Agile Reports for Techshop JU

Abdul Mukit (2592)

February 6, 2022

Contents

List of Figures

List of Tables

Listings

Chapter 1

Contribution Report

1.1 Project Summary

Techshop JU is an online e-commerce application inspired from TechshopBD. Through this application people can buy various electronic component for their own project development. This will be a reliable place where students from CSE, EEE major can get their necessary equipment for electronic lab. This application serve two basic functionality.

- User can buy component from the shop.
- User can also order their designed printed circuit board for their project.

User have to follow these process to use this application:

- User have to create an account to access this application.
- Once account is created, user can login next time.
- After authentication, user will send to product gallery where they canview and choice products.
- By clicking one the products, user shift to product detail page.
- To purchase product, user need to press 'add to cart' button. These product is then stored in cart.
- From cart user will move on order page to confirm purchase. Then provide user details.
- User can pick cash on delivery or BKash for payment.
- User can also order custom designed PCB. For doing this user need to design Schematic and PCB layout using any pcb design tool. Then generate either pdf format or pcb gerber file.

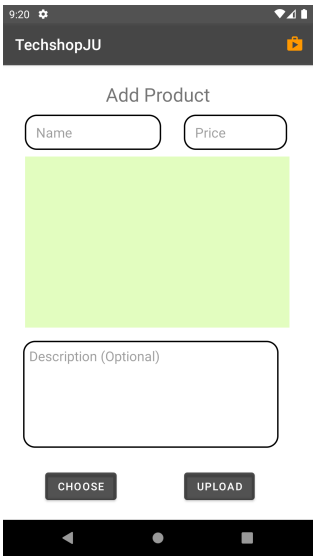

Figure 1.1: Admin: Add Product

- On the pcb section of the application, user provide his designed pcb information then place order.
- If an admin login in this application, he can creat, update or delete any product based on condition.

1.2 My Involvement

Our team follow the agile approach in this project. We work in two sprint. In each sprint every member work in developing a particular feature. In the first sprint I work on developing Admin's CRUD functionality and in sprint two, I work orderPCB features.

1.3 Admin CRUD

Source code of this feature is stored in `https://github.com/abdulmukit98/techshopJU/tree/admin`

This features will help admin to govern the product information in the application. On opening the application, user need to register/login. If the login credentials matches with admin (which is specifically assigned) admin will redirect to Add product view. Admin can insert product information such as name, price, description, images. Figure 1.1 show the user interface to add product. This application store the information in database. Firebase is used in this criteria. Product Image is stored in firebase storage. And

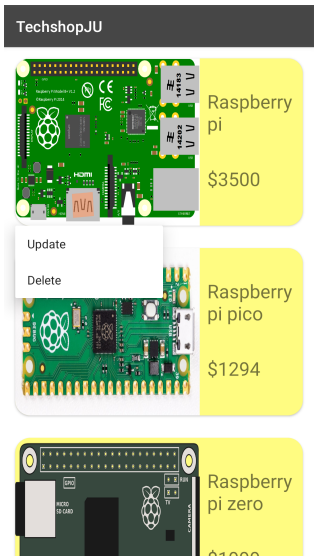

Figure 1.2: Admin: Product Gallery

the information is stored in Realtime Database. The database is viewable in `http://console.firebase.google.com`.

Once the product is stored, admin can go to product gallery. This view is consist of RecyclarView containg list of products. In each product, its image along with name and price is shown. By clicking any product, user can redirect to its detail information page. If admin hold any product in the list, a pop-up menu appear to give admin access to update or delete the product. Admin can update any information of the product when necessary. Admin can also delete the product if it is not available anymore to save database spaces.

1.4 OrderPCB

In Sprint-2 I worked in developing orderPCB feature. The source code is stored in `https://github.com/abdulmukit98/techshopJU/tree/orderPCB`

User can order their customized printed circuit board through our application. To do so, at first user need to develop his pcb layout through any pcb design tool. But KiCad is recommended since it is open-source. After designing, user need to generate gerber file or pdf of the pcb. Then in the PCB section of the application, user have to insert pcb information such as dimension, layers, masking, and the gerber file. Then user can place order by clicking add to cart button.

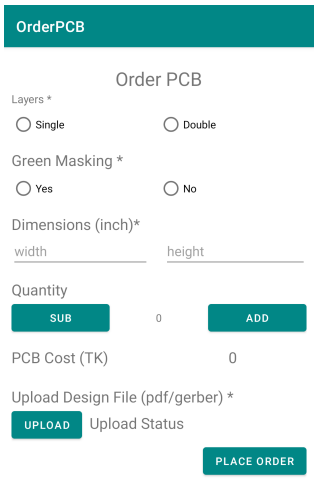

Figure 1.3: OrderPCB view

Functionality	**Tools**
Version Control	Git Bash
Burndown Chart	Trello
Time Tracking	Toggle Track
Documentation	JavaDoc, DoxyGen
Communication Platform	Slack
Meeting	Zoom
Coding ide	Android Studio
Writing tool	Overleaf
Unit Testing	JUnit4
Instrumented Testing	JUnit & Espresso
UI Testing	JUnit4
Continuous Development tool	Jenkins
Database	Firebase

Table 1.1: Tools used in techshopJU

1.5 Tool Used

1.5.1 Git-Bash

Git Bash is used for version control of this project. Our project is stored in the Remote repository `https://github.com/abdulmukit98/techshopJU` Git Bash is used to keep track our local repository with the remote repository. The recommended git-bash command used in this project is documented in github

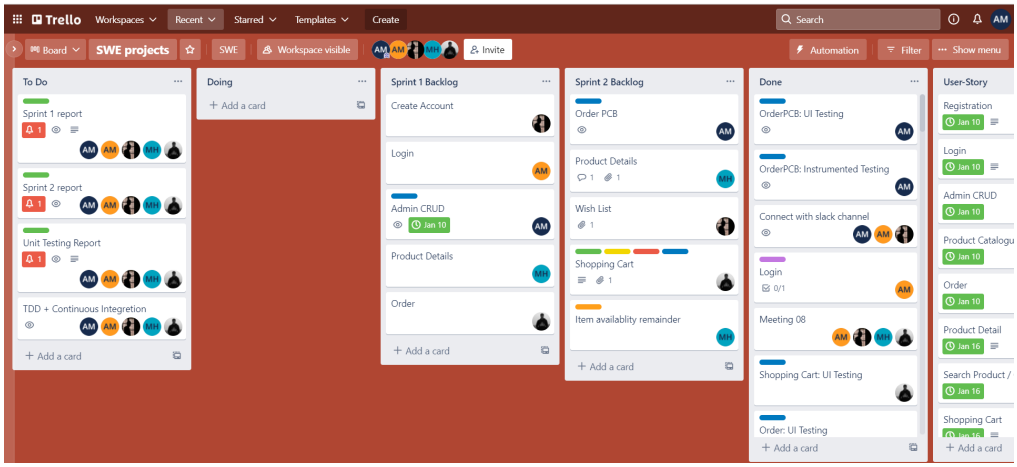

Figure 1.4: Trello Board

wiki `https://github.com/abdulmukit98/techshopJU/wiki/Git-Bash`

1.5.2 Trello

In agile method, burndown chart is required to keep track of work in the project. Trello is used for this perpose. There are different list for varity purpose. ToDo, Doing & Done list keep track of the workflow. Each list can contain multiple card representing work status. Initially a task is created in todo meaning it need tobe done in future. A task can be assigned to any member in the group with deadline.

There are also Sprint-1 Backlog and Sprint-2 Backlog representing the features need tobe performed in the sprint.

1.5.3 Toggle

Toggle is use for time tracking of the project. It will refer how many time a programmer spend in a task. A sequential tasks create a chart in toggle representing the effort of a member.

1.5.4 Documentation tools

Both JavaDoc and DoxyGen are used for documentation in this project. Initially Source code documentation is required for each code. Since we are using Android framework, Java is used for programming. In each java code successful **Block Comment** on each each method is needed. Then the documentation tools is used to generate a web-view format of the source code documentation. Demostration of using documentation tool is shown in our project's wiki page.

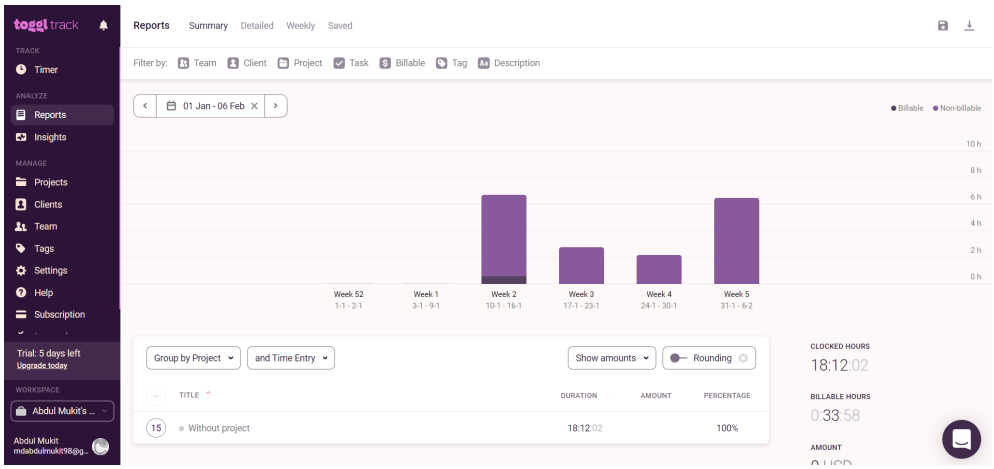

Figure 1.5: Toggle Track

JavaDoc: `https://github.com/abdulmukit98/techshopJU/wiki/Documentation-JavaDoc`
DoxyGen: `https://github.com/abdulmukit98/techshopJU/wiki/Documentation-doxyGen`

1.5.5 Slack

Slack is used as forum. Our Slack workspace id is: `softwareengin-rr29038.slack.com` Different channel are created for discussion in different topic on slack. The slack channels for our project are

- architectural pattern
- coding-standards
- documentation
- general
- unit testing
- user story

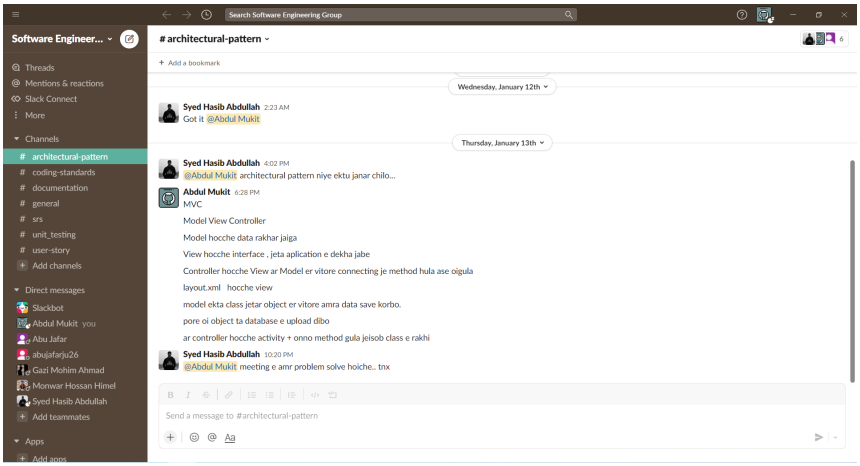

Figure 1.6: Slack Workspace

Chapter 2

Product Backlog

Index	Feature
1	Create Account
2	Login
3	Admin CRUD
4	Product Catalogue
5	Order
6	Product Details
7	Search Product
8	Category
9	Shopping Cart
10	Wish List
11	OrderPCB
12	Item Availableity Remainder

Table 2.1: Product Backlog

Chapter 3

Sprint-1 Process

3.1 About

In the beginning of sprint-1, a meeting is held to sketch the overall sprint procedure. In agile scrum development approach, it is necessary to work in iterative manner. Before sprint process meeting, an agenda need to follow notedown what need to do in the meeting. After a successful meeting, its minutes need tobe published. Oure sprint-1 planning meeting agenda and minutes are stored in the github wiki.
`https://github.com/abdulmukit98/techshopJU/wiki/Sprint-1-Planning-Agenda`
`https://github.com/abdulmukit98/techshopJU/wiki/Sprint-1-Planning-Minutes`

3.2 Sprint-1 planning meeting

In this we decide what need to do in this sprint. Slack workspace and Trello board was created after this meeting. Android is decided tobe the projects framework. Github account is created for individual member. And Github wiki pages is written for a cse programmer to understand thhis project. Coding Standards for our project is maintained with the following one `https://github.com/abdulmukit98/techshopJU/wiki/Coding-Standards-for-JAVA`.

3.3 Sprint-1 Backlog

In this meeting, 5 features is choosed for sprint-1 backlog from the product backlog. The features for Sprint-1 backlog is assigned as:

Features	**Assigned to**
Create Account	Gazi Muhim Ahmed
Login	Abu Jafar
Admin CRUD	Abdul Mukit
Product Details	Monwar Hossen
Order	Hasib Abdullah

Table 3.1: Sprint-1 Backlog

Chapter 4

Sprint-1 Retrospective

4.1 Daily Scrum

Daily scrum is the iterative approach which is performed daily to keep attached to the project in agile manner. Daily Scrum include a 10-15 min meeting where each member answer three question.

- What did you do yesterday
- What will you do today
- Do you face any problem.

4.2 My Involvement

In sprint-1 I was working on admin's crud functionality where I was establishing communication between the application with database. I used Firebase as database. Since I work on both information and image, I had to integrate both Realtime database with Firebase storage (for media file). Moreover, I had to implement a recyclar view to display the product's list.

I used MVC as architectural pattern in this feature. This concept is based on three term.

- model
- view
- controller

A Model is the simplest data repository for this project. Every model needs tobe transferred into database for flexibility.

Listing 4.1: Admin: Model

```
public class Product implements Serializable {
```

```
@NotNull
private String productId;
private String productName;
private int productPrice;
private String productImageLink;
private String productDescription;

public Product() {

}

public String getProductId() {
    return productId;
}

public void setProductId(String productId) {
    this.productId = productId;
}

public String getProductName() {
    return productName;
}

public void setProductName(String productName) {
    this.productName = productName;
}

public int getProductPrice() {
    return productPrice;
}

public void setProductPrice(int productPrice) {
    this.productPrice = productPrice;
}

public String getProductImageLink() {
    return productImageLink;
}

public void setProductImageLink(String productImageLink) {
    this.productImageLink = productImageLink;
}

public String getProductDescription() {
    return productDescription;
}
```

```
    public void setProductDescription(String productDescription) {
        this.productDescription = productDescription;
    }

    @Override
    public String toString() {
        return "Product{" +
                "productId='" + productId + '\'' +
                ", productName='" + productName + '\'' +
                ", productPrice=" + productPrice +
                ", productImageLink='" + productImageLink + '\'' +
                ", productDescription='" + productDescription + '\'' +
                '}';
    }
}
```

Basically a model is a class that contain information for the feature. It can get different methods such as Constructor, Setters, Getters, ToString etc.

A view is the layout that is displayed in the feature. User can interact with view to access model.

Listing 4.2: Admin: View

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Add Product"
        android:textSize="25sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.038" />

    <EditText
        android:id="@+id/mEdtName"
```

```
        android:layout_width="181dp"
        android:layout_height="47dp"
        android:background="@drawable/edit_text_border"
        android:hint="Name"
        android:paddingLeft="15dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.126"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.118" />

    <ImageView
        android:id="@+id/mImageView"
        android:layout_width="351dp"
        android:layout_height="227dp"
        android:background="@color/greenish"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.487"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.322" />

    <EditText
        android:id="@+id/mEdtPrice"
        android:layout_width="137dp"
        android:layout_height="47dp"
        android:background="@drawable/edit_text_border"
        android:hint="Price"
        android:inputType="number"
        android:paddingLeft="15dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.875"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.118" />

    <EditText
        android:id="@+id/mEdtDescription"
        android:layout_width="339dp"
        android:layout_height="142dp"
        android:layout_marginTop="16dp"
        android:background="@drawable/edit_text_border"
        android:gravity="top"
```

```
        android:hint="Description (Optional)"
        android:inputType="textMultiLine"
        android:padding="8dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.371"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/mImageView"
        app:layout_constraintVertical_bias="0.015" />

    <Button
        android:id="@+id/mBtnUpload"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="27dp"
        android:text="Upload"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.757"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/mEdtDescription"
        app:layout_constraintVertical_bias="0.0" />

    <Button
        android:id="@+id/mBtnChoose"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="27dp"
        android:text="Choose"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.177"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/mEdtDescription"
        app:layout_constraintVertical_bias="0.0" />

    <ProgressBar
        android:id="@+id/mProgress"
        style="@style/Widget.AppCompat.ProgressBar.Horizontal"
        android:layout_width="382dp"
        android:layout_height="16dp"
        android:indeterminate="true"
        android:indeterminateBehavior="cycle"
        android:indeterminateTint="@color/black"
        app:layout_constraintBottom_toTopOf="@+id/textView"
        app:layout_constraintEnd_toEndOf="parent"
```

```
        app:layout_constraintHorizontal_bias="0.448"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.0" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Controller class held all the logic implemented for this features

After completing all the functionality of admin crud functionality, I had to documented properly in source code. I used JavaDoc for documentation generator. The built-in javaDoc in Android Studio has a bug issue since no api package is detected in javaDoc. I used IntelliJ Idea to generate JavaDoc for my project.

I tested my code using JUnit1.4.0 Unit testing tools.

4.3 Sprint Review (product)

In sprint-1 the basic operation of the application was fulfilled. Since an user can simply authenticate the application, then view product's gallery. If he finds any product interesting, he can simply buy it by placing an order.

4.4 Sprint Retrospective (process)

In Sprint-1 User Authentication is successfully performed. User can order 1 product at at time. Sprint-1 fulfill the very basic functionality for an e-commerce application.

The following things can be improved in next sprint.

- Display a product detail information page.
- Search a product from the gallery.
- Use a shopping cart for multiple purchase.
- Have a wish list to store favourite products.

Chapter 5

Sprint-2 Process

5.1 Intro

We have complete a successful sprint last week. Chapter 4 show the working of sprint-1. Based on sprint-1, we need to prepare our Sprint-2 backlog for continuous development.

5.2 Sprint-2 Planning Meeting

Our Sprint-2 planning meeting was held in Jan,11 to Jan,12 2022. Several decision was made in this meeting.

- Time must be tracked in a gentle manner using toggle.
- Our documentation tools need to be upgrade since javaDoc failed in last sprint in android-studio ide.
- we have to strickly maintain coding standards for JAVA described in wiki `https://github.com/abdulmukit98/techshopJU/wiki/Coding-Standards-for-JAVA`
- Have some extra time in Unit testing tool (JUnit).
- Assign time limit for performing a task in burndown chart (trello).
- Reschedule Product Backlog for project improvement.

The meeting agenda and minutes are recorded here.
`https://github.com/abdulmukit98/techshopJU/wiki/Sprint-2-Planning-Meeting-Agenda`
`https://github.com/abdulmukit98/techshopJU/wiki/Sprint-2-Planning-Meeting-Minutes`

5.3 Backlog

Our Sprint-2 backlog was declared in the meeting from the product backlog in table 2.1

Features	Assigned to
Product details	Monwar Hossain
Shopping Cart	Hasib Abdullah
Wish List	Gazi Muhim Ahmed
Order PCB	Abdul Mukit
Search Product	Abu Jafar

Table 5.1: Sprint-2 Backlog

5.4 Improvement

There are few things that hlep improve our product. Our trello board reflect the workflow of the product. We can estimate time by experience and calculating total cyclomatic complexity of code. We need to store our source code in different branch and merge only after successful build. We can also use continuous integration for our project.

Chapter 6

Sprint-2 Retrospective

6.1 Daily Scrum

Daily scrum meeting is the consistent approach of solving a task. In a sprint, each Daily scrum meeting contain 3 questions.

- What was done last day
- What will be done today
- Which problem is faced in solving task.

In our project's github wiki as sequential meeting is recorded representing daily scrum in sprint.

6.2 My Involvement

In sprint-2 I was working with OrderPCB features. Using this feature an user can order his customized pcb through the application. I used mvc architectural pattern in this feature. Where each model object held the information of the ordered pcb.

Listing 6.1: orderPCB: model

```
/**
 * This is our model <br>
 * This class contain the information of the pcb we are going to order <br>
 * this model is uploaded in database.
 */
public class PCBDetails
{

    int cost, quantity;
```

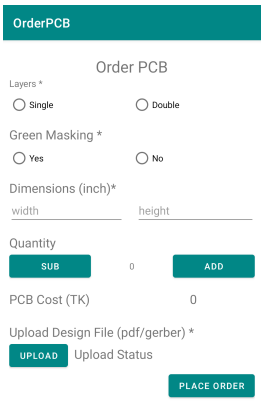

Figure 6.1: OrderPCB UI

```
private String orderId;
private boolean isSingle, isMasking;
private double width, height;
private String fileURL;

/**
 * null constructor
 */
public PCBDetails()
{

}

/**
 * Constructor with requirments of pcb.
 *
 * @param orderId    Unique identification key for the pcb
 * @param isSingle   if true then pcb is single layered, false --> double lay
 * @param isMasking  true --> green masking enable in pcb.
 * @param width      dimension
 * @param height     dimension
 * @param quantity   no of pcb
 */
public PCBDetails(String orderId, boolean isSingle, boolean isMasking, doubl
{
    this.orderId = orderId;
    this.isSingle = isSingle;
```

```
        this.isMasking = isMasking;
        this.width = width;
        this.height = height;
        this.quantity = quantity;
        this.cost = (int) (width * height * 20 * quantity);

    }

    /**
     * receive order id from another class
     *
     * @return give the order id
     */
    public String getOrderId()
    {
        return orderId;
    }

    /**
     * To set order if for a order
     *
     * @param orderId the id which will be assigned with the product
     */
    public void setOrderId(String orderId)
    {
        this.orderId = orderId;
    }

    /**
     * get if pcb is single or not
     *
     * @return
     */
    public boolean isSingle()
    {
        return isSingle;
    }

    /**
     * define pcb single or double
     *
     * @param single true or false
     */
    public void setSingle(boolean single)
    {
        isSingle = single;
```

```
}

/**
 * get the masking condition of pcb
 *
 * @return
 */
public boolean isMasking()
{
    return isMasking;
}

/**
 * define if masking will be implemented in pcb
 *
 * @param masking
 */
public void setMasking(boolean masking)
{
    isMasking = masking;
}

/**
 * get dimension
 *
 * @return double format of width
 */
public double getWidth()
{
    return width;
}

/**
 * define width of pcb
 *
 * @param width a non zero double
 */
public void setWidth(double width)
{
    this.width = width;
}

/**
 * To get height of the pcb
 *
 * @return height
```

```
 */
public double getHeight()
{
    return height;
}

/**
 * To define height of the pcb
 *
 * @param height a non zero double
 */
public void setHeight(double height)
{
    this.height = height;
}

/**
 * Get cost required in the order
 *
 * @return integer represent cost
 */
public int getCost()
{
    return cost;
}

/**
 * define cost of the order
 *
 * @param cost nonzero integer
 */
public void setCost(int cost)
{
    this.cost = cost;
}

/**
 * get the number of pcb ordered
 *
 * @return
 */
public int getQuantity()
{
    return quantity;
}
```

```
/**
 * define quntity of pcb in an order
 *
 * @param quantity
 */
public void setQuantity(int quantity)
{
    this.quantity = quantity;
}

/**
 * retrive the http location of the schematic uploaded to database
 *
 * @return Url
 */
public String getFileURL()
{
    return fileURL;
}

/**
 * set the url of schematic
 *
 * @param fileURL web url
 */
public void setFileURL(String fileURL)
{
    this.fileURL = fileURL;
}

/**
 * Show all the information of the orderPCB
 *
 * @return
 */
@Override
public String toString()
{
    return "PCBDetails{" +
            "orderId='" + orderId + '\'' +
            ", isSingle=" + isSingle +
            ", isMasking=" + isMasking +
            ", width=" + width +
            ", height=" + height +
            ", cost=" + cost +
            ", quantity=" + quantity +
```

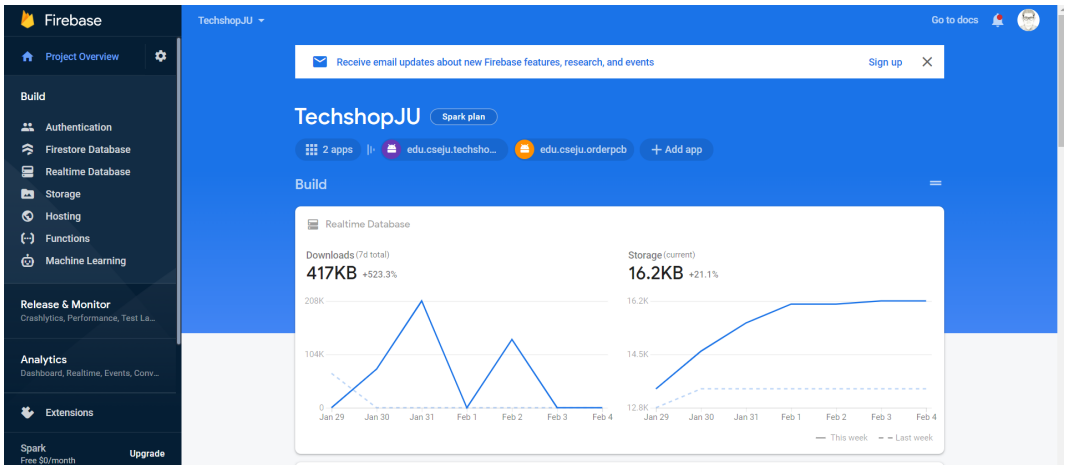

Figure 6.2: Firebase database

```
                ",␣fileURL='" + fileURL + '\'' +
                '}';
    }
}
```

This model is then stored in Firebase. View is the ui part where user can interact with the application. In android studio project, layout.xml file is the view

Listing 6.2: View XML file

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/mainLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="10dp"
    tools:context=".MainActivity">

    <ProgressBar
        android:id="@+id/progress"
        style="@style/Widget.AppCompat.ProgressBar.Horizontal"
        android:layout_width="match_parent"
        android:layout_height="15dp"
        android:indeterminate="true"
        android:indeterminateTint="@color/black" />
```

```
<TextView
    android:id="@+id/tvTitle"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center_horizontal"
    android:text="Order PCB"
    android:textSize="25sp" />


<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Layers *"
    android:textSize="15sp" />

<RadioGroup
    android:id="@+id/rgroupLayer"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="10dp"
    android:orientation="horizontal">

    <RadioButton
        android:id="@+id/rbSingle"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Single" />

    <RadioButton
        android:id="@+id/rbDouble"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Double" />
</RadioGroup>


<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Green Masking *"
    android:textSize="20sp" />

<RadioGroup
    android:id="@+id/rgroupMasking"
```

```
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="10dp"
        android:orientation="horizontal">

        <RadioButton
            android:id="@+id/rbYes"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Yes" />

        <RadioButton
            android:id="@+id/rbNo"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="No" />

    </RadioGroup>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Dimensions (inch)*"
        android:textSize="20sp" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="15dp"
        android:orientation="horizontal">

        <EditText
            android:id="@+id/edtWidth"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginRight="10dp"
            android:layout_weight="1"
            android:hint="width"
            android:inputType="number" />

        <EditText
            android:id="@+id/edtHeight"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
```

```
            android:layout_marginLeft="10dp"
            android:layout_weight="1"
            android:hint="height"
            android:inputType="number" />
</LinearLayout>

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Quantity"
    android:textSize="20sp" />

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="15dp"
    android:orientation="horizontal">

    <Button
        android:id="@+id/btnSub"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="sub" />

    <TextView
        android:id="@+id/tvQuantity"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:gravity="center_horizontal"
        android:text="0" />

    <Button
        android:id="@+id/btnAdd"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Add" />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">
```

```
    <TextView
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginBottom="25dp"
        android:layout_weight="1"
        android:text="PCB Cost (TK)"
        android:textSize="20sp" />

    <TextView
        android:id="@+id/tvCost"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginBottom="25dp"
        android:layout_weight="1"
        android:gravity="center"
        android:text="0"
        android:textSize="20sp" />
</LinearLayout>

<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Upload Design File (pdf/gerber) *"
    android:textSize="20sp" />

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">

    <Button
        android:id="@+id/btnUpload"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Upload" />

    <TextView
        android:id="@+id/tvUploadStatus"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dp"
        android:text="Upload Status"
        android:textSize="20sp" />
</LinearLayout>

<Button
```

```
        android:id="@+id/btnCart"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:text="Place order" />

</LinearLayout>
```

And Controller works as bridge between model and view. All the interconnection between model and view is the property of controller. In orderPCB feature, I used MainActivity.java as controller.

Listing 6.3: orderPCB Controller

```
public class MainActivity extends AppCompatActivity implements RadioGroup.OnChe
{

    private static final int REQUEST_PARAM = 5;
    private final boolean uploadStatus = false;
    RadioGroup rgLayers, rgMasking;
    RadioButton rbSingle, rbDouble, rbYes, rbNo;
    EditText edtWidth, edtHeight;
    TextView tvQuantity, tvCost, tvUploadStatus;
    Button btnSub, btnAdd, btnUploadFile, btnCart;
    ProgressBar progressBar;
    TestClass testClass = new TestClass();
    DatabaseReference databaseReference;
    StorageReference storageReference;
    private int mQuantity = 0, mCost = 0;
    private boolean isLayerSingle, isMaskingYes;
    private Uri fileUri;

    /**
     * This method trigger when any radio_button in the radio_group is checked.
     *
     * @param group     contain the radio_group which is checked
     * @param checkedId the id of the button.
     */
    @Override
    public void onCheckedChanged(RadioGroup group, int checkedId)
    {
        if (group == rgLayers)
        {
            switch (checkedId)
            {
                case R.id.rbSingle:
                    isLayerSingle = true;
```

```
                break;
            case R.id.rbDouble:
                isLayerSingle = false;
                break;
        }
    }
    else if (group == rgMasking)
    {
        switch (checkedId)
        {
            case R.id.rbYes:
                isMaskingYes = true;
                break;
            case R.id.rbNo:
                isMaskingYes = false;
                break;
        }
    }
    //
}

/**
 * Trigger if any button is clicked.
 *
 * @param v represent the button.
 */
@Override
public void onClick(View v)
{
    switch (v.getId())
    {
        case R.id.btnAdd:
            addItem();
            break;
        case R.id.btnSub:
            removeItem();
            break;
        case R.id.btnUpload:
            uploadSchematic();
            break;
        case R.id.btnCart:
            addToCart();
            break;
    }
    //
}
```

```
/**
 * when button add is clicked this method run <br>
 * it will add the pcb quantity for order, then show in the textview
 */
private void addItem()
{
    mQuantity = mQuantity + 1;
    tvQuantity.setText(String.valueOf(mQuantity));

    mCost = calculateCost(edtWidth.getText().toString(), edtHeight.getText()
    tvCost.setText(mCost + "");
}

/**
 * This method calculate cost required to place order pcb
 *
 * @param width     pcb width
 * @param length    pcb length
 * @param quantity no of pcb ordered
 * @return Cost to place order
 */
public int calculateCost(String width, String length, int quantity)
{
    double wid = Double.parseDouble(width);
    double len = Double.parseDouble(length);

    double cost = wid * len * 20 * quantity;

    return (int) cost;
}

/**
 * This method run to reduce an quantity of pcb ordered.
 * the quantity can not be negetive.
 */
private void removeItem()
{
    mQuantity = mQuantity - 1;
    if (testClass.checkNegetive(mQuantity) == true)
        mQuantity = 0;
    tvQuantity.setText(String.valueOf(mQuantity));

    mCost = calculateCost(edtWidth.getText().toString(), edtHeight.getText()
    tvCost.setText(mCost + "");
}
```

```
/**
 * This button send an intent to read pcb schematic file from android filesto
 * this intent is set for any type of file. <br>
 * a reques_param constant is add to uniquely identify the received file
 */
public void uploadSchematic()
{
    Intent intent = new Intent();
    intent.setAction(Intent.ACTION_GET_CONTENT);
    intent.setType("*/*");
    startActivityForResult(intent, REQUEST_PARAM);
}

/**
 * This method run to place the order and save the order information into da
 * At first check if user properly complete the order process. <br>
 * it generate an unique key to identify the order in database <br>
 * A progressbar show the database store progress <br>
 * if order successfully uploaded , success message given.
 * if there is any problem uploading order, a toast is shown containing fail
 */
private void addToCart()
{

    if (rbSingle.isChecked() == false && rbDouble.isChecked() == false)
    {
        message(getApplicationContext(), "Select Layers type");
        return;
    }
    if (rbYes.isChecked() == false && rbNo.isChecked() == false)
    {
        message(getApplicationContext(), "Select Masking");
    }

    if (fileUri == null)
    {
        message(getApplicationContext(), "upload schemaic");
        return;
    }
    String fileExtension = getFileExtension(fileUri);

    String wid = edtWidth.getText().toString();
    String len = edtHeight.getText().toString();
    mCost = calculateCost(wid, len, mQuantity);
    tvCost.setText(mCost + "");
```

```
String key = databaseReference.push().getKey();
PCBDetails pcbDetails = new PCBDetails(key, isLayerSingle, isMaskingYes,
        Double.parseDouble(wid), Double.parseDouble(len), mQuantity);

progressBar.setVisibility(View.VISIBLE);
storageReference.child(key + fileExtension + "").putFile(fileUri).addOn
{
    @Override
    public void onSuccess(UploadTask.TaskSnapshot taskSnapshot)
    {

        Task<Uri> task = taskSnapshot.getMetadata().getReference().getDo
        task.addOnSuccessListener(new OnSuccessListener<Uri>()
        {
            @Override
            public void onSuccess(Uri uri)
            {
                System.out.println(uri.toString());

                pcbDetails.setFileURL(uri.toString());
                databaseReference.child(key).setValue(pcbDetails).addOn
                {
                    @Override
                    public void onSuccess(Void unused)
                    {
                        message(getApplicationContext(), "success");
                    }
                }).addOnFailureListener(new OnFailureListener()
                {
                    @Override
                    public void onFailure(@NonNull Exception e)
                    {
                        message(getApplicationContext(), e.getMessage())
                    }
                });
                progressBar.setVisibility(View.INVISIBLE);
            }
        });

    }
}).addOnFailureListener(new OnFailureListener()
{
    @Override
    public void onFailure(@NonNull Exception e)
    {
```

```
                progressBar.setVisibility(View.INVISIBLE);
                message(getApplicationContext(), e.getMessage());
            }
        });


}

/**
 * A simple Toast show with given message
 *
 * @param context The contex where toast will be shown.
 * @param msg     Message which will display
 */
void message(Context context, String msg)
{
    if (testClass.validMessage(msg) == true)
        Toast.makeText(context, msg, Toast.LENGTH_SHORT).show();
}

/**
 * This method receive the file Uri and assume the extension type of the fil
 *
 * @param fileUri The Uri of the file
 * @return String containing the extension type
 */
public String getFileExtension(Uri fileUri)
{
    ContentResolver contentResolver = getContentResolver();
    String details = contentResolver.getType(fileUri);
    return details.substring(details.indexOf('/') + 1);
}

/**
 * If the schematic is successfully picked from storage, this method activat
 * it match the request code with the parameter sent with intent to uniquely
 * an Uri (Unique resource identifier) generate to hold the file.
 *
 * @param requestCode The constant passed with intent
 * @param resultCode  check is file is fully received
 * @param data        contain the data of the file.
 */
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable I
{
    super.onActivityResult(requestCode, resultCode, data);
```

```
        if (requestCode == REQUEST_PARAM && resultCode == RESULT_OK && data != n
        {
            fileUri = data.getData();
            tvUploadStatus.setText("Upload Successful");
        }
    }

    /**
     * Runs at the start of the activity. <br>
     * All the layout_view will be defined here.
     * initialize setup
     */
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        rgLayers = findViewById(R.id.rgroupLayer);
        rgMasking = findViewById(R.id.rgroupMasking);
        rbSingle = findViewById(R.id.rbSingle);
        rbDouble = findViewById(R.id.rbDouble);
        rbYes = findViewById(R.id.rbYes);
        rbNo = findViewById(R.id.rbNo);
        edtHeight = findViewById(R.id.edtHeight);
        edtWidth = findViewById(R.id.edtWidth);
        tvQuantity = findViewById(R.id.tvQuantity);
        tvCost = findViewById(R.id.tvCost);
        tvUploadStatus = findViewById(R.id.tvUploadStatus);
        btnAdd = findViewById(R.id.btnAdd);
        btnSub = findViewById(R.id.btnSub);
        btnCart = findViewById(R.id.btnCart);
        btnUploadFile = findViewById(R.id.btnUpload);
        progressBar = findViewById(R.id.progress);
        progressBar.setVisibility(View.INVISIBLE);

        rgMasking.setOnCheckedChangeListener(this);
        rgLayers.setOnCheckedChangeListener(this);
        btnAdd.setOnClickListener(this);
        btnSub.setOnClickListener(this);
        btnUploadFile.setOnClickListener(this);
        btnCart.setOnClickListener(this);

        mCost = 0;
        databaseReference = FirebaseDatabase.getInstance().getReference("pcb");
        storageReference = FirebaseStorage.getInstance().getReference("pcb");
```

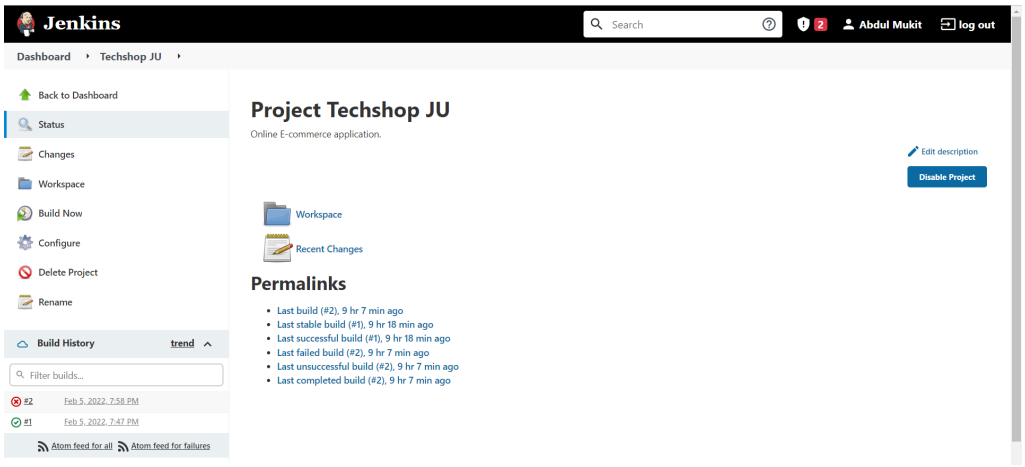

Figure 6.3: Jenkins Project Dashboard

```
    }
}
```

After efficient coding and commenting, I use doxyGen as Documentation tool. The working principal of doxyGen is stored in projects wiki page `https://github.com/abdulmukit98/techshopJU/wiki/Documentation-doxyGen`

Then I perfor Unit Testing of this feature. To do so a testClass meed to define containing function that need to be tested. Then generate testing class, test case for the function, and perform testing. I also perform UI testing and Instrumented testing for this feature.

I install Jenkins as Continuous Integration tool. Configure project and build. The initial build passed but 2nd time it failed. I am still working on troubleshooting it.

6.3 Sprint Review (product)

After Sprint-2, it is assumed that the application can fulfill the primary need of user. User can successfully visit product detail page the add product in his favorite, or cart to purchase. User can also search specific product from the catalogue. User can order his customized printed circuit board for his electrical projects.

6.4 Sprint Retrospective (process)

In sprint-2 about 70% criteria of an e-commerce application is reflected in our project. There still some features that can be improved in future.

- Digital Payment Method.

- Item Availability Remainder
- Help desk
- 2-step authentication
- password recovery and so on.

Chapter 7

Unit Testing

7.1 Intro

Unit testing is the way of assuring correct functionality of a method. An unit is a sequential set of instruction that perform specific operation. But based on the parameter and cyclomatic complexity, and unit can show multiple behaviour based on the environment. Unit testing help us verify those functionality.

7.2 Explain Unit Testing

To perform unit testing we have to perform following task

- Add following JUnit4 and Truth library dependency into build.gradle file

Listing 7.1: Unit Test Build

```
testImplementation 'junit:junit:4.+'
testImplementation 'com.google.truth:truth:1.0.1'
```

- Create a Test Model with method / unit that need to be tested

Listing 7.2: TestModel.java

```
package edu.cseju.orderpcb.TestModel;

/**
 * This class is created to perform testing
 * basically unit testing will perform here. <br>
 * method in this class will be tested with possible test cases.
 */
public class TestClass
{
```

```
    /**
     * null constructor
     */
    public TestClass()
    {
    }


    /**
     * Check if an integer is negative or not
     *
     * @param quantity the input parameter
     * @return true if quentitu -ve , false if +ve
     */
    public boolean checkNegetive(int quantity)
    {
        return (quantity < 0) ? true : false;
    }

    /**
     * If there is valid message, then show message as toast
     * @param msg   String message
     * @return true if message is valid
     */
    public boolean validMessage(String msg)
    {
        return (msg == null || msg == "")? false: true;
    }
}
```

- Generate testingClass using JUnit4 library

Listing 7.3: TestingClass.java

```
package edu.cseju.orderpcb.TestModel;

import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class TestClassTesting
{

    TestClass testClass = new TestClass();

    /**
     * check negetive
```

```
     * possible test case "boundary testing"
     * <p>
     * quantity = 0   false
     * quantity > 0   false
     * quantity < 0   true
     */
    @Test
    public void checkNegetive()
    {
        int quantity = -5;
        assertEquals(true, testClass.checkNegetive(quantity));
    }

    /**
     * valid quantity, check_negetive false
     */
    @Test
    public void checkNegetive2()
    {
        int quantity = 6;
        assertEquals(false, testClass.checkNegetive(quantity));
    }

    /**
     * null quantity
     */
    @Test
    public void checkNegetive3()
    {
        int quantity = 0;
        assertEquals(false, testClass.checkNegetive(quantity));
    }

    /**
     * For validMessage, possible test case
     * message = null        false
     * message = ""          empty false
     * message = "something"  true
     */
    @Test
    public void validMessage1()
    {
        String msg = null;
        assertEquals(false, testClass.validMessage(msg));
    }
```

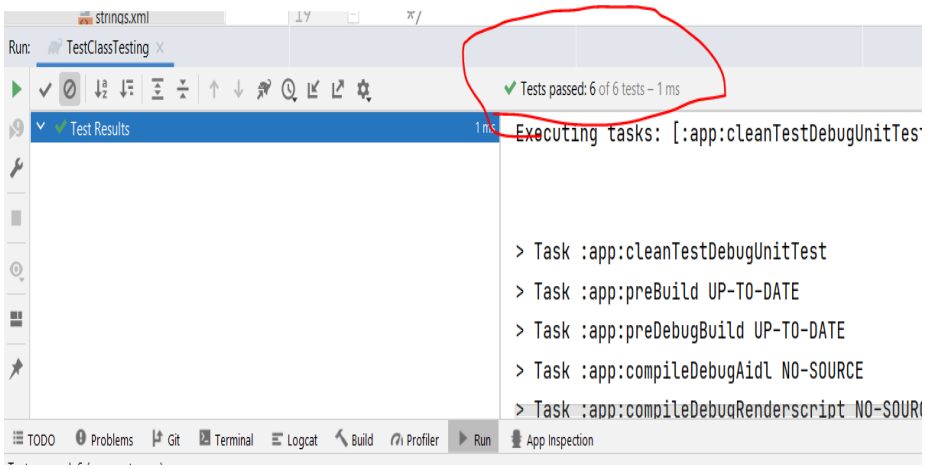

Figure 7.1: Unit Testing Verdict

```
    /**
     * empty message , expected false
     */
    @Test
    public void validMessage2()
    {
        String msg = "";
        assertEquals(false, testClass.validMessage(msg));
    }

    /**
     * for valid message
     */
    @Test
    public void validMessage3()
    {
        String msg = "message msg1";
        assertEquals(true, testClass.validMessage(msg));
    }


}
```

- assertEqual() method help us match our assumptions with outcome.
- Verdict

7.3 Effects of Unit Testing

Unit testing satisfy all the code unit work as expected. It help fixing bus easily. Besides, user is encourage user develop better logic and algorithms. To make unit testing easier, developer need to make unit simple to understand.

7.4 Personal Experience

I followed YouTube for android unit testing. Maximum videos are made with Kotlin, I spend time to realize it in equivalent Java format, then apply in my code. I also performed UI Testing and Instrumented testing.

Chapter 8

UI testing

8.1 Intro

UI testing is used to check if user interface designed in view are stable when operating. I use Espresso-3.4.0 for android user interface testing. UI testing work on API level thats why need presence of usb debugging or emulator.

8.2 Procedure

Documented on github wiki: `https://github.com/abdulmukit98/techshopJU/wiki/UI-Test-Espresso`

8.2.1 Step-1: Include gradle dependency

Listing 8.1: UI test Dependency

```
androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'
androidTestImplementation 'androidx.test:runner:1.4.0'
androidTestImplementation 'androidx.test:rules:1.4.0'
androidTestImplementation 'androidx.test:core:1.4.0'
```

8.2.2 Step-2: Generate testClass for context/fragments

Press alt+insert to generate test. Use JUnit4 and save it in **Android Test** package.

8.2.3 Test Functions

- For checking if a view is dislayed in UI

Listing 8.2: UI Test Functions

```
@Test
public void test_MainActivity_isViewed()
{
    ActivityScenario<MainActivity> scenario = ActivityScenario.launch(MainA
    onView(withId(R.id.mainLayout)).check(matches(isDisplayed()));
}

@Test
public void dimension_height_is_shown()
{
    ActivityScenario<MainActivity> scenario = ActivityScenario.launch(MainA
    onView(withId(R.id.edtHeight)).check(matches(isDisplayed()));
}
```

here **ActivityScenario** is used to launch the activity that is going to-be tested

- Test Case for checking String

Listing 8.3: UI Testing String match

```
@Test
public void is_orderPCB_textView_matches_givenString()
{
    ActivityScenario<MainActivity> scenario = ActivityScenario.launch(M
    onView(withId(R.id.tvTitle)).check(matches(withText("Order PCB")));
}
```

- Test case for checking **inoutType**

Listing 8.4: EditText InputType Test

```
@Test
public void test_isEditText_width_inputType_number()
{
    ActivityScenario<MainActivity> scenario = ActivityScenario.launch(M
    onView(withId(R.id.edtHeight)).check(matches(withInputType(TYPE_CLAS
}
```

Chapter 9

Continuous Integration: Jenkins

9.1 Intro

Continuous Devlopment and Continuous Integration is the process of automatic integration and deployment of code when multiple contributor work in a project. It is a good practics in agile software development. There are many tools that help developer to integrate their code. Some of them are:

- Jenkins
- Travis CI
- GitLab
- Circle CI
- TeamCity and so on

I am trying to use Jenkins since it is open-source.

9.2 How to use

Install Jenkins `https://www.jenkins.io/`
Jenkins required JAVA 11 or JAVA1.8 to install. After Successful installation, jenkins will be available in `http://localhost:8080/`
Authenticate to jenkins. Then Manage jenkins → configure system. Add android sdk location. Then github credentials.

Goto Manage jenkins → Global tool configuration. Set JDK, Git, Gradle location.

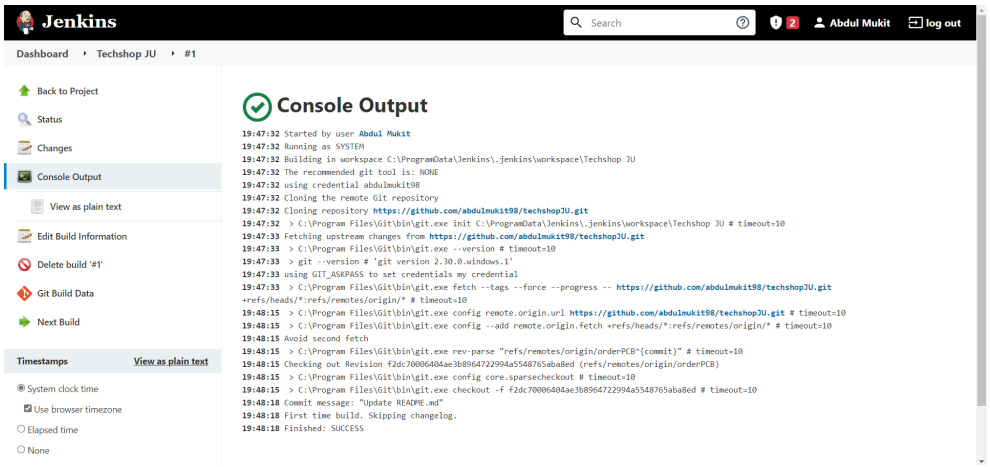

Figure 9.1: Build Success

In Dashboard, create NewItem using Freestyle Project. Then Configure project JDK, Source code management Git, specify which branch to build, invoke gradle. In Task specify **build –stacktree**. Click apply then save. Then Build the project manually.

Figure 9.1 show us the console if build successful. If the build is failed, proper guideline will be provided in console.

Figure 9.2 represent a build that failed in jenkins. Need to change approach.

9.3 My Experience

I am new at running jenkins. I am still working on fixing it. I followed this tutorial for setting up jenkins android `https://www.youtube.com/watch?v=h74i7krG33Y`

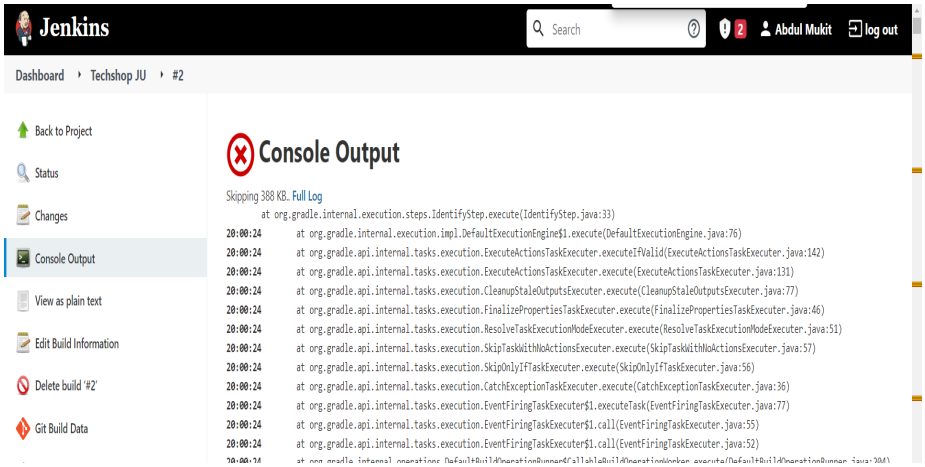

Figure 9.2: Build Failed