

# **INTERNSHIP REPORT**

FEB 2021 – JUNE 2021

Internship report submitted in partial fulfilment of the requirement  
for the degree of Bachelor of Technology In  
**COMPUTER SCIENCE ENGINEERING**

By:

Priyanshu Shukla (171284)

To



Department of Computer Science & Engineering and Information  
Technology  
**Jaypee University of Information Technology Wagnaghat, Solan-  
173234, Himachal Pradesh**

# **DECLARATION**

I hereby declare that this submission is my own work carried out at Paymentus Corporation, Mohali from Feb, 2021 to June, 2021 and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma from a university or other institute of higher learning, except where due acknowledgment has been made in the text.

Name: Priyanshu Shukla

Date: 22-05-2021

# ACKNOWLEDGEMENT

I take this opportunity to express my sincere thanks and deep gratitude to all those people who extended their wholehearted cooperation and have helped me in completing this internship successfully.

First of all, I would like to thank Mr. Gaurav, who mentored me, guided me and helped me put my skills to real world use.

I would also like to thank my family and friends who greatly supported me during the course of the Internship.

Last but not the least, I would like to thank the founders for considering me a part of the organization and providing such a great platform to learn and enhance my skills.

Priyanshu Shukla

171284

Jaypee University of Information Technology

# TABLE OF CONTENTS

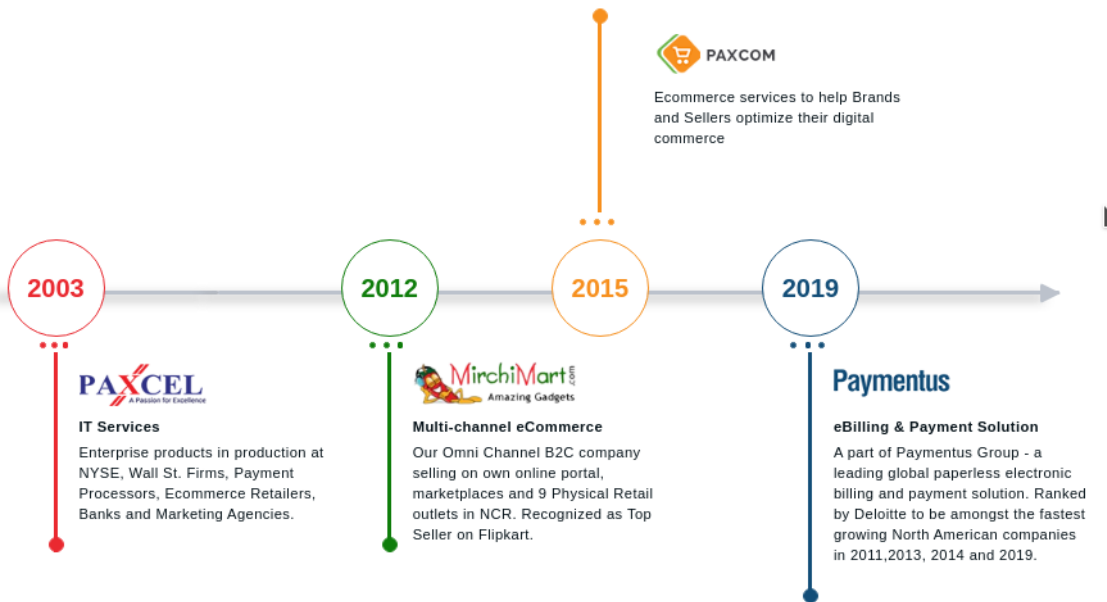
S. no.	Title	Page
1.	Introduction	1
2.	Description of the Industry	3
3.	Tools and Technologies Used	6
4.	Results & Conclusion	38
5.	References	39

# **Chapter-1**

## **Introduction**

### **Paxcom:**

We are a team of 200+ Ecommerce enthusiasts, passionate about using technology to simplify Digital Commerce and Payments for brands across the globe. Paxcom is a part of the Paymentus group - a leading global paperless electronic billing and payment solution provider.



## Paymentus:

Paymentus is a North Carolina based software company providing complete billing solutions.

I worked at Paymentus India Pvt Ltd at Mohali branch. I worked with the Hybrid-Billers Team and was successfully able to understand their working structure and pattern. I also learned soft skills like communicating within a corporate firm and working with a team. I was successfully able to understand various coding paradigms used by a company to build its

application. I got a thorough understanding of some of the company's existing products and some of the upcoming products.

The working culture of the company is great. I thoroughly enjoyed myself working there. Paymentus has a typical blend of work and fun. Although I didn't get to spend much time in the company office and started working from home after the coronavirus pandemic lockdown, I really enjoyed the weekend football and various parties at the company. And even after the lockdown the communication between me and my team was good through various meeting platforms like skype and google meets.

## **Chapter-2**

### **DESCRIPTION OF THE INDUSTRY**

## **Paymentus**

In 2004, Paymentus was brought into the world from a longing to improve the manner in which bills get paid. Vision, advancement and praiseworthy

help have moved Paymentus to turn into the main paperless electronic charging and installment arrangement available, bringing about 1,300 customers remembering the absolute biggest billers for North America.

We realize that to keep our answers current and significant, we need individuals with the expertise, drive and proclivity for encouraging a remarkably cheerful client experience. Our exceptionally dedicated, inventive representatives transformed a thought into a safe, SAAS-based Customer Engagement and Payment Platform; one that empowers direct-charge associations to give a brought together client experience and lift reception of cost-saving electronic charging and installment administrations.

Perceived by Deloitte to be among the quickest developing North American organizations in 2011, 2013, 2014, and 2016, Paymentus reliably endeavors to foster better, quicker, safer, cost-proficient charging and installment stages. We persistently look for higher incentive for our clients, in the two arrangements and administration.

It's what has prompted our exceptional development somewhat recently. We succeed when our customers succeed. They succeed when their client connections are upgraded and, thus, their clients take an interest in these



expense saving electronic administrations at high rates.

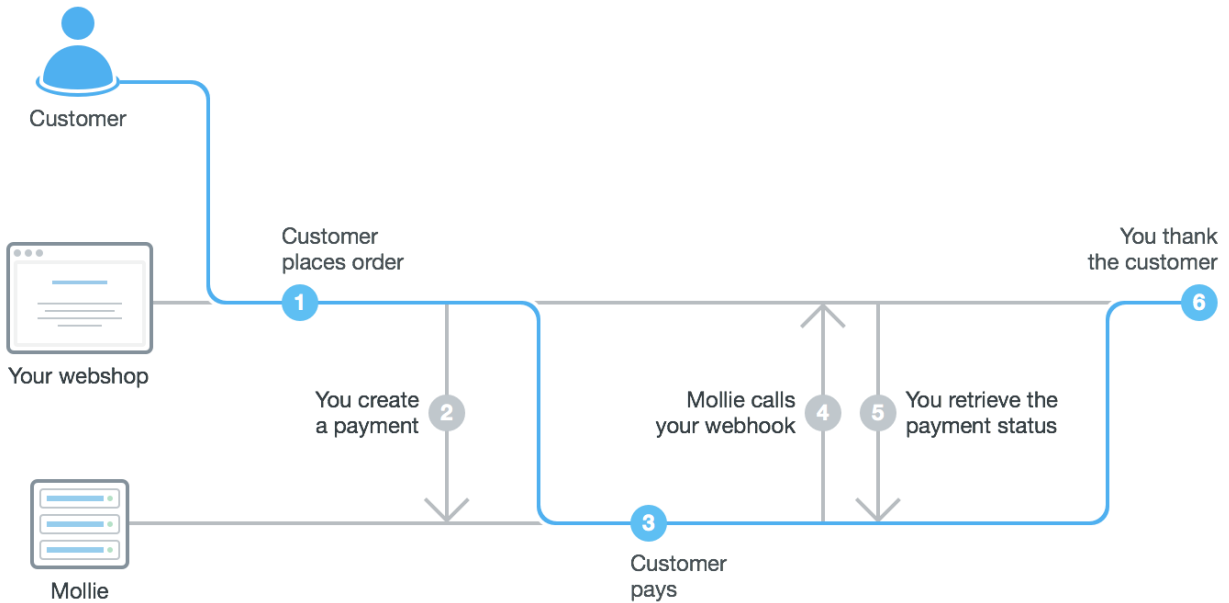
### **How Paymentus Works:**

#### **How to Make The Payment:**

Enter the URL of the association you need to pay into your internet browser and explore to the charging segment to start the installment interaction.

#### **Unsure of the URL?:**

The URL is often found on the billing statement one receives. It can be used to make utilities' payment.



## Chapter – 3

### Tools and Technologies used

My Internship at Paymentus involved learning skills in languages and tools like Angular, HTML, CSS, Typescript, GitHub, NodeJS and Javascript. Some of the technologies we also used in our day to day work are Kafka, Jira, Docker, Kubernetes, MongoDB etc. Apart from that this internship has also taught me soft skills like working culture in a corporate firm, communication within a team. I can call it my most improved skill by joining daily meetings to discuss new strategies to improve and pipeline to implement them. Although these meetings were very short time wise (ie. 5-15 mins) but informatively, they were sufficient enough to get us hooked for the day. We were assigned seniors from the company that managed our work and guided

us in right directions. They gave us feedback on every turn and helped us in very best way possible. The one thing we would like to add in this report about the seniors that managed us is that they not only taught us how to work efficiently and be more productive to the company but also how to manage work-life balance, how to overcome frustration of working in these covid times. Next I will be explaining in detail the technologies that we used.

## **JavaScript**

- It is a lightweight, interpreted programming language. It is expected for making network-oriented applications and is free to facilitate with Java. It is particularly easy to do because it is fused with HTML.
- It is the most notable programming language nowadays and that makes it a designer's unprecedented choice. At the point when you learn this, it helps you making unimaginable front-end similarly as back-end programming projects using unmistakable Javascript based frameworks like Node.js etc.
- Fabulous thing about Javascript is that we can discover enormous stacks of systems and Libraries as of late made which can be utilized plainly in your thing progress to lessen our chance to advance.



### **When should we avoid JavaScript?**

Regardless of the way that JavaScript is generally carried out and normalized language among internet browsers you ought to consistently verify whether what you are attempting to achieve should be possible in HTML, CSS, or (now and again) a worker side language first. This is on the

grounds that few more established or low fueled gadgets, for example, cell phones experience issues dealing with JavaScript. Likewise PCs with higher security needs frequently debilitate JavaScript because of possible blemishes. JavaScript is an exceptionally integral asset, yet make sure to utilize it sparingly and search for elective arrangements.

## **NODEJS**

After more than 20 years of stateless-online on the stateless mentioning reaction point of view, we at last have web applications with steady, two-way affiliations.

In one sentence: Node.js shines consistently web applications using push development over websockets. To be sure, after over twenty years of stateless online over the stateless sales response perspective, we now have web applications which are consistent, two-way affiliations, where both the client and specialist can begin correspondence, allowing them to exchange data wholeheartedly. This is as a glaring contrast to the ordinary web response perspective, where the client reliably begins correspondence. Moreover, it's completely established on the open web stack like HTML, CSS etc.

One may battle that we've had this for a significant long time as Flash and Java Applets—yet in reality, those were basically sandboxed conditions using the web as a vehicle show to be passed on to the client. They were run in detachment and every now and again worked over non-standard ports, which may requires extra approvals and such.

With the whole of its advantages, Node.js at present accepts a fundamental part in the development heap of some high-profile associations which can depend upon its novel benefits.

Basically Node.js is a laborer fit for executing JavaScript. At its middle, Node.js is a laborer engine that you can adjust and change, and it will just work after you set it up. It offers unconventional and event driven APIs so the requesting to it are dealt with as a circle (event circle), and that is the explanation Node.js is fundamentally a runtime. Being a piece of the JavaScript climate, which is great for application improvement, you can deal with it effectively nearby different JS gadgets, UIs, and connectors. All things considered, it is an open source and cross-stage system for building web applications two or three lines of code.

The discussion application is the most ordinary continuous application where

Node.js shows what it can do similar to managing various customers, genuine data, gigantic traffic, and coincidentally finding contraptions. Besides, it's inconceivable to learn Node while making a discussion application, as it covers basically all the programming of a regular Node.js application.

Because of HTML, laborer side web applications are not a customary use case for Node.js. Regardless, on the off chance that you join Node.js and Express.js, you can make praiseworthy web applications on the laborer side.

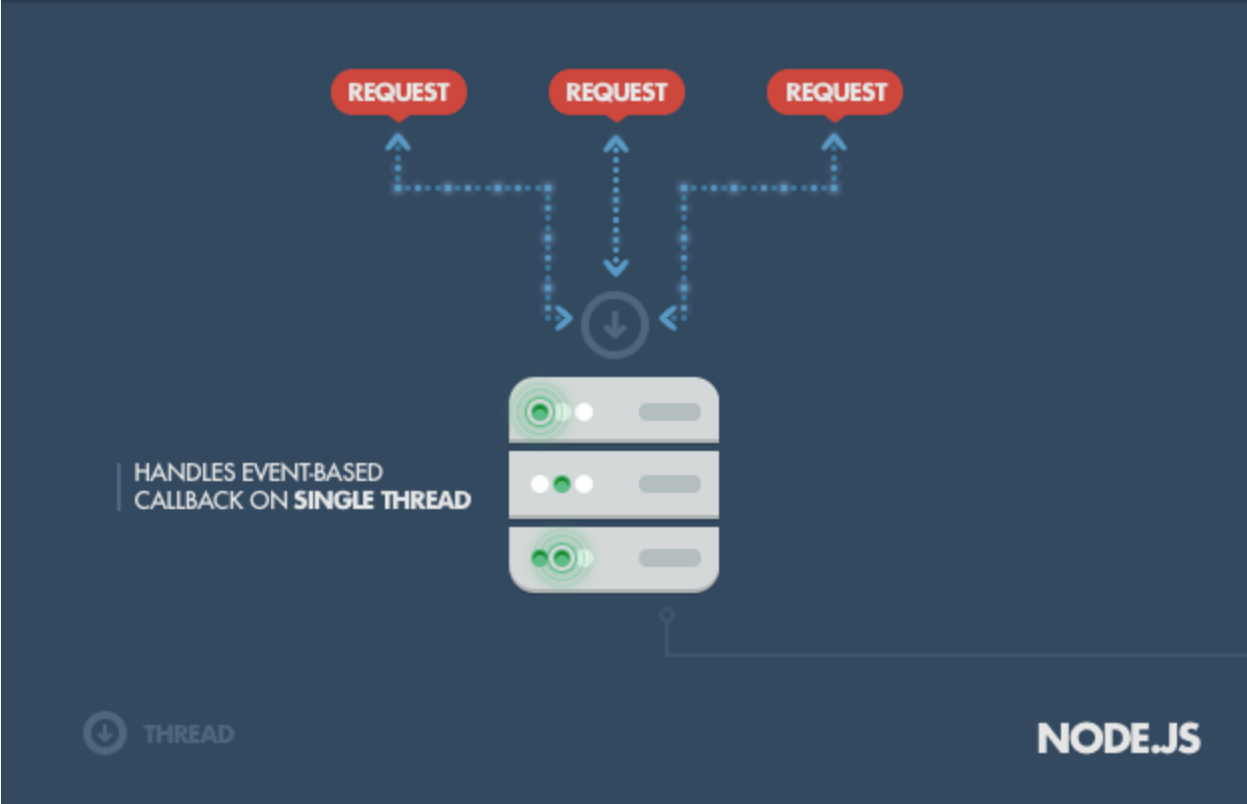
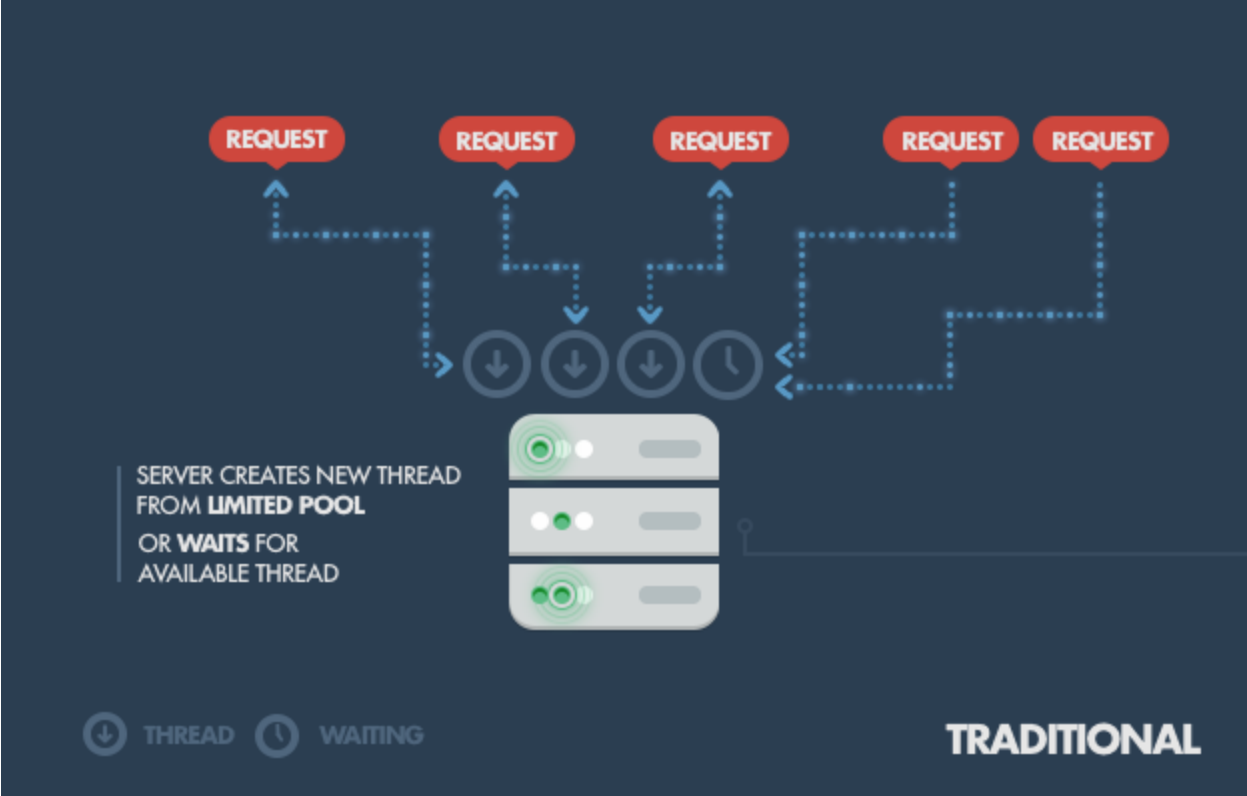
Node.js has some definite highlights in data streaming utilizing the way that HTTP sales and responses are on a very basic level data streams. For example, getting ready data while it's essentially being moved, for instance for sound/video encoding.

Another use occurrence of Node.js is checking dashboards to collect on-going information about site visitors and portrayal. Customer estimations and the ability to see what they are doing promptly, is unquestionably an unbelievable extra for associations.

Time will determine if Node is that next huge thing.

Node.js was never made for addressing the register scaling issue. The main aim of this was to deal with the Input and Output scaling issue, which it does unimaginably well. Being single-hung, Node.js might be a horrendous choice for web laborers filling in as computational specialists, since significant estimation will impede the laborer's responsiveness. In case your use case doesn't contain CPU raised activities or get to any thwarting resources, you can mishandle the potential gains of this and make quick and flexible system apps.





## **MongoDB**

MongoDB is a report situated information base. This implies that it doesn't utilize tables and columns to store its information, yet rather assortments of JSON-like archives. These reports support installed fields, so related information can be put away inside them.

MongoDB is likewise an outline less information base, so we don't have to determine the number or sort of segments prior to embeddings our information.

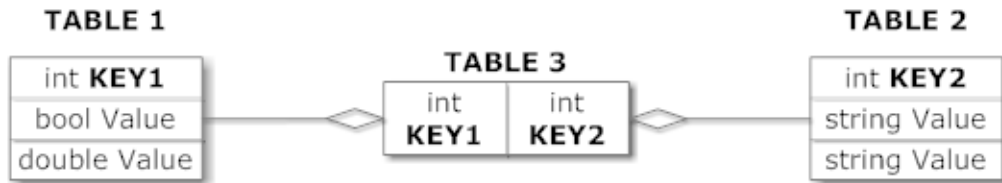
An example of a MongoDB database structure:

---

```
{
  _id: ObjectId(3da252d3902a),
  type: "Test",
  title: "MongoDB Example",
  author: "Author Name",
  tags: [ "mongodb", "compass", "crud" ],
  categories: [
    {
      name: "javascript",
      description: "Client-side and server-side JavaScript programming"
    },
    {
      name: "databases",
      description: "Different kinds of databases and their management"
    }
  ],
  content: "MongoDB is a cross-platform, open-source, NoSQL database..."
}
```

---

# Relational Model



# Document Model



## Angular

One of the middle conflicts to pick rakish over other single page application frameworks is its doubtlessly described strategy for how things ought to be done. It has an appraisal, in a way of speaking. That portrayed way enables the creation of a uniform code base over a relationship, without obsessing about describing certain standards.

Rakish as of now goes with these standards out of the container. Following them doesn't simply construct the consistency and consequently the idea of the code. It makes your application more self-evident, too. That is, if you are

as of now familiar with precise. This demanding system moreover ends up being valuable across joint effort borders. It engages new designers to consolidate into another gathering quickly, because of the great shared characteristic with the code.

What I need to state is, you ought to adhere to the rakish construction rules to profit by the precise framework. It will make your life fundamentally less complex, when coming into new endeavors and will extend the idea of your code thusly.

An average action, when learning rakish is to placed everything into the application module. Clearly, that your application will progress into an absolute mayhem. Modules are there which is as it ought to be!

Modules help to figure out your code into tinier gatherings to make finding things more straightforward. However, they are not simply something remedial. With the help of torpid stacked modules, you can in like manner assemble the customer experience simply by downloading the bits of the application, that are needed by then.

Precise is written in typescript by Google improvement group and is kept up by them routinely. It's first introductory delivery was in start of 2016. Absolute first form was called as Angular-JS so in future updates to cause it to recognize from other they named it to simply Angular and eliminated the JS. 11 stable adaptations has been delivered by the google group alongside

the local area support as of date 11 Nov 2020. To do material plan in angular, it gives precise material library.

## **GitHub**

Git is used for managing the movements to an assignment after some time. An errand might be just a lone record, a lot of archives, or an enormous number of reports. Those records can be anything from plain substance to pictures or accounts.

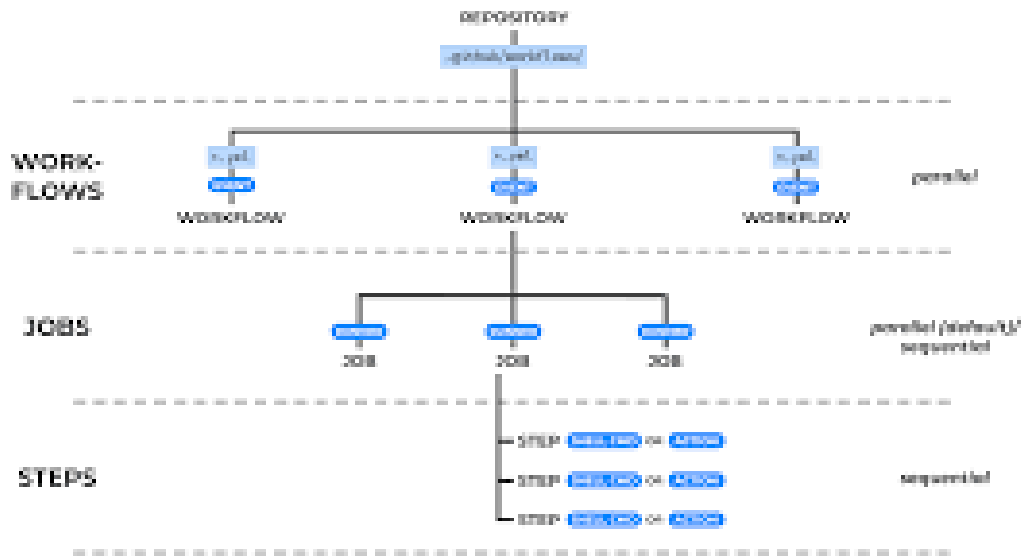
Since Git is based on supervising changes, it is routinely used as a joint exertion instrument allowing people to go after a comparable endeavor all the while. By following their individual changes, Git can join everything to the last structure. Imagine that you're creating a blog section that has various records related with it. You may have one rule content record which is the genuine post, an additional report for references, similarly as some various records that are diagrams and various pictures.

Without Git, you may have these various records taken care of in a coordinator on your PC, anyway it is very improbable to tell where the total of the reports are at a given point on schedule. Imagine you send your post to

two allies to copy change. How might you combine their movements back? Which record is the main, which is changed?

With Git you can follow the whole repo at various concentrations in time using submits while moreover giving remarks on why you decided to save the endeavor by then, at that point. Thereafter, you can per utilize this set of experiences of takes steps to see an away from of your endeavor, and besides travel back in that set of experiences, if significant. A product codebase works basically like that blog section. At its most fundamental level, it is an arrangement of records that are associated with one another.

Right when an architect is going after a particular component, Git gives a way to deal with spare a portrayal of the entire repo through a submit. This is for the most part done when slow headway has been made and a component is without bug. In making the present, the creator can give remarks explaining what was changed and why it was changed. This message adds to the verifiable scenery of the broaden and can simplify it to choose when a particular component – or even a bug – was introduced.



## Typescript

The definition from the authority site says: "a made superset out of JavaScript" yet it acknowledge you grasp what a "superset" is and what "created" implies. Maybe to keep things direct you can consider TypeScript of "a layer on top" of JavaScript.

TypeScript is a layer since you can create TypeScript code in your editor.



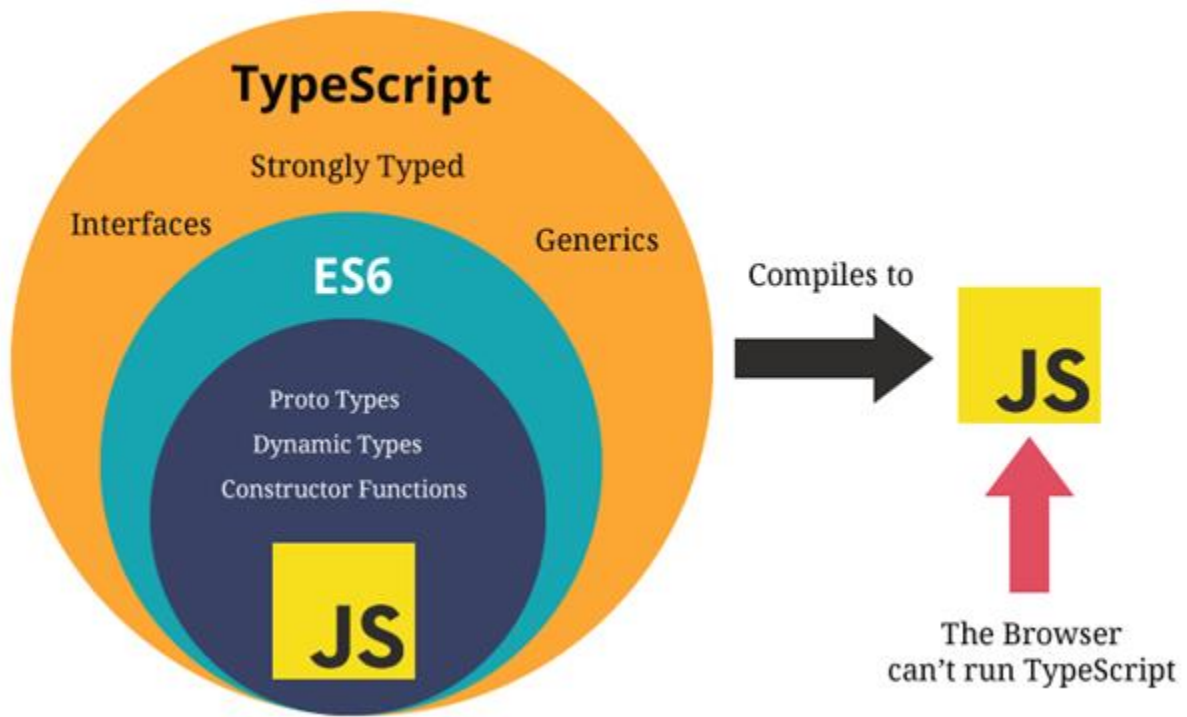
After a social event all that TypeScript stuff is gone and you're left with plain, clear JavaScript.

In case the chance of a gathering step bewilders you recollect that

JavaScript is as of now amassed and thereafter interpreted. There is a JavaScript engine that

scrutinizes and executes your code.

Nevertheless, JavaScript engines can't scrutinize TypeScript code so any TypeScript record should go under a "pre-understanding" measure, called total. Just after the important collection step you're left with unadulterated JavaScript code, arranged to run in a program. TypeScript is an outstanding kind of JavaScript yet it needs an "translator" prior to running in a Program.



## **Angular Design of The Material**

The Material Design is a visual language which can be utilized to make modernized encounters. It's a huge load of standards and rules across stages and contraptions for nature, advancement and segments that smooth out the course of action work measure for social occasions

arranging things.

The Material parts award you to make competent UIs with eminent assessed quality, theming and customization highlights.

Jaunty Material is the execution of Material Design standards and rules for Angular. It contains particular UI segments, for example,

- control structures (input, date picke)
- route designs (menu, toolbar)
- format segments (matrices, records )
- catches
- pointers which includes progress bars and spinners
- popups as well as modals

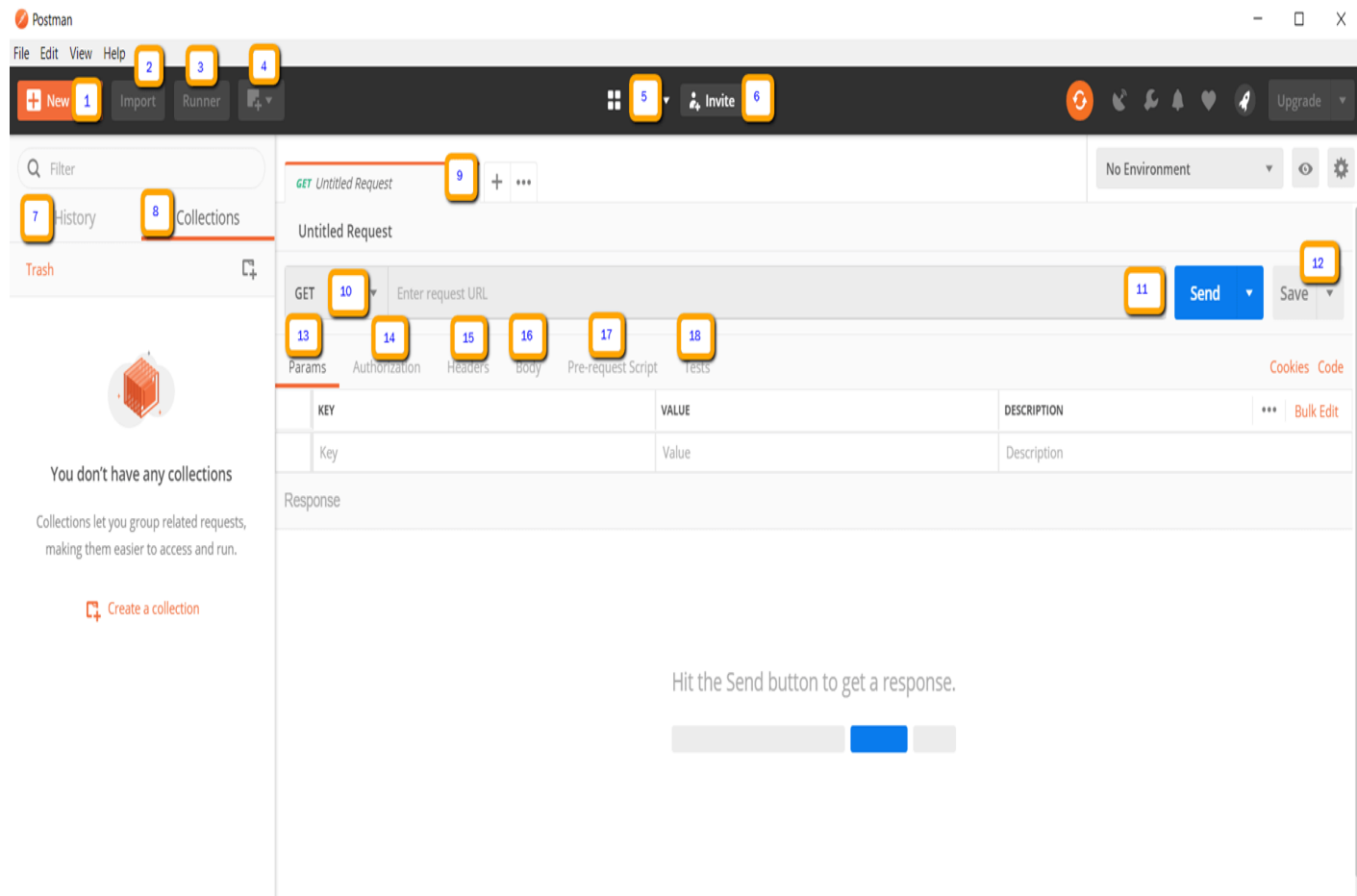
## **POSTMAN**

It is a flexible API testing instrument that quickly fuses in the CI/CD pipeline. It started in 2012 as a side errand by the founder Abhinav Asthana to enhance API work measure in testing and progression. Programming interface addresses Application Programming Interface which licenses programming apps to talk with each other through API calls.

Here are some significant reasons on why we should utilize Postman over it's contests.

With more than 4 million clients these days, this Software is becoming a tool of choice for the obvious reasons.

## The most effective method to utilize Postman to execute APIs



The following is the Postman Workspace.

1. New - The New feature is used to create a new environment where we will be using all of Postman's features.
2. Import - This is used to import an already existing environment.

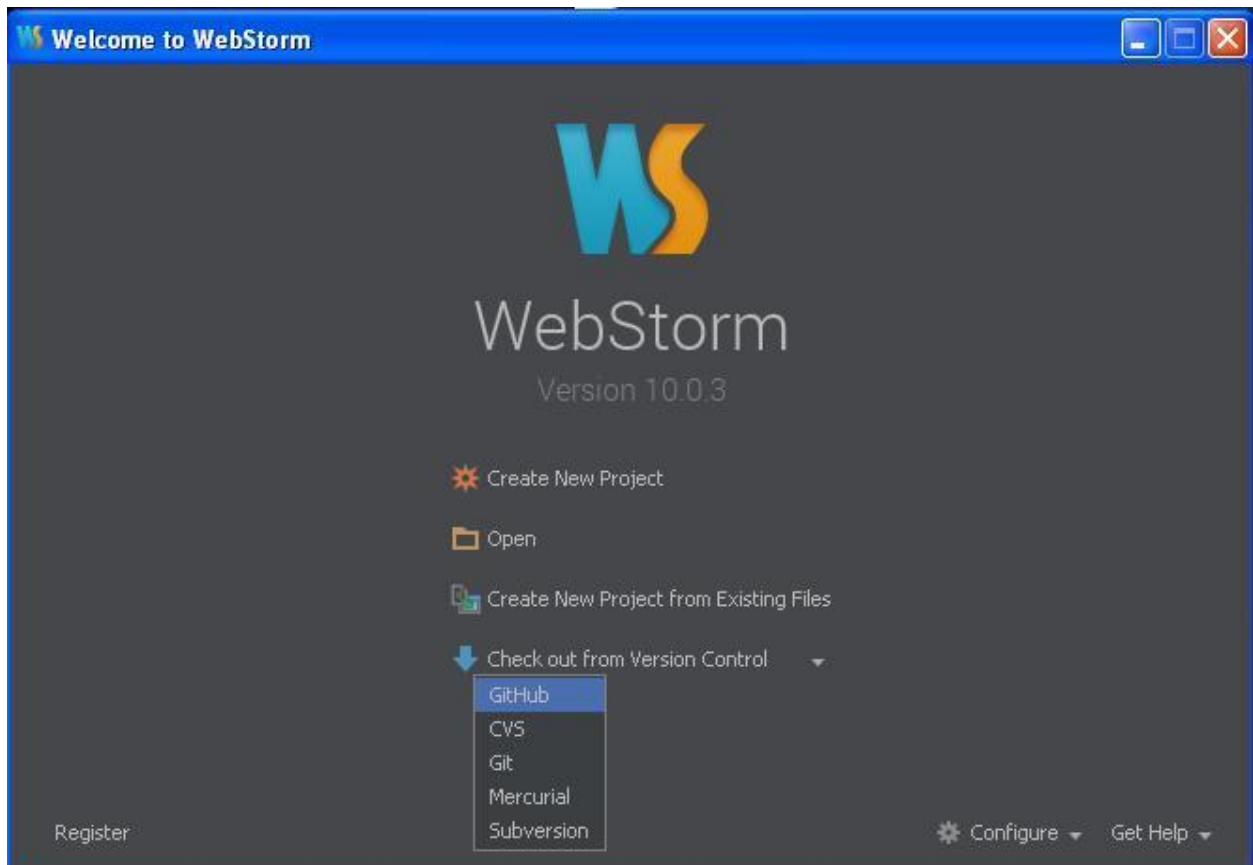
3. Runner - This feature is used to automate functions such as sending requests to a particular API.
4. Open New - This feature is used to open a new tab for sending another request
5. My Workspace - To check the workspace we are working in.
6. Welcome - Collaborate in a workspace along with your colleagues or friends.
7. History - This contains a history of the past requests we have made.
8. Groupings - This is basically to group together certain requests which we might have made and which need to be together.
9. Solicitation tab - Shows the title of the request we are currently making. "Untitled Request" is shown for unnamed requests.
10. HTTP Request - This feature is used to send all the CRUD (create, read, update, delete) requests like GET and POST, etc In Postman API testing.
11. URL - Basically the URL of the API where we will be making the request at.
12. Save - This is used to save our progress, that is, whatever

progress we have made thus far.

13. Params - The parameters we will be passing to the request.
14. Endorsement - Certain type of tokens or auth codes which might be required to make any request.
15. Headers - You can set headers, for instance, content sort JSON depending upon the necessities of the affiliation.
16. Body - The body is where the body (in JSON usually) is sent to the specified URL.
17. Pre-request Script - Here we specify the scripts are needed to be run before sending the request.
18. Tests - These are scripts executed when the request is made. It is useful when we need to check if our request is actually returning the response we need.



## JETBRAINS WEBSTORM



This is a joined improvement environment for coding in JSP and is associated with headways, including NodeJS, React etc. This makes our headway experience really beguiling, robotizing routine work and helping you with dealing with complex tasks easily.

It gives splendid code getting, autocompletion, refactoring features, on-the-fly slip-up evasion, and extensively more. Alongside help for the celebrated designs like AngularJS and Meteor and consolidated instruments for testing, exploring and code assessment and compromise with various VCS, WebStorm improves your convenience and takes your headway experience to an unfathomable level.

Firstly,JetBrains was set up in 2000 in Czech's capital of Prague by three software developers from Russia namely: S Dmitriev, V Kipyatkov and E Belyaev.

The association's first thing was IntelliJ Renamer, a tool created for coding in Java.

Come 2012, in the arise of having been the association's CEO for seemingly forever, Dmitriev supplied the association to two as of late appointed CEOs, O Stepanov and M Shafirov, and offered himself to his consistent endeavors in the field of BioInformatics.

In 2021 the New York Times asserted, in light of unidentified sources, that obscure gatherings may have installed malware in JetBrains' product that

prompted the Solar Winds hack and other inescapable security setles. JetBrains said they had not been reached by any administration or security organization, and that they had not "participated or been associated with this assault in any capacity".



```
9 <script>
10   import Welcome from "@components/Welcome";
11
12   export default {
13     components: {Welcome},
14     name: 'HelloWorld',
15   }
16 }
17
```

```
25 resolve: {
26   extensions: ['.js', '.vue', '.json'],
27   alias: {
28     'vue$': 'vue/dist/vue.esm.js',
29     '@': resolve('src'),
30   }
31 },
32 module: {
33   rules: [
```

## **JIRA**

### **What is JIRA?**

It's a tool made by the Company Atlassian which is based in Australia. The tool is used for issue and bug tracking, and add tasks to the KIBANA board. The key usage of this device is to follow issues and bugs related to your software or product and to produce production ready apps or software.

It is similarly used for project the chiefs. The JIRA dashboard basically eases the task of software development and breaks down everything into several stages. A segment of key features are recorded under.

This programming could be utilized for the accompanying motive:

Test cases as well as Requirement management

In the Agile Methods

Managing Project

Developing Software

Product as well as Task Management

## Tracking of Bugs and Defaults

Step by step process on how to use Jira software:

Step 1) Navigate to Jira using your browser or any specified software.

Step 2) Create a new project.

Step 3) Choose a new template or just start from scratch.

Step 4) Set up the columns for the software as needed.

Step 5) Create/modify any of the issues present.

Step 6) Start working along with your team members.

Jira allows anything and everything to be customized as per our needs:

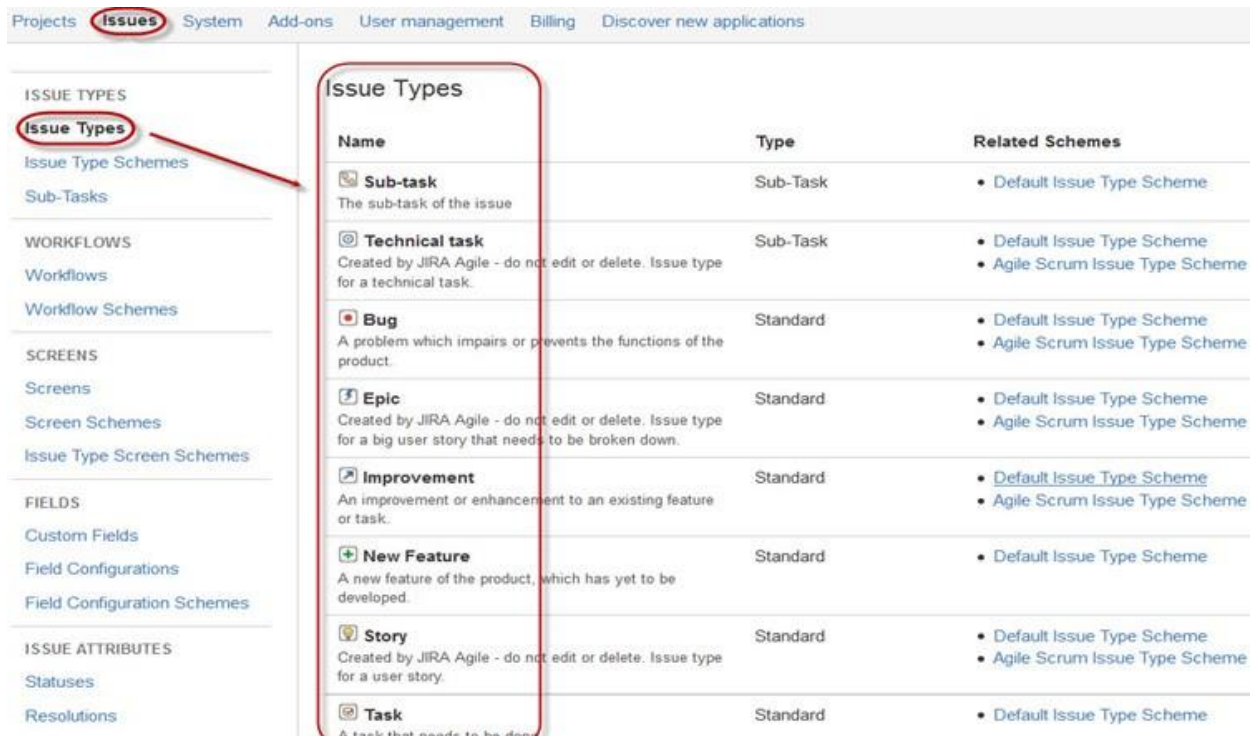
Workflows, Issue Types, Field Configuration, Notification, Permissions

Presently in this Jira Agile instructional exercise, let us see JIRA Issue exhaustively.

### **Jira Issues and their Types:**

It shows a wide list of things are needed to be worked upon or taken care of.

JIRA Issues are portrayed under various designs like new part, sub-undertaking, bug, etc as shown in the screen shot.



The screenshot shows the JIRA 'Issues' page. The left sidebar contains a navigation menu with categories: ISSUE TYPES (with 'Issue Types' selected), ISSUE TYPE SCHEMES, Sub-Tasks, WORKFLOWS (Workflows, Workflow Schemes), SCREENS (Screens, Screen Schemes, Issue Type Screen Schemes), FIELDS (Custom Fields, Field Configurations, Field Configuration Schemes), and ISSUE ATTRIBUTES (Statuses, Resolutions). The main content area is titled 'Issue Types' and contains a table with the following data:

Name	Type	Related Schemes
<b>Sub-task</b> The sub-task of the issue	Sub-Task	<ul style="list-style-type: none"><li>• <a href="#">Default Issue Type Scheme</a></li></ul>
<b>Technical task</b> Created by JIRA Agile - do not edit or delete. Issue type for a technical task.	Sub-Task	<ul style="list-style-type: none"><li>• <a href="#">Default Issue Type Scheme</a></li><li>• <a href="#">Agile Scrum Issue Type Scheme</a></li></ul>
<b>Bug</b> A problem which impairs or prevents the functions of the product.	Standard	<ul style="list-style-type: none"><li>• <a href="#">Default Issue Type Scheme</a></li><li>• <a href="#">Agile Scrum Issue Type Scheme</a></li></ul>
<b>Epic</b> Created by JIRA Agile - do not edit or delete. Issue type for a big user story that needs to be broken down.	Standard	<ul style="list-style-type: none"><li>• <a href="#">Default Issue Type Scheme</a></li><li>• <a href="#">Agile Scrum Issue Type Scheme</a></li></ul>
<b>Improvement</b> An improvement or enhancement to an existing feature or task.	Standard	<ul style="list-style-type: none"><li>• <a href="#">Default Issue Type Scheme</a></li><li>• <a href="#">Agile Scrum Issue Type Scheme</a></li></ul>
<b>New Feature</b> A new feature of the product, which has yet to be developed.	Standard	<ul style="list-style-type: none"><li>• <a href="#">Default Issue Type Scheme</a></li></ul>
<b>Story</b> Created by JIRA Agile - do not edit or delete. Issue type for a user story.	Standard	<ul style="list-style-type: none"><li>• <a href="#">Default Issue Type Scheme</a></li><li>• <a href="#">Agile Scrum Issue Type Scheme</a></li></ul>
<b>Task</b> A task that needs to be done	Standard	<ul style="list-style-type: none"><li>• <a href="#">Default Issue Type Scheme</a></li></ul>

The issue types present in Jira are as follows:

- The Default Issue Type Scheme: This is the default issue type scheme where the newly created issues are added to by default.
- Coordinated Scrum Issue Type Scheme: Issues and modifications related to agile scrum will be added to the coordinated scrum issue type scheme.

ISSUE TYPES

Issue Types

**Issue Type Schemes**

Sub-Tasks

WORKFLOWS

Workflows

Workflow Schemes

SCREENS

Screens

Screen Schemes

Issue Type Screen Schemes

FIELDS

Custom Fields














Field Configurations

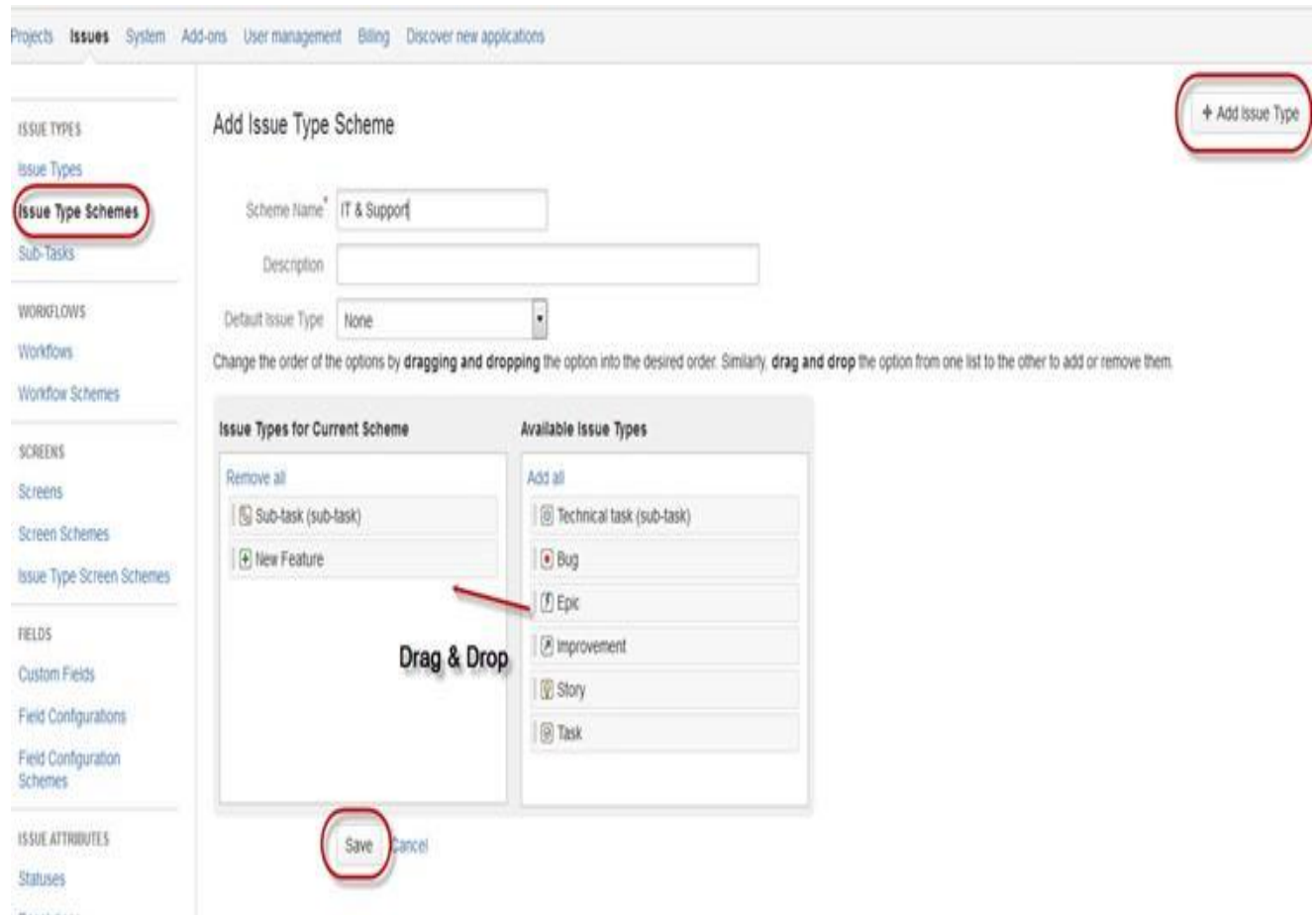
Field Configuration Schemes

ISSUE ATTRIBUTES

## Issue Type Schemes

 An issue type scheme determines which issue types will be available to a set of projects. It also allows to specify the user interface.

Name	Options	Projects
<b>Default Issue Type Scheme</b> Default issue type scheme is the list of global issue types. All newly created issue types will automatically be added to this scheme.	<ul style="list-style-type: none"><li> Bug (Default)</li><li> New Feature</li><li> Task</li><li> Improvement</li><li> Sub-task</li><li> Epic</li><li> Story</li><li> Technical task</li></ul>	Global (all unconfigured projects)
<b>Agile Scrum Issue Type Scheme</b> This issue type scheme is used by JIRA Agile's Scrum project template. Projects associated with the Scrum template will be associated to this scheme. You can modify this scheme.	<ul style="list-style-type: none"><li> Epic</li><li> Story (Default)</li><li> Technical task</li><li> Bug</li><li> Improvement</li></ul>	No projects



These Components are sub-spaces of an errand; they are used to pack issues inside an endeavor into more unobtrusive parts. Sections add a couple of developments to the exercises, isolating it into features, gatherings, modules, subprojects and that is only the start. Using fragments you can deliver reports, assemble estimations, and show it on dashboards, and so on.



## Components

Projects can be broken down into components, e.g. "Database", "User Interface". Issues can then be categorised against different components.

Name	Description	Component Lead	Default Assignee	
 <input type="text"/>	<input type="text"/>	<input type="text"/>	Project Default (Unassigned)	<input type="button" value="Add"/>
 SAP Testing	Bug detected while User Acceptance testing	 Krishna Rungta [Administrator]	Project Lead	<input type="button" value="Delete"/>

For adding new segments, as demonstrated in the above screen we can add names.

## **RESULTS AND CONCLUSION**

This internship was to be sure a pool of information, not just have I acquired information in Full Stack Development however I have likewise found out about how improvement of any venture happens, how group works, how crafted by every worker is followed, how work is disseminated between various partners, what are the various phases of advancement, what are the specialized issues that one countenances in the improvement of any undertaking, what everything is needed before the advancement of any task, what the code base ought to resemble and what standards should be continued in the turn of events. The card scanner can check different kinds of charge/Mastercards. It effectively examines non emblazoned cards yet sets aside some effort to check the embellished ones. This can be improved by additional upgrades in the examining calculation utilized.

## REFERENCES

- [Angular.io](#)
- [Github.com](#)
- [Paxcom.net](#)
- [Paymentus.com](#)
- [Wikipedia.com](#)
- [Material.angular.io](#)
- [Npmjs.com](#)
- [restfulapi.net](#)