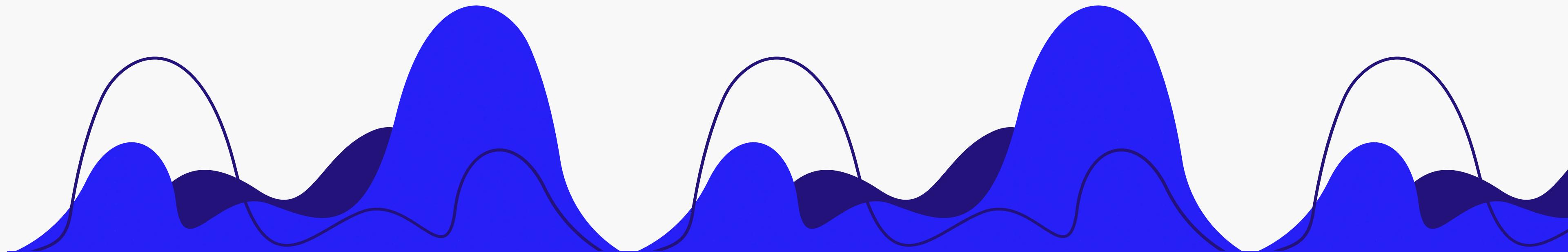
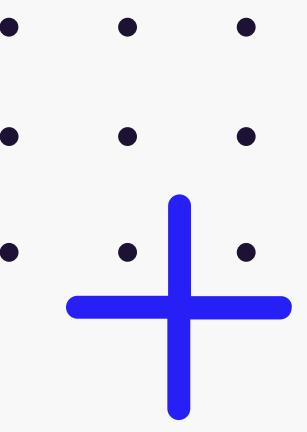


APPLICATION DE GESTION DES RÉSERVATIONS DANS UN HÔTEL



Réalisé par :

ELHARCHI MOHAMMED AMINE

AITALI YASSIR

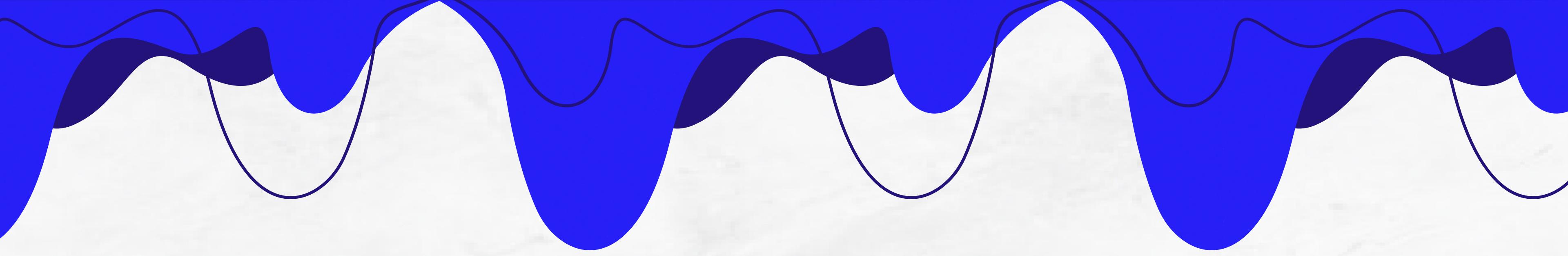
ELHADEG MEHDI

INTRODUCTION

+ Notre application est conçue pour gérer les informations liées aux hébergements hôteliers, notamment les détails sur les chambres, les clients et les réservations.

+ Elle permet aux clients d'effectuer des réservations, de vérifier la disponibilité des chambres, d'enregistrer les informations des clients et de conserver un historique des réservations.

- • •
- • •
- • •
- • •
- • •



SOMMAIRE

01

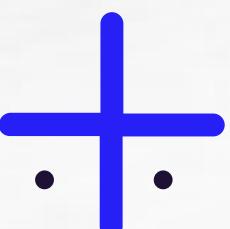
Les fonctionnalités
du projet

02

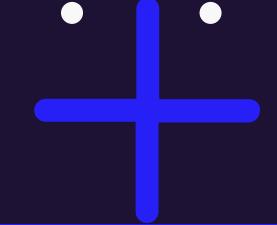
Les outils utilisés

03

La conception

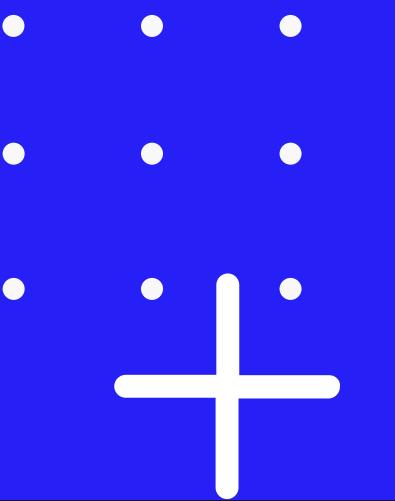


FONCTIONNALITÉS DE L'APPLICATION



- Notre application est conçue pour gérer les informations liées aux hébergements hôteliers.
- Elle englobe des détails sur les chambres, les clients et les réservations.
- Les composants clés comprennent :
 - Chambre (Room) : Cette section fournit des informations sur les chambres disponibles, notamment le numéro de chambre, le type de chambre, la capacité, les tarifs et d'autres détails pertinents.
 - Client (Client) : Cette section gère les informations clients, telles que les noms, coordonnées et autres données pertinentes.
 - Réservation (Reservation) : Cette partie contient des détails sur les réservations, comprenant les dates d'arrivée et de départ, les clients associés, les types de chambres réservées...

LES OUTILS UTILISÉS



C++



QT CREATOR 5.6.1



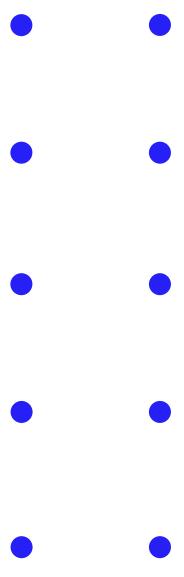
MYSQL

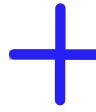


QT CREATOR

Un environnement de développement intégré (IDE); Qt Creator offre des outils pour la conception, la programmation, les tests et le débogage d'applications, en particulier celles construites avec la bibliothèque Qt, qui est une bibliothèque C++ populaire pour la création d'applications multiplateformes.

Il comprend des fonctionnalités telles qu'un éditeur de code, un concepteur visuel, une gestion de projet et des outils de débogage pour simplifier le processus de développement.

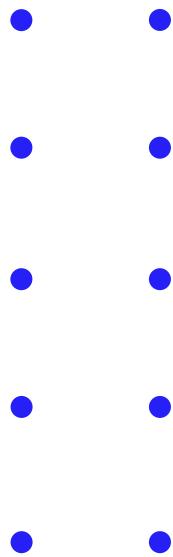




MySQL

MySQL est un système de gestion de base de données relationnelles open source.

Il utilise le langage SQL pour stocker, récupérer et gérer les données. Souvent utilisé dans le développement d'applications web, MySQL est reconnu pour sa fiabilité et ses performances.



LA CONCEPTION



Structure de la base de donnée :

- **guests (id, room_no, name, email, contact, address, checkin, checkout, identity, room_type, total_amount, paid_amount, due_amount, status, packages)**
- **packages (name, details, price, company, available, db)**
- **room (room_no, type, details, price, available, no_of_beds, wifi, tv, couple_friendly, db)**
- **users (id, email, password)**

Structure de la base de donnée

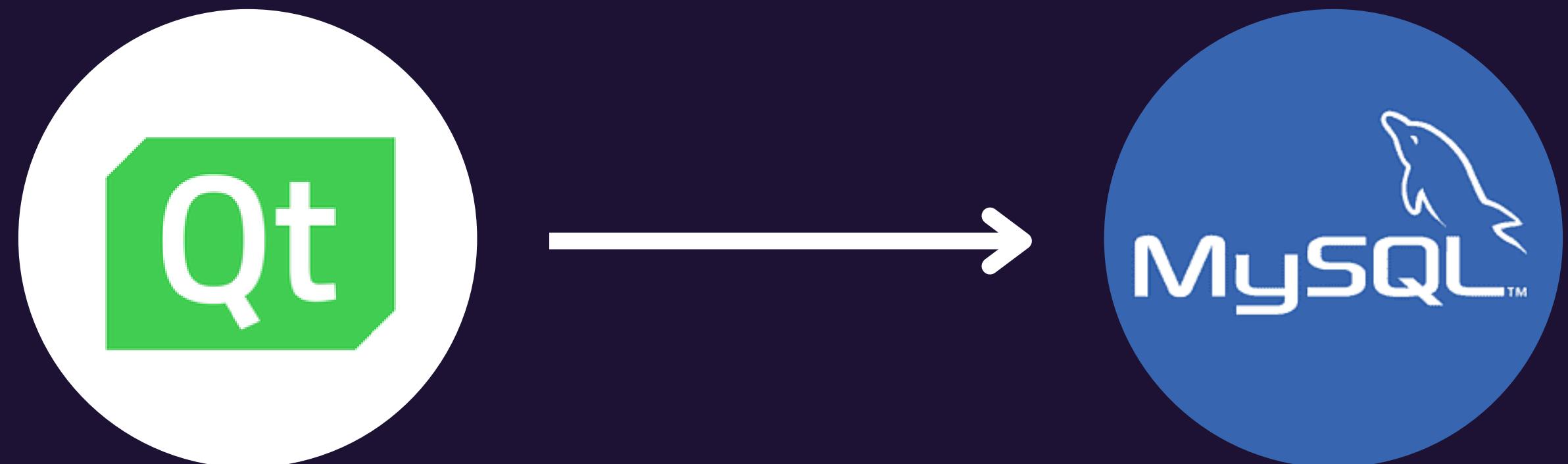
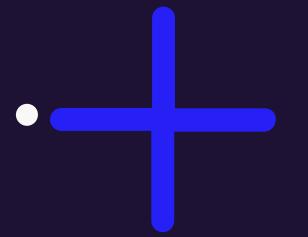
| v | o | hms users |
|---|-------------------------|-----------|
| g | id : int(11) | |
| o | email : varchar(255) | |
| o | password : varchar(255) | |

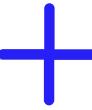
| v | o | hms packages |
|---|------------------------|--------------|
| o | name : varchar(255) | |
| o | details : varchar(255) | |
| # | price : int(11) | |
| o | company : varchar(255) | |
| # | available : tinyint(1) | |
| o | db : varchar(255) | |

| v | o | hms guests |
|---|--------------------------|------------|
| g | id : int(11) | |
| # | room_no : int(11) | |
| o | name : varchar(255) | |
| o | email : varchar(255) | |
| o | contact : varchar(255) | |
| o | address : varchar(255) | |
| o | checkin : varchar(255) | |
| o | checkout : varchar(255) | |
| o | identity : varchar(255) | |
| o | room_type : varchar(255) | |
| # | total_amount : int(11) | |
| # | paid_amount : int(11) | |
| # | due_amount : int(11) | |
| o | status : varchar(255) | |
| o | packages : varchar(255) | |

| v | o | hms room |
|---|------------------------------|----------|
| g | room_no : int(11) | |
| o | type : varchar(255) | |
| o | details : varchar(255) | |
| # | price : double | |
| # | available : tinyint(1) | |
| # | no_of_beds : int(11) | |
| # | wifi : tinyint(1) | |
| # | tv : tinyint(1) | |
| # | couple_friendly : tinyint(1) | |
| o | db : varchar(255) | |

CONNEXION AU S.G.B.D





CONNEXION AU S.G.B.D

- **Déclaration de la connexion**

```
#include <QSqlDatabase>
#include <QSqlQuery>
```

- **Etablissement de la connexion**

```
bool Database::connectDB(){
    db = QSqlDatabase::addDatabase("QMYSQL"); // Initializing Database,
    db.setHostName("localhost"); // Setting hostname
    db.setPort(3306); // setting port
    db.setDatabaseName("hms"); // setting database name
    db.setUserName("root"); //setting username
    db.setPassword(""); //setting password
    if(db.open()){
        return true;
    }else{ // If database connection is not established
        return false;
    }
}
```

INTERFACE UTILISATEUR

- **Authentification des administrateurs :**
- **Dans le but de renforcer la sécurité des données, il est nécessaire de mettre en place une interface d'administration. Cette interface exigera une authentification en tant qu'administrateur avant toute possibilité d'accéder aux données ou de les manipuler.**

INTERFACE UTILISATEUR

- Authentification des administrateurs :

Hotel Management System

Login

Email

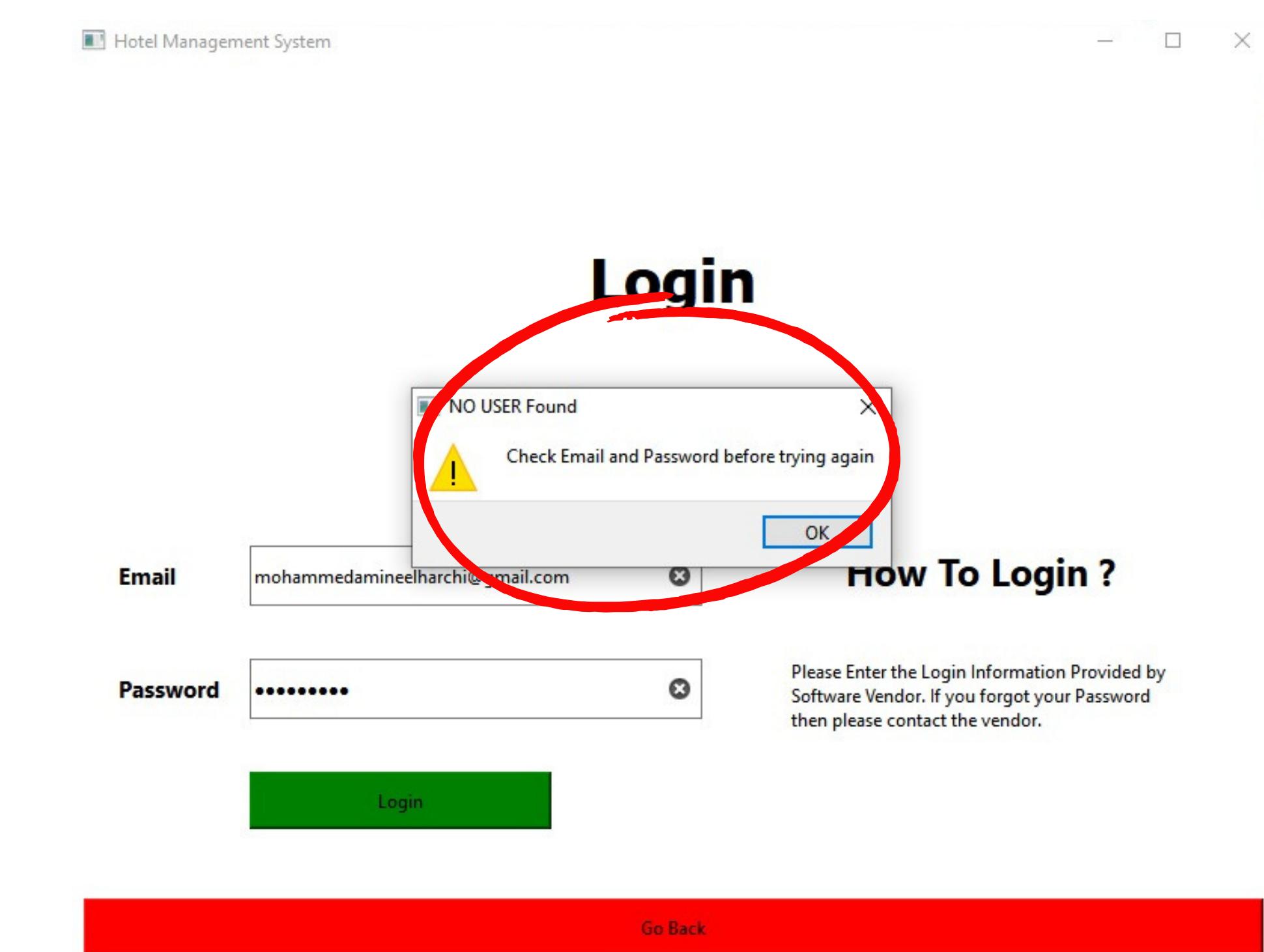
Password

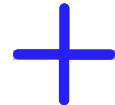
How To Login ?

Please Enter the Login Information Provided by Software Vendor. If you forgot your Password then please contact the vendor.

Login

Go Back



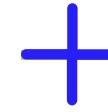


```
void MainWindow::loginButton_clicked(){
    QString email,password;
    email = MainWindow::emailText->text(); // fetch input text from email input field of loginscreen
    password = MainWindow::passwordText->text(); // fetch input text from password input field of loginscreen

    // Checks if the user with provided email and password exists in database or not.
    // Searches for record with provided email and password in users table in database.
    QSqlQuery query;
    bool user = query.exec("SELECT * FROM users WHERE email='"+email+"' AND password='"+password+"'");

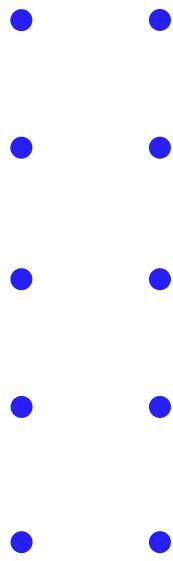
    if(user){ // condition is true if user exists
        if(query.next()){
            connect(this,SIGNAL(loggedIn()),this,SLOT(dashboard())); // Navigates to Dashboard //Receives loggedIn signal
            emit loggedIn(); // Emits loggedIn signal
        }else{
            QMessageBox::warning(window(),"NO USER Found","Check Email and Password before trying again"); // Displays No User Found message
        }
    }else{
        QMessageBox::warning(window(),"Network Error","Check Internet Connection"); // Displays Connection Error
    }
}
```

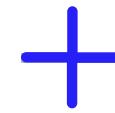
- On récupère les entrées d'email et de mot de passe, on effectue une requête SQL pour vérifier l'existence de l'utilisateur dans la base de données, et on émet un signal "loggedIn" en cas de succès, redirigeant ainsi vers le tableau de bord. En cas d'échec, on affiche des avertissements appropriés, signalant soit l'absence d'utilisateur, soit un problème de connexion réseau.



INTERFACE UTILISATEUR

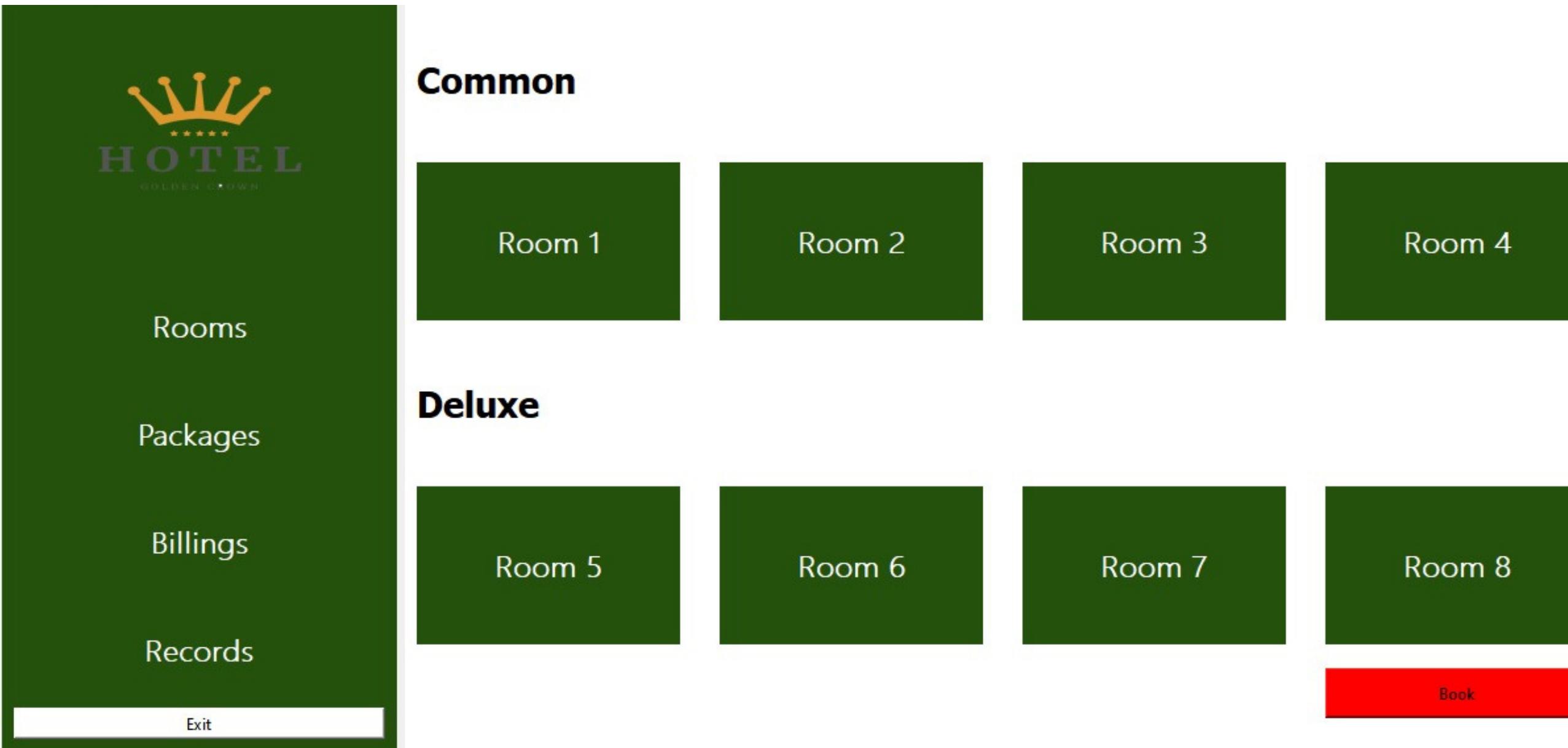
- Dashboard :





INTERFACE UTILISATEUR

- Options de gestion(affichage des chambres) :



The image shows a user interface for managing hotel rooms. On the left, a sidebar menu lists "Rooms", "Packages", "Billings", and "Records". At the bottom of the sidebar is a red "Exit" button. The main area displays two categories of rooms: "Common" and "Deluxe". Each category contains four room boxes labeled Room 1 through Room 4 for the Common category, and Room 5 through Room 8 for the Deluxe category. A red "Book" button is located at the bottom right of the room grid.

Common

Room 1 Room 2 Room 3 Room 4

Deluxe

Room 5 Room 6 Room 7 Room 8

Book

INTERFACE UTILISATEUR

- On définit une fonction **Roommain** dans la classe **MainWindow** qui configure l'affichage de la fenêtre principale pour la gestion des chambres. Cette fonction affecte un titre à la fenêtre, définit un style, crée des étiquettes pour les types de chambres (**Common** et **Deluxe**), et met en forme des boutons pour chaque chambre.
- Les boutons sont stylisés avec une feuille de style pour une apparence cohérente. En outre, un bouton "Book" est ajouté avec un style spécifique. Chaque bouton de chambre est connecté à des slots correspondants, tels que **room1()**, **room2()**, etc., pour gérer les interactions avec ces boutons.
- Finalement, la fonction configure une grille de disposition (**QGridLayout**) pour organiser les éléments graphiques, tels que les étiquettes et les boutons, sur le côté droit de la fenêtre principale.

```
void MainWindow::Roommain(){  
    setWindowTitle("Room");  
    style();  
  
    QLabel *label = new QLabel();  
    label->setText("Common");  
    QFont banner("Helvetica",20,QFont::Bold);  
    label->setFont(banner);  
    label->setFixedHeight(90);  
  
    QLabel *label2 = new QLabel();  
    label2->setText("Deluxe");  
    label2->setFont(banner);  
    label2->setFixedHeight(90);  
  
    rightside->setStyleSheet("*{background:white"  
    "};");  
    QString StyleSheetBtns = "QPushButton { color: white;  
    background-color: black; border: 1px solid black; font-size: 14px; padding: 5px; }";  
  
    QPushButton* room1Btn = new QPushButton("Room 1");  
    QPushButton* room2Btn = new QPushButton("Room 2");  
    QPushButton* room3Btn = new QPushButton("Room 3");  
    QPushButton* room4Btn = new QPushButton("Room 4");  
    QPushButton* room5Btn = new QPushButton("Room 5");  
    QPushButton* room6Btn = new QPushButton("Room 6");  
    QPushButton* room7Btn = new QPushButton("Room 7");  
    QPushButton* room8Btn = new QPushButton("Room 8");  
    ...  
}
```

INTERFACE UTILISATEUR

- Options de gestion(affichage des chambres) :

File Help contact us

Room Details

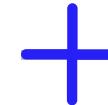
Room Number: 1

Room Type: Deluxe

Details: Spacious room with king size bed

Price: 120.5

Availability: 1



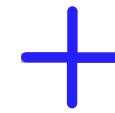
INTERFACE UTILISATEUR

- Packages :

The screenshot shows the user interface for managing hotel packages. On the left, a sidebar menu is visible with the following items: Rooms, Packages, Billings, and Records. At the bottom of the sidebar is an 'Exit' button. The main content area has a header with three buttons: 'Add Package' (white text on green background), 'Packages' (large black text on white background), and 'Remove Package' (white text on green background). Below the header, there are two sections: 'Available' and 'Not Available'. The 'Available' section contains a table with one row of data:

| ID | Name | Company | Details | |
|----|----------|---------|---------|------|
| 1 | Suitable | big | 1000 | emsi |

The 'Not Available' section is currently empty.



INTERFACE UTILISATEUR

- **On définit une fonction `PackageScreen` dans la classe `MainWindow` qui configure l'affichage de la fenêtre principale pour la gestion des packages.** Cette fonction affecte un titre à la fenêtre, définit un style, et crée des étiquettes et des boutons pour les différentes fonctionnalités liées aux packages.
- **Les boutons sont stylisés avec une feuille de style pour une apparence cohérente.** Deux étiquettes sont ajoutées pour indiquer les catégories "Available" et "Not Available". De plus, deux boutons "Add Package" et "Remove Package" sont créés pour permettre l'ajout et la suppression de packages.
- **Les étiquettes et les boutons sont configurés avec des tailles fixes et des alignements spécifiques pour une présentation visuelle optimale.**

```
void MainWindow::PackageScreen(){  
    setWindowTitle("Packages");  
    style();  
    //rightside->setStyleSheet("*{background:green;}");  
  
    QString StyleSheetBtns = "QPushButton { color: white; background-color: #4CAF50; border: none; padding: 10px; font-size: 16px; }";  
  
    QLabel *label = new QLabel();  
  
    label->setText("Packages");  
    QFont banner("Helvetica",40,QFont::Bold);  
    label->setFont(banner);  
    label->setFixedHeight(90);  
    label -> setAlignment(Qt::AlignCenter);  
    label->adjustSize();  
  
    QLabel *label1 = new QLabel();  
    label1->setText("Available");  
    QFont banner1("Helvetica",20,QFont::Bold);  
    label1->setFont(banner1);  
    label1->setFixedHeight(90);  
    label1->adjustSize();  
    label1 -> setAlignment(Qt::AlignCenter);  
  
    QLabel *label2 = new QLabel();  
    label2->setText("Not Available");  
    label2->setFont(banner1);  
    label2->setFixedHeight(90);  
    label2->adjustSize();  
    label2 -> setAlignment(Qt::AlignCenter);  
  
    QPushButton* addPackage = new QPushButton("Add Package");  
    QPushButton* removePackage = new QPushButton("Remove Package");
```



INTERFACE UTILISATEUR

- Ajout du package :

The screenshot shows a web-based application interface for adding a package. At the top, there is a navigation bar with links for 'File', 'Help', and 'contact us'. The main content area is titled 'Package Information'. On the left, there is a vertical list of form fields with corresponding input boxes:

- Available**: A checkbox input field.
- Name**: An input box.
- Company**: An input box.
- Details**: An input box.
- Price**: An input box.

At the bottom of the form area, there is a green button labeled 'Submit' and a red button labeled 'Go Back'.



INTERFACE UTILISATEUR

- La classe **Package** est créée avec des méthodes d'accès (getters) et de modification (setters) pour les propriétés d'un package, notamment **getName()**, **getDetails()**, **getPrice()**, **getCompany()**, et **isAvailable()** pour obtenir les valeurs, ainsi que **setName()**, **setDetails()**, **setCompany()**, **setPrice()**, et **setAvailable()** pour définir ou mettre à jour ces valeurs.

```
void Package::setName(QString name){  
    this->name = name;  
}  
  
void Package::setDetails(QString details){  
    this->details = details;  
}  
void Package::setCompany(QString company){  
    this->company = company;  
}  
void Package::setPrice(int price){  
    this->price = price;  
}  
void Package::setavailable(bool available){  
    this->available = available;  
}
```



INTERFACE UTILISATEUR

- Billings:

The screenshot shows the 'Billings' section of the Hotel Management System. On the left, a dark sidebar lists 'Rooms', 'Packages', 'Billings' (which is selected), and 'Records'. The main area has a header with a logo ('HOTEL GOLDEN CROWN'), a dropdown for 'Select A Room' (set to 'Room 1 (Common)'), and a message 'No user'. It includes fields for 'Name', 'Email', 'Phone No.', 'Address', and 'Nationality'. Below these are summary amounts: 'Total Amount' (0), 'Paid Amount' (0), and 'Due Amount' (0). A 'Pay:' button leads to a 'Pay Here' input field. At the bottom are 'Back', 'Pay', and 'Check out' buttons.

Billings

Select A Room

Room 1 (Common)

No user

Name

Email

Phone No.

Address

Nationality

Total Amount 0

Paid Amount 0

Due Amount 0

Pay:

Pay Here

Back Pay Check out

INTERFACE UTILISATEUR

- La fonction **Bill** de la classe **MainWindow** crée une interface de facturation avec une disposition verticale. Elle comprend une moitié supérieure avec un label "Select A Room" et une liste déroulante pour choisir une chambre. La fonction réagit à la sélection en appelant la méthode **checkout** avec le numéro de chambre correspondant. La moitié inférieure reste réservée pour d'autres éléments liés à la facturation.

```
void MainWindow::Bill(){  
    QWidget *parent = new QWidget; // parent widget  
    QVBoxLayout *main_layout = new QVBoxLayout();  
    setWindowTitle("Billings");  
    style();  
    rightside->setLayout(main_layout);  
  
    QWidget *tophalf = new QWidget;  
    // QVBoxLayout *main_layout = new QVBoxLayout();  
    QHBoxLayout *top_layout = new QHBoxLayout();  
    QLabel *selectRoomLabel = new QLabel("Select A Room");  
    QComboBox *selectRoom = new QComboBox();  
    selectRoom->addItem("Please select a Room");  
    selectRoom->addItem("Room 1 ( Common )");  
    selectRoom->addItem("Room 2 ( Common )");  
    selectRoom->addItem("Room 3 ( Common )");  
    selectRoom->addItem("Room 4 ( Common )");  
    selectRoom->addItem("Room 5 ( Deluxe )");  
    selectRoom->addItem("Room 6 ( Deluxe )");  
    selectRoom->addItem("Room 7 ( Deluxe )");  
    selectRoom->addItem("Room 8 ( Deluxe )");  
  
    top_layout->addWidget(selectRoomLabel);  
    top_layout->addWidget(selectRoom);  
    tophalf->setLayout(top_layout);  
}
```

INTERFACE UTILISATEUR

- Guests records:

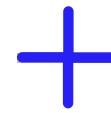
The screenshot shows a Windows application window titled "Guests Records". The window has a menu bar with "File", "Help", and "contact us". Below the menu is a search bar with a "Search" button and an "Edit" button. To the right of the search bar are buttons for "Sort By Name Ascending" and "Status Active". The main area contains a table with guest information:

| Room No | Name | Email | Contact | Address | Checkin | Checkout | Identity | Room |
|---------|-------|-----------------|------------|---------|----------------|----------|----------|------|
| 11 | kjnkj | jsjsj@gmail.com | 5555555555 | sjsjj | 2023-12-31 ... | | jsjjs | |

On the left side of the application, there is a vertical navigation bar with the following options:

- Rooms
- Packages
- Billings
- Records

At the bottom of the navigation bar is an "Exit" button.



INTERFACE UTILISATEUR

- Help/FAQ window:

The screenshot shows a software window titled "FAQ". The menu bar includes "Dashboard", "File", "Help", and "contact us". The main content area displays three questions in bold black text:

- Q1) What is this software used for?**

Ans) This hotel management software is used to keep all the recording of the datas of the costumers and enables to the smooth running of the hotel. It is basically used to digitize the record keeping of the hotel.
- Q2) How do I login to the software?**

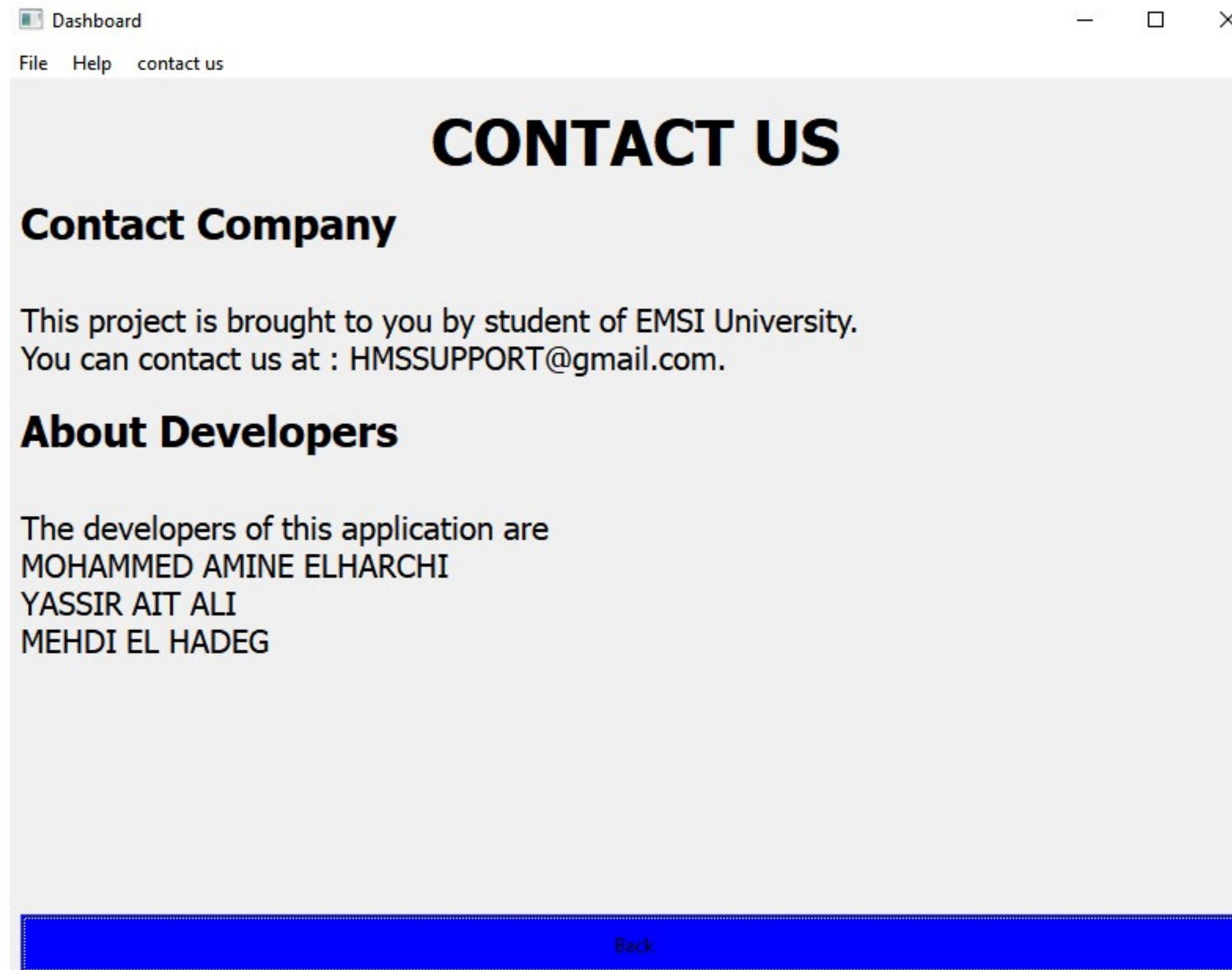
Ans) To login to the software you need to be the employee of the hotel and you need to have your login id and password provided to you by the hotel.
- Q3) How do I keep the datas of the guest?**

Ans) To input the datas of the guest, click on the rooms button and then select the preferred room by thr guest them a form pops up in the screen and then fill the information of the guest

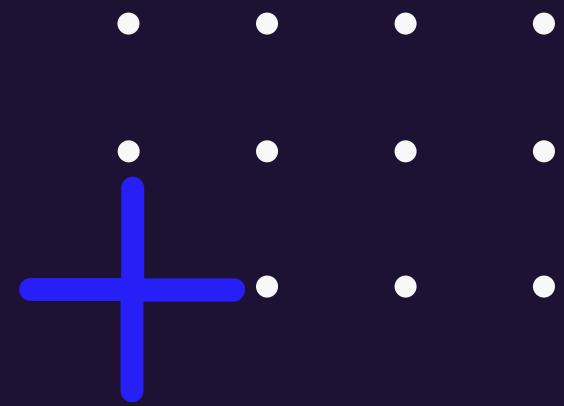
[Back](#)

INTERFACE UTILISATEUR

- Contact us window:



REMERCIEMENTS



Nous tenons à remercier sincèrement M. ABDALI Abelmounaim pour sa précieuse contribution, mettant à notre disposition les outils et la documentation indispensables qui ont grandement facilité l'achèvement de ce projet.

