P P SAVANI UNIVERSITY

TUTORIAL NO.:9

ON

SOFTWARE ENGINEERING(SSCS3010)

TITLE: Identify the Elements and Relationships by Analyzing the Class Diagram of Shop Retail Application

BACHELOR OF SCIENCE IN INFORMATION TECHNOLOGY (BSC-IT)

SUBMITTED TO:

Name: MS. HEMANGINI MEHTA(HGM)

Designation: ASSISTANT PROFESSOR

P P Savani University

SUBMITTED BY:

Name: RAJ MO FAHIM ZAKIR

Enrollment: 23SS02IT161

BSCIT5B-Batch 2023-26

Faculty Signature: _____

INSTITUTE OF COMPUTER SCIENCE AND APPLICATIONS
P P SAVANI UNIVERSITY
MANGROL, SURAT- 394125 (GUJARAT)

Student Name: RAJ MO FAHIM ZAKIR
Enrolment Number: 23SS02IT161
Subject Name: SOFTWARE ENGINEERING
Subject Code: SSCS3010

TUTORIAL-9 **Date:09/09/2025**

Aim: Identify the Elements and Relationships by Analyzing the Class Diagram of Shop Retail Application

Objective

The purpose of this tutorial is to understand how to identify **elements (classes, attributes, operations)** and **relationships (associations, aggregation, composition, inheritance)** in a **class diagram** using a **Shop Retail Application** case study.

Step 1: Recall Basics of Class Diagram

A **class diagram** in UML shows:

- **Classes** (blueprints of objects)
- **Attributes** (data members)
- **Operations/Methods** (functions of class)
- **Relationships** between classes (association, aggregation, composition, inheritance, dependency)

Step 2: Case Study – Shop Retail Application

A **Shop Retail Application** allows customers to browse products, place orders, and make payments.
Key requirements:

1. The shop sells multiple products.
2. Customers can register and place orders.
3. Each order contains one or more products.
4. A payment is linked to an order.
5. The shop has employees who manage products and orders.

Step 3: Elements (Classes and Attributes)

From the case study, we can identify these classes:

1. **Customer**
    - Attributes: CustomerID, Name, Email, Phone
    - Methods: register(), placeOrder(), viewOrder()
1. **Product**
    - Attributes: ProductID, Name, Price, StockQty
    - Methods: updateStock(), getPrice()

1. **Order**
    - Attributes: OrderID, OrderDate, Status
    - Methods: addProduct(), calculateTotal(), generateInvoice()
1. **OrderItem**
    - Attributes: Quantity
    - Methods: getSubTotal()
1. **Payment**
    - Attributes: PaymentID, Amount, Date, Mode
    - Methods: processPayment()
1. **Employee**
    - Attributes: EmpID, Name, Role
    - Methods: manageProduct(), verifyOrder()

Step 4: Relationships

- **Customer – Order**:
  A customer *places* orders. (1-to-many association)
- **Order – OrderItem – Product**:
  An order *contains* multiple order items.
  Each order item is *linked* to one product. (association, aggregation)
- **Order – Payment**:
  Each order has one payment. (1-to-1 association, composition if tightly bound)
- **Employee – Product**:
  Employees *manage* products. (association)
- **Employee – Order**:
  Employees *verify* orders. (association)

Step 5: Example Class Diagram

Step 6: Analysis

- **Elements**: Classes (Customer, Product, Order, OrderItem, Payment, Employee) with attributes & operations.
- **Relationships**: Associations (Customer–Order, Order–Payment), Aggregation (Order–OrderItem), and Management (Employee–Product/Order).
- **Multiplicity**: One customer → many orders; One order → many products; One product → many order items.

Case Study – Resume Builder Application (For Freshers)

Objective

To identify classes, attributes, operations, and relationships for a UML class diagram of a Resume Builder Application, focusing on fresh graduates who want to create, edit, and export resumes.

Step 1: Basics of Class Diagram (Recap)

- Classes: Core entities (blueprints of objects)
- Attributes: Properties of classes
- Methods/Operations: Functions performed by classes
- Relationships:
    - Association (link between classes)
    - Aggregation/Composition (whole-part relationship)
    - Inheritance (generalization-specialization)

Step 2: Case Study Description

A Resume Builder Application should:

- Allow freshers to create accounts and build resumes.
- Store personal details, education, skills, and projects.
- Provide resume templates for users to choose from.
- Allow users to preview and export resumes (PDF, Word).
- Admin/Employees can manage templates and verify formatting.

Step 3: Elements (Classes, Attributes, Operations)

User (Fresher/Customer)

- Attributes: `UserID`, `Name`, `Email`, `Password`
- Methods: `register()`, `login()`, `createResume()`, `exportResume()`

Resume

- Attributes: `ResumeID`, `Title`, `CreatedDate`, `LastUpdated`
- Methods: `addSection()`, `editSection()`, `previewResume()`, `downloadResume()`

Section (Abstract Class)

(Specialized into PersonalInfo, Education, Skills, Projects, Experience)

- Attributes: `SectionID`, `Content`
- Methods: `editContent()`, `viewContent()`

PersonalInfo (inherits Section)

- Attributes: `Name`, `Phone`, `Email`, `Address`

Education (inherits Section)

- Attributes: `Degree`, `Institution`, `Year`, `CGPA`

Skill (inherits Section)

- Attributes: `SkillName`, `ProficiencyLevel`

Project (inherits Section)

- Attributes: `ProjectTitle`, `Description`, `Technology`

Template

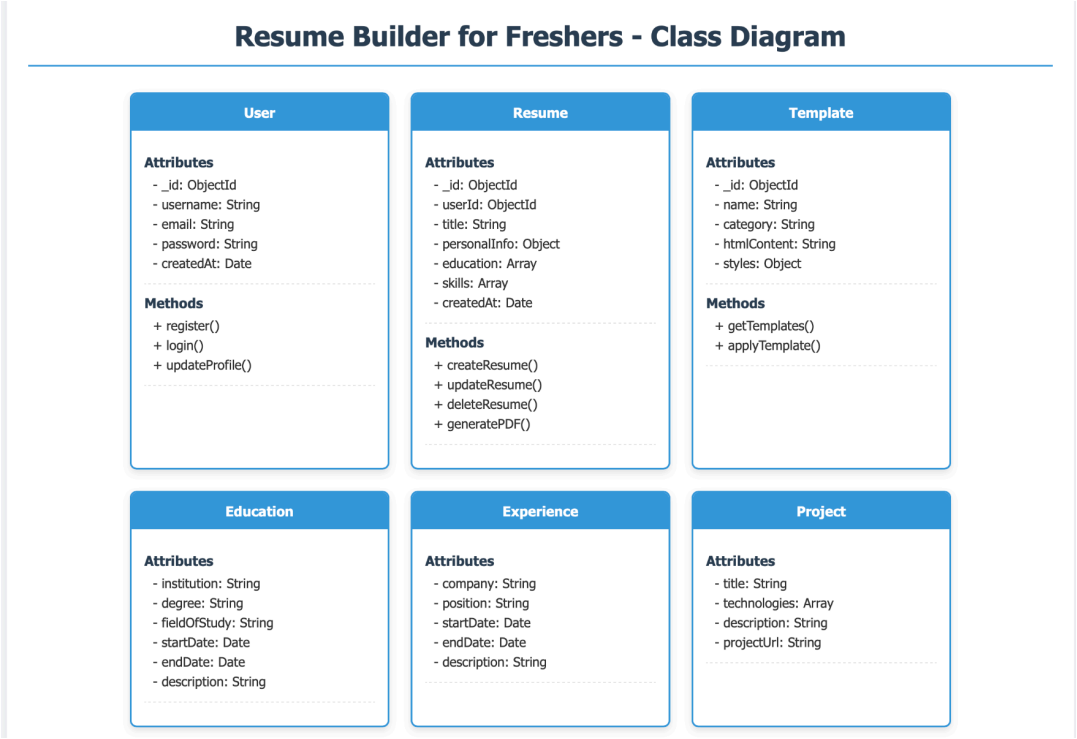
- Attributes: `TemplateID`, `Name`, `Layout`
- Methods: `applyTemplate()`, `customizeTemplate()`

Admin/Employee

- Attributes: `EmpID`, `Name`, `Role`
- Methods: `manageTemplate()`, `verifyResume()`

Step 4: Relationships

- User – Resume: One user can create multiple resumes (1-to-many association)
- Resume – Section: Resume is composed of multiple sections (composition)
- Section – Subclasses: Inheritance (PersonalInfo, Education, Skills, Projects)
- Resume – Template: A resume uses one template (association)
- Admin – Template: Admins manage templates (association)



Step 6: Analysis

- Classes Identified: User, Resume, Section (and its subtypes), Template, Admin
- Relationships:
  - Association: User–Resume, Resume–Template, Admin–Template, Admin–Resume
  - Composition: Resume–Section (sections cannot exist without a resume)
  - Inheritance: Section → (PersonalInfo, Education, Skills, Project, etc.)
- Multiplicity:
  - One user → many resumes
  - One resume → many sections
  - One section → one resume
  - One resume → one template