

INTERNSHIP REPORT

(Feb 2022-May 2022)

Internship report submitted in partial fulfillment of the requirement for the
degree of Bachelor of Technology

in

Computer Science and Engineering

By

Rana Vijay Singh

181394



Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology,

Waknaghat, 173234,

Himachal Pradesh, INDIA

DECLARATION

I hereby declare that this submission is my own work carried out at Florence Capital. - Fintech Company, Gurugram from February 2022 to May 2022 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.

Submitted by:

Rana Vijay Singh

181394

Computer Science & Engineering and Information Technology Department

Jaypee University of Information Technology

ACKNOWLEDGEMENT

I would like to thank Parnita Banerjee, Manager of HR in Talent Acquisition, of Florence Capital - A Fintech Company, Gurugram for allowing me to do an internship within the organization.

I also would like to thank Mr. Rishabh Agarwal and all the people that worked along with me at Florence Capital - A Fintech Company, Gurugram with their patience and openness they created an enjoyable working environment.

I also would like to thank Mr. Rishabh Agarwal and Mr. Abhinav Kumar for mentoring me throughout my internship.

It is indeed with a great sense of pleasure and immense sense of gratitude that I acknowledge the help of these individuals.

I am highly indebted to Mr. Pankaj Kumar, Training & Placement Coordinator of our college for the facilities provided to accomplish this internship. I would also like to thank the Head of our Department Dr.Vivek Kumar Sehgal and the faculty for teaching us the skills required for this internship.

Finally, I must acknowledge the constant support and patients of my parents.

Rana Vijay Singh

181394

Jaypee University of Information Technology

TABLE OF CONTENT

Content	Page No.
Declaration By Candidate	I
Acknowledgement	II
List of Figures	IV
Chapter 01: INTRODUCTION	1
1.1 Introduction	1
1.2 Job Description	1
1.3 Objectives	2
1.4 Internship Schedule	2
Chapter 02: COMPANY DESCRIPTION	4
2.1 Florence Capital	4
Chapter 03: TOOLS AND TECHNOLOGIES USED	6
3.1 Github	6
3.2 Postman	8
3.3 MongoDB	10
3.4 Express.js	12
3.5 React	13
3.6 Node.js	16
3.7 Jira Software	17
Chapter 04: LIVE PROJECT	20
4.1 Description	19
4.2 Features of the dashboard	21
4.3 Collection dashboard	21
4.4 Credit Dashboard Changes	22
4.5 Code	24
4.5 Results	34
Chapter 05: CONCLUSION	35
5.1 Conclusion	35
5.2 Mentor's Review	35
REFERENCES	36
PLAGIARISM REPORT	37

LIST OF FIGURES

Fig no.	Figures	Page no.
1	Objectives of Software Engineering	1
2	About Florence Capital	5
3	About Constraints	6
4	Postman Application	10
5	Example of a MongoDB Application	12
6	Express Framework	13
7	An example of a ReactJS application	14
8	Features of ReactJs	15
9	An example of an Express.js application	14
10	Jira - RoadMap	18
11	Jira - Agile Boards	18
12	Jira - WorkFlow Engine	19
13	Jira - Devops Metrics	19
14	Cdl Dashboard edit box	20
15	Collection dashboard edit box	21
16	Collection dashboard	22
17	Intel dashboard	23
18	Intel dashboard - reviews & comments	23
19	Collection dialog box	24
20	Cdl dialog box	24

Chapter 01:INTRODUCTION

1.1 Introduction

An internship is a professional teachable moment that gives students real-world experience in their field of study or professional goals. An internship enables students to learn new skills while exploring and enhancing their careers. This report is a description of my ongoing internship at Florence Capital - A Fintech Company, Gurugram. This internship report details the activities that helped me achieve a bunch of my stated objectives. I was assigned the profile of “MERN Stack Developer” for my internship. In the first month of the internship, I was asked to learn different frontend languages. After the completion of this learning process, I was assigned many different tasks on the live project of the company.

1.2 Job Description

The introduction of engineering concepts to the creation, implementation, and technical management is known as software engineering. Software engineering was designed to address the challenges of low-quality software initiatives. Problems arise when software exceeds schedules, budgets, and quality expectations. It assures that the software is built in a consistent, correct, timely, cost-effective, and specification-compliant manner. Software engineering became vital to keep up with the rapid changing of user needs and the context in which the program is expected to operate.

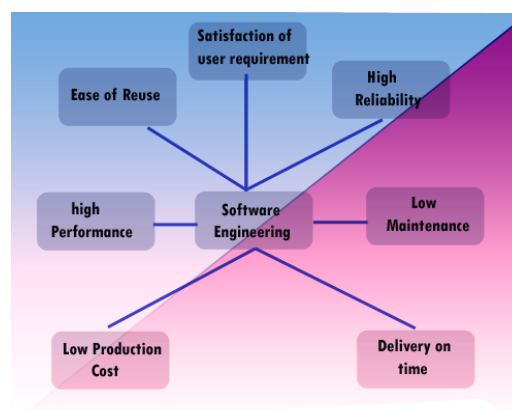


Fig 1. Objectives of Software Engineering

The Software Engineer Trainee is responsible to assist in the design and development of

software. The Software Engineer Trainee works with other members of the team to develop secure and dependable software solutions. The duties and responsibilities of this position are as follows:

- Application development (coding, programming)
- Code debugging and testing
- New software application documentation and testing
- Investigating, diagnosing, and resolving a wide range of technical challenges
- Working with senior executives
- Identifying issues and developing solutions
- Learning about new technology ahead of time

1.3 Objectives

The objective of the internship is to learn more about the programming languages like Node.js, Express.js, MongoDB, and ReactJS. These four languages are the basic requirements for my job profile. After the learning process, I was assigned to work on a live project for the company. This internship provides experience to the freshers so that they can learn more about the industry. The perspective was to gain enough knowledge so that we can work easily on the project assigned to us.

1.4 Internship Schedule

The internship plan was as follows:

➤ February 2022

This month, I revised the HTML, CSS, and Javascript which was followed by learning about Vue.js. I was assigned to develop different tasks on the frontend of the Florence capital application for hands-on experience and a better understanding of the language.

➤ March 2022

To learn more about Frontend development, I learned about React.js this month. I developed different projects for Athena Education using all three languages. This project helped me to know how the frontend works for the web development project.

➤ April 2022

After the completion of frontend development, I studied ApexChart Library and ReactJS for frontend development. On learning all the languages for MERN Stack development, I was assigned to create a Graph Task to integrate all my learnings into one project. This project helped me to understand how the web application works.

➤ May 2022

I worked on multi-selection components and internal dashboard of the company. Worked on the styling of the existing and developed pages using Material UI Framework.

Chapter 02: COMPANY DESCRIPTION

2.1 Florence Capital

Florence Capital is the first FinTech company that is dedicated to exclusively serving women. We want women to benefit from the growing economy in India.

Women have traditionally had very limited access to credit and have been left out of the credit market by and large - closing various opportunities for them. We want to open these doors through a transparent, ethical and innovative credit framework.

We envision a future in which all Indian women are financially independent. Providing loans is just the beginning of our journey towards this vision.

We are building a community of women, breaking gender stereotypes & barriers as we redefine financial freedom for women.

We use cutting-edge breakthroughs in technology and an innovative credit underwriting process to constantly innovate the process of lending and give cash loans at low-interest rates.

Florence Capital, a multi-million-dollar tech startup, is India's first fintech serving women exclusively. Countless Indian women desire financial inclusiveness; therefore we launched an app to provide credit quickly and safely. We pride ourselves on offering 100% digital loans, with zero paperwork, all within 24 hours.

At Florence, we love to work with driven team members who are eager to grow, touch lives, and improve society. Every one of us envisions a future in which all Indian women are financially independent, and providing loans is just the beginning of that journey.

Florence was founded by two Princeton-educated serial entrepreneurs and backed by some of the biggest names in global finance and technology, such as Mr. Amit Singhal (ex worldwide Head of Google Search) and Mr. Michael Novogratz (ex Fortress Investment Group, CEO of Galaxy Group Investments).

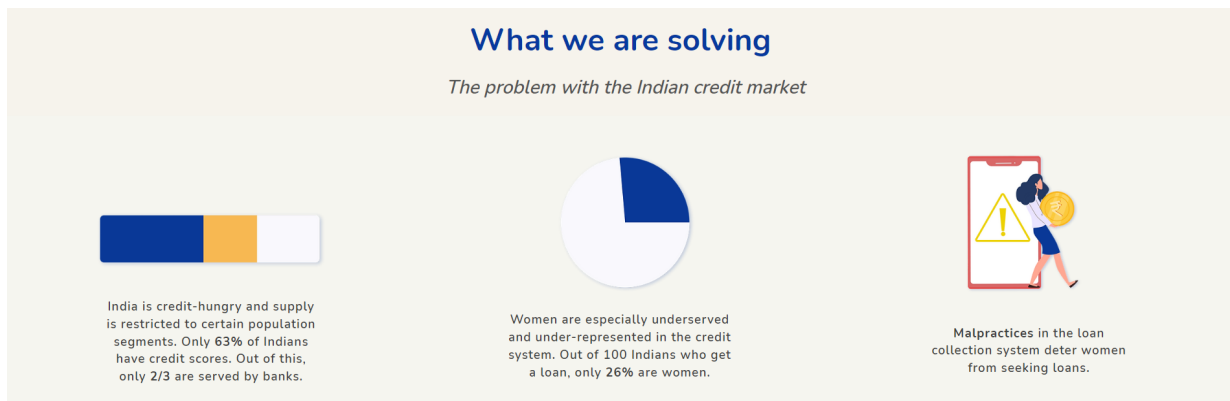


Fig 2. About Florence Capital

2.2.1 How we do it

- **Building technology and user experiences** - that are designed to resonate with them creating UI/UX which makes the loan journey smoother. Redefining gamification and how women interact with credit.
- **Strategy Automating the lending process (machine-driven lending)** - by combining innovation in credit policies and data science to automate our lending process.
- **Innovation on the credit underwriting system** - By combining the best practices in credit underwriting with unique alternate data which we are preparing, testing and perfecting in house.
- **Focusing on ethical lending** - Through a women-only customer service and loan collection team.
- **Building a community-** of women to foster professional development and entrepreneurship.

2.2.2 The constraints of the traditional credit infrastructure

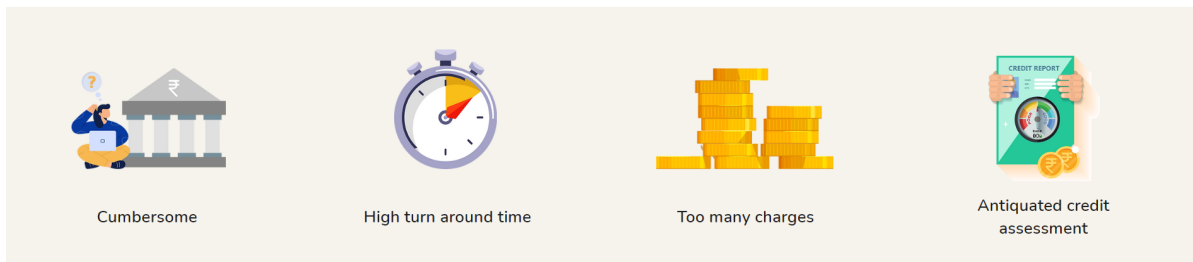


Fig 3. Constraints

Chapter 03: TOOLS AND TECHNOLOGIES USED

3.1 Github

Software developers and engineers can use GitHub to build public-facing cloud repositories for free. You can transfer a GitHub repository to your device, add and alter files locally, and then "push" your changes back to the repository, where they will be visible to the public.

Repository

A repository (sometimes shortened as "repo") is a site where all of the files for a project are kept. Each project has its repository, which you can access using a specific URL.

Forking a Repo

When you establish a new project based on an existing project, this is known as "forking." This is a fantastic tool that greatly fosters the development of new applications and projects. If you find a project on GitHub that you'd want to contribute to, you can fork it, make the modifications you want, and then republish it as a new repo. You may quickly add changes to your current fork if the original repository that you forked to create your new project is updated.

Branch

A branch represents an independent line of development. Branches serve as an abstraction for the edit/stage/commit process. You can think of them as a way to request a brand new working directory, staging area, and project history. New commits are recorded in the history for the current branch, which results in a fork in the history of the project.

Head

Git's way of referring to the current snapshot. Internally, the git checkout command simply updates the HEAD to point to either the specified branch or commit. When it points to a branch, Git doesn't complain, but when you check out a commit, it switches into a "detached HEAD" state.

Git Commit

You're ready to commit once you've staged the files you want to add. The commit message is important regardless of whether you commit using GitHub Desktop or the command line. Short commit statements that describe your change should be used. The commit messages will lead you through the history of your repository, therefore they should be informative. The following message format can be used in command-line commits:

```
git commit -m "git commit message example"
```

Pull Requests

You've forked a repository, made a fantastic change to the project, and want the original devs to notice it—perhaps even put it in the original project/repository. Create a pull request to accomplish this. The original repository's authors can view your work and decide whether or not to accept it into the official project. When you submit a pull request, GitHub provides an excellent communication channel between you and the primary project's maintainer.

3.1.2 BitBucket

Bitbucket Cloud is a Git based code hosting and collaboration tool, built for teams. Bitbucket's best-in-class Jira and Trello integrations are designed to bring the entire software team together to execute on a project. We provide one place for your team to collaborate on code from concept to Cloud, build quality code through automated testing, and deploy code with confidence.

A brief overview of Bitbucket



Best-in-class Jira & Trello integration

Bring structure to chaos and keep the entire software company, from engineering to design, in the loop. Access branches, build status, commits, and status on Jira issues or Trello cards



Build and test automatically with built-in continuous delivery

Build, test and, deploy with our integrated CI/CD solution, Bitbucket Pipelines. Benefit from configuration as code and fast feedback loops.



Secure your code

Rest easy knowing your code is secure in the Cloud and implement checks to prevent problems before they happen.



Code collaboration from concept to cloud

Transition Jira issues based on pull request status, create a merge checklist with designated approvers, and check for passing builds.



Deploy with confidence

Track, preview, and confidently promote your deployments.

3.2 Postman

Postman is a well-known API testing tool. APIs may be simply created, tested, shared, and documented with the help of this tool. Postman is an Application Program Interface (API) testing tool that may be used to create, test, develop, change, and document APIs. It's a basic graphical user interface for executing and analyzing HTTP requests and responses. When utilizing Postman for testing, you don't need to write any HTTP connection code. Instead, we use Postman to connect with the API and create test suites called collections. Almost every feature a developer can require is included in this program. This tool can do GET, POST, PUT, and PATCH HTTP queries, as well as convert APIs.

Postman is built around a set of powerful technologies that are extremely easy to use. Postman has become a useful tool for more than 8 million users. Postman is used for the following reasons:

1. Accessibility- After installing Postman on the device, simply login to the account to use it anywhere.
2. Use Collections- Users can create collections for their API calls using Postman. Each set can create multiple requests and subfolders. It will aid in the organization of test suites.
3. Test development- Every API request should include verification of a successful HTTP response status to test checkpoints.
4. Automation Testing- The Collection Runner or Newman can conduct tests in multiple repetitions or iterations, saving time for repeated tests.
5. Creating Environments- The usage of numerous environments reduces test replication because the same collection can be used in different settings.
6. Debugging- The postman console assists in successfully debugging the tests by tracking what data is being obtained.
7. Collaboration- To improve file sharing, you can import and export collections and environments. You can also share the collections via a direct connection.
8. Continuous integration- It can support continuous integration.

It's as simple as typing a URL into your browser to send a request. In Postman, we can quickly

send requests to APIs. You can use an API request to access or deliver data from a data source.

An HTTP method is required to send the API request. POST, GET, DELETE, PUT, and PATCH are some of the most widely used methods.

- GET: This HTTP method is used to access the data from an API.
- POST: This method transmits new data.
- DELETE: This is used to remove or delete the existing data.
- PATCH: This method is used to update the existing data.
- PUT: This method is used to update the existing data.

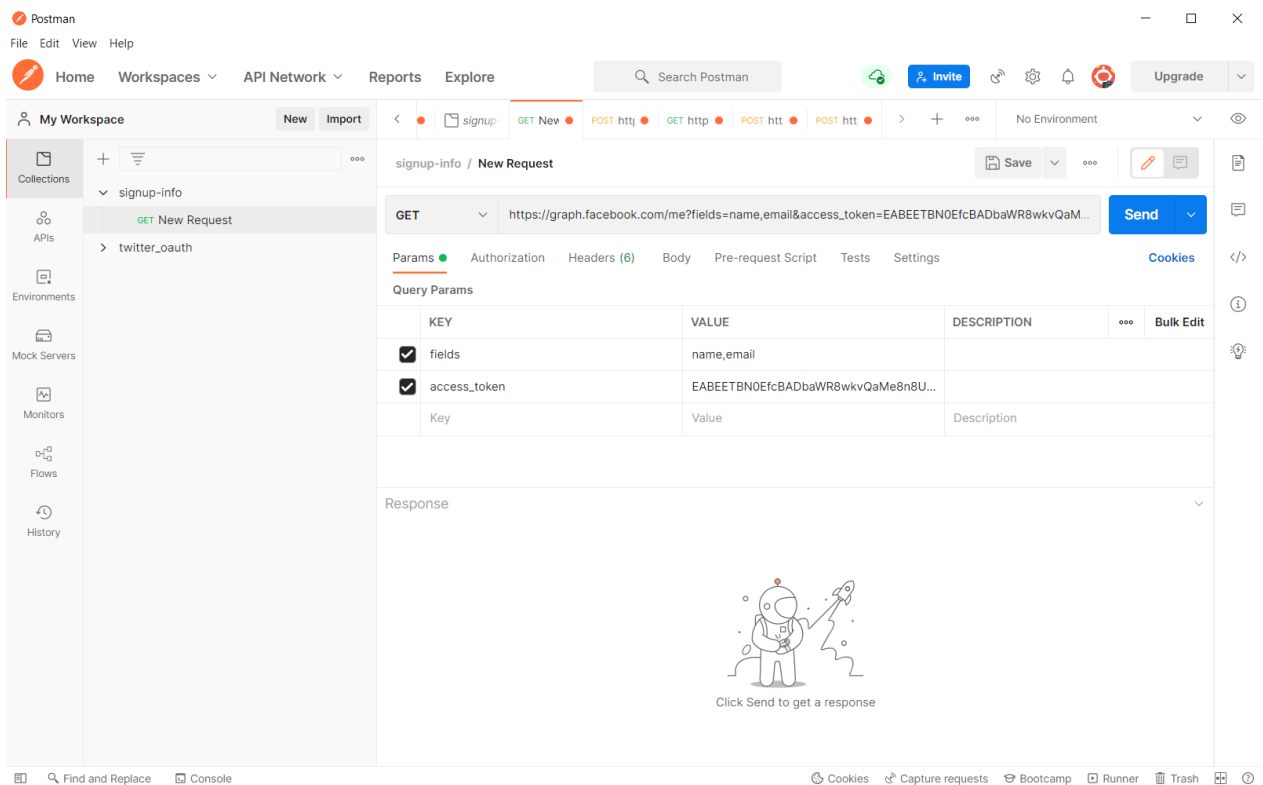


Fig 4. Postman Application

3.3 MongoDB

MongoDB is a document-oriented, cross-platform database with exceptional productivity, availability, and easy handling. MongoDB is built on the collection and document concepts.

Database

A database is a tangible repository for data. On the file system, each database has its own collection of files. Typically, a MongoDB server provides many databases.

Collection

In MongoDB, a collection is a group of documents. An RDBMS table is the same thing. A collection exists within a single database. Collections do not enforce a schema. Within a collection, different elements can be found in separate documents. In most cases, every one of the documents in a collection has the same or similar functions.

Document

A document is made up of a set of key-value pairs. Documents have a dynamic schema. Documents in the same collection don't really want to have the same number of fields or architecture, and similar variables in a collection's document can contain data.

3.4.1 Advantages of MongoDB over RDBMS:

- MongoDB is a way to store data that allows you to store multiple documents in one collection. A document's amount of features, content, and size may differ from one to the other.
- The structure of a single thing is clear.
- No complicated connections here.
- Deep querying capabilities. MongoDB supports dynamic querying on documents using a document-based markup language that's also nearly as effective as SQL.
- Tuning.
- The scalability of MongoDB is simple.

- No conversion or mapping of application components to database objects is required.
- The (windowed) work set is stored in internal memory, allowing for quicker data access.

3.4.2 Uses of MongoDB:

- Data is saved as JSON-style documents in document-oriented storage.
- Indexing can be done on any attribute.
- Replication and high availability.
- Auto-Sharding
- Rich searches
- For real-time information.
- Expert assistance is provided by MongoDB.

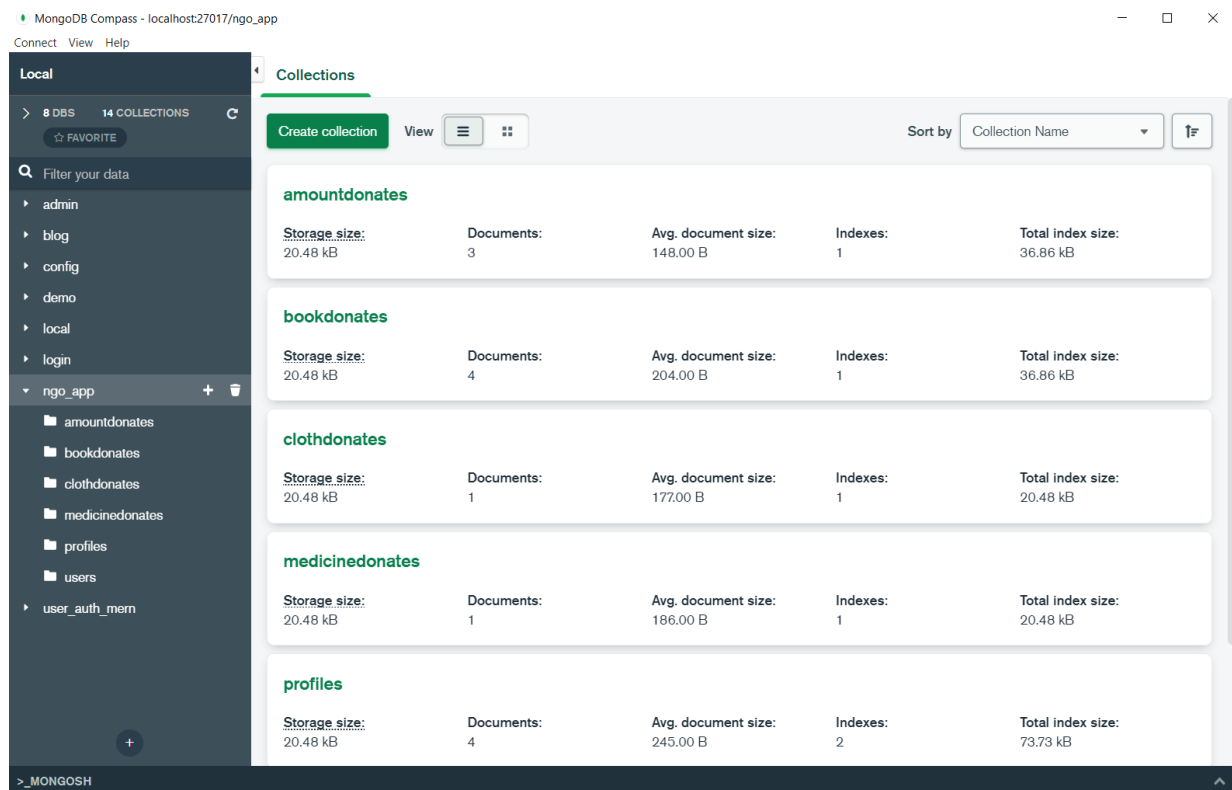


Fig 5. An example of a MongoDB Application

3.4 Express.js

It is a Node.js web framework that includes a wide range of features for developing web and mobile apps. It makes it easy to swiftly build Node-based Web apps. The following are some of the important aspects of the Express framework:

- Allows middlewares to build up responses to HTTP queries.

- Creates a routing table depending on the HTTP Method and Address to perform various actions.
- Allows you to dynamically generate HTML pages using template arguments.



Fig 6. Express Framework

3.4.1 Advantages of Express.js

1. Quickly and easily creates Node.js web applications.
2. Configuration and customization are simple.
3. You could use HTTP methods and URLs to define application routes.
4. There are various middleware modules included that can be used to carry out extra requests and responses.
5. Several template engines, such as Jade, Vash, EJS, and others, are easy to use.
6. Allows you to choose a middleware for error handling.
7. It's simple to provide static files and resources from your app.
8. Creates a REST API server for you.
9. Connecting to databases like MongoDB, Redis, and MySQL is simple.

3.5 React

ReactJS is a Javascript framework for constructing declarative, fast, and adaptable user interface components. It's a component-based front-end framework that manages the app's view layer exclusively. Jordan Walke, a software engineer at Facebook, conceived the idea. Facebook invented and maintains it, and it's later used in Facebook products like WhatsApp and Instagram. Although ReactJS was designed around 2011 for Facebook's feed area, it was first publicly published in May 2013.

The most common website architecture today is MVC (model view controller). React is the view

in the MVC design, whereas Flux or Redux is the architecture.

A ReactJS application is made up of numerous components, each being responsible for generating a reusable piece of HTML code. Components are the foundation of all React programs. Complex applications can be developed using simple building blocks by nesting these components with other components. ReactJS interconnects the DOM-based technique for populating data in the HTML DOM. Because the virtual DOM alters specific DOM components rather than refreshing the full DOM, it is fast.

To create a React app, we create React components that correspond to various aspects. The application structure is made up of these components, which are arranged into higher-level components. Consider a form containing the required fields, labels, and buttons, among other things. Each form component can be created as a React element, which we then combine to build the form component. The form elements would describe the structure and parts of the form.

ReactJS' major purpose is to construct a User Interface that aids program performance. It makes use of a virtual DOM (JavaScript objects), that speeds up the app. In JavaScript, the virtual DOM is quicker than the traditional DOM. ReactJS can also be used on server and client sides, as well as with other frameworks. Component and data structures are used to frame and make app maintenance easier.

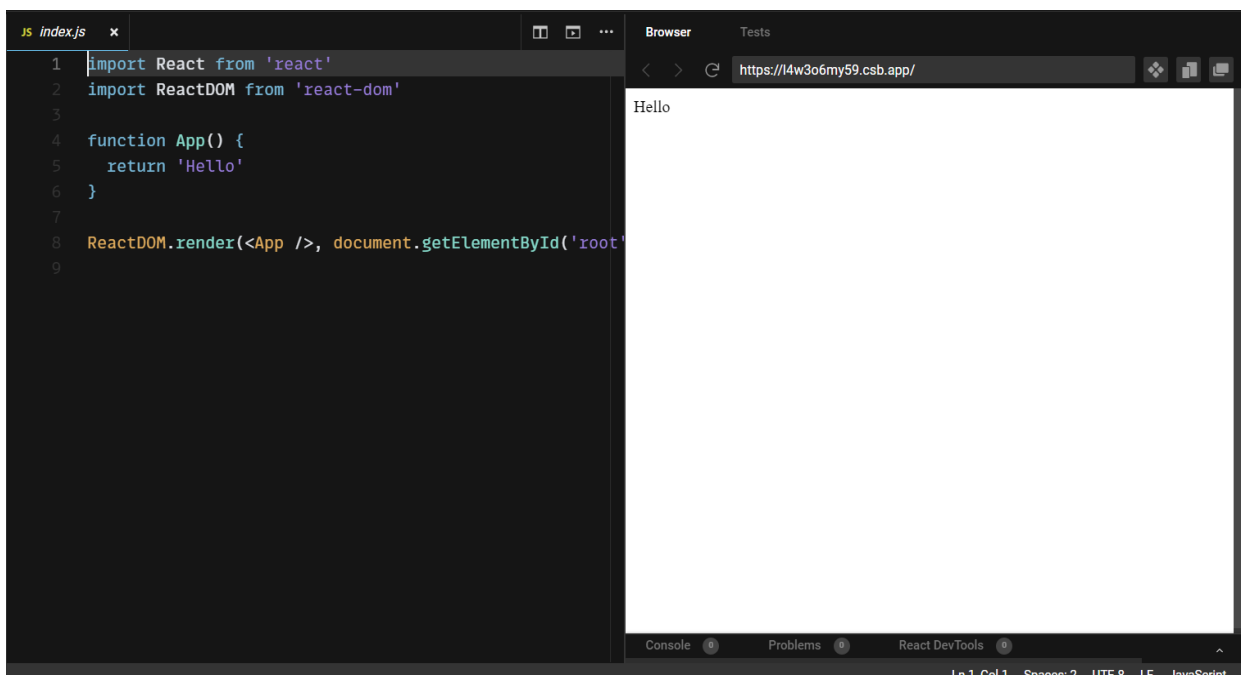


Fig 7. An example of a ReactJS application

3.5.1 Features of ReactJs

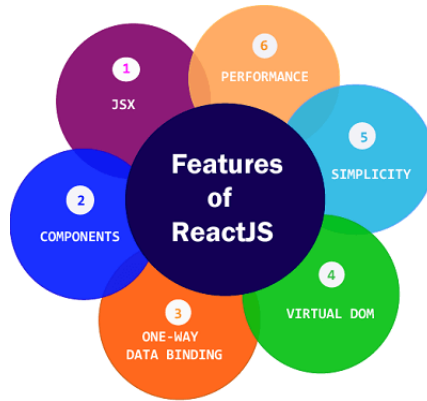


Fig 8. Features of ReactJS

ReactJS is quickly becoming the most popular JavaScript framework among web developers. It is an important part of the front-end ecology. The following are some of ReactJS' key features:

- JSX

JavaScript XML is abbreviated as JSX. It's a syntax extension for JavaScript. ReactJS uses an XML or HTML-like syntax. This syntax is converted into React Framework JavaScript calls. It enhances ES6 to allow HTML-like text and JavaScript react code to coexist. It is not required to utilize JSX, however, it is strongly recommended in ReactJS.

- Components

Components are at the heart of ReactJS. A ReactJS application consists of several components, each with its logic and controls. These components are reusable, which makes it easier to keep the code clean while working on larger projects.

- One-way Data Binding

ReactJS is designed to handle one-way data binding or unidirectional data flow. One-way

data-binding allows you to have more control over your application. If the data flow occurs oppositely, additional characteristics are required. Because components are designed to be immutable, and the data they contain cannot be modified, this is the case. Flux is a pattern that aids in data unidirectionality. This increases efficiency by making the application more versatile.

- Virtual DOM

The original DOM object is represented by a virtual DOM object. It functions similarly to a one-way data binding. The complete UI is re-rendered in virtual DOM representation whenever any changes are made to the web application. Then it compares the differences between the old and new DOM representations. After that, the true DOM will only update the things that have changed. This speeds up the application and eliminates memory waste.

- Simplicity

ReactJS makes use of a JSX file, which makes the program easy to code and comprehend. We already know that ReactJS is a component-based solution that allows you to reuse code as needed. This makes it easy to use and understand.

- Performance

The speed of ReactJS is well-known. This distinguishes it from other frameworks currently available. This is due to the fact that it manages a virtual DOM. The Document Object Model (DOM) is a cross-platform computer programming API for HTML, XML, and XHTML. The DOM is purely memory-based. As a result, we didn't even write directly to the DOM while creating a component. Instead, we'll create virtual components that will be converted into the DOM, resulting in smoother and faster performance.

3.6 Node.js

Node.js is an open-source and cross-platform JavaScript runtime environment. It is a commonly used tool for almost any task. Node.js runs the V8 JavaScript engine, which is at the core of Google Chrome, outside of the browser. As a consequence, Node.js is lightning quick.

Node.js software runs in a single operation rather than creating a new process for each request. The standard library of Node.js offers a set of asynchronous I/O semantics that prohibit JavaScript code from stopping, and modules in Node.js are frequently built using non-blocking paradigms, thus blocking behavior is an exception to the general rule.

When Node.js performs an I/O operation, such as receiving from the connection, entering a database, or entering the filesystem, instead of suspending the process and spending CPU cycles awaiting a response, Node.js will begin the activities as soon as the response is received. This allows Node.js to handle hundreds of concurrent connections without incurring the cost of thread concurrency control, which may be a major source of mistakes.

Node.js has a significant benefit since billions of frontend programmers who develop JavaScript for the internet can now generate server-side code in combination with client applications without needing to learn another language. You shouldn't have to rely on all of our users' browsers to upgrade to use the new ECMAScript standards in Node.js because you can choose ECMAScript version use by updating the Node.js edition, and you can additionally facilitate certain experimental features by running Node.js with flags.

3.7 Jira Software

Jira Software Cloud helps teams of all types manage work, in whatever flavor of agile works best. Jira embraces the real agile philosophy by giving you a simple platform that is also highly customizable so that you can implement the process and elements that work best for your case/project and with a rich out-of-the-box feature set that represents elements of multiple agile project management techniques.

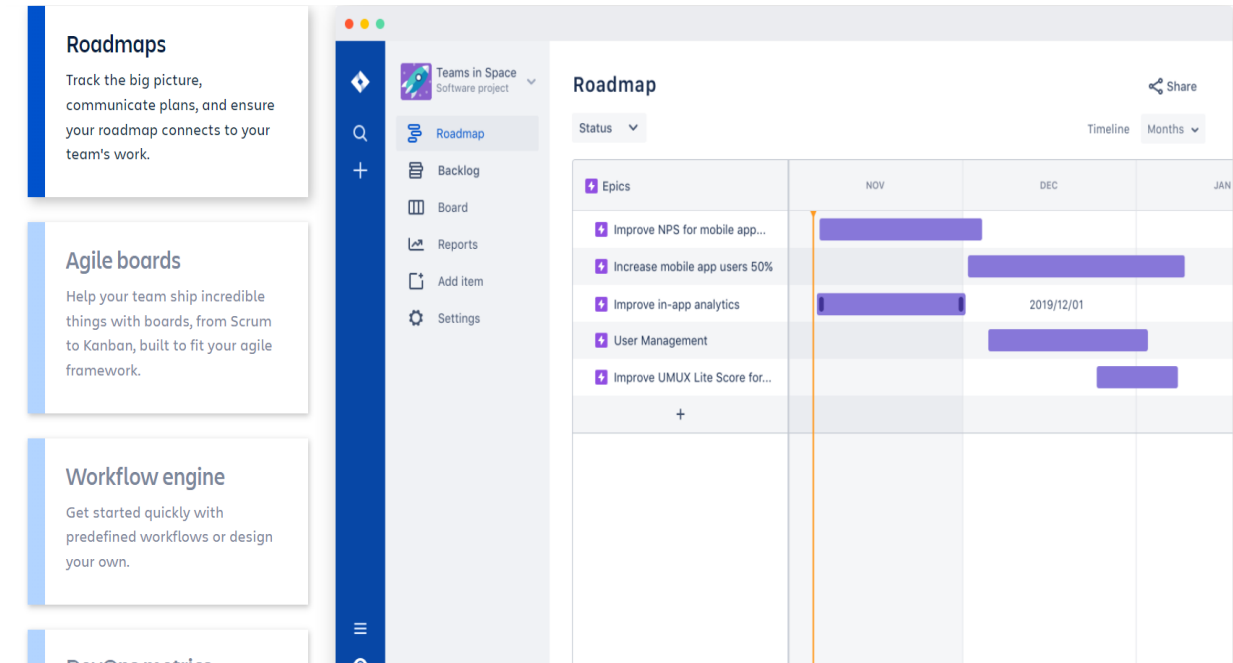


Figure 10: Jira - RoadMap

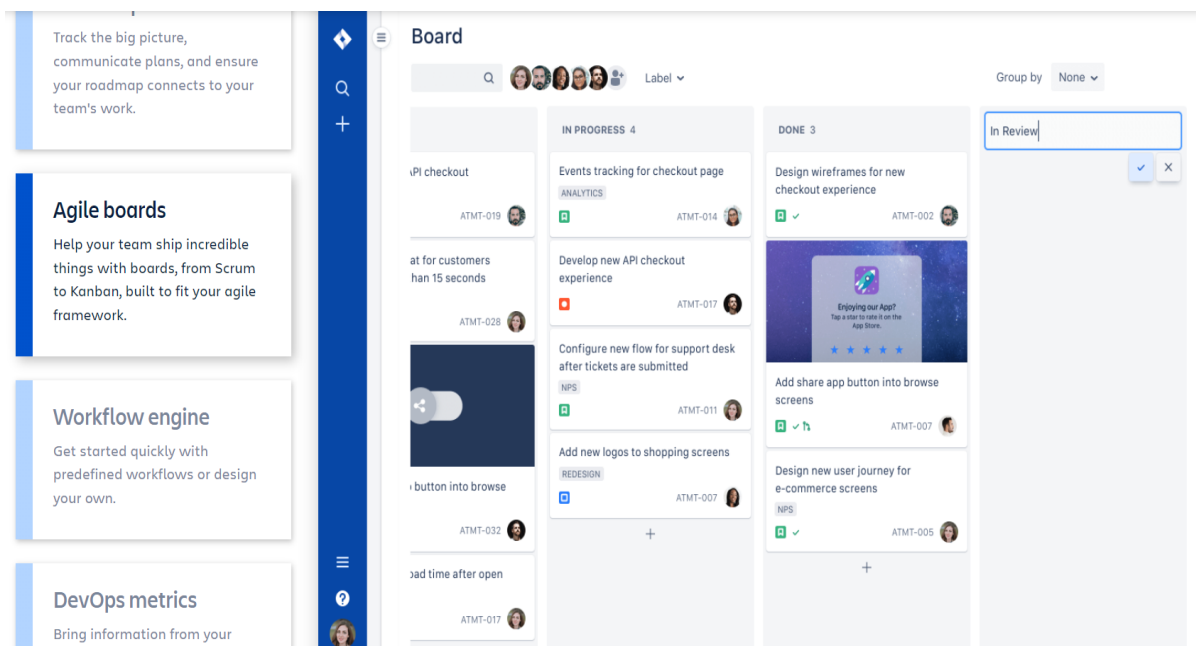


Figure 11: Jira - Agile Boards

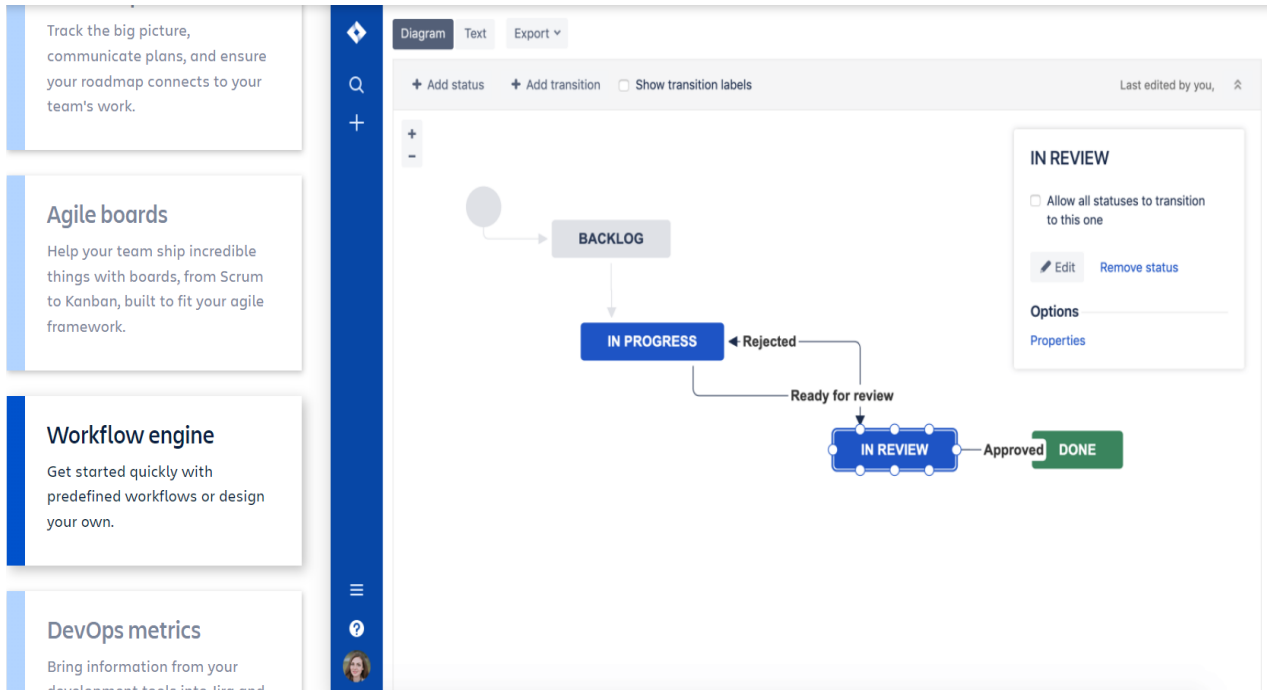


Figure 12: Jira - WorkFlow Engine

The screenshot shows the Jira DevOps Metrics view for a task titled 'Build Android application package' (ID: MOB-27 / MOB-22). The task status is 'In Review'. The assignee is Blair Bell and the reporter is Rahul Ramsey. The story point estimate is 3. The development branch has 10 commits (2 hours ago), 1 pull request (CLOSED), and 2 builds (successful). The release 'Production' is ON, and 'Deployment Feature Flag 1' is also ON. The activity log shows:

- Blair Bell (November 7, 2018, 1:51 AM): data has been sent to PA and they are reviewing (1 reaction).
- Blair Bell (November 7, 2018, 1:51 AM): what is the date for review? (1 reaction).

The sidebar on the left is identical to the one in Figure 12.

Figure 13: Jira - DevOps Metrics

Chapter 04: LIVE PROJECT

4.1 Description

The major task of the project is to Collection and CDL team comments/Feedback should be accessible to Credit Team. I mainly have to create three internal dashboards for the company. Which were Cdl dashboard, Collection Dashboard, and Credit dashboard Changes.

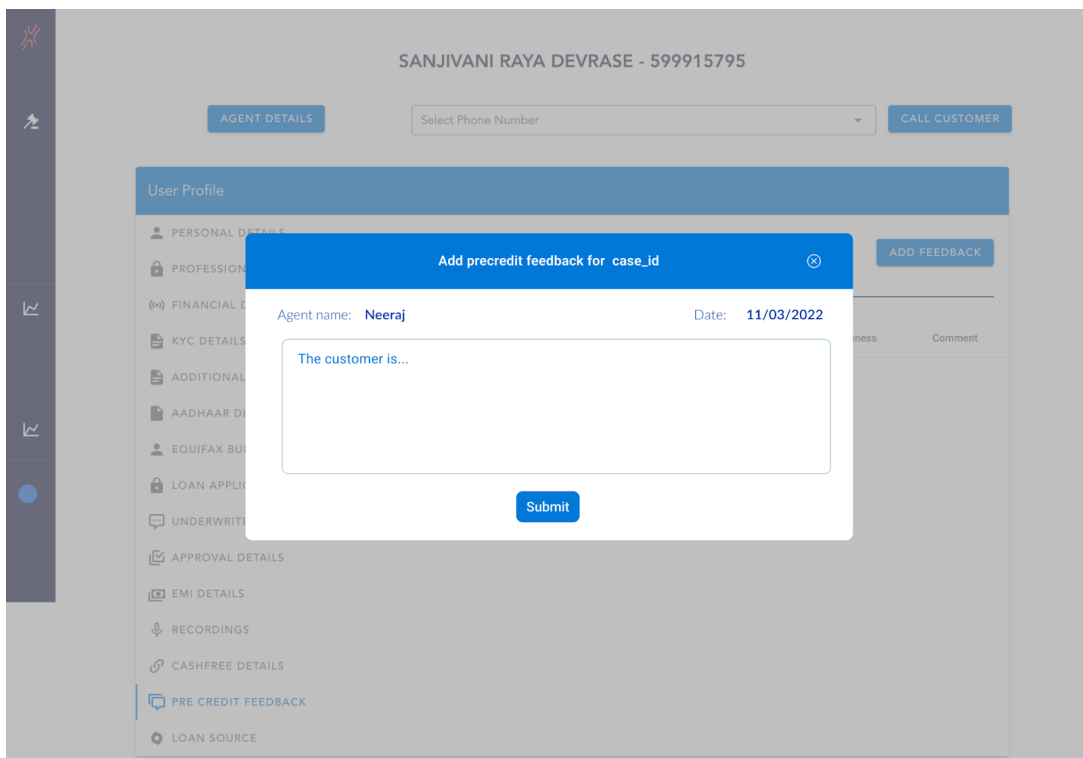


Fig 14. Cdl Dashboard edit box

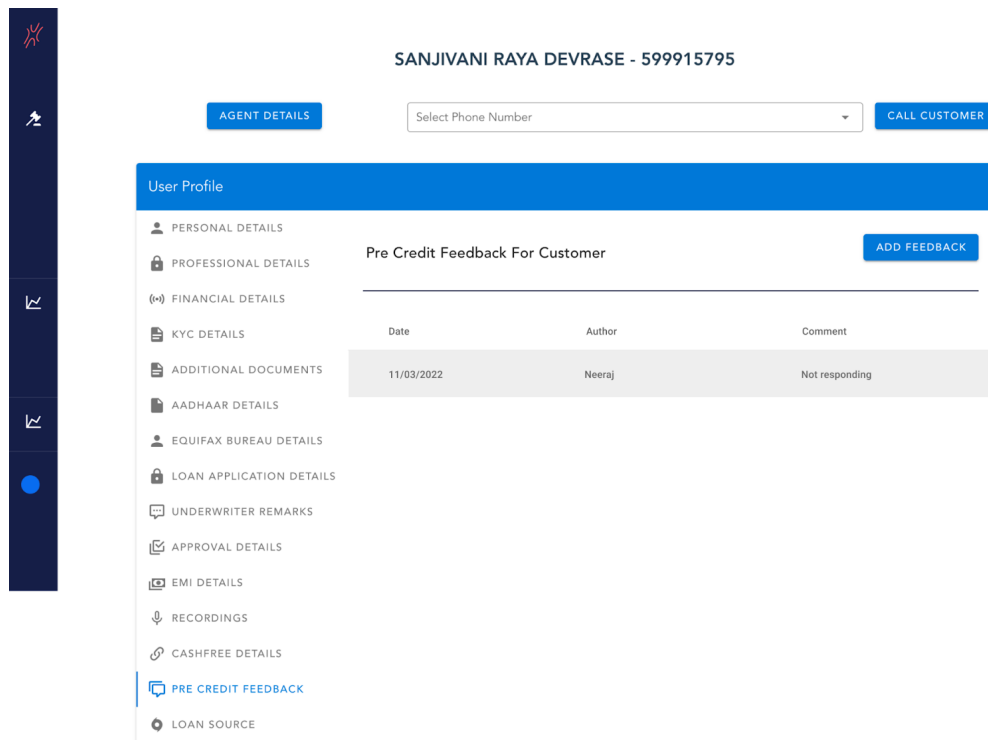


Fig 13. Cdl Dashboard

4.2 Features of the dashboard

Feature 1: In the Pre-Credit Feedback tab, Agents can write comments about customers by pressing the “Add Feedback” button. (Screen 1)

Feature 2: It will be saved by date and agent name. (Screen 2)

Feature 3: Agents can write multiple comments.

Feature 4: very same comments will be visible on Credit Dashboard as well.

4.3 Collection dashboard

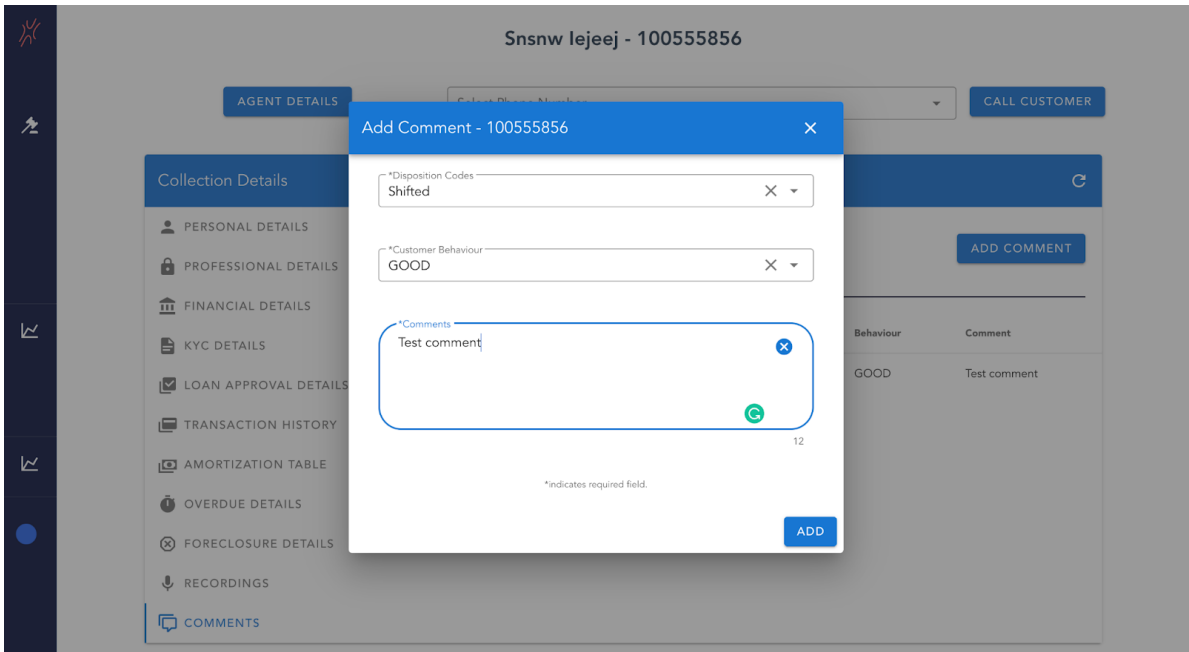


Figure 15: collection dashboard - edit box

Feature 1: Agents can write comments about customers. (present CDL design is to remain the same). (Screen 1)

Feature 2: Agents will fill the category (Drop-down). (Screen 1)

Feature 3: It will be saved by date and agent name.

Feature 4: Agents can write multiple comments.

Feature 5: very same comments will be visible on Credit Dashboard.

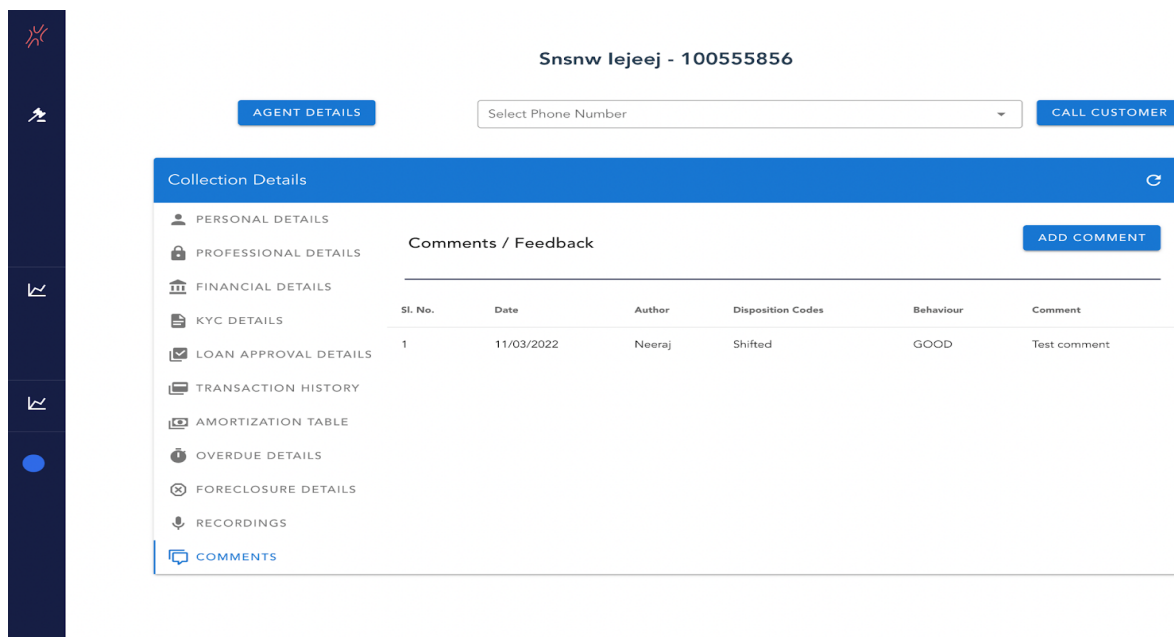


Figure 16: collection dashboard

4.4 Credit Dashboard Changes

Create Intel Page on Credit Dashboard and Integrating the Cdl dashboard and the collection dashboard in a view only mode.

Feature 1: There is to be a separate tab,"Intel" for feedback and comments of CDL and collection teams.

Feature 2: If there are any comments, a green dot will show up in the Intel tab.

Feature 3: In the Intel tab, credit officers can click on the feedback/comments and see the details left by CDL/collection.

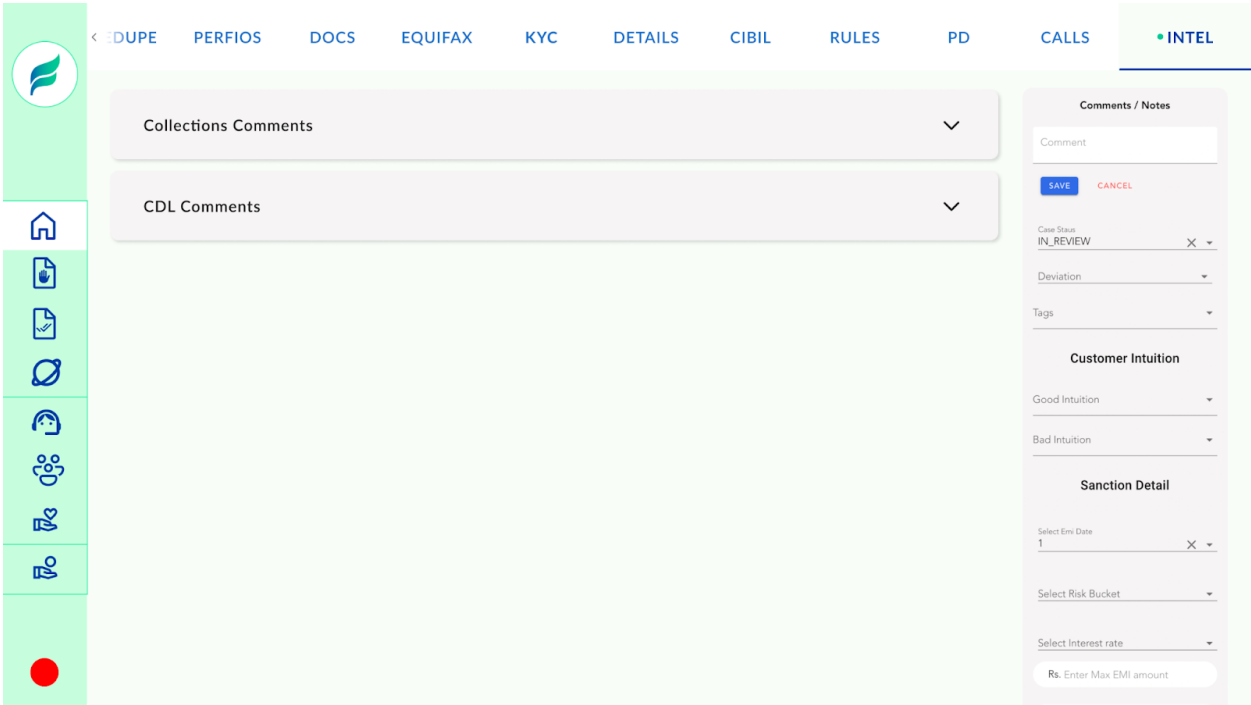


Figure 17: Intel dashboard

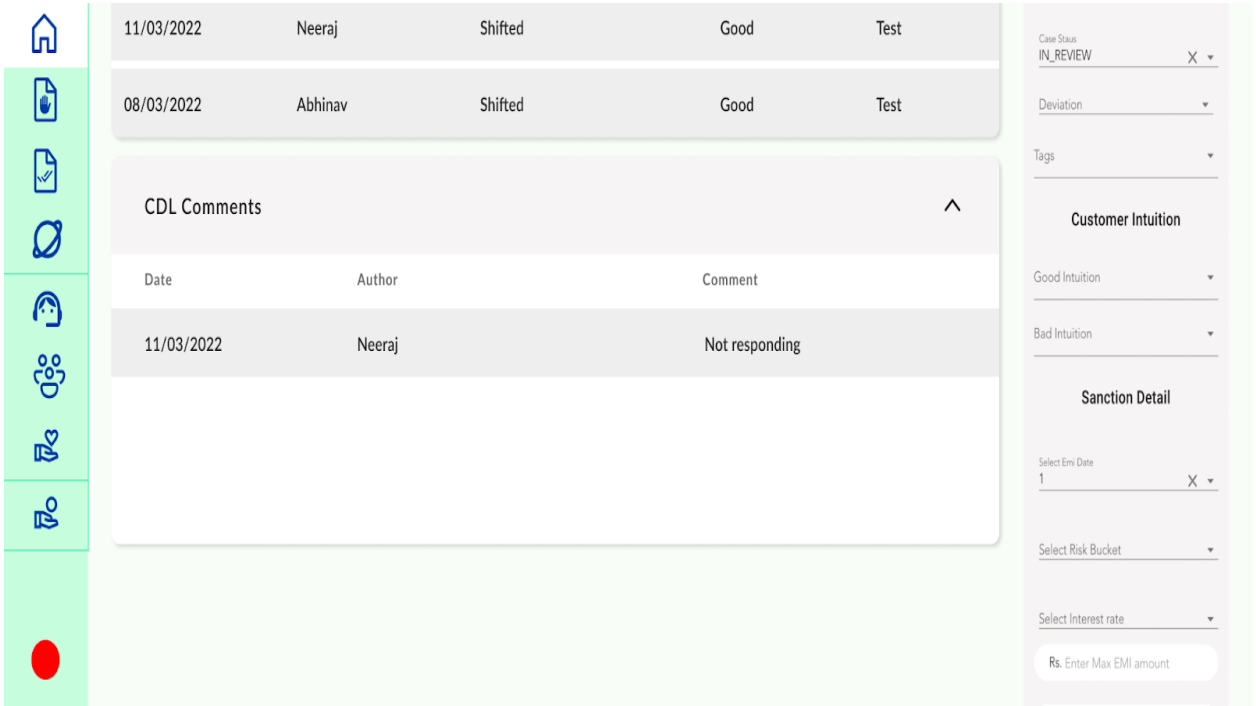


Figure 18: Intel dashboard - reviews and comments

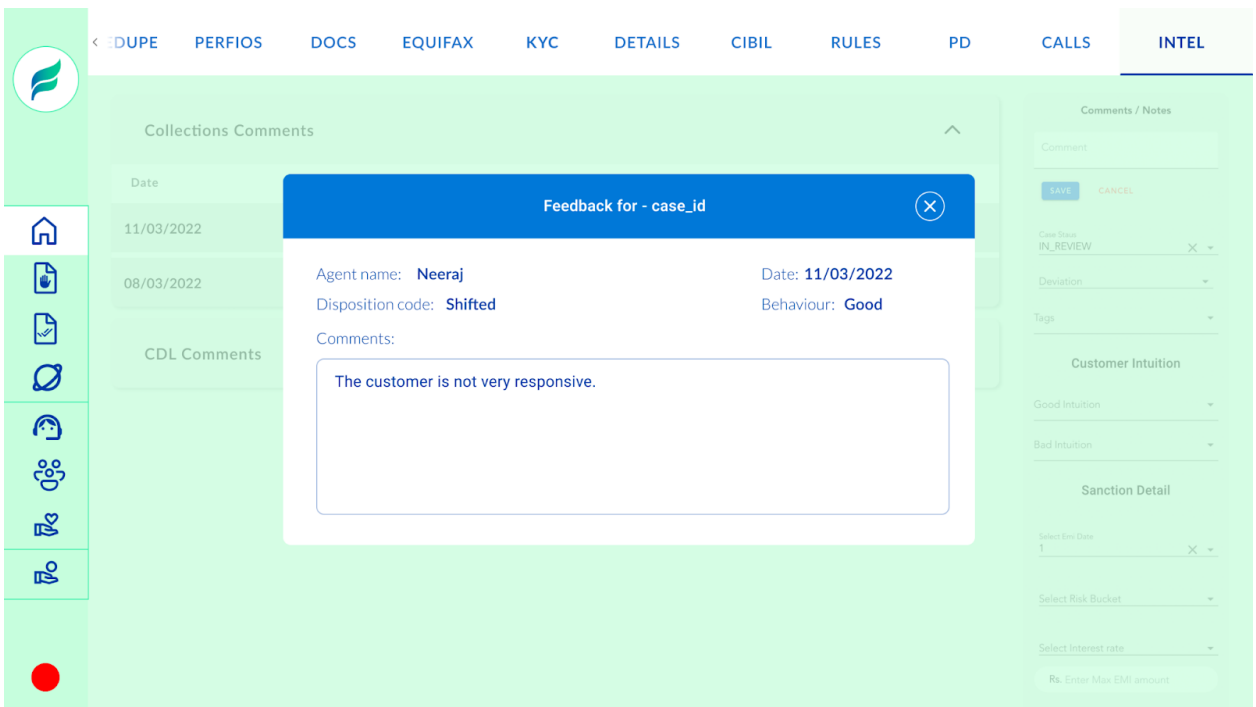


Figure 19: collection dialog box

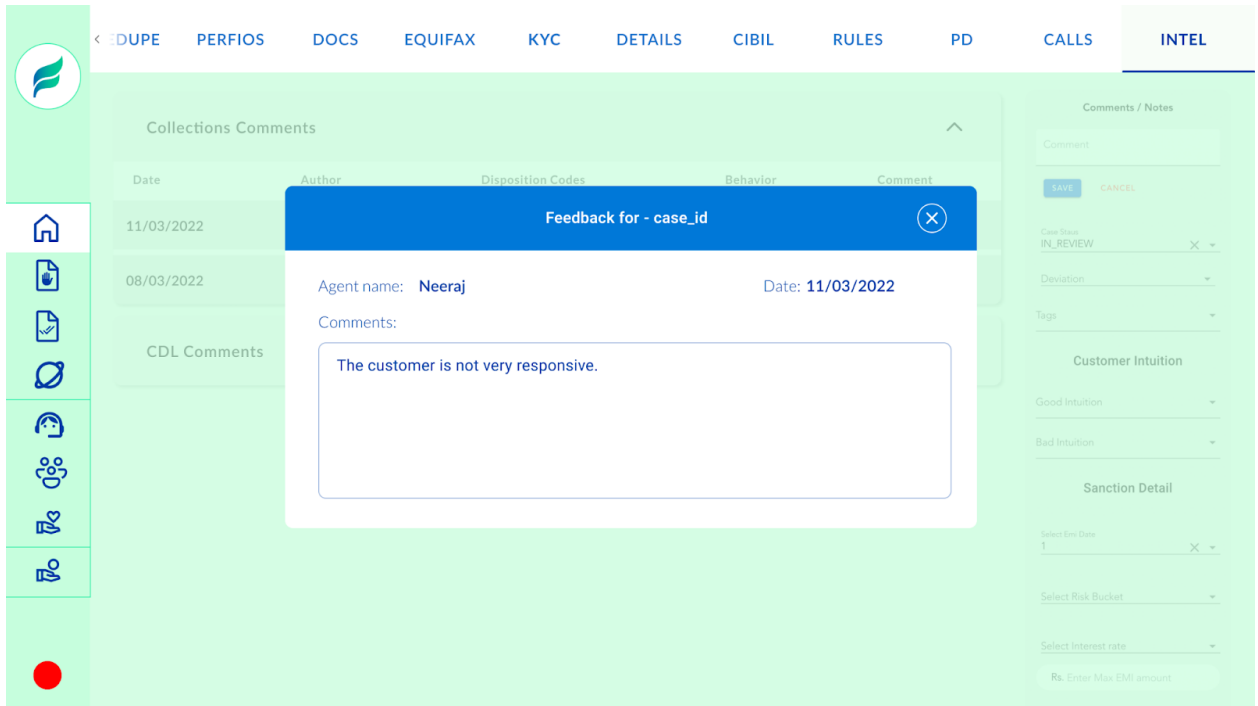


Figure 20 - cdl dialog box

4.5 Code

The basic code of background.js is as follows.

4.4.1 Cdl Dashboard

```
src > views > collections > Collections.vue {} "Collections.vue" > template
1 <template>
2 <v-container v-if="showLoans">
3 <v-row align="center" depressed justify="end">
4 <v-btn color="primary" @click="$router.push({name: 'ExportDetails'})">
5   Export Loans Details
6 </v-btn>
7 </v-row>
8 <div>
9 <p class="text-sm-left">
10   Search Applications
11 </p>
12 </div>
13 <v-row>
14 <v-col cols="6" md="4">
15 <v-list-item>
16 <v-select dense v-model="searchType" class="ml-2 mt-8" label="Search By" :items="searchTypeChoices" ></v-select>
17 </v-list-item>
18 </v-col>
19
20 <v-col cols="6" md="4">
21 <v-card-text>
22 <v-text-field v-model="searchText" placeholder="Type Here ...." ></v-text-field>
23 </v-card-text>
24 </v-col>
25 <v-col cols="6" md="4">
26 <v-card-text>
27 <v-btn v-if="searchText !== '' & searchType !== ''" @click="searchLoans">Submit</v-btn>
28 </v-card-text>
29 </v-col>
30 </v-row>
31 <div>
32 <p class="text-md-left">
33   Choose Filters
34 </p>
35 </div>
36 <v-row>
37 <v-col cols="6" md="4">
38 <v-list-item>
39 <v-select dense class="ml-2 mt-8" v-model="year" :items="yearList" label="Select Year" :clearable="true" v-on:change="getMonthList()" ></v-select>
40 </v-list-item>
41 </v-col>
42 <v-col cols="6" md="4">
43 <v-list-item>
44 <v-select dense class="ml-2 mt-8" v-model="month" :items="monthList" label="Select Month" :clearable="true" ></v-select>
45 </v-list-item>
46 </v-col>
```

feature/kyc-page | Jira: Rana Vijay Singh: > No active issue | Bitbucket: Rana Vijay Singh: | 0 0 0 | ranavijaysingh | Live Share


```

98 <script>
99 import axios from '../utils/axios'
100 import { mapActions, mapState } from 'vuex'
101 import LoadingProgress from '@components/loader.vue'
102 import Alert from '@components/Alert.vue'
103 import { formatString, daysBetween } from '../utils/utilsHelpers'
104
105 export default {
106   name: 'Collections',
107   components: {
108     Alert,
109     LoadingProgress
110   },
111   data () { ...
112 },
113   mounted () {
114     this.getYearList()
115     this.fetchLoans()
116     this.getPerfiosInstitutionsList()
117     this.pagination.page = this.numPage
118   },
119   computed: {
120     ...mapState({
121       perfiosInstitutionsList: state => state.customerSupport.perfiosInstituionList,
122       yearList: state => state.customerSupport.yearList,
123       monthList: state => state.customerSupport.monthList,
124       numPage: state => state.customerSupport.page
125     })),
126     headers () {
127       return this.fields.map(field => {
128         return {
129           text: formatString(field.key),
130           sortable: field.sortable,
131           value: field.key,
132           align: 'center'
133         }
134       })
135     },
136     pages () {
137       if (this.pagination.rowsPerPage == null || this.pagination.totalItems == null) {
138         return 0
139       }
140       return Math.ceil(
141         this.pagination.totalItems / this.pagination.rowsPerPage
142       )
143     }
144   }
145 }
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161

```

4.4.2 Collection Dashboard

```

<template>
  <v-container style="margin: 2rem">
    <v-card style="width:50rem; background-color: #f6f4f4">
      <v-simple-table class="align-table">
        <template v-slot:default>
          <tbody>
            <tr v-for="detail in userDetails" :key="detail.key">
              <td class="data-iterator">
                <!-- <v-icon :color="detail.color">{{detail.icon}} </v-icon> <br/> -->
                <span class="mr-4">{{detail.key}}</span>
              </td>
              <td>
                {{ detail.value }}
              </td>
            </tr>
          </tbody>
        </template>
      </v-simple-table>
    </v-card>
  </v-container>
</v-data-table
  :headers="headers"
  :items="loans" hide-default-footer

```

```

    style="background-color: #f6f4f4; margin-top:1rem">
    <template v-slot:top>
      <div style="height: 2rem; background-color: #161e44"></div>
    </template>
    <template v-slot:body="{ items, headers }">
      <tbody style="background-color: #f6f4f4">
        <tr v-for="item in items" :key="item.customerId" @click="navigate(item)">
          <td v-for="header in headers" :key="header.value"> {{ item[header.value] }}</td>
        </tr>
      </tbody>
    </template>
  </v-data-table>
</v-container>
</template>

<script>
import { mapState } from 'vuex'
import { formatString } from '@/utils/utilsHelpers'

export default {
  name: 'CustomerLoans',
  data () {
    return {
      fields: [],
      loans: [],
      userDetails: []
    }
  },
  mounted () {
    this.parseData()
  },
  methods: {
    parseData () {
      this.fields = [
        { key: 'loanId', sortable: false },
        { key: 'applicationStatus', sortable: false },
        { key: 'startedOn', sortable: true },
        // { key: 'approvedOn', sortable: true },
        // { key: 'amountApproved', sortable: false },
        // { key: 'tenure', sortable: false },
        { key: 'amountApplied', sortable: false },
        { key: 'approvalStatus', sortable: false }
        // { key: 'disbursalStatus', sortable: false },
      ]
      for (const loan of this.customerLoans) {
        // loan.startedOn = getDate(loan.startedOn)

```

```
computed: {
  ...mapState({
    customerLoans: state => state.customerSupport.loanSearchResults
  }),
  headers () {
    return this.fields.map(field => {
      return {
        text: formatString(field.key),
        sortable: field.sortable,
        value: field.key,
        align: 'center'
      }
    })
  }
}
}
</script>

<style scoped>
.align-table {
  background-color: #f6f4f4;
}
</style>
```

4.4.4 Intel Page

```
<template>
  <div id="app">
    <v-app id="inspire">
      <v-expansion-panels>
        <v-expansion-panel style="background-color: #f6f4f4">
          <v-expansion-panel-header>
            CDL Comments
          </v-expansion-panel-header>
          <v-expansion-panel-content>
            <v-simple-table>
              <template v-slot:default>
                <thead>
                  <tr>
                    <th class="text-left tableBorderRemove">
                      Date
                    </th>
                    <th class="text-left tableBorderRemove">
                      Author
                    </th>
                    <th class="text-left tableBorderRemove">
                      Comment
                    </th>
                  </tr>
                </thead>
              </template>
            </v-simple-table>
          </v-expansion-panel-content>
        </v-expansion-panel>
      </v-expansion-panels>
    </v-app>
  </div>
</template>
```

```

</thead>
<tbody style="background-color: #f6f4f4">
  <tr
    v-for="feedback in feedbackHistory"
    :key="feedback.id"
    @click="showReadCdICommentDialog(feedback.id)"
  >
    <td class="tableBorderRemove">{{ feedback.cdIDate }}</td>
    <td class="tableBorderRemove">{{ feedback.cdIAuthor }}</td>
    <td class="tableBorderRemove" v-if="feedback.feedbackComment !== null && feedback.feedbackComment.length > 20">{{ fe
    <td class="tableBorderRemove" v-else>{{ feedback.feedbackComment}}</td>
  </tr>
</tbody>
</template>
</v-simple-table>
<template>
<div justify="center">
  <v-dialog
    v-model="showCdICommentDialog"
    persistent
    max-width="600px"
  >
    <v-card>
      <v-toolbar
        dark
        color="primary"
        class="fixed-bar"
      >

```

```

<v-toolbar-title>
  <span class="text-h5">Feedback For Case - {{loanId}}</span>
</v-toolbar-title>
<v-spacer>
  <v-btn
    style="float:right"
    icon
    dark
    @click="showCdlCommentDialog = false"
  >
    <v-icon>mdi-close</v-icon>
  </v-btn>
</v-spacer>
</v-toolbar>
<v-card-text>
  <v-container>
    <span style="float:left">Name: {{cdlAuthor}}</span>
    <span>Date: {{cdlDate}}</span>
    <v-row class="text-left">
      <v-col cols="12">
        <v-textarea
          outlined
          readonly
          aria-placeholder="the customer is..."
          name="input-7-1"
          :value="cdlFeedback"
        >

```

```

        <v-textarea
          outlined
          readonly
          aria-placeholder="the customer is..."
          name="input-7-1"
          :value="cdlFeedback"
        >
      </v-textarea>
    </v-col>
  </v-row>
</v-container>
</v-card-text>
</v-card>
</v-dialog>
</div>
</template>
  <!--Content End-->
</v-expansion-panel-content>
</v-expansion-panel>
<v-expansion-panel style="background-color: #f6f4f4">
  <v-expansion-panel-header>
    Collections Comments
  </v-expansion-panel-header>
  <v-expansion-panel-content>
    <!--Content-->
    <v-simple-table>
      <template v-slot:default>
        <thead>

```

```

<th class="text-left tableBorderRemove">
  Author
</th>
<th class="text-left tableBorderRemove">
  Disposition Codes
</th>
<th class="text-left tableBorderRemove">
  Behaviour
</th>
<th class="text-left tableBorderRemove">
  Comments
</th>
</tr>
</thead>
<tbody style="background-color: #f6f4f4">
<tr>
  <tr>
    v-for="feedbackCollection in feedbackCollectionHistory"
    :key="feedbackCollection.id"
    @click="showReadCollectionCommentDialog(feedbackCollection.id)"
  >
    <td class="tableBorderRemove">{{ feedbackCollection.date }}</td>
    <td class="tableBorderRemove">{{ feedbackCollection.collectionAuthor }}</td>
    <td class="tableBorderRemove">{{ feedbackCollection.collectionObservation }}</td>
    <td class="tableBorderRemove">{{ feedbackCollection.customerBehaviour }}</td>
    <td class="tableBorderRemove" v-if="feedbackCollection.collectionFeedback.length > 20">{{ feedbackCollection.collect
    <td class="tableBorderRemove" v-else>{{ feedbackCollection.collectionFeedback}}</td>

```



```

import { mapState } from 'vuex'
// import axios from '@utils/axios.js'

export default {
  data () {
    return {
      feedbackHistory: [],
      feedbackCollectionHistory: [],
      showCdlCommentDialog: false,
      showCollectionCommentDialog: false,
      cdlAuthor: '',
      cdlFeedback: '',
      cdlDate: '',
      collectionDate: '',
      collectionObservation: '',
      customerBehaviour: '',
      collectionFeedback: '',
      collectionAuthor: '',
      loanId: '',
      feedbackCollections: [],
      feedbackCustomerSupport: []
    }
  },
  mounted () {
    this.getData()
  },
  computed: {
    ...mapState({

```

```

    ...mapState({
      customer: state => state.credit.customer,
      feedbacks: state => state.credit.feedbacks
    })
  },
  methods: {
    async getData () {
      this.loanId = this.feedbacks[0].loanId
      this.feedbackCollections = this.feedbacks[0].collectionsFeedbacks
      this.feedbackCustomerSupport = this.feedbacks[0].csFeedback
      this.showCollectionFeedbacks()
      this.showCdlFeedbacks()
    },
    commentSort (array) {
      array.sort(function (a, b) {
        return new Date(b.createdOn) - new Date(a.createdOn)
      })
    },
    showReadCdlCommentDialog (index) {
      this.showCdlCommentDialog = true
      this.cdlAuthor = this.feedbackHistory[index - 1].cdlAuthor
      this.cdlFeedback = this.feedbackHistory[index - 1].feedbackComment
      this.cdlDate = this.feedbackHistory[index - 1].cdlDate
    },
    showCdlFeedbacks () {
      this.feedbackHistory = []
      let i = 1

```

```

      this.collectionDate = this.feedbackCollectionHistory[index - 1].date
      this.collectionObservation = this.feedbackCollectionHistory[index - 1].collectionObservation
      this.customerBehaviour = this.feedbackCollectionHistory[index - 1].customerBehaviour
      this.collectionFeedback = this.feedbackCollectionHistory[index - 1].collectionFeedback
      this.collectionAuthor = this.feedbackCollectionHistory[index - 1].collectionAuthor
    },
    showCollectionFeedbacks () {
      this.feedbackCollectionHistory = []
      let i = 1
      this.commentSort(this.feedbackCollections)
      if (this.feedbackCollections === null || this.feedbackCollections === undefined) { return }
      for (const feedback of this.feedbackCollections) {
        this.feedbackCollectionHistory.push({
          id: i,
          date: new Date(feedback.createdOn).toLocaleDateString(),
          collectionAuthor: feedback.userName,
          collectionFeedback: feedback.collectionFeedback,
          collectionObservation: feedback.collectionObservation,
          userIdString: feedback.userId,
          customerBehaviour: feedback.customerBehaviour
        })
        i++
      }
    }
  }
}
</script>

```

4.5 Results

After building the cdl dashboard with edit and view comment feature and the collection dashboard with view and edit review functionality. As a result Correct data was displayed on the intel page of the credit dashboard, using v-dialog box in a view only mode.

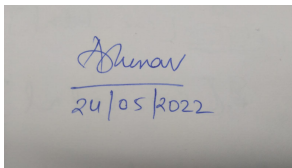
Chapter 05: CONCLUSION

5.1 Conclusion

In conclusion, this internship has become a wonderful and rewarding experience. I can say that my stay with Florence Capital - A Fintech Company was quite beneficial to me. Needless to say, the technical components of my work aren't perfect and, given enough time, could be improved. As someone who had no prior experience with MERN stack development, I think the effort I spent learning and understanding about was well spent, as it helped to construct a fully effective web service. Two of the most significant lessons I've learned are time management and self-motivation.

5.2 Mentor's Review

He started training on ReactJS and Vue.js and within a month he started working on a live project. He is a very talented, quick learner, and open to new skills and approaches.

A rectangular box containing a handwritten signature in blue ink that reads "Abhinav" and the date "24/05/2022" written below it.

Abhinav Kumar
Chief Technical Officer

REFERENCES

1. <https://www.w3schools.com/html/default.asp>
2. <https://www.w3schools.com/css/default.asp>
3. <https://www.w3schools.com/js/default.asp>
4. <https://www.w3schools.com/nodejs/>
5. <https://www.youtube.com/watch?v=SccSCuHhOw0>
6. <https://www.npmjs.com/package/express>
7. <https://www.mongodb.com/docs/>
8. <https://www.w3schools.com/REACT/DEFAULT.ASP>
9. <https://stackoverflow.com/>
10. <https://github.com/>
11. <https://www.youtube.com/watch?v=0KEpWHtG10M&list=PL4cUxeGkcC9gixLvV4VEkZ6H6H4yWuS58>
12. [\(206\) Vue JS 3 Tutorial for Beginners #1 - Introduction - YouTube](#)
13. [Bitbucket | Git solution for teams using Jira](#)

PLAGIARISM REPORT

RANA VIJAY SINGH 181394

ORIGINALITY REPORT

17%	13%	1%	13%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	bitbucket.org Internet Source	4%
2	www.coursehero.com Internet Source	2%
3	Submitted to Softwarica College Of IT & E-Commerce Student Paper	2%
4	Submitted to Universidad Anahuac México Sur Student Paper	2%
5	www.javatpoint.com Internet Source	2%
6	www.atlassian.com Internet Source	1%
7	Submitted to The University of Wolverhampton Student Paper	1%
8	Submitted to Universiti Teknologi MARA Student Paper	1%

9	Submitted to Lovely Professional University Student Paper	1%
10	Submitted to Management Development Institute Of Singapore Student Paper	1%
11	Submitted to Teaching and Learning with Technology Student Paper	1%
12	Submitted to Carnegie Mellon University Student Paper	<1%
13	sportold.ubbcluj.ro Internet Source	<1%

Exclude quotes On
Exclude bibliography On

Exclude matches < 14 words