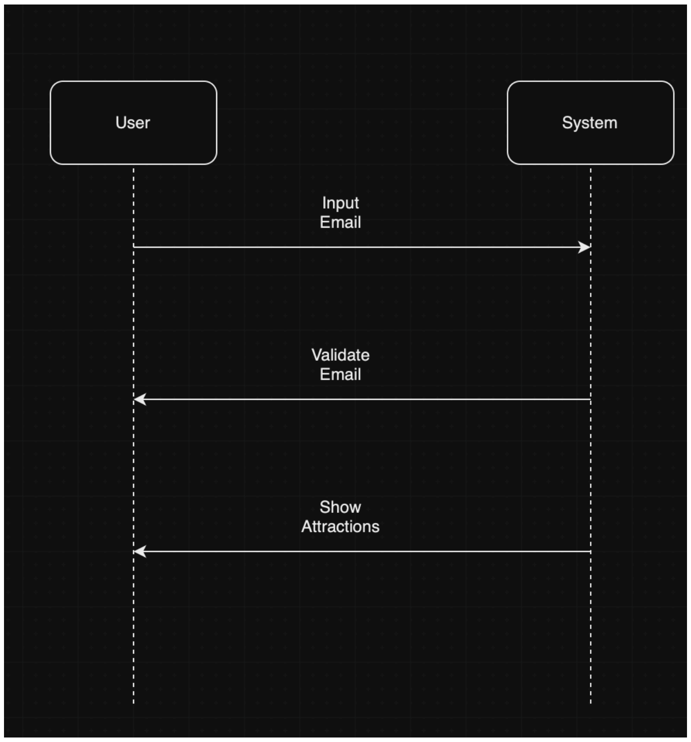
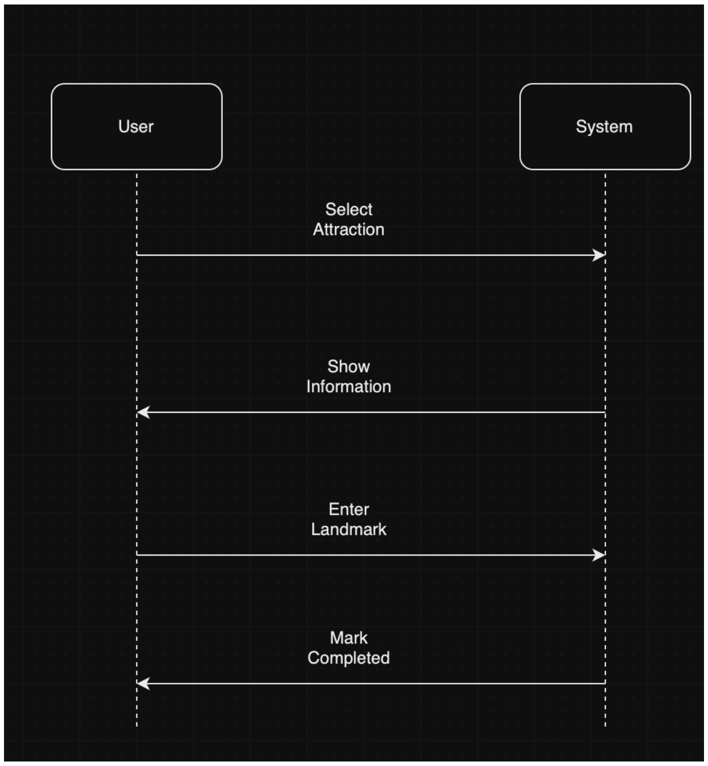
Login SSD



View Attraction SSD



Operation Contracts

Operation: validateEmail(email)

Description:

This operation validates the email input by the user during the login process. If the email is valid, the system proceeds to the attractions screen. If the email is invalid, the system shows an error message prompting the user to input a valid email address.

Preconditions:

1. The user has entered a value in the email field.
2. The email input field is not empty.
3. The email format follows a valid pattern (e.g., "user@domain.com").
4. The system has access to a method for validating the email (such as a regex pattern check or an API).

Postconditions:

1. If the email format is valid:
2. The user is logged in successfully.
3. The system transitions to the attractions screen (or next step).
4. If the email format is invalid:
5. An error message is shown to the user indicating the email is invalid (e.g., "Please enter a valid email address").
6. The system remains on the login screen, allowing the user to correct their input.

Effects:

1. Valid Email: The system triggers a transition to the next screen (such as AttractionsActivity).
2. Invalid Email: The system displays an error message, prompting the user to enter a correct email address.
3. Updates the SharedPreferences or session data, marking the user as logged in (if implemented).

EXAMPLE:
Operation: validateEmail(email)
Preconditions:
- Email field is not empty.
- The email input is in a valid format (matches a regex pattern like user@domain.com).

Postconditions:
- If email is valid, user proceeds to AttractionsActivity.
- If email is invalid, an error message is displayed (e.g., "Invalid email format").

Effect:
- Validate the email format using regex or a similar method.
- If valid, user is logged in, and the AttractionsActivity is shown.
- If invalid, display an error message and allow the user to retry.

Operation Name: showAttractions()

Description:

This operation is triggered after the user successfully logs in. It retrieves and displays a list of attractions that the user can explore. The list may come from a local database, an API, or predefined data.

<u>Preconditions:</u>

1. The user has successfully logged in (i.e., the email has been validated).
2. The system has access to the list of attractions (either from a local database or an API).
3. The AttractionsActivity has been initialized and is ready to display the list.

<u>Postconditions:</u>

1. The system displays a list of attractions on the screen.
2. If the system cannot retrieve the list (e.g., due to a network failure), it displays an error message (e.g., "Unable to load attractions").
3. The user can interact with the list (e.g., click an attraction to view details).

<u>Effects:</u>
- Success: The system fetches the list of attractions and displays them in a UI component (such as a RecyclerView or ListView).
- Failure: The system shows an error message or fallback UI indicating that attractions could not be loaded (e.g., "Please try again later").

- User interaction is enabled to allow the user to select an attraction from the list.

EXAMPLE:

Operation: showAttractions()

Preconditions:

- User has logged in successfully.
- The attractions list is available, either from a local source or a remote API.
- AttractionsActivity is active and can display the list.

Postconditions:

- The list of attractions is shown to the user.
- If data retrieval fails, an error message is shown ("Unable to load attractions").
- The user is able to click on an attraction to view more details.

Effect:

- Retrieve the list of attractions from the database or API.
- Display the list in a UI component such as a RecyclerView or ListView.
- Handle potential errors (e.g., network failure) by showing an appropriate message to the user.

Operation Name: selectAttraction(attractionID)

Description:

When the user selects an attraction from the list, this operation retrieves detailed information about the selected attraction and displays it on the screen. The user is then able to see more details, such as the attraction's description, photos, or location.

Preconditions:

1. The user is logged in and has accessed the attractions list.
2. The user has clicked on a valid attraction from the list.
3. The attractionID parameter is passed, identifying the selected attraction.

Postconditions:

1. The system fetches the details of the selected attraction.
2. The attraction details (e.g., description, image, address) are displayed on the screen.
3. If an error occurs (e.g., the attraction details cannot be fetched), an error message is shown.

Effects:

- Success: The system fetches and displays the detailed information about the selected attraction.
- Failure: If data retrieval fails, an error message is displayed (e.g., "Unable to load attraction details").

EXAMPLE:
Operation: selectAttraction(attractionID)
Preconditions:
 - User is logged in and viewing a list of attractions.
 - The user selects an attraction from the list.
 - The system has access to the attraction's detailed information.
Postconditions:
 - The details of the selected attraction are displayed (description, photos, location).
 - If there's an error retrieving the details, an error message is shown.
Effect:
 - Fetch the details for the selected attraction using the `attractionID`.
 - Display the details to the user in a new screen or fragment.
 - Handle errors gracefully by showing a relevant message to the user.

Operation Name: saveProgress(progress)

Description:

This operation stores the user's progress, such as which attractions they have already visited. The progress data will be used to track the user's journey and provide feedback.

Preconditions:

1. The user is logged in and has accessed the attractions list.
2. The progress data is available and needs to be stored (e.g., a list of visited attractions).
3. The system has access to a local storage or database to save the progress.

Postconditions:

1. The user's progress is saved in local storage or a database.
2. If the save operation fails (e.g., due to lack of storage or connectivity), an error message is displayed.

Effects:

- Success: The system saves the user’s progress (e.g., which attractions have been visited).
- Failure: An error message is displayed indicating that the progress could not be saved.

EXAMPLE:
Operation: saveProgress(progress)
Preconditions:
- User is logged in and has visited some attractions.
- The progress data (such as visited attractions) is available to be stored.

Postconditions:
- The user’s progress is saved in the local storage or database.
- If an error occurs, an error message is shown.

Effect:
- Save the user's progress data (e.g., list of visited attractions) in SharedPreferences or a database.
- If the operation fails, display an appropriate error message.