**DAO**

1. **Interface- CustomerDao**

**Method- getCustomerById()**

Parameters- int id

Returns- void

Description- This method throws SQL exception and customer not found exception in case customer is not present in the database. It takes the customer id and matches it with the database.

**Method- getCustomerByName()**

Parameters- String name

Returns- void

Description- This method throws SQL exception and customer not found exception in case customer is not present in the database. It takes the customer name and matches it with the database.

**Method- loginUsingId()**

Parameters- int id, String password

Returns- void

Description- This method throws SQL exception and customer not found exception in case customer is not present in the database. It takes the customer id and password to allow the customer to login with the help of id.

**Method- loginUsingName()**

Parameters- String name, String password

Returns- void

Description- This method throws SQL exception and customer not found exception in case customer is not present in the database. It takes the customer name and password to allow the customer to login with the help of customer name.

1. **Class CustomerDaoImpl implements CustomerDao**

**Method- getCustomerById()**

Parameters- int id

Returns- Customer object

Description- This method throws SQL exception and customer not found exception in case customer is not present in the database. It takes the customer id and matches it with the database.

**Method- getCustomerByName()**

Parameters- String name

Returns- Customer object

Description- This method throws SQL exception and customer not found exception in case customer is not present in the database. It takes the customer name and matches it with the database.

**Method- loginUsingId()**

Parameters- int id, String password

Returns- Customer object

Description- This method throws SQL exception and customer not found exception in case customer is not present in the database. It takes the customer id and password to allow the customer to login with the help of id.

**Method- loginUsingName()**

Parameters- String name, String password

Returns- Customer object

Description- This method throws SQL exception and customer not found exception in case customer is not present in the database. It takes the customer name and password to allow the customer to login with the help of customer name.

1. **Class DBUtil**

**Method-** getConnection()

Parameters- None

Returns- Connection object

Description- This method is helpful in establishing connection to the mysql database.

1. **Interface EmployeeDao**

**Method- loginUsingId()**

Parameters- int id, String password

Returns- void

Description- This method throws SQL exception and customer not found exception in case customer is not present in the database. It takes the employee id and password to allow the employee to login with the help of id.

1. **Class EmployeeDaoImpl implements EmployeeDao**

**Method- loginUsingId()**

Parameters- int id, String password

Returns- Customer object

Description- This method throws SQL exception and customer not found exception in case customer is not present in the database. It takes the employee id and password to allow the employee to login with the help of id.

1. **Interface InvoiceDao**

**Method- fetchInvoiceByOrderId()**

Parameters- int order\_id

Returns- void

Description- This method throws SQL exception and invoice not found exception in case invoice is not present in the database. It retrieves the invoice in exchange for order id.

**Method- addInvoice()**

Parameters- Date invoiceDate, int order\_id, int customer\_id, GSTType gst\_type, float total\_value, InvoiceStatus status

Returns- void

Description- It takes invoice date, order id, customer id, GST type, total value and invoice status to add invoice. This method throws SQL exception.

1. **Class InvoiceDaoImpl implements InvoiceDao**

**Method- fetchInvoiceByOrderId()**

Parameters- int order\_id

Returns- Invoice object

Description- This method throws SQL exception and invoice not found exception in case invoice is not present in the database. It retrieves the invoice in exchange for order id.

**Method- addInvoice()**

Parameters- Date invoiceDate, int order\_id, int customer\_id, GSTType gst\_type, float total\_value, InvoiceStatus status

Returns- Invoice object

Description- It takes invoice date, order id, customer id, GST type, total value and invoice status to add invoice. This method throws SQL exception.

1. **Interface OrderDao**

**Method- getAllOrdersWithoutProductList()**

Parameters- None

Returns- void

Description- This method retrieves all orders without list of products. It throws SQL exception as well.

**Method- getOrdersWithoutProductListByCustomerId()**

Parameters- int id

Returns- void

Description- This method retrieves all orders without list of products with the help of customer id. It throws SQL exception as well.

**Method- getQuotesWithoutProductListByCustomerId()**

Parameters- int id

Returns- void

Description- This method retrieves quote list for customer without list of products with the help of customer id. It throws SQL exception as well.

**Method- addQuote()**

Parameters- date order\_date, int customer\_id, String customer\_shipping\_address, float total\_order\_value, float shipping\_cost

Returns- void

Description- This method inserts new quote in the database. It throws SQL exception as well.

**Method- addOrderHasProducts()**

Parameters- Map<Integer,Integer> productMap

Returns- void

Description- This method inserts into OrderHasproducts() in the database. It throws SQL exception as well.

**Method- getOrderById()**

Parameters- int order\_id

Returns- void

Description- This method retrieves order with the help of id in the database. It throws SQL exception and order not found exception.

**Method- approveOrder()**

Parameters- int order\_id

Returns- void

Description- This method sets the order status to approve and takes order id in exchange. It throws SQL exception.

**Method- expireOrder()**

Parameters- int order\_id

Returns- void

Description- This method changes the status to expires if order approval fails and takes order id in exchange. It throws SQL exception.

1. **class OrderDaoImpl implements OrderDao**

**Method- getAllOrdersWithoutProductList()**

Parameters- None

Returns- List containing order details

Description- This method retrieves all orders without list of products. It throws SQL exception as well.

**Method- getOrdersWithoutProductListByCustomerId()**

Parameters- int id

Returns- List containing order details

Description- This method retrieves all orders without list of products with the help of customer id. It throws SQL exception as well.

**Method- getQuotesWithoutProductListByCustomerId()**

Parameters- int id

Returns- List containing order details

Description- This method retrieves quote list for customer without list of products with the help of customer id. It throws SQL exception as well.

**Method- addQuote()**

Parameters- date order\_date, int customer\_id, String customer\_shipping\_address, float total\_order\_value, float shipping\_cost

Returns- Order id

Description- This method inserts new quote in the database. It throws SQL exception as well.

**Method- addOrderHasProducts()**

Parameters- Map<Integer,Integer> productMap

Returns- void

Description- This method inserts into OrderHasproducts() in the database. It throws SQL exception as well.

**Method- getOrderById()**

Parameters- int order\_id

Returns- Object of order class

Description- This method retrieves order with the help of id in the database. It throws SQL exception and order not found exception.

**Method- approveOrder()**

Parameters- int order\_id

Returns- void

Description- This method sets the order status to approve and takes order id in exchange. It throws SQL exception.

**Method- expireOrder()**

Parameters- int order\_id

Returns- void

Description- This method changes the status to expires if order approval fails and takes order id in exchange. It throws SQL exception.

1. **Interface ProductDao**

**Method- fetchAllProducts()**

Parameters- none

Returns- void

Description- This method retrieves all product details. It throws SQL exception as well.

**Method- addProducts()**

Parameters- Set<Product> products

Returns- void

Description- This method retrieves adds product details to the database. It throws SQL exception as well.

1. **Class ProductDaoImpl implements ProductDao**

**Method- fetchAllProducts()**

Parameters- none

Returns- List of products and details

Description- This method retrieves all product details. It throws SQL exception as well.

**Method- addProducts()**

Parameters- Set<Product> products

Returns- Update count value

Description- This method retrieves adds product details to the database. It throws SQL exception as well.