ONE TIME LINKS INTEGRATION IN TO MR. COOPER WEBSITE

A PROJECT REPORT

Submitted by

CB.EN.U4CSE15522 D. KRISHNA CHAITANYA VARMA

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

AMRITA SCHOOL OF ENGINEERING, COIMBATORE

AMRITA VISHWA VIDYAPEETHAM

COIMBATORE 641 112

June 2019

AMRITA VISHWA VIDYAPEETHAM

AMRITA SCHOOL OF ENGINEERING, COIMBATORE, 641112

BONAFIDE CERTIFICATE

This is to certify that the project report entitled **"ONE TIME LINKS INTEGRATION IN TO MR. COOPER WEBSITE"** submitted by D. Krishna Chaitanya Varma (CB.EN.U4CSE15522) in partial fulfillment of the requirements for the award of the Degree **Bachelo**r **of Technology** in **Computer Science and Engineering** is a bonafide record of the work carried out under our guidance and supervision at Department of Computer Science and Engineering, Amrita School of Engineering, Coimbatore.

PROJECT GUIDES

Mr. Yogesh Srinath H
Lead Engineer
Mr. Cooper
.

Dr. S Thangavelu
Assistant Professor
Dept. of Computer Science and Engg.

CHAIRPERSON

Dr. P.N. Kumar
Dept. of Computer Science and Engg.

This project report was evaluated by us on :……..

INTERNAL EXAMINER

EXTERNAL EXAMINER

TABLE OF CONTENTS

ACKNOWLEDGEMENT

We express our gratitude to our beloved **Satguru Sri Mata Amritanandamayi Devi** for providing a bright academic climate at this university, which has made this entire task appreciable. This acknowledgement is intended to be a thanks giving measure to all those people involved directly or indirectly with our project. We would like to thank our Vice Chancellor **Dr. Venkat Rangan. P** and **Dr. Sasangan Ramanathan** Dean Engineering of Amrita Vishwa Vidyapeetham for providing us the necessary infrastructure required for completion of the project.

We express our thanks to **Dr. P.N. Kumar**, Chairperson of Department of Computer Science Engineering, Dr. P Bagavathy Sivakumar and Prof. Prashant R Nair, Vice Chairpersons of the Department of Computer Science and Engineering for their valuable help and support during our study. We express our gratitude to our guides, **Dr. S Thangavelu,** for his guidance, support and supervision.

We feel extremely grateful to **Dr. D Venkataraman, Ms. K Nalina Devi, Mr. C Arun Kumar and Ms. R Manjusha** for their feedback and encouragement which helped us to complete the project. We also thank the staff of the Department of Computer Science Engineering for their support.

We would like to extend our sincere thanks to our family and friends for helping and motivating us during the course of the project. Finally, we would like to thank all those who have helped, guided and encouraged us directly or indirectly during the project work. Last but not the least, we thank **God** for His blessings which made our project a success.

ABSTRACT

Currently in any website if a user or a customer wants to see their details or any statements, they have to Sign in where the Sign in option will be available only for the Digital customers and Non-Digital customers can't use the website. There are many websites in which users visit them only twice or thrice in a year and in that case being a Digital customer is unnecessary and the customers can also forget their password if they visit only twice or thrice in a year. So, by introducing One-Time Links any user can access the website any time and see their statements, details and etc. Even for the Digital users there will be different modes of entering into the website apart from Sign in.

The main objective of the project is to provide a customer with One-Time links when he has requested. So that he can access the website using the link generated. Initially, when the customer requests for the one-time link, his data will be stored as a record in to the COSMOS DB and it will generate a unique ID and this unique ID generated will be sent as a one-time link. The other objective of this project is to enable the paperless mode of sending the customers documents in order to reduce the paper wastage and save trees.

Although, there are many websites which provide different solutions to this problem, One-time Links will provide a better customer experience and security by authenticating customers. COSMOS DB which is being used has many benefits and also provide better experience to both the customers and the developers because of its availability and developers can query it using any DB language. It provides low latency reads and writes. Also, by enabling paperless, customer can see the requested documents in a less amount of time rather than waiting for so many days.

LIST OF FIGURES

LIST OF ABBREVATIONS

TDD	Test Driven Development
MVC	Model View Controller
ROR	Ruby on Rails
DOM	Document Object Model
REST	Representational State Transfer
API	Application Programming Interface
HTML	Hyper Text markup Language
CSS	Cascading Style Sheet
CURD	Create, Read, Update and Delete
HTTP	Hyper Text Transfer Protocol
SOAP	Simple Object Access Protocol
WSDL	Web Service Definition Language
SQL	Standardized query language
YAML	Yet Another Markup Language
JS	JavaScript
JSX	JavaScript XML
XML	Extensible Markup Language
CLI	Command Line Interface
OTP	One Time Password
SSN	Social Security Number
J2EE	Java 2 Platform, Enterprise Edition

.NET	Network Enabled Technologies
DB	Database
IVR	Interactive Voice Response
SMS	Short Message Service
ID	Identity Document
USAA	United Services Automobile Association
UWM	United Wholesale Mortgage
VU	Veterans United

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

The banking industry is one of the oldest and has ever been growing to new heights. But there was not much technological development being done to optimize the process. Thus, it was a need of hour to develop modular applications to speed up the process and involving less effort. The contribution to the project was to develop an application to generate one-time links. One-time links are sent to customers during IVR or agent interaction via email or SMS to perform a specific task on web page. Through One-time links customers can easily can see their monthly statements, tax statements instead of logging in. Also, paperless initiative has been introduced for the arbor day so that instead of sending the customer's monthly statements, tax statements etc., by post or fax, directly they will be sent to the customer's mail or phone in order to reduce the paper wastage and save trees. Since in the project designing new applications are error-prone, unit test and TDD approaches were used.

1.2 PROBLEM STATEMENT

Generating one-time links so that customer can complete the transaction outside of the call by authenticating themselves. Also, enabling paperless mode of sending the documents to the customer.

1.3 SPECIFIC OBJECTIVE

In the project, TDD approach has been used for the application developed, so that it can accommodate any number of clients as it responsive. Whenever the user requests for any of the statements, one-time link will be generated and sent to the customer's email or Phone through SMS based on the customer's selection. The data given by the customer will be verified and get inserted in to the COSOMS DB. Then the COSMOS DB will generate the unique ID and it will be sent as a link to the customer. Link sent to the customer then redirects to verification page and upon verification it will be redirected to the customer's desired page.

The application lets user to see the statements which he has requested for and also other desired statements. The 2nd application provides a user to select an option for paperless mode of communication so that customer can reduce the paper wastage and can save trees.

1.4 LIMITATIONS

The application uses media queries on one of the clients and there is an immediate UI change when the browser window is reduced in width. The meta-data is currently saved in a file and only the admin can change the generated UI. A store such as Redis must be used as cache for faster data retrieval.

The One-time link application is best suited for Mr. Cooper and UWM customers. For the other white labels USAA and VU customers, even though customer is already logged in the website, if he has requested for one-time links and entered the website through one-time links it again asks for Log In.

CHAPTER 2

LITERATURE SURVEY

The following section provides a review of the literature related to the banking process, meta-data driven development, the React library, the concept of MVVM and building apps using TDD.

A modular application is one that has loosely coupled units and have high cohesion. Using a modular application fashion makes it easy for one to develop, test, maintain and deploy. In building a web application, a very good care must be taken on how the user interface is created. Though there are lots of ways in building web application user interfaces, it is noted that experienced web application developers are usually familiar with particular “patterns”. Nevertheless, user interfaces are often still developed from scratch, even though they are built based on existing applications. This makes web application UI development a repetitive job. Adding to the complexity is that UI changes may come a lot of time during development which decreases development time.

Meta-data driven development is a very recent concept which focuses on accommodating various clients using same application by multi-tenanting it. This was initially used by a company Salesforce. They came up with a solid architecture and implementation. This concept is further taken into development and thus companies started to develop UI using meta-data. A client is identified and the elements and access levels are defined in a database for the particular client and only view needed for the client is produced and customization is offered on top of it.

In Nationstar there are many clients/in-house users who have their own responsibilities and operational methods. One such client is called ‘Apollo’. It is the front-end team for the company. Similarly, there are other clients like ‘I Assist’ for call canter and so on. Each have their own UI and implementations.

Earlier when a small change is made on UI it would not reflect to all people using the function. So, a common endpoint was needed but with capability to produce a different UI for each client. Thus, meta-data driven helps in achieving this complex requirement.

DEVELOPING APPS USING REACT

React is an Open-Source JavaScript library majorly used in many of top websites for its simplicity and modularization. It is maintained by Facebook and a community of individual developers and corporations. React processes only the View in an application. It is used fast, simple and scalable web apps. It uses a special syntax called 'JSX' which allows mixing of JavaScript and HTML in same file.

React is based on component driven architecture and support stores like Redux. It looks for a base component to start and loads it up. Since components can be nested inside another component, it is modularized and scalable. The components work on Plug-n-Play method. The store can be called only when an action occurs and action methods will call corresponding dispatcher to make changes to store. The components can subscribe to a store and the components will automatically re-render based on changes made in store.

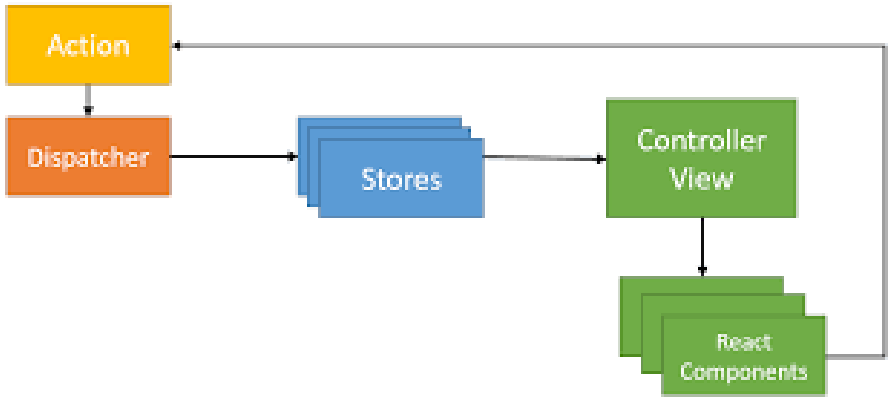

Fig. 2.1 – A generic functional flow of react apps

VIRTUAL DOM

Another interesting feature of react is the concept of Virtual DOM. The normal DOM is tree-structured and traversing trees is fairly easy. But the trees are huge nowadays. Since we are more and more pushed towards dynamic web apps, we often need to modify the tree incessantly. And this is a real performance and development hurdle.

The Virtual DOM is an abstraction of the HTML DOM. It is lightweight and detached from the browser-specific implementation details. Since the DOM itself was already an abstraction, the virtual DOM is, in fact, an abstraction of an abstraction.

Whenever an element in tree changes it internally re-renders the whole DOM and efficiently checks the difference between actual DOM and rendered DOM and only applies a patch to the actual DOM than re-rendering the whole DOM in browser.

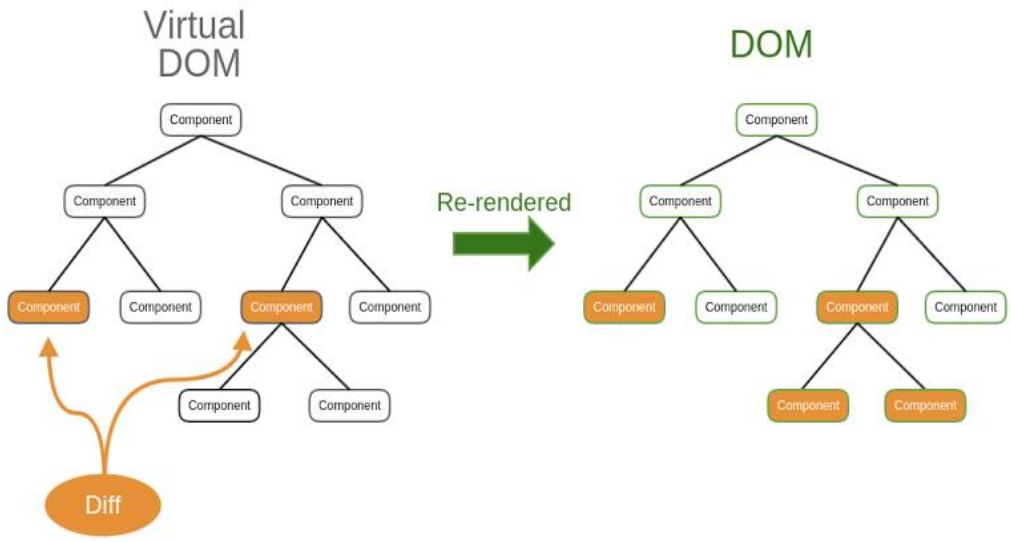

Fig 2.2 – Detection of change in Virtual DOM

DEVELOPING APPS USING RUBY ON RAILS

Ruby is a programming language. Like Java or the C language, Ruby is a general-purpose programming language, though it is best known for its use in web programming.

Rails is a software library that extends Ruby programming language. It is software code that is added to the Ruby programming language. Technically, it is a package library that is installed using the operating system command-line interface. Ruby on Rails or simply Rails is a framework for building websites. As such, Rails establishes conventions for easier collaboration and maintenance. These

conventions are codified as the Rails API (the application programming interface, or directives that control the code).

Rails combines the Ruby programming language with HTML, CSS, and JavaScript to create a web application that runs on a web server. Because it runs on a web server, Rails is considered a server-side, or back end. Rails is the central project of a vast community that produces software libraries that simplify the task of building complex websites.

Web servers deliver HTML, CSS, and JavaScript, either from static files that are stored on the server, or from an application server that creates files dynamically using a programming language such as Ruby. A software program written in Ruby and organized using Rails conventions is a web application. Rails uses Ruby to dynamically assemble HTML, CSS, and JavaScript files from component files (often adding content from a database).

Perspectives on a Rails Application

- Web Browser's Perspective
- Coder's Perspective
- Software Architect's Perspective
- Time Traveller's Perspective
- Gem Hunter's Perspective
- Tester's Perspective

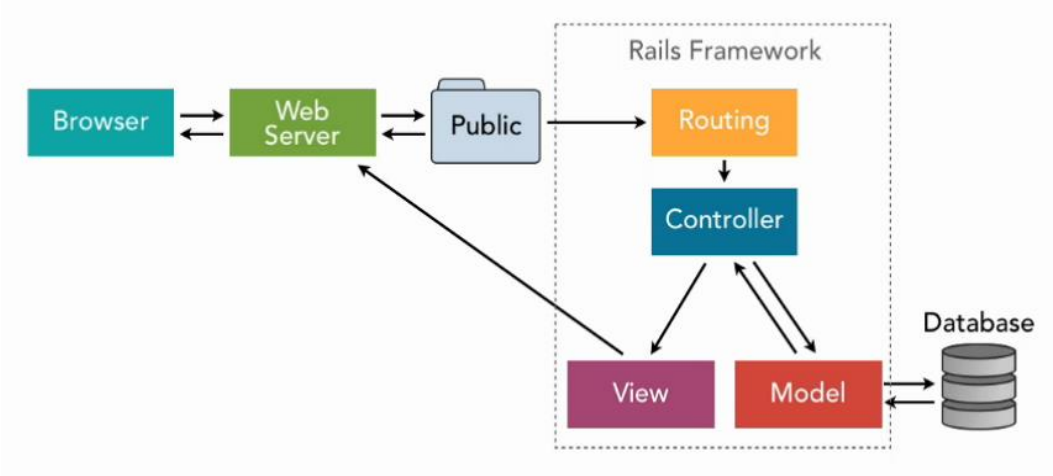

Fig. 2.3 – Flow of Rails Application

FUNCTIONING OF ISOMORPHIC APPLICATIONS

Isomorphic apps are gaining lot of attention in industry as it is used of server-side rendering and client load in heavily minimized and since server takes the load, it is faster. Moreover, without this there is a dependence on user browser that JavaScript must be enabled.

The top reasons developers develop isomorphic apps are:

- Better SEO
- Increase in performance
- Easier maintenance

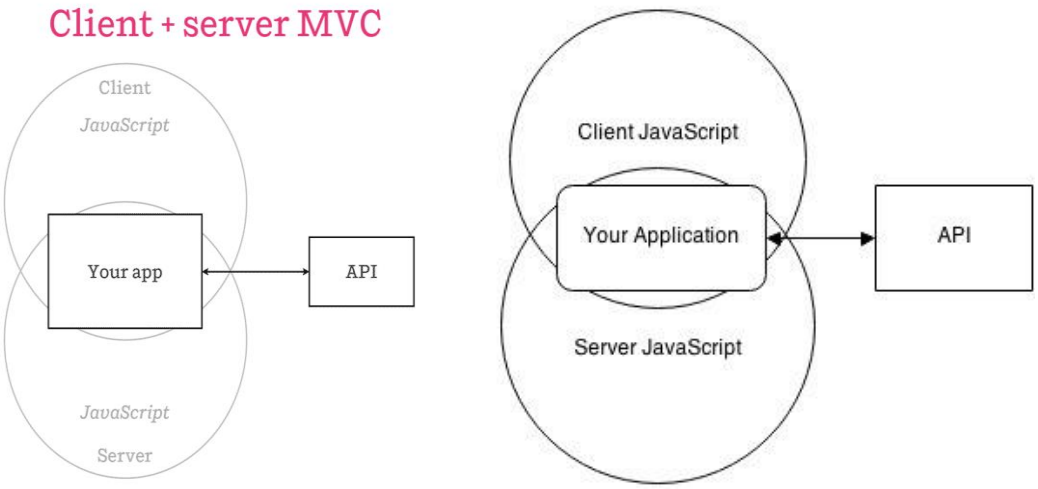

Fig 2.4 – Comparison of normal client-side app with an isomorphic application

When a page is loaded, the server provides complete HTML code of the view and it only requests for downloading the remaining dependent elements (images, stylesheets and scripts). And any further requests made are only for new resources and not HTML. This is because it bundles the whole view in a JavaScript file and this is used to change the HTML being rendered.

The drawback of such apps is that server must be built dynamic and scalable. There is also a chance that store used by react app is on server side and can cause mix ups.

Thus, a responsive and isomorphic apps can be built using these best approaches.

TESTING APPLICATION

Every component developed is worth testing to some degree, though it may be a simple test. This gives confidence that the component works as expected, even if that seems obvious at a single glance, and it gives you confidence to refactor later on.

As per Test Driven Development (TDD), the application functionality must be made as a test case even though the core functionality is not implemented as code. The code is first made to fail and is tested as the functionalities are implemented. The application is only considered complete if and only if all the test cases written passes. Thus, even if a functionality in code is changed it must be able to pass the test case written. Thus, a care must be taken test cases are perfect and all edge case is considered.

There are many frameworks which does TDD in react. But the most used packages available in NPM are Mocha and Chai. Mocha only accepts transpiled code and thus the existing code needs to be transpiled using compiler like babelJS.

The general approach is that the developer splits the setting into different environments such as development, production and testing. The development setting usually has hot loading and is less optimized for servers while production setting can have minified version of JavaScript files, stylesheets and html files. This makes the code more maintainable and organized.

The phases in which a successful testing is done is as follows:

1. Unit tests are written and verified. The key factor to be considered while writing a unit test is that all edges cases are covered and it can run independently.
2. After unit tests are successful, Integration test is written and verified if successful.
3. The performance testing helps to benchmark the application with its variants.
4. In the end a user acceptance test is done and then is moved to production.

As far as applications are concerned, most of the testing is considered under UI testing as they are more inclined towards UI. For code or functional testing,

frameworks such as NUnit can be used. The test can be run independently through their own software rather than using IDE. The NUnit not only helps in unit testing but integration testing by prioritizing test before a dependent test.

Once unit testing is done a UI automation tool like Winium is used. Selenium provides support for web apps, mobile web and mobile applications using Appium. But for windows app automation, previously we everyone used some external tools like sikuli, auto it, but now we have a Winium driver which provides support for doing windows app automation. We can write tests in any programming language which Selenium Web driver supports. Winium is basically a http client implementing the JSWP protocol. This tool is a development from web testing tool selenium.

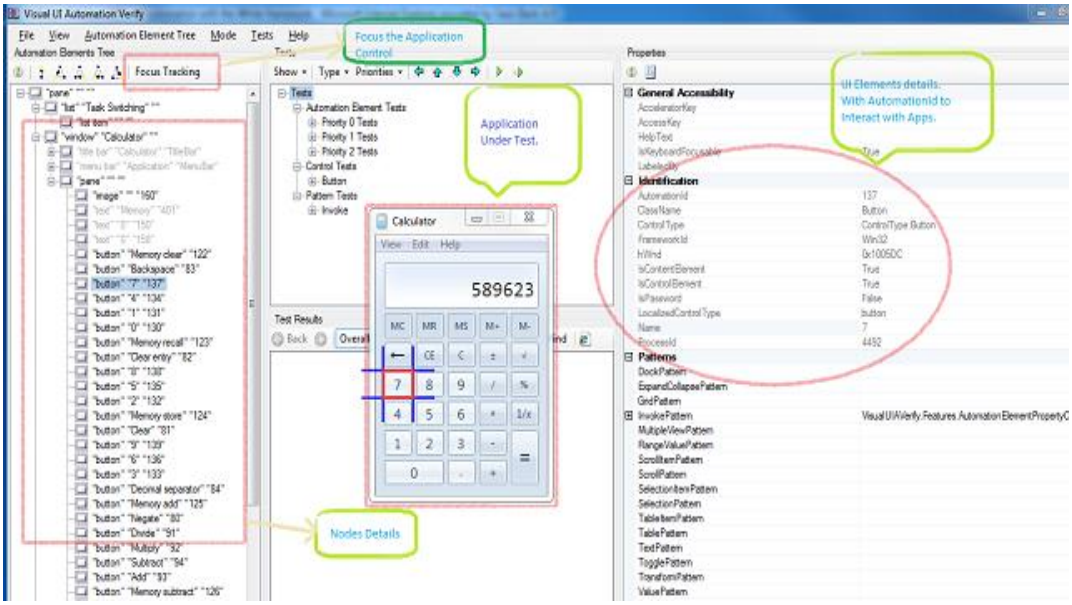

Fig. 2.5 – Visual UI automation verification

An element is first identified and its children are then traversed to get button id and related classes to call. The usual steps followed in Winium automation is

1. Winium driver should be called using its executable
2. An new session should be started at driver creation
3. Get the window class of application by method find_element_by_class_name()

4. Then we find an element in the window to click/select. Winium will simulate movement of mouse and clicks the element on calling MouseClick() method
5. We can also receive member attributes using get_attribute() method.
6. This procedure is repeated as long as the application elements are tested and verified for acceptance.

The disadvantage of using Winium is that one cannot do any other task as moving to another window will disrupt the test. Due to this it is always recommended to run tests on an isolated environment like a Virtual Machine.

CHAPTER 3

SYSTEM SPECIFICATIONS

The developed widget requires a system with the following specifications:

- Any browser including android browsers (In case of Internet Explorer – version 9 or above)
- Internet connection
- Windows 98 or above

For development of the web application a system with following requirements is needed:

- NodeJS with react package
- Any browser including android browsers (In case of Internet Explorer – version 9 or above)
- Internet connection
- IntelliJ IDEA
- Postman
- Ruby (version – 2.5.1 or greater)
- MAC

CHAPTER 4

ARCHITECTURE

4.1 BASIC ARCHITECTURE (RUBY ON RAILS AND REACT)

- **RUBY ON RAILS**

Ruby on Rails (ROR) is open source web framework written in Ruby programming language, and all the applications in Rails are written in Ruby. Ruby on rails is focused on productivity and enforces agile web development. Ruby on rails framework was designed for database-backend web applications. It was created as a response to heavy web frameworks such as J2EE and the .NET framework. In order to make development process faster, Ruby on Rails uses convention and assumptions that are considered best ways to accomplish tasks, and it's designed to encourage those. This convention eliminates configuration code and increases productivity.

Ruby on Rails architecture has the following features:

- Model-View-Controller architecture.
- REST for web services.
- Supports the major databases (MYSQL, Oracle, MS SQL Server, PostgreSQL, IBM DB2, and more).
- Open-source server-side scripting language.
- Convention over configuration.
- Scripts generators to automate tasks.
- Use of YAML machine, which is a human-readable data serialization format.

The above-described features are distributed in the following Rails components

- Action Mailer
- Action Pack
    - Action Controller
    - Action Dispatcher
    - Action View

- Active Model
- Active Record
- Active Resource
- Active Support
- Railties

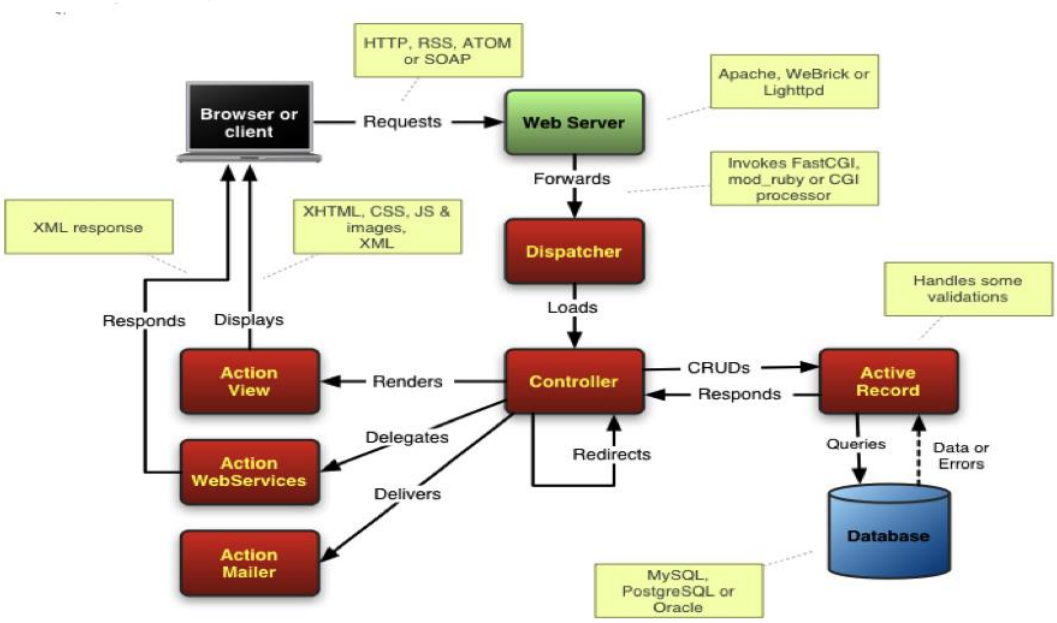

Fig. 4.1 – Reference model depicting the overall ROR framework architecture

MODEL-VIEW-CONTROLLER PATTERN

Ruby on rails users MVC architectural pattern in order to improve the maintainability of the application. The Model centralizes the business logic, the view manages the display logic, while the controller deals with the application flow. The MVC allows a clean separation from HTML views. Additionally, it improves decoupling and testing

Model

The Model layer carries the business logic of the application and the rules to manipulate the data. In Ruby on Rails, the modules are used to manage the interaction with their corresponding elements in the database. The Models represent the information in the database and do the appropriate validations.

View

The view is the front-end of the application, representing the user interface. In Ruby on Rails, views are HTML files with embedded Ruby code. The embedded Ruby code in the HTMLs is fairly simple (loops and conditionals). It is only used to display data to the user in the form of views. Views are used to provide the data to the browsers that requested the web pages. Views can server content in several formats, such as HTML, PDF, XML, RSS and more.

Controller

Controllers interact with models and views. The incoming requests from the browsers are processed by the controllers, which process the data from the models and pass it to the views for presentation.

RAILS MODULES

Action Mailer

This module is responsible for providing e-mail services. It processes incoming mails and creates new ones. This module can handle simple text or complex rich-format emails. Also, it has common tasks built-in, such as, sending out forgotten passwords, welcome messages, and fulfilling any other written-communication's need. Action Mailer is wrapped around the Action Controller. It provides ways to make email with templates in the same way that Action View uses it to render web pages.

Action Pack

The Action Pack module provides the controller and view layers of the MVC patterns. These modules capture the user requests made by the browser and map these requests to actions. These actions are defined in the controller's layer and later the actions render a view that is displayed in the browser. Action Pack is divided in 3 sub-modules, which are: Action Dispatch, Action Controller, and Action View.

- **Action Dispatch:** handles routing of web browser request. It parses the web request and does advanced processing around HTTP, such as handling cookies, sessions, request methods
- **Action Controller:** after the action dispatch has processed the request it makes the routing to its corresponding controller. This module provides a base controller from which all the other controllers can inherit. Action Controller contains actions to controls model and view. This module makes data available as needed, controls views rendering and redirection. Additionally, it manages the user sessions, application flow, caching features, helper modules and implement filters for the pre, during and post processing hooks.
- **Action View:** it is call by the Action Controller. It renders the presentation of the web page requested. Action View provides master layouts, templates lookups and view helpers that assist the generation of the HTML, feeds and other presentation formats. There are three templates schemas in Rails, which are rhtml, rxml, and rjs. The rhtml format generates HTML views to the users with ERB (embedded ruby code in HTML). The rxml is used to construct XML documents using Ruby, and rjs allow creating dynamic JavaScript code in Ruby useful to implement AJAX functionality.

Active Model

Define the interface between the Action Pack and the Active Record modules. Also, Action Record interfaces can be used outside of Rails framework to provide Object-relational mapping (ORM) functionalities.

Active Record

Active record is an architectural pattern used to manage data in relational databases through objects. In Ruby on Rails the Active Record module provides object-relational mapping to classes. This module builds the Model layer that connects the database tables with its representation in ruby classes. Rails provide tools to implement the CRUD functionality with zero-configuration. CRUD allows creating, reading, updating and deleting records from the database through ruby objects. An object represents each row in a

database table. Additionally, it also provides advance search capabilities and the ability to create relationships or associations between models. Active Records relies heavily on conventions on how the classes should be named, the tables in the database, the foreign keys and primary keys. However, the database mapping can be accomplished using configuration, but it is highly encouraged to follow the rails convention, such as active record modules.

This module is used to create model classes, which contains the business logic, handle validations and relationships, automatically maps to a table and encapsulates data access, provides getters and setters, callbacks and also supports several databases.

Active Resource

Active Resource module is used for managing the connection between RESTful web services and business objects. It follows the same principle of Active Record that is to reduce the amount of code needed to map resources. Active Resources maps model classes to remote REST resources in the same way that Active Record maps model classes to database tables. Active Resource leverages the HTTP protocol and adds code conventions to make it easy to infer complex structures and relations. Active Record also provides proxy capabilities between an Active Resource (client) and a RESTful service. This is accomplished implementing an object-relational mapping for REST web services. When a request to a remote resource is made, a REST XML is generated and transmitted, and then the result is parsed into a ruby object.

RESTful Architecture

REST is an alternative to web services, such as SOAP and WSDL. It relies in the HTTP protocol for all the CRUD operations: create, read, update and delete. RESTful web services are appropriated when the web services are completely stateless, limited bandwidth (it’s very useful for mobile devices since it doesn’t the the overhead of other protocols like SOAP), when the data is not generated dynamically so it could be cached to improve performance and when there is a mutual understanding between the service producer and the consumer.

Active Support

It is a collection of utility classes and standard Ruby libraries extensions that are useful for the development on Ruby on Rails. It includes a rich support for multi-bytes strings, internationalization, time zones and testing.

Railties

Railties is the Rails' core code that builds new applications. It is responsible for "glue"-ing all the above-describe modules all together. Additionally, it handles all the bootstrapping process, the command line interface and provides the Rails' code generators. Rake is one of the command lines used to perform database tasks, deployment, documentation, testing and cleanups. Rails also supply a built-in testing framework that generates test stubs automatically when code is generated, provides unit testing, functional testing for views and controls, test fixtures and supply test data using YAML.

- **REACT**

React is an Open-Source JavaScript library which process only the View in an application. React is based on component driven architecture and support stores like Redux. It looks for a base component to start and loads it ups. Since components can be nested inside another component, it is modularised and scalable. The components work on Plug-n-Play method. The store can be called only when an action occurs and action methods will call corresponding dispatcher to make changes to store. The components can subscribe to a store and the components will automatically re-render based on changes made in store.

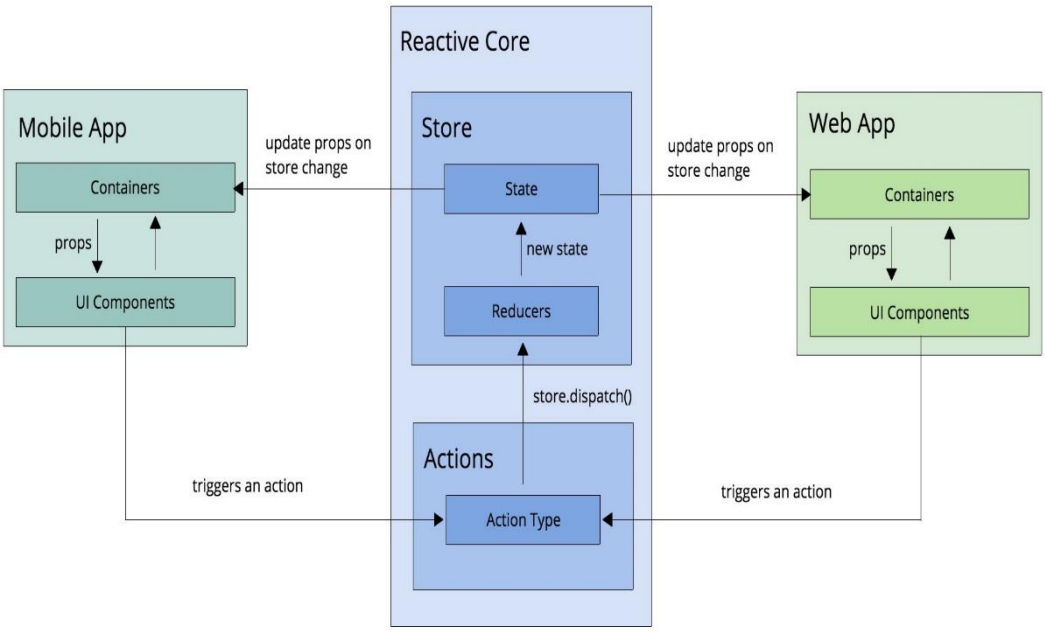

Fig. 4.2 – React Js architecture

React implements a virtual DOM that is basically a DOM tree representation in JavaScript. So, when it needs to read or write to the DOM, it will use the virtual representation of it. Then the virtual DOM will try to find the most efficient way to update the browser's DOM. Unlike browser DOM elements, React elements are plain objects and are cheap to create. React DOM takes care of updating the DOM to match the React elements. The reason for this is that JavaScript is very fast and it's worth keeping a DOM tree in it to speedup its manipulation. Although React was conceived to be used in the browser, because of its design it can also be used in the server with Node.js.

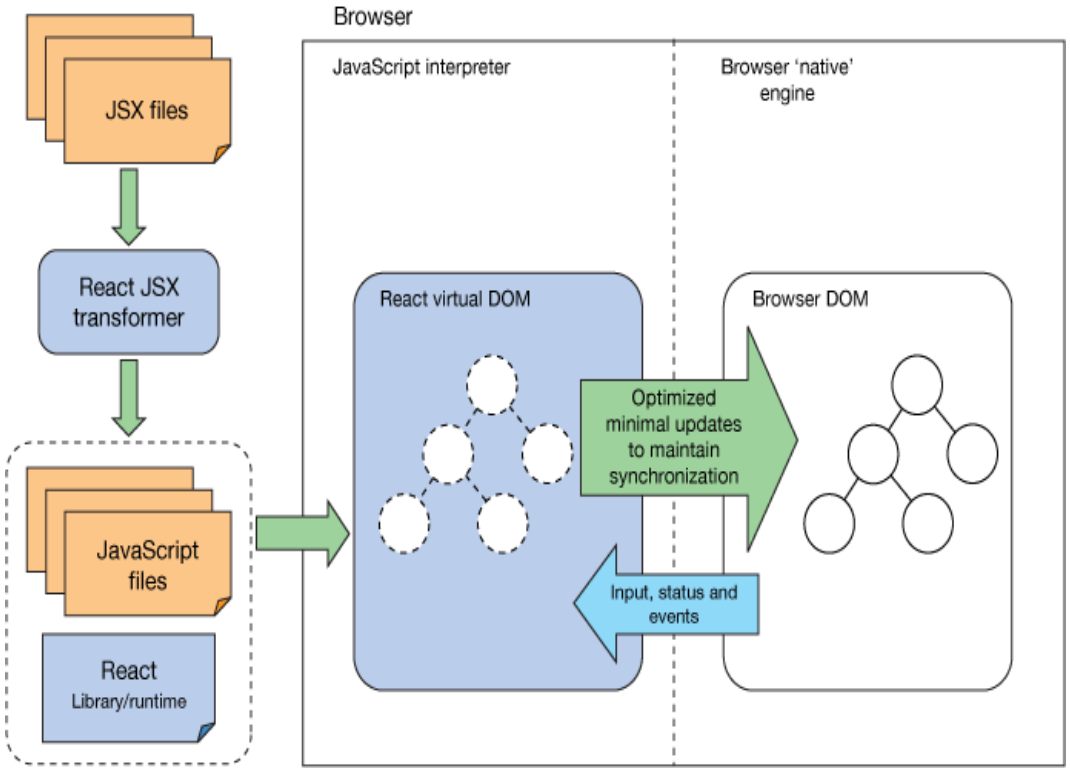

Fig. 4.3 – Working of Virtual DOM in React

Redux

Redux is a predictable state container for JavaScript apps. It is a library that acts as a state container and helps in managing application data flow. It helps to write applications that behave consistently, run in different environments (client, server, and native), and are easy to test.

In React, view components dispatch an action and the same action can be dispatched by another part of the application. This action is dispatched not to a central hub but directly to the store. Redux has only single store. The logic that decides how to change the data lives in pure functions called reducers. Once the store receives an action it asks the reducers about the new version of the state by sending the current state and the given action. Then in immutable fashion the reducer needs to return the new state. The store continues from there and updates its internal state. As a final step, the wired to the store React component gets re-rendered.

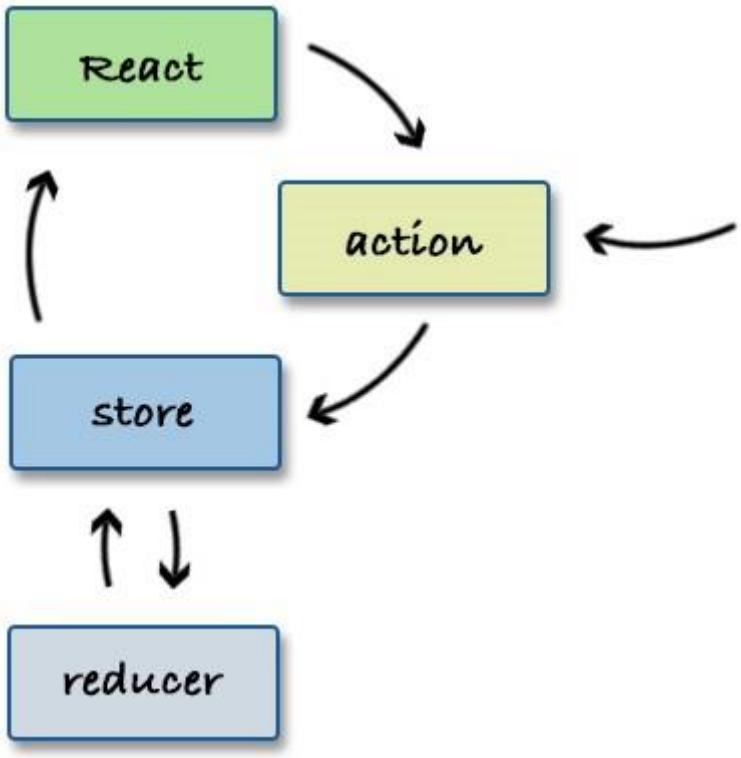

Fig. 4.4 – Redux architecture

Actions

The typical action in Redux is just an object with a type property. Everything else in that object is considered a context data and it is not related to the pattern but to the application logic.

Store

Store is a single place where the data gets stored in the Redux. After the reducer perform some logic, it updates the store and that changes in the store gets reflected in the overall application (but only to the components which have subscribed to the store).

Reducer

Reducer is a function that accepts the current state and action and returns the new state. There are two important characteristics of the reducer.

(1) It must be a pure function – it means that the function should return the exact same output evert time when the same input is given.

(2) It should have no side effects – stuff like accessing a global variable, making an async call or waiting for a promise to resolve have no place in here.

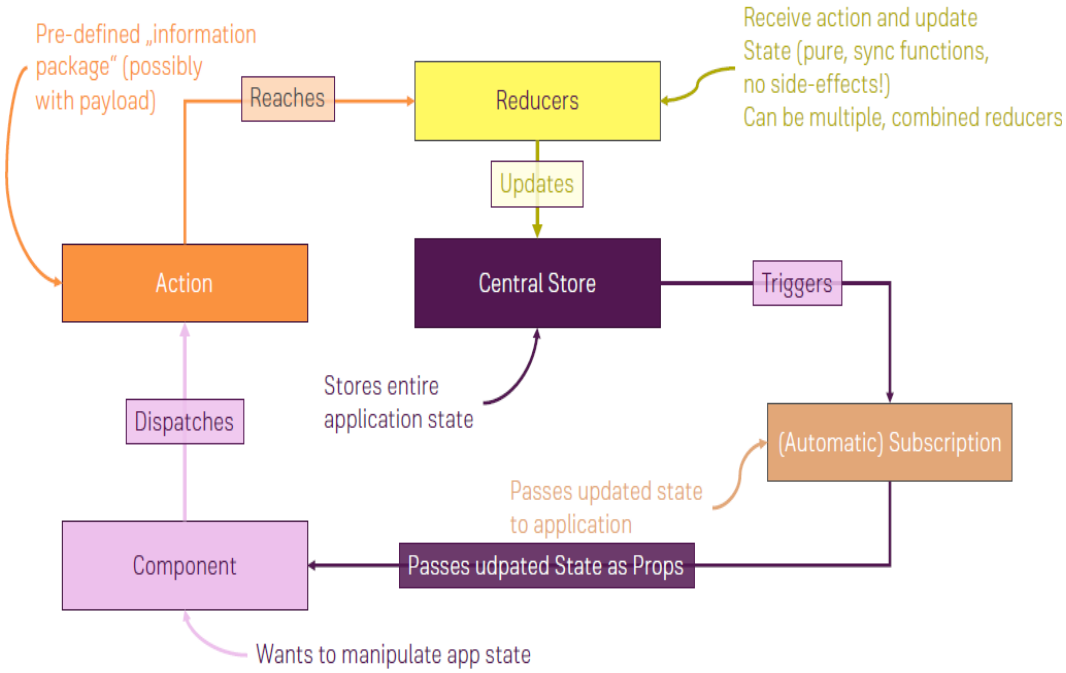

Fig. 4.5 – Redux Flow

4.2 PROJECT ARCHITECTURE

To get One-time link customer has to call IVR or iAssist.

- One-time links via IVR
- One-time links via iAssist

Both of these will provide customer with one-time link by allowing them to authenticate which leads to specific asset or task requested for.

Key Terms in the Application

Link Types:

- **Full Auth Link –** Authentication required to access the website.
- **Registration Link –** Authentication required to create website account.
- **Direct Link Access –** No authentication leads to specific task.

Auth Methods:

- **Sign In –** Username and password required
- **MFA –** Send link to customer, then one-time passcode sends via SMS.
- **Alt path –** Send link to customer, then manually enter random 4 SSN and property area zip code.
- **No Auth –** Send link to customer for direct access.

Access Levels:

- **Okta –** Full account access to website.
- **Non-Okta –** Limited account access to website.
- **Direct Access –** No authentication leads to specific task.

Link Delivery Flow

Link delivery flow is to determine which system (IVR or iAssist) is triggering the link, and based on the use-case, delivery channel is determined, and based on delivery channel right link type is triggered.

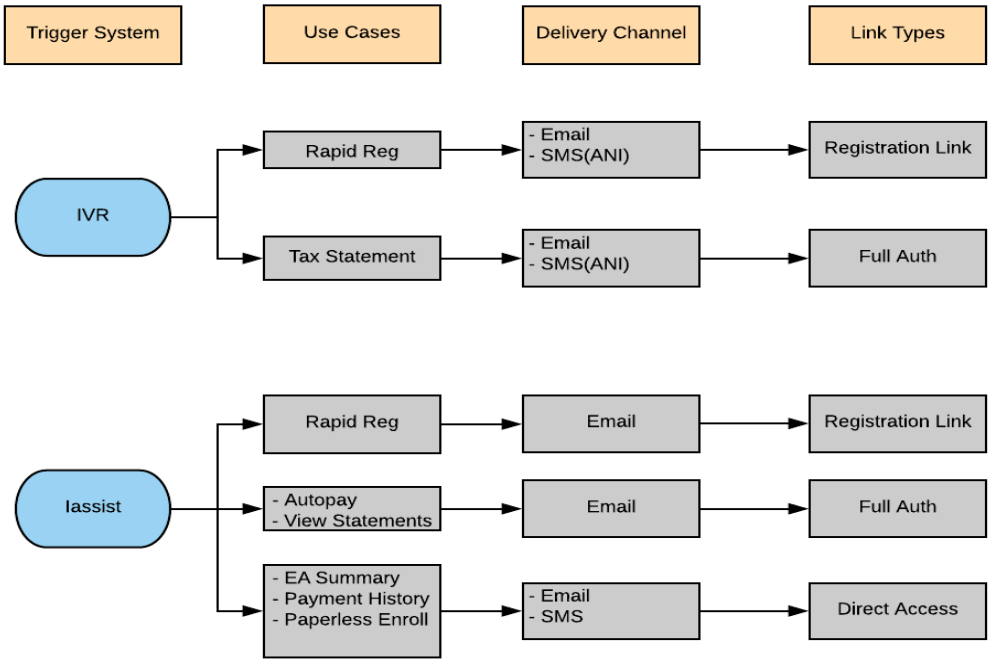

Fig. 4.6 – Link Delivery Flow

Web Interaction Flow

Web Interaction flow is to understand the website behaviour on each link type. On the basis of link type and use cases, personas (digital or non-digital) are identified, and based on person, right auth method is determined, and on the basis of authentication done by customer access level and banners are displayed.

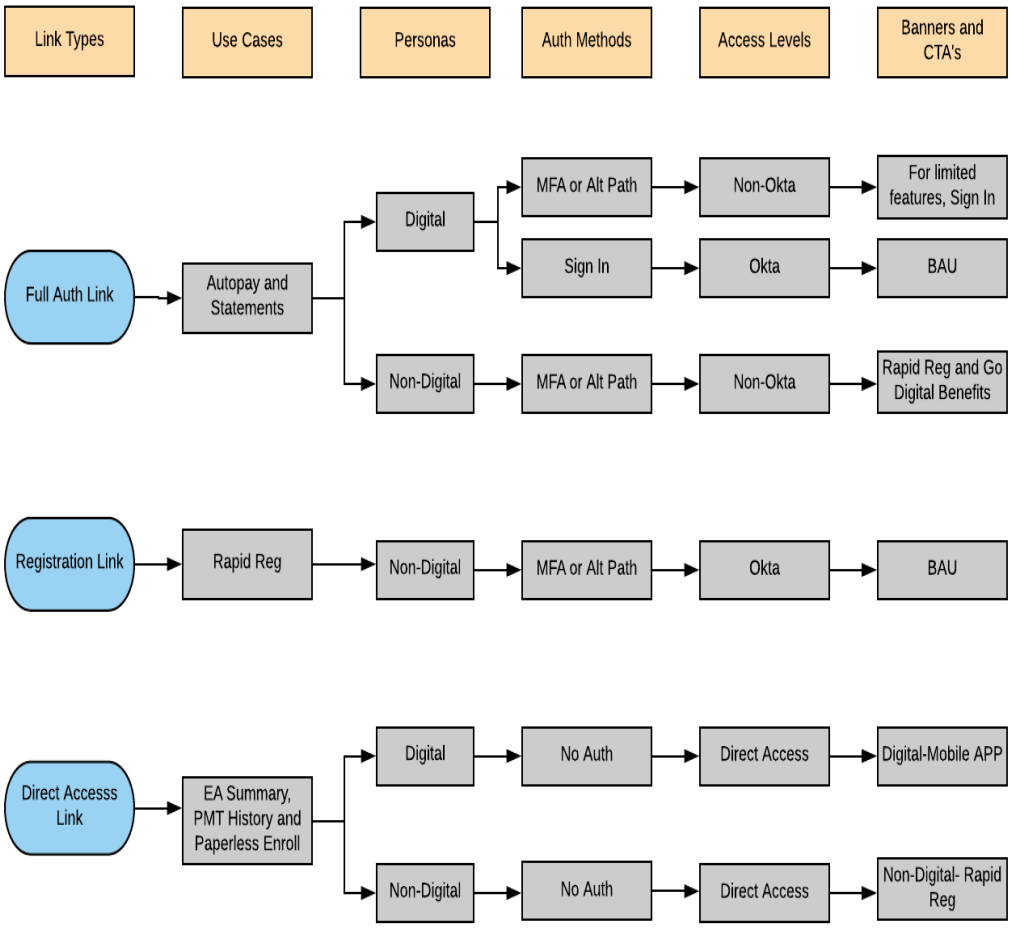

Fig. 4.7 – Web Interaction Flow

CHAPTER 5

RESULTS AND DISCUSSION

1. View of One Time Link Received on Phone Through SMS

After calling IVR or iAssist customer will get the one-time link which he has requested for so that he can authenticate himself and access the website for that particular resource type.

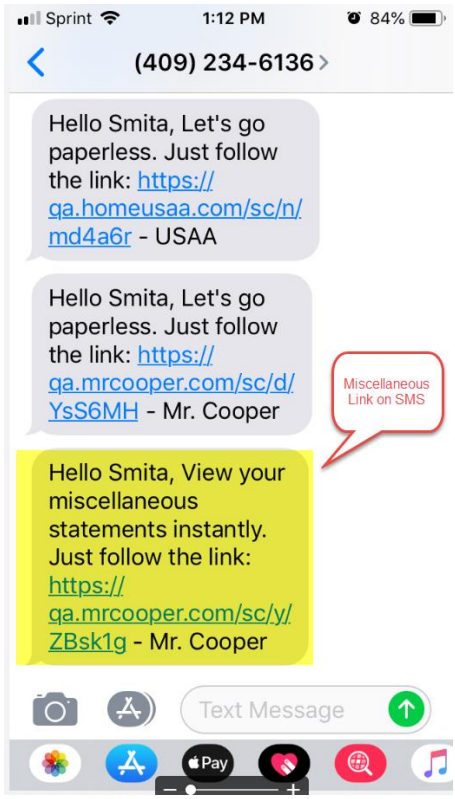

Fig. 5.1 – View of one-time link received on phone through SMS

2. View of One Time Link Received Through EMAIL

After calling IVR or iAssist customer will get the one-time link which he has requested for so that he can authenticate himself and access the website for that particular resource type.

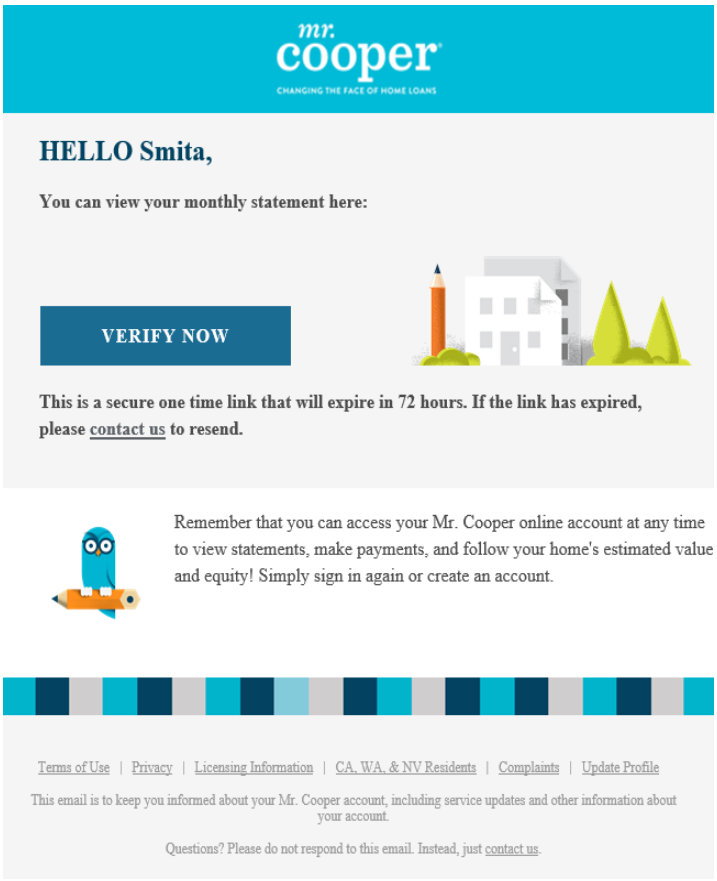

Fig. 5.2 – View of one-time link received through EMAIL

3. Verification Page View After Customer Clicks One Time Links

Customer can verify himself by different ways.

1. By Sending OTP to his registered mobile number.
2. Using Alternative path by entering random 4 digits of SSN and ZIP Code.
3. (Only for Digital Users) Sign In.

For Digital Customer

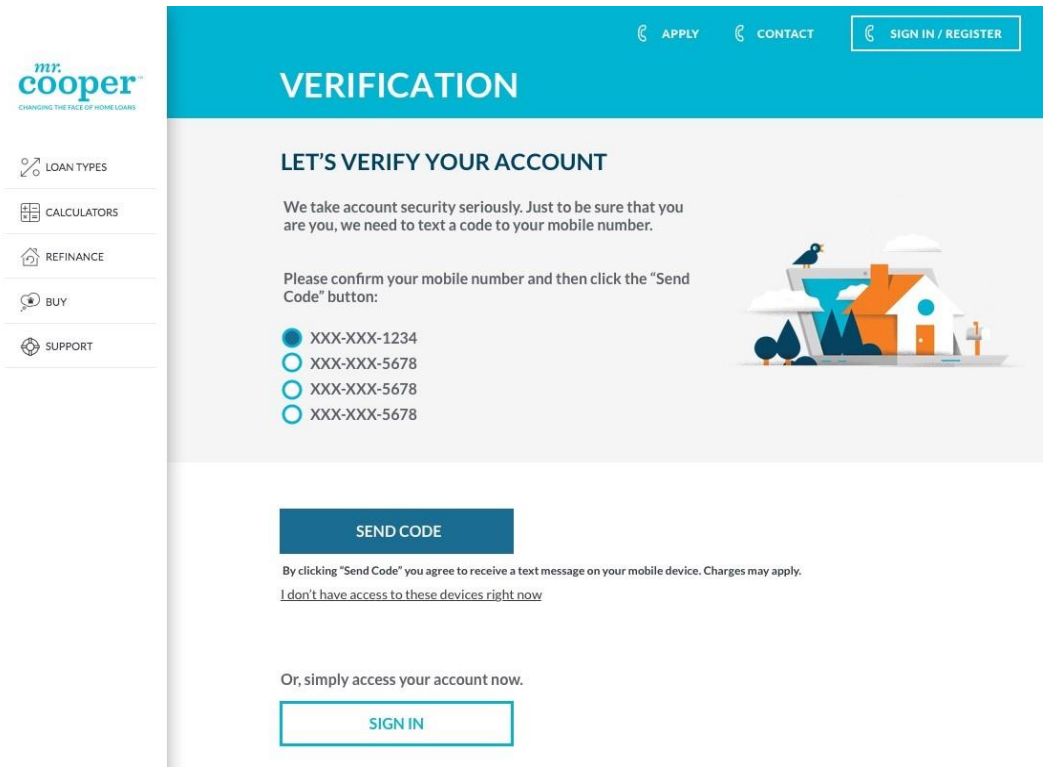

Fig. 5.3 – Verification page view for Digital Customers

For Non-Digital Customer

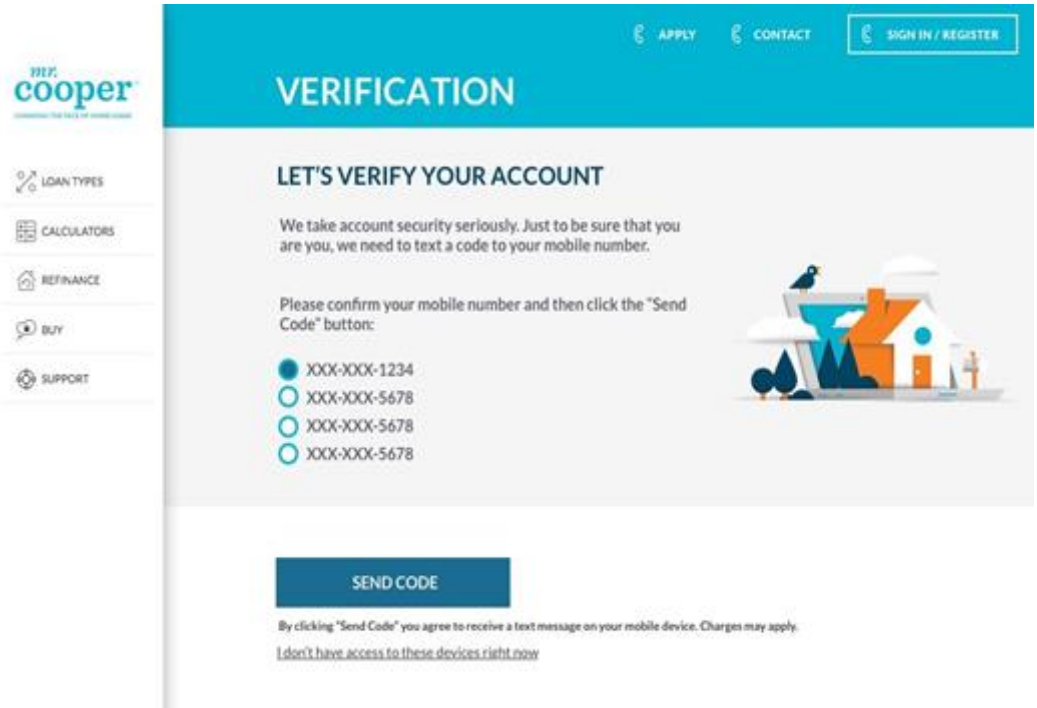

Fig. 5.4 – Verification page view for Non-Digital Customers

4. Sign In Page when Digital Customer Clicks Okta Features

Digital customer has limited access to the website when he enters through one-time links without using SIGN IN option. So, after entering the website if user clicks any Okta features he will be redirected to Sign In Page.

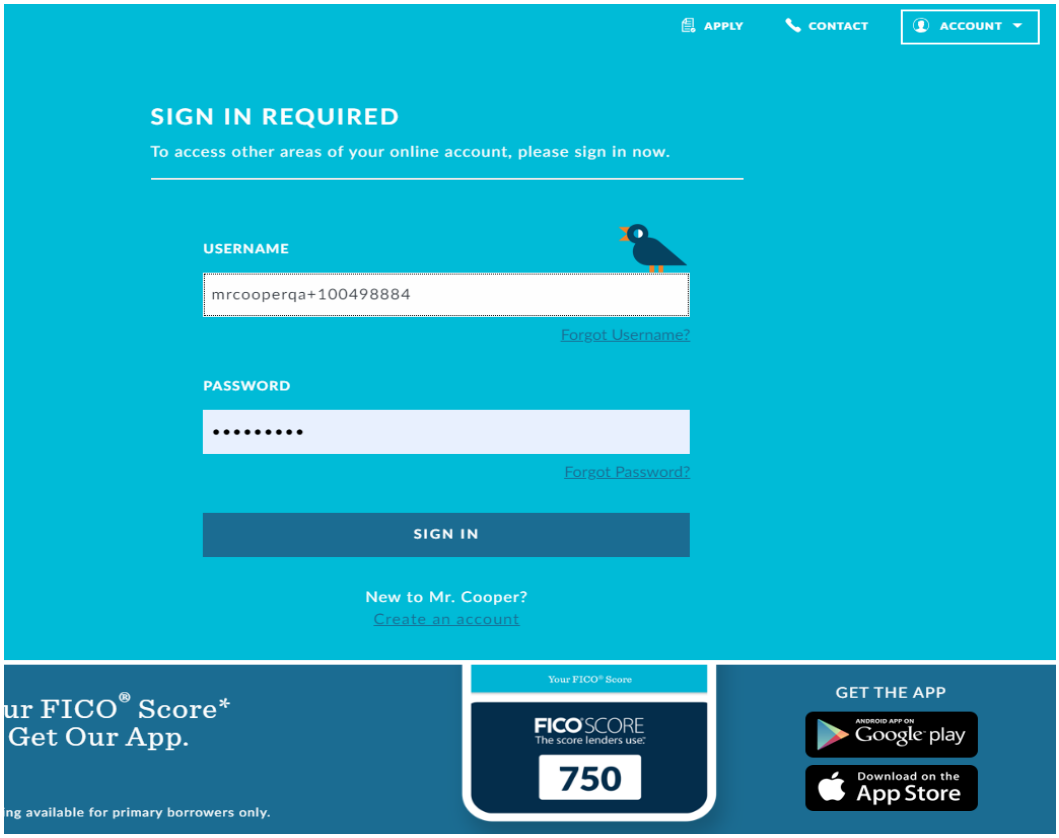

Fig. 5.5 – Sign In Page when Digital Customer clicks Okta Features

5. Create Account Page When Non-Digital Customer Clicks Okta Features

Non-Digital customer has limited access to the website when he enters through one-time links. So, after entering the website if he clicks any Okta features he will be redirected to create account page.

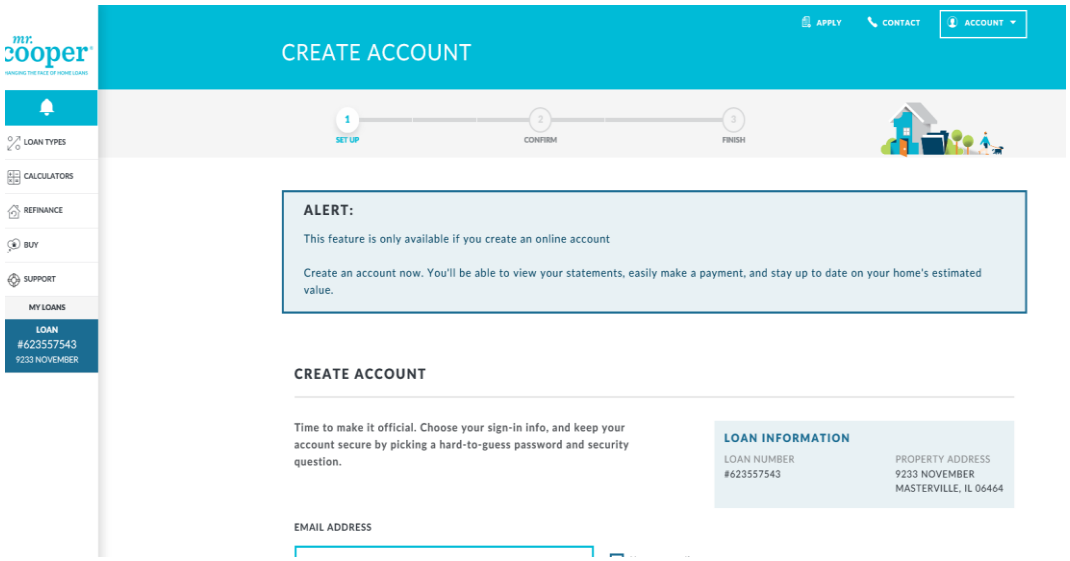

Fig. 5.6 – Create Account Page when Non-Digital Customer clicks Okta Features

6. Expired Page View

Expired page will be visible when customer clicks the one-time link for the 2nd time.

For Digital Customer

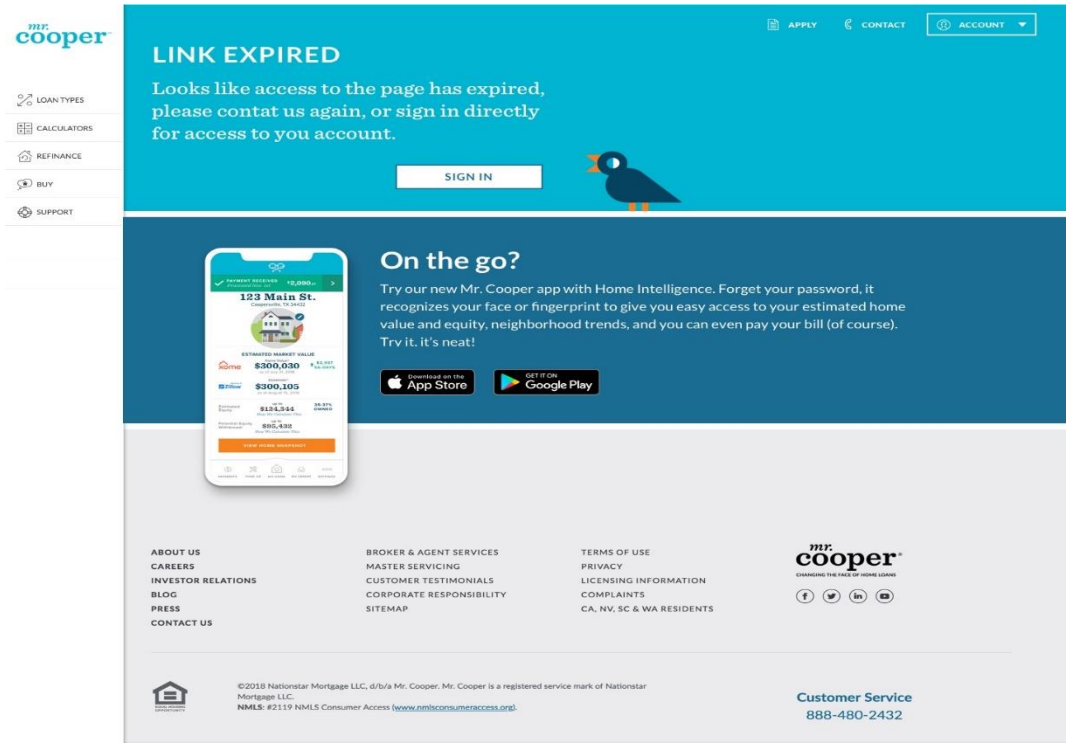

Fig. 5.7 – Expired Page view for Digital Customer

For Non-Digital Customer

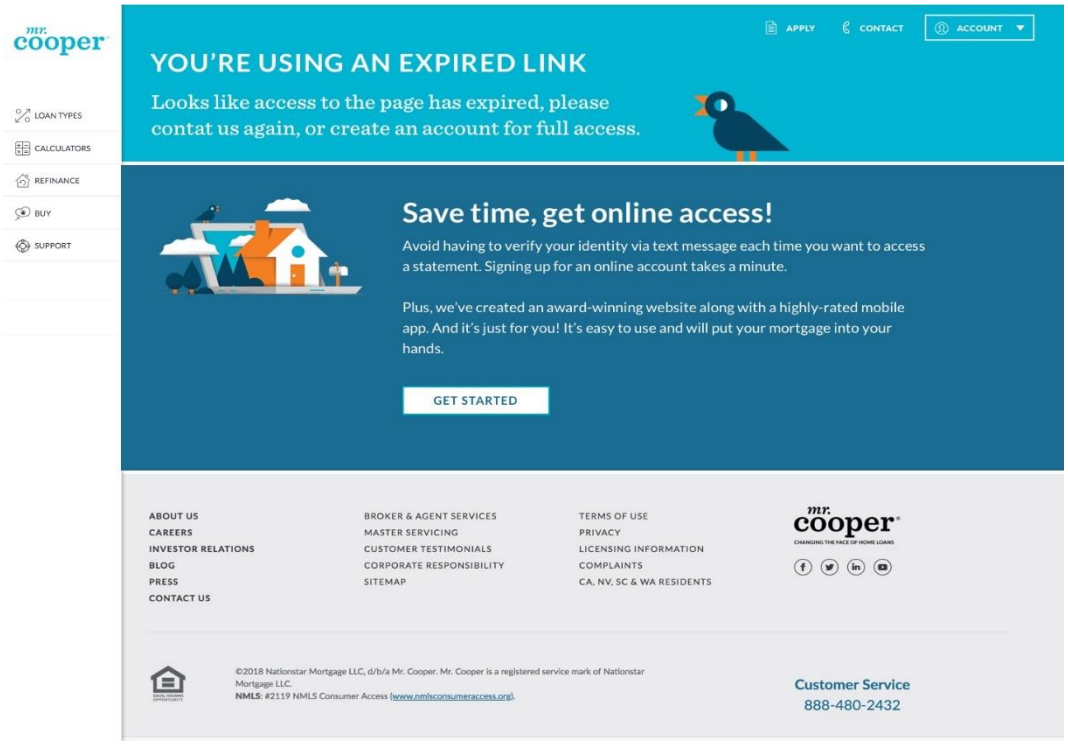

Fig. 5.8 – Expired Page view for Non-Digital Customer

7. Go Paperless Option in Create Account Flow

While creating an account customer will be give an option in Step-3 to enable paperless.

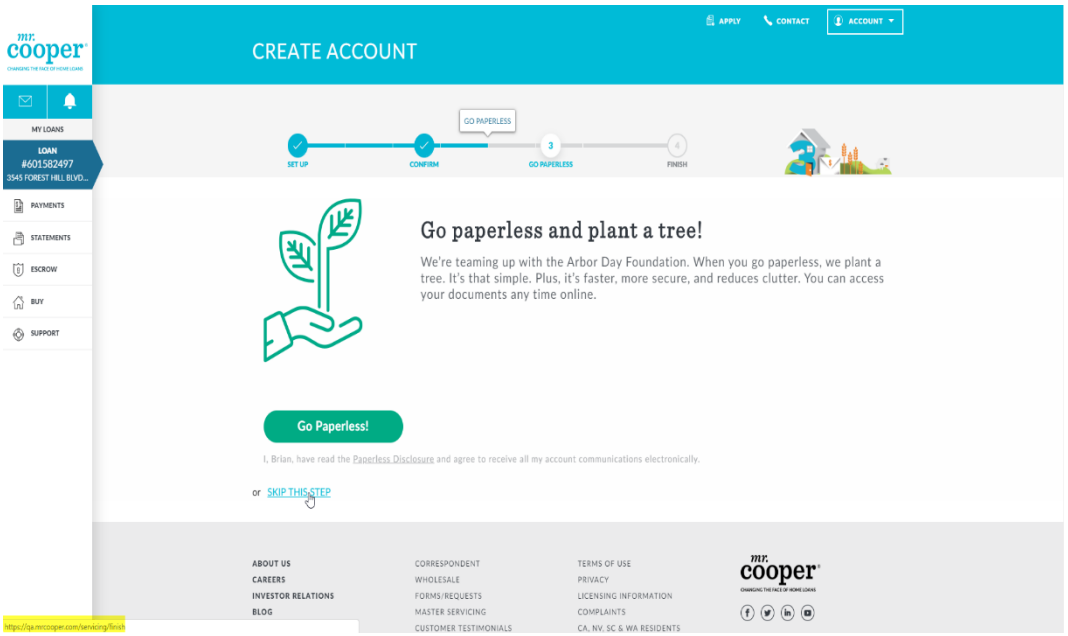

Fig. 5.9 – Go Paperless option in Create Account Flow.

8. Go Paperless Option as a Widget

Go Paperless widget is available in right side of the page in main dashboard requesting customer to go paperless.

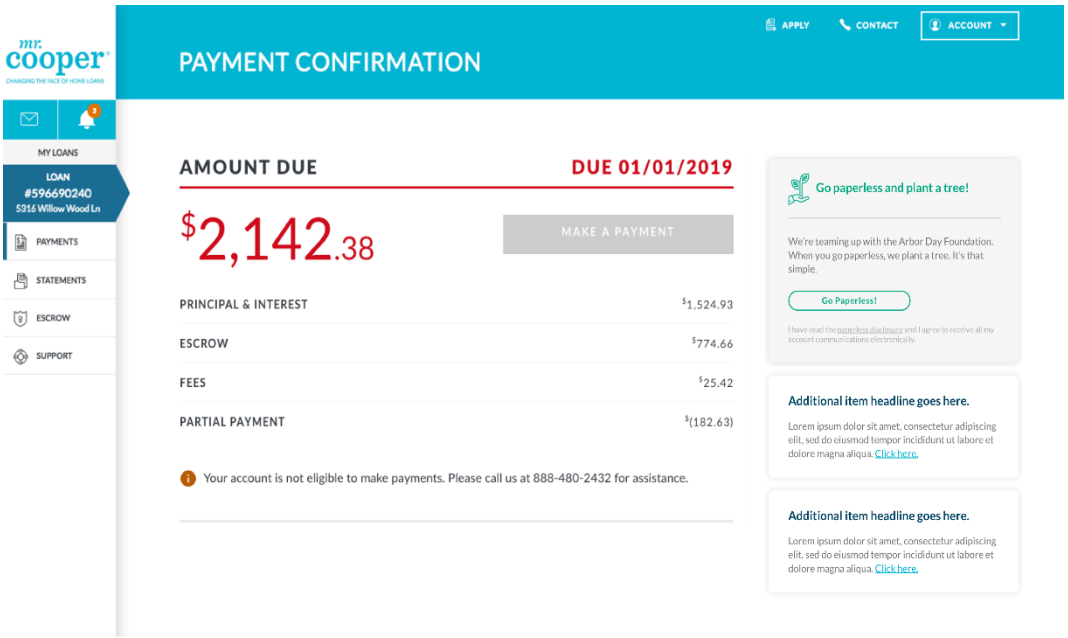

Fig. 5.10 – Go Paperless option as Widget

9. Go Paperless Option as a Banner

Go paperless banner will be shown to the customer as a banner in the top in the statements page

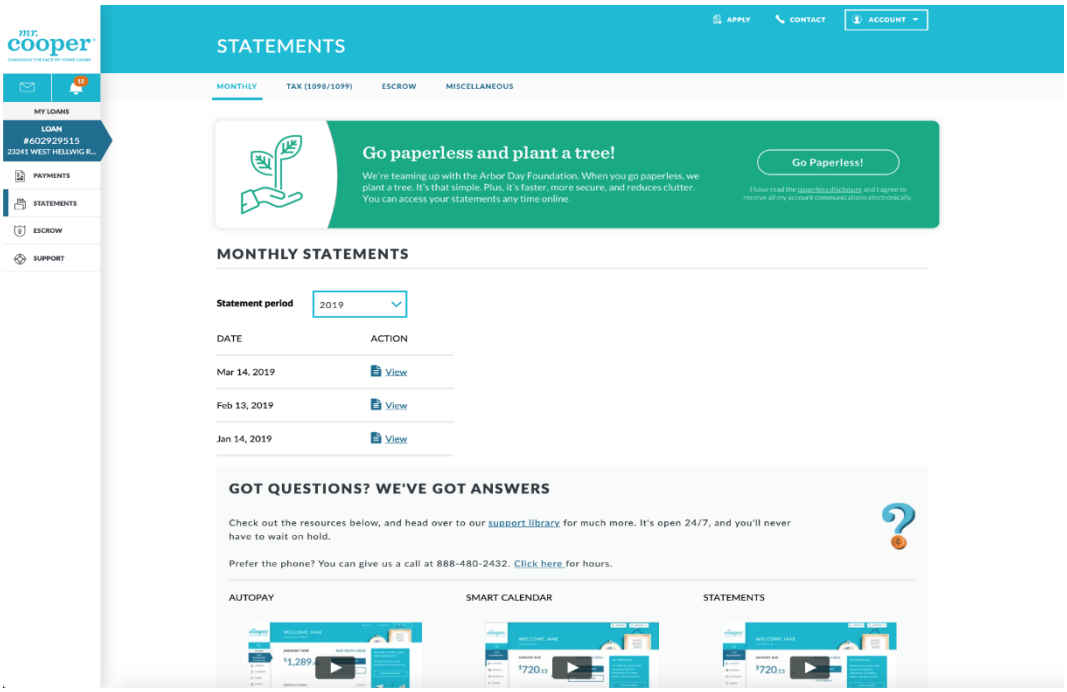

Fig. 5.11 – Go Paperless option as a Banner

10. Go Paperless Option as a Pop-Up

Go paperless pop-up banner will be shown to the customer after logging into the website to enable paperless, if customer hasn't enabled for paperless even after seeing the paperless widget in the right side of the page in main dashboard for the 3 times after logging in.

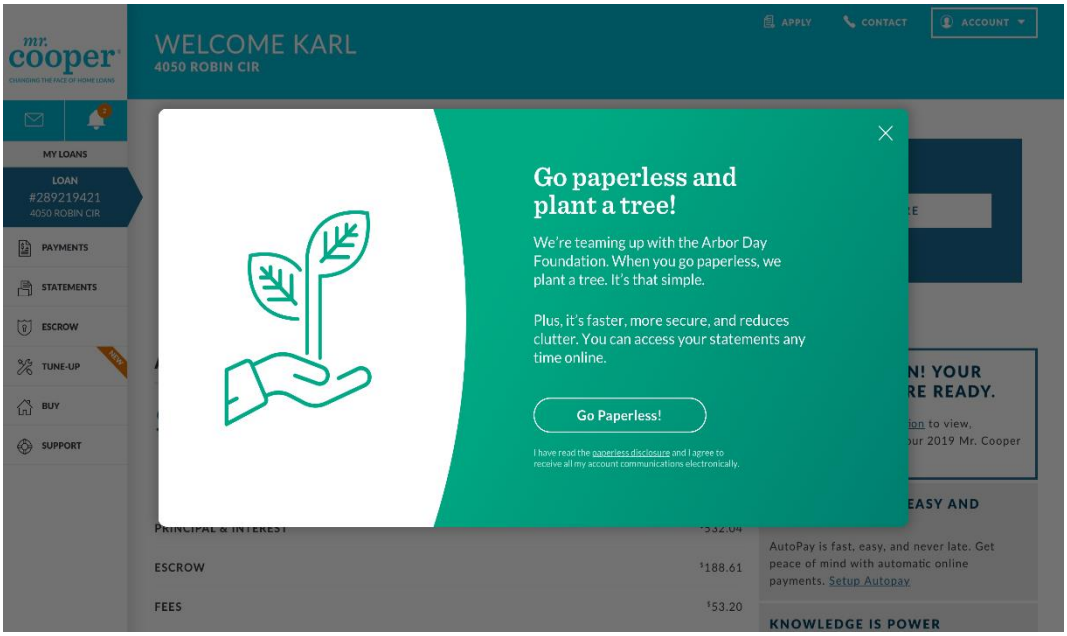

Fig. 5.12 – Go Paperless option as a Pop-up

CHAPTER 6

CONCLUSION

The one-time links application can be further developed by introducing it for different features in the website. Also, for some of the white labels it will consider all the users as Non-Digital users even though some of them are Digital users and it won't show the Sign in option in the verification page. This is because some white labels will use their own sign in page and they won't use Mr. Cooper sign in. So, the session which is created will come from their website. Thus, session which is coming through their website should be handled in the verification page and sign in option should be made available for all the Digital users.

Implementing an application of this scale should require a very strong database which should be available to store a lot of values from many customers. Currently the application is using COSMOS DB where it will generate a unique ID upon the insertion of some records of the customer and based on that unique ID one-time link will be generated. COSMOS DB is used because of its availability all over the world and any DB language can be used to query COSMOS DB. Apart from COSMOS DB many other similar DB's are available which can be used. Currently, the customer logging into the website through one-time links is not being logged which has to be implemented in near feature.

REFERENCES

[1] Hello World – React, Available at: https://facebook.gihub.io/react/docs/hello-world.html, Accessed on: 29-02-2019.

[2] Inner working of virtual DOM, Available at: https://medium.com/@rajaraodv/the-inner-workings-of-virtual-dom-666ee7ad47cf, Accessed on: 29-02-2019.

[3] Stack Overflow, Available at: https://stackoverflow.com/ , Accessed on: 22-04-2019.

[4] W3school, Available at: https://www.w3schools.com/php/ , Accessed on: 18-02-2019.

[5] React responsive, Available at: https://github.com/contra/react-responsive, Accessed on: 20-02-2019.

[6] Dimas Gilang Saputra and Fazat Nur Azizah, "A Metadata Approach for Building Web Application User Interface," in The 4th International Conference on Electrical Engineering and Informatics, ICEEI,2013, pp. 903-911.

[7] Multi-tenant Architecture, Available at: https://developer.salesforce.com/page/Multi_Tenant_Architecture, Accessed on: 20-02-2019.

[8] React on server for Beginners, Available at: https://scotch.io/tutorials/react-on-the-server-for-beginners-build-a-universal-react-and-node-app, Accessed on : 05-03-2019.

[9] Unit test your JavaScript with Mocha and Chai, Available at: https://www.sitepoint.com/unit-test-javascript-mocha-chai/, Accessed on: 20-02-2019.

[10] Nationstar University, Available at: https://nationstar.csod.com/client/nationstar/default.aspx, Accessed on: 17-01-2019.

[11] Introduction to WPF, Available at: https://msdn.microsoft.com/en-us/library/aa970268(v=vs.100).aspx, Accessed on: 08-04-2019

[12] The MVVM Pattern, Available at: https://msdn.microsoft.com/en-in/library/hh848246.aspx, Accessed on: 10-04-2019

[13] Implementing the MVVM pattern, Available at: https://msdn.microsoft.com/en-us/library/ff798384.aspx, Accessed on: 10-04-2019.

[14] Working with Excel files, Available at: https://www.dotnetperls.com/excel, Accessed on: 12-04-2019.

[15] Prism and MVVM step by step, Available at: www.c-sharpcorner.com/UploadFile/31514f/prism-and-mvvm , Accessed on: 11-04-2019.

[16] NUnit – Quick Start, Available at: http://www.nunit.org/index.php?p=quickStart&r=2.6.4, Accessed on: 10-04-2019.

[17] Ruby On Rails – Architecture, Available at: https://www.techcareerbooster.com/blog/ruby-on-rails-architecture-overview-for-beginners, Accessed on: 10-05-2019.

[18] What is Ruby on Rails, Available at: http://railsapps.github.io/what-is-ruby-rails.html, Accessed on: 05-02-2019.

[19] React + Redux Architecture, Available at: https://medium.freecodecamp.org/react-redux-architecture-part-1-separation-of-concerns-812da3b08b46 Accessed on: 05-03-2019.

[20] Ruby on Rails Architectural Design, Available at: https://adrianmejia.com/blog/2011/08/11/ruby-on-rails-architectural-design/, Accessed on: 15-05-2019.

[21] React.js (Introduction and Working), Available at: https://www.geeksforgeeks.org/react-js-introduction-working/, Accessed on: 10-01-2019.

[22] Getting Started with Redux.js, Available at: http://redux.js.org/introduction/getting-started, Accessed on: 10-02-2019.

[23] Getting to Know Flux, the react.js Architecture, Available at: https://scotch.io/tutorials/getting-to-know-flux-the-react-js-architecture, Accessed on: 10-02-2019.

[24] Alex Banks and Eve Porcello “Learning React – Functional Web Development with React and Redux”, 1607.3561 (2016).

[25] William Danielsson “React native application development – A comparison between native Android and React Native,” LIU-IDA/LITH-EX-A—16/050—SE, 2016.

[26] Shinya Hara and Yoshiro Imai “Development of Web Application for Education Assistance Environment with Wen-based Questionnaire Service,” in 4th International Conference on Electronics and Software Science (ICESS2018), 2018.