Software Requirements Specification

APPRISE

To:

Ms. Priyanka
(Mentor and Assistant professor)

From:
Kapil bakshi
Kunal Kumar Gupta,
Manish Kumar,
Pankaj Kumar

Table of Contents

1. Introduction

This section gives a scope description and overview of everything included in this SRS document. Also, the purpose for this document is described and a list of abbreviations and definitions is provided.

1.1 Purpose

The purpose of this document is to give a detailed description of the requirements for the “Apprise software. It will illustrate the purpose and complete declaration for the development of system. It will also explain system constraints, interface and interactions with other external applications.

1.2 Scope

Apptimize allows app developers to deploy multiple versions of a mobile app while maintaining a single app listing in the App Store. The developer can programmatically control which version of the app each user sees and enable new features or user flows.

1.4 References

https://en.wikipedia.org/wiki/apptimize

http://apptimize.com/docs/examples/visual-apptimizer-test/android-visual/

1.5 Overview

The remainder of this document includes three chapters and appendixes. The second one provides an overview of the system functionality and system interaction with other systems. This chapter also introduces different types of stakeholders and their interaction with the system. Further, the chapter also mentions the system constraints and assumptions about the product.

The third chapter provides the requirements specification in detailed terms and a description of the different system interfaces. Different specification techniques are used in order to specify the requirements more precisely for different audiences.

The fourth chapter deals with the prioritization of the requirements. It includes a motivation for the chosen prioritization methods and discusses why other alternatives were not chosen.

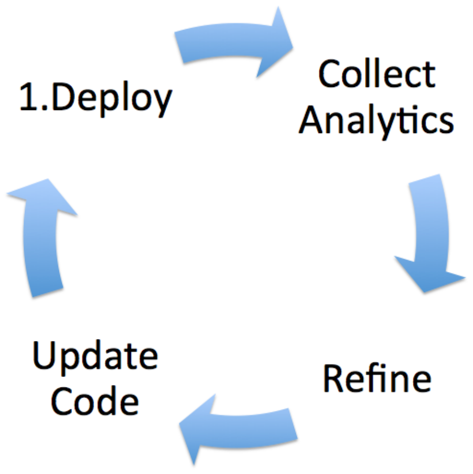
The Appendixes in the end of the document include the all results of the requirement prioritization and a release plan based on them.

2. Overall description

This section will give an overview of the whole system. The system will be explained in its context to show how the system interacts with other systems and introduce the basic functionality of it. It will also describe what type of stakeholders that will use the system and what functionality is available for each type. At last, the constraints and assumptions for the system will be presented.

2.1 Product perspective

The traditional way to A/B test looks something like this:



You must deploy your app with version A, wait for it to roll out to a significant number of users, gather analytics, redeploy with version B, wait for it to roll out, gather analytics, then you can finally make your decision on whether version A or B yields the best results. This method takes lots of time on both the development side and simply waiting for the app store and your users to update.

With Apprise A/B testing, we can deploy both versions simultaneously, gather analytics and propagate the optimized version to all of your users without ever redeploying. This saves lots of time in the development cycle and gives you more control over what your users are seeing!

2.2 Product functions

A/B testing compares two versions of a webpage against each other to determine which one performs better. By creating an A and B version of your page you can validate new design changes, test hypotheses, and improve your website's conversion rate.
A/B tests are statistical experiments that help you decide whether a change is actually making a significant impact on your product.

2.3 User characteristics

* Customers interacting with site.
* Implementers coding A/B test.
* Somebody interpreting results.

2.4 Assumptions and dependencies

One assumption about the product is that it will always be used on mobile phones that have enough performance. If the phone does not have enough hardware resources available for the application, for example the users might have allocated them with other applications, there may be scenarios where the application does not work as intended or even at all.

.

2.5 Apportioning of requirements

In the case that the project is delayed, there are some requirements that could be transferred to the next version of the application. Those requirements are to be developed in the third release.

3. Specific requirements

This section contains all of the functional and quality requirements of the system. It gives a detailed description of the system and all its features.

3.1 External interface Requirements

This section provides a detailed description of all inputs into and outputs from the system. It also gives a description of the hardware, software and communication interfaces and provides basic prototypes of the user interface.

3.1.1 User interfaces

A first-time user of the mobile application should see the log-in page when he/she opens the application, see Figure 2. If the user has not registered, he/she should be able to do that on the log-in page.

If the user is not a first-time user, he/she should be able to see the search page directly when the application is opened, see Figure 3. Here the user chooses the type of search he/she wants to conduct.

Every user should have a profile page where they can edit their e-mail address, phone number and password. Also, the user can set the mobile application to his/her preferred language.

The hardware connection to the database server is managed by the underlying operating system on the mobile phone and the web server.

3.1.2 Communications interfaces

The communication between the different parts of the system is important since they depend on each other. However, in what way the communication is achieved is not important for the system and is therefore handled by the underlying operating systems for both the mobile application and the web portal.

3.2 Functional Requirements

A/B testing allows us to show visitors two versions of the same page and let them determine the winner. Constantly testing and optimizing your page can increase revenue, donations, leads, registrations, downloads, and user-generated content, while providing teams with valuable insight about their visitors.

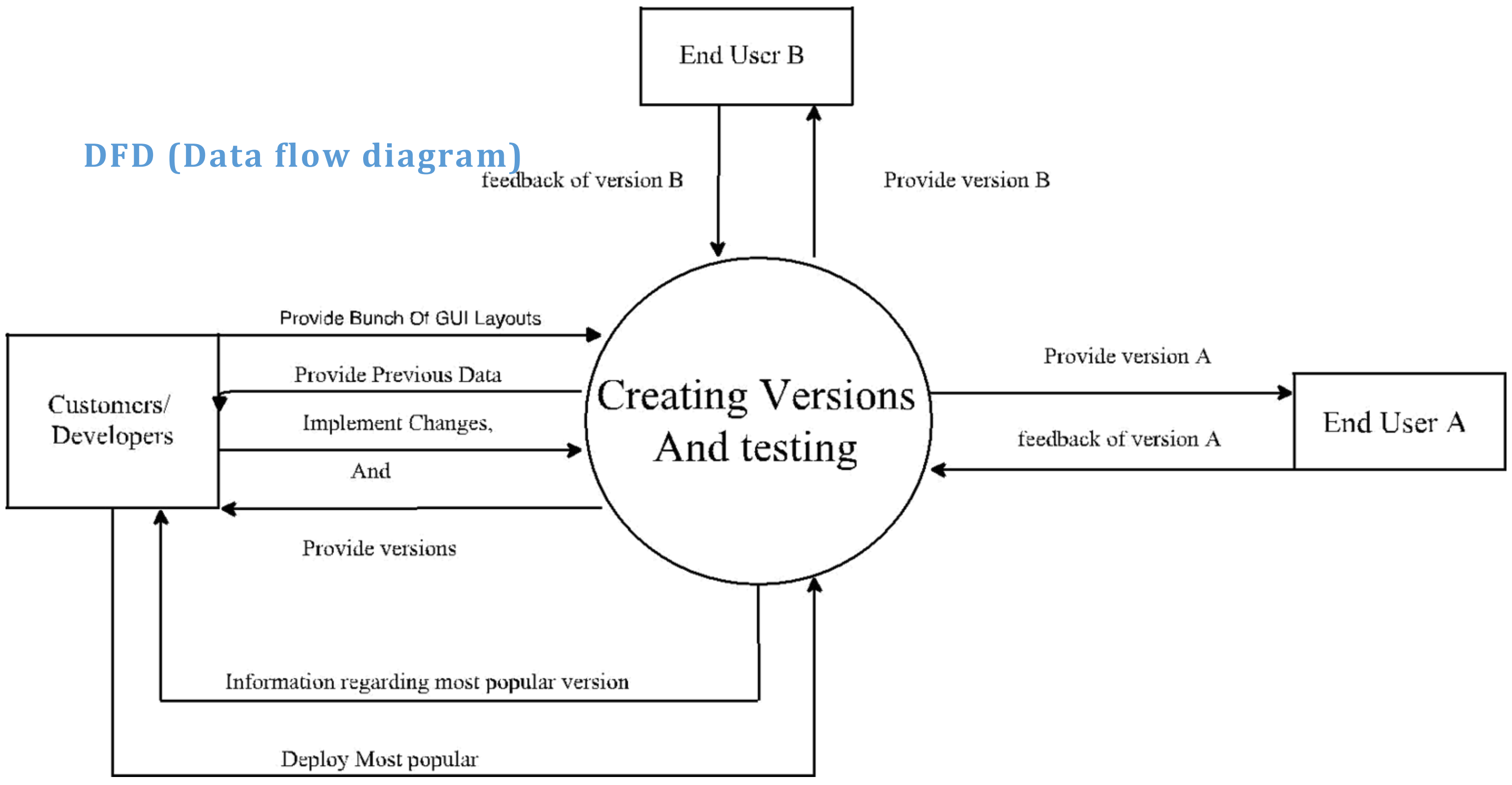
It lets you make live changes without App Store updates. This is a big deal for mobile because many platforms don't allow for fast iterations. You can even make visual changes, iterate with more advanced features, and work with cross-functional teams with all the tools everyone on the team needs to drive improvements.

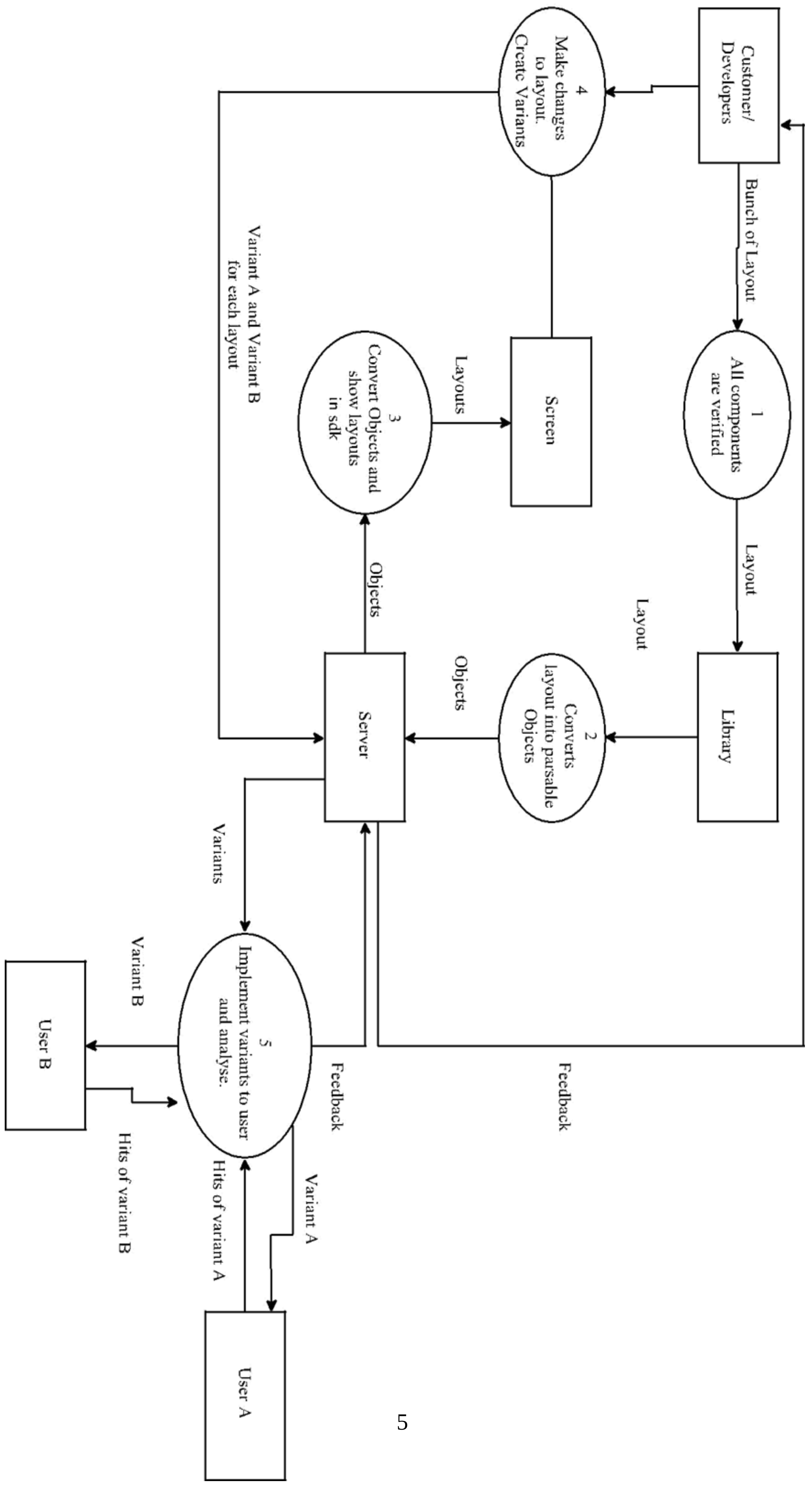
3.2 Non-functional Requirements

1.) Developers can increase conversion rate of their app.

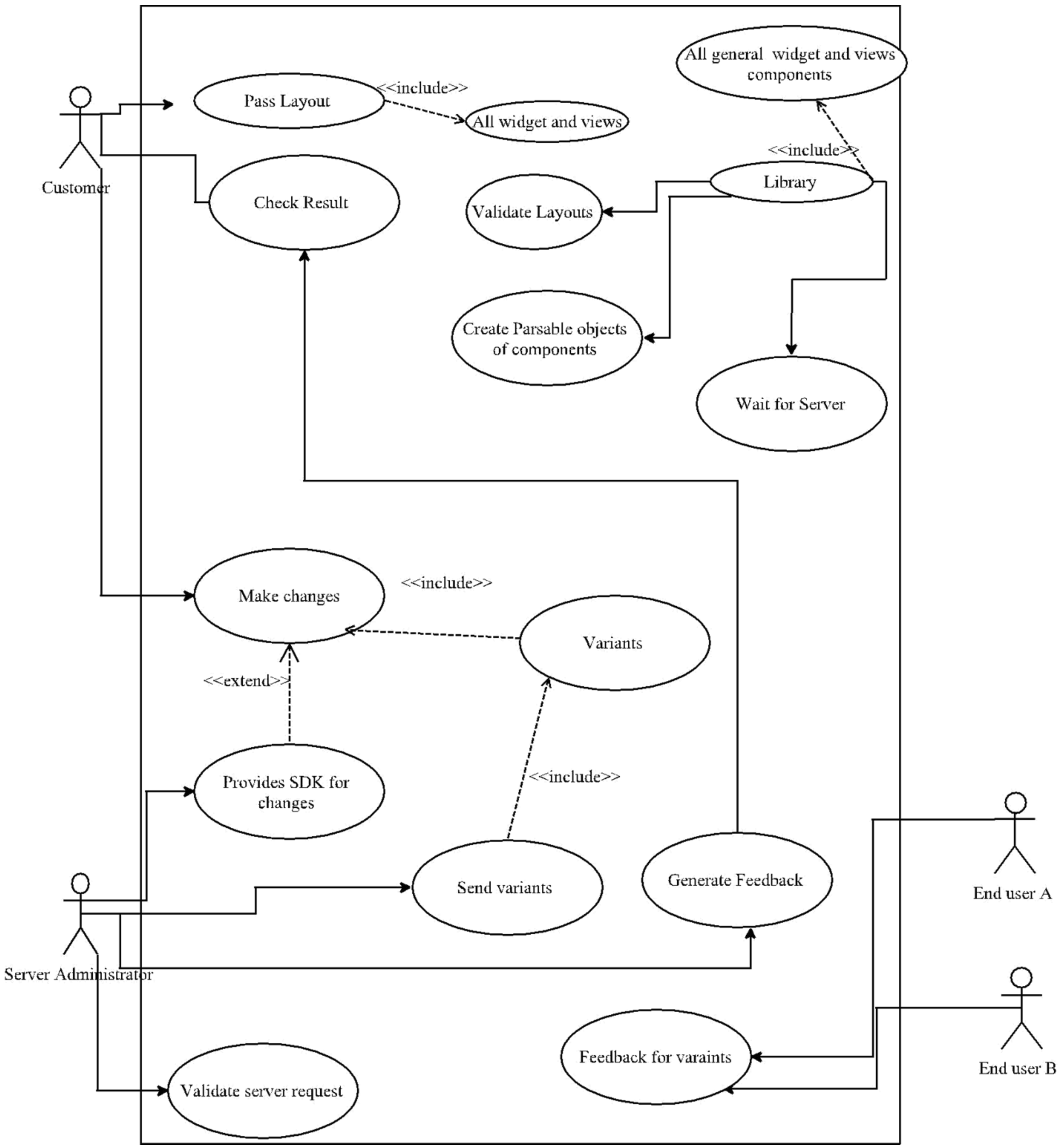
1.) Increase in user engagement and retention.

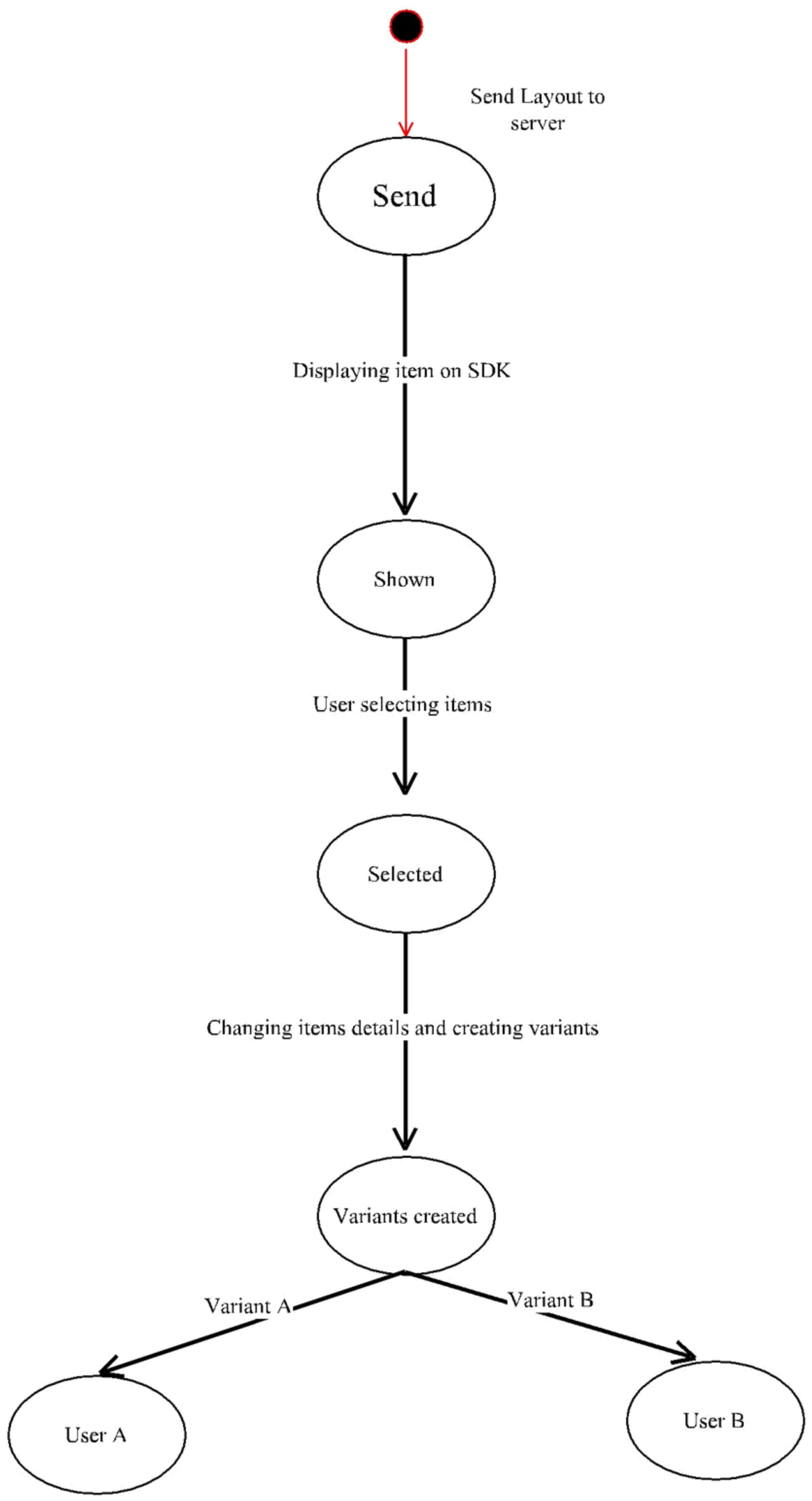
DFD (Data flow diagram)

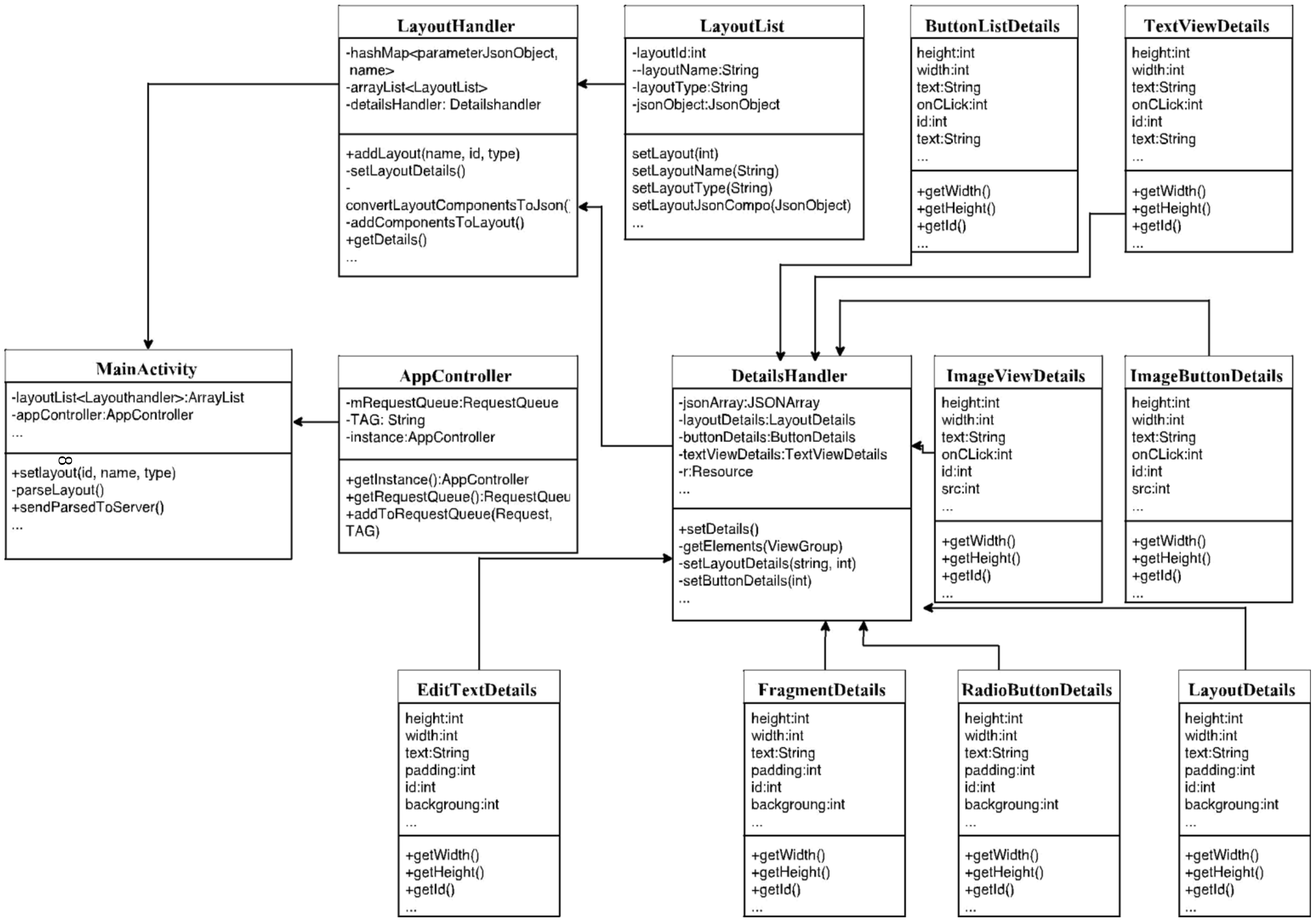




Customer/ Developers
Bunch of Layout
1 All components are verified
Layout
Library
Layout
2 Converts layout into parsable Objects
Objects
Server
4 Make changes to layout. Create Variants
Screen
Layouts
3 Convert Objects and show layouts in sdk
Objects
Variant A and Variant B for each layout
Variants
5 Implement variants to user and analyse.
Feedback
Feedback
Variant B
User B
Hits of variant B
Hits of variant A
Variant A
User A


Customer
Pass Layout
<<include>>
All widget and views
All general widget and views components
<<include>>
Library
Check Result
Validate Layouts
Create Parsable objects of components
Wait for Server
Make changes
<<include>>
Variants
<<extend>>
Provides SDK for changes
<<include>>
Send variants
Generate Feedback
End user A
Server Administrator
End user B
Feedback for varaints
Validate server request


Send Layout to server
Send
Displaying item on SDK
Shown
User selecting items
Selected
Changing items details and creating variants
Variants created
Variant A
Variant B
User A
User B


LayoutHandler
-hashMap<parameterJsonObject, name>
-arrayList<LayoutList>
-detailsHandler: Detailshandler
+addLayout(name, id, type)
-setLayoutDetails()
-convertLayoutComponentsToJson(
-addComponentsToLayout()
+getDetails()
...
LayoutList
-layoutId:int
--layoutName:String
-layoutType:String
-jsonObject:JsonObject
setLayout(int)
setLayoutName(String)
setLayoutType(String)
setLayoutJsonCompo(JsonObject)
...
ButtonListDetails
height:int
width:int
text:String
onCLick:int
id:int
text:String
...
+getWidth()
+getHeight()
+getId()
...
TextViewDetails
height:int
width:int
text:String
onCLick:int
id:int
text:String
...
+getWidth()
+getHeight()
+getId()
...
MainActivity
-layoutList<Layouthandler>:ArrayList
-appController:AppController
...
+setlayout(id, name, type)
-parseLayout()
+sendParsedToServer()
...
AppController
-mRequestQueue:RequestQueue
-TAG: String
-instance:AppController
+getInstance():AppController
+getRequestQueue():RequestQueu
+addToRequestQueue(Request, TAG)
DetailsHandler
-jsonArray:JSONArray
-layoutDetails:LayoutDetails
-buttonDetails:ButtonDetails
-textViewDetails:TextViewDetails
-r:Resource
...
+setDetails()
-getElements(ViewGroup)
-setLayoutDetails(string, int)
-setButtonDetails(int)
...
ImageViewDetails
height:int
width:int
text:String
onCLick:int
id:int
src:int
...
+getWidth()
+getHeight()
+getId()
...
ImageButtonDetails
height:int
width:int
text:String
onCLick:int
id:int
src:int
...
+getWidth()
+getHeight()
+getId()
...
EditTextDetails
height:int
width:int
text:String
padding:int
id:int
backgroung:int
...
+getWidth()
+getHeight()
+getId()
...
FragmentDetails
height:int
width:int
text:String
padding:int
id:int
backgroung:int
...
+getWidth()
+getHeight()
+getId()
...
RadioButtonDetails
height:int
width:int
text:String
padding:int
id:int
backgroung:int
...
+getWidth()
+getHeight()
+getId()
...
LayoutDetails
height:int
width:int
text:String
padding:int
id:int
backgroung:int
...
+getWidth()
+getHeight()
+getId()
...