SELF SERVICE PORTAL FOR MOBILE APPLICATION DEVELOPMENT

Project Report Submitted in Partial Fulfilment of the Requirements
for the Degree of

Bachelor of Technology

in

Computer Science and Engineering

Submitted by

Prasanna Natarajan: 1410110298

Under the Supervision of

Mathew Basil Thomas
Senior Manager, IT Development,
Des_Mob_Exp_Svcs_In

Department of Computer Science and Engineering

May, 2018

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Name of the Student _____ Signature and Date _____

Acknowledgements

The internship opportunity I had with Dell EMC was a great chance for learning and professional development. I am also grateful for having a chance to meet so many wonderful people and professionals who led us through this internship period.

I take this opportunity to express my deepest gratitude to Mathew Basil Thomas, Senior Manager, Development at Dell EMC who in spite of being busy with his duties, took time out to hear, guide and keep me on the correct path and allowing me to carry out my project at their esteemed organization.

I wish to express my sincere gratitude to Dr. Debopam Acharaya, HoD and Associate Professor, CSE Department, SNU for continuously providing me support and updates about the project expectations for the entire duration of the project.

The work could not have been possible without the support of Danny, Product Owner and Ashitha, Scrum Master. I would also take this opportunity to thank Sankar Narayanan, Mentor for guiding me throughout the project.

Also, I thank Shourya Pratap Singh, Sneha Agarwal, Sneha Reddy and Harshit Anand for being wonderful team members.

Lastly, I thank my parents and my friends for their support.

Table of Contents

List of Tables

List of Figures

List of Symbols and Abbreviations

Abbreviation	Full Form
RMADP	Rapid Mobile Application Development Platform
SSP	Self Service Portal
PCF	Pivotal Cloud Foundry
API	Application Programming Interface
REST	Representational State Transfer
JSON	JavaScript Object Notation
JDK	Java Development Kit
VM	Virtual Machine
SDK	Software Development Kit
JS	JavaScript
UI	User Interface
CSS	Cascading Style Sheets
PHP	Hypertext Preprocessor
IPA	iPhone application
APK	Android application package
PaaS	Platform as a Service

Abstract

In this project, I developed an Innovative Rapid Mobile Application Development Platform (RMADP) which is focused on enabling business teams to develop mobile apps without coding. From this portal, users will be able to create simple mobile applications by just dragging and dropping different GUI components and setting the functionalities of each as per the requirement of the application. This portal provides the user with Android App Executable (.apk) and iOS App Executable (.ipa).

The primary goal of this project was to enable at least 3 business teams with no programming experience, to develop mobile apps through this self-service portal, which was successfully fulfilled by end of this project.

For the purpose of this document, a simple mobile application is one that does not involve graphics-intensive applications and applications that exploits the native sensors of a mobile device.

1. Introduction

According to Statista [1], by the year 2019, the total number of Smartphones users worldwide are expected to be 2.71 billion. With such a large user base, more mobile apps will need to be developed to cater different needs.

With such a user base in mind, and making use of the ease employees will get with mobility, Enterprises have started developing mobile apps for different departments ranging from HR to finance to the datacenter.

1.1. Problem Definition

In a large organization like Dell with over 138,000 employees [2], a large number of Enterprise apps are required to be built for a myriad of teams. The need for each app varies based on the business team.

The Mobility Team of Dell becomes the bottleneck for all the mobile apps that get built inside Dell. A solution was needed to enable teams to build their own mobile apps without consulting the Mobility team or hiring/training developers for mobile development.

1.2. Problem Overview/Specification

1.2.1. Project Description

Develop a Self Service Portal (SSP) for enabling any Dell team to develop enterprise mobile apps without the knowledge of programming.

1.2.2. Deliverables

1. Develop and create a proposal for a self-service portal to help business teams build mobile apps without coding.

2. Define the business and technical requirements and implement for iOS and Android.

3. Develop Drag and Drop Capability for end user to pull together needed components.

4. Develop Connector mechanism to help connect the UI components to backend micro-services/databases.

5. Demonstrate the ability to develop an end to end iOS and Android app via the self-service portal that has login capability, key UI components and connectivity to a backend micro-service.

6. Develop a proposal for deployment and marketing.

1.2.3. Key Metrics

Enable at least 3 business teams with no mobile experience to develop mobile apps through the self-service portal.

1.3. Hardware Specifications

- Apple Macintosh with Mac OS X 10.11 or higher (for build server)
- Pivotal Cloud Foundry (PaaS) with 4 VM instances to run PHP (for portal), Python (for mobile app services), and Spring Boot (for middleware) Applications, and Maria DB database server.
- Android phone with minimum Android 6.0 Marshmallow OS (for running mobile apps).
- Apple iPhone with minimum iOS 9 (for running mobile apps).

1.4. Software Specifications

1.4.1. Portal

The list of software used for development of the portal are listed below in table (1.1)

(1.1) List of software used for development of the portal

Software/Technology	**Version**	**License**
HTML [3]	5	-
CSS [4]	4	-
JavaScript [5]	ECMA Script 6	-
PHP [6]	5.6.35	PHP License
Apache Web Server [7]	2.4.33	Apache License 2.0
XAMPP for Windows [8]	5.6.35	GNU GPL
Bootstrap [9]	4.0.0	MIT License
Font Awesome [10]	5.0.7	Font Awesome Free License
JQuery-Confirm [11]	3.3.0	MIT License
JQuery [12]	3.2.1	MIT License
Popper.js [13]	1.12.9	MIT License
Vue.js [14]	2.5.13	MIT License
image-picker [15]	0.3.0	MIT License
devices.css [16]	-	MIT License
Intro.js [17]	2.8.0	Commercial / Open Source License
ddSlick [18]	2.0	-
JQuery UI [19]	1.12.1	MIT License
Holder.js [20]	2.9.4	MIT License

1.4.2. Middleware and Database

The list of software used for development of the middleware layer are listed below in table (1.2)

(1.2) List of software used for the development of the middleware layer

Software/Technology	**Version**	**License**
Java Development Kit [21]	1.8.0_161	Oracle BCL
Eclipse [22]	Oygen.3 Release 4.7.3	Eclipse Public License
Spring Tool Suite [23]	3.9.4	Eclipse Public License
Apache Maven [24]	3.3.9	Apache License 2.0
Gson [25]	2.8.2	Apache License 2.0
Okio [26]	1.14.0	Apache License 2.0
OkHttp [27]	3.10.0	Apache License 2.0
MariaDB [28]	10.1.26	GNU GPL

1.4.3. Build Server and Mobile App Services

The list of software used for development of the build server and mobile application are listed below in table (1.3)

(1.3) List of software used in development of build server and the mobile application

Software/Technology	**Version**	**License**
Homebrew [29]	1.5.11	BSD License
GNU sed [30] (stream editor)	4.4	GNU GPL
Node [31]	9.6.1	MIT License
Ionic Framework [32]	3.20.0	MIT License
Cordova [33]	8.0.0	Apache 2.0 License
Android Studio with SDK [34]	3.0 with Android SDK API 23	Freeware
XCode with Command Line Tools [35]	9.3	Freeware with open source components

1.4.4. Generated Mobile Application

Uses following Ionic Plugins:

- In App Browser [36]
- Social Sharing [37]
- Chat.JS [38]
- Email Composer [39]
- File [40]
- File Transfer [41]
- Media [42]
- Streaming Media [43]

1.4.5. Other Software

The list of other software used for development are listed below in table (1.4)

(1.4) List of other software used

Software/Technology	**Version**	**License**
Git [44]	2.16.2	GNU GPL
Gitlab [45]	-	Commercial License
JIRA [46]	-	Commercial License
Postman [47]	6.0.10	Free Version

2. Literature Survey/Related Works

2.1. Existing Solutions

An extensive research on the available tools in the market was done to get a sense of potential features for the portal. All the important findings are summarized below:

2.1.1. Adobe XD

Adobe XD [48] is a UI prototyping tool by Adobe. Its important features are:

1. It consists of GUI connect, which lets the user see the flow if the application.
2. User can drag and drop the components into a phone frame.

2.1.2. Kinetise

Kinetise [49] is a self-service web portal for making Android/iOS mobile applications. Its important features are:

1. User can drag and drop the components into a phone frame.
2. The platform offers its own Content Management System (CMS).
3. The platform offers advanced user management features.
4. The platform offers map integration.
5. Data can be retrieved in tabular form.
6. User can even define the logic in Excel format.
7. This platform uses refresh policy for every second in which an event trigger like repopulate components is provided.
8. Components like charts, QR codes are also provided.
9. This platform provides native code for making complex changes.
10. It provides support for external API's like Facebook, Google Maps, Twitter, YouTube etc.

2.1.3. Kony

Kony [50] is a self-service web portal for making Android/iOS mobile applications. Its important features are:

1. Kony provides one-click deployment feature for both apps and services.

2. It has a marketplace with reusable components and apps.
3. Mapping of database and forms are done based on fields.
4. It provides native UI for Android and iOS.

2.1.4. Eachscape

Eachscape [51] is a self-service web portal for making Android/iOS mobile applications. Its important features are:

1. Eachscape provides an option to own the source code.
2. This platform provides a separate powerful user management section.

2.1.5. Thunkable

Thunkable [52] is developed by MIT. It is a self-service web portal for making Android/iOS mobile applications. Its important features are:

1. This tool provides control for voice commands.
2. It has community driven support.

2.1.6. Shoutem

Shoutem [53] is a self-service web portal for making Android/iOS mobile applications. Its important features are:

1. Shoutem provides color based themes and layouts.
2. It provides splash screen component.
3. It allows for customization of individual components.

2.1.7. Appypie

Appypie [54] is a self-service web portal for making Android/iOS mobile applications. It majorly supports pre-built templates and uses web technologies to build the app. It provides trial version for 48 hours in editing for both iOS and Android, after which the app cannot be edited unless a payment is made.

2.1.8. Appgyver

Appgyver [55] is a self-service web portal for making Android/iOS mobile applications. Its important features are:

1. Appgyver has a brilliant UI drag and drop.
2. It visually defines the front end functionalities.

2.1.9. Bizness Apps

Bizness Apps [56] is a self-service web portal for making Android/iOS mobile applications. Its important features are:

1. It consists of loyalty programs and reviews.
2. It also provides marketing options with a dedicating marketing advisor.
3. It has reseller community.

3. Requirement Analysis

3.1. Given Requirements

Develop a Self Service Portal for enabling teams to develop mobile apps. The intern will play the role of a Software Developer.

3.2. Requirement Gathering

We took interview of employees from different business teams of the organization to know and gather their requirements. During the interviews, employees gave us an overview of what platforms they work on daily basis and ideas on which simple mobile apps can be built.

Some of the question asked during the interviews are as follows:

1. Have you ever tried building an app before? Could you describe the step by step process of how you approached and did it?

2. How much time did it take?

3. Have you used something similar (say X) before?

4. Does X solve your problem? Why not? Where does X fall short of your expectation?

5. Do you have any idea about a mobile app that you would like to use on your team, to increase productivity?

6. How fast do you need to get this app out to market?

7. What budget do you have to play with?

8. Does your app involve more than 10 screens/pages?

9. Will more than 100 users be using this app?

10. How frequently will you be modifying the app?

11. Does your application involve the usage of device hardware features (Camera, GPS, and Accelerometers)?

12. Does your application need features like AR, VR, 3D touch or Contact details retrieval?

13. What technical tools or platforms do you use majorly on a daily basis?

Information gathered during these interviews cannot be disclosed as per Dell EMC's policies. Based on this information, key features for the portal were decided.

3.3. Functional Requirements

3.3.1 Portal

- Users should be able to create a new app
- User should be able to edit previously created app
- User should be able to delete existing app
- User should be able to drag and drop component
- User should be able to create screens
- User should be able to delete screens
- User should be able to build project
- User should be able to generate android apk and xcode project
- User should be able to download generated apk, zip

- User should be able to add side menu
- User should be able to add tabs
- User should be able to produce web app

3.4. Non-functional requirements

- Middleware server should balance load in build server
- Scripts if sent from frontend should not be saved in data base
- User should be restricted from building the app until every screen is in correct state
- User should be shown different states of build process when he builds an app

3.5. Scope of the project

- All the generates apps are hybrid application
- No support for push notification
- Some native features (file transfer and handling, media support and sharing) are only supported.
- Features which are hardware intensive (graphics rendering for games, access to sensors etc.) are not supported.

4. Feasibility Study

4.1. Solution 1 – Desktop Application

4.1.1. Description

The self-service portal can be a desktop application that works on Windows/Mac.

4.1.2. Technical Feasibility - Infeasible

The portal can be easily built on cross-platform desktop application development frameworks like electron. Integration with Dell EMC's authentication will be hard especially because Dell EMC's authentication that is currently in place works differently on different platforms and also does not support cross platform apps.

4.1.3. Operational Feasibility – Feasible

All Dell EMC employees are given a laptop (a windows/mac machine), therefore installation of this desktop portal shouldn't be a problem. Moreover, with the help of IT team, we can also force install this portal to make it available for all Dell EMC employees.

4.1.4. Organizational Feasibility – Feasible

In Dell EMC, employees use windows or a mac machine. Since Dell EMC is a large organization and not all app requests can be handled by the mobile experience team, simple apps can be delegated to this portal. The Dell EMC mobile app standards will still be maintained as the portal itself is maintained by the mobile experience team.

4.1.5. Overall Viability – Not Viable

Since authentication will be a challenge and without it, the portal cannot function properly this solution is not viable.

4.2. Solution 2 – Mobile Application

4.2.1. Description

The self-service portal can itself be a mobile application that works on Android/iOS.

4.2.2. Technical Feasibility - Feasible

The portal can be implemented using a cross-platform app builder like ionic. Dell EMC also has authentication libraries readily available for these different platforms.

4.2.3. Operational Feasibility – Infeasible

Dell EMC does not have any restrictions on personal smartphones that the employees use. Therefore, it will be hard to make the employees use this portal with the adherence to Dell EMC rules. Moreover, we have found out, through interviews that employees would not like to use this portal from their mobile phones since features like drag and drop will be difficult to use.

4.2.4. Organizational Feasibility – Feasible

Many Dell EMC employees use Android/iOS mobile devices. Since Dell EMC is a large organization and not all app requests can be handled by the mobile experience team, simple apps can be delegated to this portal. The Dell EMC mobile app standards will still be maintained as the portal itself is maintained by the mobile experience team.

4.2.5. Overall Viability – Not Viable

Since Dell EMC doesn’t have any control over employee personal devices, the portal being there wouldn’t serve the company’s purpose.

4.3. Solution 3 – Web Application

4.3.1. Description

The self-service portal can be a web application.

4.3.2. Technical Feasibility - Feasible

The self-service portal can be built using web technologies. The Android/iOS application that is generated out of this portal could be built using the Ionic framework, a cross-platform mobile application builder. Dell EMC's web authentication system can be easily integrated with the web portal.

4.3.3. Operational Feasibility – Feasible

A Dell EMC employee can access this portal in any web browser which supports JavaScript with any Dell EMC network registered device. Therefore it will be easy to restrict access to if needed by Dell EMC to make a strict adherence to its policies.

4.3.4. Organizational Feasibility – Feasible

Dell EMC is a large organization and not all app requests can be handled by the mobile experience team, simple apps can be delegated to this portal. The Dell EMC mobile app standards will still be maintained as the portal itself is maintained by the mobile experience team.

4.3.5. Overall Viability – Viable

This is a viable solution.

4.4. Recommended solution for further analysis

Solution 3 seems to be the only viable solution for the reasons elaborated above.

5. System analysis and design

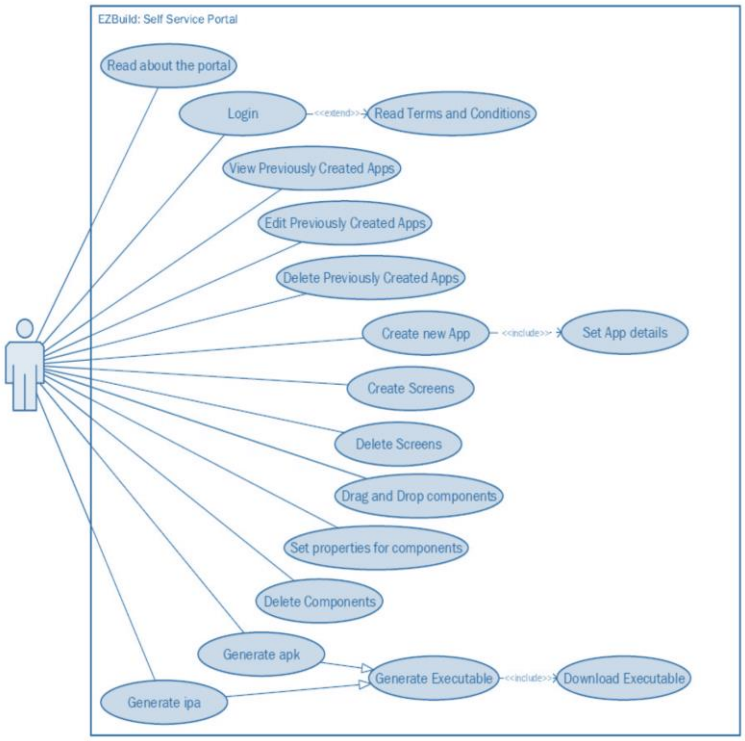
"System analysis is a process of collecting and interpreting facts, identifying the problems, and decomposition of a system into its components." [57], in this section of the report I have tried analyze the self-service portal.

I will try to analyze using various types' system designs. These types are as follows:

5.1. Use Case Diagram (Fig.5.1)

Actor: Dell Employee

Use Cases: Read about pivotal, Login, Read Terms and Condition, View previously Created App, Edit Previously Created App, Delete Previously Created App, Create new app, Set App Details, Create Screen, Delete Screen, Drag and Drop Component, Set component properties, Delete Component, Generate APK, Generate IPA, Generate Executable, Download Executable



(5.1) Use case diagram

5.2. Activity Diagram

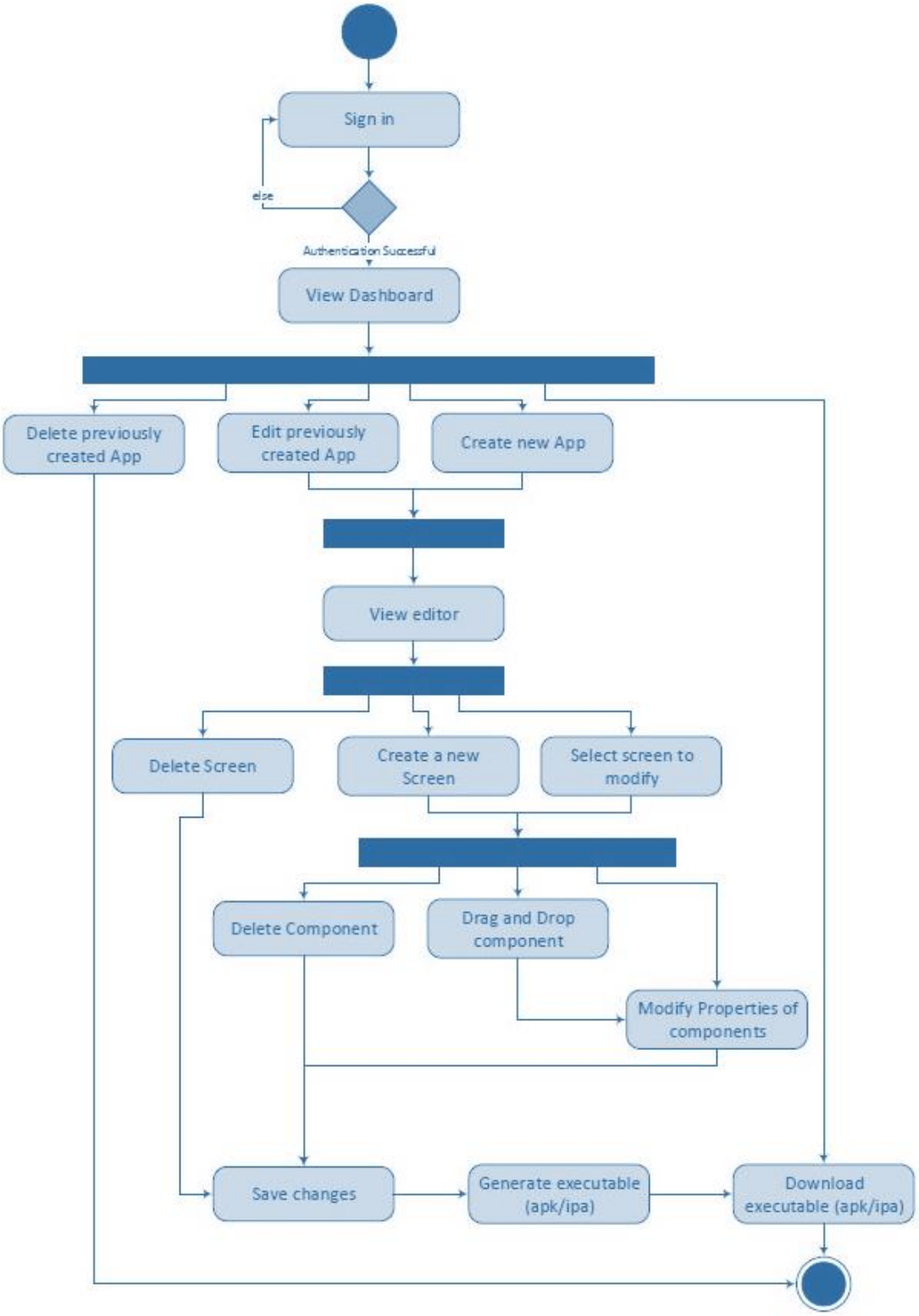
The activity diagram is used to describe flow of activity through a series of actions. [58] The Activity diagram (fig.5.2) is explained below.

When the user enters the URL of the portal, the Dell's authentication happens and if successful, the landing page opens.

Then the view Dash board is opened from the landing page where different tasks can be performed:

- Delete previously created app
- Edit previously created app
- Create new app

For creating new app and editing existing apps user goes into editor page where new screens can be created, existing screens can be modified and deleted. In the screens user can drag and drop various components and modify their properties. After this, the changes are saved and the respective ipa/apk of the app is generated and downloaded.



(5.2) Activity Diagram

5.3. Dataflow Diagram

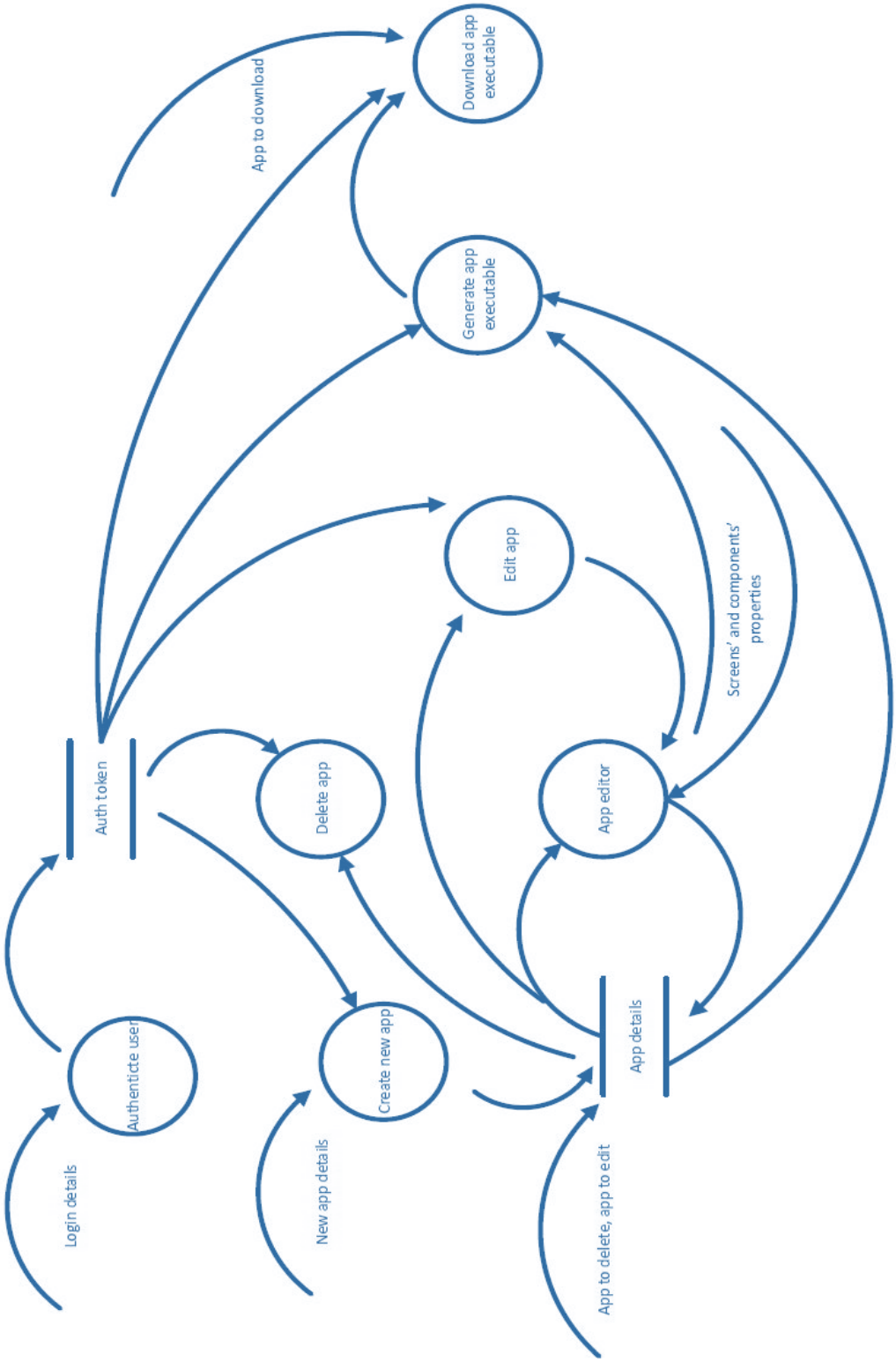
5.3.1. Level 0

The DFD Level 0 shows a data system as a whole and emphasizes the way it interacts with external entities [58]. Here (fig. 5.3) the Dell employee interacts with the portal (EZBuild) by providing Login details, new app details, app to delete, app to edit, app to download, screens and component properties. The portal responds back with the app executable (apk/ipa).

(5.3) Data Flow Diagram Level 0

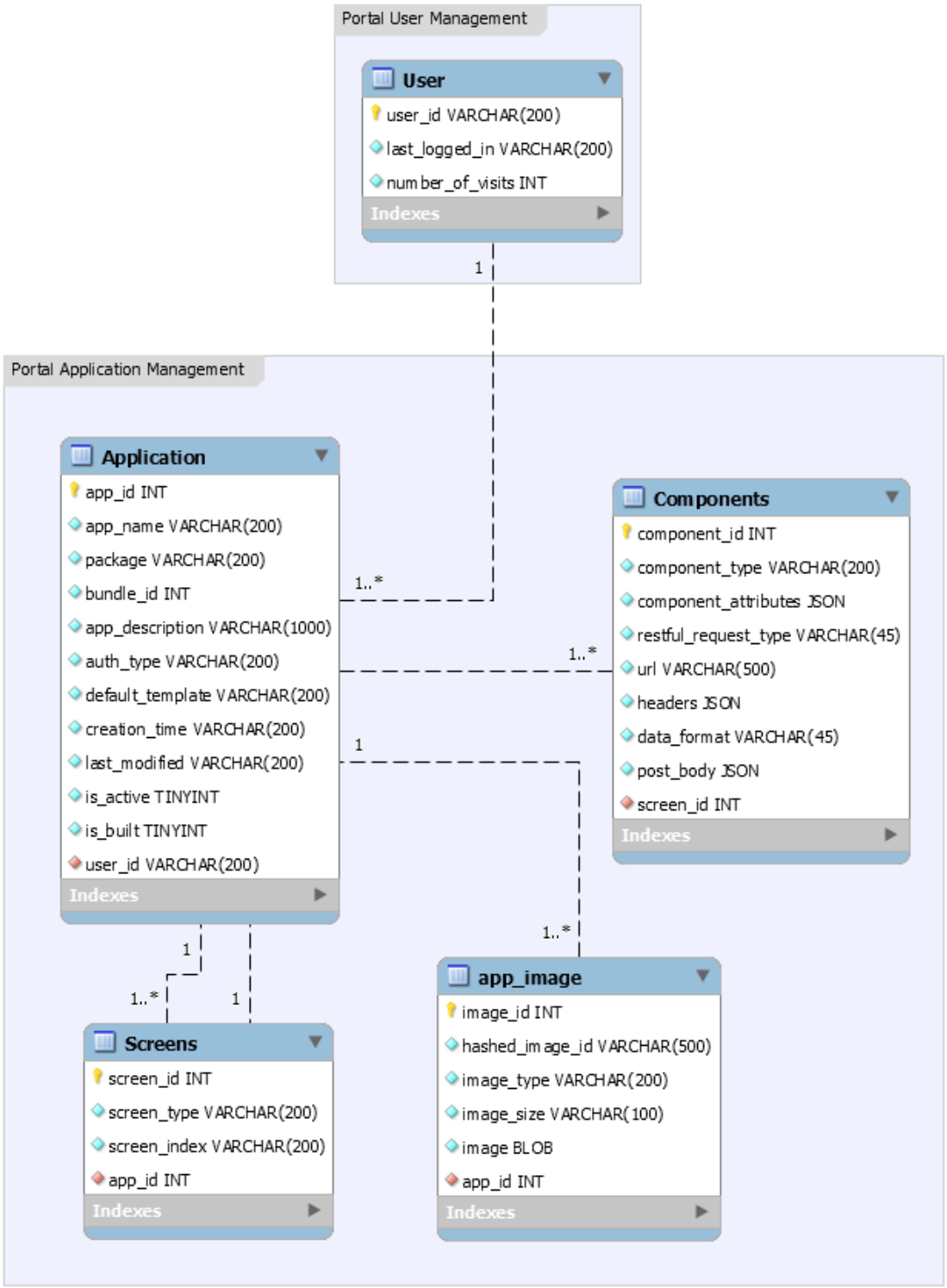
5.3.2. Level 1

The DFD Level 1 is more detailed than DFD Level 0. It breaks down the main processes into sub processes. All the inputs from level 0 are carried forward to level 1 in a detailed form [58]. We have two database stores namely auth token and app details. Auth token is accessed for authenticate user, create new app, delete app, edit app, generate and download app. App details is accessed for create new app, app editor, edit app, delete app and generate app. The following figure depicts level 1 DFD (fig. 5.4)



(5.4) Data Flow Diagram Level 1

5.4. Database Schema (Fig 5.5)



(5.5) Database Schema

5.5. Class Diagram (Fig 5.6)

In UML class diagram describes the structure of system by showing the system classes, attributes, methods and associations between classes. [58]

Class diagram consists of 6 models:

1. User Model
2. Application Model
3. Component Model
4. Screen Model
5. Nav Model
6. Status model

User Model:

User Model consists attributes of user such as user id, last logged in and number of visits.

Methods in User model:

- insert(UserModel customer);
- findByUserId(String userId);
- updateUserModelOnLogin(UserModel userModel);

They contain get and set methods for all the attributes.

Application Model:

Application model consists attributes of application such as application name, package name, bundle id, application description, authentication type, default template, user id, creation type, last modified, is Active, is Build, application id and landing screen.

Methods in Application Method:

- insert(ApplicationModel application);
- fetchApplicationWithId(int appId);
- fetchApplicationsForUser(String userId);
- updateForDelete(int appId);

- updateForBuiltApp(int appId, int value);
- updateForLandingScreen(int appId, int landingScreen);
- getSimpleDate(String date);
- isPackageExists(String packageName);

They contain get and set methods for all the attributes.

Component Model:

Component model consists attributes of component such as component id, screen id, component type, component attributes, restful request type, URL, headers, data format and post body.

Methods in Component Model:

- insert(ComponentModel component);
- fetchComponentWithId(int componentId);
- fetchComponentsForScreen(int screenId);
- update(ComponentModel component);
- delete(int componentId);

They contain get and set methods for all the attributes.

Screen Model:

Screen model consists of attributes of screen such as screen id, app id, screen type and screen index.

Methods in Screen Model:

- insert(ScreenModel screen);
- fetchScreenWithId(int screenId);
- fetchScreensForAppWithId(int appId);
- delete(int screenId);
- getIndexById(int screenId);
- update(int screenId, String screenType);

They contain get and set methods for all the attributes.

Nav Model:

Nav model consists of attributes of nav bar such as nav item id, screen id, option name, nav icon and option index.

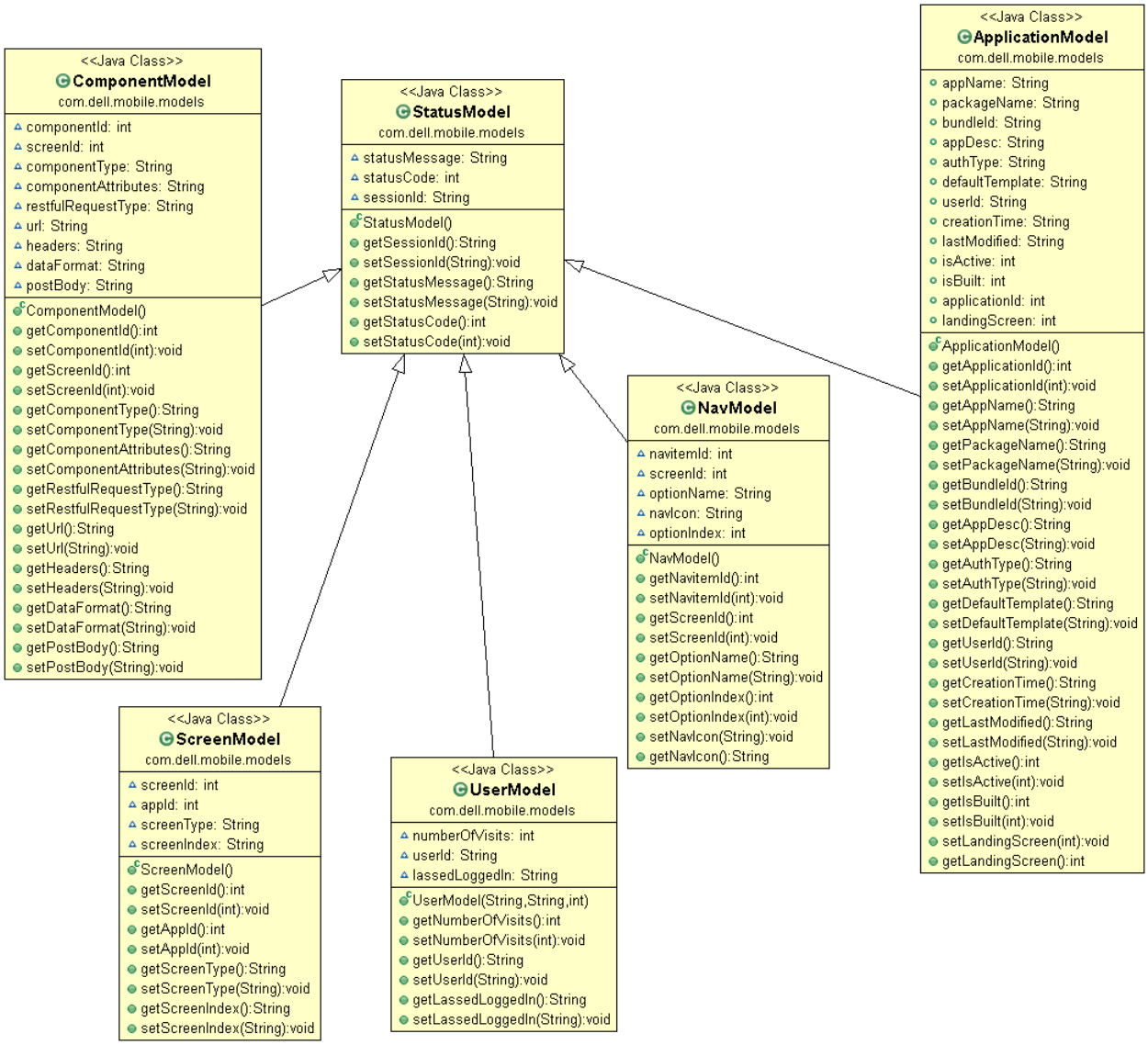
Methods in Nav Model:

- delete(int navitemId);
- update(NavModel navmodel);
- insert(NavModel nav);
- updateNav(ArrayList<NavModel> inputList);
- getNavItemByScreen(int screenId);

They contain get and set methods for all the attributes.

Status Model:

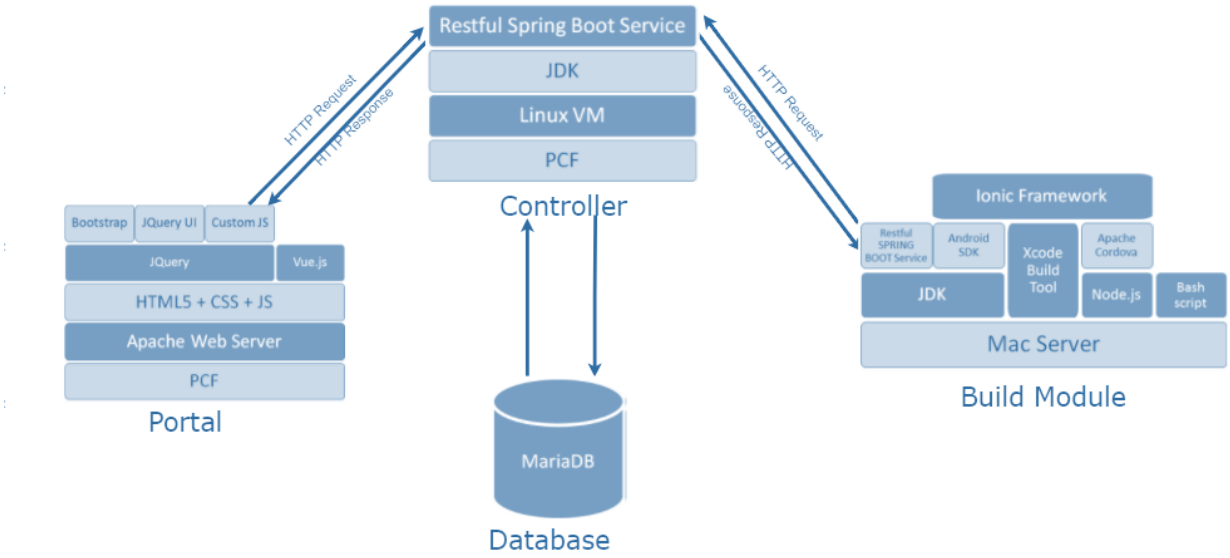
Status model consists of attributes such as status message, status code and session id. They contain get and set methods for all the attributes.



(5.6) Class Diagram

5.6. Architecture

5.6.1. HLA (Fig 5.7)



(5.7) High Level Architecture

5.6.2. The UI Module

This is the frontend portal visible to the user in the browser. It features drag and drop functionality for components to be added to app screens, and has the option to save, view, edit, delete or build app projects.

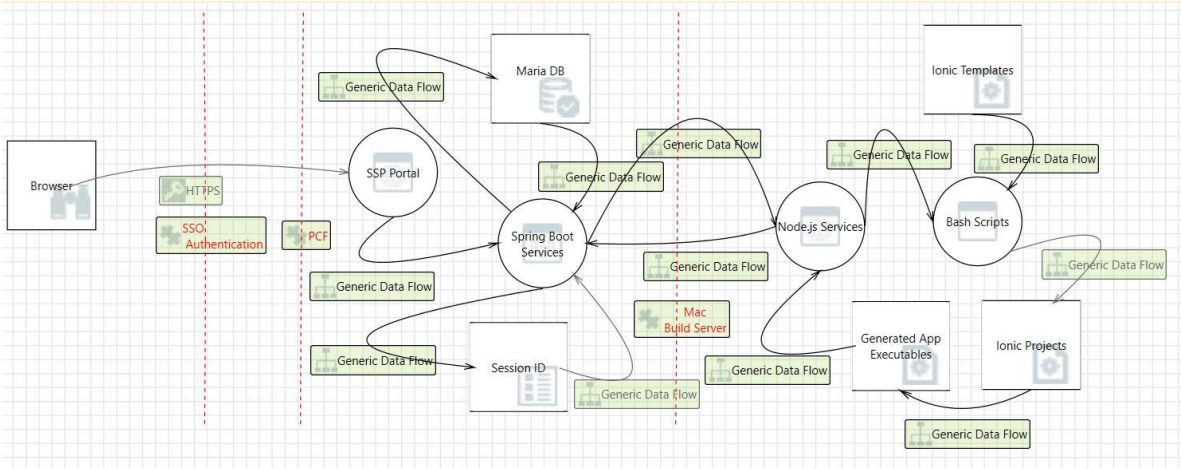
5.6.3. The Controller Module

This is the middle module that takes requests from portal, does all the database operations on the MySQL server, and forwards the build request to the Build Module. The database server stores all app project related information. Only the controller has access to this server.

5.6.4. The Build server

This module sits on top of a Mac Server. It communicates with the controller and generates the Android and iOS Apps.

5.7. Threat Modeling Diagram (Fig 5.8)



(5.8) Threat Modeling Diagram

Threat modeling diagram is used by Dell to analyze the security of this application. In the figure above (fig 5.8) the browser is how an end user will access the portal. The browser will contact the SSP portal, after Dell's Authentication, which will in turn contact the Sprint Boot Services for accessing the database and also contacting the Node.js server for building the mobile application.

6. Implementation

6.1. Methodology Used

During the development of SSP, Pivotal Labs Methodology was strictly followed. This methodology strongly promotes the following two:

6.1.1. Scrum Framework

The Scrum Framework was co-created by Jeff Sutherland and Ken Schwaber in 1993 [59]. The godfathers of Scrum, Takeuchi and Nonaka [60], defined Scrum as a holistic approach for developing products with six characteristics: built-in instability, self-organizing project teams, overlapping development phases, multi-learning, subtle control, and organizational transfer of learning.

The Scrum Team is composed of [61]: Product Owner - responsible for managing the product backlog and defining goals for each "Sprint" (time period of approximately 2-4 weeks in which product increments are released), The Development Team - across - functional and self-organizing team responsible for developing and testing each product increments, and The Scrum Master - responsible for ensuring that Scrum is being practiced correctly.

Every day before starting the work, the Development team holds a Daily Scrum Standup - which is a 15 minutes event, in which the goals for next 24 hours are set and the blockers are discussed.

Scrum Teams also use Scrum Board - a visual representation of the work flow, broken down into manageable chunks called "Stories", with each story moved along the board from the "backlog" (the to-do list), into work-in-progress (WIP), and on to completion [62].

6.1.2. Extreme Programming (XP)

The Extreme Programming methodology adapts to rapidly changing requirements [63]. It also advocates frequent incremental releases in short development cycle (about 2 weeks). XP strongly promotes the following two techniques during the development cycle:

6.1.2.1. Test Driven Development (TDD)

Unlike traditional development methods, where test cases are written after the development process is ended. XP promotes TDD, in which development is driven by automated unit test cases [64]. First of all unit test cases are created from requirements (user stories in Scrum), then code is written such that the test cases pass.

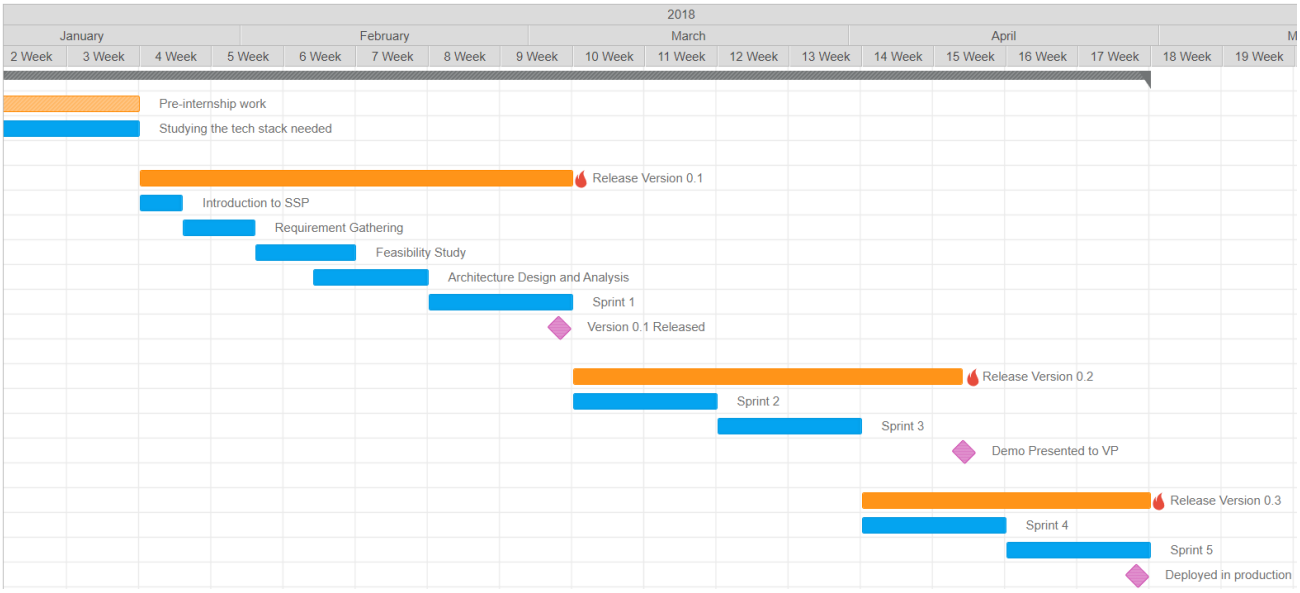
6.1.2.2. Pair Programming

In Pair Programming, two programmers in a team work on the same story at a time. As per TDD, one developer writes the test cases for each of units for the given user story, then the other developer in the pair writes the code, while the first one reviews the code. This technique not just ensures better quality code, but research [65] has shown that this technique also yields happier and confident programmers.

During development of the project, the methodology followed was that after every 2 hours of programming, there is a 1-hour break, then pair switches their positions. In this way, every developer gets to learn and work on different modules of the project and also learn while writing quality code.

6.2. Sprint-wise progress

The entire project was carried out in sprints. The Gantt chart below (fig 6.1) represents the timeline and the work done sprint wise.



(6.1) Gantt chart

6.2.1. Sprint 1 (19/02/18 – 02/03/18)

6.2.1.1. User Stories

The following high level user stories were completed in sprint 1:

- High level architectural diagram
- Research on competing products in market.
- Research on possible use cases with EMC.
- Landing page.
- App Dashboard.
- Build and Download app executable.
- Create a new app project.
- App Editor Page including Screen view, create new screen, Delete screen, components view, Mobile frame, drag and drop components, creation of components and set properties.

6.2.1.2. Features built

The following features were implemented in sprint 1:

- Basic app dashboard with sections for portal advertisement and tutorial was made.
- A form for creating new app was created.
- Adding and deleting screens in the mobile frame.
- Login view component was created.
- Drag and drop feature for component was implemented.

6.2.1.3. Accomplishments

In sprint 1 version 0.1 of the portal was released.

6.2.2. Sprint 2 (05/03/18 – 16/03/18)

6.2.2.1. User Stories

The following high level user stories were completed in sprint 2:

- Interviewing people.
- App Editor Page including landing screen for app, save project.
- Sign in page.
- Services for adding screen to app and components to screen.
- Create file for postman.

6.2.2.2. Features built

The following features were implemented in sprint 2:

- Save feature for all the changes that is made in the editor page.
- Landing screen for the app.
- Sign in page for logging into the portal.

6.2.3. Sprint 3 (19/03/18 – 30/03/18)

6.2.3.1. User Stories

The following high level user stories were completed in sprint 2:

- Terms and conditions for sign in page.
- Delete existing project.
- Interviewing operations team.

- List box components (list with and without icon)
- Tooltips for properties form.
- Dell branding and theming for landing.
- Side menu template.
- Request to controller.
- UI changes in the home page.
- Include chat bot engine component.

6.2.3.2. Features built

The following features were implemented in sprint 3:

- Delete existing project.
- Include list view components (list with and without icon).
- Sign in page for logging into the portal.
- Tooltips for properties form.
- Side menu template that is to be selected while creating a new project.

Included chat bot engine component.

6.2.4. Sprint 4 (02/04/18 – 13/04/18)

6.2.4.1. User Stories

The following high level user stories were completed in sprint 2:

- Edit existing project.
- IPA generation.
- Turning website to an app.
- Graph view component.
- Audio/Video/Documents.
- Download and share.
- Preferences component.
- Storing of downloaded existing app project built.
- Dell secure authentication integration.

6.2.4.2. Features built

The following features were implemented in sprint 4:

- Edit existing project.
- IPA generation.
- Turning website to an app.
- Graph view component.
- Audio/Video/Documents.
- Download and share.
- Preferences component.

6.2.4.3. Accomplishments

In sprint 4, the version 0.2 of the portal was released.

6.2.5. Sprint 5 (16/04/18 – 04/05/18)

6.2.5.1. User Stories

The following high level user stories were completed in sprint 2:

- Deployment in Jenkins.
- Bottom navigation tab view.
- Help page overlay.
- Documentation.
- Secure authentication deployment and PCF deployment.
- Video tutorial in landing page.
- Admin dashboard for mobile experience team.

6.2.5.2. Features built

The following features were implemented in sprint 5:

- Bottom navigation tab view is included as a template that is to be selected while creating a new project.
- Help page overlay that would ease the work of user for making the app in the editor page.

- Video tutorial in landing page.
- Admin dashboard for mobile experience team that keeps track of all the details about the users and the apps created.

6.2.5.3. Accomplishments

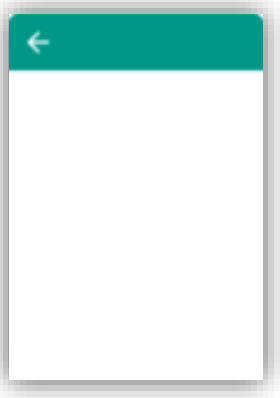
In sprint 5, the version 0.3 of the portal was released.

6.3. Templates

User can choose the template while creating a new app. The various types of templates that are available for a user to choose from are blank, sidebar, tabs and web app.

6.3.1. List of Templates

6.3.1.1. Blank Template



(6.2) Blank Template

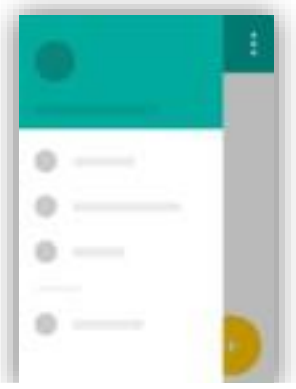
What is it?

This is a blank template. The user can choose this template to start an application fresh without any other navigation options like sidebar or tabs.

How to use it?

In the “create new app” screen, a user has to choose the template as “blank” and then click on “create” button to start an application with this template.

6.3.1.2. Sidebar Template



(6.3) Sidebar Template

What is it?

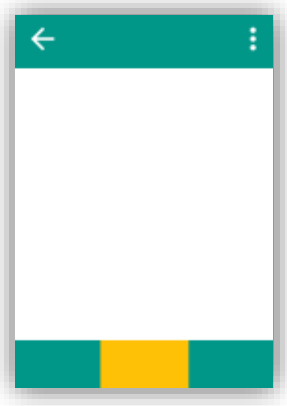
Sidebar template, also called the hamburger template is for left side navigation.

How to use it?

User has to choose sidebar template while creating a new app. Once user has chosen this template the editor page opens with nav button. After creating the app user has to click on the nav button before build. A dialog box opens for selecting the screen

and filling the screen name. Screen with Login view cannot be selected.

6.3.1.3. Tabs Template



What is it?

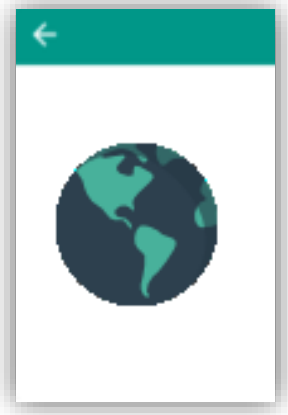
Tabs template is for bottom navigation.

How to use it?

User has to choose tabs template while creating the new app. Once user has chosen this template the editor page opens with a tab screen. User has options of choosing between 2 tabs, 3 tabs or 4 tabs. User has to select the screen and icon. Also fill screen name.

(6.4) Tabs Template

6.3.1.4. Web App Template



What is it?

Web app template is for converting website into an app.

How to use it?

User has to choose web app template while creating a new app. Once user has chosen this template, a dialog box opens asking for URL which has to be converted to app. On clicking build button, build dialogue box appears showing build process.

(6.5) Web App Template

6.4. Components

Components card contains various components. Here the user can drag the components from the component card and drop into the mobile screen. One component can be replaced by another component. Various type of components are Login view, List catalogue, Simple list view, Chat bot and Preferences.

6.4.1. List of Components

6.4.1.1. Login View

What is it?

The login view is a component with a login form (username, password and a submit button). If a user wants a login form in their application, then this component can be used. It expects the user credentials which are typically in the form of "username" and a matching "password", which helps in authentication for the application that the user is building.

How to use it?

The login view is placed in the component card in the left hand side of the editor page. If required, this component should be dragged from the component card and dropped in the mobile screen. Each component has some properties associated with it. The list of properties are present in the right hand side of the editor page and will be visible when the component. The user is supposed to fill the property details for each component. The list of properties of login view with their description and examples are as follows (Table 6.1):

(6.1) Property form for Login component

S.NO	Property Name	Property Description	Example	Remarks
1.	REST Type	What kind of HTTP request is the API using? (GET,PUT,POST DELETE)	POST	This field is fixed as POST in v1.beta.
2.	URL	The URL where the API that handles login is present.	*http://emc/my-server/api/2.2/sampleauth/signin*	It is a mandatory field.
3.	Data Type	Type of data that is expected to be found in API.(JSON,key value pair,plain text,XML)	JSON	Type is fixed as JSON in v1.beta..
4.	Header	Additional header that user might add with the API call.	{"apiKey": "Jsfhjsdhfjs12dhf"}	It is not a mandatory field (can be left blank) and should be in JSON format.
5.	Additional Content	Data that user might want to	{"User-Type": "manager"}	It is not a mandatory

		send with API call in POST Body.		field (can be left blank) and should be JSON format.
6.	Next Screen	Here the screen that comes after the login screen is selected.	Screen2	It is a mandatory field.

The API specifications are:

The API is expected to handle a post request with the following specifications:

Post Header

The header of the post request that is sent from the app has the following json (by default):

```
{
   "Content-Type":"application/json"
}
```

Any additional header that is provided in the properties form will be appended in this json.

Post Body

The body of the post request that is sent from the app has the following json:

```
{
   "username”: input_from_username_field",
```

```
  "password”: input_from_password_field"
}
```

Expected response:

The expected response from API:

1. For a successful authentication:

```
{
  "status”: “success”
}
```

2. For unsuccessful authentication:

```
{
  "status”:” failure”
}
```

6.4.1.2. List Catalogue

What is it?

In List Catalogue, list with icon, title, description and other data are present. When we click on a particular list field the description page opens with icon, title, description, back button, download button and share button. If a user wants a list view with icon in their application, then this component can be used.

How to use it?

The list catalogue is placed in the component card in the left hand side of the editor page. If required, this component should be dragged from the component card and dropped in the mobile screen. Each component has some properties associated with it. The list of properties are present in the right hand side of the editor page and will be visible when the component. The user is supposed to fill the property details

for each component. The list of properties of login view with their description and examples are as follows (Table 6.2):

(6.2) Property form for List Catalogue component

S.NO	Property Name	Property Description	Example	Remarks
1.	REST Type	What kind of HTTP request is the API using? (GET,PUT,POST DELETE)	GET	This field is fixed as GET in v1.beta.
2.	URL	The URL where the data is present.*	*http://emc/my-server/api/2.2/sampleauth/signin*	It is a mandatory field.
3.	Data Type	Type of data that is expected to be found in API.(JSON,key value pair,plain text,XML)	JSON	Type is fixed as JSON in v1.beta.
4.	Header	Additional header that user might add with the API call.	{"apiKey": "Jsfhjsdhfjs12dhf"}	It is not a mandatory field (can be left blank) and should be in JSON format.

*The API specifications are:

The API will be expected to handle a GET request of the following specifications:

The general structure of the API is:

```
[
 {section_details},
 {section_details}
]
```

An Array containing different section_details.

If sections are not required then just have one element inside the array.

Each of these section details are of the format:

```
{
 "title”: Section_title"
 "values":[
 {list_content_for_this_section},
 {list_content_for_this_section}]}
```

Each of these list content will be of the format:

1.For Description (with download and share)

```
{
"desc_type":"description" ,
"img":"url_of_any_image",
```

```
"title":"Sample_title",
"smallDesc":"Sample Description.",
"version”: “sample_version",
"description":"Sample Desc.",
"share":["share_section”, “share_section"],
"message”: “some message",
"subject”: “some subject",
"name”: sampleName",
"downloads”: “download_section"
}
```


The details of this list content are as follows(Table 6.3):

(6.3) API details for list content

S.No	List content	Description	Example
1.	desc_type	Type of content that will be displayed in the description page. For a description page this field should be "description".	----------

2.	img	It is the icon of each list content that is displayed on the left side of the title.	"*http://images/DellEMC.png*"
3.	title	Name of the list content.	"DellEMC Mobile"
4.	smallDesc	It is a content that will be displayed below the title in the list.	"It is a mobile app"
5.	version	It is a single line content that will be displayed on the right of the title.	----------
6.	description	It is the description of the respective list content that will be displayed	"sample description"

		once the list content is clicked.	
7.	share	url of the file that is to be shared. Share can be an array as multiple files can be shared.	"*https://i.imgur.com/qBnk1Bh.jpg*"
8.	message	Content that will be shared	
9.	subject	Title of the content that will be shared	
10.	name	Name of the file (if any) with extension.	"name.pdf"
11.	downloads	URL of the file that is to be downloaded.	"*https://i.imgur.com/qBnk1Bh.jpg*"

2. For Description (without download and share)

```
{
"desc_type":"description”,
"img":"url_of_any_image",
"title":"Sample_title",
"smallDesc":"Sample Description.",
"version":"sample_version",
"description":"Sample Desc."
}
```


The details of this list content are as follows (Table 6.4):

(6.4) API definition for text description view

S.No	List content	Description	Example
1.	desc_type	Type of content that will be displayed in the description page. For a description page this field should be "description".	----------
2.	Img	It is the icon of each list content that is displayed on the left side of the title.	"*http://images/DellEMC.png*"
3.	Title	Name of the list content.	"DellEMC Mobile"
4.	smallDesc	It is a content that will be displayed below the title in the list.	"It is a mobile app"
5.	version	It is a single line content that will be displayed on the right of the title.	----------

6.	description	It is the description of the respective list content that will be displayed once the list content is clicked.	"sample description"

3. For Audio

```
{
 "desc_type":"audio",
 "img":"url_of_any_image",
 "title":"sample_title",
 "version":"sample_version",
 "smallDesc":"This is a sample audio",
 "url":"sample_url"
}
```

The details of this list content are as follows (Table 6.5):

(6.5) API definition for Audio support

S.No	List content	Description	Example
1.	desc_type	Type of content that will be displayed in the description page. For a description page this field should be "audio".	----------
2.	Img	It is the icon of each list content that is displayed on the left side of the title.	"*http://images/DellEMC.png*"

3.	Title	Name of the list content.	"DellEMC Mobile"
4.	smallDesc	It is a content that will be displayed below the title in the list.	"It is a mobile app"
5.	version	It is a single line content that will be displayed on the right of the title.	----------
6.	url	It contains details about audio.	"*http://images/DellEMC.mp3*"

1. For Video

```
{
"desc_type":"video",
"img":"url_of_any_image",
"title":"sample_title",
"version":"sample_version",
"smallDesc":"This is a sample video",
"url":"video_url"

}
```

The details of this list content are as follows (Table 6.6):

(6.6) API definition for video support

S.No	List content	Description	Example
1.	desc_type	Type of content that will be displayed in the description page. For a description page this field should be "description".	----------
2.	img	It is the icon of each list content that is displayed on the left side of the title.	"*http://images/DellEMC.png*"
3.	title	Name of the list content.	"DellEMC Mobile"
4.	smallDesc	It is a content that will be displayed below the title in the list.	"It is a mobile app"
5.	version	It is a single line content that will be displayed on the right of the title.	----------
6.	url	It contains details about video	"*http://images/DellEMC..mp4*"

2. For Graphs

```
{
 "desc_type":"graph",
 "img":"url_of_any_image",
 "title":"sample_title",
 "smallDesc":"sample description",
```

```
 "version":"sample_description",
 "description":"Sample Description",
 "graphs":[
        {graph_content},
        {graph_content}
       ]
}
```

The details of this list content are as follows (Table 6.7):

(6.7) API definition for graph support in list

S.No	List content	Description	Example
1.	desc_type	Type of content that will be displayed in the description page. For a description page this field should be "description".	----------
2.	img	It is the icon of each list content that is displayed on the left side of the title.	"*http://images/DellEMC.png*"
3.	title	Name of the list content.	"DellEMC Mobile"
4.	smallDesc	It is a content that will be displayed below the title in the list.	"It is a mobile app"
5.	version	It is a single line content that will be displayed on the right of the title.	----------

6.	description	It is the description of the respective list content that will be displayed once the list content is clicked.	"sample description"
7.	graphs	It contains data of graph	{ "type": bar", "url": *http://graphs/get/data/bar*", "title":"Bar Graph" }

Each of these graph content will be of the format:

```
{

 "type":"graph_type",
 "url":"url_of_the_data",
 "title":"sampleTitle"
}
```

The details of this graph content are as follows (Table 6.8):

(6.8) API definition for graph

S.No	Graph content	Description	Example
1.	type	Type of content that will be displayed in the description page. For a description page this field should be "description".	----------
2.	url	It contains data of graph	"*http://graphs/get/data/bar*"

3.	Title	Name of the list content.	"DellEMC Mobile"

The URL should contain the data for the graph in the following format:

```
{
 "data": []
 "labels":[]
}
```

An example:

For making a list view with 2 videos, 2 audios

```
[
 {"title":"Video",
  "values":[
      {"desc_type":"video",
       "img":"url_of_any_image",
       "title":"sample_title",
       "version":"sample_version",
       "smallDesc":"This is a sample video",
       "url":"video_url"
      },
      {"desc_type":"video",
       "img":"url_of_any_image",
       "title":"sample_title",
       "version":"sample_version",
       "smallDesc":"This is a sample video",
```

```
        "url":"video_url"
       }
 ]
},
{"title":"Audio",
 "values":[
      {"desc_type":"audio",
       "img":"url_of_any_image",
       "title":"sample_title",
       "version":"sample_version",
       "smallDesc":"This is a sample audio",
       "url":"sample_url"
      },
      {"desc_type":"audio",
       "img":"url_of_any_image",
       "title":"sample_title",
       "version":"sample_version",
       "smallDesc":"This is a sample audio",
       "url":"sample_url"
      }
 ]
},
 {"title":"Normal Description",
  "values":[
        {"desc_type":"description" ,
         "img":"url_of_any_image",
         "title":"sample_title",
         "smallDesc":"Sample Description.",
         "version":"sample_version",
         "description":"Sample Desc.",
```

```
        "share":["share_url","share_url"],
        "message":"some message",
        "subject":"some_subject",
        "name":"sampleName",
        "downloads":"download_url"
       },

       {"desc_type":"description" ,
        "img":"url_of_any_image",
        "title":"sample_title",
        "smallDesc":"Sample Description.",
        "version":"sample_version",
        "description":"Sample Desc.",
        "downloads":"download_url",
        "message":"some message",
       "subject":"some_subject",
        "name":"sampleName",
        "share":"share_url"
       },

       {"desc_type":"description" ,
        "img":"url_of_any_image",
        "title":"sample_title",
        "smallDesc":"Sample Description.",
        "version":"sample_version",
        "description":"Sample Desc."
       }
     ]
},
```

```
{"title":"Graphs",
 "values":[
     {"desc_type":"graph",
      "img":"url_of_any_data",
      "title":"sample_title",
      "smallDesc":"sample_description",
      "version":"sample_version",
      "description":"Sample Description",
      "graphs":[
        {"type":"graph_type",
         "title":"graph_title",
         "url":"url_of_any_data"
        },
        {"type":"graph_type",
         "title":"graph_title",
         "url":"url_of_any_data"
        }
      ]
     },
     {"desc_type":"graph",
      "img":"url_of_any_data",
      "title":"sample_title",
      "smallDesc":"sample_description",
      "version":"sample_version",
      "description":"Sample Description",
      "graphs":[
        {"type":"graph_type",
         "title":"graph_title",
         "url":"url_of_any_data"
```

```
        }
       ]
      }
     ]
   }
]
```

6.3.1.3 Simple List View:

What is it?

In List View, list with title, description and other data are present. When we click on a particular list field the description page opens with icon, title, description, back button, download button and share button. If a user wants a list view without icon in their application, then this component can be used.

How to use it?

The list view is placed in the component card in the left hand side of the editor page. If required, this component should be dragged from the component card and dropped in the mobile screen. Each component has some properties associated with it. The list of properties are present in the right hand side of the editor page and will be visible when the component. The user is supposed to fill the property details for each component. The list of properties of login view with their description and examples are as follows (Table 6.9):

(6.9) Property form for Simple List component

S.No	Property Name	Property Description	Example	Remarks
1.	REST Type	What kind of HTTP request is the API using? (GET,PUT,POST DELETE)	GET	This field is fixed as GET in v1.beta.
2.	URL	The URL where the data is present.*	*http://emc/my-server/api/2.2/sampleauth/signin*	It is a mandatory field.
3.	Data Type	Type of data that is expected to be found in API.(JSON,key value pair,plain text,XML)	JSON	Type is fixed as JSON in v1.beta..
4.	Header	Additional header that user might add with the API call.	{"apiKey": "Jsfhjsdhfjs12dhf"}	It is not a mandatory field (can be left blank) and should be in JSON format.

*The API specifications are:

The API will be expected to handle a GET request of the following specifications:

The general structure of the API is:

```
[
 {section_details},
 {section_details}
]
```

An Array containing different section_details.

If sections are not required then just have one element inside the array.

Each of these section details are of the format:

```
{
 "title":"Section_title"
 "values":[
 {list_content_for_this_section},
 {list_content_for_this_section}
 ]
}
```

Each of these list content will be of the format:

1. For Description (with download and share)

```
{
"desc_type":"description",
"title":"DellEMC Mobile",
```

```
"smallDesc":"Sample Description.",
"version":"ver",
"description":"Sample Desc.",
"share": ["share_section","share_section"],
"message":"some message",
"subject":"some_subject",
"name":"sampleName.png",
"downloads":"download_section"
}
```


The details of this list content are as follows (Table 6.10):

(6.10) API definition for text description

S.No	List content	Description	Example
1.	desc_type	Type of content that will be displayed in the description page. For a description page this field should be "description".	----------
2.	title	Name of the list content.	"DellEMC Mobile"

3.	smallDesc	It is a content that will be displayed below the title in the list.	"It is a mobile app"
4.	version	It is a single line content that will be displayed on the right of the title.	----------
5.	description	It is the description of the respective list content that will be displayed once the list content is clicked.	"sample description"
6.	share	URL of the file that is to be shared. Share can be an array as multiple files	"*https://i.imgur.com/qBnk1Bh.jpg*"

		can be shared.	
7.	message	Content that will be shared	
8.	subject	Title of the content that will be shared	
9.	name	Name of the file (if any) with extension.	"name.pdf"
10.	downloads	URL of the file that is to be downloaded.	"*https://i.imgur.com/qBnk1Bh.jpg*"

2. For Description (without download and share)

```
{
"desc_type":"description",
"title":"Help a Customer",
"smallDesc":"Sample Description.",
"version":"v1.1.4",
"description":"Sample Desc."
}
```

The details of this list content are as follows (Table 6.11):

(6.11) API definition for text description without download and share

S.No	List content	Description	Example
1.	desc_type	Type of content that will be displayed in the description page. For a description page this field should be "description".	----------
2.	title	Name of the list content.	"DellEMC Mobile"
3.	smallDesc	It is a content that will be displayed below the title in the list.	"It is a mobile app"
4.	version	It is a single line content that will be displayed on the right of the title.	----------
5.	description	It is the description of the respective list content that will be displayed once the list content is clicked.	"sample description"

3. For Audio

```
{
  "desc_type":"audio",
  "title":"DellEMC Mobile",
  "version":"1.1.12",
  "smallDesc":"This is a sample audio",
  "url":"sample_url"
}
```

The details of this list content are as follows (Table 6.12):

(6.12) API definition for Audio support

S.No	List content	Description	Example
1.	desc_type	Type of content that will be displayed in the description page. For a description page this field should be "audio".	----------
2.	title	Name of the list content.	"DellEMC Mobile"
3.	smallDesc	It is a content that will be displayed below the title in the list.	"It is a mobile app"
4.	version	It is a single line content that will be displayed on the right of the title.	----------
5.	url	It contains details about audio.	"*http://images/DellEMC.mp3*"

4. For Video

```
{
"desc_type":"video",
"title":"DellEMC Mobile",
"version":"10.10.2018",
"smallDesc":"This is a sample video",
"url":"video_url"
}
```

The details of this list content are as follows (Table 6.13):

(6.13) API definition for Video support

S.No	List content	Description	Example
1.	desc_type	Type of content that will be displayed in the description page. For a description page this field should be "description".	----------
2.	title	Name of the list content.	"DellEMC Mobile"
3.	smallDesc	It is a content that will be displayed below the title in the list.	"It is a mobile app"
4.	version	It is a single line content that will be displayed on the right of the title.	----------
5.	url	It contains details about video	"*http://images/DellEMC..mp4*"

5. For Graphs

```
{
 "desc_type":"graph",
 "title":"DellEMC Mobile",
 "smallDesc":"Dell EMC Mobile is your companion for ",
 "version":"v3.4.4",
 "description":"Sample Description",
 "graphs":[
        {graph_content},
        {graph_content}
      ]
}
```

The details of this list content are as follows (Table 6.14):

(6.14) API definition for graph support

S.No	List content	Description	Example
1.	desc_type	Type of content that will be displayed in the description page. For a description page this field should be "description".	----------
2.	title	Name of the list content.	"DellEMC Mobile"
3.	smallDesc	It is a content that will be displayed below the title in the list.	"It is a mobile app"
4.	version	It is a single line content that will be displayed on the right of the title.	----------
5.	description	It is the description of the respective list content that will be displayed once the list content is clicked.	"sample description"
6.	graphs	It contains data of graph	{ "type": bar", "url": *http://graphs/get/data/bar*", "title":"Bar Graph" }

Each of these graph content will be of the format:

```
{
 "type":"bar",
 "url":"dellemc.com/apac/get/data/bar",
 "title":"sampleTitle"
}
```

The details of this graph content are as follows (Table 6.15):

(6.15) API definition for graph

S.No	Graph content	Description	Example
1.	type	Type of content that will be displayed in the description page. For a description page this field should be "description".	----------
2.	url	It contains data of graph	"*http://graphs/get/data/bar*"
3.	title	Name of the list content.	"DellEMC Mobile"

An example:

```
[
 {"title":"Video",
  "values":[
      {"desc_type":"video",
       "title":"sample_title",
       "version":"sample_version",
       "smallDesc":"This is a sample video",
       "url":"video_url"
```

```
        },
        {"desc_type":"video",
         "title":"sample_title",
         "version":"sample_version",
         "smallDesc":"This is a sample video",
         "url":"video_url"
        }
    ]
},
{"title":"Audio",
 "values":[
        {"desc_type":"audio",
         "title":"sample_title",
         "version":"sample_version",
         "smallDesc":"This is a sample audio",
         "url":"sample_url"
        },
        {"desc_type":"audio",
              "title":"sample_title",
         "version":"sample_version",
         "smallDesc":"This is a sample audio",
         "url":"sample_url"
        }
 ]
},
 {"title":"Normal Description",
  "values":[
        {"desc_type":"description" ,
         "title":"sample_title",
```

```
        "smallDesc":"Sample Description.",
        "version":"sample_version",
        "description":"Sample Desc.",
    "share":["share_url","share_url"],
        "message":"some message",
        "subject":"some_subject",
        "name":"sampleName",
        "downloads":"download_url"
       },

       {"desc_type":"description",
        "title":"sample_title",
        "smallDesc":"Sample Description.",
        "version":"sample_version",
        "description":"Sample Desc.",
        "downloads":"download_url",
        "message":"some message",
      "subject":"some_subject",
        "name":"sampleName",
        "share":"share_url"
       },

       {"desc_type":"description" ,
        "title":"sample_title",
        "smallDesc":"Sample Description.",
        "version":"sample_version",
        "description":"Sample Desc."
       }
     ]
},
```

```
{"title":"Graphs",
 "values":[
     {"desc_type":"graph",
      "title":"sample_title",
      "smallDesc":"sample_description",
      "version":"sample_version",
      "description":"Sample Description",
      "graphs":[
        {"type":"graph_type",
         "title":"graph_title",
         "url":"url_of_any_data"
        },
        {"type":"graph_type",
         "title":"graph_title",
         "url":"url_of_any_data"
        }
      ]
     },
     {"desc_type":"graph",
      "title":"sample_title",
      "smallDesc":"sample_description",
      "version":"sample_version",
      "description":"Sample Description",
      "graphs":[
        {"type":"graph_type",
         "title":"graph_title",
         "url":"url_of_any_data"
        }
```

```
        ]
       }
      ]
   }
]
```

6.3.1.4 Chat Bot:

What is it?

A chat bot is a computer program which conducts a conversation via textual methods. Such a bot is an automated system of communication with users. Training chat bot can be done in two ways. One way is by providing URL in which training data is present. Other way is by providing question and answers in training data. If a user wants Chat Bot in their application, then this component can be used.

How to use it?

The Chat Bot is placed in the component card in the left hand side of the editor page. If required, this component should be dragged from the component card and dropped in the mobile screen. Each component has some properties associated with it. The list of properties are present in the right hand side of the editor page and will be visible when the component. The user is supposed to fill the property details for each component. The list of properties of login view with their description and examples are as follows (Table 6.16):

(6.16) Property form for Chatbot component

S.No	Property Name	Property Description	Example	Remarks
1.	REST Type	What kind of HTTP request is the API using? (GET,PUT,POST DELETE)	GET	This field is fixed as GET in v1.beta.

2.	URL	The URL where the training data is present.*	*http://emc/my-server/api/2.2/sampleauth/signin*	It is a mandatory field.
3.	Data Type	Type of data that is expected to be found in API.(JSON,key value pair,plain text,XML)	JSON	Type is fixed as JSON in v1.beta..
4.	Header	Additional header that user might add with the API call.	{"apiKey": "Jsfhjsdhfjs12dhf"}	It is not a mandatory field (can be left blank) and should be in JSON format.
5.	Training Data	Data for training chat bot is added.	{"What is SSP?": "Self Service Portal"}	It is not a mandatory field (can be left blank) and should be JSON format.

*The API specifications are:

The API will be expected to handle a GET request of the following specifications:

Expected response from the API:

```
{
    "res":"The response from the chatbot"
    "isMe":"false"
    "image":"assets/imgs/chatbot.png"
    "time":"timestamp"
}
```

A description of the chatbot api is as follows (Table 6.17):

(6.17) API definition for chatbot

S.No	Content	Description	Example
1.	res	Reply that is expected from the chat bot.	
2.	isME	This field is always false.	---------
3.	image	Icon used for chat bot. Default icon can be used but if a customized icon is needed, one can host the icon in the API.	"*https://i.imgur.com/qBnk1Bh.jpg*"
4.	time	Current time.	

6.3.1.5 Preferences:

What is it?

Preferences page consists of Feedback, Help, Recommend and About section. If a user wants Preferences in their application, then this component can be used.

How to use it?

The Preferences is placed in the component card in the left hand side of the editor page. If required, this component should be dragged from the component card and dropped in the mobile screen. Each component has some properties associated with it. The list of properties are present in the right hand side of the editor page and will be visible when the

component. The user is supposed to fill the property details for each component. The list of properties of login view with their description and examples are as follows (Table 6.18):

(6.18) Property form for Preferences component

S.No	Property Name	Property Description	Example	Remarks
1.	App Description	It contains the description of the App.	EZ Build – This portal will enable business teams to develop simple mobile apps, without deep knowledge of programming. Through this portal, you can easily drag and drop different functionalities and components into the application as needed.	It is a mandatory field and can be written in plain text format.
2.	Help Content	It contains the steps on how the SSP Portal can be used.	Step 1: Drag the component in the screen. Step 2: Click on plus icon to add the screens. Step 3: Enter the property fields of the component added.	It is a mandatory field and can be written in plain text format.

7. Product Release

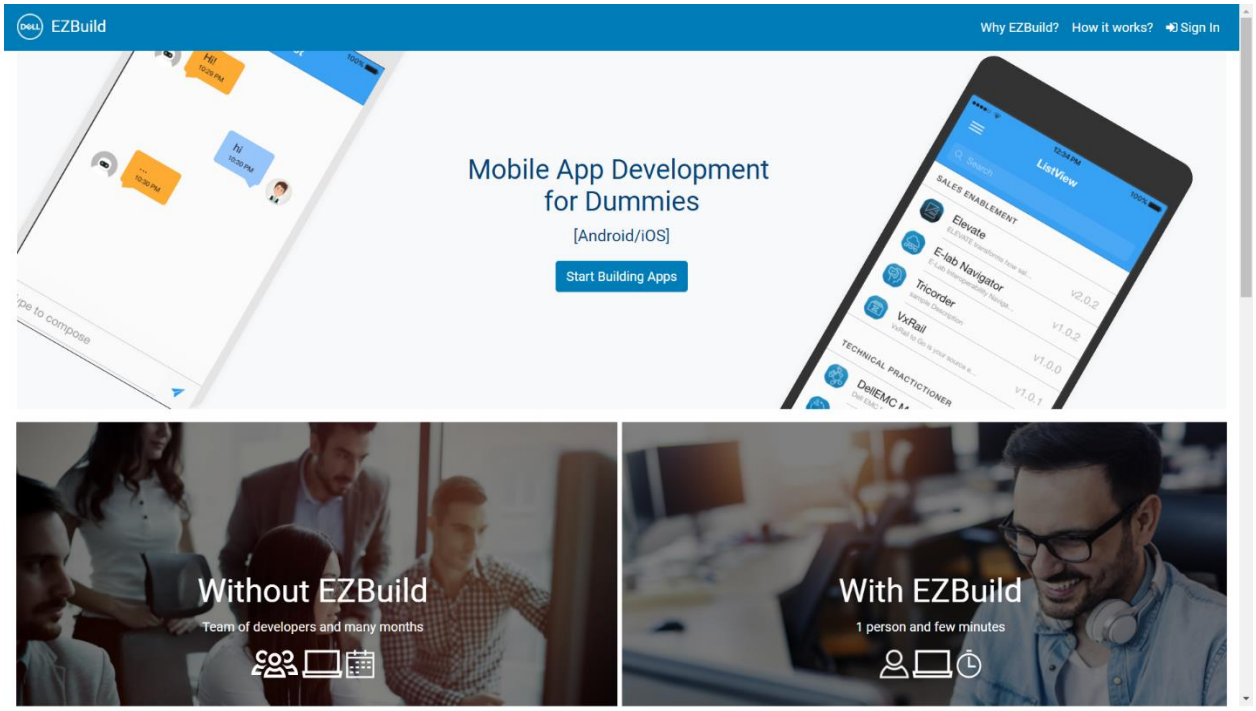
7.1 Deployment

The Self Service Portal was successfully deployed in Pivotal Cloud Foundry (PCF) with all the required security and authentication features implemented.

Screenshots:

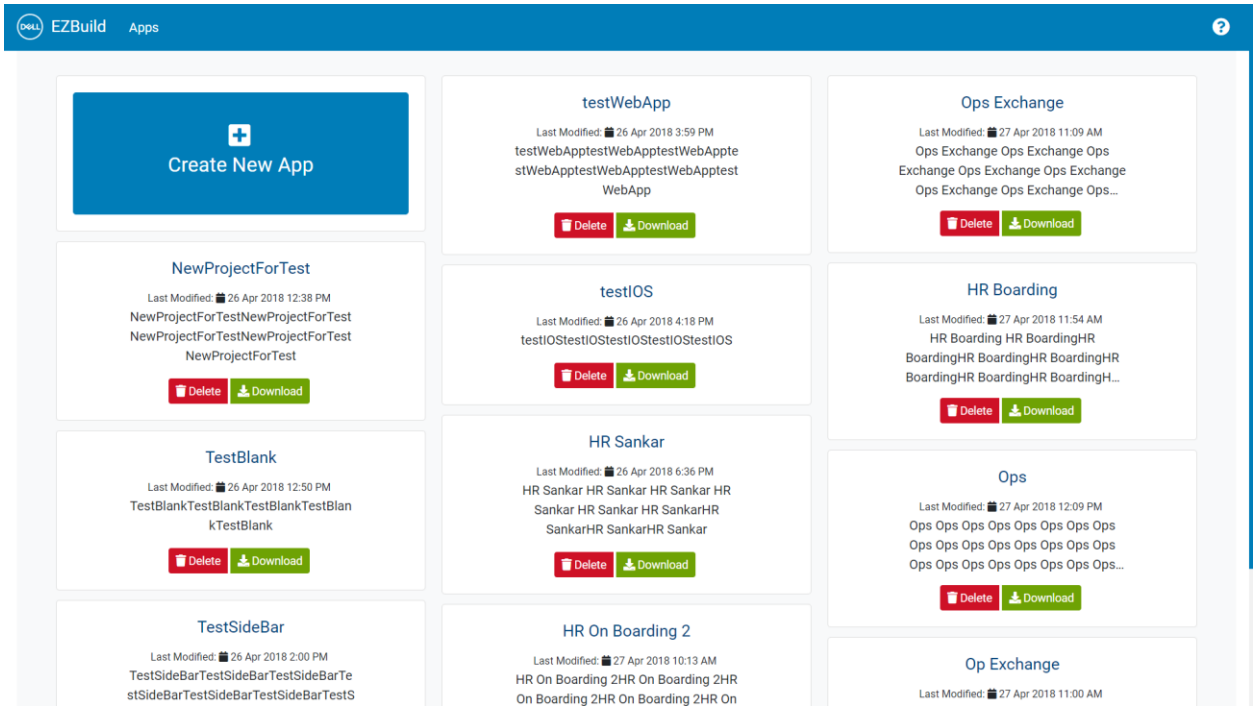
Some screenshots of the portal after deployment:

1. Landing Page (fig.7.1)



(7.1) Landing Page of EZBuild

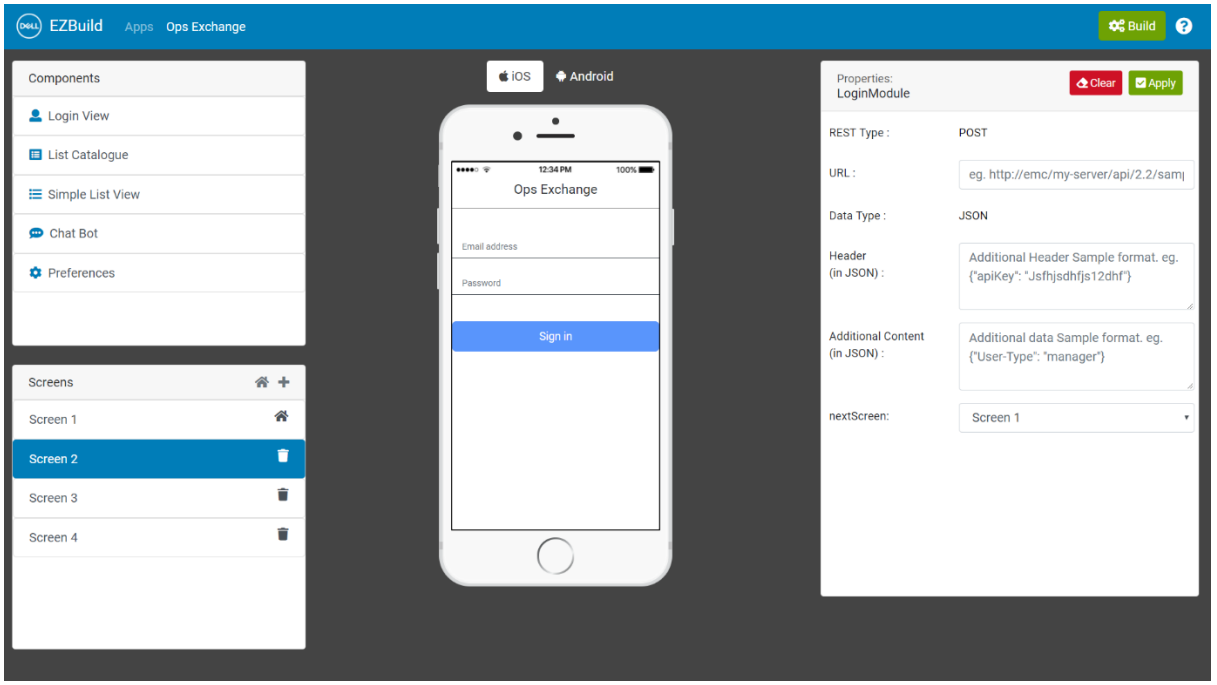
2. List of Apps Page (fig.7.2)



(7.2) App Dashboard

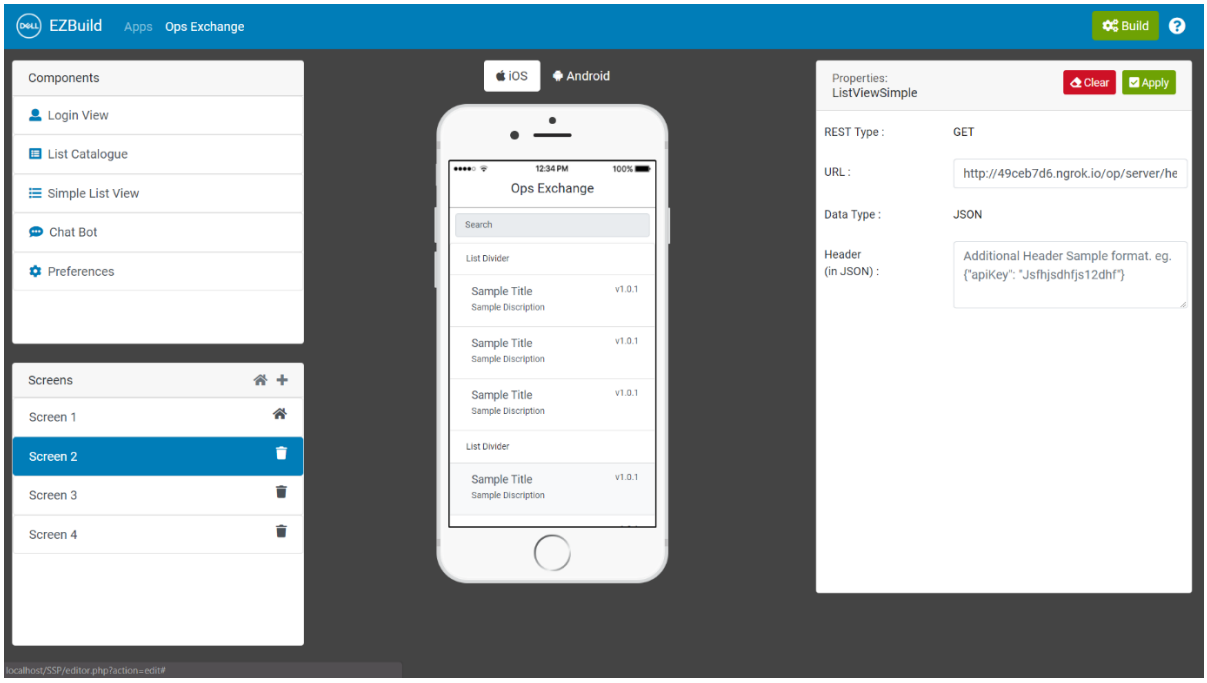
3. Editor Page

3.1 With Login Module in the mobile screen (fig.7.3)



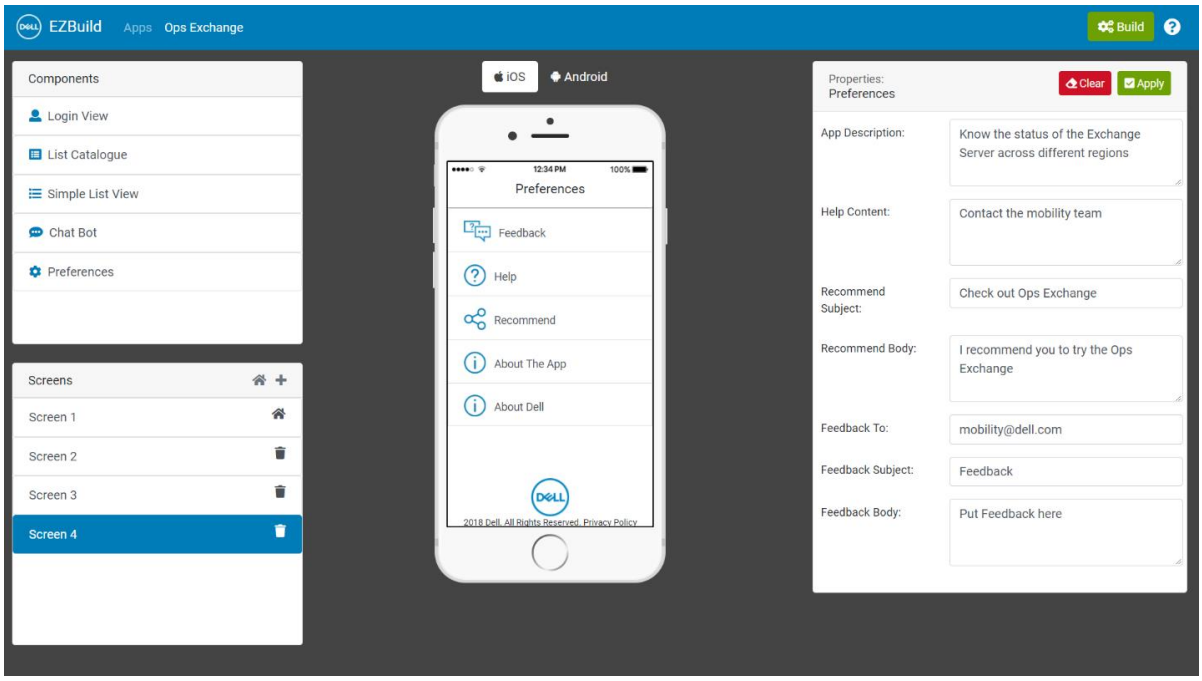
(7.3) Editor Page with login module

3.2 With a Simple List view in the mobile screen (fig.7.4)



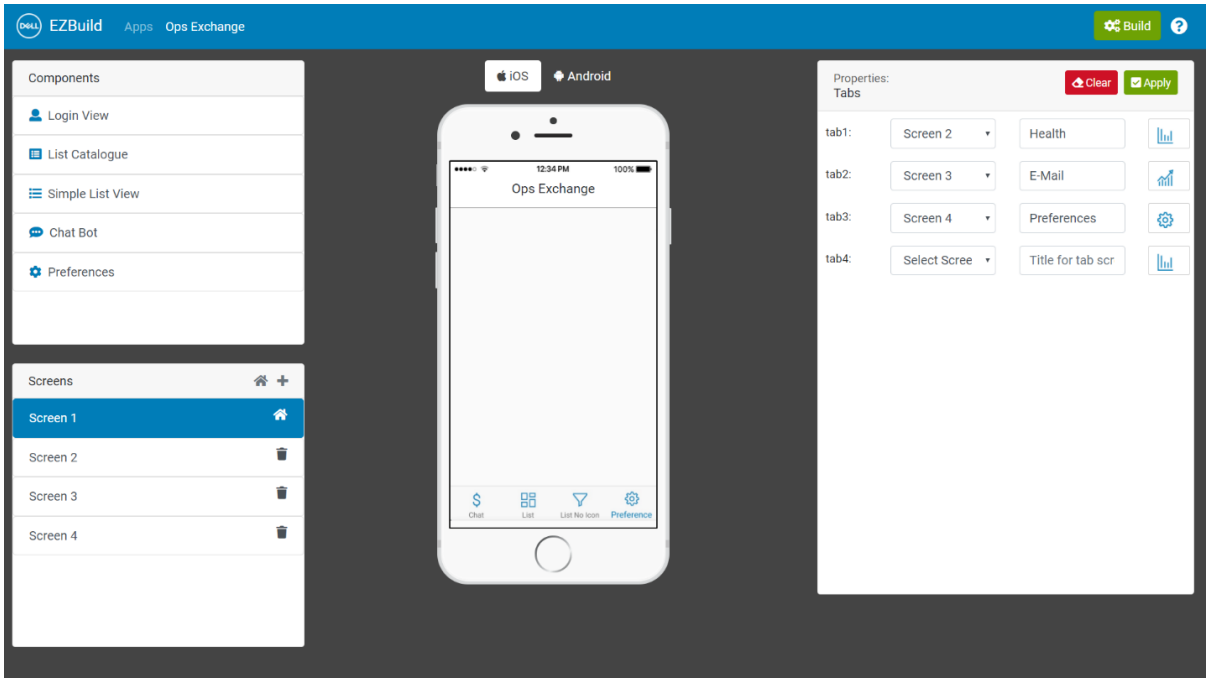
(7.4) Editor Page with simple list view component

3.3 With a Preferences Page in the mobile screen (fig.7.5)



(7.5) Editor Page with preferences component

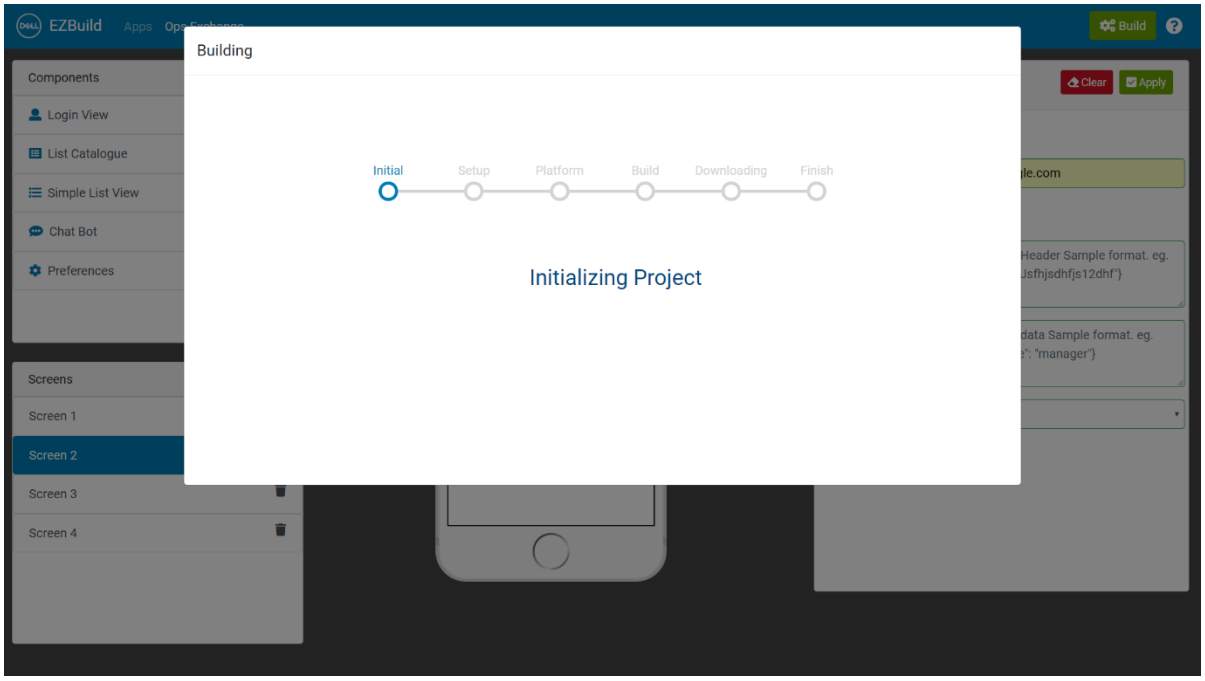
3.4 With a Tabs screen in the mobile screen (fig.7.6)



(7.6) Editor Page with Tabs screen

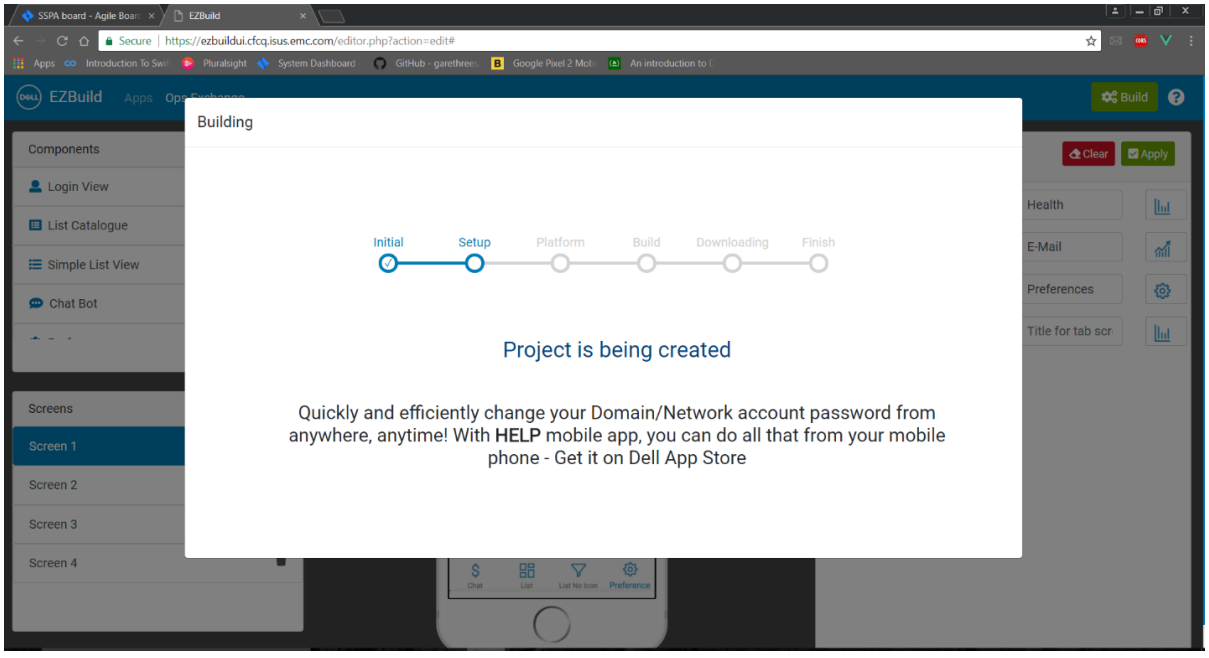
4. Build Dialog

4.1 Initializing Project (fig.7.7)



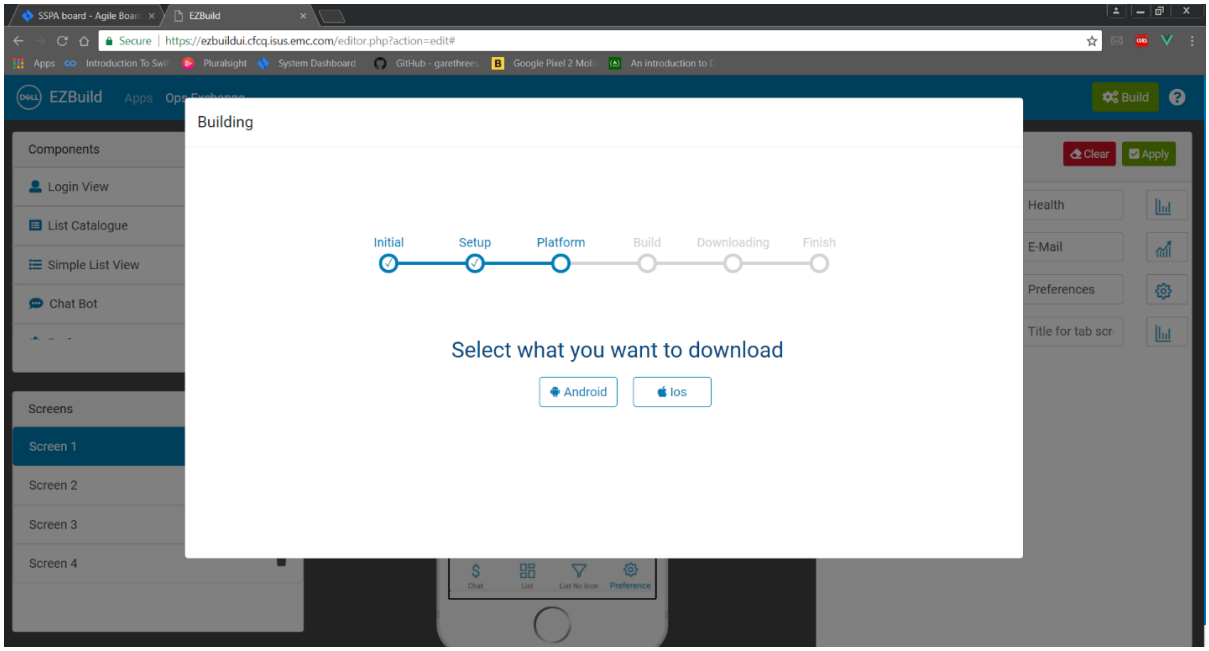
(7.7) Build start

4.2 Project Setup (fig.7.8)



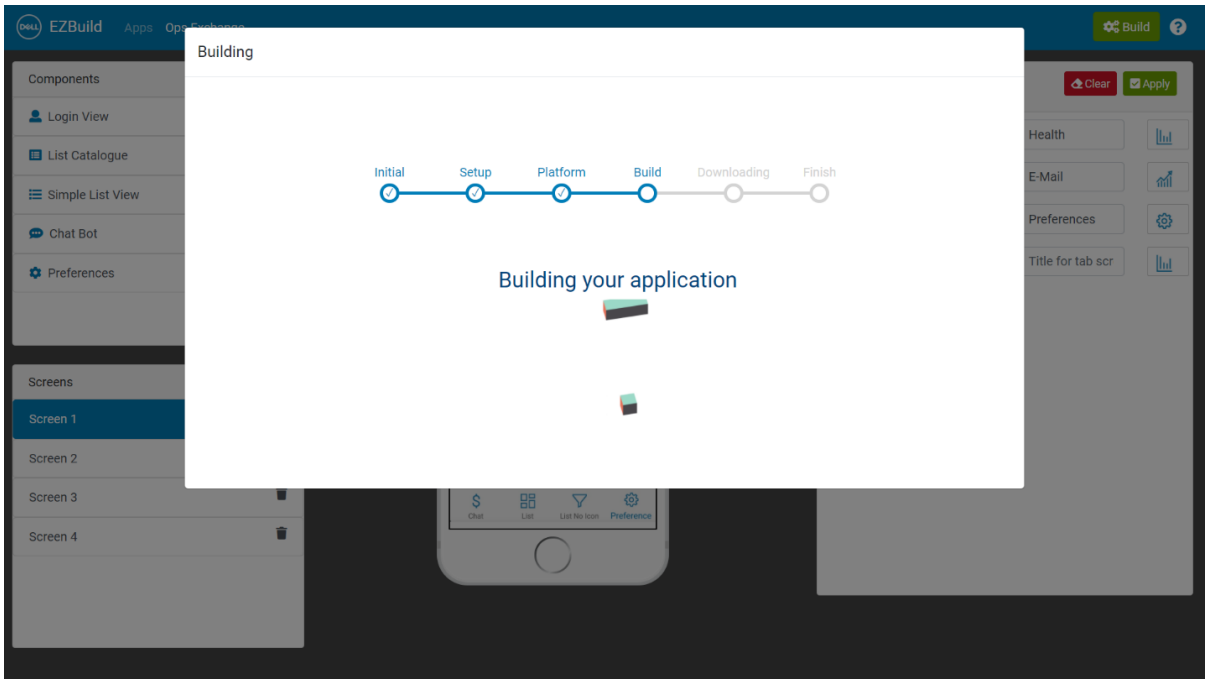
(7.8) Setup the project

4.3 Choose Platform (fig.7.9)



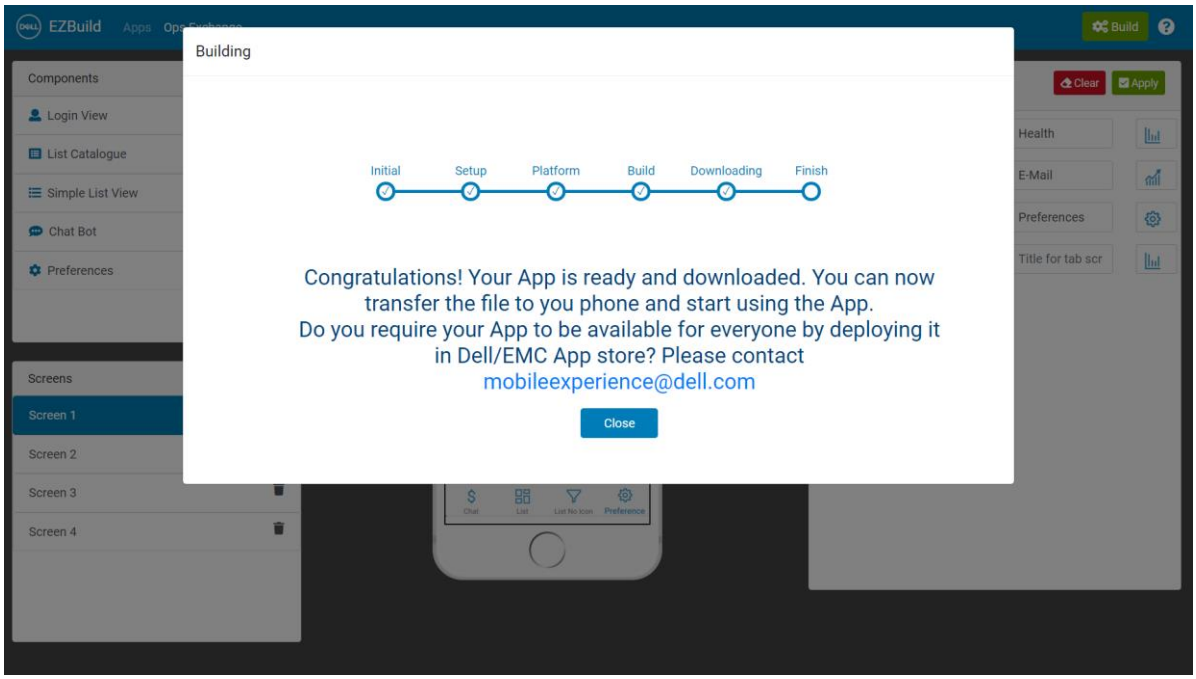
(7.9) choose which platform to build for

4.4 Building Project (fig.7.10)



(7.10) Build for the chosen platform

4.5 Download Project and Finish (fig.7.11)



(7.11) Download and finish

7.2 Business use case 1 – Spaces Chat (Mobile Experience Team)

7.2.1 Introduction

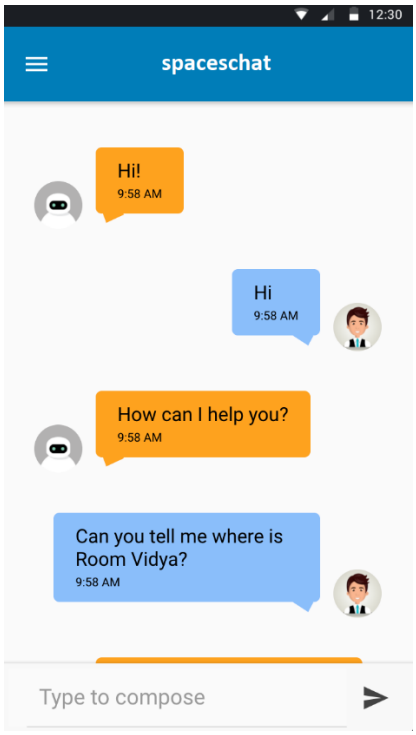
As per our interview with the mobile experience team, we came up with components such that one of their use cases could be fulfilled. So we made them make a chatbot app using the self-service portal.

7.2.2 Features

The chatbot app called “Spaces Chat” lets a user find meeting rooms, library or the cafeteria through a simple conversation. They can just ask questions like “Where’s the cafeteria?” or “Where’s the meeting room A?”, and the chatbot will reply to them with directions to reach that place. This app will save time for employees that they waste while navigating through different rooms in big offices.

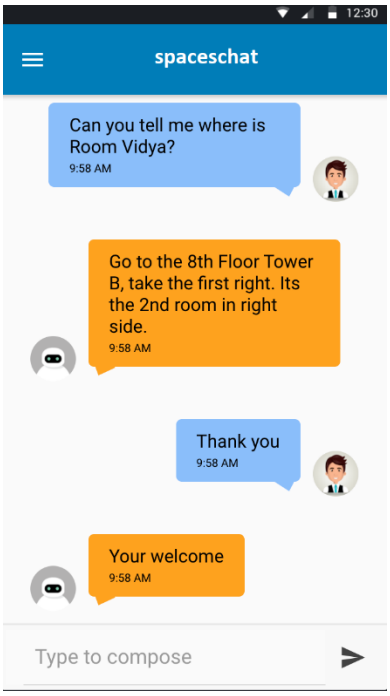
7.2.3 Screenshots:

7.2.3.1 Chatbot for Spaces India Application (fig.7.12)



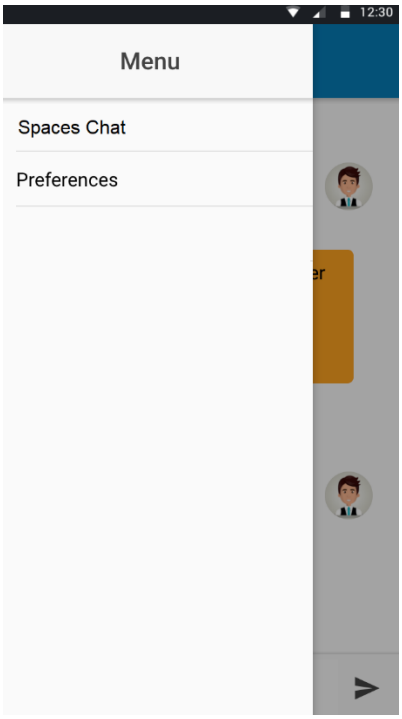
(7.12) Chatbot for Spaces India

7.2.3.2 Chatbot giving directions (fig.7.13)



(7.13) Chatbot answering with directions

7.2.3.3 The Side Menu with preferences (fig.7.14)



(7.14) Side menu for spaceschat

7.3 Business use case 2 – Onboarding App (HR)

7.3.1 Introduction

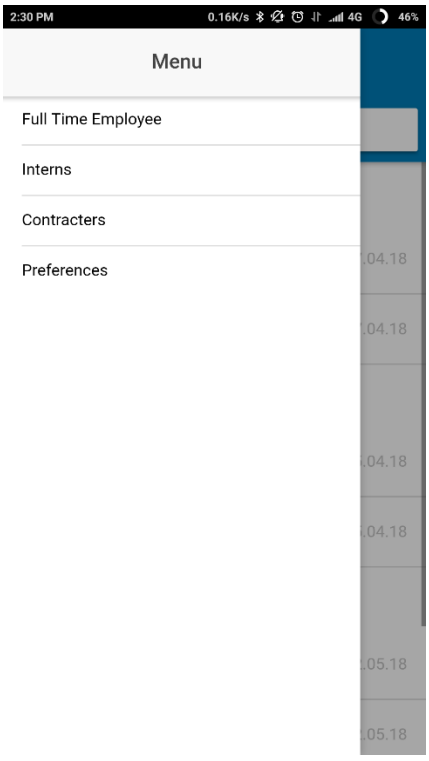
As per our interview with the HR hiring team, we came up with components such that one of their use cases could be fulfilled. So we made them make an onboarding app using the self-service portal.

7.3.2 Features

It has a side menu with FTE, Intern and Contract as elements. These elements are list items with icons. List items are divided into different sections based on time like one week ago, this week, one week from now etc. Each list item comprises of description (details of each employee like role, team, office, hiring manager), title (name of the employee), icon (image of the employee) and version (date of joining).

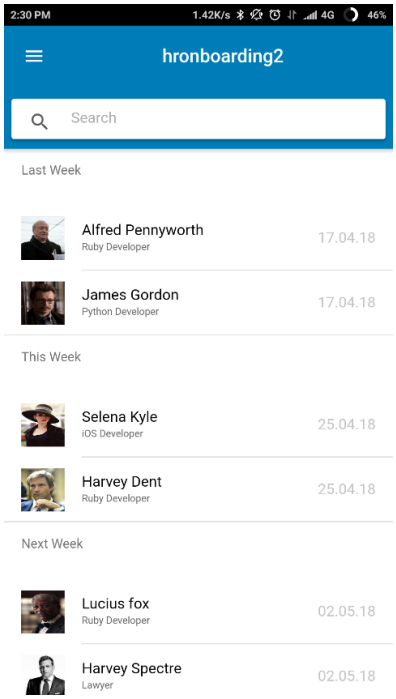
7.3.3 Screenshots:

7.3.3.1 The Side Menu (fig.7.15)



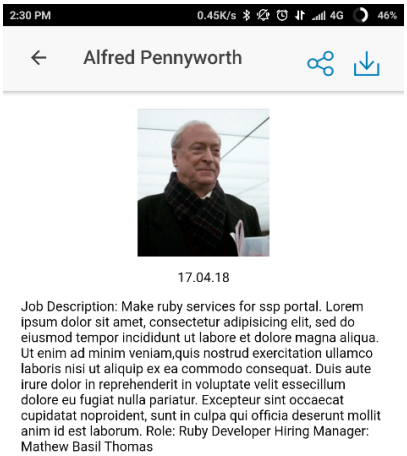
(7.15) Side Menu in HR Onboarding app

7.3.3.2 The List View (fig.7.16)



(7.16) List of candidates

7.3.3.3 The List Details View (fig.7.17)



(7.17) Description of candidates

7.4 Business use case 3 – Operations Dashboard (Operations Team)

7.4.1 Introduction

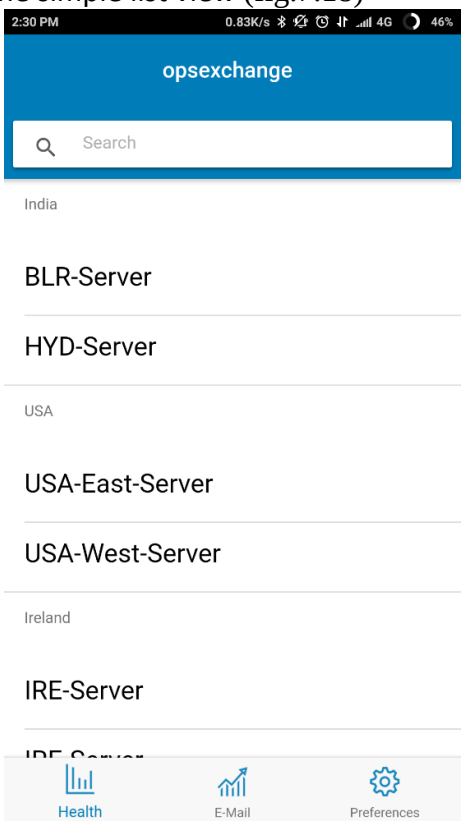
As per our interview with the Operations hiring team, we came up with components such that one of their use cases could be fulfilled. So we made them make an Operations Dashboard using the Self Service Portal.

7.4.2 Features

It has a side menu with server health and email traffic as elements. List items are divided into different sections based on location. For server health it has red and green pie cart. For memory and disk space it has line graph (parameter vs time). For email traffic it has line graph (number of mails vs time).

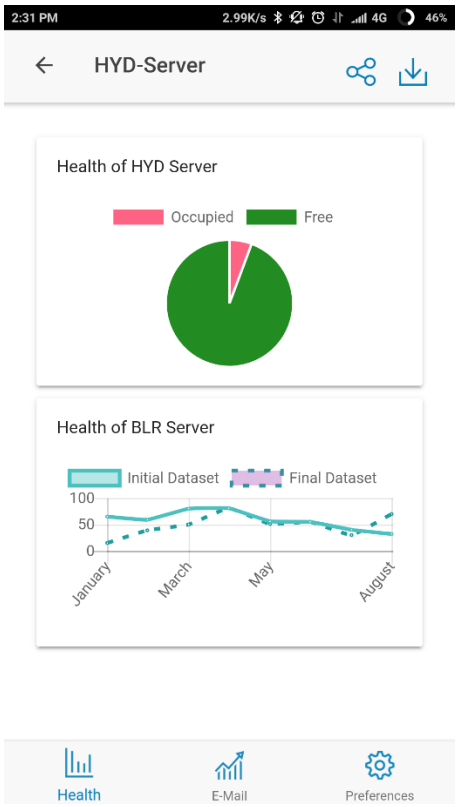
7.4.3 Screenshots

7.4.3.1 The Tabs view and the simple list view (fig.7.18)



(7.18) List of servers for operations app

7.4.3.2 The Description View for the simple list view (fig.7.19)



(7.19) Graphs for constant monitoring of server health

8. Summary

- A secure authentication was implemented for logging into the portal.
- Drag and drop feature was implemented for selecting the components into the mobile screen.
- List of components present in portal were login view, list view with and without icon, chat bot and preferences. Each of the component has properties associated with it.
- List of templates present in portal were blank, side menu, tabs and web apps.
- Auto save feature was implemented.
- APK can be downloaded for android and zipped xcode project files for IOS.
- Helper overlay was also provided for user to ease their work.

9. Conclusion

The Project was successfully completed by end of the internship. The portal was deployed in the production. Three different teams at the company were able to generate mobile apps for their use cases. Both Dell 6 and Dell 4 employees can access the portal for making apps and successfully generate the APK/IPA for their Android and IOS mobile phones.

10. Scope for future work

The Project was successfully completed by end of the internship. Three different teams at the company were able to generate mobile apps for their use cases. Some of the future works for this project can be:

1. Undo Feature

   There can be an undo option in the editor.

2. Online Simulator:

   An online web based simulator for iOS and Android can be made to preview and run the created app instead of building, transferring and running it on an actual phone every time.

3. Advanced Components:

   More advanced drag-and-drop components like maps can also be integrated within the editor.

4. Marketplace:

   Already created apps can be put on a marketplace within the portal, where users can reuse already created app projects and make their apps by modifying them.

5. Postman Integration:

   Postman [47] is a popular tool used to test web services. The portal can have an option to directly import Postman project file, and all the back-end capability can be directly set to the components from the Postman project file.

6. Form component:

   Add a new component to allow users to create forms with different form fields like buttons, text area, check box etc.

7. Messaging Control:

   Add a new component to allow user to communicate with other users using their own backend services.

8. Automated Service Creation:

   Build middleware layers between the current API endpoint and what is expected in the portal by using a query language like GraphQL [66].

9. Map Component:

   Add a map component to allow users to add functionalities like markers, navigation etc.

10. Live Preview:

    When the app is being made the user will be able to live preview the screens of the actual app.

11. Generate Preview for Flow of Screens:

    User will be able to view the flow/navigation of all the screens that is contained in the app through a flow diagram.

Reference

[1] *https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/*

[2] *https://www.sec.gov/Archives/edgar/data/1571996/000157199617000004/delltechnologiesfy1710k.htm*

[3] *http://w3.org/html*

[4] *http://w3.org/css*

[5] *http://developer.mozilla.org/en-US/docs/Web/JavaScript*

[6] *http://php.net*

[7] *http://httpd.apache.org*

[8] *http://www.apachefriends.org*

[9] *http://getbootstrap.com*

[10] *http://fontawesome.com*

[11] *http://github.com/craftpip/jquery-confirm*

[12] *http://jquery.com*

[13] *http://popper.js.org*

[14] *http://vuejs.org*

[15] *http://github.com/rvera/image-picker*

[16] *http://github.com/marvelapp/devices.css/*

[17] *http://github.com/usablica/intro.js*

[18] *http://github.com/prashantchaudhary/ddslick*

[19] *http://jqueryui.com*

[20] *http://holderjs.com*

[21] *http://www.oracle.com/technetwork/java/javase*

[22] *http://eclipse.org*

[23] *http://spring.io/tools/sts/*

[24] *http://maven.apache.org*

[25] *http://github.com/google/gson*

[26] *http://github.com/square/okio*

[27] *http://github.com/square/okhttp*

[28] *http://mariadb.org*

[29] *http://brew.sh*

[30] *http://www.gnu.org/software/sed*

[31] *http://nodejs.org*

[32] *http://ionicframework.com*

[33] *http://cordova.apache.org*

[34] *http://developer.android.com/studio/index.html*

[35] *http://developer.apple.com/xcode*

[36] *https://ionicframework.com/docs/native/in-app-browser/*

[37] *https://ionicframework.com/docs/native/social-sharing/*

[38] *https://www.chartjs.org/*

[39] *https://ionicframework.com/docs/native/email-composer/*

[40] *https://ionicframework.com/docs/native/file/*

[41] *https://ionicframework.com/docs/native/file-transfer/*

[42] *https://ionicframework.com/docs/native/media/*

[43] *https://ionicframework.com/docs/native/streaming-media/*

[44] *http://git-scm.com*

[45] *http://gitlab.com*

[46] *http://www.atlassian.com/software/jira*

[47] *http://www.getpostman.com*

[48] *https://www.adobe.com/in/products/xd.html*

[49] *https://www.kinetise.com*

[50] *https://www.kony.com/products/appplatform*

[51] *http://eachscape.com*

[52] *http://thunkable.com*

[53] *http://www.shoutem.com*

[54] *https://www.appypie.com/no-coding-app-builder*

[55] *https://www.appgyver.com*

[56] *http://www.biznessapps.com*

[57] *https://www.tutorialspoint.com/system_analysis_and_design/system_analysis_and_design_overview.htm*

[58] Rajib Mall, Fundamentals of Software Engineering. Phi Learning Private Limited, India, 2003.

[59] Sutherland, J. The Scrum Papers: Nuts, Bolts, and Origins of an Agile Process, 2011, January.

[60] Takeuchi. H and Nonaka. I, The new product development game, Harvard Business Review, 1986, January.

[61] Sutherland. J and Schwaber. K, The Scrum Guide, 2013, July.

[62] *https://www.cprime.com/2015/02/3-differences-between-scrum-and-kanban-you-need-to-know/*

[63] Beck. K, Extreme Programming Explained: Embrace Change, Addison-Wesley Longman Publishing Co., Inc., Boston, MA,USA, 2000.

[64] Beck, Test Driven Development: By Example, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[65] Williams.L, Kessler.R.R, Cunningham.W and Jef-fries.R, "Strengthening the case for pair programming", IEEE Softw. Vol 17, No. 4, pp. 19-25, July 2000.

[66] *https://graphql.org/*