

**Loan Management System
(Front-end Development)**

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

in

Computer Science and Engineering/Information Technology

by

Akhil Sahota (151225)

under the supervision of



Kuliza Technologies

to



Department of Computer Science & Engineering and Information
Technology

**Jaypee University of Information Technology Waknaghat,
Solan - 173234, Himachal Pradesh**

SUPERVISOR'S CERTIFICATE

This is to certify that the work titled "**Loan Management System (Front-end Development)**", submitted by **Akhil Sahota** for end semester (Even) of Bachelor of Technology in Computer Science and Engineering to Jaypee University of Information Technology, Waknaghat, Solan has been carried out under my supervision. This work has not been submitted partially or fully to any other university or Institute for the award of this or any other degree or diploma.

Signature of Supervisor:
Name of Supervisor:
Designation:
Date:



Mr. Devashish Jha
Product Consultant(Kuliza Technologies)

17/05/19

DECLARATION BY THE SCHOLAR

I hereby declare that the work presented in this report entitled “**Loan Management System (Front-end Development)**” submitted at **Jaypee University of Information Technology, Wagnaghat India**, is an authentic record of my work carried out under the supervision of **Mr. Devashish Jha (Product Consultant(Kuliza Technologies))**. I have not submitted this work elsewhere for any other degree or diploma.



Akhil Sahota, 151225

Department of Computer Science and Engineering/Information Technology,
Jaypee University of Information Technology, Wagnaghat, India
Date: 17-05-2019

ACKNOWLEDGEMENT

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend our sincere thanks to all of them.

I would like to express my sincere gratitude to my Project Manager, Mr. Devashish Jha for providing his invaluable guidance, comments and suggestions throughout the course of the project. I am highly indebted to Mr. Prithvi Buddharaju for his guidance and constant supervision as well as for providing necessary information regarding the project and also for his support in completing the project.

I would like to express our gratitude towards my parents and Kuliza Technologies for their kind co-operation and encouragement which helped us in completion of this project.

My thanks and appreciations also go to my colleagues in developing the project and people who have willingly helped us out with their abilities.

TABLE OF CONTENTS

Title	Page No.
1. Chapter-1 INTRODUCTION	1
1.1. Introduction	2
1.2. Problem Statement	3
1.3. Objectives	3
1.4. Methodology	3
1.5. Organization	4
2. Chapter-2 LITERATURE SURVEY	5
3. Chapter-3 SYSTEM DESIGN	8
3.1. Loan Management System	9
3.2. Client-server Model	10
3.3. Document Object Model	11
3.4. Methodology	12
3.5. Project Requirements	15
3.6. LMS Architecture Design	16
3.7. System Design	17
3.8. Modular Design	18
3.9. Authentication	19
4. Chapter-4 DEVELOPMENT	21
4.1. ReactJS	22
4.2. Redux	25
4.3. Other Technologies	29
4.4. Development Tools	33
4.5. Development Process	37
5. Chapter-5 TESTING	45
5.1 Test Plan	46
5.2 Unit and Integration Test Plan	48
6. Chapter-6 RESULTS AND PERFORMANCE	52
7. Chapter-7 CONCLUSIONS	54

LIST OF ABBREVIATIONS

1. LMS – Loan Management System
2. NBFC – Non-Banking Financial Company
3. UX Audit – User Experience Audit
4. UI – User Interface
5. DOM – Document Object Model
6. HTML – HyperText Markup Language
7. CSS – Cascading Style Sheets
8. XML – Extensible Markup Language
9. SDLC - Software Development Life Cycle
10. API – Application Programming Interface
11. JSX – JavaScript XML
12. HTTP – HyperText Transfer Protocol
13. W3C – World Wide Web Consortium
14. URL – Uniform Resource Locator
15. JSON – JavaScript Object Notation
16. VSCode – Visual Studio Code
17. NPM – Node Package Manager

LIST OF FIGURES

Title	Page No.
Figure 3.1 Client-server model	10
Figure 3.2 The HTML DOM tree of objects	11
Figure 3.3 Software Development Life Cycle	12
Figure 3.4 Agile Methodology	14
Figure 3.5 LMS system model	17
Figure 3.6 Top level use cases for LMS	19
Figure 3.7 State diagram for navigation	20
Figure 4.1 An example of JSX (React)	23
Figure 4.2 Comparison between Virtual DOM and Browser DOM in different stages of state change	24
Figure 4.3 State change process of LMS with Redux	26
Figure 4.4 Redux-saga flow diagram	28
Figure 4.5 Functionality of Webpack module bundler	32
Figure 4.6 Visual Studio Code user interface	34
Figure 4.7 Console panel of Chrome Dev Tools	35
Figure 4.8 Representation of BitBucket workflow branches	36
Figure 4.9 Jira software issue and project tracker	37
Figure 4.10 React-boilerplate folder structure(GitHub)	38
Figure 4.11 /app folder structure	39
Figure 4.12 /components folder structure	40
Figure 4.13 /containers folder structure	41
Figure 4.14 Implementation of Redux-persist	42
Figure 4.15 Snapshot of 'Organization' screen	43
Figure 4.16 'Organization' folder structure	44

ABOUT THE COMPANY

(Kuliza Technologies)



Kuliza is a leading provider of digital transformation and operational intelligence solutions for financial enterprises. Since 2006, Kuliza has executed more than 120 digital transformation projects for global startups and industry-leading global enterprises.

Lend.In is Kuliza's flagship lending product, a new-age lending system for banks and lending institutions to increase the overall efficiencies while decreasing the cost and go-to-market time for customers.

Kuliza's other suite of transformation solutions also help financial enterprises design and build end to end Mutual Funds commerce platforms & Insurance commerce platforms.

Kuliza has been recognized by Deloitte and Nasscom in the past has won several industry awards like the prestigious Technoviti Award by EY and Banking Frontiers, Digital Transformation Partner & Best Lending System of the year for NBFCs by elets technomedia, and has also been mentioned by global market research firms like Gartner, Forrester etc.

ABSTRACT

Today's market is progressing with a great pace. The people are lacking the time to accomplish all their tasks. The technology that saves time is always appreciated whether it is an automated machine in a factory or a bank providing a loan to its customers. Most of the times, banks deal with a delay in the process due to inappropriate loan processing system. This in some cases leads to the loss of their prospective customers. A suitable loan management system allows the bank to provide their customers with quick loan approvals, while at the same instance, the banks are allowed to disburse the loan amount in a fast pace which further results in happy and satisfied customers. The key aspects of the loan management system include reduced processing time and turnaround, ability to improve the performance throughout and inquire in a much lesser time, tracking of status on line and better document management, minimization of errors, details required and unwanted information requirements and better customer satisfaction with new product offerings and thus, impacting the minds of the customers. The financial sector is growing and grooming as each day passes and thus, introducing efficient new technologies in this sector will surely result in development of agencies and customers. With appropriate steps and efficient technologies, the customers and the agencies can bond together and the financial sector can gain more foothold.

Chapter 1
INTRODUCTION

1.1 Introduction

Loan Management System (LMS) is the most practical application developed to keep record of all the customer lending records, fund flows, customer records and cash flow. Any Non-Banking Financial Company(NBFC) or bank will need a Loan Management System to set up the loan product and perform client servicing. To accurately calculate the part of each payment that is to be applied to principal, late fees, interest, etc., a loan management system is required. It also comes with the benefits of an extensive report suite including monthly billings, loan payoffs, property tax renewals, accounting reports, account history, late notices, property tax renewals, promises to pay and many interest statements.

A detailed report on the customer's financial position is maintained by the loan processing system. All types of loans including balloon payments, standard mortgages, and interest only loans, etc. are handled by it. By regulating the business activities and maintaining proper lending and financial records, it simplifies the transactions. Various modules including loan tracking, lender module, mortgage module, deed of trust software, lending module, etc. can comprise a loan management system.

A loan management software system manages the loan information and the store or database. This data stored is used for cash flow information, tracking current installment payment status, loan servicing and other accounting tasks. To manage portfolio of loans and keep up with the financial data necessary is the basic purpose of such system. Some important features of this system must include create amortization schedules, sharing of principal, interest and late charge payments between two entities, charge/collect late fees, ability to handle conventional and non-conventional financing, accurately track partial payments, user friendly, powerful, affordable and fully featured.

This type of software program has a great impact on lender companies as the program is dynamic and flexible in nature. Investors, not-for-profit organizations and investors, all can extract some good from this. It builds some strong long-term client relationships.

1.2 Problem Statement

Servicing a loan does become complicated with each customer, who will have different payment dates, different terms and conditions, etc. You will have to make sure that accounting is in order, and a loan processing system aims to provide specific type of accounting to ensure your business is prepared when repayments are coming in. The current loan management systems in the market are complex and heavy by technical design aspects. This means that time to market is high and the system design is not flexible enough to meet the current and trending market needs.

1.3 Objectives

The main objective of building this product is to develop a lightweight, user-friendly, flexible and efficient loan management system. The goal is to develop a system that helps loans to get processed and service from the very beginning to the end. This creates a workflow that an employee or a business needs to manage. To make a good impression on the customers, they must be provided with an easy and understanding lending experience. Then, they get encouraged to use your organization for other loan opportunities. Since, it is a configuration-based product, the thought process is to change the UI, to suit the configurators, using Ant-design.

1.4 Methodology

Imagine you run an online shopping website. You know that your visitors interact with your homepage and what they look for in your search engines. They even get started on your checkout process. But they do not convert. This might be the time to update the user flows or the information hierarchy. But how would you know what need amendment or rejigging?

A method known as User Experience Audit (UX Audit) is used to point out less-than-perfect areas of a digital product, revealing the instances of the product or application or site that are causing headaches for users. A UX Audit, ultimately, lets you know how

to boost conversions by making it easier for users to achieve their goals on software or site.

One another primary development artifacts are User Stories. A user story is a very high-level explanation of a requirement, consisting of enough information so that the product developers can generate a reasonable effort and time estimate to implement it.

With the help of UX Audit and User Stories, the requirements from the project are made clear and the development is carried out.

1.5 Organization

The whole structure of the project has been carved up in sequential approach. There will be detailed overview of the project with implementation of front-end development technologies like React, Redux, React-router. An exploratory analysis of the each of used technologies is thoroughly studied to acknowledge their behavior.

Moreover, we will propose various tools and requirements to implement the development of our project.

Chapter 4 will be the powerhouse of our report as the critical implementation is taken care in it and the technologies implemented are explained.

Chapter 5 is mainly contrasting the documentation of test plans which were be developed by us.

The next chapters mainly deal with result and performance analysis by in-depth examination of our project scope. The evaluation of the results obtained is done.

CHAPTER 2
Literature Survey

RESEARCH PAPER I:

Learning React: Functional Web Development with React and Redux

Author: Alex Banks, Eve Porcello

Year: May-2016

Technology Used: ReactJS, Redux

Summary: The basic unit of all React applications are components. A component in React is a self-contained module that on rendering on the browser generates output. Interface elements like an input field or a checkbox can be written as a React component. A component can include one or multiple components in its output. Thus, components are composable. Redux is an open source JavaScript library for predictable state management of JavaScript applications. The reducers used by Redux are functions that compute and store the state of application without and worse effects.

RESEARCH PAPER II:

Developing a frontend application using ReactJS and Redux

Author: Khuat, Tung

Year: Nov-2018

Technology Used: ReactJS, Redux

Summary: The basic unit of all React applications are components. A component in React is a self-contained module that on rendering on the browser generates output. Redux stores the state in a simple plain JavaScript object which makes it easier to pass data and map out within the entirety of the application. The process of debugging and testing is also sped up by using a single object for storing the centralized state.

RESEARCH PAPER III:

Modern Web-Development using ReactJS

Author: Sanchit Aggarwal

Year: Mar-2018

Technology Used: ReactJS

Summary: React provides a much efficient and light weight document object model. The easy and non-complex nature of ReactJS enables one to quickly get comfortable with the framework. The reason for highly efficient performance of the framework is essentially is the virtual DOM feature of the framework. ReactJS is designed in such a way that unidirectional data flow that is downstream is allowed and supported. If bidirectional data flow is required, that additional features needs to be implemented. This was done as the components need to be immutable and data within them must not change under any circumstances.

CHAPTER 3
SYSTEM DESIGN

This section provides gets you acquainted with all the concepts and theories used in this project.

3.1 Loan Management System

Banks, being a financial institution, are engaged in the acceptance of deposit of money, granting credit (by means of overdraft, loan) and other transactions such as foreign exchange, discount of bills, etc.

Loans is one of the transactions being done in the bank which is a type of debt. Similar to all debt instruments, a loan includes redistribution of financial assets over time, between the borrower and the lender. The lender initially lends an amount of money to the borrower initially, which the borrower has to pay back to him, usually but not in regular installments. This service is generally provided at a cost which is referred to as interest on the debt. To pay off the loan, a fixed amount (for paying off and interest together) is paid periodically. A system that manages this whole system is called loan management system.

To provide decision-making assistance at the highest level of management, computers in business have risen to automate the jobs. Large corporations and industries depend on computer's accuracy and efficiency in its operation, with the environment at fast pace as it has the ability to retrieve, store and analyze data at tremendous speed at low cost. This development is viewed as an opportunity to control the data processing and increase the influence of the organization. Some organizations have their businesses totally dependent on the accurate computer operations. Thus, in the area of loan and advances, the banking industry is no exception to the advantages a computer provides, which contributes to the economic development in large measures.

Loans are readily given to people who have an account (mostly current account). Apart from being a customer, there are other conditions that necessary to be fulfilled for giving an individual a loan. Such conditions may include: having at least one guarantor, provision of collateral security and filling the bank loan management form. Thus, the role of a loan management system can never be overemphasized. A computerized loan management system helps the banks to work without stress.

3.2 Client-server model

A client-server model is a distributed communication framework that divides processes between a service provider (server) and a service requester (client). The clients and servers communicate with each other through a network by exchanging response packages or requests.

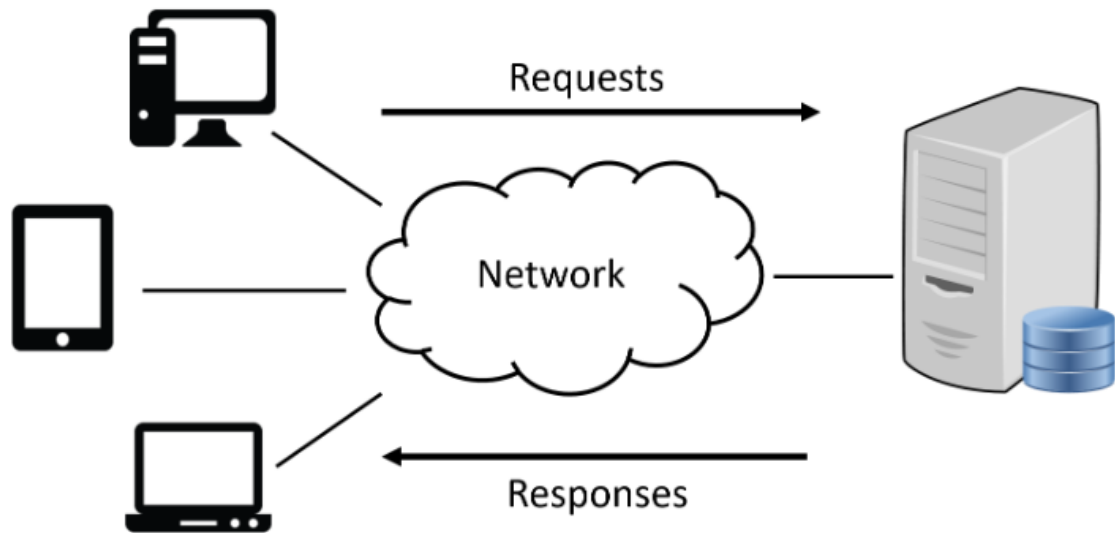


Figure 3.1 Client-server model

While using this model, the client side shares data and interacts with the database. All the resources are focused on user experience and the user interface. The server side handles business logic, database interactions and handles calculations. The servers can handle one or many different times at the same time.

However, with the rise in demand for applications with complex user interactions and adaptive interfaces rise, the application logic is moving more towards the client. This typically results in a better user experience by avoiding full page reloads after every interaction.

3.3 Document Object Model

The Document Object Model ^[6] is a programming interface for HTML and XML document. It interfaces pages to programming languages or scripts. The DOM can be seen as a page representation for programs to modify the style, content and structure of the document. The document is presented as a logical tree structure. There is a node in the end of every branch of the tree. There is an object present in each node. Event handlers can also be attached to the nodes. These events handlers get executed once an event is triggered.

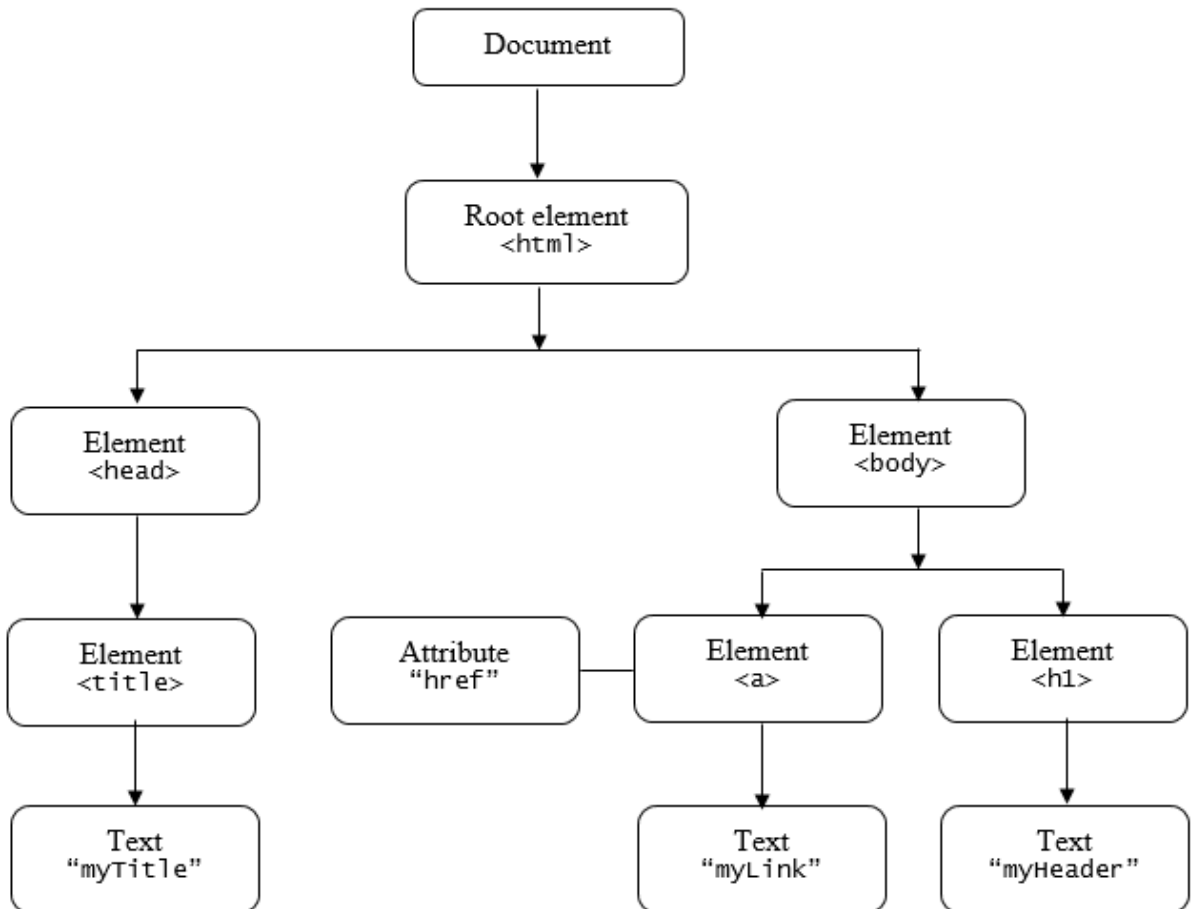


Figure 3.2 The HTML DOM tree of objects

3.4 Methodology

The framework used for the development of the LMS is Agile Development Model. To provide an overview of the project, we have used Software Development Life Cycle (SDLC).

3.4.1 Software Development Life Cycle

Software Development Life Cycle (or SDLC)^[1] is a popular framework used by the software industry to define the tasks that have to be performed in each step of the development process. To provide the best results with the resources given is the goal of SDLC.

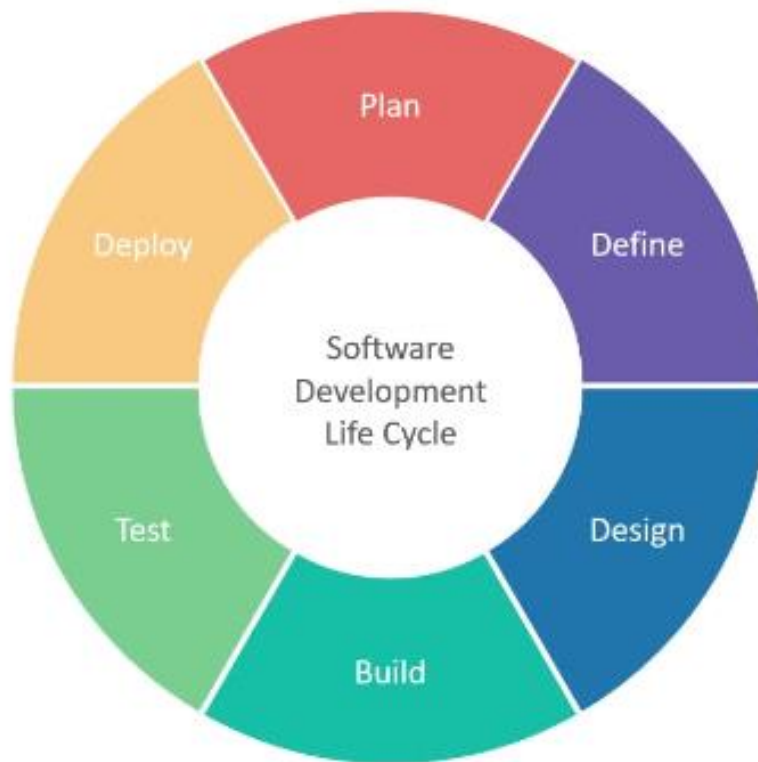


Figure 1.3 Software Development Life Cycle

Following are the stages of a Software Development Life Cycle:

- Planning and requirement analysis: A project plan is constructed, based on the objective that is hoped to achieve with the product. In accordance with the technical requirements and the economic resources the feasibility of the product is assessed. After this, the product quality assurance requirements are set. Risk assessment is also done to ensure that minimum risks are encountered in the development process.
- Defining requirements: In this stage, the requirements of the product are clearly documented and defined.
- Designing the product architecture: Based on the requirements of the product, a design document is formed which contains the proposals for the architecture of the product including the external and internal flow of data and modules to be used. A best approach is selected by the reviewers.
- Developing the product: In this stage, the product is build. According to the design selected, the programming code is generated. Depending on the type of software being developed, the programming language is chosen. Also coding standards and guidelines are set by the developer's organization.
- Testing the product: The software is tested for defects in this stage. The bugs and defects found are reported, tracked and fixed. This phase is repeated until the defined code standards are met.
- Deployment and Maintenance: The product is ready for deployment after it is carefully tested. Depending on the strategy of the organization, the product is set out for deployment. After it is released, when new bugs and errors are discovered or reported by users, the needed enhancements are taken care under the maintenance phase.

3.4.2 Agile Development Model



Figure 3.2. Agile Methodology

Agile Development Model ^[2] is a popular type of SDLC model. It focuses on adapting the process and satisfying the customer by rapid completion and deployment of working software product.

This model works in an incremental flow with multiple iterations. Each iteration consists of the basic SDLC stages with improvements in every successive iteration. These stages are as explained above: planning, defining, designing, building, testing, reviewing and launching the product. Time duration for an iteration can be from one to three weeks depending on the size of the product. Every iteration consists on teams working simultaneously on their respective areas.

In this methodology, an adaptive approach is used with no detailed planning. Step by step development take place. It focuses on what feature is to be developed next and what tasks have to be carried out to accomplish the development of that feature. The in-progress product features that are completed get reviewed by peers and customers, greatly reducing the major failure risks.

The logical reason for considering the Agile Methodology is due to its flexible nature. The Kuliza LMS project has a dynamic set of requirements as new functionalities can be added

at any time. Also, feature prioritization can be shift depending on the urgent need of the product.

Another logical basis for selecting this approach is that the company has to do everything in small amount of time encountering minimum failures and risks. To push out quality product fast, feedback is gathered regularly and the development process moves further accordingly.

Additionally, the development teams here regularly exchange ideas, work in close proximity and discuss the issues encountered. Thus, the progress is done without much documentation. Group meetings are easily arranged and revision is done.

3.5 Project Requirements:

The following section describes the requirements for the LMS. This is the defining stage of the SDLC.

Extensibility is a principle which takes into consideration the future growth during each implementation of the design. It is not possible to design everything in advance in software engineering. Considering this, the extensibility in the design of the product emphasizes properties that helps in modifying or extending the system with minimum effort. Modifiability, maintainability and scalability are the three most important properties of extensibility.

Modifiability is the ease with which one can modify a software system. The way in which each functionality is architecturally organized and separated helps in determining the modifiability. If a change in system requires as few changes in each related component as possible, a system is said to have high modifiability.

Maintainability is the endeavor required to locate the bugs and errors in an operational software and fix them. In definition, maintainability and modifiability are quite similar. The major distinguishing point between maintainability and modifiability is that the former takes correction of bugs into consideration whereas the latter does not.

Scalability is defined as “the system ability to expand in a chosen dimension without any major architectural changes”. Scalability can be quantified by seeing the ease in how a system without any major change in design can add new components and new functionalities.

Since I have been working on the frontend development team of the project, I state the quality requirements of the project from frontend perspective:

- The system should have an uncluttered, simple user interface with readable font-size and font.
- The website must be responsive to the web browsers across all the devices.
- Loading indicators are to be added for asynchronous items.
- While scrolling there must not be any shuttering and each view should have a minimum loading speed.
- The system must not crash or freeze in the browser.

3.6 LMS architecture design

This section details the Designing step in SDLC. The architecture of LMS is structured based on the requirements defined previously. Fundamentally, it uses a React-redux framework of development and the role it follows is of a client-side application.

3.7 System Design

This project has a system design which is based on the conventional client-server model.

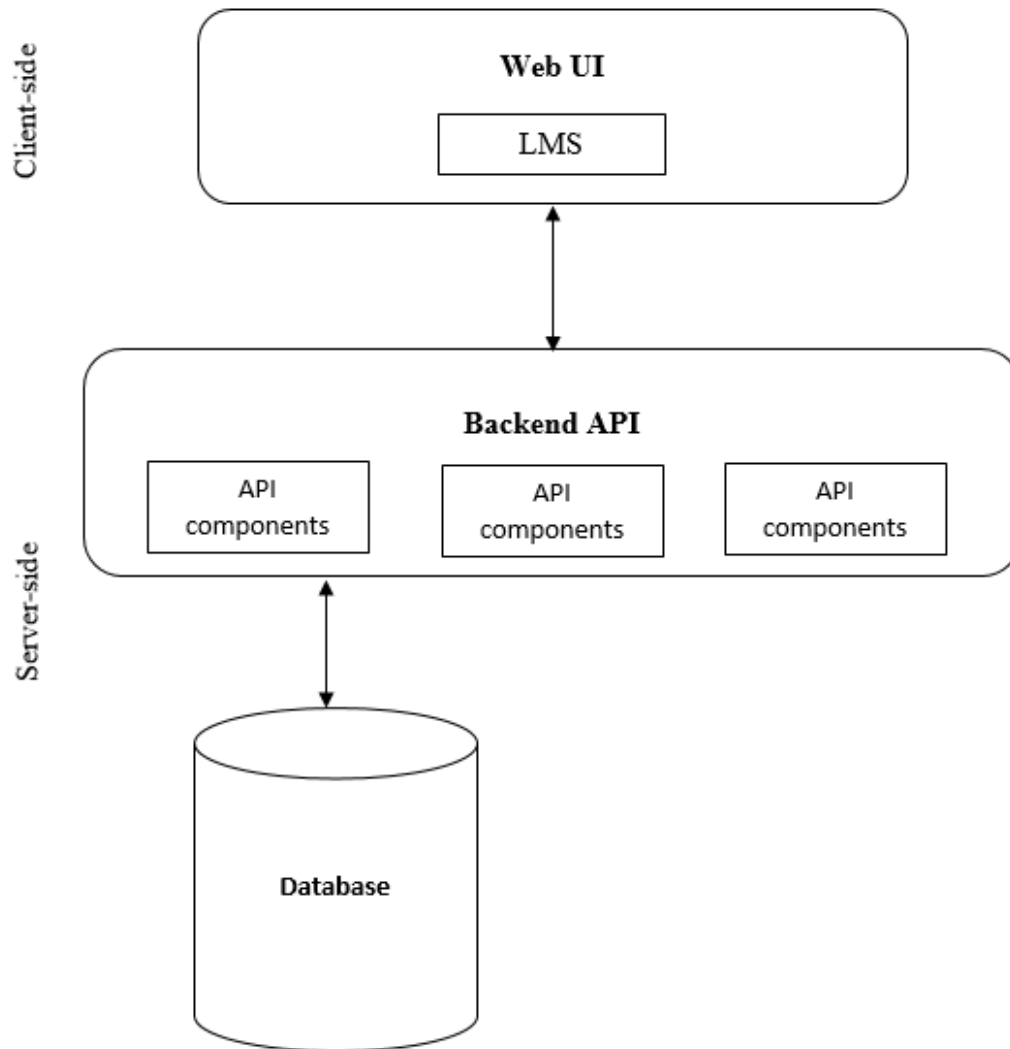


Figure 3.3 LMS system model

The requests made by the client are handled by the backend API. Business logic, calculations, database interactions are all taken care of by the backend. Frontend also contains structured ways to store data and make calculations. However, the frontend is mainly focused on providing the best UI interface with the best user experience and optimal performance. The attractiveness of the web page alone is not enough. Depending on

the type of the customer the company wishes to attract and what they specifically need right away, the website must be specifically attractive to them. Just as the first meeting depicts the personality of a business profile, the same is the case with the frontend of the website.

3.8 Modular Design

The Modular Design approach focuses on dividing a system into sub-modules which are independent of one another. A modular system comes with the advantages of low cohesion and high coupling, thus increasing scalability, modifiability, reusability and flexibility of the system. Dividing the system into sub-parts helps in creating a different flow path analysis which makes it possible to identify critical components in performance and functionality of the system. We can thus assign priorities to these components based on their importance and spend more time in perfecting them. By optimizing these components, or by replacing the low performing ones with high performing ones, it makes it possible to improve performance and efficiency of the system.

To divide the system into independent discrete modules, the initial step is to identify and elaborate the different functionalities that are possessed by the system. And then, based on these functionalities, the main components are separated. To identify critical components and analyze different flow paths, use case diagrams were used. In modern software engineering, the use case analysis is a valuable and important requirement analysis technique. It helps in capturing the functional requirements of the system and serve a basis for scheduling, estimating and validating effort. They also help in capturing additional behavior that can improve the robustness of the system. The use cases have proven to be easily understandable by the business users and thus, have proven to be an excellent bridge between end users and software developers. The incremental, evolutionary and inherent iterative nature of use cases make them well fit for agile development.

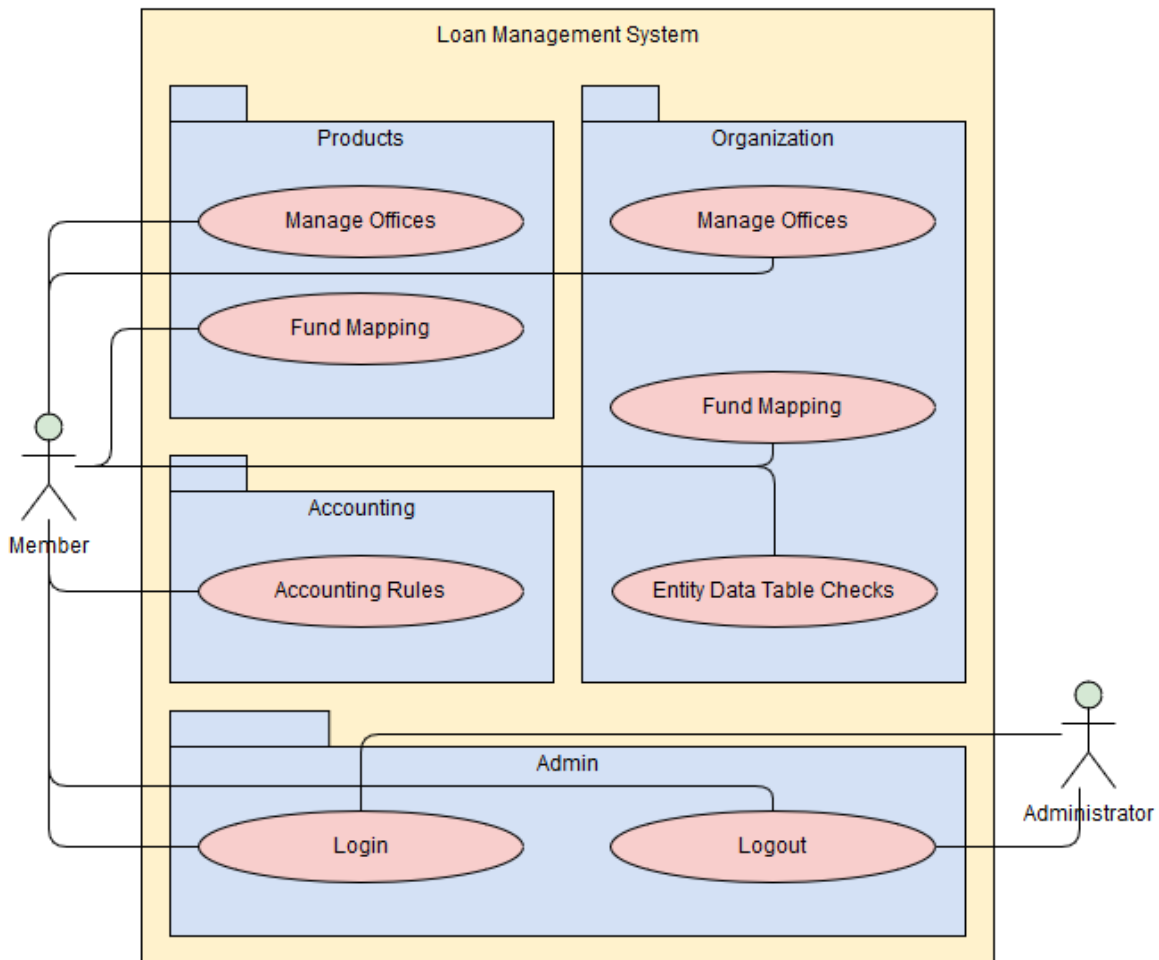


Figure 3.4 Top level use cases for LMS

In figure above, the most basic interactions between the LMS user and the backend API have been described. The Loan Management System module components are sorted based on the service provided based on this diagram. They are separated into user authentication, accounting, system and organization services.

3.9 Authentication

The user authentication is implemented using React-router, which is discussed in detail further in this project. The application design can be divided as: pages with authentication (Dashboard Page) and pages without authentication (Login Page). The objective here was to create a secure component that allows the user to access only specific number of pages when logged in or else the user will be redirected to the login page.

Whenever the user navigates to a page that requires authentication, user authentication is checked. And obviously we also require log in and log out functions.

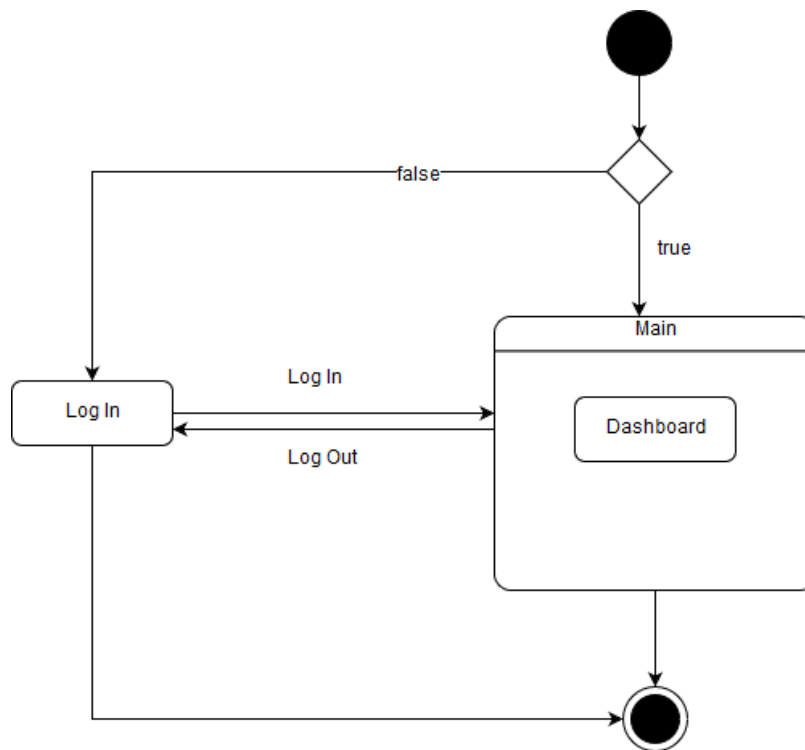


Figure 3.5 State diagram for navigation

Chapter 4
DEVELOPMENT

4.1 ReactJS

ReactJS ^[3], which is also known as React.js or React, is an open-source JavaScript library which is used for building user interfaces. The handling of the view layer in single page web and mobile applications development is done by the React. The maintenance of React is done by Facebook, Instagram and a community of developers and corporations.

The basic unit of all React applications are components. A component in React is a self-contained module that on rendering on the browser generates output. Interface elements like an input field or a checkbox can be written as a React component. A component can include one or multiple components in its output. Thus, components are composable.

React components ^[4] are written corresponding to the various interface elements. The structure of the application is defined by the higher-level components inside which we these React components. This is how React apps are written.

For example, we have to build a form. There are various interface elements like buttons, checkboxes, input fields, or labels that a form might consist of. Each of these elements are written as a React component inside the form. Then for the form itself can be written as a higher-level component. This new higher-level component specifies the structure of the form and all the interface elements comprising the form are included in it.

Interactive, complex user interfaces often comprise of complex application data and state. There are strict principles of data management that each component in the React app must put up with. React aims for scalability, speed and simplicity. Some of its notable features are Stateful components, JSX and Virtual Document Object Model.

4.1.1 Stateful components

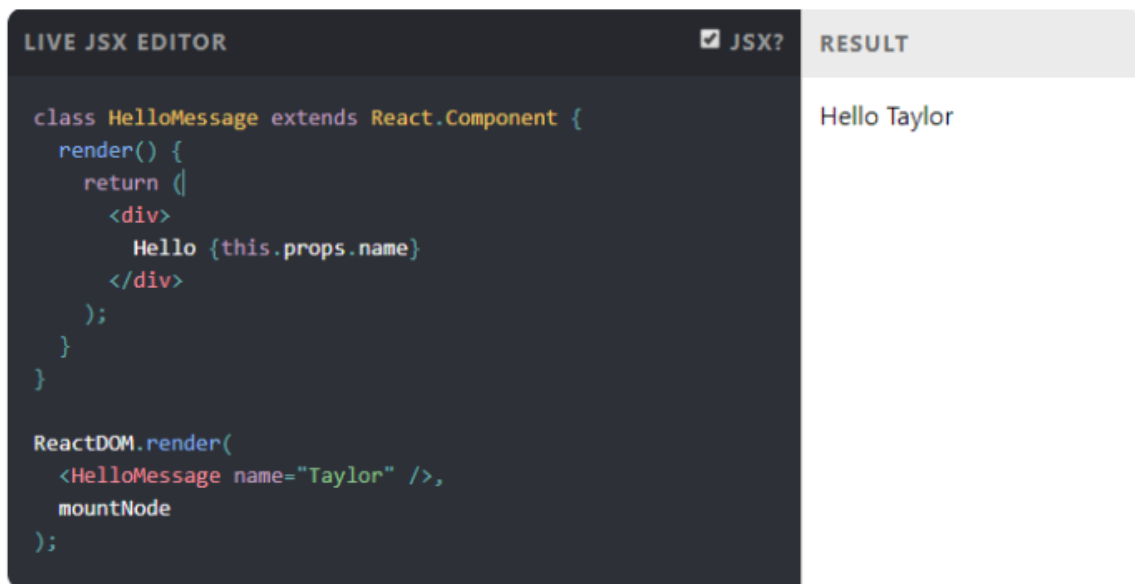
React allows to have React components which are these reusable independent pieces into which the UI is split. There is a render method implemented by each of the React components that takes the interface elements to be displayed as input and returns what to display on the browser screen. During the process to execute code at specific times there

are lifecycle methods that each component follows. These methods can be overridden at times. React API is used to call these methods.

To react and record to user events React uses a plain JavaScript object called State. The component, along with all its child components, immediately renders whenever a component state is changed. The values held by the states can be passed from a component to its child components and further to their child components as props.

4.1.2 JSX

JSX ^[5] or JavaScript XML is a syntax extension to JavaScript. It is similar to a template language but it comes with the full power of JavaScript. JSX is optional and does not require to use React. But it is recommended to use JSX with React to display the way a UI should look like. It provides a visual aid when you are using JavaScript for a UI. It allows React to show more useful warnings and error messages.



The image shows a 'LIVE JSX EDITOR' interface. On the left, there is a code editor with a dark background. The code defines a class 'HelloMessage' that extends 'React.Component'. The 'render()' method returns a JSX element: a 'div' containing the text 'Hello {this.props.name}'. Below the class definition, 'ReactDOM.render()' is called with the 'HelloMessage' component and the prop 'name="Taylor"'. On the right, there is a 'RESULT' panel with a light background, displaying the rendered output: 'Hello Taylor'. A checkbox labeled 'JSX?' is checked in the top right corner of the editor area.

```
class HelloMessage extends React.Component {
  render() {
    return (
      <div>
        Hello {this.props.name}
      </div>
    );
  }
}

ReactDOM.render(
  <HelloMessage name="Taylor" />,
  mountNode
);
```

Figure 4.1 An example of JSX (React)

4.1.3 Virtual Document Object Model

The original HTML DOM was focused on static pages and were not intended for the creation of dynamic UI. The DOM has to re-paint the entire page along with its entire layout and CSS and update each node, every time the DOM updates. It is very possible for a single page application to consist of several dynamically generated nodes that have event listeners attached to them. The HTML DOM checks for the updates and changes in every node at a regular interval in case of dynamic UI. Every time re-painting and re-rendering the entire page for even a small update in the node decreases the performance and efficiency of the web page considerably. As a solution for this inefficiency, the virtual DOM was invented.

In the Virtual DOM, an ideal representation of the UI is stored in the local memory and synced with the real DOM. It is an abstraction of the real HTML DOM. It is very lightweight and is not attached with the browser. Without affecting the real actual DOM, the actual virtual DOM can be updated. React uses a built-in module called ReactDOM. A process called reconciliation is used by React when the nodes are updated, using an algorithm that contrasts and compares changes to know which interface elements have to be updated. React then updates only those elements without affecting the rest of the elements.

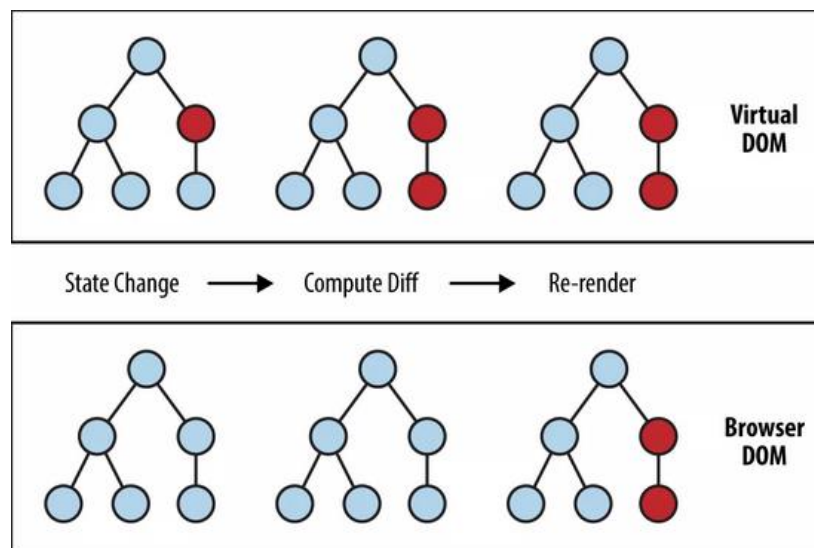


Figure 4.2 Comparison between Virtual DOM and Browser DOM in different stages of state change

4.1.4 React Router

React Router is an API that has its own usefulness in dynamic client-side routing. The React Router allows us to construct a single web page application through which we can navigate without refreshing the page as the user navigates between them. A component structure is used by it to call components, which are responsible for displaying the correct information.

A blank page or the flash of a white screen is prevented by preventing a page refresh. This is an increasingly common way of having a simple and more seamless user experience. This can be achieved using Router or Link. React Router also enables the users to utilize the functions of the browser like the refresh button and the back button while conserving the application view.

The UI is kept in sync with the URL by the React Router. It has powerful and very useful built-in features like location transition handling, dynamic route matching and lazy code loading.

4.2 Redux

Redux ^[7] is an open source JavaScript library for predictable state management of JavaScript applications. One of the most complex and important aspects of software development is state management. A source of most major errors is state mismanagement. Facebook's flux architecture which is a Model-View-Controller framework is simplified and implemented as Redux. To reduce the complexity, it uses something called as reducers. The reducers used by Redux are functions that compute and store the state of application without and worse effects.

Redux is mainly based on the following stated principles ^[11]:

- The application state in the reducers is stored in a single object. Redux stores the state in a simple plain JavaScript object which makes it easier to pass data and map out within the entirety of the application. The process of debugging and testing is also sped up by using a single object for storing the centralized state.

- The state of the application is always immutable. The values stored in the states cannot be modified in Redux at any cost. To depict or describe state changes Redux uses actions. These actions are immutable JavaScript objects. To prevent race conditions, these actions are orderly executed.
- It is specified by the reducers that how the actions will transform the state. The reducers are functions in JavaScript that take current state and action as input and return the new state as output. The data changes are thus centralized and can be acted upon on all or part of the state. It is also possible to combine multiple reducers into one and reuse them.

Having such an architecture increases scalability considerably for large and complex web applications. Also it enables a very powerful developer tools as it makes it credible to track every small mutation to the action that led to it. Knowing the current state and action, the upcoming state can be predicted with absolute certainty.

The following figure shows the flow of data in LMS with Redux.

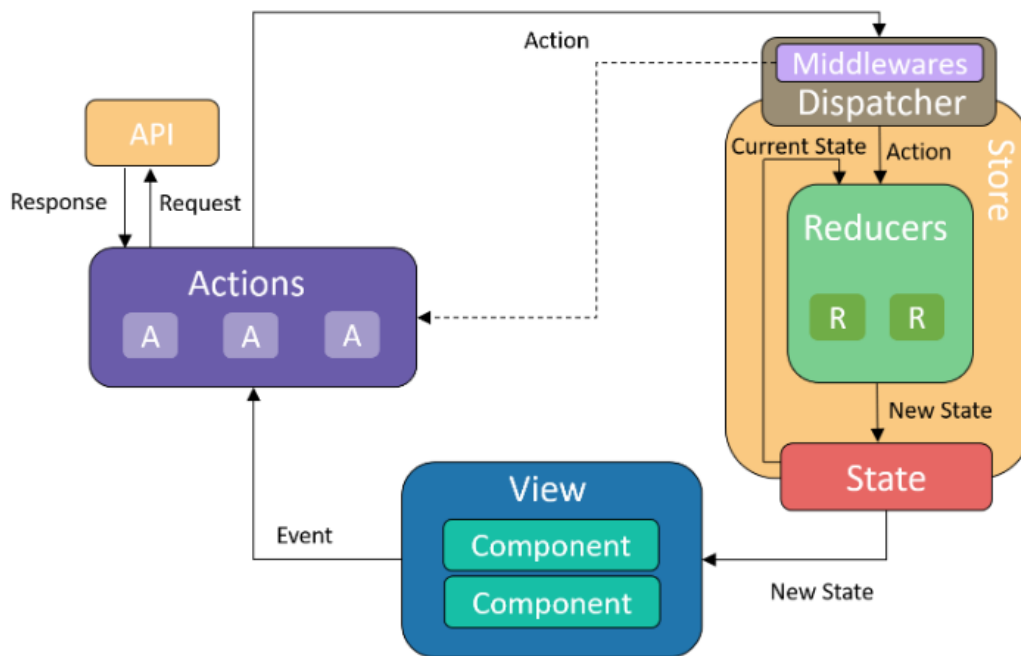


Figure 4.3 State change process of LMS with Redux

Whenever the user interacts with an HTML element, an action is dispatched by the corresponding action call. It is checked by the dispatcher that if the action requires an HTTP request to be sent to the backend API with the help of middlewares. Whenever such interactions are necessary, the action is returned by the dispatcher and as it is asynchronous, it waits for the response from the backend. The action is passed back, after the request is completed. The reducer then receives the current state and the action from the dispatcher. The reducer takes it as an input and returns the new state and interchanges it with the old one. The state that has been changed has some changed view components that are notified to the Redux store. According to the updated state, the component is re-rendered.

4.2.1 Redux-saga

Redux-saga^[8] is a middleware library for redux that have been specially designed to handle the side effects present in your redux application. It uses generators, an ES6 feature, and allows the developers to write synchronous looking asynchronous code, and is very simple and easy to test. The ‘saga’ pattern is a way to handle very long running transactions with potential side effects and failures. Also a counter-transaction is needed to revert back in case anything goes wrong.

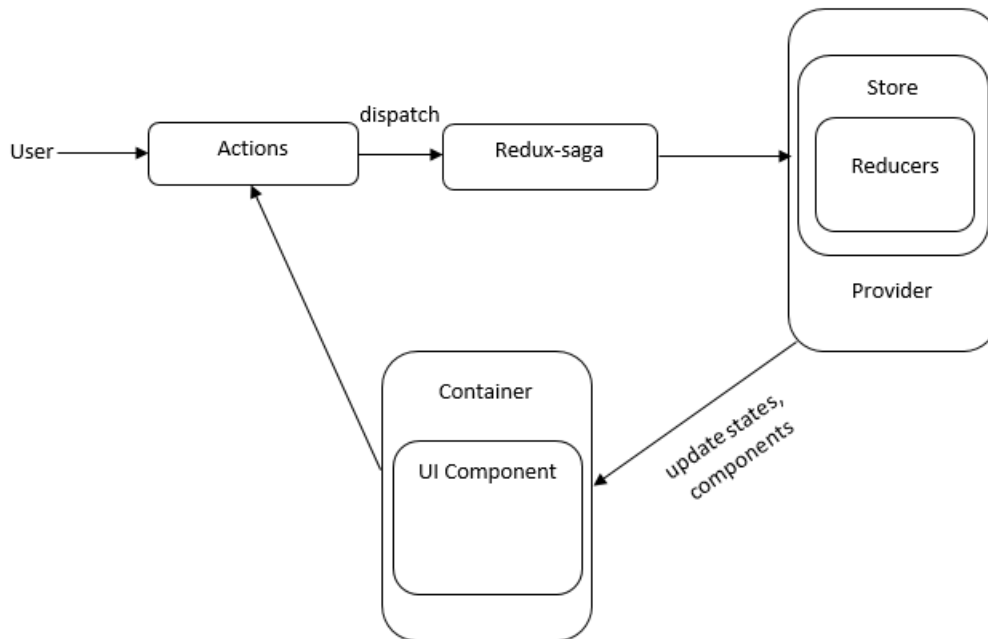


Figure 4.4 Redux-saga flow diagram

The hardest part in developing an app are asynchronous calls – how to handle timeouts, network requests, and other callbacks without further complicating the Redux reducers and actions. Using Redux alone for handling the asynchronous calls won't be able to provide much flexibility since at its core, Redux is nothing but a state container that manages synchronous data flows. Every time an action, which is a plain object describing what has happened, is passed to the store, the reducer gets called to update the container state immediately. But in case of an asynchronous flow, the response has to be waited for first, and if, there is no error or failure encountered, the state is to be updated. And this becomes worse if your application has a complex logic/workflow.

These side effects are made easier and simpler by working with sagas. Sagas are a kind of design pattern that have been taken from the distributed transactions world, where a saga manages the needed processes to be executed in a very transactional way, storing the state of the executing and compensating failed processes. In context of Redux, a saga is used as a middleware to trigger asynchronous actions and trigger them. The ES6 generators, which

are functions that have the ability to be paused and resumed, instead of executing in one go. This makes the asynchronous code easy to understand and write.

4.3 Other technologies

4.3.1 HTML

Hyper Text Markup Language (HTML) is the primary markup language that is utilized for making website pages and other data that can be shown in a web browser. It defines the meaning and structure of the web content. It describes how a document should be displayed by an internet browser. “HyperText” basically refers to links that connect multiple web pages to one another, either within a single website or between websites. A fundamental aspect of the Web are links.

HTML is written in type of HTML elements comprising of tags encased in angled brackets (like `<html>`), within the web page content. HTML tags for the most part come in pairs like `<h1>` and `</h1>`, albeit a few tags, known as empty components, are unpaired, for instance ``. The first tag in a pair is the start tag, the second tag is the end tag (also known as opening and closing tags). In between of labels, website designers can include text, content, tags, comments and different kinds of text-based content. The purpose for a web browser is to read HTML documents and compose them into visible or audible web pages. The browser does not display the tags but uses them to interpret the web page content. These elements created using the tags form the building blocks of all websites. HTML allows to embed images and objects which can be used to create interactive forms. It provides a way to create structured documents by denoting structural semantics for text such as lists, paragraphs, quotes, headings, links and other items. It can also embed scripts written in scripting languages such as JavaScript which enhance the behavior of HTML web pages.

4.3.2 CSS

CSS (Cascading Style Sheets) is a stylesheet language which is used to describe the presentation of an HTML or XML written document. It is a simply designed language intended to simplify the process to make the web pages presentable.

It enables the separation of the web document content (written in HTML) from document presentation, including the beautification elements such as colors, font and layouts. This separation improves content accessibility, control in the specification of presentation characteristics, provide more flexibility, enable multiple pages to share formatting, repetition in structural content (such as by allowing for tableless web design) and reduce complexity. It allows us to use apply styles to make web pages independent of the HTML that makes up each page. It describes how the elements have to be rendered on paper, in speech, on screen or on other media.

CSS can allow the same markup page to be presented with different rendering methods for different styles such as in print, on-screen, by voice and on tactile devices. We can also use it to allow the web page to display differently depending on the size of the screen on which the web page is being viewed. There is also a priority scheme that CSS specifies to determine which rule style to prioritize in case more than one rule matches against a particular element. The CSS standards are taken care by the World Wide Web Consortium (W3C).

A CSS comprises of style rules that the browser interprets and applies to corresponding elements in the document. A style rule set has a declaration block and a selector. The declaration block contains one or multiple declarations which are separated by semicolons. Each declaration has a CSS property name and a value, which are separated by colon. Also, it ends with a semicolon and surrounded by curly braces. The selector points to the HTML element which is to be styled.

4.3.3 JavaScript

JavaScript(JS) is an interpreted, lightweight or JIT compiled programming language with first-class functions which means that it supports returning functions as the values from other functions, passing them as arguments to other functions, storing them in data structures or assigning them to variables. It is well-known as a powerful client-side scripting language for Web pages. It is mainly used for enhancing user interaction with the web page. In other words, JavaScript helps in making the webpage livelier and interactive. The standard of JavaScript is ECMAScript ^[12].

JavaScript was originally known as LiveScript, but Netscape renamed it to JavaScript. It cannot run on its own since it is a scripting language. In fact, the JavaScript code is run by the web browser. When a user requests an HTML page which has JavaScript in it, the script is sent to the browser and it is the browser that executes it. JavaScript has a main advantage that it is supported by all modern web browsers. Also, it runs on any operating system including Mac, Windows or Linux. You just need a text editor to write JavaScript code and a browser to display the web page.

Many good features such as dynamic objects, functions, loose typing and an expressive object literal notation are provided by JavaScript.

- JavaScript functions have lexically scoped first class objects. This is what makes it a more powerful language.
- The developers feel liberated from having to form a complex class hierarchy and reduce the concerns about the type system by using loose typing objects.
- A powerful and expressive object literal notation is possessed by JavaScript. Simply by listing their components one can create the objects. This notation inspired the popular data format, JSON.

In conclusion, “JavaScript is a full-featured programming language, as complex as any and more complex than some”.

4.3.4 Webpack

Webpack ^[8] is a type of module bundler whose work is to convert the dynamic modules with dependencies into static modules. It is general to have various modules from different from either custom modules or files installed by NPM (Node Package Manager) in application development. Single or multi-layer dependencies can be contained in these modules. Through a dependency graph, Webpack bundles these modules into a single JavaScript file. The developers can also export and import modules, codes and other assets between JavaScript files using the webpack. To extend the functionality of Webpacks can be installed with plugins and loaders.

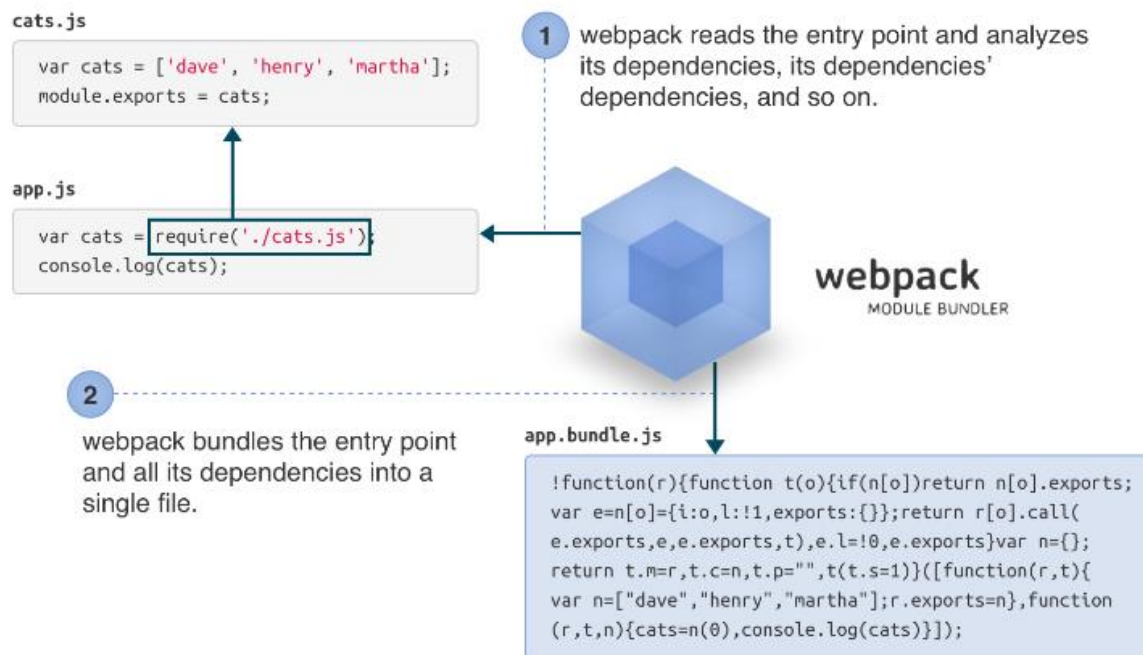


Figure 4.5 Functionality of Webpack module bundler

The following features are offered by Webpack:

- The assets which are not used to reduce file size are removed by dead asset elimination.
- The chunks of code are split into simpler parts and used only when needed by enabling code splitting
- Webpack parses most third-party libraries by using clever parsing.
- Other resources are transformed to JavaScript using loaders. Most important loaders in this project are the JSX and Babel loaders as they are required for React. ES6 syntax is used by the developers in Babel loaders which gets compiled to ES5 syntax for full browser compatibility.

The built-in development servers of Webpack have the feature of live reloading. The Webpack Development Server is a small NodeJS Express application which uses the Webpack middleware to serve up the bundled modules for local development.

4.4 Development Tools

4.4.1 Visual Studio Code

Visual Studio Code (VSCode) is the text and source code editor choice used in this project. The benefits of using VSCode are that it is a free open source editor with a built in package manager, cross platform editing and a file system browser with multiple panels ^[10].

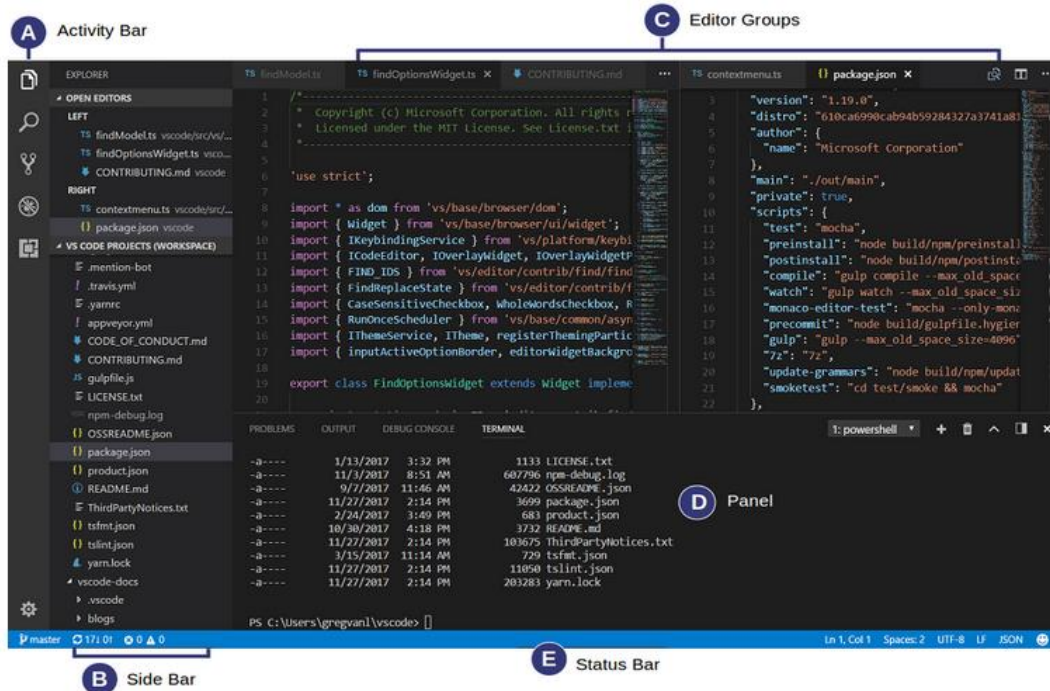


Figure 4.6 Visual Studio Code user interface

Additionally, jsLinter can be installed using VSCode’s package manager. There are several highly customizable options but by default, jsLinter looks for syntactical errors and correct them when the file is saved. The code also gets arranged in a coding standardized style such as ESLint, standard JS or Airbnb. This hugely increases maintainability and modifiability of the source code.

4.4.2 Chrome Dev Tools

Chrome Web Tools is a set of tools for web developing which come built-in directly in the Google Chrome browser. These DevTools help to diagnose bugs and problems quickly as well as edit the web pages on-the-fly. Ultimately, this leads to building better websites, faster.

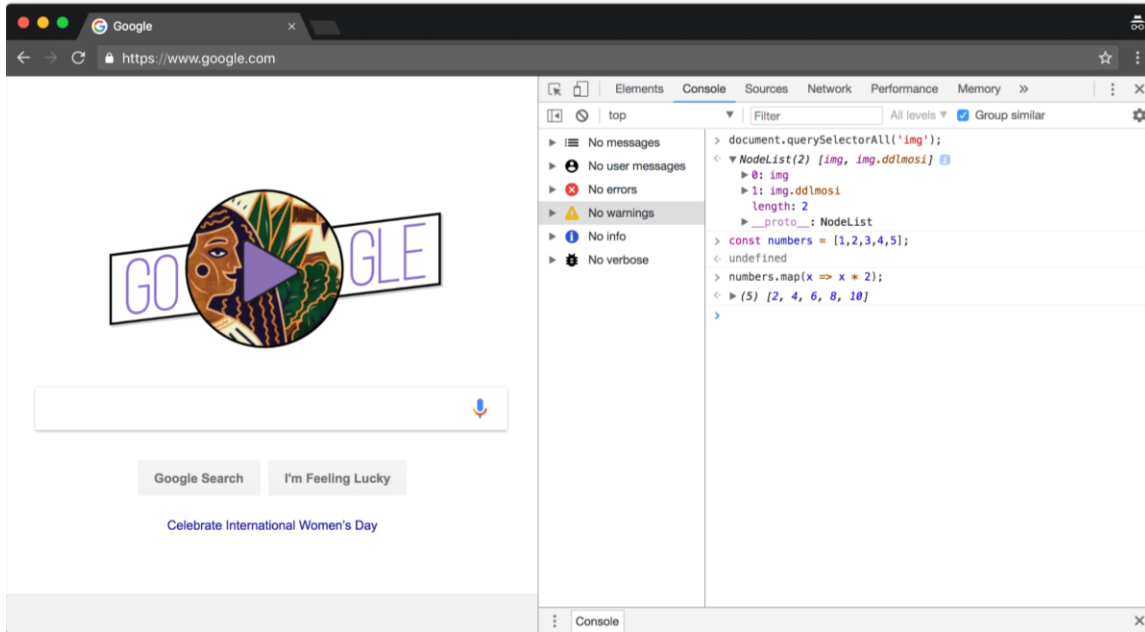


Figure 4.7 Console panel of Chrome Dev Tools

4.4.3 BitBucket

BitBucket is one of the most widely used modern version control system. It works as a powerful tool for a development team to collaborate on various instances in the same project. The most optimized service for developers to manage their source code is provided by BitBucket along with flexibility, high security and high performance.

The workflow in BitBucket is very essential to accomplish the task assigned in productive and consistent manner. The users thus get encouraged to leverage BitBucket effectively. The BitBucket workflow design defines a strict branching model based on the release of the project. A robust framework is, thus, provided to the developers for managing the larger projects.

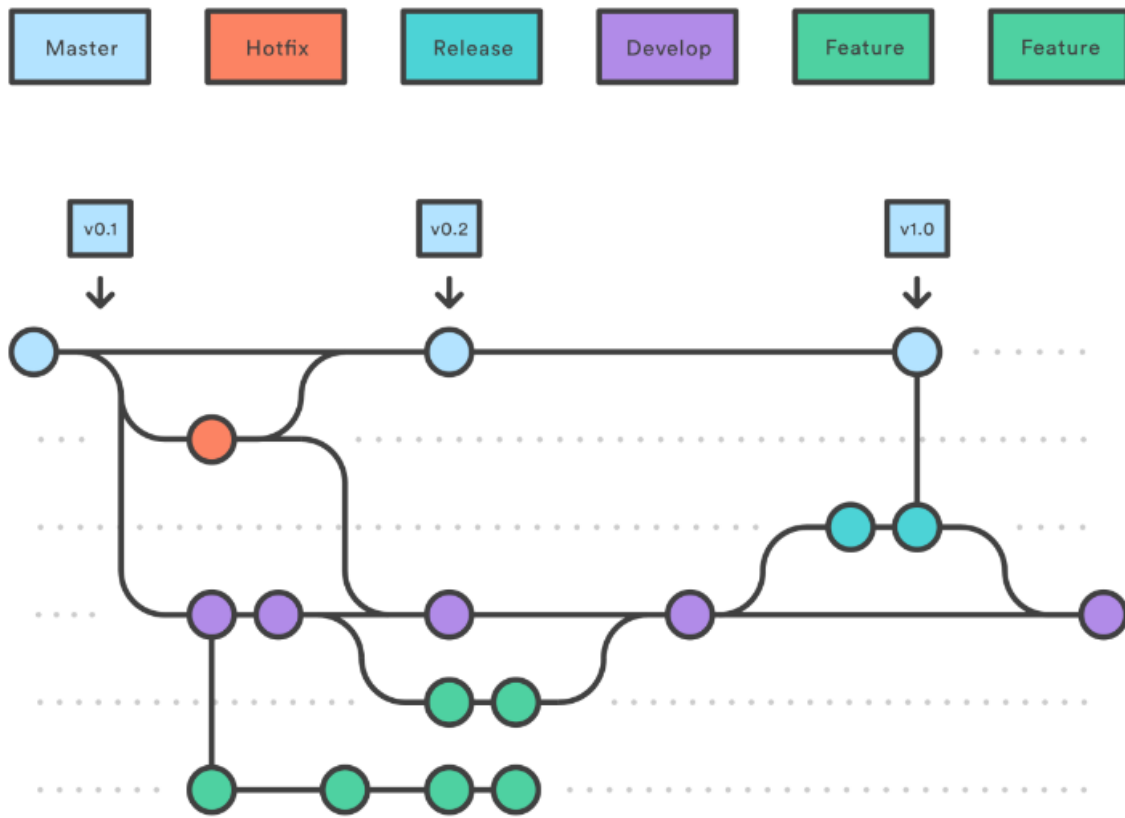


Figure 4.8 Representation of BitBucket workflow branches

4.4.4 Jira Software

The most popular software development tool which is used in Agile development is Jira Software. It provides project management functions and issue tracking. There are many rich planning features that are provided by Jira Software such as Kanban, Scrum, and Mixed methodology, which further enhance the flexibility in planning. The major use of Jira is to keep a track of the process of development with accurate estimations that help the developers to become more efficient and accurate. The order of issues, bugs and stories can be prioritized and arranged in different levels in the product backlog. One more feature

Jira has is extensive reporting functionality that provides very critical insight into the process of agile development [9].

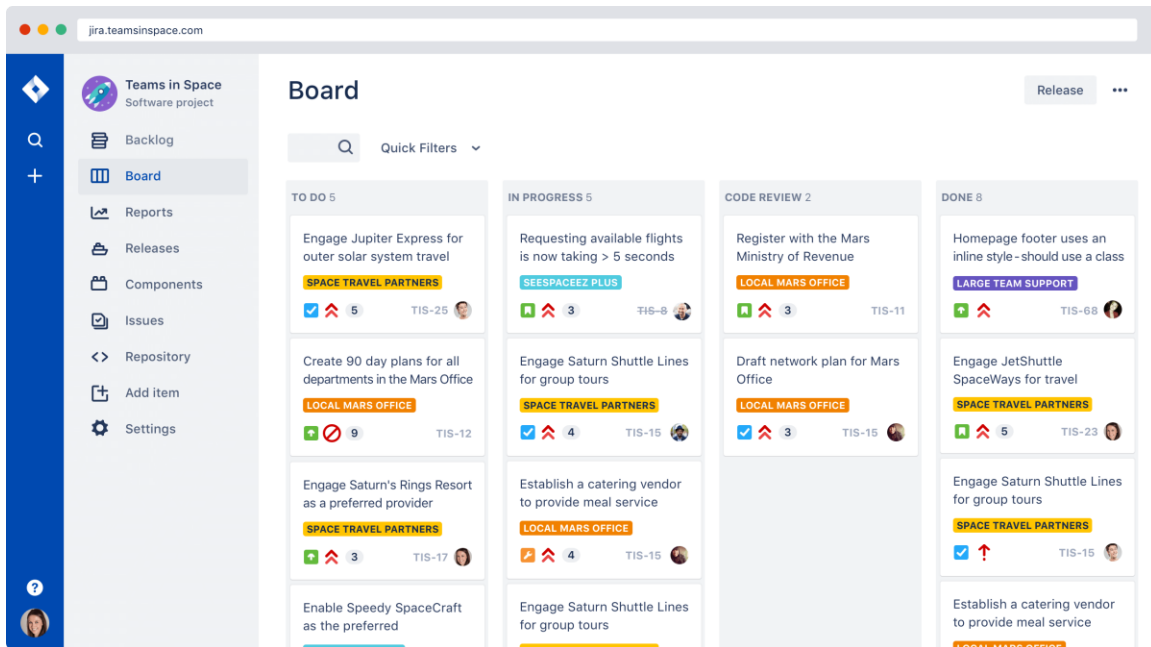


Figure 4.9 Jira software issue and project tracker

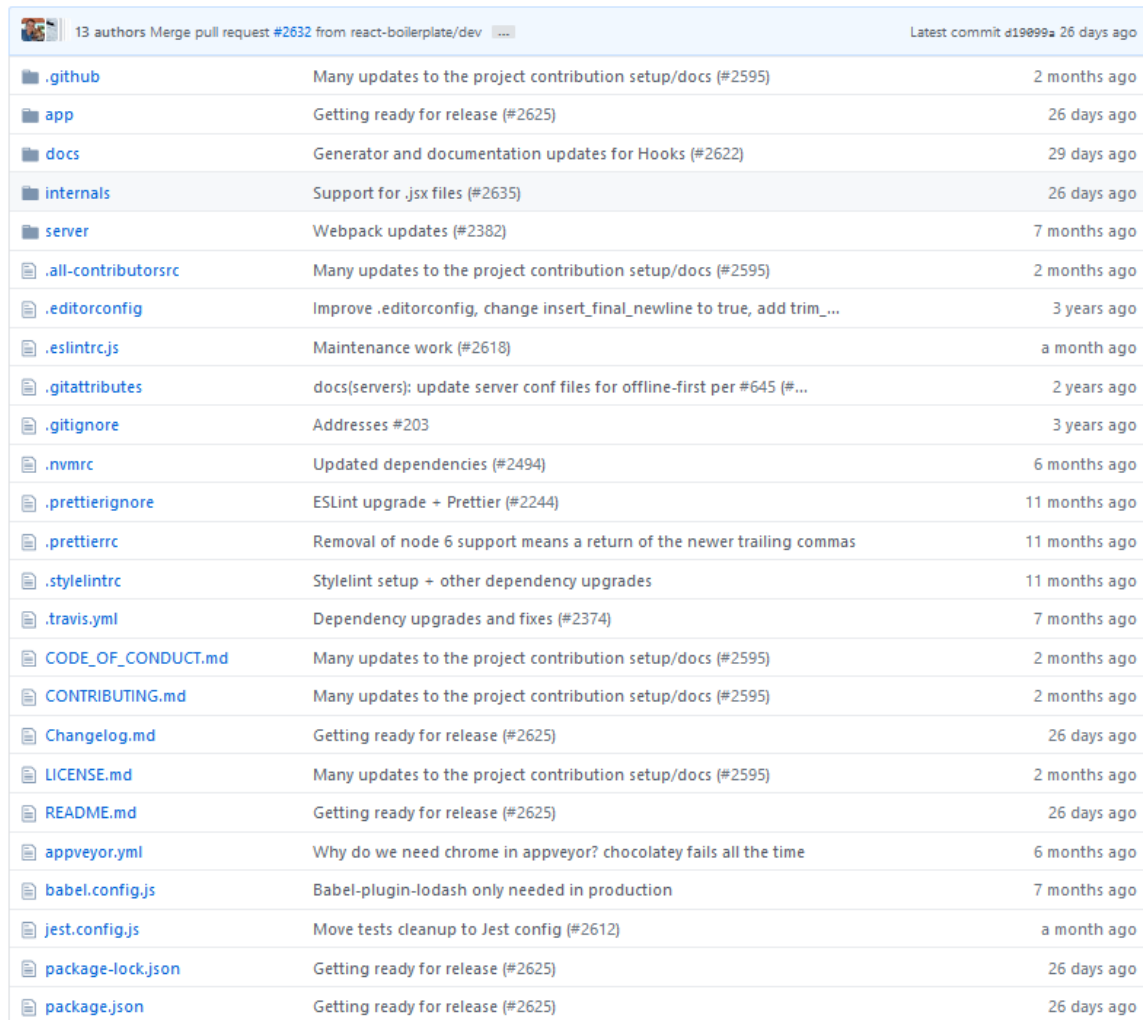
4.5 Development process

When starting to build a project in React, we need a boilerplate. In programming, the term boilerplate code is used to refer blocks of code that are used over and over again. When the development stack needed to build the project consists of several libraries, it becomes necessary to initialize all these libraries and configure them to work together in a smooth way. Every time you start with a new project, you will be repeating yourself. This may also lead to inconsistencies in how the libraries are set up. A lot of confusion will be caused when switching between projects.

In such situations, boilerplate comes in. It is a template that can be cloned and reused for every new project you start.

4.5.1 React-boilerplate

React-boilerplate is a quick setup for new offline-first, performance oriented ReactJS applications. It used Redux to take control of our application's state. It is available for cloning at GitHub¹. The development of the project started on this boilerplate. The folder structure of react-boilerplate is shown below.



File/Folder	Commit Message	Time Ago
.github	Many updates to the project contribution setup/docs (#2595)	2 months ago
app	Getting ready for release (#2625)	26 days ago
docs	Generator and documentation updates for Hooks (#2622)	29 days ago
internals	Support for .jsx files (#2635)	26 days ago
server	Webpack updates (#2382)	7 months ago
.all-contributorsrc	Many updates to the project contribution setup/docs (#2595)	2 months ago
.editorconfig	Improve .editorconfig, change insert_final_newline to true, add trim_...	3 years ago
.eslintrc.js	Maintenance work (#2618)	a month ago
.gitattributes	docs(servers): update server conf files for offline-first per #645 (#...	2 years ago
.gitignore	Addresses #203	3 years ago
.npmrc	Updated dependencies (#2494)	6 months ago
.prettierrc	ESLint upgrade + Prettier (#2244)	11 months ago
.prettierrc	Removal of node 6 support means a return of the newer trailing commas	11 months ago
.stylelintrc	Stylelint setup + other dependency upgrades	11 months ago
.travis.yml	Dependency upgrades and fixes (#2374)	7 months ago
CODE_OF_CONDUCT.md	Many updates to the project contribution setup/docs (#2595)	2 months ago
CONTRIBUTING.md	Many updates to the project contribution setup/docs (#2595)	2 months ago
Changelog.md	Getting ready for release (#2625)	26 days ago
LICENSE.md	Many updates to the project contribution setup/docs (#2595)	2 months ago
README.md	Getting ready for release (#2625)	26 days ago
appveyor.yml	Why do we need chrome in appveyor? chocolatey fails all the time	6 months ago
babel.config.js	Babel-plugin-lodash only needed in production	7 months ago
jest.config.js	Move tests cleanup to Jest config (#2612)	a month ago
package-lock.json	Getting ready for release (#2625)	26 days ago
package.json	Getting ready for release (#2625)	26 days ago

Figure 4.10 React-boilerplate folder structure(GitHub)

This is the basic structure of any development project using React. All the project development is done inside the /app folder. The good thing with React is that you can structure your application in any way you like. It is not mandatory to follow a certain folder structure. The file structure must be in such a way that when your client wants to add new

features or remove some features, you should be able to update the project without much increase in complexities.

4.5.2 Folder Structure

The folder structure inside /app folder that is used in this project is shown below.

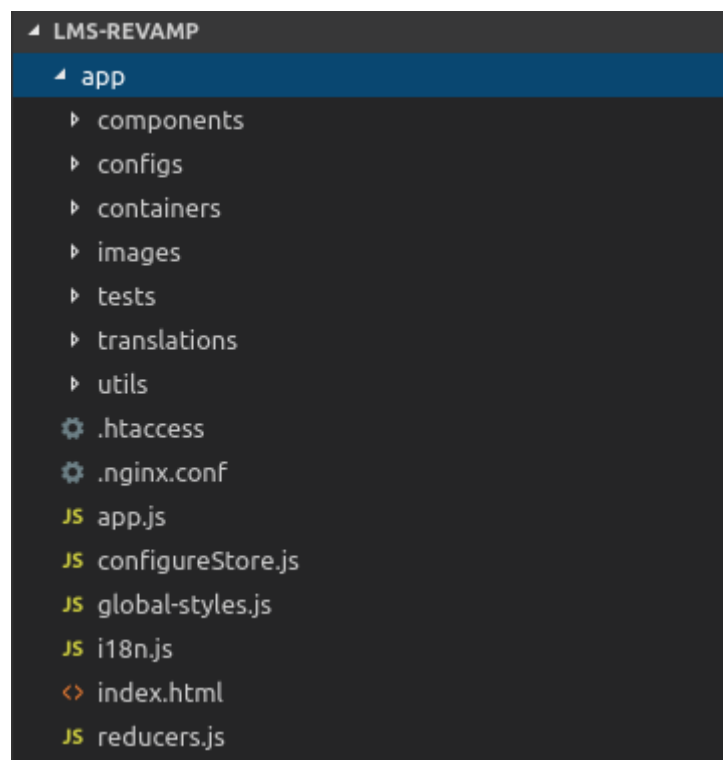


Figure 4.11 /app folder structure

Below are listed major folders within /app and their contents and use are briefly described:

- /components – This folder contains all the components that have been used throughout the project. Each component or feature has everything it needs to work on its own styles, set of actions, translations as well as unit or integration tests. This feature can be seen as an independent piece of code that you might need to reuse at different places in the project. The needed component can be imported wherever

required from this folder. Below is the list of components that were developed during the project development.

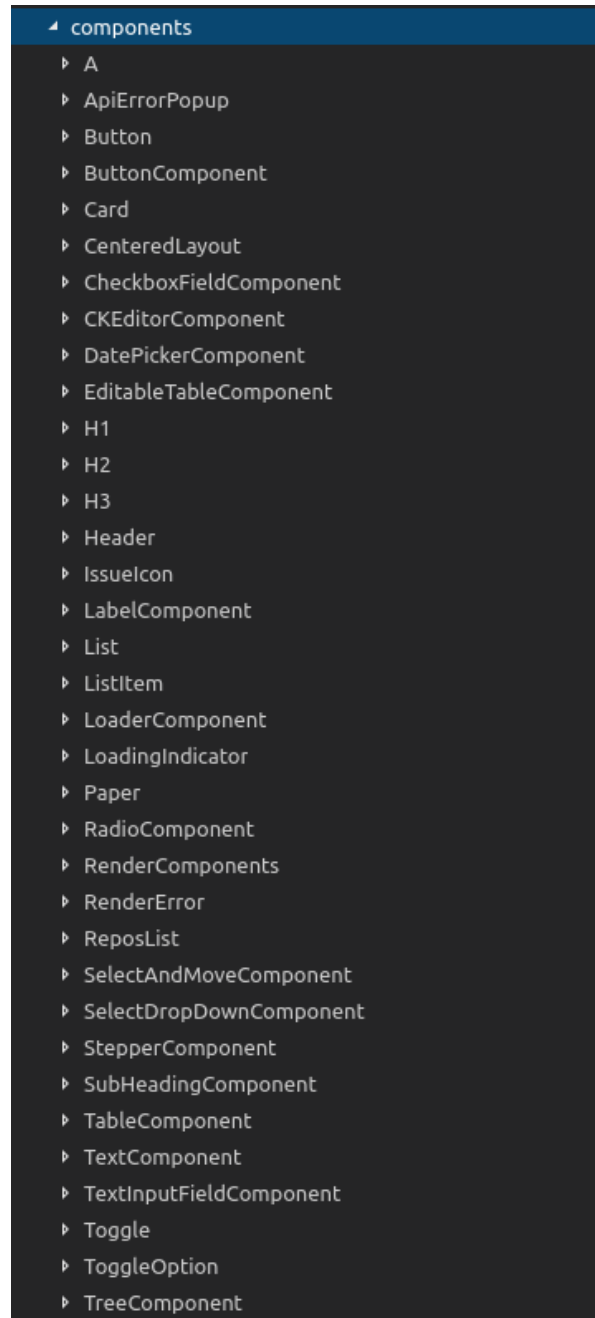


Figure 4.12 /components folder structure

- `/containers` – This folder contains all the pages developed using React. Containers receive Redux state updates and dispatch actions, and also may render DOM elements. They are basically components that are responsible for fetching data and displaying it on the screen. These components are connect-ed to a Redux store. So, you should include `connect` (react-redux) what it makes the connection, and the functions `mapStateToProps` and `mapDispatchToProps`. The containers folder structure is shown below.

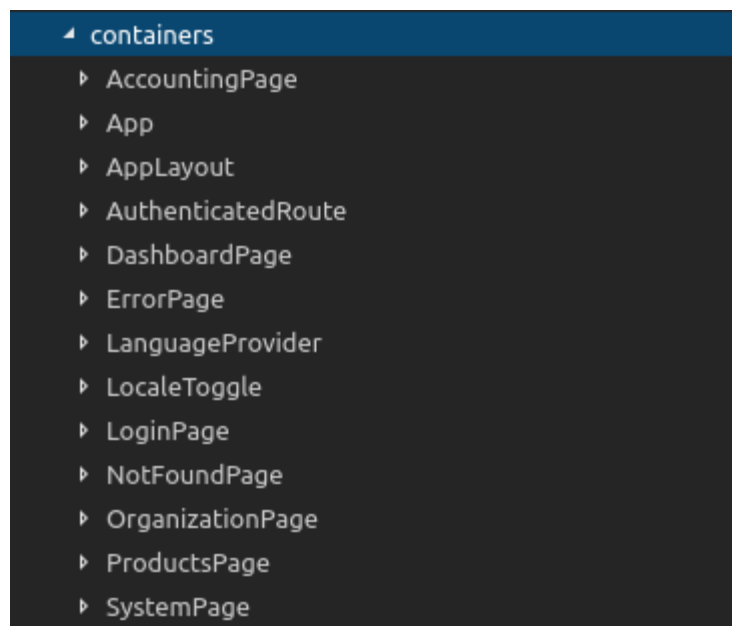


Figure 4.13 `/containers` folder structure

- `/utils` – this folder consists of utility functions that are required at multiple instances in this project. These functions have been generalized throughout the project development in such a way that they can be used anywhere with almost no modification.

4.5.3 Redux-persist

In this project, Redux-persist is used. It is used to persist and rehydrate a redux store. The root component of the project is wrapped with `PersistGate`. This delays the rendering of the app's UI until the persisted state has been retrieved and saved to redux. Why persist? Whenever app reloads, the JavaScript process is empty. The store is created from scratch and the state gets initialized. It is time consuming and expensive to re-fetch data every time the page loads. If everything is stored on disk, any given page can be refreshed in milliseconds rather than seconds. Redux-persist provides a structured, consistent, and performant way to persist state.

```
9
10
11 import { persistStore, autoRehydrate } from 'redux-persist-immutable';
12
13 persistStore(store, {
14   whitelist: ['app'],
15 });
16
```

Figure 4.14 Implementation of Redux-persist

4.5.4 Modular Development

As decided in the designing phase of the Software Development Life-Cycle (SDLC), the various the project was divided into these four modules:

- Products
- Organization
- System
- Accounting

Each of the modules and functionalities included in them were studied thoroughly and depending on their complexity, a time estimation was done for each feature inside the modules. These modules were then picked up one after other and were developed.

I was given the 'Organization' module to develop. The snapshot of the UI of this module after complete development is shown below.

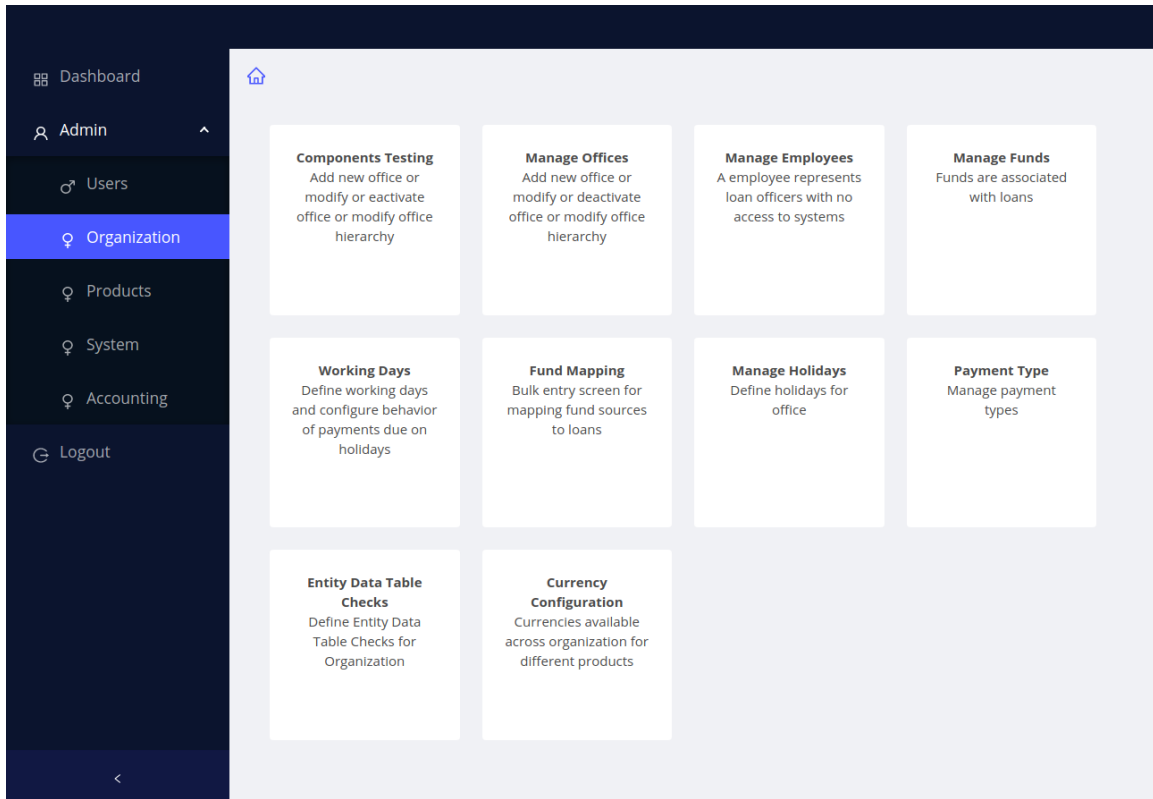


Figure 4.156 Snapshot of 'Organization' screen

Each card represents the functionalities included in the module. Every card when clicked, opens up another set of web pages depending on the functionality.

Each functionality inside these modules shared similar folder structure. The following image is the folder structure of 'Organization'.

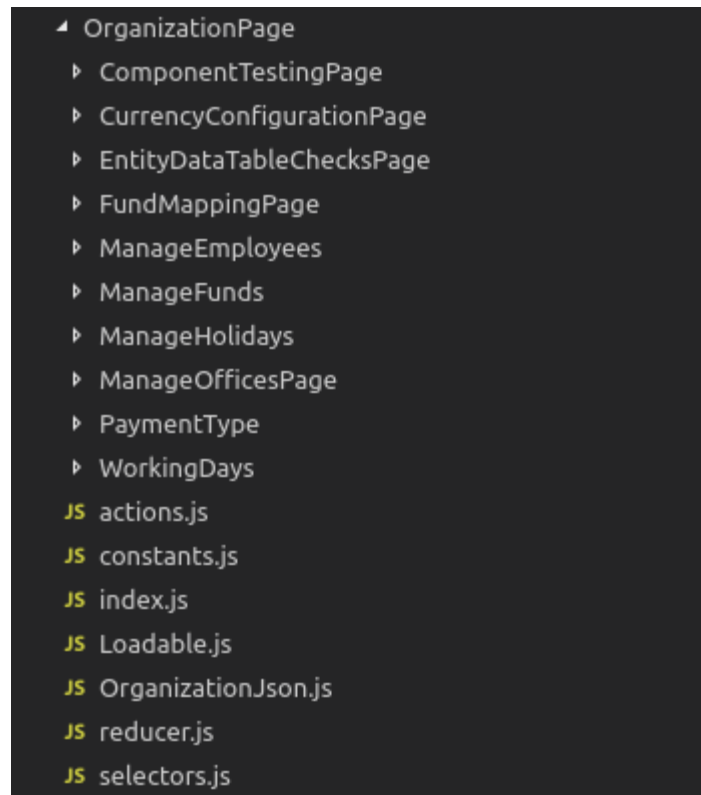


Figure 4.16 'Organization' folder structure

It took about one and a half month to complete this module. The time duration includes end-to-end development from starting the module from scratch, developing each functionality one after other, the detection and removal of bugs.

Chapter 5
TESTING

Testing is the process of confirming that the web program works as it was intended. Each feature is checked for defects in this stage. The product defects are reported, tracked using the Jira software and is fixed by the developer who is responsible for the defected feature.

5.1 Test Plan

Specifically, testing will consist of the following phases:

- Unit and integration level – adherence to coding standards and successful communication between units.

5.1.1 Test Items

Test items – Image, tables, radio button, checkbox, horizontal line, text input field, label, editable table, table, drop down, button.

5.1.2 Features to be tested

The below listed features will be tested:

- Accessibility
- Coding standards
- Security
- Usability
- Functionality
- Navigation
- Scalability

5.1.3 Features not to be tested

The intent is that all of the individual test cases that are contained in the test plan will be performed.

However, in case time does not allow, some of the low priority test cases may get dropped.

5.1.4 Strategy/Approach

The approach we followed for the testing of this project is risk-based testing. In this, each test case prepared is prioritized as, High, Medium, or Low priority and based on the categorization they are scheduled (High priority goes first). This general rule might include instances that may be treated as exceptions:

- Using small amount of resources, a large number of low priority test cases can be executed.
- A higher priority test case is preceded by a lower priority test case. For example, many of the inexpensive low priority navigational tests have to be passed necessarily before an expensive and high priority usability test.

Manual testing will be used.

5.1.5 Item Pass/Fail Criteria

The entrance criteria for each phase of testing must be satisfied before the next phase is commenced. The project manager has the power to make the decision whether the total or criticality of any or all detected bugs and defects warrant any delay in the product completion.

5.1.6 Test Deliverables

The following documents have to be generated as a result of the testing activities carried out:

- Master Test Plan (MTP)
- Deliverable documents: test plan
- Test input and output data (Test cases)

5.2 Unit and Integration Test Plan

5.2.1 Introduction

This testing phase uses multiple number of testing techniques. The decision as to which testing technique(s) is appropriate to use for any given code unit is resided by the team leader responsible for signing-off the module.

5.2.2 Features and functions to test

- Accessibility

UIAC1 - Low

The colors used in this Web page are friendly to color blind viewers.

- Coding Standards

Each of the coding units that together make the code module being tested must be coded following all the coding standards, any deviations from the standard must be prior documented and approved by the project manager.

UICS1 – Low

Any pop-up used in the application must meet following requirements:

- The pop-up follows the GUI standard
- The pop-up is not too large for its parent window
- The pop-up should have an appropriate initial position on the screen

UICS2 – Medium

All the forms that we have used in the application must necessarily meet the following requirements:

- All validations are performed (and all error messages are rightly displayed) in top-down, left-to-right fashion of the field.
- Using the Tab key allows the user client to tab through the available data input fields in a top to bottom, left to right order.
- When the form is first viewed, the browser places the cursor on the best suited control/field.
- If radio buttons are used, a default selection is always done.
- All the input fields in the form are validated for invalid data and corresponding error messages are displayed.
- All data entry fields are checked to ensure that they accept valid values and that the checking routines used can handle the invalid data appropriately.

UICS3 – High

The below stated design and syntax requirements must be followed by the code:

- Each unit of code has to be copied or inherited from the most appropriate object template and class.
- The internal workings of the program code must not be described by the error messages.
- The coding standards of the W3C HTML standard must be followed by the HTML code and it must be validated via the W3C validation service.

- Security

UISC1 – High

The input data that is received from the user client must be parsed so that it is made sure that no inappropriate meta-character sequences are contained. For example, & &.

UISC2 – High

The input data that is received from the user client must be parsed so that it is made sure that no “buffer overflow” or “out of bounds” input data is contained.

- Usability

UIUB1 – Low

Related information on the web page is grouped together on the web page to minimize the user’s eye movement.

UIUB2 – Low

The mandatory data input fields must be flagged with some visual symbol. For example, an asterisk (*) sign.

UIUB3 – Low

When the client views the product on via his/her device, it fits perfectly

- Functionality

Cut, copy, paste and zoom.

Menu bar options.

- Navigation

UING1 – High

All the links that are used on the web page will be checked to make sure they meet following specifications:

- The link should go to the appropriate location.
 - The link should not be broken.
 - The link must have an associated “Title” tag.
 - Lowercase characters for the address must be used in the internal links.
 - The link should not alter the browser’s default colors of the link.
- **Compatibility**
 - UICP1 - Medium
 - The window size of this application can be resized automatically according to the screen size of the device the user is viewing it on.

 - UICP2 - Medium
 - The content of this application is clearly readable.

5.2.3 Features and functions not to be tested

The notable features and functions that will not be tested include: None

Chapter 6
RESULTS AND PERFORMANCE ANALYSIS

This section provides an assessment on the result of the project based on the defines requirements that have been mentioned earlier in the project objectives.

6.1 Functionalities

6.1.1 Authentication

The authentication section has achieved all the requirements mentioned. Every web page request made must first authenticate the user or else the user gets directed to the login page. Additional requirements that were achieved include:

- Authentication tokens are saved using cookies.
- Users are required to have an account to be issued an access token and a refresh token. Invalid accounts will show the response message error description. Username in-inputs remove capitalize letters and spaces when the request is sent.

6.2 Qualities

The quality requirements of the project have been achieved. Quality check is done by the project leader with each new implementation in the modules.

The LMS has achieved the following quality requirement:

- The system should have an attractive, uncluttered, simple user interface with readable font-size and font-style.
- The system must not cause any crash or freezes to the browser. There must not be any shuttering while scrolling and the loading speed of each view are reduced to a minimum. React greatly reduces the loading time for each section as it only loads the parts that have been modified. Webpack provides the advantage of bundling modules into a minified file which improves performance.
- It should be responsive on both the web browser and the mobile browsers.

Chapter 7
CONCLUSION

7. Conclusion

In conclusion, the Loan Management System project (Front-end Development) was a success. The LMS achieved all the expected functions and quality requirements. With each implementation, a considerable effort was made to enhance extensibility. As an outcome, the end-product had high flexibility, scalability, modifiability and maintainability. Deployment and maintenance will be done in the upcoming future and is outside of the scope of this project.

In total, four modules were built and they are already in demo stage. They are already demoed to the clients. Deployment and maintenance will be done in the upcoming future and is outside of the scope of this project.

While the company is satisfied with the final project result, there were still many aspects that can be optimized. In my opinion, there is always a room for potential improvements in functional logic, code structure, user experience design application performance. I will continue to do code refactoring to reduce complexity, improve readability and maintainability.

REFERENCES

1. Bass, L., Clements, P. & Kazman, R. Software Architecture in Practice. 2nd ed. Boston: Addison Wesley. 2003
2. Ivanecky, N. 2016. Crash Article in Agile Development. Medium. Accessed 2018. <https://medium.com/open-product-management/crash-article-in-agile-development-da960861259e>
3. React. No date. Accessed 2018. <https://reactjs.org>
4. React. No date. React-component. Accessed 2018. <https://reactjs.org/docs/react-component.html>
5. JSX. 2014. JSX Specification. Accessed 2018. <https://facebook.github.io/jsx/>
6. MDN web docs. No date. Introduction to the DOM. Accessed 2018. https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction
7. Redux. No date. Accessed 2018. <https://redux.js.org/>
8. GitHub 2017. Usage. Webpack. Accessed 2018. <https://github.com/webpack/docs/wiki/usage>
9. Atlassian. No date. Accessed 2018. <https://www.atlassian.com/software/jira>
10. VSCode. 2019. Accessed 2019. <https://code.visualstudio.com/>
11. Geary D. 2016. Manage State with Redux-Part 1, IBM developerWorks. Accessed 2018. <https://developer.ibm.com/tutorials/wa-manage-state-with-redux-p1-david-geary/>
12. Crockford, D. JavaScript: The Good Parts. 1st ed. O'Reilly Media: O'Reilly. 2008