Design UML for CTEM

1. **Analyze System:**
   1. **Identity Actor and Use case**

Form analyze requirement, we have ? actors and ? usecases:

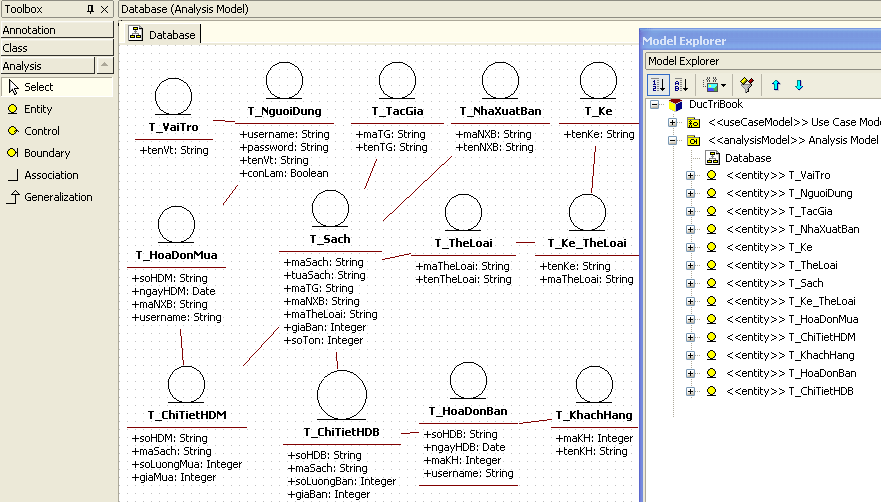
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Actor | Quantity | Operate with system | Use case |
| 1 | User | More | Login | UC\_LogIn |
| 2 | Manager DB | 1 | Manage DB Info | UC\_Mg\_DB\_Info |
| 3 | Supper Admin | 1 | Manage Menu   * Create Menu * Update Menu * Delete Menu | UC\_Mg\_Menu   * UC\_C\_Menu * UC\_U\_Menu * UC\_D\_Menu |
| 4 | Admin | 1 | Manage Employee   * View Employee   + Search Employee * Create Employee * Update Employee * Delete Employee * Delegate for Employee   Manage Rights   * Create Right * Update Right * Delete Right   Mange Role   * Create Role * Update Role * Delete Role | UC\_Mg\_Emp   * UC\_V\_Emp   + UC\_S\_Emp * UC\_C\_Emp * UC\_U\_ Emp * UC\_D\_ Emp * UC\_D\_ Emp   UC\_Mg\_Right   * UC\_C\_Right * UC\_U\_ Right * UC\_D\_ Right   UC\_Mg\_Role   * UC\_C\_Role * UC\_U\_ Role * UC\_D\_ Role |
| 5 | Manager | 1 | Manage Employee   * View Employee   + Search Employee * Create Employee * Update Employee * Delete Employee * Delegate for Employee   Manage Construction   * View Construction   + Search Construction * Create Construction   + Subcontractors (→ UC\_C\_Subc)   + Estimate (→ UC\_C\_Est)   + Non Estimate * Update Construction * Delete Construction * View Report Construction (have filters)   Manage Subcontractors   * View Subcontractors   + Search Subcontractors * View Report (have filters) * Create Subcontractors * Update Subcontractors * Delete Subcontractors   Manage Estimate   * View Estimate   + Search Estimate * View Report (have filters)   Manage Final Account   * View Final Account   + Search Final Account * View Report (have filters)   Manage Debt   * View Debt   + Search Debt * View Report (have filters)   Manage Warehouse   * View Warehouse   + Search Warehouse * View Report (have filters)   Manage Worker   * View Worker   + Search Worker * View Report (have filters) | UC\_Mg\_Emp   * UC\_V\_Emp   + UC\_S\_Emp * UC\_C\_Emp * UC\_U\_ Emp * UC\_D\_ Emp * UC\_Delegate\_ Emp   UC\_Mg\_Constr   * UC\_V\_Constr   + UC\_S\_Constr * UC\_C\_Constr   + UC\_Subc (→ UC\_C\_Subc   + UC\_Est   + UC\_Non\_Est * UC\_U\_Constr * UC\_D\_Constr * UC\_R\_Constr   UC\_Mg\_Subc   * UC\_V\_Subc   + UC\_S\_Subc * UC\_R\_Subc * UC\_C\_Subc * UC\_U\_Subc * UC\_D\_Subc   UC\_Mg\_Est   * UC\_V\_ Est   + UC\_S\_ Est * UC\_R\_ Est   UC\_Mg\_Final \_Acc   * UC\_V\_ Final \_Acc   + UC\_S\_ Final \_Acc * UC\_R\_Subc   UC\_Mg\_Debt   * UC\_V\_ Debt   + UC\_S\_Debt * UC\_R\_ Debt   UC\_Mg\_Whouse   * UC\_V\_ Whouse   + UC\_S\_ Whouse * UC\_R\_ Whouse   UC\_Mg\_Worker   * UC\_V\_ Worker   + UC\_S\_ Worker * UC\_R\_ Worker |
| 6 | Chief Accountant | 1 | Manage Employee   * View Employee   + Search Employee * Create Employee * Update Employee * Delete Employee * Delegate for Employee   Manage Construction   * View Construction   + Search Construction * Create Construction   + Subcontractors (→ UC\_C\_Subc)   + Estimate (→ UC\_C\_Est)   + Non Estimate * Update Construction * Delete Construction * View Report Construction (have filters)   Manage Subcontractors   * View Subcontractors   + Search Subcontractors * View Report (have filters) * Create Subcontractors * Update Subcontractors * Delete Subcontractors   Manage Estimate   * View Estimate   + Search Estimate * View Report (have filters)   Manage Final Account   * View Final Account   + Search Final Account * View Report (have filters)   Manage Debt   * View Debt   + Search Debt * View Report (have filters)   Manage Warehouse   * View Warehouse   + Search Warehouse * View Report (have filters)   Manage Worker   * View Worker   + Search Worker   View Report (have filters) | UC\_Mg\_Emp   * UC\_V\_Emp   + UC\_S\_Emp * UC\_C\_Emp * UC\_U\_ Emp * UC\_D\_ Emp * UC\_Delegate\_ Emp   UC\_Mg\_Constr   * UC\_V\_Constr   + UC\_S\_Constr * UC\_C\_Constr   + UC\_Subc (→ UC\_C\_Subc   + UC\_Est   + UC\_Non\_Est * UC\_U\_Constr * UC\_D\_Constr * UC\_R\_Constr   UC\_Mg\_Subc   * UC\_V\_Subc   + UC\_S\_Subc * UC\_R\_Subc * UC\_C\_Subc * UC\_U\_Subc * UC\_D\_Subc   UC\_Mg\_Est   * UC\_V\_ Est   + UC\_S\_ Est * UC\_R\_ Est   UC\_Mg\_Final \_Acc   * UC\_V\_ Final \_Acc   + UC\_S\_ Final \_Acc * UC\_R\_Subc   UC\_Mg\_Debt   * UC\_V\_ Debt   + UC\_S\_Debt * UC\_R\_ Debt   UC\_Mg\_Whouse   * UC\_V\_ Whouse   + UC\_S\_ Whouse * UC\_R\_ Whouse   UC\_Mg\_Worker   * UC\_V\_ Worker   + UC\_S\_ Worker   UC\_R\_ Worker |
| 7 | Accountant | More | Manage Employee   * View Employee   + Search Employee   Manage Construction   * View Construction   + Search Construction * View Report Construction (have filters)   Manage Subcontractors   * Manage Disbursement * View Subcontractors   + Search Subcontractors * View Report (have filters)   Manage Estimation   * Create Estimation * Update Estimation * Delete Estimation * Create Estimation Detail * Update Estimation Detail * Delete Estimation Detail * View Estimation   + Search Estimation   + View Estimation Detail     - Search Estimation Detail * View Report (have filters) * Relate to Material   Manage Final Account   * Create Final Account * Update Final Account * Delete Final Account * Create Final Account Detail * Update Final Account Detail * Delete Final Account Detail * View Final Account   + Search Final Account   + View Final Account Detail     - Search Final Account Detail * View Report (have filters) * Relate to Material * Relate to Warehouse * Relate to Estimate   Manage Debt   * Create Debt * Update Debt * Delete Debt * View Debt   + Search Debt * View Report (have filters) * Relate to Final Account   Manage Warehouse   * Create Warehouse * Update Warehouse * Delete Warehouse * View Warehouse   + Search Warehouse * View Report (have filters)   Manage Stock Out   * Create Stock Out * Update Stock Out * Delete Stock Out * View Stock Out   + Search Stock Out * View Report (have filters) * Relate to Warehouse * Relate to Estimate Detail   Manage Worker   * Create Worker Salary * Update Worker Salary * Delete Worker Salary * Create Worker * Update Worker * Delete Worker * View Worker Salary   + Search Worker Salary   + Search Worker * View Report (have filters)   Manage Material   * Create Material * Update Material * Delete Material * View Material   + Search Material | UC\_Mg\_Emp   * UC\_V\_Emp   + UC\_S\_Emp   UC\_Mg\_Constr   * UC\_V\_Constr   + UC\_S\_Constr * UC\_R\_Constr   UC\_Mg\_Subc   * UC\_Mg\_Disb * UC\_V\_Subc   + UC\_S\_Subc * UC\_R\_Subc   UC\_Mg\_Est   * UC\_C\_Est * UC\_U\_Est * UC\_D\_Est * UC\_C\_Est\_Detail * UC\_U\_Est\_Detail * UC\_D\_Est\_Detail * UC\_V\_ Est   + UC\_S\_ Est   + UC\_V\_Est\_Detail     - UC\_S\_Est\_Detail * UC\_R\_ Est * UC\_Relate\_Mat   UC\_Mg\_Final \_Acc   * UC\_C\_Final\_Acc * UC\_U\_Final\_Acc * UC\_D\_Final\_Acc * UC\_C\_Final\_Acc\_Detail * UC\_U\_Final\_Acc\_Detail * UC\_D\_Final\_Acc\_Detail * UC\_V\_ Final \_Acc   + UC\_S\_ Final \_Acc   + UC\_V\_Acc\_Detail     - UC\_S\_Acc\_Detail * UC\_R\_Final\_Acc * UC\_Relate\_Mat * UC\_Relate\_Whouse * UC\_Relate\_Est   UC\_Mg\_Debt   * UC\_C\_Debt * UC\_U\_Debt * UC\_D\_Debt * UC\_V\_ Debt   + UC\_S\_Debt * UC\_R\_ Debt * UC\_Relate\_Final\_Acc   UC\_Mg\_Whouse   * UC\_C\_Whouse * UC\_U\_Whouse * UC\_D\_Whouse * UC\_V\_ Whouse   + UC\_S\_ Whouse * UC\_R\_ Whouse   UC\_Mg\_StockOut   * UC\_C\_ StockOut * UC\_U\_ StockOut * UC\_D\_ StockOut * UC\_V\_ StockOut   + UC\_S\_ StockOut * UC\_R\_ StockOut * UC\_Relate\_Whouse * UC\_Relate\_Est\_Detail   UC\_Mg\_Worker   * UC\_C\_Wker\_Sal * UC\_U\_Wker\_Sal * UC\_D\_Wker\_Sal * UC\_C\_Worker * UC\_U\_ Worker * UC\_D\_ Worker * UC\_V\_ Wker\_Sal   + UC\_S\_Wker\_Sal   + UC\_S\_ Worker * UC\_R\_ Wker\_Sal   UC\_Mg\_Mat   * UC\_C\_Mat * UC\_U\_Mat * UC\_D\_Mat * UC\_V\_Mat   + UC\_S\_Mat |
| 8 | Cashier | 1 | Manage Employee   * View Employee   + Search Employee * Manage Employee Salary   + Create Employee Salary   + Update Employee Salary   + Delete Employee Salary   + View Employee Salary     - Search Employee Salary   + View Report (have filters) * Manage Employee Advance   + Create Employee Advance   + Update Employee Advance   + Delete Employee Advance   + View Employee Advance     - Search Employee Adv   + View Report   Manage Comparison Debt   * Create Comparison Debt * Update Comparison Debt * Delete Comparison Debt * View Comparison Debt   + Search Comparison Debt * View Report (have filters) * Relate to Debt * Relate to Final Account | UC\_Mg\_Emp   * UC\_V\_Emp   + UC\_S\_Emp * UC\_Mg\_Emp\_Sal   + UC\_C\_ Emp\_Sal   + UC\_U\_ Emp\_Sal   + UC\_D\_ Emp\_Sal   + UC\_V\_Emp\_Sal     - UC\_S\_ESa   + UC\_R\_Emp\_Sal * UC\_Mg\_Emp\_Adv   + UC\_C\_ Emp\_ Adv   + UC\_U\_ Emp\_ Adv   + UC\_D\_ Emp\_ Adv   + UC\_V\_Emp\_ Adv     - UC\_S\_EAd   + UC\_R\_Emp\_ Adv   UC\_Mg\_Compa\_Debt   * UC\_C\_ Compa\_Debt * UC\_U\_ Compa\_Debt * UC\_D\_ Compa\_Debt * UC\_V\_ Compa\_Debt   + UC\_S\_ComDebt * UC\_R\_ Compa\_Debt * UC\_Relate\_Debt * UC\_Relate\_Final\_Acc |

* 1. **Phân tích kho dữ liệu**

Từ từ điển dữ liệu, kho dữ liệu của ứng dụng được mô hình hoá bằng các bảng sau:

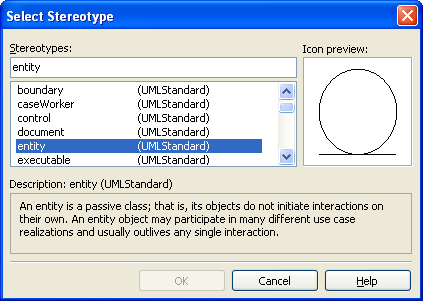
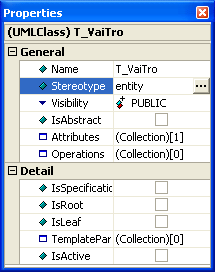
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Stt** | **Tên thực thề** | **Thuộc tính** | **Kiểu** | **Mô tả cho** |
| 1 | T\_VaiTro | tenVt | String(20) – PK | Vai trò của người chạy chương trình  Trị: Quản lý/ Kế toán/Nhân viên |
| 2 | T\_NguoiDung | username  password  tenVt  conLam | String(20)- PK  String(20)  String (20) – FK  Boolean | Người dùng có authentication   * Tên người dùng * Mật khẩu * Vai trò của người dùng * Còn làm hay nghỉ rồi |
| 3 | T\_NhaXuatBan | maNXB  tenNXB | String(20)-PK  String(20) | Nhà xuất bản   * Mã nhà xuất bản * Tên nhà xuất bản |
| 4 | T\_TacGia | maTG  tenTG | String(20)-PK  String(50) | Tác giả   * Mã tác giả * Tên tác giả |
| 5 | T\_Ke | tenKe | String(20) – PK | Kệ sách  -tên kệ sách |
| 6 | T\_TheLoai | maTheLoai  tenTheLoai | String(20)-PK  String(50) | Thể loại sách   * Mã thể loại * Tên thể loại |
| 7 | T\_Ke\_TheLoai | tenKe  maTheLoai | String(20) – PK, FK  String(20) – PK, FK | Quan hệ kệ(n)-Thể loại(n)   * Tên kệ chứa thể loại * Mã thể loại được chứa trên kệ |
| 8 | T\_Sach | maSach  tuaSach  maTG  maNXB  maTheLoai  giaBan  soTon | String(20)-PK  String (30)  String (20) –FK  String (20) - FK  String (20) - FK  Integer  Integer | Sách   * Mã sách * Tựa sách * Mã tác giả * Mã nhà xuất bản * Mã thể loại * Giá bán * Lượng tồn kho |
| 9 | T\_KhachHang | maKH  tenKH | Integer, PK, identity  String(50) | Khách hàng   * Mã khách hàng * Tên khách hàng |
| 10 | T\_HoaDonMua | soHDM  ngayHDM  maNXB  username | String(20) - PK  Date  String(20) –FK  String(20) –FK | Hoá đơn mua   * Số hoá đơn mua * Ngày hoá đơn * Mã nhà xuất bản xuất hoá đơn * Kế toán lưu hoá đơn |
| 11 | CTHDM | soHDM  maSach  soluongMua  giaMua | String(20) –PK, FK  String(20)- PK, FK  Integer  Integer | Chi tiết hoá đơn mua   * Số hoá đơn mua * Mã sách được mua * Số lượng mua * Giá mua |
| 12 | T\_HoaDonBan | soHDB  ngayPD  maKhach  username | String(20)- PK  Date  Integer  String(20) – FK | Hoá đơn bán sách   * Số hoá đơn bán * Ngày hoá đơn bán * Mã khách hàng * Nhân viên in hoá đơn |
| 13 | T\_CTHDB | soHDB  maSach  soluongBan  giaBan | String(20) - FK, PK  String(20) – FK, PF  Integer  Integer | Chi tiết hoá đơn bán   * Số hoá đơn bán * Mã sách đã bán * Số lượng * Đơn giá bán |

Các thực thể này đóng vai trò các kho dữ liệu để người dùng thao tác vào. Mô hình hoá các thực thể được biều diễn bằng mô hình sau: *(trong báo cáo chỉ để diagram, không có các cửa sổ công cụ)*



***Hướng dẫn cách tiến hành***

* *Chọn Main diagram trong Analysis Model, đổi tên thành Database ( dùng cửa sổ Properties)*
* *Lần lượt chọn biểu tượng Entity trong Toolbox để vẽ các entity*
  + *Bảo đảm mô hình biểu diễn đúng entity: Chọn đối tượng trong Model Explorer, mở cửa sổ Properties ( nếu không thấy thì dùng menu View.Properties), chọn stereoType: Entity.*

**

* + *Thay đổi cách trình bầy 1 entity trên lược đồ: Kích phải vào 1 entity trên lược đồ, chọn* ***Format/Stereotype display/Iconic*** *kết hợp với chọn* ***Format/Suppress Asttributes****. Stereotype- kiểu có thể thay đổi, lưỡng tính- là một kiểu dữ liệu của UML cho phép người thiết kế cụ thể hoá sau một đối tượng.*
  + *Vì đây là mô hình dữ liệu nên dùng những kiến thức về database để phân tích.*
  + *Hai cách add attribute cho entity:*
    - *Cách 1: Kích phải chuột vào entity trên diagram, add attribute, gõ trực tiếp tên và kiểu của attribute phân cách bằng dấu hai chấm.*
    - *Cách 2: Kích phải vào entity trong của sổ Model Explorer/ Add attribute/ ấn định tên và kiểu trong cửa sổ Properties.*
  + *Phân tich này sẽ được Database specialist thiết kế database cho ứng dụng*
  + *Không nên tạo operation cho các entity vì dữ liệu trong database và đối tượng trong bộ nhớ có khác biệt một chút. Chúng ta sẽ tạo những class ánh xạ riêng cho các bảng trong đó có các operation giúp nạp/ lưu mới/ cập nhật/ xoá giữa đối tượng thuộc lớp với database bên dưới.*
  1. **Chi tiết hoá các use case**

Các usecase đã phân tích ở trên được chi tiết hoá bằng các mô tả sau:

**1.4.1- Nhóm use case chung**

|  |  |  |
| --- | --- | --- |
| User Case ID | UC\_1 | |
| Name | UC\_LogIn | |
| Goal | Đăng nhập vào hệ thống | |
| Actors | CanBo | |
| Pre-conditions |  | |
| Post-conditions |  | |
| Main Flow | 1.Người dùng nhập username, password, tenVt | 1. Hệ thống sẽ kiểm tra tính hợp lệ của user.   Xuất giao diện phù hợp cho user này. |
| Exception | 2A: Nếu user không hợp lệ, xuất thông báo “Dữ liệu không hợp lệ” và quay về bước 1. | |
| Open Issues | N/A | |

**1.4.2- Nhóm use case cho vai trò quản lý**

|  |  |  |
| --- | --- | --- |
| User Case ID | UC\_2 | |
| Name | UC\_TK\_NS | |
| Goal | Kết hợp xem tình hình kinh doanh và quản lý nhân sự | |
| Actors | QuanLy | |
| Pre-conditions | User đã đănng nhập thành công với vai trò QuanLy | |
| Post-conditions |  | |
| Main Flow | 1.User chọn chức năng | 1. Nếu chức năng được chọn là “Thống kê”, xuất giao diện của UC\_ThongKe   Ngược lại xuất giao diện của UC\_QLNhanSu. |
| Exception |  | |
| Open Issues | N/A | |

|  |  |  |
| --- | --- | --- |
| User Case ID | UC\_3 | |
| Name | UC\_ThongKe | |
| Goal | Thống kê tình hình kinh doang trong một khoảng thời gian | |
| Actors | QuanLy | |
| Pre-conditions | User đã đănng nhập thành công với vai trò QuanLy | |
| Post-conditions |  | |
| Main Flow | 1.User nhập date1, date2 | 1. Xuât thống kê kinh doanh từ date1 đến date2. |
| Exception | 2A: Dạng ngày tháng không hợp lệ: Báo lỗi “Invalid date forma” quay lại bước 1  2B: Nếu date1>date2, hoán vị date1 với date2, tiếp tục bước 2 | |
| Open Issues | N/A | |

|  |  |  |
| --- | --- | --- |
| User Case ID | UC\_4 | |
| Name | UC\_QLNhanSu | |
| Goal | Quản lý nhân sự của nhà sách | |
| Actors | QuanLy | |
| Pre-conditions | User đã đănng nhập thành công với vai trò QuanLy | |
| Post-conditions |  | |
| Main Flow | 2.User chọn chức năng | 1. Hệ thống xuất danh sách nhân viên  3. Hệ thống cho phép user thực hiện chức năng. |
| Exception |  | |
| Open Issues | N/A | |

|  |  |  |
| --- | --- | --- |
| User Case ID | UC\_5 | |
| Name | UC\_ThemNV | |
| Goal | Thêm nhân viên mới | |
| Actors | QuanLy | |
| Pre-conditions | User đã đănng nhập thành công với vai trò QuanLy | |
| Post-conditions |  | |
| Main Flow | 1. User nhập username, password, vai trò của nhân viên mới. | 2. Hệ thống lưu thông tin nhân viên mới vào kho dữ liệu. |
| Exception | 2A: Nếu username trống, báo lỗi “username khải không trống” và quay lại bước 1  2B: Nếu password dưới 6 ký tự, báo lỗi “Password: ít nhất 6 ký tự” và quay lại bước 1  2C: Nếu username trùng với các username đang có, báo lỗi “Trùng tên” và quay lại bước 1 | |
| Open Issues | N/A | |

|  |  |  |
| --- | --- | --- |
| User Case ID | UC\_6 | |
| Name | UC\_SuaNV | |
| Goal | Sửa thông tin nhân viên | |
| Actors | QuanLy | |
| Pre-conditions | User đã đănng nhập thành công với vai trò QuanLy | |
| Post-conditions |  | |
| Main Flow | 1. User nhập username của nhân viên cần sửa  3. User nhập password, vai trò mới của nhân viên hiện hành. | 2. Hệ thống xuất thông tin của nhân viên đã được chọn.  4. Hệ thống cập nhật thông tin về nhân viên này. |
| Exception | 2A: Nếu username không tồn tại, báo lỗi :User không tồn tại”, quay lại bước 1  2B: Nếu username lại là chính user QuanLy, báo lỗi “Không thề xoá chính mình”, về bước 1  4A: Nếu password dưới 6 ký tự, báo lỗi “Password: ít nhất 6 ký tự” và quay lại bước 3  4B: Nếu vai trò không trùng với các vai trò đang có, báo lỗi “Vai trò không hợp lệ” và quay lại bước 3 | |
| Open Issues | N/A | |

**1.4.3- Nhóm use case cho vai trò kế toán**

***Bạn tự làm***

**1.4.4- Nhóm use case cho nhân viên**

***Bạn tự làm***

**1.4.5- Nhóm use case cho khách hàng**

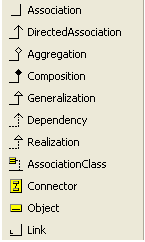
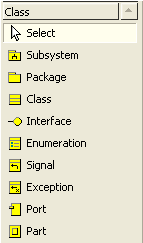
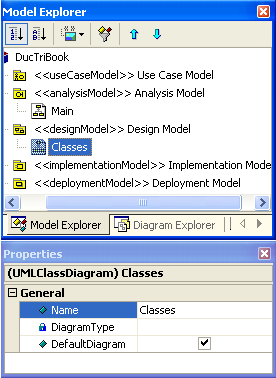
***Bạn tự làm***

1. **Thiết kế**
   1. **Sơ đồ lớp cần có của hệ thống**

***Phần giải thích***

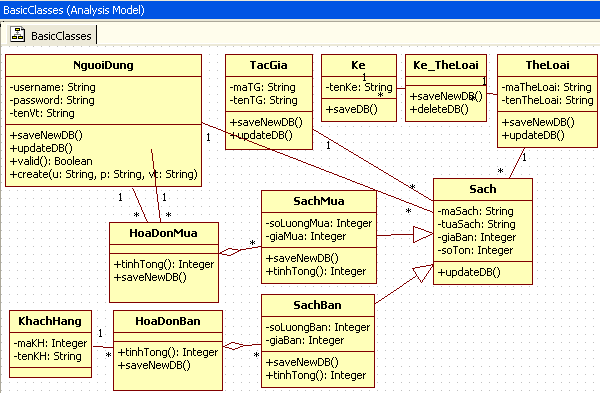
*Sơ đồ lớp là sơ đồ tĩnh vì sơ đồ này chỉ cho biết các bản mô tả về các đối tượng có trong hệ thống. Các lớp không khó được nhận diện. Gợi ý sau đây giúp nhận diện các lớp và tạo Class Diagrams:*

* *Các lớp ánh xạ cho các bảng trong database- thường thì một bảng sẽ ứng với một lớp.*
* *Các lớp tạo tiện ích như các danh sách ( chứa danh sách lấy từ database ra hoặc chứa danh sách trước khi lưu vào database).*
* *Các lớp đóng vai trò giao diện người dùng (GUI)*
* *Một số chú ý:*
  + *Class Diagrams có thể trong Analysis Model hoặc Design Model đều được vì chỉ trong hai nhóm này mới cho ta tạo class diagram.*
  + *Khai thác tính thừa kế để đơn giản hoá các lớp*
  + *Khi tạo 1 class, chỉ khai báo các dữ liệu và hành vi cần thiết. Những dữ liệu giúp mô tả quan hệ giữa các lớp cũng không cần mô tả mà dùng chức năng Association. Những hành vi cơ bản như constructors, getters, setters không cần thiết kế vì khi StarUML sinh code****- chỉ sinh ra khung của lớp – các hành vi được sinh ra đều code trống-***  *các construtor được xem như hành vi* ***bình thường ( có kiểu trả trị) trong khi đó các môi trường lập trình như NetBeans đã hỗ trợ đầy đủ các đặc điểm này. Khi sinh code, các quan hệ sẽ được mô tả tự động bằng biến tham khảo đến lớp mà lớp này có quan hệ. Việc sinh code được đề cập sau.***
  + *Các công cụ giúp thao tácvới class Diagram:*



Từ các phân tích về dữ liệu và các usecase ở trên, các lớp sau được nhận diện:

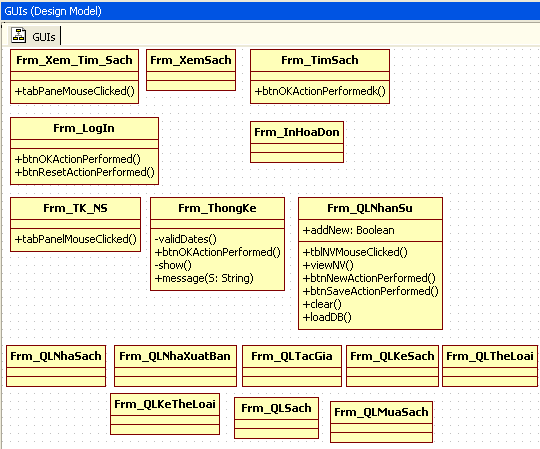
**2.1.1- Các lớp ánh xạ với kho dữ liệu**



*Các bạn làm rõ thêm các hành vi cho các lớp. Thường một lớp ánh xạ với bảng thường có các hành vi: nạp, tạo mới, sửa, xoá, đọc một danh sách các dòng ( có thể theo một điều kiện nào đó phù hợp với bài toán).*

**2.1.2- Các lớp tạo giao diện người**

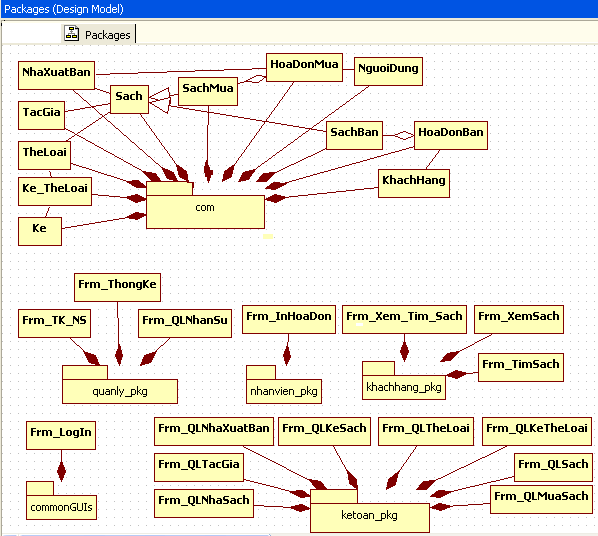
Mỗi chức năng của chương trình (usecase) cần có một giao diện người dùng ( GUI-các class). Từ phân tích usecase ở trên, các giao diện người dùng được cần đến ứng với nhu cầu tương tác của từng actor:



***Hướng dẫn cách thực hiện***

* *Trong* ***Model Explorer****, kích phải vào* ***Design Model****, chọn* ***Add Diagram/Class diagram***
* *Mở cửa sổ* ***Properties*** *đổi tên class diagram này là* ***Interfaces***
* *Dùng chức năng* ***class*** *trong* ***Class Toolbox*** *để vẽ các lớp GUI.*
* *Nếu ứng dụng được nhắm đến là ứng dụng chạy trên mạng, các interface (bộ hành vi giao tiếp lập trình) được cần đến, chúng ta cũng làm tương tự.*
* *Để thay đổi định dạng các đối tượng trên sơ đồ: chọn 1 đối tượng ( hoặc dùng chuột kéo đánh dấu một số đối tượng, hoặc Ctrl A để đánh dấu mọi đối tượng), vào menu format, chọn* ***Suppress Attributes/ Suppress Operations***.
  1. **Thiết kế gói**

Vì user được chia thành 5 vai trò và để có những thuận lợi trong việc bảo trì hệ thống, 6 gói sau được chọn:



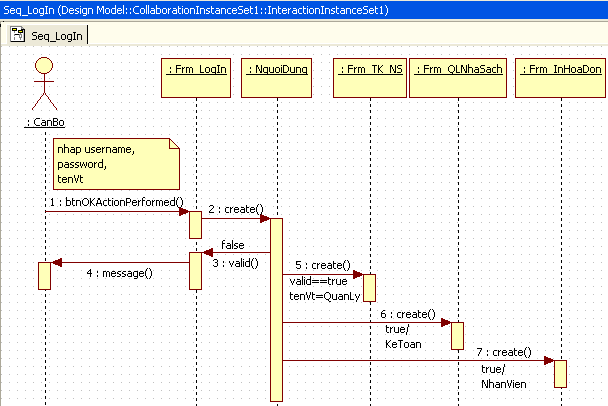
***Hướng dẫn cách tiến hành***

* *Vào* ***Model Exporer****, thêm một* ***Class diagram*** *vào* ***Design model****, đặt tên là* ***Packages***
* *Dùng chức năng  trong* ***Toolbox*** *để vẽ các packages*
* *Kéo các lớp từ* ***Model Explorer*** *vào màn hình thiết kế*
* *Chọn tất cả các phần tử trong màn hình thiết kế ( dùng chuột đánh dấu hoặc Ctrl A)*
* *Vào* ***Menu Format/Suppress Attributes*** *và* ***Suppress Operations*** *để ẩn attributes và operations giúp các lớp chỉ còn tên lớp.*
* *Dùng chức năng  trong* ***Toolbox*** *giúp vẽ các đường chỉ định gói này chứa các lớp nào.*
  1. **Sơ đồ trình tự cho các usecase**

**2.3.1- Sơ đồ trình tự cho usecase LogIn**

**Mục đích:**  Tạo giao diện và cơ chế hoạt động để cán bộ quản lý, kế toán nhân viên của nhà sach đăng nhập vào hệ thống để làm việc.

**Sơ đồ:**.



***Hướng dẫn cách tiến hành***

1. *Kích phải vào Design Model trong* ***Model Explorer***
2. *Add* ***Diagram/ Sequence Diagram****. StartUML tự động tạo một* ***CollaborationInstanceSet****, dưới đó có một* ***InteractionInstanceSet*** *và lược đồ của chúng ta nằm dưới nút này ( xem hình trên)*
3. *Đặt tên SeqLogIn*
4. *Kéo các đối tượng ( actor, class) cần có trong lược đồ từ* ***Model Explorer****.*
5. *Dùng chức năng* ***Stimulus*** *và* ***SelfStimulus*** *để vẽ các bước thao tác. StartUML không hiển thị tham số của các lời gọi hàm.*
6. *Dùng Annotaion ToolBox để đưa vào các text giải thích nếu cần.*

**Đặc tả hành vi:**

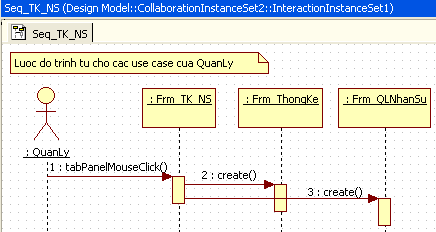
1. Người dùng gọi giao diện Login
2. Người dùng nhập dữ liệu bao gồm: username, password, vai trò
3. Hệ thống kiểm tra tính hợp lệ của người dùng
   1. Nếu dữ liệu không hợp lệ, một thông báo “Dữ liệu không hợp lệ!”
   2. Ngược lại
      1. Nếu vai trò=QuanLy, hệ thống sẽ xuất giao diện của người quản lý.
      2. Nếu vai trò=KeToan, hệ thống sẽ xuất giao diện của người kế toán.
      3. Nếu vai trò=NhanVien, hệ thống sẽ xuất giao diện của người nhân viên.

**2.3.2- Sơ đồ trình tự cho các usecase của actor QuanLy**

**Sơ đồ trình tự cho các usecase TK\_NS**

**Mục đích:** Tạo giao diện và cơ chế hoạt động tổng hợp cho người quản lý làm công việc thống kê kinh doanh và quản lý nhân sự.

**Sơ đồ:**

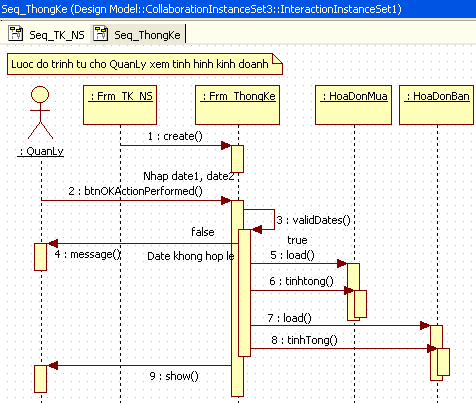


**Đặc tả hành vi:**

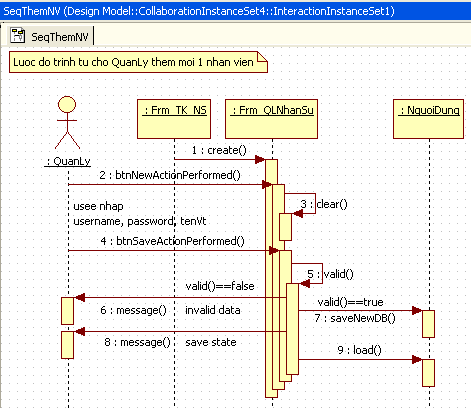
1. Người dùng chọn chức năng
2. Nếu chức năng =1 hệ thống sẽ xuất giao diện của ThongKe\_GUI

Ngược lại hệ thống sẽ xuất giao diện của QLNhanSu\_GUI

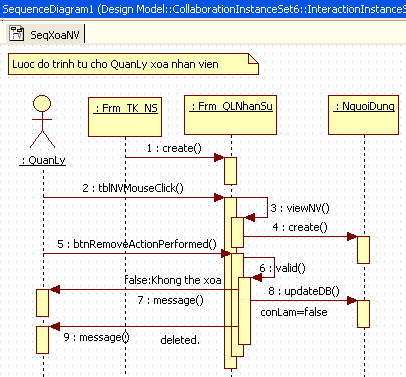
**Sơ đồ trình tự cho các usecase ThongKe**



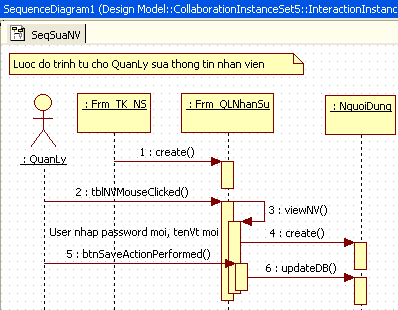
**Sơ đồ trình tự cho các usecase UC\_ThemNV**



**Lược đồ trình tự cho các usecase UC\_XoaNV**



**Lược đồ trình tự cho các usecase UC\_SuaNV**



**2.3.3- Sơ đồ trình tự cho các usecase của actor Kế toán**

Bạn tự làm

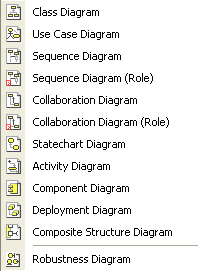
**2.3.4- Sơ đồ trình tự cho các usecase của actor NhanVien**

Bạn tự làm

**2.3.5- Sơ đồ trình tự cho các usecase của actor KhachHang**

Bạn tự làm

*Phần thiết kế này các bạn có thể làm rõ thêm thông qua các sơ đồ khác. Phần này để bạn tự xây dựng.*

**

***Sinh code***

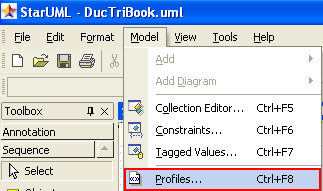
*Hầu hết các công cụ UML đề có chức năng sinh code giúp tạo khung cho các lớp đã thiết kế- tiết giảm công sức viết code- và StarUML cũng không ngoại lệ.*

*Để yêu cầu StarUML sinh code theo mõt ngôn ngữ nào đó ( thường là C++, C#, Java), người dùng phải chỉ định cho phầ mềm công cụ biết rằng mình muốn sinh code theo ngôn ngữ nào –* ***Xem lại cách ấn định Profiles*** *như đã được trình bầy ở phần đầu của tài liệu này.*

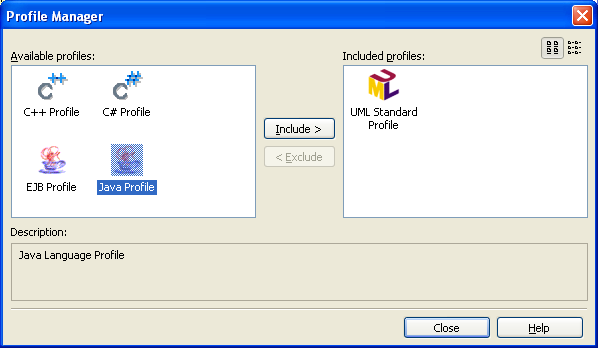
*Sau khi đã ấn định Profile cho StarUML, StarUML nhận diện các lớp cơ bản nhất của ngôn ngữ để hỗ trợ người dùng chọn kiểu của thuộc tính cũng như kiểu của trị trả về từ acc1 hành vi.*

***Cách add profile Java vào StarUML***

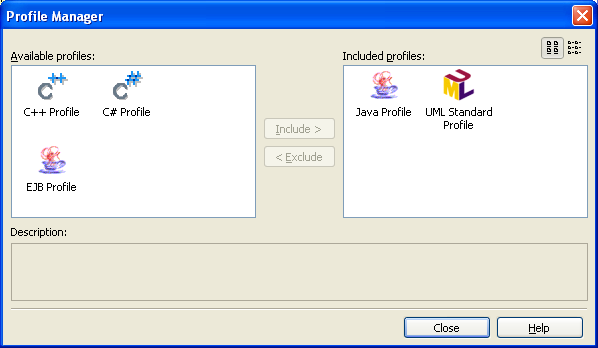
*Bước 1- Menu Model/ Profiles*

**

*Hộp thoại sau sẽ xuất hiện cho phép người dùng chọn profile được support bởi StarUML:*

**

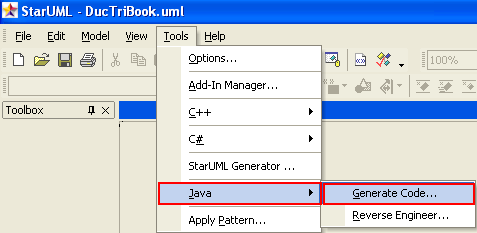
*Bước 2- Chọn Java Profile, kích nút include*

**

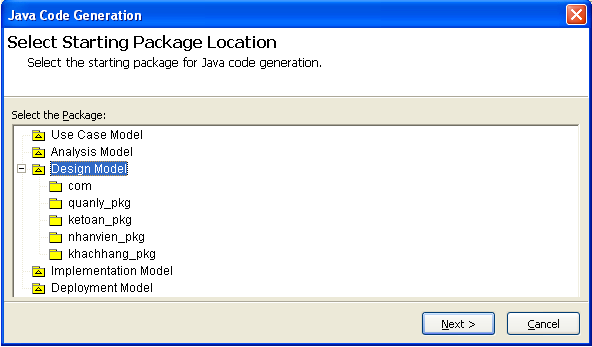
*Bước 3- Kích nút close*

***Sinh code cho mô hình***

*Bước 1- Menu*

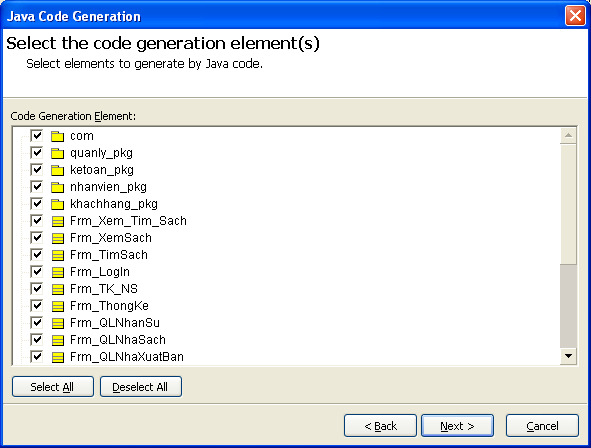
**

*Bước 2- Chọn thư mục chứa kết quả phân tích sẽ được sinh code.*

**

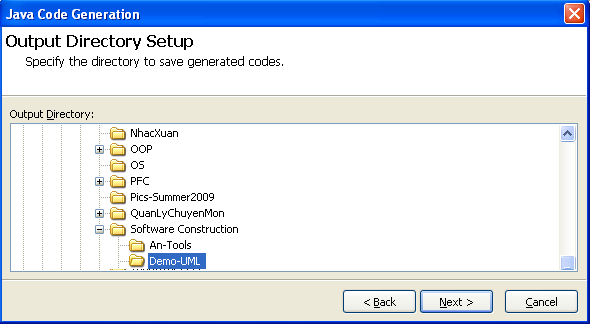
*Kích nút* ***Next***

*Bước 4- Chọn những thứ muốn sinh code (đánh dấu chọn)*



*Kích nút* ***Next***

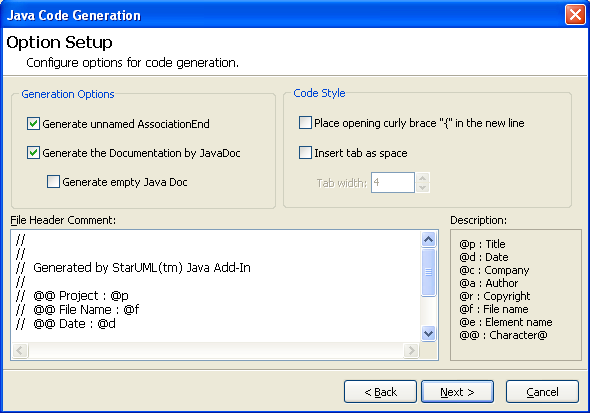
*Bước 5- Chọn thư mục chứa code*

**

*\*

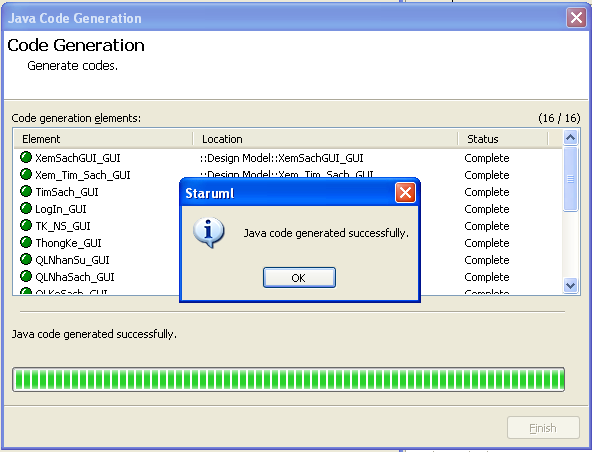
*Kích nút* ***Next***

*Bước 6- Chọn thư mục chứa code*

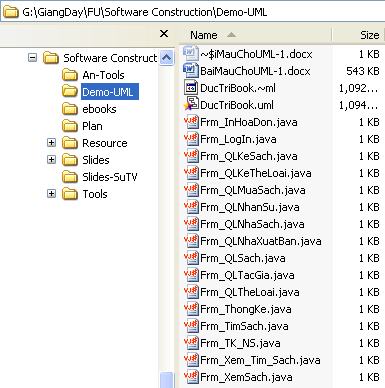
**

*Nếu đánh dấu vào mục* ***Generate unamed Association****, code của một lớp sẽ có thêm một biến tham khảo đến lớp có quan hệ.*

*Kích nút* ***Next***

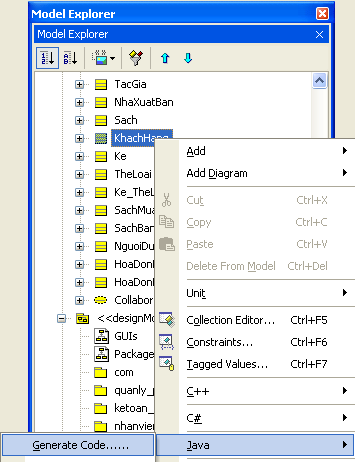
**

*Xem kết quả sinh code bằng Windows Explorer:*

**

***Chú ý: Các code sinh ra là các khai báo class với các hàm chứa code trống. Người lập trình phải tự cụ thể hoá code.***

*Bạn cũng có thể sinh code cho một đối tượng từ của số Model Explorer như sau:*

**