

CHAPTER 1

INTRODUCTION

1.1 Introduction to Aash Portfolio

Portfolio App is your own personal App, where you can present your creative or commercial work. Your portfolio is literally everything when it comes to how you're perceived as an Engineer — whether you're an artist, hobbyist, or a professional Programmer.

Portfolio app are vital for freelancers working in the digital age. While all freelancers can hand out business cards and certain freelancers, such as photographers, can distribute physical portfolios to potential clients, a portfolio App provides a way for any freelancer in any industry to reach more clients on a global scale. It also provides a way for you to get creative with the information and intricate details you share about yourself and your work.

A portfolio App is an extension of a freelancer's (or company's) résumé. It provides a convenient way for potential clients to view your work while also allowing you to expand on your skills and services. This, however, isn't the ultimate purpose of a portfolio App.

The ultimate purpose of a portfolio App is to provide a way for you to land more clients, whether that means freelance work, more clients for your agency or employment at a company. You should decide what you want to accomplish with your App before adding content to it.

1.2 Organization of the Report:

CHAPTER 2

Chapter 2 talks about the literature survey of the project. It contains all the coding languages involved to run the project. The software used for executing and testing the code Android Studio Is an integrated development environment (IDE) for Google Android OS. An Emulator – a virtual android device that can simulate variety of hardware features. It can then be used for testing purposes. Firebase - is a backend service provided by Google that offers many useful features for mobile and web apps XML- Extensible Markup Language is a text language that can be used to describe the behaviour of programming languages that process them.

CHAPTER 3

Chapter 3 gives the introduction of the project it includes current and proposed system. It tells about the architecture diagram and description of components in the system architecture. It also gives gist about functional, non-functional, hardware and software requirements and also the user interface.

CHAPTER 4

Chapter 4 talks about all the class diagrams, data flow diagrams, state chat diagrams. The algorithm design and pseudo code of important functions. The -screen shots of the different screens and also include brief description about the screen shot.

CHAPTER 5

Chapter 5 talks about software test specification and the test strategies. It also says what you except from testing and where you are performing testing specification of any testing tools (if used) and any constraints required for testing. It includes the test cases pass/fail criteria of each test case.

CONCLUSION AND FUTURE ENHANCEMENT REFERENCES

CHAPTER 2

LITERATURE SURVEY

2.1 Android Studio

Android Studio is an integrated development environment (IDE) for Android Operating System. It is built based on JetBrains' IntelliJ IDEA Community Edition, and it is specifically designed for creating applications on Android devices and to play ground with some Groove and Kotlin. Some of the key features of Android Studio are as follows

1. **Instant Run** – a feature that pushes code and resource changes to the running app. It allows changes to be made to the app without the need to restart the app, or rebuilding the APK, so that the effects can be seen instantly.
2. **An Emulator** – a virtual android device that can simulate variety of hardware features such as GPS location, network latency, motion sensors, and multi-touch input that can be used to run and install the app. It can then be used for testing purposes.
3. **Testing Tools and Frameworks** – extensive testing tools such as, JUnit 4 and functional UI test frameworks are included with Android Studio. Espresso Test Recorder can generate UI test code by recording the developer's interactions with the app on a device or emulator. The tests can be run on a device, an emulator, in Firebase Test Lab, or on a continuous integration environment.

2.2 Kotlin Programming Language

Kotlin is a programming language introduced by JetBrains in 2011, the official designer of the most intelligent Java IDE, named IntelliJ IDEA. This is a strongly statically typed general-purpose programming language that runs on JVM. In 2017, Google announced Kotlin is an official language for android development. Kotlin is an open-source programming language that combines object-oriented programming and functional features into a unique platform. The content is divided into various chapters that contain related topics with simple and useful examples.

2.3 Bootstrap (front-end framework)

Bootstrap, originally named Twitter Blueprint, was developed by Mark Otto and Jacob Thornton at Twitter as a framework to encourage consistency across internal tools. Before Bootstrap, various libraries were used for interface development, which led to inconsistencies and a high maintenance burden. According to Twitter developer Mark Otto:

A super small group of developers and I got together to design and build a new internal tool and saw an opportunity to do something more. Through that process, we saw ourselves build something much more substantial than another internal tool. Months later, we ended up with an early version of Bootstrap as a way to document and share common design patterns and assets within the company.

2.3 XML

XML or Extensible Markup Language is a text language that can be used to describe the behaviour of programming languages that process them. XML was developed XML working group in 1996. According to World Wide Web Consortium there are ten design goals for XML. These design goals are

1. XML shall be straightforwardly usable over the Internet.
2. XML shall support a wide variety of applications.

XML shall be compatible with SGML.

1. It shall be easy to write programs which process XML documents.
2. The number of optional features in XML is to be kept to the absolute minimum, ideally

XML documents should be human-legible and reasonably clear.

1. The XML design should be prepared quickly.
2. The design of XML shall be formal and concise.
3. XML documents shall be easy to create.
4. Terseness in XML markup is of minimal importance.

XML is used when transferring data from the database to the client, and in designing the visual aspect of Android applications. When data is sent from the database, it is sent using XML. This allows the data to be processed by any programming language the same way, since the data is always sent using XML. As mentioned, XML is also used to design the user interface of Android applications. This means that all the visual aspects such as, the layout of the page, the position of all button and text fields, as well as the colour of anything on the page is specified using XML. Since XML is human-legible, it makes the process of designing a page in the app relatively easy and intuitive.

CHAPTER 3

REQUIREMENT'S & SPECIFICATION

3.1 INTRODUCTION

3.1.1 CURRENT SYSTEM

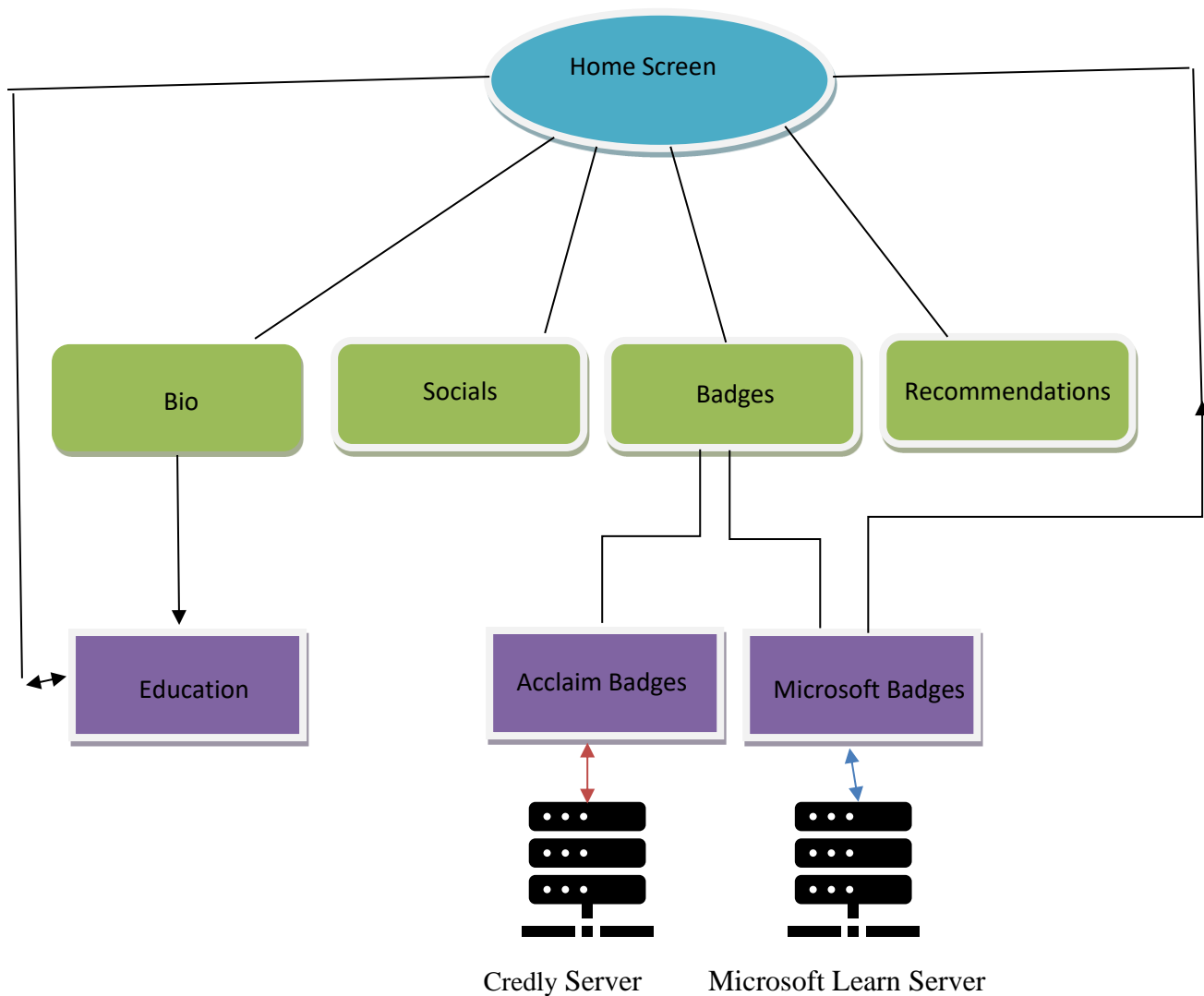
This Project is First of its Kind so there is no Existing System to portrait and Differentiate the Current System

3.1.2 PROPOSED SYSTEM

Aash Portfolio is an App designed Exclusively to show Academic and Career progress of Aashik the concept is first of its Kind There are lots of skilled professionals around. So how can your potential client understand you are a top-notch one? The only way is to leave everything behind for a while and start developing your online presence, because:

- A portfolio App presents your professionalism and dedication to the craft better and more clearly than any CV. You don't just tell; you show actual cases and examples which speak volumes. Moreover, owing to the portfolio App, you can show your mature and sophisticated approach to selling your work along with yourself.
- Your portfolio is the only web-space where everything is up to you. You are able to create a never-seen-before platform with unique content and achieve a recognizable identity the way you imagine it. A private portfolio App lets your inner creator go far beyond any client.
- An online portfolio is certain to make you available and searchable for new clients. It lets your potential employers find you the moment they get round to using a search engine or looking through candidates' applications.
- A portfolio published on the web humanizes you as a master of the craft. People come to you not only for you being an outstanding web designer, marketer or copywriter, but because they like you. You know, it's a good psychological trick. Customers are more likely to choose those, who seem to be more decent, "likable", friendly-looking than others. Whereas hard skills are often considered afterward.

3.2 System Architecture



A system architecture is a representation of a system in which there is a mapping of functionality onto hardware software architecture onto the hardware architecture, and human interaction with these components.

3.3 FUNCTIONAL REQUIREMENTS

Functional requirements are product features or functions that developers must implement to enable users to accomplish their tasks. So, it's important to make them clear both for the development team and the stakeholders. Generally, functional requirements describe system behaviour under specific conditions.

Functional requirements can be classified according to different criteria. For example, we can group them on the basis of the functions a given feature must perform in the end product. Of course, they would differ depending on the product being developed, but for the sake of an example, the types of functional requirements might be

- Authentication
- Authorization levels
- Compliance to laws or regulations
- External interfaces
- Transaction's process

3.4 NON-FUNCTIONAL REQUIREMENTS

3.4.1 USER INTERFACE

User Interface Design for any application should be very simple. We should have only a few clicks or navigation among the features when using the application to avoid hassle.

The user interface (UI) of our tic-tac-toe game, for that we use frames, panels, buttons, labels, etc.

Functions Definitions:

1. **setBackground(new Color(255,255,255))**: This function will set the background color of the UI component.
2. **setLayout(layout)**: This function will set the layout of the frame or panel. Layout can be grid, flow, gridbag, etc
setText("your text"): This function will set the text of the label, button, etc
3. **setVisible(true)**: This function will set the frame/window to be visible to the user.

4. By default, it is false.
- **setBorder(BorderFactory.createLineBorder(Color.decode("#2C6791"))):** This function will set the border around the buttons, frames, panels, etc.
 - **setSize(int width, int height):** This function is used to set the size of frame, panel, etc. It takes two parameters such as width and height. **setIconImage(icon):** This function is used to set the icon of the frame/window.
 - **add(obj):** This function is used to add the component object in frame or panel.

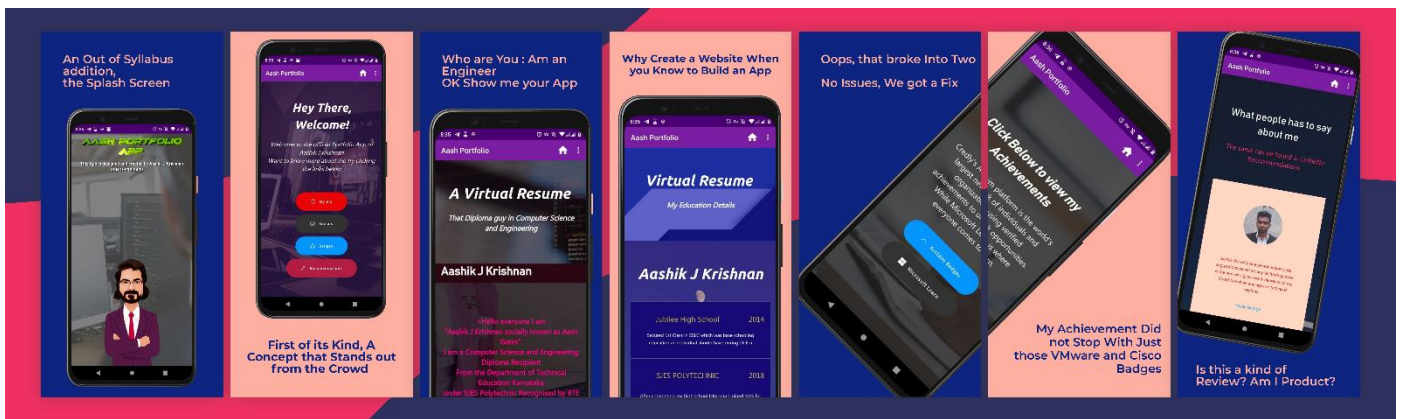







Figure 3.4.1 User interface

Figure 3.4.1 User interface diagram

Android provides an XML vocabulary for View Group and View classes, so most of your UI is defined in XML files. However, rather than teach you to write XML, this lesson shows you how to create a layout using Android Studio's Layout Editor. The Layout Editor writes the XML for you as you drag and drop views to build your layout.

Layout Editor

1. To get started, set up your workspace as follows:
2. In the Project window, open **app > res > layout > activity_main.xml**.
3. To make room for the Layout Editor, hide the **Project** window. To do so, select **View > Tool Windows > Project**, or just click **Project** on the left side of the Android Studio screen.
4. If your editor shows the XML source, click the **Design** tab at the top right of the window.
5. Click  (Select Design Surface) and select **Blueprint**.
6. Click  (View Options) in the Layout Editor toolbar and make sure that **Show All Constraints** is checked.

7. Make sure Autoconnect is off. A tooltip in the toolbar displays  (Enable Autoconnection to Parent) when Autoconnect is off.
8. Click  (Default Margins) in the toolbar and select **16**. If needed, you can adjust the margins for each view later.
9. Click  (Device for Preview) in the toolbar and select **5.5, 1440 × 2560, 560 dpi (Pixel XL)**.
10. Your Layout Editor now looks as shown in figure.

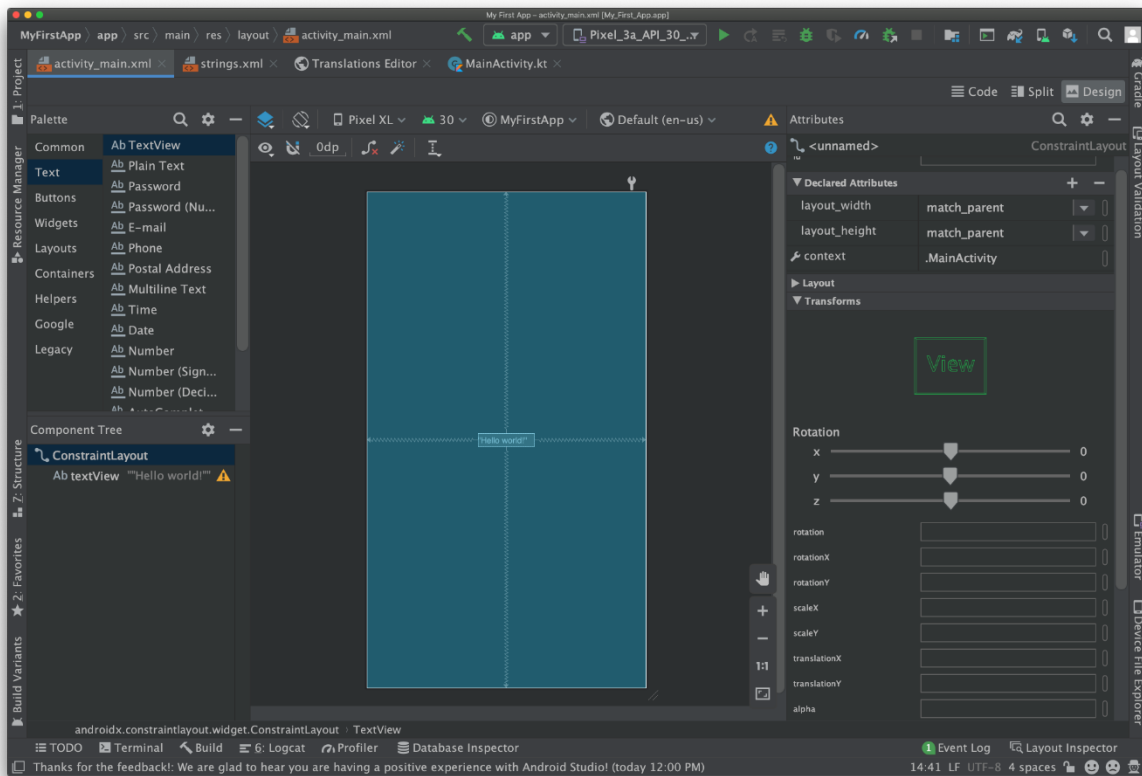


Figure 3.4.1 XML diagram

3.4.2 Software Requirements

Requirements for Running Android Studio with AVD (Android Virtual Device) Manager

Operating System	Windows /Mac OS
Bridge	Android Platform Tools
Compiler 1	RedHat OpenJDK jdk-16.0.1.9
Compiler 2	Kotlin
Language	Kotlin, Java, XML, Bootstrap

Requirements for Running the Android App

Operating System	AOSP Based Android 6.0 and above
Security Patch	Latest Security Patch released on August 5 2021 (Tested for Combability with Path released during the Demo of this Project

3.4.3 Hardware Requirements

Hardware Requirements to Run Android Studio Along with AVD Manager

Processor	AMD FX/Ryzen/Ryzen Pro/Threadripper/EPYC Series Processors Intel Pentium 6 th Gen and Above/iCore/Core/ /Xeon Processors 4 th Gen and above/ Intel evo
Storage	M.2 NVMe /SATA SSD, Hard Disk with an RPM of 7500 or above with at least 250GB Space on OS Directory for Smoother Operation
RAM	8 GB and Above with 2133MHz Plus (3000MHz Preferred)

Hardware Requirement to Run the Android App

Processor	Any Processor that Supports ARMv8-A architecture
RAM	2 GB and Above

3.5 SYSTEM MODEL

3.5.1 USE CASE DIAGRAM

Use case diagrams show business use cases, actors, and the relationships between them. The relationships between actors and business use cases state that an actor can use a certain functionality of the business system. You will not find any information.

In the Unified Modelling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. An effective use case diagram can help your team discuss and represent:

- Scenarios in which your system or application interacts with people, organizations, or external systems
- Goals that your system or application helps those entities (known as actors) achieve

CHAPTER 4

SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE AND DECOMPOSITION

A system architecture is the conceptual model that defines the structure, behaviour, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviours of the system. A system architecture can consist of system components and the sub-systems developed, that will work together to implement the overall system. There have been efforts to formalize languages to describe system architecture, collectively these are called architecture description languages.

4.2 DATA MODELS

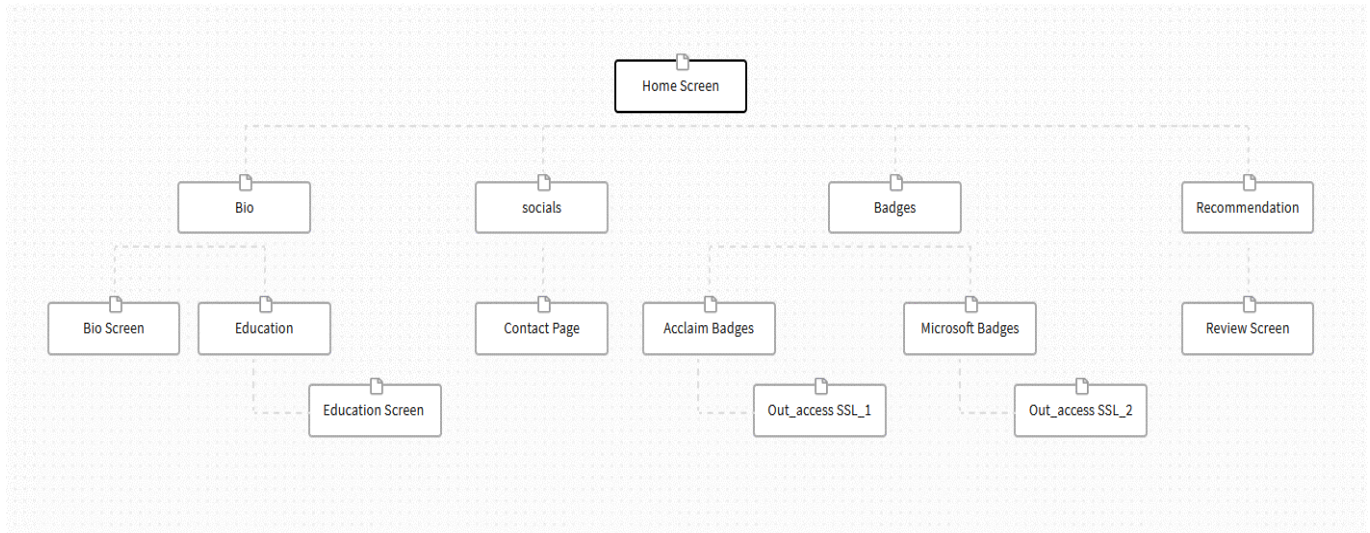
4.2.1 CLASS DIAGRAM

Class diagrams have three main constituents:

- **Class Name** - this is a mandatory section as it contains the name of the class
- **Class Attributes** - this is where all the attributes of a class can be found
- **Class Operations** - includes a list of operations that describe how a class interacts with data

4.2.2 DATA FLOW DIAGRAM

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled. They can be used to analyse an existing system or model a new one. That's why DFDs remain so popular after all these years. While they work well for data flow software and systems, they are less applicable nowadays to visualizing interactive, real-time or database-oriented software or systems



4.2.2 DATA FLOW DIAGRAM

4.2.3 STATE CHART DIAGRAM

State chart diagram is one of the five UML diagrams used to model the dynamic nature of a system. They define different states of an object during its lifetime and these states are changed by events. State chart diagrams are useful to model the reactive systems. Reactive systems can be defined as a system that responds to external or internal event. State chart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. Following are the main purposes of using State chart diagrams –

- To model the dynamic aspect of a system.
- To model the life time of a reactive system.
- To describe different states of an object during its life time.
- Define a state machine to model the states of an object.

4.3 ALGORITHM DESIGN

XML Code:

```
<?xml version="1.0" encoding="utf-8"
standalone="no"?><manifest
xmlns:android="http://schemas.android.com/apk/res/android"
android:compileSdkVersion="23"
android:compileSdkVersionCodename="6.0-2438415"
package="com.estilo.android "
platformBuildVersionCode="23"
platformBuildVersionName="6.0-2438415">

<uses-permission
android:name="android.permission.ACCESS_NETWORK_S
TATE"/>

<uses-permission
android:name="android.permission.VIBRATE"/>

<uses-permission
android:name="android.permission.INTERNET"/>

<application android:allowBackup="true"
android:fullBackupContent="true"
android:icon="@drawable/ic_launcher"
android:label="@string/app_name"
android:theme="@style/MaterialAppTheme"
android:usesCleartextTraffic="@bool/usesCleartextTraffic">

<activity
android:configChanges="keyboardHidden|orientation|screen
Size" android:hardwareAccelerated="true"
android:label="@string/app_name"
android:name="com.goyal.website2apk.MainActivity"
android:screenOrientation="portrait"
android:windowSoftInputMode="adjustResize">
```

```
<intent-filter>

<action android:name="android.intent.action.VIEW"/>

<action android:name="android.intent.action.MAIN"/>

<category
android:name="android.intent.category.LAUNCHER"/>

</intent-filter>

<intent-filter>

<action android:name="android.intent.action.VIEW"/>

<category
android:name="android.intent.category.DEFAULT"/>

<category
android:name="android.intent.category.BROWSABLE"/>

<data android:host="null" android:pathPrefix="/"
android:scheme="https"/>

</intent-filter>

</activity>

<meta-data android:name="android.support.VERSION"
android:value="26.1.0"/>

<meta-data
android:name="android.arch.lifecycle.VERSION"
android:value="27.0.0-SNAPSHOT"/>

</application>

</manifest>
```


Bootstrap Code

```
<!DOCTYPE html>

<html >

<head>

  <meta charset="UTF-8">

  <meta http-equiv="X-UA-Compatible" content="IE=edge">

  <meta name="generator" content="Mobirise v4.12.4,
mobirise.com">

  <meta name="twitter:card" content="summary_large_image"/>

  <meta name="twitter:image:src" content="assets/images/index-
meta.jpg">

  <meta property="og:image" content="assets/images/index-
meta.jpg">

  <meta name="twitter:title" content="Aash Gates">

  <meta name="viewport" content="width=device-width, initial-
scale=1, minimum-scale=1">

  <link rel="shortcut icon" href="assets/images/phoxo1-1-1.webp"
type="image/x-icon">

  <meta name="description" content="Welcome to the Official
Website of that CSE Diploma Boy">


<title>Aash Gates</title>

<link rel="stylesheet" href="assets/web/assets/mobirise-icons-
bold/mobirise-icons-bold.css">

<link rel="stylesheet" href="assets/web/assets/mobirise-
icons/mobirise-icons.css">

<link rel="stylesheet"
href="assets/bootstrap/css/bootstrap.min.css">

<link rel="stylesheet" href="assets/bootstrap/css/bootstrap-
grid.min.css">
```

```
<link rel="stylesheet" href="assets/bootstrap/css/bootstrap-
reboot.min.css">

<link rel="stylesheet" href="assets/tether/tether.min.css">

<link rel="stylesheet" href="assets/web/assets/gdpr-plugin/gdpr-
styles.css">

<link rel="stylesheet" href="assets/animatecss/animate.min.css">

<link rel="stylesheet" href="assets/theme/css/style.css">

<link href="assets/fonts/style.css" rel="stylesheet">

<link rel="preload" as="style" href="assets/mobirise/css/mbr-
additional.css"><link rel="stylesheet"
href="assets/mobirise/css/mbr-additional.css" type="text/css">

</head>

<body>

<section class="cid-rW9hQf4kpQ mbr-fullscreen mbr-parallax-
background" id="header2-c">

<div class="mbr-overlay" style="opacity: 0.7; background-color:
rgb(31, 5, 63);"></div>

<div class="container align-center">

<div class="row justify-content-md-center">

<div class="mbr-white col-md-10">

<h1 class="mbr-section-title mbr-bold pb-3 mbr-fonts-
style display-1">Hey There, Welcome! </h1>

<p class="mbr-text pb-3 mbr-fonts-style display-
5"><em>Welcome to the official Portfolio App of Aashik J
Krishnan<br>Want to know more about me try clicking the links
below<br></em><br>

</p>

<div class="mbr-section-btn"><a class="btn btn-md btn-
secondary display-4" href="Bio.html"><span class="mbri-user mbr-
iconfont mbr-iconfont-btn" style="font-size: 22px;"></span>My
Bio</a>

<a class="btn btn-md btn-black display-4"
href="Socials.html"><span class="mbri-globe mbr-iconfont mbr-
```

```
iconfont-btn" style="font-size: 22px;"></span>Socials</a> <a  
class="btn btn-md btn-primary display-4"  
href="Badges.html"><span class="mbrib-star mbr-iconfont mbr-  
iconfont-btn" style="font-size: 22px;"></span>Badges</a> <a  
class="btn btn-md btn-success display-4"  
href="RecRev.html"><span class="mbri-magic-stick mbr-iconfont  
mbr-iconfont-btn" style="font-size:  
22px;"></span>Recomendations</a></div>
```

```
</div>
```

```
</div>
```

```
</div>
```

```
</section>
```

```
<section class="engine"><a href="https://mobirise.info/o">free  
portfolio web templates</a></section><script  
src="assets/web/assets/jquery/jquery.min.js"></script>
```

```
<script src="assets/popper/popper.min.js"></script>
```

```
<script src="assets/bootstrap/js/bootstrap.min.js"></script>
```

```
<script src="assets/tether/tether.min.js"></script>
```

```
<script src="assets/parallax/jarallax.min.js"></script>
```

```
<script  
src="assets/viewportchecker/jquery.viewportchecker.js"></script>
```

```
<script src="assets/smoothscroll/smooth-scroll.js"></script>
```

```
<script src="assets/theme/js/script.js"></script>
```

```
<div id="scrollToTop" class="scrollToTop mbr-arrow-up"><a  
style="text-align: center;"><i class="mbr-arrow-up-icon mbr-arrow-  
up-icon-cm cm-icon cm-icon-smallarrow-up"></i></a></div>
```

```
<input name="animation" type="hidden">
```

```
</body>
```

```
</html>
```

4.4 USER INTERFACE DIAGRAM

4.4.1 Splash Screen:

A splash screen is mostly the first screen of the app when it is opened. It is a constant screen which appears for a specific amount of time, generally shows for the first time when the app is launched. The Splash screen is used to display some basic introductory information such as the company logo, content, etc. just before the app loads completely.

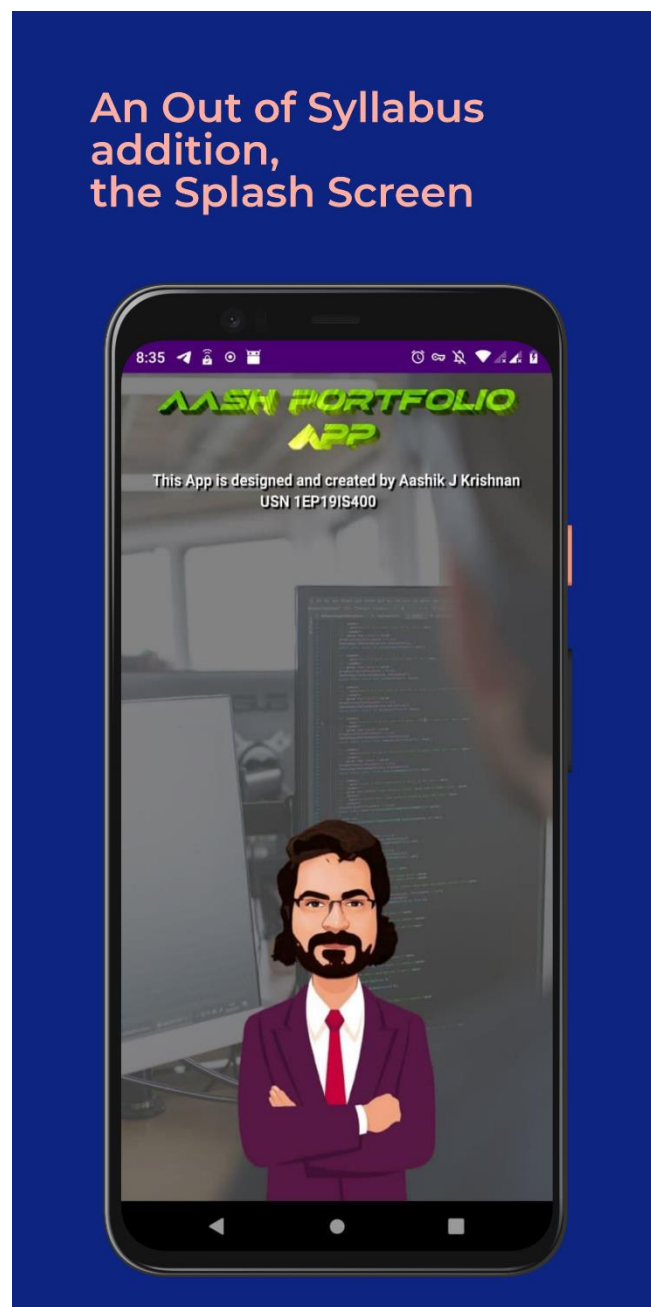


Figure 4.4.1 Splash Screen

4.4.2 HOME Screen:

The application opens up with this page when the android App is run.

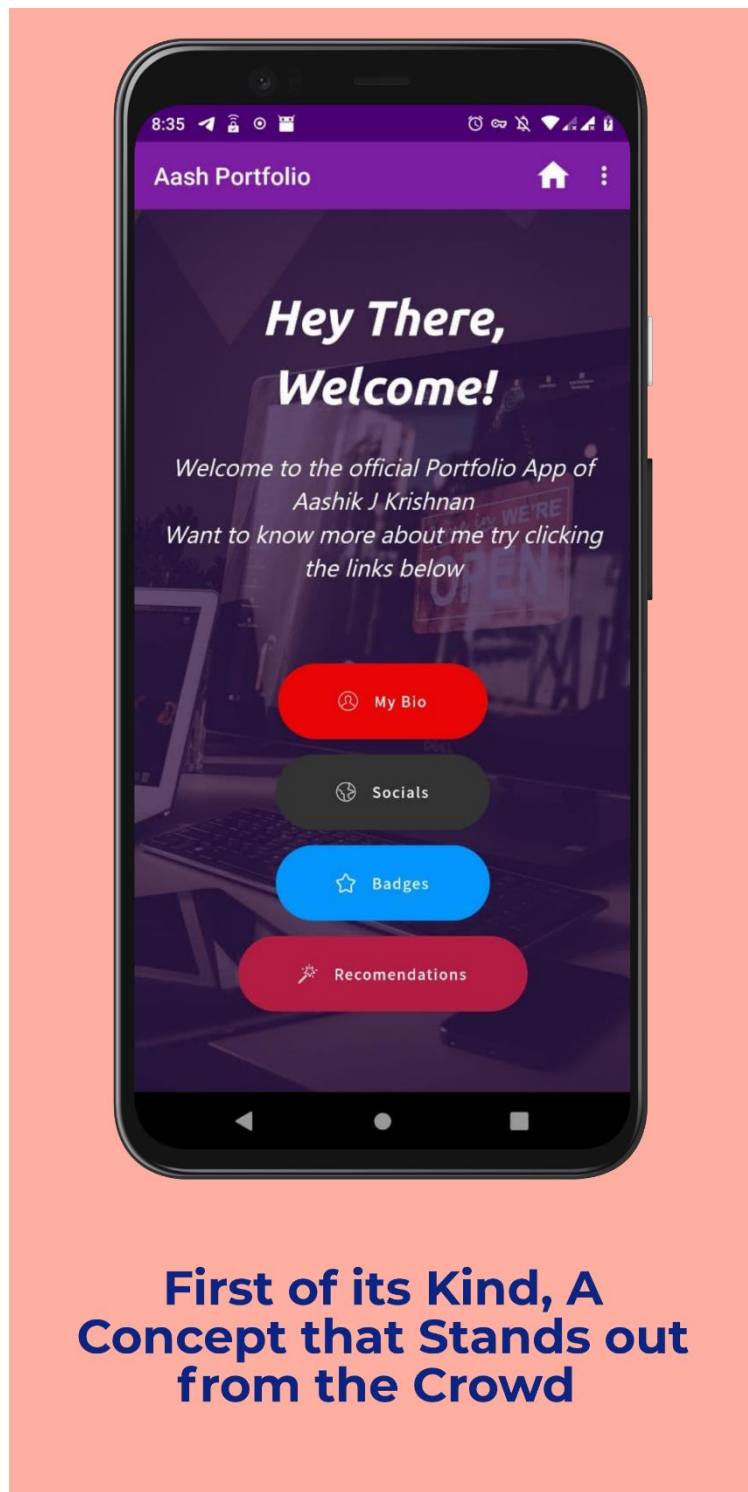


Figure 4.4.2 Home Screen

4.4.3 Mock Up

A mock-up suggests what the final design will look like, and is usually shared with clients and stakeholders between a wireframe and prototype.

Wireframes are built and shared to communicate the structure and functional requirements of the design. Mock-ups are essentially wireframes with an added surface layer that communicates the visual design (colours, images, typography).

Unlike a prototype, a mock-up is static and doesn't include any interactions.

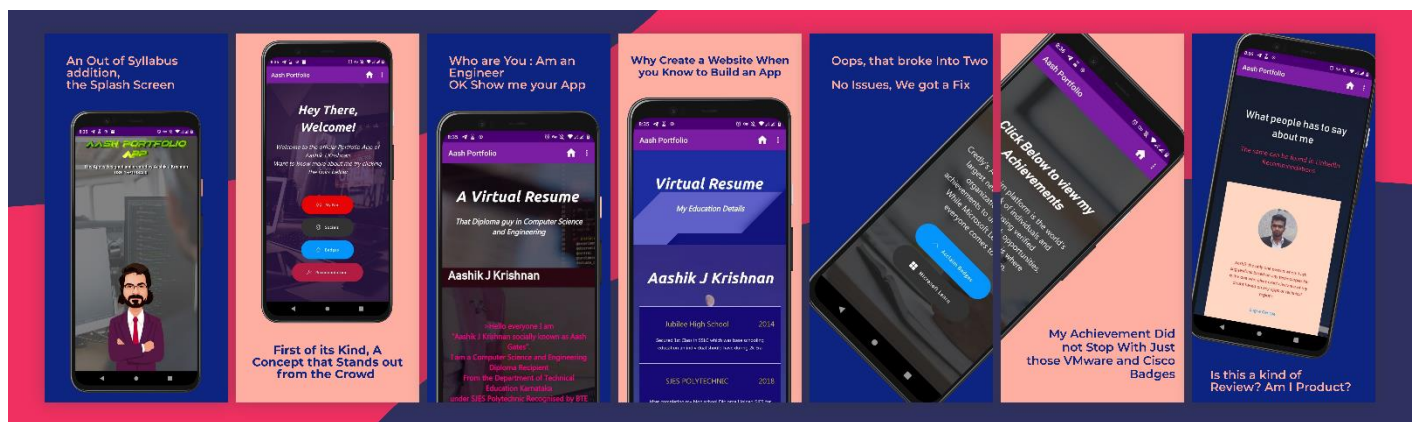


Figure 4.2.3 Mock-Up

CHAPTER 5

SOFTWARE TEST SPECIFICATION

Testing is a process, which reveals errors in the program. Once the source code has been generated, the software must be tested to uncover as many errors as possible before delivery to the customer. Software testing can be stated as the process of validating and verifying that a computer program/application/product:

- meets the requirements that guided its design and development,
- works as expected,
- Can be implemented with the same characteristics,
- Satisfies the needs of stakeholders.

5.2 TESTING STATERGIES

Is the process used to assess the quality of computer software? Software testing is an empirical technical investigation conducted to provide stakeholders with information about the quality of the product or service under test, with respect to the context in which it is intended to operate. This includes, but is not limited to, the process of executing a program or application with the intent of finding software bugs. Quality is not an absolute; it is value to some person. With that in mind, testing can never completely establish the correctness of arbitrary computer software; testing furnishes a criticism or comparison that compares the state and behaviour of the product against a specification. An important point is that software testing should be distinguished from the separate discipline of Software Quality Assurance (S.Q.A.), which encompasses all business process areas, not just testing. Over its existence, computer software has continued to grow in complexity and size. Every software product has a target audience. For example, a video game software has its audience completely different from banking software. Therefore, when an organization develops or otherwise invests in a software product, it presumably must assess whether the software product will be acceptable to its end users, its target audience,

its purchasers, and other stakeholders. Software testing is the process of attempting to make this assessment Software testing methods are traditionally divided into black box testing and white box testing. These two approaches are used to describe the point of view that a test engineer takes when designing test cases.

5.2.1 UNIT TESTING

However, is when the tester has access to the internal data structures, code, and algorithms. White box testing methods include creating tests to satisfy some code coverage criteria. For example, the test designer can create tests to cause all statements in the program to be executed at least once. Other examples of white box testing are mutation testing and fault injection methods. White box testing includes all static testing. White box testing methods can also be used to evaluate the completeness of a test suite that was created with black box testing methods. This allows the software team to examine parts of a system that are rarely tested and ensures that the most important function points have been tested. Two common forms of code coverage are function coverage, which reports on functions executed and statement coverage, which reports on the number of lines executed to complete the test. They both return a coverage metric, measured as a percentage. Unit testing tests the minimal software component, or module. Each unit (basic component) of the software is tested to verify that the detailed design for the unit has been correctly implemented. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors.

5.2.2 INTEGRATION TESTING

Integration testing exposes defects in the interfaces and interaction between integrated components (modules). Progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a system. System integration testing verifies that a system is integrated to any external or third-party systems defined in the system requirements. Before shipping the final version of software, alpha and beta testing are often done additionally.

CONCLUSION

A portfolio App is an efficient and simple tool to collect all the best works you made as a professional and put it in one place to show everyone. Creating your own site may become like building your own workshop as an artisan.

It will be your own, personal place, where you can hang your trophies and show everyone what are you capable of. Start building your best online portfolio website today – it's free! – make a significant contribution to your professional growth and career development!

Express yourself, keep up with the changes, and don't stick to any rules – just follow your taste.

Take your portfolio as an opportunity to surprise and impress your potential clients. A portfolio App lets you show off all of your work in one place. This not only looks good to a client, but it can also help inspire you on your next projects.

Tell others about your skills and personality. Express your uniqueness through case studies. Be searchable and keep the pace with your competitors. There are no rules and everything goes as long as you are able to make the visitor smile and remember you.

REFERENCE

Knowledge Grab, Text Books Used

- Android Programming: The Big Nerd Ranch Guide
- Headfirst Android Development
- Java Programming for Android Developers for Dummies
- Android Application Development All-in-one for Dummies
- GUI Design for Android Apps
- GUI Design for Android Apps
- Efron, B. & Tibshirani, R. J. (1993). *An introduction to the bootstrap*
- Hands-On Design Patterns with Kotlin
- Kotlin Programming: The Big Nerd Ranch Guide
- Davison, A. C. & Hinkley, D. V. (2009). *Bootstrap methods and their application*. New York, NY: Cambridge University Press.

Knowledge Base

- <https://kinsta.com/>
- <https://developer.android.com/>
- <https://weblium.com/>

Knowledge Grab, Online learning

1. Coursera
2. Udemy
3. IBM Digital Nation
4. IBM Skills Build
5. IBM Open P-Tech