Assignment 5

Title:

Design And Implementation Design Model From Analysis Model.

Problem Statement:

- Prepare a Design Model from Analysis Model.
- Study in detail working on systems/projects.
- Identify Design classes/ Evolve Analysis Model. Use advanced relationships. Draw Design class Model using OCL and UML2.0 Notations. Implement the design model with a suitable object-oriented language.

Objective:

- To Identify Design level Classes.
- To Draw Design level class Model using analysis model.
- To Implement Design Model-class diagram

Theory:

The analysis model is refined and formalized to get a design model. Design modeling, means refinement to analysis level models to adapt to the actual implementation environment. In the design space, yet another new dimension has been added to the analysis space to include the implementation environment. This means that we want to adopt our analysis model to fit in the implementation model at the same time as we refine it.

1. Creating Design level Class Diagrams

   Class diagrams model the static structure of a package or of a complete system. As the blueprints of the system, class diagrams model the objects that make up the system, allowing to display the relationships among those objects and to describe what the objects can do and the services they can provide.

2. Class diagrams

In UML, class diagrams are one of six types of structural diagram. Class diagrams are fundamental to the object modeling process and model the static structure of a system. Depending on the complexity of a system, you can use a single class diagram to model an entire system, or you can use several class diagrams to model the components of a system.

VERBS:

- Sign up
- Login
- View event
- Register for event
- Pay
- Check out
- Check payment status
- Create event
- Add manager
- Add volunteer
- Create team
- Select team
- Verify user
- Generate certificate

CLASSES:

- Event Class
- Event Registration Class
- User Class
    - Participant
    - Volunteer
    - Manager
    - Admin
- Team Class

ATTRIBUTES:

- **Event :** eventId, eventName, organizingDept, managerId, participationAmount, eventType, startDate, endDate
- **Event Registration :** regId, eventId, userId, regDate
- **User** : userType, name, gender, contactNo, emailId, username, password
- **Team** : teamId, eventId, teamName, noOfMembers.
- **Participant** : category, paymentHistory[]
- **Volunteer :** volunteerId, eventName
- **Admin :** adminId
- **Manager :** managerId, eventName

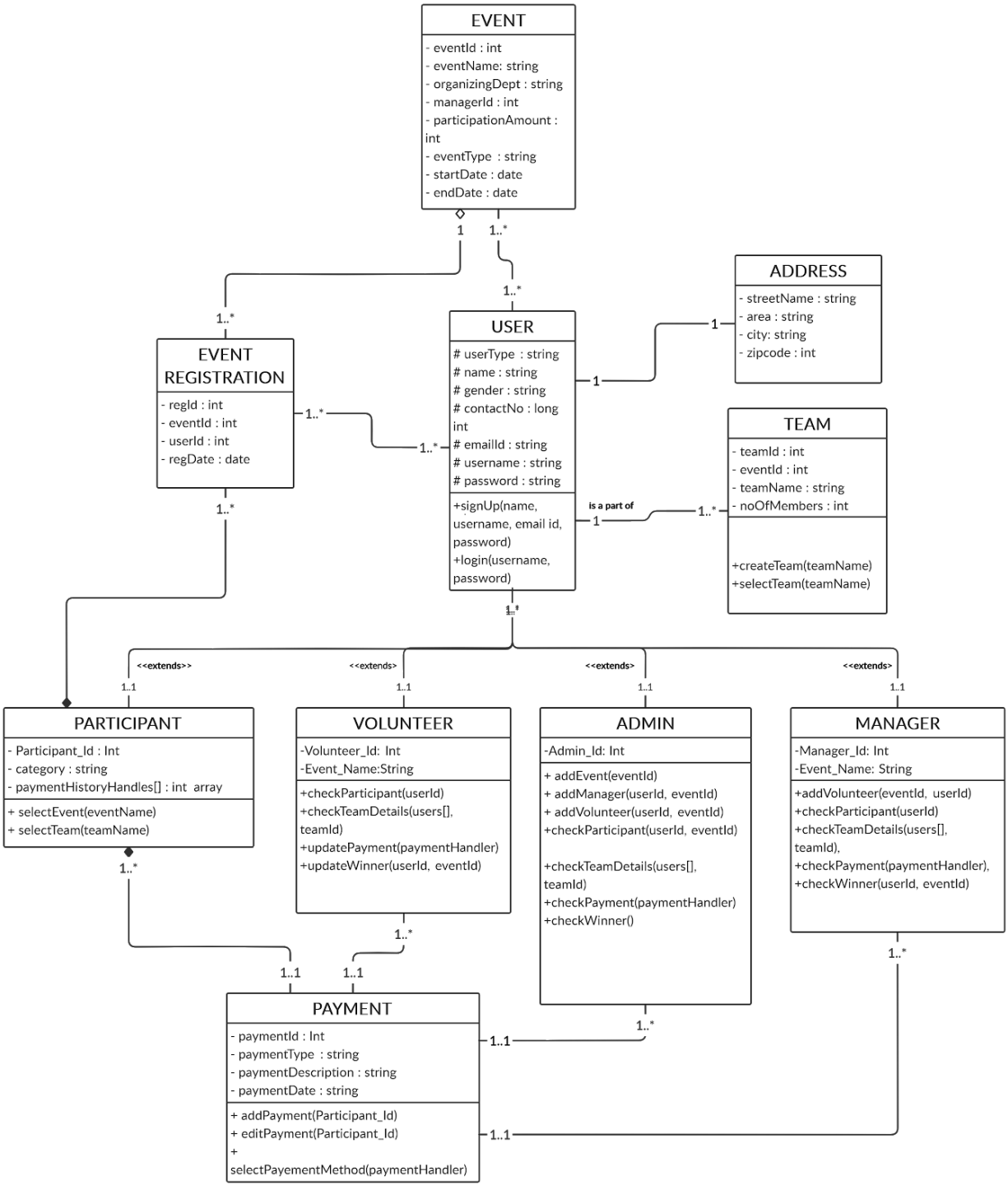
OPERATIONS:

- **User :** signUp(name, username, email id, password), login(username, password)
- **Team :** createTeam(teamName), selectTeam(teamName)
- **Participant** : selectEvent(eventName), selectTeam(teamName), makePayment(paymentMethod)
- **Volunteer :** checkParticipant(userId, eventId), checkTeamDetails(users[], teamId), updatePayment(paymentHandler), updateWinner(userId, eventId)
- **Manager** : addVolunteer(eventId, userId), checkParticipant(userId, eventId), checkTeamDetails(users[], teamId), checkPayment(paymentHandler), checkWinner(userId, eventId)
- **Admin** : addEvent(eventId), addManager(userId, eventId), addVolunteer(userId, eventId), checkParticipant(userId, eventId), checkTeamDetails(users[], teamId), checkPayment(paymentHandler), checkWinner(userId, eventId)

Strategy for Creation of Design Model

- In our assignment, we tried to decide cardinality for creation and editing of the classes by referring to the class diagram.
- Then with thinking from the point of view of the actual code implementation, additional classes are created, for eg. Address, Payment to store redundant information.
- We finalized the associations after taking in consideration our relational schema of our problem.
- Thus, the diagram contains representation of all the S/W classes that would be implemented

Design Model for Event Registration System



Conclusion:

We have thus implemented accepted standards and procedures to develop a Design Model for the project idea we have picked (Event Registration System). UML is the standard language for specifying, designing, and visualizing the artifacts of software systems. Thus we have understood that Class Diagram provides an overview of how the application is structured before studying the actual code. It certainly reduces the maintenance time.