HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA KĨ THUẬT ĐIỆN TỬ 1

BÁO CÁO

BÀI TẬP LỚN LẬP TRÌNH WEB

Giảng viên hướng dẫn : **Trần Đức Dương**

Đề tài : **Web quản lý đặt phòng khách sạn**

Nhóm bài tập lớn : **03**

Thành viên nhóm: : **Nguyễn Việt Hải – B20DCCN221**

Phạm Huy Hoàng – B20DCCN281

Nguyễn Tiến Hưng – B20DCCN341

I. Đặc tả chức năng

Website quản lý đặt phòng khách sạn có các chức năng chủ yếu sau đây:

1. Về phía admin:

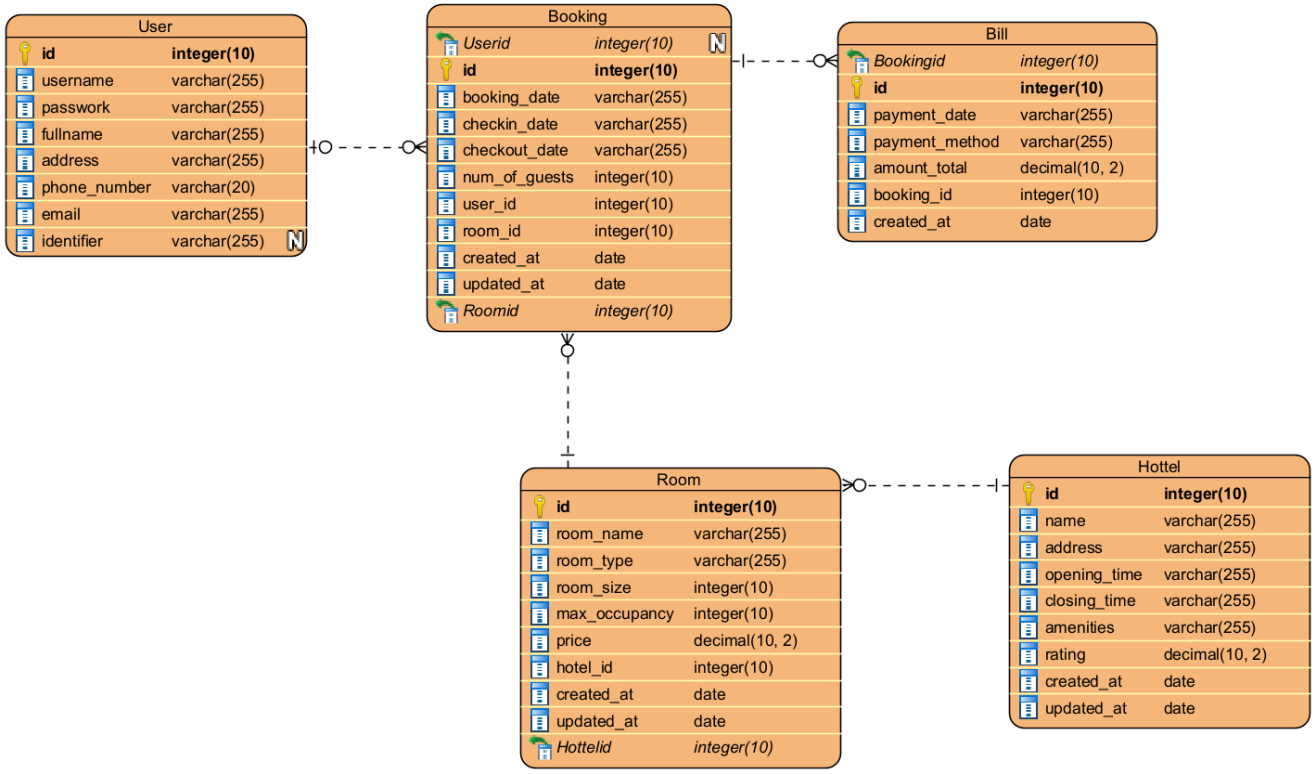
- Quản lý danh sách khách sạn (thêm, sửa, xóa)
- Quản lý danh sách phòng
- Quản lý danh sách user là khách hàng
- Xem thống kê đặt phòng
- Xem thống kê hóa đơn

2. Về phía client

- Xem danh sách khách sạn muốn đặt phòng
- Xem danh sách phòng của khách sạn đó
- Đặt phòng
- Thanh toán
- Xem danh sách hóa đơn của khách hàng đó

II. Thiết kế

1. Thiết kế database :



Nhóm bọn em thiết kế cơ sở dữ liệu gồm 5 thành phần:

- User:

  + identifier
  + username
  + password
  + full_name
  + address
  + phone_number
  + email

- Hotel

  + name
  + address
  + opening_time
  + closing_time
  + amenities
  + rating

- Room
    + room_name
    + room_type
    + room_size
    + max_occupancy
    + price
    + hotel_id
- Booking
    + booking_date
    + checkin_date
    + checkout_date
    + num_of_guests
    + user_id
    + room_id
- Bill
    + payment_date
    + payment_method
    + amount_total
    + booking_id
    + created_at
- Tạo các bảng theo thiết kế cơ sở dữ liệu trong MySQL:

```
CREATE TABLE User (
  id INT AUTO_INCREMENT PRIMARY KEY,
  identifier VARCHAR(255),
  username VARCHAR(255) NOT NULL,
  password VARCHAR(255) NOT NULL,
  full_name VARCHAR(255) NOT NULL,
  address TEXT NOT NULL,
  phone_number VARCHAR(20) NOT NULL,
  email VARCHAR(255) NOT NULL,
  created_at DATETIME DEFAULT CURRENT_TIMESTAMP(),
```

```
  updated_at DATETIME ON UPDATE CURRENT_TIMESTAMP()
);

-- Create Hotel table
CREATE TABLE Hotel (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  address TEXT NOT NULL,
  opening_time VARCHAR(255) NOT NULL,
  closing_time VARCHAR(255) NOT NULL,
  amenities VARCHAR(255) NOT NULL,
  rating DECIMAL(10,2) NOT NULL,
  created_at DATETIME DEFAULT CURRENT_TIMESTAMP(),
  updated_at DATETIME ON UPDATE CURRENT_TIMESTAMP()
);

-- Create Room table
CREATE TABLE Room (
  id INT AUTO_INCREMENT PRIMARY KEY,
  room_name VARCHAR(255) NOT NULL,
  room_type VARCHAR(255) NOT NULL,
  room_size INT NOT NULL,
  max_occupancy INT NOT NULL,
  price DECIMAL(10,2) NOT NULL,
  hotel_id INT NOT NULL,
  created_at DATETIME DEFAULT CURRENT_TIMESTAMP(),
  updated_at DATETIME ON UPDATE CURRENT_TIMESTAMP(),
  FOREIGN KEY (hotel_id) REFERENCES Hotel(id) ON DELETE CASCADE
);

-- Create Booking table
CREATE TABLE Booking (
  id INT AUTO_INCREMENT PRIMARY KEY,
```

```
  booking_date VARCHAR(255) NOT NULL,
  checkin_date VARCHAR(255) NOT NULL,
  checkout_date VARCHAR(255) NOT NULL,
  num_of_guests INT NOT NULL,
  user_id INT NOT NULL,
  room_id INT NOT NULL,
  created_at DATETIME DEFAULT CURRENT_TIMESTAMP(),
  updated_at DATETIME ON UPDATE CURRENT_TIMESTAMP(),
  FOREIGN KEY (user_id) REFERENCES User(id) ON DELETE CASCADE,
  FOREIGN KEY (room_id) REFERENCES Room(id) ON DELETE CASCADE
);

-- Create Bill table
CREATE TABLE Bill (
  id INT AUTO_INCREMENT PRIMARY KEY,
  payment_date VARCHAR(255) NOT NULL,
  payment_method VARCHAR(255) NOT NULL,
  amount_total DECIMAL(10,2) NOT NULL,
  booking_id INT NOT NULL,
  created_at DATETIME DEFAULT CURRENT_TIMESTAMP(),
  updated_at DATETIME ON UPDATE CURRENT_TIMESTAMP(),
  FOREIGN KEY (booking_id) REFERENCES Booking(id) ON DELETE CASCADE
);
```

2. Thiết kế giao diện

- Giao diện đăng nhập

- Giao diện đăng ký

- Giao diện chính của admin:

+ Giao diện quản lý danh sách khách sạn: gồm 1 bảng chứa danh sách các khách sạn cần quản lý với các cột là thuộc tính của khách sạn (id, tên khách sạn, giờ mở,

giờ đóng, đánh giá), cột cuối là cột tùy chọn với 2 chức năng sửa và xóa, có 1 nút để thêm khách sạn, khi bấm vào sẽ hiện 1 modal để người dùng nhập các thông tin cần thêm và nút xác nhận trong modal để lưu dữ liệu vào DB

+ Giao diện quản lý danh sách phòng: thiết kế tương tự giao diện quản lý khách sạn

+ Giao diện quản lý khách hàng: thiết kế tương tự giao diện quản lý khách sạn

+ Giao diện quản lý đặt phòng: thiết kế tương tự giao diện quản lý khách sạn nhưng không có nút thêm (admin chỉ có quyền sửa xóa, không có quyền thêm booking, chỉ client mới thêm được booking)

+ Giao diện thống kế hóa đơn: chỉ có duy nhất các bảng với các cột là các thuộc tính của hóa đơn, không thể thêm, sửa, xóa hóa đơn bền phía adimin

- Giao diện chính của client:

+ Giao diện quản lý danh sách khách sạn: gồm 1 bảng chứa danh sách các khách sạn cần quản lý với các cột là thuộc tính của khách sạn (id, tên khách sạn, giờ mở, giờ đóng, đánh giá), cột cuối là cột tùy chọn với 1 chức năng xem danh sách phòng

+ Khi người dùng click xem danh sách phòng trong tùy chọn sẽ hiện bảng danh sách các phòng của khách sạn đó với 1 tùy chọn là đặt phòng

+ Khi người dùng click đặt phòng, phòng sẽ tự động được thêm vào booking, người dùng chuyển sang giao diện đặt phòng để xem các phòng mình đã đặt

+ Trong giao diện này người dùng có thể click thanh toán để thanh toán cho hóa đơn của mình (có liên kết với blockchan để thanh toán, hiển thị mô hình hóa phòng 3d bằng unity)

+ Cuối dùng người dùng có thể chuyển sang giao diện hóa đơn để xem các hóa đơn đã thanh toán

III. Kiến trúc hệ thống

Hệ thống được thiết kế theo mô hình MVC gồm 2 phần:

1. Back-end (Model + Controller) tạo bằng spring boot (restful API), bao gồm:

- Để kết nối được với database (ví dụ schema hotel) sẽ khai báo file application.property như sau:

```
spring.jpa.hibernate.ddl-auto=update
spring.datasource.url=jdbc:mysql://localhost:3306/hotel
```

```
spring.datasource.username=root
spring.datasource.password=123456

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
```

- Model trong mô hình MVC bao gồm:

+ Các class entity theo thiết kế: User, Hotel, Room, Booking, Bill, các entity được kết nối với cơ sở dữ liệu bằng JPA và được tự động tạo các hàm khởi tạo (có biến, không biến), các hàm getter, setter bằng lombok

Ví dụ: entity User:

```
@Entity
@Table(name = "user")
@Data
@NoArgsConstructor
@AllArgsConstructor
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private Long id;

    @Column(name = "identifier")
    private String identifier;

    @Column(name = "username", nullable = false)
    private String username;

    @Column(name = "password", nullable = false)
    private String password;

    @Column(name = "full_name", nullable = false)
    private String fullName;

    @Column(name = "address", nullable = false)
    private String address;

    @Column(name = "phone_number", nullable = false)
    private String phoneNumber;

    @Column(name = "email", nullable = false)
    private String email;

    @CreationTimestamp
    @Column(name = "created_at", updatable = false, nullable = false)
    private LocalDateTime createdAt;

    @UpdateTimestamp
    @Column(name = "updated_at")
    private LocalDateTime updatedAt;
}
```

+ Các class dto cấu hình request nhận được và response trả về: UserRequestDto, UserResponseDto, HotelRequestDto, HotelResponseDto, …

Ví dụ: UserRequestDto (ở giao diện đăng nhập, người dùng chỉ nhập vào username và password, còn ở giao diện đăng ký, người dùng sẽ nhập đầy đủ các thông tin gồm fullName, address, phoneNumber, …)

```
@Data
@NoArgsConstructor
@AllArgsConstructor
public class UserRequestDto {
    private String username;
    private String password;
    @JsonProperty("full_name")
    private String fullName;
    private String address;
    @JsonProperty("phone_number")
    private String phoneNumber;
    private String email;
}
```

và UserResponseDto:

```
@Data
@NoArgsConstructor
@AllArgsConstructor
public class UserResponseDto {
    private Long id;
    private String identifier;
    private String username;
    private String password;
    @JsonProperty("full_name")
    private String fullName;
    private String address;
    @JsonProperty("phone_number")
    private String phoneNumber;
    private String email;
}
```

- Các class Mapper để chuyển từ RequestDto thành Entity và từ Entity thành ResponseDto:

Ví dụ: UserMapper

```
@Component
public class UserMapper { // Chuyển từ request sang entity và từ entity sang
response

    private final ModelMapper modelMapper;


    public UserMapper() {
        this.modelMapper = new ModelMapper();
```

```
    }

    public User toEntity(UserRequestDto userRequestDto) {
        return modelMapper.map(userRequestDto, User.class);
    }

    public UserResponseDto toDto(User user) {
        return modelMapper.map(user, UserResponseDto.class);
    }
}
```

- Các interface Repository để cung cấp các phương thức để thực hiện các hoạt động cơ bản như tìm kiếm, thêm, sửa đổi và xóa dữ liệu

Ví dụ: UserRepository

```
@Repository
public interface UserRepository extends JpaRepository<User, Long> {
    Optional<User> findByUsernameAndPassword(String username, String
password);
    List<User> findByIdentifier(String identifier);
}
```

- Các interface Service và class ServieImpl để triển khai Service: trung gian giữa Controller và Repository, được sử dụng để xử lý các yêu cầu từ Controller và liên kết với Repository để thực hiện các thao tác trên cơ sở dữ liệu

Ví dụ: UserService:

```
public interface UserService {
    UserResponseDto addUser(UserRequestDto userRequestDto);
    UserResponseDto updateUserPassword(Long id, String newPassword);
    UserResponseDto getUserByUsernameAndPassword(String username, String
password);
    List<UserResponseDto> getAllUsers();
    UserResponseDto getUserById(Long id);
    UserResponseDto updateUser(Long id, UserRequestDto requestDto);
    void deleteUser(Long id);
}
```

và UserServiceImpl:

```
@Service
public class UserServiceImpl implements UserService {

    private final UserRepository userRepository;
    private final UserMapper userMapper;

    @Autowired
    public UserServiceImpl(UserRepository userRepository, UserMapper
userMapper) {
        this.userRepository = userRepository;
```

```
        this.userMapper = userMapper;
    }

    @Override
    public UserResponseDto addUser(UserRequestDto userRequestDto) {
        User user = userMapper.toEntity(userRequestDto);
        user.setIdentifier("Client");
        User savedUser = userRepository.save(user);
        return userMapper.toDto(savedUser);
    }

    @Override
    public UserResponseDto updateUserPassword(Long id, String newPassword) {
        Optional<User> optionalUser = userRepository.findById(id);
        if (optionalUser.isPresent()) {
            User user = optionalUser.get();
            user.setPassword(newPassword);
            User updatedUser = userRepository.save(user);
            return userMapper.toDto(updatedUser);
        } else {
            throw new RuntimeException("User not found with id: " + id);
        }
    }

    @Override
    public UserResponseDto getUserByUsernameAndPassword(String username,
String password) {
        Optional<User> optionalUser =
userRepository.findByUsernameAndPassword(username, password);
        if (optionalUser.isPresent()) {
            User user = optionalUser.get();
            return userMapper.toDto(user);
        } else {
            throw new RuntimeException("User not found with username and
password");
        }
    }

    @Override
    public List<UserResponseDto> getAllUsers() {
        List<User> users = userRepository.findByIdentifier("Client");
        return
users.stream().map(userMapper::toDto).collect(Collectors.toList());
    }

    @Override
    public UserResponseDto getUserById(Long id) {
        User user = userRepository.findById(id)
                .orElseThrow(() -> new RuntimeException("User not found with
id: " + id));
        return userMapper.toDto(user);
    }


    @Override
    public UserResponseDto updateUser(Long id, UserRequestDto requestDto) {
        User user = userRepository.findById(id)
```

```
                .orElseThrow(() -> new RuntimeException("User not found with
id: " + id));
        user.setFullName(requestDto.getFullName());
        user.setAddress(requestDto.getAddress());
        user.setPhoneNumber(requestDto.getPhoneNumber());
        user.setEmail(requestDto.getEmail());
        user = userRepository.save(user);
        return userMapper.toDto(user);
    }

    @Override
    public void deleteUser(Long id) {
        if (!userRepository.existsById(id)) {
            throw new RuntimeException("User not found with id: " + id);
        }
        userRepository.deleteById(id);
    }
}
```

- Cuối cùng là các lớp Controller trong mô hình MVC: xử lý các yêu cầu HTTP từ phía người dùng

Ví dụ: UserController: thực hiện các thao tác CRUD cơ bản, đăng nhập,…

```
@RestController
@RequestMapping("/api/users")
@CrossOrigin
public class UserController {
    private final UserService userService;
    private final Validation validation;

    @Autowired
    public UserController(UserService userService, Validation validation) {
        this.userService = userService;
        this.validation = validation;
    }
    //Thêm 1 user (dùng cho chức năng đăng ký tài khoản)
    @PostMapping
    public ResponseEntity<?> addUser(@RequestBody UserRequestDto
userRequestDto) {
        List<String> list_error = validation.getInputError(userRequestDto);
        if (!list_error.isEmpty()){
            return new ResponseEntity<>(list_error, HttpStatus.BAD_REQUEST);
        }
        UserResponseDto userResponseDto =
userService.addUser(userRequestDto);
        return
ResponseEntity.status(HttpStatus.CREATED).body(userResponseDto);
    }
    //Thay đổi mật khẩu (dùng cho chức năng đổi mật khẩu)
    @PutMapping("/{id}/password")
    public ResponseEntity<UserResponseDto> updateUserPassword(@PathVariable
Long id, @RequestParam String newPassword) {
        UserResponseDto userResponseDto = userService.updateUserPassword(id,
```

```
newPassword);
        return ResponseEntity.ok(userResponseDto);
    }
    //Dùng cho chức năng đăng nhập và kiểm tra tài khoản là Admin (identifier
= '666') hay User
    @GetMapping("/login")
    public ResponseEntity<UserResponseDto>
getUserByUsernameAndPassword(@RequestParam String username, @RequestParam
String password) {
        UserResponseDto userResponseDto =
userService.getUserByUsernameAndPassword(username, password);
        return ResponseEntity.ok(userResponseDto);
    }
    // Lấy tất cả User
    @GetMapping
    public ResponseEntity<List<UserResponseDto>> getAllUsers() {
        List<UserResponseDto> Users = userService.getAllUsers();
        return ResponseEntity.ok(Users);
    }
    // Lấy User theo id
    @GetMapping("/{id}")
    public ResponseEntity<UserResponseDto> getUserById(@PathVariable Long id)
{
        UserResponseDto User = userService.getUserById(id);
        return ResponseEntity.ok(User);
    }
    // Sửa User
    @PutMapping("/{id}")
    public ResponseEntity<?> updateUser(@PathVariable Long id, @RequestBody
UserRequestDto requestDto) {
        List<String> list_error = validation.getInputError(requestDto);
        if (!list_error.isEmpty()){
            return new ResponseEntity<>(list_error, HttpStatus.BAD_REQUEST);
        }
        UserResponseDto user = userService.updateUser(id, requestDto);
        return ResponseEntity.ok(user);
    }
    // Xóa User
    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteUser(@PathVariable Long id) {
        userService.deleteUser(id);
        return ResponseEntity.noContent().build();
    }
}
```

- Front-end (View) được tạo bằng react, bao gồm:

+Quản lý điều hướng

```
function App() {
  const [user, setuser] = useState(null);
  const Applogin = () => (
    <BrowserRouter>
        <Routes>
          <Route path={PATH.LOGIN} element={<LoginPage setuser={setuser} />} />
          <Route path={PATH.SIGNUP} element={<Signup setuser={setuser}/> } />
        </Routes>
    </BrowserRouter>
  );
  const Appusing = () => (
    <BrowserRouter>
      <Header setuser={setuser} user={user} />
      <Component>
        <SideBar user={user} width={200} style={{ background: "#fff" }} />
        <Routes>
          <Route path={PATH.ROOM_HOTEL} element={<Room user={user} />} />
          <Route path={PATH.HOTEL} element={<Hotel user={user} />} />
          <Route path={PATH.ROOM} element={<Room user={user} />} />
          <Route path={PATH.CUSTOMER} element={<Customer user={user} />} />
          <Route path={PATH.BILL} element={<Bill user={user}/>} />
          <Route path={PATH.BILL_USER} element={<Bill user={user}/>} />
          <Route path={PATH.BOOKING} element={<Booking user={user}/>}/>
        </Routes>
      </Component>
    </BrowserRouter>
  );
  if (user !== null) {
    return Appusing(user, setuser);
  } else return Applogin(user, setuser);
}
```

Trong đó, sử dụng useState để lưu user, khi chưa đăng nhập hay đăng ký, user = null -> sẽ chỉ hiện giao diện đăng nhập hoặc đăng ký. Sau khi đăng nhập sẽ điều hướng đến giao diện danh sách khách sạn

+ Dùng thư viện axios để thay thế cho fetch rest api, từ đó lấy dữ liệu từ back end

VD: hotelApi.js

```
import axios from "axios";

axios.defaults.baseURL = "http://localhost:8080/api";

//Lấy client
export function getuser(){

}

//lay khach san theo id
export function gethotelbyid(id){
  return axios.get(`/hotels/${id}`)
}

// Lấy danh sách các khách sạn
export function gethotel() {
  return axios.get("/hotels");
}

// Thêm khách sạn
export function posthotel(data) {
  console.log(data);
  return axios.post("/hotels", {
    name: data.name,
    amenities: data.amenities,
    address: data.address,
    rating: data.rating,
    opening_time: data.opening_time,
    closing_time: data.closing_time
  });
}

// Sửa khách sạn theo id khách sạn
export function puthotel(data,id) {
  return axios.put(`/hotels/${id}`, {
    name: data.name,
    amenities: data.amenities,
    address: data.address,
    rating: data.rating,
    opening_time: data.opening_time,
    closing_time: data.closing_time
  });
}

// Xóa công ty theo công ty id
export function deletehotel(id) {
  return axios.delete(`/hotels/${id}`);
}
```

+ Xử lý dữ liệu và gắn vào component tương ứng:

VD: lấy danh sách khách sạn

```
useEffect(() => {
  gethotel()
    .then((response) => {
      setHotels(response.data);
    })
    .catch((error) => console.log(error));
}, []);
```

```
const columns = [
  {
    title: "Mã khách sạn",
    dataIndex: "id",
    key: "id",
    width: "8vw",
    ...getColumnSearchProps("id"),
  },
  {
    title: "Tên khách sạn",
    dataIndex: "name",
    key: "name",
    width: "12vw",
    ...getColumnSearchProps("name"),
  },
  {
    title: "Tiện ích",
    dataIndex: "amenities",
    key: "amenities",
  },
  {
    title: "Địa chỉ",
    dataIndex: "address",
    key: "address",
  },
  {
    title: "Giờ mở",
    dataIndex: "opening_time",
    key: "opening_time",
```

```
    width: "6vw",
  },
  {
    title: "Giờ đóng",
    dataIndex: "closing_time",
    key: "openingTime",
    width: "6vw",
  },
  {
    title: "Đánh giá",
    dataIndex: "rating",
    key: "rating",
    width: "8vw",
    sorter: (a, b) => a.rating - b.rating,
    sortDirections: ["descend", "ascend"],
  },
  {
    title: "Tùy chọn",
    key: "action",
    width: "10vw",
    render: (record) => (
      <Action>
        <Dropdown overlay={actionMenu(record)}>
          <Button>
            <Space>
              Tùy chọn
              <DownOutlined />
            </Space>
          </Button>
        </Dropdown>
      </Action>
    ),
  },
];
```

```
<HotelTable>
  <Table
    dataSource={hotels}
    columns={columns}
    scroll={{ y: 500 }}
    pagination={{
      defaultPageSize: 10,
      showSizeChanger: true,
      pageSizeOptions: ["10", "20", "30"],
    }}
  ></Table>
```

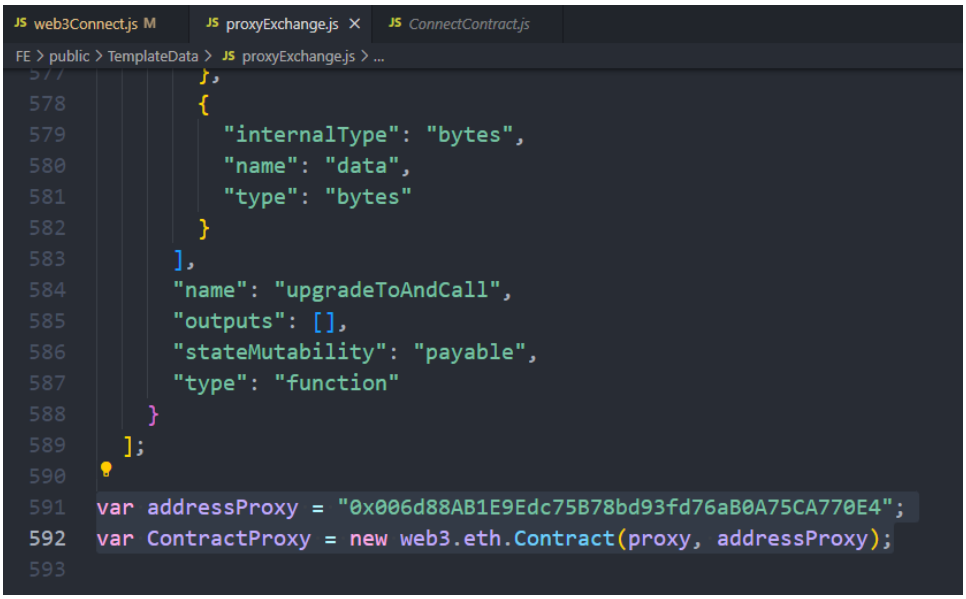
- Module Thanh toán được tạo từ frame work Unity, được chạy từ file run.html tại path : “WebProject\btl_web\FE\public\ run.html”. File run.html ngoài chức năng hiển thị giao diện còn gọi tới các file để call tới blockchain :

```
run.html
FE > public > run.html > ...
79              };
80            }).catch((message) => {
81              alert(message);
82            });
83          };
84          document.body.appendChild(script);
85        </script>
86        <script src="TemplateData/web3.min.js"></script>
87        <script src="TemplateData/web3Connect.js"></script>
88        <script src="TemplateData/ConnectContract.js"></script>
89        <script src="TemplateData/proxyExchange.js"></script>
90      </body>
91    </html>
92
```

- File web3Connect.js thực hiện chức năng kết nối tới ví metamask

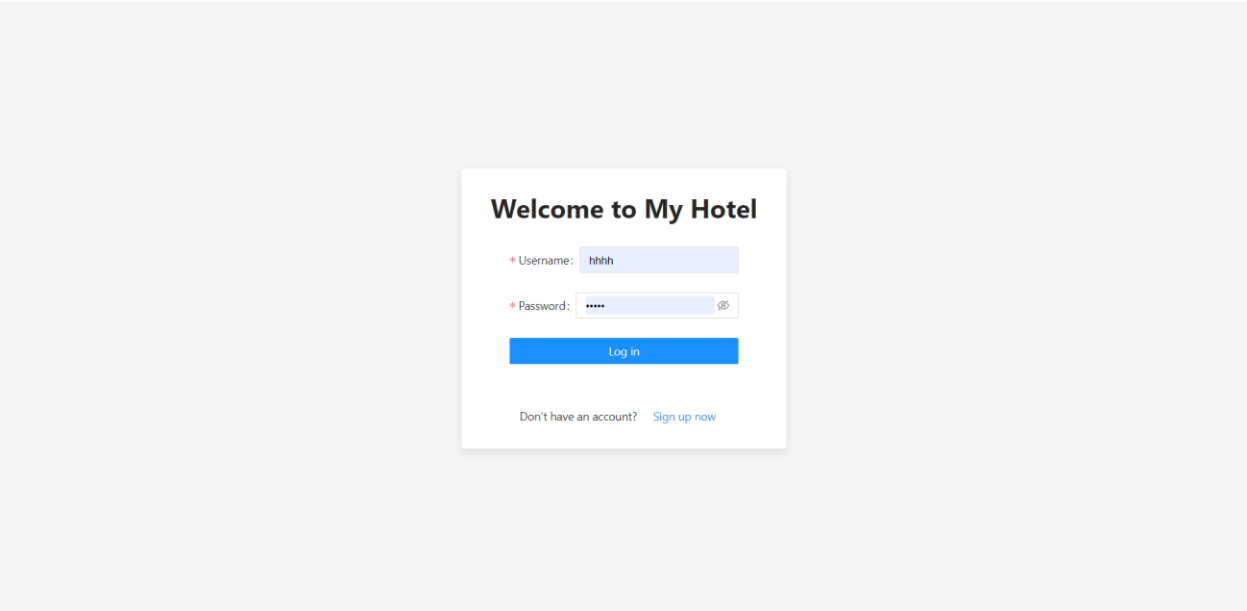
```
web3Connect.js M
FE > public > TemplateData > web3Connect.js > window.ethereum.on("accountsChanged") callback
1  //kiểm tra xem trình duyệt có hỗ trợ Ethereum hay không
2  //bằng cách kiểm tra xem window.ethereum có tồn tại hay không.
3  //Nếu trình duyệt hỗ trợ Ethereum, thì đoạn mã tạo một đối tượng Web3 mới
4  //cho phép giao tiếp với blockchain thông qua window.ethereum.
5  if (window.ethereum) {
6    web3 = new Web3(window.ethereum);
7    // gọi để hiển thị một popup kết nối khi người dùng đang truy cập vào ứng dụng.
8    ethereum.enable();
9
10   window.ethereum.on("accountsChanged", function () {
11     //để tải lại trang và cập nhật các thông tin về tài khoản mới.
12     location.reload();
13   });
14
15 }
16
```

- File proxyExchange.js dùng để cấp quyền chọc vào blockchain thông qua addressProxy (địa chỉ ví cần chuyển tiền cho thanh toán) . Sau khi tạo đối tượng ContractProxy, bọn em có thể sử dụng các phương thức của nó để tương tác với hợp đồng thông minh Ethereum, ví dụ như gọi hàm, truy xuất thông tin, lắng nghe sự kiện, và nhiều hoạt động khác.

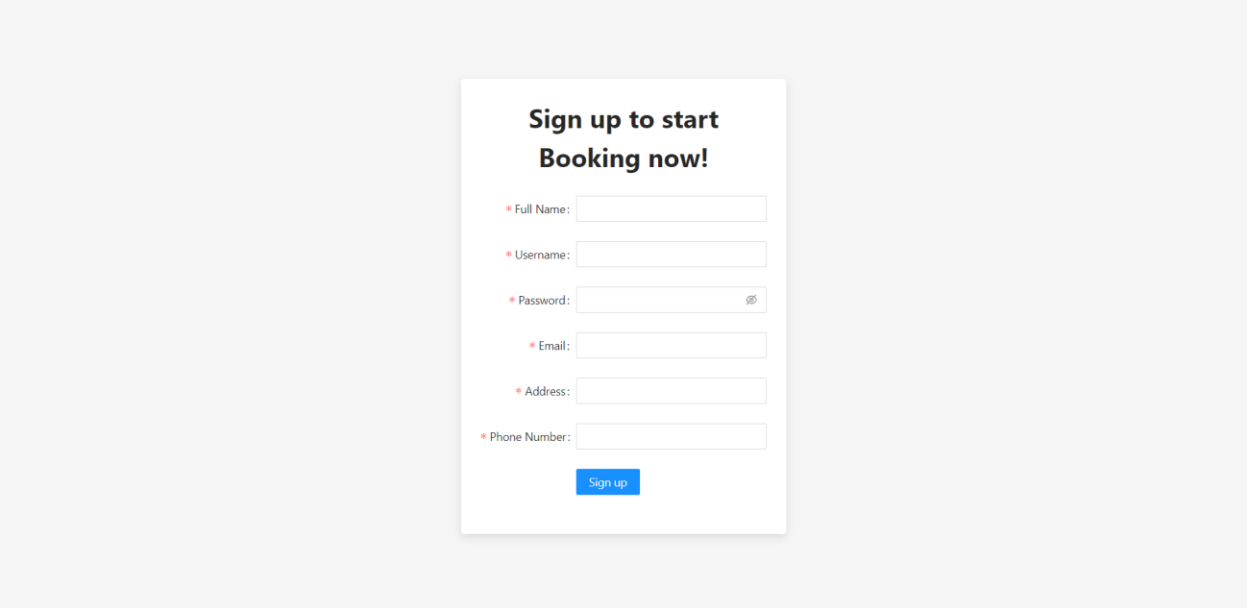


IV. Ảnh chụp màn hình các phần giao diện

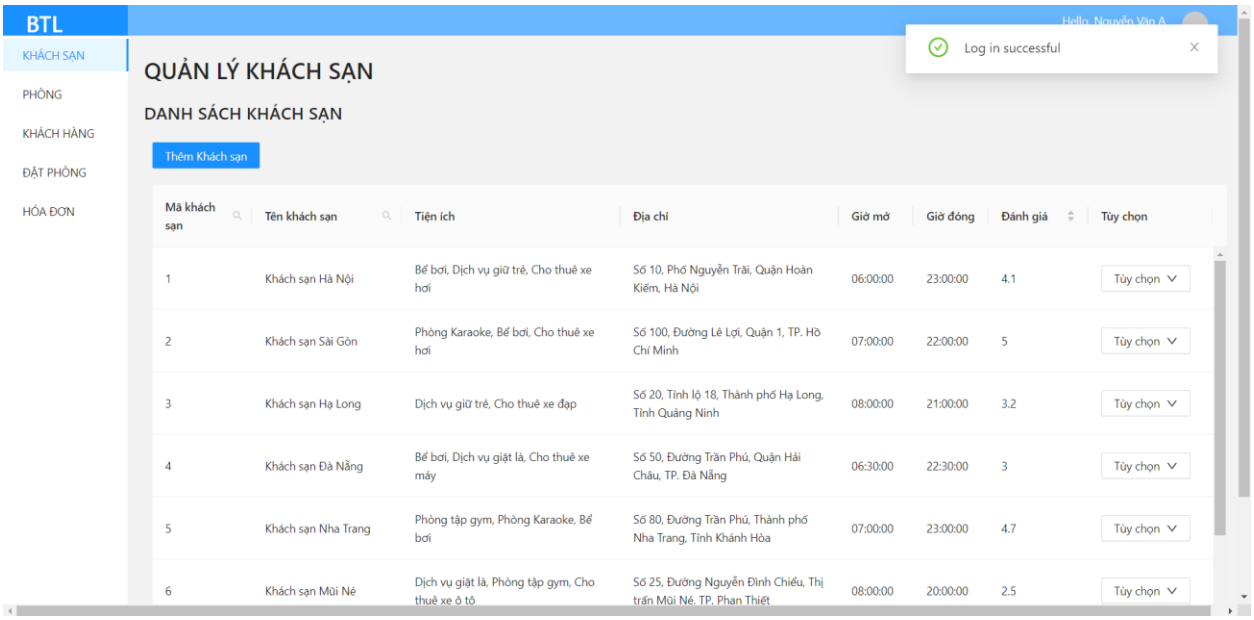
- Giao diện đăng nhập:

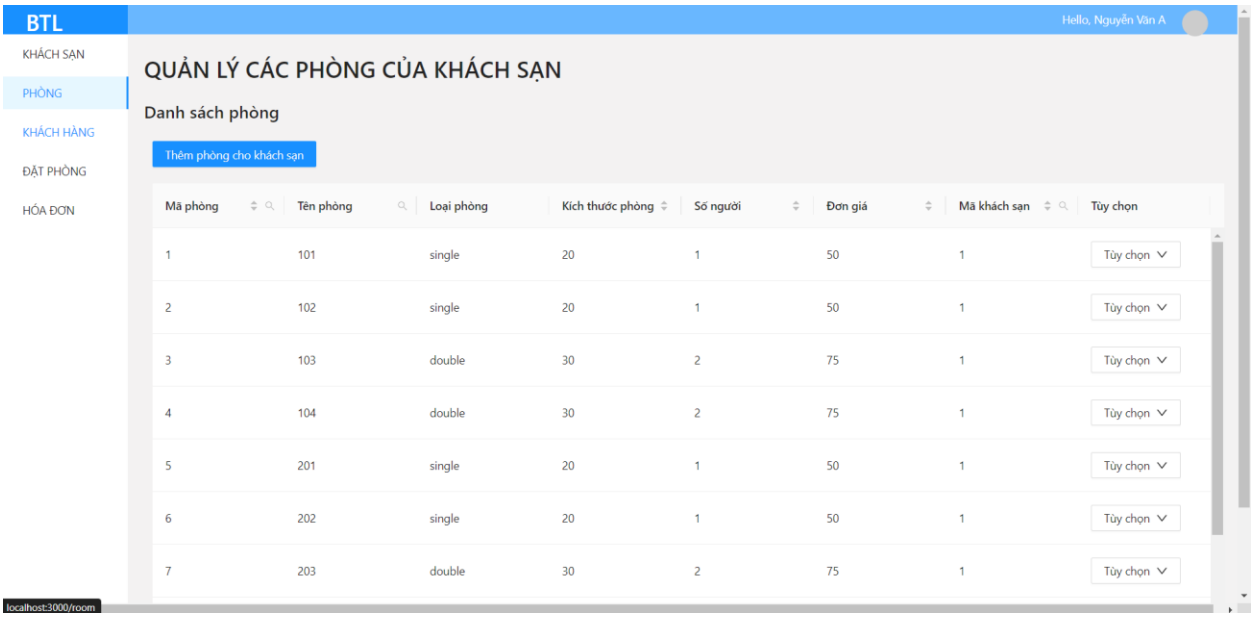


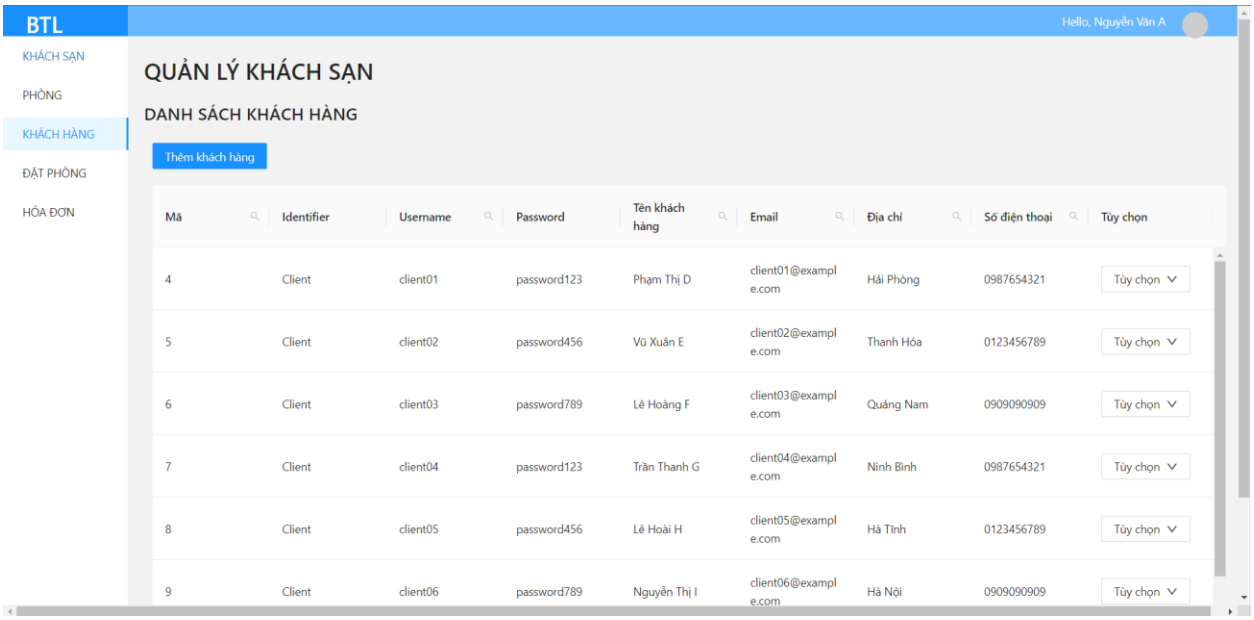
- Giao diện đăng ký:

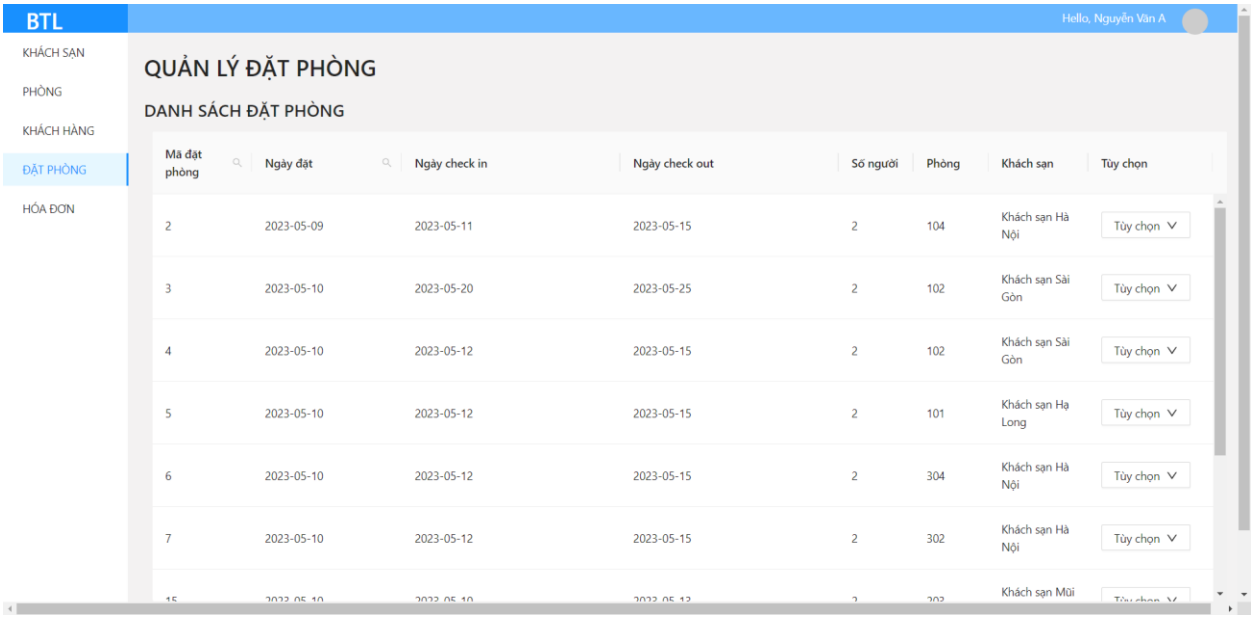


- Giao diện chính của Admin: Các giao diện quản lý khách sạn, phòng, khách hàng, đặt phòng, hóa đơn là lượt như sau:

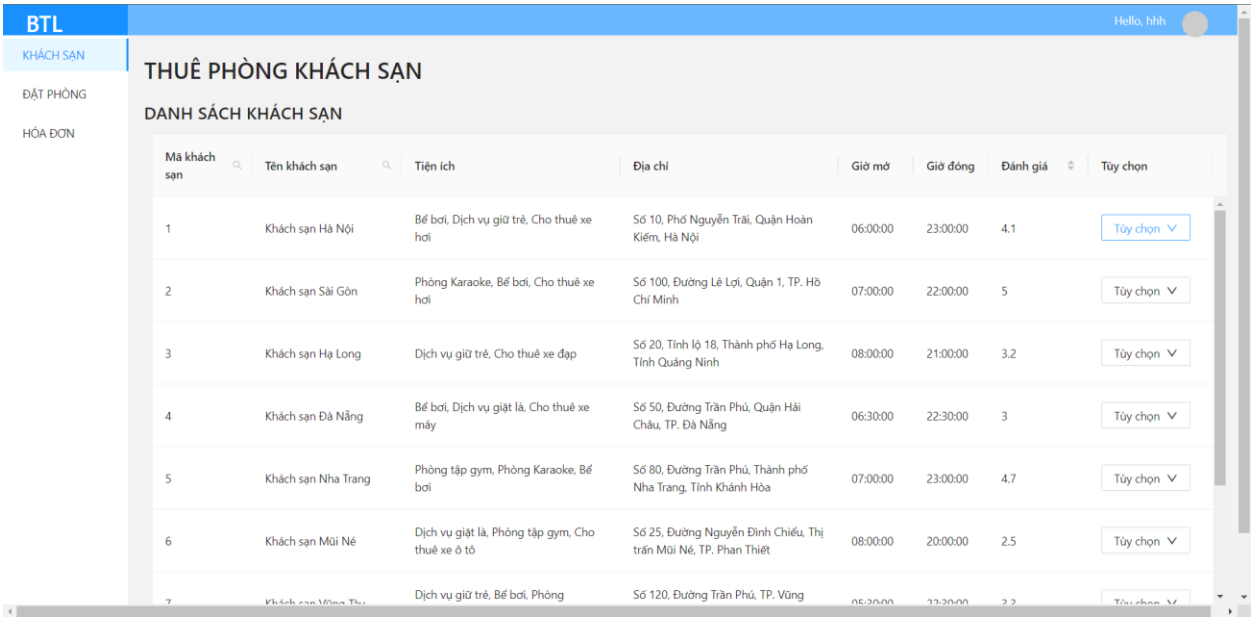


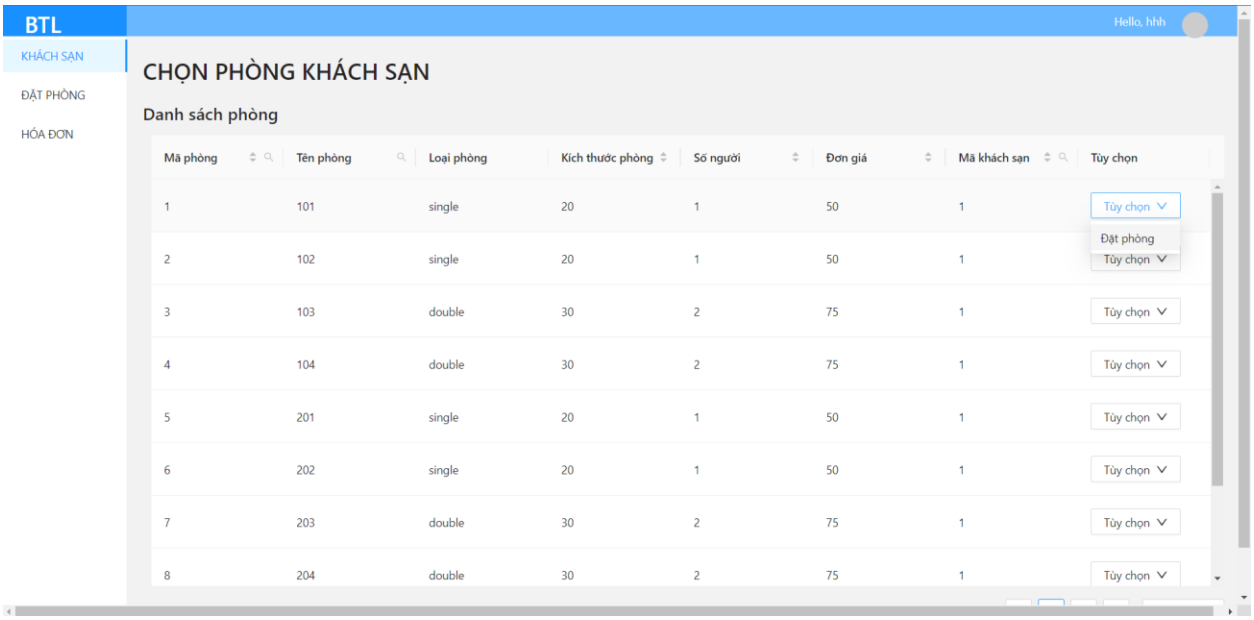

BTL
Hello, Nguyễn Văn A
KHÁCH SẠN
PHÒNG
KHÁCH HÀNG
ĐẶT PHÒNG
HÓA ĐƠN
QUẢN LÝ CÁC PHÒNG CỦA KHÁCH SẠN
Danh sách phòng
Thêm phòng cho khách sạn
Mã phòng
Tên phòng
Loại phòng
Kích thước phòng
Số người
Đơn giá
Mã khách sạn
Tùy chọn
1 101 single 20 1 50 1 Tùy chọn
2 102 single 20 1 50 1 Tùy chọn
3 103 double 30 2 75 1 Tùy chọn
4 104 double 30 2 75 1 Tùy chọn
5 201 single 20 1 50 1 Tùy chọn
6 202 single 20 1 50 1 Tùy chọn
7 203 double 30 2 75 1 Tùy chọn
localhost:3000/room

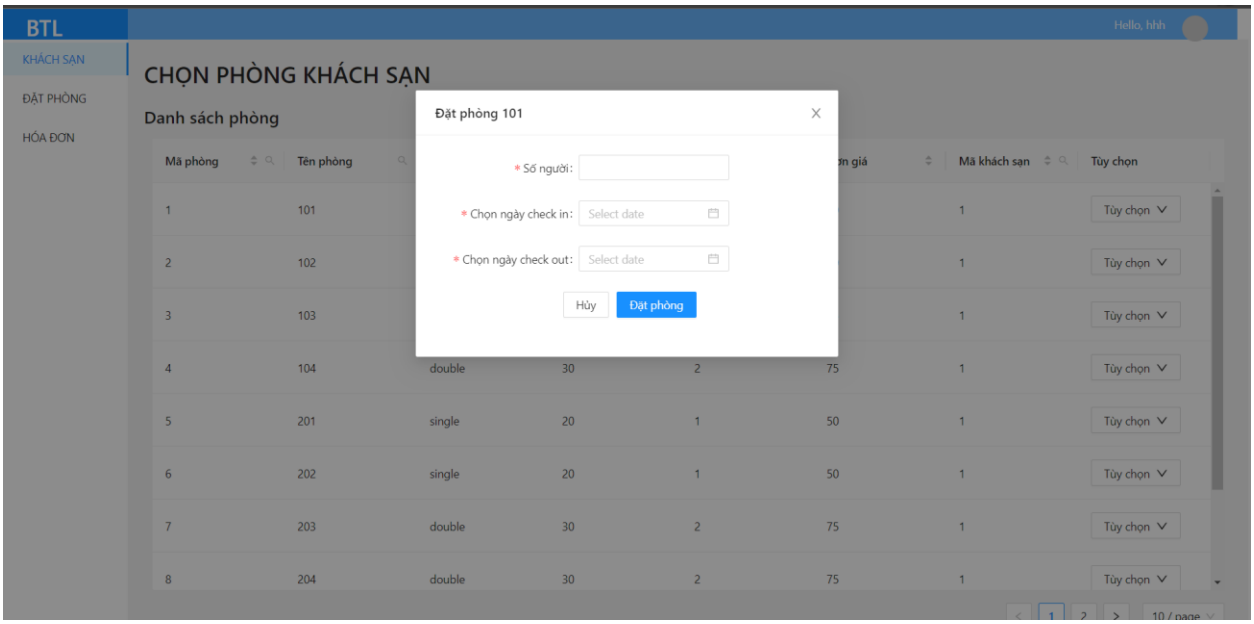

BTL
Hello, Nguyễn Văn A
KHÁCH SẠN
PHÒNG
KHÁCH HÀNG
ĐẶT PHÒNG
HÓA ĐƠN
QUẢN LÝ KHÁCH SẠN
DANH SÁCH KHÁCH HÀNG
Thêm khách hàng
Mã
Identifier
Username
Password
Tên khách hàng
Email
Địa chỉ
Số điện thoại
Tùy chọn
4 Client client01 password123 Phạm Thị D client01@example.com Hải Phòng 0987654321 Tùy chọn
5 Client client02 password456 Vũ Xuân E client02@example.com Thanh Hóa 0123456789 Tùy chọn
6 Client client03 password789 Lê Hoàng F client03@example.com Quảng Nam 0909090909 Tùy chọn
7 Client client04 password123 Trần Thanh G client04@example.com Ninh Bình 0987654321 Tùy chọn
8 Client client05 password456 Lê Hoài H client05@example.com Hà Tĩnh 0123456789 Tùy chọn
9 Client client06 password789 Nguyễn Thị I client06@example.com Hà Nội 0909090909 Tùy chọn

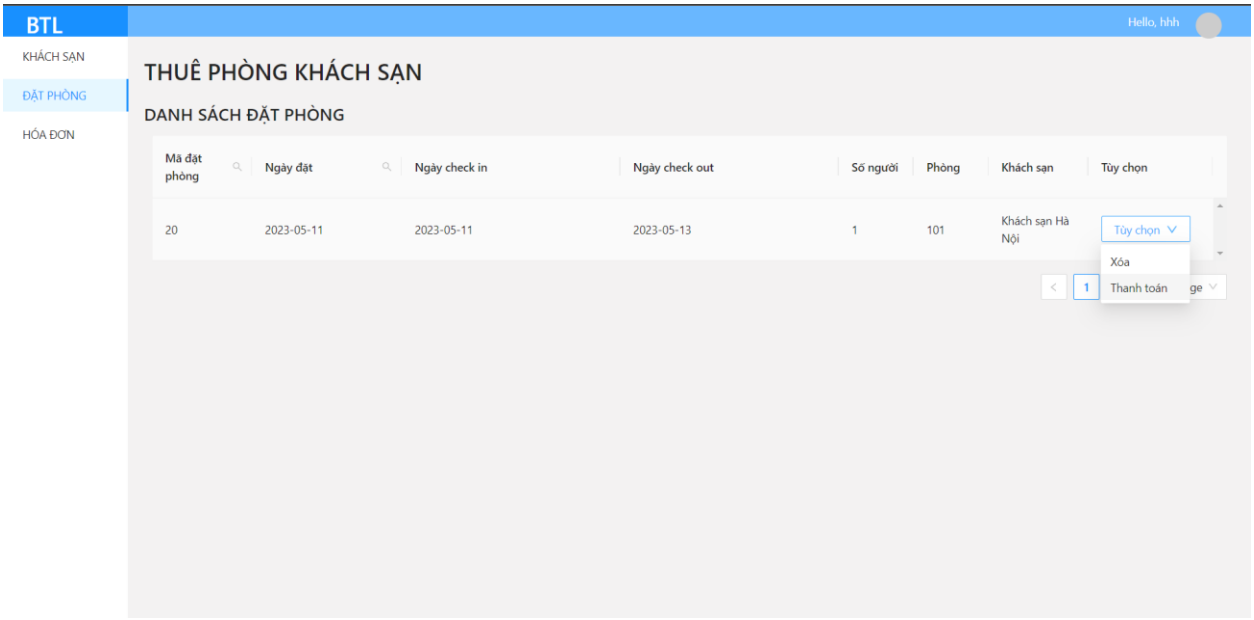


- Giao diện chính của Client: Các giao diện xem danh sách khách sạn, xem danh sách phòng của khách sạn, xem danh sách đặt phòng, xem danh sách hóa đơn của user lần lượt như sau:

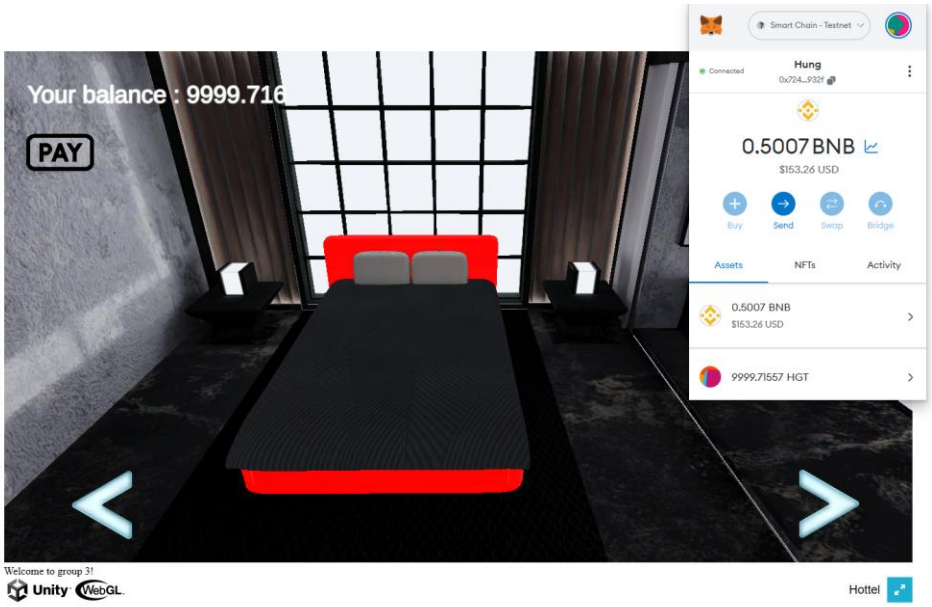


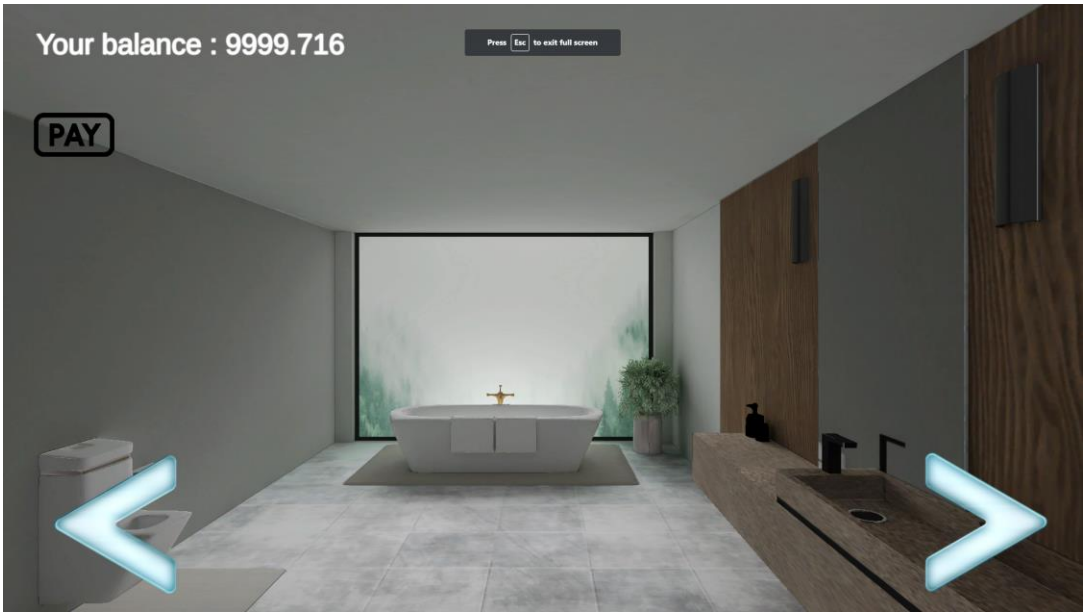

BTL
Hello, hhh
KHÁCH SẠN
ĐẶT PHÒNG
HÓA ĐƠN
CHỌN PHÒNG KHÁCH SẠN
Danh sách phòng
Mã phòng
Tên phòng
Loại phòng
Kích thước phòng
Số người
Đơn giá
Mã khách sạn
Tùy chọn
1 101 single 20 1 50 1 Tùy chọn
Đặt phòng
2 102 single 20 1 50 1 Tùy chọn
3 103 double 30 2 75 1 Tùy chọn
4 104 double 30 2 75 1 Tùy chọn
5 201 single 20 1 50 1 Tùy chọn
6 202 single 20 1 50 1 Tùy chọn
7 203 double 30 2 75 1 Tùy chọn
8 204 double 30 2 75 1 Tùy chọn


BTL
Hello, hhh
KHÁCH SẠN
ĐẶT PHÒNG
HÓA ĐƠN
CHỌN PHÒNG KHÁCH SẠN
Danh sách phòng
Đặt phòng 101
* Số người:
* Chọn ngày check in: Select date
* Chọn ngày check out: Select date
Hủy
Đặt phòng
Mã phòng
Tên phòng
Mã khách sạn
Tùy chọn
1 101 1 Tùy chọn
2 102 1 Tùy chọn
3 103 1 Tùy chọn
4 104 double 30 2 75 1 Tùy chọn
5 201 single 20 1 50 1 Tùy chọn
6 202 single 20 1 50 1 Tùy chọn
7 203 double 30 2 75 1 Tùy chọn
8 204 double 30 2 75 1 Tùy chọn
10 / page

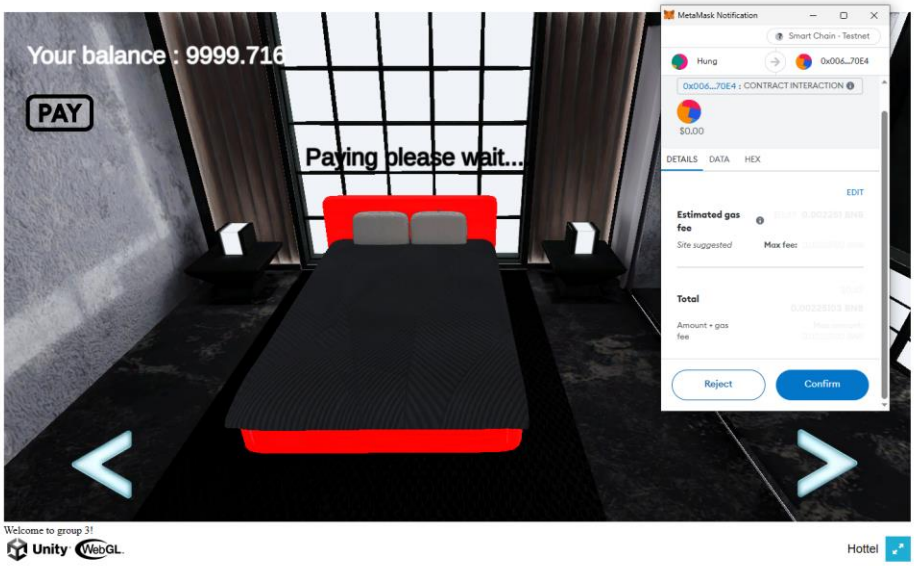


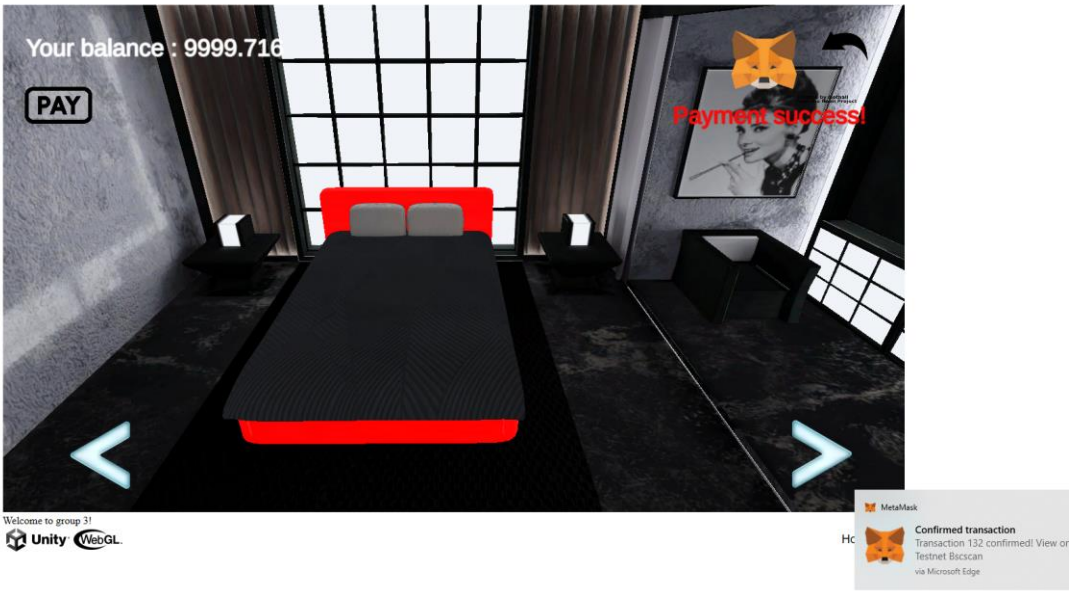
- Giao diện thanh toán bằng sử dụng ví metamask :


Your balance : 9999.716
PAY
Smart Chain - Testnet
Connected
Hung
0.5007 BNB
$153.26 USD
Buy
Send
Swap
Bridge
Assets
NFTs
Activity
0.5007 BNB
$153.26 USD
9999.71557 HGT
Welcome to group 3!
Unity WebGL
Hottel


Your balance : 9999.716
Press Esc to exit full screen
PAY


Your balance : 9999.716
PAY
Welcome to group 3!
Unity WebGL
Hottel


Your balance : 9999.716
PAY
Paying please wait...
MetaMask Notification
Smart Chain - Testnet
Hung
0x006...70E4
0x006...70E4 : CONTRACT INTERACTION
$0.00
DETAILS
DATA
HEX
EDIT
Estimated gas fee
Site suggested
Max fee:
Total
Amount + gas fee
Reject
Confirm
Welcome to group 3!
Unity WebGL
Hottel



V. Phân công công việc

- Nguyễn Việt Hải: viết back-end

- Phạm Huy Hoàng: viết front-end

- Nguyễn Tiến Hưng: thiết kế, fix lỗi + tạo thanh toán bằng blockchan + mô hình 3D unity