# GUI Components & Event Handling Functions

## GUI Components for Interface

1. **JFrame**
   * **Main Frame**: Holds the entire GUI.
   * **Setup Frame**: For initial slot input for visitor and staff cars.
   * **Main Menu Frame**: For the main application interface after setup, including all the main functionalities taken from Project 1 and more.
2. **JPanel**
   * **mainPanel**: Contains input fields and buttons for the initial setup.
   * **menuPanel**: Main menu layout with parking slot buttons and control/menu buttons.
   * **slotPanel**: Displays dynamic buttons for parking slots based on user input.
3. **JLabel**
   * Labels indicate the purpose of the text fields in the setup phase:
     + "Number of Staff Parking Slots:"
     + "Number of Visitor Parking Slots:"
4. **JTextField**
   * **staffSlotInput**: Text field for user input of the number of staff slots.
   * **visitorSlotInput**: Text field for user input of the number of visitor slots.
5. **JButton**
   * **submitButton**: Submits the number of parking slots.
   * **exitButton**: Exits the application.
   * Control buttons in the main menu:
     + **addSlotButton**: Adds a new parking slot, visitor or staff.
     + **deleteSlotButton**: Deletes a selected parking slot by slot ID.
     + **listAllButton**: Lists all parking slots, their occupation type, staff, parked time and date, and fee.
     + **deleteAllButton**: Deletes all unoccupied slots.
     + **parkCarButton**: Parks a car in a selected slot from user input of slot ID, user type, registration number, and owner name.
     + **findCarButton**: Finds a car by its registration number and lists fee, parked time, and slot ID.
     + **removeCarButton**: Removes a car from a slot by its registration number.
6. **JOptionPane**
   * Used for displaying dialog boxes for confirmations and messages, including user input prompts.
7. **StringBuilder**
   * **slotList**: Constructs a string of slot information for display in a dialog when listing all slots.

## Event Handling Functions

1. **Action Listeners**
   * **submitButton** 
     + **Action Listener**: Validates and processes user input for the number of slots.
   * **exitButton** 
     + **Action Listener (Setup)**: Closes the application.
   * **addSlotButton** 
     + **Action Listener**: Calls addParkingSlot() to handle adding new parking slots.
   * **deleteSlotButton** 
     + **Action Listener**: Calls deleteParkingSlot() for removing slots.
   * **listAllButton** 
     + **Action Listener**: Calls listAllSlots() to display all parking slots.
   * **deleteAllButton** 
     + **Action Listener**: Calls deleteAllUnoccupiedSlots() to remove all unoccupied slots.
   * **parkCarButton** 
     + **Action Listener**: Calls parkCar() to handle car parking.
   * **findCarButton** 
     + **Action Listener**: Calls findCarByRegistration() to find a car.
   * **removeCarButton** 
     + **Action Listener**: Calls removeCarByRegistration() to remove a parked car.
   * **exitButton** 
     + **Action Listener (Main Menu)**: Closes the application.
2. **SlotActionListener**
   * Handles actions on parking slot buttons:
     + **actionPerformed** method:
       - Checks if the selected slot is available or occupied.
       - Displays confirmation dialogs for parking or removing a car.
       - Calls parkCarInSlot() or removeCarFromSlot() based on user responses.
3. **Parking Slot Functions**
   * **parkCarInSlot(String slotId, String slotType)**: Handles logic for parking a car in the specified slot, validating registration and updating slot status.
   * **removeCarFromSlot(String slotId)**: Removes a car from the specified slot and resets slot information.
   * **addParkingSlot()**: Prompts for a new slot ID, adds it if valid, and refreshes the panel.
   * **deleteParkingSlot()**: Prompts for a slot ID to delete and updates the GUI if the slot is unoccupied.
   * **listAllSlots()**: Compiles and displays a list of all parking slots, including their status and details (parked time and fees).
   * **appendSlotInfo(StringBuilder slotList, String slotId, ParkingSlotInfo info)**: Appends formatted information about a specific parking slot to the StringBuilder.
4. **Refresh and Validation Functions**
   * **refreshSlotPanel()**: Updates the slot panel to reflect the current status of all parking slots.
   * **isRegistrationUnique(String registration)**: Checks if the provided registration number is unique within the system.
   * **findSlotButton(String slotId)**: Searches for the JButton corresponding to the given slot ID.
   * **findCarByRegistration()**: Prompts for a registration number and displays car information if found.
   * **removeCarByRegistration()**: Prompts for a registration number to remove a car and refreshes the UI.
   * **main(String[] args)**: Initialises the GUI by invoking the ParkingSystemGUI constructor.