Practical Module:
OpenWorld Agency

Inés González Valdés

71807452A

Bachelor of Software Engineering

Human-Computer Interaction

Table of contents

INTRODUCTION

The goal of this project is to develop a complete application in Java Swing for the management of bookings in a fictional agency OpenWorld Travel Agency, specialized in theme parks and associated accommodations. The main aim is to computerize the booking process with the creation of a system which allows customers to consult parks and accommodations, visualize detailed information of each one and book tickets and/or accommodations in a simple and intuitive way.

All the information related to parks and accommodations is obtained from two data files (parks.dat and accommodations.dat). The app shows this information clearly, allow date selection, number of people, apply discounts, validate the capacity of accommodations and generate a final summary. Moreover, it must show a minigame that may give a voucher for the next purchase.

The application allows to:

- Book tickets for theme parks
- Book accommodations
- Insert comments
- Modify or remove the reservation
- Register the customer information
- Generate a booking summary
- Play a minigame to obtain a voucher
- Save the final booking

The option to improve the mark with <u>internationalization it has been implemented</u> inside the application. The interface can change dynamically between English and Spanish, including:

- Menus
- Buttons
- Labels
- Tooltips
- JOptionPanes messages
- Texts summaries and descriptions

This internationalization has been implemented ensuring that the logic layer remains independent from the interface and that translations are handled exclusively in the user interface layer through ResourceBundle.

To enhance the visual appearance of the application, a modern Look and Feel was researched and applied, improving the aesthetic quality of the interface. The application is perfectly adapated for navigating with tab key or with mnemonics (Alt+Key).

Regarding layout management, the application uses Absolute Layout for most panels. However, two panels—ThemeParksPanel and AccommodationsPanel—use a combination of layout managers (BorderLayout, BoxLayout, GridBagLayout, and FlowLayout) to allow proper resizing and dynamic content adjustment.

APPLICATION DEVELOPMENT

LOGIC

1. Model Layer

The model package contains all the domain classes used to represent the main data of the application. They do not contain interface logic or internationalization code, which keeps them clean, reusable, and easy to maintain.

Below is a detailed explanation of each class and its most relevant methods.

1.1. Accommodation

Description

Represents an accommodation option available in the agency. It stores essential information such as type, category, price, and the associated theme park.

Main Attributes

accommodationCode: Unique identifier of the accommodation.

type: Accommodation type (HO = hotel, AP = apartment, AH = aparthotel).

category: Star rating (1–5).

name: name of the accommodation.

parkCode: Code of the theme park to which it belongs.

price: Price per night (per person in hotels, per room in apartments/aparthotels).

maxOccupancy: Maximum number of people per room (2 for hotels, 4 for apartments/aparthotels).

Relevant Methods

Constructor: Initializes all attributes and automatically assigns maxOccupancy based on the accommodation type.

Getters: Provide access to the accommodation data.

toString(): Returns the accommodation name, used for display in combo boxes and lists.

1.2. AccommodationReservation

Description

Represents a reservation for an accommodation, including the selected accommodation, entry date, number of nights, and number of poeple.

Main Attributes

accommodation: The selected Accommodation

date: Check-in date.

nights: Number of nights booked.

people: Number of guests.

Relevant Methods

Constructor: Creates a complete accommodation reservation.

Getters: Provide access to reservation details.

toString(): Generates a textual summary of the reservation, used in the final booking confirmation.

1.3. Booking

Description

Represents the final booking made by the customer. It may include a ticket reservation, an accommodation reservation, or both. It also stores all calculated prices, discounts, and whether the customer obtained a gift voucher from the minigame.

Main Attributes

customer: Customer who made the booking.

parkReservation: Ticket reservation .

accommodationReservation: Accommodation reservation

ticketPrice: Total price of theme park tickets.

accommodationPrice: Total price of the accommodation.

discount: Discount applied

totalPrice: Final price after discounts.

giftVoucher: Indicates whether the customer won a €100 voucher.

bookingDate: Date when the booking was generated.

Relevant Methods

Constructor: Initializes all booking data.

setGiftVoucher(boolean): Activates the gift voucher.

setTotalPrice(double): Updates the final price.

toString(): Generates a complete booking confirmation text, used when saving the final booking to a file.

1.4. Cell

Description

Represents a cell in the minigame grid. Each cell may or may not contain the winning prize and has an associated image.

<u>Main Attributes</u>

winning: Indicates whether the cell contains the prize.

imagePath: Path to the image displayed when the cell is revealed.

<u>Relevant Methods</u>

isWinning(): Returns whether the cell is the winning one.

setImagePath(String): Updates the image shown after clicking the cell.

1.5. Customer

<u>Description</u>

Represents the customer who completes the booking. Stores basic personal information.

<u>Main Attributes</u>

name: Customer's first name.

surname: Customer's last name.

id: Customer's identification number

<u>Relevant Methods</u>

Getters: Provide access to customer information.

1.6. ThemePark

<u>Description</u>

Represents a theme park available in the agency. Contains general information, location, description, and ticket prices.

<u>Main Attributes</u>

parkCode: Unique identifier of the park.

name: Name of the theme park.

country: Country where the park is located.

location: City .

description: Detailed description of the park.

adultPrice: Ticket price for adults.

childPrice: Ticket price for children.

Relevant Methods

Getters: Provide access to park information.

toString(): Returns the park name for display in lists and combo boxes.

1.7. TicketReservation

Description

Represents a reservation for theme park tickets, including the selected park, date, number of adults, number of children, and number of days.

Main Attributes

park: The selected ThemePark.

date: Start date of the visit.

adults: Number of adult tickets.

children: Number of child tickets.

days: Duration of the visit in days.

Relevant Methods

Constructor: Creates a complete ticket reservation.

Getters: Provide access to reservation details.

toString(): Generates a textual summary of the ticket reservation.

2. Service Layer

The service layer contains all the business logic of the application.While the model layer stores data, the service layer performs operations.

These classes do not interact with the user interface directly and do not perform translations. Instead, they return key-based text, which is later translated in the UI using ResourceBundle, ensuring a clean separation of concerns.

2.1. AccommodationService

Description

Provides all business logic related to accommodations, including filtering, capacity validation, and generating information strings.

<u>Relevant Methods</u>

filter(accommodations, type, category, park): returns the accommodations that match the selected filters.

getAccommodationInfo(a): builds a key-based string with accommodation details

getAccommodationImagePath(a): returns the file path of the accommodation image.

isValidAccommodationCapacity(acc, people, rooms): checks whether the selected number of people fits within the allowed occupancy.

getAccommodationsForSelectedPark(accommodations, park): returns accommodations assocciated to the selected theme park.

2.2. BookingService

<u>Description</u>

Handles the creation of bookings and generates reservation summaries for both the UI and the final confirmation file.

<u>Relevant Methods</u>

createBooking(customer, ticketRes, accRes, ticketPrice, accPrice, discount, total): creates and returns a complete Booking object.

buildReservationSummary(ticketRes, accRes, ticketPrice, accPrice, discount, totalWithoutDiscount, total): builds a DefaultListModel<String> summarizing the reservation using translation keys.

buildBookingSummaryText(booking, parkName): generates the full booking confirmation text using translation keys.

2.3. MiniGameService

<u>Description</u>

Implements the logic of the minigame that allows the user to win a discount voucher.

<u>Relevant Methods</u>

resetGame(): initializes the game, creates cells, and randomly selects the winning one.

selectCell(index): processes the user’s selection, reveals images, and returns whether the user won.

getCellCount(): returns the number of cells in the game.

getCell(index): returns the cell at the specified index.

hasWon(): returns whether the user won the game.

hasPlayed(): returns whether the user has already played.

getWinningIndex(): returns the index of the winning cell.

2.4. PriceService

Description

Centralizes all price calculations, including ticket prices, accommodation prices, offer discounts.

Relevant Methods

calculateParkPrice(ticketRes): computes the total cost of theme park tickets.

calculateAccommodationPrice(accRes): computes the total cost of the accommodation.

calculateDiscount(ticketRes, accRes, parkOnOffer): calculates the 15% offer discount when applicable.

calculateTotalAfterDiscount(ticketRes, accRes, parkOnOffer): computes the final total after applying discounts.

round(value): rounds a number to two decimal places.

2.5. ThemeParkService

Description

Provides operations related to theme parks, including retrieving park information, mapping parks to accommodations, and generating key-based descriptions.

Relevant Methods

getAllParks(parks): returns the full list of parks.

getParkInfo(park): builds a key-based string containing park details

getParkImagePath(park): returns the file path of the park image.

getParkNameByCode(parks, parkCode): returns the name of the park that matches the given code.

getParksForSelectedAccommodation(parks, accommodation): returns the park associated with the selected accommodation.

3. Utility Layer

The utility layer contains helper classes that support the application by handling low-level operations such as reading data files, saving bookings, and storing file-related constants.

These classes do not contain business logic or UI logic; instead, they provide reusable functionality that the controller and services rely on.

3.1. DataLoader

Description

Responsible for loading theme parks and accommodations from external .dat files.It parses each line, splits the fields, and constructs the corresponding model objects.

Relevant Methods

loadThemeParks(): reads the parks data file, parses each line, and returns a list of ThemePark objects.

loadAccommodations(): reads the accommodations data file, parses each line, and returns a list of Accommodation objects.

3.2. DataSaver

Description

Handles saving the final booking to a file.

It generates a unique filename using the customer ID and a timestamp, then writes the booking summary.

Relevant Methods

saveCurrentBooking(booking): saves the booking to a .dat file inside the files/ directory, using the booking's toString() output.

3.3. FileConstants

Description

Stores all file-related constants used throughout the application.

This avoids hard-coded strings and ensures consistency across the system.

Relevant Fields

FILES_PATH: base directory for saved files.

FILES_EXTENSION: extension used for saved booking files.

IMG_EXTENSION: extension for image files.

IMG_PATH: base path for images inside the project.

PARKS_FILE: full path to the parks data file.

ACCOMMODATION_FILE: full path to the accommodations data file.

4. Controller Layer

The controller layer acts as the intermediary between the user interface and the logic. It coordinates the flow of data between the UI panels and the service layer, ensuring that the interface never interacts directly with the model or the services.

This separation maintains it clean and keeps the UI free of business logic.

Loads all initial data, stores the current application state (selected park, selected accommodation, reservations, customer, booking), delegates operations to the appropriate service, provides processed data back to the UI, manages the minigame lifecycle, handles saving the final booking, the main class in this layer is AppController.

4.1. AppController

Description

AppController is the central coordinator of the application. It loads data from files, stores the current selections and reservations, communicates with all service classes, and exposes high-level methods used by the UI.

It maintains: Lists of parks and accommodations, the park currently on offer, the selected park and accommodation, ticket and accommodation reservations, customer information, the final booking, the minigame state, it also holds instances of all service classes and delegates operations to them.

Relevant Methods

AppController(): loads parks and accommodations, initializes services, sorts data, and selects the offer park.

chooseOfferPark(): randomly selects the theme park on offer.

setSelectedPark(p): sets the currently selected theme park.

setSelectedAccommodation(a): sets the currently selected accommodation.

resetSelections(): clears both selected park and accommodation.

getAccommodationsForSelectedPark(): returns accommodations belonging to the selected park.

getParksForSelectedAccommodation(): returns the park associated with the selected accommodation.

getTicketPrice(): returns the calculated ticket price.

getAccommodationPrice(): returns the calculated accommodation price.

getDiscount(): returns the offer discount.

getTotalPrice(): returns the final price after discounts.

setTicketReservation(r): stores the ticket reservation.

setAccommodationReservation(r): stores the accommodation reservation.

removeTicketReservation(): removes the ticket reservation.

removeAccommodationReservation(): removes the accommodation reservation.

createBooking(): creates the final booking using all current data.

getReservationSummary(): returns a summary list for the UI.

getBookingSummaryText(): returns the full booking summary text for saving.

setCustomer(c): stores the customer information.

startMiniGame(): resets and starts a new minigame.

revealCell(index): reveals a cell and returns whether it is the winning one.

hasWonMiniGame(): returns whether the user won the minigame.

saveCurrentBooking(): saves the booking to a file.

INTERFACE

The interface layer of the application has been developed using Java Swing. Each screen is implemented as a JPanel, and the main window (MainWindow) dynamically switches between panels using the method switchPanel(). This approach keeps the UI modular, maintainable, and easy to extend.

The interface is fully internationalized, meaning that all labels, buttons, menus, tooltips, and messages change dynamically when the user selects a different language (English/Spanish).

The following sections describe each screen of the application, including:

- A screenshot of the implemented screen
- The corresponding screen from the Wireframe V2
- The most relevant Swing components
- A justification for the design choices

1. MainWindow

Description

MainWindow is the root container of the application. It manages: the menu bar, language switching, help system (JavaHelp), dynamic panel switching, application icon and look & feel.

It is implemented as a JFrame and acts as the main entry point for the UI.

Most Relevant Components

JMenuBar: Contains the Options and Language menus.

JMenu (Options, Language): Groups related actions.

JMenuItem (Help, Exit, English, Spanish): Executes actions such as exiting, opening help, or changing language.

switchPanel(JPanel panel, String helpID): Replaces the current panel with another one.

Application icon (/img/logo.png): Sets the window icon.

FlatLaf Look & Feel: Provides a modern, clean UI style.

Justification of Component Selection

JFrame is the standard container for desktop applications.

JMenuBar + JMenu + JMenuItem provide a familiar, accessible navigation structure for the menu

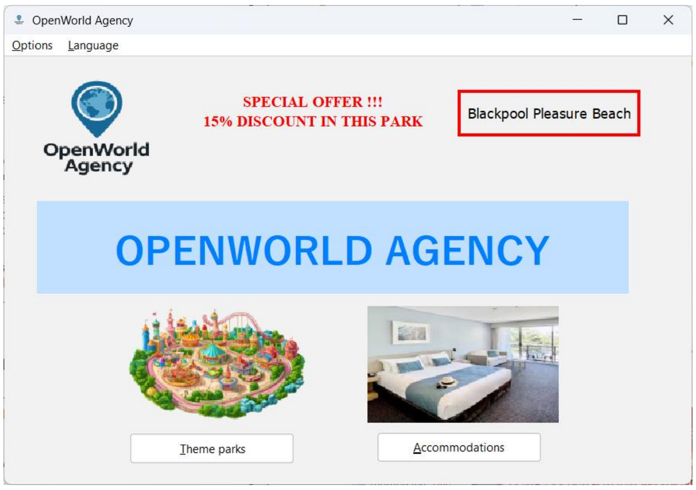
Internationalization is handled at the window level to ensure all panels update consistently.

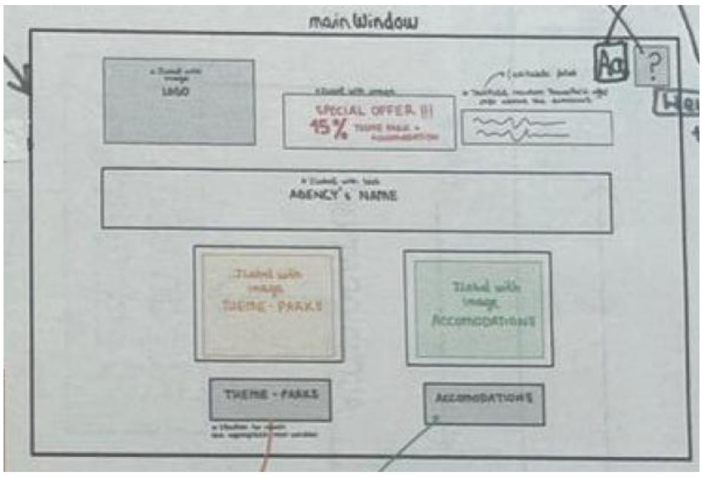
FlatLaf improves visual appearance and usability compared to default Swing look & feel.

JavaHelp integration allows contextual help for every screen.

Dynamic panel switching avoids opening multiple windows and keeps navigation simple.

2. MainMenuPanel





<u>Description</u>

MainMenuPanel is the first screen displayed when the application starts. It serves as the central navigation hub, allowing the user to choose between booking Theme Parks and booking Accommodations.

It also displays: the agency logo, the daily promotion (the park selected as the offer of the day), representative images for each category.

The panel implements the Localizable interface, meaning all text elements update dynamically when the user changes the application language.

<u>Most Relevant Components</u>

1. JLabel lblLogo. Displays the agency logo (/img/logo.png).

2. JLabel lblPromotion. Shows the “Today’s Promotion” text.

3. JLabel lblPromotionPark. Displays the name of the park currently on offer.

4. JLabel lblName. Large banner label showing the application name (“OpenWorld Agency”).

5. JLabel lblParksImg. Image representing theme parks (/img/themepark.png).

6. JLabel lblAccoImg. Image representing accommodations (/img/acc.png).

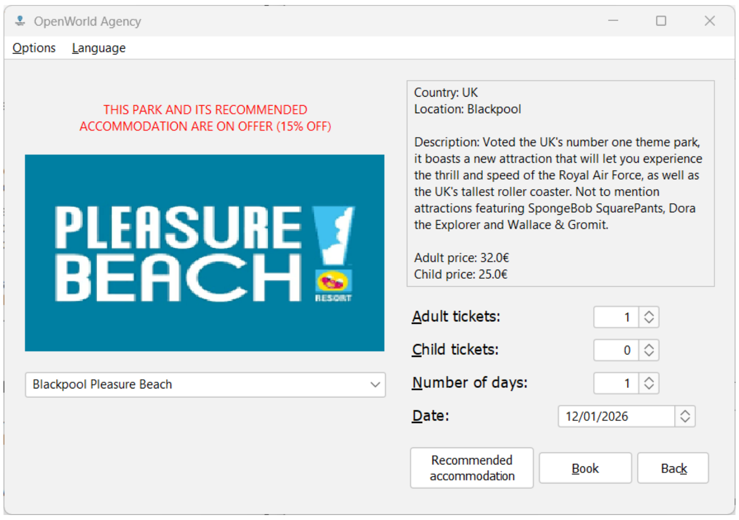
7. JButton btnThemeParks. Navigates to the Theme Parks panel. Includes mnemonic T for accessibility.

8. JButton btnAccommodations. Navigates to the Accommodations panel. Includes mnemonic A for accessibility.

<u>Justification of Component Selection</u>

- Use of Images (JLabel with icons). Images help users quickly understand the two main booking categories. They improve usability and align with the visual style defined in the Wireframe V2.
- Large Title Label. The large lblName creates a strong visual identity and makes the screen feel more professional.
- Promotion Area. The promotion label and bordered park name highlight the daily offer, one of the key features of the application. The red color and border ensure the user notices it immediately.
- Navigation Buttons. Two large buttons provide clear, simple navigation. Mnemonics (Alt+T, Alt+A) improve accessibility and keyboard usability.
- Absolute Layout. The panel uses a null layout to precisely match the positions defined in the Wireframe V2. This ensures the final UI closely resembles the planned design.
- Internationalization Support. All text is loaded through ResourceBundle, allowing the interface to switch languages dynamically without restarting the application.

3. ThemeParksPanel



OpenWorld Agency

THIS PARK AND ITS RECOMMENDED ACCOMMODATION ARE ON OFFER (15% OFF)

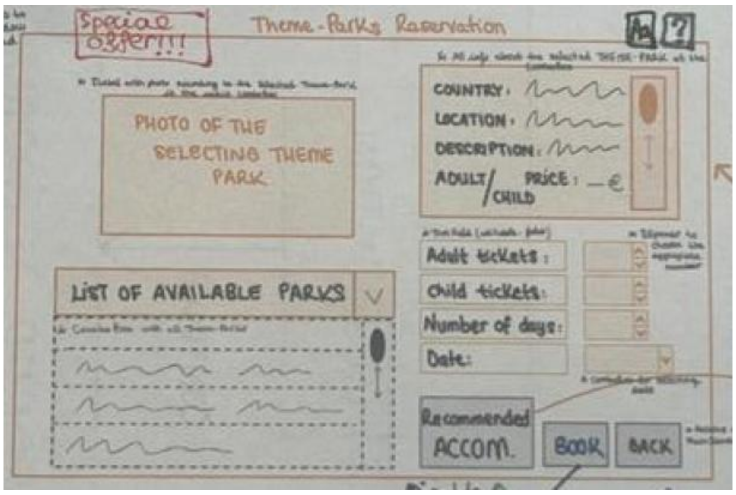
Blackpool Pleasure Beach

Country: UK
Location: Blackpool

Description: Voted the UK's number one theme park, it boasts a new attraction that will let you experience the thrill and speed of the Royal Air Force, as well as the UK's tallest roller coaster. Not to mention attractions featuring SpongeBob SquarePants, Dora the Explorer and Wallace & Gromit.

Adult price: 32.0€
Child price: 25.0€

Adult tickets:	1
Child tickets:	0
Number of days:	1
Date:	12/01/2026



Description

ThemeParksPanel is the screen where the user selects a theme park, views detailed information about it, and configures the ticket reservation (number of adults, children, days, and date).

It is divided into two main sections:

- Left panel: park selection, park image, and offer indicator
- Right panel: park description, ticket configuration fields, and action buttons

This panel implements the Localizable interface, allowing all text to update dynamically when the user changes the language.

Most Relevant Components

1. Park Selection Area (Left Panel)

1. JLabel lblSpecialOfferText. Displays whether the selected park is part of the daily promotion. Shown in red to attract attention.
2. JLabel lblParkImg. Displays the image of the selected park. Images are dynamically loaded using setImageToLabel().
3. JComboBox comboBoxParks. Allows the user to select a theme park. It is alphabetically ordered

2. Park Information Area (Right Panel)

1. JTextArea textAreaInfoPark. Shows the full description of the selected park. The text is generated by the service layer using key-based strings and translated here.
2. JScrollPane scrollPaneInfoPark. Allows long descriptions to be displayed cleanly.
3. JSpinner spinnerAdultTickets. Number of adult tickets (min 1). Mnemonic: A
4. JSpinner spinnerChildTickets. Number of child tickets (min 0). Mnemonic: C
5. JSpinner spinnerNumberDays. Number of days for the visit. Mnemonic: N
6. JSpinner spinnerDate. Visit start date. Uses a SpinnerDateModel with a formatted editor (dd/MM/yyyy). Mnemonic: D
7. Labels (lblAdultTickets, lblChildTickets, lblNumberOfDays, lblDate). Each label is linked to its corresponding spinner using setLabelFor() to improve accessibility.
8. JButton btnRecommendedAccommodation. Creates a ticket reservation and navigates to the AccommodationsPanel. Used when the user wants to book accommodation after selecting a park.
9. JButton btnBook. Creates a ticket reservation and navigates to the ReservationPanel. Used when the user wants to book only park tickets.
10. JButton btnBack. Returns to the MainMenuPanel.

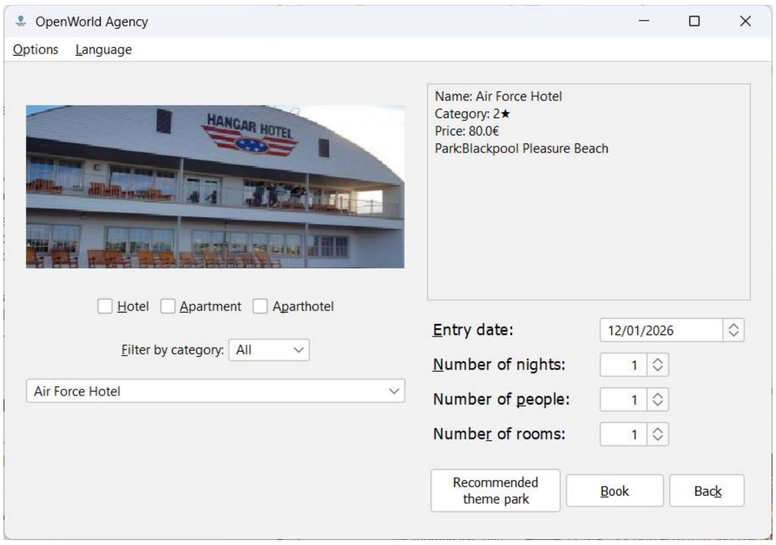
Justification of Component Selection

- Two-panel layout (left + right). Separates visual selection (left) from detailed configuration (right)
- ComboBox for park selection. Efficient for selecting from a list of parks. Automatically displays park names via toString()
- Image label for park preview Provides immediate visual feedback
- TextArea inside ScrollPane. Ideal for long descriptions
- Spinners for numeric input. Prevent invalid values

LAYOUTS

The layout of the ThemeParksPanel is organized using different layout managers, each chosen for a specific purpose. The main panel uses a BorderLayout, which cleanly separates the left and right sections. The left panel uses a vertical BoxLayout, ideal for stacking the offer text, park image, and park selector. The right panel also uses a BorderLayout to divide the park information, the ticket form, and the action buttons. The ticket fields panel employs a GridBagLayout, providing a structured and well-aligned form. Finally, the buttons panel uses a right-aligned FlowLayout, grouping the action buttons following UI conventions.

4. AccommodationsPanel



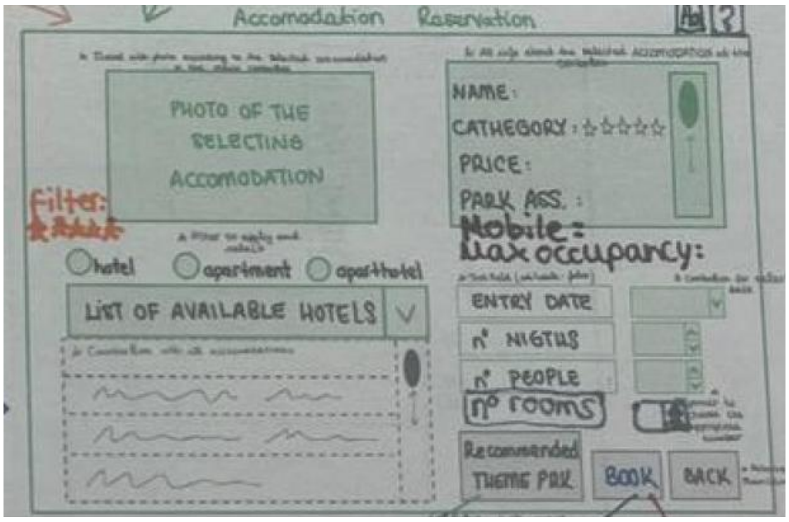
Name: Air Force Hotel
Category: 2★
Price: 80.0€
Park:Blackpool Pleasure Beach

☐ Hotel ☐ Apartment ☐ Aparthotel

Filter by category: All

Air Force Hotel

Entry date: 12/01/2026
Number of nights: 1
Number of people: 1
Number of rooms: 1



Description

AccommodationsPanel allows the user to browse, filter, and select accommodations associated with the previously chosen theme park. It also provides a reservation form where the user can configure: Entry date, number of nights, number of people, number of rooms.

However the attributes of mobile phone and max occupancy that are represented in th etext area were remove because of statement policy of not modifying .dat files.

The panel is divided into two main sections:

- Left panel: image preview, type filters, category filter, and accommodation selector
- Right panel: accommodation description, reservation form, and action buttons

The panel implements the Localizable interface, so all text updates dynamically when the user changes the language.

<u>Most Relevant Components</u>

Left Panel Components:

1. JLabel lblAccImg. Displays the image of the selected accommodation. Images are dynamically loaded using setImageToLabel().
2. JCheckBox chckbxHotel
3. JCheckBox chckbxApartment
4. JCheckBox chckbxAparthotel
5. JComboBox comboBoxCathegory. Allows filtering by star rating (1–5 stars or “All”).
6. JComboBox comboBoxAcc. Displays the list of accommodations that match the selected filters.

Right Panel Components:

1. JTextArea textAreaInfoAcc. Shows the full description of the selected accommodation. Text is generated by the service layer and translated here.
2. JScrollPane scrollPaneInfoAcc. Allows long descriptions to be displayed cleanly.
3. Reservation Form Components. Arranged using a GridBagLayout for a clean, structured form.
4. JSpinner spinnerEntryDate. Entry date for the stay. Uses a SpinnerDateModel with a formatted editor (dd/MM/yyyy).
5. JSpinner spinnerNumberNights
6. JSpinner spinnerNumberPeople
7. JSpinner spinnerNumberRooms
8. Each label is linked to its spinner using setLabelFor() to improve accessibility.
9. JButton btnRecommendedPark. Creates an accommodation reservation and navigates to the ThemeParksPanel.
10. JButton btnBook. Creates an accommodation reservation and navigates to the ReservationPanel.
11. JButton btnBack. Returns to the MainMenuPanel.

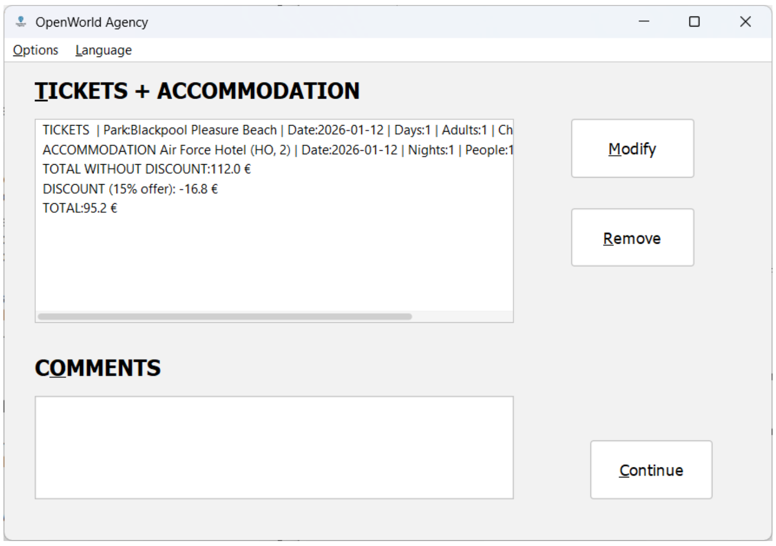
<u>Justification of Component Selection</u>

- Checkboxes and category filters allow intuitive filtering, improving usability.
- Image preview helps users visually identify accommodations.
- Scrollable tExt area is ideal for long descriptions.
- Spinners ensure valid numeric input and prevent errors.
- Buttons with mnemonics improve accessibility.
- Occupancy validation prevents invalid reservations and improves reliability.

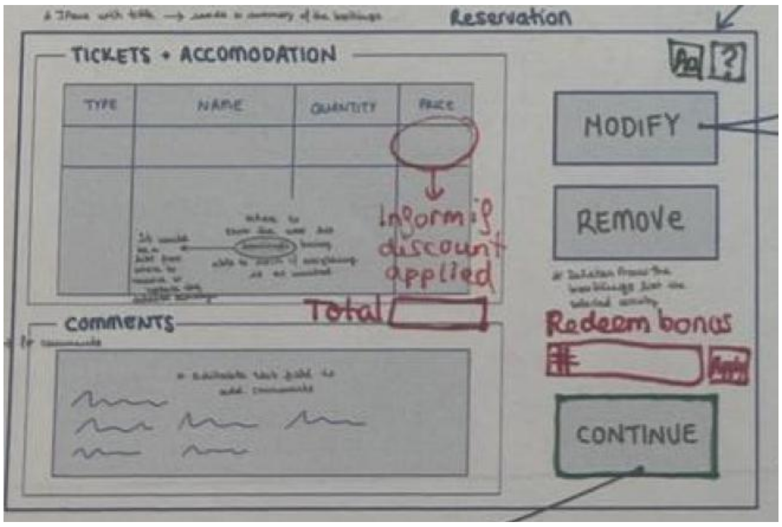
Layout

The layout of the AccommodationsPanel uses several layout managers, each chosen for a specific purpose. The main panel uses a BorderLayout, separating the left and right sections clearly. The left panel uses a vertical BoxLayout, ideal for stacking the accommodation image, type filters, category filter, and selector in a clean column. The right panel also uses a BorderLayout to organize the description area, the reservation form, and the action buttons. The reservation fields panel employs a GridBagLayout, which provides a structured and well-aligned form. The buttons panel uses a right-aligned FlowLayout, grouping the action buttons neatly. The filter panels (checkboxes and category selector) use FlowLayout, which centers the elements and keeps them compact and visually balanced.

5. ReservationPanel



```
TICKETS  | Park:Blackpool Pleasure Beach | Date:2026-01-12 | Days:1 | Adults:1 | Ch
ACCOMMODATION Air Force Hotel (HO, 2) | Date:2026-01-12 | Nights:1 | People:1
TOTAL WITHOUT DISCOUNT:112.0 €
DISCOUNT (15% offer): -16.8 €
TOTAL:95.2 €
```



Description

ReservationPanel displays a complete summary of the user's current selections, including theme park tickets and/or accommodation reservations. It allows the user to: review the reservation details, modify any part of the reservation, remove individual components (tickets or accommodation), add optional comments, proceed to the customer information screen.

This panel uses absolute positioning, allowing full control over the placement of each component to match the Wireframe V2 design precisely.

It also implements the Localizable interface, so all text updates dynamically when the user changes the language.

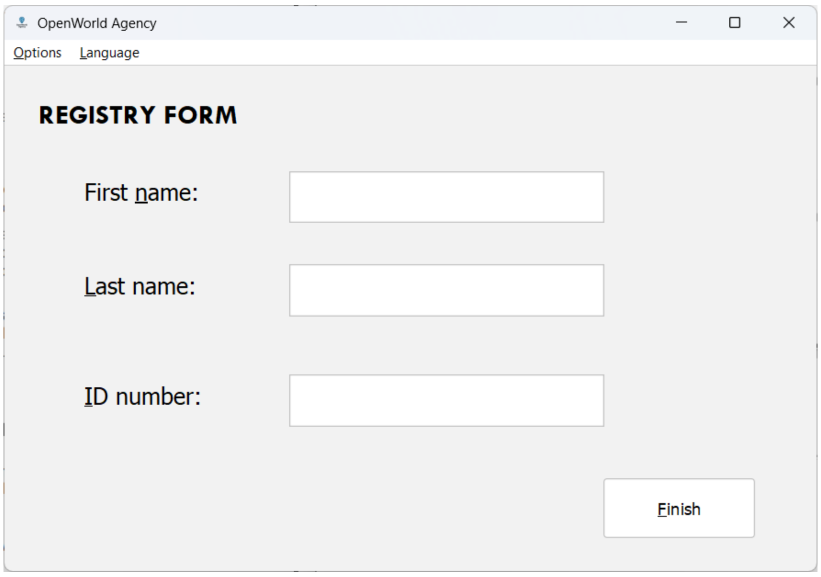
Most Relevant Components

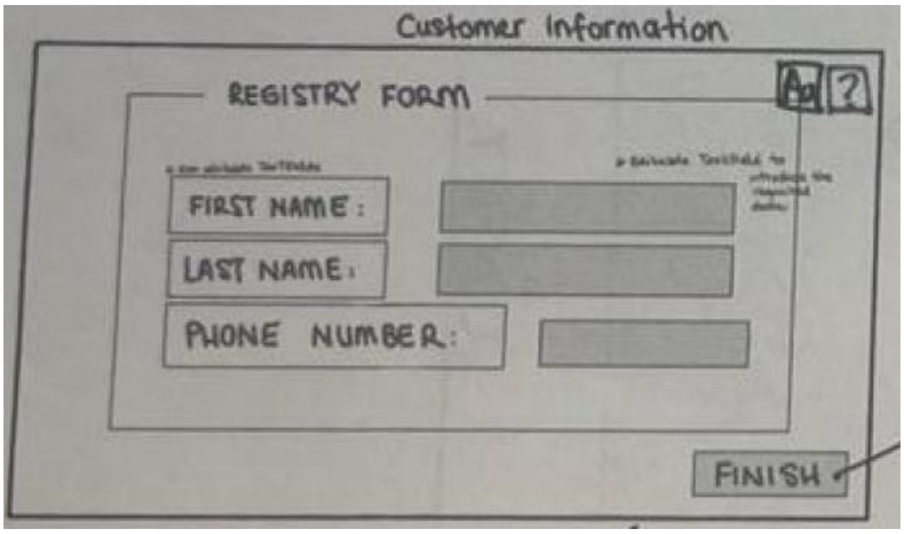
1. JList<String> listSummary. Displays the full reservation summary. The content is generated by the controller and translated dynamically.
2. JScrollPane scrollPane. Allows the summary list to be scrollable, ensuring readability even with long reservations.
3. JLabel lblComments. Label for the comments field. Includes mnemonic and tooltip for accessibility.
4. JTextField textFieldComments. Allows the user to enter optional comments for the booking.
5. JButton btnModify. Allows the user to modify the selected part of the reservation. Behavior depends on the selected list item:If the item corresponds to tickets, the panel switches to ThemeParksPanel. If it corresponds to accommodation, it switches to AccommodationsPanel.
6. JButton btnRemove. Removes the selected reservation component: Removes ticket reservation if the selected item is ticket-related. Removes accommodation reservation if the selected item is accommodation-related. After removal, the panel reloads itself.
7. JButton btnContinue. Moves to the CustomerInformationPanel. Enabled only if at least one reservation exists.

Justification of Component Selection

- JList inside a JScrollPane provides a clean and readable summary of the reservation.
- Modify and Remove buttons give the user full control over their selections.
- Continue button ensures the user cannot proceed without having at least one reservation.
- Comments field allows personalization of the booking, improving user experience.
- Absolute layout ensures the final UI matches the exact structure defined in the Wireframe V2.

6. CustomerInformationPanel





Description

CustomerInformationPanel is the screen where the user enters their personal information before finalizing the booking. It collects: First name, last name, IID number (DNI format).

However, in the wireframe because of an error while we read we misunderstood ID number with phone number, so in the UI we have to change it.

This information is required to create the final Customer object and complete the booking process.

The panel uses absolute positioning, allowing precise placement of labels, text fields, and the Finish button to match the Wireframe V2 design.

It also implements the Localizable interface, so all labels, tooltips, and button texts update dynamically when the user changes the language.

Most Relevant Components

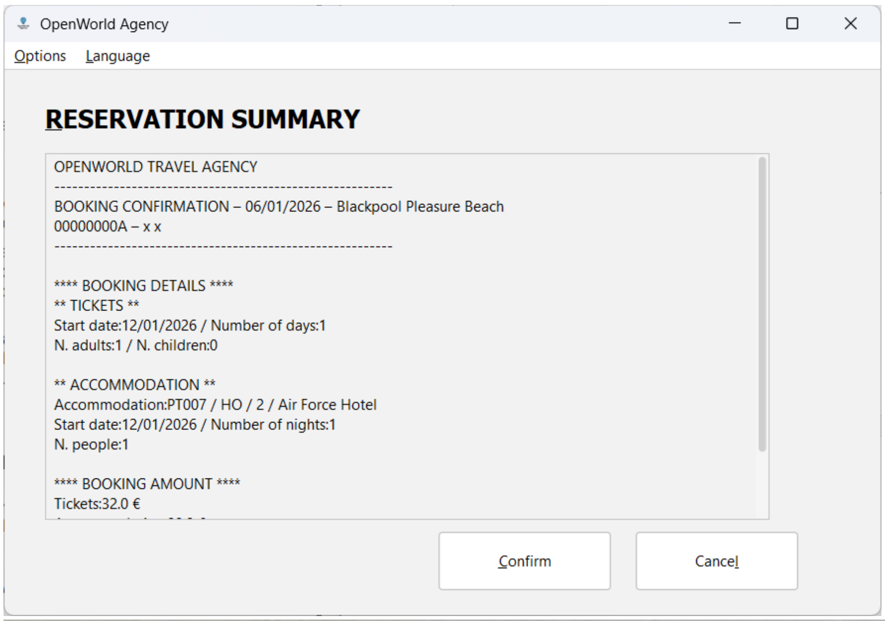
1. JLabel lblFirstName + JTextField textFieldFirstName. Field for entering the customer's first name. Includes mnemonic and tooltip for accessibility.
2. JLabel lblLastName + JTextField textFieldLastName. Field for entering the customer's last name.
3. JLabel lblIDNumber + JTextField textFieldIdNumber. Field for entering the customer's ID number (DNI). The DNI is validated using a regular expression (\\d{8}[A-Za-z]).
4. JLabel lblTitle. Displays the title of the screen ("Customer Information"). Uses a large, bold font to create visual hierarchy.
5. JButton btnFinish. When clicked; validates all fields: no empty fields, first and last name contain only letters, ID number matches DNI format. Creates a new Customer object. Stores it in the controller. Navigates to the ReservationSummaryPanel. The button includes a mnemonic (F) and tooltip.

Justification of Component Selection

- Text fields are the most intuitive and appropriate components for entering personal information.
- Labels with mnemonics improve accessibility and keyboard navigation.
- Validation with dialog messages ensures data integrity and prevents invalid bookings.

- A single Finish button keeps the interface simple and focused.
- Absolute layout allows the panel to match the exact structure defined in the Wireframe V2.

7. ReservationSummaryPanel



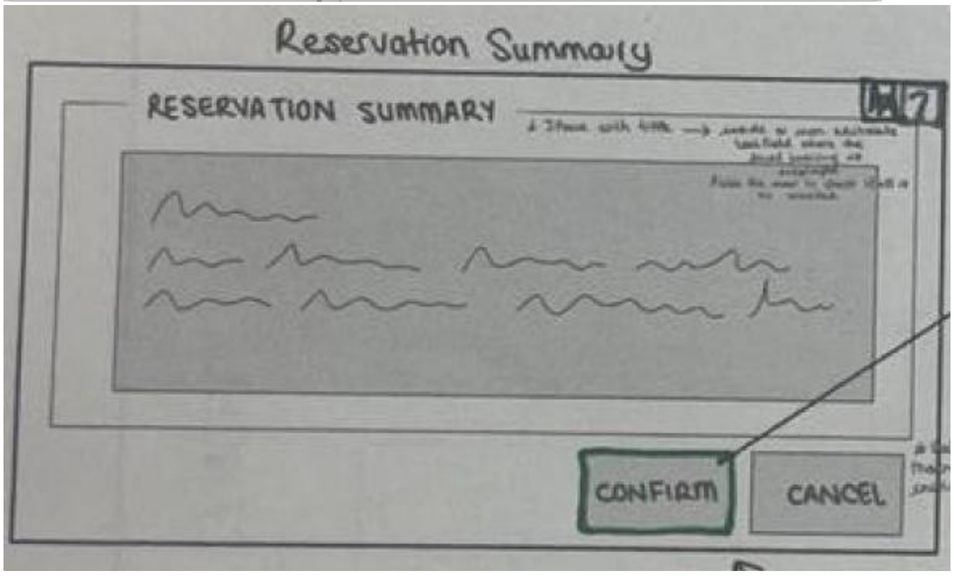
RESERVATION SUMMARY

```
OPENWORLD TRAVEL AGENCY
-------------------------------------------------------
BOOKING CONFIRMATION – 06/01/2026 – Blackpool Pleasure Beach
00000000A – x x
-------------------------------------------------------

**** BOOKING DETAILS ****
** TICKETS **
Start date:12/01/2026 / Number of days:1
N. adults:1 / N. children:0

** ACCOMMODATION **
Accommodation:PT007 / HO / 2 / Air Force Hotel
Start date:12/01/2026 / Number of nights:1
N. people:1

**** BOOKING AMOUNT ****
Tickets:32.0 €
```



Description

ReservationSummaryPanel displays the complete summary of the booking before the user proceeds to the final confirmation step.It shows all relevant information: customer details, ticket reservation details, accommodation reservation details, total price and discounts, the summary is presented in a scrollable text area, allowing the user to review the entire booking comfortably.

The panel uses absolute positioning, ensuring that the final layout matches the Wireframe V2 design precisely.

It also implements the Localizable interface, so all labels, tooltips, and button texts update dynamically when the user changes the language.

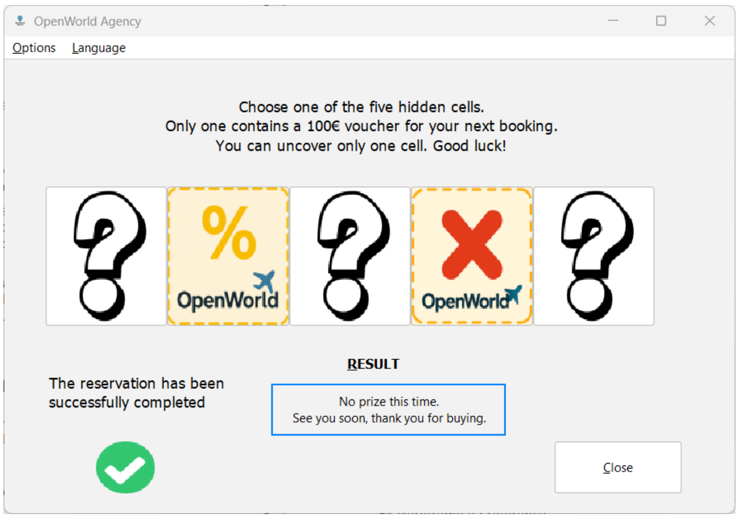
Most Relevant Components

1. JTextArea textAreaSummary, Displays the full booking summary as plain text. It is: Non-editable, scrollable, automatically translated using the custom translate() method
2. JScrollPane scrollPane. Allows the summary to be scrollable, ensuring readability even for long bookings.
3. JLabel lblTitle.. Title of the screen ("Reservation Summary"). Uses a bold font to create visual hierarchy.
4. JButton btnConfirm. When clicked: proceeds to the MiniGamePanel, where the user can play the optional minigame before finalizing the booking. Includes mnemonic (C) and tooltip.
5. JButton btnCancel. Cancels the entire booking process: Calls controller.resetAllReservations(). Returns to the MainMenuPanel. Includes mnemonic (L) and tooltip.

Justification of Component Selection

- A scrollable text area is ideal for displaying long summaries in a readable format.
- Confirm and Cancel buttons provide a clear decision point for the user.
- Absolute layout ensures the panel matches the exact structure defined in the Wireframe V2.
- Mnemonics and tooltips improve accessibility and usability.
- Translation of the summary text ensures full internationalization support.

8. MiniGamePanel

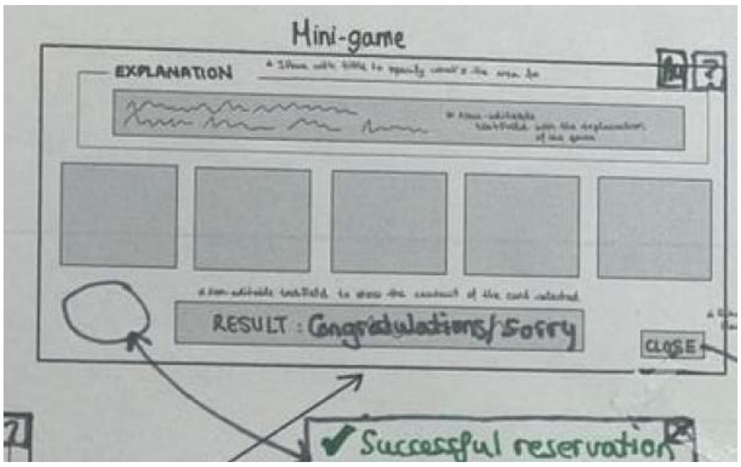


Choose one of the five hidden cells.
Only one contains a 100€ voucher for your next booking.
You can uncover only one cell. Good luck!

RESULT

The reservation has been successfully completed

No prize this time.
See you soon, thank you for buying.



Description

MiniGamePanel presents the optional minigame that the user can play before finalizing the booking. The game consists of several hidden cells; the user clicks one of them to reveal whether they have won a gift voucher. Only one attempt is allowed.

The panel uses absolute positioning for all elements except the grid of buttons, which is handled internally by a GridLayout.This allows the final UI to match the Wireframe V2 design precisely.

The panel implements the Localizable interface, so all labels, tooltips, and button texts update dynamically when the user changes the language.

Most Relevant Components

1. JLabel lblExplanation. Displays the instructions for the minigame. Centered and styled with a readable font.
2. JPanel panelButtons. Contains the clickable cells of the minigame. Each cell is represented by a JButton with an initial hidden image.
3. Dynamic Button Creation. In initialize():The controller is asked for the number of cells. A button is created for each cell. Each button stores its index using putClientProperty("index", i). Clicking a button triggers the reveal logic. When a cell is clicked: All cells update their icons to show their true content,the result label displays either a win or lose message. If the user wins, the booking is marked with a gift voucher
4. JLabel lblResult. Displays the outcome of the minigame. Styled with a colored border to highlight the result.
5. JLabel lblConfirmation. Displays a confirmation message related to the booking.
6. JLabel lblTickImage. Shows a tick icon to reinforce the confirmation visually.
   JButton btnClose. When clicked: Saves the booking to file, resets all reservations, resets the minigame, returns to the MainMenuPanel. Includes mnemonic (C) and tooltip.

Justification of Component Selection

- Buttons as hidden cells provide an intuitive and interactive minigame experience.
- A grid layout inside the button panel ensures the cells are evenly spaced and visually consistent.
- Result labels and tick icon give immediate feedback to the user.
- Close button provides a clear exit point after the game.
- Absolute layout allows precise placement of all elements to match the Wireframe V2 design.
- Dynamic image scaling ensures icons always fit the button size.

TESTING

This section documents the results obtained after executing the seven validation scenarios defined during the wireframe phase. Each scenario was tested once the full module development was completed.

For every person-scenario, the following aspects were evaluated:

- Whether the interface supported the expected user flow
- Whether the user could complete the scenario without confusion
- Whether any usability or functional issues appeared
- Whether the final implementation matched the wireframe intentions
- Whether any errors required correction

If an issue was detected, the scenario, the error, and the implemented solution are documented below.

1. Beck

Scenario Summary

Beck wants to choose a high-quality hotel for her anniversary trip. She tries to filter accommodations by star rating.

Issue Detected

- The initial version of the interface did not include a category (star rating) filter, making it difficult for her to quickly locate high-quality hotels.

Solution Implemented

✓ A category filter was added to the Accommodations Panel.
✓ Users can now filter accommodations by 1–5 stars or select "All".

2. Gabriel

Scenario Summary

Gabriel, an elderly user, values readability and clarity. He also expects accommodations to include a phone contact.

Issues Detected

- No phone number field existed for accommodations.
- Text size was small for elderly users.
- Some parts lacked icons or tooltips.

Solution Implemented

✓ Tooltips were added to all buttons.
✓ Icons were added where appropriate (e.g., tick icon, images in minigame).
✓ Font sizes were increased in key labels and titles.

Not Implemented (Justification)

✗ Adding a phone number attribute to accommodations was not allowed, because:

The .dat files cannot be modified.The statement explicitly forbids adding new attributes or extra functionalities. This limitation is documented and justified.

3. Macarena

Scenario Summary

Macarena relies heavily on the help system before booking.

Issues Detected

- Help was only accessible from the Main Window.
- No category filter existed for accommodations.

Solutions Implemented

✓ Help access was added to every panel using hb.enableHelpKey(panel, helpID, hs).

✓ Category filter implemented (same fix as Beck's scenario).

✓ Short explanations were added to tooltips to improve clarity.

4. Antonio

Scenario Summary

Antonio books for himself and his three children and wants to understand occupancy and discounts clearly.

Issues Detected

- Maximum occupancy was not clearly displayed.
- Discount information on the main screen was not prominent.

Solutions Implemented

- ✓ Maximum occupancy, could not be display in the textArea as an attribute of accommodation because we cannot modify the attributes of the class accommodation because then it cannot pass the tests in the teachers correction. However it is implemented in the logic and is shown clearly in the accommodation reservation a JOptionPane is displayed informing if the capacity is full.
- ✓ The discount label on the Main Menu was highlighted using color and border emphasis, also is indicated when selected the theme park and in the summary there are lines indicating the discount applied.

5. Eve

Scenario Summary

Eve books discounted tickets and wants to verify that the discount was applied correctly

Issue Detected

- There was no explicit confirmation message indicating that the discount had been applied.

Solution Implemented

- ✓ A confirmation message was added to the Reservation and Reservation Summary Panel, clearly stating when a discount has been applied.

6. Manuel

<u>Scenario Summary</u>

Manuel books for a large family and needs flexibility with rooms and clear confirmation messages.

<u>Issues Detected</u>

- No warning appeared when exceeding accommodation capacity.
- No option existed to select number of rooms.
- No confirmation message appeared after completing the booking.

<u>Solutions Implemented</u>

- ✓ A capacity warning dialog was added using JOptionPane.
- ✓ A number of rooms spinner was added to the Accommodations Panel.
- ✓ A final confirmation label was added after finishing the booking in the minigame.

7. Marga

<u>Scenario Summary</u>

Marga wins a voucher in the minigame and wants to redeem it during the booking process.

<u>Issue Detected</u>

- There was no option to redeem the voucher code.

<u>Not Implemented (Justification)</u>

- ✗ A voucher redemption field was not added, because: The statement explicitly forbids adding extra functionalities not included in the base specification. The wireframe originally included this idea, but it had to be removed to comply with the assignment rules.

<u>Implemented Instead</u>

- ✓ The minigame still awards a gift voucher flag, which is applied automatically to the booking (as allowed by the statement).
- ✓ The user is clearly notified of the prize.

Errors detected

Some suggestions from testers were not implemented intentionally, due to assignment constraints:

- ✗ Adding new attributes (e.g., phone number or max occupancy in accommodations). Would break .dat compatibility rule.
- ✗ Adding a voucher redemption field. Although planned in the wireframe, it was removed because the statement forbids adding new features beyond the required ones.

These decisions are documented to justify why certain usability improvements were not included.

DECLARATION OF THE USE OF ARTIFICIAL INTELLIGENCE TOOLS

This chapter documents the use of Artificial Intelligence tools during the development of this module.

1. Tool

Microsoft Copilot

2. Purpose of Use

Microsoft Copilot was used exclusively for documentation-related tasks, help html tasks, clarification of Java concepts, and assistance with internationalization. It was not used to generate new functionalities, modify the .dat files, or write core logic of the application.

The purposes were:

- Formatting and organizing documentation text
- Translating interface texts for .properties files (internationalization)
- Clarifying doubts about Java Swing, event handling, and utility methods
- Helping rewrite or format code without altering its logic
- Assisting in writing the html's in JavaHelp
- Rewrite classes for readability (spacing, indentation) without modifying the code.

3. Prompts or Instructions Provided

Below is a representative list of prompts used during the project.

Documentation and Formatting

- "Rewrite this documentation in a formal academic style."
- "Organize this text into sections and improve clarity."
- "Generate a professional description of this panel."
- "Help me write the Testing Results Report based on these scenarios."
- "Help writing the html help for this class"

Code Formatting (no logic changes)

- "Format this Java class without modifying any logic."
- "Improve indentation and spacing of this class."
- "Explain what this method does"

Internationalization

- "Translate these UI strings into English for a .properties file."
- "Generate the Spanish version of these keys."
- "Help me create consistent tooltip texts for all components."

Technical Clarifications

- "Explain how SpinnerDateModel and the editor works."
- "What can I do a translate method for the keys to translate them in the UI?"
- "How do I correctly scale an image in a JLabel?"
- "Is this filter().stream() usage correct?"

Specific Methods Discussed

- translate(String raw, ResourceBundle texts)
- setImageToLabel(JLabel label, String path)
- applyFilters() in AccommodationsPanel
- checkOccupancy()
- buildReservation()
- initialize() in MiniGamePanel
- onCellClicked(int index)
- fillSummary() in ReservationPanel

These methods were explained, not rewritten.

4. Use of the Generated Content

The content generated by Microsoft Copilot was integrated into the project in the following ways:

- ✓ Adapted to fit the style of the document. Most AI-generated text was rewritten, expanded, or adapted to match the tone and structure of the final documentation.
- ✓ Used as a basis and then reviewed and edited. Copilot provided initial drafts for: Panel descriptions, testing report, personas summaries, internationalization explanations

These drafts were manually revised and edited before inclusion.

- ✓ Translations. Some .properties translations were used directly when they required no further modification.
- ✗ Not used to generate application logic. No controller logic, model classes, or functional code was generated by AI. All logic was implemented manually to comply with the assignment rules.

5. Specific Parts of the Work Generated with AI

The following parts of the documentation were created with the assistance of Microsoft Copilot:

<u>Documentation Sections</u>

Full descriptions of:

- MainMenuPanel
- ThemeParksPanel
- AccommodationsPanel
- ReservationPanel
- CustomerInformationPanel
- ReservationSummaryPanel
- MiniGamePanel
- Testing Results Report
- Personas and Scenarios summaries
- Justification of UI components
- Final conclusions and summaries

Internationalization

Draft versions of:

- texts_en.properties
- texts_es.properties
- Tooltip descriptions
- Button labels
- Error messages

Code Formatting

Reformatting of:

- translate() method
- applyFilters()
- checkOccupancy()
- initialize() in MiniGamePanel
- fillSummary() in ReservationPanel
- Adding spacing, indentation.
- scaleImageToLabel()
- toString() with a defined model in the statement
- buildReservationSummary() in BookingService
- buildBookingSummaryText() in BookingService
- swicthPanel() in MainWindow
- IsOnlyLetter() in Customer Infor Panel
- IsValidID() in Customer Info Panel

Technical Clarifications

AI was used to clarify:

- How to use SpinnerDateModel
- How to use GridBagConstraints
- How it works stream(), sorted(), ToList(), filter() methods