**PROJECT DEVELOPMENT PHASE**

**SPRINT - 2**

|  |  |
| --- | --- |
| DATE | 05 NOVEMBER 2022 |
| TEAM ID | PNT2022TMID11845 |
| PROJECT NAME | SMART FASHION RECOMMENDER APPLICATION |

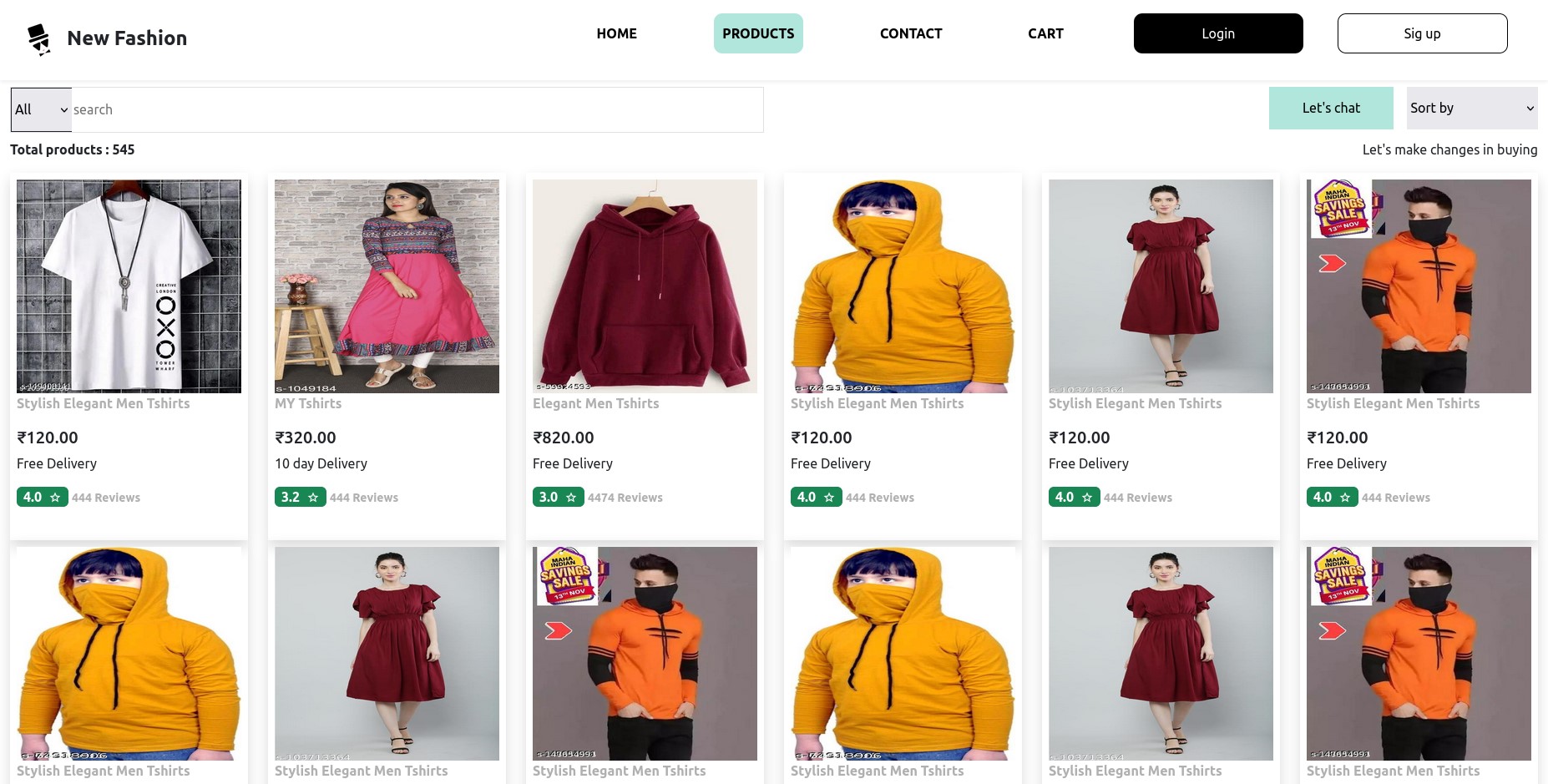
**PRODUCT LIST PAGE :**

|  |
| --- |
| <section class="container-fluid pt-3 pt-md-4"> |
| <!-- search filters --> |
| <div class="row"> |
| <div class="col-12 col-md-6"> |
| <div class="d-flex border">  <select name="category" class="border border-end-0 |
| border-dark py-3"> |
| <option value="all">All</option> |
| <option value="t-shirt">T-shirt</option> |
| <option value="t-shirt">T-shirt</option> |
| <option value="t-shirt">T-shirt</option> |
| </select>  <input type="text" placeholder="search" style="flex: 1;" |
| class="border-0 py-2"> |
| </div> |
| </div> |
| <div class="col-12 col-md-6"> |
| <div class="d-flex justify-content-around |
| justify-content-md-end mt-3 mt-md-0">  <!-- chat bot --> |
| <button class="chat-btn me-md-3">Let's chat</button> |
| <!-- sort by --> |
| <select name="sort" class="border-0 py-3"> |
| <option value="">Sort by</option> |
| <option value="low-high">Price: Low to High</option> |
| <option value="high-low">Price: High to Low</option>  </select> |
| </div> |
| </div> |
| </div> |
| <!-- product list items --> |
| <div class="row mt-2"> |
| <div class="col-12"> |
| <div class="d-flex justify-content-between"> |
| <p class="fw-bold">Total products : 545</p>  <p class="d-none d-md-block">Let's make changes in |
| buying</p> |
| </div> |
| <app-product-card |
| [product]="productDatails"></app-product-card> |
| </div> |
| <!-- pagination -->  <div class="col-12 mt-5 d-flex justify-content-center"> |
| <nav aria-label="Page navigation example"> |
| <ul class="pagination"> |
| <li class="page-item"><a class="page-link" |
| href="#">1</a></li> |
| <li class="page-item"><a class="page-link" |
| href="#">2</a></li>  <li class="page-item"><a class="page-link" |
| href="#">3</a></li> |
| <li class="page-item"><a class="page-link" |
| href="#">Next</a></li> |
| </ul> |
| </nav> |
| </div>  </div> |
| </section> |

CSS:

|  |
| --- |
| .chat-btn { |
| border: none; |
| padding: 0.7rem 2.5rem; |
| background-color: var(--primary-color);  } |
|  |
| .chat-btn:hover { |
| border: 3px solid var(--primary-color); |
| background-color: transparent; |
| } |
| .pagination .page-link { |
| color: black; |
| border: 3px solid var(--primary-color); |

OUTPUT SCREEN



**BACKEND API’S:**

**Login API**

|  |
| --- |
| from flask import Blueprint,jsonify,g,request |
| import ibm\_db |
| from passlib.hash import sha256\_crypt |
| import jwt |
|  |
| from ..lib import validation\_error from ..lib import exception |
| from ..lib import db |
|  |
| auth\_bp = Blueprint("auth",\_\_name\_\_) |
|  |
| @auth\_bp.route("/",methods=["GET"]) |
| def check(): |
| print(g.get("db")) |
| return jsonify({"msg":"hi"}) |
|  |
| @auth\_bp.route('/register',methods=['POST']) |
| def reg():  try: |
| data = request.get\_json() |
| name=data['name'] |

|  |
| --- |
| email=data['email'] |
| password=data['password'] |
| mobile\_no=data['mobileNo'] |
| print(email,password,name,mobile\_no) insert\_sql="INSERT INTO |
| USER(name,email,password,role,mobilenumber) VALUES(?,?,?,?,?)" |
| prep\_stmt = ibm\_db.prepare(db.get\_db(), insert\_sql) |
| ibm\_db.bind\_param(prep\_stmt,1,name) |
| ibm\_db.bind\_param(prep\_stmt,2,email) |
| ibm\_db.bind\_param(prep\_stmt,3,sha256\_crypt.encrypt(password)) |
| ibm\_db.bind\_param(prep\_stmt,4,"user") ibm\_db.bind\_param(prep\_stmt,5,mobile\_no) |
| ibm\_db.execute(prep\_stmt) |
| return {"message":'Created'},201 |
|  |
| except Exception as e: |
| return exception.handle\_exception(e) |
|  |
| @auth\_bp.route('/me',methods=['GET']) |
| def getMe(): |
| try: |
| token = request.headers['Authorization'] |
| if (not token): |
| return validation\_error.throw\_validation("Please login",401) decoded = jwt.decode(token,"secret",algorithms=["HS256"]) |
| select\_sql = "SELECT \* FROM USER WHERE ID=?" |
| prep\_stmt = ibm\_db.prepare(db.get\_db(), select\_sql) |
| ibm\_db.bind\_param(prep\_stmt,1,decoded['id']) |
| ibm\_db.execute(prep\_stmt) |
| isUser=ibm\_db.fetch\_assoc(prep\_stmt) |
| return isUser except Exception as e: |
| return exception.handle\_exception(e) |
|  |
| @auth\_bp.route('/login',methods=['POST']) |
| def auth\_log(): |
| try:  data = request.get\_json() |
| print(data) |
| email=data['email'] |
| password=data['password'] |
| select\_sql = "SELECT \* FROM USER WHERE EMAIL=?" |
| prep\_stmt = ibm\_db.prepare(db.get\_db(), select\_sql) |
| ibm\_db.bind\_param(prep\_stmt,1,email) |
| ibm\_db.execute(prep\_stmt)  isUser=ibm\_db.fetch\_assoc(prep\_stmt) |
| print(isUser) |
| if not isUser: |
| return validation\_error.throw\_validation("Invalid |
| Credentials",400) |
| if not sha256\_crypt.verify(password,isUser['PASSWORD']): |
| return validation\_error.throw\_validation("Invalid  Credentials",400) |
| encoded\_jwt = |
| jwt.encode({"id":isUser['ID'],"role":isUser['ROLE']},"secret",algorithm |
| ="HS256") |
| isUser["token"] = encoded\_jwt |
| return isUser |
| except Exception as e:  return exception.handle\_exception(e) |

**Category API**

|  |
| --- |
| from flask import Blueprint,request |
| import ibm\_db  from ..lib import exception |
| from ..lib import db |
|  |
| category\_bp = Blueprint("category",\_\_name\_\_) |
|  |
| @category\_bp.route("/",methods=["GET"]) |
| def get\_category(): |
| try: |
| select\_sql = "SELECT \* FROM CATEGORY WHERE" |
| prep\_stmt = ibm\_db.prepare(db.get\_db(), select\_sql) |
| ibm\_db.execute(prep\_stmt) |
| categories=[]  category=ibm\_db.fetch\_assoc(prep\_stmt) |
| while(category != False): |
| categories.append(category) |
| category = ibm\_db.fetch\_assoc(prep\_stmt) |
| print(categories) |
| return categories,200 |
| except Exception as e: |
| return exception.handle\_exception(e) |
| @category\_bp.route("/",methods=["POST"]) |
| def add\_category(): |
| try: |
| data = request.get\_json() |
| category = data['category'] |
| insert\_sql="INSERT INTO CATEGORY(category\_name) VALUES(?)" |
| prep\_stmt = ibm\_db.prepare(db.get\_db(), insert\_sql) ibm\_db.bind\_param(prep\_stmt,1,category) |
| ibm\_db.execute(prep\_stmt) |
| return {"message":'Created'},201 |
| except Exception as e: |
| return exception.handle\_exception(e) |
|  |
| @category\_bp.route("/<id>",methods=["DELETE"]) |
| def get\_category\_id(id): |
| try: |
| print(id) |
| select\_sql = "DELETE FROM CATEGORY WHERE ID=?" |
| prep\_stmt = ibm\_db.prepare(db.get\_db(), select\_sql) |
| ibm\_db.bind\_param(prep\_stmt,1,id) ibm\_db.execute(prep\_stmt) |
|  |
| return {"message":'Deleted'},200 |
| except Exception as e: |
| return exception.handle\_exception(e) |
|  |

|  |
| --- |
|  |
| from flask import Blueprint,request |
| import ibm\_db |
| from ..lib import exception |
| from ..lib import db |
|  |
| product\_bp = Blueprint("product",\_\_name\_\_) |
|  |

**Product API**

|  |
| --- |
| @product\_bp.route("/",methods=['POST']) |
| def add\_product(): |
| try: |
| data = request.get\_json() name=data['name'] |
| category=data['category'] |
| description = data['description'] |
| stock=data['stock'] |
| specificity = data['specificity'] |
| price = data['price'] |
| brand=data['brand'] insert\_sql="INSERT INTO |
| PRODUCT(product\_name,category,description,stock,specificity,price,brand |
| ) VALUES(?,?,?,?,?,?,?)" |
| prep\_stmt = ibm\_db.prepare(db.get\_db(), insert\_sql) |
| ibm\_db.bind\_param(prep\_stmt,1,name) |
| ibm\_db.bind\_param(prep\_stmt,2,category) |
| ibm\_db.bind\_param(prep\_stmt,3,description) ibm\_db.bind\_param(prep\_stmt,4,stock) |
| ibm\_db.bind\_param(prep\_stmt,5,specificity) |
| ibm\_db.bind\_param(prep\_stmt,6,price) |
| ibm\_db.bind\_param(prep\_stmt,7,brand) |
| ibm\_db.execute(prep\_stmt) |
| return {"message":'Created'},201 |
| except Exception as e:  return exception.handle\_exception(e) |
|  |
| @product\_bp.route("/",methods=['GET']) |
| def get\_product(): |
| try: |
| select\_sql = "SELECT PRODUCT.ID AS product\_id, |
| category,category\_name,product\_name,description,price,stock,image,brand ,specificity FROM PRODUCT JOIN CATEGORY ON |
| CATEGORY.ID=PRODUCT.CATEGORY" |
| prep\_stmt = ibm\_db.prepare(db.get\_db(), select\_sql) |
| ibm\_db.execute(prep\_stmt) |
| products=[] |
| product=ibm\_db.fetch\_assoc(prep\_stmt) |
| while(product != False): products.append(product) |
| product = ibm\_db.fetch\_assoc(prep\_stmt) |
| print(products) |
| return products or [],200 |

|  |
| --- |
| except Exception as e: |
| return exception.handle\_exception(e) |
|  |
| @product\_bp.route("/<id>",methods=['GET']) |
| def get\_product\_id(id): |
| try: |
| select\_sql = "SELECT PRODUCT.ID AS product\_id, |
| category,category\_name,product\_name,description,price,stock,image,brand |
| ,specificity FROM PRODUCT JOIN CATEGORY ON CATEGORY.ID=PRODUCT.CATEGORY |
| WHERE PRODUCT.ID=?" prep\_stmt = ibm\_db.prepare(db.get\_db(), select\_sql) |
| ibm\_db.bind\_param(prep\_stmt,1,id) |
| ibm\_db.execute(prep\_stmt) |
| product=ibm\_db.fetch\_assoc(prep\_stmt) |
| print(product) |
| return product or [],200 |
| except Exception as e:  return exception.handle\_exception(e) |
|  |
| @product\_bp.route("/<id>",methods=['PUT']) |
| def update\_product(id): |
| try: |
| data = request.get\_json() name=data['name'] |
| category=data['category'] |
| description = data['description'] |
| stock=data['stock'] |
| specificity = data['specificity'] |
| price = data['price'] |
| brand=data['brand']  insert\_sql="UPDATE PRODUCT SET |
| product\_name=?,category=?,description=?,stock=?,specificity=?,price=?,b |
| rand=? WHERE ID=?" |
| prep\_stmt = ibm\_db.prepare(db.get\_db(), insert\_sql) |
| ibm\_db.bind\_param(prep\_stmt,1,name) |
| ibm\_db.bind\_param(prep\_stmt,2,category) |
| ibm\_db.bind\_param(prep\_stmt,3,description) ibm\_db.bind\_param(prep\_stmt,4,stock) |
| ibm\_db.bind\_param(prep\_stmt,5,specificity) |
| ibm\_db.bind\_param(prep\_stmt,6,price) |
| ibm\_db.bind\_param(prep\_stmt,7,brand) |
| ibm\_db.bind\_param(prep\_stmt,8,id) |
| ibm\_db.execute(prep\_stmt) |
| return {"message":'Updated'},200 |
| except Exception as e:  return exception.handle\_exception(e) |
|  |
| @product\_bp.route("/<id>",methods=['DELETE']) |
| def delete\_product(id): |
| try: |
| insert\_sql="DELETE FROM PRODUCT WHERE ID=?" prep\_stmt = ibm\_db.prepare(db.get\_db(), insert\_sql) |
| ibm\_db.bind\_param(prep\_stmt,1,id) |
| ibm\_db.execute(prep\_stmt) |
| return {"message":'Deleted'},200 |
| except Exception as e: |
| return exception.handle\_exception(e) |

**Cart API**

|  |
| --- |
|  |
| from flask import Blueprint,request |
| import ibm\_db |
| from ..lib import validation\_error from ..lib.auth import check\_auth |
| from ..lib import exception |
| from ..lib import db |
|  |
| cart\_bp = Blueprint("cart",\_\_name\_\_) |
|  |
| @cart\_bp.route("/",methods=['POST']) |
| def add\_cart(): |
| try: |
| user\_id =check\_auth(request) |
| data=request.get\_json() |
| product=data['product']  select\_sql = "SELECT \* FROM PRODUCT WHERE ID=?" |
| prepare\_select =ibm\_db.prepare(db.get\_db(),select\_sql) |
| ibm\_db.bind\_param(prepare\_select,1,product) |
| ibm\_db.execute(prepare\_select) |
| is\_product = ibm\_db.fetch\_assoc(prepare\_select) |
|  |
| print(is\_product) |



|  |
| --- |
| @cart\_bp.route("/",methods=['GET']) |
| def get\_cart(): |
| try: |
| user\_id =check\_auth(request)  insert\_sql="SELECT PRODUCT.ID AS product\_id,cart\_id, |
| category,category\_name,product\_name,description,price,stock,image,brand |
| ,specificity,CART.user as user FROM CART JOIN PRODUCT ON |
| CART.PRODUCT=PRODUCT.ID JOIN CATEGORY ON PRODUCT.CATEGORY = CATEGORY.ID |
| WHERE CART.USER=?" |
| prep\_stmt = ibm\_db.prepare(db.get\_db(), insert\_sql) |
| ibm\_db.bind\_param(prep\_stmt,1,user\_id) |
| ibm\_db.execute(prep\_stmt) |
| products=[] |
| product=ibm\_db.fetch\_assoc(prep\_stmt) |
| while(product != False): |
| products.append(product) |
| product = ibm\_db.fetch\_assoc(prep\_stmt) print(products) |
| return products or [],200 |
|  |
| except Exception as e: |
| return exception.handle\_exception(e) |
|  |
| @cart\_bp.route("/<product>/<id>",methods=['DELETE']) |
| def delete\_cart(product,id): |
| try: |
| user\_id =check\_auth(request) |
| print(product,id,user\_id) |
|  |
| select\_sql = "SELECT \* FROM PRODUCT WHERE ID=?" prepare\_select =ibm\_db.prepare(db.get\_db(),select\_sql) |
| ibm\_db.bind\_param(prepare\_select,1,product) |
| ibm\_db.execute(prepare\_select) |
| is\_product = ibm\_db.fetch\_assoc(prepare\_select) |
|  |
| print(is\_product) |
| if not is\_product: |
| return validation\_error.throw\_validation("No Product found",404) |
|  |
| print("ff") |

insert\_sql

=

"DELETE FROM CART WHERE CART\_ID=? AND user=?"

prep\_stmt

=

ibm\_db

.prepare(

db

.

get\_db

()

,

insert\_sql

)

ibm\_db

.bind\_param(

prep\_stmt

,

1

,

id

)

ibm\_db

.bind\_param(

prep\_stmt

,

2

,

user\_id

)

ibm\_db

.execute(

prep\_stmt

)

print

(

"aa"

)

update\_sql

=

"UPDATE PRODUCT SET stock=? WHERE ID=?"

update\_stmt

=

ibm\_db

.prepare(

db

.

get\_db

()

,

update\_sql

)

ibm\_db

.bind\_param(

update\_stmt

,

1

,

is\_product

[

'STOCK'

]+

1

)

ibm\_db

.bind\_param(

update\_stmt

,

2

,

product

)

ibm\_db

.execute(

update\_stmt

)

return

{

"message"

:

'Deleted'

}

,

200

except

Exception

as

e

:

return

exception

.

handle\_exception

(

e

)

**Order API:**

|  |
| --- |
|  |
| from flask import Blueprint,request |
| import ibm\_db |
| from ..lib import exception from ..lib import db,auth |
|  |
| order\_bp = Blueprint("order",\_\_name\_\_) |
|  |
| @order\_bp.route("/",methods=['POST']) def add\_order(): |
| try: |
| user\_id =auth.check\_auth(request) |
| data=request.get\_json() |
| products=data['products'] |
| insert\_sql="SELECT ORDER\_ID FROM FINAL TABLE (INSERT INTO |
| ORDER(user) VALUES(?))" prep\_stmt = ibm\_db.prepare(db.get\_db(), insert\_sql) |
| ibm\_db.bind\_param(prep\_stmt,1,user\_id) |
| ibm\_db.execute(prep\_stmt) |
| order = ibm\_db.fetch\_assoc(prep\_stmt) |
| print(order) |
|  |
| for product in products: |

print

(

product

)

insert1\_sql

=

"INSERT INTO ORDERDETAIL(order,product)

VALUES(?,?)"

prep1\_stmt

=

ibm\_db

.prepare(

db

.

get\_db

()

,

insert1\_sql

)

ibm\_db

.bind\_param(

prep1\_stmt

,

1

,

order

[

'ORDER\_ID'

])

ibm\_db

.bind\_param(

prep1\_stmt

,

2

,

product

)

ibm\_db

.execute(

prep1\_stmt

)

return

{

"message"

:

'Created'

}

,

201

except

Exception

as

e

:

return

exception

.

handle\_exception

(

e

)

@

order\_bp

.route

(

"/<id>"

,

methods

=[

'GET'

])

def

get\_order

(

id

):

try

:

insert\_sql

=

"SELECT PRODUCT.ID AS product\_id,

category,category\_name,product\_name,description,price,stock,image,brand

,specificity,paid FROM ORDERDETAIL JOIN ORDER ON

ORDERDETAIL.ORDER=ORDER.ORDER\_ID JOIN PRODUCT ON

ORDERDETAIL.PRODUCT=PRODUCT.ID JOIN CATEGORY ON PRODUCT.CATEGORY =

CATEGORY.ID WHERE ORDER.USER=?"

prep\_stmt

=

ibm\_db

.prepare(

db

.

get\_db

()

,

insert\_sql

)

ibm\_db

.bind\_param(

prep\_stmt

,

1

,

id

)

ibm\_db

.execute(

prep\_stmt

)

products

=[]

product

=

ibm\_db

.fetch\_assoc(

prep\_stmt

)

while

(

product

!=

False

):

products

.

append

(

product

)

product

=

ibm\_db

.fetch\_assoc(

prep\_stmt

)

print

(

products

)

return

products

or

[]

,

200

except

Exception

as

e

:

return

exception

.

handle\_exception

(

e

)