Project Structure :-

The Whole Project is distributed in 3 files :-

- **User : Where we create the user and store his wallet.**
- **Camera : where we store camera and its attributes.**
- **CRA : The main file to access both files and do all functions that are shown in output.**

Camera.java ::

```
package CRA;

class Camera {
	private int id;
	private String brand;
	private String model;
	private double price;
	private String status;
	public Camera(int id, String brand, String model, double price,
String status) {
	this.id = id;
	this.brand = brand;
	this.model = model;
	this.price = price;
	this.status = status;
	}
      public int getId() {
	return id;
	}
	public String getBrand() {
	return brand;
	}
	public String getModel() {
	return model;
	}
	public double getPrice() {
	return price;
	}
	public String getStatus() {
	return status;
	}
	public void setStatus(String status) {
	this.status = status;
	}
	@Override
	public String toString() {
	return String.format("%-10s%-15s%-15s%-20.2f%-15s", id, brand,
model, price,
      status);
	}
}
```

User.java ::

```
package CRA;

import java.util.ArrayList;

class User {
 private String username;
 private String password;
 private ArrayList<Camera> myCameras;
 private double wallet;
 public User(String username, String password, double wallet) {
 this.username = username;
 this.password = password;
 myCameras = new ArrayList<>();
 // wallet = 0;
 this.wallet = wallet;
 }

 public User(String username, String password, ArrayList<Camera> myCameras,
double
wallet) {
super();
this.username = username;
this.password = password;
 myCameras = new ArrayList<>();
 wallet = 0;
}
public String getUsername() {
 return username;
 }
public String getPassword() {
 return password;
 }
 public ArrayList<Camera> getMyCameras() {
 return myCameras;
 }
 public void addCamera(Camera camera) {
 myCameras.add(camera);
 }

 public void setMyCameras(ArrayList<Camera> myCameras) {
 this.myCameras = myCameras;
 }
 public void removeCamera(int id) {
 for (Camera camera : myCameras) {
 if (camera.getId() == id) {
 myCameras.remove(camera);
 System.out.println("Camera with ID " + id + " removed from your list.");
 return;
 }
 }
 System.out.println("Camera with ID " + id + " not found in your list.");
 }
 public void viewMyCameras() {
 if (myCameras.isEmpty()) {
 System.out.println("You have no cameras in your list.");
 return;
 }
 System.out.println(String.format("%-10s%-15s%-15s%-20s%-15s", "CAMERA ID",
"BRAND", "MODEL", "PRICE (PER DAY)", "STATUS"));
```

```
 for (Camera camera : myCameras) {
 System.out.println(camera);
 }
 }

 public double getWallet() {
 return wallet;
 }
 public void addMoneyToWallet(double amount) {
 wallet += amount;
 }
 public void deductMoneyFromWallet(double amount) {
 if (wallet < amount) {
 System.out.println("Insufficient balance in your wallet.");
 return;
 }
 wallet -= amount;
 System.out.println("Amount of INR " + amount + " deducted from your
wallet.");
 }

 public void rentCamera(Camera camera, int days) {
 if (wallet < camera.getPrice() * days) {
 System.out.println("Insufficient balance in your wallet.");
 return;
 }
 camera.setStatus("RENTED");
 double rentAmount = camera.getPrice() * days;
 wallet -= rentAmount;
 System.out.println("Amount of INR " + rentAmount + " deducted from your
wallet for renting camera with ID " + camera.getId() + ".");
 }
}
```

CRA.java ::

```
package CRA;

import java.util.*;

public class CRA {
static Scanner sc = new Scanner(System.in);
static List<Camera> cameraList = new ArrayList<>();
static List<User> users = new ArrayList<>();
static Map<String, Double> wallet = new HashMap<>();
static String loggedInUser = null;
public static void main(String[] args) {
cameraList.add(new Camera(1, "Samsung", "s123", 2900.0, "Available"));
cameraList.add(new Camera(2, "Nikon", "n918", 4000.0, "Available"));
users.add(new User("Nirmal", "Nirmal321", 10000));
System.out.println("+----------------------------------------+");
System.out.println("| WELCOME TO CAMERA RENTAL APP |");
System.out.println("+----------------------------------------+");
System.out.println("PLEASE LOGIN TO CONTINUE -");
// login
while (true) {
try {
```

```
System.out.print("USERNAME - ");
String username = sc.nextLine();
System.out.print("PASSWORD - ");
String password = sc.nextLine();
for (int i = 0; i < users.size(); i++) {
User user = users.get(i);
if (user.getUsername().equalsIgnoreCase(username) &&
user.getPassword().equalsIgnoreCase(password)) {
loggedInUser = username;
break;
}
}
if (loggedInUser == null) {
System.out.println("Invalid credentials. Please try again.");
} else {
break;
}
} catch (Exception e) {
System.out.println("Please Enter creds in alphanumeric & valid ones\n");
}
}
// main menu
while (true) {
System.out.println("1. MY CAMERA");
System.out.println("2. RENT A CAMERA");
System.out.println("3. VIEW ALL CAMERAS");
System.out.println("4. MY WALLET");
System.out.println("5. EXIT");
try {
int choice = Integer.parseInt(sc.nextLine());
switch (choice) {
case 1:
myCamera(loggedInUser);
break;
case 2:
viewAllCameras();
rentCamera(loggedInUser);
break;
case 3:
viewAllCameras();
break;
case 4:
myWallet(loggedInUser);
break;
case 5:
System.out.println("Thank you for using the app!");
System.exit(0);
default:
System.out.println("Invalid choice. Please try again.");
break;
}
} catch (NumberFormatException | InputMismatchException e) {
System.out.println("Please enter valid number from 1 to 5\n");
}
}
}
public static void myWallet(String loggedInUser) {
for (int i = 0; i < users.size(); i++) {
User user = users.get(i);
if (user.getUsername().equalsIgnoreCase(loggedInUser)) {
```

```
System.out.println("YOUR CURRENT WALLET BALANCE IS - INR."+
user.getWallet());
System.out.print("DO YOU WANT TO DEPOSIT MORE AMOUNT TO YOUR WALLET?(1.YES
2.NO) - ");
String input = sc.nextLine();
if (input.equals("1")) {
try {
System.out.print("ENTER THE AMOUNT (INR) -");
String input2 = sc.nextLine();
int amountUpdate = Integer.parseInt(input2);
// Find the user with the specified username
User userToUpdateWallet = null;
for (int j = 0; j < users.size(); j++) {
User user1 = users.get(j);
if
(user.getUsername().equalsIgnoreCase(loggedInUser)) {
userToUpdateWallet = user1;
break;
}
}
// Update the user wallet with money
if (userToUpdateWallet != null) {
userToUpdateWallet.addMoneyToWallet(amountUpdate);
System.out.println("YOUR WALLET BALANCE UPDATED SUCCESSFULLY. CURRENT
WALLET BALANCE - INR."+ user.getWallet());
} else {
System.out.println("User with username " + loggedInUser + " not found.");
}
} catch (NumberFormatException |
InputMismatchException e) {
System.out.println("\nPlease enter valid amount in numbers");
}
} else {
break;
}
}
}
public static void myCamera(String loggedInUser) {
Scanner sc = new Scanner(System.in);
boolean backToMenu = false;
while (!backToMenu) {
try {
System.out.println("1. ADD");
System.out.println("2. REMOVE");
System.out.println("3. VIEW MY CAMERAS");
System.out.println("4. GO TO PREVIOUS MENU");
// System.out.print("Enter your choice: ");
int choice = sc.nextInt();
switch (choice) {
case 1:
addCamera();
break;
case 2:
removeCamera();
break;
case 3:
viewMyCameras(loggedInUser);
break;
case 4:
backToMenu = true;
```

```
break;
default:
System.out.println("Invalid choice. Please try again.");
break;
}
} catch (InputMismatchException e) {
System.out.println("\nPlease enter valid number from 1 to 4\n");
break;
}
}
// sc.close();
}
@SuppressWarnings("resource")
private static void rentCamera(String loggedInUser) {
Scanner sc = new Scanner(System.in);
try {
// Get user input
System.out.print("ENTER THE CAMERA ID YOU WANT TO RENT - ");
int cameraCode = sc.nextInt();
// Get the camera from the camera list
Camera getcamera = getCameraById(cameraList, cameraCode);
if (getcamera == null) {
System.out.println("Camera with ID " + cameraCode + " not found.");
return;
}
if(getcamera.getStatus().equalsIgnoreCase("Not Available")){
System.out.println("Camera not available");
return;
}
// Get rental period
System.out.print("ENTER RENTAL PERIOD (in days) - ");
int rentalPeriod = sc.nextInt();
sc.nextLine(); // Consume the newline character left by nextInt()
getcamera.setStatus("RENTED");
double rentAmount = getcamera.getPrice() * rentalPeriod;
// Find the user with the specified username
User userToUpdate = null;
for (int i = 0; i < users.size(); i++) {
User user = users.get(i);
if (user.getUsername().equalsIgnoreCase(loggedInUser)) {
userToUpdate = user;
break;
}
}
if (rentAmount > userToUpdate.getWallet()) {
System.out.println("ERROR : TRANSACTION FAILED DUE TO INSUFFIECIENT WALLET
BALANCE. PLEASE DEPOSIT THE AMOUNT TO YOUR WALLET.");
return;
}
// Update the myCameras ArrayList of the found user object
if (userToUpdate != null) {
userToUpdate.addCamera(getcamera);
userToUpdate.deductMoneyFromWallet(rentAmount);
}
System.out.println("Camera rented successfully for " +
rentalPeriod + " days. Your wallet balance is now $"+
userToUpdate.getWallet());
System.out.println("YOUR TRANSACTION FOR CAMERA - " + "with rent INR." +
rentAmount + " HAS SUCCESSFULLY COMPLETED.");
} catch (NumberFormatException | InputMismatchException e) {
System.out.println("Please enter valid number\n");
```

```
}
// sc.close();
}
public static Camera getCameraById(List<Camera> cameraList, int id) {
for (Camera camera : cameraList) {
if (camera.getId() == id) {
return camera;
}
}
return null; // Camera with given id not found in the list
}
public static void addCamera() {
try {
// System.out.println("\nADD CAMERA");
System.out.print("ENTER THE CAMERA BRAND - ");
String brand = sc.nextLine();
System.out.print("ENTER THE MODEL - ");
String model = sc.nextLine();
System.out.print("ENTER THE PER DAY PRICE (INR) - ");
double price = Double.parseDouble(sc.nextLine());
int id = generateCameraId();
Camera camera = new Camera(id, brand, model, price, "Available");
cameraList.add(camera);
System.out.println("YOUR CAMERA HAS BEEN SUCCESSFULLY ADDED TO THE
LIST.\n");
} catch (NumberFormatException | InputMismatchException e) {
System.out.println("Please enter valid the valid data\n");
}
}
public static int generateCameraId() {
int lastCameraId = 0;
if (!cameraList.isEmpty()) {
Camera lastCamera = cameraList.get(cameraList.size() - 1);
lastCameraId = lastCamera.getId();
}
int newCameraId = lastCameraId + 1;
return newCameraId;
}
public static void removeCamera() {
 Scanner scanner = new Scanner(System.in);
 try {

System.out.println("==========================================================
=========="
);
 System.out.println("CAMERA ID BRAND MODEL PRICE (PER DAY) STATUS");

System.out.println("==========================================================
======== ");
 for (int i = 0; i < cameraList.size(); i++) {
 Camera camera = cameraList.get(i);
 System.out.format("%-10d%-10s%-10s%-18s%s%n", camera.getId(),
camera.getBrand(), camera.getModel(),
 camera.getPrice(),
camera.getStatus().equalsIgnoreCase("Available") ?
"Available" : "Not Available");
 }
 System.out.println("----------------------------------------------------
");
 System.out.print("ENTER THE CAMERA ID TO REMOVE - ");
 // Use hasNextInt() method to check if there is an integer input
```

```
 if (scanner.hasNextInt()) {
 int cameraId = scanner.nextInt();
 // process the input
 boolean found = false;
 for (int i = 0; i < cameraList.size(); i++) {
 Camera camera = cameraList.get(i);
 if (camera.getId() == cameraId) {
 cameraList.remove(camera);
 found = true;
 System.out.println("CAMERA SUCCESSFULLY REMOVED FROM THE LIST - ");
 break;
 }
 }
 if (!found) {
 System.out.println("Camera with ID " + cameraId + " not found.");
 }
 } else {
 System.out.println("Invalid input. Please enter an integer.");
 }
 } catch (NumberFormatException | InputMismatchException e) {
 System.out.println("Please enter valid data\n");
 } finally {
// scanner.close();
 }
}
public static void viewAllCameras() {
System.out.println("\nFOLLOWING IS THE LIST OF AVAILABLE CAMERA(S)-");
System.out.println("==========================================================
========================");
if (cameraList.size() == 0) {
System.out.println("No cameras available for rent.");
} else {
System.out.println("CAMERA ID\tBRAND\t\tMODEL\t\tPRICE (PER DAY)\tSTATUS");
System.out.println("==========================================================
========================");
for (Camera camera : cameraList) {
System.out.println(camera.getId() + "\t\t" +
camera.getBrand() + "\t\t" + camera.getModel() + "\t\t"
+ camera.getPrice() + "\t\t" +
camera.getStatus());
}
}
System.out.println("==========================================================
========================");
}
public static void viewMyCameras(String loggedInUser) {
for (User user : users) {
if (user.getUsername().equals(loggedInUser)) {
user.viewMyCameras();
return;
}
}
System.out.println("User with username " + loggedInUser + " not found.");
}
}
```