INVENTORY MANAGEMENT FOR RETAILERS

(TAILORING APPLICATION)

1. INTRODUCTION:

The dress is part of our lifestyle, when it comes to ladies, mainly their dress was stitched by buying materials. This project report for the stitching tuning / Tailoring unit explains the viability of the project in this locality. As the fashion changes, population increases, the demand for stitching units will also increase. Prompt delivery and affordable price are the key elements to success here. The sector also created newer avenues for many businesses and entrepreneurs based in the locality. Though there are a lot of stitching units, the demand for another stitching unit is also very high. This profession gives the entrepreneur a decent income and the opportunity to employ multiple ladies.

1.1 PROJECT OVERVIEW:

The online tailoring management system will eliminate all these manual interventions and increase the speed of the whole process. The system will allow customers to register online and successfully submit their measurements. The system has inbuilt validation system to validate the entered data. The customer can login to the system to check on the status of the clothes for collection. The system will show the already completed garments for clients to collect. The system also provides information about the cost of each garment the customer intends to get knit. This data will be stored in the database for further reference or audit

1.2 PURPOSE:

Automate the current manual tailoring system and maintain a searchable customer, product database, maintain data security and user rights. To enable customers to send their measurements to their tailors for their clothes to be made. To provide information about the cost, the fabric type, the urgency at which a customer wants the dress finished, the type of material to be

used and quantity in terms of pairs needed. To compute the total cost depending on the selected fabric, type of material, quantity and duration and avails that information to the customer. To enable report generation: it is able to give a report of finished garments to the clients for collection and bookings made, administrator is able to view all the customers and their details, finished garments and all the bookings made. To create a data bank for easy access or retrieval of customer details, orders placed and the users who registered to the system.

2. LITERATURE SURVEY:

2.1 EXISTING PROBLEM:

Tailors work in poor working environment, overwork, job insecurity, lack of opportunities. Tailors face many health related problems like Headache, vision related problems, pain in joint, wheezing problems, tension, stomach ache/ ulcer, and anemia. Many jobs have been lost in this industry.Questionable future job prospects.Tailors have a rather low social status.Relatively low salary for tailors.You may even need a second job to pay your bills.Hard to plan your future as a tailor.

2.2 REFERENCES:

[1] Paula Deitz (25 August 1996). "Savile Row's Ambassador to the Court of Kings".The New York Times.Retrieved 9 January 2009.

[2] Dunn, Bill (14 April 2003). "The Battle for Savile Row".BusinessWeek.Retrieved 9 January 2009.

[3]Cooper, H. (1998). Synthesizing Research: A Guide for literature Reviews

[4]Norton, Kate (31 October 2006). "Savile Row Never Goes Out of Style". Business Week. Retrieved 9 January 2009.

[5] "Hardy Amies UK stores to close following sale to Fung Capital". Retail Week. 2008-11-11. Retrieved 2009-10-08.

[6] Piet Schreuders, Adam Smith, Mark Lewisohn (30 Jun 2008). Beatles London: The Ultimate Guide to Over 400 Beatles Sites in and Around London. Anova Books.pp. 53.

[7] U.S. Bureau of Labor Statistics (BLS),

http://degreedirectory.org/articles/Tailor_How_to_Become_a_Professional_Tailor_in_5_Ste ps.html

[8] Tailoring software (For ladies/gents tailoring shop) - www.assersoft.com

2.3 PROBLEM STATEMENT DEFINITION:

At present customers need to walk around the tailor shops to get their estimations takenfor the fitting of their vestments. Their nuances are taken and kept on papers. Customersadditionally need to move from their working environments to continue to check for thepieces of clothing whether there complete or not. This is monotonous and extravagant.Due to the manual systems being utilized, the whole strategy will as a rule be moderate.Customers additionally have no prior information on cost of work their pieces of attire.

3. IDEATION AND PROPOSED SOLUTION:

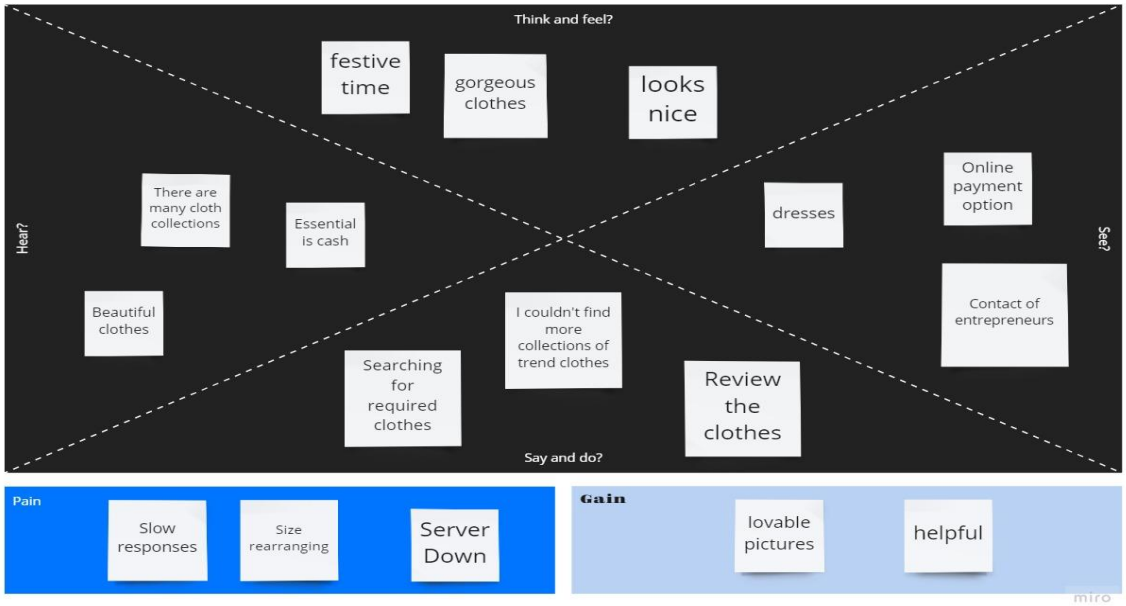
3.1 EMPATHY MAP CAVAS:

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviours and attitudes.

It is a useful tool to helps teams better understand their users. Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.

Example:

Entrepreneur Tailoring app (buying a clothes)



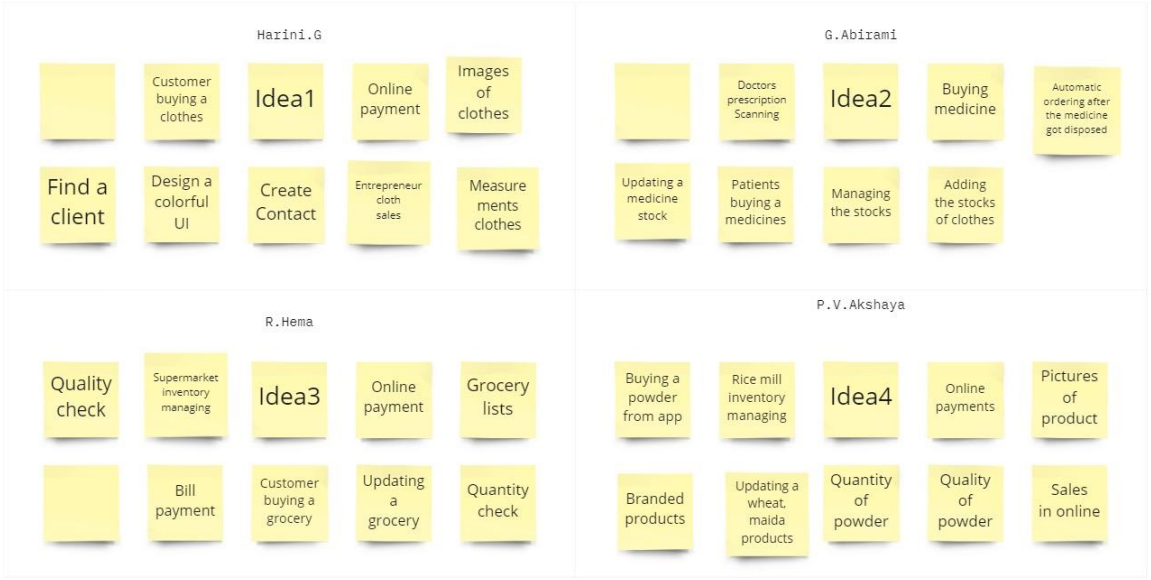
3.2 IDEATION AND BRAINSTORMING:

Step-1: Team Gathering, Collaboration and Select the Problem Statement

Problem Statement:

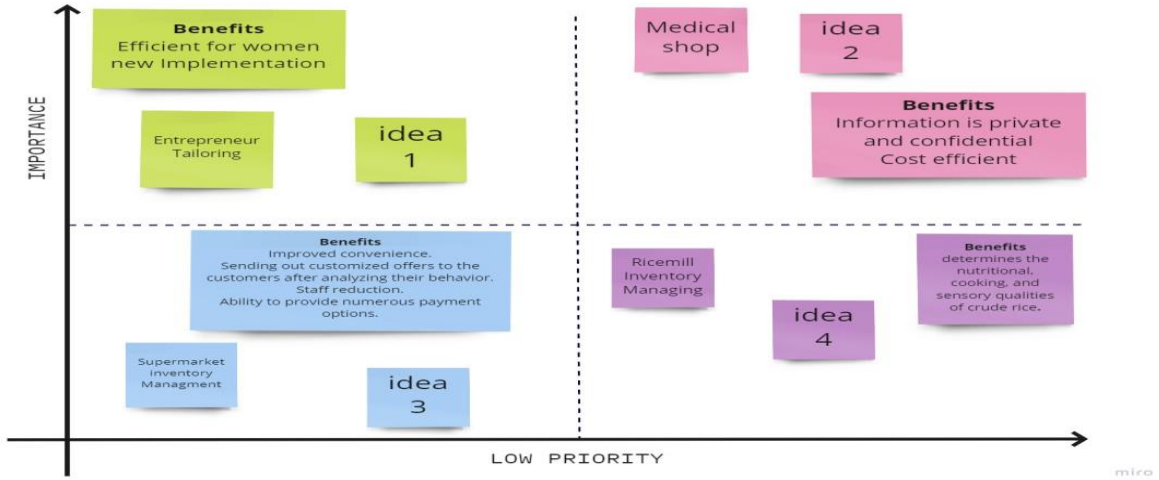
The problem faced by the company is they do not have any systematic system to record and keep their inventory data. It is difficult for the admin to record the inventory data quickly and safely because they only keep it in the logbook and not properly organized.

Step-2: Brainstorm, Idea Listing and Grouping



miro

Step-3: Idea Prioritization



IDEA PRIORITIZATION:

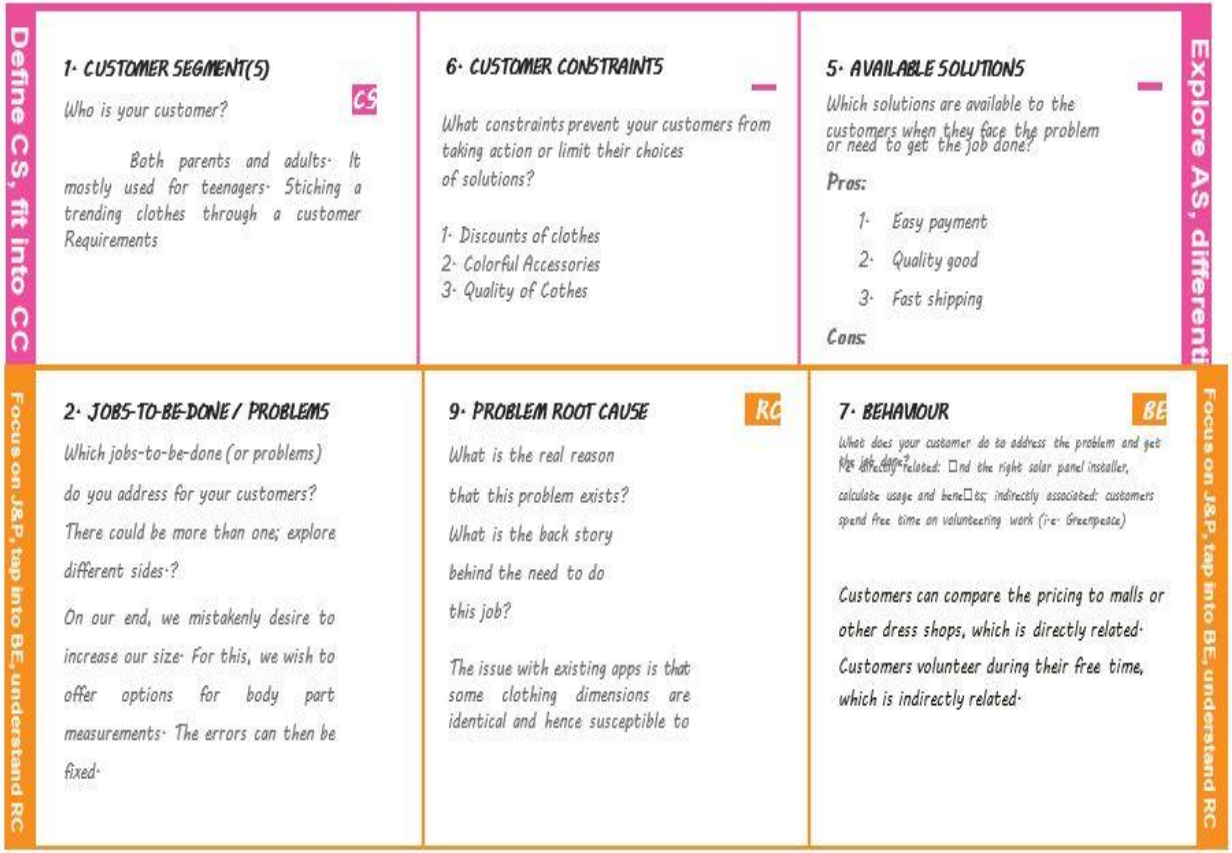
Entrepreneur Tailoring

1. Good Convenience
2. Efficient for women
3. Customers increment

3.3 PROPOSED SOLUTION:

I am (Customer)	**I'm trying to**	**But**	**Because**	**Which makes me feel**
Consumer	Buy a thing	It takes a long time	The website is not responsive and it takes longer time to load	Frustrated
Consumer	Buy a Product	I cannot click the buy button	If I click the button it shows	Frustrated

3.4PROBLEM SOLUTION FIT:



4.REQUIREMENT ANALYSIS:

4.1 FUNCTIONAL REQUIREMENTS:

- Capture customer information, store it and make it open at the crucial point in time.
- Present the customers with a constant introduction on the bits of dress status.
- Generate reports exactly and fortunate.

- Search and show customer information nuances.
- Computes the total cost of a bit of apparel depending upon the picked surface, kind of material, sum and term and benefits that information to the customer.

4.2 NON-FUNCTIONAL REQUIREMENTS:

- The system has first class and faithful quality level.
- The break between frustrations,mean time to fix, and precision are high.
- The system has straightforward interfaces.
- This ensures the straight forwardness with which the structure can be learned or used.
- The structure can empower customers to present and work it with for all intents and purposes no arrangement.
- Handles creating proportions of work in a spry manner as can be expeditiously widened for instance the straightforwardness, with which the system can be changed to manage a gigantic augmentation in customers, exceptional main job or trades.
- The system prevents unapproved access to the structure with customer check through login-on structure.

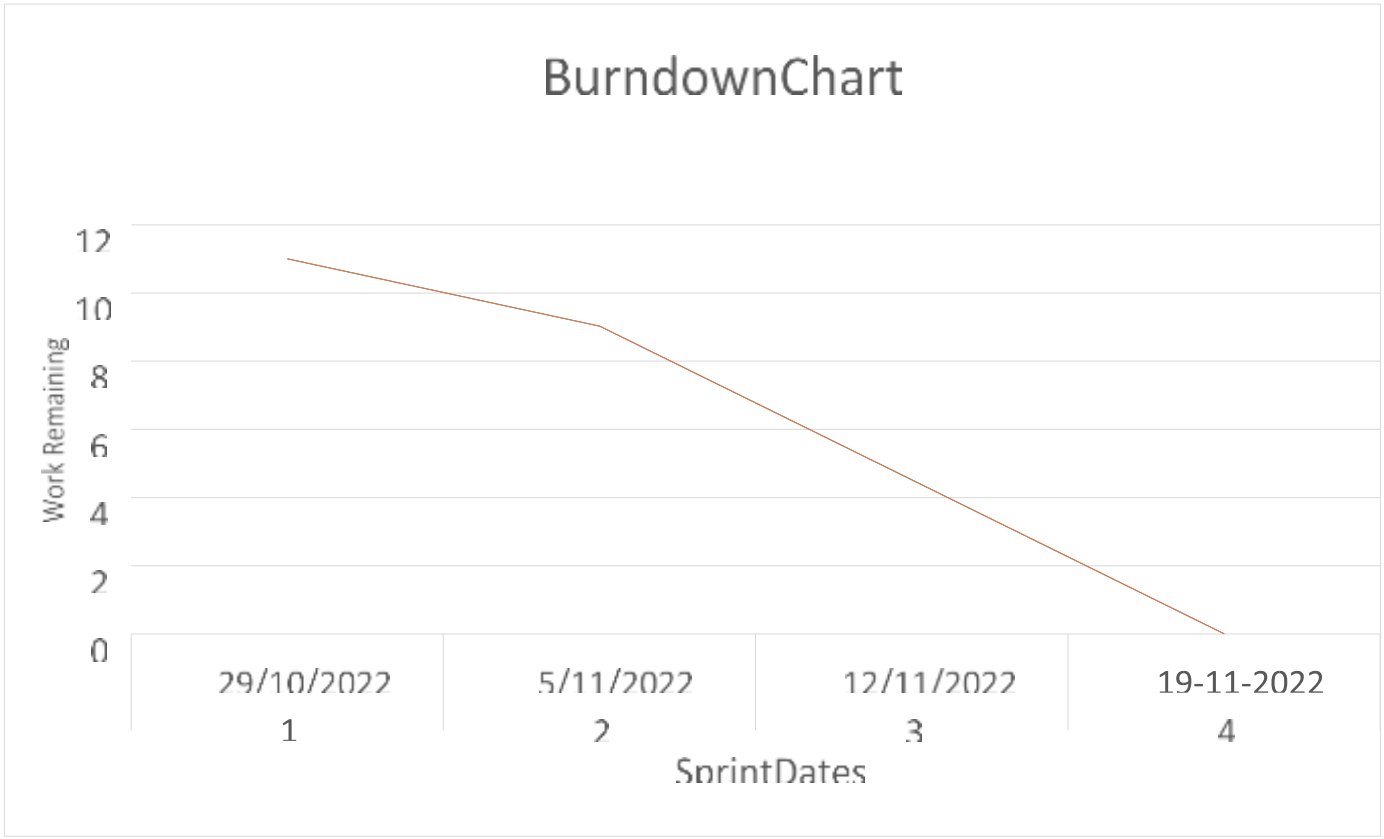
6. PROJECT PLANNING AND SCHEDULING

6.1 Sprint Planning and Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	2	High	1
Sprint-1	Confirmation mail sending	USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	1
Sprint-2	Creating Account	USN-3	As a user, I can register for the application through Facebook	2	Low	2
Sprint-1	Homepage	USN-4	As a user, I can register for the application through Gmail	2	Medium	3
Sprint-1	Login and Dashboard	USN-5	As a user, I can log into the application by entering email & password	1	High	4

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

$$AV = \frac{sprint\ duration}{velocity} = \frac{20}{10} = 2$$

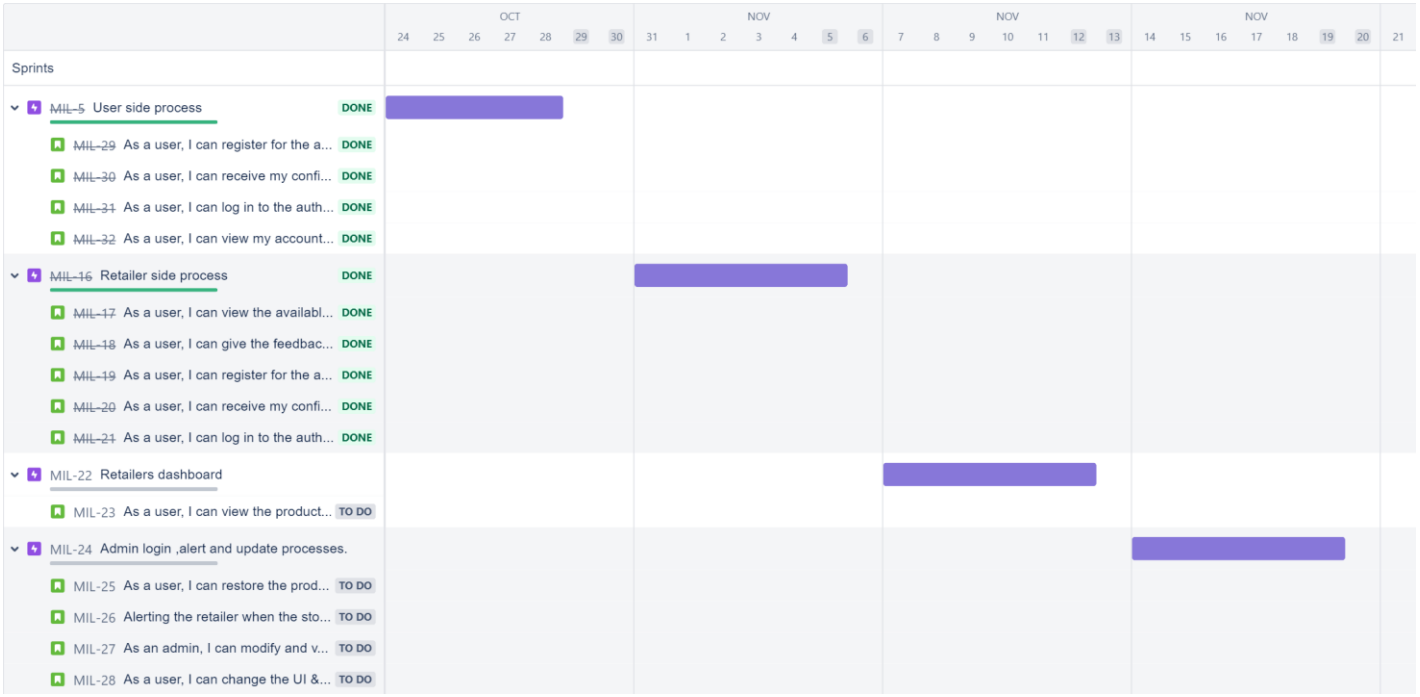

BurndownChart
Work Remaining
12
10
8
6
4
2
0
29/10/2022
5/11/2022
12/11/2022
19-11-2022
1
2
3
4
SprintDates

6.2 SPRINT DELIVERY AND SCHEDULE

TITLE	DESCRIPTION	DATE
Literature Survey & Information Gathering	Literature survey on the most relevant 5 papers have been made and the information gathered have been submitted.	17 SEPTEMBER 2022
Prepare Empathy Map	Prepare an empathy map canvas to capture the user's pains & gains and almost 4 WH questions have been answered and prepare the list of problem statements.	17 SEPTEMBER 2022
Ideation	To list by organizing brainstorming sessions And prioritize the top three ideas based on feasibility and importance.	17 SEPTEMBER 2022
Proposed Solution	To prepare the proposed solution documents, which includes the novelty, feasibility of ideas, business model, social impact, scalability of the solution, etc.	24 SEPTEMBER 2022

Problem Solution Fit	Preparing the problems Solution fit document.	25 SEPTEMBER 2022
Development Phase	Preparing the documentation and source code	13 NOVEMBER 2022

6.3 Reports from JIRA



7. Coding Solutions

```
def home():
  if not session.get("name"):
        return render_template('home.html')
  return render_template('home.html', session = session)

@app.route('/register')
def new_student():
  return render_template('Register.html')


@app.route('/addrec',methods = ['POST', 'GET'])
def addrec():
  if request.method == 'POST':
    fname = request.form['fname']
    lname = request.form['lname']
    cname = request.form['cname']
    state = request.form['state']
    city = request.form['city']
    mobileno = request.form['mobileno']
    emailid = request.form['emailid']
    password = request.form['password']
    pincode = request.form['pincode']

    sql = "SELECT * FROM Users WHERE EMAILID =?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,1,emailid)
    ibm_db.execute(stmt)
    account = ibm_db.fetch_assoc(stmt)
```

```
if account:
  users = []
  sql = "SELECT * FROM Users"
  stmt = ibm_db.exec_immediate(conn, sql)
  dictionary = ibm_db.fetch_both(stmt)
  while dictionary != False:
    # print ("The Name is : ",  dictionary)
    users.append(dictionary)
    dictionary = ibm_db.fetch_both(stmt)
  return render_template('list.html', msg="You are already a member, please login using your details", users = users)
else:

  var_list.append(fname)
  var_list.append(lname)
  var_list.append(cname)
  var_list.append(state)
  var_list.append(city)
  var_list.append(mobileno)
  var_list.append(emailid)
  var_list.append(password)
  var_list.append(pincode)

  bodytemp = r'D:\IBM\GUIDED PROJECT\INVENTORY MANAGEMENT SYSTEM FOR RETAILERS\templates\email.html'
  with open(bodytemp, "r", encoding='utf-8') as f:
      html= f.read()


  # Set up the email addresses and password. Please replace below with your email address and password
  email_from = 'padhu10a@gmail.com'
  epassword = 'rbjibzkssszsbrjo'
  email_to = emailid
```

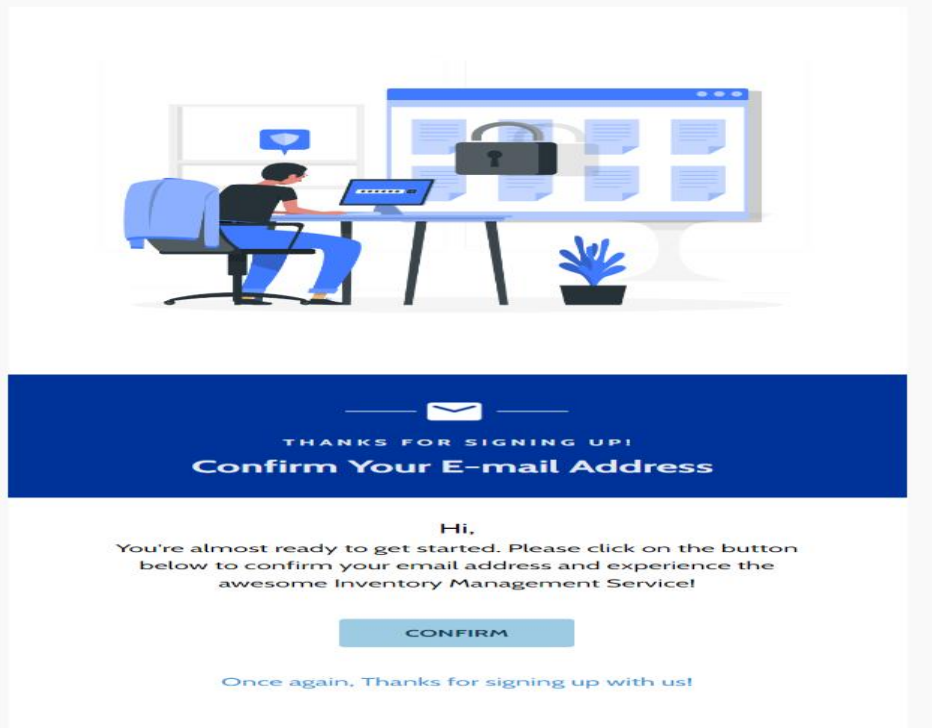
```
    # Generate today's date to be included in the email Subject
    date_str = pd.Timestamp.today().strftime('%Y-%m-%d')

    # Create a MIMEMultipart class, and set up the From, To, Subject fields
    email_message = MIMEMultipart()
    email_message['From'] = email_from
    email_message['To'] = email_to
    email_message['Subject'] = f'Report email - {date_str}'

    # Attach the html doc defined earlier, as a MIMEText html content type to the MIME message
    email_message.attach(MIMEText(html, "html"))
    # Convert it as a string
    email_string = email_message.as_string()

    # Connect to the Gmail SMTP server and Send Email
    context = ssl.create_default_context()
    with smtplib.SMTP_SSL("smtp.gmail.com", 465, context=context) as server:
        server.login(email_from, epassword)
        server.sendmail(email_from, email_to, email_string)
    return render_template('notify.html')

@app.route('/confirm')
def confirnation():
  insert_sql = "INSERT INTO Users (FIRSTNAME, LASTNAME, COMPANYNAME, STATE, CITY, MOBILENO, EMAILID, PASSWORD,
PINCODE)  VALUES (?,?,?,?,?,?,?,?,?)"
  prep_stmt = ibm_db.prepare(conn, insert_sql)
  ibm_db.bind_param(prep_stmt, 1, var_list[0])
  ibm_db.bind_param(prep_stmt, 2, var_list[1])
  ibm_db.bind_param(prep_stmt, 3, var_list[2])
  ibm_db.bind_param(prep_stmt, 4, var_list[3])
  ibm_db.bind_param(prep_stmt, 5, var_list[4])
  ibm_db.bind_param(prep_stmt, 6, var_list[5])
```

```
    ibm_db.bind_param(prep_stmt, 7, var_list[6])
    ibm_db.bind_param(prep_stmt, 8, var_list[7])
    ibm_db.bind_param(prep_stmt, 9, var_list[8])
    ibm_db.execute(prep_stmt)
    return render_template('confirm.html')



@app.route('/login', methods =['POST', 'GET'])
def login():
    msg = ''
    if request.method == 'POST' and 'email' in request.form and 'password' in request.form:
        email = request.form['email']
        password = request.form['password']

        sql = "SELECT * FROM Users WHERE EMAILID =? AND PASSWORD =?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,email)
        ibm_db.bind_param(stmt,2,password)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)

        if account:
            session['loggedin'] = True
            session['id'] = account['ID']
            session['email'] = account['EMAILID']
            session['name'] = account['FIRSTNAME']
            msg = 'Logged in successfully !'
            return render_template('home.html', msg = msg)
        else:
            msg = 'Incorrect email / password !'
```

```
    return render_template('login.html', msg = msg)

@app.route('/logout')
def logout():
    session.pop('loggedin', None)
    session.pop('id', None)
    session.pop('email', None)
    session.pop('name', None)
    return redirect(url_for('home'))

@app.route('/list')
def list():
  users = []
  sql = "SELECT * FROM Users"
  stmt = ibm_db.exec_immediate(conn, sql)
  dictionary = ibm_db.fetch_both(stmt)
  while dictionary != False:
    # print ("The Name is : ",  dictionary)
    users.append(dictionary)
    dictionary = ibm_db.fetch_both(stmt)

  if users:
    return render_template("list.html", users = users , session = session)
```

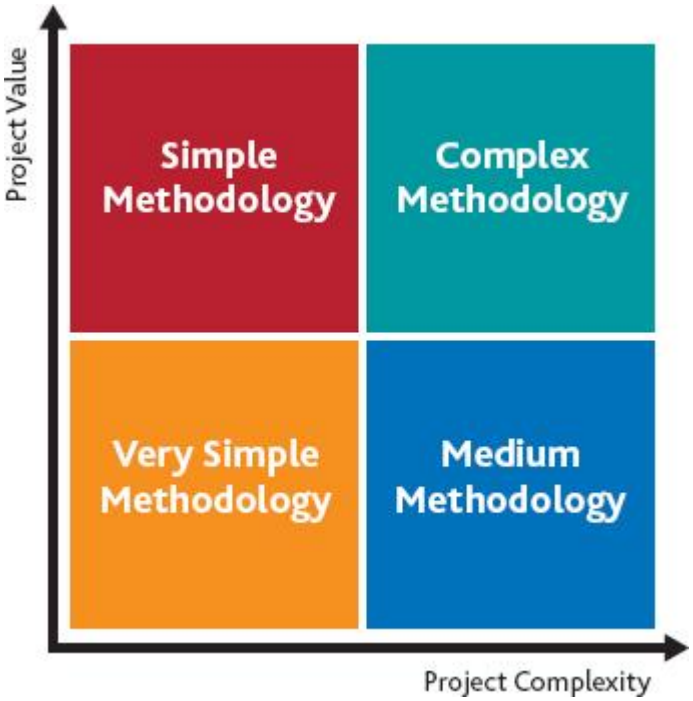

home
about
signin
signup
Search..
Saree blouse
Kurtis
Salwars
Men's suit
Shirts
Pants
Frock and gowns
Wedding designer

FEATURE 2



8. Results

Alterations make your garments fit you correctly, providing comfort, ease of movement and above all a flattering finish. Tailoring does not have to be a luxury reserved for only formal and business wear, and we believe there doesn't need to be a special occasion in order for you to want to look and feel special.



10. Advantages and Disadvantages

Advantages:

Alterations make your garments fit you correctly, providing comfort, ease of movement and above all a flattering finish. Tailoring does not have to be a luxury reserved for only formal and business wear, and we believe there doesn't need to be a special occasion in order for you to want to look and feel special.

Disadvantages:

1) **Sizes could be an issue**: Yes, at times you may like a dress, but have to drop it because of the reason that you do not have a size that fits you.
2) The proper fit could be an issue: It often happens that you buy a ready-made garment that you have liked, but unfortunately it does not fit you properly.

11. Conclusion

The way toward tailoring software development approach has been utilized since decades either intentionally or unwittingly, this procedure guarantees that procedures utilized in development are altered dependent on the particular task to be executed, the need to give systems that would fill its particular need without interference or disturbance is as yet a subject for conflict, reliability engineering as talked about in this examination paper is proposed as a powerful component towards quality assurance of software systems.

Most associations that are into software development buy in to the spry development approach and declaration which guarantees and proposes emphasess during development. Emphasess alone as talked about in this examination can't settle or relieve software issues totally in spite of the fact that the software development procedure can be upgraded dependent on a lithe statement which recommends the utilization of devices and methods to enhance quality during development just as quality assurance of systems.

12. Future Scope

The online Tailoring Management System will permit to enlist and pass on estimations to the tailor for the accompanying system to seek after. It in like manner keeps up clients' information and making various reports about the tailor shop. The central customers of the endeavor are clients and system Administrator. It furthermore engages customers to check the status of their bits of garments for instance in case arranged or not for collection. The structure gives information about the cost, the surface kind the customer need his/her dress weave from, the length a customer needs the dress finished, the kind of material to be used, sum the extent that sets required and specifically, the system enlists the hard and fast cost and benefits that information to the customer. Be that as it

may, online installment has not been achieved,but the client is relied upon topay either by means of versatile cash move administrations like m-pesa, pesapal or money when they come to pick their garments.

14. Appendix

```
<?php
if (file_get_contents('function.php') == '') {
	header('Location: './install/');
}
require_once('function.php');
session_start();

if (is_user()) {
	redirect('home.php');
}
?>


<!DOCTYPE html>
<html lang="en">
<head>

	<meta charset="utf-8">
	<meta http-equiv="X-UA-Compatible" content="IE=edge">
	<meta name="viewport" content="width=device-width, initial-scale=1">
	<title>Sign In</title>

	<!-- CSS -->
	<link rel="stylesheet" href="http://fonts.googleapis.com/css?family=Roboto:400,100,300,500">
	<link rel="stylesheet" href="assets/bootstrap/css/bootstrap.min.css">
	<link rel="stylesheet" href="assets/font-awesome/css/font-awesome.min.css">
```

```
                <link rel="stylesheet" href="assets/css/form-elements.css">
        <link rel="stylesheet" href="assets/css/style.css">
<style>

.pass {
width: 100%;
height: 50px;
}
</style>
        <!-- HTML5 Shim and Respond.js IE8 support of HTML5 elements and media queries -->
        <!-- WARNING: Respond.js doesn't work if you view the page via file:// -->
        <!--[if lt IE 9]>
            <script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>
            <script src="https://oss.maxcdn.com/libs/respond.js/1.4.2/respond.min.js"></script>
        <![endif]-->


    </head>

    <body>

                <!-- Top menu -->

        <!-- Top content -->
        <div class="top-content">

            <div class="inner-bg">
                <div class="container">
                    <div class="row">
                        <div class="col-sm-8 col-sm-offset-2 text">
                            <h1><strong>ADMIN </strong> Login</h1>
                        </div>
```

```
</div>
<div class="row">

   <div class="col-md-6 col-md-offset-3 form-box">
          <div class="form-top">
                 <div class="form-top-left">
                        <h3>Sign In</h3>
                 <p>Fill in the form below to get instant access:</p>
                 </div>
                 <div class="form-top-right">
                        <i class="fa fa-user"></i>
                 </div>
     </div>
     <div class="form-bottom">



                                    <?php if (!empty($_GET['error'])): ?>
              <div class="alert alert-danger alert-dismissable">
          <button type="button" class="close" data-dismiss="alert" aria-hidden="true">&times;</button>
                                             <?php echo $_GET['error']?>
          </div>
         <?php endif ?>


                                                 <form role="form" action="signin_post.php" method="post" class="registration-form">

<div class="form-group">
<input type="text" name="username" value="admin" class="form-first-name form-control">
</div>
<div class="form-group">
<input type="password" name="password" value="admin" class="pass form-control">
```

```
</div>

                                        <button type="submit" class="btn"> Submit</button>
                                    </form>
                                </div>
                            </div>
                        </div>
                    </div>
                </div>

            </div>


            <!-- Javascript -->
            <script src="assets/js/jquery-1.11.1.min.js"></script>
            <script src="assets/bootstrap/js/bootstrap.min.js"></script>
            <script src="assets/js/jquery.backstretch.min.js"></script>
            <script src="assets/js/retina-1.1.0.min.js"></script>
            <script src="assets/js/scripts.js"></script>

            <!--[if lt IE 10]>
                <script src="assets/js/placeholder.js"></script>
            <![endif]-->

        </body>
</html>

from turtle import st
from flask import Flask, render_template, request, redirect, url_for, session
from markupsafe import escape

import ibm_db
```

```
conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32733;SECURITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=lkc93724;PWD=zAzNGa6DaNk6xvle","","")

import smtplib, ssl
## email.mime subclasses
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
## The pandas library is only for generating the current date, which is not necessary for sending emails
import pandas as pd

app = Flask(__name__)

var_list = []
app.secret_key = 'your secret key'

@app.route('/')
def home():
  if not session.get("name"):
        return render_template('home.html')
  return render_template('home.html', session = session)

@app.route('/register')
def new_student():
  return render_template('Register.html')


@app.route('/addrec',methods = ['POST', 'GET'])
def addrec():
  if request.method == 'POST':
    fname = request.form['fname']
    lname = request.form['lname']
```

```
cname = request.form['cname']
state = request.form['state']
city = request.form['city']
mobileno = request.form['mobileno']
emailid = request.form['emailid']
password = request.form['password']
pincode = request.form['pincode']

sql = "SELECT * FROM Users WHERE EMAILID =?"
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt,1,emailid)
ibm_db.execute(stmt)
account = ibm_db.fetch_assoc(stmt)

if account:
  users = []
  sql = "SELECT * FROM Users"
  stmt = ibm_db.exec_immediate(conn, sql)
  dictionary = ibm_db.fetch_both(stmt)
  while dictionary != False:
    # print ("The Name is : ",  dictionary)
    users.append(dictionary)
    dictionary = ibm_db.fetch_both(stmt)
  return render_template('list.html', msg="You are already a member, please login using your details", users = users)
else:

  var_list.append(fname)
  var_list.append(lname)
  var_list.append(cname)
  var_list.append(state)
  var_list.append(city)
  var_list.append(mobileno)
```

```
var_list.append(emailid)
var_list.append(password)
var_list.append(pincode)

bodytemp = r'D:\IBM\GUIDED PROJECT\INVENTORY MANAGEMENT SYSTEM FOR RETAILERS\templates\email.html'
with open(bodytemp, "r", encoding='utf-8') as f:
    html= f.read()


# Set up the email addresses and password. Please replace below with your email address and password
email_from = 'padhu10a@gmail.com'
epassword = 'rbjibzkssszsbrjo'
email_to = emailid

# Generate today's date to be included in the email Subject
date_str = pd.Timestamp.today().strftime('%Y-%m-%d')

# Create a MIMEMultipart class, and set up the From, To, Subject fields
email_message = MIMEMultipart()
email_message['From'] = email_from
email_message['To'] = email_to
email_message['Subject'] = f'Report email - {date_str}'

# Attach the html doc defined earlier, as a MIMEText html content type to the MIME message
email_message.attach(MIMEText(html, "html"))
# Convert it as a string
email_string = email_message.as_string()

# Connect to the Gmail SMTP server and Send Email
context = ssl.create_default_context()
with smtplib.SMTP_SSL("smtp.gmail.com", 465, context=context) as server:
    server.login(email_from, epassword)
```

```
        server.sendmail(email_from, email_to, email_string)
    return render_template('notify.html')

@app.route('/confirm')
def confirnation():
  insert_sql = "INSERT INTO Users (FIRSTNAME, LASTNAME, COMPANYNAME, STATE, CITY, MOBILENO, EMAILID, PASSWORD,
PINCODE)  VALUES (?,?,?,?,?,?,?,?,?)"
  prep_stmt = ibm_db.prepare(conn, insert_sql)
  ibm_db.bind_param(prep_stmt, 1, var_list[0])
  ibm_db.bind_param(prep_stmt, 2, var_list[1])
  ibm_db.bind_param(prep_stmt, 3, var_list[2])
  ibm_db.bind_param(prep_stmt, 4, var_list[3])
  ibm_db.bind_param(prep_stmt, 5, var_list[4])
  ibm_db.bind_param(prep_stmt, 6, var_list[5])
  ibm_db.bind_param(prep_stmt, 7, var_list[6])
  ibm_db.bind_param(prep_stmt, 8, var_list[7])
  ibm_db.bind_param(prep_stmt, 9, var_list[8])
  ibm_db.execute(prep_stmt)
  return render_template('confirm.html')



@app.route('/login', methods =['POST', 'GET'])
def login():
    msg = ''
    if request.method == 'POST' and 'email' in request.form and 'password' in request.form:
        email = request.form['email']
        password = request.form['password']

        sql = "SELECT * FROM Users WHERE EMAILID =? AND PASSWORD =?"
        stmt = ibm_db.prepare(conn, sql)
```

```
        ibm_db.bind_param(stmt,1,email)
        ibm_db.bind_param(stmt,2,password)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)

        if account:
            session['loggedin'] = True
            session['id'] = account['ID']
            session['email'] = account['EMAILID']
            session['name'] = account['FIRSTNAME']
            msg = 'Logged in successfully !'
            return render_template('home.html', msg = msg)
        else:
            msg = 'Incorrect email / password !'
    return render_template('login.html', msg = msg)

@app.route('/logout')
def logout():
    session.pop('loggedin', None)
    session.pop('id', None)
    session.pop('email', None)
    session.pop('name', None)
    return redirect(url_for('home'))

@app.route('/list')
def list():
  users = []
  sql = "SELECT * FROM Users"
  stmt = ibm_db.exec_immediate(conn, sql)
  dictionary = ibm_db.fetch_both(stmt)
  while dictionary != False:
    # print ("The Name is : ",  dictionary)
```

```
  users.append(dictionary)
  dictionary = ibm_db.fetch_both(stmt)

if users:
  return render_template("list.html", users = users , session = session)



return "No users..."
```