

**TICKET MANAGEMENT SYSTEM**

Submitted in Partial fulfillment of Requirement for the award of the Degree of

**MASTER OF COMPUTER APPLICATIONS**

Dilip Kumar. M

1MS17MCA13

Under the Guidance of

Dr. Manish KumarMr. Indrajit Naiya

Assistant Professor Senior Software Developer

Department of Master of Computer Application KLoc Technologies Pvt Ltd

Ramaiah Institute of Technology

**Department of Master of Computer Applications**

**RAMAIAH INSTITUTE OF TECHNOLOGY**

(Autonomous Institute, Affiliated to VTU)

Accredited by National Board of Accreditation & NAAC with ‘A’ Grade MSR Nagar, MSRIT Post, Bangalore-560054 www.msrit.edu 2018



**CERTIFICATE**

This is to certify that the dissertation work entitled “Ticket Management System” is carried out by Dilip Kumar. M (1MS17MCA13), a bonafide student of Ramaiah Institute of Technology, Bangalore, in partial fulfillment for the award of Master of Computer Applications of the Visvesvaraya Technological University, Belgaum, during the year 2020. It is certified that all corrections / suggestions indicated for Internal Assessment have been incorporated in the project report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect to dissertation work prescribed for the said degree.

Guide HoD

(Dr. Manish Kumar) (Dr. Yogish H K)

Name & Signature of Examiners with Date:-

1)

2)



**DECLARATION**

I, Dilip Kumar. M, a student of Master of Computer Applications, Ramaiah Institute of technology, Bangalore hereby declare that the project entitles “Ticket Management System” has been carried out independently under the Guidance of Dr. Manish Kumar and Mr. Indrajit Naiya.

I hear by declare that work submitted in this project report is my own, except where acknowledge in the text and has not been previously submitted for the award of the degree of Visvesvaraya Technological University, Belgaum or any other institute or University.

Place: Bangalore Dilip Kumar. M

Date: 1MS17MCA13

**ACKNOWLEDGEMENT**

It gives me pleasure to present my project “Ticket Management System” With deep sense of gratitude, I am thankful to our beloved Principal. I would like to thank our most respected HoD, Dr. Yogish H K for his pillared support, encouragement and for giving me an opportunity to embark upon this topic and for his constant encouragement.

I would like to thank Dr. Manish Kumar, Assistant Professor, Department of Master of Computer Applications, for his help and support during the course of this project. I would also like to thank Mr. Indrajit Naiya, Senior Software Developer, KLoc Technologies Pvt Ltd, for ably guiding me during the course of the internship and mentoring me during the project.

**ABSTRACT**

This project is built on NodeJS and ReactJS, which is a smarter choice as this framework provides lots of free ready-made solutions to implement basic functionality for online stores. It operates on MVC architecture. Model describes the app logic, View describes how the part of the model may be shown to the user’s interface, while Controller is responsible for app management by processing the user’s action.

This project provides an easy way to raise an issue for the kloc’s applications. When a user land on an e-commerce site, they expect to find what they are looking quickly and easily. Kloc application users can find there application and have a concern about the kloc’s application they can raise an issue in this platform, which helps us to manage the issues and resolve them as soon as possible

**TABLE OF CONTENTS**

[1. Introduction 1](#_Toc42713586)

[1.1. Overview 1](#_Toc42713587)

[1.2. Feasibility Study 1](#_Toc42713588)

[1.3. Existing System 2](#_Toc42713589)

[1.4. Proposed System 2](#_Toc42713590)

[2. Literature Survey 4](#_Toc42713591)

[2.1. Existing System Study 4](#_Toc42713592)

[2.2. Tools And Technology Used 6](#_Toc42713593)

[3. Hardware And Software Requirements 10](#_Toc42713594)

[3.1. Hardware Requirements For Server 10](#_Toc42713595)

[3.2. Software Requirements 10](#_Toc42713596)

[4. Software Requirement Specification 12](#_Toc42713597)

[4.1. Introduction 12](#_Toc42713598)

[4.2. Overall Description 12](#_Toc42713599)

[4.3. External Interface Requirements 16](#_Toc42713600)

[4.4. System Features 17](#_Toc42713601)

[4.5. Other Nonfunctional Requirements 18](#_Toc42713602)

[5. System Design Description 20](#_Toc42713603)

[5.1. Introduction 20](#_Toc42713604)

[5.2. System Overview 21](#_Toc42713605)

[5.3. Data Dictionary 21](#_Toc42713606)

[5.4. Functional Design 22](#_Toc42713607)

[6. Implementation 26](#_Toc42713608)

[6.1. Introduction 26](#_Toc42713609)

[6.2. Pseudo Code 26](#_Toc42713610)

[7. Testing 29](#_Toc42713611)

[7.1. Testing 29](#_Toc42713612)

[7.2. Test Cases 29](#_Toc42713613)

[8. Conclusion 32](#_Toc42713614)

[9. Future Scope And Futher Enhancement 34](#_Toc42713615)

[10. Bibliography 36](#_Toc42713616)

[10.1. Website Reference 36](#_Toc42713617)

[11. User Manual 38](#_Toc42713618)

[11.1. Screenshot 38](#_Toc42713619)

# 

# Introduction

## Overview

Kloc is a software solutions provider with a keen focus on customer satisfaction. We partner with businesses to address all their IT requirements be it application development, integration, maintenance or quality initiatives. We do not offer cookie cutter solutions but tailor our offerings to adapt to our customer’s enterprise thereby ensuring high return on investment. As our slogan goes — we don’t just develop solutions, – we think IT through.

Our vision is to be a technology enterprise that empowers organizations and individuals to innovate, invent and inspire the world with their creative imagination and technical brilliance.

KLoc Technologies to be recognized among the most innovative enterprises in the IT business. We will offer our customers high quality & cost-effective services within geographies and market segments that can benefit from our diverse talent base and process-centric solution delivery.

## Feasibility Study

### Technical Feasibility

Tickets Management will be a used to raise any issues in kloc’s application, which provide various features to track the status of the raised issue’s. Various customer all over the globe use this website for raising kloc applications issues. It provides a convenient way for raising tickets.

### Economic Feasibility

This application is built on NodeJS and ReactJS. As the frameworks used for developing the application is open source the application can be easily developed in a cost-effective manner. The project is divided among many teams, the development team which is further divided into front-end and back-end.

## Existing System

Previously the website was really difficult to handle. It was a very complex website. For any changes it took a lot of effort. Moreover, error handling was also very difficult.

## Proposed System

The current system is developed using ReactJS, which is used to show the graph in usage of the applications. How many tickets are raised and how many tickers are solved. It provides more convenient way to manage errors. Debugging becomes easier and even new features can be added in a more convenient manner.

2. LITERATURE SURVEY

# Literature Survey

In the present situation kloc’s customers don’t have a platform in which allows them to raise an issue for the kloc’s different application. Customer’s have to call or send an email Kloc’s service for any issue. Here in this application the customer is allow to raise an issue for Kloc’s application. It allows customers to get solutions if the same issue is solved previously and can check FAQ’s for issues as well. It allows customers to see other solved issues and can be check the status of the raised issue.

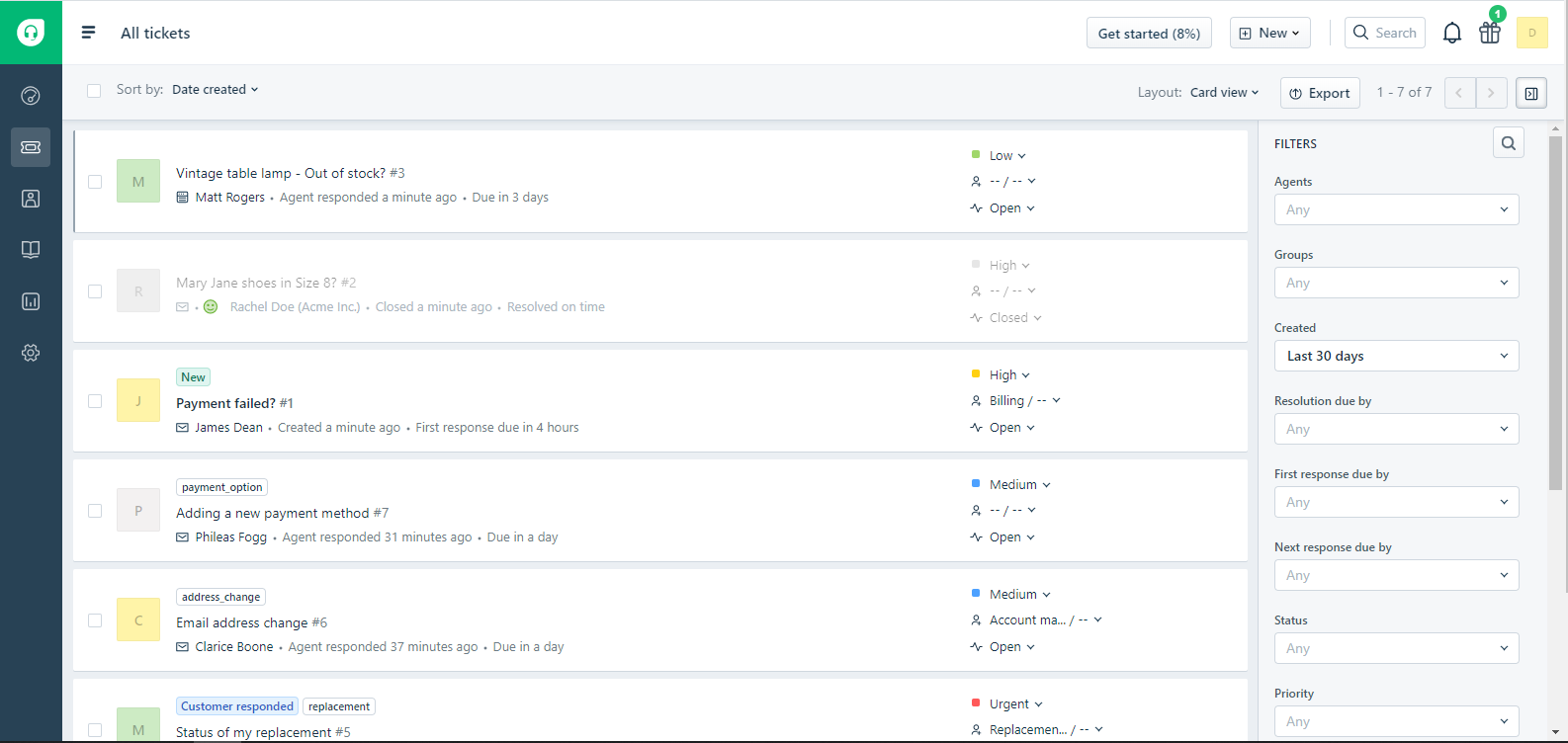
To understand the working of real life scenario, where user earlier had problem to search Kloc’s email, contact no and then need to speak about the issue. After the issue is complained customer’s have no idea of the status of the issue? This web app overcomes these problems by providing to raise an issue and

## Existing System Study

There are numerous existing systems which falls under the same category. We have looked into a number of such systems and have come up with the idea of our system. Apart from the basic features we have also included some of our own ideas and features to make the application much more user friendly and stable. The various systems or application which we have referred in the development of this project are shown below.

### Freshdesk.Com

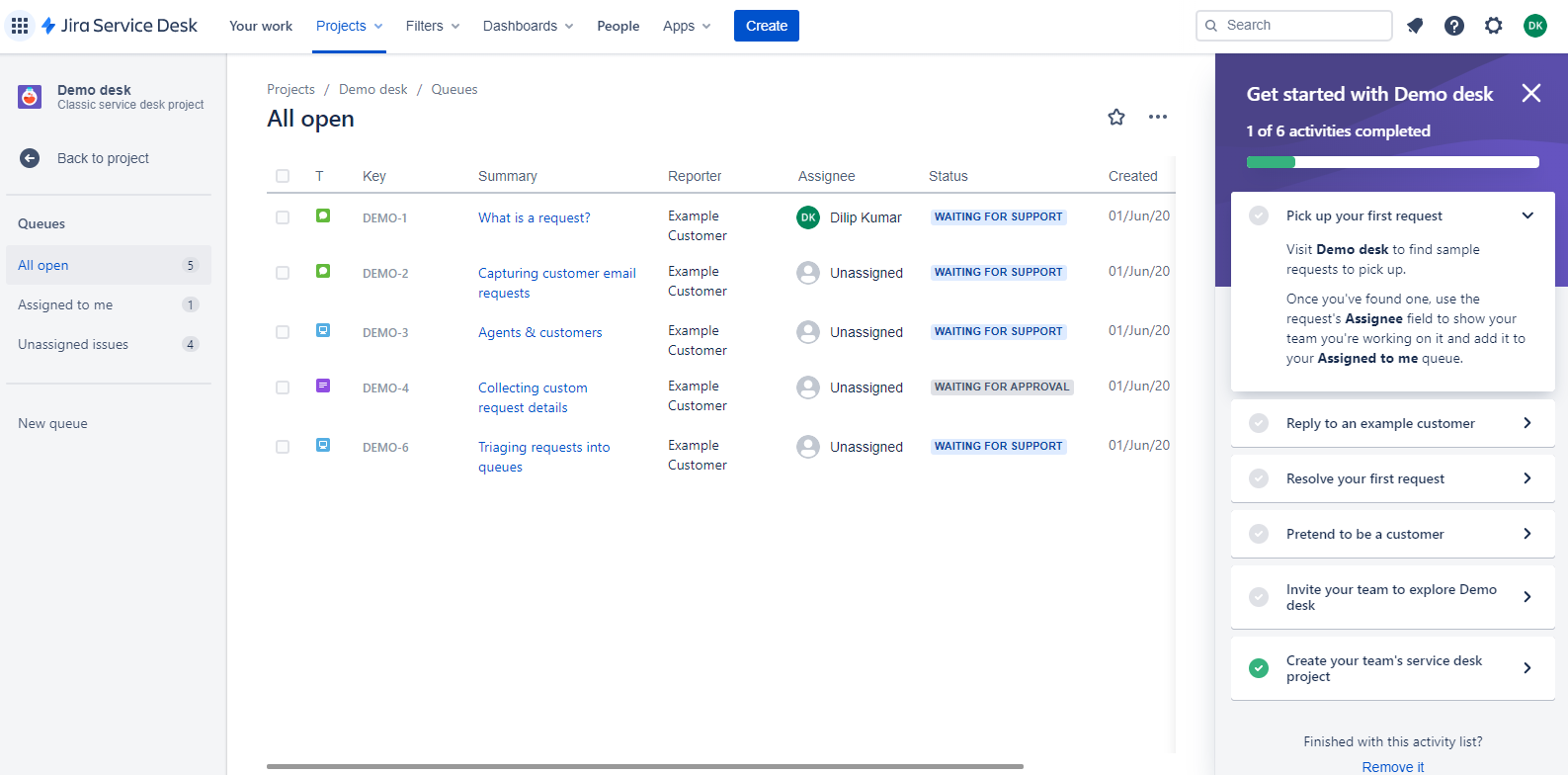
The first system which we referred was freshdesk.com. A screenshot of the freshdesk is shown in Fig 2.1 for issuing tickets. It is one of the most mainstream websites in India in this field. Startup from all over the country visit this website and provide feedback.



**Fig 2.1: freshdesk ticket issuing page**

### Atlassian.Com

Fig 2.2 shows Atlassian generating tickets based on the projects. This is also one of the best websites running for this field. Based on the company and projects registered under the projects can be viewed



**Fig 2.2: atlassian.com**

## Tools And Technology Used

### Reactjs

React (also known as React.js or ReactJS) is a JavaScript library for building user interfaces. It is maintained by Facebook and a community of individual developers and companies.

React can be used as a base in the development of single-page or mobile applications. However, React is only concerned with rendering data to the DOM, and so creating React applications usually requires the use of additional libraries for state management and routing. Redux and React Router are respective examples of such libraries.

React code is made of entities called components. Components can be rendered to a particular element in the DOM using the React DOM library. When rendering a component, one can pass in values that are known as "props

JSX, or JavaScript XML, is an extension to the JavaScript language syntax. Similar in appearance to HTML, JSX provides a way to structure component rendering using syntax familiar to many developers. React components are typically written using JSX, although they do not have to be (components may also be written in pure JavaScript). JSX is similar to another extension syntax created by Facebook for PHP called XHP.

### Nodejs

Node.js is an open-source, cross-platform, JavaScript runtime environment that executes JavaScript code outside of a web browser. Node.js lets developers use JavaScript to write command line tools and for server-side scripting—running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm, unifying web-application development around a single programming language, rather than different languages for server- and client-side scripts.

Though .js is the standard filename extension for JavaScript code, the name "Node.js" doesn't refer to a particular file in this context and is merely the name of the product. Node.js has an event-driven architecture capable of asynchronous I/O. These design choices aim to optimize throughput and scalability in web applications with many input/output operations, as well as for real-time Web applications (e.g., real-time communication programs and browser games).

The Node.js distributed development project was previously governed by the Node.js Foundation, and has now merged with the JS Foundation to form the OpenJS Foundation, which is facilitated by the Linux Foundation's Collaborative Projects program.

Node.js brings event-driven programming to web servers, enabling development of fast web servers in JavaScript. Developers can create scalable servers without using threading, by using a simplified model of event-driven programming that uses callbacks to signal the completion of a task. Node.js connects the ease of a scripting language (JavaScript) with the power of Unix network programming.

Node.js was built on the Google V8 JavaScript engine since it was open-sourced under the BSD license. It is proficient with internet fundamentals such as HTTP, DNS, TCP. JavaScript was also a well-known language, making Node.js accessible to the web development community.

### Mongodb

MongoDB is a document database, which means it stores data in JSON-like documents. We believe this is the most natural way to think about data, and is much more expressive and powerful than the traditional row/column model.

#### Rich Json Documents

* + The most natural and productive way to work with data.
  + Supports arrays and nested objects as values.
  + Allows for flexible and dynamic schemas.

#### Powerful Query Language

* + Rich and expressive query language that allows you to filter and sort by any field, no matter how nested it may be within a document.
  + Support for aggregations and other modern use-cases such as geo-based search, graph search, and text search.
  + Queries are themselves JSON, and thus easily compassable. No more concatenating strings to dynamically generate SQL queries.

#### All The Power Of A Relational Database, And More...

* + Full ACID transactions.
  + Support for joins in queries.
  + Two types of relationships instead of one: reference and embedded.

3. HARDWARE AND SOFTWARE REQUIREMENTS

# Hardware and Software Requirements

## Hardware Requirements for Server

Ticket Management system is a web-based application, it can run on any hardware platform capable of running modern browser. The following configuration is the bare minimum required to run any modern browser.

• Processor : Intel Core i3 higher version

• RAM : 4 GB

• Hard Disk : 1 GB of free space

• Output : Color monitor

• Input : Keyboard, Mouse

• Network card with broadband connection x

## Software Requirements

### For Developer

* Front-end: ReactJS
* Backend: NodeJS
* Database: MongoDB

### For End User

* Operating system: Ubuntu or macOS or Windows
* Browser: Latest versions of Google chrome or Mozilla Firefox

4. SOFTWARE AND SOFTWARE REQUIREMENTS

# Software Requirement Specification

## Introduction

### Purpose

* Ticket Management System is web application is intended to provide complete solutions for customers through a single gateway using the internet as the sole medium. Ticket Management system is a multichannel ticketing system collects all support tickets from different channels and organizes them in one tab. This helps agent’s reply to all of them from the same tab.
* This document is meant to discuss the features of Ticket Management system, so as to serve as a guide to the developers on one hand and a software validation document for the prospective client on the other.

### Intended Audience and Reading Suggestions

This project is a prototype for Ticket management system and it is restricted within the college premises. This has been implemented under the guidance of senior software engineer. This project is useful for the Ticket management system team as well as to the customers.

### Product Scope

This project helps Kloc’s customers to raise an issue for the kloc’s application. The main advantage of this application is to identify the issue and if the issues is caused by any other previous customers then an solution for the issue can be found. And if the issue is solved it can be found in the FAQ’s

## Overall Description

We are able to generate a ticket or raise an issue for the particular application where the customer or merchant have been facing issues with the application features. The issue will be managed until the issue get’s a resolution. And we can view the metrics based on the tickets count, recurring issues with the application, based on the resolved duration, based on the issues .We can raise a ticket on an application and also with the FAQ’s.

### Product Functions

#### Admin Features

1. We are creating the admin at the backend after signing up as a customer.
2. Admin can invite Moderators:

Admin can invite the moderators through their mail ids. An email sign up link will be sent to the moderators. Moderators should login to their respective mail ids. And they should be able to sign up through the same mail id with the email being pre-filled in the sign up form.

1. Admin can create/read/update/delete and can search an application:

Admin can create an application. An application can be created by selecting the moderators. If the moderators are linked, only those moderators will get the mails of the application creation. An application can be created by selecting no moderators and is linked to all the moderators. That tells us that the application is in charge of all the moderators.

1. Admin can search the application and raise a ticket:

Admin can search the list of applications and can select any application among them and will also be able to raise a ticket. Admin can raise a ticket with admin email being prefilled and the application name being pre-filled and attached to the ticket.

1. Admin can assign the tickets to the Moderators:

Admin can assign the tickets to any of the moderators, if the application is linked to all the moderators.

1. Admin can archive/delete the ticket:

Admin can archive/delete a ticket, on clicking the delete application link, the ticket will not be deleted but works in a similar fashion by archiving them.  
These tickets can be considered as deleted.

1. Admin can read/update his profile details and can update his password:

Admin can archive/delete a ticket, on clicking the delete application link, the ticket will not be deleted but works in a similar fashion by archiving them.  
These tickets can be considered as deleted.

1. Admin can create/read/update/delete comments and sub comments on the tickets:

Admin will have complete access to create a comment on the ticket and to view it. He will also have permission to update the comments.

1. Admin can create/read/update/delete faqs:

Admin can create and read the faqs. He can search through them. He will be having complete access to update and archive/delete the faqs.

1. Admin can raise a ticket with the faqs:

Admin can raise a ticket with faq attached.

1. Admin can read the resolved tickets:

Admin can read all the tickets whose status is resolved. And he’ll be able to search through them.

1. Admin can view the graphs:

Admin has the permissions to view the graphs / to obtain the metrics for the following scenarios:

* 1. An application has how many bugs?
  2. What are the problem areas?
  3. How long did it take to fix those?
  4. Which are the recurring issues?
  5. Overall ticket count.

#### Moderator Features (Module 1)

1. A Moderator will be created via an email invite from Admin.
2. Moderators can read their assigned tickets:

Moderators can read all the tickets that are assigned to him.

1. Moderators can read their resolved tickets:

Moderators can read all the tickets whose status are resolved and which belongs to him.

1. Moderators can read their unassigned tickets:

A Moderator can read all of their tickets raised through the applications in which he’s a moderator under unassigned tickets.

1. Moderators can assign tickets to themselves:

Moderators can assign tickets from the list of unassigned tickets. Once the tickets are assigned to a moderator that particular ticket will longer be available for any other moderators to assign.

#### Customer Feature

1. A customer will be created through a sign up.
2. Customers can search through the applications and raise a ticket.
3. Customers can view their tickets.
4. Customers can view their resolved tickets.

### User Classes Andf Characters

* The user who is using the product which company provides, if the users are facing any issue on the company product then they can use this ticket management system to raise a ticket.
* The user should be familiar with the Internet.

### Operating Environment

The operating environment for this application can be any device E.g. PC, Mobile, I Pad etc. The users of this application must have a web browser to open this web application. Users of this application must have internet connection enabled.

### Design And Implementation Constraints

Design of the interface is carried under Nodejs and ReactJS web framework which is an open source framework for developing web applications. It follows Model View Controller architecture pattern.

The database which is used for the application is MongoDB. It is a powerful, open source database for NoSQL. It has earned strong reputation for storing data in Json format.

* Users should use web browser to use this applications
* Ticket management system is a website that shall operate in all famous browsers, for a model we are taking Microsoft Internet Explorer versions 7.0, 8.0 and 9.0 .

### User Documentation

A well-documented user manual will be provided; the user manual contains all the screenshots of the application. These screenshots are well described in order to make the user understand the flow of working of the application.

### Assumption And Dependencies

* Shopify users, who is using our product, can raise the issue.

## External Interface Requirements

### Hardware Interfaces

* Any standard Operating system
* A browser which supports HTML, CSS & Javascript.

### Software Interfaces

**Tools/Packages Used in Ticket Management System:**

* Linux
* Database: MongoDB(nosql)
* Tools:
  + Gitlab – for version control
  + Ngrok – for public URL’s
  + Balsamiq – for wireframing
  + DBDesigner.net – for database modelling.
  + VSCode – for code editing.
  + Postman – for developing API’s
* Packages (npm):
  + JWT(json web token) for authentication and authorization
  + Express for web application framework
  + AWS s3 for file uploads
  + React-Paginate for pagination
  + Chrome debugger
  + Mongoose for Database Design and Modelling Schema
  + Bcrypt for encrypting the password

### Communications Interfaces

* FTP for file uploading
* SMTP for sending an email
* HTTPS to transfer the data over the web

## System Features

### Description And Priority

It maintains the information of all Kloc’s applications. It also sorts the issues according to the priority, fetching data from the cookies it displays the recently searched FAQ’s and similar questions

### Stimulus/Response Sequence

It displays the issue’s and allows the customers to raise an issue moreover user can close the issue is the customer can solve is themselves.

## Other Nonfunctional Requirements

### Performance Requirements

This application must be fit according to the performance wise. It should use less memory and will be easily accessible to the user. Memory Management should be done wisely so that none of the memory part is wasted. The most important factor in the working of the whole application is its connectivity with the server and the mode of connection.

### Safety Requirements

If there is extensive damage to a wide portion of the database due to catastrophic failure, such as a disk crash, the recovery method restores a past copy of the database that was backed up to archival storage (typically tape) and reconstructs a more current state by reapplying or redoing the operations of committed transactions from the backed up log, up to the time of failure.

### Security Requirements

Security systems need database storage just like many other applications. However, the special requirements of the security market mean that vendors must choose their database partner carefully.

### Software Quality Attributes

* Availability - The products in the store should be available when the need of the product is increased.
* Correctness - When a customer is ordering a product then the correct product should be shipped.
* **Maintainability** - The administrator should maintain the product stock.
* **Usability** - The product should satisfy the customer’s needs.

#### 5. System Design Description

# System Design Description

## Introduction

### Purpose

The purpose of this document is to provide an architectural design of Ticket Management System, an e-commerce application, which provides options to raise an ticket for kloc applications online. The document intends to serve as the graphical representation of architecture put together after understanding the requirements specified in the preceding document, the Software Requirement Specification.

### Scope

The document will cover the design phase of the product. The objective of Ticket Management System is to complete the software in the given period of time. NodeJS is used in combination of ReactJS and MongoDB database to develop this website with Model View Controller architecture.

### Overview

This document will contain the general definition and feature of Ticket Management System, its design constraint the overall system architecture. With the help of UML diagrams, the design of the system and the sub-systems/modules will be explained visually in order to help the programmer.

### Constraints

The hardware, software and technology used should have following specifications

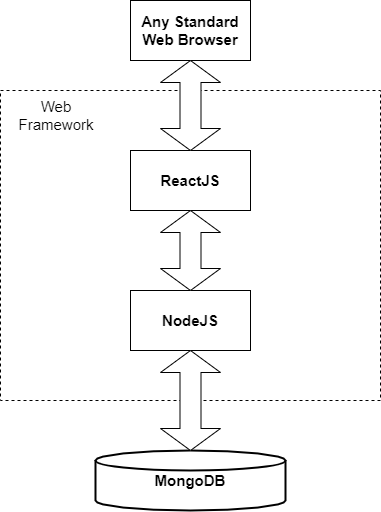
* Any device with web browser.
* The system should have internet connection.

**Some Other Constraints**

* Any device should have 384 \* 640 screen size for better visibility.
* Only registered users can use the services provided by the application.

## System Overview

### System Architecture



**Fig 5.1 System Architecture**

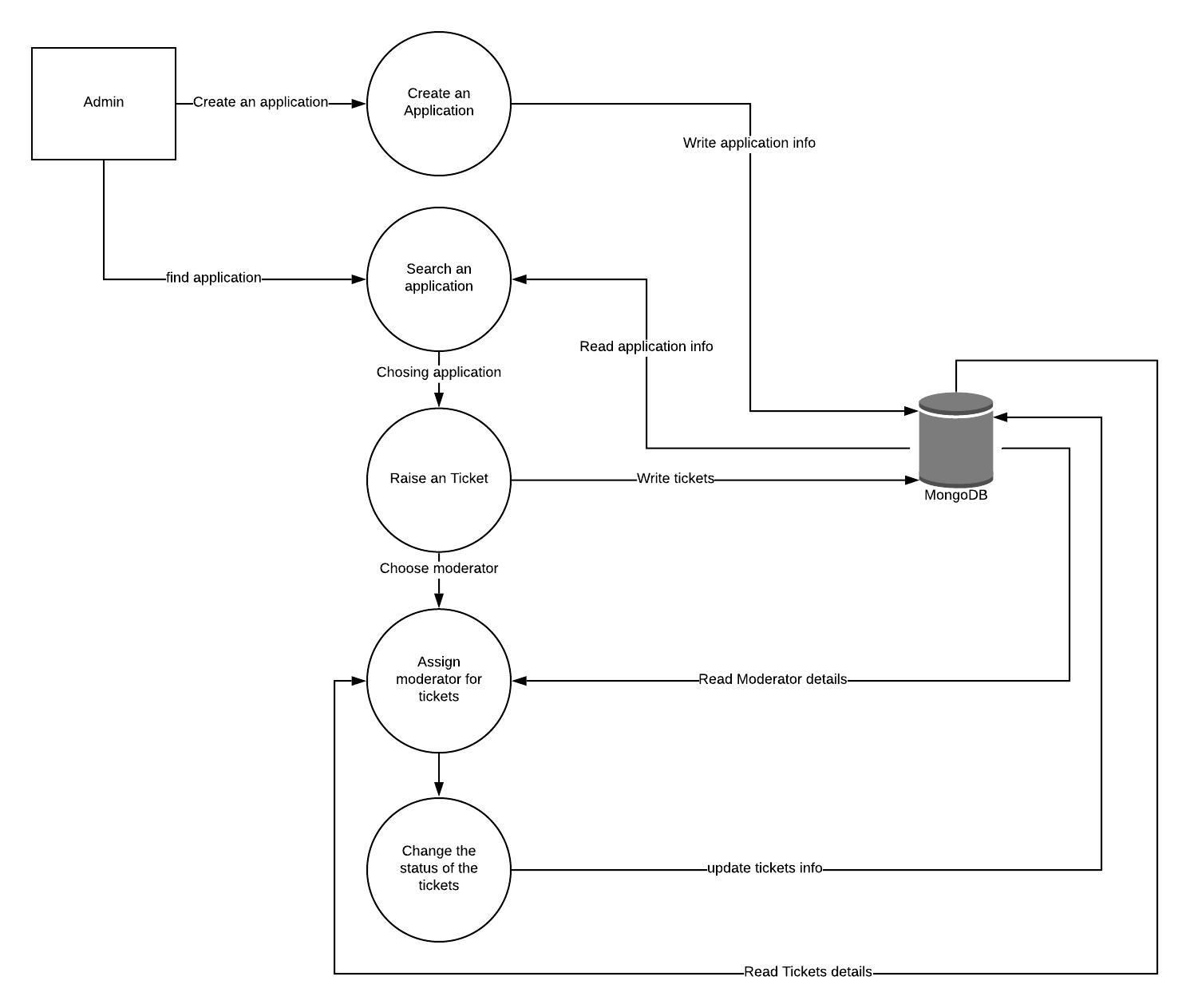
In Fig 5.1. Representation of system architecture is shown, User can use any web browser on any device having internet connectivity to access this web application. Application is developed on combination of ReactJS and NodeJS web framework, and databases used in it is MongoDB.

## Data Dictionary

Due to confidentiality of the databases maintained, the data dictionary cannot be published in any form not owned by the company.

## Functional Design

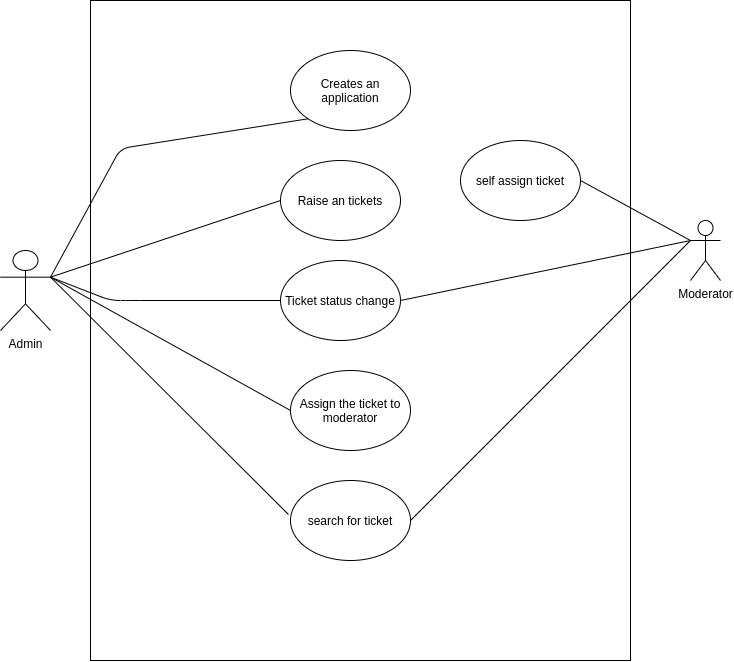
### Data Flow Diagram



**Fig. 5.2 Data Flow Diagram Level 1**

Fig. 5.2 show the data flow diagram for the admin module. The data flows between database and the activity of an action.

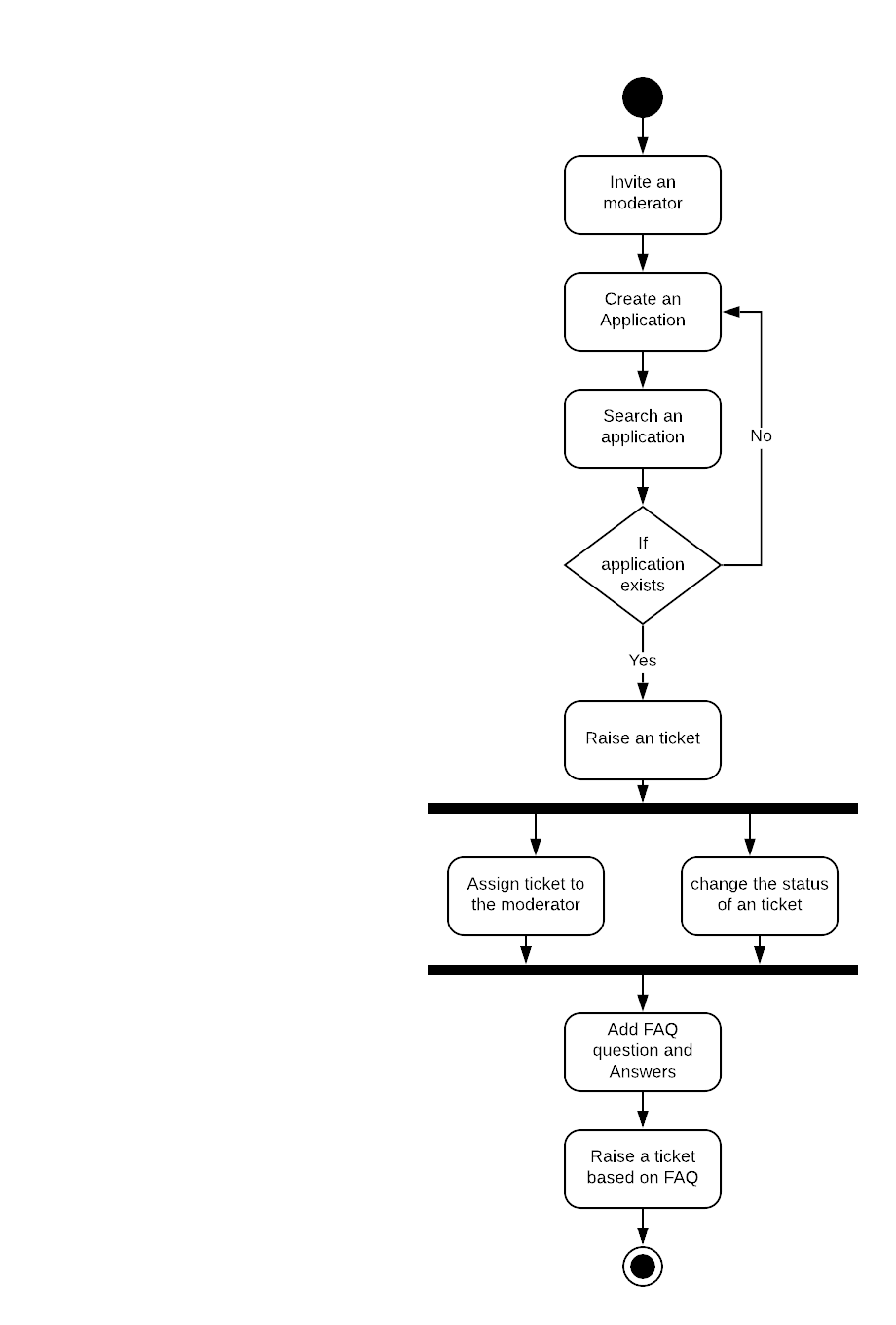
### Use Case Diagram

****

**Fig 5.3 Use Case diagram for admin**

Here, the fig.5.3 show the actor is the Admin and Moderator. The admin can creates an application, raise a tickets, change the ticket status, and assign the ticket to moderator and Search for ticket. The Moderator can assign the ticket for himself, moderator can change the status of the ticket assigned to him and search for ticket.

### Activity Diagram



**Fig 5.4 Activity Diagram for admin module**

**6. IMPLEMENTATAION**

# Implementation

## Introduction

The project shall be implemented using the MongoDB as the database, NodeJS as Backend and ReactJS as the front end. NodeJS and ReactJS runs in different server. For each action an API call is made from the ReactJS to the NodeJS. Every CRUD operation is performed by NodeJS for every CRUD action ReactJS makes an API call to the NodeJS and displays the output in ReactJS

## Pseudo Code

Due to confidentiality of the databases maintained, the data dictionary cannot be published in any form not owned by the company.

### Creating User / Groups / Functions / Roles

**Steps involved:**

* Make a request(JSON) to the endpoint that hits the function associated with the endpoint and send back a response.
* Populate the database according to the response, as to which table and which column to insert the data into.

**Implementation Logic:**

* The URL mentioned in the controller is called depending on what function is to be done.
* The corresponding method under that URL is called with a request body containing the JSON request(user-details).
* Then the wrapper and service classes are called, the business logic is applied and then it return the response as a JSON(user-details, with the success message if user is successfully created).

### Updating User / Groups / Functions / Roles

As soon as the admin/user clicks on the update icon, the UI will be populated with the searches based on the parameters selected by the admin/user

**Steps involved:**

* Make a request (JSON) to the endpoint that hits the function associated with the endpoint and send back a response. Parameters are selected based on which the search is to be done.
* Populate the UI with the list of searches sent as response

**Implementation Logic:**

* The URL mentioned in the controller is called depending on what function is to be done. Search by many ways, like search by user\_id, group\_name, function\_name etc.
* The corresponding method under that URL is called with a request body containing the JSON request(user\_id, group\_name, function\_name).
* Send response with list of searches

### Disable User / Groups / Functions / Roles

As soon as the admin/user clicks on the disable icon , the record will not be deleted, but the enabled column in the database will be updated to false.

**Steps involved:**

* Make a request(JSON) to the endpoint that hits the function associated with the endpoint and send back a response.
* Populate the UI with the list of hooks sent as response

**Implementation Logic:**

* The URL mentioned in the controller is called depending on what function is to be done.
* The user\_id, group\_id, function\_id is provide in the request body of the JSON which is to be disabled.
* The corresponding method under that URL is called with a request body containing the JSON request(user\_id, group\_name, function\_name).

**7. TESTING**

# Testing

## Testing

To access web applications, we use web browsers. These browsers runs on different devices like PC, handheld device. With each different devices their screen sizes differ, web apps should be compatible to sun on any screen size.

The general scenarios of testing activities performed are:

* Unit testing
* Integration testing
* System testing

## Test Cases

### Login Page Test Cases

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Case No | Test Case Name | Test Case  Description | Inputs | Expected  Result | Actual Result | Status |
| 1 | Login Test#1 | Login test with invalid password | Login id with incorrect password | Login failed | Login failed | Pass |
| 2 | Login Test#2 | Login test with invalid login id | Incorrect login id with password | Login failed | Login failed | Pass |
| 3 | Login Test#3 | Login test with empty user name and password | Leave two fields empty | Login failed | Login failed | Pass |
| 4 | Login Test#4 | Login test with valid id and valid password | Valid id and valid password | Login successful | Login successful | Pass |

### Create Customer/ Ticket/ Faq/ Moderator And Metrics (Graph) Display Cases

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Case No | Test Case Name | Test Case  Description | Inputs | Expected  Result | Actual Result | Status |
| 1 | Customer create test | Check if the Customer is successfully created | All the mandatory attributes of user must be given | Customer successfully create | As Expected | Pass |
| 2 | Moderator create test | Check if the moderator is successfully created | All the mandatory attributes must be given | Moderator successfully create | As Expected | Pass |
| 3 | Ticket create test | Check if the ticket is successfully created | All the mandatory attributes of ticket details must be given | Ticket successfully create | As Expected | Pass |
| 4 | FAQ create test | Check if the FAQ is successfully created | All the mandatory attributes of FAQ must be given | FAQ successfully create | As Expected | Pass |
| 5 | Metrics | Check if the Metrics is displayed proper data | Selecting the Year | Displaying the data in graph wise | As Expected | Pass |

**Appropriate messages are displayed in all the above cases.**

**8. CONCLUSION**

# Conclusion

Kloc is a software solutions provider with a keen focus on customer satisfaction. We partner with businesses to address all their IT requirements be it application development, integration, maintenance or quality initiatives. We do not offer cookie cutter solutions but tailor our offerings to adapt to our customer’s enterprise thereby ensuring high return on investment. As our slogan goes — we don’t just develop solutions, – we think IT through.

Tickets Management web application is intended to provide complete solutions for customers through a single gateway using the internet as the sole medium. It set up options for Kloc customers to raise an issue for the kloc shopify apps, customer to browse through the website and raise a ticket through online.

This project provides an easy way to raise an issue for the kloc’s applications. When a user land on an e-commerce site, they expect to find what they are looking quickly and easily.

**9. FUTURE SCOPE AND FURTHER ENHANCEMENT**

# Future Scope And Futher Enhancement

This web application was developed to be used for Kloc’s application users. This app currently targeting Kloc’s application users. At present this helps Kloc’s application uses to raise an ticket and see the status of the issue.

In future Machine Learning will be introduced, voice interpreted chat bots will be introduced. And as soon as an issue is resolved an email will be triggered to the issue’s raised customer.

**10. BIBLIOGRAPHY**

# Bibliography

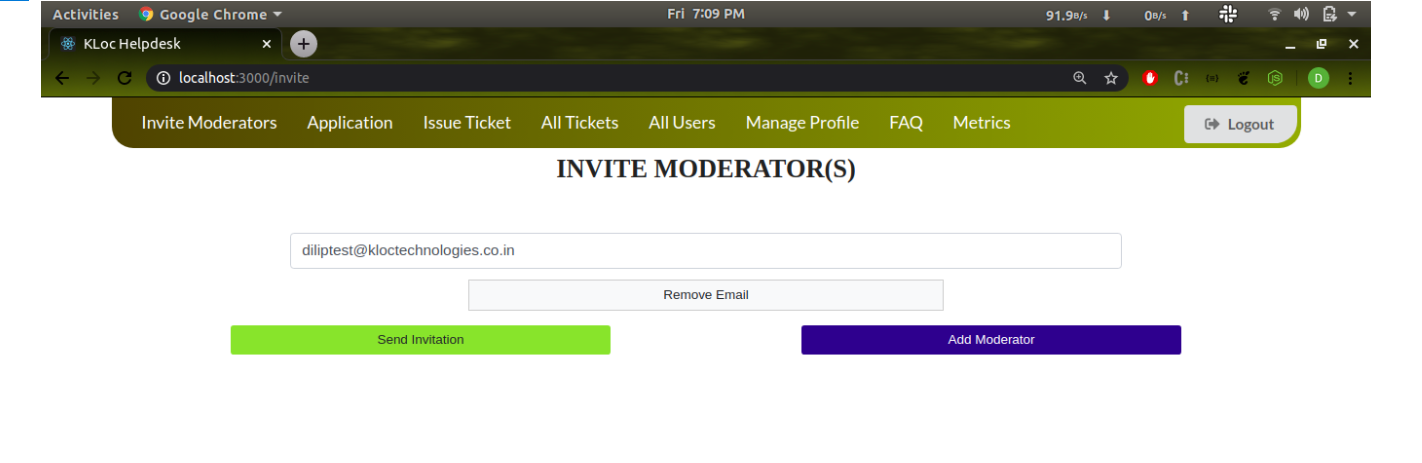
## Website Reference

1. <https://en.wikipedia.org/wiki/Node.js> (Accessed on: 04-June-2020)
2. <https://en.wikipedia.org/wiki/React_(web_framework)> (Accessed on: 04-June-2020)
3. <https://nodejs.org/en/> (Accessed on: 04-June-2020)
4. <https://reactjs.org/> (Accessed on: 04-June-2020)
5. <https://reactjs.org/blog/2020/02/26/react-v16.13.0.html> (Accessed on: 04-June-2020)
6. <https://reactjs.org/tutorial/tutorial.html> (Accessed on: 04-June-2020)
7. <https://www.youtube.com/watch?v=2jqok-WgelI> (Accessed on: 04-June-2020)
8. <https://www.youtube.com/watch?v=8SkiIAbFbNs> (Accessed on: 04-June-2020)
9. <https://www.youtube.com/watch?v=9hb_0TZ_MVI&list=PLC3y8-rFHvwgg3vaYJgHGnModB54rxOk3&index=2> (Accessed on: 04-June-2020)
10. <https://www.youtube.com/watch?v=hzLDsxPGctY> (Accessed on: 04-June-2020)
11. <https://www.youtube.com/watch?v=QFaFIcGhPoM&list=PLC3y8-rFHvwgg3vaYJgHGnModB54rxOk3> (Accessed on: 04-June-2020)
12. <https://www.youtube.com/watch?v=Y2hgEGPzTZY&list=PLC3y8-rFHvwgg3vaYJgHGnModB54rxOk3&index=4> (Accessed on: 04-June-2020)

**11. USER MANUAL**

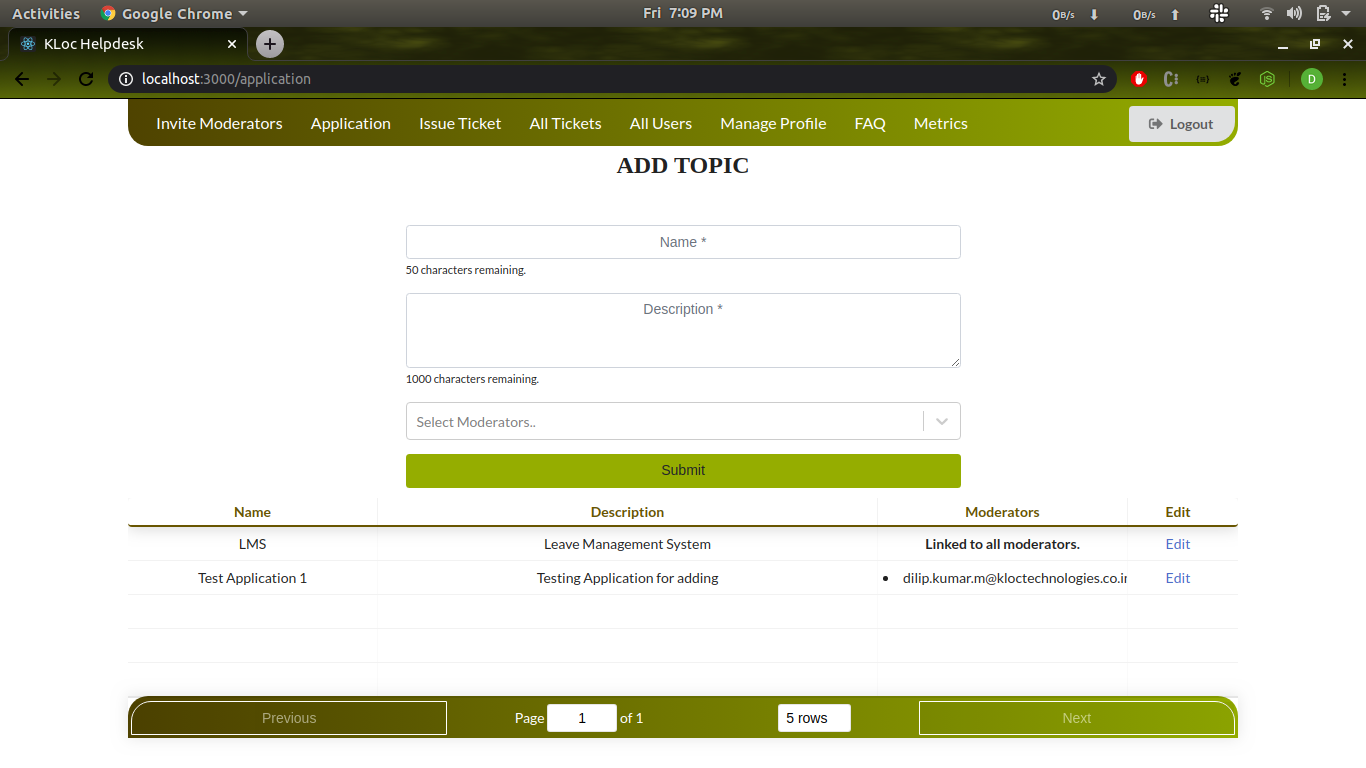
# User Manual

## Screenshot



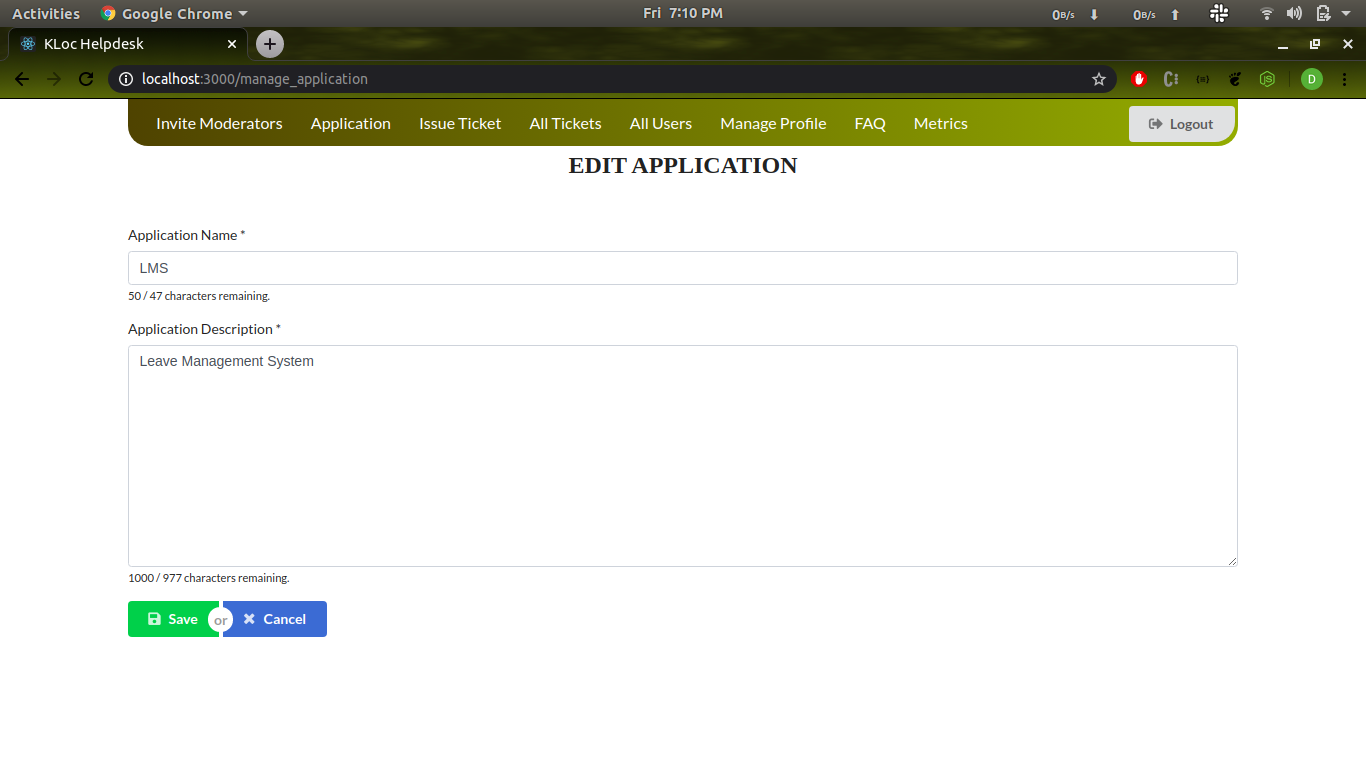
**Fig: 11.1 Admin inviting Moderator**

Fig: 11.1 In this page admin will add the moderator email and an invitation will be sent to the moderator’s email. Moderator then will access this web application and then register using his/her name and password.



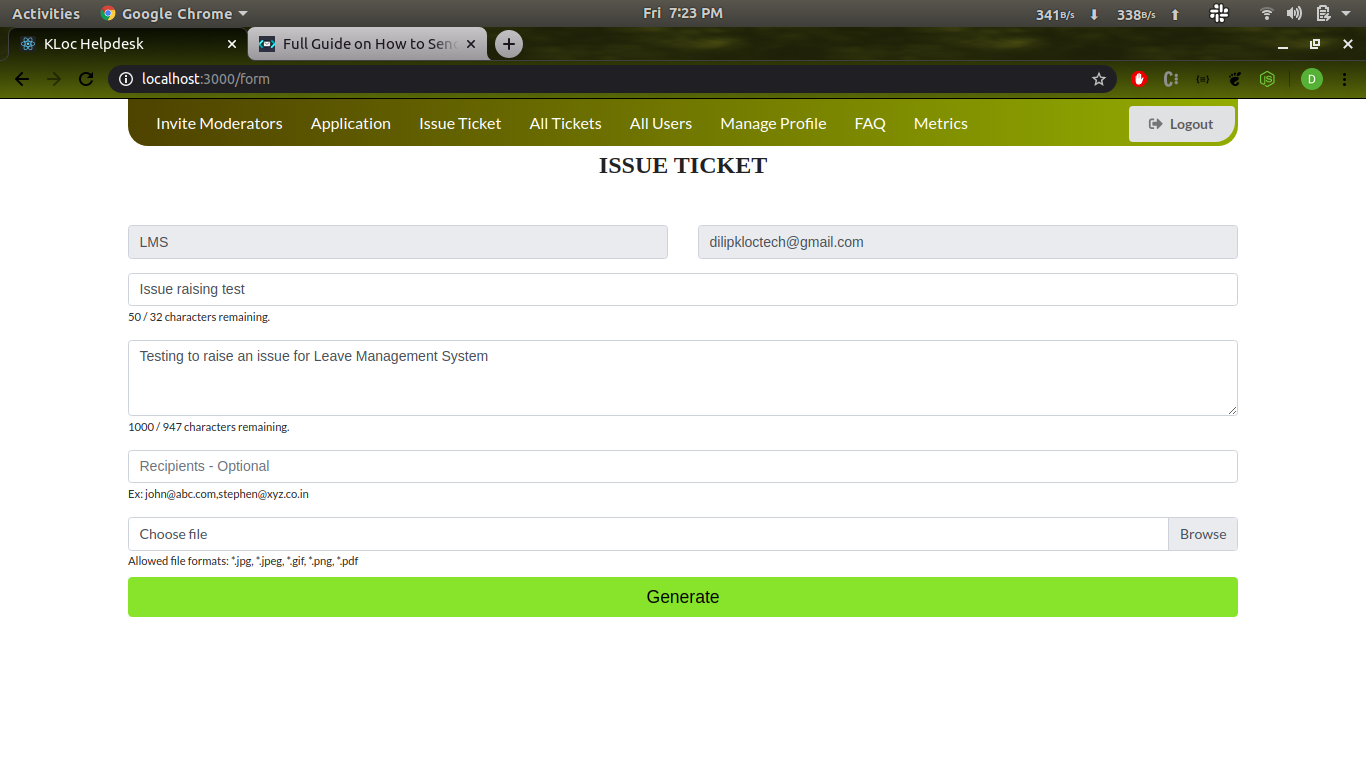
**Fig: 11.2 Admin Adding Application**

Fig 11.2 In this page admin will add the Kloc’s running application, then the application can be used by moderator’s and Kloc’s registered application, users to raise any issues for this application and even moderator’s can raise issues for the registered application.



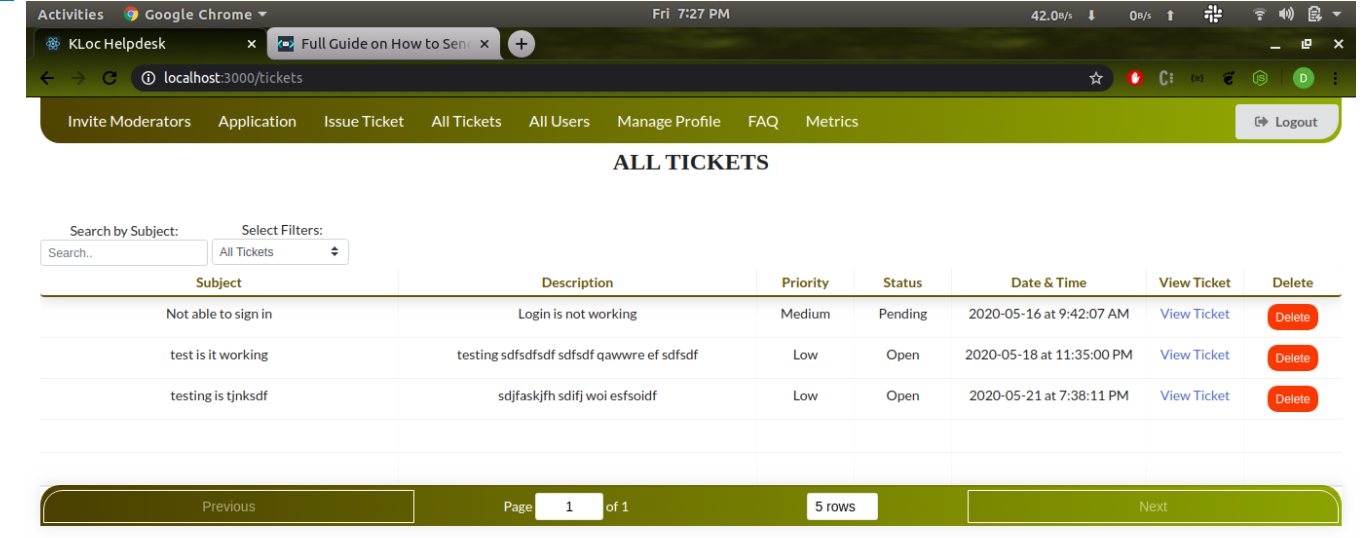
**Fig: 11.3 Admin can edit the application**

Fig 11.3 In this page admin can edit the registered application, admin can edit the application name and application description. In this the edited application will get updated and customers will see this the updated application details only.



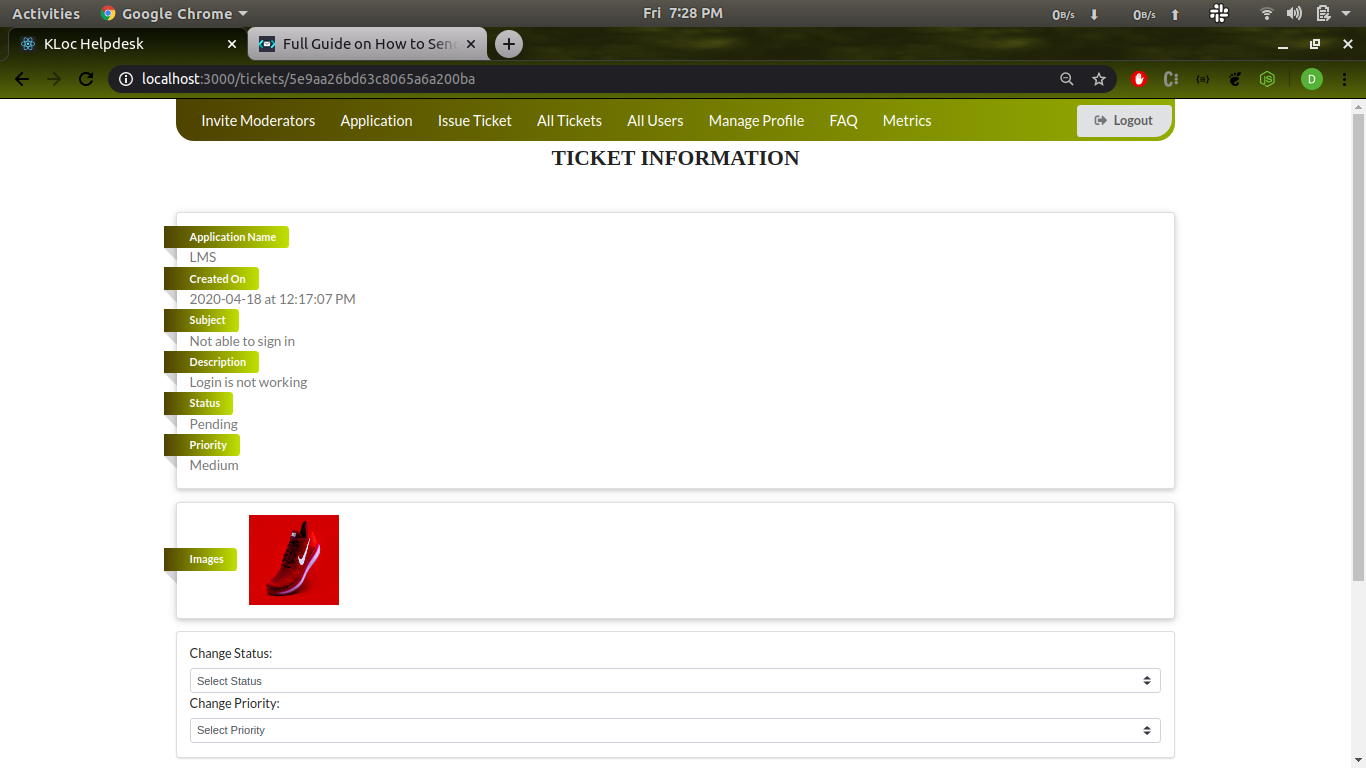
**Fig: 11.4 Admin Raising an issue**

Fig 11.4 In this page admin can raise an issue ticket for an existing application by providing the subject, description and choose the moderator for the issue. The raised ticket can be viewed by the moderator’s and can self-assign the raised tickets if the issue is not assigned to any moderator by the admin. Admin can also upload the images like screen shots.



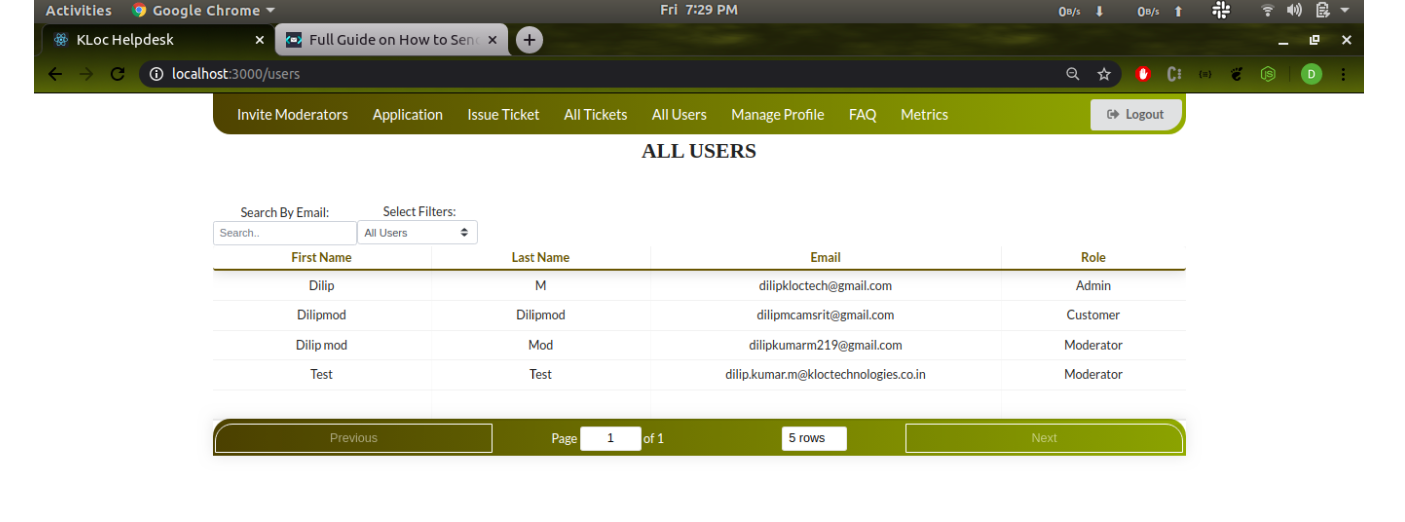
**Fig 11.5 All Tickets**

Fig 11.5 Admin can view all the Raised Tickets by moderator, customers or tickets raised by admin as well. Admin can view the status, priority and the date and time raised the tickets. Here admin can also delete the raised tickets as well. Admin can choose how many tickets can be viewed per time selectecting 5 rows or 10 rows.



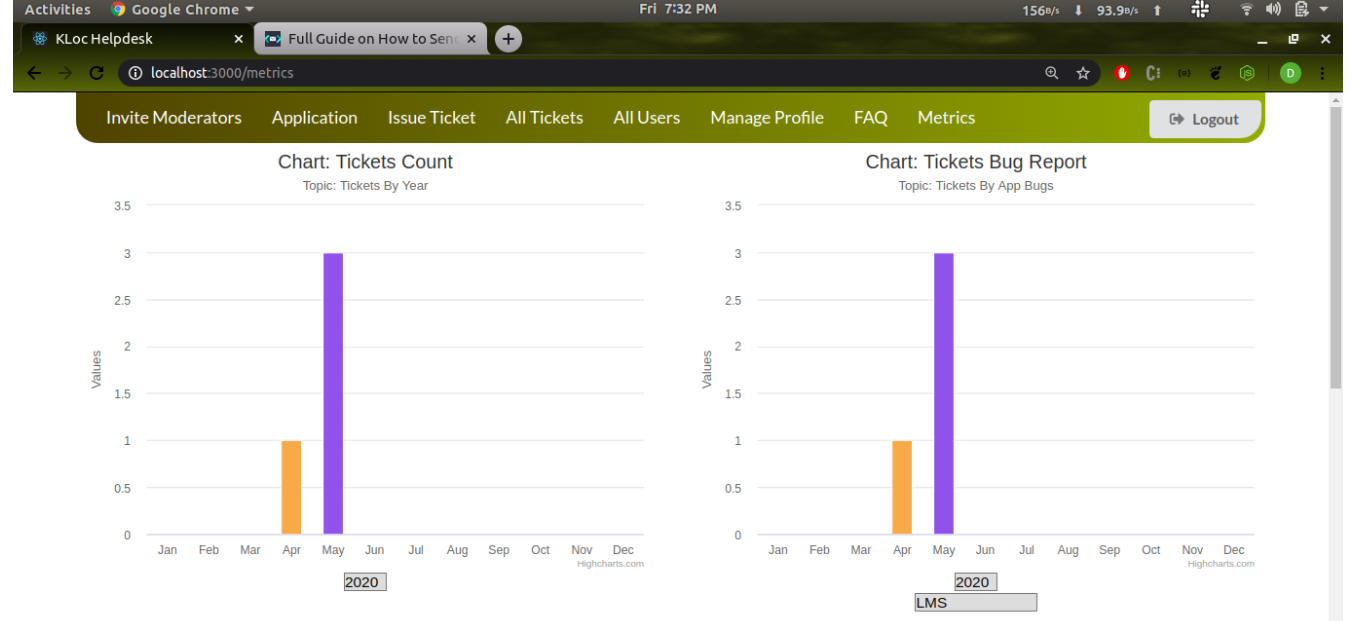
**Fig 11.6 Ticket Information**

Fig 11.6 In this form admin can view the Application name, created date time, subject, description, status, priority. Admin can change the status of the raised issue ticket. Admin can also change the priority of the raised ticket.



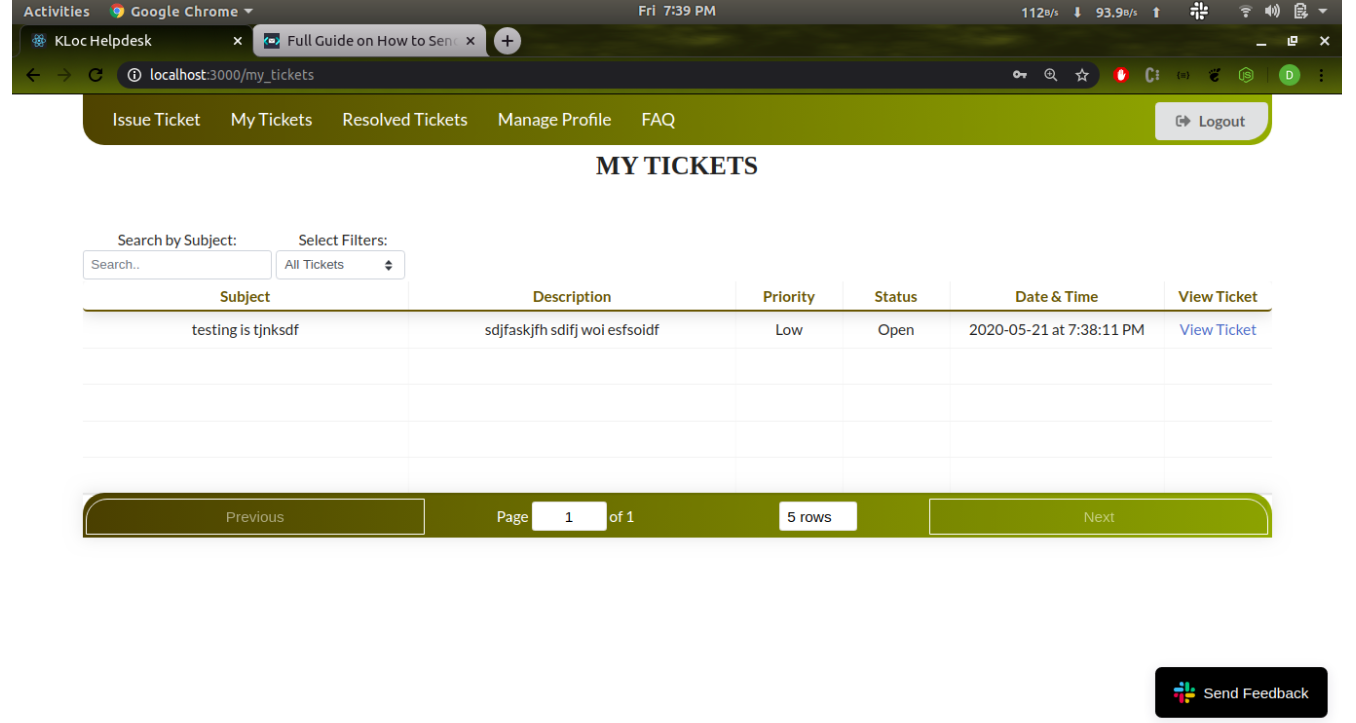
**Fig: 11.7 All Users**

Fig 11.7 Admin Can view all the users of the Kloc’s application users registered under this site. Each users role can be viewed by the admin.



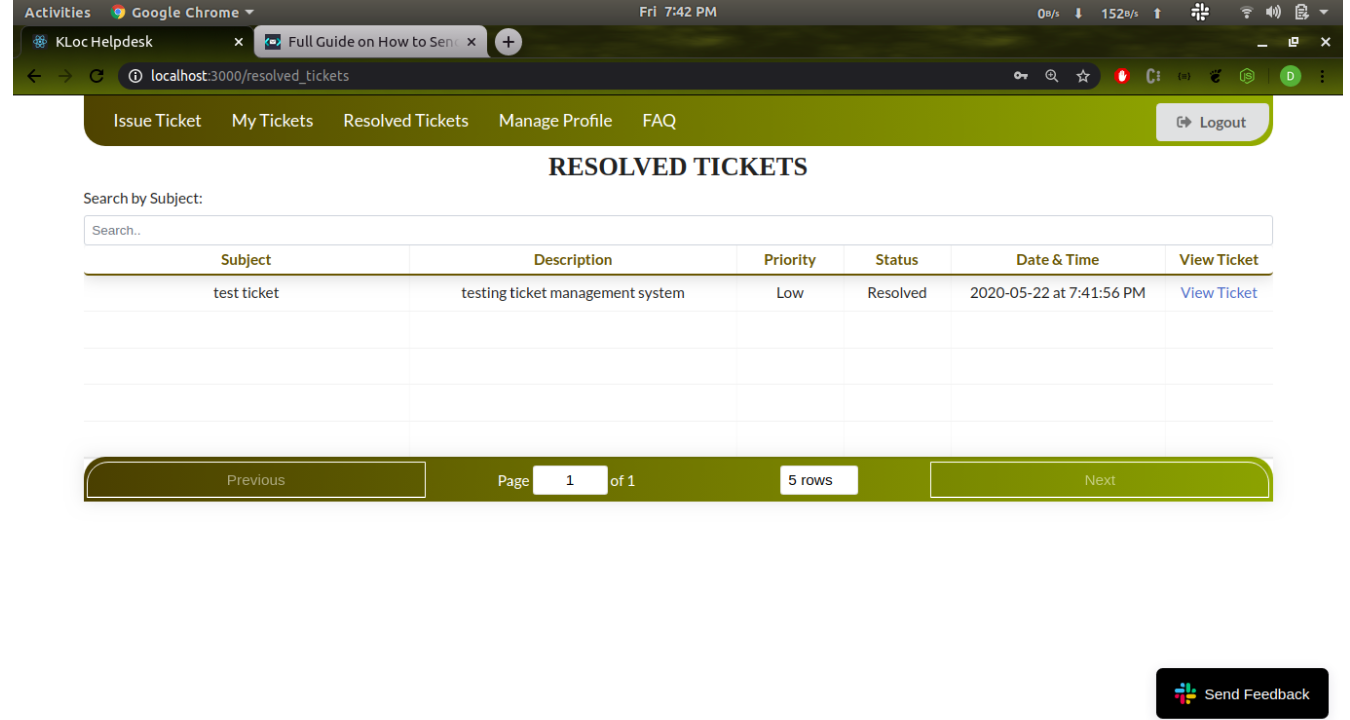
**Fig: 11.8 Metrics**

Fig 11.8 Admin can view the progress in chart format. Based of tickets raised by year. Admin can choose different years to view the issues raised by a year. Admin can select the application and view the issue raised.



**Fig: 11.9 My Tickets**

Fig 11.9 Moderator Can View all the tickets raised by him/her. Moderator can view the priority, status, description and subject of the raised tickets.



**Fig: 11.10 Resolved Tickets**

Fig 11.10 Moderator Can View all the resolved tickets. The tickets can be solved by other moderators as well. Moderator can view the priority, status, description and subject of the raised tickets.