

**PROJECT REPORT**

<b>Team ID</b>	<b>TEAM ID - PNT2022TMID15450</b>
<b>Project Name</b>	<b>INVENTORY MANAGEMENT SYSTEM FOR RETAILERS</b>

**1. INTRODUCTION****1.1 Project Overview:**

Retail inventory management is the process of ensuring you carry merchandise that shoppers want, with neither too little nor too much on hand. By managing inventory, retailers meet customer demand without running out of stock or carrying excess supply. In practice, effective retail inventory management results in lower costs and a better understanding of sales patterns. Retail inventory management tools and methods give retailers more information on which to run their businesses. Applications have been developed to help retailers track and manage stocks related to their own products. The System will ask retailers to create their accounts by providing essential details. Retailers can access their accounts by logging into the application. In today's more turbulent environment there is no longer any possibility of manufacturing and marketing acting independently of each other. It is now generally accepted that the need to understand and meet customer requirements is a prerequisite for survival. At the same time, in the search for improved cost competitiveness, manufacturing management has been the subject of massive renaissance. The last decade has seen the rapid introduction of flexible manufacturing systems, of new approaches to inventory based on materials requirement planning (MRP) and just in time (JIT) methods, a sustained emphasis on quality. Equally there has been a growing recognition of the critical role that procurement plays in creating and sustaining competitive advantage as part of an integrated logistics process. In this scheme of things, logistics is therefore essentially an integrative concept that seeks to develop a system wide view of the firm. It is fundamentally a planning concept that seeks to create a framework through which the needs of the manufacturing strategy and plan, which in turn link into a strategy and plan for procurement.

**1.2 Purpose:**

Once retailers successfully log in to the application they can update their inventory details, also users will be able to add new stock by submitting essential details related to the stock. They can view details of the current inventory. The System will automatically send an email alert to the retailers if there is no stock found in their accounts. So that they can order new stock. In the industries there will be a competitor who will be a low cost producer and will have greater sales volume in that sector. This is partly due to economies of scale, which enable fixed costs to spread over a greater volume but more particularly to the impact of the experience curve. It is possible to identify and predict improvements in the rate of output of workers as they become more skilled in the processes and tasks on which they work. Bruce Henderson extended this concept by demonstrating that all costs, not just production costs, would decline at a given rate as volume increased. This cost decline applies only to value added, i.e. costs other than bought in supplies. Traditionally it has been suggested that the main route to cost reduction was by gaining greater sales volume and there can be no doubt about the close linkage between relative market share and relative costs. However it must also be recognized that logistics management can provide a multitude of ways to increase efficiency and productivity and hence contribute significantly to reduced unit costs.



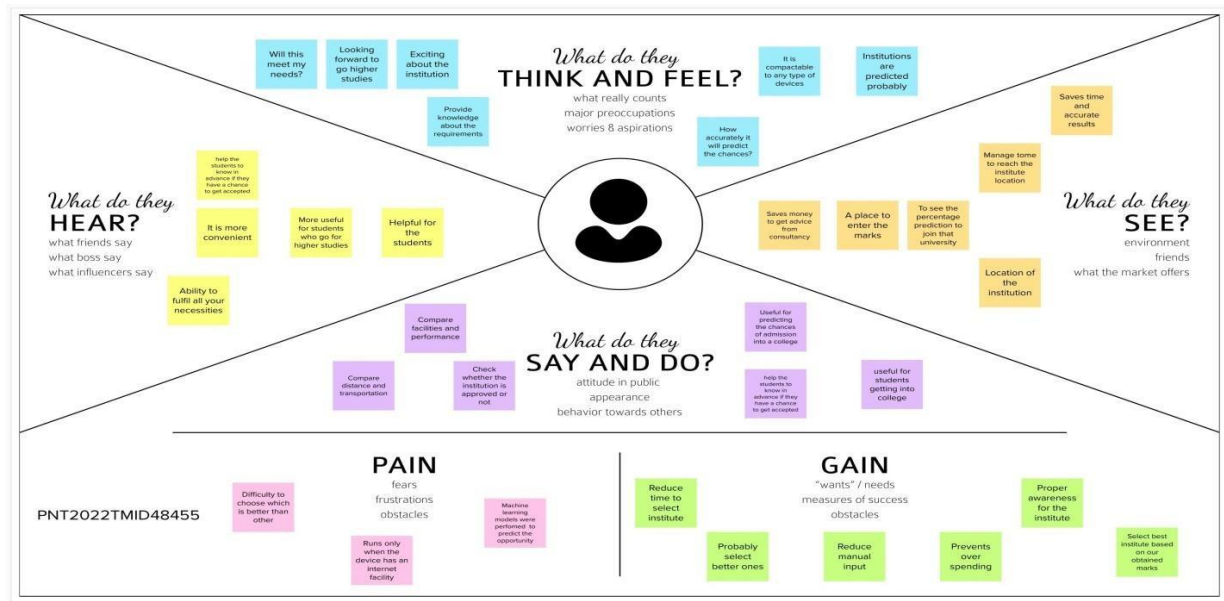


### 1.3 Problem Statement Definition

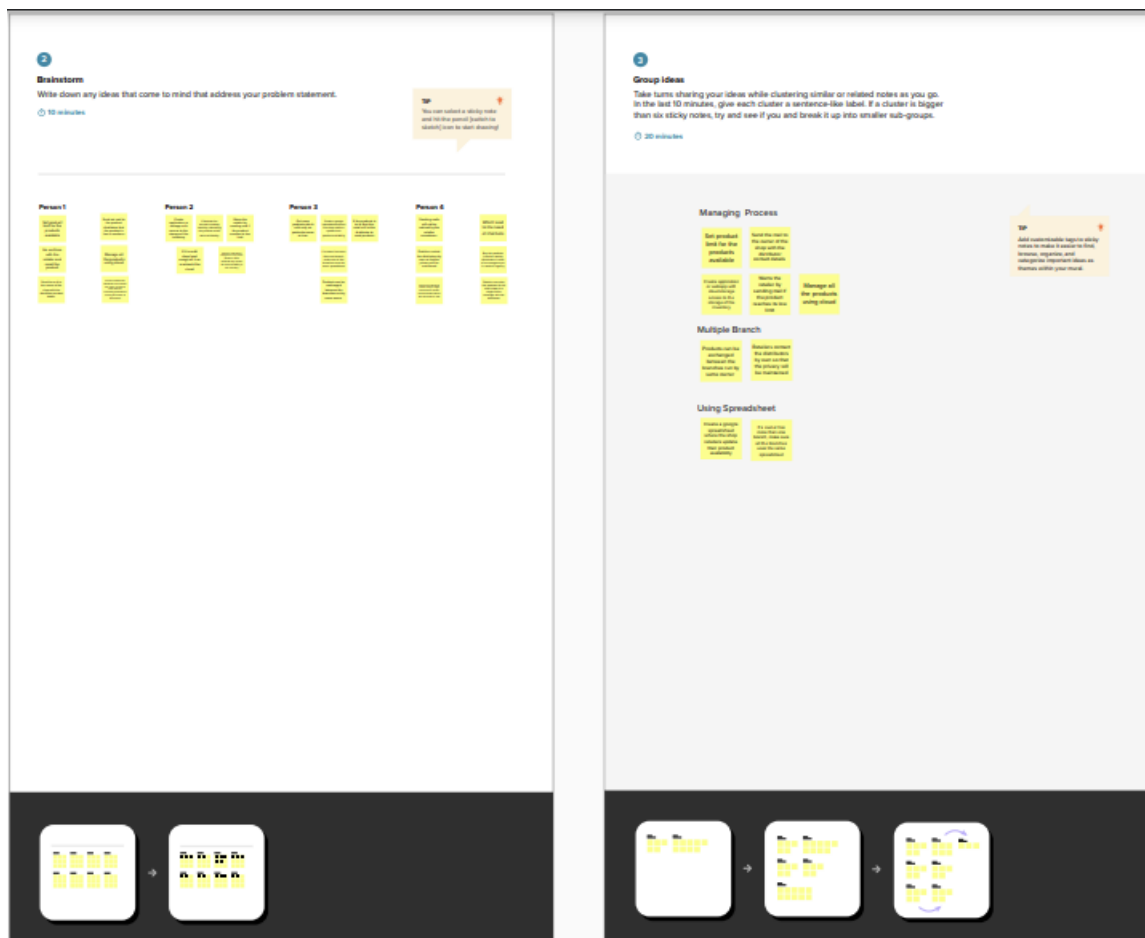
<b>Problem Statement (PS)</b>	<b>I am (Customer)</b>	<b>I'm trying to</b>	<b>But</b>	<b>Because</b>	<b>Which makes me feel</b>
PS-1	I am a rice shop owner who keeps his stock updated	Update the current stock in my inventory according to the customer's purchases	I 'm unable to upload and update the data	The server is very bad due to the network in remote areas	Frustrated and unable to update the products which users needed.
PS-II	I am Murugesan, who is a customer to a shop.	Update the products which are not available at the store.	the data has been entered wrongly	heavy network traffic to update	to go for alternative decisions like searching on another shop, leave the product out etc.

## 2.IDEATION & PROPOSED SOLUTION

## 2.1 Empathy Map Canvas



## 2.2 Ideation & Brainstorming



4

**Prioritize**

Your team should all be on the same page about what's important moving forward. Place your ideas on the grid to determine which ideas are important and which are feasible.

20 minutes



5

**After you collaborate**

You can export the mural as an image or pdf to share with members of your company who might find it helpful.

**Quick add-ons****Share the mural**

Share a share link to the mural with stakeholders to keep them in the loop about the outcomes of the session.

**Export the mural**

Export a copy of the mural as a PNG or PDF to attach to emails, include in slides, or save to your drive.

**Keep moving forward****Strategy Blueprint**

Define the components of a new idea or strategy.

[Open the template](#)

**Customer experience journey map**

Understand customer needs, motivations, and behaviors for an experience.

[Open the template](#)

**Strengths, weaknesses, opportunities & threats**

Identify strengths, weaknesses, opportunities, and threats (SWOT) to develop a plan.

[Open the template](#)

**Share template feedback**

### 2.3 Proposed Solution

S. No.	Parameter	Description
1	Problem Statement	To solve the need that the shopkeepers doesn't have the systematic way to keep their record of inventory data.
2	Idea / Proposed Solution	An application which retailers successfully log in to the application, that they can update their inventory details, also users will be able to add new stock by submitting essential details related to the stock. They can view details of the current inventory. The System will automatically send an email alert to the retailers, if the stock reduced to the limited amount found in the inventory. So that they can order new stock.
3	Novelty / Uniqueness	With this inventory management system, the shopkeeper not only can fill the inventory but also reduce the wastage of goods. The users can register the stocks that they need by logging in from their account.
4	Social Impact / Customer Satisfaction	Customer Satisfaction is entirely depend on the services which they expected. If the retailer's system exceeds with customer's expectation, the customers will be satisfied.
5	Business Model	With the better inventory management system, Update the inventory without any need of manpower. Retailer can live up with user's need and be on the flow with current sale products and they can update the inventory with that products.
6	Scalability of the Solution	To create a scalable inventory management system, the retailer have to <ol style="list-style-type: none"> <li>1. Keeping low inventory levels as much as possible</li> <li>2. Keep an eye on Sales Projections</li> <li>3. Use ODM (On-Demand Manufacturing). ODM refers to manufacture or in this case, update the products which are highly in demand.</li> </ol>

## 2.4 Problem Solution fit

Define CS, fit into	<b>1. CUSTOMER SEGMENT(S)</b> <b>CS</b> <p>Customer segmentation is an important marketing tool.</p> <p>Effective customer segmentation helps the enterprises increase profits and improve customer service level.</p> <p>On the other hand, due to possible detrimental consequences, supply disruptions have been receiving more and more attention.</p>	<b>6. CUSTOMER CONSTRAINTS</b> <b>CC</b> <p>limits on raw materials, machine capacity, workforce capacity, inventory investment, storage space, or the total number of orders placed.</p>	<b>5. AVAILABLE SOLUTIONS</b> <b>AS</b> <ul style="list-style-type: none"> <li>Lift per store sales by 5%</li> <li>70% time saved in store audits</li> <li>&gt;97% accurate retail insights in real-time</li> <li>Improved adherence to store compliance</li> <li>Brand &amp; SKU level competitive</li> </ul>	Focus on J&P, tap into BE, understand
	<b>2. JOBS-TO-BE-DONE / PROBLEMS</b> <b>J&amp;P</b> <ul style="list-style-type: none"> <li>Inconsistent Tracking</li> <li>Warehouse Efficiency</li> <li>Inaccurate Data</li> <li>Changing Demand</li> <li>Limited Visibility</li> <li>Manual Documentation</li> <li>Problem Stock</li> <li>Supply Chain Complexity</li> </ul>	<b>9. PROBLEM ROOT CAUSE</b> <b>RC</b> <ul style="list-style-type: none"> <li>Network issue</li> <li>Server down</li> <li>Data loss</li> </ul>	<b>7. BEHAVIOUR</b> <b>BE</b> <ul style="list-style-type: none"> <li>The data will be secure.</li> <li>Check the stock regularly.</li> <li>The process will be on time.</li> </ul>	
Identify	<b>3. TRIGGERS</b> <b>TR</b> <p>Retail inventory management techniques help stores and ecommerce sellers satisfy customers, reduce costs and increase profits.</p>	<b>10. YOUR SOLUTION</b> <b>SL</b> <ul style="list-style-type: none"> <li>Create a System to Get Accurate and Accessible Information</li> </ul>	<b>8. CHANNELS of BEHAVIOUR</b> <b>CH</b> <p>Online:</p> <ul style="list-style-type: none"> <li>Stock update</li> </ul>	Extract online

<b>4. EMOTIONS: BEFORE / AFTER</b> <p><b>Before:</b></p> <p>Over work and stock unavailable</p> <p><b>After:</b></p> <p>Easy work</p> <b>EM</b>	<ul style="list-style-type: none"> <li>Create a Unique Process Customized for Business Type.</li> <li>Keep an eye on Contemporary trends in the industry.</li> <li>Be prepared for fluctuations in supply and demand.</li> </ul>	<ul style="list-style-type: none"> <li>Stock needed</li> <li>Maintaining the stock above the warning level</li> <li>Calculating the current stock by using the billing info</li> </ul>
--	--	--

### Solution Architecture



### 3.REQUIREMENT ANALYSIS

#### 3.1 Functional requirement

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement(Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Email
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Login	Log into the application by entering the Email and Password
FR-4	Dashboard	View the products availability
FR-5	Add items to cart	Users they wish to buy products, they can add it to the cart.
FR-6	Stock Update	If the desired product is unavailable, they can update the products into the list for buying products.

#### 3.2 Non-Functional requirements

Following are the non-functional requirements of the proposed solution.

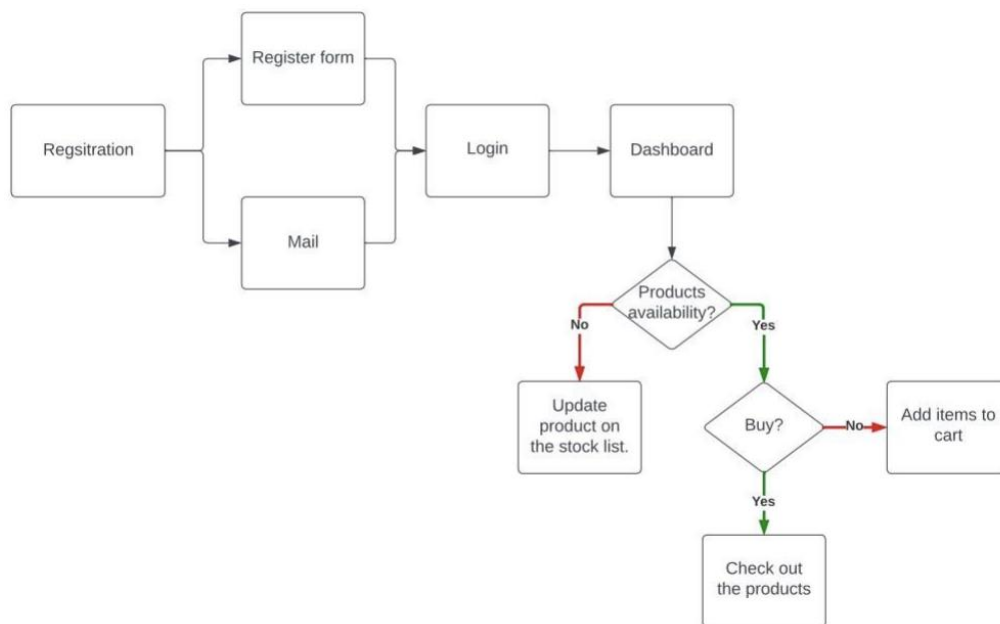
FR No.	Non-Functional Requirement	Description
NFR-1	Usability	While usability determines how effective implementing an inventory tracking system is in your business. If it takes hours for your staff to learn the ins and outs of the software, then it's probably not worth buying.
NFR-2	Security	The process of ensuring the safety and optimum management control of stored goods. It is of central importance for optimum warehouse management because the performance of a company stands or falls with the safety and efficiency of a warehouse.
NFR-3	Reliability	Relying on manual inventory counts to know what you have will only guarantee high inefficiencies and a loss of customers.

NFR-4	<b>Performance</b>	Creating systems to log products, receive them into inventory, track changes when sales occur, manage the flow of goods from purchasing to final sale and check stock counts.
NFR-5	<b>Availability</b>	Whether a specific item is available for customer orders. Additional information provided by retailers may include the quantity available.
NFR-6	<b>Scalability</b>	They should use an automated inventory management system for inventory tracking. This will make your business much more scalable so that you can continue building consistent growth and take advantage of increased sales.

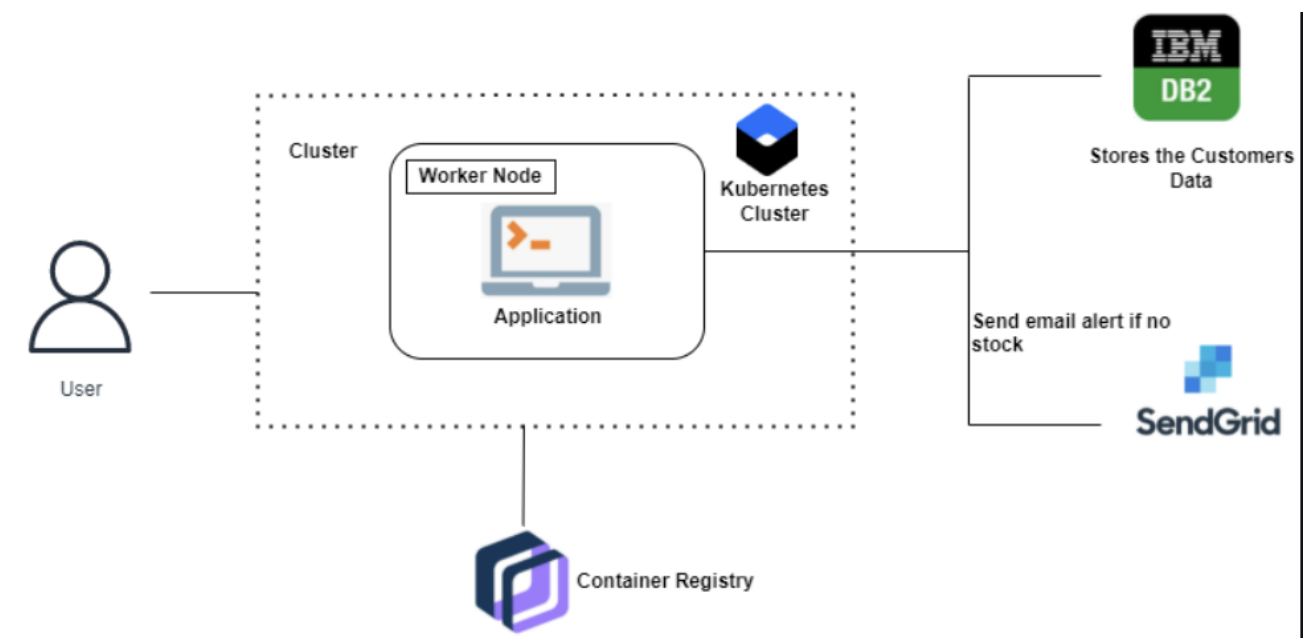
## 4.PROJECT DESIGN

### 4.1 Data Flow Diagrams

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.



### 4.2 Solution & Technical Architecture:



**Table-1:Components&Technologies:**

S.No	Component	Description	Technology
1.	User Interface	How user interacts with application e.g.Web UI, Mobile App, Chatbot etc.	HTML, CSS, JavaScript / AngularJs / React Js etc.
2.	Application Logic	Logic for a process in the application	Python-Flask
3.	Database	Data Type, Configurations etc.	MySQL, NoSQL, etc.
4.	Cloud Database	Database Service on Cloud	IBM DB2
5.	File Storage	File storage requirements	IBM Cloud Object Storage
6.	App Container	Contain the whole application in a singlecontainer	Docker Container / IBMContainer Registry
7.	Infrastructure (Server / Cloud)	Application Deployment on Local System /Cloud Local Server Configuration: Cloud Server Configuration:	Local, Cloud Foundry, Kubernetes, etc.
8.	Send Mails	Sending mails about stocks available in theInventory to the Retailer	SendGrid

**Table-2: Application Characteristics:**

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	List the open-source frameworks used	React Js / Flask
2.	Security Implementations	List all the security / access controlsimplemented, use of firewalls etc.	IBM Cloud Security
3.	Availability	Justify the availability of application (e.g., use of load balancers, distributedservers etc.)	Python-Flask

### 4.3 User Stories

Use the below template to list all the user stories for the product.

User Type	Functional Requirement(Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Custoer (Web user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I can register for the application through E-mail	I can access my account / dashboard	Medium	Sprint-1
	Confirmation	USN-3	As a user, I will receive confirmation email once I haveregistered for the application	I can get confirmation for myemail and passwordand create authent icated accoun t.	Medium	Sprint-1
	Login	USN-4	As a user, I can log into the application by entering email & password	I can log onto the application with verified email and password	High	Sprint-1
	Dashboard	USN-5	As a user, I can view the products which are available	Once I log on to the application, I can view products to buy.	High	Sprint-2

User Type	Functional Requirement(Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
	Add items tocart	USN-6	As a user, I can add the products Iwish to buy to the carts.	As a user, I can buyany products or addit to my cart for buying it later.	Medium	Sprint-2

	Stock Update	USN-7	As a user, I can add products which are not available in the dashboard to the stock list.	If any of the products which are not available, as a user I can update the inventory and send mail to the owner.	Medium	Sprint-3
Customer Care Executive	Request to Customer Care	USN-8	As a user, I can contact the Customer Care Executive and request any services I want from the customer care.	As a user, I can contact Customer Care and get support from them.	Low	Sprint-4
Administrator	Contact Administrator	USN-9	I can be able to report any difficulties I experience as a report	As user, I can give my support in my possible ways to administrator and the administration.	Medium	Sprint-4

## 5. PROJECT PLANNING & SCHEDULING

### 5.1 Sprint Planning & 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	vethanathan
Sprint-1		USN-2	As a user, I can register for theapplication through E-mail	1	Medium	Vignesh Shree Kishore
Sprint-1	Confirmation	USN-3	As a user, I will receive confirmation email once I have registered for theapplication	2	Medium	Srivatsan, Vethanathan

Sprint	Functional Requirement(Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Login	USN-4	As a user, I can log into the application by entering email & password	2	High	Srivatsan
Sprint-2	Dashboard	USN-5	As a user, I can view the productswhich are available	4	High	vethanathan
Sprint-2	Add items to cart	USN-6	As a user, I can add the products Iwish to buy to the carts.	5	Medium	Vethanathan
Sprint-3	Stock Update	USN-7	As a user, I can add products which are not available in the dashboard tothe stock list.	5	Medium	Vignesh Shree Kishore
Sprint-4	Request to Customer Care	USN-8	As a user, I can contact the Customer Care Executive and requestany services I want from the customer care.	5	Low	Vethanathan
Sprint-4	Contact Administrator	USN-9	I can be able to report any difficultiesI experience as a report	5	Medium	Shree kishore Srivatsan

## 6.2. Sprint Delivery Schedule

<b>Sprint</b>	<b>Total Story Points</b>	<b>Duration</b>	<b>Sprint StartDate</b>	<b>Sprint End Date(Planned)</b>	<b>Story Points Completed (ason Planned End Date)</b>	<b>Sprint Release Date(Actual)</b>
Sprint-1	7	6 Days	24 Oct 2022	29 Oct 2022	7	29 Oct 2022
Sprint-2	9	6 Days	31 Oct 2022	05 Nov 2022	9	05 Nov 2022
Sprint-3	5	6 Days	07 Nov 2022	12 Nov 2022	5	12 Nov 2022
Sprint-4	10	6 Days	14 Nov 2022	19 Nov 2022	10	19 Nov 2022



## 6. CODING & SOLUTIONING

### 6.1 Feature 1

```
from flask import Flask, render_template, flash, redirect, url_for, session, request, logging
from wtforms import Form, StringField, TextAreaField, PasswordField, validators, SelectField, IntegerField
import ibm_db
from passlib.hash import sha256_crypt
from functools import wraps

from sendgrid import *

#creating an app instance
app = Flask(__name__)

app.secret_key='a'

conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=31198;SECURITY=SSL;SSLS ererCertificate=DigiCertGlobalRootCA.crt;UID=ldy44383;PWD=iW50uSQCx3l7ckXh",",")

#Index
@app.route('/')
def index():
    return render_template('home.html')

#Products
@app.route('/products')
def products():
    sql = "SELECT * FROM products"
    stmt = ibm_db.prepare(conn, sql)
    result=ibm_db.execute(stmt)

    products=[]
    row = ibm_db.fetch_assoc(stmt)
    while(row):
        products.append(row)
        row = ibm_db.fetch_assoc(stmt)
    products=tuple(products)
    #print(products)

    if result>0:
        return render_template('products.html', products = products)
    else:
        msg='No products found'
        return render_template('products.html', msg=msg)

#Locations
@app.route('/locations')
def locations():

    sql = "SELECT * FROM locations"
    stmt = ibm_db.prepare(conn, sql)
    result=ibm_db.execute(stmt)
```

```

locations=[]
row = ibm_db.fetch_assoc(stmt)
while(row):
    locations.append(row)
    row = ibm_db.fetch_assoc(stmt)
locations=tuple(locations)
#print(locations)

if result>0:
    return render_template('locations.html', locations = locations)
else:
    msg='No locations found'
    return render_template('locations.html', msg=msg)

#Product Movements
@app.route('/product_movements')
def product_movements():

    sql = "SELECT * FROM productmovements"
    stmt = ibm_db.prepare(conn, sql)
    result=ibm_db.execute(stmt)

    movements=[]
    row = ibm_db.fetch_assoc(stmt)
    while(row):
        movements.append(row)
        row = ibm_db.fetch_assoc(stmt)
    movements=tuple(movements)
    #print(movements)

    if result>0:
        return render_template('product_movements.html', movements = movements)
    else:
        msg='No product movements found'
        return render_template('product_movements.html', msg=msg)

#Register Form Class
class RegisterForm(Form):
    name = StringField('Name', [validators.Length(min=1, max=50)])
    username = StringField('Username', [validators.Length(min=1, max=25)])
    email = StringField('Email', [validators.length(min=6, max=50)])
    password = PasswordField('Password', [
        validators.DataRequired(),
        validators.EqualTo('confirm', message='Passwords do not match')
    ])
    confirm = PasswordField('Confirm Password')

#user register
@app.route('/register', methods=['GET','POST'])
def register():
    form = RegisterForm(request.form)
    if request.method == 'POST' and form.validate():
        name = form.name.data
        email = form.email.data
        username = form.username.data
        password = sha256_crypt.encrypt(str(form.password.data))

```

```

sql1="INSERT INTO users(name, email, username, password) VALUES(?,?,?,?)"
stmt1 = ibm_db.prepare(conn, sql1)
ibm_db.bind_param(stmt1,1,name)
ibm_db.bind_param(stmt1,2,email)
ibm_db.bind_param(stmt1,3,username)
ibm_db.bind_param(stmt1,4,password)
ibm_db.execute(stmt1)
#for flash messages taking parameter and the category of message to be flashed
flash("You are now registered and can log in", "success")

#when registration is successful redirect to home
return redirect(url_for('login'))
return render_template('register.html', form = form)

#User login
@app.route('/login', methods = ['GET', 'POST'])
def login():
    if request.method == 'POST':
        #Get form fields
        username = request.form['username']
        password_candidate = request.form['password']

        sql1="Select * from users where username = ?"
        stmt1 = ibm_db.prepare(conn, sql1)
        ibm_db.bind_param(stmt1,1,username)
        result=ibm_db.execute(stmt1)
        d=ibm_db.fetch_assoc(stmt1)
        if result > 0:
            #Get the stored hash
            data = d
            password = data['PASSWORD']

            #compare passwords
            if sha256_crypt.verify(password_candidate, password):
                #Passed
                session['logged_in'] = True
                session['username'] = username

                flash("you are now logged in","success")
                return redirect(url_for('dashboard'))
            else:
                error = 'Invalid Login'
                return render_template('login.html', error=error)
            #Close connection
            cur.close()
        else:
            error = 'Username not found'
            return render_template('login.html', error=error)
    return render_template('login.html')

#check if user logged in
def is_logged_in(f):
    @wraps(f)
    def wrap(*args, **kwargs):

```

```

        if 'logged_in' in session:
            return f(*args, **kwargs)
        else:
            flash('Unauthorized, Please login','danger')
            return redirect(url_for('login'))
    return wrap

#Logout
@app.route('/logout')
@is_logged_in
def logout():
    session.clear()
    flash("You are now logged out", "success")
    return redirect(url_for('login'))

#Dashboard
@app.route('/dashboard')
@is_logged_in
def dashboard():
    sql2="SELECT product_id, location_id, qty FROM product_balance"
    sql3="SELECT location_id FROM locations"
    stmt2 = ibm_db.prepare(conn, sql2)
    stmt3 = ibm_db.prepare(conn, sql3)

    result=ibm_db.execute(stmt2)
    ibm_db.execute(stmt3)

    products=[]
    row = ibm_db.fetch_assoc(stmt2)
    while(row):
        products.append(row)
        row = ibm_db.fetch_assoc(stmt2)
    products=tuple(products)

    locations=[]
    row2 = ibm_db.fetch_assoc(stmt3)
    while(row2):
        locations.append(row2)
        row2 = ibm_db.fetch_assoc(stmt3)
    locations=tuple(locations)

    locs = []
    for i in locations:
        locs.append(list(i.values())[0])

    if result>0:
        return render_template('dashboard.html', products = products, locations = locs)
    else:
        msg='No products found'
        return render_template('dashboard.html', msg=msg)

#Product Form Class
class ProductForm(Form):
    product_id = StringField('Product ID', [validators.Length(min=1, max=200)])
    product_cost = StringField('Product Cost', [validators.Length(min=1, max=200)])
    product_num = StringField('Product Num', [validators.Length(min=1, max=200)])

```

```

#Add Product
@app.route('/add_product', methods=['GET', 'POST'])
@is_logged_in
def add_product():
    form = ProductForm(request.form)
    if request.method == 'POST' and form.validate():
        product_id = form.product_id.data
        product_cost = form.product_cost.data
        product_num = form.product_num.data

        sql1="INSERT INTO products(product_id, product_cost, product_num) VALUES(?,?,?)"
        stmt1 = ibm_db.prepare(conn, sql1)
        ibm_db.bind_param(stmt1,1,product_id)
        ibm_db.bind_param(stmt1,2,product_cost)
        ibm_db.bind_param(stmt1,3,product_num)

        ibm_db.execute(stmt1)

        flash("Product Added", "success")

        return redirect(url_for('products'))

    return render_template('add_product.html', form=form)

#Edit Product
@app.route('/edit_product/<string:id>', methods=['GET', 'POST'])
@is_logged_in
def edit_product(id):
    sql1="Select * from products where product_id = ?"
    stmt1 = ibm_db.prepare(conn, sql1)
    ibm_db.bind_param(stmt1,1,id)
    result=ibm_db.execute(stmt1)
    product=ibm_db.fetch_assoc(stmt1)

    print(product)
    #Get form
    form = ProductForm(request.form)

    #populate product form fields
    form.product_id.data = product['PRODUCT_ID']
    form.product_cost.data = str(product['PRODUCT_COST'])
    form.product_num.data = str(product['PRODUCT_NUM'])

    if request.method == 'POST' and form.validate():
        product_id = request.form['product_id']
        product_cost = request.form['product_cost']
        product_num = request.form['product_num']

        sql2="UPDATE products SET product_id=?,product_cost=?,product_num=? WHERE product_id=?"
        stmt2 = ibm_db.prepare(conn, sql2)
        ibm_db.bind_param(stmt2,1,product_id)
        ibm_db.bind_param(stmt2,2,product_cost)
        ibm_db.bind_param(stmt2,3,product_num)
        ibm_db.bind_param(stmt2,4,id)

```

```

        ibm_db.execute(stmt2)

        flash("Product Updated", "success")

        return redirect(url_for('products'))

    return render_template('edit_product.html', form=form)

#Delete Product
@app.route('/delete_product/<string:id>', methods=['POST'])
@is_logged_in
def delete_product(id):

    sql2="DELETE FROM products WHERE product_id=?"
    stmt2 = ibm_db.prepare(conn, sql2)
    ibm_db.bind_param(stmt2,1,id)

    ibm_db.execute(stmt2)

    flash("Product Deleted", "success")

    return redirect(url_for('products'))

#Location Form Class
class LocationForm(Form):
    location_id = StringField('Location ID', [validators.Length(min=1, max=200)])

#Add Location
@app.route('/add_location', methods=['GET', 'POST'])
@is_logged_in
def add_location():
    form = LocationForm(request.form)
    if request.method == 'POST' and form.validate():
        location_id = form.location_id.data

        sql2="INSERT into locations VALUES(?)"
        stmt2 = ibm_db.prepare(conn, sql2)
        ibm_db.bind_param(stmt2,1,location_id)
        ibm_db.execute(stmt2)

        flash("Location Added", "success")

        return redirect(url_for('locations'))

    return render_template('add_location.html', form=form)

#Edit Location
@app.route('/edit_location/<string:id>', methods=['GET', 'POST'])
@is_logged_in
def edit_location(id):

    sql2="SELECT * FROM locations where location_id = ?"
    stmt2 = ibm_db.prepare(conn, sql2)
    ibm_db.bind_param(stmt2,1,id)
    result=ibm_db.execute(stmt2)

```

```

location=ibm_db.fetch_assoc(stmt2)
#Get form
form = LocationForm(request.form)
print(location)

#populate article form fields
form.location_id.data = location['LOCATION_ID']

if request.method == 'POST' and form.validate():
    location_id = request.form['location_id']

    sql2="UPDATE locations SET location_id=? WHERE location_id=?"
    stmt2 = ibm_db.prepare(conn, sql2)
    ibm_db.bind_param(stmt2,1,location_id)
    ibm_db.bind_param(stmt2,2,id)
    ibm_db.execute(stmt2)

    flash("Location Updated", "success")

    return redirect(url_for('locations'))

return render_template('edit_location.html', form=form)

#Delete Location
@app.route('/delete_location/<string:id>', methods=['POST'])
@is_logged_in
def delete_location(id):
    sql2="DELETE FROM locations WHERE location_id=?"
    stmt2 = ibm_db.prepare(conn, sql2)
    ibm_db.bind_param(stmt2,1,id)
    ibm_db.execute(stmt2)

    flash("Location Deleted", "success")

    return redirect(url_for('locations'))

#Product Movement Form Class
class ProductMovementForm(Form):
    from_location = SelectField('From Location', choices=[])
    to_location = SelectField('To Location', choices=[])
    product_id = SelectField('Product ID', choices=[])
    qty = IntegerField('Quantity')

class CustomError(Exception):
    pass

#Add Product Movement
@app.route('/add_product_movements', methods=['GET', 'POST'])
@is_logged_in
def add_product_movements():
    form = ProductMovementForm(request.form)

    sql2="SELECT product_id FROM products"
    sql3="SELECT location_id FROM locations"
    stmt2 = ibm_db.prepare(conn, sql2)
    stmt3 = ibm_db.prepare(conn, sql3)

```

```
result=ibm_db.execute(stmt2)
ibm_db.execute(stmt3)
```

```
products=[]
row = ibm_db.fetch_assoc(stmt2)
while(row):
    products.append(row)
    row = ibm_db.fetch_assoc(stmt2)
products=tuple(products)
```

```
locations=[]
row2 = ibm_db.fetch_assoc(stmt3)
while(row2):
    locations.append(row2)
    row2 = ibm_db.fetch_assoc(stmt3)
locations=tuple(locations)
```

```
prods = []
for p in products:
    prods.append(list(p.values())[0])
```

```
locs = []
for i in locations:
    locs.append(list(i.values())[0])
```

```
form.from_location.choices = [(l,l) for l in locs]
form.from_location.choices.append(("Main Inventory","Main Inventory"))
form.to_location.choices = [(l,l) for l in locs]
form.to_location.choices.append(("Main Inventory","Main Inventory"))
form.product_id.choices = [(p,p) for p in prods]
```

```
if request.method == 'POST' and form.validate():
    from_location = form.from_location.data
    to_location = form.to_location.data
    product_id = form.product_id.data
    qty = form.qty.data
```

```
if from_location==to_location:
    raise CustomError("Please Give different From and To Locations!!")
```

```
elif from_location=="Main Inventory":
    sql2="SELECT * from product_balance where location_id=? and product_id=?"
    stmt2 = ibm_db.prepare(conn, sql2)
    ibm_db.bind_param(stmt2,1,to_location)
    ibm_db.bind_param(stmt2,2,product_id)
    result=ibm_db.execute(stmt2)
    result=ibm_db.fetch_assoc(stmt2)
    print("-----")
    print(result)
    print("-----")
    app.logger.info(result)
```



```

if result!=False:
    if(len(result))>0:
        Quantity = result["QTY"]
        q = Quantity + qty

        sql2="UPDATE product_balance set qty=? where location_id=? and product_id=?"
        stmt2 = ibm_db.prepare(conn, sql2)
        ibm_db.bind_param(stmt2,1,q)
        ibm_db.bind_param(stmt2,2,to_location)
        ibm_db.bind_param(stmt2,3,product_id)
        ibm_db.execute(stmt2)

        sql2="INSERT into productmovements(from_location, to_location, product_id, qty)
VALUES(?, ?, ?, ?)"
        stmt2 = ibm_db.prepare(conn, sql2)
        ibm_db.bind_param(stmt2,1,from_location)
        ibm_db.bind_param(stmt2,2,to_location)
        ibm_db.bind_param(stmt2,3,product_id)
        ibm_db.bind_param(stmt2,4,qty)
        ibm_db.execute(stmt2)
    else:

        sql2="INSERT into product_balance(product_id, location_id, qty) values(?, ?, ?)"
        stmt2 = ibm_db.prepare(conn, sql2)
        ibm_db.bind_param(stmt2,1,product_id)
        ibm_db.bind_param(stmt2,2,to_location)
        ibm_db.bind_param(stmt2,3,qty)
        ibm_db.execute(stmt2)

        sql2="INSERT into productmovements(from_location, to_location, product_id, qty)
VALUES(?, ?, ?, ?)"
        stmt2 = ibm_db.prepare(conn, sql2)
        ibm_db.bind_param(stmt2,1,from_location)
        ibm_db.bind_param(stmt2,2,to_location)
        ibm_db.bind_param(stmt2,3,product_id)
        ibm_db.bind_param(stmt2,4,qty)
        ibm_db.execute(stmt2)

sql = "select product_num from products where product_id=?"
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt,1,product_id)
current_num=ibm_db.execute(stmt)
current_num = ibm_db.fetch_assoc(stmt)

sql2="Update products set product_num=? where product_id=?"
stmt2 = ibm_db.prepare(conn, sql2)
ibm_db.bind_param(stmt2,1,current_num["PRODUCT_NUM"]-qty)
ibm_db.bind_param(stmt2,2,product_id)
ibm_db.execute(stmt2)

```

```

alert_num=current_num['PRODUCT_NUM']-qty

if(alert_num<=0):
    alert("Please update the quantity of the product { }, Atleast { } number of pieces must be added to
finish the pending Product Movements!".format(product_id,-alert_num))

elif to_location=="Main Inventory":
    sql2="SELECT * from product_balance where location_id=? and product_id=?"
    stmt2 = ibm_db.prepare(conn, sql2)
    ibm_db.bind_param(stmt2,1,from_location)
    ibm_db.bind_param(stmt2,2,product_id)
    result=ibm_db.execute(stmt2)
    result=ibm_db.fetch_assoc(stmt2)

    app.logger.info(result)
    if result!=False:
        if(len(result))>0:
            Quantity = result["QTY"]
            q = Quantity - qty

            sql2="UPDATE product_balance set qty=? where location_id=? and product_id=?"
            stmt2 = ibm_db.prepare(conn, sql2)
            ibm_db.bind_param(stmt2,1,q)
            ibm_db.bind_param(stmt2,2,to_location)
            ibm_db.bind_param(stmt2,3,product_id)
            ibm_db.execute(stmt2)

            sql2="INSERT into productmovements(from_location, to_location, product_id, qty)
VALUES(?, ?, ?, ?)"
            stmt2 = ibm_db.prepare(conn, sql2)
            ibm_db.bind_param(stmt2,1,from_location)
            ibm_db.bind_param(stmt2,2,to_location)
            ibm_db.bind_param(stmt2,3,product_id)
            ibm_db.bind_param(stmt2,4,qty)
            ibm_db.execute(stmt2)

            flash("Product Movement Added", "success")

        sql = "select product_num from products where product_id=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,product_id)
        current_num=ibm_db.execute(stmt)
        current_num = ibm_db.fetch_assoc(stmt)

        sql2="Update products set product_num=? where product_id=?"
        stmt2 = ibm_db.prepare(conn, sql2)
        ibm_db.bind_param(stmt2,1,current_num['PRODUCT_NUM']+qty)
        ibm_db.bind_param(stmt2,2,product_id)
        ibm_db.execute(stmt2)

        alert_num=q
        if(alert_num<=0):
            alert("Please Add { } number of { } to { } warehouse!".format(-q,product_id,from_location))
        else:
            raise CustomError("There is no product named { } in { }.".format(product_id,from_location))

```

```

else: #will be executed if both from_location and to_location are specified
    f=0
    sql = "SELECT * from product_balance where location_id=? and product_id=?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,1,from_location)
    ibm_db.bind_param(stmt,2,product_id)
    result=ibm_db.execute(stmt)
    result = ibm_db.fetch_assoc(stmt)

    if result!=False:
        if(len(result))>0:
            Quantity = result["QTY"]
            q = Quantity - qty

            sql2="UPDATE product_balance set qty=? where location_id=? and product_id=?"
            stmt2 = ibm_db.prepare(conn, sql2)
            ibm_db.bind_param(stmt2,1,q)
            ibm_db.bind_param(stmt2,2,from_location)
            ibm_db.bind_param(stmt2,3,product_id)
            ibm_db.execute(stmt2)
            f=1

            alert_num=q
            if(alert_num<=0):
                alert("Please Add { } number of { } to { } warehouse!".format(-q,product_id,from_location))

        else:
            raise CustomError("There is no product named { } in { }.".format(product_id,from_location))

    if(f==1):
        sql = "SELECT * from product_balance where location_id=? and product_id=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,to_location)
        ibm_db.bind_param(stmt,2,product_id)
        result=ibm_db.execute(stmt)
        result = ibm_db.fetch_assoc(stmt)

        if result!=False:
            if(len(result))>0:
                Quantity = result["QTY"]
                q = Quantity + qty

                sql2="UPDATE product_balance set qty=? where location_id=? and product_id=?"
                stmt2 = ibm_db.prepare(conn, sql2)
                ibm_db.bind_param(stmt2,1,q)
                ibm_db.bind_param(stmt2,2,to_location)
                ibm_db.bind_param(stmt2,3,product_id)
                ibm_db.execute(stmt2)

```

```

else:

    sql2="INSERT into product_balance(product_id, location_id, qty) values(?, ?, ?)"
    stmt2 = ibm_db.prepare(conn, sql2)
    ibm_db.bind_param(stmt2,1,product_id)
    ibm_db.bind_param(stmt2,2,to_location)
    ibm_db.bind_param(stmt2,3,qty)
    ibm_db.execute(stmt2)
    sql2="INSERT into productmovements(from_location, to_location, product_id, qty)
VALUES(?, ?, ?, ?)"
    stmt2 = ibm_db.prepare(conn, sql2)
    ibm_db.bind_param(stmt2,1,from_location)
    ibm_db.bind_param(stmt2,2,to_location)
    ibm_db.bind_param(stmt2,3,product_id)
    ibm_db.bind_param(stmt2,4,qty)
    ibm_db.execute(stmt2)

    flash("Product Movement Added", "success")

    render_template('products.html',form=form)

    return redirect(url_for('product_movements'))

    return render_template('add_product_movements.html', form=form)

#Delete Product Movements
@app.route('/delete_product_movements/<string:id>', methods=['POST'])
@is_logged_in
def delete_product_movements(id):

    sql2="DELETE FROM productmovements WHERE movement_id=?"
    stmt2 = ibm_db.prepare(conn, sql2)
    ibm_db.bind_param(stmt2,1,id)
    ibm_db.execute(stmt2)

    flash("Product Movement Deleted", "success")

    return redirect(url_for('product_movements'))

if __name__ == '__main__':
    app.secret_key = "secret123"
    #when the debug mode is on, we do not need to restart the server again and again
    app.run(debug=True)

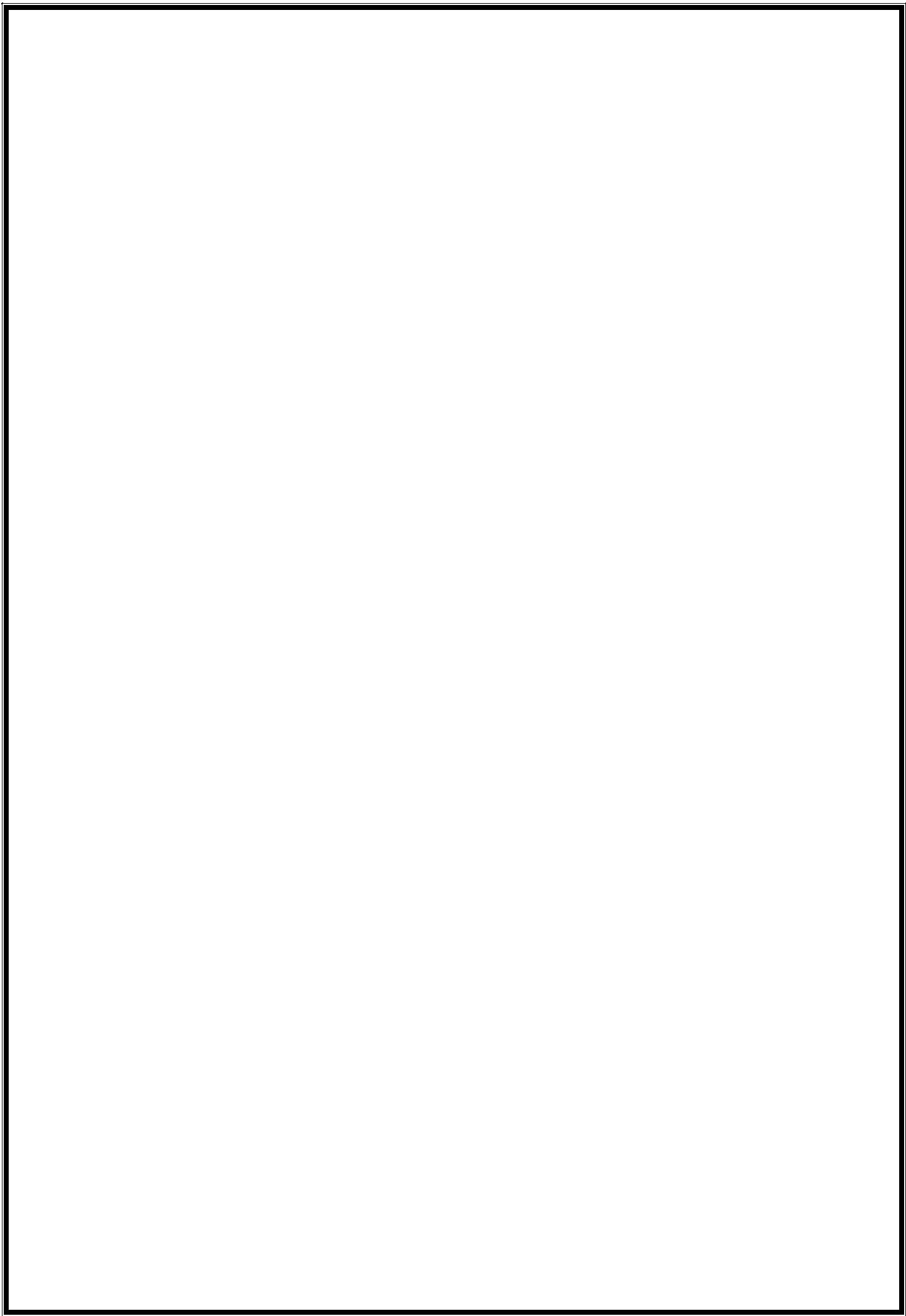
```

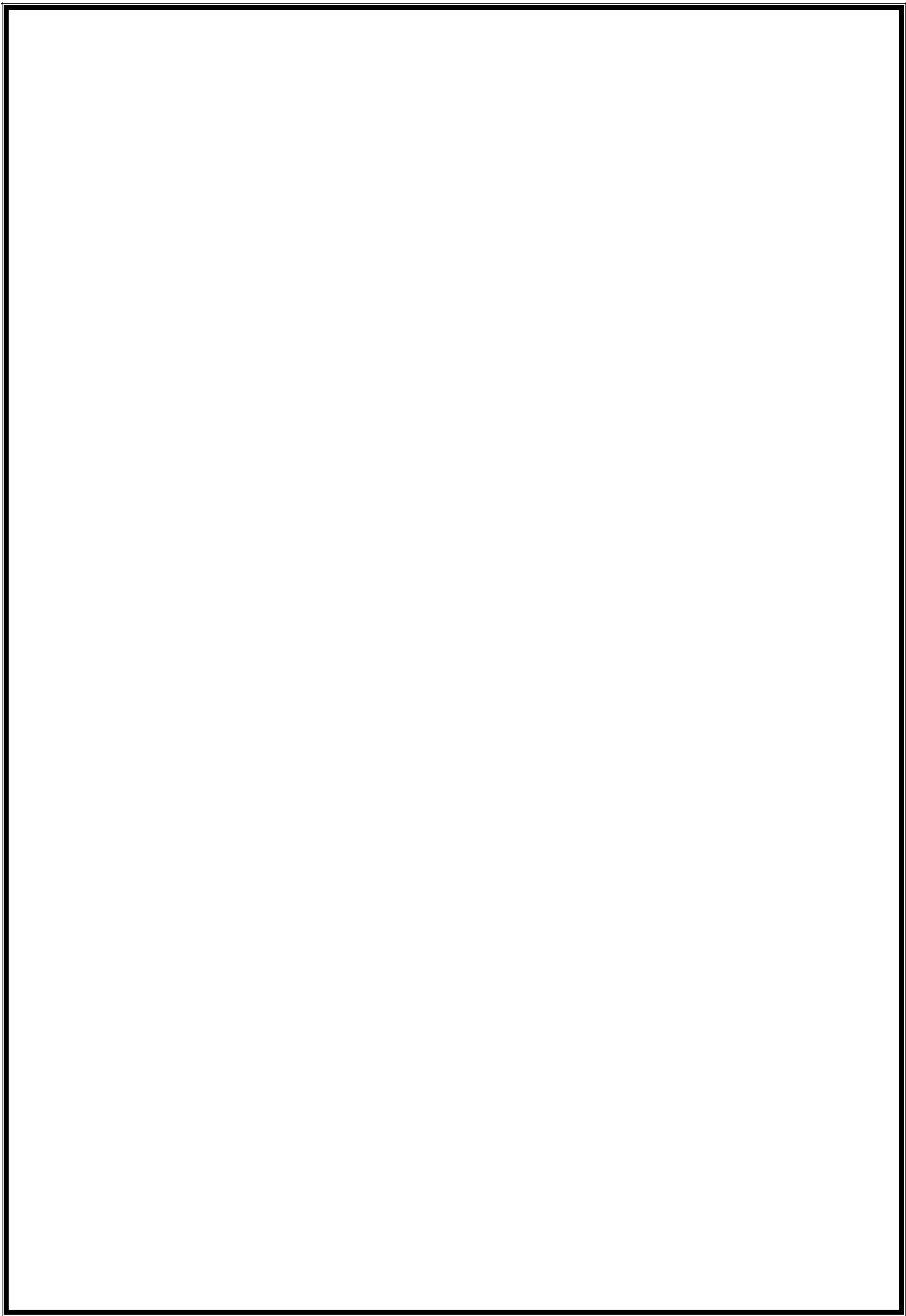
## 6.2 Feature 2

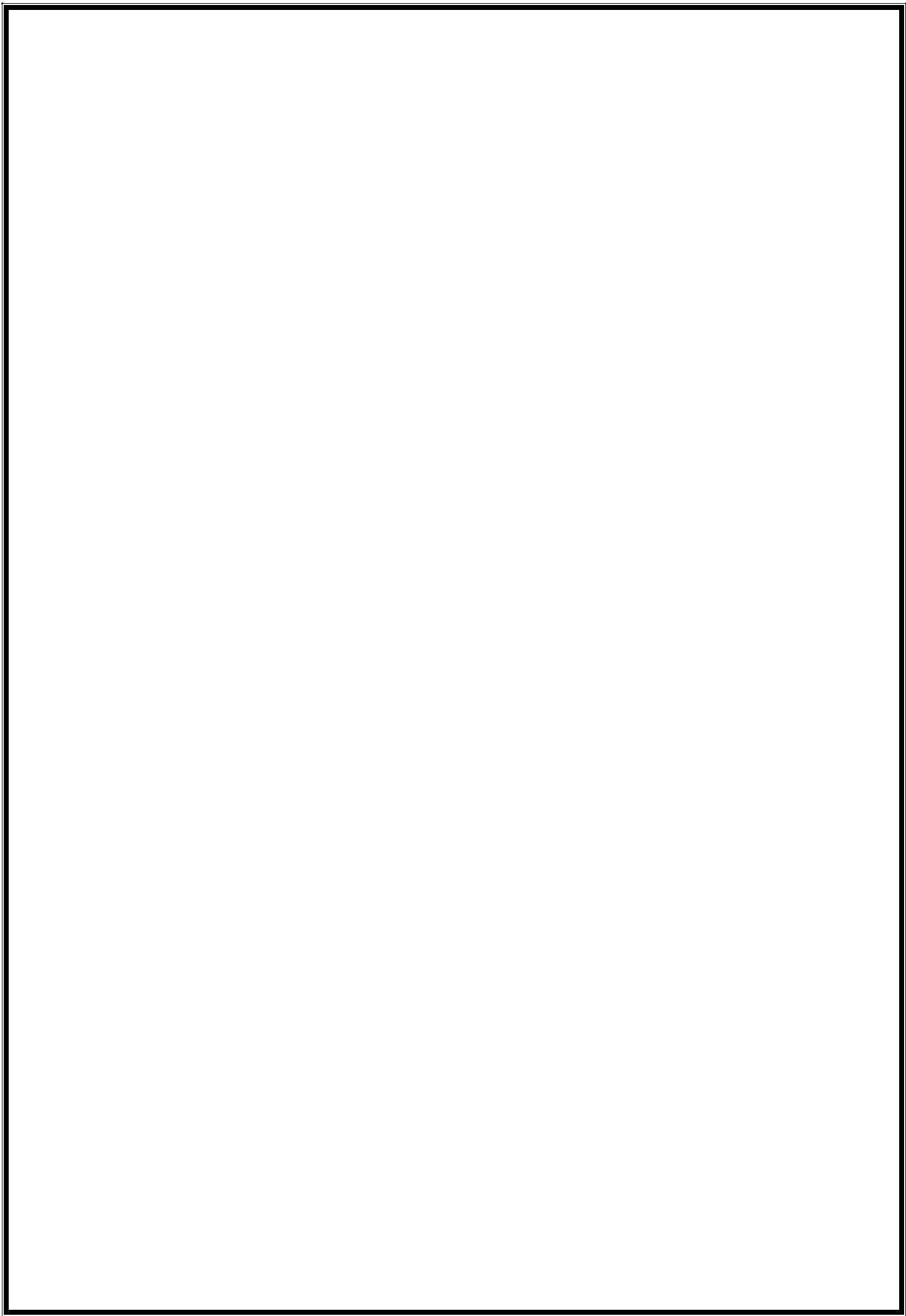
```
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.base import MIMEBase

def alert(main_msg):
    mail_from = 'ucs19430@rmd.ac.in'
    mail_to = 'vethanathanvk@gmail.com'
    msg = MIMEMultipart()
    msg['From'] = mail_from
    msg['To'] = mail_to
    msg['Subject'] = '!Alert Mail On Product Shortage! - Regards'
    mail_body = main_msg
    msg.attach(MIMEText(mail_body))

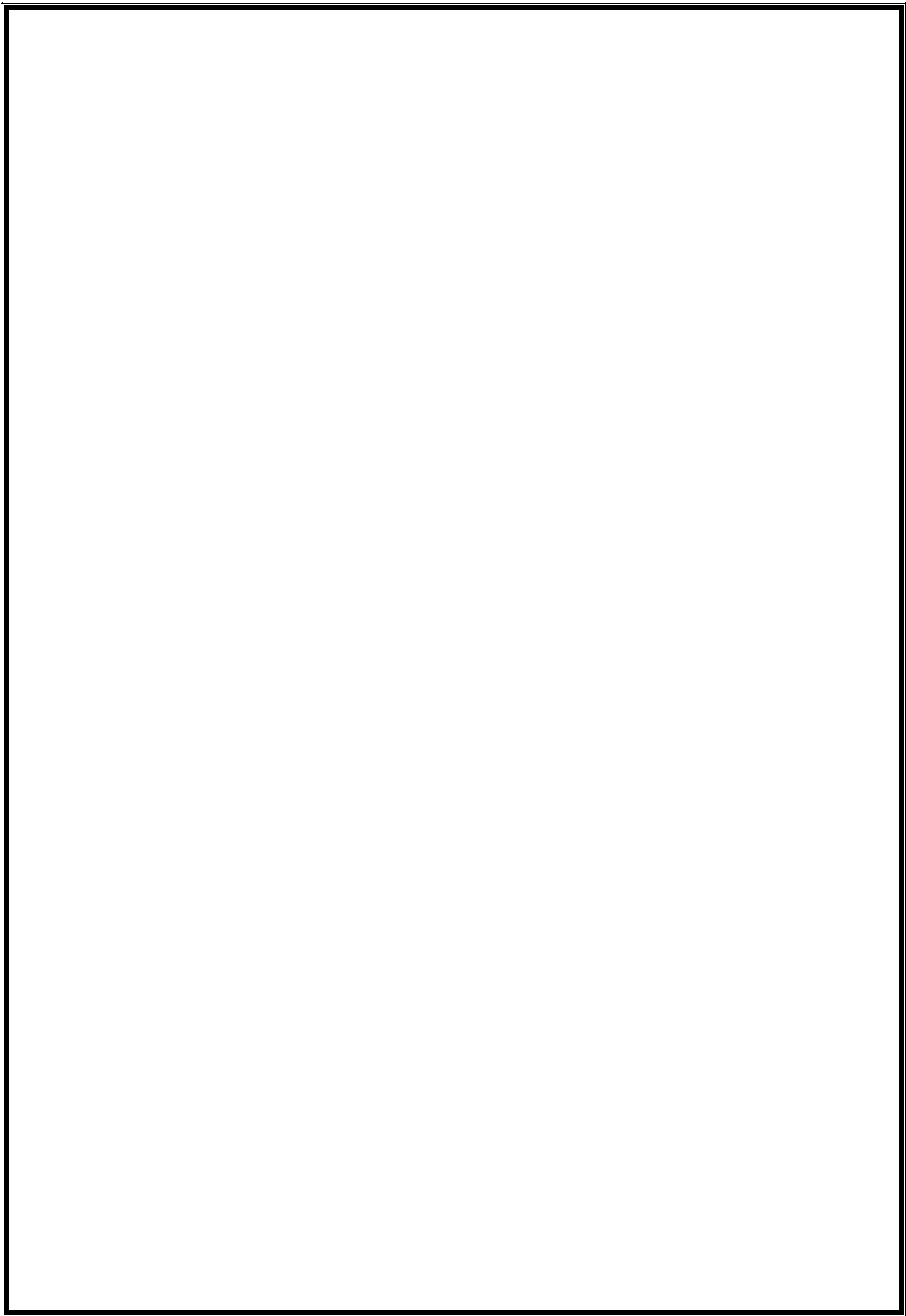
    try:
        server = smtplib.SMTP_SSL('smtp.sendgrid.net', 465)
        server.ehlo()
        server.login('apikey', 'API_KEY')
        server.sendmail(mail_from, mail_to, msg.as_string())
        server.close()
        print("Mail sent successfully!")
    except:
        print("Some Issue, Mail not Sent :(")
```

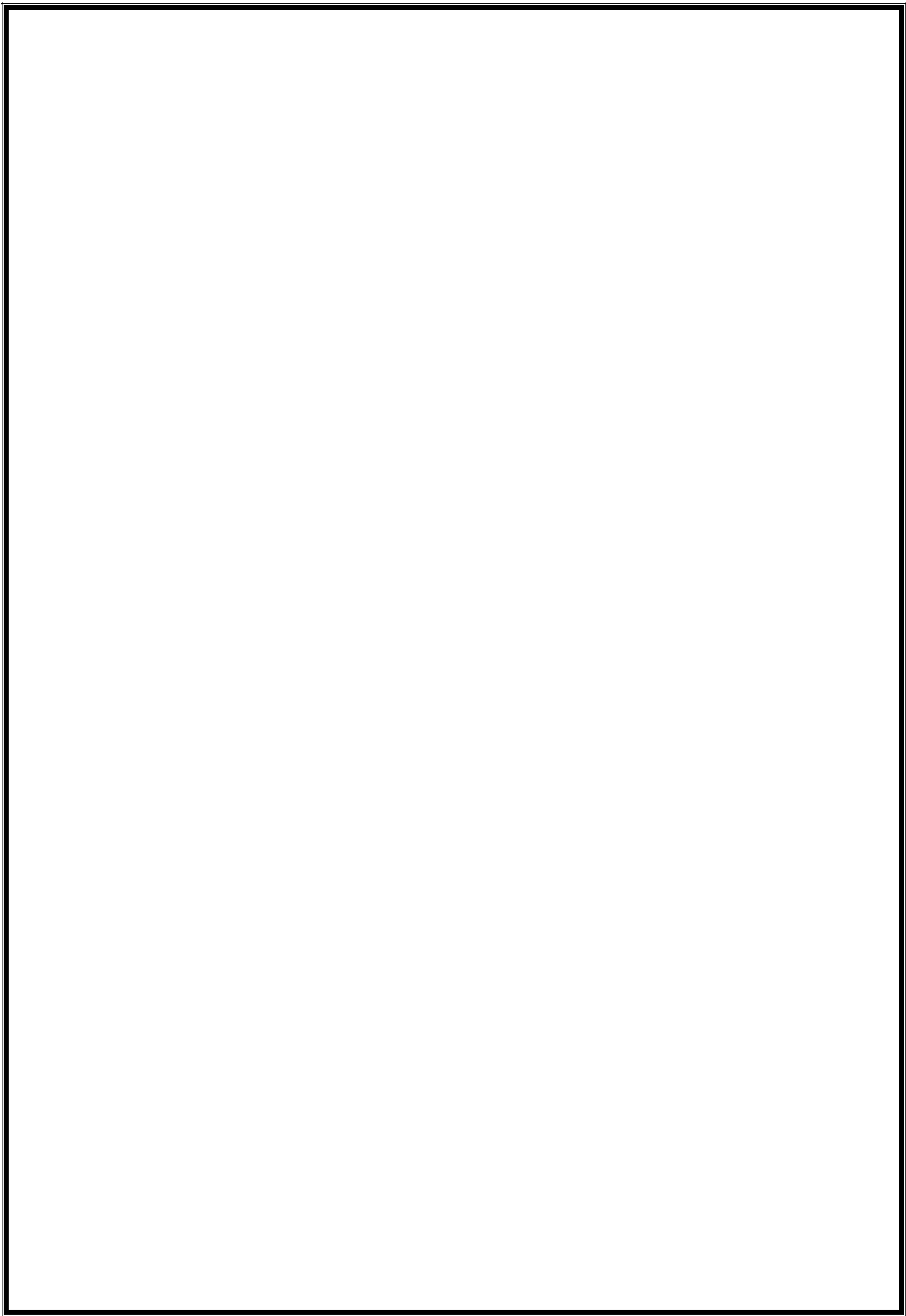


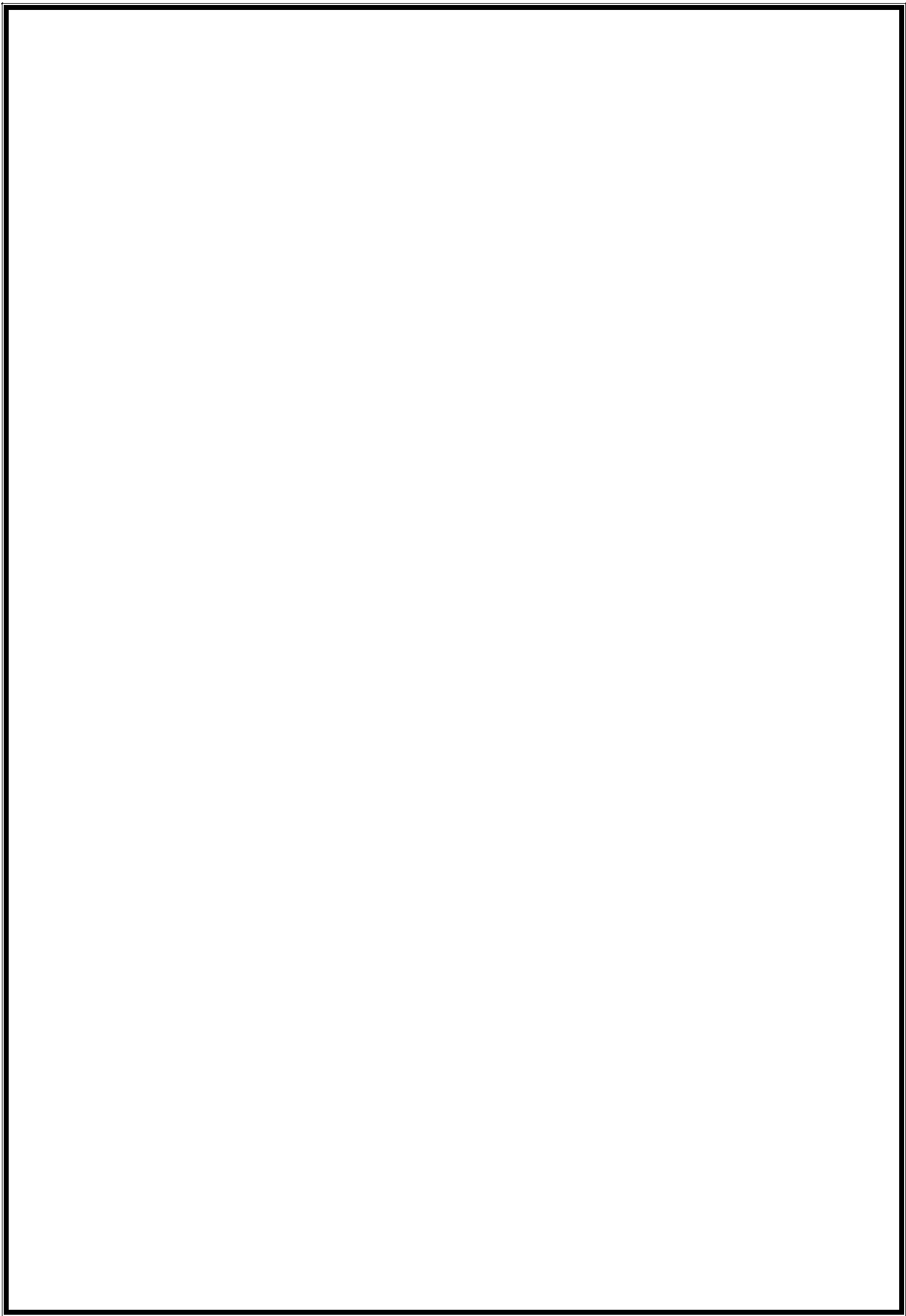












### **6.3 User Acceptance Testing**

User Acceptance Testing (UAT) is a type of testing performed by the end user or the client to verify/accept the software system before moving the software application to the production environment. UAT is done in the final phase of testing after functional, integration and system testing is done.

## **Purpose of UAT**

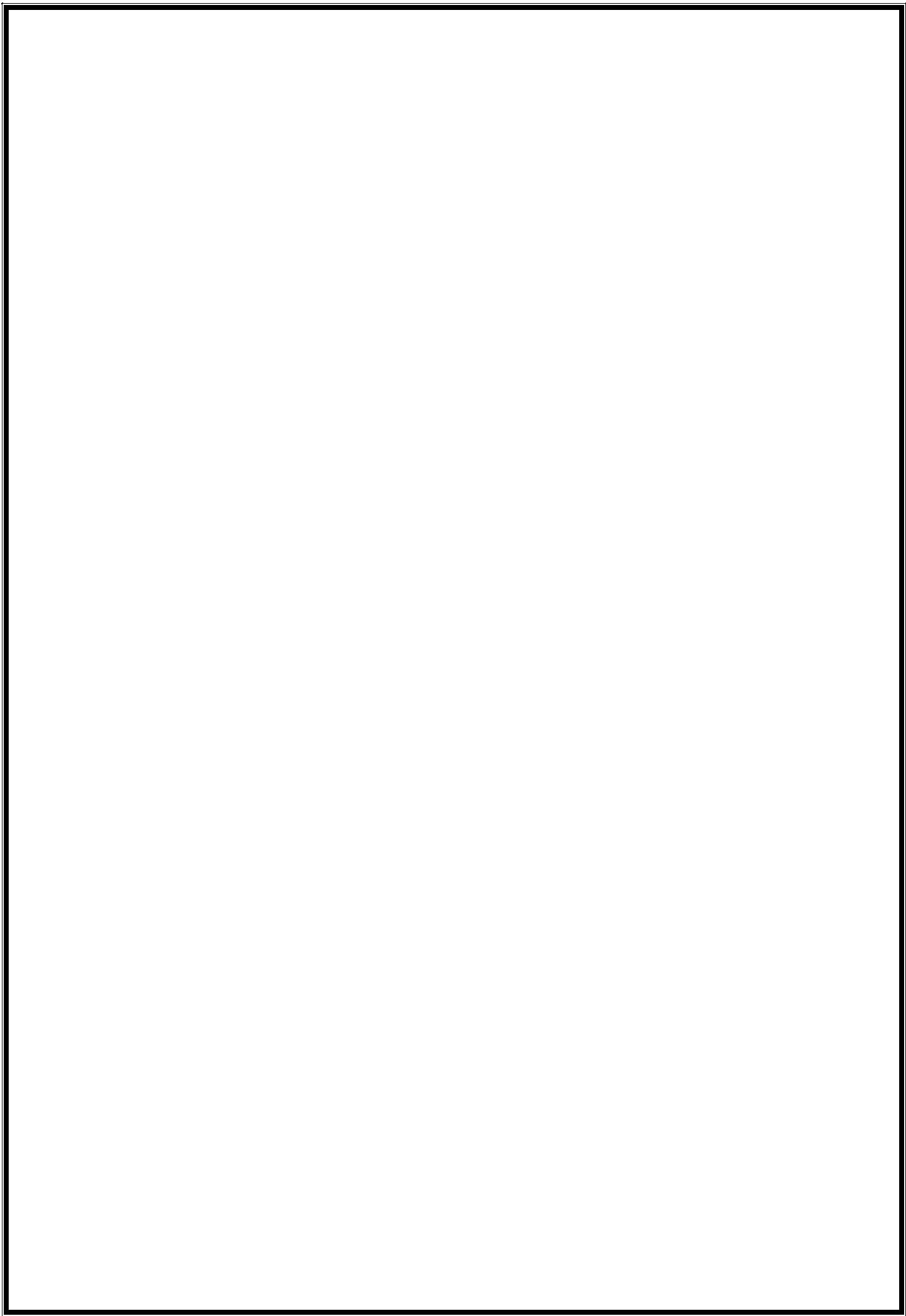
The main Purpose of UAT is to validate end to end business flow. It does not focus on cosmetic errors, spelling mistakes or system testing. User Acceptance Testing is carried out in a separate testing environment with production-like data setup. It is kind of black box testing where two or more end-users will be involved. UAT is performed by -

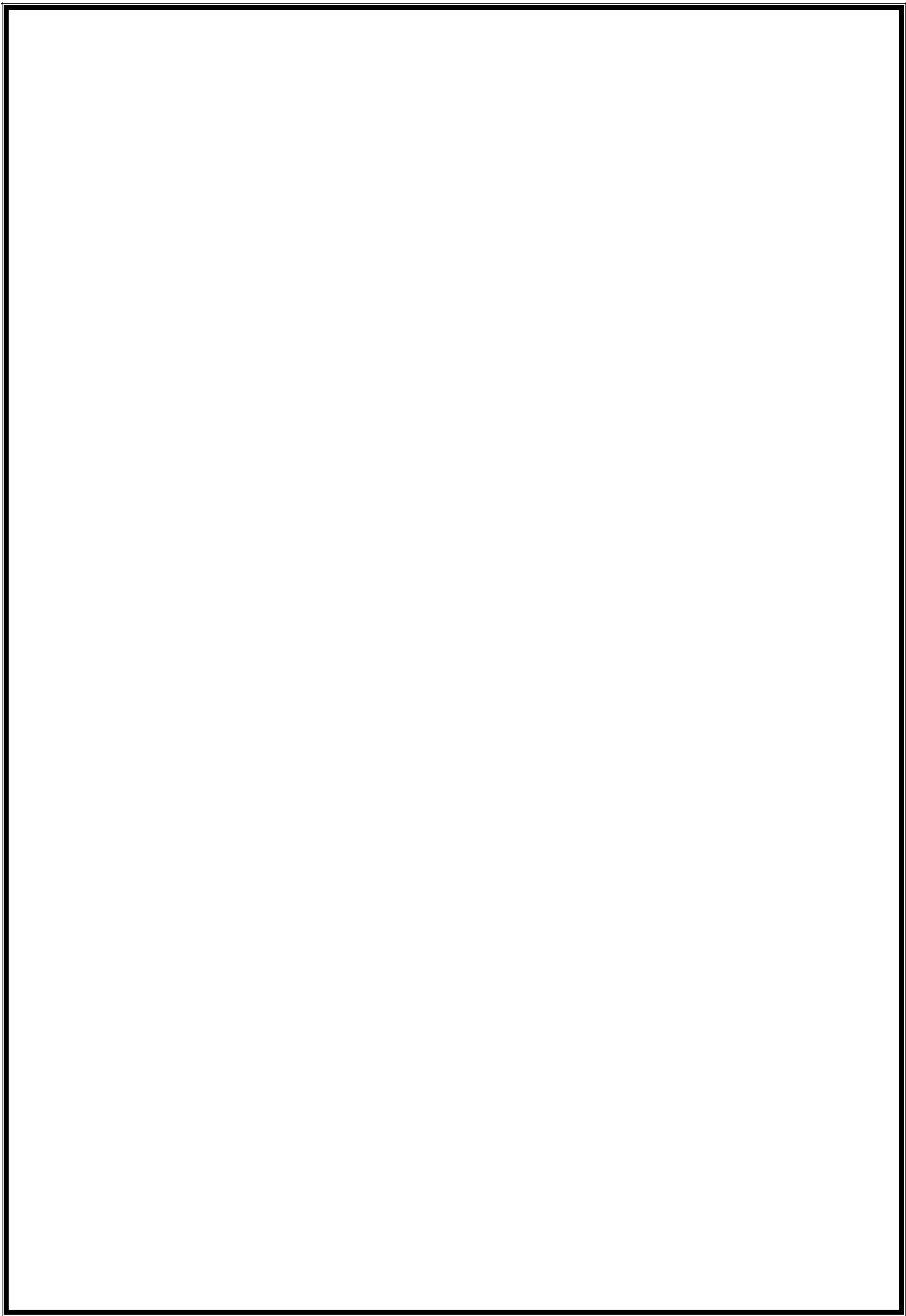
- Client
- End users

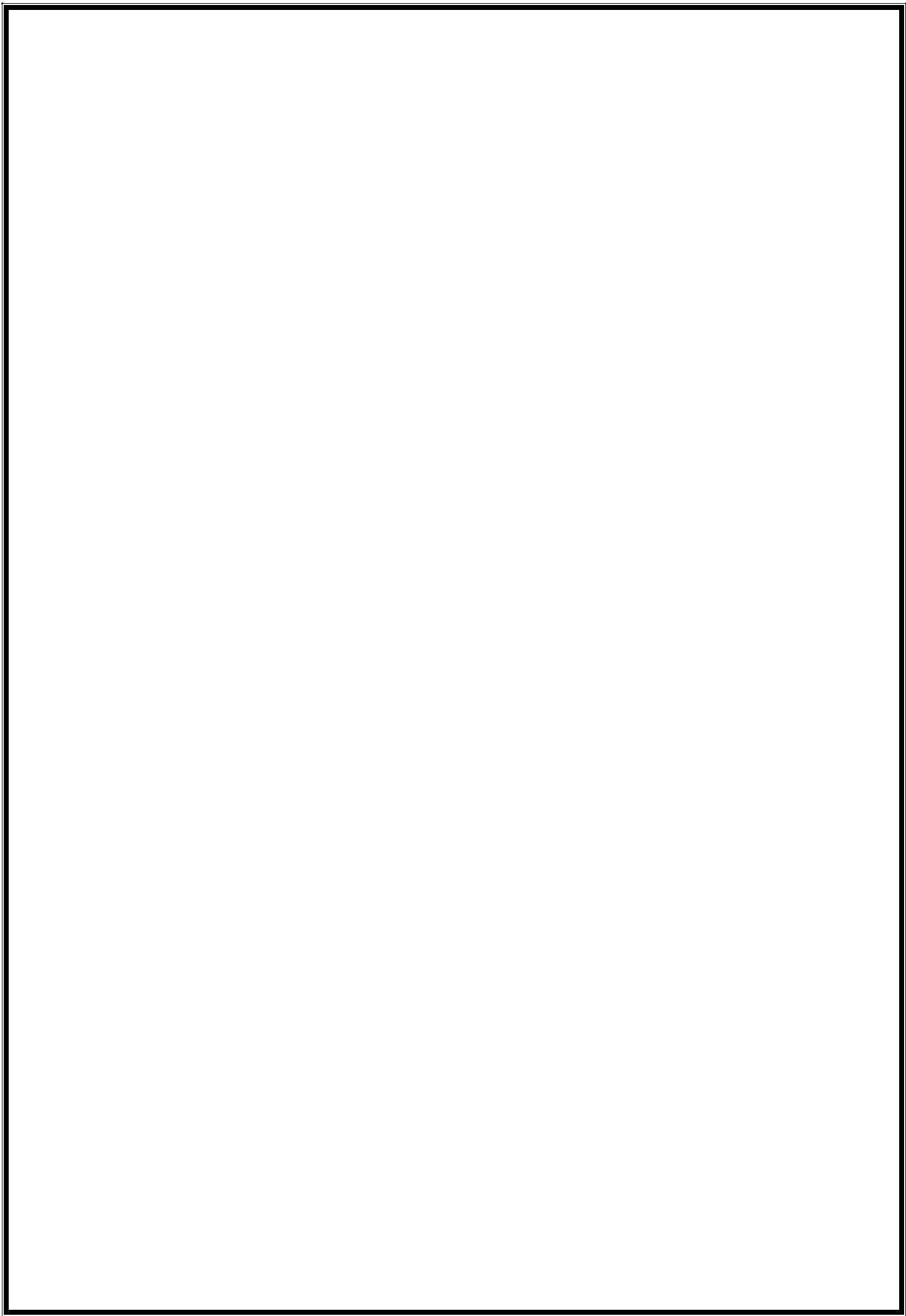
Need of User Acceptance Testing arises once software has undergone Unit, Integration and System testing because developers might have built software based on requirements document by their own understanding and further required changes during development may not be effectively communicated to them, so for testing whether the final product is accepted by client/end-user, user acceptance testing is needed.

Developers code software based on requirements document which is their “own” understanding of the requirements and may not actually be what the client needs from the software.

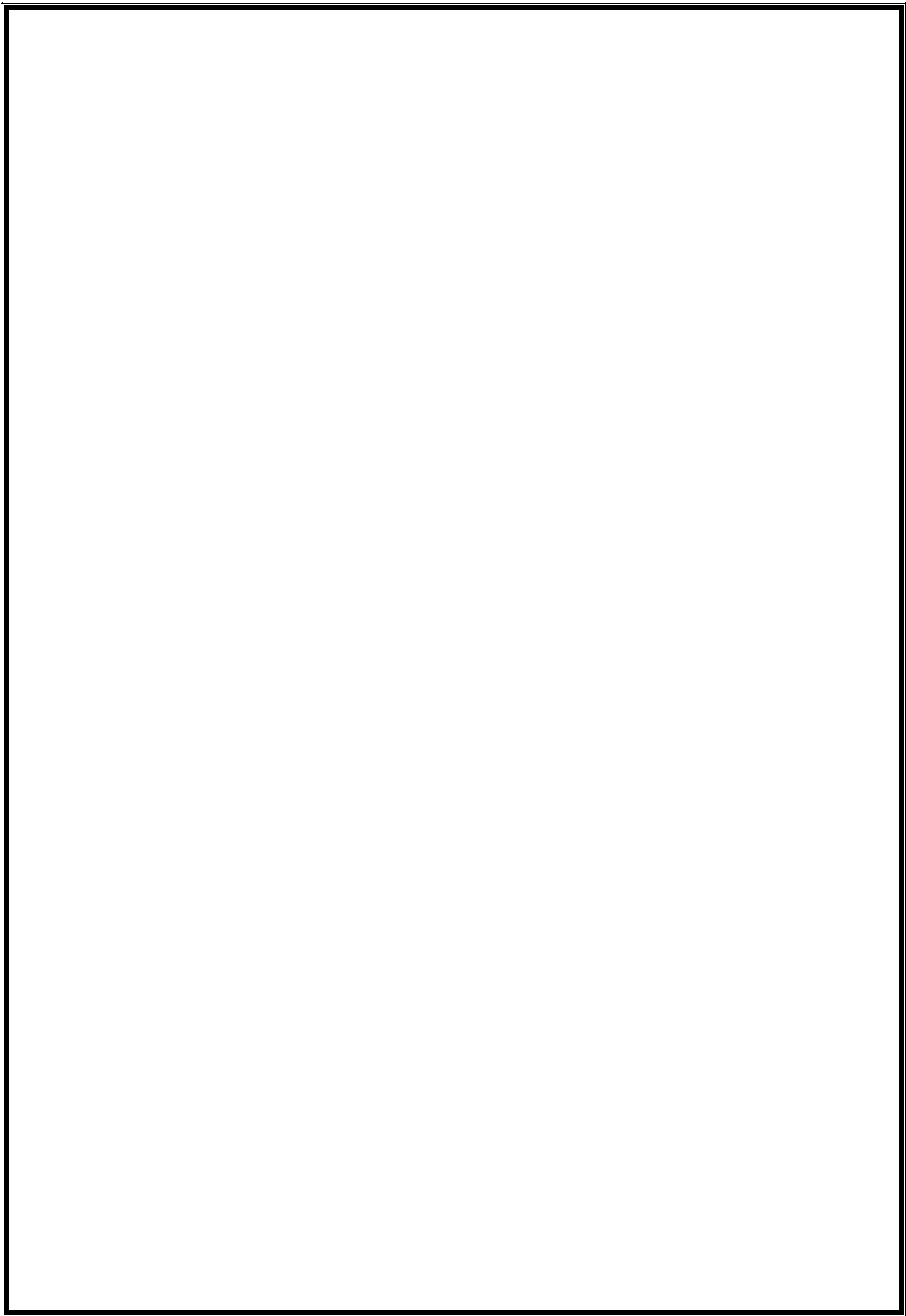
Requirements changes during the course of the project may not be communicated effectively to the developers.

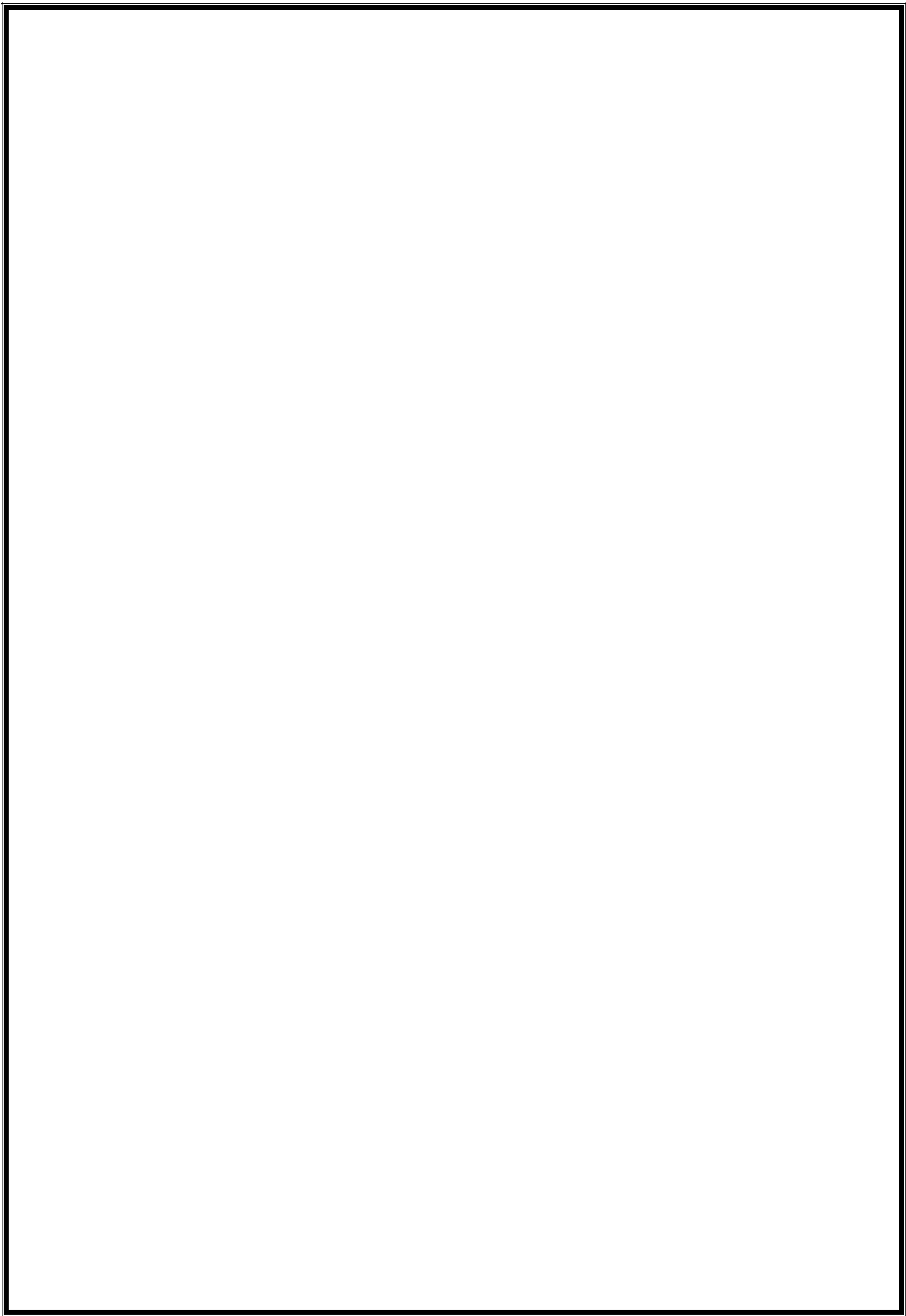


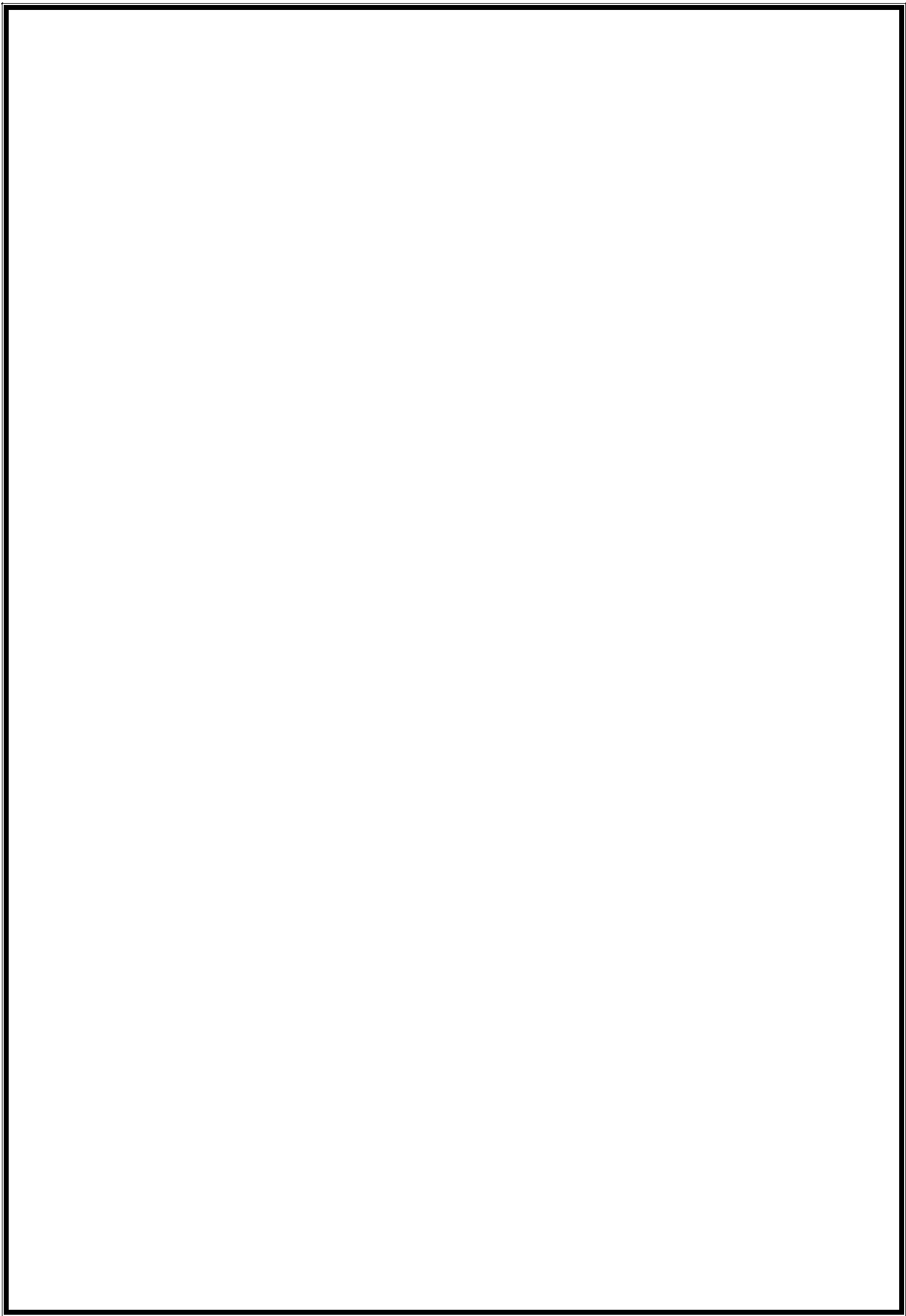


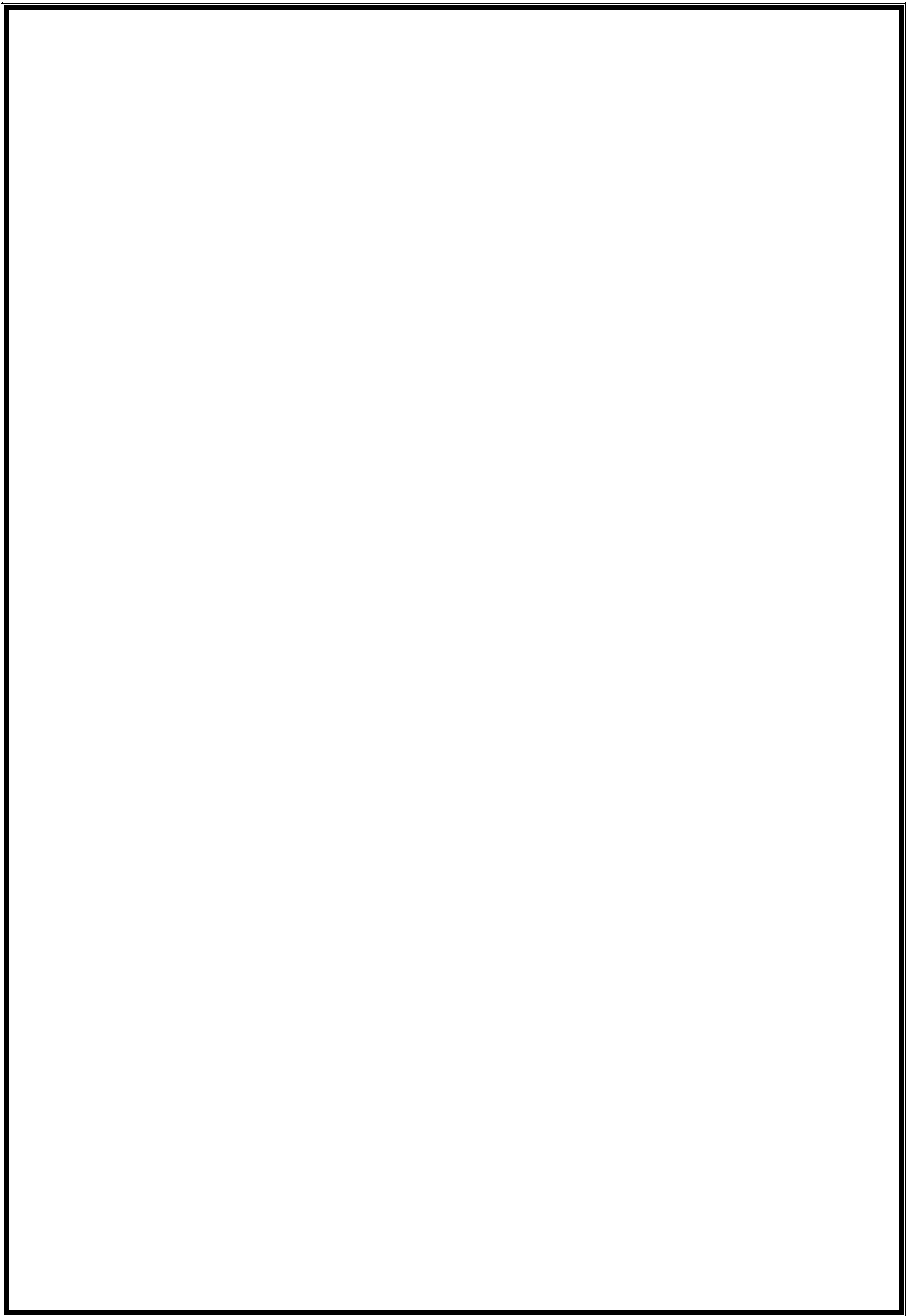


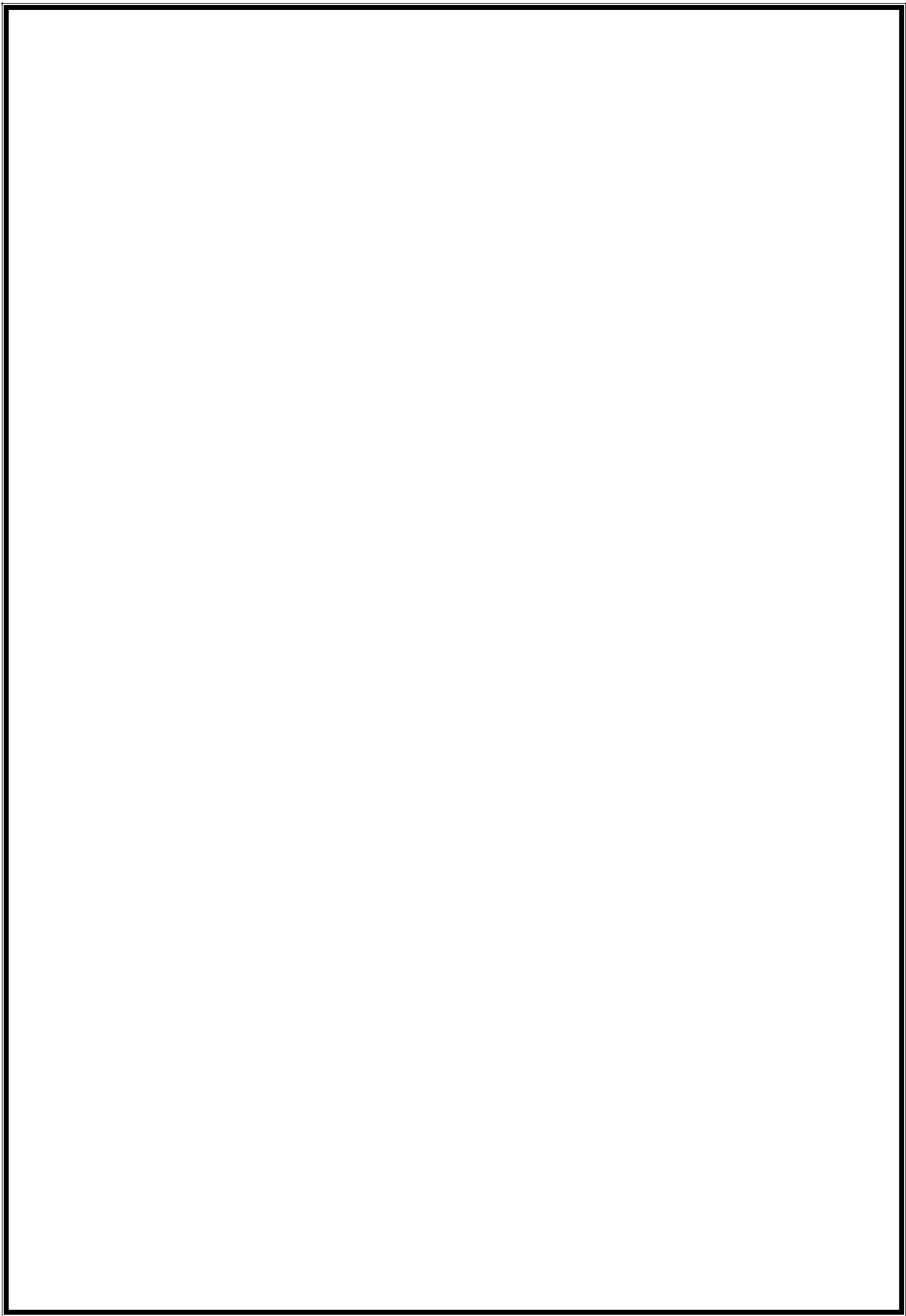


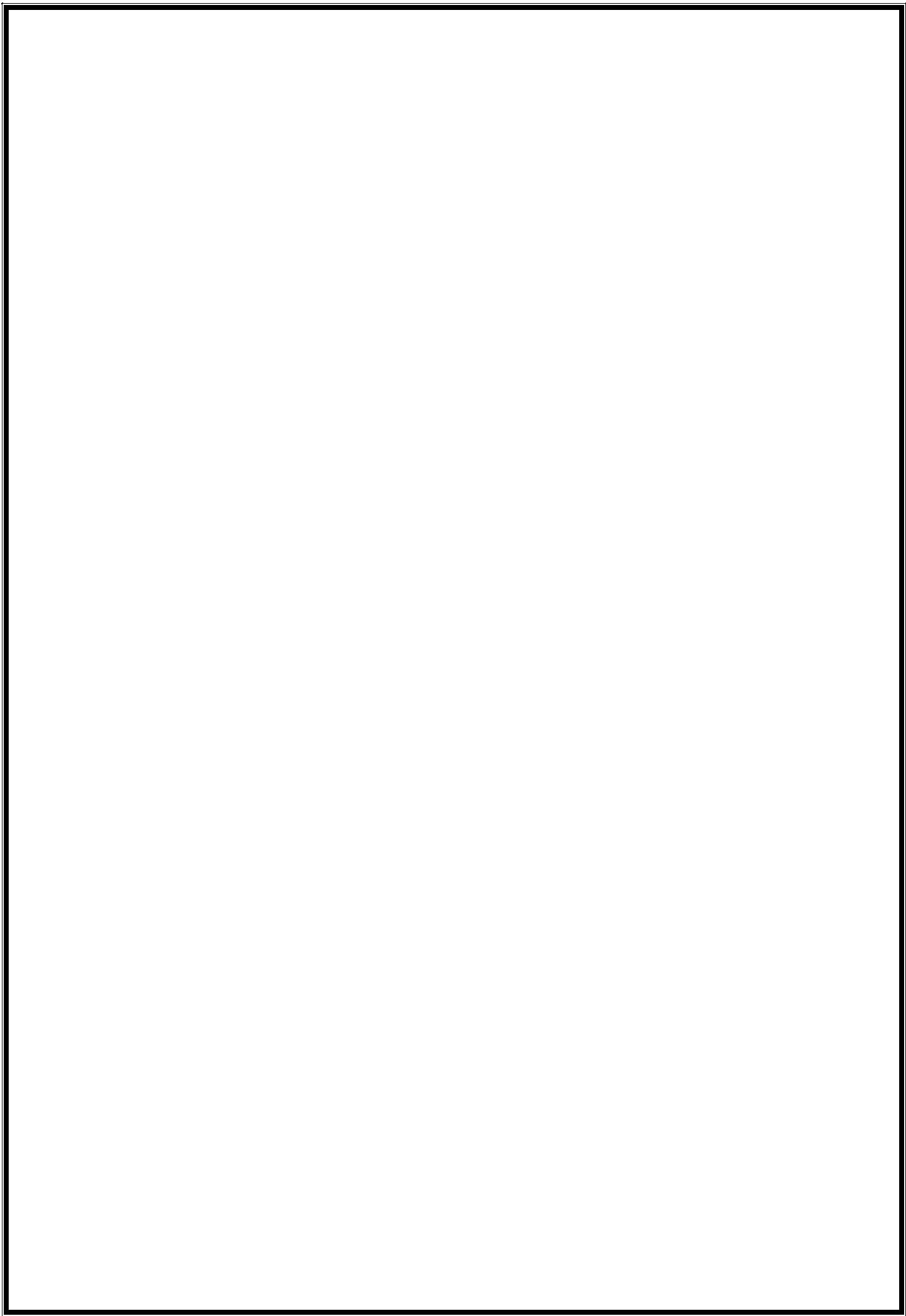










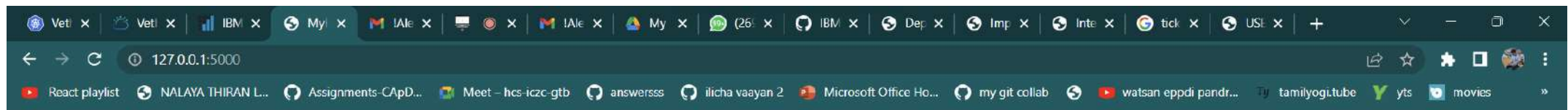


**GitHub Link:**

<https://github.com/IBM-EPBL/IBM-Project-33420-1660220018>

**Video Demo Link:**

<https://drive.google.com/file/d/1jRxI9auXeUBfIzRVz9bR1T3OMkXT-e9W/view?usp=sharing>



VETHA Home Products Location Product Movements

Logout Dashboard

# Inventory Management System for Retailers

## Developed by

- VETHANATHAN V K
- SRIVATSAN KUMAR S
- VIGNESH N
- SHREE KISHORE S

asset



Browser tabs: Vetl x, Vetl x, IBM x, My x, lAle x, x, lAle x, My x, (26 x, IBM x, Dep: x, Imp x, Inte x, tick x, USL x, +

Address bar: 127.0.0.1:5000/login

Browser bookmarks: React playlist, NALAYA THIRAN L..., Assignments-CApD..., Meet - hcs-iczc-gtb, answersss, ilicha vaayan 2, Microsoft Office Ho..., my git collab, watsan eppdi pandr..., familyogi.tube, yts, movies

You are now logged out

## Login

Username

Password

Submit

Browser tabs: Vetl x, Vetl x, IBM x, My x, lAle x, x, lAle x, My x, (26 x, IBM x, Dep x, Imp x, Inte x, tick x, USL x, +

Address bar: 127.0.0.1:5000/register

Browser bookmarks: React playlist, NALAYA THIRAN L..., Assignments-CApD..., Meet - hcs-iczc-gtb, answersss, ilicha vaayan 2, Microsoft Office Ho..., my git collab, watsan eppdi pandr..., familyogi.tube, yts, movies

# Register

Name

Email

Username

Password

Confirm Password

Submit

## Products

Add Product

Product ID	Product Cost	Product Quantity		
Lays RED flavour	50	-55	Edit	Delete
Milk Bikis	10	99	Edit	Delete
good day	15	40	Edit	Delete
MUNCH	10	76	Edit	Delete

## Add Location

Location ID		
Chennai	<a href="#">Edit</a>	<a href="#">Delete</a>
banglore	<a href="#">Edit</a>	<a href="#">Delete</a>
HYDRABED	<a href="#">Edit</a>	<a href="#">Delete</a>

VetI x VetI x IBM x My x IAlc x x x IAlc x My x (26 x IBM x Dep x Imp x Inte x tick x USL x +

127.0.0.1:5000/product\_movements

React playlist NALAYA THIRAN L... Assignments-CApD... Meet - hcs-iczc-gtb answersss ilicha vaayan 2 Microsoft Office Ho... my git collab watsan eppdi pandr... familyogi.tube yts movies

VETHA Home Products Location Product Movements Logout Dashboard

# Product Movements

Add Product Movements

Movement ID	Time	From Location	To Location	Product ID	Quantity	
		Main Inventory	Chennai	Lays RED flavour	100	Delete
		Main Inventory	Chennai	good day	10	Delete
		Chennai	banglore	good day	100	Delete
		Main Inventory	HYDRABED	MUNCH	4	Delete
		HYDRABED	banglore	MUNCH	2	Delete
		HYDRABED	Chennai	MUNCH	20	Delete

## Chennai

banglore

HYDRABED

NRN19864.USERS

Export to CSV

NAME	EMAIL	USERNAME	PASSWORD
Vethanathan V K	vethanathanvk@gmail.com	vethanathan	\$5\$rounds=535000\$8rUJfHPkCeZyLZux\$Xlqiv92/AyR0z/gGl9aqZ5.AkHLjggGStMVvaQmdg11







SQL

Tables

BALANCE\_PRODUCT

LOCATIONS

PERSONS

PRODUCTMOVEMENTS

PRODUCTS

PRODUCT\_BALANCE

USERS

Views

MQTs

Aliases

Nicknames

Data objects

Filter objects

NRN19864

Tables

BALANCE\_PRODUCT

LOCATIONS

PERSONS

PRODUCTMOVEMENTS

PRODUCTS

PRODUCT\_BALANCE

USERS

Views

MQTs

Aliases

Nicknames

\*Untitled - 1

drop table productmovements;

create table productmovements(from\_location varchar(255),to\_location varchar(255),qty int,product\_id varchar(255));

History

Script	Date	Status	Runtime
Untitled - 1	Nov 18, 2022 11:52:21 AM	2	0.095 s
drop table productmovements			0.043 s
create table productmovements(from_location varchar(255),to_location varcha...			0.052 s
Untitled - 1	Nov 18, 2022 11:49:59 AM	1	0.013 s
insert into product_balance(qty,product_id,location_id) values (5,'good day...			0.013 s
Untitled - 1	Nov 18, 2022 11:49:32 AM	2	0.060 s
create table product_balance(qty int,product_id varchar(255),location_id va...			0.049 s
insert into product_balance(qty,product_id,location_id) values (2,'Milk Bik...			0.011 s
Untitled - 1	Nov 18, 2022 11:47:39 AM	1	0.034 s
drop table product_balance			0.034 s
Untitled - 1	Nov 18, 2022 11:40:11 AM	1	0.021 s
insert into product_balance (qty,product_id,location_id) values(2,'Milk Bik...			0.021 s

Veth

Veth

IBM

MyFi

IAler

S

IAler

My L

(269)

IBM

Depl

Impl

Integ

tick

+

eu-de.containers.cloud.ibm.com/kubeproxy/clusters/cds5ukcf0ari9gibn5g0/service/#/workloads?namespace=default

React playlist

NALAYA THIRAN L...

Assignments-CApD...

Meet - hcs-iczc-gtb

answerss

ilicha vaayan 2

Microsoft Office Ho...

my git collab

watsan eppdi pandr...

tamilyogi.tube

yts

movies

»

kubernetes

default

Search

Workloads

Workloads

Cron Jobs

Daemon Sets

Deployments

Jobs

Pods

Replica Sets

Replication Controllers

Stateful Sets

Service

Ingresses

Ingress Classes

Services

Config and Storage

Config Maps

Persistent Volume Claims

Secrets

Storage Classes

Workload Status

Running: 1

Deployments

Running: 1

Pods

Running: 1

Replica Sets

Deployments

Name	Images	Labels	Pods	Created
<div></div> app	<a href="#">Show all</a>	-	1 / 1	4 hours ago

Pods

Final code

File Home Share View

Pin to Quick access

C:\Windows\System32\cmd.exe - docker build -t app:latest.

Microsoft Windows [Version 10.0.19044.2130]  
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP\Downloads\IBM-Project-33420-1660220018-main (1)\IBM-Project-33420-1660220018-main\Final Deliverables\Final code>docker build -t app:latest .

[+] Building 19.7s (5/11)

=> [internal] load build definition from Dockerfile 0.2s

=> => transferring dockerfile: 281B 0.0s

=> [internal] load .dockerignore 0.2s

=> => transferring context: 2B 0.0s

=> [internal] load metadata for docker.io/library/python:2.7 4.2s

=> [auth] library/python:pull token for registry-1.docker.io 0.0s

=> [internal] load build context 0.3s

=> => transferring context: 43.58kB 0.2s

=> [1/6] FROM docker.io/library/python:2.7@sha256:cfa62318c459b1fde9e0841c619906d15ada5910d625176e24bf692cf8a26 15.1s

=> => resolve docker.io/library/python:2.7@sha256:cfa62318c459b1fde9e0841c619906d15ada5910d625176e24bf692cf8a260 0.1s

=> => sha256:cfa62318c459b1fde9e0841c619906d15ada5910d625176e24bf692cf8a2601d 2.14kB / 2.14kB 0.0s

=> => sha256:c934af72b8bd03b9804d5bde2569c320926e70392d708d113a2e71bcf98c8a20 2.22kB / 2.22kB 0.0s

=> => sha256:68e7be49c28c89833fc91b44a8c1db6982b52695749d57c3d6fe995a9bb1df5f 8.93kB / 8.93kB 0.0s

=> => sha256:09b6f03ffac4cb4e42f8ab0bfc480bd3a3fa20e1dde37784db63bc886b0cbb3 7.81MB / 7.81MB 5.6s

=> => sha256:7e2b2a5af8f65687add6d864d5841067e23bd435eb1a051be6fe1ea2384946b4 16.78MB / 50.38MB 15.0s

=> => sha256:dc3f0c679f0f4c39597721c1df5c5db4f9685b26bd789a44eeb406835a1800d5f 10.00MB / 10.00MB 9.4s

=> => sha256:fd4b47407fc30b8206971ec60f280b107b00df8007da2fb912ebb8656b53695e 10.49MB / 51.83MB 15.0s

=> => sha256:b32f6bf7d96d26a22dc62da6522f384dc936c30c88b233d378e06cf127346d 5.24MB / 192.17MB 15.0s

Downloads

Music

Pictures

Videos

10 items

Satisfactory air 16:58

IBM Cloud

Search resources and products...

Catalog

Manage

VETHANATHAN VK's A...

Help

Kubernetes dashboard

Actions...

Clusters /

Vethanathan

Normal

Expires in 30 days

Add tags

Overview

Worker nodes

Worker pools

DevOps

Pool: Filter...

Search

Add

	Name	Status	Worker pool	Zone	Private IP	Public IP	Version
<input type="checkbox"/>	00000010	Normal	default	Milan 01	10.144.183.53	169.51.200.58	1.24.7_1543


Items per page: 25


1-1 of 1 item

1 of 1 page


Docker Desktop


Upgrade plan








vethanathan














Containers



Images



Volumes





Dev Environments

BETA

Extensions

BETA






Add Extensions

Containers

[Give feedback](#)




A container packages up code and its dependencies so the application runs quickly and reliably from one computing environment to another.


[Learn more](#)







☐

Only show running containers



Search



<input type="checkbox"/>	NAME	IMAGE	STATUS	PORT(S)	STARTED	ACTIONS
<input type="checkbox"/>	<div><div></div><div>quizzical_brahmagupta</div><div>7ecde2aed29a</div><div></div></div>	<a href="#">app:latest</a>	Running	<a href="#">5000:5000</a> 	3 hours ago	<div><div></div><div></div><div></div></div>

Showing 1 items





## Containers

Images

Volumes

 Dev Environments **BETA**Extensions **BETA**

Add Extensions

Containers [Give feedback](#)

A container packages up code and its dependencies so the application runs quickly and reliably from one computing environment to another. [Learn more](#)

☐ Only show running containers

<input type="checkbox"/>		NAME	IMAGE	STATUS	PORT(S)	STARTED	ACTIONS
<input type="checkbox"/>		<b>quizzical_brahmagupta</b> 7ecde2aed29a	<a href="#">app:latest</a>	Running	<a href="#">5000:5000</a>	3 hours ago	

Showing 1 items



RAM 1.80GB

CPU 0.06%

Connected to Hub

v4.13.1

IBM Cloud

Search resources and products...

Catalog

Manage

VETHANATHAN VK's A...

?

Clusters /

Vethanathan

Normal

Expires in 30 days

Add tags

Help

Kubernetes dashboard

Actions...

Overview

Worker nodes

Worker pools

DevOps

New

Expires in 30 days:

Be sure to back up your data, your cluster will be deleted in 30 days. To access the full capabilities of the service, try out a [standard cluster](#).

Node status

1 of 1

Normal

Details

Add-on status

0 of 0

Normal

Details

Master status

Normal

Docs

Ingress status

Healthy

Docs

Details

Cluster ID

cds5ukcf0ari9gibn5g0

Version

1.24.8\_1544

Infrastructure

Classic

Zones

Milan 01

Created

19/11/2022, 10:15 am

Resource group

Default

Image security enforcement

Enable

Node health

Worker node details

1 total nodes

