**Lotte Cinema Management**

**Coding document**

MỤC LỤC

[MỤC LỤC 2](#_Toc135458851)

[CHƯƠNG 1 – MÔ TẢ VỀ CHỨC NĂNG CỦA MÃ NGUỒN 18](#_Toc135458852)

[CHƯƠNG 2 – KIẾN TRÚC THIẾT KẾ 18](#_Toc135458853)

[1.1 Lớp DataAccessLayer (DAO) 18](#_Toc135458854)

[1.2 Lớp BusinessLogicLayer (BUS) 18](#_Toc135458855)

[1.3 Lớp PresentationLayer 18](#_Toc135458856)

[1.4 Lớp Data Tranfer Oject ( DTO) 18](#_Toc135458857)

[1.5 Singleton 18](#_Toc135458858)

[CHƯƠNG 3 – QUY CHUẨN LẬP TRÌNH 19](#_Toc135458859)

[3.1 Quy chuẩn về cách đặt tên 19](#_Toc135458860)

[CHƯƠNG 4 – CÁC CHỨC NĂNG VÀ HÀM THỰC THI 19](#_Toc135458861)

[4.1 Chức năng Đăng nhập 19](#_Toc135458862)

[4.1.1 Lớp GUI 19](#_Toc135458863)

[4.1.1.1 Hàm **btn\_Login\_Click – Form : Login** 19](#_Toc135458864)

[4.1.1.2 Hàm **btn\_Show\_Click – Form : Login** 20](#_Toc135458865)

[4.1.1.3 Hàm **btn\_Hide\_Click – Form : Login** 20](#_Toc135458866)

[4.1.1.4 Hàm **txt\_Password\_KeyDown – Form : Login** 21](#_Toc135458867)

[4.1.1.5 Hàm **txt\_User\_KeyDown – Form : Login** 21](#_Toc135458868)

[4.1.2 Lớp BUS 22](#_Toc135458869)

[4.1.2.1 Hàm **login – UserBUS** 22](#_Toc135458870)

[4.1.3 Lớp DTO 23](#_Toc135458871)

[4.1.4 Lớp DAO 23](#_Toc135458872)

[4.1.4.1 **Hàm login - UserDAO** 23](#_Toc135458873)

[4.2 Chức năng Đăng Kí 23](#_Toc135458874)

[4.3 Chức năng Quản lí Nguyên Vật liệu – Thêm nguyên vật liệu 23](#_Toc135458875)

[4.3.1 Lớp GUI 23](#_Toc135458876)

[4.3.1.1 Hàm **Supply\_Add\_Load – Form : Product\_List** 23](#_Toc135458877)

[4.3.1.2 Hàm **btn\_Add\_Click – Form : Product\_Add** 24](#_Toc135458878)

[4.3.1.3 Hàm **btn\_Reset\_Click – Form : Product\_Add** 25](#_Toc135458879)

[4.3.1.4 Hàm **btn\_Add\_Click – Form : Product\_List** 25](#_Toc135458880)

[4.3.2 Lớp BUS 26](#_Toc135458881)

[4.3.2.1 Hàm **create – ProductBUS** 26](#_Toc135458882)

[4.3.3 Lớp DTO 26](#_Toc135458883)

[4.3.4 Lớp DAO 26](#_Toc135458884)

[4.3.4.1 Hàm **create - ProductDAO** 26](#_Toc135458885)

[4.4 Chức năng Quản lí Nguyên Vật liệu – Sửa nguyên vật liệu 27](#_Toc135458886)

[4.4.1 Lớp GUI 27](#_Toc135458887)

[4.4.1.1 Hàm **btn\_Edit\_Click – Form : Product\_List** 27](#_Toc135458888)

[4.4.2.2 Hàm **btn\_Edit\_Click – Form : Product\_Edit** 27](#_Toc135458889)

[4.4.2 Lớp BUS 27](#_Toc135458890)

[4.4.2.1 Hàm **update - ProductBUS** 27](#_Toc135458891)

[4.4.3 Lớp DTO 28](#_Toc135458892)

[4.4.4 Lớp DAO 28](#_Toc135458893)

[4.4.4.1 Hàm **update - ProductDAO** 28](#_Toc135458894)

[4.5 Chức năng Quản lí Nguyên Vật liệu –Xóa nguyên vật liệu 28](#_Toc135458895)

[4.5.1 Lớp GUI 28](#_Toc135458896)

[4.5.1.1 Hàm **btn\_Delete\_Click – Form : Product\_Edit** 28](#_Toc135458897)

[4.5.2 Lớp BUS 29](#_Toc135458898)

[4.5.2.1 Hàm **delete – ProductBUS** 29](#_Toc135458899)

[4.5.3 Lớp DTO 29](#_Toc135458900)

[4.5.4 Lớp DAO 29](#_Toc135458901)

[4.5.4.1 Hàm **delete – ProductDAO** 29](#_Toc135458902)

[4.6 Chức năng Quản lí Nguyên Vật liệu –Hiển thị nguyên vật liệu 30](#_Toc135458903)

[4.6.1 Lớp GUI 30](#_Toc135458904)

[4.6.1.1 Hàm **Supply\_List\_Load – Form : Product\_List** 30](#_Toc135458905)

[4.6.2 Lớp BUS 30](#_Toc135458906)

[4.6.2.1 Hàm **view - ProductBUS** 30](#_Toc135458907)

[4.6.3 Lớp DTO 30](#_Toc135458908)

[4.6.4 Lớp DAO 30](#_Toc135458909)

[4.6.4.1 Hàm **selectAll - ProductDAO** 30](#_Toc135458910)

[4.7 Chức năng Quản lí Nguyên Vật liệu – Lọc nguyên vật liệu 31](#_Toc135458911)

[4.7.1 Lớp GUI 31](#_Toc135458912)

[4.7.1.1 Hàm **cbb\_SupplyType\_SelectedIndexChanged – Product\_List** 31](#_Toc135458913)

[4.7.2 Lớp BUS 32](#_Toc135458914)

[4.7.2.1 Hàm **filter - ProductBUS** 32](#_Toc135458915)

[4.7.3 Lớp DTO 32](#_Toc135458916)

[4.7.4 Lớp DAO 32](#_Toc135458917)

[4.7.4.1 Hàm **filterByType - ProductDAO** 32](#_Toc135458918)

[4.8 Chức năng Quản lí Nguyên Vật liệu – Tìm kiếm nguyên vật liệu 33](#_Toc135458919)

[4.8.1 Lớp GUI 33](#_Toc135458920)

[4.8.1.1 Hàm **txt\_Search\_TextChanged – Product\_List** 33](#_Toc135458921)

[4.8.2 Lớp BUS 33](#_Toc135458922)

[4.8.2.1 Hàm **search – ProductBUS** 33](#_Toc135458923)

[4.8.3 Lớp DTO 33](#_Toc135458924)

[4.8.4 Lớp DAO 33](#_Toc135458925)

[4.8.4.1 Hàm **search – ProductDAO** 33](#_Toc135458926)

[4.9 Chức năng Quản lí Nguyên Vật liệu – Đặt nguyên vật liệu 34](#_Toc135458927)

[4.9.1 Lớp GUI 34](#_Toc135458928)

[4.9.1.1 Hàm **btn\_Order\_Click – Form : Product\_List** 34](#_Toc135458929)

[4.9.1.2 Hàm **bnt\_Order\_Click – Form : Product\_Order** 34](#_Toc135458930)

[4.10 Chức năng Quản lí Phim– Hiển thị phim 35](#_Toc135458931)

[4.10.1 Lớp GUI 35](#_Toc135458932)

[4.10.1.1 Hàm **Movie\_List\_Load – Movie\_List** 35](#_Toc135458933)

[4.10.2 Lớp BUS 35](#_Toc135458934)

[4.10.2.1 Hàm **selectAll – MovieBUS** 35](#_Toc135458935)

[4.10.3 Lớp DTO 35](#_Toc135458936)

[4.10.4 Lớp DAO 35](#_Toc135458937)

[4.10.4.1 Hàm **selectAll – MovieDAO** 35](#_Toc135458938)

[4.11 Chức năng Quản lí Phim– Thêm Phim 36](#_Toc135458939)

[4.11.1 Lớp GUI 36](#_Toc135458940)

[4.11.1.1 Hàm **btn\_Add\_Click – Movie\_List** 36](#_Toc135458941)

[4.11.1.2 Hàm **btn\_Browse\_Click – Movie\_Add** 36](#_Toc135458942)

[4.11.1.3 Hàm **btn\_Create \_Click – Movie\_Add** 37](#_Toc135458943)

[4.11.2 Lớp BUS 38](#_Toc135458944)

[4.11.2.1 Hàm **create – MovieBUS** 38](#_Toc135458945)

[4.11.3 Lớp DTO 39](#_Toc135458946)

[4.11.4 Lớp DAO 39](#_Toc135458947)

[4.11.*4.1* Hàm **create – MovieDAO** 39](#_Toc135458948)

[4.12 Chức năng Quản lí Phim–Sửa thông tin Phim 40](#_Toc135458949)

[4.12.1 Lớp GUI 40](#_Toc135458950)

[4.12.1.1 Hàm **btn\_Edit\_Click – Form : Movie\_List** 40](#_Toc135458951)

[4.12.1.2 Hàm **btn\_Edit\_Click – Form : Movie\_Edit** 40](#_Toc135458952)

[4.12.1.2 Hàm **btn\_Browse\_Click** 40](#_Toc135458953)

[4.12.2 Lớp BUS 40](#_Toc135458954)

[4.12.1.1 Hàm **update - MovieBUS** 40](#_Toc135458955)

[4.12.3 Lớp DTO 41](#_Toc135458956)

[4.12.4 Lớp DAO 41](#_Toc135458957)

[4.12.4.1 Hàm **update – MovieDAO** 41](#_Toc135458958)

[4.13 Chức năng Quản lí Phim–Xóa Phim 42](#_Toc135458959)

[4.13.1 Lớp GUI 42](#_Toc135458960)

[4.13.1.1 Hàm **btn\_Delete\_Click – Movie\_List** 42](#_Toc135458961)

[4.13.2 Lớp BUS 42](#_Toc135458962)

[4.13.2.1 Hàm **delete – MovieBUS** 42](#_Toc135458963)

[4.13.3 Lớp DTO 42](#_Toc135458964)

[4.13.4 Lớp DAO 43](#_Toc135458965)

[4.13.4.1 Hàm **delete – MovieDAO** 43](#_Toc135458966)

[4.14 Chức năng Quản lí Nhân viên– Hiển thị danh sách nhân viên 43](#_Toc135458967)

[4.14.1 Lớp GUI 43](#_Toc135458968)

[4.14.1.1 Hàm **Staff\_List\_Load – Staff\_List** 43](#_Toc135458969)

[4.14.2 Lớp BUS 43](#_Toc135458970)

[4.14.2.1 Hàm **view – StaffBUS** 43](#_Toc135458971)

[4.14.3 Lớp DTO 43](#_Toc135458972)

[4.14.4 Lớp DAO 44](#_Toc135458973)

[4.14.4.1 Hàm **selectAll – StaffDAO** 44](#_Toc135458974)

[4.15 Chức năng Quản lí Nhân viên– Thêm Nhân viên 44](#_Toc135458975)

[4.15.1 Lớp GUI 45](#_Toc135458976)

[4.15.1.1 Hàm **btn\_Add\_Click – Form : Staff\_List** 45](#_Toc135458977)

[4.15.1.2 Hàm **btn\_Add\_Click – Form : Staff\_Add** 45](#_Toc135458978)

[4.15.2 Lớp BUS 45](#_Toc135458979)

[4.15.2.1 Hàm **register** 45](#_Toc135458980)

[4.15.3 Lớp DTO 45](#_Toc135458981)

[4.15.4 Lớp DAO 45](#_Toc135458982)

[4.15.4.1 Hàm **register - StaffDAO** 46](#_Toc135458983)

[4.16 Chức năng Quản lí Nhân viên– Sửa Nhân viên 46](#_Toc135458984)

[4.16.1 Lớp GUI 46](#_Toc135458985)

[4.16.1.1 Hàm **btn\_Edit\_Click – Form : Staff\_List** 46](#_Toc135458986)

[4.16.1.2 Hàm **btn\_Edit\_Click – Form : Staff\_Edit** 46](#_Toc135458987)

[4.16.2 Lớp BUS 47](#_Toc135458988)

[4.16.2.1 Hàm **update – StaffBUS** 47](#_Toc135458989)

[4.16.2.2 Hàm **getRoleID – RoleBUS** 47](#_Toc135458990)

[4.16.3 Lớp DTO 47](#_Toc135458991)

[4.16.4 Lớp DAO 47](#_Toc135458992)

[4.16.4.1 Hàm **update – StaffDAO** 47](#_Toc135458993)

[4.16.4.2 Hàm **selectByName** 47](#_Toc135458994)

[4.17 Chức năng Quản lí Nhân viên– Xóa Nhân viên 48](#_Toc135458995)

[4.17.1 Lớp GUI 48](#_Toc135458996)

[4.17.1.1 Hàm **btn\_Delete\_Click – Staff\_List** 48](#_Toc135458997)

[4.17.2 Lớp BUS 48](#_Toc135458998)

[4.17.2.1 Hàm **delete – StaffBUS** 48](#_Toc135458999)

[4.17.3 Lớp DTO 48](#_Toc135459000)

[4.17.4 Lớp DAO 48](#_Toc135459001)

[4.17.4.1 Hàm **delete – StaffDAO** 48](#_Toc135459002)

[4.18 Chức năng Quản lí Nhân viên– Tìm kiếm nhân viên 48](#_Toc135459003)

[4.18.1 Lớp GUI 49](#_Toc135459004)

[4.18.1.1 Hàm **txt\_Search\_TextChanged – Staff\_List** 49](#_Toc135459005)

[4.18.2 Lớp BUS 49](#_Toc135459006)

[4.18.2.1 Hàm **search – StaffBUS** 49](#_Toc135459007)

[4.18.1 Lớp DTO 49](#_Toc135459008)

[4.18.1 Lớp DAO 49](#_Toc135459009)

[4.18.1.1 Hàm **Search - StaffDAO** 49](#_Toc135459010)

[4.19 Chức năng đổi mật khẩu người dùng 50](#_Toc135459011)

[4.19.1 Lớp GUI 50](#_Toc135459012)

[4.19.1.1 Hàm **btn\_changepass\_Click – Staff\_List** 50](#_Toc135459013)

[4.19.1.2 **Hàm btn\_Confirm\_Click – Staff\_List** 50](#_Toc135459014)

[4.19.2 Lớp BUS 51](#_Toc135459015)

[4.19.1.1 Hàm **changePassword – StaffBUS** 51](#_Toc135459016)

[4.19.3 Lớp DTO 51](#_Toc135459017)

[4.19.4 Lớp DAO 51](#_Toc135459018)

[4.19.4.1 Hàm **changePass - StaffDAO** 51](#_Toc135459019)

[4.20 Chức năng chốt sổ 52](#_Toc135459020)

[4.20.1 Lớp GUI 52](#_Toc135459021)

[4.20.1.1 Hàm **btn\_Aggreate\_Click – Form : Aggreate** 52](#_Toc135459022)

[4.20.1.2 Hàm **Aggregate\_Load – Form : Aggreate** 52](#_Toc135459023)

[4.20.1.3 Hàm **btn\_confirm\_Click** 52](#_Toc135459024)

[4.20.2 Lớp BUS 53](#_Toc135459025)

[4.20.2.1 Hàm **create - RevenueBUS** 53](#_Toc135459026)

[4.20.3 Lớp DTO 53](#_Toc135459027)

[4.20.4 Lớp DAO 53](#_Toc135459028)

[4.20.4.1 Hàm **create - RevenueDAO** 53](#_Toc135459029)

[4.21 Chức năng Quản lí mã giảm giá – Hiển thị tất cả mã giảm giá 54](#_Toc135459030)

[4.20.1 Lớp GUI 54](#_Toc135459031)

[4.20.1.1 Hàm **Promotion\_List\_Load – Form : Promotion\_List** 54](#_Toc135459032)

[4.20.2 Lớp BUS 54](#_Toc135459033)

[4.20.2.1 Hàm **getAllPromotions - PromotionBUS** 54](#_Toc135459034)

[4.20.3 Lớp DTO 55](#_Toc135459035)

[4.20.4 Lớp DAO 55](#_Toc135459036)

[4.20.4.1 Hàm **getAllPromotions – PromotionDAO** 55](#_Toc135459037)

[4.22 Chức năng Quản lí mã giảm giá – Thêm mã giảm giá 56](#_Toc135459038)

[4.22.1 Lớp GUI 56](#_Toc135459039)

[4.22.1.1 Hàm **btn\_Add\_Click** 56](#_Toc135459040)

[4.22.1.2 Hàm **btn\_Create\_Click** 56](#_Toc135459041)

[4.22.2 Lớp BUS 56](#_Toc135459042)

[4.22.2.1 Hàm **addPromotion – PromotionBUS** 56](#_Toc135459043)

[4.22.3 Lớp DTO 57](#_Toc135459044)

[4.22.4 Lớp DAO 57](#_Toc135459045)

[4.22.4.1 Hàm **AddPromotion – PromotionDAO** 57](#_Toc135459046)

[4.23 Chức năng Quản lí mã giảm giá –Sửa mã giảm giá 58](#_Toc135459047)

[4.23.1 Lớp GUI 58](#_Toc135459048)

[4.23.1.1 Hàm **btn\_Edit\_Click() – Form : Promotion\_List** 58](#_Toc135459049)

[4.23.1.2 Hàm **btn\_Edit\_Click() – Form : Promotion\_Edit** 58](#_Toc135459050)

[4.23.2 Lớp BUS 58](#_Toc135459051)

[4.23.2.1 Hàm **UpdatePromotion – PromotionBUS** 58](#_Toc135459052)

[4.23.3 Lớp DTO 59](#_Toc135459053)

[4.23.4 Lớp DAO 59](#_Toc135459054)

[4.23.4.1 Hàm **UpdatePromotion – PromotionDAO** 59](#_Toc135459055)

[4.24 Chức năng Quản lí mã giảm giá –Xóa mã giảm giá 59](#_Toc135459056)

[4.24.1 Lớp GUI 59](#_Toc135459057)

[4.24.1.1 Hàm **btn\_Delete\_Click – Promotion\_List** 59](#_Toc135459058)

[4.24.2 Lớp BUS 60](#_Toc135459059)

[4.24.2.1 Hàm **DeletePromotion – PromotionBUS** 60](#_Toc135459060)

[4.24.3 Lớp DTO 60](#_Toc135459061)

[4.24.4 Lớp DAO 61](#_Toc135459062)

[4.24.4.1 Hàm **DeletePromotion - PromotionDAO** 61](#_Toc135459063)

[4.25 Chức năng Quản lí lịch chiếu phim – Hiển thị lịch chiếu 61](#_Toc135459064)

[4.25.1 Lớp GUI 61](#_Toc135459065)

[4.25.1.1 Hàm **Schedule\_List\_Load** 61](#_Toc135459066)

[4.25.1.2 Hàm **loadDGV** 62](#_Toc135459067)

[4.25.2 Lớp BUS 62](#_Toc135459068)

[4.25.3 Lớp DTO 62](#_Toc135459069)

[4.25.4 Lớp DAO 62](#_Toc135459070)

[4.25.2.1 Hàm **getScheduleOfShowroom - ScheduleDAO** 62](#_Toc135459071)

[4.26 Chức năng Quản lí lịch chiếu phim – Thêm lịch chiếu phim 63](#_Toc135459072)

[4.26.1 Lớp GUI 63](#_Toc135459073)

[4.26.1.1 Hàm **btn\_Add\_Click** 63](#_Toc135459074)

[4.26.1.2 Hàm **Schedule\_Add\_Load** 63](#_Toc135459075)

[4.26.1.3 Hàm **btn\_Create\_Click** 64](#_Toc135459076)

[4.26.2 Lớp BUS 64](#_Toc135459077)

[4.26.2.1 Hàm **create - ScheduleBUS** 64](#_Toc135459078)

[4.26.2.2 Hàm **loadMovie – ScheduleBUS** 65](#_Toc135459079)

[4.26.2.3 Hàm **loadShowRoom** 65](#_Toc135459080)

[4.26.3 Lớp DTO 66](#_Toc135459081)

[4.26.4 Lớp DAO 66](#_Toc135459082)

[4.26.4.1 Hàm **create – ScheduleDAO** 66](#_Toc135459083)

[4.26.4.2 Hàm **loadShowRoom – ScheduleDAO** 66](#_Toc135459084)

[4.26.4.3 Hàm **loadShowRoom – ScheduleDAO** 67](#_Toc135459085)

[4.27 Chức năng Quản lí lịch chiếu phim –Sửa lịch chiếu 68](#_Toc135459086)

[4.27.1 Lớp GUI 68](#_Toc135459087)

[4.27.1.1 Hàm **btn\_Edit\_Click – Form : Schedule\_List** 68](#_Toc135459088)

[4.27.1.2 Hàm **btn\_Edit\_Click – Form : Schedule\_Edit** 68](#_Toc135459089)

[4.27.2 Lớp BUS 68](#_Toc135459090)

[4.27.2.1 Hàm **update – ScheduleBUS** 68](#_Toc135459091)

[4.27.3 Lớp DTO 69](#_Toc135459092)

[4.27.4 Lớp DAO 69](#_Toc135459093)

[4.27.4.1 Hàm **update – ScheduleDAO** 69](#_Toc135459094)

[4.28 Chức năng Quản lí lịch chiếu phim –Xóa lịch chiếu 70](#_Toc135459095)

[4.28.1 Lớp GUI 70](#_Toc135459096)

[4.28.1.1 Hàm **btn\_Delete\_Click** 70](#_Toc135459097)

[4.28.2 Lớp BUS 70](#_Toc135459098)

[4.28.2.1 Hàm **delete – ScheduleBUS** 70](#_Toc135459099)

[4.28.3 Lớp DTO 70](#_Toc135459100)

[4.28.4 Lớp DAO 71](#_Toc135459101)

[4.28.4.1 Hàm **delete – ScheduleDAO** 71](#_Toc135459102)

[4.29 Chức năng Quản lí phòng chiếu–Hiển thị phòng chiếu 71](#_Toc135459103)

[4.29.1 Lớp GUI 71](#_Toc135459104)

[4.29.1.1 Hàm **Theater\_List\_Load** 71](#_Toc135459105)

[4.29.1.2 Hàm **GV\_DataLoad** 71](#_Toc135459106)

[4.29.2 Lớp BUS 71](#_Toc135459107)

[4.29.3 Lớp DTO 71](#_Toc135459108)

[4.29.4 Lớp DAO 71](#_Toc135459109)

[4.29.4.1 Hàm **selectAll – ShowroomDAO** 72](#_Toc135459110)

[4.30 Chức năng Quản lí phòng chiếu–Tạo phòng chiếu 72](#_Toc135459111)

[4.30.1 Lớp GUI 72](#_Toc135459112)

[4.30.1.1 Hàm **btn\_Add\_Click** 72](#_Toc135459113)

[4.30.1.1 Hàm **btn\_Create\_Click** 72](#_Toc135459114)

[4.30.2 Lớp BUS 72](#_Toc135459115)

[4.30.2.1 Hàm **create – ShowRoomBUS** 72](#_Toc135459116)

[4.30.3 Lớp DTO 73](#_Toc135459117)

[4.30.4 Lớp DAO 73](#_Toc135459118)

[4.30.4.1 Hàm **create – ShowRoomDAO** 73](#_Toc135459119)

[4.31 Chức năng Quản lí phòng chiếu –Sửa thông tin phòng chiếu 73](#_Toc135459120)

[4.31.1 Lớp GUI 73](#_Toc135459121)

[4.31.1.1 Hàm **btn\_Edit\_Click – Form : Showroom\_List** 73](#_Toc135459122)

[4.31.1.2 Hàm **btn\_Edit\_Click – Form : Showroom\_Edit** 74](#_Toc135459123)

[4.31.2 Lớp BUS 74](#_Toc135459124)

[4.31.2.1 Hàm **update – ShowRoomBUS** 74](#_Toc135459125)

[4.31.3 Lớp DTO 74](#_Toc135459126)

[4.31.4 Lớp DAO 74](#_Toc135459127)

[4.30.4.1 Hàm **update – ShowRoomDAO** 74](#_Toc135459128)

[4.32 Chức năng Quản lí phòng chiếu–Xóa phòng chiếu 75](#_Toc135459129)

[4.32.1 Lớp GUI 75](#_Toc135459130)

[4.32.1.1 Hàm **btn\_Delete\_Click – Showroom\_List** 75](#_Toc135459131)

[4.32.1 Lớp BUS 75](#_Toc135459132)

[4.32.1.1 Hàm **delete - ShowroomBUS** 75](#_Toc135459133)

[4.32.1 Lớp DTO 75](#_Toc135459134)

[4.32.1 Lớp DAO 76](#_Toc135459135)

[4.30.4.1 Hàm **delete – ShowroomDAO** 76](#_Toc135459136)

[4.33 Chức năng Quản lí Phim – Tìm kiếm Phim 76](#_Toc135459137)

[4.33.1 Lớp GUI 76](#_Toc135459138)

[4.33.1.1 Hàm **txt\_Search\_TextChanged – Showroom\_List** 76](#_Toc135459139)

[4.33.2 Lớp BUS 76](#_Toc135459140)

[4.8.2.1 Hàm **search – MovieBUS** 76](#_Toc135459141)

[4.33.3 Lớp DTO 76](#_Toc135459142)

[4.33.4 Lớp DAO 77](#_Toc135459143)

[4.33.4.1 Hàm **search – MovieDAO** 77](#_Toc135459144)

[4.34 Chức năng Quản lí doanh thu theo ngày – Xem doanh thu 77](#_Toc135459145)

[4.34.1 Lớp GUI 77](#_Toc135459146)

[4.34.1.1 Hàm **dtp\_Day\_ValueChanged** 77](#_Toc135459147)

[4.34.2 Lớp BUS 78](#_Toc135459148)

[4.34.2.1 Hàm **GetRevenuesByDay – RevenueBUS** 78](#_Toc135459149)

[4.34.3 Lớp DTO 78](#_Toc135459150)

[4.34.4 Lớp DAO 78](#_Toc135459151)

[4.34.4.1 Hàm **GetRevenuesByDay -**  **RevenueDAO** 78](#_Toc135459152)

[4.35 Chức năng Quản lí doanh thu theo ngày – Cập nhật doanh thu tiền mặt 79](#_Toc135459153)

[4.35.1 Lớp GUI 79](#_Toc135459154)

[4.35.1.1 Hàm **btn\_UpdateCash\_Click** 79](#_Toc135459155)

[4.35.2 Lớp BUS 79](#_Toc135459156)

[4.35.2.1 Hàm **UpdateRevenue - RevenueBUS** 79](#_Toc135459157)

[4.35.3 Lớp DTO 79](#_Toc135459158)

[4.35.4 Lớp DAO 80](#_Toc135459159)

[4.35.4.1 Hàm **UpdateRevenue – RevenueDAO** 80](#_Toc135459160)

[4.36 Chức năng Quản lí doanh thu theo tháng – Hiển thị doanh thu theo tháng. 80](#_Toc135459161)

[4.36.1 Lớp GUI 80](#_Toc135459162)

[4.36.1.1 Hàm **dtp\_Day\_ValueChanged** 80](#_Toc135459163)

[4.36.1.2 Hàm **loadChart** 80](#_Toc135459164)

[4.36.1.3 Hàm **dtp\_Day\_ValueChanged** 81](#_Toc135459165)

[4.36.2 Lớp BUS 81](#_Toc135459166)

[4.36.2.1 Hàm **GetRevenuesByMonth - RevenueBUS** 81](#_Toc135459167)

[4.36.3 Lớp DTO 82](#_Toc135459168)

[4.36.4 Lớp DAO 82](#_Toc135459169)

[4.36.4.1 Hàm **GetRevenuesByMonth - RevenueDAO** 82](#_Toc135459170)

[4.37 Chức năng Quản lí doanh thu theo tháng – Lập báo cáo gửi lên trụ sở chính 83](#_Toc135459171)

[4.37.1 Lớp GUI 83](#_Toc135459172)

[4.37.1.1 Hàm **btn\_Report\_Click -** 83](#_Toc135459173)

[4.37.1.2 Hàm **btn\_UpdateCash\_Click** 83](#_Toc135459174)

[4.37.1.3 Hàm **btn\_Confirm\_Click** 84](#_Toc135459175)

[4.37.2 Lớp BUS 84](#_Toc135459176)

[4.37.3 Lớp DTO 84](#_Toc135459177)

[4.37.4 Lớp DAO 84](#_Toc135459178)

[4.38 Chức năng Chốt sổ 84](#_Toc135459179)

[4.38.1 Lớp GUI 84](#_Toc135459180)

[4.38.1.1 Hàm **Aggregate\_Load** 84](#_Toc135459181)

[4.38.1.2 Hàm **btn\_Confirm\_Click – Form Aggregate** 84](#_Toc135459182)

[4.38.2 Lớp BUS 85](#_Toc135459183)

[4.38.2.1 Hàm **create – RevenueBUS** 85](#_Toc135459184)

[4.38.3 Lớp DTO 85](#_Toc135459185)

[4.38.4 Lớp DAO 85](#_Toc135459186)

[4.38.4.1 Hàm **add – RevenueDAO** 85](#_Toc135459187)

[4.38 Chức năng Đặt vé trực tuyến 86](#_Toc135459188)

[4.38.1 Lớp GUI 86](#_Toc135459189)

[4.38.1.1 Hàm **btn\_Ticket\_Click** 86](#_Toc135459190)

[4.39.1.2 Hàm **Ticket\_Movie\_Load** 86](#_Toc135459191)

[4.39.1.3 Hàm **btn\_Confirm\_Click – Form : Ticket\_Movie** 87](#_Toc135459192)

[4.39.1.4 Hàm **Ticket\_Schedule\_Load** 87](#_Toc135459193)

[4.39.1.5 Hàm **btn\_Confirm\_Click – Form : Ticket\_Schedule** 87](#_Toc135459194)

[4.39.1.5 Hàm **CreateSeats** 88](#_Toc135459195)

[4.39.1.6 Hàm **Ticket\_ComboFood\_Load** 94](#_Toc135459196)

[4.39.1.7 Hàm **insertUC\_ProductItem** 95](#_Toc135459197)

[4.39.1.8 Hàm **btn\_Confirm\_Click** 95](#_Toc135459198)

[4.39.1.9 Hàm **Ticket\_Infor\_Load** 96](#_Toc135459199)

[4.39.1.10 Hàm **btn\_CheckPromoCode\_Click** 97](#_Toc135459200)

[4.39.1.10 Hàm **btn\_Payment\_Click** 97](#_Toc135459201)

[4.39.2 Lớp BUS 97](#_Toc135459202)

[4.39.2.1 Hàm **getMoviesInSchedule : MovieBUS** 97](#_Toc135459203)

[4.39.2.2 Hàm **getFoodCombos : BookingBUS** 98](#_Toc135459204)

[4.39.2.3 Hàm **passDataToTicketInfor : BookingBUS** 98](#_Toc135459205)

[4.39.2.4 Hàm **checkPromotion: BookingBUS** 100](#_Toc135459206)

[4.39.3 Lớp DTO 101](#_Toc135459207)

[4.39.4 Lớp DAO 101](#_Toc135459208)

[4.39.4.1 Hàm **selectAll: MovieDAO** 101](#_Toc135459209)

[4.39.4.2 Hàm **getPromotionByCode: PromotionDAO** 102](#_Toc135459210)

[4.39.4.3 Hàm **getMoviesInSchedule : MovieDAO** 102](#_Toc135459211)

[4.39.4.4 Hàm **getByID: MovieDAO,ProductDAO,ScheduleDAO,** 103](#_Toc135459212)

[**ShowRoomDAO** 103](#_Toc135459213)

[4.40 Chức năng Thanh toán 105](#_Toc135459214)

[4.40.1 Lớp GUI 105](#_Toc135459215)

[4.40.1.1 Hàm **btn\_Confirm\_Click – Form : CashPayment** 105](#_Toc135459216)

[4.40.1.2 Hàm **btn\_Confirm\_Click – Form : ATMPayment** 106](#_Toc135459217)

[4.40.1.3 Hàm **btn\_Confirm\_Click – Form : MomoPayment** 106](#_Toc135459218)

[4.40.2 Lớp BUS 107](#_Toc135459219)

[4.40.2.1 Hàm **createBooking** 107](#_Toc135459220)

[4.40.3 Lớp DTO 108](#_Toc135459221)

[4.40.4 Lớp DAO 108](#_Toc135459222)

[4.40.4.1 Hàm **addFoodBooking : BookingDAO** 108](#_Toc135459223)

[4.40.4.2 Hàm **addProductBooking : BookingDAO** 109](#_Toc135459224)

[4.40.4.3 Hàm **Add : TicketDAO** 109](#_Toc135459225)

[4.41 Các hàm liên quan đến xử lí File 109](#_Toc135459226)

[4.41.1 Lớp DAO 109](#_Toc135459227)

[4.41.1.1 Hàm **SendPasswordResetEmail** 109](#_Toc135459228)

[4.41.1.2 Hàm **addRevenueToExcel** 110](#_Toc135459229)

[4.41.1.3 Hàm **addBillToExcel** 111](#_Toc135459230)

[4.41.1.4 Hàm **uploadToGUI** 112](#_Toc135459231)

[4.42 Các hàm kết nối và thực thi SQL 113](#_Toc135459232)

[4.42.1 Lớp DAO 113](#_Toc135459233)

[4.42.1.1 Hàm **ExecuteQuery – DataProvider** 113](#_Toc135459234)

[4.42.1.2 Hàm **ExecuteNonQuery – DataProvider** 114](#_Toc135459235)

[4.42.1.2 Hàm **ExecuteScalar - DataProvider** 114](#_Toc135459236)

[4.43 Các lớp hỗ trợ cho việc 115](#_Toc135459237)

[4.43.1 Movie DTO 115](#_Toc135459238)

[4.43.2 FoodCombo DTO 117](#_Toc135459239)

[4.43.3 Items DTO 118](#_Toc135459240)

[4.43.4 Product DTO 118](#_Toc135459241)

[4.43.4 Promotion DTO 120](#_Toc135459242)

[4.43.5 Revenue DTO 120](#_Toc135459243)

[4.43.6 Role DTO 122](#_Toc135459244)

[4.43.6 Revenue DTO 122](#_Toc135459245)

[4.43.7 Seat DTO 124](#_Toc135459246)

[4.43.8 Session 124](#_Toc135459247)

[4.43.9 Staff DTO 125](#_Toc135459248)

[4.43.10 Ticket DTO 127](#_Toc135459249)

[4.43.11 User DTO 128](#_Toc135459250)

CHƯƠNG 1 – MÔ TẢ VỀ CHỨC NĂNG CỦA MÃ NGUỒN

Mã nguồn này được viết để xây dựng một ứng dụng quản lý cho rạp phim Lotte Cinema quận 7. Nó cho phép người dùng thực hiện các nghiệp vụ quản lý trong rạp chiếu phim như: đặt vé, quản lý nhân viên, quản lý sản phẩm, quản lý lịch chiếu phim, quản lý các bộ phim,…. Ứng dụng được viết bằng ngôn ngữ C# và sử dụng mô hình 3 lớp để phân tách logic dữ liệu, logic nghiệp vụ và giao diện người dùng.

CHƯƠNG 2 – KIẾN TRÚC THIẾT KẾ

* 1. Lớp DataAccessLayer (DAO)

Lớp này chứa các phương thức để thao tác với database, bao gồm tạo mới, cập nhật, xóa và tìm kiếm. Nó sử dụng kết nối tới database để thực hiện các thao tác này.

* 1. Lớp BusinessLogicLayer (BUS)

Lớp này chứa các phương thức để xử lý nghiệp vụ của ứng dụng, bao gồm xác thực dữ liệu, kiểm tra tính hợp lệ và tính toán. Nó sử dụng các phương thức từ lớp DataAccessLayer để truy xuất dữ liệu từ database.

* 1. Lớp PresentationLayer

Lớp này chứa các form và control để hiển thị giao diện người dùng. Nó sử dụng các phương thức từ lớp BusinessLogicLayer để thực hiện các tính năng của ứng dụng.

* 1. Lớp Data Tranfer Oject ( DTO)

Lớp này đây không hẳn là layer, đây chỉ là 1 gói dữ liệu đươc trao đổi giữa các lớp. Gói dữ liệu này được xây dựng dưới dạng lớp đối tượng. Mục đích sử dụng chính của DTO đó là giảm số lần gọi các method giữa các tiến trình xử lý.

* 1. Singleton

Phần mêm được thiết kế theo quy tắt singleton để đảm bảo rằng chỉ có một phiên bản duy nhất của một lớp được tạo ra và cung cấp một điểm truy cập toàn cục đến phiên bản đó.

CHƯƠNG 3 – QUY CHUẨN LẬP TRÌNH

3.1 Quy chuẩn về cách đặt tên

* Đặt tên file:
* Các file trong lớp DAO và BUS cần có hậu tố thể hiện lớp mà file đó thuộc về. Ví dụ: StaffBUS, StaffDAO,…
* Các file trong DTO thể hiện tên của đối tượng mà nó thể hiện. Ví dụ: Staff, Theater,…
* Ở thư mục GUI: cần chia thành các thư mục và đặt tên theo nghiệp vụ mà nó thể hiện. Trong các thư mục này cần chia thành các file có tiền tố là tên của đối tượng và hậu tố là tên của chức năng. Ví dụ: staff\_list, ticket\_info, staff\_add,….
* Các biến được đặt tên thể hiện rõ chức năng của bản thân.
* Các biến có kiểu dữ liệu là List, ArrayList, … đặt tên theo kiểu dữ liệu bên trong và thêm hậu tố s, ví dụ tickets
* Tên phương thức thể hiện rõ chức năng của phương thức đó.
* Các tên nên đặt ngắn gọn và nêu bật lên chức năng chính.
* Tên biến luôn đặt theo chuẩn camelCases.
* Tên phương thức đặt theo chuẩn camelCases
* Quy chuẩn về độ dài của một dòng code: tránh viết các dòng code quá dài (80 ký tự) ngắt dòng để tăng khả năng đọc.

CHƯƠNG 4 – CÁC CHỨC NĂNG VÀ HÀM THỰC THI

4.1 Chức năng Đăng nhập

4.1.1 Lớp GUI

4.1.1.1 Hàm **btn\_Login\_Click – Form : Login**

Hàm **btn\_Login\_Click** : hàm này có chức năng bắt sự kiện đăng nhập của người dùng khi nhấn nút đăng nhập. Hệ thống sẽ lấy dữ liệu của người dùng nhập vào là username và password. Sau đó gọi đến hàm **login** của lớp BUS vào truyền đối số vào để kiểm tra đăng nhập.

Code :

  private void btn\_Login\_Click(object sender, EventArgs e)

        {

            if (BUS.StaffBUS.Instance.login(txt\_User.Text, txt\_Password.Text))

            {

                Admin frm = new Admin();

                frm.ShowDialog();

                this.Hide();

            }

            else

            {

                if (txt\_User.Text.Length == 0)

                {

                    lbl\_Error.Text = "Vui lòng nhập tên đăng nhập";

                }

                else if (txt\_Password.Text.Length == 0)

                {

                    lbl\_Error.Text = "Vui lòng nhập mật khẩu";

                }

                else

                {

                    lbl\_Error.Text = "Tên tài khoản hoặc mật khẩu không chính xác";

                }

            }

        }

4.1.1.2 Hàm **btn\_Show\_Click – Form : Login**

Hàm **btn\_Show\_Click** : hàm để bắt sự kiện nhấn vào kí tự hiện mật khẩu.

        private void btn\_Show\_Click(object sender, EventArgs e)

        {

            if (txt\_Password.PasswordChar == '●')

            {

                btn\_Hide.BringToFront();

                txt\_Password.PasswordChar = '\0';

            }

        }

4.1.1.3 Hàm **btn\_Hide\_Click – Form : Login**

Hàm **btn\_Show\_Click**  : hàm để bắt sự kiện nhấn vào kí tự ẩn mật khẩu.

Code :

        private void btn\_Hide\_Click(object sender, EventArgs e)

        {

            if (txt\_Password.PasswordChar == '\0')

            {

                btn\_Show.BringToFront();

                txt\_Password.PasswordChar = '●';

            }

        }

4.1.1.4 Hàm **txt\_Password\_KeyDown – Form : Login**

Hàm **txt\_Password\_KeyDown** :bắt sự kiện nhấn nút Enter của người dùng khi con trỏ đang trong txt\_password. Hàm sẽ nhận thông tin đăng nhập

Code :

  private void txt\_Password\_KeyDown(object sender, KeyEventArgs e)

        {

            if (e.KeyCode == Keys.Enter)

            {

                if (BUS.StaffBUS.Instance.login(txt\_User.Text, txt\_Password.Text))

                {

                    Admin frm = new Admin();

                    frm.ShowDialog();

                    this.Hide();

                }

                else

                {

                    if (txt\_User.Text.Length == 0)

                    {

                        lbl\_Error.Text = "Vui lòng nhập tên đăng nhập";

                    }

                    else if (txt\_Password.Text.Length == 0)

                    {

                        lbl\_Error.Text = "Vui lòng nhập mật khẩu";

                    }

                    else

                    {

                        lbl\_Error.Text = "Tên tài khoản hoặc mật khẩu không chính xác";

                    }

                }

            }

        }

4.1.1.5 Hàm **txt\_User\_KeyDown – Form : Login**

Hàm **txt\_User\_KeyDown** :bắt sự kiện nhấn nút Enter của người dùng khi con trỏ đang trong txt\_user.

Code :

 private void txt\_User\_KeyDown(object sender, KeyEventArgs e)

        {

            if (e.KeyCode == Keys.Enter)

            {

                if (BUS.StaffBUS.Instance.login(txt\_User.Text, txt\_Password.Text))

                {

                    Admin frm = new Admin();

                    frm.ShowDialog();

                    this.Hide();

                }

                else

                {

                    if (txt\_User.Text.Length == 0)

                    {

                        lbl\_Error.Text = "Vui lòng nhập tên đăng nhập";

                    }

                    else if (txt\_Password.Text.Length == 0)

                    {

                        lbl\_Error.Text = "Vui lòng nhập mật khẩu";

                    }

                    else

                    {

                        lbl\_Error.Text = "Tên tài khoản hoặc mật khẩu không chính xác";

                    }

                }

            }

        }

4.1.2 Lớp BUS

4.1.2.1 Hàm **login – UserBUS**

Hàm Login nhận vào hai tham số là *email* và *password*, đại diện cho thông tin đăng nhập người dùng. Hàm sử dụng lớp **UserDAO** để gọi phương thức **Login** và thực hiện quá trình xác thực đăng nhập. Kết quả trả về từ phương thức **Login** là một giá trị *boolean*, biểu thị xem quá trình đăng nhập có thành công hay không.

Code :

        public bool Login(string email, string password)

        {

            return DAO.UserDAO.Instance.Login(email, password);

        }

4.1.3 Lớp DTO

Đối tượng **session** dùng để lưu trữ dữ liệu của người dùng khi người dùng đang sử dụng ứng dụng. Ở chức năng Đăng nhập, **session** đặt thuộc tính id, name và role.

4.1.4 Lớp DAO

4.1.4.1 **Hàm login - UserDAO**

Hàm **login** : gọi thủ tục “Login” trong Database để kiểm tra người dùng có tồn tại trong cơ sở dữ liệu hay không. Hàm cần 2 đối số *username, password*. Hàm gọi đến **ExecuteQuery** của lớp DataProvider để thực thi lệnh SQL và kết quả trả về dưới kiểu dữ liệu là DataTable. Xét Datatable, nếu tồn tại thì sẽ đặt lại dữ liệu cho đối tượng session lần lượt là *id,name,role* và trả về kết quả là True. Nếu không tồn tại thì trả về kết quả là False.

Code :

        public bool Login(string email, string password)

        {

            string procname = "exec Login @email , @password";

            using (DataTable dataTable = DataProvider.Instance.ExecuteQuery(procname, new object[] { email, password }))

            {

                DataRow row = dataTable.Rows[0];

                return row["Result"].ToString().Equals("true") ? true : false;

            }

        }

4.2 Chức năng Đăng Kí

4.3 Chức năng Quản lí Nguyên Vật liệu – Thêm nguyên vật liệu

4.3.1 Lớp GUI

4.3.1.1 Hàm **Supply\_Add\_Load – Form : Product\_List**

Hàm Supply\_Add\_Load bắt sự kiện đang thực hiện của Form Product\_Add.

Code :

        private void Supply\_Add\_Load(object sender, EventArgs e)

        {

            ProductBUS.Instance.getTypes(cbb\_SupplyType);

            cbb\_SupplyType.DataSource = ProductBUS.Instance.sellectAllProductType();

            //dtp\_Expiration.MaxDate = DateTime.Now;

            if (session.lang == null)

            {

                session.lang = "vi-VN";

            }

            SetLanguage(session.lang);

        }

4.3.1.2 Hàm **btn\_Add\_Click – Form : Product\_Add**

Hàm **btn\_Add\_Click** bắt sự kiện click chuột vào nút thêm nguyên vật liệu. Sau đó, hàm sẽ kiểm tra thông tin của nguyên vật liệu được nhập bằng các hàm **checkNumber** của **ValidatorHelper** có hợp lệ hay không . Nếu thông tin hợp lệ sẽ gọi đến hàm **create()** của lớp ProductBUS để thực hiện thêm nguyên vật liệu, nếu không hợp lệ thì hệ thống sẽ hiện lỗi và yêu cầu người dùng nhập lại.

Code :

  private void btn\_Add\_Click(object sender, EventArgs e)

        {

            // Check before Add

            if (cbb\_SupplyType.SelectedIndex == 0)

            {

                lbl\_Error.Text = "Vui lòng chọn loại hàng hóa";

                cbb\_SupplyType.Focus();

            }

            else if (txt\_SupplyName.Text.Length == 0)

            {

                lbl\_Error.Text = "Vui lòng nhập tên hàng hóa";

                txt\_SupplyName.Focus();

            }

            else if (BUS.ValidatorHelper.Instance.checkName(txt\_SupplyName.Text) == false)

            {

                lbl\_Error.Text = "Tên hàng hóa không được có ký tự";

                txt\_Quantity.Focus();

            }

            else if (txt\_Quantity.Text.Length == 0)

            {

                lbl\_Error.Text = "Vui lòng nhập số lượng";

                txt\_Quantity.Focus();

            }

            else if (BUS.ValidatorHelper.Instance.checkNumber(txt\_Quantity.Text) == false)

            {

                lbl\_Error.Text = "Số lượng không được có ký tự";

                txt\_Quantity.Focus();

            }

            else if (txt\_Price.Text.Length == 0)

            {

                lbl\_Error.Text = "Vui lòng nhập đơn giá";

                txt\_Price.Focus();

            }

            else if (BUS.ValidatorHelper.Instance.checkNumber(txt\_Price.Text) == false)

            {

                lbl\_Error.Text = "Đơn giá không được có ký tự";

                txt\_Price.Focus();

            }

            else // Add event

            {

                ProductBUS.Instance.create(txt\_SupplyName.Text, cbb\_SupplyType.Text,

                    float.Parse(txt\_Price.Text), int.Parse(txt\_Quantity.Text),dtp\_Expiry.Value, int.Parse(txt\_Quantity.Text));

                this.Close();

            }

        }

4.3.1.3 Hàm **btn\_Reset\_Click – Form : Product\_Add**

Hàm **btn\_Reset\_Click** bắt sự kiện click chuột vào nút “Đặt lại”. Hệ thống sẽ đặt lại thông tin giá trị rỗng và số lương bằng 0.

Chức năng : đặt lại thông tin đang nhập với giá trị rỗng và số lượng bằng 0.

4.3.1.4 Hàm **btn\_Add\_Click – Form : Product\_List**

Hàm, **btn\_Add\_Click** bắt sự kiện click chuột của người dùng khi nhấn vào nút thêm nguyên vật liệu trên Form Product\_List. Người dùng sẽ được chuyển đến Form Product\_Add.

Code :

  private void btn\_Reset\_Click(object sender, EventArgs e)

        {

            txt\_SupplyName.Text = string.Empty;

            txt\_Quantity.Text = string.Empty;

            txt\_Price.Text = string.Empty;

            cbb\_SupplyType.SelectedIndex = 0;

            dtp\_Expiry.Value = DateTime.Now;

        }

4.3.2 Lớp BUS

4.3.2.1 Hàm **create – ProductBUS**

Hàm **create** được truyền các tham số đầu vào là *string name, string type, string price, string quantity.* Hàm sẽ gọi đến hàm **create** của lớp **ProductDAO** và kiểm tra kết quả trả về . Nếu thành công thì sẽ hiển thị thành công. Còn không thì sẽ hiển thị không thành công.

Code :

        public void create(string name, string type, string price, string quantity)

        {

            int i = DAO.ProductDAO.Instance.create(new Product(name,float.Parse(price),0,type,0));

            if (i > 0)

            {

                MessageBox.Show("Thêm sản phẩm thành công");

            }

            else

            {

                MessageBox.Show("Thêm sản phẩm không thành công");

            }

        }

4.3.3 Lớp DTO

Đối tượng **Product** được sử dụng để lưu trữ thông tin của một nguyên vật liệu, ở đây sử dụng đối tượng Product với 4 đối số là *nam,type,price,quantity* vì *id* sẽ tăng ở cơ sở dữ liệu.

4.3.4 Lớp DAO

4.3.4.1 Hàm **create - ProductDAO**

Hàm **create** có đối số truyền vào là một đối tượng *Product* , gọi lệnh insert lên cơ sở dữ liệu để thểm dữ liệu vào bảng. Lệnh SQL sẽ được truyền vào hàm **ExecuteNonQuery** của lớp DAO.DataProvider. Kiểu dữ liệu trả về là *int*.

Code :

        public int create(Product product)

        {

            string sql = "INSERT INTO[dbo].[PRODUCT]([NAME],[TYPE],[PRICE],[QUANTITY])VALUES( @name , @type , @price , @quantity )";

            return DataProvider.Instance.ExecuteNonQuery(sql, new Object[] { product.Name, product.Type, product.Price, product.Quantity });

        }

4.4 Chức năng Quản lí Nguyên Vật liệu – Sửa nguyên vật liệu

4.4.1 Lớp GUI

4.4.1.1 Hàm **btn\_Edit\_Click – Form : Product\_List**

Hàm **btn\_Edit\_Click** sẽ bắt sự kiện của người dùng khi click chuột vào nút Sửa nguyên vật liệu. Sau đó Form Product\_Edit sẽ được mở ra và các dữ liệu của nguyên vật liệu đó sẽ được đổ vào các TextBox, Combobox tương ứng.

Code :

4.4.2.2 Hàm **btn\_Edit\_Click – Form : Product\_Edit**

Hàm **btn\_Edit \_Click** bắt sự kiện click chuột vào nút sửa nguyên vật liệu. Sau đó, hàm sẽ kiểm tra thông tin của nguyên vật liệu được nhập có hợp lệ hay không . Nếu thông tin hợp lệ sẽ gọi đến hàm **update** của lớp **ProductBUS** để thực hiện sửa nguyên vật liệu, nếu không hợp lệ thì hệ thống sẽ hiện lỗi và yêu cầu người dùng nhập lại.

4.4.2 Lớp BUS

4.4.2.1 Hàm **update - ProductBUS**

Hàm **update** được truyền các tham số đầu vào là *string id*,*string name, string type, string price, string quantity.* Hàm sẽ gọi đến hàm **update** của lớp ProductDAO và kiểm tra kết quả trả về . Nếu thành công thì sẽ hiển thị thành công. Còn không thì sẽ hiển thị không thành công.

Code :

        public void update(string id, string name, string type, string price, string quantity, int

lastOrder)

        {

            int i = DAO.ProductDAO.Instance.update(new Product(id,name,float.Parse(price),int.Parse(quantity),type, lastOrder));

            if (i > 0)

            {

                MessageBox.Show("Cập nhật thành công");

            }

            else

            {

                MessageBox.Show("Cập nhật không thành công");

            }

        }

4.4.3 Lớp DTO

Đối tượng **Product** được sử dụng để lưu trữ thông tin của một nguyên vật liệu, ở đây sử dụng đối tượng Product với 5 đối số là *id,nam,type,price,quantity*.

4.4.4 Lớp DAO

4.4.4.1 Hàm **update - ProductDAO**

Hàm **update** có đối số truyền vào là một đối tượng *Product* , gọi lệnh insert lên cơ sở dữ liệu để thểm dữ liệu vào bảng. Lệnh SQL sẽ được truyền vào hàm **ExecuteNonQuery** của lớp DAO.DataProvider. Kiểu dữ liệu trả về là *int*.

Chức năng : Hàm **update** dùng để truyền lệnh update và các tham số để thêm sửa nguyên vật liệu trên cơ sở dữ liệu.

Code :

        public int update(Product product)

        {

            string sql = "UPDATE [dbo].[PRODUCT]\r\n   SET [NAME] = @name ,[TYPE] = @type ,[PRICE] = @price ,[QUANTITY] = @quantity WHERE id = @id ";

            return DataProvider.Instance.ExecuteNonQuery(sql, new Object[] { product.Name,product.Type,product.Price,product.Quantity,product.ID });

        }

4.5 Chức năng Quản lí Nguyên Vật liệu –Xóa nguyên vật liệu

4.5.1 Lớp GUI

4.5.1.1 Hàm **btn\_Delete\_Click – Form : Product\_Edit**

Hàm **btn\_Delete\_Click** sẽ bắt sự kiện của người dùng khi click chuột vào nút xóa nguyên vật liệu và lấy id của nguyên vật liệu từ bảng Datagridview.

Chức năng : Hàm **btn\_Delete\_Click** là lấy id của nguyên vật liệu và xác nhận người dùng có xóa nguyên vật liệu hay không.

4.5.2 Lớp BUS

4.5.2.1 Hàm **delete – ProductBUS**

Hàm **delete** được truyền các tham số đầu vào là *string id*. Hàm sẽ gọi đến hàm **delete** của lớp ProductDAO và kiểm tra kết quả trả về . Nếu thành công thì sẽ hiển thị thành công. Còn không thì sẽ hiển thị không thành công.

Chức năng : Sửa nguyên vật liệu, kiểm tra và hiển thị kết quả .

Code :

        public void delete(string id)

        {

            int i = DAO.ProductDAO.Instance.delete(id);

            if (i > 0)

            {

                MessageBox.Show("Xóa thành công");

            }

            else

            {

                MessageBox.Show("Xóa không thành công");

            }

        }

4.5.3 Lớp DTO

4.5.4 Lớp DAO

4.5.4.1 Hàm **delete – ProductDAO**

Hàm **delete** có đối số truyền vào là *string id*, gọi lệnh delete lên cơ sở dữ liệu để xóa dữ liệu trên bảng. Lệnh SQL sẽ được truyền vào hàm **ExecuteNonQuery** của lớp DAO.DataProvider. Kiểu dữ liệu trả về là *int*.

Chức năng : Hàm **delete** dùng để truyền lệnh delete và tham số *id* để xóa sửa nguyên vật liệu trên cơ sở dữ liệu.

Code :

        public int delete(string id)

        {

            string sql = "DELETE FROM product WHERE id = @id ";

            return DataProvider.Instance.ExecuteNonQuery(sql, new Object[] { id });

        }

4.6 Chức năng Quản lí Nguyên Vật liệu –Hiển thị nguyên vật liệu

4.6.1 Lớp GUI

4.6.1.1 Hàm **Supply\_List\_Load – Form : Product\_List**

Hàm **Supply\_List\_Load** sẽ bắt sự kiện của đang hiển thị của Form Product\_List và hiển thị ra tất cả nguyên vật liệu. Sau đó gọi đến hàm **get\_Type** để hiển thị tất cả loại nguyên vật liệu lên Combobox.

4.6.2 Lớp BUS

4.6.2.1 Hàm **view - ProductBUS**

Hàm **view** gọi đến hàm **selectAll** của ProductDAO để lấy danh sách các nguyên vật liệu.

Chức năng : Lấy tất cả nguyên vật liệu.

Code :

        public List<Product> selectAll()

        {

            return DAO.ProductDAO.Instance.selectAll();

        }

4.6.3 Lớp DTO

Đối tượng **Product** được sử dụng để lưu trữ thông tin của một nguyên vật liệu, ở đây sử dụng đối tượng Product với 5 đối số là *id,nametype,price,quantity*.

4.6.4 Lớp DAO

4.6.4.1 Hàm **selectAll - ProductDAO**

Hàm **selectAll**  gọi lệnh select lên cơ sở dữ liệu để lấy tất cả nguyên vật liệu trên cơ sở dữ liệu. Lệnh SQL sẽ được truyền vào hàm **ExecuteQuery** của lớp DAO.DataProvider. Hàm sẽ thêm tất cả nguyên vật liệu vào List<Product> và trả về List<Product> này.

Chức năng : Hàm **selectAll** dùng để truyền lệnh select để lấy tất cả nguyên vật liệu trên cơ sở dữ liệu.

Code :

public List<Product> selectAll()

        {

            List<Product> products = new List<Product>();

            string sql = "Select \* from product order by [QUANTITY]";

            DataTable dataTable = DataProvider.Instance.ExecuteQuery(sql);

            foreach (DataRow row in dataTable.Rows)

            {

                string id = row["ID"].ToString();

                string name = row["NAME"].ToString();

                float price = float.Parse(row["Price"].ToString());

                int quantity = int.Parse(row["Quantity"].ToString());

                int lastOrder = int.Parse(row["LAST\_ORDER"].ToString());

                string type = row["TYPE"].ToString();

                //string expiration = row["EXPIRATION"].ToString();

                Product product = new Product(id, name, price, quantity, type, lastOrder);

                products.Add(product);

            }

            return products;

        }

4.7 Chức năng Quản lí Nguyên Vật liệu – Lọc nguyên vật liệu

4.7.1 Lớp GUI

4.7.1.1 Hàm **cbb\_SupplyType\_SelectedIndexChanged – Product\_List**

Hàm **cbb\_SupplyType\_SelectedIndexChanged**bắt sự kiện thay đổi trên combobox SupplyType để lọc nguyên vật liệu giữa Drink và Food. Khi combobox có giá trị thay đổi thì hàm này sẽ đưa giá trí vào hàm **filter** của ProductBUS để lấy danh sách các nguyên vật liệu đã được lọc và đưa vào hàm **GV\_LoadData** để hiển thị lên DataGirdView.

4.7.2 Lớp BUS

4.7.2.1 Hàm **filter - ProductBUS**

Hàm **filter** có tham số đầu vào là *string key* . Hàm sẽ trả về List<Product> lấy từ hàm **filterByType** của ProductDAO.

Code :

        public List<Product> filter(string key)

        {

            return DAO.ProductDAO.Instance.filterByType(key);

        }

4.7.3 Lớp DTO

Đối tượng **Product** được sử dụng để lưu trữ thông tin của một nguyên vật liệu, ở đây sử dụng đối tượng Product với 5 đối số là *id,name,type,price,quantity*.

4.7.4 Lớp DAO

4.7.4.1 Hàm **filterByType - ProductDAO**

Hàm **filterByType**  gọi lệnh select lên cơ sở dữ liệu với tham số là *string key* để lấy tất cả nguyên vật liệu có loại là *key* trên cơ sở dữ liệu. Lệnh SQL sẽ được truyền vào hàm **ExecuteQuery** của lớp DAO.DataProvider. Hàm sẽ thêm tất cả nguyên vật liệu vào List<Product> và trả về List<Product> này.

Chức năng : Hàm **filterByType** dùng để truyền lệnh select để lấy tất cả nguyên vật liệu có loại bằng với tham số *key* trên cơ sở dữ liệu.

Code :

        public List<Product> filterByType(string key)

        {

            List<Product> products = new List<Product>();

            string sql = "Select \* from Product where Type = @ProductType ";

            DataTable dataTable = DataProvider.Instance.ExecuteQuery(sql, new Object[] { key });

            foreach (DataRow row in dataTable.Rows)

            {

                string id = row["ID"].ToString();

                string name = row["NAME"].ToString();

                float price = float.Parse(row["Price"].ToString());

                int quantity = int.Parse(row["Quantity"].ToString());

                string type = row["TYPE"].ToString();

                int lastOrder = int.Parse(row["LAST\_ORDER"].ToString());

                Product product = new Product(id, name, price, quantity, type, lastOrder);

                products.Add(product);

            }

            return products;

        }

4.8 Chức năng Quản lí Nguyên Vật liệu – Tìm kiếm nguyên vật liệu

4.8.1 Lớp GUI

4.8.1.1 Hàm **txt\_Search\_TextChanged – Product\_List**

Hàm **txt\_Search\_TextChanged** bắt sự kiện thay đổi trên thanh tìm kiếm và lấy giá trị được nhập vào để đưa vào hàm **search** của ProductBUS.Sau đó lấy kết quả và đưa vào hàm **GV\_LoadData** để hiển thị trên màn hình.

4.8.2 Lớp BUS

4.8.2.1 Hàm **search – ProductBUS**

Hàm **search** có tham số đầu vào là *string key* . Hàm sẽ trả về List<Product> lấy từ hàm **search** của ProductDAO.

Code :

        public List<Product> search( string key)

        {

            return DAO.ProductDAO.Instance.search(key);

        }

4.8.3 Lớp DTO

Đối tượng **Product** được sử dụng để lưu trữ thông tin của một nguyên vật liệu, ở đây sử dụng đối tượng Product với 5 đối số là *id,name,type,price,quantity*.

4.8.4 Lớp DAO

4.8.4.1 Hàm **search – ProductDAO**

Hàm **search** gọi thủ tục “search\_product” trên cơ sở dữ liệu với tham số là *string key* để tìm kiếm các nguyên vật liệu liên quan đến *key* trên cơ sở dữ liệu. Lệnh SQL sẽ được truyền vào hàm **ExecuteQuery** của lớp DAO.DataProvider. Hàm sẽ thêm tất cả nguyên vật liệu vào List<Product> và trả về List<Product> này.

Code :

  public List<Product> search(string key)

        {

            List<Product> products = new List<Product>();

            string sql = "exec search\_product @type";

            DataTable dataTable = DataProvider.Instance.ExecuteQuery(sql, new Object[] { key });

            foreach (DataRow row in dataTable.Rows)

            {

                string id = row["ID"].ToString();

                string name = row["NAME"].ToString();

                string type = row["TYPE"].ToString();

                float price = float.Parse(row["Price"].ToString());

                int lastOrder = int.Parse(row["LAST\_ORDER"].ToString());

                int quantity = int.Parse(row["Quantity"].ToString());

                Product product = new Product(id, name, price, quantity, type, lastOrder);

                products.Add(product);

            }

            return products;

        }

4.9 Chức năng Quản lí Nguyên Vật liệu – Đặt nguyên vật liệu

4.9.1 Lớp GUI

4.9.1.1 Hàm **btn\_Order\_Click – Form : Product\_List**

Hàm **btn\_Order\_Click** bắt sự nhấn vào nút đặt hàng của người dùng và chuyền đến Form Product\_Order . Đồng thời , đặt lại giá trị cho các thông tin tương ứng.

Chức năng : đưa người dùng đến Form Product\_Order và hiện lên thông tin của nguyên vật liệu.

4.9.1.2 Hàm **bnt\_Order\_Click – Form : Product\_Order**

Hàm **btn\_Order\_Click** bắt sự nhấn vào nút đặt hàng của người dùng và kiếm tra số lượng của nguyên vật liệu . Nếu còn dưới 20% thì sẽ được đặt hàng và ngược lại .

Chức năng : Gửi yêu cầu đặt hàng lên hệ trụ sở chính.

4.10 Chức năng Quản lí Phim– Hiển thị phim

4.10.1 Lớp GUI

4.10.1.1 Hàm **Movie\_List\_Load – Movie\_List**

Hàm **Movie\_List\_Load** bắt sự kiện đang thực hiện của Form Movie\_List. Hàm này sẽ lấy *List<Movie>* từ hàm **selectAll** của **MovieDAO** và truyền vào hàm **GV\_DataLoad** để hiển thị lên màn hình.

4.10.2 Lớp BUS

4.10.2.1 Hàm **selectAll – MovieBUS**

Hàm gọi đến hàm **selectAll** của **MovieDAO** để lấy danh sách các bộ phim.

Code :

        public List<Movie> selectAll()

        {

           return  DAO.MovieDAO.Instance.selectAll();

        }

4.10.3 Lớp DTO

Đối tượng **Movie** được sử dụng để lưu trữ thông tin của một nguyên vật liệu, ở đây sử dụng đối tượng Movie với 5 đối số là *id, title, genre, duration, rating, story, poster*

4.10.4 Lớp DAO

4.10.4.1 Hàm **selectAll – MovieDAO**

Hàm **selectAll** gọi lệnh select lên cơ sở dữ liệu để lấy tất cả nguyên vật liệu trên cơ sở dữ liệu. Lệnh SQL sẽ được truyền vào hàm **ExecuteQuery** của lớp DAO.DataProvider. Hàm sẽ thêm tất cả nguyên vật liệu vào List<Movie> và trả về List< Movie > này.

Code :

        public List<Movie> selectAll()

        {

            List<Movie> movies = new List<Movie>();

            string sql = "Select \* from Movie";

            DataTable dataTable = DataProvider.Instance.ExecuteQuery(sql);

            foreach (DataRow row in dataTable.Rows)

            {

                string id = row["ID"].ToString();

                string title = row["title"].ToString();

                string genre = row["genre"].ToString();

                int duration = int.Parse(row["duration"].ToString());

                float rating = float.Parse(row["rating"].ToString());

                string story = row["story"].ToString();

                string poster = row["poster"].ToString();

                Movie movie = new Movie(id, title, genre, duration, rating, story, poster);

                movies.Add(movie);

            }

            return movies;

        }

4.11 Chức năng Quản lí Phim– Thêm Phim

*4.11.1 Lớp GUI*

4.11.1.1 Hàm **btn\_Add\_Click – Movie\_List**

Hàm **btn\_Add\_Click** bắt sự kiện click chuột vào nút thêm phim , hàm sẽ chuyển người dùng dến Form Movie\_Add.

4.11.1.2 Hàm **btn\_Browse\_Click – Movie\_Add**

Hàm **btn\_Browse\_Click()** bắt sự kiện nhấn vào nút browse dùng cho thêm poster phim dưới dạng hình ảnh.bằng hàm **uploadToGUI()** ở BUS.FileHelper.

Code :

  private void btn\_Browse\_Click(object sender, EventArgs e)

        {

            string[] flieName = BUS.FileHelper.Instance.uploadToGUI();

            if(flieName != null )

            {

                txt\_Poster.Text= flieName[1];

                this.fileName = flieName[0];

            }

            else

            {

                txt\_Poster.Text = "Lỗi upload file";

            }

        }

4.11.1.3 Hàm **btn\_Create \_Click – Movie\_Add**

Hàm **btn\_Create \_Click** bắt sự kiện click chuột vào nút thêm phim. Sau đó, hàm sẽ kiểm tra thông tin của phim được nhập có hợp lệ hay không . Nếu thông tin hợp lệ sẽ gọi đến hàm **create**của lớp MovieBUS để thực hiện thêm phim, nếu không hợp lệ thì hệ thống sẽ hiện lỗi và yêu cầu người dùng nhập lại.

Code :

private void btn\_Create\_Click(object sender, EventArgs e)

        {

            if (txt\_MovieName.Text.Length == 0)

            {

                lbl\_Error.Text = "Vui lòng nhập tên phim";

                txt\_MovieName.Focus();

                Default\_Label\_Color();

                lbl\_MovieName.ForeColor = Color.Red;

            }

            else if (cbb\_MovieType.SelectedIndex == 0)

            {

                lbl\_Error.Text = "Vui lòng chọn thể loại phim";

                cbb\_MovieType.Focus();

                Default\_Label\_Color();

                lbl\_MovieType.ForeColor = Color.Red;

            }

            else if (txt\_Duration.Text.Length == 0)

            {

                lbl\_Error.Text = "Vui lòng nhập thời lượng phim";

                txt\_Duration.Focus();

                Default\_Label\_Color();

                lbl\_Duration.ForeColor = Color.Red;

            }

            else if (BUS.ValidatorHelper.Instance.checkNumber(txt\_Duration.Text) == false)

            {

                lbl\_Error.Text = "Thời lượng phin không chứa ký tự";

                txt\_Duration.Focus();

                Default\_Label\_Color();

                lbl\_Duration.ForeColor = Color.Red;

            }

            else if (int.Parse(txt\_Duration.Text) <= 0)

            {

                lbl\_Error.Text = "Thời lượng phim phải lớn hơn 0";

                txt\_Duration.Focus();

                Default\_Label\_Color();

                lbl\_Duration.ForeColor = Color.Red;

            }

            else if (txt\_Rating.Text.Length == 0)

            {

                lbl\_Error.Text = "Vui lòng nhập đánh giá phim";

                txt\_Rating.Focus();

                Default\_Label\_Color();

                lbl\_Rating.ForeColor = Color.Red;

            }

            else if (BUS.ValidatorHelper.Instance.checkRatingNumber(txt\_Rating.Text) == false)

            {

                lbl\_Error.Text = "Đánh giá phim là số thập phân > 0 và <= 10";

                txt\_Rating.Focus();

                Default\_Label\_Color();

                lbl\_Rating.ForeColor = Color.Red;

            }

            else if (txt\_Story.Text.Length == 0)

            {

                lbl\_Error.Text = "Vui nhập mô tả của phim";

                txt\_Story.Focus();

                Default\_Label\_Color();

                lbl\_Story.ForeColor = Color.Red;

            }

            else if (txt\_Poster.Text.Length == 0)

            {

                lbl\_Error.Text = "Vui lòng lòng đưa hình ảnh của phim";

                txt\_Poster.Focus();

                Default\_Label\_Color();

                lbl\_Poster.ForeColor = Color.Red;

            }

            else

            {

                MovieBUS.Instance.create(txt\_MovieName.Text, cbb\_MovieType.SelectedItem.ToString(), Int32.Parse(txt\_Duration.Text), float.Parse(txt\_Rating.Text), txt\_Story.Text, this.fileName);

                this.Close();

            }

        }

4.11.2 Lớp BUS

4.11.2.1 Hàm **create – MovieBUS**

Hàm **create** có các tham số đầu vào *là string title, string genre, int duration, float rating, string story,string poster.* Hàm sẽ gọi đến hàm **create** của lớp MovieDAO và kiểm tra kết quả trả về . Nếu thành công thì sẽ hiển thị thành công. Còn không thì sẽ hiển thị không thành công.

Code :

 public void create(string title,

         string genre,

         int duration,

         float rating,

         string story,

         string poster)

        {

            int i = DAO.MovieDAO.Instance.create(title, genre, duration, rating, story, poster);

            if (i > 0)

            {

                MessageBox.Show("Thêm thành công");

            }

        }

4.11.3 Lớp DTO

Đối tượng **Movie** được sử dụng để lưu trữ thông tin của một nguyên vật liệu, ở đây sử dụng đối tượng Movie với 4 đối số là *title, genre, duration, rating, story, poster*

4.11.4 Lớp DAO

4.11.4.1 Hàm **create – MovieDAO**

Hàm **create** trong đoạn mã trên được sử dụng để tạo một bản ghi mới trong bảng "MOVIE" của cơ sở dữ liệu. Nó lấy các tham số *title, genre, duration, rating, story, poster*. Hàm sử dụng một truy vấn SQL INSERT để chèn các giá trị này vào bảng "MOVIE" và trả về số dòng bị ảnh hưởng bởi truy vấn (số bản ghi đã được chèn thành công).

Code :

        public int create(

         string title,

         string genre,

         int duration,

         float rating,

         string story,

         string poster)

        {

            string sql = "INSERT INTO [dbo].[MOVIE] ([TITLE] ,[GENRE] ,[DURATION] ,[RATING] ,[STORY] ,[POSTER]) VALUES ( @TITLE , @GENRE , @DURATION , @RATING , @STORY , @POSTER )\r\n";

            return DataProvider.Instance.ExecuteNonQuery(sql, new Object[] { title, genre, duration, rating, story, poster });

        }

4.12 Chức năng Quản lí Phim–Sửa thông tin Phim

4.12.1 Lớp GUI

4.12.1.1 Hàm **btn\_Edit\_Click – Form : Movie\_List**

Hàm **btn\_Edit\_Click** trong đoạn mã trên là một xử lý sự kiện được gọi khi người dùng nhấn vào nút "Edit" trên giao diện. Gán giá trị các trường dữ liệu trên form chỉnh sửa (frm) với các giá trị tương ứng của bản ghi phim được chọn từ danh sách (GV\_Data).

Vòng lặp for được sử dụng để lấy giá trị của trường *Poster* từ danh sách tất cả các phim (**selectAll**) và gán giá trị của phần tử cuối cùng vào trường *txt\_Poster* trên form chỉnh sửa.

4.12.1.2 Hàm **btn\_Edit\_Click – Form : Movie\_Edit**

Hàm **btn\_Edit\_Click** trong đoạn mã trên có chức năng thực hiện xử lý sự kiện khi người dùng nhấn vào nút "Edit" trên giao diện. Hàm sẽ kiểm tra và xử lý các điều kiện các trường . Nếu dữ liệu đủ và hợp lệ thì sẽ được thêm vào hàm **update**.

4.12.1.2 Hàm **btn\_Browse\_Click**

Hàm "btn\_Browse\_Click" trong đoạn mã trên có chức năng xử lý sự kiện khi người dùng nhấn vào nút "Browse".Hàm cho phép người dùng chọn một tệp hình ảnh từ hệ thống và hiển thị đường dẫn của tệp đó trong trường "txt\_Poster" và gọi đến hàm **uploadToGUI** từ lớp FileHelper để mở hộp thoại chọn tệp và lấy đường dẫn của tệp đã chọn

4.12.2 Lớp BUS

4.12.1.1 Hàm **update - MovieBUS**

Hàm **update** có các tham số của hàm là *string id,string title, string genre, int duration, float rating, string story, string poster.* Các tham số được truyền vào hàm **update** của **MovieDAO**.

Code :

  public void update(string title,

         string genre,

         int duration,

         float rating,

         string story,

         string poster,

         string id)

        {

            int i = DAO.MovieDAO.Instance.update(title, genre, duration, rating, story, poster, id);

            if (i > 0)

            {

                MessageBox.Show("Cập nhật thành công");

            }

        }

4.12.3 Lớp DTO

Đối tượng **Movie** được sử dụng để lưu trữ thông tin của một nguyên vật liệu, ở đây sử dụng đối tượng Movie với 5 đối số là *id*,*title, genre, duration, rating, story, poster*

4.12.4 Lớp DAO

4.12.4.1 Hàm **update – MovieDAO**

Các tham số của hàm bao gồm *id*,*title, genre, duration, rating, story, poster.* Xây dựng câu lệnh SQL để thực hiện cập nhật thông tin bản ghi phim. Câu lệnh SQL này sẽ cập nhật các trường *id*,*title, genre, duration, rating, story, poster.*  trong bảng "MOVIE" dựa trên giá trị của các tham số tương ứng và điều kiện @*id* = *id*.

Code :

        public int update(string title,

         string genre,

         int duration,

         float rating,

         string story,

         string poster,

         string id)

        {

            string sql = "UPDATE [dbo].[MOVIE]\r\n   SET [TITLE] = @TITLE ,[GENRE] = @GENRE ,[DURATION] = @DURATION ,[RATING] = @RATING  ,[STORY] = @STORY ,[POSTER] = @POSTER WHERE ID = @ID";

            return DataProvider.Instance.ExecuteNonQuery(sql, new Object[] { title, genre, duration, rating, story, poster, id });

        }

4.13 Chức năng Quản lí Phim–Xóa Phim

4.13.1 Lớp GUI

4.13.1.1 Hàm **btn\_Delete\_Click – Movie\_List**

Hàm **btn\_Delete\_Click**  bắt sự kiện nhấn vào nút “Xóa” . Nếu người dùng xác nhận muốn xóa bộ phim, thì hàm sẽ gọi hàm **delete** từ MovieBUS với tham số là *id* được lấy từ DataGridView. để thực hiện việc xóa. Sau đó, hàm gọi lại sự kiện "Movie\_List\_Load" để tải lại danh sách phim trong Form gọi.

4.13.2 Lớp BUS

4.13.2.1 Hàm **delete – MovieBUS**

Hàm **delete** gọi phương thức "delete" từ MovieDAO và truyền ID của bản ghi cần xóa. Lấy giá trị trả về (i) từ hàm **delete** và kiểm tra nếu giá trị đó lớn hơn 0, tức là xóa thành công, thì hiển thị hộp thoại thông báo "Xóa thành công".

Code :

        public void delete(string id)

        {

            int i = DAO.MovieDAO.Instance.delete(id);

            if (i > 0)

            {

                MessageBox.Show("Xóa thành công");

            }

        }

4.13.3 Lớp DTO

Đối tượng **Movie** được sử dụng để lưu trữ thông tin của một nguyên vật liệu, ở đây sử dụng đối tượng Movie với 5 đối số là *id*,*title, genre, duration, rating, story, poster*

4.13.4 Lớp DAO

4.13.4.1 Hàm **delete – MovieDAO**

Hàm **delete** tạo câu truy vấn SQL để xóa bản ghi phim từ bảng "Movie" dựa trên điều kiện *id = @id*. Gọi phương thức **ExecuteNonQuery** từ đối tượng DataProvider.Instance và truyền câu truy vấn SQL và tham số id. Trả về giá trị trả về của phương thức **ExecuteNonQuery**, thể hiện số lượng bản ghi bị ảnh hưởng (số lượng bản ghi đã bị xóa).

Code :

        public int delete(string id)

        {

            string sql = "DELETE FROM Movie WHERE id = @id ";

            return DataProvider.Instance.ExecuteNonQuery(sql, new Object[] { id });

        }

4.14 Chức năng Quản lí Nhân viên– Hiển thị danh sách nhân viên

4.14.1 Lớp GUI

4.14.1.1 Hàm **Staff\_List\_Load – Staff\_List**

Hàm **Staff\_List\_Load** được bắt sự kiện "Load" của Form. Hàm **Staff\_List\_Load** gọi hàm **view** từ StaffBUS để lấy danh sách nhân viên và sau đó gọi hàm **GV\_DataLoa** để hiển thị danh sách nhân viên lên DataGridView "GV\_Data".

4.14.2 Lớp BUS

4.14.2.1 Hàm **view – StaffBUS**

Hàm **view**  lấy danh sách nhân viên bằng cách gọi hàm **selectAll** từ lớp **StaffDAO**.

Code :

        public List<Staff> view()

        {

            return DAO.StaffDAO.Instance.selectAll();

        }

4.14.3 Lớp DTO

Đối tượng **Staff** được sử dụng để lưu trữ thông tin của một nguyên vật liệu, ở đây sử dụng đối tượng **Staff** với 11 đối số là *id, username, fname, lname, password, gender, birthday, city, role, phone, salary*

4.14.4 Lớp DAO

4.14.4.1 Hàm **selectAll – StaffDAO**

Hàm **selectAll** thực hiện truy vấn đến cơ sở dữ liệu để lấy tất cả các bản ghi từ bảng "staff" và trả về một List<Staff>. Các thông tin của mỗi bản ghi được trích xuất từ DataRow và tạo thành đối tượng Staff tương ứng, sau đó được thêm vào danh sách trước khi trả về. Các trường thông tin của đối tượng Staff bao gồm *id, username, fname, lname, password, gender, birthday, city, role, phone, salary*

Code :

        public List<Staff> selectAll()

        {

            List<Staff> users = new List<Staff>();

            string sql = "Select \* from staff";

            DataTable dataTable = DataProvider.Instance.ExecuteQuery(sql);

            foreach (DataRow row in dataTable.Rows)

            {

                string id = row["ID"].ToString();

                string username = row["email"].ToString();

                string fname = row["firstName"].ToString();

                string lname = row["lastName"].ToString();

                string pwd = row["password"].ToString();

                string gender = row["sex"].ToString();

                int role = int.Parse(row["ROLE\_ID"].ToString());

                string city = row["address"].ToString();

                string phone = row["phone"].ToString();

                float salary = float.Parse(row["salary"].ToString());

                DateTime? birthday = row["birthday"].ToString() == string.Empty ? null : (DateTime?)DateTime.Parse(row["birthday"].ToString());

                Staff useN = new Staff(id, username, fname, lname, pwd, gender, birthday, city, role, phone, salary);

                users.Add(useN);

            }

            return users;

        }

4.15 Chức năng Quản lí Nhân viên– Thêm Nhân viên

4.15.1 Lớp GUI

4.15.1.1 Hàm **btn\_Add\_Click – Form : Staff\_List**

Hàm **btn\_Add\_Click** bắt sự kiện người dùng nhấp vào nút "Thêm" (btn\_Add), Form "Staff\_Add" được hiển thị dưới dạng hộp thoại. Sau khi Form "Staff\_Add" được đóng lại, danh sách nhân viên được tải lại trong form gọi.

4.15.1.2 Hàm **btn\_Add\_Click – Form : Staff\_Add**

Hàm **btn\_Add\_Click** bắt sự kiện người dùng bấm vào nút “Thêm”.Hàm sẽ kiểm tra và xử lý dữ liệu nhập từ các trường, sau đó gọi hàm **register()** từ đối tượng StaffBUS để đăng ký nhân viên mới và hiển thị thông báo tương ứng.

4.15.2 Lớp BUS

4.15.2.1 Hàm **register**

Hàm nhận các thông tin nhân viên như *email, lastname, firstname, password ,gender ,birthday , phonenumber ,address ,role và salary.* Sau đó, hàm tạo một đối tượng Staff từ các thông tin này và gọi hàm **register** từ lớp **StaffDAO** để thực hiện thêm nhân viên vào cơ sở dữ liệu. Hàm trả về giá trị *boolean* để chỉ ra kết quả của việc đăng ký (thành công hay không).

Code :

        public bool register(string email, string fname, string lname, string pwd,

            string gender, DateTime birth, string phone, string city, string role, string salary)

        {

            Staff staff = new Staff(null, email, fname, lname, pwd, gender, birth, city, getRoleID(role), phone, float.Parse(salary));

            return DAO.StaffDAO.Instance.register(staff);

            //return false;

        }

4.15.3 Lớp DTO

Đối tượng **Staff** được sử dụng để lưu trữ thông tin của một nguyên vật liệu, ở đây sử dụng đối tượng **Staff** với 11 đối số là *id, username, fname, lname, password, gender, birthday, city, role, phone, salary*

4.15.4 Lớp DAO

4.15.4.1 Hàm **register - StaffDAO**

Hàm **register** thực hiện việc chèn thông tin nhân viên mới vào bảng "Staff" với mật khẩu được đăt mặc định là “123456” trong cơ sở dữ liệu. Nếu việc chèn thành công, hàm trả về giá trị true, ngược lại trả về giá trị false.

Code :

  public bool register(Staff staff)

        {

            staff.Password = "123456";

            string sql = "insert into Staff values( @email , @pwd , @fname , @lname , @gender , @birth ,  @phone , @city , @salary , @role )";

            int i = DataProvider.Instance.ExecuteNonQuery(sql, new object[] { staff.UserName, staff.Password, staff.Fname, staff.Lname, staff.Sex, staff.Birthday, staff.Phone, staff.City, staff.Salary, staff.Role });

            if (i == 0)

            {

                return false;

            }

            return true;

        }

4.16 Chức năng Quản lí Nhân viên– Sửa Nhân viên

4.16.1 Lớp GUI

4.16.1.1 Hàm **btn\_Edit\_Click – Form : Staff\_List**

Hàm **btn\_Edit\_Click** bắt được sự kiện khi người dùng nhấn nút "Edit". Nó mở giao diện Staff\_Edit để cho phép người dùng chỉnh sửa thông tin của nhân viên được chọn trong DataGridView. Sau khi người dùng hoàn tất chỉnh sửa và đóng giao diện Staff\_Edit, danh sách nhân viên trên DataGridView được cập nhật lại bằng cách gọi hàm Staff\_List\_Load..

4.16.1.2 Hàm **btn\_Edit\_Click – Form : Staff\_Edit**

Hàm sẽ kiểm tra tính hợp lệ của các trường thông tin như *username, fname, lname, password, gender, birthday, city, role, phone, salary* bằng các hàm **checkName, isValidEmail, checkPhone** của **DAO.ValidatorHelper** Nếu các trường thông tin đều hợp lệ, hàm sẽ gọi phương thức **update** từ đối tượng StaffBUS để cập nhật thông tin nhân viên. Sau đó, hàm sẽ hiển thị thông báo thành công nếu việc chỉnh sửa thành công, hoặc thông báo lỗi nếu không thành công.

4.16.2 Lớp BUS

4.16.2.1 Hàm **update – StaffBUS**

Hàm tạo một đối tượng Staff với các thông tin như *id,username, fname, lname, password, gender, birthday, city, role, phone, salary*. Sau đó, gọi hàm **update** từ lớp DAO.StaffDAO.Instance và trả về true nếu cập nhật thành công, ngược lại trả về false.

4.16.2.2 Hàm **getRoleID – RoleBUS**

Hàm **getRole** được truyền tham số là *string name* và trả về id của một Role.

Code :

        private int getRoleID(string id)

        {

            return int.Parse(RoleDAO.Instance.selectByName(id).ID);

        }

4.16.3 Lớp DTO

Đối tượng **Staff** được sử dụng để lưu trữ thông tin của một nguyên vật liệu, ở đây sử dụng đối tượng **Staff** với 11 đối số là *id, username, fname, lname, password, gender, birthday, city, role, phone, salary.*

4.16.4 Lớp DAO

4.16.4.1 Hàm **update – StaffDAO**

Hàm này gọi đến câu truy vấn SQL được sử dụng để thực hiện việc cập nhật thông tin nhân viên. Các tham số của câu truy vấn được thay thế bằng các giá trị tương ứng từ đối tượng **staff**. Cuối cùng, hàm **ExecuteNonQuery** của **DataProvider** được gọi để thực thi câu truy vấn và trả về số hàng bị ảnh hưởng (số bản ghi được cập nhật).

4.16.4.2 Hàm **selectByName**

Hàm **selectByName** thực hiện câu truy vấn SQL để tìm kiếm vai trò theo tên, sau đó tạo đối tượng Role và gán các giá trị từ dòng dữ liệu tương ứng vào đối tượng đó. Cuối cùng, nó trả về đối tượng Role chứa thông tin về vai trò tìm thấy.

4.17 Chức năng Quản lí Nhân viên– Xóa Nhân viên

4.17.1 Lớp GUI

4.17.1.1 Hàm **btn\_Delete\_Click – Staff\_List**

Hàm **btn\_Delete\_Click** sẽ bắt sự kiện của người dùng khi click chuột vào nút “xóa” và lấy id của nhân viên từ bảng Datagridview.

4.17.2 Lớp BUS

4.17.2.1 Hàm **delete – StaffBUS**

Hàm **delete** trong lớp BUS nhận vào một tham số **id** kiểu chuỗi đại diện cho mã nhân viên cần xóa. Hàm sử dụng lớp DAO để gọi phương thức **delete** và thực hiện xóa nhân viên dựa trên *id*. Kết quả trả về từ phương thức **delet** của **StaffDAO** được kiểm tra bằng toán tử ba ngôi, trả về true nếu số lượng bản ghi bị ảnh hưởng là 1 và ngược lại trả về false.

4.17.3 Lớp DTO

Đối tượng **Staff** được sử dụng để lưu trữ thông tin của một nguyên vật liệu, ở đây sử dụng đối tượng Movie với 11 đối số là *id, username, fname, lname, password, gender, birthday, city, role, phone, salary.*

4.17.4 Lớp DAO

4.17.4.1 Hàm **delete – StaffDAO**

Hàm **delete** thực hiện việc xóa nhân viên từ cơ sở dữ liệu bằng cách thực hiện câu lệnh SQL với tham số **id** tương ứng. Nó sử dụng lớp DataProvider và phương thức **ExecuteNonQuery** để thực thi câu lệnh SQL và trả về số lượng bản ghi bị ảnh hưởng trong quá trình xóa. Kết quả này được trả về để biểu thị số lượng nhân viên đã bị xóa.

4.18 Chức năng Quản lí Nhân viên– Tìm kiếm nhân viên

4.18.1 Lớp GUI

4.18.1.1 Hàm **txt\_Search\_TextChanged – Staff\_List**

Hàm **txt\_Search\_TextChanged** bắt sự kiện thay đổi ở TextBox Search. Giá trị trong TextBox sẽ được truyền vào hàm **search** của lớp StaffBUS. Kết quả trả về một List<Staff> và truyền vào hàm **GV\_DataLoa** để hiển thị lên màn hình.

4.18.2 Lớp BUS

4.18.2.1 Hàm **search – StaffBUS**

Hàm **search** thực hiện tìm kiếm nhân viên dựa trên từ khóa **key**. Hàm gọi phương thức **search** của lớp DA.StaffDAO để thực hiện quá trình tìm kiếm. Kết quả trả về từ phương thức **Search** là một danh sách các đối tượng **Staff** được tìm thấy dựa trên từ khóa. Hàm trả về danh sách này để biểu thị kết quả tìm kiếm.

Code :

        public List<Staff> search(string key)

        {

            return DAO.StaffDAO.Instance.Search(key);

        }

4.18.1 Lớp DTO

Đối tượng **Staff** được sử dụng để lưu trữ thông tin của một nguyên vật liệu, ở đây sử dụng đối tượng **Staff** với 11 đối số là *id, username, fname, lname, password, gender, birthday, city, role, phone, salary.*

4.18.1 Lớp DAO

4.18.1.1 Hàm **Search - StaffDAO**

Hàm **Search** nhận vào một tham số key kiểu chuỗi, đại diện cho từ khóa tìm kiếm. Hàm thực hiện việc tìm kiếm nhân viên dựa trên từ khóa key bằng cách gọi thủ tục lưu trữ **search\_staff** trong cơ sở dữ liệu. Hàm sử dụng lớp DataProvider để gọi hàm **ExecuteQuery** và thực hiện câu truy vấn SQL để tìm kiếm nhân viên. Kết quả trả về là một bảng dữ liệu DataTable chứa kết quả tìm kiếm. Hàm duyệt qua từng dòng trong dataTable và lấy các thông tin nhân viên từ các cột tương ứng. Sau đó, tạo đối tượng Staff mới và thêm vào danh sách users.

Code :

        public List<Staff> Search(string key)

        {

            List<Staff> users = new List<Staff>();

            string sql = "exec search\_staff @type";

            DataTable dataTable = DataProvider.Instance.ExecuteQuery(sql, new Object[] { key });

            foreach (DataRow row in dataTable.Rows)

            {

                int id = int.Parse(row["ID"].ToString());

                string username = row["email"].ToString();

                string fname = row["firstName"].ToString();

                string lname = row["lastName"].ToString();

                string pwd = row["password"].ToString();

                string gender = row["sex"].ToString();

                int role = int.Parse(row["ROLE\_ID"].ToString());

                string city = row["address"].ToString();

                string phone = row["phone"].ToString();

                float salary = float.Parse(row["salary"].ToString());

                DateTime? birthday = row["birthday"].ToString() == string.Empty ? null : (DateTime?)DateTime.Parse(row["birthday"].ToString());

                Staff useN = new Staff(id, username, fname, lname, pwd, gender, birthday, city, role, phone, salary);

                users.Add(useN);

            }

            return users;

        }

4.19 Chức năng đổi mật khẩu người dùng

4.19.1 Lớp GUI

4.19.1.1 Hàm **btn\_changepass\_Click – Staff\_List**

Hàm **btn\_changepass\_Click**  bắt sự kiện người dùng click chuột vào nút “Đổi mật khẩu”. Hàm sẽ mở Form đổi mật khẩu để cho người dùng nhập thông tin.

4.19.1.2 **Hàm btn\_Confirm\_Click – Staff\_List**

Hàm **btn\_Confirm\_Click** kiểm tra tính hợp lệ của mật khẩu cũ và mật khẩu mới, sau đó thực hiện thay đổi mật khẩu người dùng. Nếu mọi điều kiện đều đúng, hàm sẽ truyền các tham số vào hàm **changePassword** của lớp StaffBUS. Nếu kết quả của hàm **changePassword** trả về thành công thì hiển thị thông báo thành công và đóng cửa sổ hiện tại. Ngược lại, hàm hiển thị thông báo lỗi cho người dùng.

4.19.2 Lớp BUS

4.19.1.1 Hàm **changePassword – StaffBUS**

Hàm **changePassword** nhận vào ba tham số là *oldpassword* ,*newpassword* và **id** (ID của người dùng). Hàm sử dụng lớp DAO.StaffDAO để gọi phương thức **changePass** và thực hiện thay đổi mật khẩu của người dùng dựa trên các tham số truyền vào.Kết quả trả về từ phương thức **changePass** của DAO được kiểm tra bằng toán tử ba ngôi. Nếu số lượng bản ghi bị ảnh hưởng trong quá trình thay đổi mật khẩu là 1, hàm trả về true, ngược lại trả về false.

Code :

        public bool changePassword(string oldpassword, string newpassword, int id)

        {

            return DAO.StaffDAO.Instance.changePass(oldpassword, newpassword, id) == 1 ? true : false;

        }

4.19.3 Lớp DTO

4.19.4 Lớp DAO

4.19.4.1 Hàm **changePass - StaffDAO**

Sử dụng câu lệnh SQL để gọi thủ tục lưu trữ **change\_password** với các tham số như *oldpassword, newpassword, id (id của người dùng ).* Hàm trả về số lượng bản ghi bị ảnh hưởng trong quá trình thay đổi mật khẩu. Giá trị trả về là 1 nếu mật khẩu được thay đổi thành công, ngược lại là 0.

Code :

        public int changePass(string oldpassword, string newpassword, int id)

        {

            string sql = "exec change\_password @StaffID , @OldPassword ,  @NewPassword ";

            return DataProvider.Instance.ExecuteNonQuery(sql, new Object[] { id, oldpassword, newpassword });

        }

4.20 Chức năng chốt sổ

4.20.1 Lớp GUI

4.20.1.1 Hàm **btn\_Aggreate\_Click – Form : Aggreate**

Hàm **btn\_Aggreate\_Click** bắt sự kiện người dùng nhấn vào nút “Chốt sổ”. Sau đó hàm sẽ mở Form **Aggregate.**

4.20.1.2 Hàm **Aggregate\_Load – Form : Aggreate**

Hàm **Aggregate\_Load** sẽ bắt sự kiện Form đang chạy và hiển thị tên, doanh thu của nhân viên đang đang nhập vào tài khoản bằng hàm **get\_Total\_Price** trong lớp StaffBUS với tham số đầu vào là id của nhân viên.

Code :

        private void Aggregate\_Load(object sender, EventArgs e)

        {

            lbl\_Name.Text = session.name;

            lbl\_Revenu.Text = BUS.BookingBUS.Instance.get\_Total\_Price(session.id).ToString();

        }

4.20.1.3 Hàm **btn\_confirm\_Click**

Hàm **btn\_confirm\_Clic** bắt sự kiện nhân viên bấm vào nút “Chốt sổ” . Hàm sẽ lưu thông tin vào cơ sở dữ liệu thông qua hàm **create** trong RevenueBUS và Lưu thông tin vào file Excel thông qua hàm **addRevenueToExcel** trong lớp ExcelHelper . Nếu thực hiện lưu thành công thì sẽ trả về thành công và đóng form , nếu không thành công thì sẽ thông báo không thành công.

Code :

        private void btn\_Confirm\_Click(object sender, EventArgs e)

        {

            if (BUS.RevenueBUS.Instance.create(session.id, lbl\_Revenu.Text, DateTime.Now, "Chờ xử lý", txt\_Note.Text)

               && BUS.ExcelHelper.Instance.addRevenueToExcel(session.name, lbl\_Revenu.Text, null, DateTime.Now, "Chờ xử lý"))

            {

                MessageBox.Show("Lưu thông tin thành công");

                this.Close();

            }

            else

            {

                MessageBox.Show("Lưu thông tin không thành công");

            }

        }

4.20.2 Lớp BUS

4.20.2.1 Hàm **create - RevenueBUS**

Hàm tạo một đối tượng **Revenue** mới với các thông tin truyền vào từ các tham số *staff\_id, total\_price, created\_at, status, note*. Sau đó, gọi phương thức **add** của lớp **RevenueDAO** để thêm đối tượng **Revenue** vào cơ sở dữ liệu. Kết quả trả về từ phương thức **add** là số lượng bản ghi bị ảnh hưởng trong quá trình thêm. Hàm trả về **true** nếu số lượng bản ghi bị ảnh hưởng là 1, ngược lại trả về **false**.

Code :

        public bool create(int staff\_id, string total\_price, DateTime created\_at, string status, string

note)

        {

            return DAO.RevenueDAO.Instance.add(new Revenue(staff\_id,session.name, float.Parse(total\_price), 0, created\_at, status, note)) == 1 ? true : false;

        }

4.20.3 Lớp DTO

Đối tượng **Revenue** được sử dụng để lưu trữ thông tin của một nguyên vật liệu, ở đây sử dụng đối tượng **Revenue** với 6 đối số là *id,* *staff\_id, total\_price, created\_at, status, note.*

4.20.4 Lớp DAO

4.20.4.1 Hàm **create - RevenueDAO**

Trong hàm, câu lệnh SQL được sử dụng để thêm một bản ghi vào bảng STATITICS trong cơ sở dữ liệu. Các tham số *staff\_id, total\_price, created\_at, status, note* được truyền vào câu lệnh SQL để truyền giá trị tương ứng.

Hàm sử dụng lớp **DataProvider** để gọi phương thức **ExecuteNonQuery** và thực hiện câu lệnh SQL. Kết quả trả về từ phương thức **ExecuteNonQuery** là số lượng bản ghi bị ảnh hưởng trong quá trình thêm. Hàm trả về giá trị này để biểu thị số lượng bản ghi đã được thêm vào.

Code :

 public int add(Revenue revenue)

        {

            string sql = "INSERT INTO [dbo].[STATITICS] ([STAFF\_ID] ,[TOTAL\_PRICE] ,[TOTAL\_CASH] ,[CREATED\_AT] ,[STATUS] ,[NOTE])" +

                " VALUES( @Staff\_id , @TotalPrice , @TotalCash , @Created\_at , @Status , @Note )";

            return DataProvider.Instance.ExecuteNonQuery(sql, new Object[] { revenue.Staff\_id, revenue.TotalPrice, revenue.Cash, revenue.CreatedAt, revenue.Status, revenue.Note});

        }

4.21 Chức năng Quản lí mã giảm giá – Hiển thị tất cả mã giảm giá

4.20.1 Lớp GUI

4.20.1.1 Hàm **Promotion\_List\_Load – Form : Promotion\_List**

Hàm **Promotion\_List\_Loa** được bắt sự kiện "Load" của Form. Hàm **Promotion\_List\_Load** gọi hàm **view** từ PromotionBUSđể lấy danh sách mã giảm giá và sau đó gọi hàm **GV\_DataLoad** để hiển thị danh sách mã giảm giá lên DataGridView "GV\_Data".

4.20.2 Lớp BUS

4.20.2.1 Hàm **getAllPromotions - PromotionBUS**

Hàm **getAllPromotions** không nhận tham số đầu vào. Hàm sử dụng lớp PromotionDAO để gọi phương thức **GetAllPromotions** và trả về danh sách các đối tượng **Promotion**.

Code :

        public List<Promotion> getAllPromotions()

        {

            return DAO.PromotionDAO.Instance.GetAllPromotions();

        }

4.20.3 Lớp DTO

Đối tượng **Promotion** được sử dụng để lưu trữ thông tin của một nguyên vật liệu, ở đây sử dụng đối tượng **Promotion** với 6 đối số là *id,* *code, discount, status, Note.*

4.20.4 Lớp DAO

4.20.4.1 Hàm **getAllPromotions – PromotionDAO**

Hàm **GetAllPromotions** thực hiện truy vấn đến cơ sở dữ liệu để lấy tất cả các bản ghi từ bảng " Promotion " và trả về một List< Promotion >. Các thông tin của mỗi bản ghi được trích xuất từ DataRow và tạo thành đối tượng Promotion tương ứng, sau đó được thêm vào danh sách trước khi trả về. Các trường thông tin của đối tượng Promotion bao gồm *id,* *code, discount, status, Note.*

Code :

        public List<DTO.Promotion> GetAllPromotions()

        {

            List<DTO.Promotion> promotions = new List<DTO.Promotion>();

            string query = "SELECT \* FROM PROMOTION";

            DataTable dataTable = DataProvider.Instance.ExecuteQuery(query);

            foreach (DataRow row in dataTable.Rows)

            {

                int id = Convert.ToInt32(row["ID"]);

                string code = Convert.ToString(row["CODE"]);

                float discount = Convert.ToSingle(row["DISCOUNT"]);

                string status = Convert.ToString(row["STATUS"]);

                if (status == "1")

                {

                    status = "Đang hoạt động";

                }

                else

                {

                    status = "Đã kết thúc";

                }

                string note = Convert.ToString(row["NOTE"]);

                DTO.Promotion promotion = new DTO.Promotion()

                {

                    ID = id,

                    Code = code,

                    Discount = discount,

                    Status = status,

                    Note = note

                };

                promotions.Add(promotion);

            }

            return promotions;

        }

4.22 Chức năng Quản lí mã giảm giá – Thêm mã giảm giá

4.22.1 Lớp GUI

4.22.1.1 Hàm **btn\_Add\_Click**

Hàm **btn\_Add\_Click** bắt sự kiện người dùng nhấn vào nút “Thêm” và mở ra Form Thêm mã giảm giá

Chức năng : mở Form thêm mã giảm giá.

4.22.1.2 Hàm **btn\_Create\_Click**

Hàm **btn\_Create\_Click** bắt sự kiện người dùng bấm vào nút “Thêm”.Hàm sẽ kiểm tra và xử lý dữ liệu nhập từ các trường, sau đó gọi hàm **addPromotion** từ PromotionBUS để tạo mã giảm giá mới và hiển thị thông báo tương ứng.

Chức năng : Kiểm tra thông tin nhập vào , tạo vào thêm mới đối tượng Promotion.

4.22.2 Lớp BUS

4.22.2.1 Hàm **addPromotion – PromotionBUS**

Hàm **addPromotion** nhận vào một đối tượng **Promotion** là thông tin về khuyến mãi cần thêm. Hàm sử dụng lớp **PromotionDAO** để gọi phương thức **AddPromotion** và thực hiện thêm khuyến mãi vào cơ sở dữ liệu. Nếu việc thêm thành công (trả về giá trị lớn hơn 0), hiển thị thông báo "Thêm thành công!". Ngược lại, hiển thị thông báo "Có lỗi xảy ra!"

Chức năng : thêm một khuyến mãi vào cơ sở dữ liệu thông qua lớp PromotionDAO và hiển thị thông báo kết quả cho người dùng.

Code :

        public void addPromotion(Promotion promotion)

        {

            if(DAO.PromotionDAO.Instance.AddPromotion(promotion) > 0)

            {

                MessageBox.Show("Thêm thành công!");

            }

            else

            {

                MessageBox.Show("Có lỗi xảy ra!");

            };

        }

4.22.3 Lớp DTO

Đối tượng **Promotion** được sử dụng để lưu trữ thông tin của một nguyên vật liệu, ở đây sử dụng đối tượng **Promotion** với 6 đối số là *id,* *code, discount, status, Note.*

4.22.4 Lớp DAO

4.22.4.1 Hàm **AddPromotion – PromotionDAO**

Hàm **AddPromotion** được sử dụng để thêm một chương trình khuyến mãi mới vào cơ sở dữ liệu. Nó thực hiện câu truy vấn SQL để chèn dữ liệu vào bảng PROMOTION, sử dụng các thông tin từ đối tượng **Promotion** được truyền vào. Kết quả trả về là số lượng bản ghi đã được thêm.

Code :

        public int AddPromotion(DTO.Promotion promotion)

        {

            string query = "INSERT INTO PROMOTION (CODE, DISCOUNT, STATUS, NOTE) VALUES ( @Code , @Discount , @Status , @Note )";

             return DataProvider.Instance.ExecuteNonQuery(query, new object[] { promotion.Code, promotion.Discount, promotion.Status, promotion.Note });

        }

4.23 Chức năng Quản lí mã giảm giá –Sửa mã giảm giá

4.23.1 Lớp GUI

4.23.1.1 Hàm **btn\_Edit\_Click() – Form : Promotion\_List**

Hàm **btn\_Edit\_Click** được kích hoạt khi người dùng nhấn nút "Sửa" trên giao diện. Nó thực hiện việc lấy thông tin của chương trình khuyến mãi từ hàng được chọn trong bảng dữ liệu và hiển thị trong một form chỉnh sửa. Form này cho phép người dùng cập nhật các thông tin như mã chương trình, giảm giá, trạng thái và ghi chú. Sau khi người dùng hoàn tất chỉnh sửa, danh sách **Promotion** trên giao diện được cập nhật lại.

4.23.1.2 Hàm **btn\_Edit\_Click() – Form : Promotion\_Edit**

Hàm trên được sử dụng để kiểm tra và cập nhật thông tin chương trình khuyến mãi. Trước khi thực hiện cập nhật, hàm kiểm tra xem các trường nhập liệu như mã khuyến mãi và giá trị giảm giá đã được nhập hay chưa. Nếu có lỗi, hàm hiển thị thông báo lỗi và tập trung vào trường tương ứng. Nếu không có lỗi, hàm tạo một đối tượng **Promotion** mới với các giá trị đã nhập và gọi phương thức **UpdatePromotion** từ lớp **PromotionBUS** để thực hiện cập nhật thông tin. Sau đó, hàm đóng cửa sổ hiện tại.

Chức năng : kiểm tra và cập nhật thông tin của một đối tượng khuyến mãi.

4.23.2 Lớp BUS

4.23.2.1 Hàm **UpdatePromotion – PromotionBUS**

Hàm **UpdatePromotion** nhận vào một đối tượng **Promotion** và thực hiện cập nhật thông tin của khuyến mãi đó trong cơ sở dữ liệu. Nếu quá trình cập nhật thành công, hiển thị thông báo "Sửa thành công!". Ngược lại, hiển thị thông báo "Có lỗi xảy ra!".

Code :

        public void UpdatePromotion(Promotion promotion)

        {

            if (DAO.PromotionDAO.Instance.UpdatePromotion(promotion) > 0)

            {

                MessageBox.Show("Sửa thành công!");

            }

            else

            {

                MessageBox.Show("Có lỗi xảy ra!");

            };

        }

4.23.3 Lớp DTO

Đối tượng **Promotion** được sử dụng để lưu trữ thông tin của một nguyên vật liệu, ở đây sử dụng đối tượng **Promotion** với 6 đối số là *id,* *code, discount, status, Note.*

4.23.4 Lớp DAO

4.23.4.1 Hàm **UpdatePromotion – PromotionDAO**

Hàm UpdatePromotion nhận vào một đối tượng khuyến mãi và thực hiện cập nhật thông tin của khuyến mãi đó trong cơ sở dữ liệu. Câu truy vấn SQL được truyền vào trong đó là *id,* *code, discount, status, Note* các tham số được truyền vào. Hàm trả về số lượng bản ghi được cập nhật trong cơ sở dữ liệu.

Code :

        public int UpdatePromotion(DTO.Promotion promotion)

        {

            string query = "UPDATE PROMOTION SET CODE = @Code , DISCOUNT = @Discount , STATUS = @Status , NOTE = @Note WHERE ID = @ID ";

            return DataProvider.Instance.ExecuteNonQuery(query, new object[] { promotion.Code, promotion.Discount, promotion.Status, promotion.Note, promotion.ID });

        }

4.24 Chức năng Quản lí mã giảm giá –Xóa mã giảm giá

4.24.1 Lớp GUI

4.24.1.1 Hàm **btn\_Delete\_Click – Promotion\_List**

**btn\_Delete\_Click** là một sự kiện được kích hoạt khi người dùng nhấn vào nút "Xóa" trên giao diện.Hàm sẽ kiểm tra xem có hàng khuyến mãi được chọn hay không. Nếu có, lấy chỉ mục của hàng đó và lấy giá trị của ô đầu tiên trong hàng được chọn. Tiếp theo, hiển thị một hộp thoại xác nhận xóa khuyến mãi và nếu người dùng chọn OK, hàm gọi phương thức **DeletePromotion** từ lớp **PromotionBUS** để xóa khuyến mãi có ID tương ứng.

4.24.2 Lớp BUS

4.24.2.1 Hàm **DeletePromotion – PromotionBUS**

Hàm **DeletePromotion** thực hiện việc xóa một khuyến mãi dựa trên ID được cung cấp. Cụ thể, hàm này kiểm tra tính hợp lệ của ID, sau đó gọi phương thức **DeletePromotion** từ lớp **PromotionDAO** để xóa khuyến mãi tương ứng. Nếu việc xóa thành công (số hàng bị ảnh hưởng lớn hơn 0), hiển thị thông báo "Xóa thành công!". Trong trường hợp không có hàng nào bị ảnh hưởng, hiển thị thông báo "Có lỗi xảy ra!". Điều này giúp người dùng biết kết quả của việc xóa khuyến mãi và có thể tiếp tục các thao tác khác nếu cần.

Code :

        public void DeletePromotion(int id)

        {

            if (id <= 0)

            {

                throw new ArgumentException("Invalid promotion ID. Please provide a valid ID.");

            }else if (DAO.PromotionDAO.Instance.DeletePromotion(id) >0)

            {

                MessageBox.Show("Xóa thành công!");

            }

            else

            {

                MessageBox.Show("Có lỗi xảy ra!");

            };

        }

4.24.3 Lớp DTO

Đối tượng **Promotion** được sử dụng để lưu trữ thông tin của một nguyên vật liệu, ở đây sử dụng đối tượng **Promotion** với 6 đối số là *id,* *code, discount, status, Note.*

4.24.4 Lớp DAO

4.24.4.1 Hàm **DeletePromotion - PromotionDAO**

Hàm **DeletePromotion** được sử dụng để xóa một khuyến mãi dựa trên ID được cung cấp. Hàm này thực hiện câu truy vấn SQL để xóa khuyến mãi từ bảng PROMOTION với điều kiện ID tương ứng. Sau đó, hàm trả về kết quả của phương thức **ExecuteNonQuery** từ lớp **DataProvider**, cho biết số bản ghi bị ảnh hưởng bởi câu truy vấn xóa.

Code :

        public int DeletePromotion(int id)

        {

            string query = "DELETE FROM PROMOTION WHERE ID = @ID ";

            return DataProvider.Instance.ExecuteNonQuery(query, new object[] { id });

        }

4.25 Chức năng Quản lí lịch chiếu phim – Hiển thị lịch chiếu

4.25.1 Lớp GUI

4.25.1.1 Hàm **Schedule\_List\_Load**

Hàm Schedule\_List\_Load được sử dụng để tải và hiển thị danh sách lịch chiếu cho một rạp chiếu phim cụ thể trên giao diện người dùng. Đầu tiên, hàm gọi phương thức **loadShowroom** từ lớp **ScheduleBUS** để tải danh sách các rạp chiếu phim và hiển thị chúng trong combobox cbb\_theater, cho phép người dùng lựa chọn rạp chiếu phim để xem lịch chiếu. Tiếp theo, hàm gọi phương thức **getScheduleOfShowroom** từ lớp **ScheduleDAO** để lấy danh sách các lịch chiếu cho rạp chiếu phim có mã "1" hoặc một giá trị cụ thể khác (tùy thuộc vào nguồn dữ liệu). Kết quả trả về là một danh sách các mảng chuỗi chứa thông tin về lịch chiếu. Cuối cùng, hàm gọi phương thức **loadDGV** để hiển thị danh sách lịch chiếu trong DataGridView hoặc thành phần tương tự trên giao diện người dùng.

Code :

        private void Schedule\_List\_Load(object sender, EventArgs e)

        {

            List<string[]> list = ScheduleDAO.Instance.getScheduleOfShowroom(cbb\_theater.SelectedValue.ToString());

            loadDGV(list);

            if (session.lang == null)

            {

                session.lang = "vi-VN";

            }

            SetLanguage(session.lang);

        }

4.25.1.2 Hàm **loadDGV**

Hàm **loadDGV** được sử dụng để tải danh sách lịch chiếu vào một DataGridView. Trước khi tải danh sách mới, hàm sẽ xóa sạch các dòng hiện tại trong DataGridView để chuẩn bị cho việc hiển thị danh sách mới. Sau đó, hàm duyệt qua từng phần tử trong danh sách và tạo một dòng mới trong DataGridView, sau đó gán giá trị từ danh sách vào các ô dữ liệu của dòng đó. Cuối cùng, hàm thêm dòng vào DataGridView để hiển thị thông tin lịch chiếu. Tóm lại, chức năng của hàm loadDGV là hiển thị danh sách lịch chiếu trong DataGridView để người dùng có thể xem thông tin chi tiết về các lịch chiếu.

4.25.2 Lớp BUS

4.25.3 Lớp DTO

Đối tượng **Schedule**được sử dụng để lưu trữ thông tin của một nguyên vật liệu, ở đây sử dụng đối tượng **Schedule** với 6 đối số là *id,* *showroomID, movieID, startTime, endTime, price.*

4.25.4 Lớp DAO

4.25.2.1 Hàm **getScheduleOfShowroom - ScheduleDAO**

Hàm **getScheduleOfShowroom** được sử dụng để lấy danh sách các lịch chiếu cho một rạp chiếu phim cụ thể. Khi được gọi, hàm này sẽ thực hiện truy vấn vào cơ sở dữ liệu để lấy thông tin lịch chiếu từ bảng tương ứng. Sau khi truy vấn thành công, hàm sẽ duyệt qua từng dòng dữ liệu trong kết quả trả về và trích xuất các thông tin cần thiết như ID, số rạp chiếu, tiêu đề phim, thời gian bắt đầu và kết thúc, số ghế trống và giá vé. Dựa trên số ghế trống và số ghế tổng cố định, hàm tính toán và tạo chuỗi đại diện cho số lượng ghế đã được đặt và tổng số ghế. Sau đó, thông tin lịch chiếu của mỗi dòng sẽ được lưu trữ trong một mảng chuỗi obj, và mảng này sẽ được thêm vào danh sách strings. Cuối cùng, hàm trả về danh sách **strings** chứa thông tin lịch chiếu cho rạp chiếu phim đã được chỉ định.

Code :

        public List<string[]> getScheduleOfShowroom(string showroomnumber)

        {

            List<string[]> strings = new List<string[]>();

            string sql = "exec get\_schedule\_by\_showroom @showroomnumber";

            DataTable dataTable = DataProvider.Instance.ExecuteQuery(sql, new Object[] { showroomnumber });

            foreach (DataRow row in dataTable.Rows)

            {

                string id = row["ID"].ToString();

                string showroom = row["showroomnumber"].ToString();

                string movieTitle = row["title"].ToString();

                string starttime = row["starttime"].ToString();

                string endtime = row["endtime"].ToString();

                int emptySeat = int.Parse(row["emptyseat"].ToString());

                int seatCount = 90;

                string price = row["price"].ToString();

                string[] obj = { id, showroom, movieTitle, starttime, endtime, string.Format("{0:d}/{1:d}", seatCount - emptySeat, seatCount), price };

                strings.Add(obj);

            }

            return strings;

        }

4.26 Chức năng Quản lí lịch chiếu phim – Thêm lịch chiếu phim

4.26.1 Lớp GUI

4.26.1.1 Hàm **btn\_Add\_Click**

Hàm **btn\_Add\_Click** được gọi khi người dùng nhấn vào nút "Thêm" trên giao diện người dùng. Chức năng của hàm là hiển thị cửa sổ Schedule\_Add để người dùng có thể thêm thông tin lịch chiếu mới. Sau khi cửa sổ Schedule\_Add được đóng lại, hàm gọi lại phương thức **Schedule\_List\_Load** để cập nhật danh sách lịch chiếu trên giao diện người dùng.

4.26.1.2 Hàm **Schedule\_Add\_Load**

Trong hàm Schedule\_Add\_Load, định dạng hiển thị cho DateTimePicker dtp\_StartTime và dtp\_EndTime được thiết lập là "yyyy/MM/dd hh:mm". Điều này cho phép người dùng chọn ngày và giờ trong đúng định dạng này. Sau đó, hàm gọi phương thức **loadMovie** từ lớp **MovieBUS** để tải danh sách phim vào combobox cbb\_MovieID. Điều này cho phép người dùng lựa chọn phim cho lịch chiếu. Tiếp theo, hàm gọi phương thức **loadShowroom** từ lớp **ScheduleBUS** để tải danh sách các rạp chiếu phim vào combobox cbb\_ShowroomNumber. Điều này cho phép người dùng lựa chọn rạp chiếu phim cho lịch chiếu.

4.26.1.3 Hàm **btn\_Create\_Click**

Hàm **btn\_Create\_Click** được sử dụng để xử lý sự kiện tạo lịch chiếu mới khi người dùng nhấn nút "Tạo". Chức năng của hàm là kiểm tra các điều kiện và thực hiện tạo lịch chiếu mới. Sau khi kiểm tra và thực hiện, cửa sổ hiện tại sẽ được đóng.

Nếu người dùng chưa chọn số hiệu rạp chiếu trong combobox cbb\_ShowroomNumber, hàm sẽ hiển thị thông báo lỗi và yêu cầu người dùng chọn số hiệu rạp bằng cách điều chỉnh màu sắc và trỏ chuột vào combobox. Các nhãn liên quan cũng sẽ được điều chỉnh màu sắc để chỉ ra lỗi ở combobox cbb\_ShowroomNumber.

Sau đó, hàm gọi phương thức **create** từ lớp **ScheduleBUS** để tạo một lịch chiếu mới. Các tham số truyền vào bao gồm *id,* *showroomID, movieID, startTime, endTime, price* được lấy từ các thành phần giao diện tương ứng.

4.26.2 Lớp BUS

4.26.2.1 Hàm **create - ScheduleBUS**

Phương thức **create** được định nghĩa trong lớp **ScheduleBUS** và được sử dụng để tạo một lịch chiếu mới trong cơ sở dữ liệu. Phương thức này nhận các tham số như số hiệu rạp chiếu *id,* *showroomID, movieID, startTime, endTime, price.*

Trong phương thức **create**, nó gọi phương thức create từ lớp **ScheduleDAO** để thực hiện thêm lịch chiếu mới vào cơ sở dữ liệu. Kết quả trả về từ phương thức **create** được gán cho biến i. Nếu giá trị của i lớn hơn 0, tức là thêm lịch chiếu thành công, hàm sẽ hiển thị một thông báo thành công bằng hộp thoại MessageBox.

Code :

public void create(int showroomID, int movieID, DateTime startTime, DateTime endTime, float

price)

        {

            int i = DAO.ScheduleDAO.Instance.create(showroomID, movieID, startTime, endTime, price);

            if (i > 0)

            {

                MessageBox.Show("Thêm thành công");

            }

        }

4.26.2.2 Hàm **loadMovie – ScheduleBUS**

Trong phương thức **loadMovie**, các thuộc tính DisplayMember và ValueMember của combobox cbb được thiết lập để xác định cách hiển thị và giá trị của các mục trong combobox. Sau đó, dữ liệu từ lớp **MovieDAO** được truy xuất thông qua phương thức **loadMovie** và gán vào DataSource của combobox cbb. Điều này đảm bảo rằng danh sách phim được hiển thị trong combobox và các mục trong danh sách sẽ được hiển thị theo thuộc tính *Name* và giá trị của mục sẽ là thuộc tính *Id*.

Code :

        public void loadMovie(KryptonComboBox cbb)

        {

            cbb.DisplayMember = "Name";

            cbb.ValueMember = "Id";

            cbb.DataSource = DAO.MovieDAO.Instance.loadMovie();

        }

4.26.2.3 Hàm **loadShowRoom**

Trong phương thức **loadShowRoom**, các thuộc tính DisplayMember và ValueMember của combobox cbb được thiết lập để xác định cách hiển thị và giá trị của các mục trong combobox. Sau đó, dữ liệu từ lớp **ShowRoomDAO** được truy xuất thông qua phương thức **loadShowRoom** và gán vào DataSource của combobox cbb. Điều này đảm bảo rằng danh sách phim được hiển thị trong combobox và các mục trong danh sách sẽ được hiển thị theo thuộc tính Name và giá trị của mục sẽ là thuộc tính Id.

Code

        public void loadShowroom(KryptonComboBox cbb)

        {

            cbb.DisplayMember = "Name";

            cbb.ValueMember = "Id";

            cbb.DataSource = DAO.ShowRoomDAO.Instance.loadShowRoom();

        }

4.26.3 Lớp DTO

Đối tượng **Schedule** được sử dụng để lưu trữ thông tin của một nguyên vật liệu, ở đây sử dụng đối tượng **Schedule** với 6 đối số là *id,* *showroomID, movieID, startTime, endTime, price.*

4.26.4 Lớp DAO

4.26.4.1 Hàm **create – ScheduleDAO**

Hàm **create** truy cập cơ sở dữ liệu để gọi đến thủ tục **create\_schedule** với các tham số là *id,* *showroomID, movieID, startTime, endTime, price* thông qua hàm ExecuteNonQuery thông qua lớp DataProdiver.

Code :

        public int create(

         int theaterID, int movieID, DateTime startTime, DateTime endTime, float price)

        {

            string sql = "create\_schedule @theaterID , @movieID , @startTime , @endTime , @price ";

            return DataProvider.Instance.ExecuteNonQuery(sql, new Object[] { theaterID, movieID, startTime, endTime, price });

        }

4.26.4.2 Hàm **loadShowRoom – ScheduleDAO**

Trong phương thức này, đầu tiên ta khởi tạo một danh sách rỗng có kiểu Items là **theaters**. Tiếp theo, một đối tượng **Items** với id và name là "Default" được thêm vào danh sách để đại diện cho giá trị mặc định. Sau đó, một câu truy vấn SQL được thực thi để lấy dữ liệu từ bảng **showroom**, trả về các cột id và **showRoomNumber**. Dữ liệu trả về được lưu vào một đối tượng DataTable có tên dataTable. Tiếp theo, với mỗi dòng trong dataTable, ta trích xuất giá trị id và **showRoomNumber** từ cột tương ứng và tạo một đối tượng **Items** mới với id và name là "Rạp phim " + **showRoomNumber**. Đối tượng này được thêm vào danh sách **theaters**.

Code :

        public List<Items>  loadShowRoom()

        {

            List<Items> theaters = new List<Items>();

            theaters.Add(new Items(0, "Default"));

            string sql = "Select id,showRoomNumber from showroom order by showRoomNumber";

            DataTable dataTable = DataProvider.Instance.ExecuteQuery(sql);

            foreach (DataRow row in dataTable.Rows)

            {

                int id = int.Parse(row["ID"].ToString());

                string showrRooNumber = row["showRoomNumber"].ToString();

                theaters.Add(new Items(id, "Rạp phim " + showrRooNumber));

            }

            return theaters;

        }

4.26.4.3 Hàm **loadShowRoom – ScheduleDAO**

Trong phương thức này, đầu tiên ta khởi tạo một danh sách rỗng có kiểu **Items** là movies. Tiếp theo, một đối tượng **Items** với id và name là "Default" được thêm vào danh sách để đại diện cho giá trị mặc định. Sau đó, một câu truy vấn SQL được thực thi để lấy dữ liệu từ việc thực hiện thủ tục **getIshowingMovie**. Dữ liệu trả về được lưu vào một đối tượng DataTable có tên dataTable. Tiếp theo, với mỗi dòng trong dataTable, ta trích xuất giá trị id và title từ cột tương ứng và tạo một đối tượng **Items** mới với *id* và *title*. Đối tượng này được thêm vào danh sách movies.

Code :

        public List<Items> loadMovie()

        {

            List<Items> movies = new List<Items>();

            movies.Add(new Items(0, "Default"));

            string sql = "exec getIshowingMovie";

            DataTable dataTable = DataProvider.Instance.ExecuteQuery(sql);

            foreach (DataRow row in dataTable.Rows)

            {

                int id = int.Parse(row["ID"].ToString());

                string title = row["title"].ToString();

                movies.Add(new Items(id, title));

            }

            return movies;

        }

4.27 Chức năng Quản lí lịch chiếu phim –Sửa lịch chiếu

4.27.1 Lớp GUI

4.27.1.1 Hàm **btn\_Edit\_Click – Form : Schedule\_List**

Tạo một đối tượng mới của lớp **Schedule\_Edit** để hiển thị giao diện chỉnh sửa lịch chiếu. Điền thông tin của lịch chiếu cần chỉnh sửa vào các thành phần của giao diện chỉnh sửa tương ứng (txt\_ScheduleID, cbb\_ShowroomNumber, cbb\_MovieID, dtp\_StartTime, dtp\_EndTime, txt\_Price) bằng cách lấy giá trị từ các ô trên dòng được chọn trong bảng hiển thị lịch chiếu (**GV\_Data**). Và hiển thị giao diện chỉnh sửa lịch chiếu

4.27.1.2 Hàm **btn\_Edit\_Click – Form : Schedule\_Edit**

Hàm **btn\_Edit\_Click** được gọi khi người dùng nhấn vào nút "Sửa" trên giao diện người dùng. Hàm sẽ kiểm tra và hiển thị thông báo lỗi nếu người dùng chưa chọn số hiệu rạp chiếu hoặc phim, hoặc chưa nhập đơn giá. Kiểm tra và hiển thị thông báo lỗi nếu đơn giá nhập vào chứa ký tự không hợp lệ.

Nếu không có lỗi, cập nhật thông tin lịch chiếu bằng cách gọi phương thức **update** từ lớp **ScheduleBUS** với các thông tin tương ứng.

4.27.2 Lớp BUS

4.27.2.1 Hàm **update – ScheduleBUS**

Phương thức **update** được định nghĩa trong lớp **ScheduleBUS** và được sử dụng để cập nhật thông tin lịch chiếu trong cơ sở dữ liệu. Phương thức này nhận ba tham số: *startTime* ,*endTime* và *id* (ID của lịch chiếu cần được cập nhật).

Trong phương thức update, nó gọi phương thức update từ lớp **ScheduleDAO** để thực hiện cập nhật thông tin lịch chiếu trong cơ sở dữ liệu. Kết quả trả về từ phương thức update được gán cho biến i. Nếu giá trị của i lớn hơn 0, tức là cập nhật thành công, hàm sẽ hiển thị một thông báo thành công bằng hộp thoại MessageBox

Code :

        public void update(DateTime startTime, DateTime endTime, string id)

        {

            int i = DAO.ScheduleDAO.Instance.update(startTime, endTime, id);

            if (i > 0)

            {

                MessageBox.Show("Sửa thành công");

            }

        }

4.27.3 Lớp DTO

Đối tượng **Schedule** được sử dụng để lưu trữ thông tin của một lịch chiếu, ở đây sử dụng đối tượng **Schedule** với 6 đối số là *id,* *showroomID, movieID, startTime, endTime, price.*

4.27.4 Lớp DAO

4.27.4.1 Hàm **update – ScheduleDAO**

Trong phương thức update, câu lệnh SQL được thiết lập để cập nhật thông tin lịch chiếu trong bảng SCHEDULE. Thông qua câu lệnh UPDATE, cột starttime sẽ được cập nhật thành giá trị *startTime*, cột *endtime* sẽ được cập nhật thành giá trị endTime cho hàng có ID là *id*.

Code :

        public int update(DateTime startTime, DateTime endTime, string id)

        {

            string sql = "Update SCHEDULE set starttime = @starttime , endtime = @endtime where id = @id ";

            return DataProvider.Instance.ExecuteNonQuery(sql, new Object[] { startTime, endTime, id });

        }

4.28 Chức năng Quản lí lịch chiếu phim –Xóa lịch chiếu

4.28.1 Lớp GUI

4.28.1.1 Hàm **btn\_Delete\_Click**

Hàm **btn\_Delete\_Click** được gọi khi người dùng nhấn vào nút "Xóa" trên giao diện và kiểm tra xem người dùng đã chọn một ô trong DataGridView hay chưa. Nếu có, lấy thông tin hàng được chọn và giá trị của ô đầu tiên (cột ID lịch chiếu). Nếu người dùng nhấn OK, gọi phương thức **delete** từ lớp **ScheduleBUS** để xóa lịch chiếu dựa trên *ID* lịch chiếu được chọn. Nếu xảy ra lỗi, hiển thị một thông báo lỗi tương ứng.

4.28.2 Lớp BUS

4.28.2.1 Hàm **delete – ScheduleBUS**

Gọi phương thức **delete** từ lớp **ScheduleDAO** để xóa lịch chiếu dựa trên *ID* lịch chiếu được truyền vào. Kết quả trả về từ phương thức delete được gán cho biến i. Nếu giá trị của i lớn hơn 0, tức là xóa lịch chiếu thành công, hàm sẽ hiển thị một thông báo thành công bằng hộp thoại MessageBox.

Code :

        public void delete(string id)

        {

            int i = DAO.ScheduleDAO.Instance.delete(id);

            if (i > 0)

            {

                MessageBox.Show("Xóa thành công");

            }

        }

4.28.3 Lớp DTO

Đối tượng **Schedule** được sử dụng để lưu trữ thông tin của một lịch chiếu, ở đây sử dụng đối tượng **Schedule** với 6 đối số là *id,* *showroomID, movieID, startTime, endTime, price.*

4.28.4 Lớp DAO

4.28.4.1 Hàm **delete – ScheduleDAO**

Trong phương thức delete, chuỗi SQL được tạo để xóa dòng dữ liệu từ bảng **Schedule** trong cơ sở dữ liệu với điều kiện ID lịch chiếu bằng giá trị được truyền vào. Sau đó, phương thức **ExecuteNonQuery** của lớp **DataProvider** được gọi để thực thi câu lệnh SQL và trả về số lượng hàng bị ảnh hưởng .

Code :

        public int delete(string id)

        {

            string sql = "DELETE FROM Schedule WHERE id = @id ";

            return DataProvider.Instance.ExecuteNonQuery(sql, new Object[] { id });

        }

4.29 Chức năng Quản lí phòng chiếu–Hiển thị phòng chiếu

4.29.1 Lớp GUI

4.29.1.1 Hàm **Theater\_List\_Load**

Hàm **Theater\_List\_Load** bắt sự kiện Form đang mở và hiện tất cả các phòng chiếu bằng hàm **GV\_Load.**

4.29.1.2 Hàm **GV\_DataLoad**

Hàm **GV\_DataLoad** lấy tất cả phòng chiếu từ hàm **selectAll** của lớp **ShowRoomDAO** để hiển thị lên DataGridView.

4.29.2 Lớp BUS

4.29.3 Lớp DTO

Đối tượng **ShowRoom** được sử dụng để lưu trữ thông tin của phòng chiếu, ở đây sử dụng đối tượng **ShowRoom** với 2 đối số là *id,* *showRoomNumvber.*

4.29.4 Lớp DAO

4.29.4.1 Hàm **selectAll – ShowroomDAO**

Hàm gọi câu lệnh SQL dùng hàm **ExecuteQuery** của lớp DataProvider để xử lí và lấy dữ liệu.

        public List<ShowRoom> selectAll()

        {

            List<ShowRoom> theaters = new List<ShowRoom>();

            string sql = "Select \* from ShowRoom";

            DataTable dataTable = DataProvider.Instance.ExecuteQuery(sql);

            foreach (DataRow row in dataTable.Rows)

            {

                string id = row["ID"].ToString();

                string showrRooNumber = row["showroomnumber"].ToString();

                ShowRoom theater = new ShowRoom(id, showrRooNumber);

                theaters.Add(theater);

            }

            return theaters;

        }

4.30 Chức năng Quản lí phòng chiếu–Tạo phòng chiếu

4.30.1 Lớp GUI

4.30.1.1 Hàm **btn\_Add\_Click**

Hàm **btn\_Add\_Click** bắt sự kiện nhấn vào nút thêm phòng chiếu. Tiếp đó thì Form Showroom\_Add được mở ra dể

4.30.1.1 Hàm **btn\_Create\_Click**

Hàm **btn\_Create\_Click** bắt sự kiến nhấn nút thêm để xác nhận và thêm phòng chiếu. Hàm kiểm tra thông tin, nếu hợp lệ thì sẽ đưa thông tin vào hàm **create** ở lớp ShowroomBUS để thêm phòng chiếu.

4.30.2 Lớp BUS

4.30.2.1 Hàm **create – ShowRoomBUS**

Hàm **create** nhận tham số là *theaterNumber* để tạo thêm phòng mới sau đó truyền tham số vào hàm **create** của lớp ShowroomDAO. Nếu kết quả trả về lớn hơn 0 thì thông tin báo thêm thông báo thành công.

Code :

        public void create(string theaterNum)

        {

            int i = DAO.ShowRoomDAO.Instance.create(theaterNum);

            if (i > 0)

            {

                MessageBox.Show("Thêm thành công");

            }

        }

4.30.3 Lớp DTO

Đối tượng **ShowRoom** được sử dụng để lưu trữ thông tin của phòng chiếu, ở đây sử dụng đối tượng **ShowRoom** với 6 đối số là *id,* *showRoomNumvber.*

4.30.4 Lớp DAO

4.30.4.1 Hàm **create – ShowRoomDAO**

Hàm gọi thủ tục **create\_showroom** với tham số *showroomNumber* dùng hàm **ExecuteQuery** của lớp DataProvider để xử lí và tạo một phòng chiếu mới vào bảng Showroom trên cơ sở dữ liệu.

Chức năng : Thêm mới phòng chiếu vào cơ sở dữ liệu

Code :

        public int create(string showrRooNumber)

        {

            string sql = "exec create\_showroom @showRoomNumber";

            return DataProvider.Instance.ExecuteNonQuery(sql, new Object[] { showrRooNumber });

        }

4.31 Chức năng Quản lí phòng chiếu –Sửa thông tin phòng chiếu

4.31.1 Lớp GUI

4.31.1.1 Hàm **btn\_Edit\_Click – Form : Showroom\_List**

Hàm **btn\_Edit\_Click** bắt sự kiện nhấn vào nút Sửa trên Form **Showroom\_List** . Sau đó hàm sẽ mở Form Showroom\_Edit và đặt lại thông tin của phòng chiếu được chọn.

4.31.1.2 Hàm **btn\_Edit\_Click – Form : Showroom\_Edit**

Hàm **btn\_Edit\_Click** bắt sự kiến nhấn nút thêm để xác nhận và thêm phòng chiếu. Hàm kiểm tra thông tin, nếu hợp lệ thì sẽ đưa thông tin bao gồm **showroomNumber, ShowroomID** vào hàm **update** ở lớp ShowroomBUS để thêm phòng chiếu.

4.31.2 Lớp BUS

4.31.2.1 Hàm **update – ShowRoomBUS**

Hàm **update** nhận tham số là *theaterNumber, theaterID* để tạo thêm phòng mới sau đó truyền tham số vào hàm **update** của lớp ShowroomDAO. Nếu kết quả trả về lớn hơn 0 thì thông tin báo thêm thông báo thành công.

Code :

        public void update(string theaterNum,

         string id)

        {

            int i = DAO.ShowRoomDAO.Instance.update(theaterNum, id);

            if (i > 0)

            {

                MessageBox.Show("Cập nhật thành công");

            }

        }

4.31.3 Lớp DTO

Đối tượng **ShowRoom** được sử dụng để lưu trữ thông tin của phòng chiếu, ở đây sử dụng đối tượng **ShowRoom** với 2 đối số là *id,* *showRoomNumvber.*

4.31.4 Lớp DAO

4.30.4.1 Hàm **update – ShowRoomDAO**

Hàm gọi câu lệnh update với tham số *showroomNumber, showroomID* dùng hàm **ExecuteNonQuery** của lớp DataProvider để xử lí và cập nhật thông tin một phòng chiếu ở bảng Showroom trên cơ sở dữ liệu.

Code :

        public int update(string showrRooNumber,

         string id)

        {

            string sql = "UPDATE [dbo].[showroom] SET [showRoomNumber] = @showRoomNumber WHERE ID = @ID";

            return DataProvider.Instance.ExecuteNonQuery(sql, new Object[] { showrRooNumber, id });

        }

4.32 Chức năng Quản lí phòng chiếu–Xóa phòng chiếu

4.32.1 Lớp GUI

4.32.1.1 Hàm **btn\_Delete\_Click – Showroom\_List**

Hàm bắt sự kiện người dùng nhấn vào nút xóa và khi đó hàm sẽ lấy *id*  của phòng chiếu đó và xác nhận lại xem người dùng muốn xóa phòng đó hay không , nếu có thì sẽ gọi đến hàm **delete** ở lớp ShowRoomBUS với tham số là *id .*

4.32.1 Lớp BUS

4.32.1.1 Hàm **delete - ShowroomBUS**

Hàm **delete** nhận tham số là *id* để xóa phòng chiếu sau đó truyền tham số vào hàm **delete** của lớp ShowroomDAO. Nếu kết quả trả về lớn hơn 0 thì thông tin báo thêm thông báo thành công.

Code :

        public void delete(string id)

        {

            int i = DAO.ShowRoomDAO.Instance.delete(id);

            if (i > 0)

            {

                MessageBox.Show("Xóa thành công");

            }

        }

4.32.1 Lớp DTO

Đối tượng **ShowRoom** được sử dụng để lưu trữ thông tin của phòng chiếu, ở đây sử dụng đối tượng **ShowRoom** với 2 đối số là *id,* *showRoomNumvber.*

4.32.1 Lớp DAO

4.30.4.1 Hàm **delete – ShowroomDAO**

Hàm gọi câu lệnh **delete** với tham số *id* dùng hàm **ExecuteNonQuery** của lớp DataProvider để xử lí và xóa một phòng chiếu ở bảng Showroom trên cơ sở dữ liệu.

Code :

        public int delete(string id)

        {

            string sql = "DELETE FROM showroom WHERE id = @id ";

            return DataProvider.Instance.ExecuteNonQuery(sql, new Object[] { id });

        }

4.33 Chức năng Quản lí Phim – Tìm kiếm Phim

4.33.1 Lớp GUI

4.33.1.1 Hàm **txt\_Search\_TextChanged – Showroom\_List**

Hàm **txt\_Search\_TextChanged** bắt sự kiện thay đổi trên thanh tìm kiếm và lấy giá trị được nhập vào để đưa vào hàm **search** của MovieBUS.Sau đó lấy kết quả và đưa vào hàm **GV\_LoadData** để hiển thị trên màn hình.

4.33.2 Lớp BUS

4.8.2.1 Hàm **search – MovieBUS**

Hàm **search** có tham số đầu vào là *string key* . Hàm sẽ trả về List<Showroom > lấy từ hàm **search** của ShowroomDAO.

Code :

        public List<Movie> search( string key)

        {

            return  DAO.MovieDAO.Instance.search(key);

        }

4.33.3 Lớp DTO

Đối tượng **Movie** được sử dụng để lưu trữ thông tin của một nguyên vật liệu, ở đây sử dụng đối tượng Movie với 5 đối số là *id, title, genre, duration, rating, story, poster*

4.33.4 Lớp DAO

4.33.4.1 Hàm **search – MovieDAO**

Hàm **search** gọi thủ tục “search\_Movie” trên cơ sở dữ liệu với tham số là *string key* để tìm kiếm các phim liên quan đến *key* trên cơ sở dữ liệu. Lệnh SQL sẽ được truyền vào hàm **ExecuteQuery** của lớp DAO.DataProvider. Hàm sẽ thêm tất cả phim vào List<Movie > và trả về List< Movie > này.

Code :

        public List<Movie> search(string key)

        {

            List<Movie> movies = new List<Movie>();

            string sql = "exec search\_Movie @type";

            DataTable dataTable = DataProvider.Instance.ExecuteQuery(sql, new Object[] { key });

            foreach (DataRow row in dataTable.Rows)

            {

                string id = row["ID"].ToString();

                string title = row["title"].ToString();

                string genre = row["genre"].ToString();

                int duration = int.Parse(row["duration"].ToString());

                float rating = float.Parse(row["rating"].ToString());

                string story = row["story"].ToString();

                string poster = row["poster"].ToString();

                Movie movie = new Movie(id, title, genre, duration, rating, story, poster);

                movies.Add(movie);

            }

            return movies;

        }

4.34 Chức năng Quản lí doanh thu theo ngày – Xem doanh thu

4.34.1 Lớp GUI

4.34.1.1 Hàm **dtp\_Day\_ValueChanged**

Hàm lấy thông tin thời gian của datetimepicker để truyền vào hàm **GetRevenuesByDay** của lớp RevenueBUS.

4.34.2 Lớp BUS

4.34.2.1 Hàm **GetRevenuesByDay – RevenueBUS**

Hàm truyền đối tham số *Datetime day*  vào hàm GetRevenusByDay của lớp RevenueDAO để thấy dữ liệu

Code :

        public List<Revenue> GetRevenuesByDay(DateTime day)

        {

            return RevenueDAO.Instance.GetRevenuesByDay(day);

        }

4.34.3 Lớp DTO

Đối tượng **Revenue** được sử dụng để lưu trữ thông tin của phòng chiếu, ở đây sử dụng đối tượng **Revenue** với 8 đối số là *id, staff\_id, staffName, total\_price, cash , created\_at, status, note.*

4.34.4 Lớp DAO

4.34.4.1 Hàm **GetRevenuesByDay -**  **RevenueDAO**

Hàm truy xuất dữ liệu từ cơ sở dữ liệu để lấy thông tin doanh thu của ngày được truyền làm tham số là *Datatime day .* Hàm sẽ trả về một List<Revenue> lưu thông tin của doanh của ngày được truyền vào .

Code :

public List<Revenue> GetRevenuesByDay(DateTime day)

        {

            List<Revenue> revenues = new List<Revenue>();

            string sql = "SELECT STATITICS.\*, STAFF.FIRSTNAME, STAFF.LASTNAME " +

                         "FROM STATITICS " +

                         "INNER JOIN STAFF ON STATITICS.STAFF\_ID = STAFF.ID " +

                         "WHERE CAST(STATITICS.CREATED\_AT AS DATE) = CAST( @day AS DATE) " +

                         "ORDER BY STATITICS.CREATED\_AT";

            DataTable dataTable = DataProvider.Instance.ExecuteQuery(sql, new object[] { day });

            foreach (DataRow row in dataTable.Rows)

            {

                string id = row["ID"].ToString();

                int staffId = Convert.ToInt32(row["Staff\_ID"]);

                string fname = row["firstName"].ToString();

                string lname = row["lastName"].ToString();

                string staffName = fname + " " + lname;

                float totalPrice = Convert.ToSingle(row["Total\_Price"]);

                float cash = Convert.ToSingle(row["Total\_Cash"]);

                DateTime createdAt = Convert.ToDateTime(row["Created\_At"]);

                string status = row["Status"].ToString();

                string note = row["Note"].ToString();

                Revenue revenue = new Revenue(id, staffId, staffName, totalPrice, cash, createdAt, status, note);

                revenues.Add(revenue);

            }

            return revenues;

        }

4.35 Chức năng Quản lí doanh thu theo ngày – Cập nhật doanh thu tiền mặt

4.35.1 Lớp GUI

4.35.1.1 Hàm **btn\_UpdateCash\_Click**

Hàm nhận sự kiến xác nhận cập nhật số tiền mặt của một ngày doanh thu đó. Sau đó sẽ kiểm tra các thông tin được nhập vào nếu hợp lệ sẽ gửi thông tin vào hàm **UpdateRevenue** của lớp RevenueBUS.

4.35.2 Lớp BUS

4.35.2.1 Hàm **UpdateRevenue - RevenueBUS**

Hàm truyền các tham số vào **UpdateRevenue** của hàm **UpdateRevenue** ở lớp RevenueDAO và trả về kết quả cập nhât.

Code :

        public bool UpdateRevenue(int id, float cash, string note)

        {

            int rowsAffected = RevenueDAO.Instance.UpdateRevenue(id, cash, note);

            return rowsAffected > 0;

        }

4.35.3 Lớp DTO

Đối tượng **Revenue** được sử dụng để lưu trữ thông tin của phòng chiếu, ở đây sử dụng đối tượng **Revenue** với 8 đối số là *id, staff\_id, staffName, total\_price, cash , created\_at, status, note.*

4.35.4 Lớp DAO

4.35.4.1 Hàm **UpdateRevenue – RevenueDAO**

Hàm sẽ thực thi lệnh cập nhật bảng STATITICS với các tham số truyền vào và dùng hàm **ExecuteNonQuery** của lớp DataProvider để thực thi câu lệnh SQL, kết quả trả về là số dòng ảnh hưởng trên cơ sở dữ liệu.

Code :

        public int UpdateRevenue(int id, float cash, string note )

        {

            string sql = "UPDATE STATITICS SET Total\_Cash = @cash , Status = @status , Note = @note WHERE ID = @id ";

            return DataProvider.Instance.ExecuteNonQuery(sql, new object[]

            {

             cash.ToString(),

             "Đã xử lý",

             note,

             id

            });

        }

4.36 Chức năng Quản lí doanh thu theo tháng – Hiển thị doanh thu theo tháng.

4.36.1 Lớp GUI

4.36.1.1 Hàm **dtp\_Day\_ValueChanged**

Hàm lấy thông tin thời gian của datetimepicker để truyền vào hàm **GetRevenuesByMonth** của lớp RevenueBUS.

Code :

        private void dtp\_Day\_ValueChanged(object sender, EventArgs e)

        {

            List<DTO.Revenue> data = RevenueBUS.Instance.GetRevenuesByDay(dtp\_Day.Value);

            GV\_DataLoad(GV\_Data, data);

        }

4.36.1.2 Hàm **loadChart**

Hàm nhận List<Revenue> để lấy thông tin doanh thu các ngày của tháng đó dùng để vẽ biểu đồ thống kê doanh thu

Code :

  void loadChart(List<DTO.Revenue> revenues)

        {

            totalPrice = 0;

            chart\_RevenueByMonth.ChartAreas[0].AxisX.Interval = 1;

            chart\_RevenueByMonth.Series["Doanh Thu"].Points.Clear();

            lbl\_ChartTitle.Text = "Thống kê doanh thu theo tháng" + revenues[0].CreatedAt.ToString("MM/yyyy");

            foreach (DTO.Revenue row in revenues)

            {

                DateTime date = row.CreatedAt.Date;

                string thang = date.ToString("dd");

                float total = row.TotalPrice;

                totalPrice += total;

                chart\_RevenueByMonth.Series["Doanh Thu"].Points.AddXY(thang, total.ToString("N0"));

            }

            lbl\_TotalPrice.Text = totalPrice.ToString("N0") + " VND";

        }

4.36.1.3 Hàm **dtp\_Day\_ValueChanged**

Hàm sẽ lấy dữ lấy liệu dưới dạng List<Revenue> để hiển thị ra màn hình bằng DataGridView.

Code :

   private void GV\_DataLoad(DataGridView GV\_Data, List<DTO.Revenue> data)

        {

            GV\_Data.Rows.Clear();

            for (int i = 0; i < data.Count; i++)

            {

                DataGridViewRow zRow = new DataGridViewRow();

                zRow.CreateCells(GV\_Data);

                zRow.Cells[0].Value = data[i].CreatedAt.ToString("dd/MM/yyyy");

                zRow.Cells[1].Value = data[i].TotalPrice.ToString("N0");

                GV\_Data.Rows.Add(zRow);

            }

        }

4.36.2 Lớp BUS

4.36.2.1 Hàm **GetRevenuesByMonth - RevenueBUS**

Hàm truyền đối tham số *Datetime month* vào hàm GetRevenusByDay của lớp RevenueDAO để thấy dữ liệu.

Code :

        public List<Revenue> GetRevenuesByMonth(DateTime motnh)

        {

            return RevenueDAO.Instance.GetRevenuesByMonth(motnh);

        }

4.36.3 Lớp DTO

Đối tượng **Revenue** được sử dụng để lưu trữ thông tin của phòng chiếu, ở đây sử dụng đối tượng **Revenue** với 8 đối số là *id, staff\_id, staffName, total\_price, cash , created\_at, status, note.*

4.36.4 Lớp DAO

4.36.4.1 Hàm **GetRevenuesByMonth - RevenueDAO**

Hàm truy xuất dữ liệu từ cơ sở dữ liệu bằng cách gọi thủ tục **get\_revenue\_by\_month** để lấy thông tin doanh thu của các ngày trong tháng được truyền làm tham số là *Datatime month.* Hàm sẽ trả về một List<Revenue> lưu thông tin của doanh của các ngày trong tháng được truyền vào .

Code :

        public List<Revenue> GetRevenuesByMonth(DateTime motnh)

        {

            List<Revenue> revenues = new List<Revenue>();

            string sql = "EXEC get\_revenue\_by\_month @month ";

            DataTable dataTable = DataProvider.Instance.ExecuteQuery(sql, new object[] { motnh });

            foreach (DataRow row in dataTable.Rows)

            {

                float totalPrice = Convert.ToSingle(row["TotalPrice"]);

                DateTime createdAt = Convert.ToDateTime(row["BookingDate"]);

                Revenue revenue = new Revenue(createdAt, totalPrice);

                revenues.Add(revenue);

            }

            return revenues;

        }

4.37 Chức năng Quản lí doanh thu theo tháng – Lập báo cáo gửi lên trụ sở chính

4.37.1 Lớp GUI

4.37.1.1 Hàm **btn\_Report\_Click -**

Hàm xác nhận lập báo cáo để gửi lên trụ sở chính .

Code :

  private void btn\_Report\_Click(object sender, EventArgs e)

        {

            Report frm = new Report();

            frm.txt\_TotalPrice.Text = this.totalPrice.ToString();

            frm.dtp\_Day.Value = this.dtp\_Day.Value;

            frm.ShowDialog();

        }

4.37.1.2 Hàm **btn\_UpdateCash\_Click**

Hàm sẽ kiểm tra thông tin nhập vào từ Form Report, nếu hợp lệ thì sẽ gửi thông tin lên Form ReportConfirm để báo cáo doanh thu.

Code :

  private void btn\_UpdateCash\_Click(object sender, EventArgs e)

        {

            if (string.IsNullOrEmpty(txt\_TotalPrice.Text))

            {

                lbl\_Error.Text = "Vui lòng nhập tổng doanh thu";

                lbl\_Totalprice.ForeColor = Color.Red;

                txt\_TotalPrice.Focus();

                return;

            } else if (!BUS.ValidatorHelper.Instance.checkNumber(txt\_TotalPrice.Text))

            {

                lbl\_Error.Text = "Tổng doanh thu không hợp lệ! vui lòng nhập 1 số nguyên dương";

                lbl\_Totalprice.ForeColor = Color.Red;

                txt\_TotalPrice.Focus();

            }

            else

            {

                lbl\_Totalprice.ForeColor = Color.Black;

                ReportConfirm frm = new ReportConfirm(this);

                frm.lbl\_ReportTitle.Text = "Báo cáo doanh thu tháng " +dtp\_Day.Value.ToString("MM/yyyy");

                frm.lbl\_Note.Text = txt\_Note.Text;

                frm.lbl\_TotalPrice.Text =float.Parse(txt\_TotalPrice.Text).ToString("N0") + " VND";

                this.Visible = false;

                frm.ShowDialog();

            }

        }

4.37.1.3 Hàm **btn\_Confirm\_Click**

Khi được xác nhận hàm sẽ gửi thông tin lên trụ sở chính và thông báo thành công.

Code :

        private void btn\_Confirm\_Click(object sender, EventArgs e)

        {

            MessageBox.Show("Đã gửi báo cáo lên trụ sở");

            report.Close();

            this.Close();

        }

4.37.2 Lớp BUS

4.37.3 Lớp DTO

4.37.4 Lớp DAO

4.38 Chức năng Chốt sổ

4.38.1 Lớp GUI

4.38.1.1 Hàm **Aggregate\_Load**

Khi nhấn chốt sổ ở dao diện Admin thì hàm sẽ được gọi đến và hiện ra tổng số doanh thu của nhân viên đạt được trong ngày bằng hàm **get\_Total\_Price**.

Code :

        private void Aggregate\_Load(object sender, EventArgs e)

        {

            lbl\_Name.Text = session.name;

            lbl\_Revenu.Text = BUS.BookingBUS.Instance.get\_Total\_Price(session.id).ToString();

        }

4.38.1.2 Hàm **btn\_Confirm\_Click – Form Aggregate**

Lấy thông tin của nhân viên, thông tin doanh thu, vào ghi chú lưu vào hàm **create** ở lớp **RevenueBUS** và lưu các thông tin trên vào file Excel bằng hàm **addRevenueToExcel** trong lớp ExcelHelper.

Code :

  private void btn\_Confirm\_Click(object sender, EventArgs e)

        {

            if (BUS.RevenueBUS.Instance.create(session.id, lbl\_Revenu.Text, DateTime.Now, "Chờ xử lý", txt\_Note.Text)

               && BUS.ExcelHelper.Instance.addRevenueToExcel(session.name, lbl\_Revenu.Text, null, DateTime.Now, "Chờ xử lý"))

            {

                MessageBox.Show("Lưu thông tin thành công");

                this.Close();

            }

            else

            {

                MessageBox.Show("Lưu thông tin không thành công");

            }

        }

4.38.2 Lớp BUS

4.38.2.1 Hàm **create – RevenueBUS**

Hàm tạo đối tượng **Revenue** bằng các tham số đầu vào và truyền vào hàm **add** của lớp RevenueDAO. Hàm trả về kết quả thêm .

Code :

public bool create(int staff\_id, string total\_price, DateTime created\_at, string status,

string note)

        {

            return DAO.RevenueDAO.Instance.add(new Revenue(staff\_id,session.name, float.Parse(total\_price), 0, created\_at, status, note)) == 1 ? true : false;

        }

4.38.3 Lớp DTO

Đối tượng **Revenue** được sử dụng để lưu trữ thông tin của phòng chiếu, ở đây sử dụng đối tượng **Revenue** với 8 đối số là *id, staff\_id, staffName, total\_price, cash , created\_at, status, note.*

4.38.4 Lớp DAO

4.38.4.1 Hàm **add – RevenueDAO**

Hàm nhận tham số đầu vào là một đối tượng Revenue và lấy thông tin của đối tượng truyền vào cơ sở dữ liệu qua lệnh Insert . Sau đó dùng hàm **ExecuteNonQuery** ở lớp DataProvider để thực thi câu lệnh.

Code :

        public int add(Revenue revenue)

        {

            string sql = "INSERT INTO [dbo].[STATITICS] ([STAFF\_ID] ,[TOTAL\_PRICE] ,[TOTAL\_CASH] ,[CREATED\_AT] ,[STATUS] ,[NOTE])" +

                " VALUES( @Staff\_id , @TotalPrice , @TotalCash , @Created\_at , @Status , @Note )";

            return DataProvider.Instance.ExecuteNonQuery(sql, new Object[] { revenue.Staff\_id, revenue.TotalPrice, revenue.Cash, revenue.CreatedAt, revenue.Status, revenue.Note});

        }

4.38 Chức năng Đặt vé trực tuyến

4.38.1 Lớp GUI

4.38.1.1 Hàm **btn\_Ticket\_Click**

Hàm gọi đến Form Ticket\_Movie để tiến hành đặt vé.

Code :

        private void btn\_Ticket\_Click(object sender, EventArgs e)

        {

            Ticket\_Movie frm = new Ticket\_Movie(this) { Dock = DockStyle.Fill, TopLevel = false, TopMost = true };

            this.panel\_Load.Controls.Add(frm);

            frm.Show();

        }

4.39.1.2 Hàm **Ticket\_Movie\_Load**

Hàm hiển thị các bộ phim đang có lịch chiếu lên màn hình bằng hàm **GV\_DataLoad** và với danh sách phim lấy từ hàm **getMoviesInSchedule** lấy từ MovieBUS.

Code :

        private void Ticket\_Movie\_Load(object sender, EventArgs e)

        {

            pro\_Bar.Value = Convert.ToInt32(1);

            List<DTO.Movie> data = MovieBUS.Instance.getMoviesInSchedule();

            GV\_DataLoad(GV\_Data, data);

        }

4.39.1.3 Hàm **btn\_Confirm\_Click – Form : Ticket\_Movie**

Hàm xác nhận bộ phim được chọn và lấy *ID* phim và *Tilte* phim lưu vào đối tượng **session**. Sau đó hiển thị Form Ticket\_Schedule.

Code :

        private void btn\_Confirm\_Click(object sender, EventArgs e)

        {

            session.movieID = Convert.ToInt32(getIDMovie());

            session.movieTitle = getMovieTitle();

            Ticket\_Schedule frm = new Ticket\_Schedule(this, this.\_admin) { Dock = DockStyle.Fill, TopLevel = false, TopMost = true };

            this.\_admin.panel\_Load.Controls.Add(frm);

            this.Visible = false;

            frm.Show();

        }

4.39.1.4 Hàm **Ticket\_Schedule\_Load**

Hiển thị các lịch chiếu của phim đã chọn bằng hàm **loadDGV** và danh sách lịch chiếu lấy từ hàm **getScheduleOfMovie** của lớp ScheduleDAO.

Code :

        private void Ticket\_Schedule\_Load(object sender, EventArgs e)

        {

            pro\_Bar.Value = Convert.ToInt32(25);

            List<string[]> list = ScheduleDAO.Instance.getScheduleOfMovie(session.movieID.ToString());

            loadDGV(list);

        }

4.39.1.5 Hàm **btn\_Confirm\_Click – Form : Ticket\_Schedule**

Hàm xác nhận chọn lịch chiếu của bộ phim đã chọn và lấy *ID*,*Tilte*,Schedule\_id của bộ phim đặt vào đối tượng **session**. Sau đó hiển thị Form Ticket\_Seat.

Code :

        private void btn\_Confirm\_Click(object sender, EventArgs e)

        {

            session.showRoomID = Convert.ToInt32(getShowRoomNumber());

            session.scheduleID = Convert.ToInt32(getScheduleID());

            session.startTime = getStartTime();

            Ticket\_Seat frm = new Ticket\_Seat(this, this.admin ) { Dock = DockStyle.Fill, TopLevel = false, TopMost = true };

            this.ticket\_Movie.\_admin.panel\_Load.Controls.Add(frm);

            this.Visible = false;

            frm.Show();

        }

4.39.1.5 Hàm **CreateSeats**

Khi hàm **CreateSeats** được thực thi , hàm sẽ tạo ra các Button là các ô ghế ra màn hình. Có 2 loại ghế là Couple và Standard với kích thước khác nhau. Chỗ ngồi đã được đặt thì sẽ có màu DimGray và Button đó được đặt thuộc tính là Enabled = false .Còn những thế chưa đặt thì có màu WhiteSmoke.

Code :

            foreach (Seat seat in seats)

            {

                Button button = new Button();

                button.Name = "btnSeat" + seat.ID.ToString();

                button.Text = seat.SeatNumber;

                if (seat.Type == "Couple")

                {

                    button.Size = new Size(170, 50);

                }

                else

                {

                    button.Size = new Size(85, 50);

                }

                button.FlatStyle = FlatStyle.Flat;

                button.FlatAppearance.BorderSize = 0;

                button.BackColor = Color.WhiteSmoke;

                button.Enabled = true; // enable tất cả các ghế

                button.BackColor = seat.Type == "Standard" ? Color.WhiteSmoke : Color.Tomato;

                if (seat.IsBooked)

                {

                    button.BackColor = Color.DimGray;

                    button.Enabled = false;

                }

                // Thêm button vào panel

                fpnl\_Seats.Controls.Add(button);

            }

// Event click của từng button ghế

            foreach (Control control in fpnl\_Seats.Controls)

            {

                if (control is Button button)

                {

                    button.Click += (sender, e) =>

                    {

                        Seat seat = seats.FirstOrDefault(s => s.SeatNumber == button.Text);

                        int groupCount = 0;

                        if (seat != null)

                        {

                            // Check if the selected seat is available

                            if (button.BackColor == Color.WhiteSmoke || button.BackColor == Color.Tomato)

                            {

                                if (numberOfSeats == 0)

                                {

                                    MessageBox.Show("Chưa chọn số lượng hoặc đã chọn đủ ghế!");

                                }

                                else

                                {

                                    groupCount = int.Parse(getValueOfRadio(pnl\_SeatsLen));

                                    bool flag = true;

                                    List<Seat> availableSeats = new List<Seat>();

                                    for (int i = 1; i <= groupCount; i++)

                                    {

                                        availableSeats = getAvailableSeats(seat, groupCount, i);

                                        if (isValid(availableSeats, groupCount))

                                        {

                                            break;

                                        }

                                    }

                                    if (!isValid(availableSeats, groupCount))

                                    {

                                        flag = false;

                                    }

                                    if (flag)

                                    {

                                        foreach (Seat availableSeat in availableSeats)

                                        {

                                            availableSeat.IsBooked = true;

                                            selectedSeats.Add(availableSeat);

                                            Button selectedButton = fpnl\_Seats.Controls.Find("btnSeat" + availableSeat.ID.ToString(), true).FirstOrDefault() as Button;

                                            selectedButton.BackColor = Color.Red;

                                        }

                                        numberOfSeats -= groupCount;

                                        validateRadio(numberOfSeats);

                                    }

                                    else

                                    {

                                        MessageBox.Show("Ghế đã chọn không hợp lệ!");

                                    }

                                }

                            }

                            else if (button.BackColor == Color.Red)

                            {

                                int couter = 0;

                                // deselect left side

                                int id = seats.FindIndex(s => s.ID == seat.ID);

                                Seat deselectedSeat = seats[id];

                                Button deselectedButton = fpnl\_Seats.Controls.Find("btnSeat" + deselectedSeat.ID.ToString(), true).FirstOrDefault() as Button;

                                while (deselectedButton.BackColor == Color.Red)

                                {

                                    couter++;

                                    deselectedSeat.IsBooked = false;

                                    selectedSeats.Remove(deselectedSeat);

                                    deselectedButton.BackColor = deselectedSeat.Type == "Standard" ? Color.WhiteSmoke : Color.Tomato;

                                    id--;

                                    if (id < 0)

                                    {

                                        break;

                                    }

                                    deselectedSeat = seats[id];

                                    deselectedButton = fpnl\_Seats.Controls.Find("btnSeat" + deselectedSeat.ID.ToString(), true).FirstOrDefault() as Button;

                                }

                                // deselect right side

                                id = seats.FindIndex(s => s.ID == seat.ID) + 1;

                                deselectedSeat = seats[id];

                                deselectedButton = fpnl\_Seats.Controls.Find("btnSeat" + deselectedSeat.ID.ToString(), true).FirstOrDefault() as Button;

                                while (deselectedButton.BackColor == Color.Red)

                                {

                                    couter++;

                                    deselectedSeat.IsBooked = false;

                                    selectedSeats.Remove(deselectedSeat);

                                    deselectedButton.BackColor = deselectedSeat.Type == "Standard" ? Color.WhiteSmoke : Color.Tomato;

                                    id++;

                                    if (id >= 90)

                                    {

                                        break;

                                    }

                                    deselectedSeat = seats[id];

                                    deselectedButton = fpnl\_Seats.Controls.Find("btnSeat" + deselectedSeat.ID.ToString(), true).FirstOrDefault() as Button;

                                }

                                numberOfSeats += couter;

                                validateRadio(numberOfSeats);

                            }

                        }

                    };

                }

            }

Các ghế sẽ được load từ database và hệ thống sẽ tạo ra các button tương ứng đại diện cho các ghế. Dựa vào thuộc tính Type và IsBooked của các ghế để định dạng kích thước và màu sắc cũng như việc có enable (cho phép chọn) ghế đó hay không. Sau khi đã định dạng xong hệ thống sẽ thêm các ghế vào Flowlayoutpanel để hiển thị lên cho người dùng. Sau đó sẽ thêm các event click cho từng ghế. Nếu click vào ghế chưa được chọn và numberOfSeats > 0 thì sẽ chọn các ghế bằng cách lấy các ghế hợp lệ bằng hàm **getAvailbleSeats** và kiểm tra xem các ghế đó có hợp lệ hay không bằng hàm **isValid.** Nếu click vào ghế đang chọn thì sẽ bỏ chọn tất cả các ghế liền sát ghế đó. Nếu đã chọn đủ số ghế sẽ thông báo cho người dùng.

Hàm **isValid():**

Code:

private bool isValid(List<Seat> availableSeats, int groupCount)

        {

            if (availableSeats.Count != groupCount || availableSeats.Count == 0)

            {

                return false;

            }

            Seat firstSeat = availableSeats.First();

            Seat lastSeat = availableSeats.Last();

            char row = firstSeat.SeatNumber[0];

            int seatNumber = int.Parse(firstSeat.SeatNumber.Substring(1));

            // check left side

            string seatLeft1 = row.ToString() + (seatNumber - 1).ToString();

            Seat left1 = seats.FirstOrDefault(s => s.SeatNumber == seatLeft1);

            string seatLeft2 = row.ToString() + (seatNumber - 2).ToString();

            Seat left2 = seats.FirstOrDefault(s => s.SeatNumber == seatLeft2);

            if (left1 != null)

            {

                if (!left1.IsBooked)

                {

                    if (left2 == null || left2.IsBooked)

                    {

                        return false;

                    }

                }

            }

            // check right side

            seatNumber = int.Parse(lastSeat.SeatNumber.Substring(1));

            string seatRight1 = row.ToString() + (seatNumber + 1).ToString();

            Seat right1 = seats.FirstOrDefault(s => s.SeatNumber == seatRight1);

            string seatRight2 = row.ToString() + (seatNumber + 2).ToString();

            Seat right2 = seats.FirstOrDefault(s => s.SeatNumber == seatRight2);

            if (right1 != null)

            {

                if (!right1.IsBooked)

                {

                    if (right2 == null || right2.IsBooked)

                    {

                        return false;

                    }

                }

            }

            return true;

        }

Hàm này sẽ kiểm tra 1 chuỗi các ghế có hợp lệ hay không. 1 chuỗi ghế hợp lệ là chuỗi các ghế sao cho các ghế ngoài cùng bên trái ngoài cùng không bị lủng (có 1 ghế chưa đặt giữ chuỗi ghế đang chọn và các ghế khác đã đặt hoặc không có ghế).

Hàm **getAvaiableSeats():**

private List<Seat> getAvailableSeats(Seat seat, int groupCount, int index)

        {

            List<Seat> availableSeats = new List<Seat>();

            char row = seat.SeatNumber[0];

            int seatNumber = int.Parse(seat.SeatNumber.Substring(1));

            int i = seatNumber - groupCount + index;

            availableSeats.Clear();

            for (int j = i; j < i + groupCount; j++)

            {

                string seatToCheck = row.ToString() + j.ToString();

                Seat selectedSeat = seats.FirstOrDefault(s => s.SeatNumber == seatToCheck);

                if (selectedSeat != null)

                {

                    if (!selectedSeat.IsBooked)

                    {

                        availableSeats.Add(selectedSeat);

                    }

                    else

                    {

                        break;

                    }

                }

            }

            if (availableSeats.Count == groupCount)

            {

                return availableSeats;

            }

            return availableSeats;

        }

Hàm này sẽ kiểm tra các dãy ghế liên tiếp với ghế đực chọn và trả về chuỗi các ghế có thể đặt được và có ghế được chọn.

4.39.1.6 Hàm **Ticket\_ComboFood\_Load**

Khi Form Ticket\_ComboFood được hiển thị , các ComboFood sẽ được lấy từ hàm **getFoodCombo** của lớp **BookingBUS**  và tạo ra các UC\_FoodComboCard để hiển thị lên hàm hình.

Code :

 private void Ticket\_ComboFood\_Load(object sender, EventArgs e)

        {

            pro\_Bar.Value = Convert.ToInt32(75);

            this.products = new List<Items>();

            this.combos = new List<Items>();

            List<DTO.FoodCombo> foodCombos = BUS.BookingBUS.Instance.getFoodCombos();

            foreach (DTO.FoodCombo foodCombo in foodCombos)

            {

                Image image = null;

                try

                {

                    var outPutDirectory = Path.GetDirectoryName(Assembly.GetExecutingAssembly().CodeBase);

                    int index = outPutDirectory.IndexOf("GUI");

                    outPutDirectory = outPutDirectory.Substring(0, index);

                    var iconPath = Path.Combine(outPutDirectory, "GUI\\img\\" + foodCombo.image);

                    string icon\_path = new Uri(iconPath).LocalPath;

                     image = Bitmap.FromFile(icon\_path);

                }catch(Exception ex)

                {

                    MessageBox.Show("Có lỗi xảy ra khi load ảnh");

                }

                string infor = foodCombo.foodQuantity.ToString() +" "+ foodCombo.food + " + "

                            + foodCombo.drinkQuantity.ToString() +" "+ foodCombo.drink;

                UC\_FoodComboCard uC\_FoodComboCard = new UC\_FoodComboCard(image, foodCombo.name, infor, foodCombo.price, foodCombo.id);

                uC\_FoodComboCard.addButton.Click += (sender, e) =>

                {

                    if (!uC\_FoodComboCard.isSelected)

                    {

                        string infor = uC\_FoodComboCard.information;

                        string name = uC\_FoodComboCard.name + " - " + uC\_FoodComboCard.price.ToString();

                        Items combo = new Items(int.Parse(uC\_FoodComboCard.id), name);

                        List<Items> combos = new List<Items>();

                        combos.Add(combo);

                        this.comboQuantity++;

                        UC\_ProductItem uC\_ProductItem = new UC\_ProductItem(combos, this.comboQuantity);

                        fpnl\_SelectedCombo.Controls.Add(uC\_ProductItem);

                        uC\_FoodComboCard.isSelected = true;

                        uC\_ProductItem.removeButton.Click += (sender, e) =>

                        {

                            fpnl\_SelectedCombo.Controls.Remove(uC\_ProductItem);

                            uC\_FoodComboCard.isSelected = false;

                        };

                    }

                };

                fpnl\_Combo.Controls.Add(uC\_FoodComboCard);

            }

            List<Items> products = BUS.ProductBUS.Instance.getProducts();

            insertUC\_ProductItem(products, 1);

            this.itemsQuantity = 1;

            this.comboQuantity= 0;

        }

4.39.1.7 Hàm **insertUC\_ProductItem**

Hàm sẽ thêm List<Items> vào UC\_ProductItem. Hàm sẽ bắt sự kiện thêm hoặc xóa để thêm xóa các UC\_ProductItem trên Panel.

Code ;

        private void insertUC\_ProductItem(List<Items> data, int id)

        {

            UC\_ProductItem uC\_ProductItem = new UC\_ProductItem(data,id);

            uC\_ProductItem.removeButton.Click += (sender, e) =>

            {

                fpnl\_product.Controls.Remove(uC\_ProductItem);

            };

            fpnl\_product.Controls.Add(uC\_ProductItem);

        }

4.39.1.8 Hàm **btn\_Confirm\_Click**

Hàm sẽ kiểm tra các UC\_ProductItem và UC\_ProductItem và thêm các foodcom và product và session. Sau đó gọi đến Form Ticket\_Infor.

Code :

   private void btn\_Confirm\_Click(object sender, EventArgs e)

        {

            this.products.Clear();

            this.combos.Clear();

            foreach (Control control in fpnl\_product.Controls)

            {

                if (control is UC\_ProductItem productItem)

                {

                    if (int.Parse(productItem.quantity)  > 0 && productItem.name != "Default")

                    {

                        Items product = new Items(int.Parse(productItem.id), productItem.quantity);

                        this.products.Add(product);

                    }

                }

            }

            foreach (Control control in fpnl\_SelectedCombo.Controls)

            {

                if (control is UC\_ProductItem comboItem)

                {

                    if (int.Parse(comboItem.quantity) > 0 )

                    {

                        Items combo = new Items(int.Parse(comboItem.id), comboItem.quantity);

                        this.combos.Add(combo);

                    }

                }

            }

            session.products = this.products;

            session.foodCombos = this.combos;

            Ticket\_Infor frm = new Ticket\_Infor(this,\_admin) { Dock = DockStyle.Fill, TopLevel = false, TopMost = true };

            this.\_admin.panel\_Load.Controls.Add(frm);

            this.Visible = false;

            frm.Show();

        }

4.39.1.9 Hàm **Ticket\_Infor\_Load**

Hàm sẽ hiểm thị hóa đơn ra bao gồm người lập hóa đơn , thời gian tạo từ session và tổng số tiền của hóa đơn từ hàm **passDataToTicketInfor** từ BookingBUS.

  private void Ticket\_Infor\_Load(object sender, EventArgs e)

        {

            pro\_Bar.Value = Convert.ToInt32(100);

            lbl\_Seller.Text = "Seller: "+ session.name;

            lbl\_Created.Text = session.name;

            this.totalMoney =  BUS.BookingBUS.Instance.passDataToTicketInfor(lbl\_Created, lbl\_MovieTitle, lbl\_StartTime, lbl\_ShowRoomNumber, lbl\_TotalTicket

                ,fpnl\_Ticket, fpnl\_Product, lbl\_TotalProductPrice, lbl\_TotalPrice, lbl\_TotalMoney);

        }

4.39.1.10 Hàm **btn\_CheckPromoCode\_Click**

Hàm sẽ tính lại tổng tiền của hóa đơn bằng cách kiểm tra mã giảm giá của người dùng nhập vào bằng hàm **checkPromotion** ở lớp BookingBUS.

Code :

        private void btn\_CheckPromoCode\_Click(object sender, EventArgs e)

        {

          this.totalMoney =  BUS.BookingBUS.Instance.checkPromotion(txt\_PromotionCode.Text, lbl\_TotalMoney,lbl\_Error, lbl\_Promotion);

        }

4.39.1.10 Hàm **btn\_Payment\_Click**

Hàm sẽ hiển Form Ticket\_PaymentMethod khi dùng nhấn vào Button Thanh toán.

Code :

        private void btn\_Payment\_Click(object sender, EventArgs e)

        {

            Ticket\_PaymentMethod frm = new Ticket\_PaymentMethod(this.totalMoney, this);

            frm.ShowDialog();

        }

4.39.2 Lớp BUS

4.39.2.1 Hàm **getMoviesInSchedule : MovieBUS**

Lấy thông tin các phim đang có lịch chiếu từ cơ sở dữ liệu thông qua hàm , **getMoviesInSchedule** trong lớp MovieDAO.

Code :

        public List<Movie> getMoviesInSchedule()

        {

            return DAO.MovieDAO.Instance.GetMoviesInSchedule();

        }

4.39.2.2 Hàm **getFoodCombos : BookingBUS**

Hàm lấy thông tin các FoodCombo từ cơ sở dữ liệu thông qua hàm **selectAll** của lớp FoodComboDAO.

Code :

        public List<FoodCombo> getFoodCombos()

        {

            return DAO.FoodComboDAO.Instance.selectAll();

        }

4.39.2.3 Hàm **passDataToTicketInfor : BookingBUS**

Hàm sẽ lấy dữ liệu được truyền vào và lấy thông tin lịch chiếu từ hàm **getByID** của lớp ScheduleDAO, thông tin phim từ hàm **getByID** của lớp MovieDAO, thông tin phòng chiếu từ hàm **getByID** của lớp ShowRoomDAO, thông tin của sản phẩm từ **getByID** của ProductDAO , thông tin FoodCombo từ FoodComboDAO để tính tổng số tiền của hóa đơn.

Code :

 public float passDataToTicketInfor(KryptonLabel lbl\_Created, KryptonLabel lbl\_MovieTitle, KryptonLabel

lbl\_StartTime

            , KryptonLabel lbl\_ShowRom,  KryptonLabel lbl\_TotalTicketPrice, FlowLayoutPanel fpnl\_Ticket

            , FlowLayoutPanel fpnl\_Product, KryptonLabel lbl\_TotalProductPrice, KryptonLabel lbl\_TotalPrice

            , KryptonLabel lbl\_TotalMoney)

        {

            session.bSeats = "";

            session.bFoods = "";

            Schedule schedule = DAO.ScheduleDAO.Instance.getByID(session.scheduleID.ToString());

            DateTime startTime = (DateTime) schedule.StartTime;

            lbl\_StartTime.Text = startTime.ToString("HH:mm - dd/MM/yyyy");

            startTime = DateTime.Now;

            lbl\_Created.Text = startTime.ToString("HH:mm - dd/MM/yyyy");

            Movie movie = DAO.MovieDAO.Instance.getByID(session.movieID.ToString());

            session.bMovie = movie.Title + " | " + movie.Duration.ToString() + " phút";

            lbl\_MovieTitle.Text = session.bMovie;

            ShowRoom showRoom = DAO.ShowRoomDAO.Instance.getShowRoomByID(schedule.Theater\_id.ToString());

            session.bShowroom = showRoom.showRoomNumber;

            lbl\_ShowRom.Text = session.bShowroom;

            this.totalTicket = 0;

            foreach (Seat seat in session.seats)

            {

                string seatPrice = "";

                if (seat.Type == "Standard")

                {

                    this.totalTicket += schedule.price;

                }

                else

                {

                    this.totalTicket += schedule.price + 30000;

                }

                seatPrice = seat.SeatNumber + " - " + schedule.price.ToString("N0") + " VND";

                session.bSeats += seatPrice +",";

                KryptonLabel lbl\_seat = new Krypton.Toolkit.KryptonLabel();

                lbl\_seat.StateCommon.ShortText.Color1 = System.Drawing.Color.Black;

                lbl\_seat.StateCommon.ShortText.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point);

                lbl\_seat.TabIndex = 95;

                lbl\_seat.Text= seatPrice;

                fpnl\_Ticket.Controls.Add(lbl\_seat);

            }

            lbl\_TotalTicketPrice.Text = this.totalTicket.ToString("N0") + " VND";

            this.totalProduct = 0;

            foreach (Items product in session.products)

            {

                Product productDTO = DAO.ProductDAO.Instance.getByID(product.Id.ToString());

                float price = productDTO.Price \* int.Parse(product.Name);

                string productPrice = productDTO.Name + " - X"+product.Name + " - "

                                        + price.ToString("N0") + " VND";

                session.bFoods += productPrice + ",";

                KryptonLabel lbl\_seat = new Krypton.Toolkit.KryptonLabel();

                lbl\_seat.StateCommon.ShortText.Color1 = System.Drawing.Color.Black;

                lbl\_seat.StateCommon.ShortText.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point);

                lbl\_seat.TabIndex = 95;

                lbl\_seat.Text = productPrice;

                fpnl\_Product.Controls.Add(lbl\_seat);

                this.totalProduct += price;

            }

            foreach (Items combo in session.foodCombos)

            {

                FoodCombo foodCombo = DAO.FoodComboDAO.Instance.getByID(combo.Id.ToString());

                float price = foodCombo.price \* int.Parse(combo.Name);

                string productPrice = $"{foodCombo.name} - X{combo.Name} - {price.ToString("N0")} VND";

                session.bFoods += productPrice + ",";

                KryptonLabel lbl\_seat = new Krypton.Toolkit.KryptonLabel();

                lbl\_seat.StateCommon.ShortText.Color1 = System.Drawing.Color.Black;

                lbl\_seat.StateCommon.ShortText.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point);

                lbl\_seat.TabIndex = 95;

                lbl\_seat.Text = productPrice;

                fpnl\_Product.Controls.Add(lbl\_seat);

                this.totalProduct += price;

            }

            lbl\_TotalProductPrice.Text = this.totalProduct.ToString("N0") + " VND";

            this.totalPrice= this.totalProduct + this.totalTicket;

            lbl\_TotalPrice.Text = this.totalPrice.ToString("N0") + " VND";

            lbl\_TotalMoney.Text = this.totalPrice.ToString("N0") + " VND";

            session.promotionID = -1;

            session.discount = 0;

            return this.totalPrice;

        }

4.39.2.4 Hàm **checkPromotion: BookingBUS**

Hàm sẽ lấy mã giảm giá để kiểm tra bằng hàm **GetPromotionByCode** của lớp PromotionDAO. Nếu mã giảm giá đúng thì tổng số tiền hóa đơn sẽ được tính lại hiển thị ra màn hình .

Code :

public float  checkPromotion(string code, KryptonLabel lbl\_TotalMoney, KryptonLabel lbl\_Error,

KryptonLabel lbl\_Promotion)

        {

           DTO.Promotion promotion =  PromotionDAO.Instance.GetPromotionByCode(code);

            float totalMoney = this.totalPrice;

            if (promotion == null)

            {

                lbl\_Error.Text = "Mã giảm giá không tồn tại";

                lbl\_Promotion.Text = "0 VND";

                session.promotionID = -1 ;

            }

            else if (promotion.Status == "Đã kết thúc")

            {

                lbl\_Error.Text = "Chương trình khuyến mãi đã hết!";

                lbl\_Promotion.Text = "0 VND";

                session.promotionID = -1;

            }

            else

            {

                lbl\_Error.Text = string.Empty;

                float discout = promotion.Discount \* this.totalPrice / 100;

                lbl\_Promotion.Text ="-"+ discout.ToString("N0") + " VND";

                totalMoney = (this.totalPrice - discout);

                session.promotionID = promotion.ID;

                session.discount = promotion.Discount;

            }

            lbl\_TotalMoney.Text = totalMoney.ToString("N0") + " VND";

            return totalMoney;

        }

4.39.3 Lớp DTO

Sử dụng các DTO Movie, Schedule, FoodCombo, Product, Ticket.

4.39.4 Lớp DAO

4.39.4.1 Hàm **selectAll: MovieDAO**

Hàm **selectAll** gọi lệnh select lên cơ sở dữ liệu để lấy tất cả nguyên vật liệu trên cơ sở dữ liệu. Lệnh SQL sẽ được truyền vào hàm **ExecuteQuery** của lớp DAO.DataProvider. Hàm sẽ thêm tất cả nguyên vật liệu vào List<Movie> và trả về List< Movie > này.

Code :

        public List<Movie> selectAll()

        {

            List<Movie> movies = new List<Movie>();

            string sql = "Select \* from Movie";

            DataTable dataTable = DataProvider.Instance.ExecuteQuery(sql);

            foreach (DataRow row in dataTable.Rows)

            {

                string id = row["ID"].ToString();

                string title = row["title"].ToString();

                string genre = row["genre"].ToString();

                int duration = int.Parse(row["duration"].ToString());

                float rating = float.Parse(row["rating"].ToString());

                string story = row["story"].ToString();

                string poster = row["poster"].ToString();

                Movie movie = new Movie(id, title, genre, duration, rating, story, poster);

                movies.Add(movie);

            }

            return movies;

        }

4.39.4.2 Hàm **getPromotionByCode: PromotionDAO**

Hàm sẽ gọi lệnh Select với tham số là mã code được truyền vào và lấy ra thông tin Promotion. Dùng hàm **ExecuteQuery** của lớp DataProvider để thực thi.

Code :

public Promotion GetPromotionByCode(string promoCode)

        {

            string sql = "SELECT \* FROM Promotion WHERE CODE = @promoCode";

            DataTable dataTable = DataProvider.Instance.ExecuteQuery(sql, new object[] { promoCode });

            // Kiểm tra xem có dòng dữ liệu trả về hay không

            if (dataTable.Rows.Count > 0)

            {

                DataRow row = dataTable.Rows[0];

                // Đọc các giá trị từ dòng dữ liệu và tạo đối tượng Promotion tương ứng

                int id = Convert.ToInt32(row["ID"]);

                string code = row["CODE"].ToString();

                float discount = Convert.ToSingle(row["DISCOUNT"]);

                string status = row["STATUS"].ToString();

                string note = row["NOTE"].ToString();

                DTO.Promotion promotion = new DTO.Promotion()

                {

                    ID = id,

                    Code = code,

                    Discount = discount,

                    Status = status,

                    Note = note

                };

                return promotion;

            }

            return null;

        }

4.39.4.3 Hàm **getMoviesInSchedule : MovieDAO**

Lấy các phim trên cơ sở dữ liệu đang có lịch chiếu bằng cách gọi thủ tục get\_movies\_in\_schedule , thực thi bằng cách gọi hàm **ExecuteQuery** .

Code :

 public List<Movie> GetMoviesInSchedule()

        {

            List<Movie> movies = new List<Movie>();

            string sql = "exec get\_movies\_in\_schedule";

            DataTable dataTable = DataProvider.Instance.ExecuteQuery(sql);

            foreach (DataRow row in dataTable.Rows)

            {

                string id = row["ID"].ToString();

                string title = row["title"].ToString();

                string genre = row["genre"].ToString();

                int duration = int.Parse(row["duration"].ToString());

                float rating = float.Parse(row["rating"].ToString());

                string story = row["story"].ToString();

                string poster = row["poster"].ToString();

                Movie movie = new Movie(id, title, genre, duration, rating, story, poster);

                movies.Add(movie);

            }

            return movies;

        }

4.39.4.4 Hàm **getByID: MovieDAO,ProductDAO,ScheduleDAO,**

**ShowRoomDAO**

Các hàm sẽ lấy các đối tượng dựa trên ID được truyền và thực thi bằng hàm **ExecuteQuery** của DataProvider.

Code GetByID Movie :

public Movie getByID(string id)

        {

            List<Movie> movies = new List<Movie>();

            string sql = "Select \* from Movie where id = @id ";

            DataTable dataTable = DataProvider.Instance.ExecuteQuery(sql, new Object[] {id});

            try

            {

                DataRow row = dataTable.Rows[0];

                string title = row["title"].ToString();

                string genre = row["genre"].ToString();

                int duration = int.Parse(row["duration"].ToString());

                float rating = float.Parse(row["rating"].ToString());

                string story = row["story"].ToString();

                string poster = row["poster"].ToString();

                Movie movie = new Movie(id, title, genre, duration, rating, story, poster);

                return movie;

            }

            catch (Exception e)

            {

                return null;

            }

        }

Code GetByID Product :

        public Product getByID(string id)

        {

            string sql = "SELECT \* FROM product WHERE ID = @ID";

            DataTable dataTable = DataProvider.Instance.ExecuteQuery(sql, new object[] { id });

            if (dataTable.Rows.Count > 0)

            {

                DataRow row = dataTable.Rows[0];

                string name = row["NAME"].ToString();

                float price = float.Parse(row["Price"].ToString());

                int quantity = int.Parse(row["Quantity"].ToString());

                int lastOrder = int.Parse(row["LAST\_ORDER"].ToString());

                string type = row["TYPE"].ToString();

                //DateTime? expiry\_date = row["expiry\_date"].ToString() == string.Empty ? null : (DateTime?)DateTime.Parse(row["expiry\_date"].ToString());

                Product product = new Product(id, name, price, quantity, type,  lastOrder);

                return product;

            }

            return null; // ID không tồn tại trong danh sách sản phẩm

        }

Code GetByID Schedule :

  public Schedule getByID(string id)

        {

            string sql = "Select \* from Schedule where id = @id ";

            DataTable dataTable = DataProvider.Instance.ExecuteQuery(sql, new Object[] {id});

            try

            {

                DataRow row = dataTable.Rows[0];

                int showRoom = int.Parse(row["showroom\_id"].ToString());

                int movie\_id = int.Parse(row["movie\_id"].ToString());

                float price = float.Parse(row["price"].ToString());

                DateTime? starttime = row["starttime"].ToString() == string.Empty ? null : (DateTime?)DateTime.Parse(row["starttime"].ToString());

                DateTime? endtime = row["endtime"].ToString() == string.Empty ? null : (DateTime?)DateTime.Parse(row["starttime"].ToString());

                Schedule schedule = new Schedule(id, showRoom, movie\_id, starttime, endtime,price);

                return schedule;

            }catch (Exception ex)

            {

                return null;

            }

        }

Code GetByID Showrooom :

 public ShowRoom getShowRoomByID(string id)

        {

            string sql = "Select \* from ShowRoom where id = @id ";

            DataTable dataTable = DataProvider.Instance.ExecuteQuery(sql, new Object[] { id });

            try

            {

                DataRow row = dataTable.Rows[0];

                string showrRooNumber = row["showroomnumber"].ToString();

                ShowRoom showRoom = new ShowRoom(id, showrRooNumber);

                return showRoom;

            }

            catch (Exception)

            {

                return null;

            }

        }

4.40 Chức năng Thanh toán

4.40.1 Lớp GUI

4.40.1.1 Hàm **btn\_Confirm\_Click – Form : CashPayment**

Hàm lấy thông tin số tiền được nhập vào từ khách hàng và nhập thông tin. Hàm sẽ kiểm tra . Nếu số tiền nhập chính xác thì sẽ tạo **createBooking** trong lớp BookingBUS và thông báo tiền thừa. Còn không sẽ hiển thị thông báo lỗi.

Code :

        private void btn\_Confirm\_Click(object sender, EventArgs e)

        {

            float moneyIn = float.Parse(txt\_MoneyIn.Text);

            if (moneyIn < this.totalMoney)

            {

                string message = $"Tổng hóa đơn: {totalMoney.ToString("N0")} VND" +

                    $"\nThiếu: {(totalMoney - moneyIn).ToString("N0")} VND" +

                    $"\nThanh toán Thất bại! Vui lòng đưa thêm tiền.";

                MessageBox.Show(message);

            }

            else

            {

                BookingBUS.Instance.createBooking();

                string message = $"Tổng hóa đơn: {totalMoney.ToString("N0")} VND" +

                    $"\nTiền thừa: {(moneyIn - totalMoney).ToString("N0")} VND" +

                    $"\nThanh toán thành công!";

                MessageBox.Show(message);

                this.\_PaymentMethod.Close();

                this.\_PaymentMethod.\_ticketInfor.Close();

                this.Close();

            }

        }

4.40.1.2 Hàm **btn\_Confirm\_Click – Form : ATMPayment**

Khi thanh toán bằng thẻ tín dụng thành công thì hàm sẽ tạo ra một **createBooking** trong lớp BookingBUS và thông báo tiền thừa. Còn không sẽ hiển thị thông báo lỗi.

Code :

 private void CountdownTimer\_Tick(object sender, EventArgs e)

        {

            // Giảm giá trị đếm ngược

            remainingTime = remainingTime.Subtract(TimeSpan.FromSeconds(1));

            if (remainingTime.TotalSeconds >= 0)

            {

                UpdateCountdownLabel();

            }

            else

            {

                // Dừng đếm ngược

                countdownTimer.Stop();

                // Hiển thị thông báo khi đếm ngược đạt đến 0

                MessageBox.Show("Thanh toán không thành công!");

            }

        }

4.40.1.3 Hàm **btn\_Confirm\_Click – Form : MomoPayment**

Khi thanh toán bằng thẻ tín dụng thành công thì hàm sẽ tạo ra một **createBooking** trong lớp BookingBUS và thông báo tiền thừa. Còn không sẽ hiển thị thông báo lỗi.

Code :

private void btn\_Confirm\_Click(object sender, EventArgs e)

        {

            BookingBUS.Instance.createBooking();

            this.\_PaymentMethod.Close();

            this.\_PaymentMethod.\_ticketInfor.Close();

            this.Close();

        }

4.40.2 Lớp BUS

4.40.2.1 Hàm **createBooking**

Hàm sẽ cập nhật thông tin vé bằng dữ liệu lấy từ **createBooking** của lớp BookingDAO, hàm **getByID** của lớp ScheduleDAO và session. Sau đó cập nhật thông tin vé và thêm vé vào cơ sở dữ liệu. Thêm các thông tin vào FoodBooking và ProductBooking vào cơ sở dữ liệu thông qua hàm **addFoodBooking**, **addProductBooking.** Cuối cùng thêm thông tin hóa đơn vào File Excel.

Code :

   public void createBooking()

        {

            int bookingID = DAO.BookingDAO.Instance.createBooking(session.id, session.promotionID);

            Schedule schedule = DAO.ScheduleDAO.Instance.getByID(session.scheduleID.ToString());

            // Update ticket

            foreach (Seat seat in session.seats)

            {

                float price = schedule.price;

                if (seat.Type == "Couple")

                {

                    price += price;

                }

                Ticket ticket = new Ticket("0", session.scheduleID.ToString(),bookingID.ToString(), seat.ID.ToString() , price.ToString());

                if (!TicketDAO.Instance.Add(ticket))

                {

                    MessageBox.Show("Có lỗi xảy ra khi thêm vé");

                }

            }

            // Insert FOOD\_BOOKING

            foreach (Items combo in session.foodCombos)

            {

                BookingDAO.Instance.addFoodBooking(combo.Id, bookingID, int.Parse(combo.Name));

            }

            foreach (Items product in session.products)

            {

                BookingDAO.Instance.addProductBooking(product.Id, bookingID, int.Parse(product.Name));

            }

            bool excel = ExcelHelper.Instance.addBillToExcel(session.id.ToString(), session.name, bookingID.ToString(),

            this.totalTicket, this.totalProduct, DateTime.Now, session.discount);

            if (excel)

            {

                MessageBox.Show("Đã xuất hóa đơn ra excel!");

            }

            else

            {

                MessageBox.Show("Có lỗi xảy ra khi xuất hóa đơn!");

            }

            session.promotionID = -1;

            session.products = null;

            session.foodCombos = null;

            session.movieID = -1;

            session.seats = null;

            session.scheduleID = -1;

            session.startTime = null;

            session.movieTitle = null;

        }

4.40.3 Lớp DTO

4.40.4 Lớp DAO

4.40.4.1 Hàm **addFoodBooking : BookingDAO**

Thêm mới thông tin vào bảng FOOD\_BOOKING bằng cách gửi câu lệnh vào hàm **ExecuteNonQuery** vào cơ sở dữ liệu .

Code :

        public int addFoodBooking(int foodID, int bookingID, int quantity)

        {

            string query = "insert into FOOD\_BOOKING VALUES ( @foodID , @bookingID , @quantity )";

            return DataProvider.Instance.ExecuteNonQuery(query, new object[] { foodID, bookingID,   quantity });

4.40.4.2 Hàm **addProductBooking : BookingDAO**

Thêm mới thông tin vào bảng PRODUCT\_BOOKING bằng cách gửi câu lệnh vào hàm **ExecuteNonQuery** vào cơ sở dữ liệu .

Code :

        public int addProductBooking(int productID, int bookingID, int quantity)

        {

            string query = "insert into PRODUCT\_BOOKING VALUES ( @productID , @bookingID , @quantity )";

            return DataProvider.Instance.ExecuteNonQuery(query, new object[] { productID, bookingID, quantity });

        }

4.40.4.3 Hàm **Add : TicketDAO**

Thêm vé mới vào cơ sở dữ liệu bằng cách gửi câu lệnh vào hàm **ExecuteNonQuery** của lớp DataProdiver.

Code :

        public bool Add(Ticket ticket)

        {

            string sql = "INSERT INTO [dbo].[TICKET]([BOOKING\_ID],[PRICE], [SCHEDULE\_ID],[SEAT\_ID])VALUES( @booking\_id , @price , @schedule\_id , @seat\_id  )";

            int i = DataProvider.Instance.ExecuteNonQuery(sql, new object[] { ticket.BOO\_ID, ticket.PRICE, ticket.SCH\_ID, ticket.SEA\_ID });

            if (i == 0)

            {

                return false;

            }

            return true;

        }

4.41 Các hàm liên quan đến xử lí File

4.41.1 Lớp DAO

4.41.1.1 Hàm **SendPasswordResetEmail**

Hàm hỗ trỡ gửi Email cho người dùng gửi lại mật khẩu

Code :

  public void SendPasswordResetEmail(string recipientEmail, string newPassword)

            {

                try

                {

                    // Thiết lập thông tin cấu hình WebMail

                    var smtpClient = new SmtpClient();

                    smtpClient.Host = "smtp.gmail.com";

                    smtpClient.Port = 587;

                    smtpClient.EnableSsl = true;

                    // Thiết lập thông tin xác thực

                    smtpClient.Credentials = new NetworkCredential(this.email, this.pass);

                    // Tạo đối tượng MailMessage để tạo email

                    var mailMessage = new MailMessage();

                    mailMessage.From = new MailAddress(this.email);

                    mailMessage.To.Add(recipientEmail); // Địa chỉ email của người nhận

                    mailMessage.Subject = "IIEX Cinema - Password Reset"; // Chủ đề của email

                    mailMessage.Body = $"Mật khẩu của bạn là: {newPassword}\n"; // Nội dung email

                    // Gửi email

                    smtpClient.Send(mailMessage);

                    MessageBox.Show("Đã gửi mật khẩu mới vào email!");

                }

                catch (Exception ex)

                {

                    MessageBox.Show(ex.Message);

                    throw;

                }

        }

4.41.1.2 Hàm **addRevenueToExcel**

Hàm này dùng để thêm dữ liệu doanh thu vào File Excel

Code :

public bool addRevenueToExcel(string staffName, string total\_price, string cash, DateTime

created\_at, string status)

        {

            try

            {

                var outPutDirectory = Path.GetDirectoryName(Assembly.GetExecutingAssembly().CodeBase);

                int index = outPutDirectory.IndexOf("GUI");

                outPutDirectory = outPutDirectory.Substring(0, index);

                var filePath = Path.Combine(outPutDirectory, "GUI//Excel//Statitics.xlsx");

                string absoluteFilePath = new Uri(filePath).LocalPath;

                Workbook workbook = new Workbook();

                workbook.LoadFromFile(absoluteFilePath);

                Worksheet worksheet = workbook.Worksheets[0];

                int lastRow = worksheet.LastRow + 1;

                worksheet.InsertRow(lastRow);

                worksheet.Range[lastRow, 1].Value = staffName;

                worksheet.Range[lastRow, 2].Value = total\_price;

                worksheet.Range[lastRow, 3].Value = created\_at.ToString("HH:mm dd/MM/yy");

                worksheet.Range[lastRow, 4].Value = cash;

                worksheet.Range[lastRow, 5].Value = status;

                workbook.Save();

                return true;

            }

            catch (Exception ex) { return false; }

        }

4.41.1.3 Hàm **addBillToExcel**

Hàm dùng để thêm hóa đơn vào File Excel

Code :

 public bool addBillToExcel(string staffID,string staffName, string bookingID, float ticketPrice,float

productPrice, DateTime created\_at, float discout)

        {

            try

            {

                var outPutDirectory = Path.GetDirectoryName(Assembly.GetExecutingAssembly().CodeBase);

                int index = outPutDirectory.IndexOf("GUI");

                outPutDirectory = outPutDirectory.Substring(0, index);

                var filePath = Path.Combine(outPutDirectory, "GUI//Excel//Bill.xlsx");

                string absoluteFilePath = new Uri(filePath).LocalPath;

                Workbook workbook = new Workbook();

                workbook.LoadFromFile(absoluteFilePath);

                Worksheet worksheet = workbook.Worksheets[0];

                int lastRow = worksheet.LastRow + 1;

                worksheet.InsertRow(lastRow);

                worksheet.Range[lastRow, 1].Value = staffID;

                worksheet.Range[lastRow, 2].Value = staffName;

                worksheet.Range[lastRow, 3].Value = bookingID;

                worksheet.Range[lastRow, 4].Value = session.scheduleID.ToString();

                worksheet.Range[lastRow, 5].Value = session.bShowroom;

                worksheet.Range[lastRow, 6].Value = session.bMovie;

                worksheet.Range[lastRow, 7].Value = session.startTime;

                worksheet.Range[lastRow, 8].Value = session.bSeats;

                worksheet.Range[lastRow, 9].Value = session.bFoods;

                worksheet.Range[lastRow, 10].Value = created\_at.ToString("HH:mm dd/MM/yy");

                worksheet.Range[lastRow, 11].Value = ticketPrice.ToString();

                worksheet.Range[lastRow, 12].Value = productPrice.ToString();

                worksheet.Range[lastRow, 13].Value = discout.ToString();

                float total = (productPrice+ticketPrice)\*discout/100;

                worksheet.Range[lastRow, 14].Value = total.ToString();

                workbook.Save();

                return true;

            }

            catch (Exception ex) { return false; }

        }

4.41.1.4 Hàm **uploadToGUI**

Hàm dùng để load hình ảnh lên Form

Code :

 public string[] uploadToGUI()

        {

            OpenFileDialog fdlg = new OpenFileDialog();

            fdlg.Title = "C# Corner Open File Dialog";

            fdlg.InitialDirectory = @"c:\";

            fdlg.Filter = "All files (\*.\*)|\*.\*|All files (\*.\*)|\*.\*";

            fdlg.FilterIndex = 2;

            fdlg.RestoreDirectory = true;

            if (fdlg.ShowDialog() == DialogResult.OK)

            {

                // Lấy đường dẫn tới thư mục GUI/Image trong dự án

                var outPutDirectory = Path.GetDirectoryName(Assembly.GetExecutingAssembly().CodeBase);

                int index = outPutDirectory.IndexOf("GUI");

                outPutDirectory = outPutDirectory.Substring(0, index);

                var iconPath = Path.Combine(outPutDirectory, "GUI//img//");

                string imageDirectory = new Uri(iconPath).LocalPath;

                // Kiểm tra nếu thư mục GUI/Image không tồn tại, tạo mới

                if (!Directory.Exists(imageDirectory))

                {

                    Directory.CreateDirectory(imageDirectory);

                }

                // Tạo tên duy nhất cho tệp mới

                string uniqueFileName = Path.GetRandomFileName();

                string newFilePath = Path.Combine(imageDirectory, uniqueFileName);

                // Sao chép tệp đến thư mục GUI/Image

                File.Copy(fdlg.FileName, newFilePath, true);

                // Lấy phần tên tệp và gán vào trường txt\_Poster

                string fileName = Path.GetFileName(newFilePath);

                string[] res = {fileName, fdlg.FileName };

                return res;

            }

            return null;

        }

4.42 Các hàm kết nối và thực thi SQL

4.42.1 Lớp DAO

4.42.1.1 Hàm **ExecuteQuery – DataProvider**

Hàm sẽ thực thi câu lệnh truy vấn được truyền vào và kết quả trả về dạng DataTable .

Code :

public DataTable ExecuteQuery(string query, object[] parameter = null)

        {

            DataTable data = new DataTable();

            using (SqlConnection connection = new SqlConnection(connectionSTR))

            {

                connection.Open();

                SqlCommand command = new SqlCommand(query, connection);

                if (parameter != null)

                {

                    string[] listPara = query.Split(' ');

                    int i = 0;

                    foreach (string item in listPara)

                    {

                        if (item.Contains('@'))

                        {

                            command.Parameters.AddWithValue(item, parameter[i]);

                            i++;

                        }

                    }

                }

                SqlDataAdapter adapter = new SqlDataAdapter(command);

                adapter.Fill(data);

                connection.Close();

            }

            return data;

        }

4.42.1.2 Hàm **ExecuteNonQuery – DataProvider**

Hàm sẽ thực thi câu lệnh SQL được truyền vào và kết quả trả về số dòng thay đổi trong bảng trên cơ sở dữ liệu.

Code :

 public int ExecuteNonQuery(string query, object[] parameter = null)

        {

            int data = 0;

            using (SqlConnection connection = new SqlConnection(connectionSTR))

            {

                connection.Open();

                SqlCommand command = new SqlCommand(query, connection);

                if (parameter != null)

                {

                    string[] listPara = query.Split(' ');

                    int i = 0;

                    foreach (string item in listPara)

                    {

                        if (item.Contains('@'))

                        {

                            command.Parameters.AddWithValue(item, parameter[i]);

                            i++;

                        }

                    }

                }

                try

                {

                    data = command.ExecuteNonQuery();

                }

                catch (SqlException e)

                {

                    data= 0;

                }

                connection.Close();

            }

            return data;

        }

4.42.1.2 Hàm **ExecuteScalar - DataProvider**

Hàm sẽ thực thi câu lệnh truy vấn được truyền vào và kết quả trả về cột đầu tiên của hàng đầu tiên trong tập kết quả.

Code :

  public object ExecuteScalar(string query, object[] parameter = null)

        {

            object data = 0;

            using (SqlConnection connection = new SqlConnection(connectionSTR))

            {

                connection.Open();

                SqlCommand command = new SqlCommand(query, connection);

                if (parameter != null)

                {

                    string[] listPara = query.Split(' ');

                    int i = 0;

                    foreach (string item in listPara)

                    {

                        if (item.Contains('@'))

                        {

                            command.Parameters.AddWithValue(item, parameter[i]);

                            i++;

                        }

                    }

                }

                try

                {

                    data = command.ExecuteScalar();

                }

                catch (Exception)

                {

                    data = 0;

                    throw;

                }

                connection.Close();

            }

            return data;

        }

4.43 Các lớp hỗ trợ cho việc

4.43.1 Movie DTO

MovieDTO bao gồm các thuộc tính *string id, string title, string genre, int duration, float rating, string story, string poster .*

Code :

 public class Movie

    {

        private string id;

        private string title;

        private string genre;

        private int duration;

        private float rating;

        private string story;

        private string poster;

        public Movie(string id, string title, string genre, int duration, float rating, string story, string poster)

        {

            this.id = id;

            this.title = title;

            this.genre = genre;

            this.duration = duration;

            this.rating = rating;

            this.story = story;

            this.poster = poster;

        }

        public Movie(string title, string genre, int duration, float rating, string story, string poster)

        {

            this.title = title;

            this.genre = genre;

            this.duration = duration;

            this.rating = rating;

            this.story = story;

            this.poster = poster;

        }

        public string ID

        {

            get { return id; }

            set { id = value; }

        }

        public string Title

        {

            get { return title; }

            set { title = value; }

        }

        public string Genre

        {

            get { return genre; }

            set { genre = value; }

        }

        public int Duration

        {

            get { return duration; }

            set { duration = value; }

        }

        public float Rating

        {

            get { return rating; }

            set { rating = value; }

        }

        public string Story

        {

            get { return story; }

            set { story = value; }

        }

        public string Poster

        {

            get { return poster; }

            set { poster = value; }

        }

    }

4.43.2 FoodCombo DTO

FoodCombo bao gồm các thuộc tín *string id, string name, string food, int foodQuantity, string drink, int drinkQuantity, float price, string image.*

Code :

    public class FoodCombo

    {

        public FoodCombo(string id, string name, string food, int foodQuantity

                        , string drink, int drinkQuantity, float price, string image)

        {

            this.id = id;

            this.name = name;

            this.food = food;

            this.foodQuantity = foodQuantity;

            this.drink = drink;

            this.drinkQuantity = drinkQuantity;

            this.price = price;

            this.image = image;

        }

        public string id { get; set; }

        public string name { get; set; }

        public string food { get; set; }

        public int foodQuantity { get; set; }

        public string drink { get; set; }

        public int drinkQuantity { get; set; }

        public float price { get; set; }

        public string image { get; set; }

    }

}

4.43.3 Items DTO

Items bao gồm các thuộc tín *int id, string name*

Code :

    public class Items

    {

        public Items(int id, string name)

        {

            Id = id;

            Name = name;

        }

        public int Id { get; set; }

        public string Name { get; set; }

    }

4.43.4 Product DTO

Product bao gồm các thuộc tính *string id, string name, float price, int quantity, string type,DateTime? expiry\_date, int lastOrder.*

Code :

    public class Product

    {

        private string id;

        private string name;

        private float price;

        private int quantity;

        private string type;

        private int lastOrder;

        private DateTime? expiry\_date;

        //private DateTime expiration;

        public string ID

        {

            get { return id; }

            set { id = value; }

        }

        public string Name

        {

            get { return name; }

            set { name = value; }

        }

        public string Type

        { get { return type; } set { type = value; } }

        public float Price

        { get { return price; } set { price = value; } }

        public int Quantity

        { get { return quantity; } set { quantity = value; } }

        public DateTime? Expiry\_date

        {

            get { return expiry\_date; }

            set { expiry\_date = value; }

        }

        public int LastOrder { get { return lastOrder; } set { lastOrder = value; } }

        public Product(string id, string name, float price, int quantity, string type,DateTime? expiry\_date, int lastOrder)

        {

            this.id = id;

            this.name = name;

            this.price = price;

            this.quantity = quantity;

            this.type = type;

            this.expiry\_date = expiry\_date;

            this.lastOrder = lastOrder;

        }

        public Product(string name, float price, int quantity, string type, DateTime? expiry\_date, int lastOrder)

        {

            this.name = name;

            this.price = price;

            this.quantity = quantity;

            this.type = type;

            LastOrder = lastOrder;

            this.expiry\_date = expiry\_date;

        }

    }

4.43.4 Promotion DTO

Promotion bao gồm các thuộc tính *int ID, string Code, float Discount, string Status, string Note..*

Code :

    public class Promotion

    {

        public int ID { get; set; }

        public string Code { get; set; }

        public float Discount { get; set; }

        public string Status { get; set; }

        public string Note { get; set; }

    }

4.43.5 Revenue DTO

Revenue bao gồm các thuộc tính *string id, int staff\_id, string staffName, float total\_price, float cash, DateTime created\_at, string status, string note .*

Code :

public class Revenue

    {

        private string id;

        private int staff\_id;

        private string staffName;

        private float total\_price;

        private float cash;

        private DateTime created\_at;

        private string status;

        private string note;

        public Revenue()

        {

        }

        public Revenue(string id, int staff\_id, string staffName, float total\_price, float cash, DateTime created\_at, string status, string note)

        {

            this.id = id;

            this.staff\_id = staff\_id;

            this.staffName = staffName;

            this.total\_price = total\_price;

            this.cash = cash;

            this.created\_at = created\_at;

            this.status = status;

            this.note = note;

        }

        public Revenue(DateTime created\_at, float total\_price)

        {

            this.created\_at = created\_at;

            this.total\_price = total\_price;

        }

        public Revenue(int staff\_id, string staffName, float total\_price, float cash, DateTime created\_at, string status, string note)

        {

            this.staff\_id = staff\_id;

            this.staffName = staffName;

            this.total\_price = total\_price;

            this.cash = cash;

            this.created\_at = created\_at;

            this.status = status;

            this.note = note;

        }

        public string ID

        {

            get { return id; }

            set { id = value; }

        }

        public int Staff\_id

        {

            get { return staff\_id; }

            set { staff\_id = value; }

        }

        public string StaffName

        {

            get { return staffName; }

            set { staffName = value; }

        }

        public float TotalPrice

        {

            get { return total\_price; }

            set { total\_price = value; }

        }

        public float Cash

        {

            get { return cash; }

            set { cash = value; }

        }

        public DateTime CreatedAt

        {

            get { return created\_at; }

            set { created\_at = value; }

        }

        public string Status

        {

            get { return status; }

            set { status = value; }

        }

        public string Note

        {

            get { return note; }

            set { note = value; }

        }

    }

4.43.6 Role DTO

Role bao gồm các thuộc tín *string id, string name.*

Code :

    public class Role

    {

        private string name;

        private string id;

        public string ID

        {

            get { return id; }

            set { id = value; }

        }

        public string NAME

        {

            get { return name; }

            set { name = value; }

        }

        public Role() { }

        public Role(string id, string name)

        {

            this.name = name;

            this.id = id;

        }

    }

4.43.6 Revenue DTO

Revenue bao gồm các thuộc tính string id, int theater\_id, int movie\_id, DateTime? starttime, DateTime? endtime, float price

Code :

  public class Schedule

    {

        public float price { get; set; }

        private int id;

        public int ID

        {

            get { return id; }

            set { id = value; }

        }

        private int theaterID;

        public int TheaterID

        {

            get { return theaterID; }

            set { theaterID = value; }

        }

        private int movieID;

        public int MovieID

        {

            get { return movieID; }

            set { movieID = value; }

        }

        private DateTime? startTime;

        public DateTime? StartTime

        {

            get { return startTime; }

            set { startTime = value; }

        }

        private DateTime? endTime;

        public Schedule(string id1, int theater\_id, int movie\_id, DateTime? starttime, DateTime? endtime, float price)

        {

            Id = id1;

            Theater\_id = theater\_id;

            Movie\_id = movie\_id;

            startTime = starttime;

            endTime = endtime;

            this.price = price;

        }

        public DateTime? EndTime

        {

            get { return endTime; }

            set { endTime = value; }

        }

        public string Id { get; }

        public int Theater\_id { get; }

        public int Movie\_id { get; }

    }

4.43.7 Seat DTO

Seat bao gồm các thuộc tính int ID, string SeatNumber, bool IsBooked ,string Type.

Code :

    public class Seat

    {

        public int ID { get; set; }

        public string SeatNumber { get; set; }

        public bool IsBooked { get; set; }

        public string Type { get; set; }

    }

4.43.8 Session

Session bao gồm các thuộc tính *int ID,string name, int role, int movieID, movieTitile, int scheduleID, int showRoomID, string startTime, List<Seat> seats , List<Items> foodCombos , List<Items> products , int promotionID , float discount , string lang , string bSeats,string bFoods,string bShowroom,string bMovie.*

Code :

    public class session

    {

        public static int id { get; set; }

        public static string name { get; set; }

        public static int role { get; set; }

        // Booking

        public static int movieID { get; set; }

        public static string movieTitle { get; set; }

        public static int scheduleID { get; set; }

        public static int showRoomID { get; set; }

        public static string startTime { get; set; }

        public static List<Seat> seats { get; set; }

        public static List<Items> foodCombos { get;set; }

        public static List<Items> products { get;set; }

        public static int promotionID { get; set; }

        public static float discount { get; set; }

        public static string lang { get; set; }

        //Bill

        public static string bSeats { get; set; }

        public static string bFoods { get; set; }

        public static string bShowroom { get; set; }

        public static string bMovie { get; set; }

    }

4.43.9 Staff DTO

Staff bao gồm các thuộc tính int id, string username, string fname, string lname, string password, string sex, DateTime? birthday, string address, int role, string phone, float salary

Code :

 public class Staff

    {

        //private string id;

        private int id;

        private string username;

        private string password;

        private string fname;

        private string lname;

        private string sex;

        private DateTime? birthday;

        private string address;

        private int role;

        private string phone;

        private float salary;

        public Staff(int id, string username, string fname, string lname, string password, string sex, DateTime? birthday, string address, int role, string phone, float salary)

        {

            this.id = id;

            this.username = username;

            this.fname = fname;

            this.lname = lname;

            this.password = password;

            this.sex = sex;

            this.birthday = birthday;

            this.address = address;

            this.role = role;

            this.phone = phone;

            this.salary = salary;

        }

        public Staff(string username, string fname, string lname, string password, string gender, DateTime? birthday, string address, int role, string phone, float salary)

        {

            this.username = username;

            this.fname = fname;

            this.lname = lname;

            this.password = password;

            this.sex = gender;

            this.birthday = birthday;

            this.address = address;

            this.role = role;

            this.phone = phone;

            this.salary = salary;

        }

        public int ID

        {

            get { return id; }

            set { id = value; }

        }

        public string UserName

        {

            get { return username; }

            set { username = value; }

        }

        public string Password

        {

            get { return password; }

            set { password = value; }

        }

        public string Fname

        {

            get { return fname; }

            set { fname = value; }

        }

        public string Lname

        {

            get { return lname; }

            set { lname = value; }

        }

        public string Sex

        {

            get { return sex; }

            set { sex = value; }

        }

        public DateTime? Birthday

        {

            get { return birthday; }

            set { birthday = value; }

        }

        public string City

        {

            get { return address; }

            set { address = value; }

        }

        public int Role

        {

            get { return role; }

            set { role = value; }

        }

        public string Phone

        {

            get { return phone; }

            set { phone = value; }

        }

        public float Salary

        {

            get { return salary; }

            set { salary = value;}

        }

    }

4.43.10 Ticket DTO

Ticket bao gồm các thuộc tính string id, string sch\_id, string boo\_id, string sea\_id, string price

Code :

public class Ticket

    {

        private string id;

        private string sch\_id;

        private string boo\_id;

        private string sea\_id;

        private string price;

        public Ticket(string id, string sch\_id, string boo\_id, string sea\_id, string price)

        {

            this.id = id;

            this.sch\_id = sch\_id;

            this.sea\_id = sea\_id;

            this.price = price;

            this.boo\_id = boo\_id;

        }

        public Ticket(string sch\_id, string boo\_id, string sea\_id, string price)

        {

            this.sch\_id = sch\_id;

            this.sea\_id = sea\_id;

            this.price = price;

            this.boo\_id = boo\_id;

        }

        public string ID

        {

            get { return id; }

            set { id = value; }

        }

        public string SCH\_ID

        {

            get { return sch\_id; }

            set { sch\_id = value; }

        }

        public string BOO\_ID

        {

            get { return boo\_id; }

            set { boo\_id = value; }

        }

        public string SEA\_ID

        {

            get { return sea\_id; }

            set { sea\_id = value; }

        }

        public string PRICE

        {

            get { return price; }

            set { price = value; }

        }

    }

4.43.11 User DTO

User bao gồm các thuộc tính string id, string username, string email, string fname, string lname, string password, string gender, DateTime? birthday, string city.

Code :

  public class User

    {

        private string id;

        private string username;

        private string email;

        private string fname;

        private string lname;

        private string password;

        private string gender;

        private DateTime? birthday;

        private string city;

        public User(string id, string username, string email, string fname, string lname, string password, string gender, DateTime? birthday, string city)

        {

            this.id = id;

            this.username = username;

            this.email = email;

            this.fname = fname;

            this.lname = lname;

            this.password = password;

            this.gender = gender;

            this.birthday = birthday;

            this.city = city;

        }

        public User(string username, string email, string fname, string lname, string password, string gender, DateTime? birthday, string city)

        {

            this.username = username;

            this.email = email;

            this.fname = fname;

            this.lname = lname;

            this.password = password;

            this.gender = gender;

            this.birthday = birthday;

            this.city = city;

        }

        public string UserId

        {

            get { return id; }

            set { id = value; }

        }

        public string UserName

        {

            get { return username; }

            set { username = value; }

        }

        public string Email

        {

            get { return email; }

            set { email = value; }

        }

        public string Password

        {

            get { return password; }

            set { password = value; }

        }

        public string Fname

        {

            get { return fname; }

            set { fname = value; }

        }

        public string Lname

        {

            get { return lname; }

            set { lname = value; }

        }

        public string Gender

        {

            get { return gender; }

            set { gender = value; }

        }

        public DateTime? Birthday

        {

            get { return birthday; }

            set { birthday = value; }

        }

        public string City

        {

            get { return city; }

            set { city = value; }

        }

    }