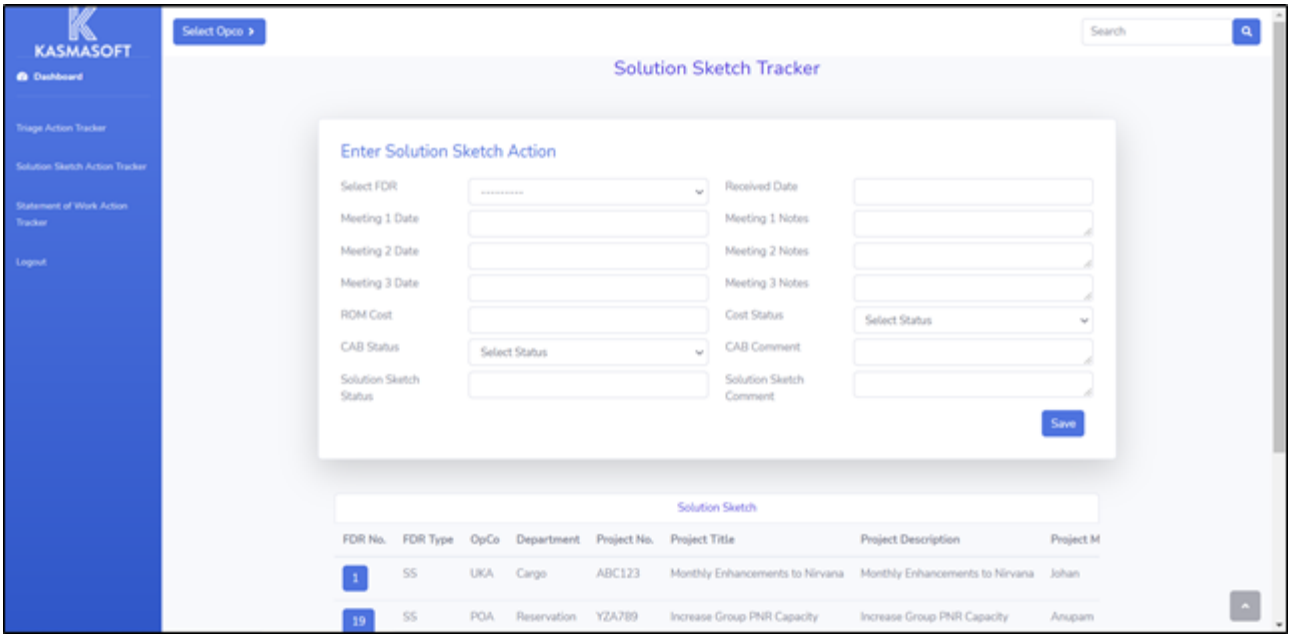


Figure : 8.8 Solution Sketch Tracker

Chapter 9

Project Timeline

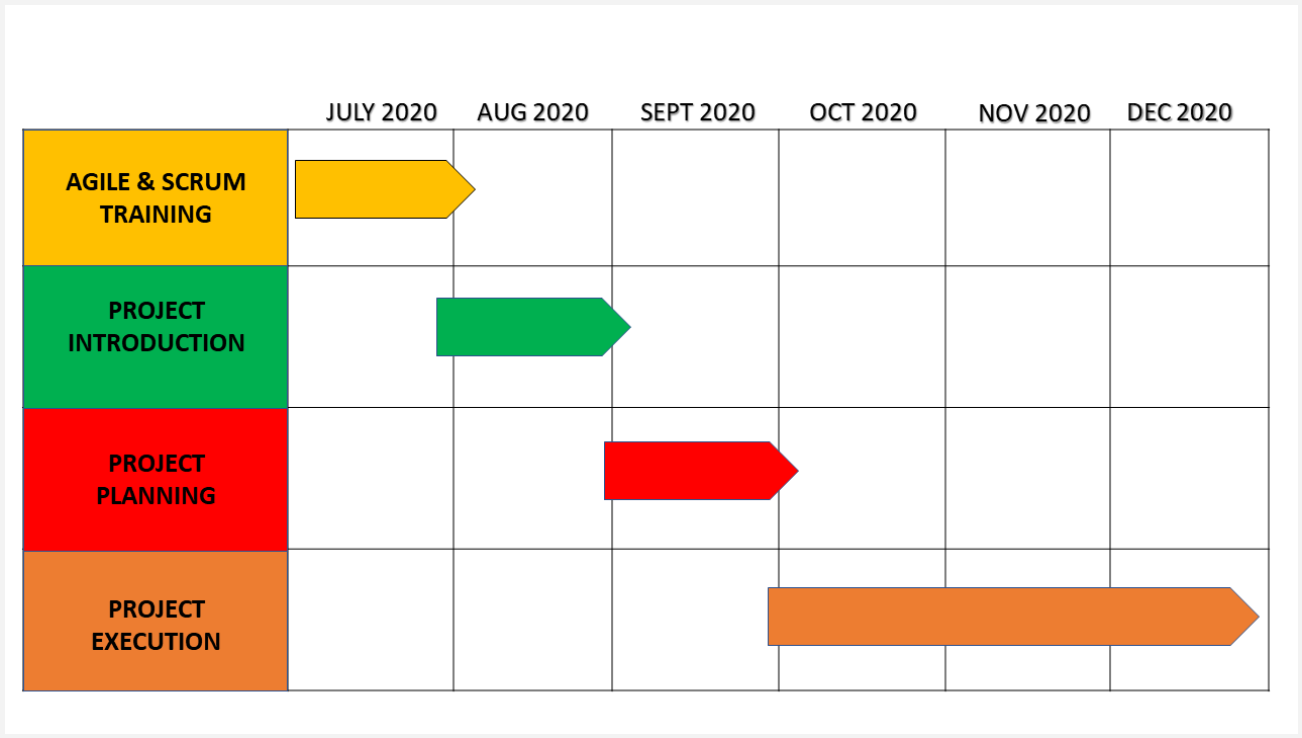


FIG 9.1 :TIMELINE CHART (JULY 2020 TO DECEMBER 2020)

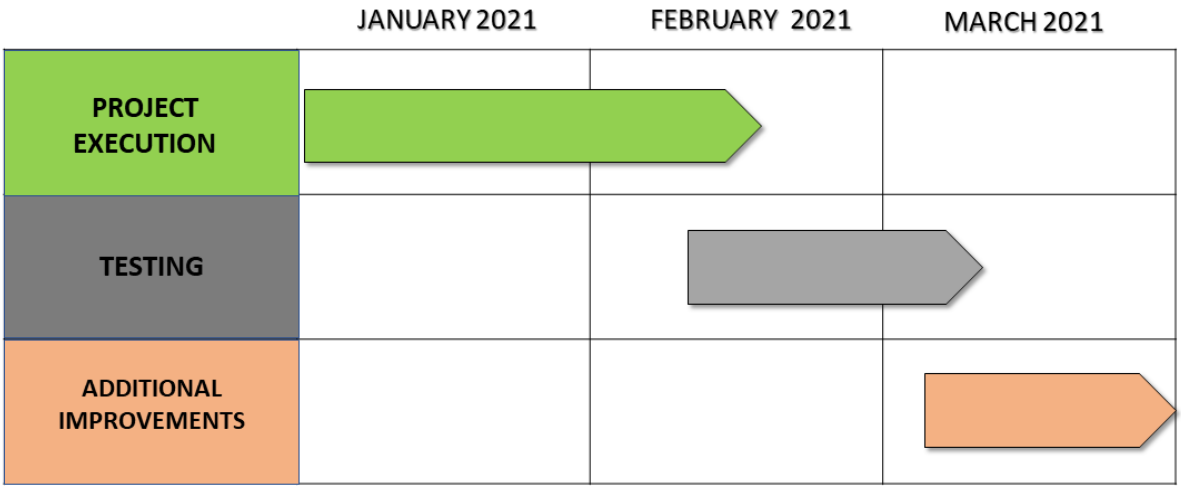


FIG 9.2: TIMELINE CHART (JANUARY 2021 TO MARCH 2021)

Chapter 10

Task Distribution

Sprint Breakdown Report

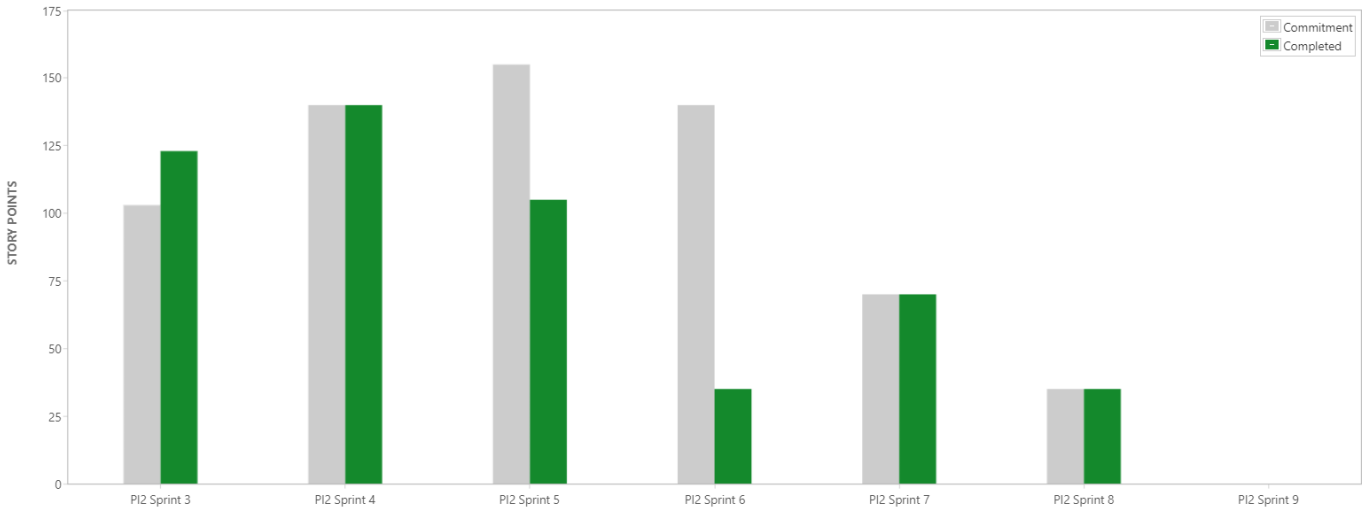


Fig: Breakdown Report

Sprint	Commitment	Completed
PI2 Sprint 3	103	123
PI2 Sprint 4	140	140
PI2 Sprint 5	155	105
PI2 Sprint 6	140	35
PI2 Sprint 7	70	70
PI2 Sprint 8	35	35
PI2 Sprint 9	0	0

Fig: Sprint Overview

The following table shows the overall task distribution of the project undertaken:

Task	Abuzar Shaikh	Shivam Tiwari	Needa Shaikh
Project Planning and Requirements Gathering			
Agile Training	✓	✓	✓
Jira Setup	✓	✓	✓
Survey of Existing System	✓	✓	✓
Study of Technologies	✓	✓	✓
System Designing	✓	✓	✓
Documentation	✓	✓	✓
Project Execution Phase			
Schema Designing	✓	✓	
Database setup	✓	✓	✓
UI Development			✓
BackEnd Development	✓	✓	✓
Integration	✓	✓	✓
Testing		✓	
Deployment	✓		
Documentation	✓	✓	✓

Chapter 11

Conclusion and Future Scope

11.1 Conclusion

We summarized the entire workflow of the functioning of the application and started implementing modules required throughout the application such as the account for logging into the system, the dashboard for displaying tables that are fetched from the database and action trackers for tracking of progress during projects registered into the application. We have implemented the tasks of assigning Project Manager and Proposal Lead for the FDR received for a particular project.The Action Tracker ,Solution Sketch and Statement of Work are also built where meeting notes, FDR creation date and FDR received dates are stored.

11.2 Future Scope

A mobile application of this automated ITSM system can be built having built the web application.

References

[1] Hendro Gunawan : Strategic Management for IT Services Using the Information Technology Infrastructure Library (ITIL) Framework

[2] Abdulazeez Ftahi, Abdul Hafeez-Baig, Raj Gururajan : Towards Effective Knowledge Application Capability in ITSM through Socialisation, Externalisation, Internalisation and Combination.

[3] Vipul Jain, O. P. Wali and V. Raveendra Saradhi **:** Information Technology Service Management [ITSM] Research: A Literature Review of Practices, Solutions and Measurement

[4] Anup Shrestha, Aileen Cater-Steel & Mark Toleman **:** Innovative decision support for IT service management .

[5] ALI YAZICI , ALOK MISHRA,PAUL KONTOGIORGIS : IT Service Management (ITSM) Education and Research: Global View.

[6] **Narges Shahsavarani,Shaobo Ji :** Research in Information Technology Service Management (ITSM) (2000 – 2010) : An Overview**.**

[7] J Iden, T. R. Eikebrokk : Implementing IT Service Management: A systematic literature review

[8] Jon Idena, Tom Roar Eikebrokk : IT Service Management Implementation.

[9] Thorsten Proehl, Koray Erek,Felix Limbach,Ruediger Zarnekow : Topics and Applied Theories in IT Service Management.

[10] Antti Lahtela, Marko Jantti, Jukka Kaukola Tieto : Implementing an ITIL-based IT Service Management Measurement System.

[11] Official Django Documentation, ,https://docs.djangoproject.com/en/3.1/

[12] Vitor Frietas, https://simpleisbetterthancomplex.com/series/beginners-guide/1.11/

Source Code

manage.py

```
#!/usr/bin/env python

"""Django's command-line utility for administrative tasks."""
import os
import sys


def main():
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'ITSM.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)


if __name__ == '__main__':
    main()
```

admin.py

```
from django.contrib import admin
from django import forms
from django.contrib.auth.models import Group
from django.contrib.auth.admin import UserAdmin as BaseUserAdmin
from django.contrib.auth.forms import ReadOnlyPasswordHashField
from .models import Account

# class Account_Admin(UserAdmin):
#     model=Account
#
list_display=('email','first_name','user_operating_company','user_designatio
n')
#     search_fields=('email',)
#     readonly_fields=()
#     ordering=()
#     filter_horizontal=()
#     list_filter=()
#     fieldsets=()


# admin.site.register(Account)
# Register your models here.

class UserCreationForm(forms.ModelForm):
    """A form for creating new users. Includes all the required
```

```
    fields, plus a repeated password."""
    password1 = forms.CharField(label='Password',
widget=forms.PasswordInput)
    password2 = forms.CharField(label='Password confirmation',
widget=forms.PasswordInput)

    class Meta:
        model = Account
        fields = ('email','first_name',
        'last_name',
        'user_operating_company',
        'user_designation',
        'is_user_UKAIR',
        'is_user_BELAIR',
        'is_user_POAIR',
        'is_user_SPANAIR',
        'is_user_IRAIR',
        'is_user_TT')

    def clean_password2(self):
        # Check that the two password entries match
        password1 = self.cleaned_data.get("password1")
        password2 = self.cleaned_data.get("password2")
        if password1 and password2 and password1 != password2:
            raise forms.ValidationError("Passwords don't match")
        return password2

    def save(self, commit=True):
        # Save the provided password in hashed format
        user = super().save(commit=False)
        user.set_password(self.cleaned_data["password1"])
        if commit:
            user.save()
        return user


class UserChangeForm(forms.ModelForm):
    """A form for updating users. Includes all the fields on
    the user, but replaces the password field with admin's
    password hash display field.
    """
    password = ReadOnlyPasswordHashField()

    class Meta:
        model = Account
        fields = ('email','first_name',
        'last_name',
        'user_operating_company',
        'user_designation',
        'is_user_UKAIR',
        'is_user_BELAIR',
        'is_user_POAIR',
        'is_user_SPANAIR',
        'is_user_IRAIR',
        'is_user_TT')

    def clean_password(self):
        # Regardless of what the user provides, return the initial value.
```

```
        # This is done here, rather than on the field, because the
        # field does not have access to the initial value
        return self.initial["password"]


class UserAdmin(BaseUserAdmin):
    # The forms to add and change user instances
    form = UserChangeForm
    add_form = UserCreationForm

    # The fields to be used in displaying the User model.
    # These override the definitions on the base UserAdmin
    # that reference specific fields on auth.User.
    list_display = ('email','first_name',
        'last_name',
        'user_operating_company',
        'user_designation',)
    search_fields=('email',)
    list_filter = ()
    fieldsets = (
        (None, {'fields': (
            'email',
            'password',
            'first_name',
            'last_name',
            'user_operating_company',
            'user_designation',
            'is_user_UKAIR',
            'is_user_BELAIR',
            'is_user_POAIR',
            'is_user_SPANAIR',
            'is_user_IRAIR',
            'is_user_TT'
                )
            }),
        # ('Personal info', {'fields': ('date_of_birth',)}),
        # ('Permissions', {'fields': ('is_admin',)}),
    )
    # add_fieldsets is not a standard ModelAdmin attribute. UserAdmin
    # overrides get_fieldsets to use this attribute when creating a user.
    add_fieldsets = (
        (None, {
            'classes': ('wide',),
            'fields': (
                'email',
                'first_name',
                'last_name',
                'user_operating_company',
                'user_designation',
                'password1',
                'password2','is_user_UKAIR',
                'is_user_BELAIR',
                'is_user_POAIR',
                'is_user_SPANAIR',
                'is_user_IRAIR',
                'is_user_TT'
            ),
        }),
```

```
    )
    search_fields = ('email',)
    ordering = ()
    filter_horizontal = ()


# Now register the new UserAdmin...
admin.site.register(Account, UserAdmin)
# ... and, since we're not using Django's built-in permissions,
# unregister the Group model from admin.
admin.site.unregister(Group)
```

backends.py

```
from django.contrib.auth.backends import BaseBackend
from django.http import request
from .models import Account
# from django_middleware_global_request.middleware import get_request

class CustomAuthenticate(BaseBackend):
    def authenticate(self, request, username=None, password=None, **kwargs):
        # self.db = db
        if username is None:
            username = kwargs.get(Account.USERNAME_FIELD)
        # print('---------->>>>>>> INSIDE AUTHENTICATION')
        try:
            user = Account.objects.get(email=username)
            # print('---------->>>>>>>', user, user.first_name)
            if user.check_password(password):
                # print('---------->>>>>>> PW Validated')
                return user
        except Account.DoesNotExist:
            return None

    def get_user(self, user_id):
        # print('---------->>>>>>> get_user')
        try:
            # print('---------->>>>>>> Inside try block\n---------->>>>>>>',
user_id)
            user =  Account.objects.get(pk=user_id)
        except Account.DoesNotExist:
            return None
        return user if self.user_can_authenticate(user) else None

    def user_can_authenticate(self, user):
        is_active = getattr(user, 'is_active', None)
        return is_active or is_active is None
```

forms.py

```
from django import forms
from django.contrib.auth.forms import AuthenticationForm
from django.db.models.enums import Choices
from .models import Account


class New_Login_Form(AuthenticationForm):
    user_operating_company =
```

```
forms.ChoiceField(choices=Account.Operating_companies)
    user_designation = forms.ChoiceField(choices=Account.Designations)
```

models.py

```
from django.db import models
from django.contrib.auth.models import AbstractBaseUser, BaseUserManager


class MyAccountManager(BaseUserManager):
    def create_user(
        self,
        email,
        first_name,
        last_name,
        user_operating_company,
        user_designation,
        is_user_UKAIR,
        is_user_BELAIR,
        is_user_POAIR,
        is_user_SPANAIR,
        is_user_IRAIR,
        is_user_TT,
        password=None
        ):
        if not email:
            raise ValueError('Users must have an email address')
        if not first_name:
            raise ValueError('Users must have a First Name')
        if not last_name:
            raise ValueError('Users must have a Last Name')
        if not user_operating_company:
            raise ValueError('Users must have a Operating Company')
        if not user_designation:
            raise ValueError('Users must have a Designations')

        user = self.model(
            email = self.normalize_email(email),
            first_name = first_name.lower(),
            last_name = last_name.lower(),
            user_operating_company = user_operating_company,
            user_designation = user_designation,
            is_user_UKAIR = is_user_UKAIR,
            is_user_BELAIR = is_user_BELAIR,
            is_user_POAIR = is_user_POAIR,
            is_user_SPANAIR = is_user_SPANAIR,
            is_user_IRAIR = is_user_IRAIR,
            is_user_TT = is_user_TT
            )

        user.set_password(password)
        user.save(using=self._db)
        return user

    def create_superuser(
        self,
        email,
```

```
        password,
        first_name,
        last_name,
        user_operating_company,
        user_designation,
        is_user_UKAIR,
        is_user_BELAIR,
        is_user_POAIR,
        is_user_SPANAIR,
        is_user_IRAIR,
        is_user_TT
        ):
        user = self.create_user(
            email=self.normalize_email(email),
            password=password,
            first_name=first_name.lower(),
            last_name=last_name.lower(),
            user_operating_company=user_operating_company,
            user_designation=user_designation,
            is_user_UKAIR = True,
            is_user_BELAIR = True,
            is_user_POAIR = True,
            is_user_SPANAIR = True,
            is_user_IRAIR = True,
            is_user_TT = True
        )
        user.is_superuser = True
        user.is_admin = True
        user.is_staff = True
        user.save(using=self._db)
        return user


class Account(AbstractBaseUser):

    Operating_companies=[
        ('','Select Operating Companies'),
        ('Belgian_air_db','Belgian Airways'),
        ('UK_air_db','UK Airways'),
        ('Portugal_air_db','Air Portugal'),
        ('Irish_air_db','Irish Air'),
        ('Spanish_air_db','Spanish Air'),
        ('Tag_tech_db','TAG Tech'),
    ]

    operating_companies_bool = {
        'Belgian_air_db': 'is_user_BELAIR',
        'UK_air_db': 'is_user_UKAIR',
        'Portugal_air_db': 'is_user_POAIR',
        'Irish_air_db': 'is_user_IRAIR',
        'Spanish_air_db': 'is_user_SPANAIR',
        'Tag_tech_db': 'is_user_TT',
    }

    Designations=[
        ('','Select User'),
        ('BU','Business User'),
        ('PL','Proposal Lead'),
```

```
        ('PM','Project Manager'),
        ('II','IT/Infra'),
    ]
    email                   = models.EmailField(verbose_name="email",
max_length=60, unique=True)
    first_name              = models.CharField(max_length= 255)
    last_name               = models.CharField(max_length=255)
    user_operating_company  = models.CharField(max_length=255,
choices=Operating_companies)
    user_designation        = models.CharField(max_length=2,
choices=Designations)
    date_joined             = models.DateTimeField(verbose_name='date
joined', auto_now_add=True)
    last_login              = models.DateTimeField(verbose_name='last
login', auto_now=True)
    is_admin                = models.BooleanField(default=False)
    is_active               = models.BooleanField(default=True)
    is_staff                = models.BooleanField(default=False)
    is_superuser            = models.BooleanField(default=False)
    is_user_UKAIR           = models.BooleanField(default=False)
    is_user_BELAIR          = models.BooleanField(default=False)
    is_user_POAIR           = models.BooleanField(default=False)
    is_user_SPANAIR         = models.BooleanField(default=False)
    is_user_IRAIR           = models.BooleanField(default=False)
    is_user_TT              = models.BooleanField(default=False)


    USERNAME_FIELD = 'email'
    REQUIRED_FIELDS = [
                    'first_name',
                    'last_name',
                    'user_operating_company',
                    'user_designation',
                    'is_user_UKAIR',
                    'is_user_BELAIR',
                    'is_user_POAIR',
                    'is_user_SPANAIR',
                    'is_user_IRAIR',
                    'is_user_TT'
        ]

    objects = MyAccountManager()

    def __str__(self):
        return self.email

    # For checking permissions. to keep it simple all admin have ALL
permissons
    def has_perm(self, perm, obj=None):
        return self.is_admin

    # Does this user have permission to view this app? (ALWAYS YES FOR
SIMPLICITY)
    def has_module_perms(self, app_label):
        return True

    def user_matches_opco(self, opco):
        mask = self.operating_companies_bool.get(opco, False)
```

```
        if mask:
            return getattr(self, mask)
        return False
```

urls.py

```
from django.urls import path, include
from django.contrib.auth import views as auth_views
from . import views

urlpatterns=[
    path('', views.HomeView.as_view(), name='home'),
    path('logout/',views.logout_view,name='logout'),
    path('login/', views.login_authentication, name='login'),
    path('switch-account/<str:opco>', views.switch_account,
name='switch-account'),
    path(
        'reset-password/',

auth_views.PasswordResetView.as_view(template_name='password_reset/password_
reset.html'),
        name='reset_password'
    ),
    path(
        'reset-password-sent/',

auth_views.PasswordResetDoneView.as_view(template_name='password_reset/passw
ord_reset_sent.html'),
        name='password_reset_done'
    ),
    path(
        'reset/<uidb64>/<token>',

auth_views.PasswordResetConfirmView.as_view(template_name='password_reset/pa
ssword_reset_form.html'),
        name='password_reset_confirm'
    ),
    path(
        'reset-password-complete/',

auth_views.PasswordResetCompleteView.as_view(template_name='password_reset/p
assword_reset_done.html'),
        name='password_reset_complete'
    ),
]
```

views.py

```
import django
from django.views.generic import View
from django.shortcuts import render, redirect
from django.http import HttpResponse
from .forms import New_Login_Form
from django.contrib import auth
from django.contrib.auth import login, authenticate, logout
from django.contrib.auth.decorators import login_required

# Create your views here.
```

```
class HomeView(View):
    def post(self,request):
        pass

    def get(self,request):
        form = New_Login_Form()
        return render(request, 'index.html', {'form': form})

def login_authentication(request):
    if request.method == "POST":
        # print('---------->>>>>>> inside if')
        # print('---------->>>>>>>', request.POST)
        form = New_Login_Form(data=request.POST)
        print('---------->>>>>>>', form.errors)
        print('---------->>>>>>>', form.is_valid())
        if form.is_valid():
            print('---------->>>>>>> Inside form')
            email = request.POST['username']
            password = request.POST['password']
            user_opco = request.POST['user_operating_company']
            user_desig = request.POST['user_designation']
            user = authenticate(request, email=email, password=password)
            if user.user_matches_opco(user_opco) and user_desig ==
user.user_designation:
                if user:
                    request.session['user_opco'] = user_opco
                    request.session['user_designation']=user_desig
                    print(f'\n{user_opco}\n')
                    login(request, user)
                    return redirect('dashboard')
            else:
                form = New_Login_Form()
                return render(request, 'index.html', {'form': form, 'err':
'Invalid Fields'})
        else:
            form = New_Login_Form()
            return render(request, 'index.html', {'form': form})
    else:
        form = New_Login_Form()
        return render(request, 'index.html', {'form': form})

@login_required(login_url='home')
def logout_view(request):
    logout(request)
    return redirect('home')

@login_required(login_url='home')
def switch_account(request, opco, backend =
'django.contrib.auth.backends.ModelBackend'):

    request.session['user_opco'] = opco
    # if(desig != 'NULL'):
    #     request.session['user_designation']=desig

    # Abuzar Changes
    if request.session['user_designation'] == 'PL':
        request.session['user_designation'] = 'PM'
```

```
        # user.user_designation='PM'
        # request.session.modified = True

    if opco == 'Tag_tech_db' and request.session['user_designation'] ==
'PM':
        request.session['user_designation'] = 'PL'
        # user.user_designation='PL'
        # request.session.modified = True

    print('[INFO] SESSION DATA USER OPCO', request.session['user_opco'])
    print('[INFO] SESSION DATA USER DESIGNATION',
request.session['user_designation'])

    # logout(request)
    # user = authenticate(request, email=email, password=password)
    # if user_opco == user.user_operating_company and user_desig ==
user.user_designation:
    #     if user:
    #         login(request, user)
    #         return redirect('dashboard')
    # else:
    #     form = New_Login_Form()
    #     return render(request, 'index.html', {'form': form, 'err':
'Invalid Fields'})
    # if request.user:
    #     login(request, request.user,
backend='django.contrib.auth.backends.ModelBackend')
    #     return redirect('dashboard')
    # else:
    #     return render(request, 'test.html')

    return redirect('dashboard')
```

operations.py

```
from django import template

register = template.Library()

def multiply(value, *args):
    ans = 1
    for i in args:
        ans *= i
    return value * ans

@register.simple_tag(name='addition')
def addition(value, *args):
    return value + sum(args)

@register.simple_tag(name='special_add')
def special_addition(rate_1, unit_1, rate_2, unit_2, rate_3, unit_3, rate_4,
unit_4, rate_5, unit_5):
    return
((rate_1*unit_1)+(rate_2*unit_2)+(rate_3*unit_3)+(rate_4*unit_4)+(rate_5*uni
t_5))
```

```
@register.simple_tag(takes_context=True, name='set')
def set(context, key, value):
    context.dicts[0][key] = value
    return ''

# @register.simple_tag(name='slice')
# def slice(string, str2):
#     return list(str(string).split(':'))[0]
```

admin.py

```
from django.contrib import admin
from .models import CostTable, Cost_sheet, FDR, Financial_year_rate,
TriageActionTracker, SolutionSketchActionTracker
# Register your models here.
class MultiDBModelAdmin(admin.ModelAdmin):
    # A handy constant for the name of the alternate database.
    using = 'central_db'

    def save_model(self, request, obj, form, change):
        # Tell Django to save objects to the 'other' database.
        obj.save(using=self.using)

    def delete_model(self, request, obj):
        # Tell Django to delete objects from the 'other' database
        obj.delete(using=self.using)

    def get_queryset(self, request):
        # Tell Django to look for objects on the 'other' database.
        return super().get_queryset(request).using(self.using)

    def formfield_for_foreignkey(self, db_field, request, **kwargs):
        # Tell Django to populate ForeignKey widgets using a query
        # on the 'other' database.
        return super().formfield_for_foreignkey(db_field, request,
using=self.using, **kwargs)

    def formfield_for_manytomany(self, db_field, request, **kwargs):
        # Tell Django to populate ManyToMany widgets using a query
        # on the 'other' database.
        return super().formfield_for_manytomany(db_field, request,
using=self.using, **kwargs)

admin.site.register(FDR, MultiDBModelAdmin)
admin.site.register(Cost_sheet,MultiDBModelAdmin)
admin.site.register(CostTable,MultiDBModelAdmin)
admin.site.register(TriageActionTracker, MultiDBModelAdmin)
admin.site.register(SolutionSketchActionTracker, MultiDBModelAdmin)
```

forms.py

```
from django import forms
from .models import FDR, CostTable, Tower,
OPERATING_COMPANY,TriageActionTracker,\
    SolutionSketchActionTracker, StatementOfWorkActionTracker
from accounts.models import Account

class NewFDRForm(forms.ModelForm):
```

```
    class Meta:
        x=Account.objects.all().filter(user_designation ='PM')
        pm_name=tuple([(i.first_name, i.first_name) for i in x])
        model=FDR
        project_manager=forms.ChoiceField(choices=pm_name)
        widgets = {
            'project_description': forms.Textarea(attrs={'rows':4,
'cols':15}),
        }
        fields=[
            'fdr_no',
            # 'fdr_status',
            'fdr_type','opco','project_no',
            'department','project_title','project_description',
            'project_manager',
            'proposal_lead','pl_comment','pl_status',
            'tower_1','comment_t1','status_t1',
            'tower_2','comment_t2','status_t2',
            'tower_3','comment_t3','status_t3',
            'tower_4','comment_t4','status_t4',
            'tower_5','comment_t5','status_t5',
            'tower_6','comment_t6','status_t6',
            'image'
        ]
class CostTableForm(forms.ModelForm):
    form_types = (
            ('','Select Type'),
            ('IC', 'Internal Capex'),
            ('IO', 'Internal Opex'),
            ('EC', 'External Capex'),
            ('EO', 'External Opex'),
        )
    form_type = forms.ChoiceField(choices=form_types)
    class Meta:
        model = CostTable
        tower = forms.ChoiceField(choices=Tower)
        is_capex = forms.BooleanField()
        is_internal = forms.BooleanField()
        is_external = forms.BooleanField()
        is_opex = forms.BooleanField()
        opco = forms.ChoiceField(choices=OPERATING_COMPANY)
        widgets = {
            'cost_description': forms.Textarea(attrs={'rows': 1, 'cols':
25}),

        }
        fields = [
            'tower',
            'cost_description',
            'form_type',
            'cost_type',
            'opco',
            'is_capex',
            'is_internal',
            'is_opex',
            'is_external',
            'rate_1',
            'unit_1',
```

```
            'rate_2',
            'unit_2',
            'rate_3',
            'unit_3',
            'rate_4',
            'unit_4',
            'rate_5',
            'unit_5',
        ]
    def __init__(self, *args, **kwargs):
        initial = kwargs.get('initial', {})
        initial['is_capex']=False
        initial['is_opex']=False
        initial['is_internal']=False
        initial['is_external']=False
        kwargs['initial']=initial
        super(CostTableForm, self).__init__(*args, **kwargs)

class DateInput(forms.DateInput):
    input_type = 'date'

class TriageActionTrackerForm(forms.ModelForm):
    # x=list(FDR.objects.all().using('central_db'))
    # print(x)

fdr_no=forms.ModelChoiceField(queryset=FDR.objects.all().using('central_db')
)
    received_date = forms.DateTimeField(input_formats=['%d/%m/%Y'])
    class Meta:
        model= TriageActionTracker
        widgets = {
            'action_tracker': forms.Textarea(attrs={'rows': 1, 'cols': 25}),
        }
        fields=[
            'fdr_no',
            'action_owner',
            'action_tracker',
            'received_date',
        ]

class SolutionSketchActionTrackerForm(forms.ModelForm):
    cost_status_choices=(
        ('','Select Status'),
        ('Provided','Cost Provided'),
        ('Pending','Cost Pending'),
    )
    cab_status_choices=(
        ('','Select Status'),
        ('Pending','Pending'),
        ('Approved','Approved'),
        ('Reject','Reject'),
    )

fdr_no=forms.ModelChoiceField(queryset=FDR.objects.all().using('central_db')
)
    received_date = forms.DateTimeField(input_formats=['%d/%m/%Y'])
    meeting_1_date = forms.DateTimeField(input_formats=['%d/%m/%Y'])
    meeting_2_date = forms.DateTimeField(input_formats=['%d/%m/%Y'])
```

```
    meeting_3_date = forms.DateTimeField(input_formats=['%d/%m/%Y'])
    cost_status = forms.ChoiceField(choices=cost_status_choices)
    CAB_status = forms.ChoiceField(choices=cab_status_choices)
    class Meta:
        model = SolutionSketchActionTracker
        widgets = {
            'meeting_1_notes': forms.Textarea(attrs={'rows': 1, 'cols':
25}),
            'meeting_2_notes': forms.Textarea(attrs={'rows': 1, 'cols':
25}),
            'meeting_3_notes': forms.Textarea(attrs={'rows': 1, 'cols':
25}),
            'CAB_comment': forms.Textarea(attrs={'rows': 1, 'cols': 25}),
            'SS_comment': forms.Textarea(attrs={'rows': 1, 'cols': 25}),
        }
        fields=[
            'fdr_no',
            'received_date',
            'meeting_1_notes',
            'meeting_2_notes',
            'meeting_3_notes',
            'meeting_1_date',
            'meeting_2_date',
            'meeting_3_date',
            'ROM_cost',
            'cost_status',
            'CAB_status',
            'CAB_comment',
            'SS_status',
            'SS_comment',
        ]

    def __init__(self, *args, **kwargs):
        super(SolutionSketchActionTrackerForm, self).__init__(*args,
**kwargs)
        self.fields['meeting_2_date'].required = False
        self.fields['meeting_3_date'].required = False
        self.fields['meeting_2_notes'].required = False
        self.fields['meeting_3_notes'].required = False


class StatementOfWorkActionTrackerForm(forms.ModelForm):
    cost_status_choices=(
        ('','Select Status'),
        ('Provided','Cost Provided'),
        ('Pending','Cost Pending'),
    )
    cab_status_choices=(
        ('','Select Status'),
        ('Pending','Pending'),
        ('Approved','Approved'),
        ('Reject','Reject'),
    )

fdr_no=forms.ModelChoiceField(queryset=FDR.objects.all().using('central_db')
)
    received_date = forms.DateTimeField(input_formats=['%d/%m/%Y'])
    meeting_1_date = forms.DateTimeField(input_formats=['%d/%m/%Y'])
```

```
    meeting_2_date = forms.DateTimeField(input_formats=['%d/%m/%Y'])
    meeting_3_date = forms.DateTimeField(input_formats=['%d/%m/%Y'])
    cost_status = forms.ChoiceField(choices=cost_status_choices)
    CAB_status = forms.ChoiceField(choices=cab_status_choices)
    class Meta:
        model = StatementOfWorkActionTracker
        widgets = {
            'meeting_1_notes': forms.Textarea(attrs={'rows': 1, 'cols':
25}),
            'meeting_2_notes': forms.Textarea(attrs={'rows': 1, 'cols':
25}),
            'meeting_3_notes': forms.Textarea(attrs={'rows': 1, 'cols':
25}),
            'CAB_comment': forms.Textarea(attrs={'rows': 1, 'cols': 25}),
            'SOW_comment': forms.Textarea(attrs={'rows': 1, 'cols': 25}),
        }
        fields=[
            'fdr_no',
            'received_date',
            'meeting_1_notes',
            'meeting_2_notes',
            'meeting_3_notes',
            'meeting_1_date',
            'meeting_2_date',
            'meeting_3_date',
            'committed_cost',
            'cost_status',
            'CAB_status',
            'CAB_comment',
            'SOW_status',
            'SOW_comment',
        ]

    def __init__(self, *args, **kwargs):
        super(StatementOfWorkActionTrackerForm, self).__init__(*args,
**kwargs)
        self.fields['meeting_2_date'].required = False
        self.fields['meeting_3_date'].required = False
        self.fields['meeting_2_notes'].required = False
        self.fields['meeting_3_notes'].required = False
```

models.py

```
from django.db.models.deletion import CASCADE
from djongo import models
from django.db import transaction
import datetime
from accounts.models import Account
from django.contrib import admin

# Choice Variables
OPERATING_COMPANY = (
        ('','Select Operating Company'),
        ('UKA','UK Air'),
        ('IRA','Irish Air'),
        ('SPA','Spansih Air'),
        ('BLA','Belgian Air'),
        ('POA','Air Portugal'),
```

```
    )
Tower = (
    ('', 'Select Tower'),
    ('EUC','End User Computing'),
    ('NET','Networks'),
    ('SOP','Security Operations'),
    ('SDC','Servers & Data Centers'),
    ('ADM','Application Development & Maintainence'),
    ('MIS','Miscellaneous Infrstructure'),
)

users = Account.objects.all().using('default')
class FDR(models.Model):
    fdr_no = models.AutoField(primary_key=True)

    FDR_STATUS=(
        ('','Select FDR Status'),
        ('O','Open'),
        ('C','Closed'),
    )
    fdr_status=models.CharField(max_length=1,choices=FDR_STATUS,default='O')

    date = models.DateField(auto_now_add=True)

    FDR_TYPE=(
        ('','Select FDR Type'),
        ('SS','Solution Sketch'),
        ('SOW','Statement Of Work'),
    )
    fdr_type=models.CharField(max_length=3,choices=FDR_TYPE)

    opco=models.CharField(max_length=3,choices=OPERATING_COMPANY)

    project_no=models.CharField(max_length=15)

    department=models.CharField(max_length=255)

    project_title=models.CharField(max_length=500)

    project_description=models.TextField()

    x=users.filter(user_designation ='PM')
    pm_name=tuple([(i.first_name, i.first_name) for i in x])

    x=users.filter(user_designation ='PL')
    pl_name=tuple([(i.first_name, i.first_name) for i in x])

    x=users.filter(user_designation ='II')
    ii_name=tuple([(i.first_name, i.first_name) for i in x])

    project_manager=models.CharField(max_length=50,
choices=pm_name,blank=True)
    PL_STATUS=(
        ('','Select Status'),
        ('NA','Not Assigned'),
        ('A','Assigned'),
        ('ROT','Ready For Triage'),
        ('APP','Approved'),
```

```
        ('P','Pending'),
        ('IP','In Progress'),
        ('C','Completed')
    )
    T_STATUS=(
        ('','Select Status'),
        ('NA','Not Assigned'),
        ('A','Assigned'),
        ('CPN','Cost Pending'),
        ('CPD','Cost Provided'),
    )
    proposal_lead=models.CharField(max_length=50,choices=pl_name,blank=True)
    pl_comment=models.CharField(max_length=500,blank=True)
    pl_status=models.CharField(max_length=3,choices=PL_STATUS,blank=True)

    tower_1=models.CharField(max_length=50,choices=ii_name,blank=True)
    comment_t1=models.CharField(max_length=500,blank=True)
    status_t1=models.CharField(max_length=3,choices=T_STATUS,blank=True)

    tower_2=models.CharField(max_length=50,choices=ii_name,blank=True)
    comment_t2=models.CharField(max_length=500,blank=True)
    status_t2=models.CharField(max_length=3,choices=T_STATUS,blank=True)

    tower_3=models.CharField(max_length=50,choices=ii_name,blank=True)
    comment_t3=models.CharField(max_length=500,blank=True)
    status_t3=models.CharField(max_length=3,choices=T_STATUS, blank=True)

    tower_4=models.CharField(max_length=50,choices=ii_name, blank=True)
    comment_t4=models.CharField(max_length=500, blank=True)
    status_t4=models.CharField(max_length=3,choices=T_STATUS, blank=True)

    tower_5=models.CharField(max_length=50,choices=ii_name, blank=True)
    comment_t5=models.CharField(max_length=500, blank=True)
    status_t5=models.CharField(max_length=3,choices=T_STATUS, blank=True)

    tower_6=models.CharField(max_length=50,choices=ii_name, blank=True)
    comment_t6=models.CharField(max_length=500, blank=True)
    status_t6=models.CharField(max_length=3,choices=T_STATUS, blank=True)

    image = models.ImageField(null=True, blank=True, upload_to='images/')

    def __str__(self):
        return str(self.fdr_no) + ': ' + self.project_title + '-' +
self.opco

# Costing sheet model below
class Cost_sheet(models.Model):
    sheet_no = models.AutoField(primary_key = True)
    Last_update=models.DateTimeField()
    fdr_no = models.ForeignKey(FDR, on_delete=models.CASCADE,
related_name='fdr_cost_link',default= None)

    year_1 = models.IntegerField()
    year_2 = models.IntegerField()
    year_3 = models.IntegerField()
    year_4 = models.IntegerField()
    year_5 = models.IntegerField()
    year_6 = models.IntegerField()
```

```
    def save(self, *args, **kwargs):
        self.Last_update=datetime.datetime.now()
        super(Cost_sheet, self).save(*args, **kwargs)

    def __str__(self):
        return f'Cost Sheet No. {self.sheet_no} | FDR No. {self.fdr_no}'

class CostTable(models.Model):
    sheet_no=models.ForeignKey(Cost_sheet,on_delete=models.CASCADE,
related_name='costtable_costsheet_link')
    sr_no = models.AutoField(primary_key=True)
    tower = models.CharField(max_length=50, choices=Tower, blank=True)
    cost_description = models.TextField(max_length=500)
    cost_type = models.CharField(max_length=50)
    is_capex=models.BooleanField(default=False)
    is_opex=models.BooleanField(default=False)
    is_internal=models.BooleanField(default=False)
    is_external=models.BooleanField(default=False)
    opco=models.CharField(max_length=3,choices=OPERATING_COMPANY)

    rate_1 = models.FloatField()
    unit_1 = models.IntegerField()

    rate_2 = models.FloatField()
    unit_2 = models.IntegerField()

    rate_3 = models.FloatField()
    unit_3 = models.IntegerField()

    rate_4 = models.FloatField()
    unit_4 = models.IntegerField()

    rate_5 = models.FloatField()
    unit_5 = models.IntegerField()

    def __str__(self):
        return f'Cost Table {self.sr_no} | {self.sheet_no}'

    def net(self):
        return self.rate_1 * self.unit_1 + self.rate_2 * self.unit_2 +
self.rate_3 * self.unit_3 + self.rate_4 * self.unit_4 + self.rate_5 *
self.unit_5

# Table for financial year
class Financial_year_rate(models.Model):

sr_no=models.ForeignKey(CostTable,on_delete=models.CASCADE,related_name='cos
ting_year_link')
    _id=models.ObjectIdField(primary_key=True)
    year=models.DateField()
    rate=models.FloatField()
    unit=models.IntegerField()

    def __str__(self):
        return f'Rate Table for Year {self.year} | {self.sr_no}'

    # class Meta:
```

```
    #     unique_together=['sr_no','year']

class TriageActionTracker(models.Model):
    fdr_no = models.ForeignKey(FDR, on_delete=models.CASCADE,
related_name='triage_action_tracker_link')
    # sr_no = models.AutoField(primary_key=True)
    _id = models.ObjectIdField(primary_key=True)
    action_tracker = models.TextField()
    owners = tuple([(i.first_name, i.first_name) for i in users])
    action_owner = models.CharField(max_length=255, choices=owners)
    received_date = models.DateField()

    def __str__(self):
        return str(self.fdr_no) + ' ' + str(self.action_owner)

    def get_fdr_no(self):
        return str(self.fdr_no.fdr_no)


class SolutionSketchActionTracker(models.Model):
    fdr_no = models.ForeignKey(FDR, on_delete=models.CASCADE,
related_name='solution_sketch_action_tracker_link')
    _id = models.ObjectIdField(primary_key=True)
    received_date = models.DateField(blank=True)
    meeting_1_date = models.DateField(blank=True)
    meeting_1_notes = models.TextField(max_length=500, blank=True)
    meeting_2_date = models.DateField(blank=True)
    meeting_2_notes = models.TextField(max_length=500, blank=True)
    meeting_3_date = models.DateField(blank=True)
    meeting_3_notes = models.TextField(max_length=500, blank=True)
    ROM_cost = models.IntegerField(blank=True)
    cost_status = models.CharField(max_length=20, blank=True)
    CAB_status = models.CharField(max_length=20, blank=True)
    CAB_comment = models.TextField(blank=True)
    SS_status = models.CharField(max_length=20, blank=True)
    SS_comment = models.TextField(blank=True)

    def __str__(self):
        return str(self.fdr_no) + ' ' + str(self.fdr_no.project_no)

    def get_fdr_no(self):
        return str(self.fdr_no.fdr_no)

class StatementOfWorkActionTracker(models.Model):
    fdr_no = models.ForeignKey(FDR, on_delete=models.CASCADE,
related_name='statement_of_work_action_tracker_link')
    _id = models.ObjectIdField(primary_key=True)
    received_date = models.DateField(blank=True)
    meeting_1_date = models.DateField(blank=True)
    meeting_1_notes = models.TextField(max_length=500, blank=True)
    meeting_2_date = models.DateField(blank=True)
    meeting_2_notes = models.TextField(max_length=500, blank=True)
    meeting_3_date = models.DateField(blank=True)
    meeting_3_notes = models.TextField(max_length=500, blank=True)
    committed_cost = models.IntegerField(blank=True)
    cost_status = models.CharField(max_length=20, blank=True)
    CAB_status = models.CharField(max_length=20, blank=True)
    CAB_comment = models.TextField(blank=True)
```

```
    SOW_status = models.CharField(max_length=20, blank=True)
    SOW_comment = models.TextField(blank=True)

    def __str__(self):
        return str(self.fdr_no) + ' ' + str(self.fdr_no.project_no)

    def get_fdr_no(self):
        return str(self.fdr_no.fdr_no)
```

urls.py

```
from django.urls import path
from . import views
from accounts.views import logout_view
from django.conf.urls.static import static
from django.contrib.staticfiles.urls import staticfiles_urlpatterns
from django.conf import settings

urlpatterns=[
    path('', views.dashboard_home, name='dashboard'),
    path('createfdr',views.create_fdr,name='fdr'),
    path('updatefdr/<str:pk>/',views.update_fdr,name='view_fdr'),
    path('costing/<str:fk>/',views.costing_page,name='view_costing_sheet'),
    path('action-tracker/', views.view_triage_action_tracker,
name='triage-action-tracker'),
    path('solution-sketch-tracker/',
views.view_solution_sketch_action_tracker,
name='solution-sketch-action-tracker'),
    path('statement-of-work-tracker/',
views.view_statement_of_work_action_tracker,
name='statement-of-work-action-tracker'),
    path('dummy/', views.dummy, name='dummy'),
]

if settings.DEBUG:
    urlpatterns += staticfiles_urlpatterns()
    urlpatterns += static(settings.MEDIA_URL,
document_root=settings.MEDIA_ROOT)
```

views.py

```
from django.shortcuts import render,redirect
from .forms import NewFDRForm, CostTableForm, TriageActionTrackerForm, \
    SolutionSketchActionTrackerForm, StatementOfWorkActionTrackerForm
import datetime
from .models import FDR,Cost_sheet,CostTable, Financial_year_rate,
TriageActionTracker, \
    SolutionSketchActionTracker, StatementOfWorkActionTracker
from django.contrib.auth.decorators import login_required
from django.db.models import Sum, ExpressionWrapper, F, FloatField
from pprint import pprint
import itertools

# Create your views here.
@login_required(login_url='home')
def dashboard_home(request):
    # current_user = request.user
```

```
    context={}
    # print(current_user.user_operating_company)
    opco_filter = None
    if request.session['user_opco'] == 'UK_air_db':
        opco_filter = 'UKA'
    elif request.session['user_opco'] == 'Spanish_air_db':
        opco_filter = 'SPA'
    elif request.session['user_opco'] == 'Portugal_air_db':
        opco_filter = 'POA'
    elif request.session['user_opco'] == 'Irish_air_db':
        opco_filter = 'IRA'
    elif request.session['user_opco'] == 'Belgian_air_db':
        opco_filter = 'BLA'

    if not opco_filter:
        context['FDR']=FDR.objects.using('central_db')
    else:

context['FDR']=FDR.objects.filter(opco=opco_filter).using('central_db')

    return render(request,'dash_home.html',context)


@login_required(login_url='home')
def create_fdr(request):
    if request.method == 'POST':
        form = NewFDRForm(request.POST, request.FILES)
        # object of cost sheet
        obj=Cost_sheet()
        print(form.errors)
        if form.is_valid():
            f = form.save(commit=False)
            f.save(using='central_db')
            obj.fdr_no=f
            x=datetime.datetime.now().year
            obj.year_1=x
            obj.year_2=x + 1
            obj.year_3=x + 2
            obj.year_4=x + 3
            obj.year_5=x + 4
            obj.year_6=x + 5
            obj.save(using='central_db')
            return redirect('dashboard')
        return render(request,'create_fdr.html',{'form':form})
    else:
        form = NewFDRForm()
        return render(request,'create_fdr.html',{'form':form})


@login_required(login_url='home')
def update_fdr(request, pk):
    fdr=FDR.objects.using('central_db').get(fdr_no=pk)
    if request.method == 'POST':
            form = NewFDRForm(request.POST, request.FILES, instance=fdr)
            print(form.errors)
            if form.is_valid():
                f = form.save(commit=False)
                f.save(using='central_db')
```

```
                return redirect('dashboard')
    form=NewFDRForm(instance=fdr)
    context={'form':form, 'fdr_no': pk}
    # print(context['fdr_no'])
    return render(request,'view_fdr.html',context)

def calculate_grand(external_data):
    grand_external = {}
    for key, group in itertools.groupby(external_data, lambda x:
x['tower']):
        # print(key, list(group))
        val_list = {f'year_{i+1}':0 for i in range(5)}
        iter_list = [('rate_1', 'unit_1'), ('rate_2', 'unit_2'), ('rate_3',
'unit_3'), ('rate_4', 'unit_4'), ('rate_5', 'unit_5')]
        for g in group:
            for idx, tup in enumerate(iter_list):
                val_list[f'year_{idx + 1}'] += g[tup[0]] * g[tup[1]]
        val_list['total'] = sum(val_list.values())
        grand_external[key] = val_list
    grand_total = sum([grand_external[key]['total'] for key in
grand_external])
    return grand_external, grand_total

@login_required(login_url='home')
def costing_page(request,fk):
    cost_sheet=Cost_sheet.objects.using('central_db').get(fdr_no=fk)
    fdr=FDR.objects.using('central_db').get(fdr_no=fk)
    if request.method =='POST':
        f=CostTableForm(request.POST)
        print(f.errors)
        if f.is_valid():
            form=f.save(commit=False)
            f_type=f.cleaned_data['form_type']
            print(f_type)
            if f_type == 'IO':
                form.is_internal = True
                form.is_opex = True
            elif f_type == 'IC':
                form.is_internal = True
                form.is_capex = True
            elif f_type == 'EC':
                form.is_external = True
                form.is_capex = True
            else:
                form.is_external = True
                form.is_opex = True
            form.sheet_no=cost_sheet
            form.save(using='central_db')
            cost_sheet.save(using='central_db')
            return redirect('view_costing_sheet',fk)
    cost_table_form = CostTableForm()

    entries =
CostTable.objects.filter(sheet_no=cost_sheet).using('central_db')
    grand_external, grand_external_total =
calculate_grand(list(entries.filter(is_external=True).order_by('tower').valu
es()))
    grand_internal, grand_internal_total =
```

```
calculate_grand(list(entries.filter(is_internal=True).order_by('tower').valu
es()))
    internal_capex_data = entries.filter(is_internal = True, is_capex =
True)
    internal_opex_data = entries.filter(is_internal = True, is_opex = True)
    external_capex_data = entries.filter(is_external = True, is_capex =
True)
    external_opex_data = entries.filter(is_external = True, is_opex = True)

    context={
        'cost_sheet': cost_sheet,
        'fdr': fdr,
        'cost_table_form': cost_table_form,
        'internal_capex_data': internal_capex_data,
        'internal_opex_data': internal_opex_data,
        'external_capex_data': external_capex_data,
        'external_opex_data': external_opex_data,
        'external_grand_data': grand_external,
        'internal_grand_data' : grand_internal,
        'external_grand_total':grand_external_total,
        'internal_grand_total':grand_internal_total,
    }
    return render(request,'cost_sheet/cost_sheet.html',context)

@login_required(login_url='home')
def view_triage_action_tracker(request):
    if request.method == 'POST':
        f=TriageActionTrackerForm(request.POST)
        print(f.errors)
        if f.is_valid:
            form=f.save(commit=False)
            form.save(using='central_db')
        return redirect('triage-action-tracker')
    else:
        triage_action_trackers =
TriageActionTracker.objects.using('central_db')
        form=TriageActionTrackerForm()
        context = {
            'trackers': triage_action_trackers,
            'form':form,
        }
        return render(request, 'action_trackers/triage.html', context)

@login_required(login_url='home')
def view_solution_sketch_action_tracker(request):
    if request.method == 'POST':
        f=SolutionSketchActionTrackerForm(request.POST)
        print(f.errors)
        if f.is_valid:
            form=f.save(commit=False)
            form.save(using='central_db')
        return redirect('solution-sketch-action-tracker')
    else:
        solution_sketch_trackers  =
SolutionSketchActionTracker.objects.using('central_db')
        form=SolutionSketchActionTrackerForm()
        context = {
            'form':form,
```

```
            'trackers': solution_sketch_trackers,
        }
        return render(request, 'action_trackers/solution_sketch.html',
context)

@login_required(login_url='home')
def view_statement_of_work_action_tracker(request):
    if request.method == 'POST':
        f=StatementOfWorkActionTrackerForm(request.POST)
        print(f.errors)
        if f.is_valid:
            form=f.save(commit=False)
            form.save(using='central_db')
        return redirect('statement-of-work-action-tracker')
    else:
        statement_of_work_trackers  =
StatementOfWorkActionTracker.objects.using('central_db')
        form=StatementOfWorkActionTrackerForm()
        context = {
            'form': form,
            'trackers': statement_of_work_trackers,
        }
        return render(request, 'action_trackers/statement_of_work.html',
context)

def dummy(request):
        srh = request.POST['query']
        opco_filter = None
        if request.session['user_opco'] == 'UK_air_db':
            opco_filter = 'UKA'
        elif request.session['user_opco'] == 'Spanish_air_db':
            opco_filter = 'SPA'
        elif request.session['user_opco'] == 'Portugal_air_db':
            opco_filter = 'POA'
        elif request.session['user_opco'] == 'Irish_air_db':
            opco_filter = 'IRA'
        elif request.session['user_opco'] == 'Belgian_air_db':
            opco_filter = 'BLA'
        fdr =
FDR.objects.filter(project_description__icontains=srh,opco=opco_filter).usin
g('central_db')
        for i in fdr:
            print(i)
        params = {'search_results': fdr}
        return render(request,'test.html',params)

def SearchPage(request):
        srh = request.GET['query']
        fdr = FDR.objects.filter(name__icontains=srh)
        params = {'search_results': fdr, 'search':srh}
        return render(request,'test.html',params)
```

settings.py

```
-------------------------------------------------------------------------------------------------------------------------
"""
Django settings for ITSM project.

Generated by 'django-admin startproject' using Django 3.0.4.
```

```
For more information on this file, see
https://docs.djangoproject.com/en/3.0/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/3.0/ref/settings/
"""

import os

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/3.0/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'zi6h58th-o^vr&yj*jj0f5l(ujqf%c1ti-%mf&!a$e&os0w$^2'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'dashboard',
    'widget_tweaks',
    'accounts',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'ITSM.urls'
EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
EMAIL_HOST = 'smtp.gmail.com'
EMAIL_PORT = 587
EMAIL_USE_TLS = True
EMAIL_HOST_USER = 'djangotestforme@gmail.com'
EMAIL_HOST_PASSWORD = 'test-123'
# SESSION_SAVE_EVERY_REQUEST = True
```

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR,'template')],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

AUTH_USER_MODEL = 'accounts.Account'

WSGI_APPLICATION = 'ITSM.wsgi.application'

AUTHENTICATION_BACKENDS = ['accounts.backends.CustomAuthenticate',
'django.contrib.auth.backends.ModelBackend']


# Database
# https://docs.djangoproject.com/en/3.0/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'djongo',
        "CLIENT": {
            "name":"users_db",

"host":"mongodb+srv://admin:admin@cluster0.u8ytw.mongodb.net/users_db?retryW
rites=true&w=majority",
            "username":"admin",
            "password":"admin",
            "authMechanism":"SCRAM-SHA-1"
        }
    },
    'Belgian_air_db': {
        'ENGINE': 'djongo',
        "CLIENT":{
            "name":"Belgian_air_db",

"host":"mongodb+srv://admin:admin@cluster0.u8ytw.mongodb.net/Belgian_air_db?
retryWrites=true&w=majority",
            "username":"admin",
            "password":"admin",
            "authMechanism":"SCRAM-SHA-1"
        }
    },
    'Irish_air_db': {
        'ENGINE': 'djongo',
        "CLIENT": {
            "name":"Irish_air_db",
```

```
"host":"mongodb+srv://admin:admin@cluster0.u8ytw.mongodb.net/Irish_air_db?re
tryWrites=true&w=majority",
            "username":"admin",
            "password":"admin",
            "authMechanism":"SCRAM-SHA-1"
        }
    },
    'Portugal_air_db': {
        'ENGINE': 'djongo',
        "CLIENT": {
            "name":"Portugal_air_db",

"host":"mongodb+srv://admin:admin@cluster0.u8ytw.mongodb.net/Portugal_air_db
?retryWrites=true&w=majority",
            "username":"admin",
            "password":"admin",
            "authMechanism":"SCRAM-SHA-1"
        }
    },
    'Spanish_air_db': {
        'ENGINE': 'djongo',
        "CLIENT": {
            "name":"Spanish_air_db",

"host":"mongodb+srv://admin:admin@cluster0.u8ytw.mongodb.net/Spanish_air_db?
retryWrites=true&w=majority",
            "username":"admin",
            "password":"admin",
            "authMechanism":"SCRAM-SHA-1"
        }
    },
    'UK_air_db': {
        'ENGINE': 'djongo',
        "CLIENT": {
            "name":"UK_air_db",

"host":"mongodb+srv://admin:admin@cluster0.u8ytw.mongodb.net/UK_air_db?retry
Writes=true&w=majority",
            "username":"admin",
            "password":"admin",
            "authMechanism":"SCRAM-SHA-1"
        }
    },
    'Tag_tech_db': {
        'ENGINE': 'djongo',
        "CLIENT": {
            "name":"Tag_tech_db",

"host":"mongodb+srv://admin:admin@cluster0.u8ytw.mongodb.net/Tag_tech_db?ret
ryWrites=true&w=majority",
            "username":"admin",
            "password":"admin",
            "authMechanism":"SCRAM-SHA-1"
        }
    },
    'central_db': {
        'ENGINE': 'djongo',
        "CLIENT": {
```

```
            "name":"central_db",

"host":"mongodb+srv://admin:admin@cluster0.u8ytw.mongodb.net/central_db?retr
yWrites=true&w=majority",
            "username":"admin",
            "password":"admin",
            "authMechanism":"SCRAM-SHA-1"
        }
    },
}

PASSWORD_HASHERS = [
    'django.contrib.auth.hashers.PBKDF2PasswordHasher'

]

# Password validation
#
https://docs.djangoproject.com/en/3.0/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/3.0/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.0/howto/static-files/

STATIC_URL = '/static/'
```

```
STATICFILES_DIRS=[
    os.path.join(BASE_DIR,'static')
]
STATIC_ROOT=os.path.join(BASE_DIR,'assets')

MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
MEDIA_URL = '/media/'
```