Developer Project Equipment

A PROJECT REPORT

Submitted By

PRIYANKA PATEL

University Roll No 1900290140025

Submitted in partial fulfilment of

the Requirements for the Degree

of

MASTER OF COMPUTER APPLICATION

Under the Supervision of

Mrs. Vidushi (Assistant Professor)

ASSOCIATE PROFESSOR

Submitted to

DEPARTMENT OF COMPUTER APPLICATIONS

KIET GROUP OF INSTITUTIONS, DELHI-NCR, GHAZIABAD UP

(June, 2022)

Declaration

I undersigned hereby declare that the project report ("**Klickfie**"), submitted for partial fulfillment of the requirements for the requirement for the award of the degree of Master of Computer Applications by the 'KIET GROUP OF INSTITUTION'S GHAZIABAD' is a Bonafede work done by me under supervision of (Pro. Vidushi Mishra). This submission represents my ideas in my own words and where ideas or words of others have been included, I have adequately and accurately cited and referenced the original sources. I also declare that I have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission.

Name: Priyanka Patel

Roll No: 1900290140025

Branch: MCA (6th Sem)

(Candidate Signature)

Certificate

Certified that **PRIYANKA PATEL (Enrollment no. 1900290140025)** has carried out the project work having "Klickfie" for Master of Computer Applications from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Technical University, Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself/herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Date:

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:

Pro. Vidushi Mishra

Associate Professor

Department of Computer Applications

KIET Group of Institutions, Ghaziabad

Signature of Internal Examiner **Signature of External Examiner**

0120 4174793
info@appinventiv.com

Date: 23rd May 2022

Sub: Letter of Internship

To Whomsoever It May Concern

This letter confirms **Priyanka Patel (AI1581)** is an employee of **Appinventiv Technologies Pvt Ltd**. She issuccessfully working on the project assigned by the organisation. The training period is on going from 7 February 2022.

The same will be continuing and the letter confirms his successful internship period.

Sincerely,

Nikita PurkayasthaHR Manager

Abstract

It is good source of inter activity among Employees and Organization's managerial department. It is done in order to manage timing and also improve Organization's Productivity.

This is a shopping web application. The name of this web application is 'Order Guide'.

In this web application the uniqueness is that, the user can order multiple items in single order according to their requirement.

Because in any big organization it's became time consuming to order welcome kit for their new hired employee if hiring is in bulk so it will save time.

ACKNOWLEDGEMENTS

I wish to record my heartfelt gratitude and sincere thanks to Pro. Vidushi Mishra, Asst. Professor, Department of Computer Applications, KIET Group of Institutions, Ghaziabad, for his kind support and inspiration given to me till the end of my project.

Words are not enough to express my gratitude to Dr. Ajay Kumar Shrivastava, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help at various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions. And i also want to thank my organization members who helped me in completing my project.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

Priyanka Patel

Table of Contents

S.NO	Tittle	Page
	Declaration	
	Certificate	
	Abstract	
	ACKNOWLEDGEMENTS	
CHAPTER 1	INTRODUCTION	1
1.1	PROJECT DESCRIPION	1
1.2	PROJECT SCOPE	2
1.3	Hardware/Software used in Project	3
CHAPTER 2	TECHNICAL FEASIBILITY	4
2.1	Operational Feasibility	4
2.2	Technical Feasibility	5
2.3	Technology Description	5
CHAPTER 3	BACKEND DESIGN	10
3.1	MongoDB	10
3.2	Database views	11
3.3	Data dictionary tables	12
CHAPTER 4	FRONTEND DESIGN	15
4.1	Android	15
4.2	IOS	17
4.3	Firebase	19
4.4	Java	21

CHAPTER 5	TESTING	24
5.1	Unit Testing	24
5.2	Integration Testing	25
5.3	Software Verification and Validation	27
5.4	Black Box Testing	29
5.5	White Box Testing	30
5.6	System Testing	32
CHAPTER 6	WORKFLOW	33
CHAPTER 7	REPORT	40
7.1	UI Action	40
7.2	Client Script	41
7.3	UI Policy	43
7.4	Data Policy	44
7.5	Business Rule	44
7.6	Notification	47
7.7	SLA	53
CHAPTER 8	RESULT	60
8.1	Create a Developer Project Equipment	62
8.2	Request a Developer Project Equipment	63
CHAPTER 9	ADVANTAGE	72
CHAPTER 10	CONCLUSION	74

CHAPTER 1

INTRODUCTION

1.1 PROJECT DESCRIPION

Developer Project Equipment submits a single service catalog request that generates several items.

For example, a New Employee Hire Developer Project Equipment can contain several items that new employees commonly need, such as business cards, computer, and cell phone. After selecting this Developer Project Equipment, the customer can then provide information about the new employee, including location and job title. The Developer Project Equipment then submits an order for catalog items like business cards, based on the details provided.

'Developer Project Equipment's determine which catalog items to order by evaluating Developer Project Equipment rule conditions. Information the customer enters within the Developer Project Equipment can be passed as cascading variables to the ordered items, allowing common information to be reused across multiple items.

Administrators and catalog administrators can create Developer Project Equipment's for the service catalog.

Developer Project Equipment's can be run automatically, generating a set of ordered items without needing to manually submit a service catalog request. For example,

an onboarding workflow for a new employee can automatically run a Developer Project Equipment to order items for that employee.

1.2 PROJECT SCOPE

Its major use is to order items in bundle for again and again like: New Employee Hire Developer Project Equipment can contain several items that new employees commonly need, such as business cards, computer, and cell phone. After selecting this Developer Project Equipment, the customer can then provide information about the new employee, including location and job title. The Developer Project Equipment then submits an order for catalog items like business cards, based on the details provided.

This project has a wide scope as it is better than the manual orders the individual same items again and again. The requirement deals with Developer Project Equipment's…specifically with the ability to control the execution order of individual items within a Developer Project Equipment. Anywhere where it's demand of items in bulks there will be demand of Developer Project Equipment.

1.3 Hardware / Software used in Project

1.3.1 Hardware Requirements

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware.
Memory – All software, when run, resides in the random-access memory (RAM) of a computer. Memory requirements are defined after considering demands of the application, operating system, supporting software and files, and other running processes. Optimal performance of other unrelated software running on a multi-tasking computer system is also considered when defining this requirement.

Number	Description
1	PC with 250 GB or more Hard disk
2	PC with 4 GB RAM.
3	PC with Pentium 1 and above

Table 1.1 Hardware Requirements

1.3.2 Software Requirements

The software requirements are description of features and functionalities of the target system. Requirements convey the expectations of users from the software product. The requirements can be obvious or hidden, known or unknown, expected or unexpected from client's point of view. Every project needs

software. We should try to understand what sort of requirements may arise in the requirement elicitation phase and what kinds of requirements are expected from the software system.

Number	Description	Type
1	Operating System	Windows XP / Windows
2	Language	JavaScript, TypeScript, Android, IOS
3	Database	MongoDB
4	PDI	Service Now
5	Browser	Google Chrome / Internet Explorer

Table 1.2 Software Requirements

CHAPTER 2

TECHNICAL FEASIBILITY

FEASIBILITY STUDY

A feasibility study is a high-level capsule version of the entire System analysis and Design Process. The study begins by classifying the problem definition. Feasibility is to determine if it's worth doing. Once an acceptance problem definition has been generated, the analyst develops a logical model of the system. A search for alternatives is analysed carefully. There are 3 parts in feasibility study.

2.1 Operational Feasibility

Operational feasibility is the measure of how well a proposed system solves the problems, and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development. The operational feasibility assessment focuses on the degree to which the proposed development projects fits in with the existing business environment and objectives with regard to development schedule, delivery date, corporate culture and existing business processes. To ensure success, desired operational outcomes must be imparted during design and development. These include such design-dependent parameters as reliability, maintainability, support-ability, usability, product-ability, sustainability, affordability and others. These parameters are required to be considered at

the early stages of design if desired operational behaviour is to be realized. A system design and development require appropriate and timely application of engineering and management efforts to meet the previously mentioned parameters. A system may serve its intended purpose most effectively when its technical and operating characteristics are engineered into the design. Therefore, operational feasibility is a critical aspect of systems engineering that needs to be an integral part of the early design phases.

2.2 TECHNICAL FEASIBILITY

This involves questions such as whether the technology needed for the system exists, how difficult it will be to build, and whether the firm has enough experience using that technology. The assessment is based on outline design of system requirements in terms of input, processes, output, fields, programs and procedures. This can be qualified in terms of volume of data, trends, frequency of updating in order to give an introduction to the technical system. The application is the fact that it has been developed on windows XP platform and a high configuration of 1GB RAM on Intel Pentium Dual core processor. This is technically feasible. The technical feasibility assessment is focused on gaining an understanding of the present technical resources of the organization and their applicability to the expected needs of the proposed system. It is an evaluation of the hardware and software and how it meets the need of the proposed system.

2.3 Technology Description

2.3.1 ServiceNow

`ServiceNow is a cloud-based platform, which was mainly developed for workflow and process automation as per the ITIL principles. However, it is highly customisable and also can be used for other purposes. ServiceNow is an American based company and was founded in 2004 by Fred Leddy. It has a unique way for naming its versions. They name the versions based on the major cities of the world. The latest version of ServiceNow is Orlando.

ServiceNow offers many ready to use solutions, workflows and products for an organisation. The organisation can develop the customised applications and modules as per the business requirement using the ServiceNow scripting and existing tools.

2.3.2 Services of ServiceNow

Some of the important offerings and most widely used services of ServiceNow are explained below −

IT Service management

ServiceNow is mainly used as a ticketing tool to manage incidents, problems and changes. It has many advanced features, analytics and insights that impacts the speed and delivery of IT.

HR management

ServiceNow can be used for almost all HR delivery services like leave management, timesheet management, employee document management, new onboarding management, performance management, etc.

IT Asset management

With ServiceNow, we can manage our hardware and software assets to optimise cost and increase efficiency. ServiceNow has features such as licence management, warranty management, CI management, advanced reporting and insights, etc.

Finance operation management

ServiceNow manages all the activities related to finance close and automates the financial processes.

Apart from offerings mentioned above, ServiceNow also offers services for IT business management, security operations, virtual chatbots, etc.

ServiceNow is built using Java and Tomcat web server running on Linux. Although to develop new modules and applications in ServiceNow the JavaScript knowledge is sufficient.

2.3.3 ServiceNow Instance

A ServiceNow instance is a set of databases, applications, virtual machines, libraries grouped together to provide the required services to a specific customer. ServiceNow customer instance is built on multi-instance architecture.

The figure given below shows multi-instance architecture −

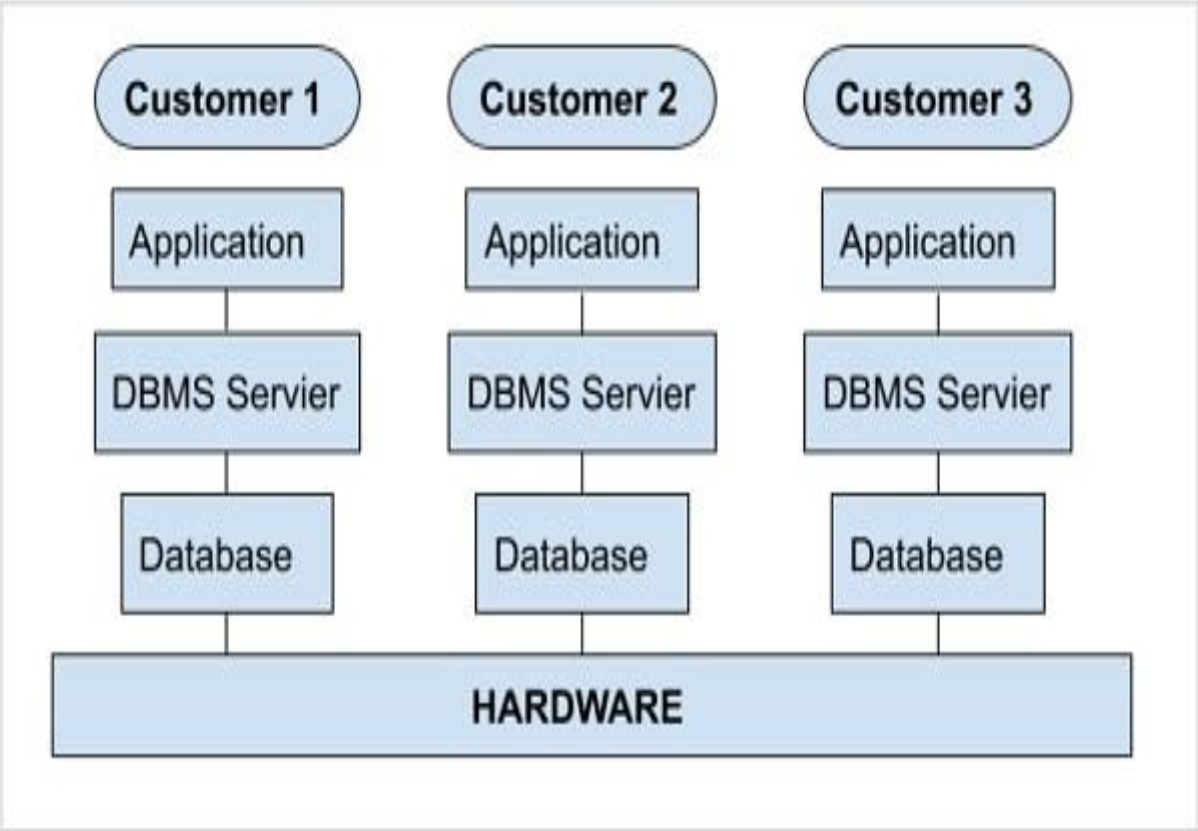


Fig.2.1 multi-instance architecture

The important point to note here is that, each customer has a separate customised application(s) along with separate database(s) running on shared hardware resources. The customer data is encrypted and therefore, is completely secure. The deployment of ServiceNow is very flexible and it can also be implemented in a private cloud.

ServiceNow also offers developer instance, which is a community edition free of cost. ServiceNow developer instance was launched, to promote the use of ServiceNow

and to provide the resources, so that the developers/administrators can learn, build, enhance and customise the applications in ServiceNow.

2.4 Technology used in project

ServiceNow ITSM overview

Improved employee and customer experiences are essential for successful digital transformation. However, using siloed IT tools for your digital transformation creates information silos. With disparate data, processes and excessive amounts of time and money spent on firefighting, you're left with limited resources to create delightful employee experiences.

ServiceNow IT Service Management (ITSM) is a modern, cloud-based, silo-busting service management solution. With ServiceNow ITSM, you can consolidate on-premises legacy tools to a single cloud platform and stop wasting your money and harness shared data and analytic with automated workflows on the Now Platform® in the Nonstop cloud. Platform-native AI and machine learning along with natural language virtual agent chat-bots unburden your IT staff and boost productivity 30%. ServiceNow ITSM lets you:

- Empower employees to self-solve issues 24/7, raise questions, and get relevant, accurate, and consistent information to improve

- employee satisfaction.

- Make smarter decisions, automate 20%1 of your services, and continually improve your services in role-based workspace.

- Triage, collaborate, and enable agents to resolve incidents, find answers, and stay connected from anywhere to resolve high impact incidents and improve agent productivity by 30%

Performance Analytic

Enable stakeholders—workers, owners, and executives— responsible for service delivery to make smarter, real-time decisions with Performance Analytic. Use data visualizations to anticipate trends, prioritize resources, and drive IT alignment with business goals.

Continual Improvement Management

Collaborate and prioritize data, people, and business goals to manage your strategic IT road-map investments with the structured framework and workflow of Continual Improvement Management (CIM).

CHAPTER 3

BACKEND DESIGN

3.1 MongoDB

MongoDB is an open-source, fast reliable, and flexible non-relational database management system, typically used with android and IOS application. This chapter is an introductory chapter about MongoDB, what is MongoDB, and the main features of MongoDB are described here

3.1.1 What is MongoDB

- MongoDB is a database system used for developing web-based software applications.
- MongoDB used for both small and large applications.
- MongoDB is a non-relational database management system.
- MongoDB is fast, reliable, and flexible and easy to use.
- MongoDB supports standard SQL (Structured Query Language).
- MongoDB is free to download and use.

- MongoDB was developed by Michael Wideners and David A-Mark in 1994.

- MongoDB is presently developed, distributed, and supported by Oracle Corporation.
- MongoDB Written in C, C++.

3.1.2Main Feature of MongoDB

- MongoDB server design is multi-layered with independent modules

.

- MongoDB is fully multi-threaded by using kernel threads. It can handle multiple CPUs if they are available.

- MongoDB provides transactional and non-transactional storage engines.

- MongoDB has a high-speed thread-based memory allocation system

.

- MongoDB supports in-memory heap table.

- MongoDB Handles large databases.

- MongoDB Server works in client/server or embedded systems.

- MongoDB Works on many different platforms.

3.2 Database views

A database view defines table joins for reporting purposes.

For example, a database view can join the Incident table to the Metric Definition and Metric Instance tables. This view can be used to report on incident metrics and may include fields from any of these three tables.

Several useful database views are installed with the Database View plugin and the Database Views for Service Management plugin. These database views cover most metric reporting needs and greatly reduce the need to define new ones.

Any user who can create a report can use database views as the report source, but ACLs on the underlying tables are honoured.

Note:

- The accumulated impact on performance grows as the number of tables that are included in the view and the number of records that those tables contain increases. To maximize the performance of the database view, ensure that the 'where' clauses that are defined in the database view are based on indexed fields.

- A database view is not treated like a custom table, so there is no licensing impact.

3.3 Data dictionary tables

The system defines data dictionary, data modelling, and entity relationship information in multiple tables.

- Tables [sys_db_object]: Contains a record for each table.

- Dictionary Entries [sys_dictionary]: Contains additional details for each table and the definition for every column on each table. Each row represents either a column on a table or a table.

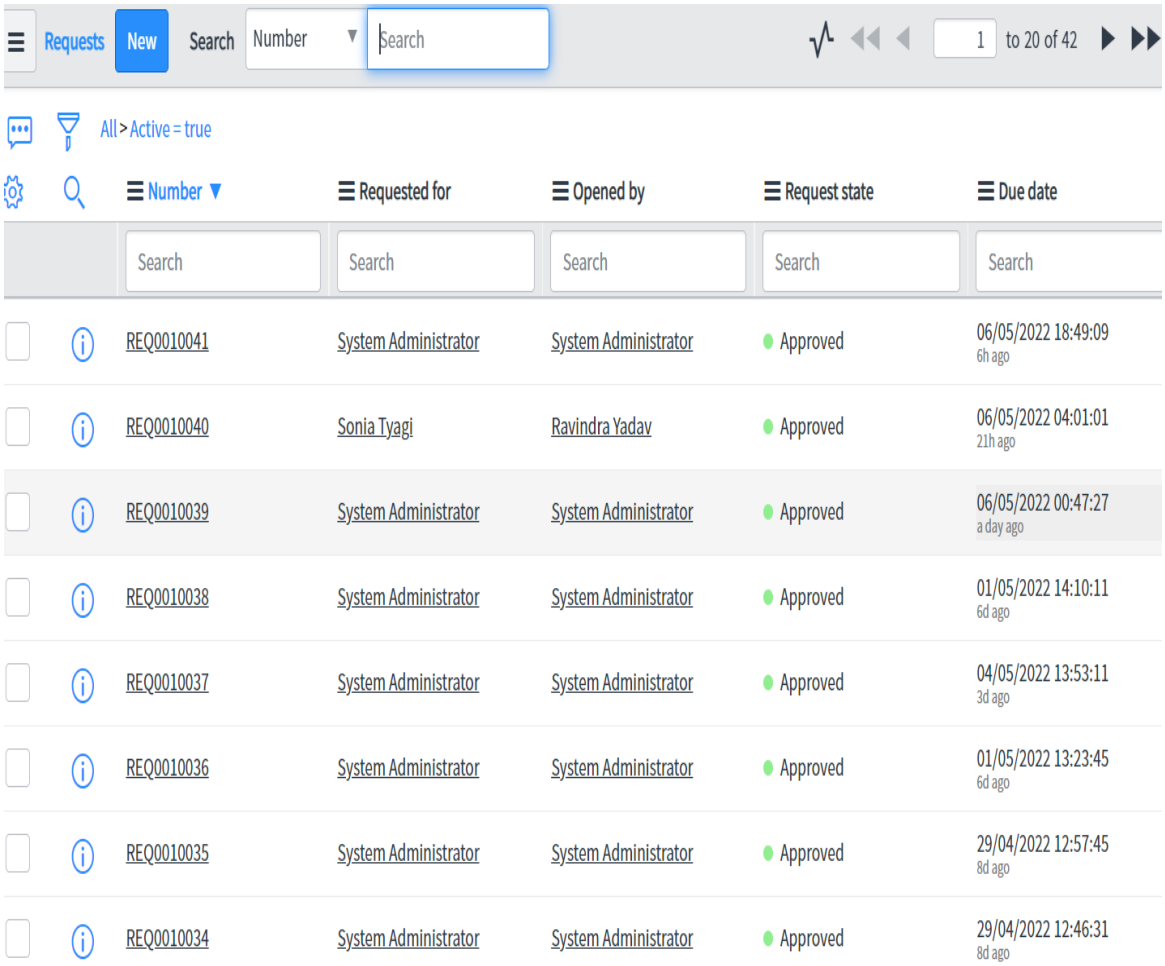
- Field Labels [sys_documentation]: Contains the human-readable labels and language information.
-

Tables

The Tables [sys_db_object] table contains a record for each table in the database.

Access the Tables list by navigating to System Definition > Tables. Administrators can create a custom table, add or modify columns in a searchable and sortable embedded list, and define the auto-number format.

The following image shows a list of the tables that extend the Application File table.



Img 3.1 Extended Application File table

System dictionary

The system dictionary is a table, called Dictionary Entry [sys_dictionary], that contains details for each table and the definition for every column on each table in an instance.

Each row in the system dictionary represents either a table or a column in one of the tables. The system dictionary provides options for administrators to modify tables and fields, which in turn define lists and forms.

Use caution when changing system dictionary records because changes can have a high impact on functionality. In particular, changes to dictionary entries for system tables, which are tables that begin with sys_, can create system-wide issues such as the inability to use update sets.

Dictionary changes are difficult to reverse. Also, dictionary changes automatically apply to all extended tables unless a dictionary override is defined. Be sure that changes are well-tested before applying them to a production instance.

Creation options

When you create a field from the system dictionary, it is automatically added at the end of the first section of the default form view.

In most cases, use the following interfaces rather than creating entries directly on the system dictionary:

- To create tables and fields, use the Tables module.

- To create fields, configure the table form.

Dictionary overrides

Dictionary overrides provide the ability to define a field on an extended table differently from the field on the parent table.

Modify dictionary entries

You can modify dictionary entries by configuring a field on a form or from the Dictionary module.

CHAPTER 4

FRONTEND DESIGN

4.1 CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language like HTML.CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.CSS is designed to enable the separation of presentation and content, including layout, colour, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .CSS file, and reduce complexity and repetition in the structural content. CSS information can be provided from various sources. These sources can be the web browser, the user and the author. The information from the author can be further classified into inline, media type, importance, selector specificity, rule order, inheritance and property definition. CSS style information can be in a separate document or it can be embedded into an HTML document. Multiple style sheets can be imported. Different styles can be applied depending on the output device being used; for example, the screen version can be quite different from the printed version, so that authors can tailor the presentation appropriately for each medium. The style sheet

with the highest priority controls the content display. Declarations not set in the highest priority source are passed on to a source of lower priority, such as the user agent style. The process is called cascading. One of the goals of CSS is to allow users greater control over presentation. Someone who finds red italic headings difficult to read may apply a different style sheet.

Depending on the browser and the web site, a user may choose from various style sheets provided by the designers, or may remove all added styles and view the site using the browser's default styling, or may override just the red italic heading style without altering other attributes.

Cascading Style Sheets, fondly referred to as CSS, is a simple design language intended to simplify the process of making web pages presentable.

CSS handles the look and feel part of a web page. Using CSS, you can control the colour of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colour are used, layout designs, variations in display for different devices and screen sizes as well as a variety of other effects.

CSS is easy to learn and understand but it provides powerful control over the presentation of an HTML document. Most commonly, CSS is combined with the markup languages HTML or XHTML.

Who Creates and Maintains CSS?

CSS is created and maintained through a group of people within the W3C called the CSS Working Group. The CSS Working Group creates documents called specifications. When a specification has been discussed and officially ratified by the W3C members, it becomes a recommendation.

These ratified specifications are called recommendations because the W3C has no control over the actual implementation of the language. Independent companies and organizations create that software.

NOTE − The World Wide Web Consortium, or W3C is a group that makes recommendations about how the Internet works and how it should evolve.

CSS Versions

Cascading Style Sheets level 1 (CSS1) came out of W3C as a recommendation in December 1996. This version describes the CSS language as well as a simple visual formatting model for all the HTML tags.

CSS2 became a W3C recommendation in May 1998 and builds on CSS1. This version adds support for media-specific style sheets e.g., printers and aural devices, downloadable fonts, element positioning and tables.

4.2 Bootstrap

Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains CSS- and (optionally) JavaScript-based design templates for typography, forms, buttons, navigation and other interface components. Bootstrap is the third-most starred project on GitHub, with more than 135,000 stars, behind only free Code Camp (almost 305,000 stars) and marginally behind Vue.js framework. According to Alexa Rank, Bootstrap getbootstrap.com is in the top-2000 in US while vuejs.org is in top-7000 in US. Bootstrap is a web framework that focuses on simplifying the development of informative web pages (as opposed to web apps). The primary purpose of adding it to a web project is to apply Bootstrap's choices of colour, size, font and layout to that project. As such, the primary factor is whether the developers in charge find those choices to their liking. Once added to a project, Bootstrap provides basic style definitions for all HTML elements. The result is a uniform appearance for prose, tables and form elements across web browsers. In addition, developers can take advantage of CSS classes defined in Bootstrap to further customize the appearance of their contents. For example, Bootstrap has provisioned for light- and dark-colored tables, page headings, more prominent pull quotes, and text with a highlight. Bootstrap is a web framework that focuses on simplifying the development of informative web pages (as opposed to web apps). The primary purpose of adding it to a web project is to apply Bootstrap's choices of color, size, font and layout to that project. As such, the primary factor is whether the developers in charge find those choices to their liking. Once added to a project, Bootstrap provides basic style definitions for all HTML elements. The

result is a uniform appearance for prose, tables and form elements across web browsers. In addition, developers can take advantage of CSS classes defined in Bootstrap to further customize the appearance of their contents. For example, Bootstrap has provisioned for light- and dark-coloured tables, page headings, more prominent pull quotes, and text with a highlight.

Bootstrap was developed by Mark Otto and Jacob Thornton at Twitter.

It was released as an open-source product in August 2011 on GitHub.

In June 2014 Bootstrap was the No.1 project on GitHub.

Why use Bootstrap

Following are the main advantage of Bootstrap:

- It is very easy to use. Anybody having basic knowledge of HTML and CSS can use Bootstrap.
- It facilitates users to develop a responsive website.
- It is compatible on most of browsers like Chrome, Firefox, Internet Explorer, Safari and Opera etc.

What is a responsive website?

A website is called responsive website which can automatically adjust itself to look good on all devices, from smart phones to desktops etc

What Bootstrap package contains

Scaffolding: background.

CSS:

Bootstrap provides a basic structure with Grid System, link styles, and Bootstrap comes with the feature of global CSS settings, fundamental HTML elements style and an advanced grid system.

Components:

Bootstrap contains a lot of reusable components built to provide iconography, drop-downs, navigation, alerts, pop-overs, and much more.

JavaScript Plugins:

Bootstrap also contains a lot of custom J-Query plugins. You can easily include them all, or one by one.

Is Bootstrap Best?

Bootstrap is more than efficient to create a responsive and mobile first website but it is not the best in the industry. There is an alternative of Bootstrap named W3.CSS which is smaller, faster, and easier to use.

Customize:

Bootstrap components are customizable and you can customize Bootstrap's components, LESS variables, and J-Query plugins to get your own style.

4.3 HTML

HTML stands for Hyper Text Markup Language. It is used to design web pages using a markup language. HTML is the combination of Hypertext and Markup language. Hypertext defines the link between the web pages. A markup language is used to define the text document within tag which defines the structure of web pages. This language is used to annotate (make notes for the computer) text so that a machine can understand it and manipulate text accordingly. Most markup languages (e.g. HTML) are human-readable. The language uses tags to define what manipulation has to be done on the text.

HTML is a markup language used by the browser to manipulate text, images, and other content, in order to display it in the required format. HTML was created by Tim Berners-Lee in 1991. The first-ever version of HTML was HTML 1.0, but the first standard version was HTML 2.0, published in 1999.

Elements and Tags:

HTML uses predefined tags and elements which tell the browser how to properly display the content. Remember to include closing tags. If omitted, the browser applies the effect of the opening tag until the end of the page.

HTML page structure:

The basic structure of an HTML page is laid out below. It contains the essential building-block elements (i.e., doctype declaration, HTML, head, title, and body elements) upon which all web pages are created.

<!DOCTYPE html>:

This is the document type declaration (not technically a tag). Itv declares a document as being an HTML document. The doctype declaration is not case-sensitive.

<html>:

This is called the HTML root element. All other elements are contained within it.

<head>:

The head tag contains the “behind the scenes” elements for a web-page. Elements within the head aren’t visible on the front-end of a web-page. HTML elements used inside the <head> element include:

- <style>
- <title>
- <base>

- <script>
- <meta>
- <link>

<body>:

The body tag is used to enclose all the visible content of a web-page. In other words, the body content is what the browser will show on the front-end. An HTML document can be created using any text editor. Save the text file using .html or .htm. Once saved as an HTML document, the file can be opened as a web-page in the browser.

NOTE: Basic/built-in text editors are Notepad (Windows) and Text-Edit (Macs). Basic text editors are entirely sufficient for when you're just getting started. As you progress, there are many feature-rich text editors available which allow for greater function and flexibility.

4.4 JavaScript

JavaScript (JS) is a light-weight object-oriented programming language which is used by several websites for scripting the web-pages. It is an interpreted, full- fledged programming language that enables dynamic interactivity on websites when applied to an HTML document. It was introduced in the year 1995 for adding programs to the web-pages in the Netscape Navigator browser.

Since then, it has been adopted by all other graphical web browsers. With JavaScript, users can build modern web applications to interact directly without reloading the page every time. The traditional website uses JS to provide several forms of interactivity and simplicity.

Although, JavaScript has no connectivity with Java programming language. The name was suggested and provided in the times when Java was gaining popularity in the market. In addition to web browsers, databases such as Couch DB and MongoDB uses JavaScript as their scripting and query language.

Features of JavaScript:

1. All popular web browsers support JavaScript as they provide built-in execution environments.

2. JavaScript follows the syntax and structure of the C programming language. Thus, it is a structured programming language.

3. JavaScript is a weakly typed language, where certain types are implicitly cast (depending on the operation).

4. JavaScript is an object-oriented programming language that uses prototypes rather than using classes for inheritance.

5. It is a light-weighted and interpreted language.

6. It is a case-sensitive language.

7. JavaScript is supportable in several operating systems including, Windows, mac OS, etc.

8. It provides good control to the users over the web browsers.

History of JavaScript:

In 1993, Mosaic, the first popular web browser, came into existence. In the year 1994, Netscape was founded by Marc Andreessen. He realized that the web needed to become more dynamic. Thus, a 'glue language' was believed to be provided to HTML to make web designing easy for designers and part-time programmers. Consequently, in

1995, the company recruited Brendan Each intending to implement and embed Scheme programming language to the browser. But, before Brendan could start, the company merged with Sun Micro-systems for adding Java into its Navigator so that it could compete with Microsoft over the web technologies and platforms. Now, two languages were there: Java and the scripting language. Further, Netscape decided to give a similar name to the scripting language as Java's. It led to 'JavaScript'. Finally, in May 1995, Marc Andreessen coined the first code of JavaScript named 'Mocha'. Later, the marketing team replaced the name with 'Live Script'. But, due to trademark reasons and certain other reasons, in December 1995, the language was finally renamed to 'JavaScript'. From then, JavaScript came into existence.

Application of JavaScript:

JavaScript is used to create interactive websites. It is mainly used for:

- Client-side validation,
- Dynamic drop-down menus,
- Displaying date and time,
- Displaying pop-up windows and dialog boxes (like an alert dialog box, confirm dialog box and prompt dialog box),
- Displaying clocks etc.

‘

CHAPTER 5

TESTING

5.1: UNIT TESTING

5.1.1 Introduction

In computer programming, unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application. In procedural programming, a unit could be an entire module, but it is more commonly an individual function or procedure. In object- oriented programming, a unit is often an entire interface, such as a class, but could be an individual method. Unit tests are short code fragments created by programmers or occasionally by white box testers during the development process. It forms the basis for component testing. Ideally, each test case is independent from the others. Substitutes such as method stubs, mock objects, fakes, and test harnesses can be used to assist testing a module in isolation. Unit tests are typically written and run by software developers to ensure that code meets its design and behaves as intended.

5.1.2Benefits:

The goal of unit testing is to isolate each part of the program and show that the individual parts are correct. A unit test provides a strict, written contract that the piece of code must satisfy. As a result, it affords several benefits.

Find problems early:

Unit testing finds problems early in the development cycle. In test-driven development (TDD), which is frequently used in both extreme programming and scrum, unit tests are created before the code itself is written. When the tests pass, that code is considered complete. The same unit tests are run against that function frequently as the larger code base is developed either as the code is changed or via an automated process with the build. If the unit tests fail, it is considered to be a bug either in the changed code or the tests themselves. The unit tests then allow the location of the fault or failure to be easily traced. Since the unit tests alert the development team of the problem before handing the code off to testers or clients, it is still early in the development process.

Facilitates Change:

Unit testing allows the programmer to refactor code or upgrade system libraries at a later date, and make sure the module still works correctly (e.g., in regression testing). The procedure is to write test cases for all functions and methods so that whenever a change causes a fault, it can be quickly identified. Unit tests detect changes which may break a design contract.

Simplifies Integration:

Unit testing may reduce uncertainty in the units themselves and can be used in a bottom-up testing style approach. By testing the parts of a program first and then testing the sum of its parts, integration testing becomes much easier.

Documentation:

Unit testing provides a sort of living documentation of the system. Developers looking to learn what functionality is provided by a unit, and how to use it, can look at the unit tests to gain a basic understanding of the unit's interface (API). Unit test cases

embody characteristics that are critical to the success of the unit. These characteristics can indicate appropriate/inappropriate use of a unit as well as negative behaviour's that are to be trapped by the unit.

5.2 INTEGRATION TESTING

Integration testing (sometimes called integration and testing, abbreviated I&T) is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before validation testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

5.2.1 Purpose

The purpose of integration testing is to verify functional, performance, and reliability requirements placed on major design items. These "design items", i.e., assemblages (or groups of units), are exercised through their interfaces using black-box testing, success and error cases being simulated via appropriate parameter and data inputs. Simulated usage of shared data areas and inter-process communication is tested and individual subsystems are exercised through their input interface. Test cases are constructed to test whether all the components within assemblages interact correctly, for example across procedure calls or process activation, and this is done after testing individual modules, i.e., unit testing. The overall idea is a "building block" approach, in which verified assemblages are added to a verified base which is then used to support the integration testing of further assemblages. Software integration testing is performed according to the software development life cycle (SDLC) after module and functional tests. The cross dependencies for software integration testing are: schedule for integration testing, strategy and selection of the tools used for integration, define the cyclomatic complexity of the software and software architecture, reuse-ability of modules and life-cycle and versioning management. Some different types of integration testing are big-bang, top-down, and bottom-up, mixed (sandwich) and risky-hardest. Other Integration Patterns [2] are: collaboration integration, backbone integration, layer integration, client-server integration, distributed services integration and high-frequency integration.

5.2.1.1 Bang

Big In the big-bang approach, most of the developed modules are coupled together to form a complete software system or major part of the system and then used for integration testing. This method is very effective for saving time in the integration testing process. However, if the test cases and their results are not recorded properly, the entire integration process will be more complicated and may prevent the testing team from achieving the goal of integration testing. A type of big-bang integration testing is called "usage model testing" which can be used in both software and hardware integration testing. The basis behind this type of integration testing is to run user-like workloads in integrated user-like environments. In doing the testing in this manner, the environment is proofed, while the individual components are proofed indirectly through their use. Usage Model testing takes an optimistic approach to testing, because it expects to have few problems with the individual components. The strategy relies heavily on the component developers to do the isolated unit testing for their product. The goal of the strategy is to avoid redoing the testing done by the developers, and instead flesh-out problems caused by the interaction of the components in the environment.

5.2.1.2 Top-down And Bottom-up

Bottom-up testing is an approach to integrated testing where the lowest level components are tested first, then used to facilitate the testing of higher-level components. The process is repeated until the component at the top of the hierarchy is tested. All the bottom or low-level modules, procedures or functions are integrated and then tested. After the integration testing of lower-level integrated modules, the next level of modules will be formed and can be used for integration testing. This approach is helpful only when all or most of the modules of the same development level are ready. This method also helps to determine the levels of software developed and makes it easier to report testing progress in the form of a percentage. Top-down testing is an approach to integrated testing where the top integrated modules are tested and the branch of the module is tested step by step until the end of the related module. Sandwich testing is an approach to combine top down testing with bottom up testing.

5.3 SOFTWARE VERIFICATION AND VALIDATION

5.3.1 Introduction

In software project management, software testing, and software engineering, verification and validation (V&V) is the process of checking that a software system meets specifications and that it fulfils its intended purpose. It may also be referred to as software quality control. It is normally the responsibility of software testers as part of the software development life-cycle. Validation checks that the product design satisfies or fits the intended use (high-level checking), i.e., the software meets the user requirements. This is done through dynamic testing and other forms of review. Verification and validation are not the same thing, although they are often confused. Boehm succinctly expressed the difference between

Validation: Are we building the right product?

Verification: Are we building the product, right?

According to the Capability Maturity Model (CMMI-SW v1.1)

Software Verification: The process of evaluating software to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase.

Software Validation: The process of evaluating software during or at the end of the development process to determine whether it satisfies specified requirements. In other words, software verification is ensuring that the product has been built according to the requirements and design specifications, while software validation ensures that the product meets the user's needs, and that the specifications were correct in the first place. Software verification ensures that "you built it right". Software validation ensures that "you built the right thing". Software validation confirms that the product, as provided, will fulfil its intended use.

From Testing Perspective

Fault – wrong or missing function in the code.

Failure – the manifestation of a fault during execution.

Malfunction – according to its specification the system does not meet its specified functionality.

Both verification and validation are related to the concepts of quality and of software quality assurance. By themselves, verification and validation do not guarantee software quality; planning, traceability, configuration management and other aspects of software engineering are required. Within the modelling and simulation (M&S) community, the definitions of verification, validation and accreditation are similar:

M&S Verification is the process of determining that a computer model, simulation, or federation of models and simulations implementations and their associated data accurately represent the developer's conceptual description and specifications.

M&S Validation is the process of determining the degree to which a model, simulation, or federation of models and simulations, and their associated data are accurate representations of the real world from the perspective of the intended use(s).

5.3.2Classification of Methods

In mission-critical software systems, where flawless performance is absolutely necessary, formal methods may be used to ensure the correct operation of a system. However, often for non-mission critical software systems, formal methods prove to be very costly and an alternative method of software V&V must be sought out. In such cases, syntactic methods are often used.

5.3.3Test Cases

A test case is a tool used in the process. Test cases may be prepared for software verification and software validation to determine if the product was built according to the requirements of the user. Other methods, such as reviews, may be used early in the life cycle to provide for software validation.

5.4 Black-Box Testing

Black-box testing is a method of software testing that examines the functionality of an application without peering into its internal structures or workings. This method of test can be applied virtually to every level of software testing: unit, integration, system and acceptance. It typically comprises most if not all higher-level testing, but can also dominate unit testing as well.

5.4.1 Test Procedures

Specific knowledge of the application's code/internal structure and programming knowledge in general is not required. The tester is aware of what the software is supposed to do but is not aware of how it does it. For instance, the tester is aware that a particular input returns a certain, invariable output but is not aware of how the software produces the output in the first place.

5.4.2 Test Cases

Test cases are built around specifications and requirements, i.e., what the application is supposed to do. Test cases are generally derived from external descriptions of the software, including specifications, requirements and design parameters. Although the tests used are primarily functional in nature, non-functional tests may also be used. The test designer selects both valid and invalid inputs and determines the correct output, often with the help of an oracle or a previous result that is known to be good, without any knowledge of the test object's internal structure.

5.5 White-Box Testing

White-box testing (also known as clear box testing, glass box testing, transparent box testing, and structural testing) is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality (i.e., black-box testing). In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. This is analogous to testing nodes in a circuit, e.g., in-circuit testing (ICT). White-box testing can be applied at the unit, integration and system levels of the software testing process. Although traditional testers tended to think of white-box testing as being done at the unit level, it is used for

integration and system testing more frequently today. It can test paths within a unit, paths between units during integration, and between subsystems during a system–level test. Though this method of test design can uncover many errors or problems, it has the potential to miss unimplemented parts of the specification or missing requirements.

5.5.1 Levels

1) Unit testing:

White-box testing is done during unit testing to ensure that the code is working as intended, before any integration happens with previously tested code.

White-box testing during unit testing catches any defects early on and aids in any defects that happen later on after the code is integrated with the rest of the application and therefore prevents any type of errors later on.

2) **Integration testing**:

White-box testing at this level are written to test the interactions of each interface with each other. The Unit level testing made sure that each code was tested and working accordingly in an isolated environment and integration examines the correctness of the behaviour in an open environment through the use of white-box testing for any interactions of interfaces that are known to the programmer.

3) Regression testing:

White-box testing during regression testing is the use of recycled white- box test cases at the unit and integration testing levels.

5.5.2 Procedures

White-box tasting's basic procedures involves the tester having a deep level of understanding of the source code being tested. The programmer must have a deep understanding of the application to know what kinds of test cases to create so that every visible path is exercised for testing. Once the source code is understood then then source code can be analysed for test cases to be created. These are the three basic steps that white-box testing takes in order to create test cases:

Input involves different types of requirements, functional specifications, detailed designing of documents, proper source code, security specifications. This is the preparation stage of white-box testing to layout all of the basic information.

Processing involves performing risk analysis to guide whole testing process, proper test plan, execute test cases and communicate results. This is the phase of building test cases to make sure they thoroughly test the application the given results are recorded accordingly.

Output involves preparing final report that encompasses all of the above preparations and results.

5.6 SYSTEM TESTING

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black-box testing, and as such, should require no knowledge of the inner design of the code or logic. As a rule, system testing takes, as its input, all of the "integrated" software components that have passed integration testing and also the software system itself integrated with any applicable hardware system(s).

The purpose of integration testing is to detect any inconsistencies between the software units that are integrated together (called assemblages) or between any of the assemblages and the hardware. System testing is a more limited type of testing; it seeks to detect defects both within the "inter-assemblages" and also within the system as a whole.

System testing is performed on the entire system in the context of a Functional Requirement Specification(s) (FRS) and/or a System Requirement Specification (SRS). System testing tests not only the design, but also the behaviour and even the believed expectations of the customer. It is also intended to test up to and beyond the bounds defined in the software/hardware requirements specification(s).

Note: Because my Project is in progress therefore my Project is not completely done and some step according this file is not done like testing.

CHAPTER 6

WORLFLOW

WORKFLOW

Workflow provides a drag-and-drop interface for automating multi-step processes across the platform. Each workflow consists of a sequence of activities, such as generating records, notifying users of pending approvals, or running scripts. The graphical Workflow Editor represents workflows visually as a type of flowchart. It shows activities as boxes labelled with information about that activity and transitions from one activity to the next as lines connecting the boxes.

Workflow life cycle

A workflow starts when a triggering event occurs. Common triggers include a record being inserted into a specific table, or a particular field in a table being set to a specified value. For example, you might create a workflow that runs whenever a user requests approval for an item they want to order from the catalogue. You can also schedule workflows to run periodically or call them from scripts such as business rules.

When an activity completes, the workflow transitions to the next activity. An activity might have several different possible transitions to various activities, depending on the outcome of the activity. Continuing the example above, if the user's request is approved, the activity might transition to an activity that notifies someone to order the item. If the user's request is denied, the activity might transition to notifying the user that their request has been denied.

The graphical Workflow Editor represents workflows visually as a type of flowchart. It shows activities as boxes labelled with information about that activity and transitions from one activity to the next as lines connecting the boxes.

At each step in a workflow:

I. An activity is processed and an action defined by that activity occurs.

II. At the completion of an action by an activity, the workflow checks the activity's conditions.

III. For each matching condition, the workflow follows the transition to the next activity.

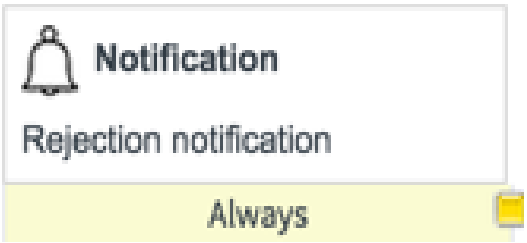
When the workflow runs out of activities, the workflow is complete. The Workflow Context stores the execution history of the activities and transitions run. The Workflow Version stores the design history of the activities, transitions, and exit conditions available to run.

Workflow activities

A workflow activity contains instructions that are processed by the workflow.

Activities can include running scripts, manipulating records, waiting for a set period of time, or logging an event. Workflow conditions determine whether or not the activity is performed. Activities can be added, removed, or rearranged. Transitions can be drawn between activities.

This is an activity that triggers a notification:



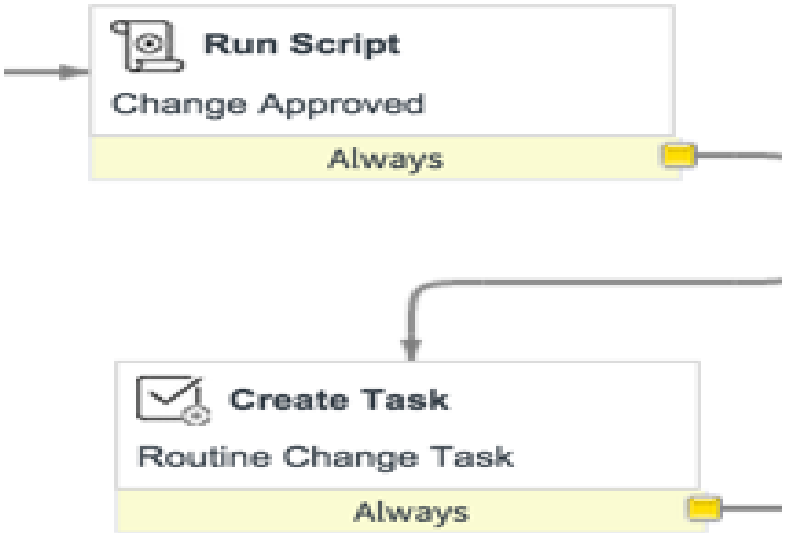
Workflow runs activities as the user session that starts them. Workflows started from record operations will run activities as the user session that performed the record operation. Workflows started from schedules or restarted from timers run activities as the System user. Workflows started from script calls run activities as the user session that started the script.

Transitions

After the workflow condition is evaluated, the workflow transition determines which activity is performed when the workflow condition is met.

This is a transition that always leads from the Change Approved script to the Change Task activity:

Sample transition

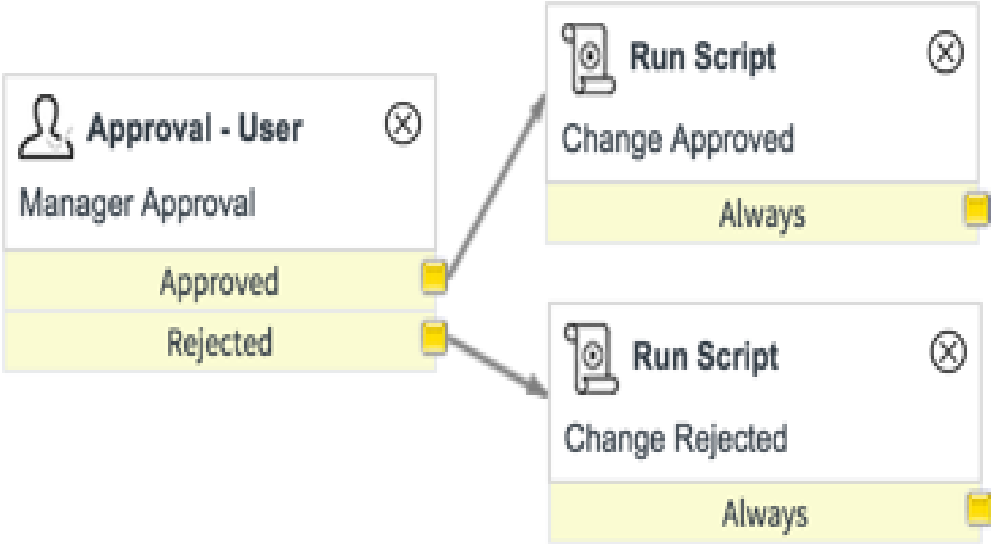


Exit conditions:

After a workflow activity is performed, the workflow condition is evaluated to determine which transition is activated.

The condition determines behaviour based on a change being approved or rejected:

Sample exit conditions



Workflow editor

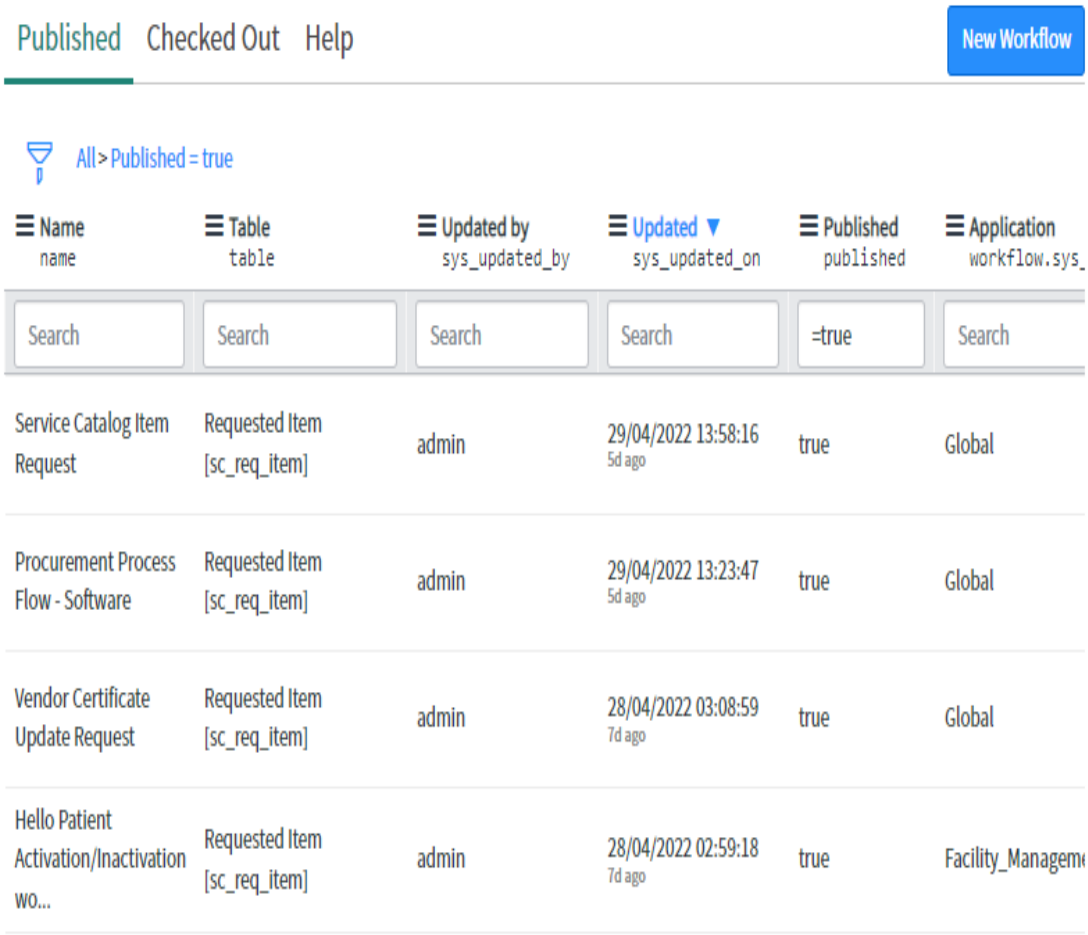
The Workflow Editor is an interface for creating and modifying workflows by arranging and connecting activities to drive processes.

You can manage multiple workflows in the same screen, create custom workflow activities, and use existing activities as data sources. Users with the workflow creator role can create workflows. Users with the workflow_admin role can create, modify, delete, and publish workflows.

To open the Workflow Editor, navigate to **Workflow** > **Workflow Editor.**

Welcome screen

The editor opens with the Welcome page, which displays a list of active, published workflows. From this tab, you can open existing workflows, create new workflows, and open help resources related to workflow.



Welcome screen

List display filters

Published: Click to view list of published workflows

Checked Out: Click to view list of workflows checked out to current user

Help: Click to view links to help resources for workflow

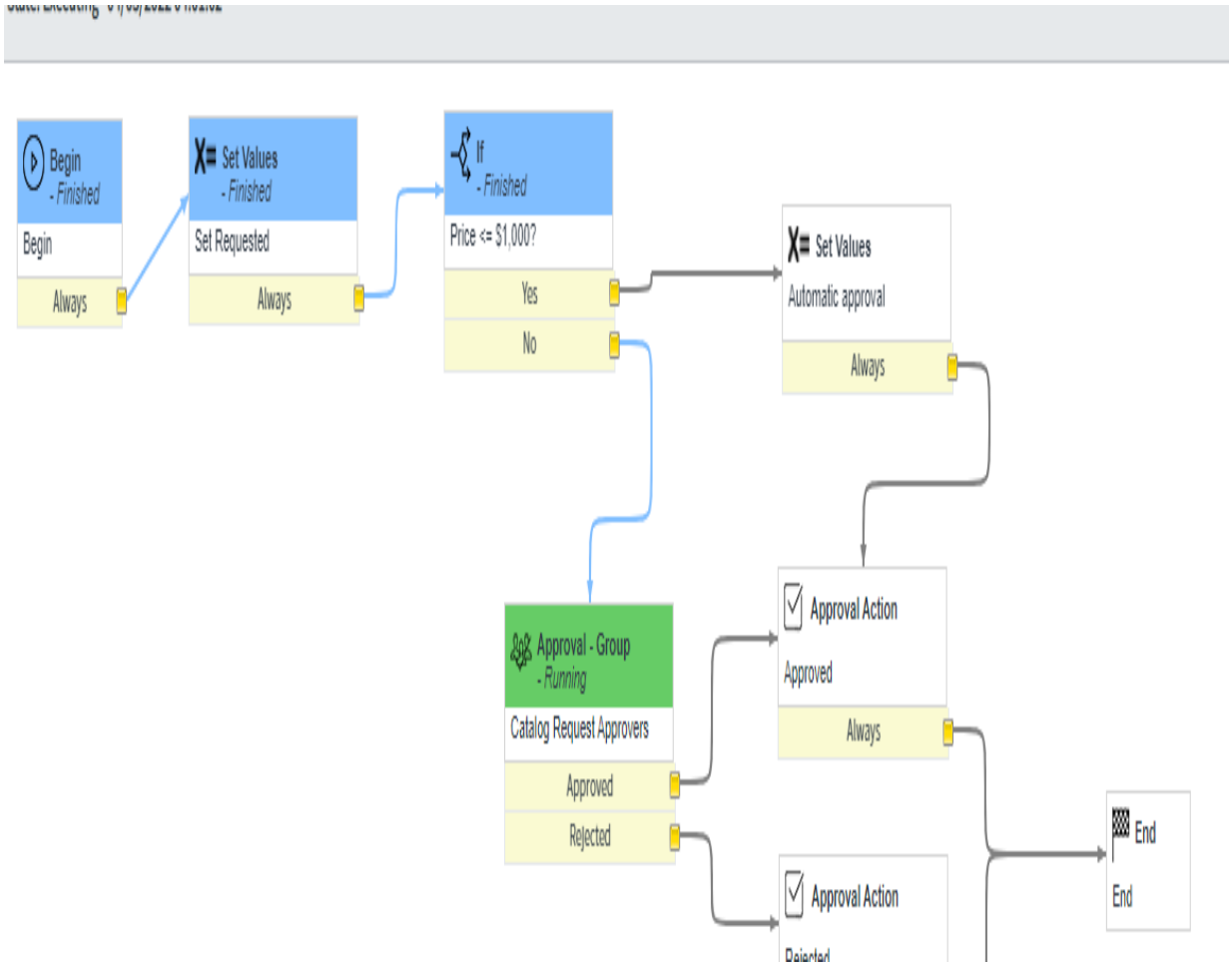
New Workflow button

Click to create a new workflow

Workflow for this module:

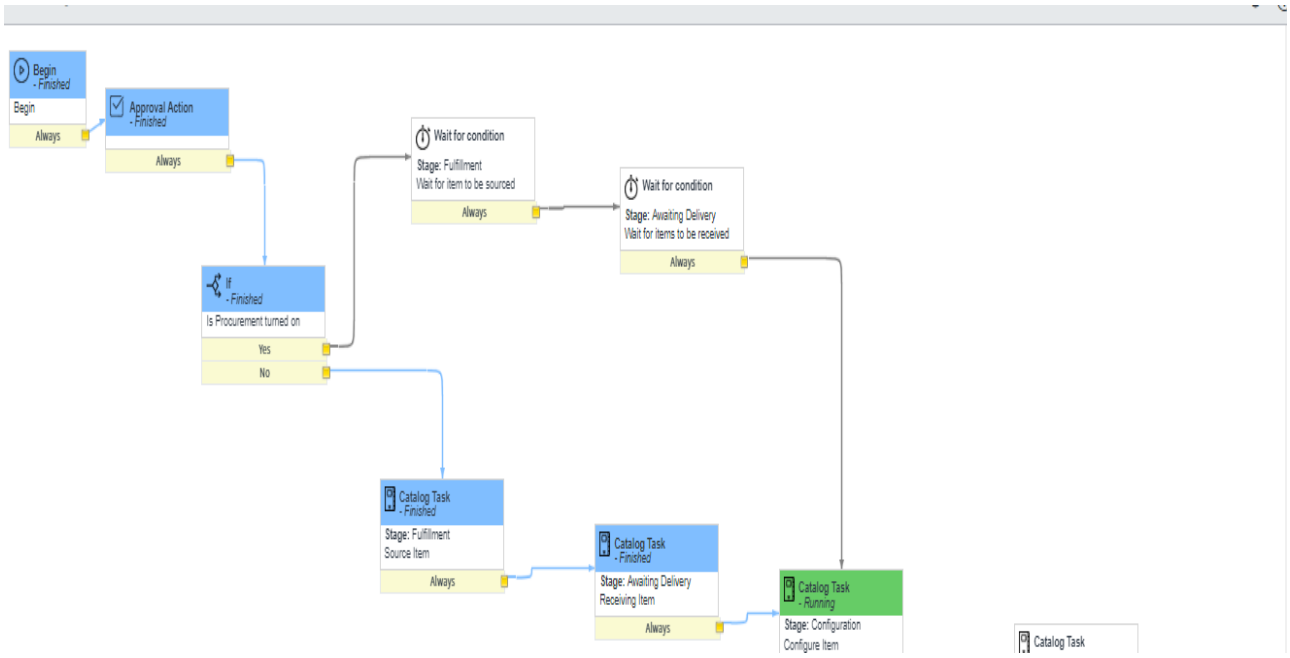
I have created two work flow for this module. First is created for 'Request Item' and second is created for 'Task'.

Workflow for Request Item:



When user Order any bundle then this workflow will work step by step and this is for only Approval the request. If approval will approve then task will generate otherwise task will not generate.

Workflow for Task:



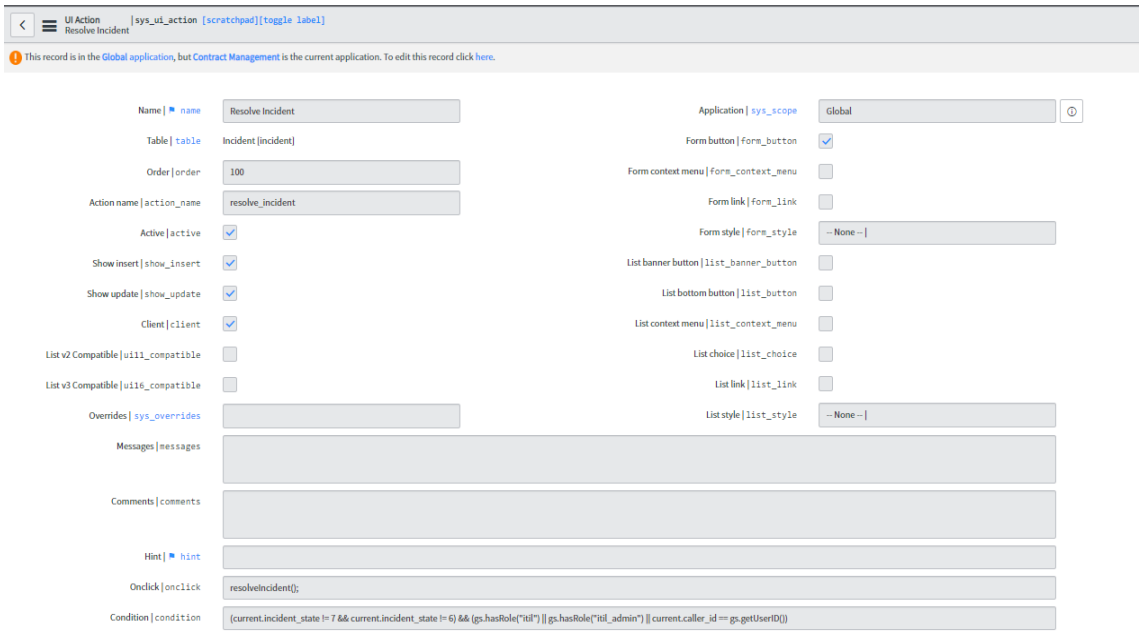
When user Order any bundle then after approval that request task will generate and then this workflow will work until the item will not delivered.

CHAPTER 7

REPORT

7.1 UI Actions:

UI Actions are UI elements that can show up on a form or a list as a button, link, or context menu. When these UI elements are clicked, they execute some JavaScript. Most of the time UI Actions are used to perform some server-side update to a record or records. UI actions can be configured to run either server side, or client side. It is also possible to configure a UI Action to run some code on the client, and other code on the server.



7.2 CLIENT SCRIPTS:

Client Script means scripts (java script) which runs on client side or on client browser. Client Scripts come in four basic types: onload, on Change, on Submit, and onCellEdit. Each type runs under different conditions, and often has a different use than the others.

Warranty expiration alert:

Name | name: Warranty expiration alert
Application | sys_scope: Global
Table | table: Incident [incident]
Active | active: ☑
UI Type | ui_type: Desktop | 0
Inherited | applies_extended: ☐
Type | type: onChange | onChange
Global | global: ☑
Field name | field: cmdb_ci
Isolate script | isolate_script: ☑
Description | description:
Messages | messages:
Script | script:

```
function onChange(control, oldValue, newValue, isLoading, isTemplate) {
    if (isLoading || newValue === '') {
        return;
    }
    g_form.clearMessages();

    var ci = new GlideRecord('cmdb_ci');
    ci.get(newValue);

    if(ci.sys_class_name == 'u_cmdb_ci_dr_panel' || ci.sys_class_name == 'cmdb_ci_computer') {
        var d = new Date();
        var exp = ci.warranty_expiration;

        if(exp.length > 0) {
```

Highlight VIP Caller (Smartphone):

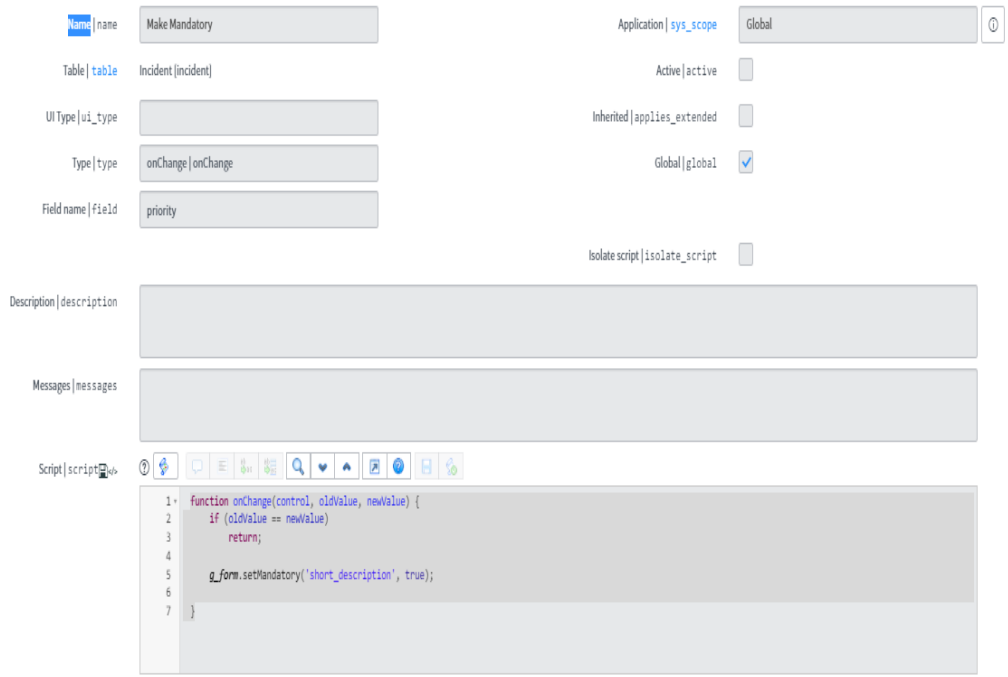
Name | name: Highlight VIP Caller (Smartphone)
Application | sys_scope: Global
Table | table: Incident [incident]
Active | active: ☑
UI Type | ui_type: Mobile / Service Portal | 1
Inherited | applies_extended: ☐
Type | type: onChange | onChange
Global | global: ☑
Field name | field: caller_id
Isolate script | isolate_script: ☐
Description | description:
Messages | messages:
Script | script:

```
function onChange(control, oldValue, newValue, isLoading) {
    // if the caller_id field is not present, then we can't add an icon anywhere
    if (!g_form.hasField('caller_id'))
        return;

    if (!newValue)
        return;

    g_form.getReference('caller_id', function(ref) {
        g_form.removeDecoration('caller_id', 'icon-star', 'VIP');

        if (ref.getValue('vip') == 'true')
            g_form.addDecoration('caller_id', 'icon-star', 'VIP');
```

function onChange(control, oldValue, newValue, isLoading) {

```
// if the caller_id field is not present, then we can't add an icon anywhere
if (!g_form.hasField('caller_id'))
return;
if (!newValue)
return;
g_form.getReference('caller_id', function(ref) {
g_form.removeDecoration('caller_id', 'icon-star', 'VIP');
if (ref.getValue('vip') == 'true')
g_form.addDecoration('caller_id', 'icon-star', 'VIP');
});
}
```

Make Mandatory:



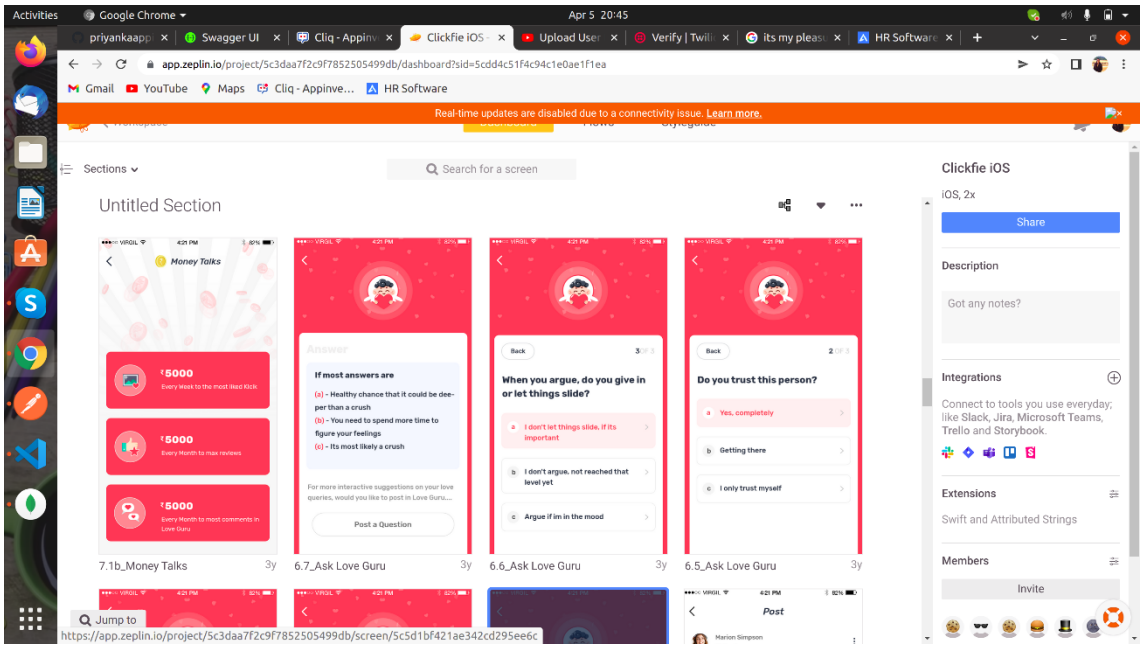

```
function onChange(control, oldValue, newValue) {
 if (oldValue == newValue)
 return;
 g_form.setMandatory('short_description', true);
```

}

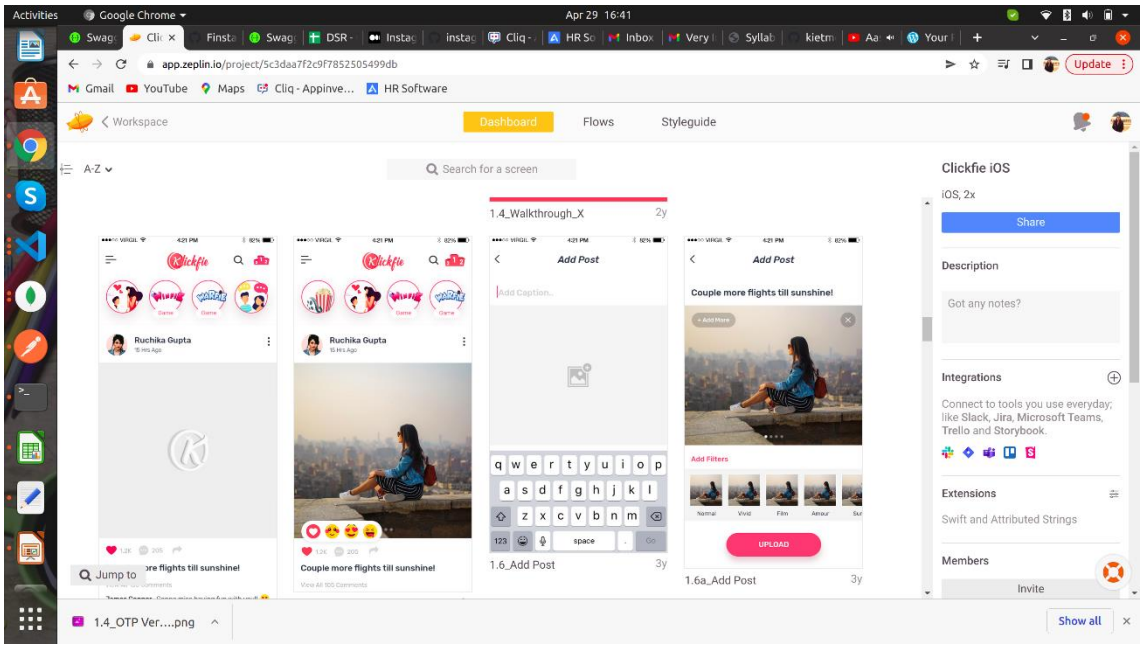
7.3 UI Policy:

UI Policies are a user-friendly way to control whether fields on a form are mandatory, read-only, or even whether they're visible.

Laptop/Desktop Computer CI Must Be Replacement:



Make fields read-only on close:



7.4 Data Policy:

Data Policies enforce data consistency by setting mandatory and read-only field attributes. Unlike UI Policies, Data Policies execute server-side.

Data Policy VS UI Policy:

Data Policies enforce data consistency by setting mandatory and read-only field attributes. Unlike UI Policies, Data Policies execute server-side.

It's often possible to convert a UI Policy to a Data Policy, and vice versa. On the UI Policy form, you'll find a UI Action called convert this to Data Policy.

There is also no Advanced view, no scripted conditions, and no UI or server-side scripting possible through a data policy (except to the extent that scripting is possible inside of any condition builder to make a determination about whether the data policy should run).

7.5 Business rule:

Business rules is the server-side script which means that it will execute on server or database. Business rule runs faster than other script in ServiceNow. The script or code

written in business rule area will get executed when record is inserted, displayed, updated, deleted or when table is queried.
The four types of business rule in ServiceNow are:]

• Display Business Rule

• Before Business Rule

• After Business Rule

• Async Business Rule

7.5.1 Display Business Rule in ServiceNow:

Code written in display business rule get executed before the form is presented to the user and just after the data is read from the database.

For e.g., you have written the code that when xyzzy user click on information box then only data related to that user specific country will get displayed to user. It means that user from US can see US specific data and user from India can see India specific data

7.5.2 Before Business Rule in ServiceNow:

Code written in before business rule get executed when user submits the form and data is not saved in database.
Let's say User click on submit ***button --> Before business rule code executes --> information will save in database.***

For e.g. Let's say you have written the code that when user click on submit button then some extra information which in not filled by user such as user current location, user manager name and user past activities will get save when user click on submit button.

7.5.3 After Business Rule in ServiceNow:

Code written in after business rule get executed when user submits the form and data saved in database. Let's say ***User click on submit button --> data saved in database --> Now after business rule code get executed.***

For e.g., there is parent incident and child incident and you want that related child incident will get closed automatically after the parent incident get closed by user.

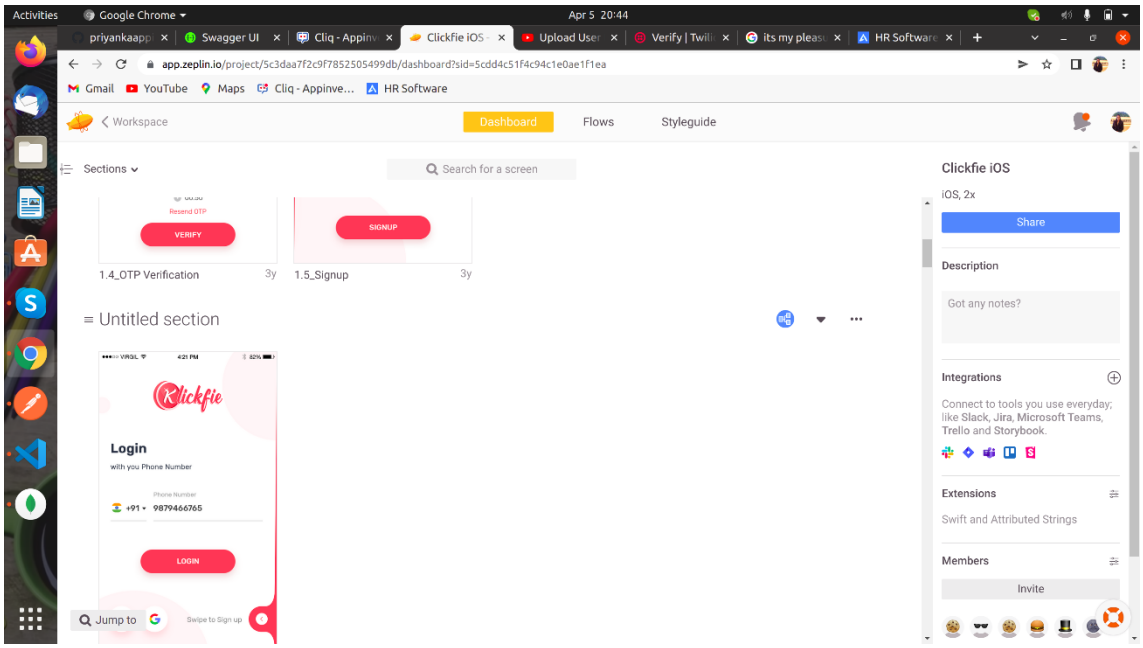
7.5.4 Async Business Rule in ServiceNow:

Async business rules are like after business rule but it runs in the background simultaneously with other processes. Means async business rule run after the data is saved into the database.

Running on background means that use can proceed with other functionality and code will run on the background which will not impact the user while doing other transitions.

For e.g., Incident ticket is in pending customer action status and will be auto closed after 5 days if user did not provide any update on incident ticket.

Response from User:



7.6 Notifications:

Template:

```
import user from "../model/userModel";
import { Request, Response } from "express";
import { HydratedDocument } from "mongoose";
import { iUser, userData } from "../interface/userInterface";
import { otpCode } from "../interface/otpInterface";
import Otp from "../model/otpModel";
import { otp } from "../services/otpService";
import { generateToken } from "../services/jwtServices";
import { constant } from "../constant/constant";
import { responses } from "../helper/response";

const signUp = async (req: Request, res: Response): Promise<void> => {
  try {
    const { name, email, phoneNumber, imageUrl }: userData = req.body;
    const userExist: HydratedDocument<iUser> | null = await user.findOne({
      email,
      phoneNumber,
    });
    if (userExist) {
      responses.status.statusCode = 400;
      responses.status.status = false;
      responses.status.message = constant.message.signUp;
      res.status(constant.statusCode.invalid).json(responses.status);
    } else {
      const User: HydratedDocument<iUser> | null = new user({
        name: name,
        email: email,
        phoneNumber: phoneNumber,
        imageUrl: imageUrl,
      });
      const data: HydratedDocument<iUser> | null = await User.save();
      const otpCode: string = await otp.generateOTP(phoneNumber);
      const dataCode = new Otp({
        code: otpCode,
```

```
      });
      const msg: HydratedDocument<otpCode> | null = await dataCode.save();
      const code = otpCode;
      responses.status.message = constant.message.signUpMsg;
      responses.status.statusCode = 200;
      responses.status.status = true;
      res
        .status(constant.statusCode.success)
        .json({...responses.status,OTP:code});
    }
  } catch (err) {
    responses.status.statusCode = 406;
    responses.status.message = constant.message.signUpFail;
    responses.status.status = false;
    res.status(constant.statusCode.alreadyLoggedIn).json( responses.status);
  }
};

const login = async (req: Request, res: Response): Promise<void> => {
  try {
    const { phoneNumber }: userData = req.body;
    const userActive: HydratedDocument<iUser> | null = await user.findOne({
      phoneNumber,
      is_active: true,
    });
    if (userActive) {
      const otpCode: string = await otp.generateOTP(phoneNumber);
      const data = new Otp({
        code: otpCode,
      });
      const msg: HydratedDocument<otpCode> | null = await data.save();
      const code = otpCode;
      responses.status.statusCode = 200;
      responses.status.status = true;
```

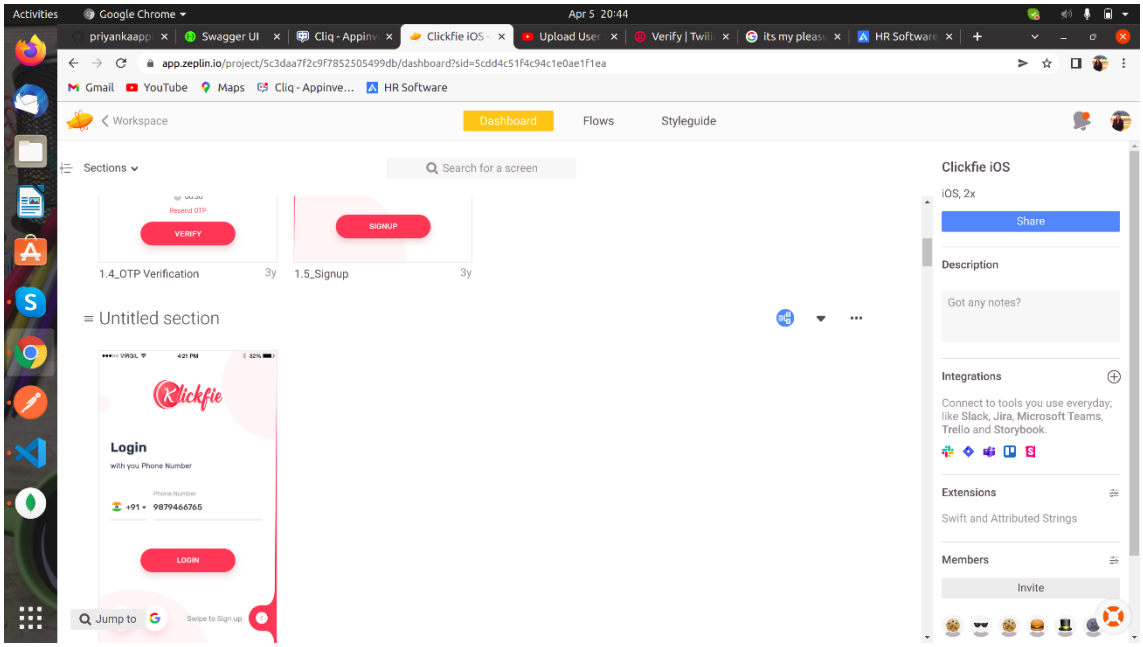
```
      responses.status.message = constant.message.otpMsg;
      res
        .status(constant.statusCode.success)
        .json({...responses.status, code: otpCode} );
    } else {
      responses.status.message = constant.message.loginInvalidMsg;
      responses.status.statusCode = 400;
      responses.status.status = false;
      res.status(constant.statusCode.invalid).json(responses.status);
    }
  } catch (err: any) {
    responses.status.message = constant.message.loginFailed;
    responses.status.statusCode = 401;
    responses.status.status = false;
    res.status(constant.statusCode.loginFailed).json(responses.status);
  }
};

const verifyOtp = async (req: Request, res: Response): Promise<void> => {
  try {
    const { phoneNumber, code }: userData = req.body;
    const data: HydratedDocument<iUser> | null = await user.findOne({
      phoneNumber,
    });
    const info: HydratedDocument<otpCode> | null = await Otp.findOne({
      code,
    });
    if (data && info && data?.phoneNumber && info?.code) {
      if (
        data.phoneNumber === req.body.phoneNumber &&
        info.code === req.body.code
      ) {
        const jwtToken = await generateToken(data._id);
        console.log(data.phoneNumber, code);
```

```
        responses.status.message = jwtToken;
        responses.status.statusCode = 200;
        responses.status.status = true;
        responses.status.message = constant.message.verifyMsg;
        res
          .status(constant.statusCode.success)
          .json({...responses.status, data: jwtToken });
      } else {
        responses.status.statusCode = 400;
        responses.status.message = constant.message.authenticationFailed;
        responses.status.status = false;
        res.status(constant.statusCode.invalid).json(responses.status);
      }
    } else {
      responses.status.statusCode = 400;
      responses.status.message = constant.message.invalidNumber;
      responses.status.status = false;
      res.status(constant.statusCode.invalid).json(responses);
    }
  } catch (error: any) {
    responses.status.statusCode = 500;
    responses.status.message = constant.message.serverError;
    responses.status.status = false;
    res.status(constant.statusCode.serverError).json(responses);
  }
};

const socialMediaFb = async (req: Request, res: Response): Promise<void> => {
  try {
    const { name, email }: userData = req.body;
    const userExist: HydratedDocument<iUser> | null = await user.findOne({
      email,
    });
    if (userExist) {
```

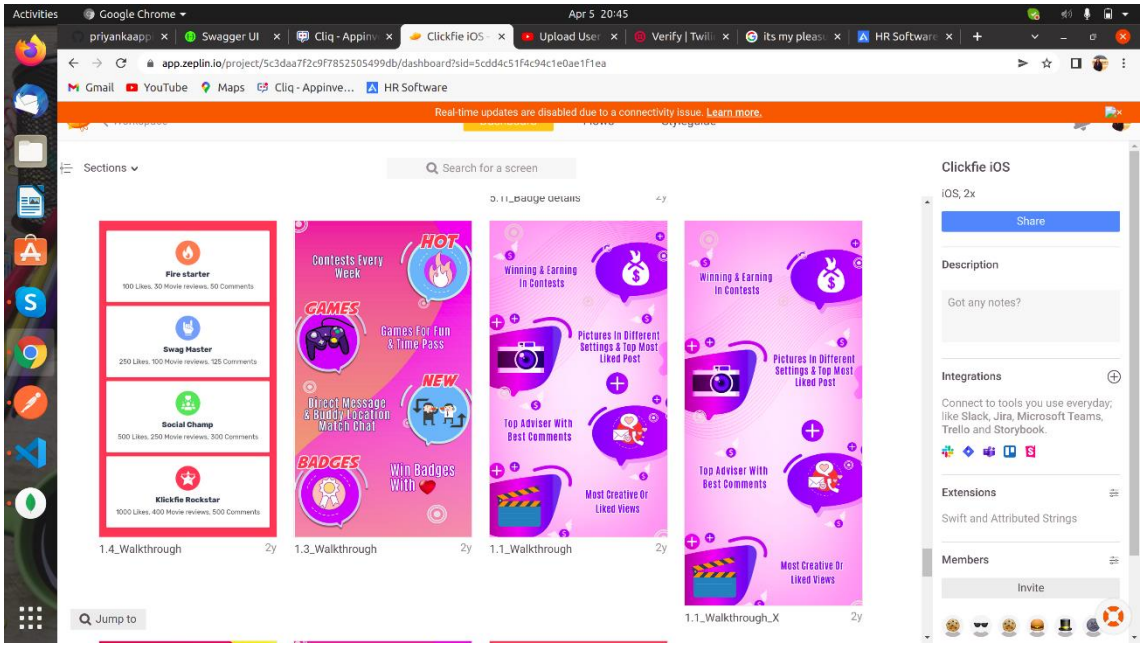
```
      responses.status.statusCode = 200;
      responses.status.status = true;
      responses.status.message = constant.message.signUp;
      res.status(constant.statusCode.success).json(responses.status);
    } else {
      const User: HydratedDocument<iUser> | null = new user({
        name: name,
        email: email,
      });
      const data: HydratedDocument<iUser> | null = await User.save();
      responses.status.message = constant.message.signUpMsg;
      responses.status.statusCode = 200;
      responses.status.status = true;
      res
        .status(constant.statusCode.success)
        .json(responses.status);
    }
  } catch (err: any) {
    console.log(err);
    responses.status.message = constant.message.serverError;
    responses.status.statusCode = 500;
    responses.status.status = false;
    res.status(constant.statusCode.serverError).json(responses.status);
  }
};

const socialMediaGoogle = async (
  req: Request,
  res: Response
): Promise<void> => {
  try {
    const { name, email }: userData = req.body;
    const userExist: HydratedDocument<iUser> | null = await user.findOne({
      email,
```

```
    });
    if (userExist) {
      responses.status.statusCode = 200;
      responses.status.status = true;
      responses.status.message = constant.message.signUp;
      res.status(constant.statusCode.success).json( responses.status);
    } else {
      const User: HydratedDocument<iUser> | null = new user({
        name: name,
        email: email,
      });
      const data: HydratedDocument<iUser> | null = await User.save();
      responses.status.message = constant.message.signUpMsg;
      responses.status.statusCode = 200;
      responses.status.status = true;
      res
        .status(constant.statusCode.success)
        .json(responses.status);
    }
  } catch (err: any) {
    responses.status.message = constant.message.serverError;
    responses.status.statusCode = 500;
    responses.status.status = false;
    res.status(constant.statusCode.serverError).json(responses.status);
  }
};

export default {
  signUp,
  login,
  verifyOtp,
  socialMediaFb,
  socialMediaGoogle,
};
```

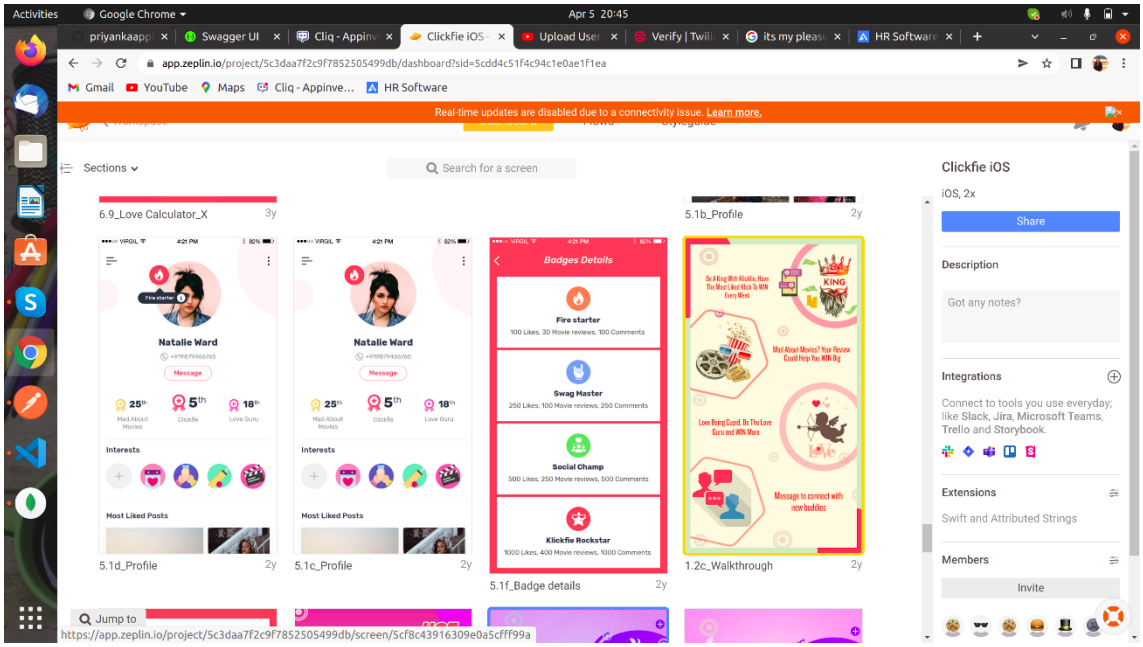
Reset Password:



Event registry:



Closed:



assigned to my group:

Record Producer:

Record producers are a great piece of ServiceNow functionality that allows for the creation of records in any table via the standard Service Catalog interface.

A record producer is a specific type of catalog item that allows end users to create task-based records, such as incident records, from the service catalog. Use record producers to provide a better end-user experience instead of using the regular task-based form for creating records.

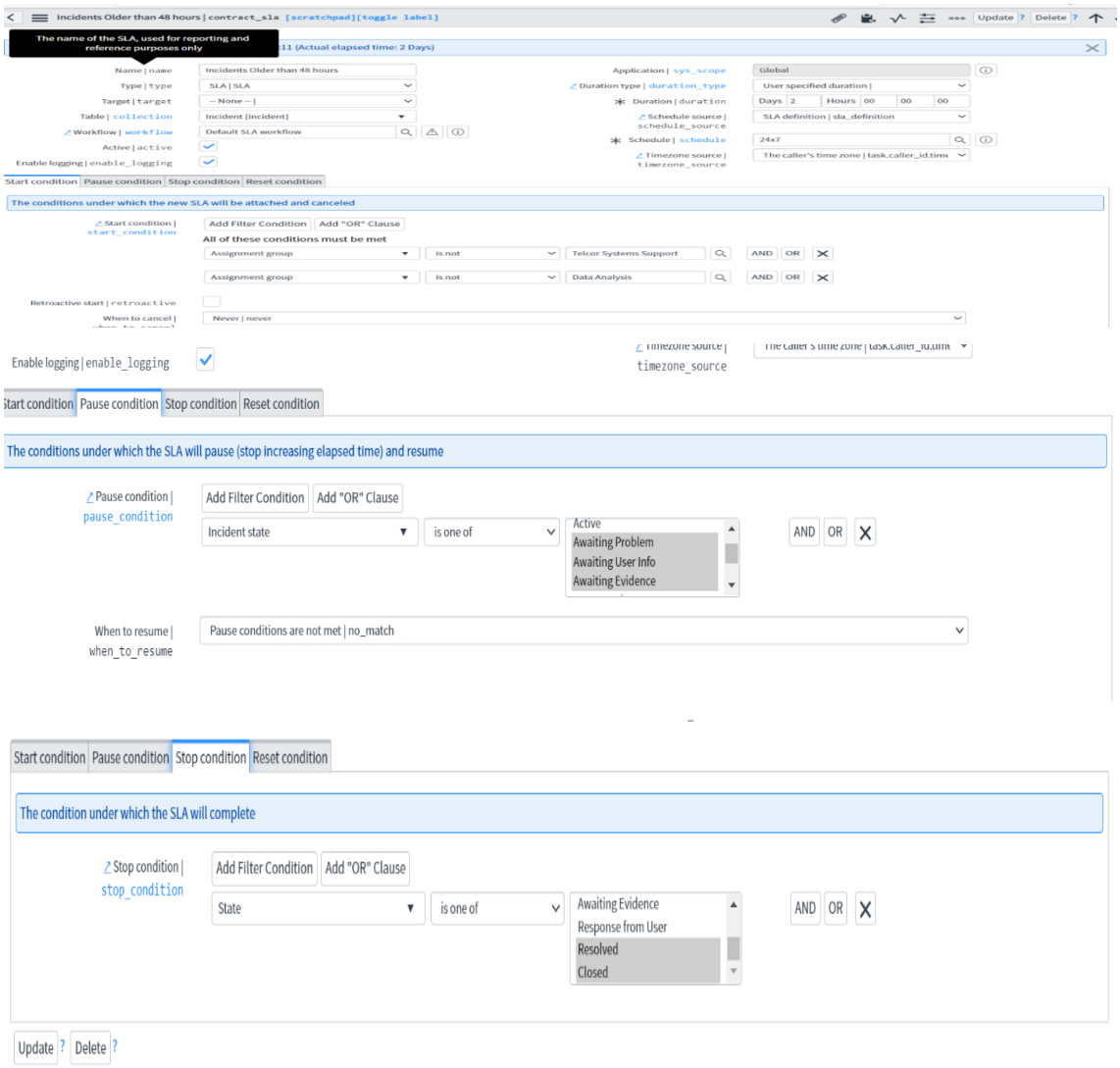
7.7 Service Level Agreement:(SLA)

Configure a Service Level Agreement (SLA) to define a set amount of time for a task to reach a certain condition. This ensures that incidents are closed or resolved according to the expectations set for customers.

An SLA definition record defines the timings, conditions, workflows, and other information required to create task SLAs. This will enable you to use an SLA system for your group's tasks.

This feature is available to those with the dept_admin role only.

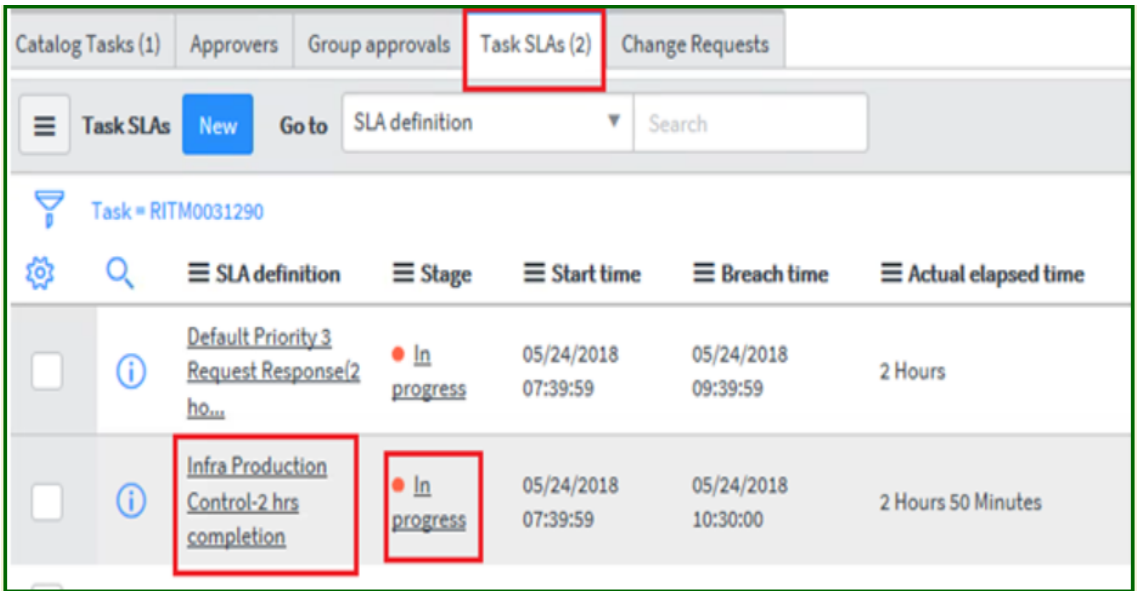
Older than 48 hours:



How to check SLAs

- To view the SLAs of the current RITM, navigate to the related list at the bottom of the RITM

- The SLAs will be displayed along with the Start time and Breach time for Each SLAs



Understanding the Request, RITM, Task

Here we see basic terminology circling around RITM, REQ and TASK. These terms are very important while working on ITSM module in ServiceNow.

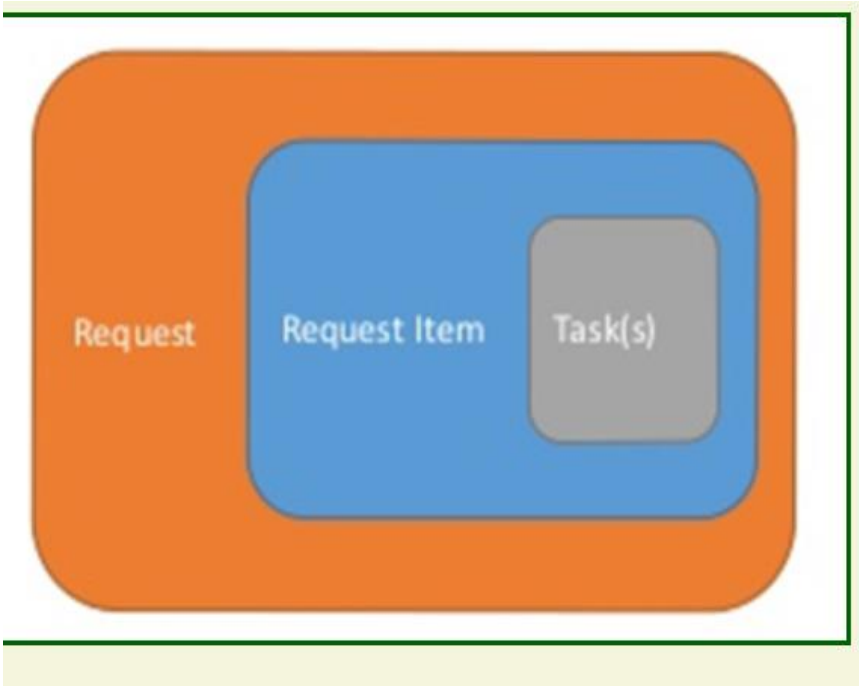
Understanding Request

Request: After clicking the correct Catalogue Item the corresponding Catalogue item form will get displayed. This form needs to be filled up with adequate information. Click on submit button. Once we submit the form, we can find the Request number.

For 1 Request there will be multiple RITM (requested Items)

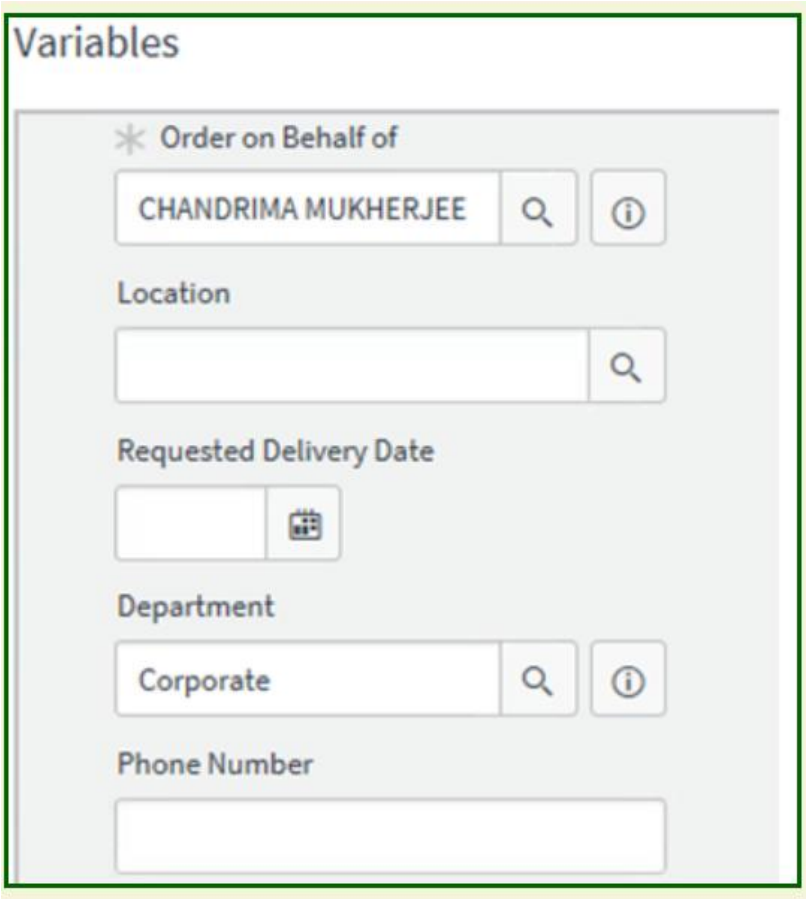
Understanding the RITM

- Each Request can have request item (RITM).
- Each RITM can have multiple tasks assigned to different groups for fulfilment
- A RITM is marked as complete only when all its Task and Subtask gets completed



How to check for the variables in RITM

- The operator might like to view the data as entered by user in the service portal form.
- This data gets captured in Variables in RITM.
- To view this data, click on any of the RITM and scroll down to view the Variables section



Understanding the Catalogue Task

- One liner: Request –> RITM –> Catalogue Task

- Request ticket starts with REQ

- Request Item ticket starts with RITM

- Task ticket starts with SCTASK

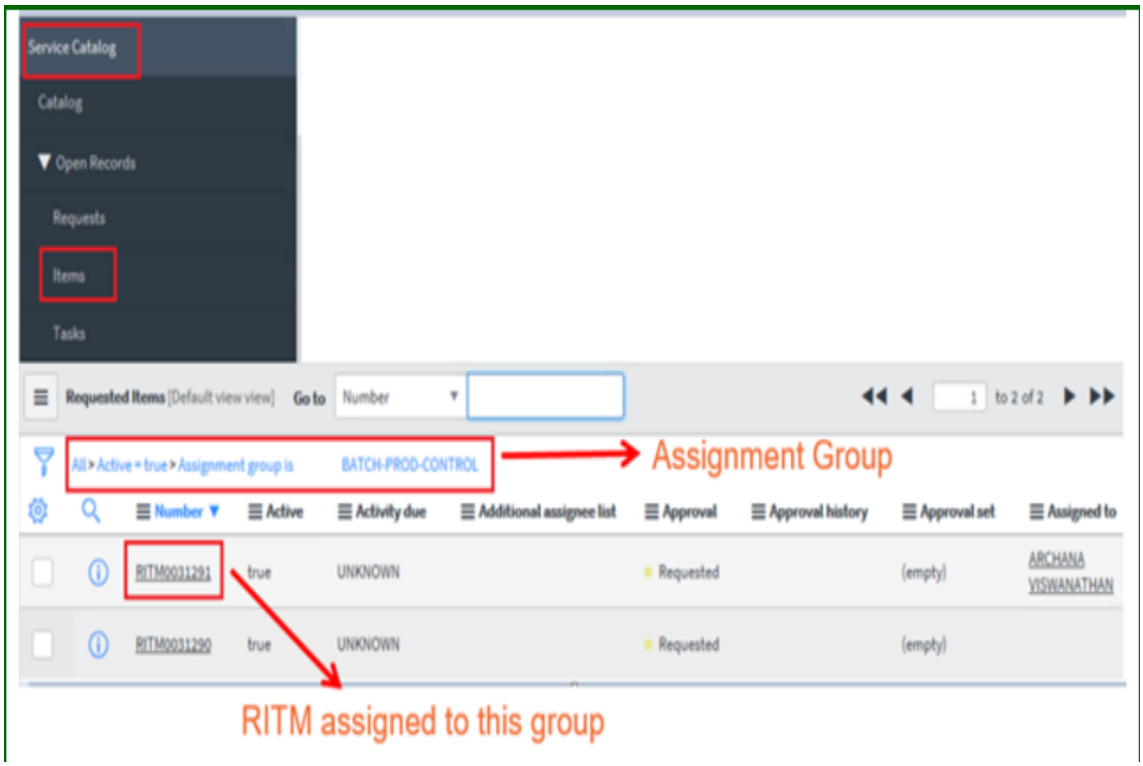
How to check for the RITM?

- Login into ServiceNow

- From the left Navigation panel click on Service Catalogue > items

- All open RITMs assigned to your group will be displayed in the list view

- Filter the records further with the respective Approval



How to Complete a request Item

- To complete a RITM the associated Task needs to be completed first.

- If all the task of a RITM gets completed the RITM will automatically get to Closed Complete status.

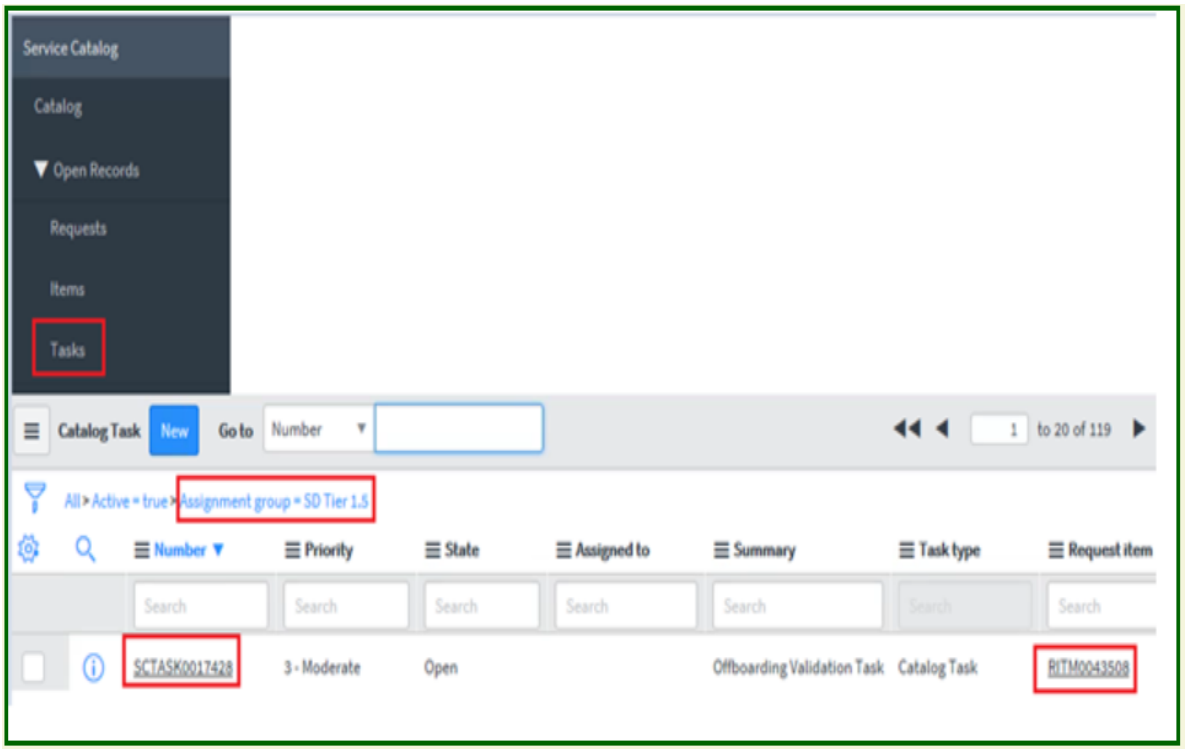
- Attached SLA will get completed.

How to check for the catalogue task details?

- Login into ServiceNow.

- From the left Navigation panel click on Service Catalogue > Tasks.

- All open Tasks assigned to your group will be displayed in the list view

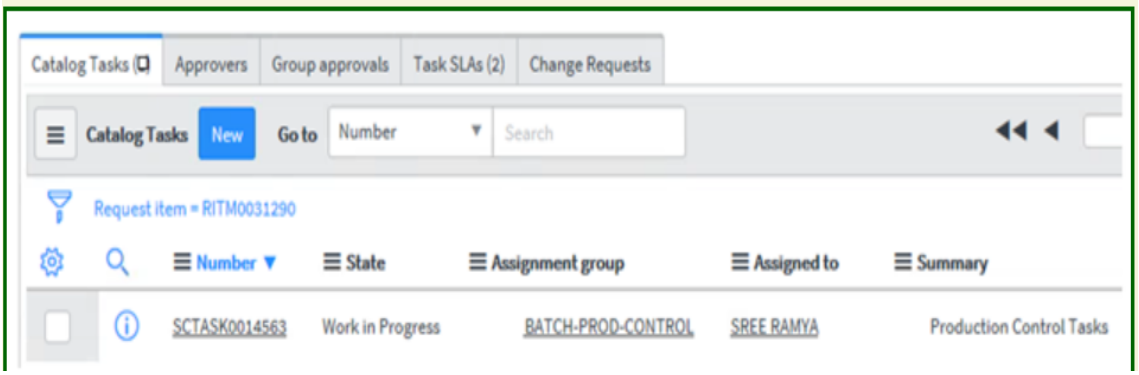


How to check catalog task in RITM

- As mentioned earlier an RITM can have one or more catalog tasks performing specific actions for completion of the request item.

- Click on the desired RITM and scroll down to the related list at the bottom.

- Operator now can view the catalog task along with the State, Assignment Group, whom the Task is currently assigned etc.



CHAPTER 8

RESULT

Developer Project Equipment:

Developer Project Equipment submits a single service catalog request that generates several items.

For example, a New Employee Hire Developer Project Equipment can contain several items that new employees commonly need, such as business cards, computer, and cell phone. After selecting this Developer Project Equipment, the customer can then provide information about the new employee, including location and job title. The Developer Project Equipment then submits an order for catalog items like business cards, based on the details provided.

Administrators and catalog administrators can create Developer Project Equipment's for the service catalog.

Developer Project Equipment's can be run automatically, generating a set of ordered items without needing to manually submit a service catalog request. For example, an onboarding workflow for a new employee can automatically run an Developer Project Equipment to order items for that employee.

The script field in a Developer Project Equipment can be used to add or remove catalog items to or from the Developer Project Equipment. It can be added to the Developer Project Equipment form by configuring the form layout.

- To add a catalog item that is not added to the Developer Project Equipment via a rule base, write the following code in the script field:

```
guide.add("<sys_id_of_cat_item>")
```

❖ To remove a catalog item that is added to the Developer Project Equipment via a rule base, write the following code in the script field:

```
guide.remove("<sys_id_of_cat_item>")
```

➢ **Create a Developer Project Equipment**

You can create a Developer Project Equipment with a two-step or three-step ordering process.

Request a Developer Project Equipment

Once created, Developer Project Equipment's present the customer with a three-step ordering process by default.

➢ **Create a Developer Project Equipment rule to add a catalog item**

You can add a catalog item to a Developer Project Equipment using specific rules.

➢ **Create a Developer Project Equipment variable**

You can create variables within a Developer Project Equipment.

➢ **Avoid enforcement of a mandatory field**

You can avoid enforcing a mandatory field within tabs.

➢ **Run Developer Project Equipment's automatically**

You can run a Developer Project Equipment automatically from within a workflow or a server script, passing parameters to that Developer Project Equipment to define variable values.

8.1 Create a Developer Project Equipment

You can create a Developer Project Equipment with a two-step or three-step

ordering process.

1. Navigate to Service Catalog > Catalog Definition > Developer Project Equipment's.

2. Click New.

3. Fill in the fields as appropriate.

4. Right-click the form header and click Save.

5. In the Rule base related list, define the rules that determine which items are included in an order.

6. (Optional) In the Variables related list, define any variables required.

Order guide

Field	Description
Name	Order guide name that appears in the catalog.
Active	Check box to indicate whether the order guide is active or not.
Category	Category heading under which the order guide appears in the catalog.
Two step	Check box to enable two step ordering instead of the default three-step ordering, omitting the final step. With two step ordering, selecting **Check out** submits the request immediately, then displays the order confirmation screen. This is not applicable for Service Portal.

	guide.
Short description	Summary of the order guide purpose.
Description	Description that appears on the first page of the order guide. Apply formatting with the HTML editor.
Meta	Comma-separated list of tags used to search for the order guide. See Configure keyword search for catalog items.
Hide on Service Portal	If selected, the order guide is not available on Service Portal.
Picture	[Optional] Image of the item.

8.2 Request a Developer Project Equipment

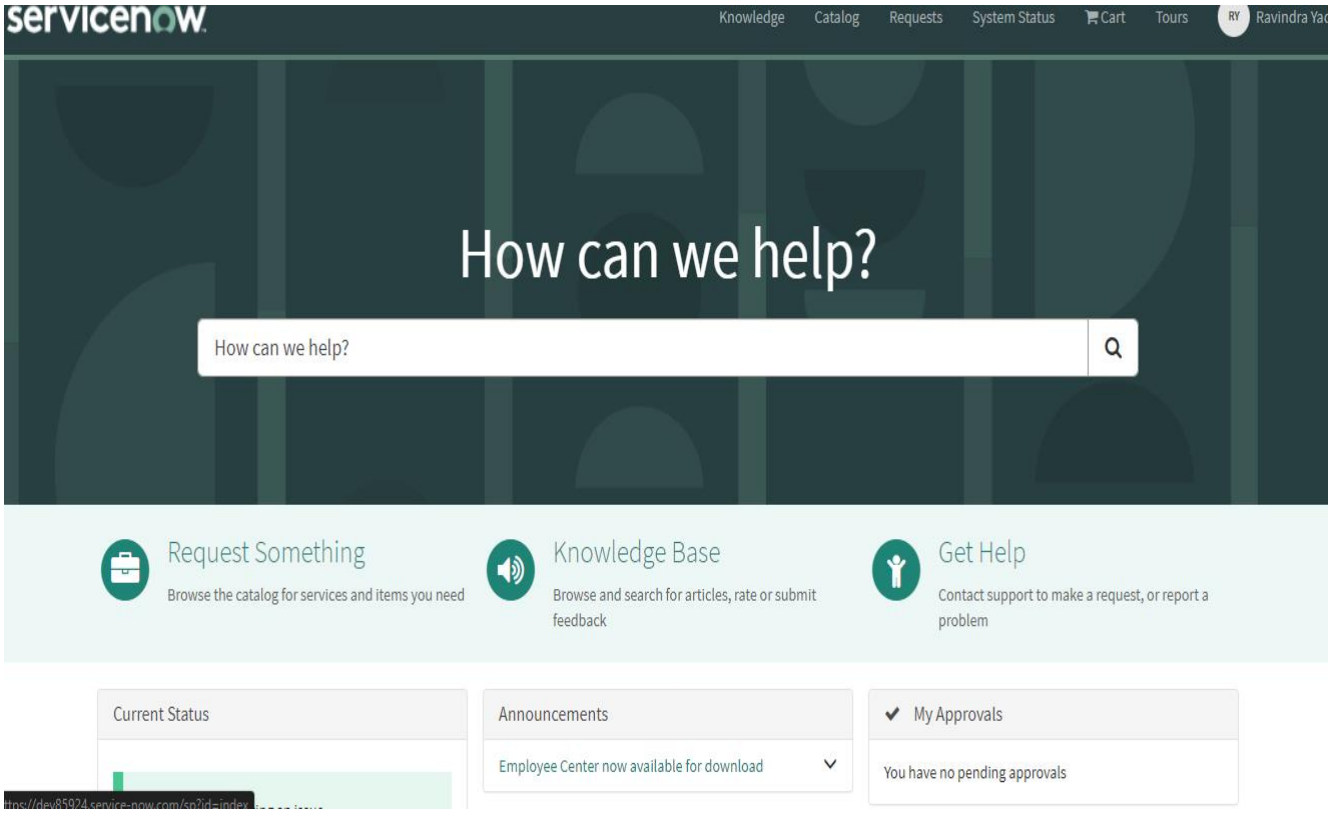
Once created, Developer Project Equipment's present the customer with a three-step ordering process by default.

a. **Describe Needs:** Enter information as prompted. Developer Project Equipment rules evaluate this information to determine which catalog items to order.

b. **Choose Options:** Select configuration options for the ordered items and provide any additional information needed, including attachments.

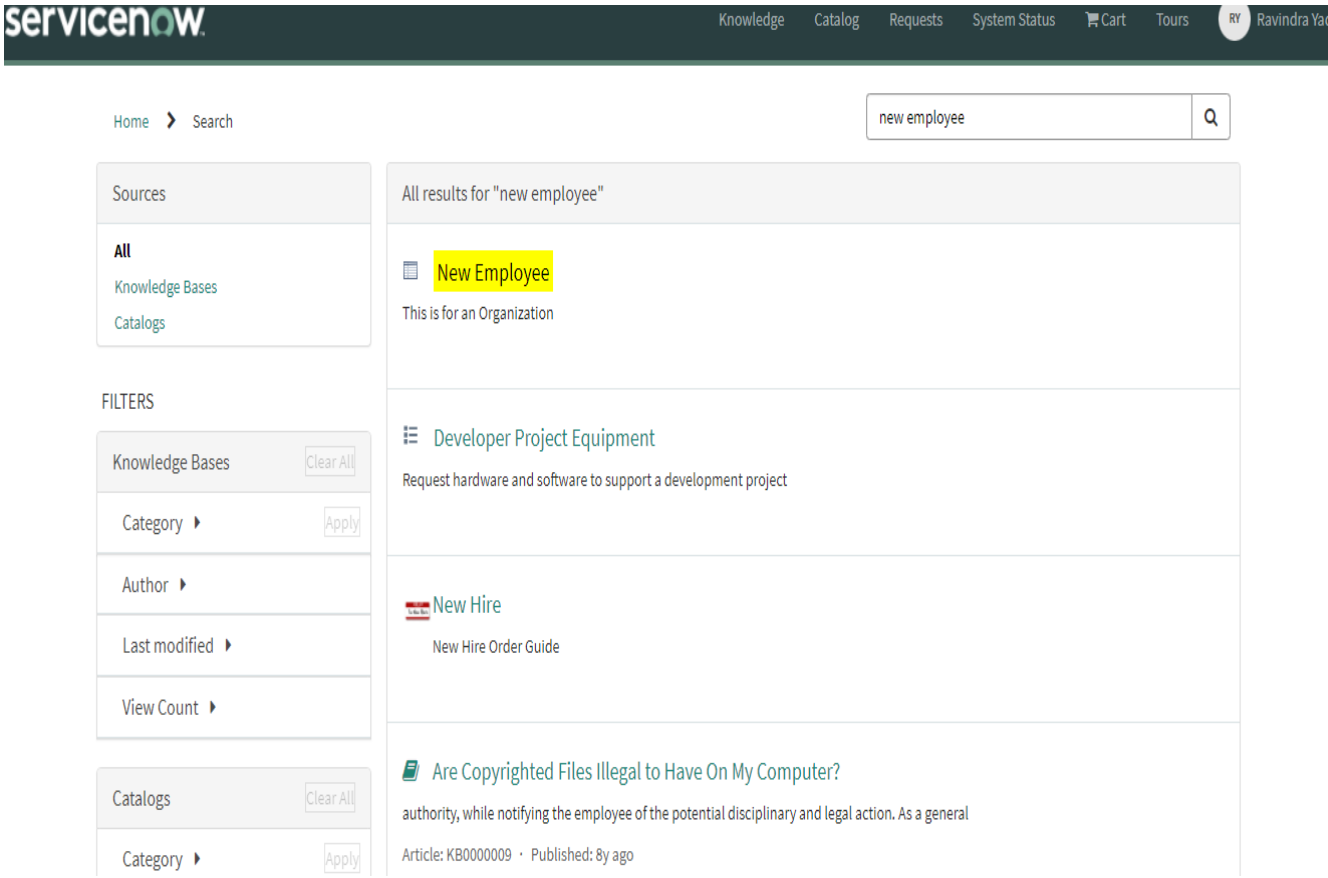
c. **Check Out:** Review and edit item information, then click Submit Order to place the request. If the requested item was ordered as part of a Developer Project Equipment, the Developer Project Equipment field on the Requested Item form shows the Developer Project Equipment name.

Review an Developer Project Equipment example

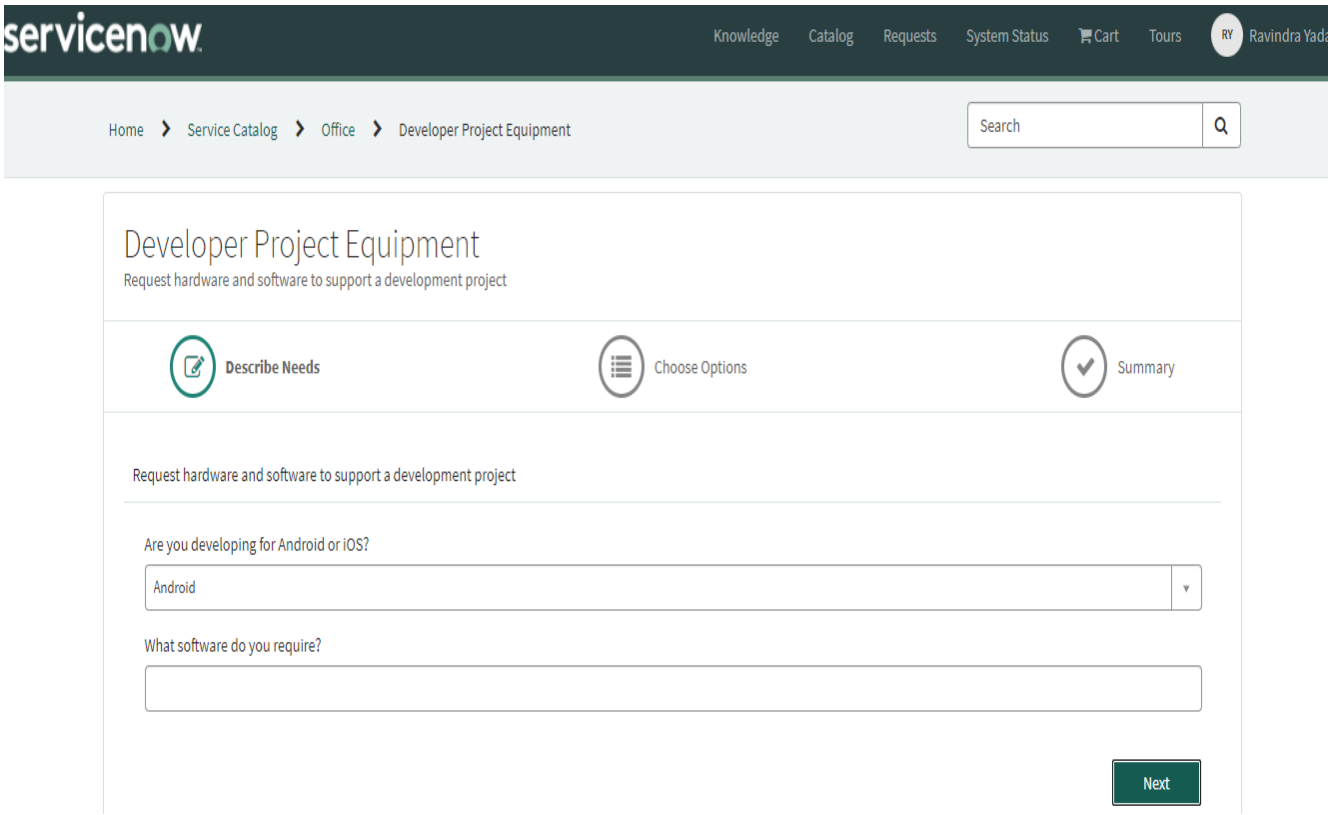


Consider a scenario where you have a New Employee Hire Developer Project Equipment that provides services and items as part of the onboarding process.

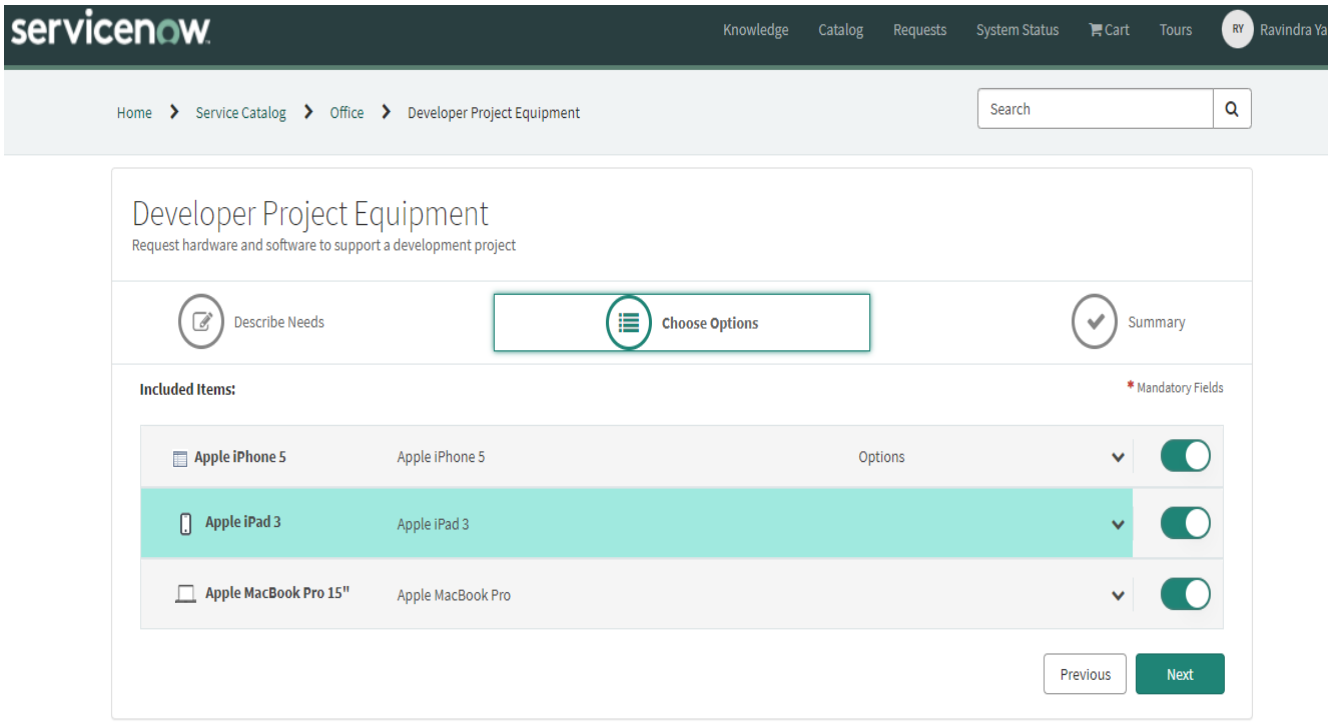
This will be the first page or we can sit index page from where user can order their items. It will be for user which will be visible for everyone. Here in search bar, we will find our application and fill that.



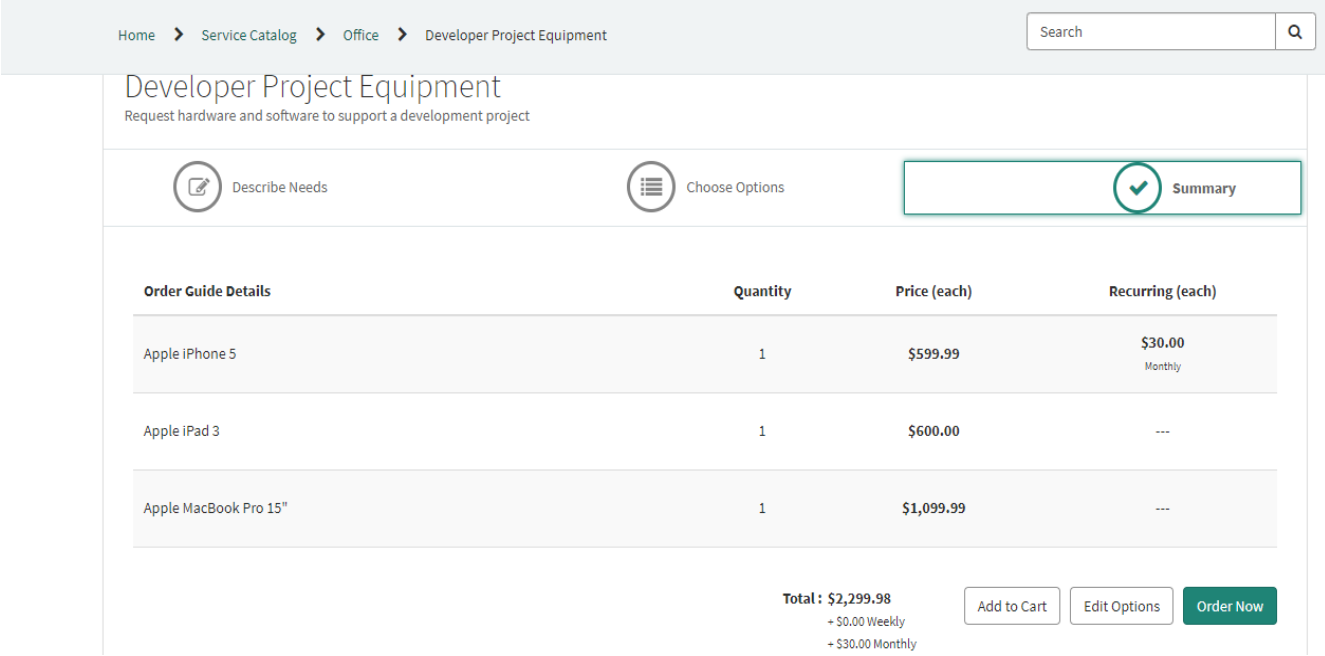
After search in search bar, it will be open like this screen shot and the name of our application is 'Developer Project Equipment', we will click at that application.



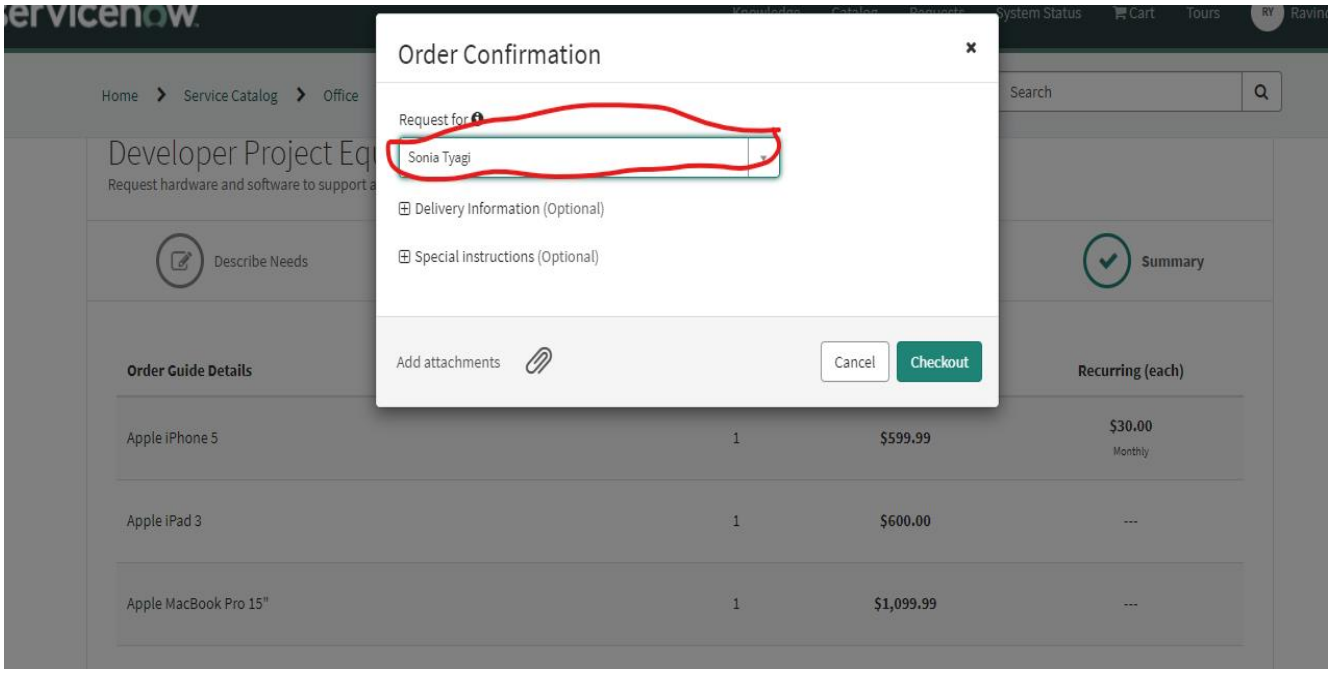
The User Interface will be like this screenshot and user can fill the form for request item. There can be add or remove field on user's demand. This will be first step of ordering items.



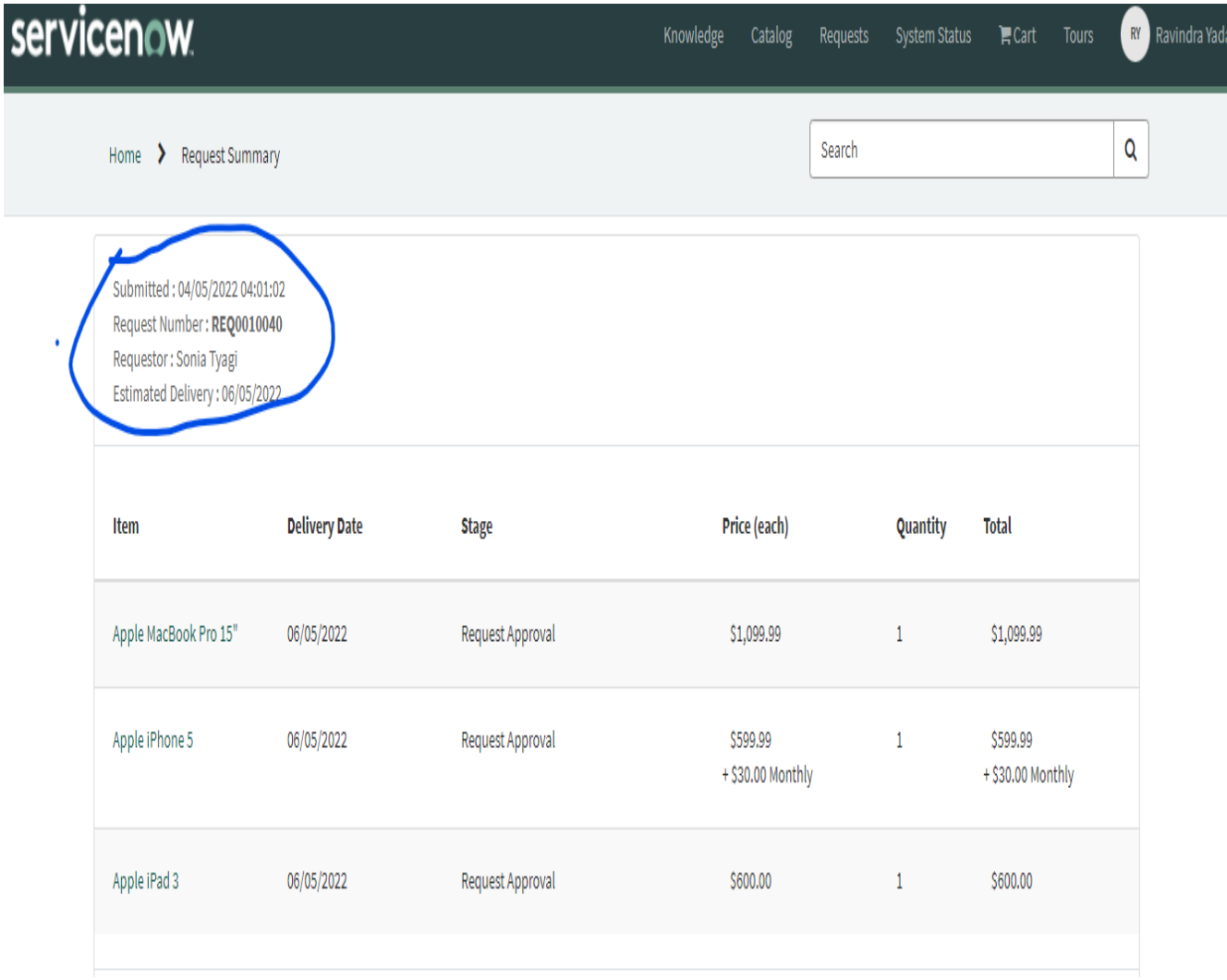
This will be second step where user can see the bundle of some items and he can remove item from click on button. After that user will click on next button for next step.



This will be the next step for user to order item. This page will show price of all item which user wants to order and after verify the price he will click on 'Order Now' Button for last step.



Here at ‘Requested For’ field user can select those names for whom he wants to order the items and after that he will click ‘checkout’ Button.



Here in the blue circle will be details of order item. Here will be the name of requested for user and ‘Request Number’, Order Date and Estimated Delivery Date. Using this Request Number user and others can trace this requested order.

Service Desk can see the all requested item from **‘sc_request_list.do’** table.

Vendor Cert | Global | System Administrator | REQ0010040

Request REQ0010040 | sc_request [scratchpad][toggle label]

Follow | Update | Add New Item | Cancel Request | Copy | Dele

Number | number	REQ0010040	Opened | opened_at	04/05/2022 04:01:02
Requested for | requested_for	Sonia Tyagi	Opened by | opened_by	Ravindra Yadav
Location | location		Approval | approval	Requested | requested
Due date | due_date	06/05/2022 04:01:01	Request state | request_state	Pending Approval | requested
Price | price	$2,299.98		
Description | description			
Short description | short_description			
Special instructions | special_instructions			

Update ? | Add New Item ? | Cancel Request ? | Copy ? | Delete ?

Related Links

Show Workflow?

Request REQ0010040 | sc_request [scratchpad][toggle label]

Follow | Update | Add New Item | Cancel Request | Copy | Del

Related Links

Show Workflow?

Workflow Context?

Requested Items (3) | Approvers (1) | Group approvals (1) | Recurring Prices (1)

Requested Items | sc_req_item Search for text | zztextsearchyy Search 1 to 3 of 3

Request = REQ0010040

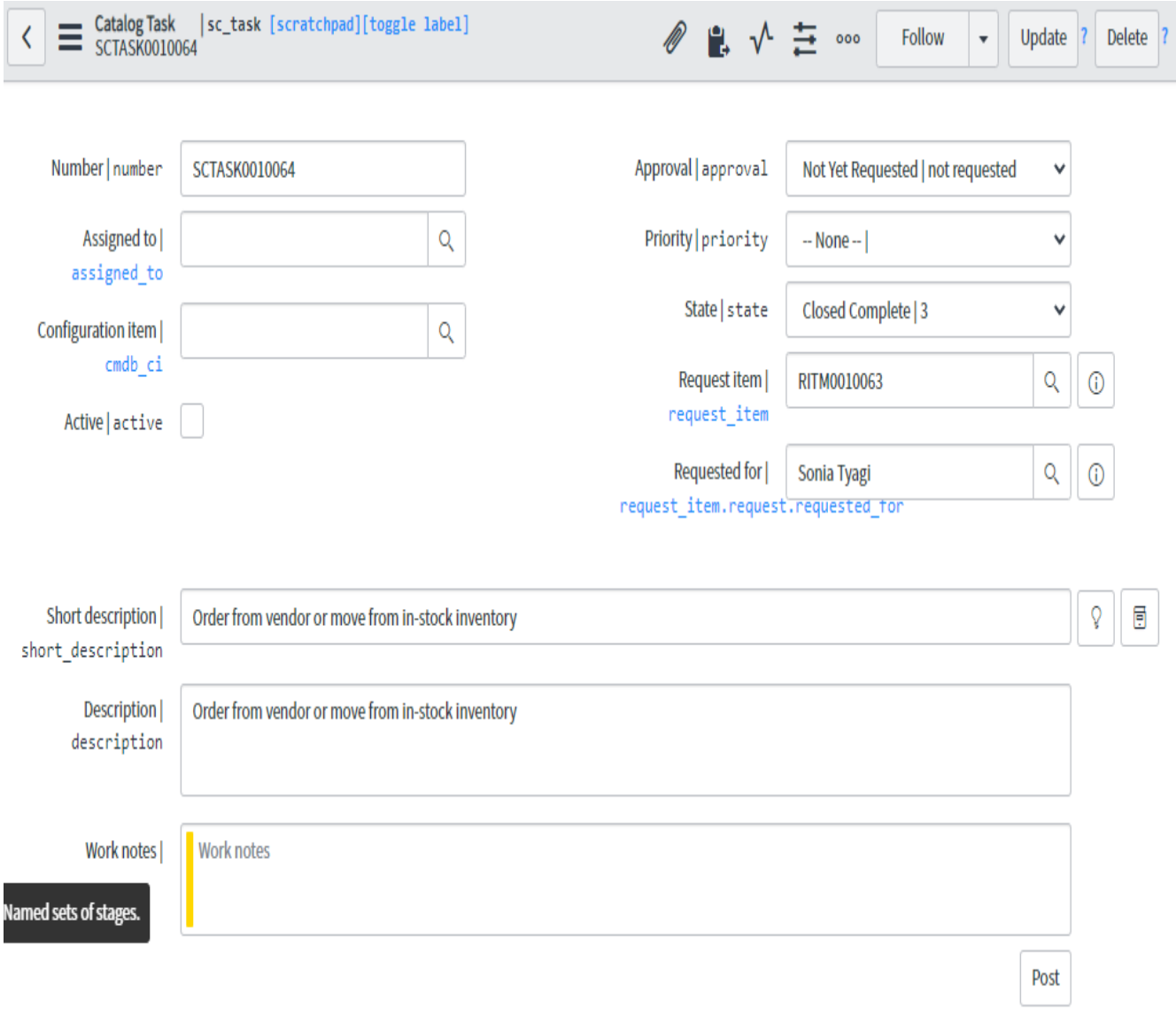
Number number	Approval approval	Quantity quantity	Catalog sc_catalog	Item cat_item	Due date due_date	Price price	Assigned to assigned_to	Stage stage
RITM0010065	Not Yet Requested	1	(empty)	Apple MacBook Pro 15"	06/05/2022 04:01:01 2d from now	$1,099.99	(empty)	
RITM0010064	Not Yet Requested	1	(empty)	Apple iPhone 5	06/05/2022 04:01:01 2d from now	$599.99	(empty)	
RITM0010063	Not Yet Requested	1	(empty)	Apple iPad 3	06/05/2022 04:01:01 2d from now	$600.00	(empty)	

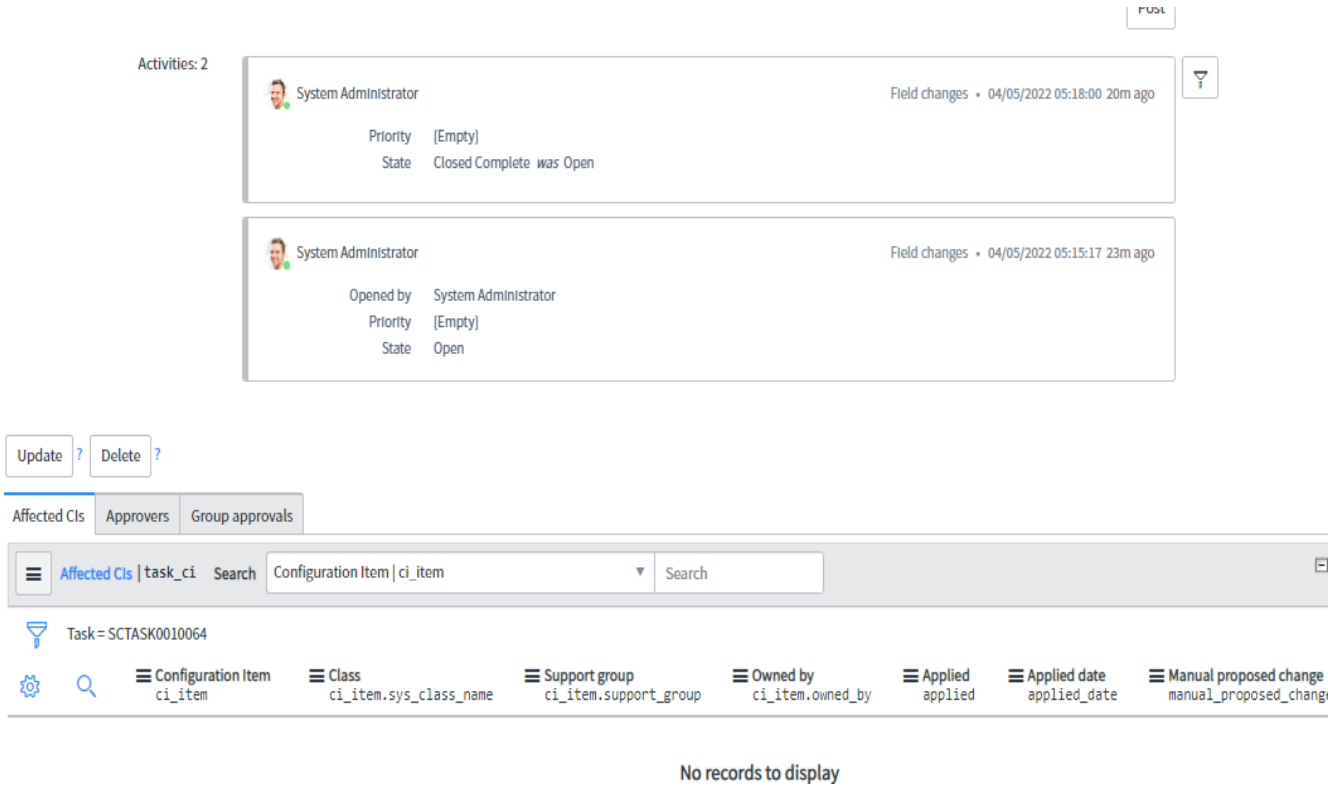
Actions on selected rows... | x 1 to 3 of 3

The form will be visible to service Desk like above screenshot and for every item there will be create different task.

For request number **REQ0010041,** Here generated three Request item and for each request item there will be generated different task.

Firstly, there will be need to approve it. It should be approved either any member of specified group or by Eric Schroeder. After approval, this request will change in task and there will be individual task for every item.





After generating the task, the task team can see the task form and they will appear like above screenshot to task team and he will have to work on this.

At the last after delivered the order the workflow will sow the end point.

All above is the step-by-step process from order the items to delivered the item.

CHAPTER 9

ADVANTAGE

- **It saves more time:**

It saves time by ordering the bundle of items. It avoids for ordering the same bundle again and again and order the items in simple three click.

- **Increase Production**

The primary benefit of automated workflow is that it saves time. The term automation itself refers to mechanization and computerization, which in turn is the process related to saving time to complete a job with accuracy by improving the speed. Saving time for completing a job is directly related to saving the employee time so that she can get involved in other productive tasks rather than doing a job with mechanical repetition of steps.

- **Save costs**.

When an employee gets involved in more tasks that generate the most revenue for your business in the same working hours, it leads to more productivity. More productivity means saving more money for the organization as time and money are always interrelated.

- **User can trace their items:**

User can trace their items and its state. How many items has been ordered and how many time will take to reached item etc.

- **Process automates:**

All Process will be automated by work flow. Here after generating Request item task will be auto generate and notification will be auto fire.

CHAPTER 10

CONCLUSION

A large number of items can be ordered in single order and it will save time of users. The major use of this application will be in big organizations, where they hire employee in bulks and they have to order ‘Welcome Kit’ they can order in single order and they will avoid again and again step to order same set of items. All process will be automated therefore it will save cost and time of users.

REFERENCES

1- https://link.springer.com/book/10.1007/978-1-4842-2571-4

2- https://link.springer.com/chapter/10.1007/978-1-4842-2571-4_12

3- http://wiki.servicenow.com/index.php?title=Differences_Among_Scripts

4-https://store.servicenow.com

5- https://link.springer.com/chapter/10.1007/978-1-4842-2571-4_2

6- https://link.springer.com/chapter/10.1007/978-1-4842-2571-4_10

7- https://link.springer.com/chapter/10.1007/978-1-4842-2571-4_4

8- https://link.springer.com/chapter/10.1007/978-1-4842-2571-4_8

9- https://link.springer.com/chapter/10.1007/978-1-4842-2571-4_1

10- https://link.springer.com/chapter/10.1007/978-1-4842-2571-4_9

11- https://link.springer.com/chapter/10.1007/978-1-4842-2571-4_11

12- https://link.springer.com/chapter/10.1007/978-1-4842-2571-4_7

13- https://link.springer.com/chapter/10.1007/978-1-4842-2571-4_6

14- https://link.springer.com/chapter/10.1007/978-1-4842-2571-4_14

15- https://link.springer.com/chapter/10.1007/978-1-4842-2571-4_13