1. INTRODUCTION

a. Overview

Payment book application is designed to maintain customers, payments, and their purchases. A retailer will be an admin of the application and each customer of the retailer's shop will be the user. Customers can create their account in the payment book app by reading and agree to the terms and conditions of the shop

b. Purpose

The main objectives of digital transactions are to reduce the costs and risks of handling cash and to increase the ease of conducting transactions. How do digital payments work? …..Digital payments can also be defined as any payments that are done online via the internet or mobile-enabled services.

2. LITERATURE SURVEY

a. Existing problem.

In the existing system the exams are done only manually but in proposed system we have to computerize the exams using this application.

1. Faster, easier, more convenient.
2. Economical and less transaction fee.
3. Waivers, discounts and cashbacks.
4. Digital record of transactions.
5. One stop solution for paying bills
6. Minimum time required

b. Proposed solution

The aim of proposed system is to develop a system of improved facilities. The proposed system can overcome all the limitations of the existing system. The system

1. Digital payments allow buyers to pay directly from their banks.

2. Digital payments allow merchants to sell to other countries and customers to pay in foreign countries.
3. Tracks and maintains the digital record of every transaction
4. No fear of losing cash or getting fake currency
5. Merchants can collect payments remotely
6. Digital payments offer cashbacks or reward points

3. THEORITICAL ANALYSIS

a. Block diagram

Diagrammatic overview of the project.

b. Hardware / Software designing

1. Hardware Requirement

- 64-bit processor with Second Level Address Translation (SLAT)
- 4GB system RAM.
- BIOS-level hardware virtualization support must be enabled in the BIOS settings. For more information, see Virtualization

2. Software Requirement

- HTML
- CSS
- Bootstrap
- Python -Flask
- Remote MySQL

4. EXPERIMENTAL INVESTIGATIONS

A digital payment saves card details of the user and lets him/her make purchases online. For example, a Master Pass digital wallet asks customers to first register for the service by entering card details. The information then has to be verified through an OPT authentication process, post which, registration will be complete. After this process is complete, one can choose the ‘Buy with Master Pass’ option as a payment mode on e-commerce websites that have this option. The payment can be made by entering the Master Pass card ID and password and the transaction will be authenticated through a 3D-secure pin or OTP

5. FLOWCHART

Diagram showing the control flow of the solution

6. RESULT

Final findings (Output) of the project along with screenshots.

7. ADVANTAGES & DISADVANTAGES

ADVANTAGES

- Low labor costs:

  Since online payments are usually automatic, they have lower labor costs than manual payment methods, such as cheese, money order, cash and EFTPOS.
- Convenience for online sales:

  Online payment methods allow conveniently selling goods and services online
- Automatic:

  Online payments can be automatic, which can be convenient for you and your customers.
- Fast transaction speed:

  Online transactions quickly provide feedback to you and your customers.
- Low risk of theft:

  After processing delays, online payments generally go straight into

your bank account, so they have a low risk of theft

DISADVANTAGES

- Service fees:

  Payment gateways and third-party payment processors charge service fees.
- Inconvenient for offline sales:

  Online payment methods are inconvenient for offline sales.
- Vulnerability to cybercriminals:

  Cybercriminals can disable online payment methods or exploit them to steal people's money or information. Visit the Australian Cybercrime Reporting Network's Learn about cybercrime page to learn more about cybercrime.
- Reliance on telecommunication infrastructure:

  Internet and server problems can disable online payment methods.
- Technical problems:

  Online payment methods can go down due to technical problems.

8. APPLICATIONS

Ease and convenience

One of the most significant advantages of digital payment is the seamless experience they provide to customers. Reduced dependency on cash, fast transfer speed, and the ease of transacting make online payments a preferred option. Traditional payment methods like cash and cheques add to factors like risk, steps, and physical presence. With digital payment, you can send and receive funds from anywhere in the world at the click of a button.

Economic progress

Customers transact more online when they see the ease, convenience, and

security of online payments. This means that more and more people feel comfortable buying online, investing digitally, and transferring funds via electronic mediums. The increase in money movement and online business contributes to the progress of the economy. This is why online ventures are being launched every day and even more are making profits daily.

Safety and efficient tracking

Handling and dealing in cash is a cumbersome and tedious task. Along with the risk of losing money, there is the hassle of carrying cash everywhere you go and keeping it safe. With digital payments, one can keep their funds secured in online format effortlessly. Nowadays, your mobile phone alone is enough to make and receive payments – thanks to UPI, net banking, and mobile wallets. Additionally, most digital payment channels provide regular updates, notifications, and statements for a customer to track his funds.

9. CONCLUSION

In future the digital payments are going to be a must and so the change in the habits of the people to accept the digital payment is also must. The cashless transition is not only safer than the cash transaction but is less time consuming. It also helps in record of the all the transaction done.

Going digital will help in keeping track of the monetary transactions taking place and will pose more security on individual's wealth. Digital payments will also be a step for an ecofriendly environment as the usage of paper reduces. A drawback factor to the making of a digital India will be the high rates of illiteracy and poverty.
In conclusion, electronic transfer funds have been around for many years and the economy has greatly benefited from this technological advance. An electronic payment system such as credit cards has facilitated monetary transactions and even provides a way to finance everyday purchases through credit.

10. FUTURE SCOPE

The future is increasingly digital, which means a greater take-up of digital payment methods.

These methods include automated clearing hours payments, which are expected to rise in prominence, particularly with the global movement towards immediate or real-time payments.

the future of payment technology, from mobile payments to tokenization. On this course, you will learn new ways of making payments from consumer-to-business (C2B), from consumer-to-consumer (C2C), and from business-to-business (B2B).

11. BIBILOGRAPHY

References of previous works or websites visited/books referred for analysis about the project, solution previous findings etc.

APPENDIX

A. Source Code

app.py

```
1 from flask import Flask, render_template, abort, redirect,
```

```
    request, url_for, flash, session
2   from flask_migrate import Migrate
3   from flask_login import login_user, login_required,
    logout_user, current_user
4   from forms import *
5   from models import db, login_manager, User, Ppay, Hpay
6   from flask_mail import Mail, Message
7
8   app = Flask(__name__)
9
10  app.config['SECRET_KEY'] = 'supersecretkeydonthack'
11  # app.config['SQLALCHEMY_DATABASE_URI'] =
    'mysql://root:@localhost/digitalpayment'
12  app.config['SQLALCHEMY_DATABASE_URI'] =
    'mysql://8NxFFu2fxx:LOMVTDxNtg@remotemysql.com/8NxFFu2fxx'
13  app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
14  app.config['MAIL_SERVER'] = 'smtp.gmail.com'
15  app.config['MAIL_PORT'] = 465
16  app.config['MAIL_USERNAME'] =
    'pratikshaprojectmail@gmail.com'
17  app.config['MAIL_PASSWORD'] = 'Digital@8600'
18  app.config['MAIL_USE_TLS'] = False
19  app.config['MAIL_USE_SSL'] = True
20  mail = Mail(app)
21
22  db.init_app(app)
23  Migrate(app, db)
24
25  login_manager.init_app(app)
26
27  login_manager.login_view = "login"
28
29  @app.errorhandler(404)
30  @login_required
31  def page_not_found(e):
32      user = session['user']
33      return render_template('404.html', user=user), 404
```

```
34
35
36 @app.route('/')
37 def index():
38     return redirect(url_for('register'))
39
40
41 @app.route('/addadmin', methods=['GET', 'POST'])
42 def addadmin():
43     form = RegistrationForm()
44     if form.validate_on_submit():
45         email =
   User.query.filter_by(email=form.email.data).first()
46         if email is None:
47             username =
   User.query.filter_by(username=form.username.data).first()
48             if username is None:
49                 register =
   User(firstname=form.firstname.data,
   lastname=form.lastname.data,
50                                      email=form.email.data,
   username=form.username.data, password=form.password.data,
51                                      is_admin=True)
52                 db.session.add(register)
53                 db.session.commit()
54                 flash('Congrats you have registered
   successfully!, Please check your inbox')
55                 welcome = form.firstname.data + ' ' +
   form.lastname.data + ' Welcome to Digital Payment Book'
56                 msg = Message(
57                     'Welcome',
58
   sender='pratikshaprojectmail@gmail.com',
59                     recipients=[form.email.data]
60                 )
61                 msg.body = welcome
```

```
62                 mail.send(msg)
63
64                 return redirect(url_for('login'))
65             else:
66                 flash('Username already exists, please try
   again!')
67                 return redirect(url_for('addadmin'))
68         else:
69             flash('Email already exists, please try logging
   in!')
70             return redirect(url_for('addadmin'))
71
72     return render_template('addadmin.html', form=form)
73
74
75 @app.route('/register', methods=['GET', 'POST'])
76 def register():
77     form = RegistrationForm()
78     if form.validate_on_submit():
79         email =
   User.query.filter_by(email=form.email.data).first()
80         if email is None:
81             username =
   User.query.filter_by(username=form.username.data).first()
82             if username is None:
83                 register =
   User(firstname=form.firstname.data,
   lastname=form.lastname.data,
84                                      email=form.email.data,
   username=form.username.data, password=form.password.data,
85                                      is_admin=False)
86                 db.session.add(register)
87                 db.session.commit()
88                 flash('Congrats you have registered
   successfully!, Please check your inbox')
89                 welcome = form.firstname.data + ' ' +
```

```
   form.lastname.data + ' Welcome to Digital Payment Book'
90                 msg = Message(
91                     'Welcome',
92
   sender='pratikshaprojectmail@gmail.com',
93                     recipients=[form.email.data]
94                 )
95                 msg.body = welcome
96                 mail.send(msg)
97
98                 return redirect(url_for('register'))
99             else:
100                 flash('Username already exists, please try
   again!')
101                 return redirect(url_for('register'))
102         else:
103             flash('Email already exists, please try
   logging in!')
104             return redirect(url_for('register'))
105
106     return render_template('register.html', form=form)
107
108
109 @app.route('/login', methods=['GET', 'POST'])
110 def login():
111     form = LoginForm()
112     if form.validate_on_submit():
113         user =
   User.query.filter_by(username=form.username.data).first()
114         if user is not None:
115             if user.check_password(form.password.data):
116                 # session['user'] = form.username.data
117                 login_user(user)
118                 if user.is_admin == True:
119                     return redirect(url_for('customers'))
120                 else:
121                     next = request.args.get('next')
```

```
                    if next == None or not next[0] == '/':
                        next = url_for('userpending')
                    return redirect(next)
        flash('Username/password not found!')
        return redirect(url_for('login'))
    return render_template('login.html', form=form)


@app.route('/customers', methods=['GET', 'POST'])
@login_required
def customers():
    user = current_user.username
    if current_user.is_admin == True:
        view_customers =
  User.query.filter_by(is_admin=False)
        return render_template('customers.html',
  view_customers=view_customers, user=user)
    else:
        return abort(404)


@app.route('/addpayment/<string:id>', methods=['GET',
  'POST'])
@login_required
def addpayment(id):
    user = current_user.username
    user_email =
  db.session.query(User.email).filter(User.id == id).first()
    uemail = user_email[0]
    payments = Ppay.query.filter_by(user_id=id).first()
    if payments:
        flash('Previous Amount Pending!')
        return redirect(url_for('pending'))
    else:
        form = AddPaymentForm()
        if form.validate_on_submit():
```

```
154             product = form.product.data
155             total_amount = form.total_amount.data
156             paid_amount = form.paid_amount.data
157             pending_amount = total_amount - paid_amount
158             if pending_amount == 0:
159                 pass
160             else:
161                 payment = Ppay(product=product,
  total_amount=total_amount, paid_amount=paid_amount,
162
  pending_amount=pending_amount, user_id=id)
163                 welcome ='We hope you loved shopping with
  us. Please pay your Pending amount as soon as possible.'
164                 msg = Message(
165                     'Alert',
166
  sender='pratikshaprojectmail@gmail.com',
167                     recipients=[uemail]
168                 )
169                 msg.body = welcome
170                 mail.send(msg)
171                 db.session.add(payment)
172                 db.session.commit()
173                 flash('Payment added successfuly')
174
175                 return redirect(url_for('customers'))
176         return render_template('addpayment.html',
  form=form, user=user)
177
178
179 @app.route('/pending')
180 @login_required
181 def pending():
182     user = current_user.username
183     if current_user.is_admin == True:
184         pending = Ppay.query.all()
185         return render_template('pending.html',
```

```
  pending=pending, user=user)
186     else:
187         return abort(404)
188 
189 @app.route('/history')
190 @login_required
191 def history():
192     user = current_user.username
193     if current_user.is_admin == True:
194         history = Hpay.query.all()
195         return render_template('history.html',
  history=history, user=user)
196     else:
197         return abort(404)
198 
199 
200 @app.route('/close/<string:id>', methods=['GET', 'POST'])
201 @login_required
202 def close(id):
203     if current_user.is_admin == True:
204         customer_email =
  db.session.query(User.email).filter(User.id == id).first()
205         total =
  db.session.query(Ppay.total_amount).filter(Ppay.user_id ==
  id).first()
206         product =
  db.session.query(Ppay.product).filter(Ppay.user_id ==
  id).first()
207         close = Hpay(customer_email=customer_email[0],
  product=product[0], amount=total[0], user_id=id)
208         db.session.add(close)
209         Ppay.query.filter_by(user_id=id).delete()
210         db.session.commit()
211         flash('Payment Closed!')
212         return redirect(url_for('customers'))
213     else:
214         return abort(404)
```

```
215
216
217 @app.route('/userpending')
218 @login_required
219 def userpending():
220     if current_user.is_admin == False:
221         user = current_user.username
222         user_id =
  db.session.query(User.id).filter(User.username ==
  user).first()
223         payments =
  Ppay.query.filter_by(user_id=user_id.id)
224         return render_template('userpending.html',
  payments=payments, user=user)
225     else:
226         return abort(404)
227
228
229 @app.route('/userhistory')
230 @login_required
231 def userhistory():
232     if current_user.is_admin == False:
233         user = current_user.username
234         user_id =
  db.session.query(User.id).filter(User.username ==
  user).first()
235         payments =
  Hpay.query.filter_by(user_id=user_id.id)
236         return render_template('userhistory.html',
  payments=payments, user=user)
237     else:
238         return abort(404)
239
240
241 @app.route('/logout')
242 @login_required
```

```
def logout():
    logout_user()
    return redirect(url_for('login'))


@app.route('/messages', methods=['GET', 'POST'])
@login_required
def messages():
    if current_user.is_admin == False:
        user = current_user.username
        form = MessageForm()
        admin =
  db.session.query(User.email).filter(User.is_admin ==
  True).first()
        admine = admin[0]
        if form.validate_on_submit():
            user = current_user.email
            issue = 'From: ' + user + 'Message: ' +
  form.messages.data
            msg = Message(
                'Issue',
                sender='pratikshaprojectmail@gmail.com',
                recipients=[admine]
            )
            msg.body = issue
            mail.send(msg)
            flash('Issue sent Successfully')
            return redirect(url_for('messages'))
        return render_template('messages.html', form=form,
  user=user)
    else:
        return abort(404)


if __name__ == "__main__":
    app.run(debug=True)
```

models.py

```
1  from werkzeug.security import
   generate_password_hash,check_password_hash
2  from flask_login import UserMixin
3  from flask_login import LoginManager
4  from flask_sqlalchemy import SQLAlchemy
5  from datetime import datetime
6
7  db = SQLAlchemy()
8  login_manager = LoginManager()
9
10 @login_manager.user_loader
11 def load_user(user_id):
12     return User.query.get(user_id)
13
14
15 class User(db.Model, UserMixin):
16     # table name (overridden)
17     __tablename__ = 'user'
18
19     # columns of table
20     id = db.Column(db.Integer, primary_key=True)
21     firstname = db.Column(db.String(100))
22     lastname = db.Column(db.String(100))
```

```
23      email = db.Column(db.String(150), unique=True, index=True)
24      username = db.Column(db.String(64), unique=True, index=True)
25      created_at = db.Column(db.DateTime, default=datetime.now)
26      password_hash = db.Column(db.String(128))
27      is_admin = db.Column(db.Boolean, default=False)
28
29      # customer_id = db.relationship('Customer', backref='user',
    lazy='dynamic')
30      pending_pay = db.relationship('Ppay', backref='user',
    lazy='dynamic')
31      pay_history = db.relationship('Hpay', backref='user',
    lazy='dynamic')
32
33      def __init__(self, firstname, lastname, email, username,
    password, is_admin):
34          self.firstname = firstname
35          self.lastname = lastname
36          self.email = email
37          self.username = username
38          self.password_hash = generate_password_hash(password)  #
    hashed the password
39          self.is_admin = is_admin
40
41      # function to check password
42      def check_password(self, password):
43          return check_password_hash(self.password_hash, password)
```

```
44
45
46 class Ppay(db.Model):
47     id = db.Column(db.Integer, primary_key=True)
48     product = db.Column(db.String(100))
49     total_amount = db.Column(db.Integer, nullable=True)
50     paid_amount = db.Column(db.Integer, nullable=True)
51     pending_amount = db.Column(db.Integer, nullable=True)
52     created_at = db.Column(db.DateTime, default=datetime.now)
53
54     user_id = db.Column(db.Integer, db.ForeignKey('user.id',
   ondelete='CASCADE'))
55
56     def __init__(self, product, total_amount, paid_amount,
   pending_amount, user_id):
57         self.product = product
58         self.total_amount = total_amount
59         self.paid_amount = paid_amount
60         self.pending_amount = pending_amount
61         self.user_id = user_id
62
63
64 class Hpay(db.Model):
65     id = db.Column(db.Integer, primary_key=True)
66     customer_email = db.Column(db.String(200))
```

```
    product = db.Column(db.String(200))
    amount = db.Column(db.Integer, nullable=True)
    closed_at = db.Column(db.DateTime, default=datetime.now)

    user_id = db.Column(db.Integer, db.ForeignKey('user.id',
ondelete='CASCADE'))

    def __init__(self, customer_email, product, amount, user_id):
        self.customer_email = customer_email
        self.product = product
        self.amount = amount
        self.user_id = user_id
```

forms.py

```
from flask_wtf import FlaskForm
from wtforms import (StringField, SubmitField, PasswordField,
IntegerField, TextAreaField)
from wtforms.validators import DataRequired, Email, EqualTo,
Length



class RegistrationForm(FlaskForm):
    firstname = StringField('First Name',
validators=[DataRequired(), Length(min=3, max=50)])
    lastname = StringField('Last Name',
validators=[DataRequired(), Length(min=3, max=50)])
    email = StringField('Email', validators=[DataRequired(),
Email()])
    username = StringField('Username',
```

```
    validators=[DataRequired(), Length(min=4, max=20,
    message='Username must be min 4 to max 20 characters!')])
13      password = PasswordField('Password',
    validators=[DataRequired(), EqualTo('pass_confirm',
    message='Passwords Must Match!')])
14      pass_confirm = PasswordField('Confirm password',
    validators=[DataRequired()])
15      submit = SubmitField('Register')
16
17  class LoginForm(FlaskForm):
18      username = StringField('Username',
    validators=[DataRequired()])
19      password = PasswordField('Password',
    validators=[DataRequired()])
20      submit = SubmitField('Login')
21
22
23  class AddPaymentForm(FlaskForm):
24      product = StringField('Product', validators=[DataRequired()])
25      paid_amount = IntegerField('Paid Amount',
    validators=[DataRequired()])
26      total_amount = IntegerField('Total Amount',
    validators=[DataRequired()])
27      submit = SubmitField('Submit')
28
29  class MessageForm(FlaskForm):
30      messages = TextAreaField('Messages',
    validators=[Length(max=200)])
31      submit = SubmitField('Send')
```