EVENT PLANING SYSTEM

Project Team

Student Name	Roll Number	Class	Email Address
Nafeesa Yousaf	FL21503	BSSE-5B	numl-f22-38910@numls.edu.pk
Muhammad Bilal	FL23847	BSSE-5B	numl-f23-88991@numls.edu.pk

Submitted to:
Ms. Fatima Gillani

Project Deliverable 2

Department of Software Engineering

NATIONAL UNIVERSITY OF MODERN LANGUAGES

ISLAMABAD

December, 2023

TABLE OF CONTENTS

Introduction **3**

Purpose **4**

System Design **5**

UML Use case Diagram 6

Object Diagram 8

Sequence Diagram 10

Communication Diagram 12

Petri Nets 13

INTRODUCTION

The second deliverable of the Event Planning/Management System focuses on the detailed design documentation, which provides a visual and structured representation of the system's architecture, interactions, and workflows. This deliverable is essential for bridging the gap between the requirements and the implementation phase, ensuring clarity, consistency, and alignment among all stakeholders.

The design diagrams included in this document illustrate the system's static and dynamic behavior, showcasing how various actors interact with the system to achieve the desired functionalities. These diagrams help in understanding the underlying structure, data flow, and interactions, enabling the development team to implement the system efficiently and accurately.

PURPOSE

The purpose of this document is to:

1. **Provide clarity:** Outline the detailed design and interactions within the system to facilitate understanding among team members and stakeholders.
2. **Ensure completeness:** Highlight all key components, interactions, and workflows to cover the system's functional and non-functional requirements.
3. **Establish a development roadmap:** Serve as a reference point for developers to guide the implementation of the system.

SYSTEM DESIGN

In the system design phase, we create the following system design diagrams:

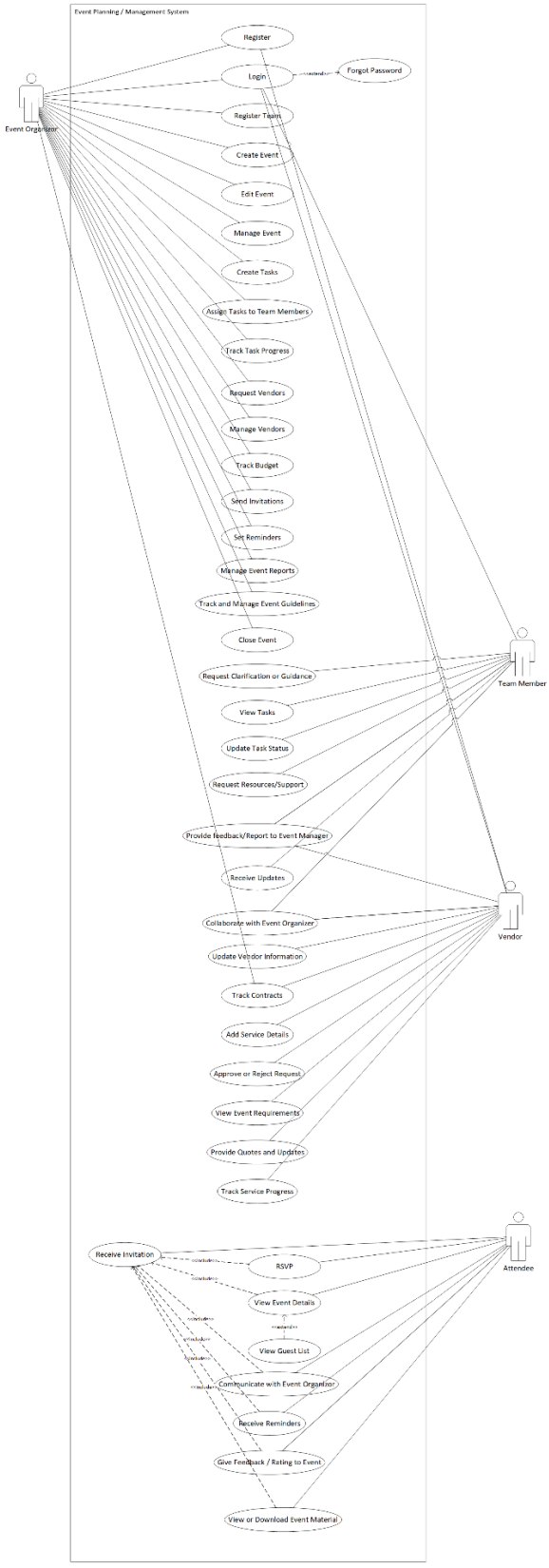
1. **Use-case Diagram:** Illustrates the interactions between various actors (Event Organizer, Vendors, Team Members, Attendees, and Admin) and the system's use cases.
2. **Object Diagram:** Depicts the static structure of the system, showing classes, objects, and their relationships.
3. **Sequence Diagram:** Demonstrates the sequence of interactions between objects during specific scenarios (e.g., event creation or task assignment).
4. **Communication Diagram:** Represents the message flow and interactions between components or objects in the system.
5. **Petri-Nets:** Models concurrency and synchronization within the system, highlighting states, transitions, and token movement.
6. **Timing Diagram:** Visualizes timing constraints and the temporal aspects of various events and processes in the system.

These diagrams collectively provide a comprehensive view of the Event Planning System's design, ensuring the system's functionality is well-documented and implementable.

Note:

The font size may appear small after pasting the diagram into the Word document due to compatibility issues with the page size. The diagram exceeds the dimensions of the page, causing the text to shrink, blur, and break pixels. Please zoom in to read the details clearly. We have included the diagram images in a zipped file for your convenience if you continue to experience issues.

UML USE CASE DIAGRAM

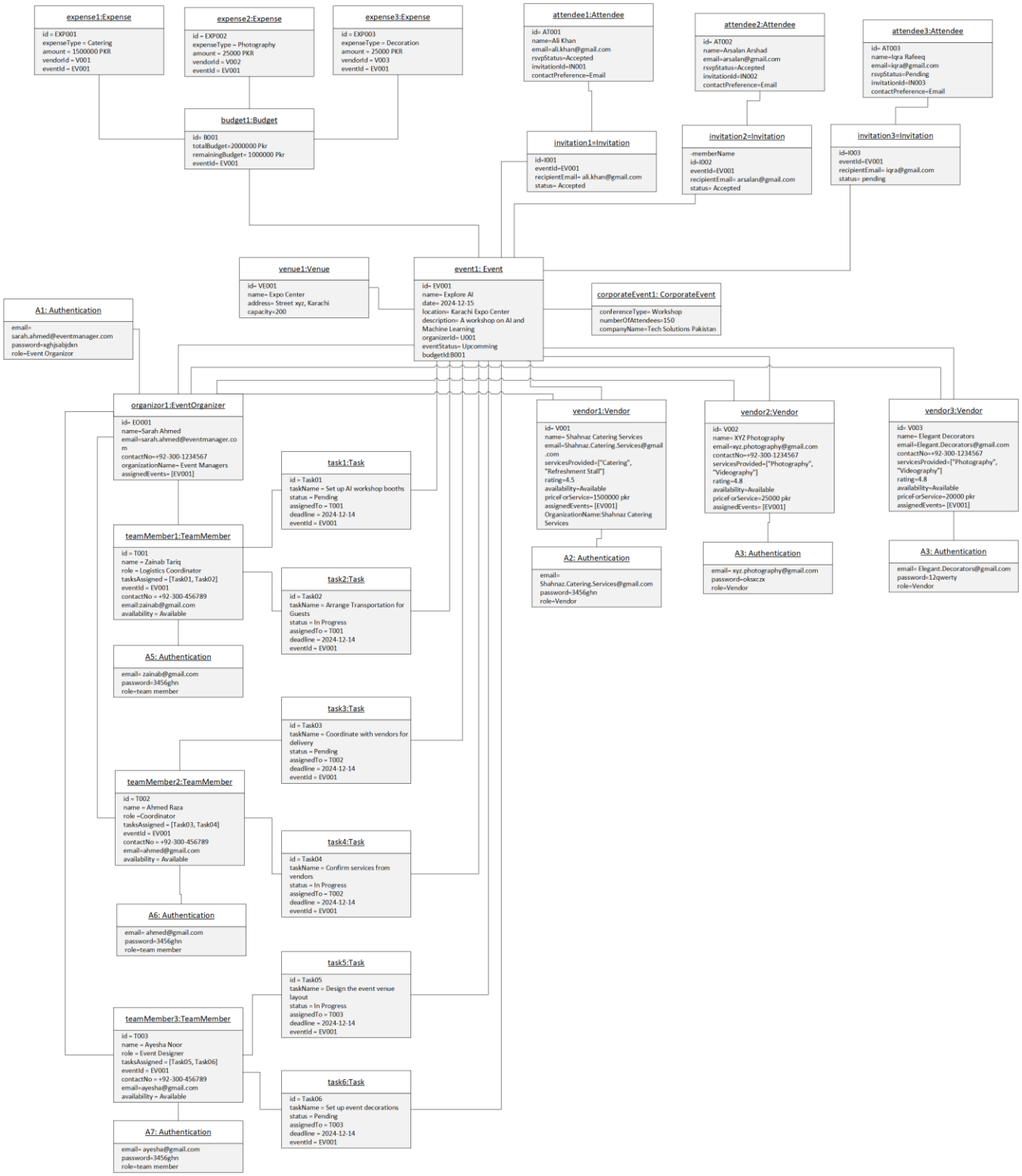


Why Team Members, Vendors, and Attendees Are Secondary Actors

The Event Organizer is the primary actor because they are the main user of the system who initiates and manages all core functions. They create events, register team members, assign tasks, hire vendors, send invitations, and oversee the entire event planning process.

All other actors—Team Members, Vendors, and Attendees—are secondary because their roles are supportive and reactive. Team Members only log in to view and update their assigned tasks. Vendors interact with the system to accept assignments and provide service updates, while Attendees only respond to invitations and RSVPs. These secondary actors rely on the Event Organizer's actions to interact with the system, making the Event Organizer the central figure driving the system's functionality.

OBJECT DIAGRAM



In this object diagram, we have focused on covering a single scenario of the system to keep the representation concise and understandable. The scenario chosen highlights the key interactions between the event, event organizer, team members, vendors, attendees, tasks, and other related objects.

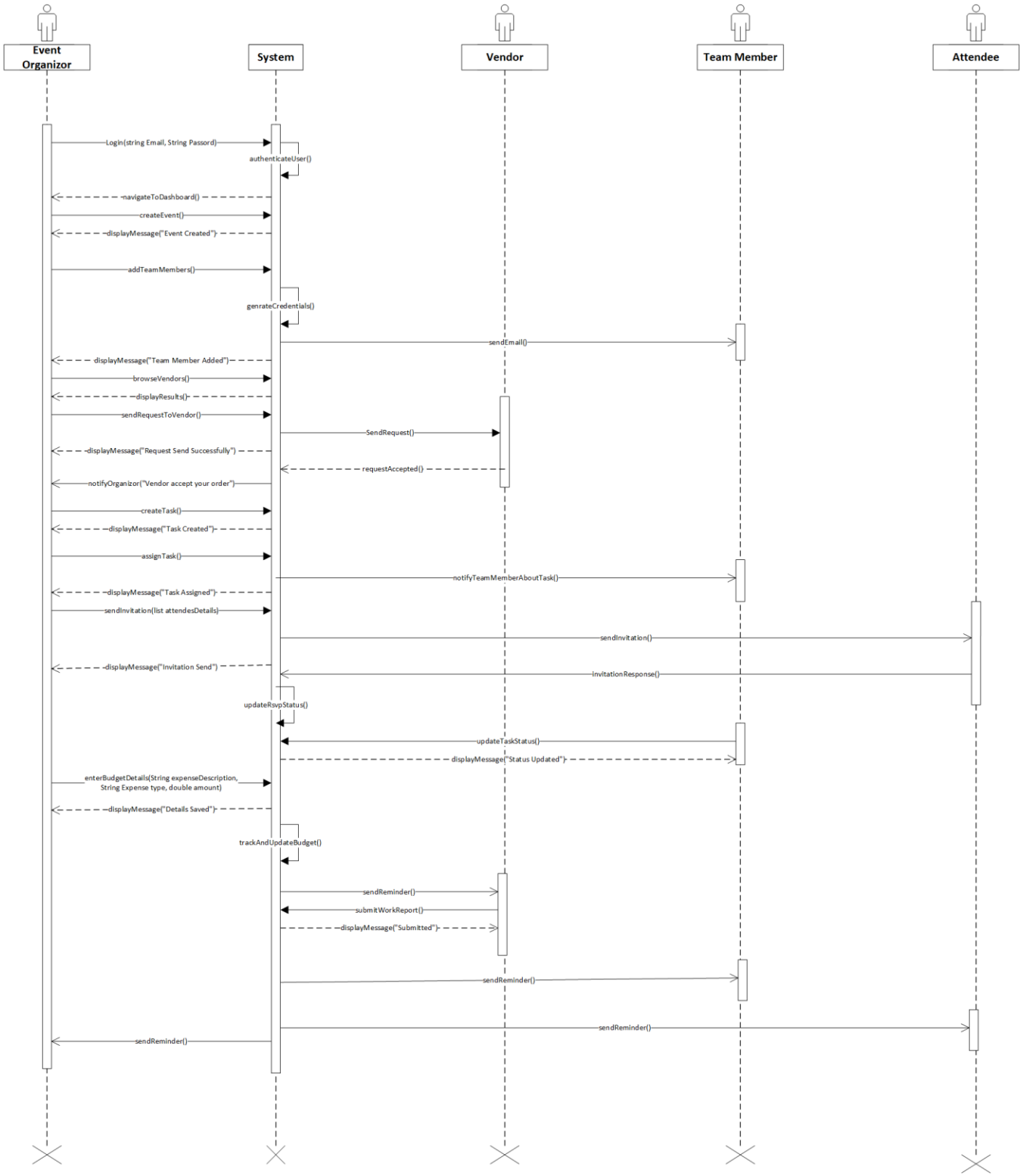
If we were to include multiple scenarios to account for all possible interactions and variations within the system, the object diagram would become significantly more complex and challenging to interpret. Therefore, for simplicity and clarity, this diagram represents one specific use case, ensuring the core relationships and associations are accurately depicted.

SEQUENCE DIAGRAM

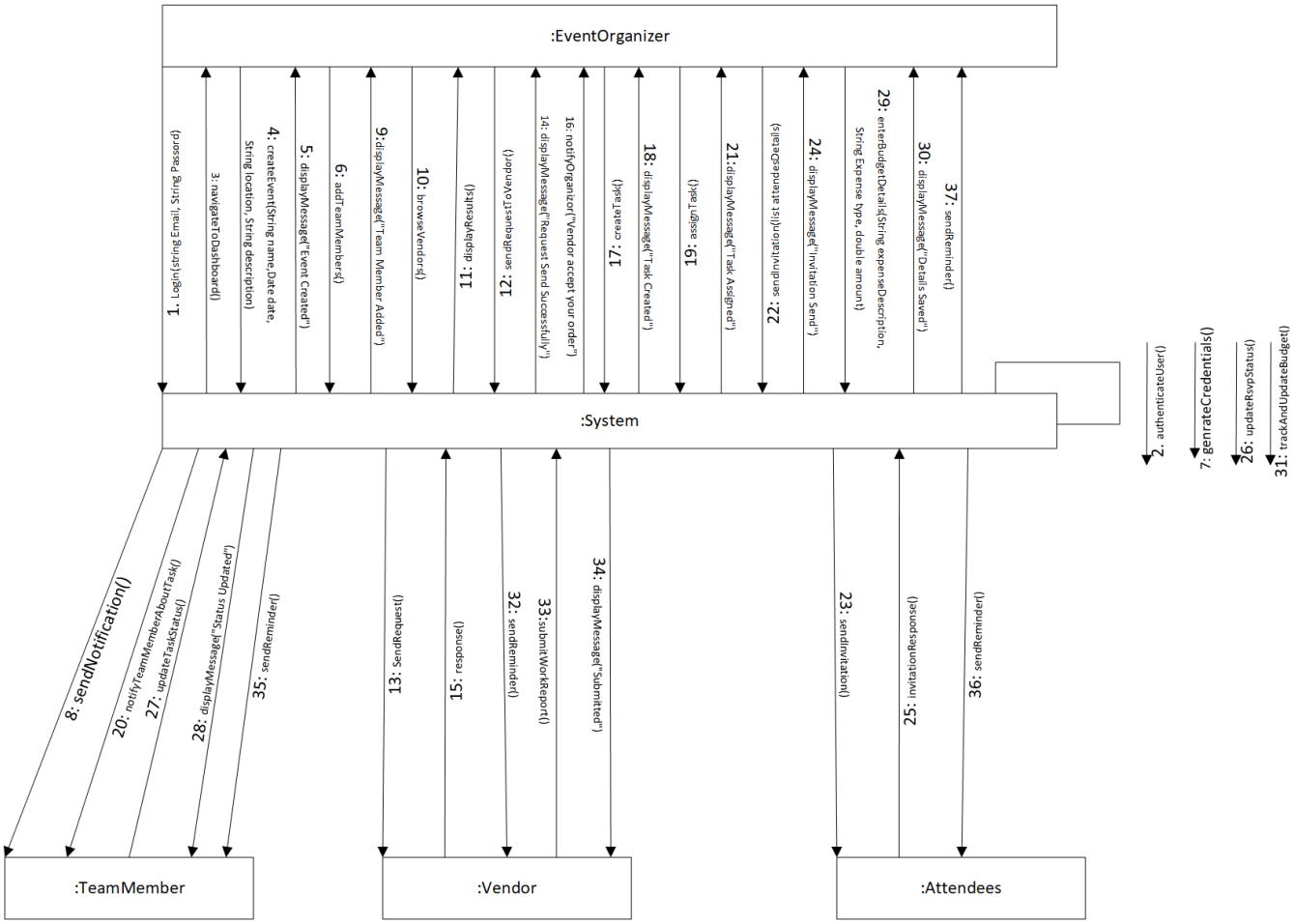
This sequence diagram shows interaction among objects or parts of the system during the specified use-case specific situation of the management of corporate event "Explore AI".

Interaction between objects/components of the system can be described by a sequence of the following steps:

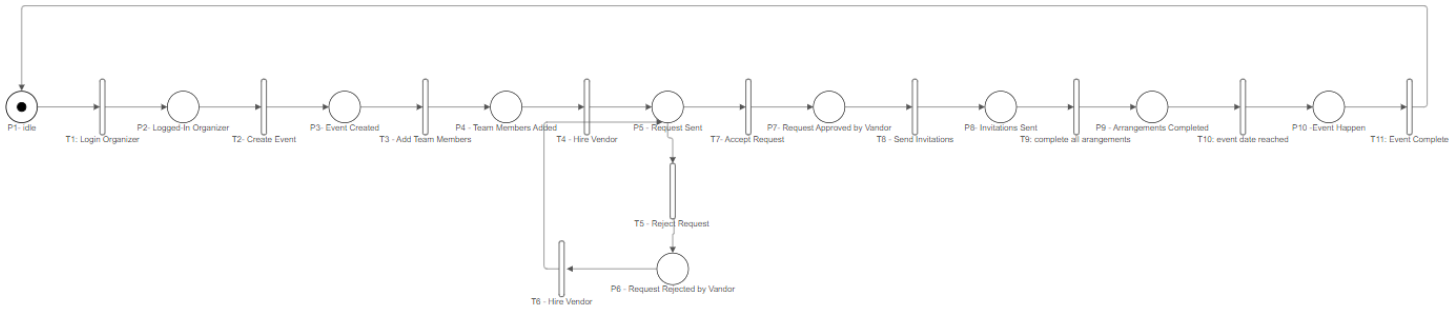
- Event Organizer Logs In and Creates an Event
- Event Organizer Registers Team Members
- Assigning Tasks to Team Members
- Event Organizer Selects and Hires Vendors
- Event Organizer Sends Invitations to Attendees
- Budget Management and Expense Tracking
- Sending Notifications and Reminders


Event Organizor
System
Vendor
Team Member
Attendee
Login(string Email, String Passord)
authenticateUser()
navigateToDashboard()
createEvent()
displayMessage("Event Created")
addTeamMembers()
genrateCredentials()
sendEmail()
displayMessage("Team Member Added")
browseVendors()
displayResults()
sendRequestToVendor()
SendRequest()
displayMessage("Request Send Successfully")
requestAccepted()
notifyOrganizor("Vendor accept your order")
createTask()
displayMessage("Task Created")
assignTask()
notifyTeamMemberAboutTask()
displayMessage("Task Assigned")
sendInvitation(list attendesDetails)
sendInvitation()
displayMessage("Invitation Send")
InvitationResponse()
updateRsvpStatus()
updateTaskStatus()
displayMessage("Status Updated")
enterBudgetDetails(String expenseDescription, String Expense type, double amount)
displayMessage("Details Saved")
trackAndUpdateBudget()
sendReminder()
submitWorkReport()
displayMessage("Submitted")
sendReminder()
sendReminder()
sendReminder()

COMMUNICATION DIAGRAM



PETRI-NETS



Input Table:

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
T1	1	0	0	0	0	0	0	0	0	0
T2	0	1	0	0	0	0	0	0	0	0
T3	0	0	1	0	0	0	0	0	0	0
T4	0	0	0	1	0	0	0	0	0	0
T5	0	0	0	0	1	0	0	0	0	0
T6	0	0	0	0	0	1	0	0	0	0
T7	0	0	0	0	1	0	0	0	0	0
T8	0	0	0	0	0	0	1	0	0	0
T9	0	0	0	0	0	0	0	1	0	0
T10	0	0	0	0	0	0	0	0	1	0
T11	0	0	0	0	0	0	0	0	0	1

Output Table:

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
T1	0	1	0	0	0	0	0	0	0	0
T2	0	0	1	0	0	0	0	0	0	0
T3	0	0	0	1	0	0	0	0	0	0
T4	0	0	0	0	1	0	0	0	0	0
T5	0	0	0	0	0	1	0	0	0	0
T6	0	0	0	0	1	0	0	0	0	0
T7	0	0	0	0	0	0	1	0	0	0
T8	0	0	0	0	0	0	0	1	0	0
T9	0	0	0	0	0	0	0	0	1	0
T10	0	0	0	0	0	0	0	0	0	1
T11	1	0	0	0	0	0	0	0	0	0

Firing Table:

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
T0	1	0	0	0	0	0	0	0	0	0
T1	0	1	0	0	0	0	0	0	0	0
T2	0	0	1	0	0	0	0	0	0	0
T3	0	0	0	1	0	0	0	0	0	0
T4	0	0	0	0	1	0	0	0	0	0
T5	0	0	0	0	0	1	0	0	0	0
T6	0	0	0	0	1	0	0	0	0	0
T7	0	0	0	0	0	0	1	0	0	0
T8	0	0	0	0	0	0	0	1	0	0
T9	0	0	0	0	0	0	0	0	1	0
T10	0	0	0	0	0	0	0	0	0	1
T11	1	0	0	0	0	0	0	0	0	0