

CAMPING TRIP PLANNER

The most efficient way to prepare your memorable days on a camping trip together with your family, friends, or even your partners!



WHAT IT DOES?

- Collect all important information regarding your upcoming trip.
- Automatically calculate all your budgets.
- Archives all your previous trips.

OBJECTIVE

- **USER FRIENDLY INTERFACE**

To create a more easy and conductive interface for our user to use our Camping Trip Planner system by using Pop-ups windows

- **EASY TRACKING**

By providing each function in its own dedicated windows. Thus making users not easily lost and can quickly understand how the system functions

UML DIAGRAM & FLOW CHART

<https://www.figma.com/design/3J7o59rMRz1gzDfU6gQPCt/Untitled?node-id=36-231&t=p0e7kcudDYLj9xdq-1>

MODULE FUNCTIONALITY

CAMPINGTRIPS

- To gather all information regarding trip name, date and time, budgets, items, and location of the campground

MODULE FUNCTIONALITY

INIT

- To show the system to the user in graphical user interface format
- Save all the data into a campingTrips vector object
- Show all data previously held inside the vector

MODULE FUNCTIONALITY

DATEANDTIME

- Collecting and validating date and time information
- Ensures inputs are in the correct format and valid, essential for accurate trip planning and scheduling.
- Includes methods to verify and retrieve formatted date and time values

MODULE FUNCTIONALITY

BUDGET

- Collect information regarding the trip's budget
- Acts as the Parent class for Transportation and Activities
- Gives the remaining budget balance based on the budget spend that was stated

MODULE FUNCTIONALITY

TRANSPORTATION

- Collect information about the User's transportation method to the camping site
- It is the Child class to the Parent class Budget
- Send transportation cost value to the method in Budget so that we can tabulate the total budget spend

MODULE FUNCTIONALITY

ACTIVITIES

- Subclass of the `Budget` class, designed to manage and represent activity-related expenses in the trip planning application.
- Supports functionality to add activity expenses to the overall budget.
- Can track the total budget spend

MODULE FUNCTIONALITY

ITEMS

- Retrieve information of the items for the camping trip.
- It is a aggregation to the CampingTrips class.
- Parent class for the Food class
- Include the method to calculate cost for item.

MODULE FUNCTIONALITY

FOOD

- Retrieve information of the food only for the camping trip.
- Functionality of retrieve details from superclass Item.
- It inherits from the Item class as it is child class for the Item class.
- Include the method to add quantity of the food

MODULE FUNCTIONALITY

LOCATION

- Name and description are straightforward attributes representing basic information about the location.
- Returns the name of the location as a string.
- This method is declared as abstract

MODULE FUNCTIONALITY

CAMPGROUND

- Represents the number of camping sites that are currently available in the campground
- This constructor initializes all attributes of the campground
- Returns the number of available camping sites as an integer.



TECHNICAL IMPLEMENTATION

- Encapsulation & Data Hiding
 - Aggregation
 - Composition
 - Inheritance
 - Polymorphism
 - Exception Handling
 - User Interface
- 

ENCAPSULATION & DATA HIDING



```
public class Activities extends Budget {  
    private String activityName;  
    private double activityCost;  
    private Color activityColor;
```

```
    public String getName() {  
        return activityName;  
    }
```

```
    public Color getColor() {  
        return activityColor;  
    }
```


AGGREGATION



```
class CampingTrips {  
    private String name;  
    private DateAndTime dateTime;  
    private Vector<Budget> budget = new Vector<Budget>();  
    private Vector<Item> item = new Vector<Item>();  
    private Location campground = new Campground();  
  
    public CampingTrips(String name, DateAndTime dateTime,  
        Vector<Budget> tripBudgets, Vector<Item> item,  
        String campName, String campDescription,  
        int campSites) {  
  
        this.name = name;  
        this.dateTime = dateTime;  
        this.budget = tripBudgets;  
        this.item = item;  
    };  
};
```


COMPOSITION



```
class CampingTrips {  
    private String name;  
    private DateAndTime dateTime;  
    private Vector<Budget> budget = new Vector<Budget>();  
    private Vector<Item> item = new Vector<Item>();  
    private Location campground = new Campground();  
  
    public CampingTrips(String name, DateAndTime dateTime,  
        Vector<Budget> tripBudgets, Vector<Item> item,  
        String campName, String campDescription,  
        int campSites) {  
        this.name = name;  
        this.dateTime = dateTime;  
        this.budget = tripBudgets;  
        this.item = item;  
        this.campground = new Campground(campName, campDescription, campSites);  
    };  
};
```


INHERITANCE

```
class Item {  
    private String name;  
    private int quantity;  
    private double price;  
    private String description;  
  
    public Item() {  
    }  
  
    public Item(String name, int quantity, double price) {  
        this.name = name;  
        this.price = price;  
        this.quantity = quantity;  
    }  
}
```

```
public class Food extends Item {  
    public Food(String n, int q, double p, String e, Boolean t) {  
        super(n, q, p);  
        expirationDate = e;  
        isVegetarian = t;  
    }  
}
```



POLYMORPHISM

```
abstract class Location {  
    public abstract String displayLocationInfo();  
}
```

```
class Campground extends Location {  
    public String displayLocationInfo() {  
        String str = String.format("Campground Name: %s\nDescription: %s\nSites Available: %d",  
            getName(), getDescription(), sitesAvailable);  
        return str;  
    }  
}
```



EXCEPTION HANDLING



```
// Method untuk check date format
public static boolean isValidDate(String date) {
    SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
    dateFormat.setLenient(false);
    try {
        dateFormat.parse(date);
        return true;
    } catch (ParseException e) {
        return false;
    }
}

// Method untuk check time format
public static boolean isValidTime(String time) {
    SimpleDateFormat timeFormat = new SimpleDateFormat("HH:mm");
    timeFormat.setLenient(false);
    try {
        timeFormat.parse(time);
        return true;
    } catch (ParseException e) {
        return false;
    }
}
```


USER INTERFACE



```
private static JTextField addTextField(String labelText, int x, int y, int width, int height) {
    JLabel label = new JLabel(labelText);
    label.setBounds(x, y, width, height);
    frame.add(label);

    JTextField textField = new JTextField();
    textField.setBounds(x, y + 30, width, height);
    frame.add(textField);

    return textField;
}

private static JComboBox<String> addComboBox(String labelText, int x, int y, int width, int height,
    String[] options) {
    JLabel label = new JLabel(labelText);
    label.setBounds(x, y, width, height);
    frame.add(label);

    JComboBox<String> comboBox = new JComboBox<>(options);
    comboBox.setBounds(x, y + 30, width, height);
    frame.add(comboBox);

    return comboBox;
}

private static JComboBox<ColoredItem> addColorComboBox(String labelText, int x, int y, int width, int height,
    ColoredItem[] items) {
    JLabel label = new JLabel(labelText);
    label.setBounds(x, y, width, height);
    frame.add(label);

    JComboBox<ColoredItem> comboBox = new JComboBox<>(items);
    comboBox.setRenderer(new ColorfulRenderer());
    comboBox.setBounds(x, y + 30, width, height);
    frame.add(comboBox);

    return comboBox;
}

private static void addButton(String buttonText, int x, int y, ActionListener actionListener) {
    JButton button = new JButton(buttonText);
    button.setBounds(x, y, width:150, height:30);
    button.addActionListener(actionListener);
    frame.add(button);
}

private static void addLabel(String labelText, int x, int y, int width, int height, Font font) {
    JLabel label = new JLabel(labelText);
    label.setBounds(x, y, width, height);
    label.setFont(font);
    frame.add(label);
}
```


CAMPING TRIP PLANNER

DEMO

