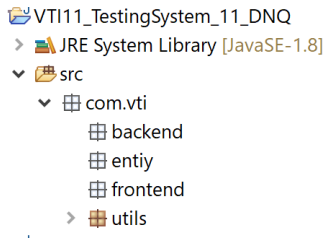
Exercise 1: Basic

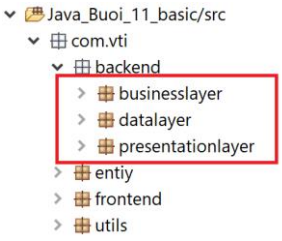
Demo trên bảng dữ liệu Group với mô hình 3 Layer

Tạo mới 1 project: VTI11_TestingSystem_11_DNQ

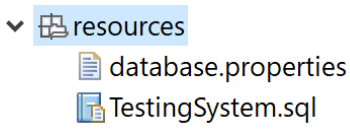
Tạo các Package trong src: com.vti.backend, com.vti.entiy, com.vti.frontend, com.vti.utils



Trong backend tạo lần lượt các package: com.vti.backend.businesslayer, com.vti.backend.datalayer, com.vti.backend.presentationlayer



Tạo Package com.vti.resources:



Tạo file: database.properties

```
# ----------------------
# database information
# ----------------------
url         =
jdbc:mysql://localhost:3306/TestingSystem?autoReconnect=true&useSSL=false&characterEncoding=latin1
username    = root
password    = root
driver      = com.mysql.cj.jdbc.Driver
```

Tạo file: TestingSystem.sql

TestingSystem.sql

Tạo Class **JdbcUltis** trong package ultis:

```
public class JdbcUltis {
        private Properties property;
        private Connection connection;

        public JdbcUltis() throws FileNotFoundException, IOException {
                property = new Properties();
                property.load(new FileInputStream("C:\\Users\\daonq\\eclipse-
workspace\\VTI11_TestingSystem_11_DNQ\\src\\com\\vti\\resources\\database.properties"));
        }

        public void connnectionTestting() throws ClassNotFoundException, SQLException {
```

```
                String url = property.getProperty("url");
                String Username = property.getProperty("username");
                String password = property.getProperty("password");
                String dirver = property.getProperty("driver");

                Class.forName(dirver);
                connection = DriverManager.getConnection(url, Username, password);
                System.out.println("Connect Success");
        }

        public Connection getConnection() throws ClassNotFoundException, SQLException {
                String url = property.getProperty("url");
                String Username = property.getProperty("username");
                String password = property.getProperty("password");
                String dirver = property.getProperty("driver");
                Class.forName(dirver);
                connection = DriverManager.getConnection(url, Username, password);
                return connection;
        }

        public void disConnection() throws SQLException {
                connection.close();
        }

        public ResultSet executeQuery(String sql) throws ClassNotFoundException, SQLException {
                Connection connnection = getConnection();
                Statement statement = connnection.createStatement();
                ResultSet result = statement.executeQuery(sql);
                return result;
        }

        public PreparedStatement createPrepareStatement(String sql) throws ClassNotFoundException,
SQLException {
                Connection connnection = getConnection();
                PreparedStatement preStatement = connnection.prepareStatement(sql);
                return preStatement;
        }

}
```

Tạo Class Group trong Entity:

```
public class Group {
        private int id;
        private String name;

        public Group(int id, String name) {
                this.id = id;
                this.name = name;
        }

        public int getId() {
                return id;
        }

        public String getName() {
                return name;
        }

        @Override
        public String toString() {
                return "Id: " + id + " / Name: " + name;
        }

}
```

Tạo Class GroupRepository trong datalayer:

```
public class GroupRepository {
```

```
	private JdbcUltis jdbc;

	public GroupRepository() throws FileNotFoundException, IOException {
		jdbc = new JdbcUltis();
	}

	public List<Group> getListGroups() throws ClassNotFoundException, SQLException {

		List<Group> groups = new ArrayList<>();
		// Create a statement object
		String sql = "SELECT * FROM `Group`";
		// execute query
		ResultSet resultSet = jdbc.executeQuery(sql);
		// handling result set
		while (resultSet.next()) {
			Group group = new Group(resultSet.getInt("GroupID"),
resultSet.getString("GroupName"));
			groups.add(group);
		}
		jdbc.disConnection();
		return groups;
	}
}
```

Tạo Class `GroupService` trong businesslayer:

```
public class GroupService {

	private GroupRepository repository;

	public GroupService() throws FileNotFoundException, IOException {
		repository = new GroupRepository();
	}

	public List<Group> getListGroups() throws ClassNotFoundException,
			SQLException {
		// logic service
		return repository.getListGroups();
	}
}
```

Tạo Class `GroupController` trong presentationlayer:

```
public class GroupController {

	private GroupService groupService;

	public GroupController() throws FileNotFoundException, IOException {
		groupService = new GroupService();
	}

	public List<Group> getListGroups() throws ClassNotFoundException,
			SQLException {
		// validation TODO
		return groupService.getListGroups();
	}
}
```

Tạo Class `Demo1` trong frontend:

```
public class Demo1 {
	public static void main(String[] args)
			throws FileNotFoundException, IOException, ClassNotFoundException,
SQLException {
//		JdbcUltis jdbc1 = new JdbcUltis();
//		jdbc1.connnectionTestting();
		GroupController controller = new GroupController();
		List<Group> groups = controller.getListGroups();
```

```
                for (Group group : groups) {
                        System.out.println(group);
                }
        }
}
```

Question 1: (Sử dụng Database Testing System đã xây dựng ở SQL)
Tạo structure 3-tiers.
Gợi ý:
Tạo project maven & config Jdbc library
Tạo 3 package: frontend, backend, database, utils
Trong package utils copy hết các file IOManager, FileManager,
ScannerUtils, JDBCUtils, …

Question 2: tiếp tục question 1 (entity)
Trong package entity tạo các class: Account, Group, …
Chú ý: config theo Object hướng đối tượng như bài 1 Java (không config groupID như trong SQL)

Question 3: tiếp tục question 1 (backend)
Trong package backend tạo 3 subpackage: **presentationlayer, businesslayer, datalayer**.
a) Trong subpackage datalayer tạo class AccountRepository, trong class tạo method getListAccounts() để lấy tất cả danh sách account từ database
Gợi ý: sử dụng JDBC để lấy tất cả danh sách account từ database
b) Trong subpackage businesslayer tạo ra class AccountService, trong class tạo method **getListAccounts()** và return ra **List<Account>**
Gợi ý: trong method getListAccounts() sẽ gọi tới method getListAccounts() của class AccountRepository
c) Trong subpackage presentationlayer tạo ra class AccountController, trong class tạo method **getListAccounts()** và return ra **List<Account>**
Gợi ý: trong method getListAccounts() sẽ gọi tới method getListAccounts() của class AccountService

Question 4:
Trong package frontend tạo class Program.java demo method trên

Exercise 2: Basic (interface)

Question 1: tiếp tục Exercise 1
Để tránh phụ thuộc giữa các package ở backend ta sẽ tạo interface cho các service và repository (đặt tên như trong hướng dẫn slide).

Question 2:
Tương tự tạo các chức năng createAccount, getListAccounts, getAccountByID(int id), isAccountExists(String username), isAccountExists(int id), updateAccountByID(int id), deleteAccount(int id)

Demo trên bảng dữ liệu Account, Department, Position với mô hình 3 Layer

Tạo các class trong Package Ultis

Tạo class jdbcUltis:

```
package com.vti.ultis;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
```

```java
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Properties;

public class jdbcUltis {
	private Properties property;
	private Connection connection;

	public jdbcUltis() throws FileNotFoundException, IOException {
		property = new Properties();
		property.load(new FileInputStream("G:\\My Drive\\DNQ_VTI\\2. JAVA CORE\\DAONQ_TL\\Source
Code\\TestingSystem\\10.TesttingSystem_Assignment_10_DNQ\\src\\com\\vti\\resources\\database.properties"));
	}

	public void connnectionTestting() throws ClassNotFoundException, SQLException {
		String url = property.getProperty("url");
		String Username = property.getProperty("username");
		String password = property.getProperty("password");
		String dirver = property.getProperty("driver");

		Class.forName(dirver);
		connection = DriverManager.getConnection(url, Username, password);
		System.out.println("Connect Success");
	}

	public Connection getConnection() throws ClassNotFoundException, SQLException {
		String url = property.getProperty("url");
		String Username = property.getProperty("username");
		String password = property.getProperty("password");
		String dirver = property.getProperty("driver");
		Class.forName(dirver);
		connection = DriverManager.getConnection(url, Username, password);
		return connection;
	}

	public void disConnection() throws SQLException {
		connection.close();
	}

	public ResultSet executeQuery(String sql) throws ClassNotFoundException, SQLException {
		Connection connnection = getConnection();
		Statement statement = connnection.createStatement();
		ResultSet result = statement.executeQuery(sql);
		return result;
	}

	public PreparedStatement createPrepareStatement(String sql) throws ClassNotFoundException, SQLException {
		Connection connnection = getConnection();
		PreparedStatement preStatement = connnection.prepareStatement(sql);
		return preStatement;
	}

}
```

Tạo class ScannerUltis:

```java
package com.vti.ultis;

import java.text.SimpleDateFormat;
import java.time.LocalDate;
import java.util.Scanner;

public class ScannerUltis {
	private static Scanner sc = new Scanner(System.in);

	public static int inputInt() {
		while (true) {
			try {
				return Integer.parseInt(sc.next().trim());
			} catch (Exception e) {
				System.err.println("Nhập lại:");
			}
		}
	}

	public static int inputIntPositive() {
		while (true) {
			try {
				int intPositive = Integer.parseInt(sc.next());
				if (intPositive >= 0) {
					return intPositive;
				} else {
					System.err.println("Nhập lại:");
				}

			} catch (Exception e) {
				System.err.println("Nhập lại:");
			}

		}
```

```
	}

	public static Float inputFloat() {
		while (true) {
			try {
				return Float.parseFloat(sc.next());
			} catch (Exception e) {
				System.err.println("Nhập lại:");
			}
		}
	}

	public static Double inputDouble() {
		while (true) {
			try {
				return Double.parseDouble(sc.next());
			} catch (Exception e) {
				System.err.println("Nhập lại:");
			}
		}
	}

	public static String inputString() {
		while (true) {
			String string = sc.next().trim();
			if (!string.isEmpty()) {
				return string;
			} else {
				System.err.println("Nhập lại:");
			}
		}
	}

	public static LocalDate inputLocalDate() {
		System.out.println("Nhập theo định dạng yyyy-MM-dd");
		SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd");
		while (true) {
			String localdate = sc.next().trim();
			try {
				if (format.parse(localdate) != null) {
					LocalDate lc = LocalDate.parse(localdate);
					return lc;
				}
			} catch (Exception e) {
				System.err.println("Nhập lại:");
			}

		}
	}

	public static String inputEmail() {
		while (true) {
			String email = ScannerUltis.inputString();
			if (email == null || !email.contains("@")) {
				System.out.print("Nhập lại: ");
			} else {
				return email;
			}
		}
	}

	public static int inputFunction(int a, int b, String errorMessage) {
		while (true) {
			int number = ScannerUltis.inputInt();
			if (number >= a && number <= b) {
				return number;
			} else {
				System.err.println(errorMessage);
			}
		}
	}

	public static String inputPassword() {
		while (true) {
			String password = ScannerUltis.inputString();
			if (password.length() < 6 || password.length() > 12) {
				System.out.print("Nhập lại: ");
				continue;
			}

			boolean hasAtLeast1Character = false;

			for (int i = 0; i < password.length(); i++) {
				if (Character.isUpperCase(password.charAt(i)) == true) {
					hasAtLeast1Character = true;
					break;
				}
			}
			if (hasAtLeast1Character == true) {
				return password;
			} else {
				System.out.print("Mời bạn nhập lại password: ");
			}
		}
	}

	public static String inputPhoneNumber() {
		while (true) {
			String number = ScannerUltis.inputString();
```

```
                                if (number.length() > 12 || number.length() < 9) {
                                        continue;
                                }

                                if (number.charAt(0) != '0') {
                                        continue;
                                }

                                boolean isNumber = true;

                                for (int i = 0; i < number.length(); i++) {
                                        if (Character.isDigit(number.charAt(i)) == false) {
                                                isNumber = false;
                                                break;
                                        }
                                }
                                if (isNumber == true) {
                                        return number;
                                } else {
                                        System.out.print("Nhập lại: ");
                                }

                        }
                }

}
```

Tạo các class trong Package ultis.properties

Tạo class DatabaseProperties:

```
package com.vti.ultis.properties;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.Properties;

public class DatabaseProperties {

        public static final String RESOURCE_FOLDER_URL = "C:\\Users\\pc\\eclipse-
workspace\\TestingSystem_Assignment_11_Exercise_3_4\\src\\main\\java\\resources\\";

        private Properties properties;

        public DatabaseProperties() throws FileNotFoundException, IOException {
                properties = new Properties();
                properties.load(new FileInputStream(RESOURCE_FOLDER_URL + "database.properties"));
        }

        public String getProperty(String key) {
                return properties.getProperty(key);
        }
}
```

Tạo các class trong Package resources

Tạo file database.properties:

```
# ----------------------
# database information
# ----------------------
url         = jdbc:mysql://localhost:3306/TestingSystem?autoReconnect=true&useSSL=false&characterEncoding=latin1
username    = root
password    = root
driver      = com.mysql.cj.jdbc.Driver
```

Tạo file TestingSystem_DB.sql:

TestingSystem_DB.sql

Tạo các class trong Package Entity

Tạo Class Account trong Entity:

```
package com.vti.entiy;

import java.time.LocalDate;
```

```java
public class Account {
	private int id;
	private String email;
	private String username;
	private String fullName;
	private Department department;
	private Position position;
	private LocalDate createDate;

	public int getId() {
		return id;
	}

	public String getEmail() {
		return email;
	}

	public String getDepartment() {
		return department.getName();
	}

	public void setId(int id) {
		this.id = id;
	}

	public String getPosition() {
		return position.getName();
	}

	public void setPosition(Position position) {
		this.position = position;
	}

	public void setID(int iD) {
		id = iD;
	}

	public LocalDate getCreateDate() {
		return createDate;
	}

	public void setCreateDate(LocalDate createDate) {
		this.createDate = createDate;
	}

	public void setEmail(String email) {
		this.email = email;
	}

	public String getUsername() {
		return username;
	}

	public void setUsername(String username) {
		this.username = username;
	}

	public String getFullName() {
		return fullName;
	}

	public void setFullName(String fullName) {
		this.fullName = fullName;
	}

	public void setDepartment(Department department) {
		this.department = department;
	}

	@Override
	public String toString() {
		return "Account [ID=" + id + ", email=" + email + ", Username=" + username + ",
FullName=" + fullName
```

```
                                + ", department=" + department + ", position=" + position + ",
CreateDate=" + createDate + "]";
        }

}
```

Tạo Class Department trong Entity:

```
package com.vti.entiy;

public class Department {
        private int id;
        private String name;

        public Department(int id, String name) {
                this.id = id;
                this.name = name;
        }

        public int getId() {
                return id;
        }

        public void setId(int id) {
                this.id = id;
        }

        public String getName() {
                return name;
        }

        public void setName(String name) {
                this.name = name;
        }

        @Override
        public String toString() {
                return name;
        }

}
```

Tạo Class Position trong Entity:

```
package com.vti.entiy;

public class Position {
        private int id;
        private String name;

        public int getId() {
                return id;
        }

        public void setId(int id) {
                this.id = id;
        }

        public String getName() {
                return name;
        }

        public void setName(String name) {
                this.name = name;
        }

        public Position(int id, String name) {
                super();
                this.id = id;
                this.name = name;
```

```
	}

	@Override
	public String toString() {
		return "Position [id=" + id + ", name=" + name + "]";
	}

}
```

Tạo các class trong Package datalayer

Tạo Interface IAccountRepository trong datalayer:

```
package com.vti.backend.datalayer;

import java.io.FileNotFoundException;
import java.io.IOException;
import java.sql.SQLException;
import java.util.List;

import com.vti.entiy.Account;

public interface IAccountRepository {

	public List<Account> getListAccounts()
			throws FileNotFoundException, ClassNotFoundException, SQLException,
IOException;

	public Account getAccountByID(int id)
			throws SQLException, ClassNotFoundException, FileNotFoundException,
IOException;

	public Boolean isAccNameExists(String name) throws ClassNotFoundException, SQLException;

	public boolean isAccIdExists(int id) throws SQLException, ClassNotFoundException;

	public boolean createAccount(Account acc, int depId, int posId) throws SQLException,
ClassNotFoundException;

	public boolean delAccByID(int ID) throws SQLException, ClassNotFoundException;

	public boolean updateByEmai(int id, String newEmail) throws ClassNotFoundException,
SQLException;

	public boolean updateByUserName(int id, String newUserName) throws SQLException,
ClassNotFoundException;

	public boolean updateByFullName(int id, String newFullName) throws SQLException,
ClassNotFoundException;

	public boolean updateByDepId(int id, int idDep) throws ClassNotFoundException,
SQLException;

	public boolean updateByPosId(int id, int idPos) throws SQLException,
ClassNotFoundException;
}
```

Tạo Class IDepartmentRepository trong datalayer:

```
package com.vti.backend.datalayer;

import java.sql.SQLException;
import java.util.List;

import com.vti.entiy.Department;

public interface IDepartmentRepository {

	public List<Department> getListDepartment() throws SQLException, ClassNotFoundException;

	public Department getDepByID(int id) throws SQLException, ClassNotFoundException;

	public Boolean isDepartmentNameExists(String name) throws SQLException,
ClassNotFoundException;
```

```
        public boolean createDep(String name) throws ClassNotFoundException, SQLException;

        public boolean updateDepartmentName(int id, String newName) throws ClassNotFoundException,
SQLException;

        public boolean delDepByID(int id) throws ClassNotFoundException, SQLException;

}
```

Tạo Class IPossitionRepository trong datalayer:

```
package com.vti.backend.datalayer;

import java.sql.SQLException;
import java.util.List;

import com.vti.entiy.Position;

public interface IPossitionRepository {
        public List<Position> getListPosition() throws ClassNotFoundException, SQLException;

        public Position getPosByID(int id) throws ClassNotFoundException, SQLException;
}
```

Tạo Class PossitionRepository trong datalayer:

```
package com.vti.backend.datalayer;

import java.io.FileNotFoundException;
import java.io.IOException;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

import com.vti.entiy.Position;
import com.vti.ultis.jdbcUltis;

public class PossitionRepository implements IPossitionRepository {
        private jdbcUltis jdbc;

        public PossitionRepository() throws FileNotFoundException, IOException {
                jdbc = new jdbcUltis();
        }

        @Override
        public List<Position> getListPosition() throws ClassNotFoundException, SQLException {
                List<Position> listPosition = new ArrayList<Position>();
                String sql = "SELECT * FROM position";
                ResultSet resultSet = jdbc.executeQuery(sql);
                while (resultSet.next()) {
                        Position pos = new Position(resultSet.getInt(1), resultSet.getString(2));
                        listPosition.add(pos);
                }
                jdbc.disConnection();
                return listPosition;
        }

        @Override
        public Position getPosByID(int id) throws ClassNotFoundException, SQLException {
                String sql = "SELECT * FROM position WHERE PositionID = ?";
                PreparedStatement preStatement = jdbc.createPrepareStatement(sql);
                preStatement.setInt(1, id);
                ResultSet result = preStatement.executeQuery();
                if (result.next()) {
                        Position pos = new Position(result.getInt(1), result.getString(2));
                        return pos;
                } else {
                        jdbc.disConnection();
                        return null;
```

```
                }

        }

}
```

Tạo Class AccountRepository trong datalayer:

```
package com.vti.backend.datalayer;

import java.io.FileNotFoundException;
import java.io.IOException;
import java.sql.Date;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.time.LocalDate;
import java.util.ArrayList;
import java.util.List;

import com.vti.entiy.Account;
import com.vti.entiy.Department;
import com.vti.entiy.Position;
import com.vti.ultis.jdbcUltis;

public class AccountRepository implements IAccountRepository {
        private jdbcUltis jdbc;

        public AccountRepository() throws FileNotFoundException, IOException {
                jdbc = new jdbcUltis();
        }

        @Override
        public List<Account> getListAccounts()
                        throws FileNotFoundException, ClassNotFoundException, SQLException,
IOException {
                String sql = "SELECT * FROM account ORDER BY AccountID";
                ResultSet resultSet = jdbc.executeQuery(sql);
                List<Account> listAcc = new ArrayList<Account>();
                while (resultSet.next()) {
                        Account acc = new Account();
                        acc.setID(resultSet.getInt(1));
                        acc.setEmail(resultSet.getString(2));
                        acc.setUsername(resultSet.getString(3));
                        acc.setFullName(resultSet.getString(4));

                        IDepartmentRepository depRepository = new DepartmentRepository();
                        Department dep = depRepository.getDepByID(resultSet.getInt(5));
                        acc.setDepartment(dep);

                        IPossitionRepository posDao = new PossitionRepository();
                        Position pos = posDao.getPosByID(resultSet.getInt(6));
                        acc.setPosition(pos);

                        LocalDate lcd = resultSet.getDate(7).toLocalDate();
                        acc.setCreateDate(lcd);

                        listAcc.add(acc);
                }
                return listAcc;
        }

        @Override
        public Account getAccountByID(int id)
                        throws SQLException, ClassNotFoundException, FileNotFoundException,
IOException {
                String sql = "SELECT * FROM account WHERE AccountID = ?";
                PreparedStatement preStatement = jdbc.createPrepareStatement(sql);
                preStatement.setInt(1, id);
                ResultSet resultSet = preStatement.executeQuery();
                if (resultSet.next()) {
                        Account acc = new Account();
                        acc.setID(resultSet.getInt(1));
                        acc.setEmail(resultSet.getString(2));
```

```
                        acc.setUsername(resultSet.getString(3));
                        acc.setFullName(resultSet.getString(4));

                        IDepartmentRepository depDao = new DepartmentRepository();
                        Department dep = depDao.getDepByID(resultSet.getInt(5));
                        acc.setDepartment(dep);

                        IPossitionRepository posDao = new PossitionRepository();
                        acc.setPosition(posDao.getPosByID(resultSet.getInt(6)));

                        LocalDate lcd =
Date.valueOf(resultSet.getDate(7).toString()).toLocalDate();
                        acc.setCreateDate(lcd);
                        return acc;
                } else {
                        jdbc.disConnection();
                        return null;
                }

        }

        @Override
        public Boolean isAccNameExists(String name) throws ClassNotFoundException, SQLException {
                String sql = "SELECT * FROM account WHERE Username = ?";
                PreparedStatement preStatement = jdbc.createPrepareStatement(sql);
                preStatement.setNString(1, name);
                ResultSet result = preStatement.executeQuery();

                if (result.next()) {
                        jdbc.disConnection();
                        return true;
                } else {
                        jdbc.disConnection();
                        return false;
                }
        }

        @Override
        public boolean createAccount(Account acc, int depId, int posId) throws SQLException,
ClassNotFoundException {
                String sql = "INSERT INTO account (Email, Username, FullName, DepartmentID,
PositionID, CreateDate) VALUES (?, ?, ?,?,?,now());";

                PreparedStatement preStatement = jdbc.createPrepareStatement(sql);
                preStatement.setNString(1, acc.getEmail());
                preStatement.setNString(2, acc.getUsername());
                preStatement.setNString(3, acc.getFullName());
                preStatement.setInt(4, depId);
                preStatement.setInt(5, posId);
//              preStatement.setDate(6, java.util.Date.parse(LocalDate.now().toString()) );
                int result = preStatement.executeUpdate();
                if (result == 1) {
                        jdbc.disConnection();
                        return true;
                } else {
                        jdbc.disConnection();
                        return false;
                }

        }

        @Override
        public boolean delAccByID(int ID) throws SQLException, ClassNotFoundException {
                String sql = "DELETE FROM account WHERE (AccountID = ?);";
                PreparedStatement preStatement = jdbc.createPrepareStatement(sql);
                preStatement.setInt(1, ID);
                int result = preStatement.executeUpdate();
                if (result == 1) {
                        jdbc.disConnection();
                        return true;
                } else {
                        jdbc.disConnection();
                        return false;
                }
```

```
	}

	@Override
	public boolean updateByEmai(int id, String newEmail) throws ClassNotFoundException,
SQLException {
		String sql = "UPDATE account SET Email = ? WHERE (AccountID = ?);";
		PreparedStatement preStatement = jdbc.createPrepareStatement(sql);
		preStatement.setString(1, newEmail);
		preStatement.setInt(2, id);
		int result = preStatement.executeUpdate();
		if (result == 1) {
			jdbc.disConnection();
			return true;
		} else {
			jdbc.disConnection();
			return false;
		}
	}

	@Override
	public boolean updateByUserName(int id, String newUserName) throws SQLException,
ClassNotFoundException {
		String sql = "UPDATE account SET Username = ? WHERE (AccountID = ?);";
		PreparedStatement preStatement = jdbc.createPrepareStatement(sql);
		preStatement.setString(1, newUserName);
		preStatement.setInt(2, id);
		int result = preStatement.executeUpdate();
		if (result == 1) {
			jdbc.disConnection();
			return true;
		} else {
			jdbc.disConnection();
			return false;
		}
	}

	@Override
	public boolean updateByFullName(int id, String newFullName) throws SQLException,
ClassNotFoundException {
		String sql = "UPDATE account SET FullName = ? WHERE (AccountID = ?);";
		PreparedStatement preStatement = jdbc.createPrepareStatement(sql);
		preStatement.setString(1, newFullName);
		preStatement.setInt(2, id);
		int result = preStatement.executeUpdate();
		if (result == 1) {
			jdbc.disConnection();
			return true;
		} else {
			jdbc.disConnection();
			return false;
		}
	}

	@Override
	public boolean updateByDepId(int id, int idDep) throws ClassNotFoundException, SQLException
{
		String sql = "UPDATE account SET DepartmentID = ? WHERE (AccountID = ?);";

		PreparedStatement preStatement = jdbc.createPrepareStatement(sql);
		preStatement.setInt(1, idDep);
		preStatement.setInt(2, id);
		int result = preStatement.executeUpdate();
		if (result == 1) {
			jdbc.disConnection();
			return true;
		} else {
			jdbc.disConnection();
			return false;
		}

	}

	@Override
```

```
	public boolean updateByPosId(int id, int idPos) throws SQLException, ClassNotFoundException
{
		String sql = "UPDATE account SET PositionID = ? WHERE (AccountID = ?);";
		PreparedStatement preStatement = jdbc.createPrepareStatement(sql);
		preStatement.setInt(1, idPos);
		preStatement.setInt(2, id);
		int result = preStatement.executeUpdate();
		if (result == 1) {
			jdbc.disConnection();
			return true;
		} else {
			jdbc.disConnection();
			return false;
		}
	}

	@Override
	public boolean isAccIdExists(int id) throws SQLException, ClassNotFoundException {
		String sql = "SELECT * FROM account WHERE AccountID = ?";
		PreparedStatement preStatement = jdbc.createPrepareStatement(sql);
		preStatement.setInt(1, id);
		ResultSet result = preStatement.executeQuery();

		if (result.next()) {
			jdbc.disConnection();
			return true;
		} else {
			jdbc.disConnection();
			return false;
		}
	}

}
```

Tạo Class DepartmentRepository trong datalayer:

```
package com.vti.backend.datalayer;

import java.io.FileNotFoundException;
import java.io.IOException;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

import com.vti.entiy.Department;
import com.vti.ultis.jdbcUltis;

public class DepartmentRepository implements IDepartmentRepository {
	private jdbcUltis jdbc;

	public DepartmentRepository() throws FileNotFoundException, IOException {
		jdbc = new jdbcUltis();
	}

	@Override
	public List<Department> getListDepartment() throws SQLException, ClassNotFoundException {
		String sql = "SELECT * FROM Department ORDER BY DepartmentID";
		ResultSet resultSet = jdbc.executeQuery(sql);
		List<Department> listDep = new ArrayList<Department>();
		while (resultSet.next()) {
			Department dep = new Department(resultSet.getInt("DepartmentID"),
resultSet.getString("DepartmentName"));
			listDep.add(dep);
		}
		jdbc.disConnection();
		return listDep;

	}

	@Override
```

```
	public Department getDepByID(int id) throws SQLException, ClassNotFoundException {
		String sql = "SELECT * FROM Department WHERE DepartmentID = ?";
		PreparedStatement preStatement = jdbc.createPrepareStatement(sql);
		preStatement.setInt(1, id);
		ResultSet result = preStatement.executeQuery();
		if (result.next()) {
			Department dep = new Department(result.getInt("DepartmentID"),
result.getNString("DepartmentName"));
			return dep;
		} else {
			jdbc.disConnection();
			return null;
		}
	}

	@Override
	public Boolean isDepartmentNameExists(String name) throws SQLException,
ClassNotFoundException {
		String sql = "SELECT * FROM Department WHERE DepartmentName = ?";
		PreparedStatement preStatement = jdbc.createPrepareStatement(sql);
		preStatement.setNString(1, name);
		ResultSet result = preStatement.executeQuery();

		if (result.next()) {
			jdbc.disConnection();
			return true;
		} else {
			jdbc.disConnection();
			return false;
		}
	}

	@Override
	public boolean createDep(String name) throws ClassNotFoundException, SQLException {
		String sql = "INSERT INTO Department (DepartmentName) VALUES (?);";
		PreparedStatement preStatement = jdbc.createPrepareStatement(sql);
		preStatement.setNString(1, name);
		int result = preStatement.executeUpdate();
		if (result == 1) {
			jdbc.disConnection();
			return true;
		} else {
			jdbc.disConnection();
			return false;
		}
	}

	@Override
	public boolean updateDepartmentName(int id, String newName) throws ClassNotFoundException,
SQLException {
		Department depID = getDepByID(id);
		if (depID == null) {
			return false;
		} else {
			String sql = "UPDATE Department SET DepartmentName = ? WHERE (DepartmentID
= ?);";
			PreparedStatement preStatement = jdbc.createPrepareStatement(sql);
			preStatement.setNString(1, newName);
			preStatement.setInt(2, id);
			int result = preStatement.executeUpdate();
			if (result == 1) {
				jdbc.disConnection();
				return true;
			} else {
				jdbc.disConnection();
				return false;
			}

		}
	}

	@Override
	public boolean delDepByID(int id) throws ClassNotFoundException, SQLException {
		Department depID = getDepByID(id);
```

```
                if (depID == null) {
                        return false;
                } else {
                        String sql = "DELETE FROM department WHERE (DepartmentID = ?);";
                        PreparedStatement preStatement = jdbc.createPrepareStatement(sql);
                        preStatement.setInt(1, id);
                        int result = preStatement.executeUpdate();
                        if (result == 1) {
                                jdbc.disConnection();
                                return true;
                        } else {
                                jdbc.disConnection();
                                return false;
                        }

                }
        }

}
```

Tạo các Class trong Package businesslayer

Tạo Interface IAccountService trong businesslayer:

```
package com.vti.backend.businesslayer;

import java.io.FileNotFoundException;
import java.io.IOException;
import java.sql.SQLException;
import java.util.List;

import com.vti.entiy.Account;

public interface IAccountService {
        public List<Account> getListAccounts()
                        throws FileNotFoundException, ClassNotFoundException, SQLException,
IOException;

        public Account getAccountByID(int id)
                        throws ClassNotFoundException, FileNotFoundException, SQLException,
IOException;

        public Boolean isAccNameExists(String name) throws ClassNotFoundException, SQLException;

        public boolean createAccount(Account acc, int depId, int posId) throws
ClassNotFoundException, SQLException;

        public boolean delAccByID(int ID) throws ClassNotFoundException, SQLException;

        public boolean updateByEmai(int id, String newEmail) throws ClassNotFoundException,
SQLException;

        public boolean updateByUserName(int id, String newUserName) throws ClassNotFoundException,
SQLException;

        public boolean updateByFullName(int id, String newFullName) throws ClassNotFoundException,
SQLException;

        public boolean updateByDepId(int id, int idDep) throws ClassNotFoundException,
SQLException;

        public boolean updateByPosId(int id, int idPos) throws ClassNotFoundException,
SQLException;

        public boolean isAccIdExists(int id) throws ClassNotFoundException, SQLException;
}
```

Tạo Interface IDepartmentService trong businesslayer:

```
package com.vti.backend.businesslayer;

import java.sql.SQLException;
```

```java
import java.util.List;

import com.vti.entiy.Department;

public interface IDepartmentService {
	public List<Department> getListDepartment() throws ClassNotFoundException, SQLException;

	public Department getDepByID(int id) throws ClassNotFoundException, SQLException;

	public Boolean isDepartmentNameExists(String name) throws ClassNotFoundException,
SQLException;

	public boolean createDep(String name) throws ClassNotFoundException, SQLException;

	public boolean updateDepartmentName(int id, String newName) throws ClassNotFoundException,
SQLException;

	public boolean delDepByID(int id) throws ClassNotFoundException, SQLException;
}
```

Tạo Interface IPossitionService trong businesslayer:

```java
package com.vti.backend.businesslayer;

import java.sql.SQLException;
import java.util.List;

import com.vti.entiy.Position;

public interface IPossitionService {
	public List<Position> getListPosition() throws ClassNotFoundException, SQLException;

	public Position getPosByID(int id) throws ClassNotFoundException, SQLException;
}
```

Tạo Class AccountService trong businesslayer:

```java
package com.vti.backend.businesslayer;

import java.io.FileNotFoundException;
import java.io.IOException;
import java.sql.SQLException;
import java.util.List;

import com.vti.backend.datalayer.AccountRepository;
import com.vti.backend.datalayer.IAccountRepository;
import com.vti.entiy.Account;

public class AccountService implements IAccountService {

	private IAccountRepository accountRepository;

	public AccountService() throws FileNotFoundException, IOException {
		accountRepository = new AccountRepository();
	}

	@Override
	public List<Account> getListAccounts()
			throws FileNotFoundException, ClassNotFoundException, SQLException,
IOException {

		return accountRepository.getListAccounts();
	}

	@Override
	public Account getAccountByID(int id)
			throws ClassNotFoundException, FileNotFoundException, SQLException,
IOException {

		return accountRepository.getAccountByID(id);
```

```
	}

	@Override
	public Boolean isAccNameExists(String name) throws ClassNotFoundException, SQLException {

		return accountRepository.isAccNameExists(name);
	}

	@Override
	public boolean createAccount(Account acc, int depId, int posId) throws
ClassNotFoundException, SQLException {

		return accountRepository.createAccount(acc, depId, posId);
	}

	@Override
	public boolean delAccByID(int ID) throws ClassNotFoundException, SQLException {

		return accountRepository.delAccByID(ID);
	}

	@Override
	public boolean updateByEmai(int id, String newEmail) throws ClassNotFoundException,
SQLException {

		return accountRepository.updateByEmai(id, newEmail);
	}

	@Override
	public boolean updateByUserName(int id, String newUserName) throws ClassNotFoundException,
SQLException {

		return accountRepository.updateByUserName(id, newUserName);
	}

	@Override
	public boolean updateByFullName(int id, String newFullName) throws ClassNotFoundException,
SQLException {

		return accountRepository.updateByFullName(id, newFullName);
	}

	@Override
	public boolean updateByDepId(int id, int idDep) throws ClassNotFoundException, SQLException
{

		return accountRepository.updateByDepId(id, idDep);
	}

	@Override
	public boolean updateByPosId(int id, int idPos) throws ClassNotFoundException, SQLException
{

		return accountRepository.updateByPosId(id, idPos);
	}

	@Override
	public boolean isAccIdExists(int id) throws ClassNotFoundException, SQLException {

		return accountRepository.isAccIdExists(id);
	}

}
```


Tạo Class DepartmentService trong businesslayer:

```
package com.vti.backend.businesslayer;

import java.io.FileNotFoundException;
import java.io.IOException;
import java.sql.SQLException;
import java.util.List;
```

```
import com.vti.backend.datalayer.DepartmentRepository;
import com.vti.entiy.Department;

public class DepartmentService implements IDepartmentService {
	private DepartmentRepository departmentRepository;

	public DepartmentService() throws FileNotFoundException, IOException {
		departmentRepository = new DepartmentRepository();
	}

	@Override
	public List<Department> getListDepartment() throws ClassNotFoundException, SQLException {

		return departmentRepository.getListDepartment();
	}

	@Override
	public Department getDepByID(int id) throws ClassNotFoundException, SQLException {

		return departmentRepository.getDepByID(id);
	}

	@Override
	public Boolean isDepartmentNameExists(String name) throws ClassNotFoundException,
SQLException {

		return departmentRepository.isDepartmentNameExists(name);
	}

	@Override
	public boolean createDep(String name) throws ClassNotFoundException, SQLException {

		return departmentRepository.createDep(name);
	}

	@Override
	public boolean updateDepartmentName(int id, String newName) throws ClassNotFoundException,
SQLException {

		return departmentRepository.updateDepartmentName(id, newName);
	}

	@Override
	public boolean delDepByID(int id) throws ClassNotFoundException, SQLException {

		return departmentRepository.delDepByID(id);
	}

}
```

Tạo Class PossitionService trong businesslayer:

```
package com.vti.backend.businesslayer;

import java.io.FileNotFoundException;
import java.io.IOException;
import java.sql.SQLException;
import java.util.List;

import com.vti.backend.datalayer.PossitionRepository;
import com.vti.entiy.Position;

public class PossitionService implements IPossitionService {
	private PossitionRepository possitionRepository;

	public PossitionService() throws FileNotFoundException, IOException {
		possitionRepository = new PossitionRepository();
	}

	@Override
	public List<Position> getListPosition() throws ClassNotFoundException, SQLException {
```

```
		return possitionRepository.getListPosition();
	}

	@Override
	public Position getPosByID(int id) throws ClassNotFoundException, SQLException {

		return possitionRepository.getPosByID(id);
	}

}
```

Tạo các Class trong Package presentationlayer

Tạo Class AccountController trong presentationlayer:

```
package com.vti.backend.presentationlayer;

import java.io.FileNotFoundException;
import java.io.IOException;
import java.sql.SQLException;
import java.util.List;

import com.vti.backend.businesslayer.AccountService;
import com.vti.backend.businesslayer.IAccountService;
import com.vti.entiy.Account;

public class AccountController {
	private IAccountService accountService;

	public AccountController() throws FileNotFoundException, IOException {
		accountService = new AccountService();
	}

	public List<Account> getListAccounts()
			throws FileNotFoundException, ClassNotFoundException, SQLException,
IOException {

		return accountService.getListAccounts();
	}

	public Account getAccountByID(int id) throws Exception {
		try {
			return accountService.getAccountByID(id);
		} catch (Exception e) {
			return null;
		}

	}

	public boolean createAccount(Account acc, int depId, int posId)
			throws ClassNotFoundException, SQLException, Exception {

		if (accountService.isAccNameExists(acc.getUsername())) {
			return false;
		} else {
			return accountService.createAccount(acc, depId, posId);
		}
	}

	public Boolean isAccNameExists(String name) throws ClassNotFoundException, SQLException {

		return accountService.isAccNameExists(name);

	}

	public boolean updateByEmai(int id, String newEmail) throws ClassNotFoundException,
SQLException {

		return accountService.updateByEmai(id, newEmail);

	}
```

```
	public boolean updateByUserName(int id, String newUserName) throws ClassNotFoundException,
SQLException {

		return accountService.updateByUserName(id, newUserName);

	}

	public boolean updateByFullName(int id, String newFullName) throws ClassNotFoundException,
SQLException {

		return accountService.updateByFullName(id, newFullName);

	}

	public boolean updateByDepId(int id, int idDep) throws ClassNotFoundException, SQLException
{

		return accountService.updateByDepId(id, idDep);
	}

	public boolean updateByPosId(int id, int idPos) throws ClassNotFoundException, SQLException
{

		return accountService.updateByPosId(id, idPos);
	}

	public boolean delAccByID(int ID) throws ClassNotFoundException, SQLException {

		return accountService.delAccByID(ID);
	}

}
```

Tạo Class DepartmentController trong presentationlayer:

```
package com.vti.backend.presentationlayer;

import java.io.FileNotFoundException;
import java.io.IOException;
import java.sql.SQLException;
import java.util.List;

import com.vti.backend.businesslayer.DepartmentService;
import com.vti.backend.businesslayer.IDepartmentService;
import com.vti.entiy.Department;

public class DepartmentController {
	private IDepartmentService departmenttService;

	public DepartmentController() throws FileNotFoundException, IOException {
		departmenttService = new DepartmentService();

	}

	public List<Department> getListDepartment() throws ClassNotFoundException, SQLException {

		return departmenttService.getListDepartment();

	}

	public Department getDepByID(int id) throws ClassNotFoundException, SQLException {

		return departmenttService.getDepByID(id);
	}


	public Boolean isDepartmentNameExists(String name) throws ClassNotFoundException,
SQLException {

		return departmenttService.isDepartmentNameExists(name);
	}
```

```
	public boolean createDep(String name) throws ClassNotFoundException, SQLException {

		return departmenttService.createDep(name);
	}


	public boolean updateDepartmentName(int id, String newName) throws ClassNotFoundException,
SQLException {

		return departmenttService.updateDepartmentName(id, newName);
	}


	public boolean delDepByID(int id) throws ClassNotFoundException, SQLException {

		return departmenttService.delDepByID(id);
	}

}
```

Tạo Class PossitionController trong presentationlayer:

```
package com.vti.backend.presentationlayer;

import java.io.FileNotFoundException;
import java.io.IOException;
import java.sql.SQLException;
import java.util.List;
import com.vti.backend.businesslayer.IPossitionService;
import com.vti.backend.businesslayer.PossitionService;
import com.vti.entiy.Position;

public class PossitionController {
	private IPossitionService possitionService;

	public PossitionController() throws FileNotFoundException, IOException {
		possitionService = new PossitionService();
	}

	public List<Position> getListPosition() throws ClassNotFoundException, SQLException {
		return possitionService.getListPosition();

	}

	public Position getPosByID(int id) throws ClassNotFoundException, SQLException {
		return possitionService.getPosByID(id);

	}
}
```

Tạo các Class trong Package frontend

Tạo Class accountFunction trong frontend:

```
package com.vti.frontend;

import java.io.FileNotFoundException;
import java.io.IOException;
import java.sql.SQLException;
import java.util.List;

import com.vti.backend.presentationlayer.AccountController;
import com.vti.backend.presentationlayer.DepartmentController;
import com.vti.backend.presentationlayer.PossitionController;
import com.vti.entiy.Account;
import com.vti.entiy.Department;
import com.vti.entiy.Position;
import com.vti.ultis.ScannerUltis;

public class accountFunction {
	public static void getListAllAccount()
				throws FileNotFoundException, ClassNotFoundException, SQLException, IOException {
		AccountController accountControler = new AccountController();
		List<Account> listAcc1 = accountControler.getListAccounts();
		String leftAlignFormat = "| %-2d | %-21s | %-15s | %-21s | %-14s | %-16s | %-16s | %n";
		System.out.format(
```

```
                                        "+----+----------------------+----------------+----------------------+---------------
+------------------+------------------+%n");
                        System.out.format(
                                        "|ID  | Email                | Username       |   FullName           | Department
| Position         | Create Date      |%n");
                        System.out.format(
                                        "+----+----------------------+----------------+----------------------+---------------
+------------------+------------------+%n");

                        for (Account acc : listAcc1) {
                                System.out.format(leftAlignFormat, acc.getId(), acc.getEmail(), acc.getUsername(),
acc.getFullName(),
                                                acc.getDepartment(), acc.getPosition(), acc.getCreateDate());
                        }
                        System.out.format(
                                        "+----+----------------------+----------------+----------------------+---------------
+------------------+------------------+%n");

        }

        public static void getAccountByID() throws Exception {
                System.out.println("Tìm kiếm Account theo ID: ");
                System.out.println("Nhập vào ID cần tìm kiếm: ");
                int idFind = ScannerUltis.inputIntPositive();
                AccountController accountControler = new AccountController();
                Account acc2 = accountControler.getAccountByID(idFind);
                if (acc2 != null) {
                        String leftAlignFormat = "| %-2d | %-21s | %-15s | %-21s | %-14s | %-16s | %-16s | %n";
                        System.out.format(
                                        "+----+----------------------+----------------+----------------------+-----
----------+-----------------+------------------+%n");
                        System.out.format(
                                        "|ID  | Email                | Username       |   FullName           |
Department     | Position         | Create Date      |%n");
                        System.out.format(
                                        "+----+----------------------+----------------+----------------------+-----
----------+-----------------+------------------+%n");

                        System.out.format(leftAlignFormat, acc2.getId(), acc2.getEmail(), acc2.getUsername(),
acc2.getFullName(),
                                        acc2.getDepartment(), acc2.getPosition(), acc2.getCreateDate());

                        System.out.format(
                                        "+----+----------------------+----------------+----------------------+-----
----------+-----------------+------------------+%n");

                } else {
                        System.out.println("Không tồn tại account này trên HT");
                }

        }

        public static void isAccNameExists()
                        throws ClassNotFoundException, SQLException, FileNotFoundException, IOException {
                System.out.println("Kiểm tra tên account đã có trên hệ thống? ");
                System.out.println("Nhập vào tên cần kiểm tra: ");
                String nameCheck = ScannerUltis.inputString();
                AccountController accountControler = new AccountController();
                if (accountControler.isAccNameExists(nameCheck)) {
                        System.out.println("Tên đã có trên hệ thống.");
                } else {
                        System.out.println("Tên chưa có trên hệ thống.");
                }
        }

        public static void createAccount() throws ClassNotFoundException, SQLException, Exception {
                Account acc = new Account();
                System.out.println("Nhập vào Email: ");
                acc.setEmail(ScannerUltis.inputEmail());
                System.out.println("Nhập vào UserName: ");
                acc.setUsername(ScannerUltis.inputString());
                System.out.println("Nhập vào FullName: : ");
                acc.setFullName(ScannerUltis.inputString());
                System.out.println("Hãy chọn phòng nhân viên: ");
                int depId = getDep();
                System.out.println("Hãy chọn Position nhân viên: ");
                int posid = getPos();
                AccountController accountControler = new AccountController();
                if (accountControler.createAccount(acc, depId, posid)) {
                        System.out.println("Tạo thành công: ");
                        getListAllAccount();
                } else {
                        System.out.println("Tạo không thành công, hãy kiểm tra lại");
                }
        }

        private static int getPos() throws ClassNotFoundException, SQLException, FileNotFoundException, IOException {
                while (true) {
                        PossitionController possitionControler = new PossitionController();
                        List<Position> listpos = possitionControler.getListPosition();
                        String leftAlignFormat = "| %-6d | %-21s |%n";

                        System.out.format("+--------+----------------------+%n");
                        System.out.format("|   ID   | Position Name        |%n");
                        System.out.format("+--------+----------------------+%n");
                        for (Position position : listpos) {
                                System.out.format(leftAlignFormat, position.getId(), position.getName());
                        }
                        System.out.format("+--------+----------------------+%n");
```

```
                        System.out.println("Chọn Position theo ID:");
                        int choosePos = ScannerUltis.inputIntPositive();
                        if (possitionControler.getPosByID(choosePos) != null) {
                                return choosePos;
                        } else {
                                System.out.println("Không có Position này, hãy chọn lại: ");
                        }
                }
        }

        private static int getDep() throws FileNotFoundException, IOException, ClassNotFoundException, SQLException {
                while (true) {
                        DepartmentController departmentControler = new DepartmentController();
                        List<Department> listDep = departmentControler.getListDepartment();
                        String leftAlignFormat = "| %-6d | %-21s |%n";

                        System.out.format("+--------+-----------------------+%n");
                        System.out.format("|   ID   | Depament Name         |%n");
                        System.out.format("+--------+-----------------------+%n");
                        for (Department department : listDep) {
                                System.out.format(leftAlignFormat, department.getId(), department.getName());
                        }
                        System.out.format("+--------+-----------------------+%n");
                        System.out.println("Chọn phòng theo ID:");
                        int chooseDep = ScannerUltis.inputIntPositive();
                        if (departmentControler.getDepByID(chooseDep) != null) {
                                return chooseDep;
                        } else {
                                System.out.println("Không có phòng này, hãy chọn lại: ");
                        }
                }

        }

        public static void updateAccount() throws Exception {
                AccountController accountController = new AccountController();
                DepartmentController departmentController = new DepartmentController();
                while (true) {
                        switch (getMenuQues6()) {
                        case 1:
                                int id = getidCase1();
                                System.out.println("Nhập vào New Email: ");
                                String newEmail = ScannerUltis.inputEmail();
                                if (accountController.updateByEmai(id, newEmail)) {
                                        System.out.println("Update thành công.");
                                } else {
                                        System.out.println("update không thành công, kiểm tra lại.");
                                }
                                break;
                        case 2:
                                int id2 = getidCase1();
                                System.out.println("Nhập vào New UserName: ");
                                String newUserName = ScannerUltis.inputString();
                                if (accountController.updateByUserName(id2, newUserName)) {
                                        System.out.println("Update thành công.");
                                } else {
                                        System.out.println("update không thành công, kiểm tra lại.");
                                }
                                break;
                        case 3:
                                int id3 = getidCase1();
                                System.out.println("Nhập vào New FullName: ");
                                String newFullName = ScannerUltis.inputString();
                                if (accountController.updateByFullName(id3, newFullName)) {
                                        System.out.println("Update thành công.");
                                } else {
                                        System.out.println("update không thành công, kiểm tra lại.");
                                }
                                break;
                        case 4:
                                int id4 = getidCase1();
                                int idDep = getNewIDDep();
                                if (accountController.updateByDepId(id4, idDep)) {
                                        System.out.println("Update thành công.");

                                } else {
                                        System.out.println("Có lỗi xảy ra, Hãy kiểm tra lại");
                                }
                                break;
                        case 5:
                                int id5 = getidCase1();
                                int idPos = getNewIDPos();
                                if (accountController.updateByPosId(id5, idPos)) {
                                        System.out.println("Update thành công.");

                                } else {
                                        System.out.println("Có lỗi xảy ra, Hãy kiểm tra lại");
                                }
                                break;
                        case 6:
                                return;
                        }
                }

        }

        private static int getNewIDPos() throws ClassNotFoundException, SQLException, FileNotFoundException, IOException {
                PossitionController possitionControler = new PossitionController();
                System.out.println("Chọn Position:");
```

```
                while (true) {
                        List<Position> listpos = possitionControler.getListPosition();
                        String leftAlignFormat = "| %-6d | %-21s |%n";

                        System.out.format("+--------+----------------------+%n");
                        System.out.format("|   ID   | Position Name        |%n");
                        System.out.format("+--------+----------------------+%n");
                        for (Position position : listpos) {
                                System.out.format(leftAlignFormat, position.getId(), position.getName());
                        }
                        System.out.format("+--------+----------------------+%n");
                        System.out.println("Chọn ID của Position cần Update:");
                        int id = ScannerUltis.inputIntPositive();
                        if (possitionControler.getPosByID(id) != null) {
                                return id;
                        } else {
                                System.out.println("Không có Position này, hãy nhập lại: ");
                        }
                }
        }

        private static int getNewIDDep() throws ClassNotFoundException, SQLException, FileNotFoundException, IOException {
                DepartmentController departmentController = new DepartmentController();
                System.out.println("Chọn phòng");
                while (true) {
                        List<Department> listDep = departmentController.getListDepartment();
                        String leftAlignFormat = "| %-6d | %-21s |%n";

                        System.out.format("+--------+----------------------+%n");
                        System.out.format("|   ID   | Depament Name        |%n");
                        System.out.format("+--------+----------------------+%n");
                        for (Department department : listDep) {
                                System.out.format(leftAlignFormat, department.getId(), department.getName());
                        }
                        System.out.format("+--------+----------------------+%n");
                        System.out.println("Chọn ID của phòng cần Update:");
                        int id = ScannerUltis.inputIntPositive();
                        if (departmentController.getDepByID(id) != null) {
                                return id;
                        } else {
                                System.out.println("Không có phòng này, hãy nhập lại: ");
                        }
                }
        }

        private static int getidCase1() throws Exception {
                while (true) {
                        System.out.println("Nhập vào ID của account cần Update: ");
                        int id = ScannerUltis.inputIntPositive();
                        AccountController accountControler = new AccountController();
                        if (accountControler.getAccountByID(id) != null) {
                                return id;
                        } else {
                                System.out.println("Không có account này trên hệ thống, Nhập lại: ");
                        }
                }

        }

        private static int getMenuQues6() {
                while (true) {
                        System.out.println("Bạn muốn update trường nào??");
                        System.out.println("1.Email, 2.UserName, 3.FullName, 4.Department, 5.Position, 6.Exit ");
                        int i = ScannerUltis.inputIntPositive();
                        if (i == 1 || i == 2 || i == 3 || i == 4 || i == 5 || i == 6) {
                                return i;
                        } else {
                                System.out.println("Chọn lại: ");
                        }
                }

        }

        public static void deleteById() throws Exception {
                AccountController accountController = new AccountController();
                int id = getIdDel();
                if (accountController.delAccByID(id)) {
                        System.out.println("Xóa thành công");
                } else {
                        System.out.println("Đã có lỗi xảy ra.");
                }
        }

        private static int getIdDel() throws Exception {
                AccountController accountController = new AccountController();
                while (true) {
                        System.out.println("Nhập vào ID Account cần xóa: ");
                        int id = ScannerUltis.inputIntPositive();
                        if (accountController.getAccountByID(id) != null) {
                                return id;
                        } else {
                                System.out.println("Không có Account này trên hệ thống, Nhập lại: ");
                        }
                }
        }

}
```

Tạo Class departmentFunction trong frontend:

```
package com.vti.frontend;

import java.io.FileNotFoundException;
import java.io.IOException;
import java.sql.SQLException;
import java.util.List;

import com.vti.backend.presentationlayer.DepartmentController;
import com.vti.entiy.Department;
import com.vti.ultis.ScannerUltis;

public class departmentFunction {
	public static void getListDepartment() throws ClassNotFoundException, SQLException, FileNotFoundException, IOException {
		DepartmentController departmentController = new DepartmentController();
		List<Department> listDep1 = departmentController.getListDepartment();
		String leftAlignFormat = "| %-6d | %-21s |%n";
		System.out.format("+--------+----------------------+%n");
		System.out.format("|   ID   | Department Name      |%n");
		System.out.format("+--------+----------------------+%n");

		for (Department department : listDep1) {
			System.out.format(leftAlignFormat, department.getId(), department.getName());
		}
		System.out.format("+--------+----------------------+%n");
	}

	public static void getDepByID() throws ClassNotFoundException, SQLException, FileNotFoundException, IOException {
		DepartmentController departmentController = new DepartmentController();
		System.out.println("Tìm kiếm phòng theo ID: ");
		System.out.println("Nhập vào ID cần tìm kiếm: ");
		int idFind = ScannerUltis.inputIntPositive();
		Department depQues3 = departmentController.getDepByID(idFind);
		if (depQues3 != null) {
			String leftAlignFormat = "| %-6d | %-21s |%n";
			System.out.format("+--------+----------------------+%n");
			System.out.format("|   ID   | Department Name      |%n");
			System.out.format("+--------+----------------------+%n");
			System.out.format(leftAlignFormat, depQues3.getId(), depQues3.getName());
			System.out.format("+--------+----------------------+%n");
		} else {
			System.out.println("Không tồn tại phòng này trên HT");
		}

	}

	public static void isDepartmentNameExists()
			throws ClassNotFoundException, SQLException, FileNotFoundException, IOException {
		DepartmentController departmentController = new DepartmentController();
		System.out.println("Kiểm tra tên phòng đã có trên hệ thông? ");
		System.out.println("Nhập vào tên cần kiểm tra: ");
		String nameCheck = ScannerUltis.inputString();
		Boolean checkResult = departmentController.isDepartmentNameExists(nameCheck);
		if (checkResult) {
			System.out.println("Tên đã có trên hệ thống.");
			getListDepartment();
		} else {
			System.out.println("Tên chưa có trên hệ thống.");
		}
	}

	public static void createDep() throws ClassNotFoundException, SQLException, FileNotFoundException, IOException {
		DepartmentController departmentController = new DepartmentController();
		String newNameDep = getNewName();
		if (departmentController.createDep(newNameDep)) {
			System.out.println("Tạo thành công.");
			getListDepartment();
		} else {
			System.out.println("Đã có lỗi xảy ra");
		}

	}

	private  static String getNewName() throws ClassNotFoundException, SQLException, FileNotFoundException, IOException {
		DepartmentController departmentController = new DepartmentController();
		while (true) {
			System.out.println("Nhập vào tên phòng cần tạo: ");
			String newName = ScannerUltis.inputString();
			if (departmentController.isDepartmentNameExists(newName)) {
				System.out.println("Đã có phòng trên hệ thống");
			} else {
				return newName;
			}
		}
	}

	public static void updateDepartmentName() throws ClassNotFoundException, SQLException, FileNotFoundException, IOException
{
		DepartmentController departmentController = new DepartmentController();
		int updateID = getIdUpdate();
		System.out.println("Nhập vào tên cần Updare: ");
		String newName = ScannerUltis.inputString();
		if (departmentController.updateDepartmentName(updateID, newName)) {
			System.out.println("Update tên phòng thành công: ");
			getListDepartment();
		} else {
			System.out.println("Đã có lỗi xảy ra");
		}
```

```
	}

	private static int getIdUpdate() throws ClassNotFoundException, SQLException, FileNotFoundException, IOException {
		DepartmentController departmentController = new DepartmentController();
		while (true) {
			System.out.println("Nhập ID phòng cần thao tác: ");
			int id = ScannerUltis.inputIntPositive();
			Department dep = departmentController.getDepByID(id);
			if (dep == null) {
				System.out.println("Không có ID này trên HT");
			} else {
				return id;
			}
		}
	}

	public static void delDepByID() throws ClassNotFoundException, SQLException, FileNotFoundException, IOException {
		DepartmentController departmentController = new DepartmentController();
		int updateID = getIdUpdate();
		if (departmentController.delDepByID(updateID)) {
			System.out.println("Xóa phòng thành công");
			getListDepartment();
		} else {
			System.out.println("Đã có lỗi xảy ra");
		}
	}

}
```

Tạo Class PossitionFunciton trong frontend:

```
package com.vti.frontend;

import java.io.FileNotFoundException;
import java.io.IOException;
import java.sql.SQLException;
import java.util.List;

import com.vti.backend.presentationlayer.PossitionController;
import com.vti.entiy.Position;
import com.vti.ultis.ScannerUltis;

public class PossitionFunciton {

	public static void getListPossition()
			throws ClassNotFoundException, SQLException, FileNotFoundException, IOException {
		PossitionController possitionController = new PossitionController();
		List<Position> listpos1 = possitionController.getListPosition();
		String leftAlignFormat = "| %-6d | %-21s |%n";
		System.out.format("+--------+-----------------------+%n");
		System.out.format("|   ID   | Possition  Name       |%n");
		System.out.format("+--------+-----------------------+%n");

		for (Position possition : listpos1) {
			System.out.format(leftAlignFormat, possition.getId(), possition.getName());
		}
		System.out.format("+--------+-----------------------+%n");

	}

	public static void getPossitonByID()
			throws ClassNotFoundException, SQLException, FileNotFoundException, IOException {

		PossitionController possitionController = new PossitionController();
		System.out.println("Tìm kiếm phòng theo ID: ");
		System.out.println("Nhập vào ID cần tìm kiếm: ");
		int idFind = ScannerUltis.inputIntPositive();
		Position pos = possitionController.getPosByID(idFind);
		if (pos != null) {
			String leftAlignFormat = "| %-6d | %-21s |%n";
			System.out.format("+--------+-----------------------+%n");
			System.out.format("|   ID   |  Possition Name       |%n");
			System.out.format("+--------+-----------------------+%n");
			System.out.format(leftAlignFormat, pos.getId(), pos.getName());
			System.out.format("+--------+-----------------------+%n");
		} else {
			System.out.println("Không tồn tại");
		}
	}
}
```

Tạo Class Demo1 trong frontend:

```
package com.vti.frontend;

import java.io.FileNotFoundException;
import java.io.IOException;
import java.sql.SQLException;
import java.util.List;

import com.vti.backend.presentationlayer.AccountController;
import com.vti.entiy.Account;
import com.vti.ultis.ScannerUltis;

public class Demo1 {
	public static void main(String[] args) throws Exception {
```

```
		while (true) {
			String leftAlignFormat = "| %-72s |%n";
			System.out.format("+------------------------------------------------------------------------+%n");
			System.out.format("|                        Choose please                                   |%n");
			System.out.format("+------------------------------------------------------------------------+%n");
			System.out.format(leftAlignFormat, "1. Quản lý nhân viên.");
			System.out.format(leftAlignFormat, "2. Quản lý phòng ban.");
			System.out.format(leftAlignFormat, "3. Quản lý vị trí.");
			System.out.format(leftAlignFormat, "4. Thoát chương trình.");
			switch (ScannerUltis.inputIntPositive()) {
			case 1:
				getMenuAccount();
				break;
			case 2:
				getMenuDepartment();
				break;
			case 3:
				getMenuPossition();
				break;
			case 4:
				return;
			default:
				System.out.println("Nhập lại:");
				break;
			}
		}

	}

	private static void getMenuPossition() throws ClassNotFoundException, FileNotFoundException, SQLException, IOException {
		while (true) {
			String leftAlignFormat = "| %-72s |%n";
			System.out.format("+------------------------------------------------------------------------+%n");
			System.out.format("|                        Choose please                                   |%n");
			System.out.format("+------------------------------------------------------------------------+%n");
			System.out.format(leftAlignFormat, "1. Get list possition");
			System.out.format(leftAlignFormat, "2. Get possition by id");
			System.out.format(leftAlignFormat, "3. Exit");
			System.out.format("+------------------------------------------------------------------------+%n");
			switch (ScannerUltis.inputIntPositive()) {
			case 1:
				PossitionFunciton.getListPossition();
				break;
			case 2:
				PossitionFunciton.getPossitonByID();
				;
				break;
			case 3:
				return;
			default:
				System.out.println("Nhập lại:");
				break;
			}
		}

	}

	private static void getMenuDepartment()
			throws ClassNotFoundException, FileNotFoundException, SQLException, IOException {
		while (true) {
			String leftAlignFormat = "| %-72s |%n";
			System.out.format("+------------------------------------------------------------------------+%n");
			System.out.format("|                        Choose please                                   |%n");
			System.out.format("+------------------------------------------------------------------------+%n");
			System.out.format(leftAlignFormat, "1. Get list department");
			System.out.format(leftAlignFormat, "2. Get department by id");
			System.out.format(leftAlignFormat, "3. Check department name exists");
			System.out.format(leftAlignFormat, "4. create new department");
			System.out.format(leftAlignFormat, "5. update department");
			System.out.format(leftAlignFormat, "6. Delete department ID");
			System.out.format(leftAlignFormat, "7. Exit");
			System.out.format("+------------------------------------------------------------------------+%n");
			switch (ScannerUltis.inputIntPositive()) {
			case 1:
				departmentFunction.getListDepartment();
				break;
			case 2:
				departmentFunction.getDepByID();
				;
				break;
			case 3:
				departmentFunction.isDepartmentNameExists();
				;
				break;
			case 4:
				departmentFunction.createDep();
				;
				break;
			case 5:
				departmentFunction.updateDepartmentName();
				;
				break;
			case 6:
				departmentFunction.delDepByID();
				;
				break;
			case 7:
				return;
			default:
```

```
                                        System.out.println("Nhập lại:");
                                        break;
                                }
                        }

                }

                private static void getMenuAccount() throws Exception {

                        while (true) {
                                String leftAlignFormat = "| %-72s |%n";
                                System.out.format("+--------------------------------------------------------------------------+%n");
                                System.out.format("|                        Choose please                             |%n");
                                System.out.format("+--------------------------------------------------------------------------+%n");
                                System.out.format(leftAlignFormat, "1. Read data - get list account");
                                System.out.format(leftAlignFormat, "2. Read data - get account by id");
                                System.out.format(leftAlignFormat, "3. Check account name exists");
                                System.out.format(leftAlignFormat, "4. create account");
                                System.out.format(leftAlignFormat, "5. update account");
                                System.out.format(leftAlignFormat, "6. Delete by ID");
                                System.out.format(leftAlignFormat, "7. Exit");
                                System.out.format("+--------------------------------------------------------------------------+%n");
                                switch (ScannerUltis.inputIntPositive()) {
                                case 1:
                                        accountFunction.getListAllAccount();
                                        break;
                                case 2:
                                        accountFunction.getAccountByID();
                                        break;
                                case 3:
                                        accountFunction.isAccNameExists();
                                        break;
                                case 4:
                                        accountFunction.createAccount();
                                        break;
                                case 5:
                                        accountFunction.updateAccount();
                                        break;
                                case 6:
                                        accountFunction.deleteById();
                                        break;
                                case 7:
                                        return;
                                default:
                                        System.out.println("Nhập lại:");
                                        break;
                                }
                        }
                }

}
```

Exercise 3: Validation

Question 1: Validation
Tạo validation & xử lý exception ở class controller, repository cho các chức năng trên.

Question 2: Handling business logic Handling business logic ở class service cho các chức năng trên.

Exercise 4: Demo

Question 1: Viết menu ở frontend để demo các chức năng

Exercise 5 (Optional):

Question 1:
Làm tương tự với entity Group

Exercise 6 (Optional):

Chuẩn bị project template để thi

- Tạo sẵn template database và kết nối JDBC
- Tạo sẵn File ScannerUtil có các method nhập vào Email, sđt, ...
- Làm sẵn các chức năng CRUD và demo
- Tạo template comment sẵn

Bài thi sẽ xoay quanh CRUD và login, validate