Write up

1 . application offers the following capabilities:

• It provides a comprehensive list of cameras available for rent, including their brand, model,

and daily rental rates.

• Users can browse through the camera listing and select the one they wish to rent.

• The application allows users to view or add funds to their wallet, which can be used to pay

for the camera rentals.

• Users can easily navigate from the current execution context to the main context, enabling

them to access additional features or information.

• Finally, users have the option to close the application when they are finished.

**2. Appearance:**

• Create a welcome screen with the name and developer details.

• Include options that the user can interact with to access various features of the application.

**3. User Interactions:**

• Welcome

• Display the application name and developer details.

• Provide options for the user to access different features

**Camera Listing:**

▪ The camera listing shows all available cameras for rent.

▪ Each camera is displayed with its brand, model, and per-day rental amount.

▪ The user can browse through the list to find their desired camera.

▪ Once a camera is selected, the user can proceed to the "Rent Camera" feature.

**Rent Camera:**

▪ Detailed information about the selected camera is displayed, including specifications and

additional details.

▪ The total rental cost is calculated based on the per-day rental amount and the desired

duration of rental.

▪ If the user has sufficient funds in their wallet, the rental cost is deducted from their wallet

amount.

▪ The user is then prompted to confirm the rental and provided with any relevant instructions

regarding pickup or delivery.

**Wallet:**

▪ The user can view their current wallet amount, which displays the available funds for rental

payments.

▪ An option is provided to add funds to the wallet, allowing the user to conveniently manage

their balance.

**Navigation:**

▪ At any point, the user can navigate back to the main context from any subcontext within the

application.

▪ This ensures easy access to other features and information without losing the current

progress.

**Close Application:**

▪ The user has the option to close or exit the application when they are finished using it.

▪ This allows for a seamless and user-friendly experience.

**4. application flow is as follows:**

▪ The user launches the application.

▪ The welcome screen is displayed, showcasing the application name and developer details.

▪ The user is presented with options to access different features.

▪ Upon selecting the "Camera Listing" option, the application retrieves and displays a list of

available cameras.

▪ The user chooses a camera from the list.

▪ The application presents detailed information about the selected camera and requests the

user to input the rental duration.

▪ The user provides the rental duration.

▪ The application calculates the total rental cost based on the per-day rental amount and

duration, deducting it from the user's wallet amount.

▪ The application confirms the rental and provides any relevant instructions for pickup or

delivery.

▪ The user can navigate back to the main context to access other features or view the camera

listing again.

▪ The user has the option to view their wallet amount and add funds if desired.

▪ Finally, the user can close the application when they are finished using it

**5 . Sprint Planning:**

**Sprint 1 (1 week):**

▪ Set up version control with Git and create a GitHub account for collaborative development.

▪ Create the project structure and initialize the application.

▪ Implement the welcome screen, displaying the application name and developer details.

▪ Implement basic navigation functionality to switch between different contexts.

▪ Create a mock camera listing to be used for testing and development purposes.

**Sprint 2 (2 weeks):**

▪ Implement the camera listing functionality, retrieving and displaying the actual list of

available cameras.

▪ Enable camera selection, allowing users to view detailed information about a selected

camera.

▪ Implement the wallet functionality, allowing users to view their current wallet amount.

▪ Implement the option to add funds to the wallet for convenient payment during the rental

process.

**Sprint 3 (1 week):**

▪ Implement the camera rental functionality, including calculating the total rental cost based

on the per-day rental amount and rental duration.

▪ Deduct the rental cost from the user's wallet amount and update the wallet balance

accordingly.

▪ Confirm the rental with the user and provide any relevant instructions regarding pickup or

delivery.

▪ Conduct thorough testing, fix any bugs or issues, and make necessary improvements based

on feedback or requirements.

**6 .The camera rental application will implement the following Java concepts and data structures:**

▪ Object-oriented programming principles to design and implement classes for cameras,

wallet, and other entities.

▪ Collection classes such as ArrayList or LinkedList to efficiently store and manage camera

data.

▪ Sorting and searching algorithms to enhance the camera listing functionality, allowing users

to easily find and select cameras.

▪ File handling techniques to save and retrieve data, ensuring persistent storage of camera and

user information.

Algorithm:

1. Start the CameraRentalApp class.

2.Initialize the cameraList, MycameraList, scanner, cameraIdCounter, and wallet variables.

3.Create a list of User objects and add predefined user credentials.

4.Create a list of Camera objects and add predefined camera details.

5.Prompt the user to enter their username and password.

6.Validate the user's credentials by comparing them with the list of User objects.

7.If the credentials are invalid, display an error message and exit the program.

8.Display the main menu.

9.Loop indefinitely until the program is terminated.

1. Prompt the user to enter their choice.

11.Based on the user's choice, perform the following actions:

a. If the choice is 1, go to the "My Camera" submenu.

b. If the choice is 2, go to the "Rent a Camera" submenu.

c. If the choice is 3, go to the "View All Cameras" submenu.

d. If the choice is 4, go to the "My Wallet" submenu.

e. If the choice is 5, exit the program.

f. If the choice is invalid, display an error message.

12.Implement the "My Camera" submenu:

a. Display the submenu options.

b. Prompt the user to enter their choice.

c. Based on the user's choice, perform the following actions:

i. If the choice is 1, go to the "Add Camera" function.

ii. If the choice is 2, go to the "Remove Camera" function.

iii. If the choice is 3, go to the "View My Cameras" function.

iv. If the choice is 4, return to the main menu.

v. If the choice is invalid, display an error message.

13.Implement the "Add Camera" function:

a. Prompt the user to enter the camera brand, model, and rental amount.

b. Create a new Camera object with the entered details and add it to the cameraList.

c. Display a success message.

14.Implement the "Remove Camera" function:

a. Prompt the user to enter the camera ID to remove.

b. Iterate through the cameraList and remove the camera with the matching ID.

c. Display a success message if the camera is found and removed, or an error message if not found.

15.Implement the "View My Cameras" function:

a. Check if the MycameraList is empty.

b. If empty, display a message indicating no cameras.

c. If not empty, display the details of each camera in the MycameraList.

16Implement the "View All Cameras" submenu:

a. Check if the cameraList is empty.

b. If empty, display a message indicating no cameras.

c. If not empty, display the details of each camera in the cameraList.

17.Implement the "Rent a Camera" function:

a. Display the available cameras for rent.

b. Prompt the user to enter the camera ID they want to rent.

c. Check if the user has sufficient balance in their wallet.

d. If the balance is sufficient, set the camera as rented, add it to the MycameraList, and deduct the rental amount from the wallet balance.

e. Display a success message if the transaction is successful, or an error message if the balance is insufficient.

18.Implement the "My Wallet" submenu:

a. Display the current balance in the wallet.

b. Prompt the user if they want to deposit an amount to the wallet

1. Finally implement the “Exit” in the menu.