KUALA LUMPUR UNIVERSITY
OF SCIENCE AND TECHNOLOGY
吉隆坡科技大学

CSC 1514-PROGRAMMING FUNDAMENTAL
OCTOBER 2026

Project – group project

NAME	1: Ismail Abdirisaq Abdiqani 2: MONIR SAAD BIN 3: LARBAOUI YASSINE 4: MD MASHRAFI SARKER MOIN 5: Hassan Osama Ibrahim Ali
MATRIC NO.	1: 2: 3: 4: 5:
DATE	28/10/2026

Table of Contents

1. Project Overview
2. Project Idea and Background
3. Problem Statement
4. Aim and Objectives
5. Project Scope
6. Users and Functionalities
7. System Requirements
8. System Design and Program Flow
9. User Interface Screens
10. Testing and Validation
11. Limitations and Future Enhancements
12. Team Contribution
13. Conclusion

1. Project Overview

GlampEase is a Java Swing desktop application prototype that demonstrates a front-end reservation and payment process for a glamping service. It guides a customer from accommodation selection through entering booking details, reviewing a calculated total, selecting a simulated payment method, and viewing a confirmation screen.

The project demonstrates GUI programming, screen navigation, basic input validation, transfer of booking information between forms, and presentation of a booking summary and receipt. Payment is simulated; the application does not connect to a bank or transfer real money, in line with the project specification.

Item	Description
Application type	Java desktop GUI prototype
Technology	Java, Swing GUI forms, NetBeans
Main flow	Room selection → Booking details → Payment → Confirmation
Shared data	Booking class stores the current booking while the application runs
Payment	Simulated only; no real financial transaction

2. Project Idea and Background

Glamping combines an outdoor stay with selected comforts of conventional accommodation. Customers need to select an accommodation option, enter guest and stay details, understand the estimated cost, choose a payment method, and receive a confirmation. GlampEase models these steps in a connected desktop interface.

The group selected glamping reservations as the scenario for the required front-end reservation and payment system. The aim is to create a clear, easy-to-follow demonstration rather than a production booking platform.

3. Problem Statement

A reservation process can become confusing when accommodation choices, guest details, stay duration, price calculation, payment selection, and confirmation are presented unclearly or across disconnected screens. Input mistakes can also prevent a booking from being processed correctly. A front-end prototype can address these concerns through a guided screen sequence, a readable booking summary, basic validation, and a confirmation view.

GlampEase addresses this design problem by connecting the customer screens and displaying key booking information before and after the simulated payment step. It does not claim to provide live room availability, permanent booking storage, or real payment processing.

4. Aim and Objectives

Aim: To develop a professional Java GUI prototype demonstrating a clear glamping reservation, simulated payment, and confirmation process.

Objectives:

- Present accommodation options to the customer.
- Collect the customer's name and number of nights, and display the guest count assigned from the selected accommodation capacity.
- Calculate and display the booking total using the selected accommodation rate and stay duration.

- Validate basic inputs and provide helpful error messages.
- Pass booking information between booking, payment, and confirmation screens.
- Require selection of a payment method before proceeding.
- Display a confirmation screen containing the booking details and total.
- Organize the program using separate GUI classes and a shared booking-data class.

5. Project Scope

5.1 Included

- Room/accommodation selection interface.
- Booking form for customer name and nights, with the selected accommodation and fixed guest count displayed.
- Booking summary and price calculation.
- Basic validation for an empty customer name and non-numeric night values.
- Payment screen with selectable payment options.
- Simulated payment-success message and confirmation screen.
- Confirmation view with customer, accommodation, guests, nights, and total.
- Navigation back to booking and room selection.

5.2 Outside current scope

- Real bank, card, or e-wallet processing.
- Database-backed storage or permanent history of multiple bookings.
- Live accommodation availability or inventory management.
- Verified user accounts or an administrator login/dashboard.
- Email/SMS notifications or online deployment.

The scope is a desktop demonstration of one reservation at a time, with simulated payment and an on-screen confirmation.

6. Users and Functionalities

6.1 Primary user

The primary user is a customer who wants to make a glamping booking. The customer uses the GUI to select accommodation, enter booking details, choose a simulated payment method, and view the confirmation.

6.2 Functionalities

ID	Functionality	Expected behaviour
F01	Select accommodation	Room displays four accommodation options with capacities and nightly prices.
F02	Enter customer name	Booking form accepts the customer's name.
F03	Enter nights and view guests	Customer enters nights; guest count is set to the selected room capacity and is read-only.
F04	Choose accommodation type	Room selection supplies the accommodation and nightly rate; the dropdown in Main is locked.
F05	Calculate total	Rate multiplied by nights is shown in the booking summary.

ID	Functionality	Expected behaviour
F06	Validate input	Missing name or invalid numeric input triggers a message.
F07	Review booking summary	Customer reviews name, accommodation, guests, nights and total before payment.
F08	Select payment method	Customer must select an available option before proceeding.
F09	Simulate payment	Application shows a success message without real money transfer.
F10	Show confirmation	Confirmation displays customer and booking details with the total.
F11	Navigate between screens	Back actions open the relevant previous/start screen.

7. System Requirements

7.1 Hardware

- Desktop or laptop capable of running a supported Java runtime and NetBeans.
- Keyboard and pointing device for data entry and GUI interaction.
- Sufficient memory and storage for the IDE, source files, and runtime.

7.2 Software

Component	Purpose
Java Development Kit (JDK)	Compiles and runs the Java application.
Apache NetBeans IDE	Opens the project, edits forms, compiles, and supports testing.
Java Swing	Builds desktop windows, labels, text fields, buttons, and selection controls.
Booking class	Holds current booking values while moving between screens.

7.3 Non-functional considerations

Quality	Design consideration
Usability	Clear labels, consistent sequence, readable booking information.
Reliability	Validate basic input before calculating or proceeding.
Maintainability	Separate screen classes and centralize shared values in Booking.
Accuracy	Calculate the total from the selected rate and number of nights.
Transparency	Clearly state that payment is simulated.

8. System Design and Program Flow

8.1 Main flow

1. Customer opens Room and continues to the booking form.
2. Main loads the accommodation selected in Room, sets the guest count to its capacity, and collects the customer name and number of nights.
3. When Calculate is selected, the application validates input and displays a summary.
4. When the customer reserves, booking values are copied into the shared Booking class.
5. Payment reads the saved values and displays the booking summary.
6. Customer selects a payment method and chooses Pay Now.
7. The application checks the selection, displays a simulated success message, and opens Confirmation.
8. Confirmation displays booking details and total; Back to Home returns to Room.

8.2 Main classes

Class	Responsibility
Room	Starting point and accommodation-selection screen.
Main	Booking form, input checks, price calculation, and transition to Payment.
Booking	Shared static fields for room, price, capacity, name, guests, nights, breakfast flag, and total.
Payment	Displays booking details, requires a payment-method selection, and simulates payment.
Confirmation	Displays booking details after simulated payment and provides Back to Home navigation.

8.3 Price calculation

The nightly rates are Deluxe Tent — RM180; Luxury Cabin — RM300; Family Glamping — RM400; Single Tent — RM100. These rates match the room-selection handlers and booking calculation branches. The guest count does not multiply the nightly price.

Total stay cost = Nightly accommodation rate × Number of nights

8.4 Data handling

The Booking class uses static fields so the current booking can be read by multiple screens. This supports one active booking moving through the demonstration flow. Data is held in memory and is not permanently saved, so this implementation does not provide database-backed booking history.

8.5 Payment behaviour

Pay Now checks whether a payment option is selected. If neither option is selected, a validation message is shown. If an option is selected, the application displays “Payment successful!” and opens Confirmation. This is a simulated result, not a real payment gateway integration.

8.6 Code Connecting the GUI Screens

The listings below are excerpts from the project source. Generated GUI layout code is omitted so that the screen connections and shared data are easy to follow. The complete Java project remains the source-code deliverable.

Listing 1 Shared booking data

```
package glamping;
public class Booking {
    // "static" means there is only one shared copy, so every page can use it
    public static String room = "";
    public static int price = 0;
    public static int capacity = 0;
    public static String name = "";
    public static int guests = 0;
    public static int nights = 0;
    public static boolean breakfast = false;
    public static double total = 0;
}
```

Static fields provide one shared copy of the active reservation. Room sets accommodation values, Main stores customer and stay values, and later screens read them. The breakfast field is declared but is not used in the current calculation.

Listing 2 Room selection and transition to Main

```
private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {
Booking.room = "Single Tent";
Booking.price = 100;
Booking.capacity = 1;
javax.swing.JOptionPane.showMessageDialog(this, "Single Tent selected");
    }
    private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
if (Booking.room.equals("")) {
    javax.swing.JOptionPane.showMessageDialog(this, "Please choose a room first.");
} else {
    new Main().setVisible(true);
    this.dispose();
}
```

The Single Tent handler stores the room name, RM100 rate and capacity of one. Continue blocks an empty room value; otherwise new Main() constructs the booking form, setVisible(true) displays it and dispose() closes Room.

Listing 3 Loading the selected accommodation

```
public Main() {
    initComponents();
    setAccommodation(Booking.room);
}
// In setAccommodation after matching the room in the dropdown:
jComboBox1.setEnabled(false);
jTextField3.setText(String.valueOf(Booking.capacity));
jTextField3.setEditable(false);
```

The constructor creates the controls before loading the room. The selected room is matched in the dropdown, and capacity becomes the fixed guest count. The final listing shows selected statements from setAccommodation.

8.7 Calculation and Reservation Code

Listing 4 Calculation and basic validation

```
String name = jTextField1.getText();
if (name.equals("")) {
    javax.swing.JOptionPane.showMessageDialog(this,
        "Please enter the customer name.");
    return;
}
// Inside the try block:
int guests = Integer.parseInt(jTextField3.getText());
int nights = Integer.parseInt(jTextField4.getText());
// The selected index determines price and room.
int total = price * nights;
```

The Calculate handler rejects an empty name and parses guest and night values as integers. It maps the selected accommodation to its rate, multiplies rate by nights, and writes the summary. A NumberFormatException handler displays the numeric-input message. The listing shows selected calculation statements; price and room are assigned by the intervening selection branches.

Listing 5 Saving details and opening Payment

```
// Make sure Calculate was clicked
if (jTextField5.getText().equals("Booking Summary")) {
    javax.swing.JOptionPane.showMessageDialog(
        this, "Please click Calculate first.");
    return;
}
try {
    // Save booking details in the shared Booking class
    Booking.name = jTextField1.getText().trim();
    Booking.guests = Integer.parseInt(jTextField3.getText().trim());
    Booking.nights = Integer.parseInt(jTextField4.getText().trim());
```

The Reserve handler first checks the summary placeholder, then saves the current input. Its accommodation branches assign Booking.room and Booking.price before the following transition:

```
Booking.total = Booking.price * Booking.nights;
// Open the Payment page
Payment paymentPage = new Payment();
paymentPage.setLocationRelativeTo(null);
paymentPage.setVisible(true);
// Close the booking page
this.dispose();
```

The total is stored before Payment is constructed. This order lets the Payment constructor read the completed shared values immediately. Centering the window with setLocationRelativeTo(null) improves its initial placement.

8.8 Payment and Return Navigation Code

Listing 6 Loading the payment summary

```
public Payment() {
    initComponents();
    loadBookingSummary();
}
private void loadBookingSummary() {
    lblCustomerName.setText(Booking.name);
    lblRoom.setText(Booking.room);
    lblNights.setText(
        Booking.guests + " guests / " + Booking.nights + " nights"
    );
    lblTotal.setText(String.format("RM %.2f", Booking.total));
}
```

After component initialization, loadBookingSummary() fills the labels from Booking. String.format uses two decimal places for the monetary amount.

Listing 7 Payment validation and confirmation

```
if (!rbcard.isSelected() && !rbWallet.isSelected()) {
    javax.swing.JOptionPane.showMessageDialog(
        this,
        "Please select a payment method."
    );
    return;
}
javax.swing.JOptionPane.showMessageDialog(
    this,
    "Payment successful!"
);
Confirmation confirmationPage = new Confirmation();
confirmationPage.setLocationRelativeTo(null);
confirmationPage.setVisible(true);
this.dispose();
```

If neither method is selected, return stops the handler. Otherwise a dialog announces simulated success, Confirmation is created and displayed, and the Payment window closes. The payment choice is not stored in Booking.

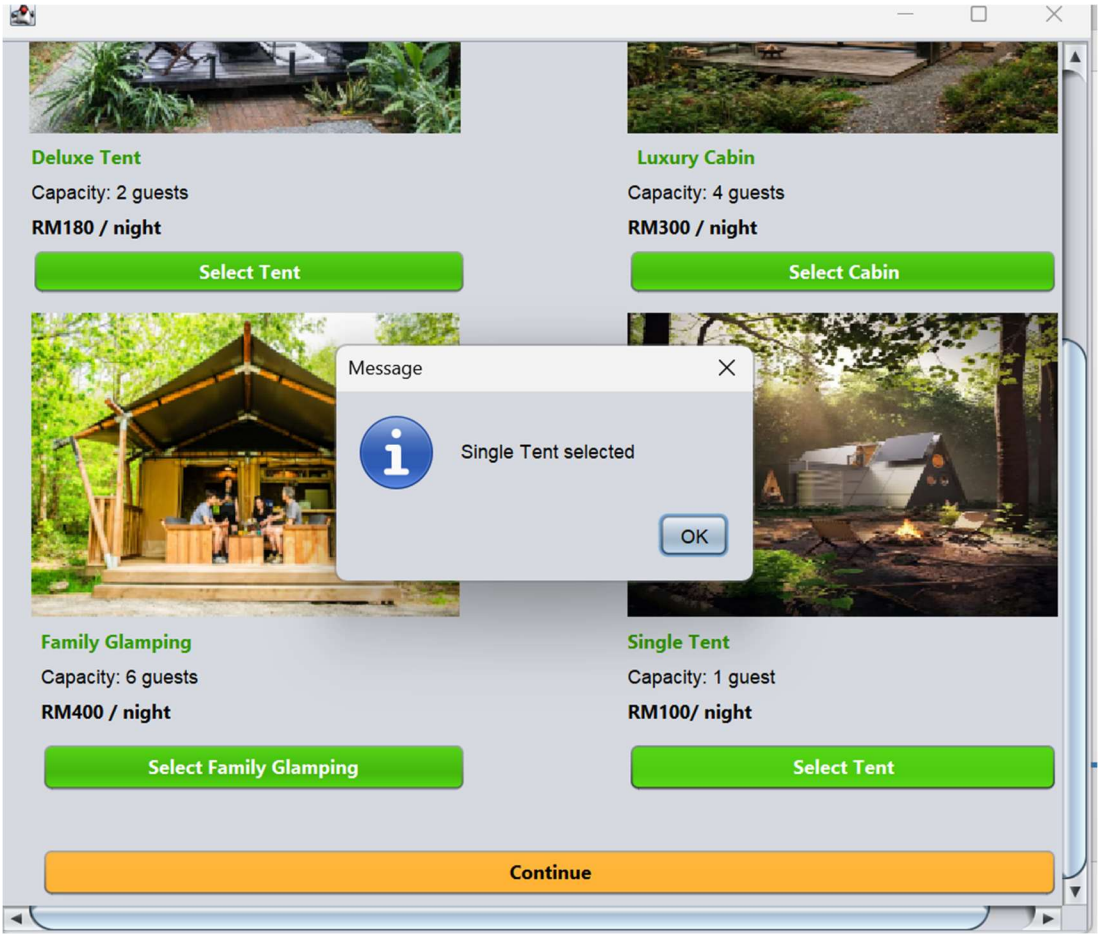
Listing 8 Loading confirmation and returning home

```
private void loadBookingDetails() {
    lblCustomerName.setText(Booking.name);
    lblRoom.setText(Booking.room);
    lblGuests.setText(String.valueOf(Booking.guests));
    lblNights.setText(String.valueOf(Booking.nights));
    // Display the actual total beside "Total Paid"
    jLabel17.setText(String.format("RM %.2f", Booking.total));
}
Room roomPage = new Room();
roomPage.setLocationRelativeTo(null);
roomPage.setVisible(true);
this.dispose();
```

Confirmation reads the same shared values and writes the amount beside Total Paid. Back to Home explicitly creates Room, so the customer returns to accommodation selection.

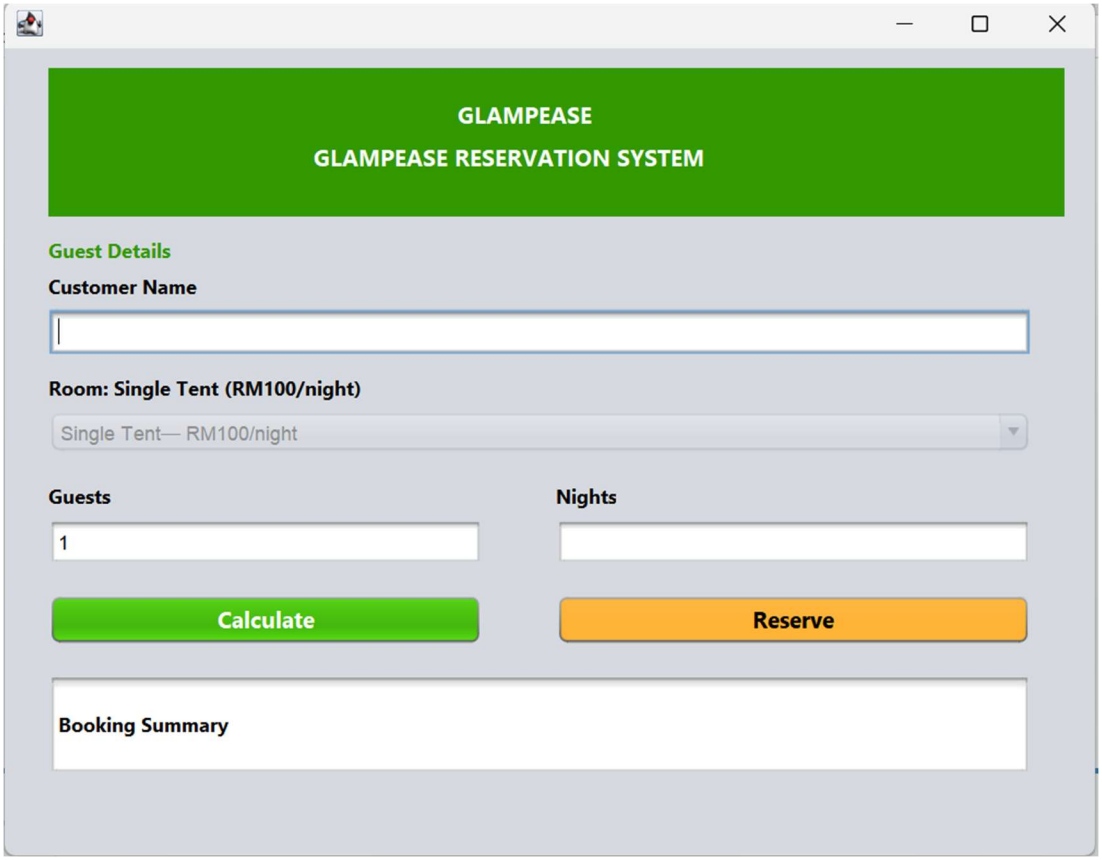
9. User Interface Screens

Figure 1. Room Selection Screen



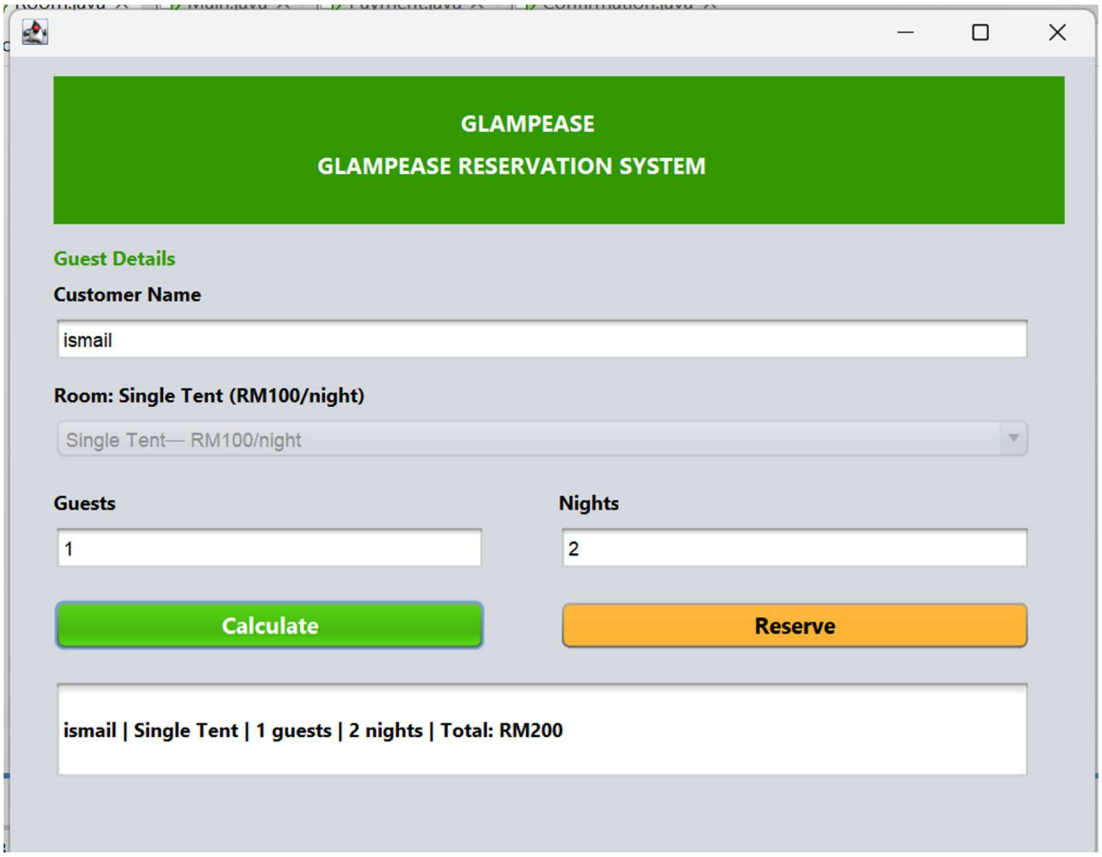
Room displays four accommodation options with images, capacities and nightly rates. Selecting Single Tent stores its room name, RM100 rate and capacity of one in Booking. The dialog acknowledges the selection; Continue opens Main.

Figure 2. Booking Details Screen



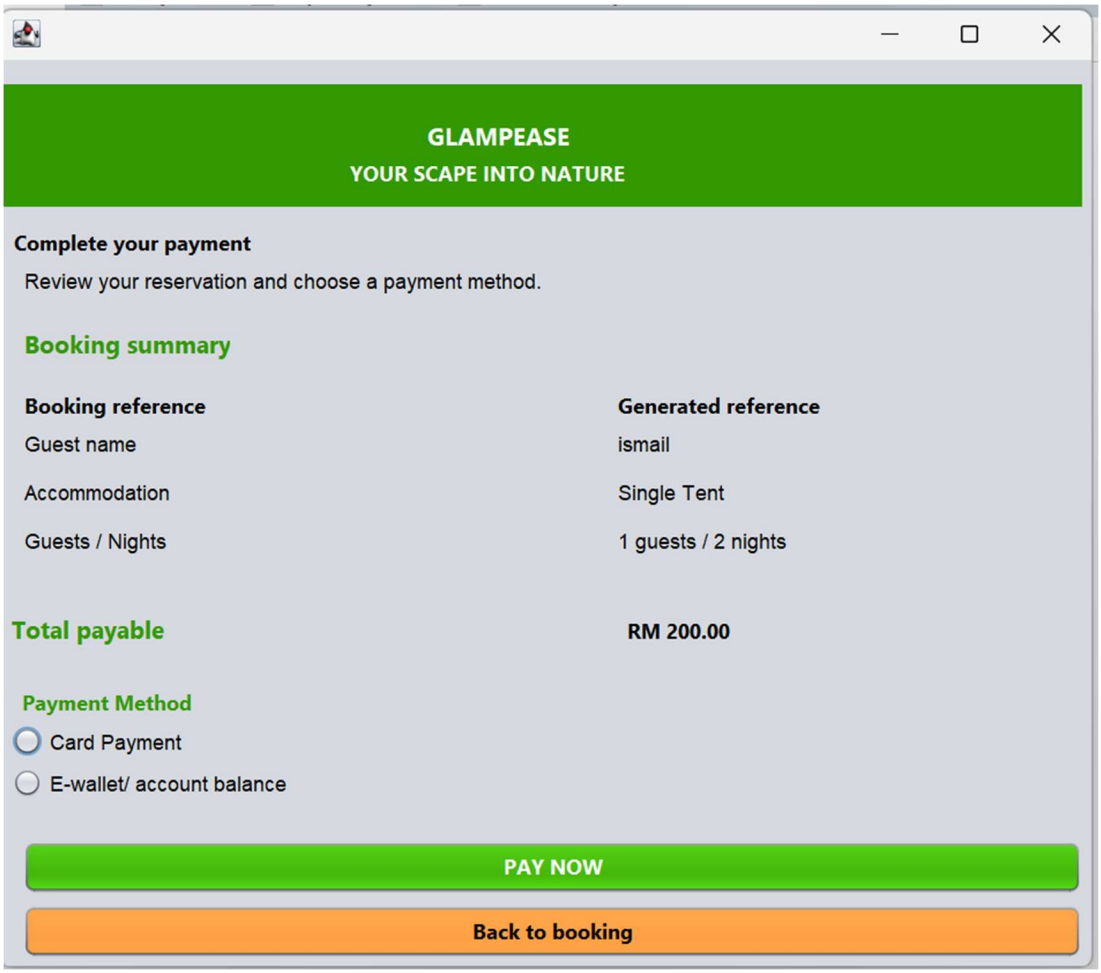
Main carries forward Single Tent at RM100 per night. Accommodation selection is locked and guests are fixed to capacity. The customer enters a name and number of nights, then chooses Calculate.

Figure 3. Calculated Booking Summary



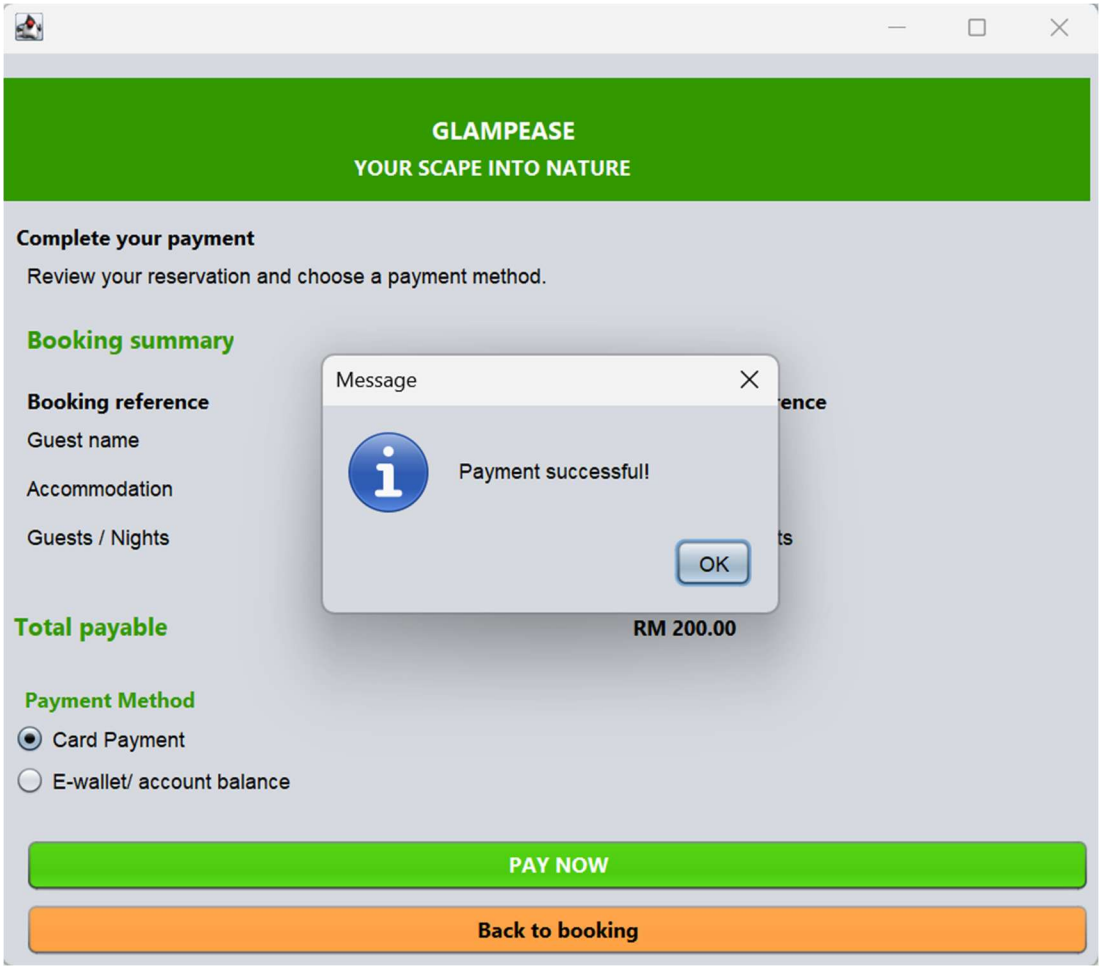
The sample booking is Single Tent for one guest over two nights. The summary shows RM200, calculated as RM100 × 2. The customer reviews this output before selecting Reserve.

Figure 4. Payment Screen



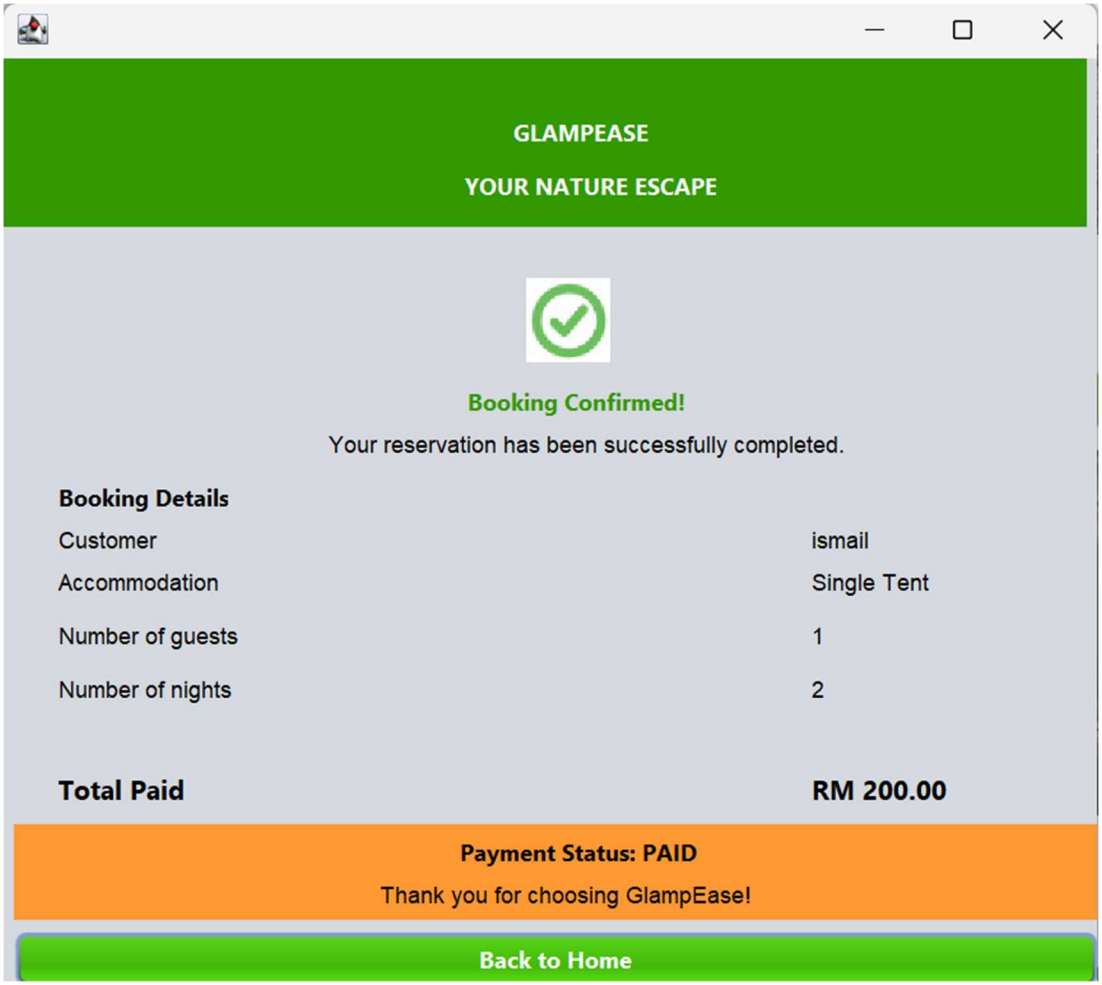
Payment loads the stored customer and reservation details and displays RM200 payable. The customer chooses Card Payment or E-wallet/account balance. Only one option can be selected. Back to booking opens Main. Generated reference is currently a static placeholder.

Figure 5. Payment Success Dialog



PAY NOW requires a selected payment method and displays Payment successful. This is a simulated result without a real bank transfer or account deduction. After the dialog is dismissed, Confirmation opens.

Figure 6. Confirmation and Receipt Screen



The confirmation displays the customer, room, guests, nights, RM200 total paid and PAID status. It acts as the on-screen receipt for the simulated reservation. Back to Home opens the Room selection GUI and closes Confirmation.

10. Testing and Validation

The demonstration below uses Single Tent for one guest and two nights. The screenshots show a consistent RM200 total through booking, payment and confirmation. The team reports that the main flow is complete. Results distinguish screenshot evidence from validation checks identified in the source code.

ID	Test scenario	Expected result	Actual result / status
T01	Open Room and continue.	Next screen opens without error.	Room selection and booking form are shown in Figures 1–2.
T02	Enter valid booking details and click Calculate.	Summary shows correct room, quantities, and total.	Figure 3 shows Single Tent, 1 guest, 2 nights and RM200.
T03	Leave name blank and calculate.	Validation message appears; flow does not continue.	Empty-name guard is implemented in Calculate.
T04	Enter letters in the nights field.	Numeric-input validation message appears.	Numeric parsing and exception handling are implemented.
T05	Reserve after valid calculation.	Booking values appear consistently on Payment.	Figure 4 displays the same booking and RM200 total.
T06	Click Pay Now without selecting a method.	Application asks the customer to select a method.	Payment-method guard is implemented in Pay Now.
T07	Select a method and click Pay Now.	Simulated success message appears and Confirmation opens.	Figures 5–6 show success feedback and confirmation.
T08	Compare Confirmation with booking details.	Name, room, guests, nights, and total match.	Figure 6 matches the demonstrated booking and total.
T09	Click Back to Home.	Room opens and Confirmation closes.	Team confirms return to Room; the source creates Room.
T10	Click Back to booking from Payment.	Main opens and Payment closes.	The source creates Main and disposes Payment.

10.1 Test summary

The demonstrated reservation retains the same accommodation, guest count, nights and RM200 amount across all output screens. Validation guards are documented from the source; these entries do not claim an independent runtime test.

11. Limitations and Future Enhancements

11.1 Current limitations

- Booking details are held in memory rather than a persistent database.
- The shared data structure represents one active booking, not a history of multiple reservations.
- Payment is simulated and does not contact a financial service.
- The prototype does not verify real accommodation availability.
- The current payment screen displays a static booking-reference placeholder rather than generating a unique reference.
- The current booking form fixes guests to room capacity. Payment back navigation creates a fresh booking form, and returning home retains the shared booking values.
- Input checks do not yet reject zero or negative nights or names containing only spaces. Reserve does not repeat all name checks after details change.
- Images use absolute Windows file paths. Packaging them as project resources would improve portability.

11.2 Possible enhancements

- Store bookings in a database and generate a unique booking reference.
- Add an administrator dashboard to view and manage stored bookings.
- Allow guest counts within the room capacity and validate positive nights, trimmed customer names and changed inputs before reservation.
- Add a printable receipt or exportable invoice.
- Improve keyboard navigation, accessibility, and error-message clarity.

12. Team Contribution

Work was distributed across interface development, testing and documentation as shown below.

Member	Contribution
Member 1	Developed the Room GUI.
Member 2	Payment GUI.
Member 3	Confirmation/receipt GUI.
Member 4	Main GUI.
Member 5	Testing, documentation and report preparation.

13. Conclusion

GlampEase demonstrates a front-end glamping reservation and simulated payment flow using Java Swing. A customer can move from accommodation selection to booking details, review a calculated total, select a payment option, and view a confirmation screen. The shared Booking class transfers the current booking values between the screens.

The project demonstrates Java Swing interface design, action handling, input parsing, exception handling, shared data and screen navigation. The confirmation provides a receipt-style reporting output for the simulated transaction. Further development can strengthen validation, preserve booking history and support printable receipts.