B. TECH. PROJECT REPORT

On

Software Development Internship at Big Binary Solutions PVT. LTD.

BY

Nityash Agrawal

DISCIPLINE OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

Dec 2019

Software Development Internship at Big Binary Solutions PVT. LTD.

A PROJECT REPORT

Submitted in partial fulfilment of the requirements for the award of the degrees

of
BACHELOR OF TECHNOLOGY
in

ELECTRICAL ENGINEERING

Submitted by:
Nityash Agrawal

Guided by:
Dr Saptarshi Ghosh | Assistant Professor
Discipline of Electrical Engineering, IIT Indore

INDIAN INSTITUTE OF TECHNOLOGY INDORE
Dec 2019

CANDIDATE'S DECLARATION

I hereby declare that the project entitled **"Software Development Internship at Big Binary Solutions PVT. LTD."** submitted in partial fulfilment for the award of the degree of Bachelor of Technology in 'Electrical Engineering' completed under the supervision of **Dr Saptarshi Ghosh, Assistant Professor, Discipline of Electrical Engineering,** IIT Indore is an authentic work.

Further, I declare that I have not submitted this work for the award of any other degree elsewhere.

Nityash Agrawal
B.Tech. IV Year
Discipline of Electrical Engineering
IIT Indore

CERTIFICATE by BTP Guide(s)

It is certified that the above statement made by the students is correct to the best of my/our knowledge.

Dr Saptarshi Ghosh
Assistant Professor
Discipline of Electrical Engineering
IIT Indore

Preface

This report on “Software Development Internship at Big Binary Solutions PVT. LTD." is prepared under the guidance of *Dr Saptarshi Ghosh*.

In this project report, I will describe my experiences throughout the period of the internship. The report contains a description of the internship company and the projects, tasks and activities that I underwent throughout the internship. I have tried to discover the relationship between practical and theoretical type of knowledge and to bridge the gap between practical necessities and theoretical assumptions. With these objectives, I have made all the necessary investigations and possible efforts to submit this report in an enlightened form. I have tried to the best of my knowledge and abilities to explain the content in a lucid manner and to eliminate errors from the report. I have also incorporated various figures to make it more illustrative.

Nityash Agrawal
B.Tech. IV Year
Discipline of Electrical Engineering
IIT Indore

Acknowledgements

I would like to express my special thanks of gratitude to my B.T.P guide *Dr Saptarshi Ghosh* for his kind support and valuable guidance. I am grateful to him for giving me this great opportunity to embark on this wonderful project and for providing me with all the facilities that were required.

I would also like to thank Mr Neeraj Singh for giving the opportunity to work in Big Binary Solutions Pvt. Ltd. who helped me to grab the rare opportunities to learn the real-world knowledge. I am also thankful to all the members of Big Binary Team for providing several figures, services, documents, data and papers as well as sharing their valuable experience with me and teaching me different methods to enhance my knowledge.

Without their support, this report would not have been possible.

Nityash Agrawal
B.Tech. IV Year
Discipline of Electrical Engineering
IIT Indore

Abstract

During the internship, I have worked on two of the company's internal mobile application projects, namely AceInvoice and Zindi. I also worked on the website for the AceInvoice App. AceInvoice is a time tracking and invoicing tool, through which a user can keep track of the amount of time he has worked and can also send it to the client in the form of an invoice which can then be paid for the client. The invoices can easily be edited and the application will keep track of all of the user's previous invoices so that the user can get his older invoices at any time. Options are provided to view the time entries in a daily, weekly and monthly view. Also, a user can be part of multiple organisations having multiple projects whose time entries can be accessed. There are various tabs in the application namely, Timesheet, Invoices, Projects, Team, Clients, More and Profile. There is also a Notifications screen where the notifications are categorised as read and unread. There are various levels of control over the application such as regular user and admin privilege. A regular user can only view his entries while a user with admin privilege can read, edit or delete others' entries as well.

I was also assigned to build a test application on any idea of my choice before starting to work on the actual application. Hence, I created an app that utilizes multiple free APIs and enables the user to get various useful information such as the user's IP address, location, the location corresponding to a manually entered IP address and current time at any location.

List of Contents

List of Figures

Chapter 1

Introduction

AceInvoice is a management tool which basically involves time-tracking and invoicing. It already has a fully functional website up and running for a long time and it also has a desktop application which is used by the company employees on a day-to-day basis. The company decided to expand its reach even more and decided to build a smartphone application for AceInvoice in 2018. The AceInvoice smartphone application was in the nascent stages of development when I joined the company and by the completion of this internship, the app is now available on the Google Play Store and iOS App Store with all the features implemented.

AceInvoice Logo

The software is developed by BigBinary Solutions Pvt. Ltd. and this software is made use of extensively company-wide.

1.1 About BigBinary

BigBinary is a software consultancy firm that builds custom mobile and web apps. BigBinary, based in USA is a group of software developers which builds mobile applications using GraphQL and React Native and web applications mainly using ReactJS and Ruby on

Rails for clients. BigBinary team members speak regularly at multiple conferences around the globe ranging from Golden Gate Ruby Conf, San Francisco to Ruby Kaigi, Japan. They not only write high-quality codes but also author technical blogs on subjects ranging from the inner workings of Kubernetes to the internals of Ruby.

Fig 1.1: BigBinary Company Logo

ImageRef: https://www.bigbinary.com/

With clients like UL, Cloud Logistics, DataStax, Gumroad, AAPC and several others BigBinary has been mentioned in the Inc. 5000 list as one of the fastest-growing companies in USA. Blogs written by BigBinary are featured regularly in Ruby Inside, Ruby Weekly and other publications.

Fig 1.2: BigBinary is listed in Inc. 5000 list for second year in a row

ImageRef: https://www.bigbinary.com/

1.2 Problem Statement

BigBinary is a remote company which means that its employees don't work from a single place or office. Rather, the employees work at their own homes and are scattered all around the world! This not only makes sure that the developers can work without any hassle but also gives them the flexibility in working hours.

But this also leads to a series of management problems:

- Managing the data regarding projects and also the associated clients.
- Requirement of a fool-proof and accurate system that can generate invoices which can be used for the purpose of the clients' billing.
- Keeping a track of the developers' daily progress regarding the work.
- Managing the expenses of the company may it be regarding an employee's ticket reimbursement or some expense related to infrastructure.

1.3 Internship Objective

This internship emphasized on learning and acquiring skills in the field of software development by delivering and developing a solution to the aforementioned organizational problem that is performance-efficient and user-friendly by making use of efficient language, latest frameworks and cutting-edge technologies during the development of the AceInvoice smartphone app.

Some of the key objectives of this internship were:

- Getting acquainted with the fundamentals of core development programming frameworks and languages such as React-Native, JavaScript, Ruby-on-Rails and ReactJS.
- Acquiring the skills to use Git command line interface and GitHub for version control.
- AceInvoice mobile app development for both iOS and Android platforms and deployment to the respective application stores.

- Problem-solving in the website of AceInvoice on both back-end and front-end.

The total internship duration can be effectively divided into two phases:

- **Learning Phase:** The internship's first month was dedicated to getting acquainted with the fundamentals of the code conventions, work-flow and the frameworks, especially Git. I also worked on the development of a React-Native application as practice for better learning of the basics.
- **Development Phase:** After acquiring the basic experience with React Native and the workflow, I was assigned to the AceInvoice app project where I worked on fixing several bugs and issues, working both on backend and UI/UX.

Chapter 2

Learning Phase

The initial month of the internship was dedicated to getting acquainted with the fundamentals of the frameworks, code conventions and the work-flow, especially Git. The tech stack for the AceInvoice app consists of:

- Node.js
- Git
- React-Native
- React-Redux
- GraphQL

I developed a React-Native app as a practice application to learn the basics better.

2.1 Node.js

Node.js is a runtime environment based on JavaScript which enables the execution of JavaScript code outside of a browser. It is cross-platform, open-source and allows the use of JavaScript for server-side scripting and to write command-line tools. It is capable of an asynchronous I/O and has an event-driven architecture.

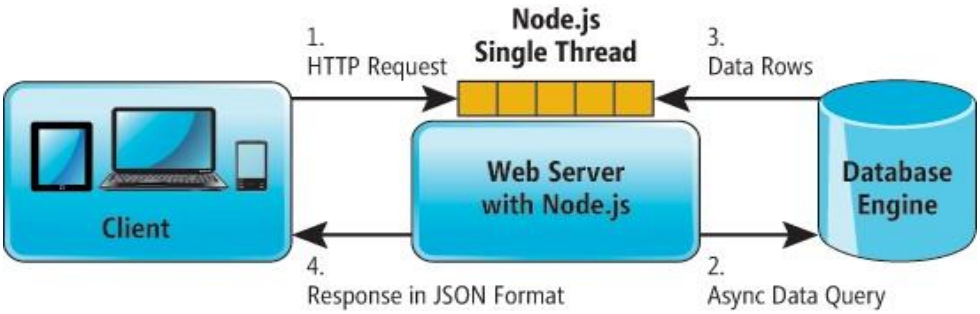

Fig 2.1: A schematic illustration of how Node.js works

ImageRef: https://vmokshagroup.com/wpcontent/uploads/2017/01/NodeArchitechure.jpg

It is mainly used to build network programs such as Web servers and network tools using a collection of modules handle various core functionalities along with JavaScript. Node.js focusses on event-driven programming to let developers build scalable servers by using call-backs to signal the completion of a task instead of threading.

Some important features of Node.js are:

- It connects the power of Unix network programming with the ease of a scripting language. It can support concurrent connections without incurring the cost of thread context switching.
- The execution of tasks in a parallel manner in Node.js is handled by a thread pool. Posts tasks to the shared task queue are called by the main thread function, which threads in the thread pool pull and executes.
- Inherently blocking system functions like file I/O run on their own threads in a blocking way, while Inherently non-blocking system functions such as networking translate to kernel-side non-blocking sockets.
- The main thread wakes up and executes the registered call-back when it is informed by a thread in the thread pool when it completes a task.

2.2 Git

Git is a distributed version control system. It is used for tracking changes in code during development. It offers strong support for non-linear development, distributed development, compatibility with existent systems and protocols, efficient handling of large projects, cryptographic authentication of history, toolkit-based design and pluggable merge strategies. The primary advantages of using Git are a feature-based workflow and frictionless context switching. The version control is done using a set of instructions called Git commands. Some popular commands are: *init, add, commit, status, merge, push, pull, rebase, branch, checkout, clone, stash, etc.*

2.2.1 GitHub

GitHub is a service that enables developers to store and manage their code using Git. The main advantage of using GitHub is that provides backup through GitHub servers rather than backing things on local machines.

- Issues are raised on a particular repository about a fault or a feature request. The developers then assign to the issue, work on it and make commits.
- Commits are the changes in code done to resolve the issue locally.
- The commit is then pushed to the server and a pull request is raised, which is then reviewed.
- Upon successful review, the pull request is merged thus changing the original file.

The major advantage in this process is that the entire history of pull requests is accessible and if at a later stage, some fault is encountered due to a previous pull request, it can be easily rolled-back.

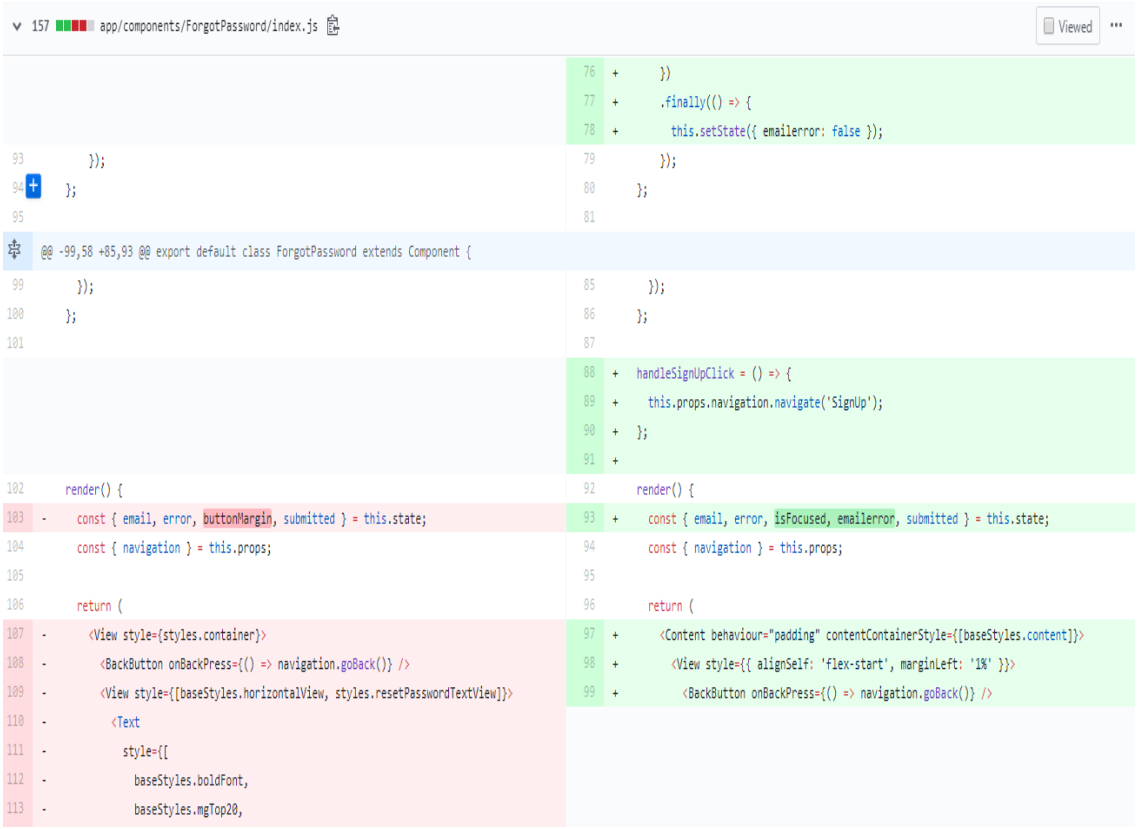

Fig 2.2: Git diff showing the code changes in a file in a pull request

2.3 React-Native

React Native is an open-source application framework written in JavaScript, created by Facebook in 2015, that enables developers to use React along with native platform capabilities. React Native enables developers to develop Android and iOS apps with a single cross-platform codebase.

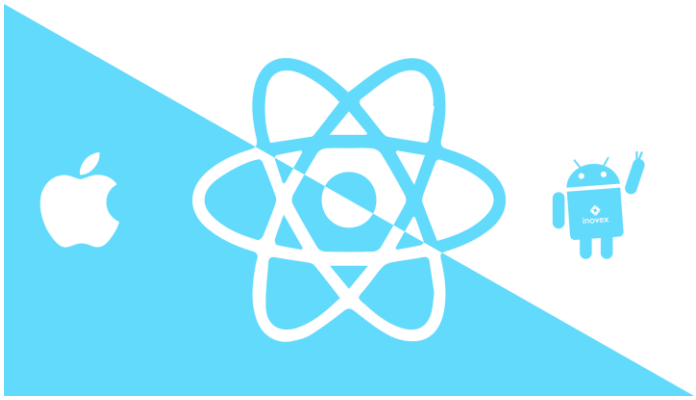

Fig 2.3: An illustration depicting the cross-platform ability of React-Native
ImageRef: https://www.inovex.de/blog/react-native-vs-native-development/#iLightbox[gallery12780]/0

With time, React Native has grown popular among mobile app developers as is evident from the graphical analysis by Google Trends. More and more of the companies are migrating their codebase to RN.

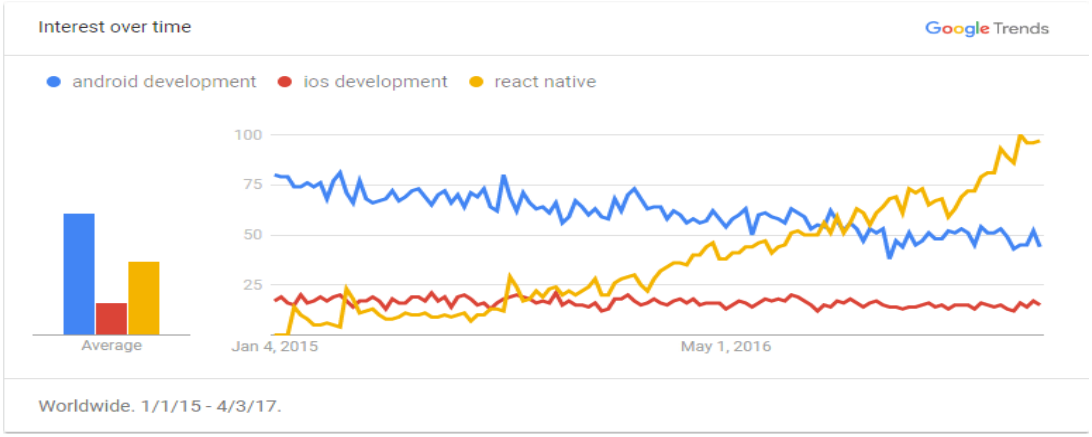

Fig 2.4: An illustration of increasing popularity of React-Native among developers
ImageRef: https://en.wikipedia.org/wiki/Document_Object_Model#/media/File:DOM-model.svg

2.3.1 Basic Code Structure

Every component or class we create in React Native has the following predefined properties:

- **State:** State variable is a JavaScript object which has dynamic properties which control the component rendering. This allows React Native to trigger re-rendering of only those components which need to be updated and hence makes React Native very fast and resource-friendly.
- **Lifecycle Methods:** Every class has some lifecycle methods which are triggered on events such as state update, component mount and unmount.
- **render() method:** This is the part which uses DOM (Document Object Model) paradigm to create different layouts and styles and handles the complete UI (User Interface). This function takes the JSX tags which produce React elements which are then converted to their respective native rendering code.

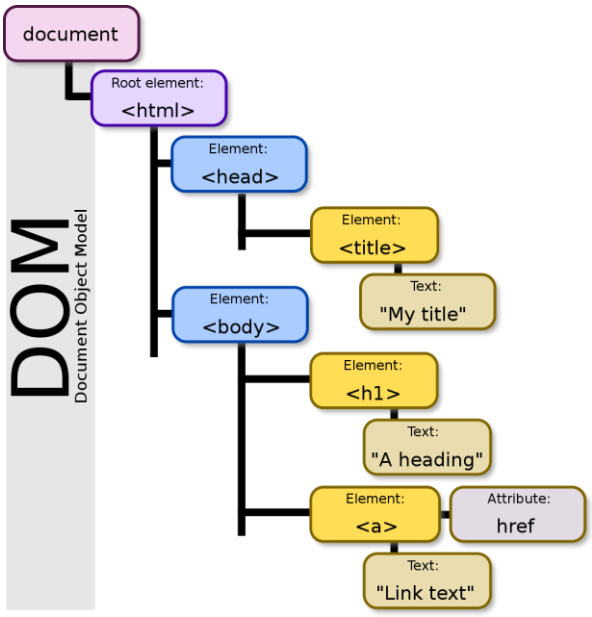

Fig 2.5: An illustration depicting the Document Object Model of React-Native

ImageRef: https://en.wikipedia.org/wiki/Document_Object_Model#/media/File:DOM-model.svg

Fig 2.6: A React Native class illustrating the aforementioned properties

2.4 React-Redux

React-Redux is a powerful framework used to centralise the state of the JavaScript apps and is used for real-time app-wide updates. It is helpful to make app-wide updates through a single state mutation and is incredibly easy to use and understand.

There are some pieces of information that an app fetches from the server and are needed app-wide. Redux helps provide this information anywhere throughout the app and the changes you make at one place are reflected throughout.

2.4.1 Architecture

Redux uses a middleware to connect the Redux store to the React component. This is achieved by using a *connect()* function from react-redux library. This whole process of data retention and centralisation is achieved using what are called *Store*, *Actions* and *Reducers.*

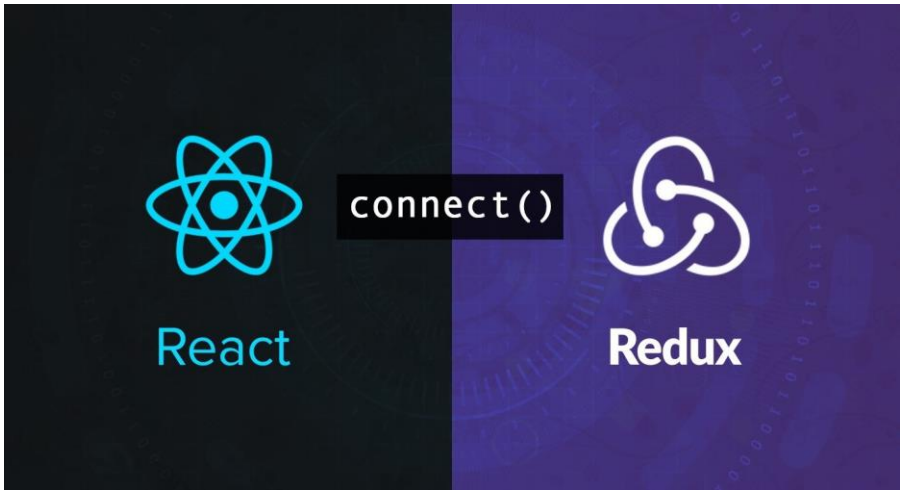

Fig 2.7: An illustration of how React and Redux are integrated using connect() function

ImageRef: https://blog.logrocket.com/react-redux-connect-when-and-how-to-use-it-f2a1edab2013/

- **Store:** Redux store is an object which contains the *immutable* state variables and the reducers which are responsible for modifying these variables.
- **Actions:** Redux uses an action to trigger a state change. Every action must be of the form:

```
{
        type: ACTION_TYPE,
        payload: ACTION_DATA
}
```

Whenever an action is triggered, it is dispatched to each and every reducer in the store. Hence, to identify the correct reducer, a *type* is used. The *payload* is the updated data that is to be stored in the state variable.

- **Reducers:** Reducers are JavaScript functions which are used to update the state variables. Each reducer must update only a single state variable.

The above process can be easily understood using a graphical flowchart and actual code snippets from AceInvoice:

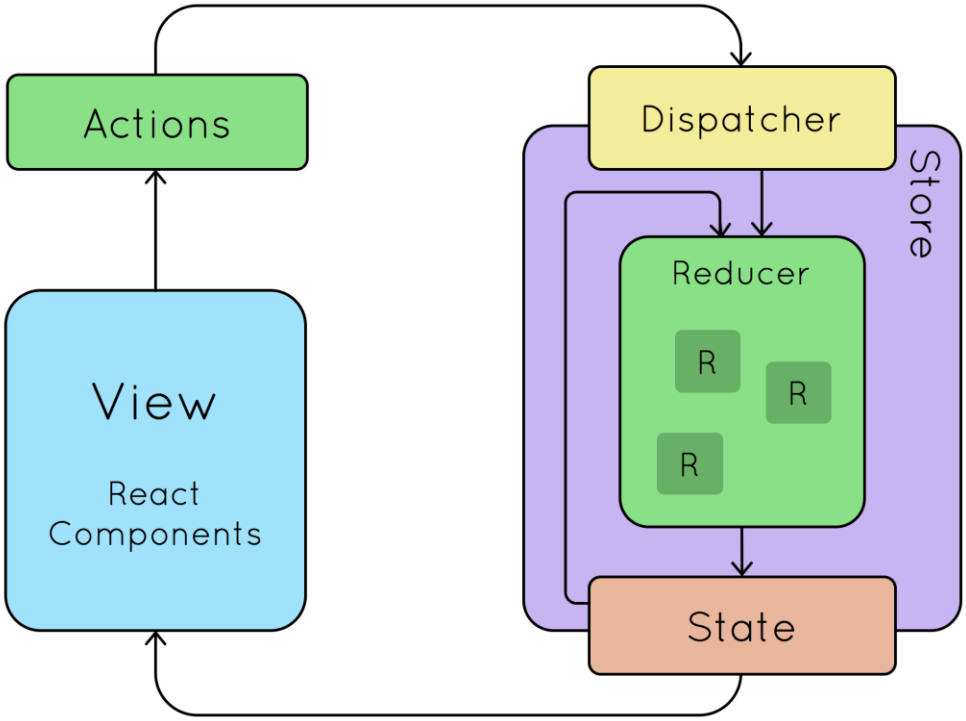

Fig 2.8: Redux data flow and how the react components are connected to the redux store.

ImageRef: https://www.esri.com/arcgis-blog/wp-content/uploads/2017/09/react-redux-overview.png

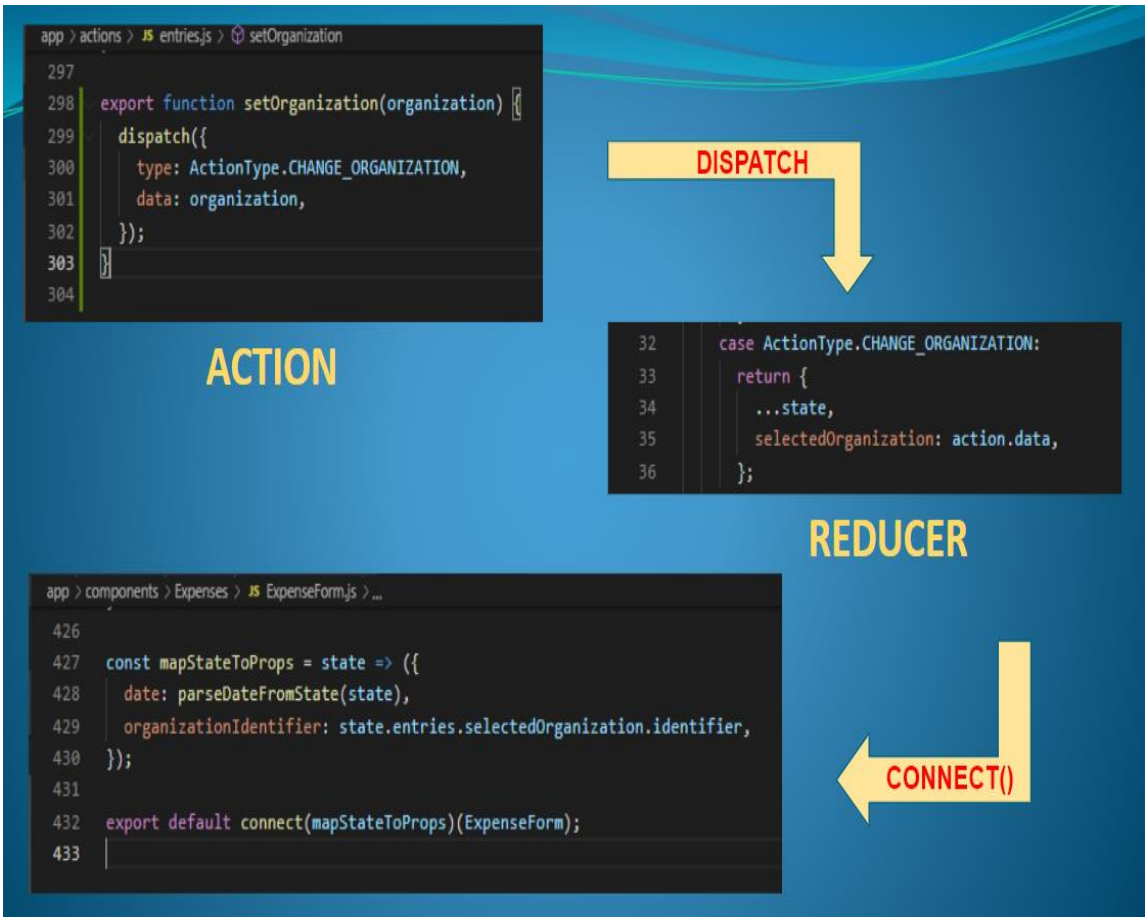

Fig 2.9: Code snippet illustrating the working of actions, reducers and connect function for the organization state variable.

2.5 GraphQL

GraphQL is an open-source data manipulation and query language for APIs, and a runtime for fulfilling queries with existing data. It gives clients the power to ask the server for specifically and exactly what they need. GraphQL APIs are not organized in terms of endpoints but in terms of types and fields. The full capabilities of data are accessed from a single endpoint. Types are used so as to ensure that Apps only ask for what's possible and clear and helpful errors are provided. Types can be used by Apps to avoid writing manual parsing code.

Some of its key features are:

- It has been compared and contrasted with REST and other web service architectures and provides an approach to developing web APIs.
- It consists of a type introspection, query language, type system, static validation and execution semantics.
- It supports subscribing to changes to data, writing (mutating) and reading.
- The structure of the data required is defined first and then the server returns the same structure of the data, thus preventing excessively large amounts of data from being returned.

Fig: 2.10: Code snippet illustrating a GraphQL query to retrieve only the mentioned fields from the server endpoint.

Chapter 3

Development Phase

After having acquired the fundamentals of the tech stack, I was assigned to the AceInvoice app and Zindi app projects where I solved numerous issues and fixed multiple bugs. GitHub was used as version control system for both of the projects. This chapter will cover the highlights of some of the major issues and bugs I worked on.

3.1 AceInvoice App

I worked primarily on the AceInvoice app for the majority of the duration of this internship. The app was in very early stages with several bugs and very few features when I started working on the project. Currently, it is a fully functional application available for use on the Google Play Store. I am ranked 3rd in order of commits made to the project across all the team working on it. The main issues which I worked in the AceInvoice app are explained in this section:

3.1.1 Settings Section

Settings section manages the settings of the organization and the user. I created this section completely from scratch. This section had mainly two screens:

- **Company Info screen:** This screen displays the information regarding the company such as the address. The user was provided with the option to edit this information. The information had to be fetched from the back-end database and displayed on the screen.

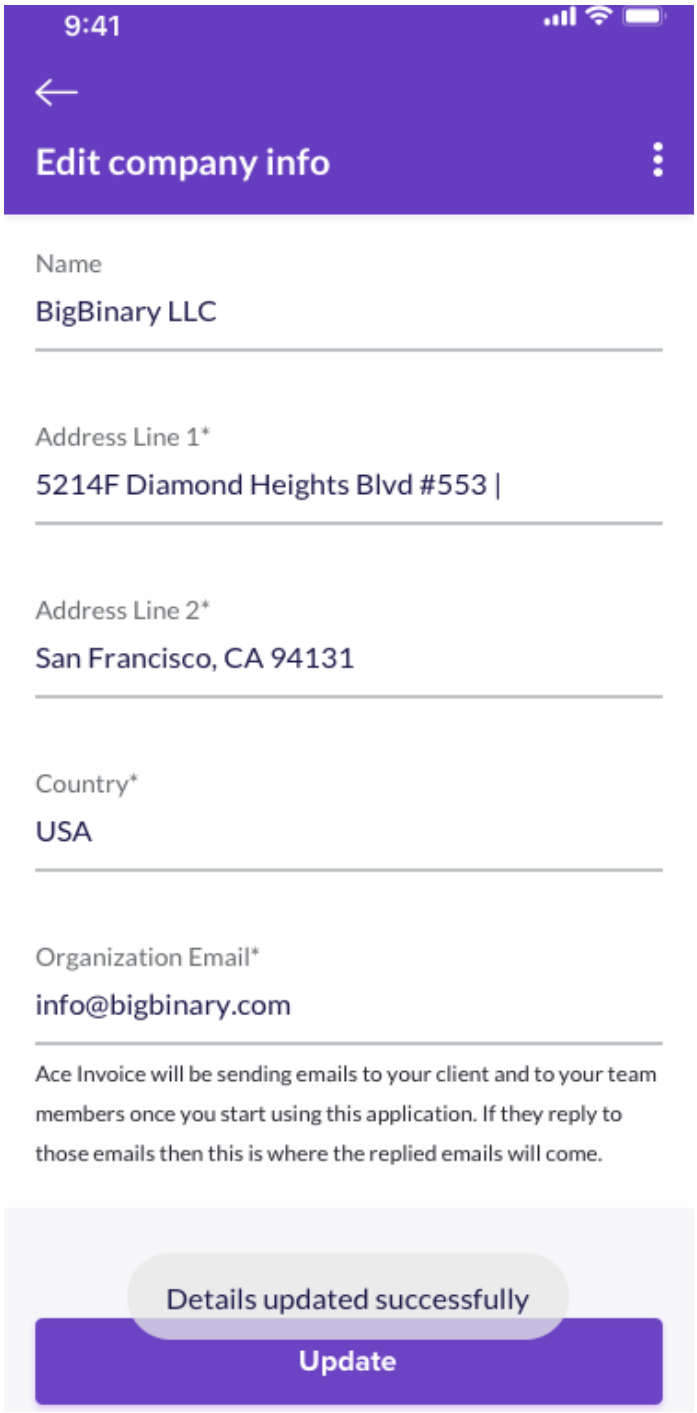

Fig 3.1: Company Info screen

- **Delete Organisation screen:** I provided the option to delete the organisation as well. This was a very sensitive issue and double confirmation was required from the user's side because once the organisation was deleted, there was no option of undoing the action and the organisation and all its related data would have been permanently removed from the database.

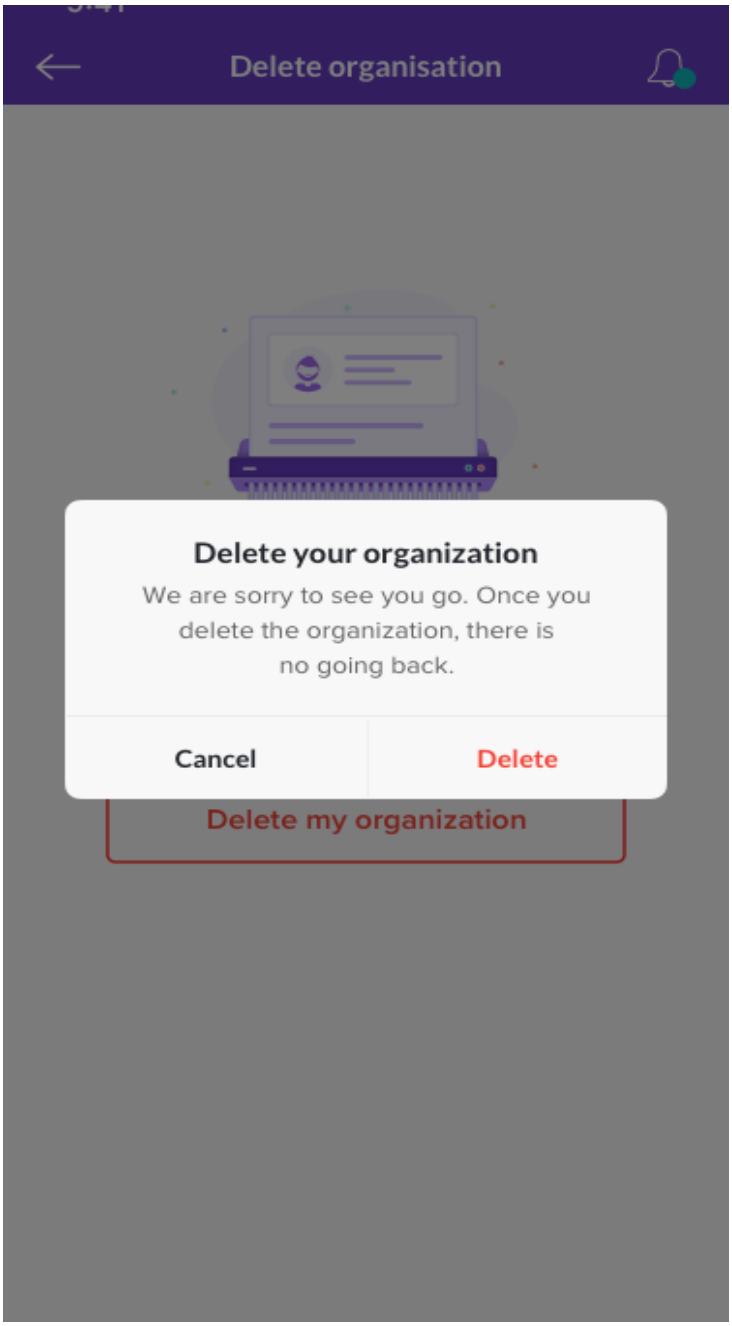

Fig 3.2: Delete Organisation screen

3.1.2 React-Native version upgrade

I was assigned to perform a major React-Native version upgrade on which the application was running and I successfully performed it. I had to modify a large part of the existing code as React- native had discontinued support for various libraries that were deprecated or dependencies that were replaced which the application was using. It was extremely critical because the application was experiencing frequently increasing crashes due to the then-current version being unsupported and the whole of the team was finding it difficult to work on. I updated the version from 0.58.6 to 0.60.4 .

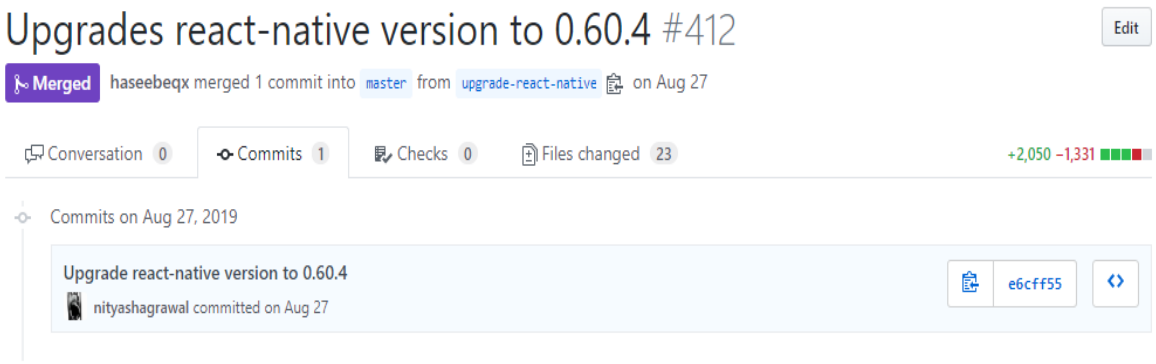

Fig 3.3: GitHub summary of the pull request with more than 2000 lines of code change in 23 files

3.1.3 Empty State designs

When I started working on the project, if there was no data under any tab such as projects or clients, a blank page used to load. I fixed it by removing the blank page across all the tabs by the respective empty state designs provided. This was a very interesting task as I had to know whether the page had any data or not before the page loaded and that was a very typical problem. I solved it by creating separate actions and reducers in the redux store of the application which are fired before the page gets loaded and fetch the data from the website using different URLs which were also created by me.

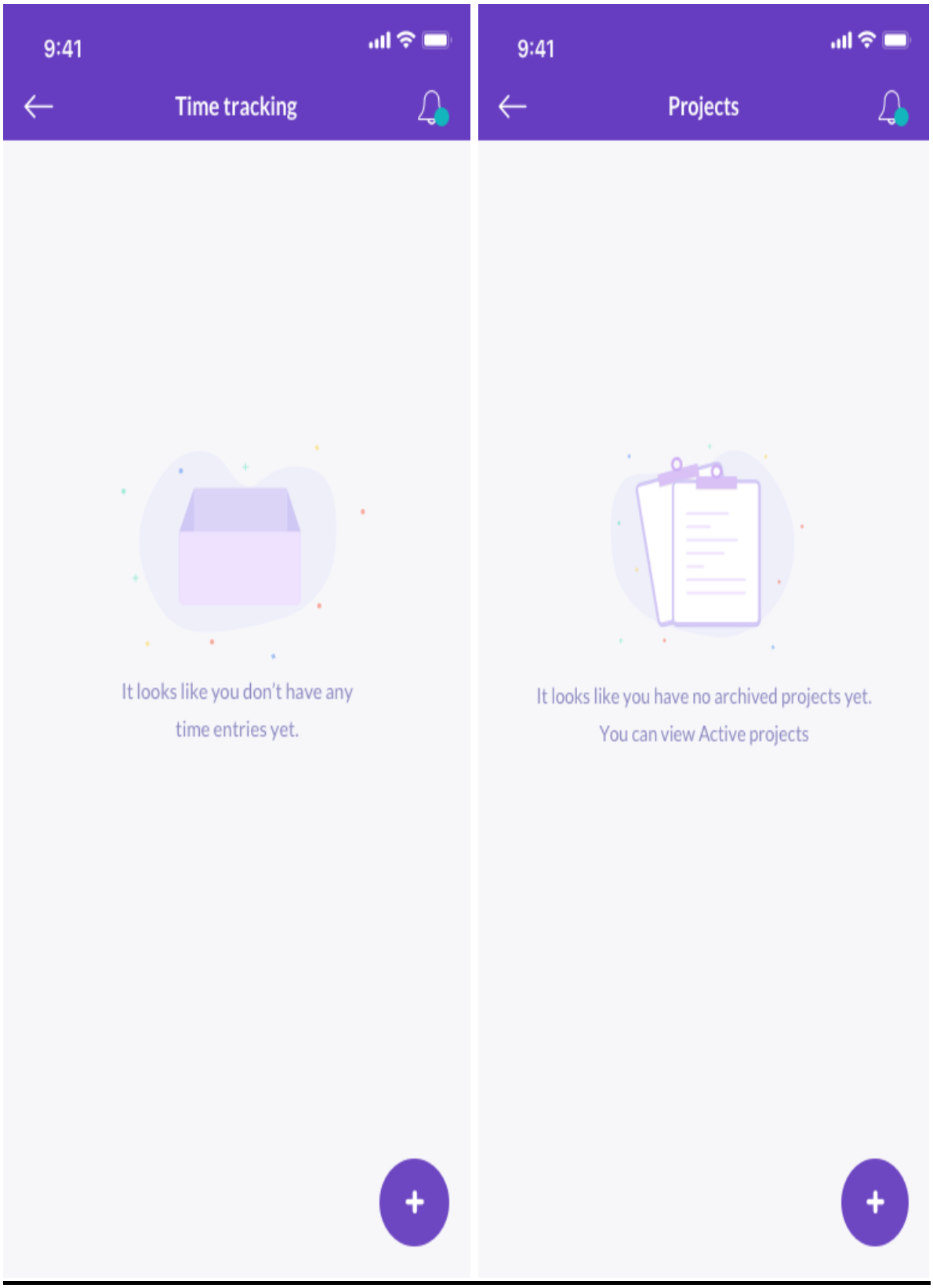

Fig 3.4: Empty state designs of timesheet and projects screen

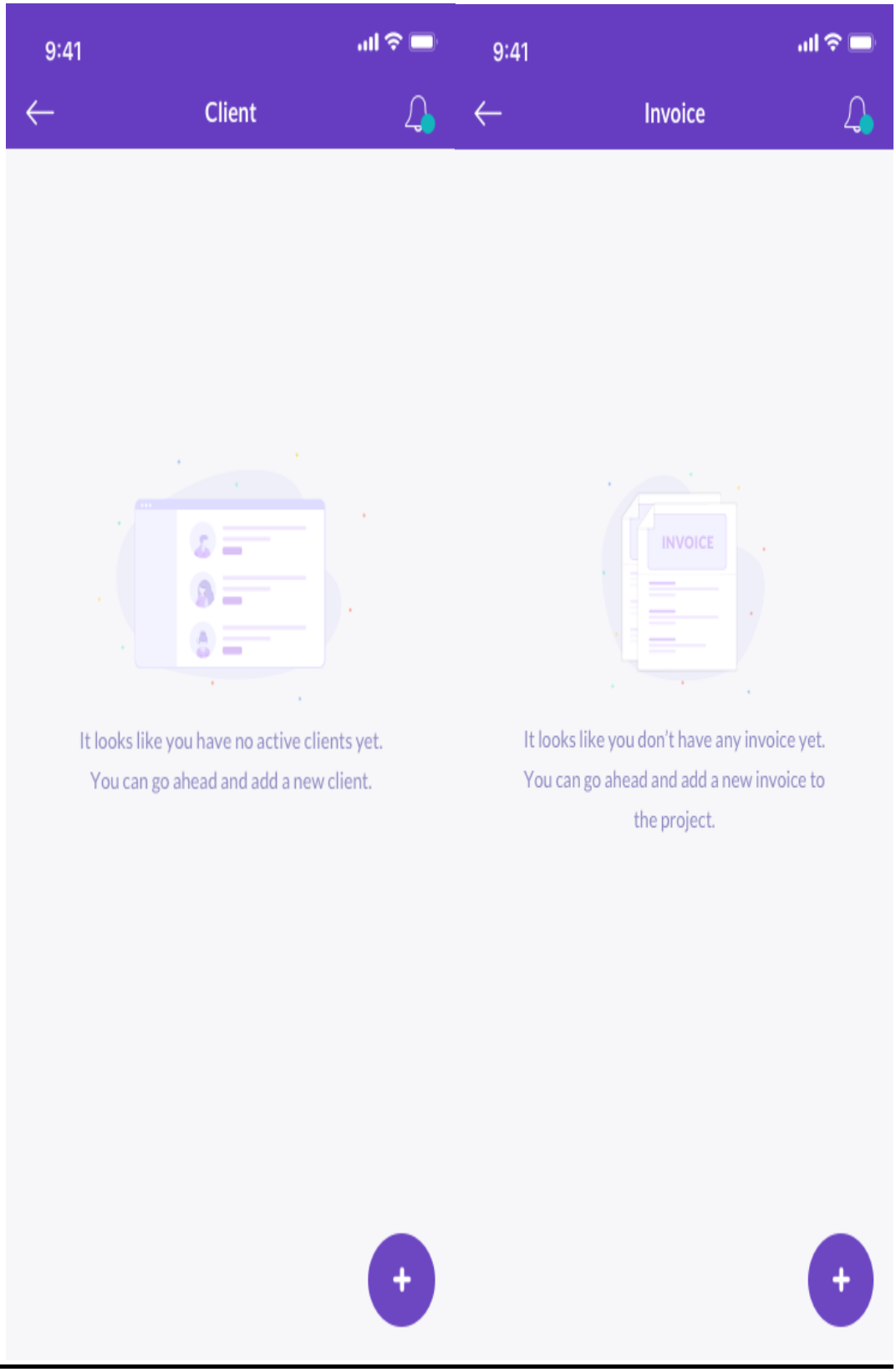

Fig 3.5: Empty state designs of clients and invoices screen

3.1.4 Revamping the existing notifications screen

The notifications screen handles all the notifications related to the user in the application. It was in a very critical state as well. The back button was not responding when pressed. The unread notification bell label was not working as expected. There was also an option to mark all the notifications as read as well. The API calls needed a lot of re-formatting as some of the properties were unique to each of the notification. So, the challenging task here was to write non-redundant and consistent code while minimising the API calls and efficiently re-using the information.

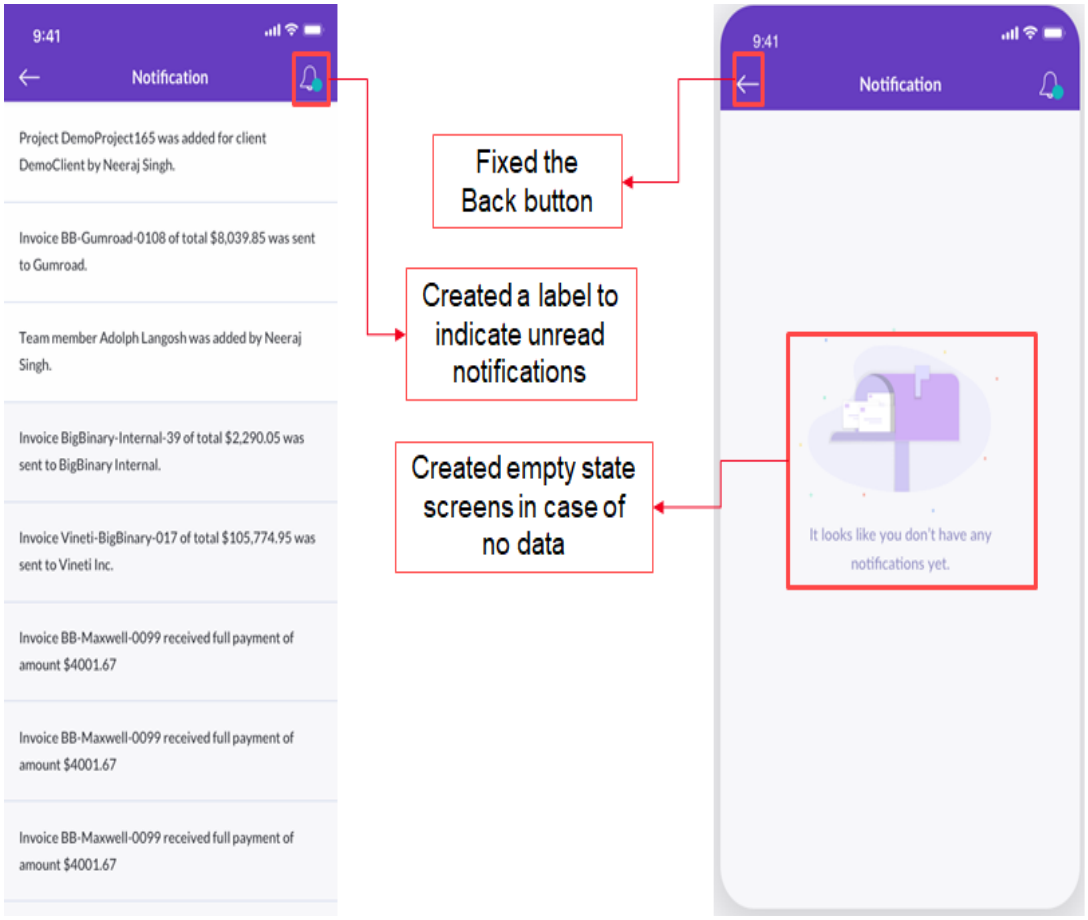

Fig 3.6: Notifications screen showing the various changes

Previously, the user had to manually click on a button to load more notifications after the initial number of notifications. I replaced that with an autoloading feature wherein, more notifications get automatically loaded whenever a user scrolls past the existing notifications.

I had to change the notification list from functional to class-based component and add various listeners and conditions to effectively fetch more notifications without duplication. This greatly improved the performance of the application as only a limited number of notifications were loading now as and when prompted by the user.

3.1.5 Updated Login screen UI

The login screen is the first screen that the user encounters in the application. It is the most important screen as this decides whether or not to provide the user with access to further screens. It is extremely critical from security point of view as well because the database is queried to know whether the user is an admin or not. Hence, this feature must be as user-friendly as possible so as to make the logging in easier for users, the foremost purpose of this app.

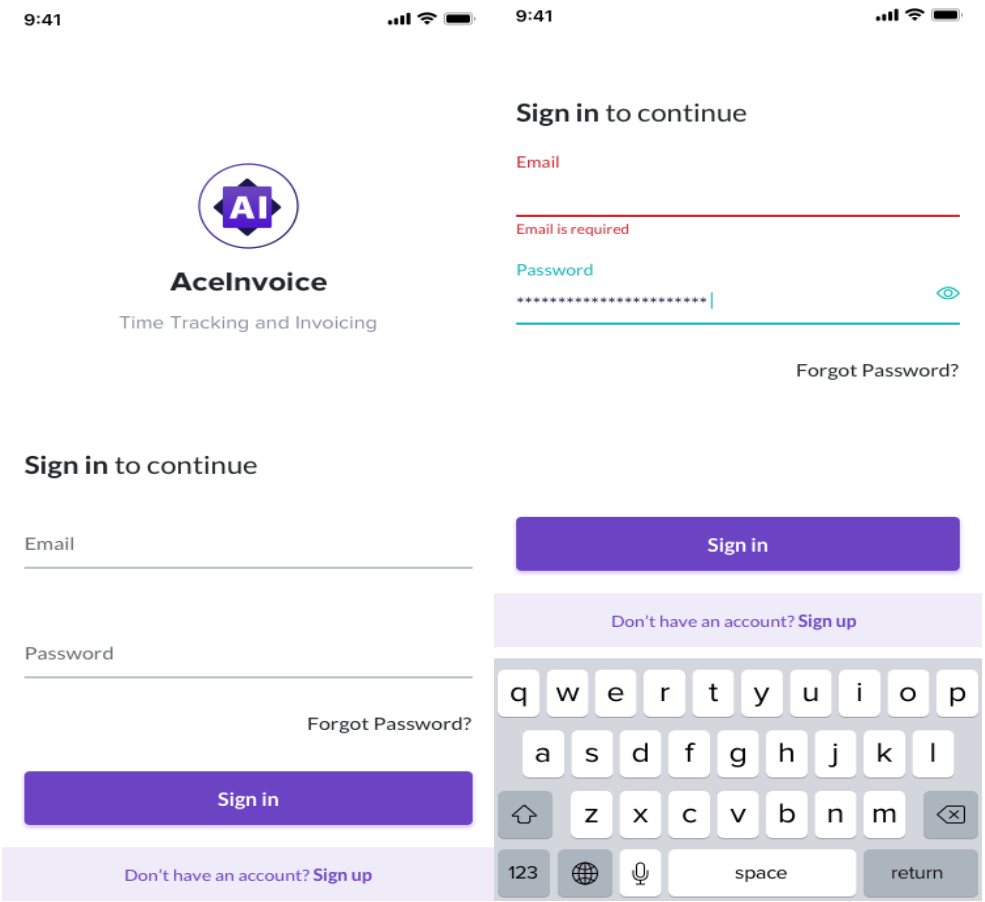

Fig 3.7: Login screen showing error and success messages

The Login, Signup and Forgot Password UI needed to be updated to the new prototype. These sections involve several routines and updating the UI for each and every routine wasn't easy. The primary reason for this was scalability. The UI changes are reflected differently on different devices due to varying screen sizes and pixel density. And not only that, since the same code would be used on both iOS and Android devices, it was a challenge to make UI changes consistent across devices and operating systems. Hence, the changes had to be made by limiting hardcoded values to a minimum and using the appropriate styles.

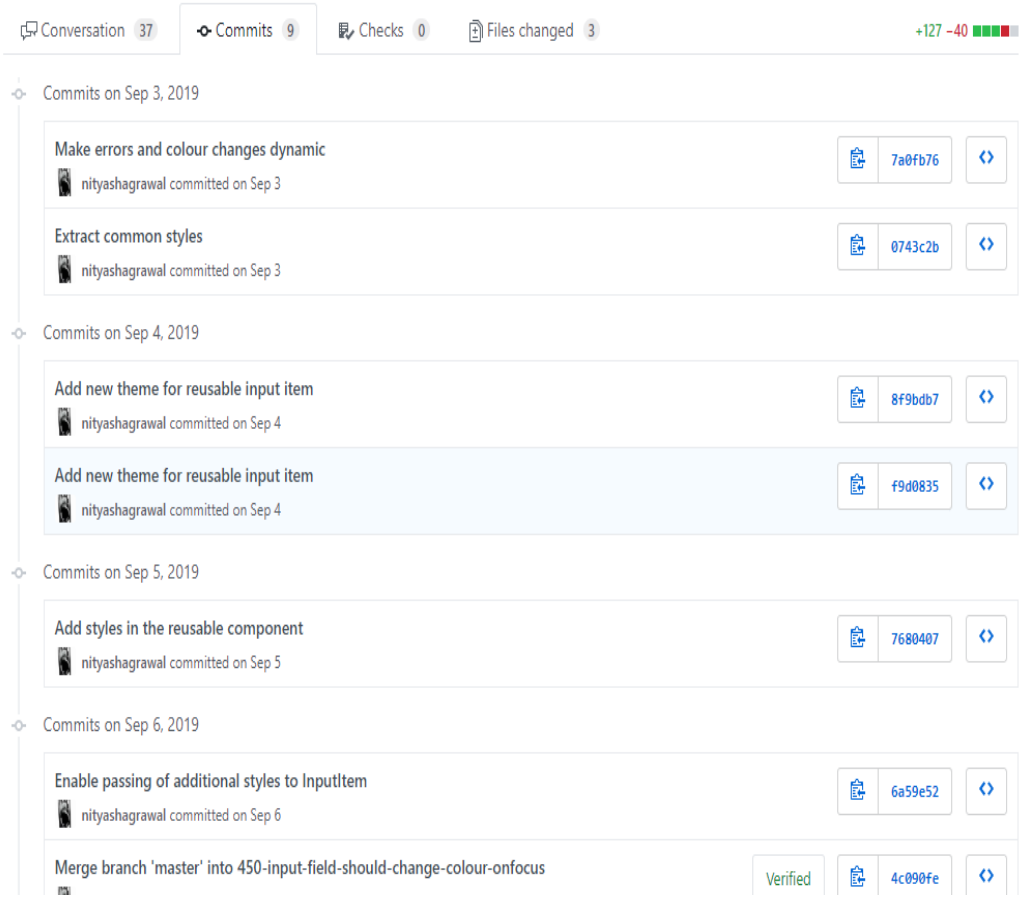

Fig 3.8: GitHub summary of the theme extraction pull request.

I also made a custom input field consistent with the designs which would change colour when under focus or when an error was encountered. I had to extract that and other related components to a themes folder and had to make it as general as possible so as to ensure that it could be used across all the pages and screens in the

application. This was done in order to promote reusability of code and to reduce the amount of duplicate and redundant code.

3.1.6 Displaying the status of time entries

The time entries logged by a user can be sent to the client for review and payment. I provided the feature of displaying the status of a time entry by fetching it from the database whether the entry was billed by the client or not. In case the entry was billed, then I disabled the edit option for that entry.

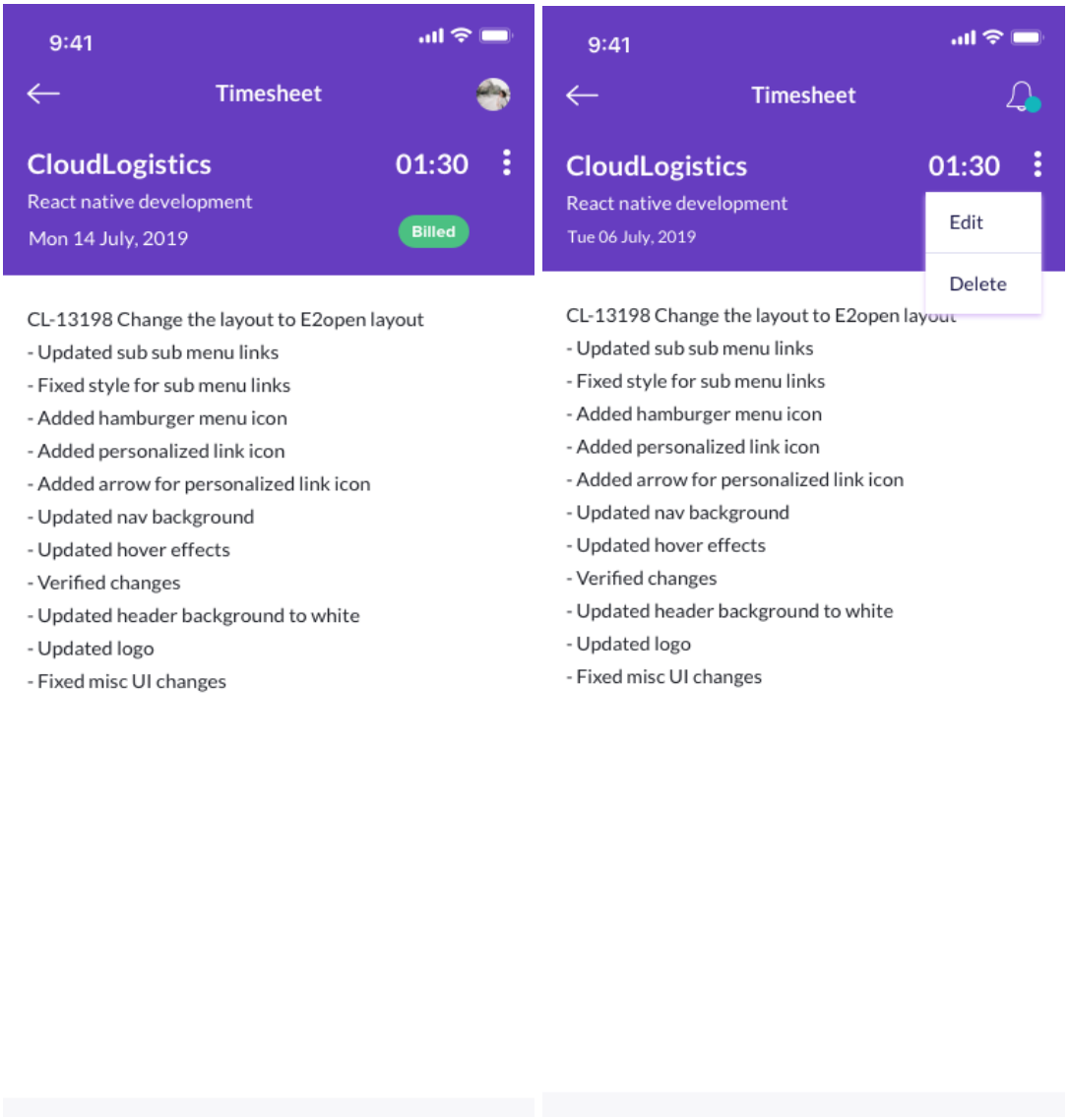

Fig 3.9: Timesheet screen showing the status of an entry

3.2 Zindi App

Zindi is a mobile application which makes online help docs contextual. It is a ticketing application where each ticket is a kind of grievance filed by the user which can be categorised by status - open or closed, date or priority- high, low, moderate or urgent.

3.2.1 Implemented a sorting function

I implemented a unique sorting function to sort tickets on various parameters in both ascending and descending order. I provided a toggling option so as to sort the tickets in the order the user wishes. I learned and used MomentJS and RamdaJS to solve this. MomentJS helps in handling of dates while RamdaJS provides an alternative way to sort data as compared to conventional sorting methods.

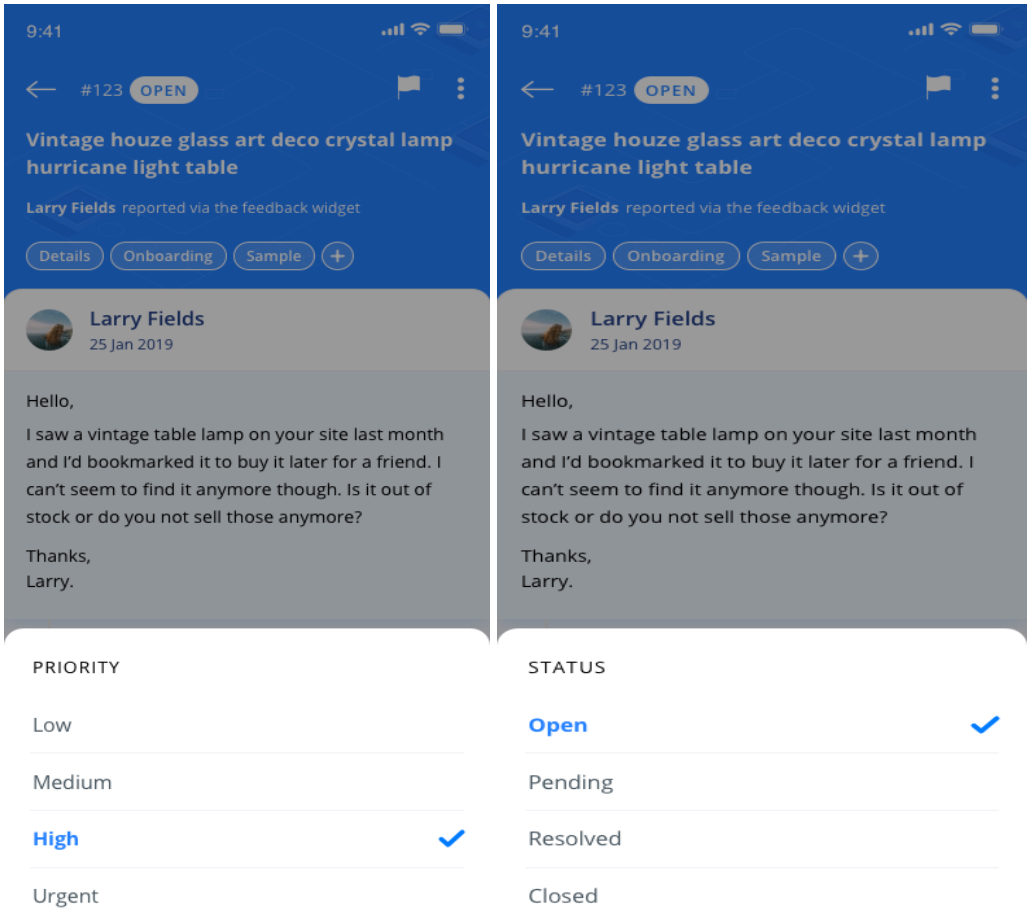

Fig 3.10: Ticket Details screens showing sorting options.

Chapter 4

Conclusion

The AceInvoice app had a very basic framework at the start of my internship. By the end of this internship, the app is production-ready with all the necessary features and is ready to be deployed on both iOS and Android app stores. The app is currently under the Google Play Store verification process and iOS testflight. Many users worldwide are already using the application.

This internship has been a great learning curve. It has been a valuable experience dealing with real-life problems, getting to know about new technologies and writing live code.

4.1 Contribution Stats

GitHub keeps a log of all the contributions and commits by a user. Here is a summary of the total contribution to the *aceinvoice-rn* GitHub repository:

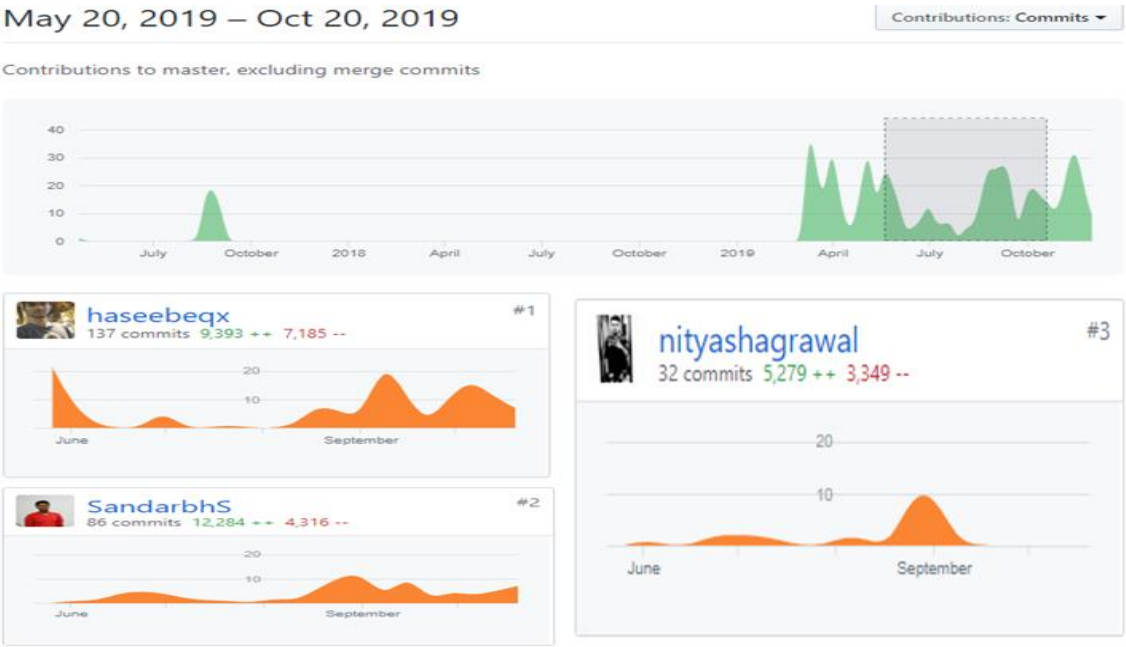

Fig 4.1: Contribution stats on GitHub

With **32 commits** and over **8.5k lines** of code modifications, I stood third in the overall contributors to the repository with 81 contributions overall.

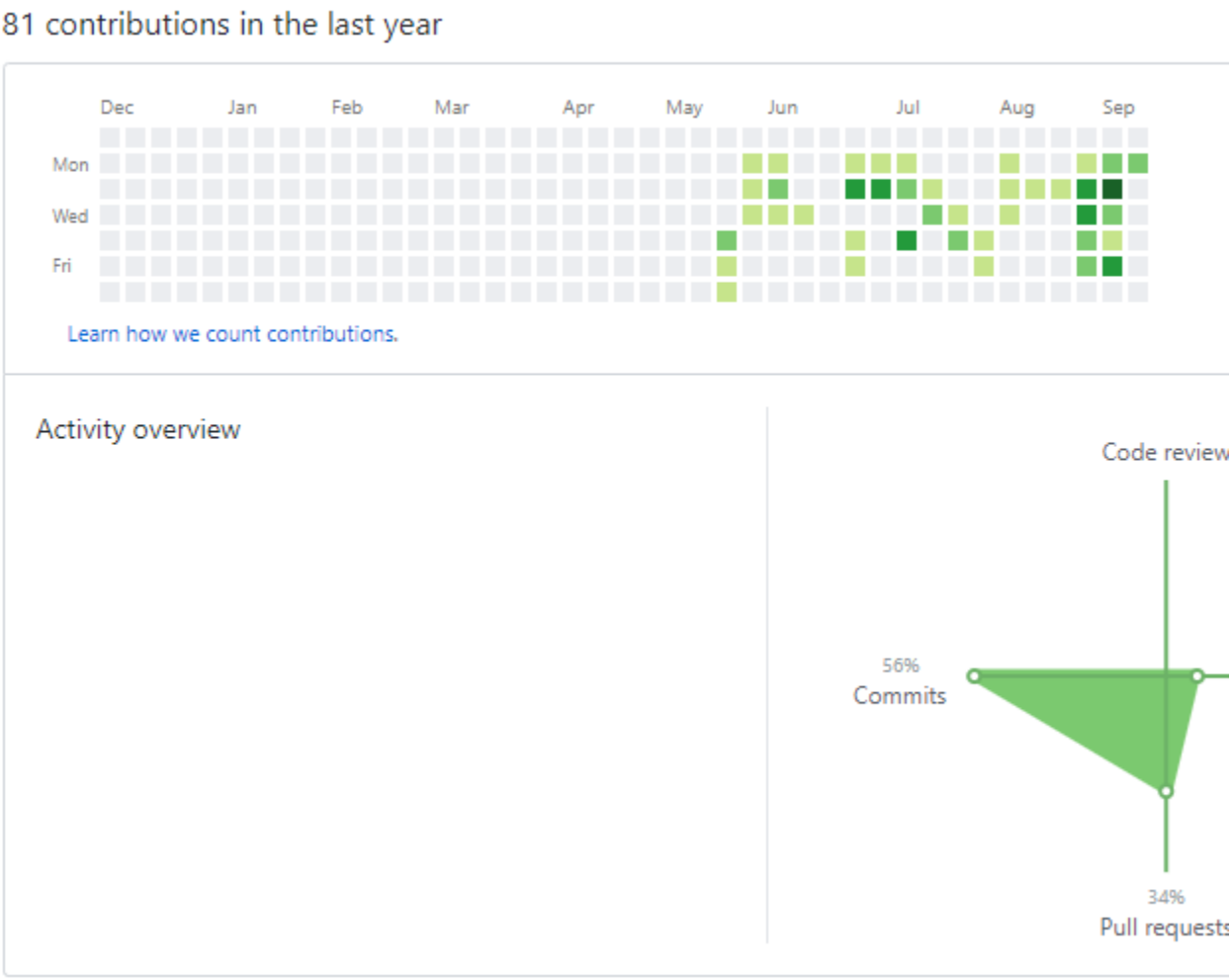

Fig 4.2: Graphical analysis of the GitHub contribution

References

- *Node.js docs: https://nodejs.org/en/about/*
- *React-Native docs: https://facebook.github.io/react-native/*
- *React-Redux official website: https://react-redux.js.org/*
- *BigBinary official website: https://bigbinary.com/*
- *JavaScript MDN Web Docs: https://developer.mozilla.org/en-US/docs/Web/JavaScript*
- *GitHub: https://github.com/*
- *https://dev.to/jessicabennett/react-native-still-a-profitable-choice-in-2019-5km*
- *https://www.inovex.de/blog/react-native-vs-native-development/*