Lab 1: Introduction to CASE (Computer-Aided Software Engineering) Tools

Objective:

Learn and explore various CASE tools to understand their importance and practical usage in software engineering.

Description:

1. StarUML

StarUML is a powerful software used for creating and managing diagrams based on the Unified Modeling Language (UML). It is widely used by developers, system designers, and architects due to its advanced features. Some of the key features include:

- **Wide UML Support:** It supports different types of UML diagrams like class diagrams, sequence diagrams, use case diagrams, and activity diagrams.
- **Code Engineering:** It allows you to generate source code from UML diagrams and also reverse the process, supporting programming languages like Java, C#, and Python.
- **Custom Extensions:** StarUML has a flexible design that allows users to add extra features by creating custom extensions.
- **Cross-Platform:** It works on different operating systems, including Windows, macOS, and Linux.

2. Draw.io

Draw.io is a free, web-based tool used to create various types of diagrams. It is known for being simple, versatile, and easy to use. Some key features include:

- **Variety of Diagrams:** It can be used to create flowcharts, entity-relationship diagrams, network diagrams, UML diagrams, and more.
- **Easy to Use:** The drag-and-drop interface makes it simple to add shapes, icons, and templates to your diagrams.
- **Cloud Integration:** It connects easily with cloud services like Google Drive, Microsoft OneDrive, Dropbox, and GitHub, making it easy to save and share your work.
- **Offline Mode:** It also has desktop versions available, so you can use it without an internet connection.

3. Lucidchart

Lucidchart is an online tool used to create and share diagrams, particularly for business processes and technical designs. It's especially good for teamwork and collaborating with others. Some of the main features include:

- **Comprehensive Diagramming:** It allows you to create flowcharts, mind maps, network diagrams, wireframes, UML diagrams, and more.
- **Data Integration:** Lucidchart can pull data from platforms like Google Sheets, Excel, and Salesforce to make diagrams that are based on real-time data.
- **Real-Time Collaboration:** Multiple people can work on the same diagram at the same time, with options for comments, chat, and tracking changes.
- **Pre-designed Templates:** There are many ready-made templates available, making it easier to get started with your diagrams.

Lab 2: Flowchart Using System Design Tool

Objective:

Learn to design flowcharts representing system workflows.

Description:

A flowchart is a visual representation of a process, system, or workflow. It uses **symbols, shapes, and arrows** to depict the sequence of steps or decisions involved in the process. Flowcharts are widely used in various fields, such as software development, business management, education, and engineering, to simplify complex procedures and enhance understanding.

Key Components of a Flowchart

- **Start/End (Oval)**
    - Represents the beginning or end of a process.
- **Process (Rectangle)**
    - Represents a step or action taken in the process.
- **Decision (Diamond)**
    - Represents a point in the process where a decision must be made.
- **Arrows**
    - Indicate the flow or direction of the process from one step to the next.
- **Input/Output (Parallelogram)**
    - Represents input provided to or output generated by the system.

Advantages of Flowchart

- **Visual Clarity**: Simplifies complex processes by breaking them down into clear, visual steps.
- **Effective Communication**: Facilitates better understanding and communication among team members.
- **Problem Identification**: Helps identify bottlenecks, inefficiencies, and areas for improvement.
- **Standardized Representation**: Ensures consistent depiction of processes across teams and projects.

Disadvantages of Flowchart

- **Time-Consuming**: Creating detailed flowcharts for complex processes can be time-intensive.
- **Rigidity**: Difficult to update or modify as processes change.
- **Limited Scalability**: Becomes cluttered and hard to follow for large or intricate processes.
- **Dependence on Symbols**: Requires knowledge of standard symbols for proper interpretation.
- **Tool Dependency**: May require specific software or tools for creation, which can incur costs.

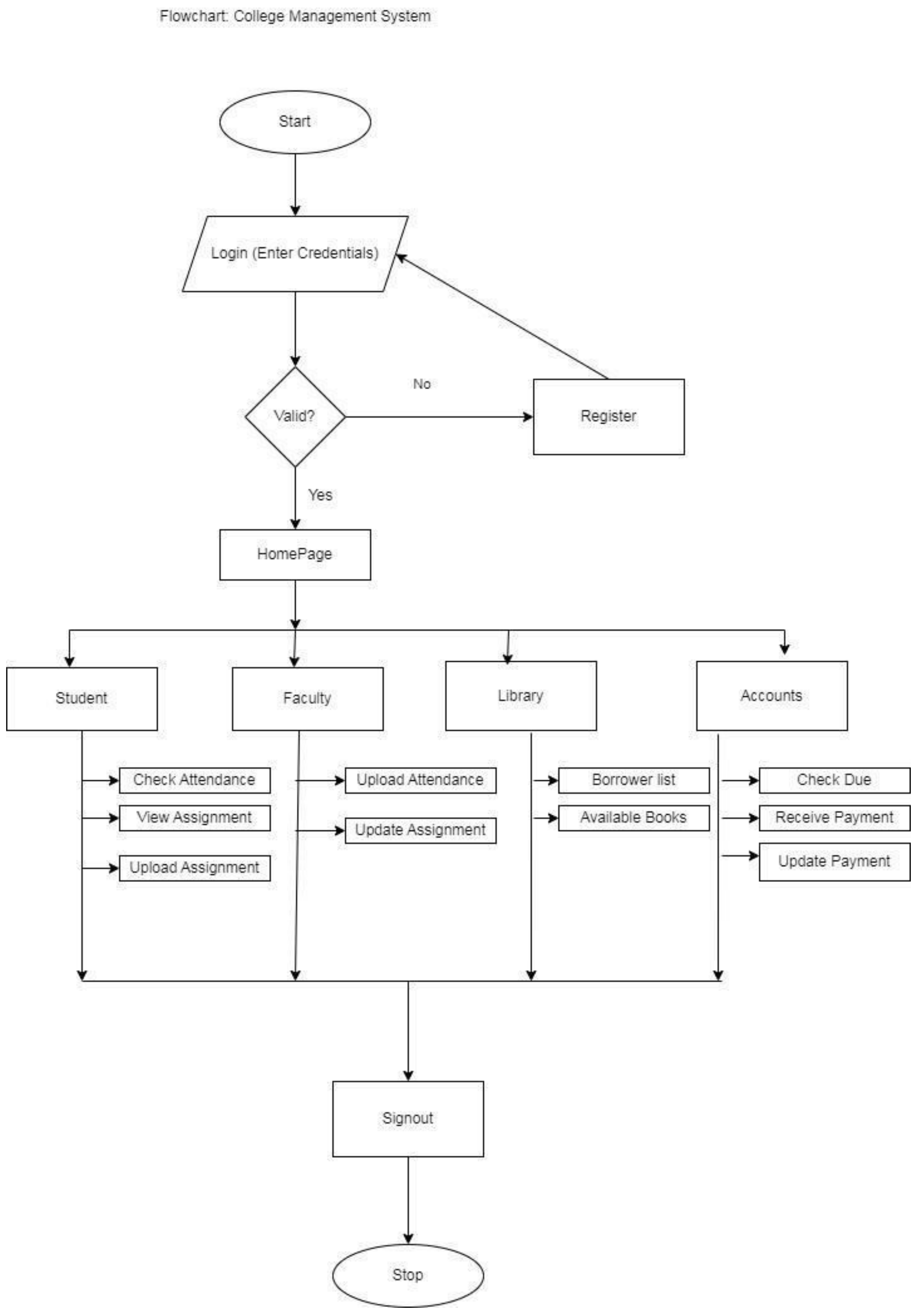


Fig: Flowchart of College Management System

Lab 3: ER Diagram Using System Design Tool

Objective:

Develop an Entity-Relationship (ER) diagram to represent system entities and their relationships.

Description:

Entity-Relationship (ER) Diagram

An ER Diagram is a graphical tool used to represent entities, their attributes, and relationships within a system or database.

Key Components of an ER Diagram

- **Entities**: Represented as rectangles, they denote objects or concepts in the database (e.g., Student, Course).
- **Attributes**: Represented as ovals connected to entities, showing the properties or details of entities (e.g., Name, Age).
- **Relationships**: Represented as diamonds (or lines) connecting entities, they indicate how entities are related (e.g., "Enrolled In").
- **Primary Key (PK)**: A unique identifier for each instance of an entity (e.g., Student ID).
- **Foreign Key (FK)**: An attribute in one entity that refers to the primary key of another entity, establishing a link between them.

Advantages of ER Diagrams

- **Clear Visualization**: Offers an easy-to-understand way to view data and relationships.
- **Simplifies Database Design**: Assists in structuring and organizing data efficiently.
- **Enhances Communication**: Serves as a universal language among database designers, developers, and stakeholders.
- **Problem Identification**: Helps identify issues like redundancies or missing entities early in the design phase.

Disadvantages of ER Diagrams

- **Complexity in Large Systems**: Can become hard to interpret for very large and detailed databases.
- **Limited Representation**: Does not capture dynamic aspects like processes or data behavior.
- **Dependency on Tools**: Creating and editing ER diagrams often requires specialized software.

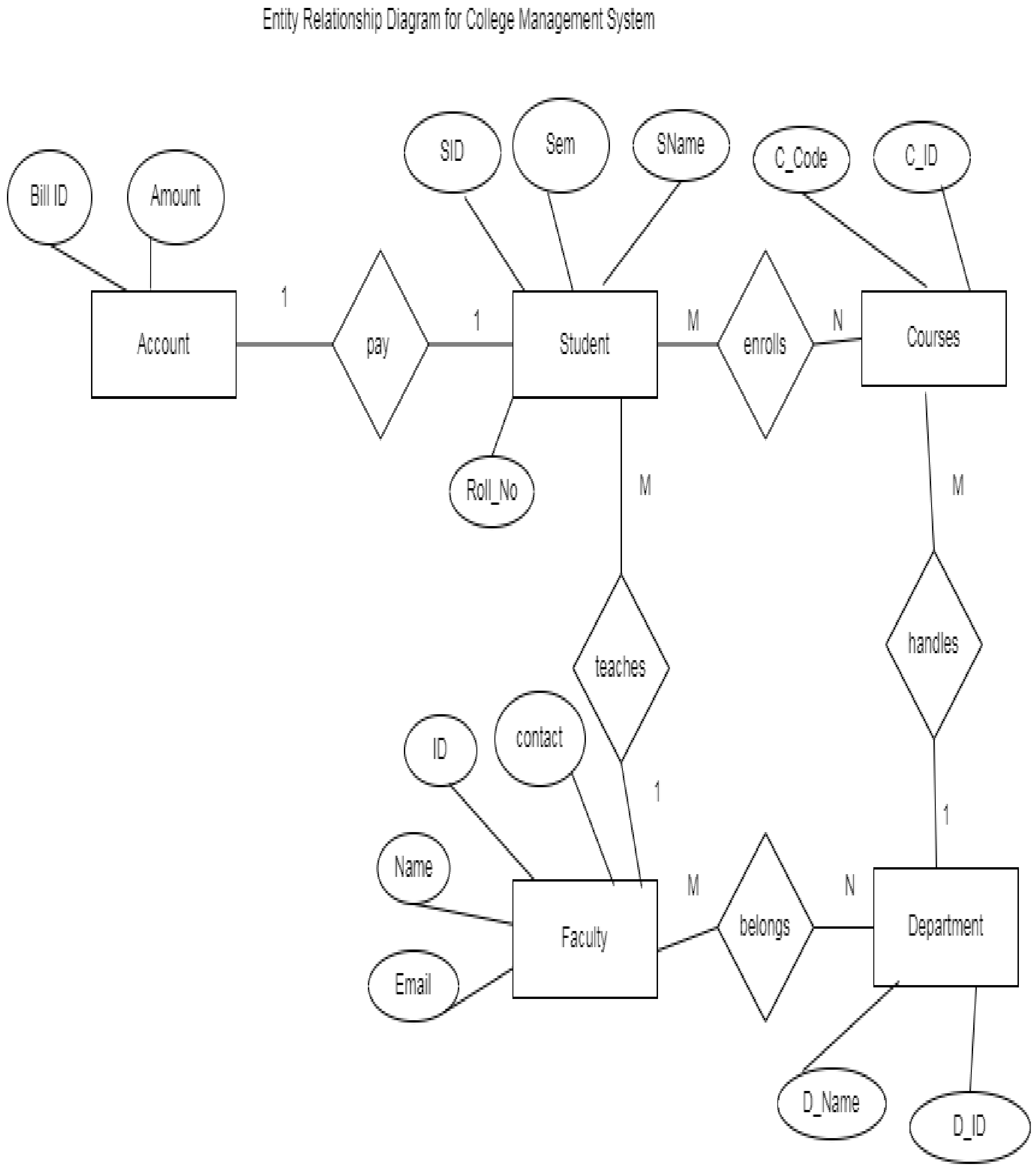


Fig : ER Diagram for College Management System

Lab 4: User Database Design Using Tabular Method

Objective:

Learn to design a database schema using a tabular method.

Description:

The tabular method is a systematic way to organize and present data in table format. It outlines the key elements of a data structure. This method ensures clarity and consistency, making it easier to design, analyze, and manage data structures.

- **Key Components of the Tabular Method**
- **Fields (Columns):**Represent the attributes or properties of the data structure.
- **Data Types:**Specifies the kind of data stored in each field.
- **Constraints:**Defines rules or conditions applied to the data.
- **Relationships (Optional):**Specify links between different tables in the system.

Advantages:

- **Structured Representation**: Ensures data is presented in a clear and organized format.
- **Simplifies Database Development**: Makes designing and managing databases easier and more efficient.
- **Constraint Validation**: Facilitates the verification of constraints and relationships within the data.
- **Data Clarity**: Clearly depicts data types, formats, and rules, improving understanding and consistency.

Disadvantages:

- **High Effort for Complexity**: Requires considerable effort to represent complex schemas accurately.
- **Limited Dynamic Insights**: May fail to capture dynamic interactions or behaviors within the system.
- **Lack of Visual Intuition**: Does not provide the intuitive understanding that diagrams can offer..

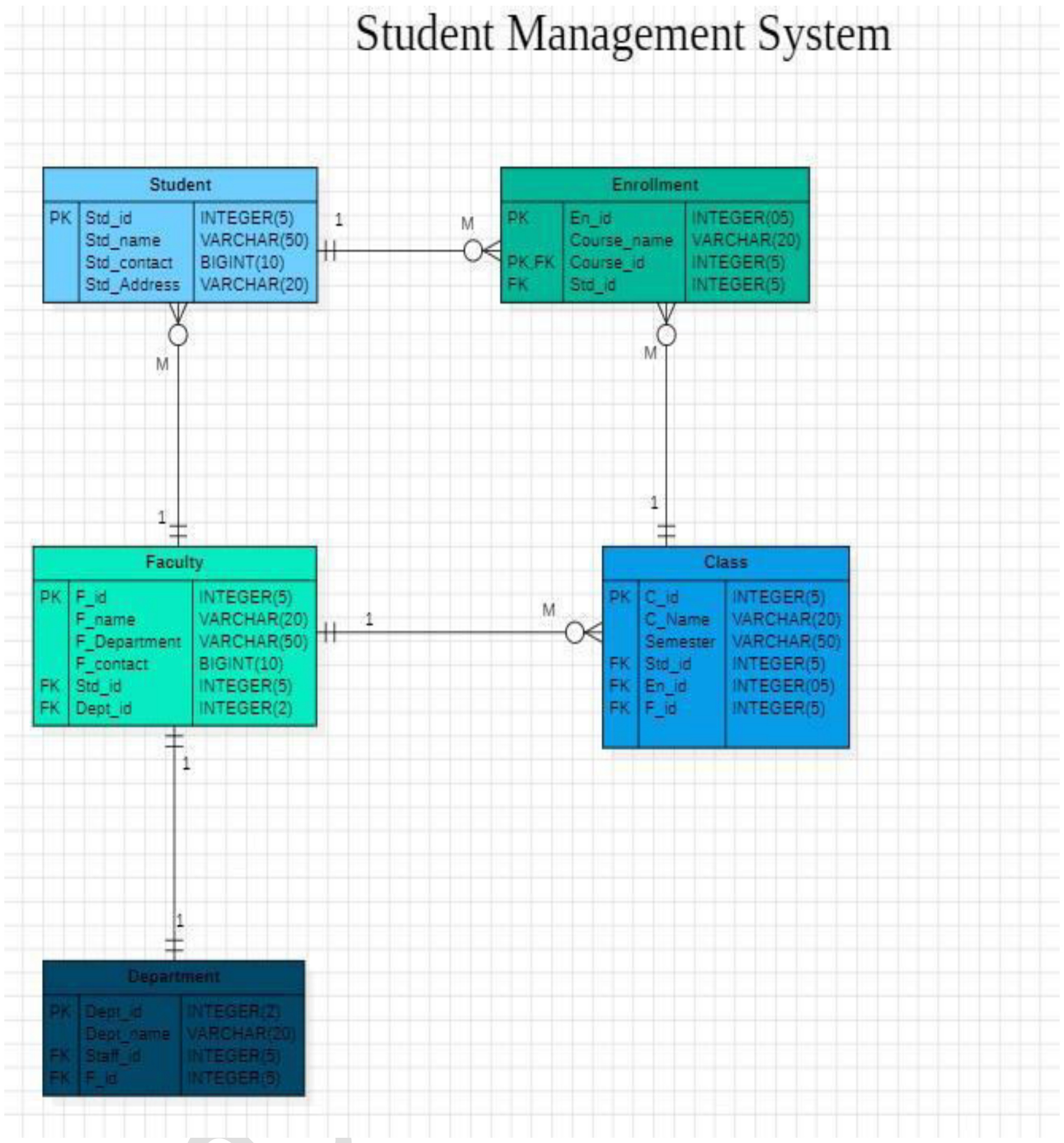


Fig: User Database Design of College Management System using Tabular method

Lab 5: Use Case Diagram Using System Design Tool

Objective:

Understand and create use case diagrams to visualize system functionality.

Description:

A Use Case Diagram is a behavioral diagram in Unified Modeling Language (UML) that depicts the interactions between users (actors) and a system. It highlights the system's functionality and the roles of the actors involved.

Key Elements of a Use Case Diagram

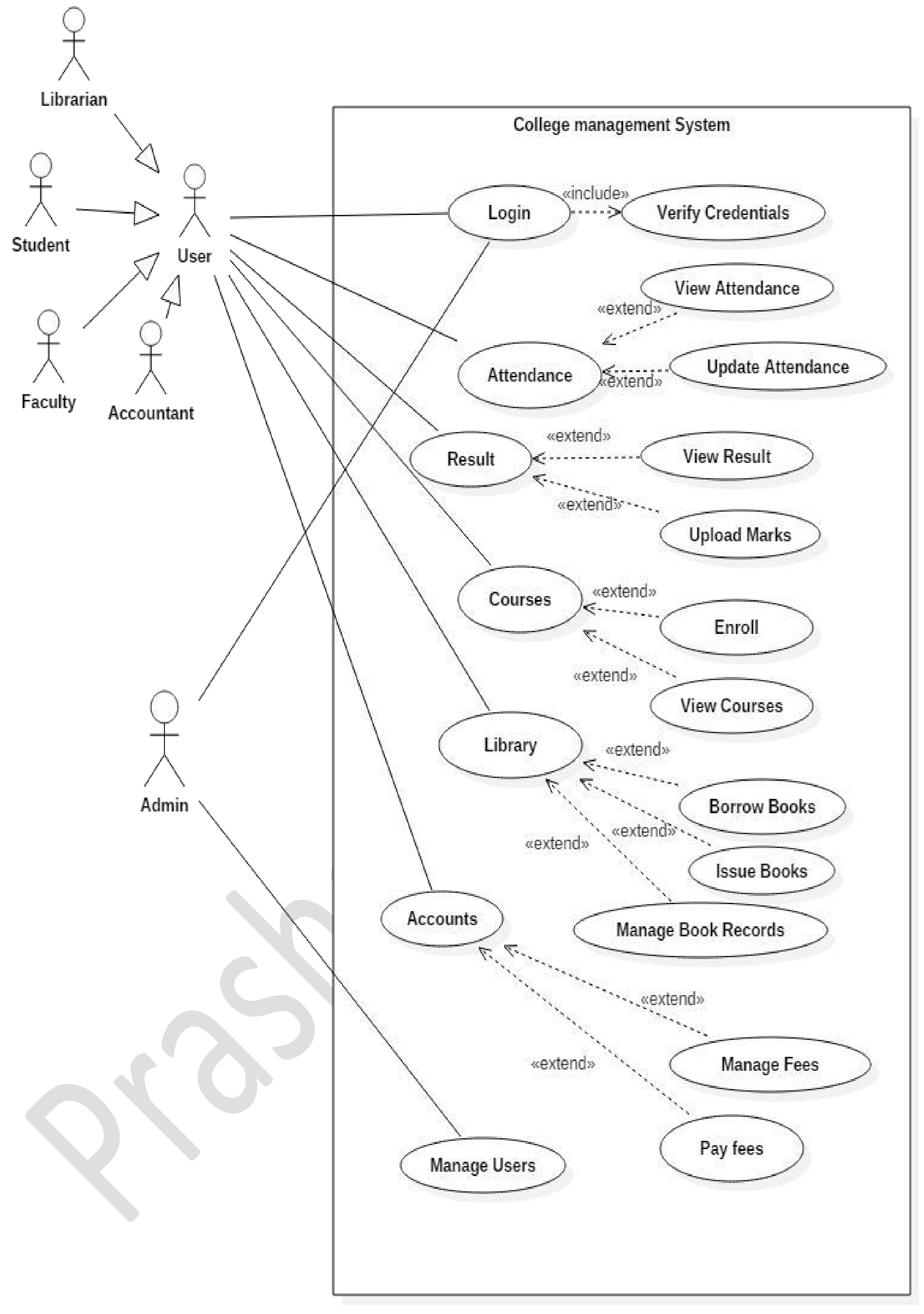
- **Actors**: Represent entities (users or external systems) that interact with the system.
- **Use Cases**: Represent specific actions or functionalities performed by the system.
- **System Boundary**: Defines the scope of the system being modeled.
- **Relationships**: Show the connections between actors and use cases or interactions among use cases.

Advantages:

- **System-User Interaction Overview**: Offers a clear visualization of how users interact with the system.
- **Improves Requirement Clarity**: Helps in understanding functional requirements effectively.
- **Non-Technical Communication**: Simplifies discussions with non-technical stakeholders.
- **Task Prioritization**: Aids in identifying and prioritizing development tasks.

Disadvantages:

- **Oversimplification**: May oversimplify complex systems, leaving out critical details.
- **Limited Dynamic Representation**: Does not effectively capture dynamic behaviors or system workflows.
- **Expertise Required**: Designing accurate diagrams requires knowledge of UML principles.
- **Time-Consuming**: Creating diagrams for systems with many use cases can be labor-intensive.



<u>**Fig**: Usecase diagram of college management system</u>

Lab 6: Activity Diagram Using System Design Tools

Objective:

Develop activity diagrams to represent the dynamic flow of control in the system.

Description:

An Activity Diagram is a behavioral diagram in Unified Modeling Language (UML) that illustrates the workflow or sequence of activities within a system or process. It highlights the flow of control and decisions involved in completing a task.

Key Elements of an Activity Diagram

- **Start Node**:
    - Represents the beginning of the process.
    - Depicted as a filled black circle.
- **Activity**:
    - Represents a task, operation, or action performed in the process.
    - Shown as a rounded rectangle.
- **Decision Node**:
    - Represents a point where the workflow branches based on conditions.
    - Depicted as a diamond.
- **Merge Node**:
    - Combines multiple alternate paths into a single flow.
    - Also represented as a diamond.
- **Fork Node**:
    - Represents a point where the process splits into parallel paths.
    - Shown as a thick horizontal or vertical bar.
- **Join Node**:
    - Combines parallel paths back into a single flow.
    - Depicted as a thick bar, similar to a fork.
- **Transitions/Arrows**:
    - Indicate the flow of control or data between elements.

    - Represented as arrows.

- **End Node**:
    - Represents the end of the process.
    - Depicted as a black circle surrounded by a larger unfilled circle.

- **Swimlanes (Optional)**:
    - Divide the diagram into sections to show which actor, system, or department is responsible for each activity.

Advantages:

- **Intuitive Workflow Representation**: Clearly illustrates the flow of activities in a system or process.
- **Highlights Key Elements**: Effectively shows decision points and parallel actions within workflows.
- **Versatile Audience**: Easy to understand for both technical and non-technical stakeholders.
- **Process Improvement**: Helps identify inefficiencies or bottlenecks in workflows.

.

Disadvantages:

- **Increased Complexity**: Becomes harder to manage and understand as the system grows larger.
- **Requires UML Knowledge**: Designing effective diagrams demands familiarity with UML symbols and conventions.
- **Hard to Modify**: Changes become difficult once the design becomes complex.

Activity Diagram for College Management System

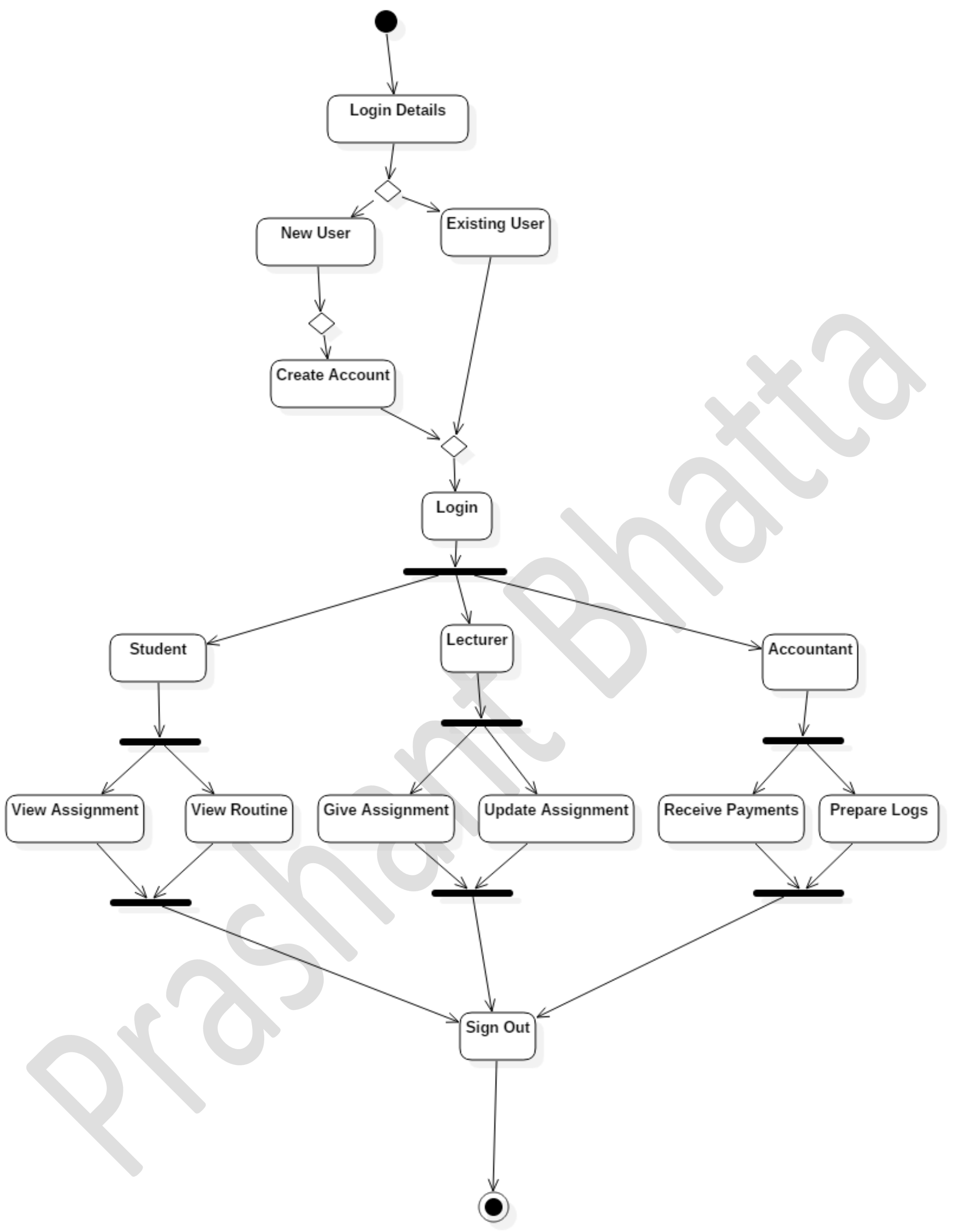


Fig : Activity Diagram for College Management system

Lab 7: Class Diagram Using System Design Tools

Objective:

Understand and create class diagrams to represent the static structure of a system.

Description:

A Class Diagram is a static structure diagram in Unified Modeling Language (UML) that depicts the system's structure by illustrating its classes, attributes, methods, and the relationships between objects. It provides a blueprint of the system's design and its key components.

Key Components of a Class Diagram

- **Class**: A rectangle divided into three parts (name, attributes, methods).
- **Attributes**: Properties or data members of a class.
- **Methods**: Functions or behaviors a class can perform.
- **Relationships**:

- **Association**: A link between two classes.
- **Aggregation**: Whole-part relationship; parts can exist independently.
- **Composition**: Stronger aggregation; parts depend on the whole.
- **Generalization**: "Is-a" relationship (inheritance).
- **Realization**: Contract between an interface and its implementation.
- **Visibility**: Access levels for attributes and methods (+ public, - private, # protected).

Advantages:

- **Clear Representation**: Clearly shows system classes and their relationships.
- **Object-Oriented Design**: Essential for designing object-oriented systems.
- **Detailed Overview**: Highlights attributes and methods effectively.
- **Dependency Analysis**: Helps identify system dependencies and hierarchies.

Disadvantages:

- **Complexity**: Can become overwhelming with too many classes.
- **Time-Consuming**: Requires significant effort to design and maintain.
- **Iterative Refinement**: Often needs multiple updates for accuracy.
- **UML Expertise**: Requires knowledge of UML for proper understanding and usage.

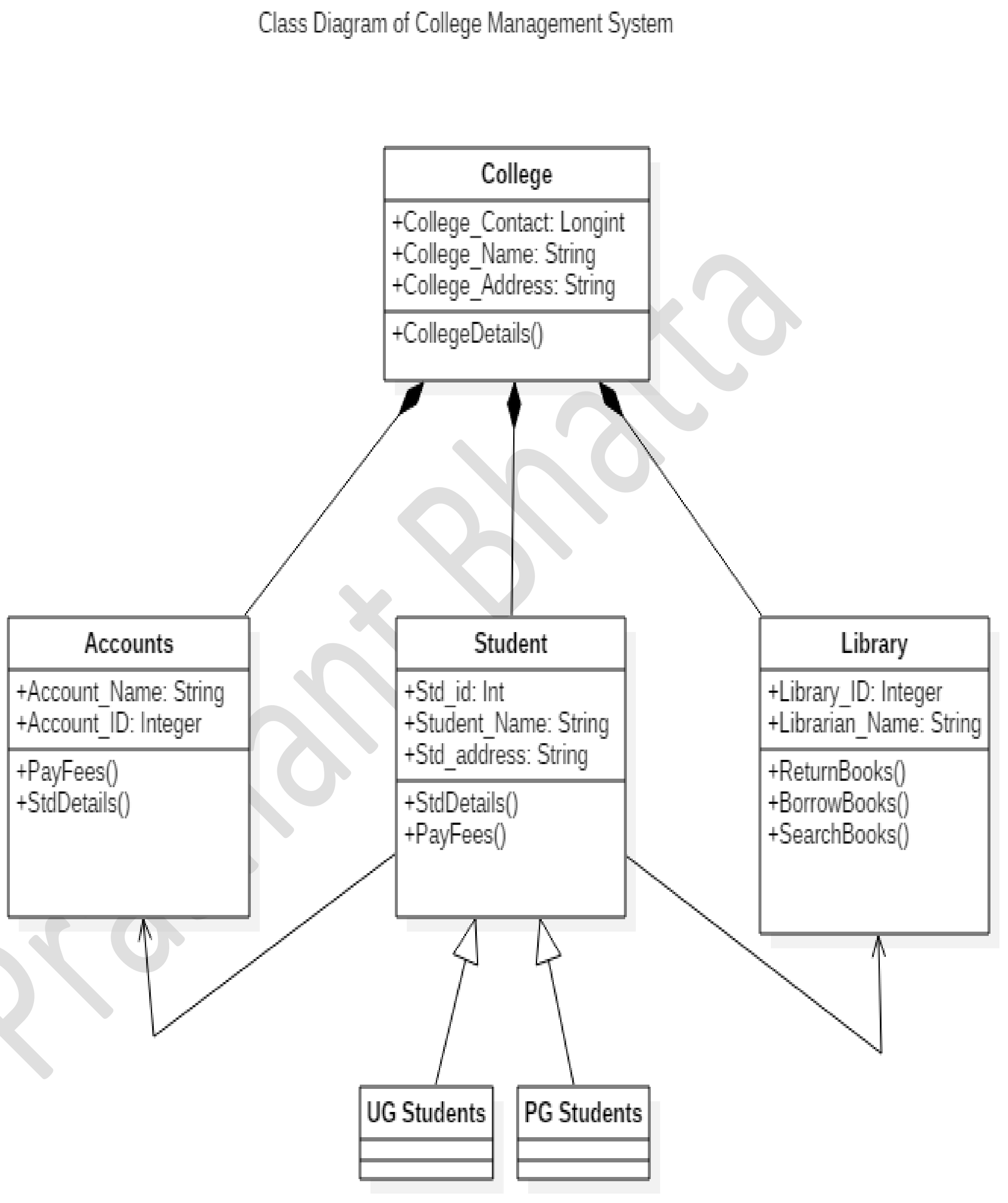


Fig: Class Diagram of College Management System

Lab 8: Object Diagram Using System Design Tools

Objective:

Design object diagrams to visualize object relationships at a specific time.

Description:

An Object Diagram is a UML (Unified Modeling Language) diagram that captures a snapshot of a system at a specific moment in time. It depicts instances of classes (objects) and the relationships between them, providing a concrete representation of the system's state.

Key Components of an Object Diagram

- **Objects:**Represent instances of classes.
- **Attributes:**Represent the specific values of attributes for a particular object.
- **Links (Relationships):**Represent the relationships or references between objects.
- **Multiplicity (Optional):**Indicates how many instances of one object can be associated with another.

Advantages:

- **System State Representation**: Offers a snapshot of the system's state at a specific time.
- **Debugging and Testing**: Useful for identifying and analyzing issues in the system.
- **Object Interaction**: Enhances understanding of how objects interact in real scenarios.
- **Class Diagram Complement**: Provides a dynamic representation to support class diagrams.

.**Disadvantages:**

1. **Limited Applicability**: Useful only in specific scenarios.
2. **Time-Intensive**: Designing for large systems can be laborious.
3. **Requires System Knowledge**: Demands a detailed understanding of the system's structure.
4. **Consistency Challenges**: Difficult to maintain consistency across multiple iterations.

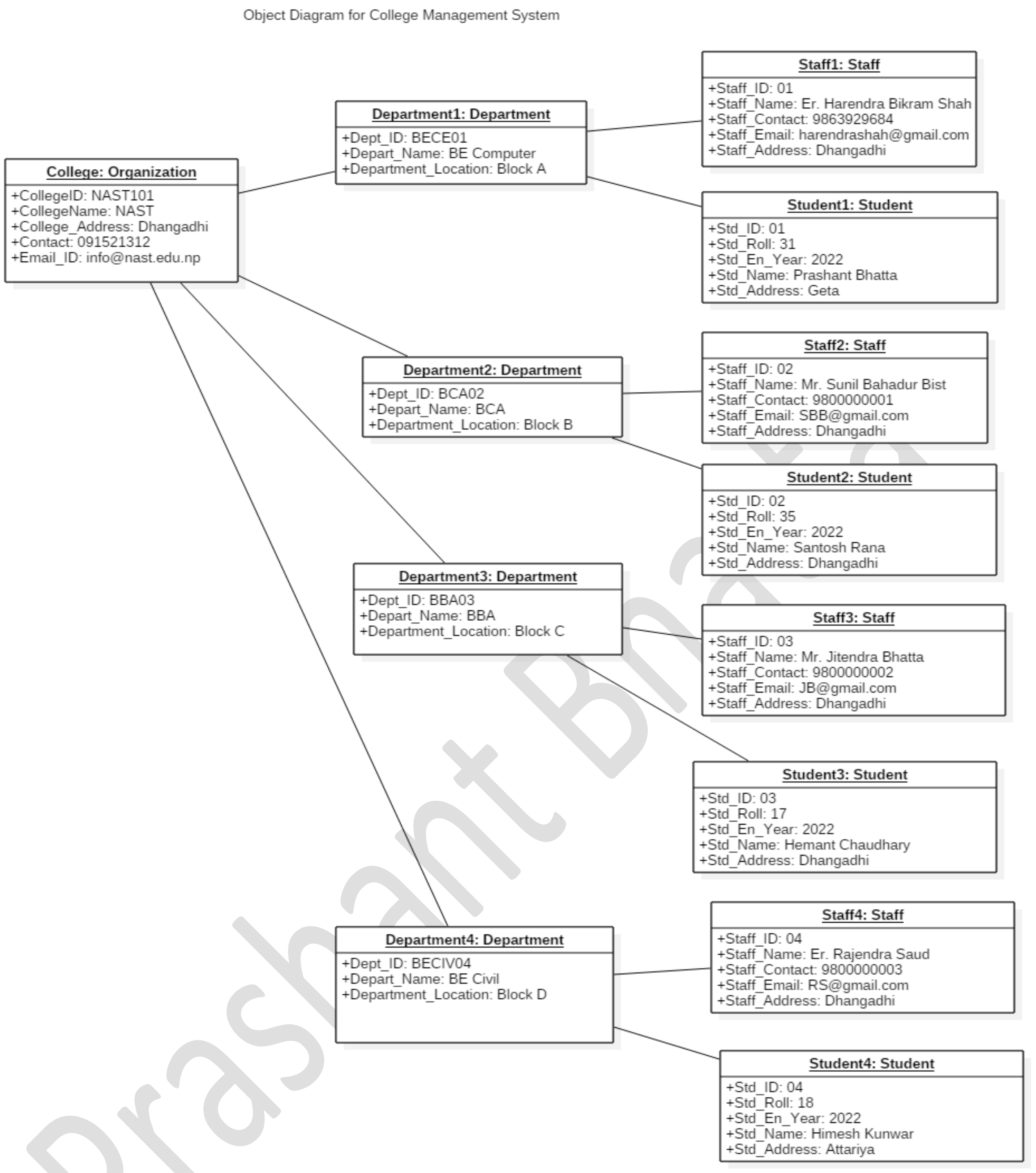


Fig: Object diagram of College Management System

Lab 9: Sequence Diagram Using System Design Tools

Objective:

To understand and practice the development of a sequence diagram to model interactions between objects in a system.

Description:

A Sequence Diagram is a UML (Unified Modeling Language) diagram that illustrates the interaction between objects in a system in a time-sequential order. It highlights the dynamic behavior of the system by showing how messages are exchanged between objects to complete a specific task.

Purpose:

- Visualize the flow of messages in a system.
- Represent dynamic behavior.
- Analyze and design a system.

Key Components of an Sequence Diagram

- **Actors**: Represent external users or systems interacting with the system
- **Objects/Entities**: Represent system components or classes involved in interaction.
- **Messages**: Arrows show communication between objects.
- **Lifelines**: Dashed lines extending downward to show the lifetime of an object.
- **Activation Bars**: Indicate periods during which an object is active.
- **Return Messages**: Dotted arrows showing responses

Advantages:

- **Clear Message Flow**: Clearly depicts the flow of messages between objects.
- **Object Interactions**: Highlights how objects interact over time.
- **Dynamic Behavior Understanding**: Useful for analyzing and understanding the system's dynamic behavior.
- **Design and Debugging Aid**: Assists in designing and debugging system processes effectively.

Disadvantages:

- **Complexity**: Becomes harder to interpret with many objects.
- **Time-Consuming**: Creating detailed interactions can be labor-intensive.
- **Difficult to Modify**: Adjustments can be challenging after the diagram is created.

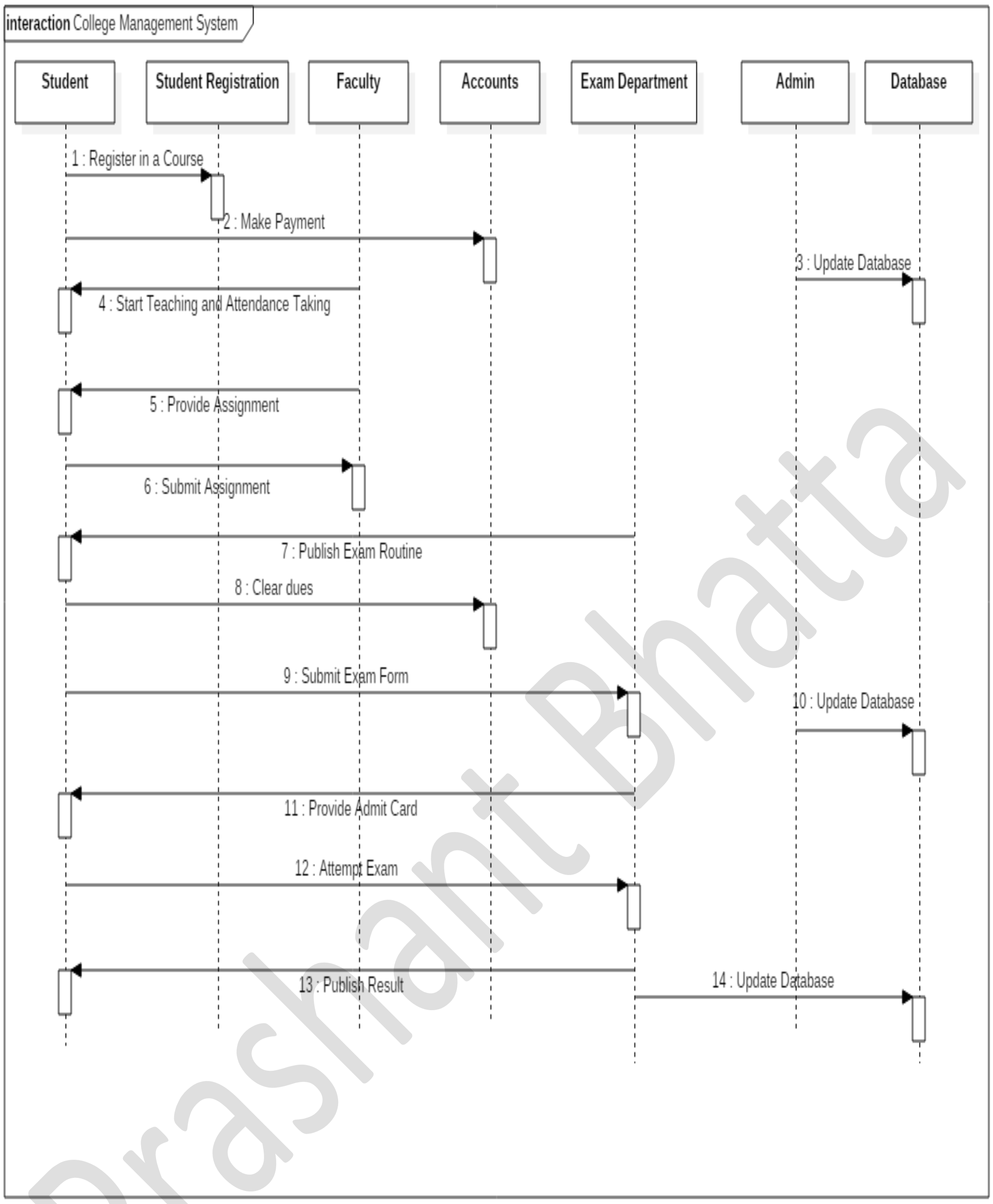


Fig : Sequence Diagram of College Management System

Lab 10: Communication Diagram Using System Design Tools

Objective:

Design communication diagrams to visualize object or component interactions in a system during runtime.

Description:

A Communication Diagram is a UML (Unified Modeling Language) diagram that focuses on the interactions between objects or components through message exchanges. It illustrates how objects collaborate to perform specific system behaviors or tasks.

Key Tools for Communication Diagram in StarUML:

- **Lifeline**: Represents an object or participant in the communication.
- **Connector**: Represents the links or relationships between objects, indicating possible communication.
- **Self Connector**:Represents self-referencing communication (when an object interacts with itself).
- **Forward Message**:Represents a **message** sent from one object to another.Includes the **sequence number** and a description of the action (e.g., 1: processOrder()).
- **Reverse Message**:Represents a **response message** (e.g., confirmation or data returned) sent back to the caller.

Advantages:

- **Object Collaborations**: Effectively highlights how objects collaborate.
- **Simple Message Exchange**: Provides a simple representation of message exchanges between objects.
- **Runtime Interactions**: Useful for understanding how objects interact during runtime.
- **Sequence Diagram Complement**: Complements sequence diagrams by offering another dynamic view.

Disadvantages:

- **Non-Intuitive for Non-Tech Users**: Can be hard for non-technical users to understand.
- **Time-Consuming**: Requires significant time for complex systems.
- **Limited Process Timing**: Does not capture the timing of processes well..

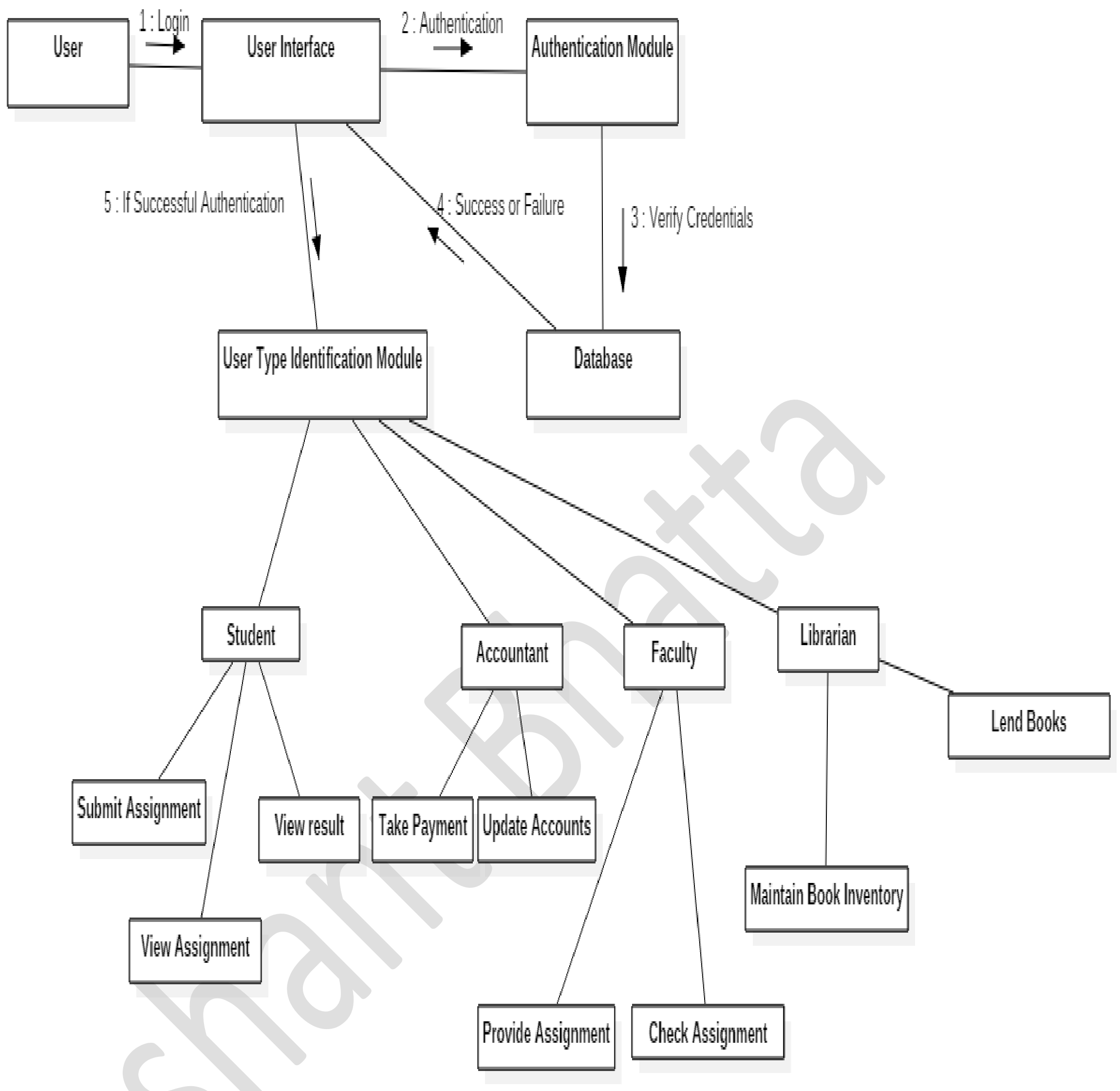


Fig: Communication Diagram of College Management System

Lab 11: Designing a DFD (Data Flow Diagram) Using System Design Tools

Objective:

To understand and apply the principles of designing a Data Flow Diagram (DFD) using system design tools, illustrating the flow of data within a system.

Description:

A Data Flow Diagram (DFD) is a graphical representation of the flow of data through a system. It helps visualize how data is processed, stored, and communicated in a system.

Components of a DFD:

- **External Entities:** Represent sources or destinations of data.
- **Processes:** Represent operations or transformations performed on data.
- **Data Stores:** Represent storage locations for data.
- **Data Flows:** Represent the movement of data between entities, processes, and data stores.

The levels of Data Flow Diagrams (DFD’s) are as follows:

Type 0: **Context Diagram:** This is the highest-level DFD that represents the system as a single process and shows its interaction with external entities. It doesn't break down any processes inside the system.

- **Type 1**: **Level 1 DFD** (or sometimes **First-Level DFD**): This DFD shows the system’s major sub-processes or functional components. It provides a breakdown of the system but still remains relatively high-level.
- **Type 2**: **Level 2 DFD** (or sometimes **Second-Level DFD**): This diagram further decomposes the processes from the Level 1 DFD into more detailed sub-processes. It provides a deeper level of granularity.
- **Type 3**: **Level 3 DFD** (or sometimes **Third-Level DFD**): This level provides even more detail than Level 2, breaking down complex processes into specific tasks, conditions, or interactions.

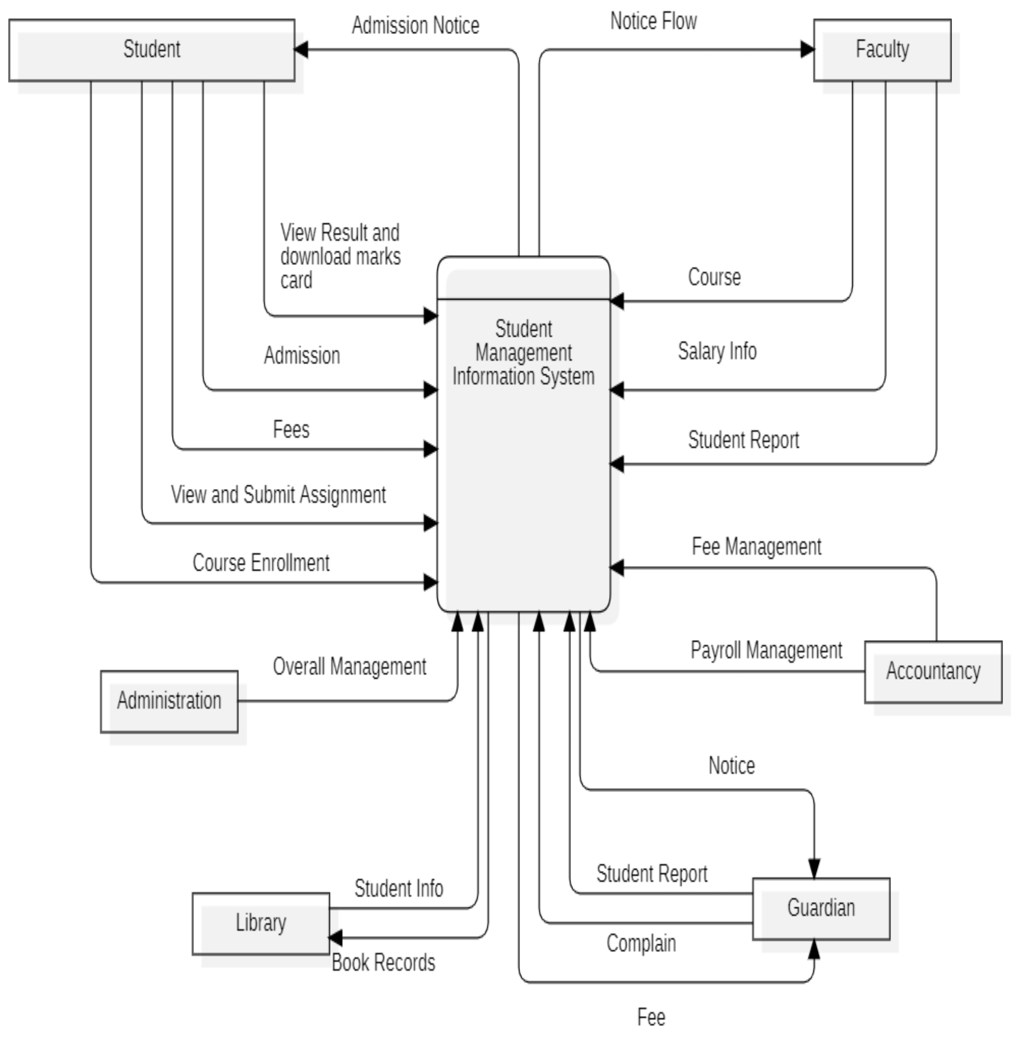


Fig: DFD of College Management System

Lab 12: Designing a Package Diagram Using System Design Tools

Objective

To understand and apply the principles of designing a Package Diagram using system design tools, illustrating the modular structure and dependencies within a system.

Description

A Package Diagram is a UML (Unified Modeling Language) structure diagram that illustrates the organization and dependencies among different packages within a system. It offers a high-level view of the system's modular architecture by grouping related elements, like classes or subsystems, into packages. This makes complex software architectures easier to understand and manage.

Components of a Package Diagram:

- **Packages**: Represent a group of related classes, interfaces, or sub-packages.
- **Classes/Interfaces**: Show specific elements within a package.
- **Dependencies**: Represent relationships between packages, indicating how changes in one package might affect another.
- **Sub-Packages**: Represent nested packages that further divide a package into more specific groups.

Advantages:

- **High-Level Overview**: Offers a clear view of the system's structure.
- **Simplifies Understanding**: Makes it easier to comprehend system modules.
- **Dependency Visualization**: Highlights dependencies and relationships between packages.
- **Manages Complexity**: Helps organize and manage complex architectures effectively.

Disadvantages:

- **Limited Detail**: Does not show the internal workings of packages.
- **UML Expertise Required**: Needs knowledge of UML for proper creation and interpretation.

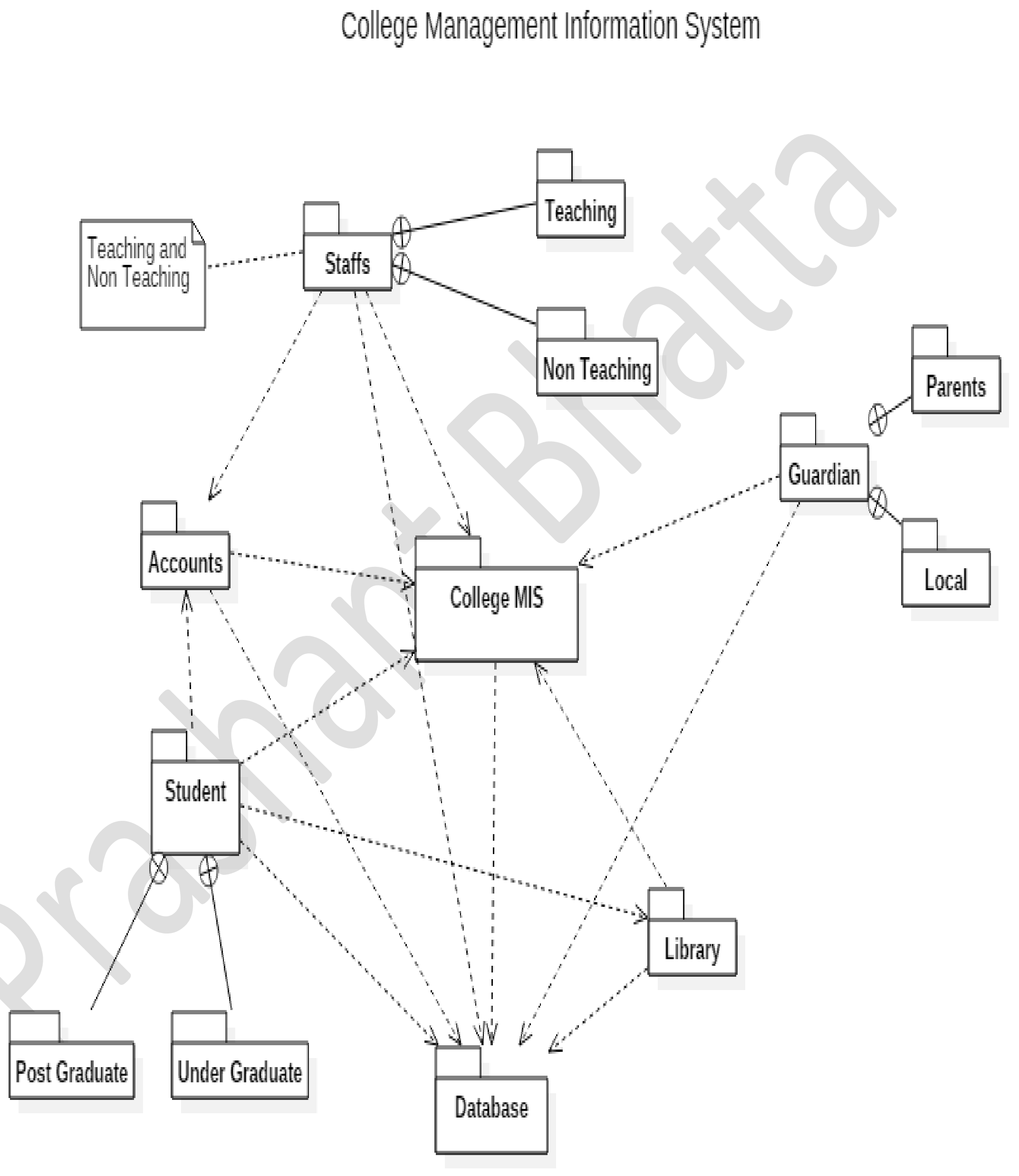


Fig: Package Diagram of College MIS

Lab 13: Designing a Deployment Diagram Using System Design Tools

Objective

The primary objective of this lab is to design a deployment diagram using system design tools. The deployment diagram visually represents the physical deployment of artifacts (software components) on nodes (hardware devices) in a system.

Introduction

A Deployment Diagram is an important UML tool used in system design to represent the physical architecture of a system. It maps software artifacts to hardware nodes, illustrating how software components are deployed on physical hardware.

It highlights the relationship between software and hardware, aiding in understanding the system architecture and deployment process.

Components :

- **Nodes**: Represent physical or virtual machines where components are deployed.
- **Artifacts**: Software components such as executables, libraries, or database scripts.
- **Connections**: Define the communication between nodes

Advantages:

- **Clear Mapping**: Maps software components to hardware nodes, providing clarity.
- **System Architecture Understanding**: Enhances understanding of the physical structure of the system.
- **Deployment & Maintenance**: Crucial for planning deployment and system maintenance.
- **Communication Highlights**: Shows the communication and interactions between nodes.

Disadvantages:

- **Complexity**: Becomes more complex with large-scale systems.
- **Limited Runtime Representation**: Doesn't capture dynamic runtime processes effectively.
- **Hardware Knowledge Required**: Requires in-depth understanding of the system's hardware.
- **Difficult to Update**: Challenging to update when there are system changes.

.

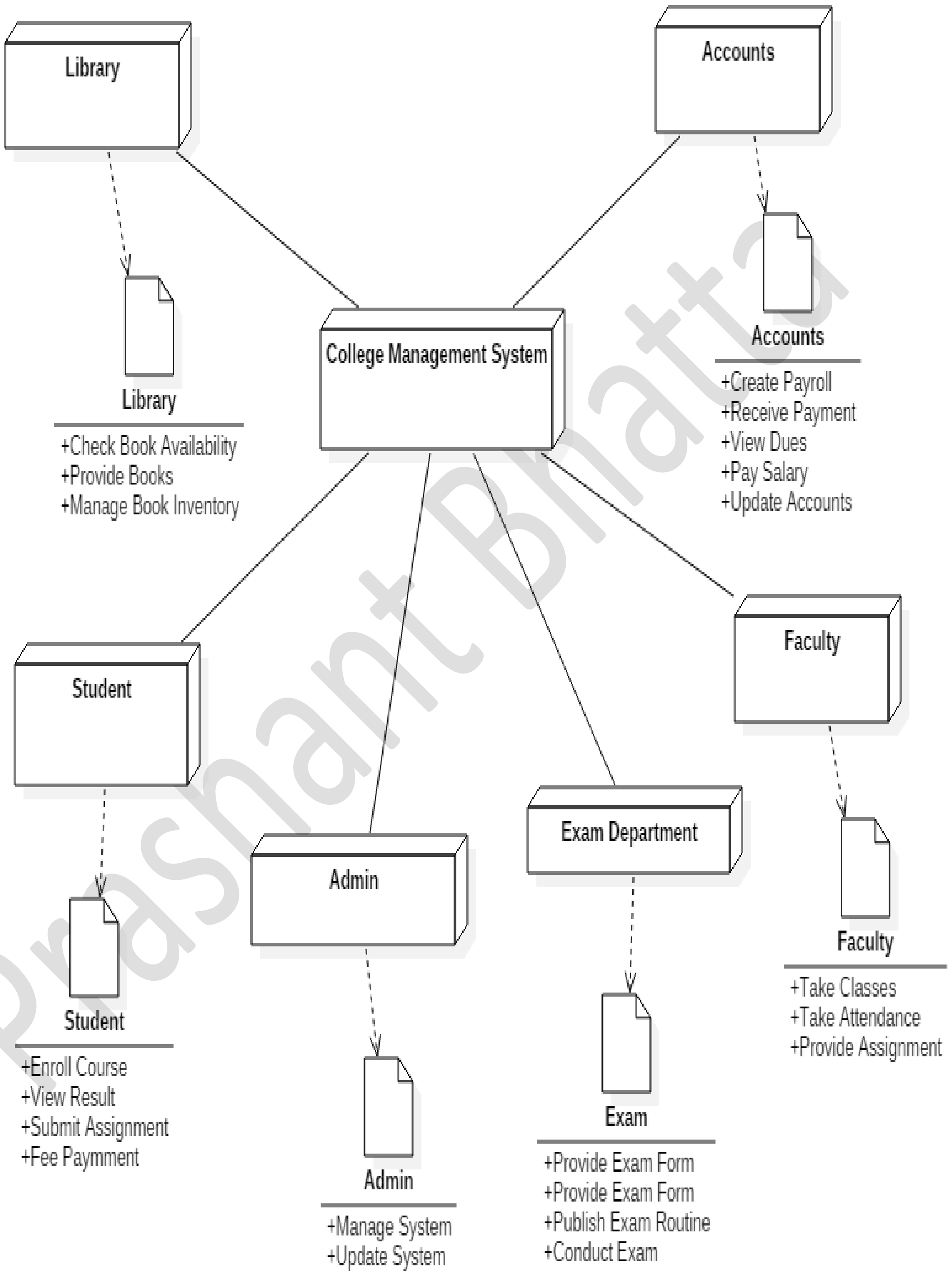


Fig : Deployment Diagram of College Management System

Lab 14: Designing a Data Flow Diagram Using System Design Tools

Objective:

To understand and apply the principles of designing a Data Flow Diagram (DFD) using system design tools, illustrating the flow of data within a system.

Description:

A Data Flow Diagram (DFD) is a visual tool that represents the flow of data within a system. It illustrates how data is processed, stored, and transferred between different components or entities in the system.

Components of a DFD:

- **External Entities:** Represent sources or destinations of data outside the system.
- **Processes:** Represent operations or transformations that occur on the data.
- **Data Stores:** Represent storage locations for data.
- **Data Flows:** Represent the movement of data between entities, processes, and data stores.

Advantages of DFD (Data Flow Diagram):

- **Simplifies System Understanding**: Provides a clear and visual representation of how data flows through a system, making it easier for stakeholders to understand complex processes.
- **Identifies System Requirements**: Helps in identifying inputs, outputs, and processing requirements, aiding in effective system analysis and design.
- **Improves Communication**: Acts as a communication bridge between developers, analysts, and non-technical stakeholders.
- **Facilitates Problem Identification**: Makes it easier to pinpoint inefficiencies, redundancies, or missing components in a system.

Disadvantages of DFD:

- **Lacks Detail**: Does not specify how data processing is performed (e.g., algorithms or coding logic).
- **Time-Consuming**: Creating detailed DFDs for complex systems can be time intensive and laborious.

The levels of Data Flow Diagrams (DFD's) are as follows:

Type 0: **Context Diagram:** This is the highest-level DFD that represents the system as a single process and shows its interaction with external entities. It doesn't break down any processes inside the system.

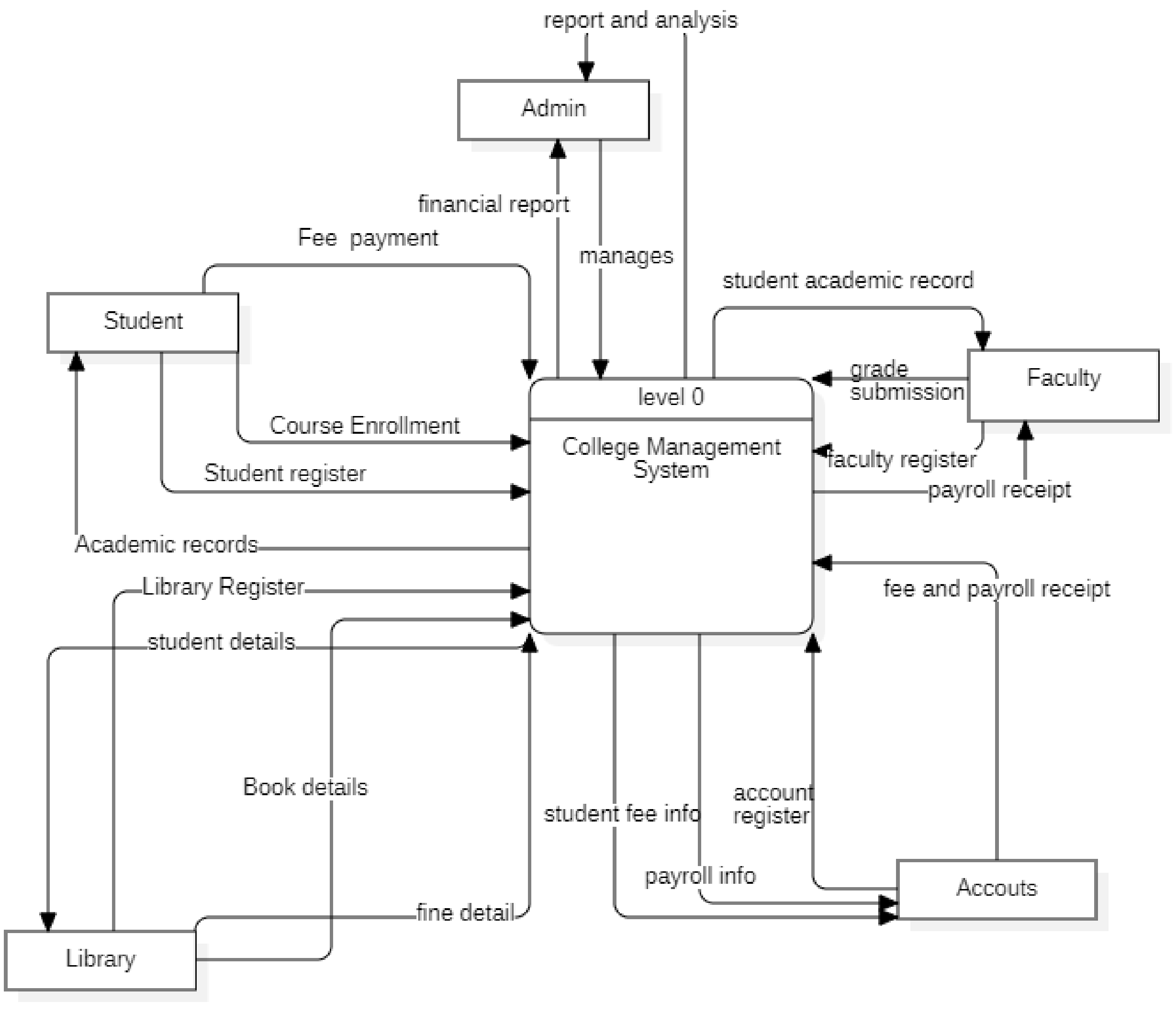


Fig : Level 0 DFD of College management system

Type 1: **Level 1 DFD** (or sometimes **First-Level DFD**): This DFD shows the system's major sub-processes or functional components. It provides a breakdown of the system but still remains relatively high-level.

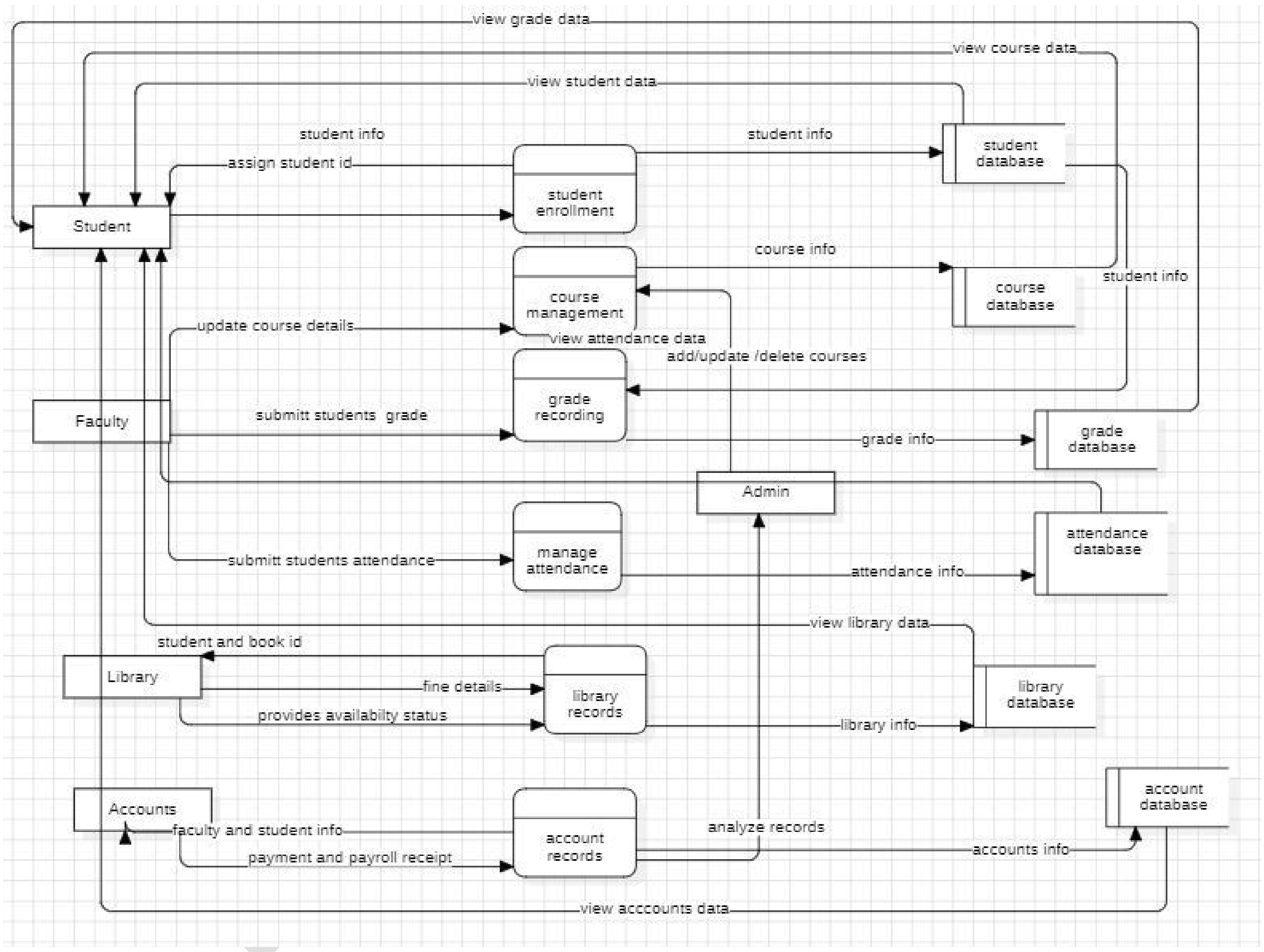


Fig : Level 1 DFD of College management system

Type 2: **Level 2 DFD** (or sometimes **Second-Level DFD**): This diagram further decomposes the processes from the Level 1 DFD into more detailed sub-processes. It provides a deeper level of granularity.

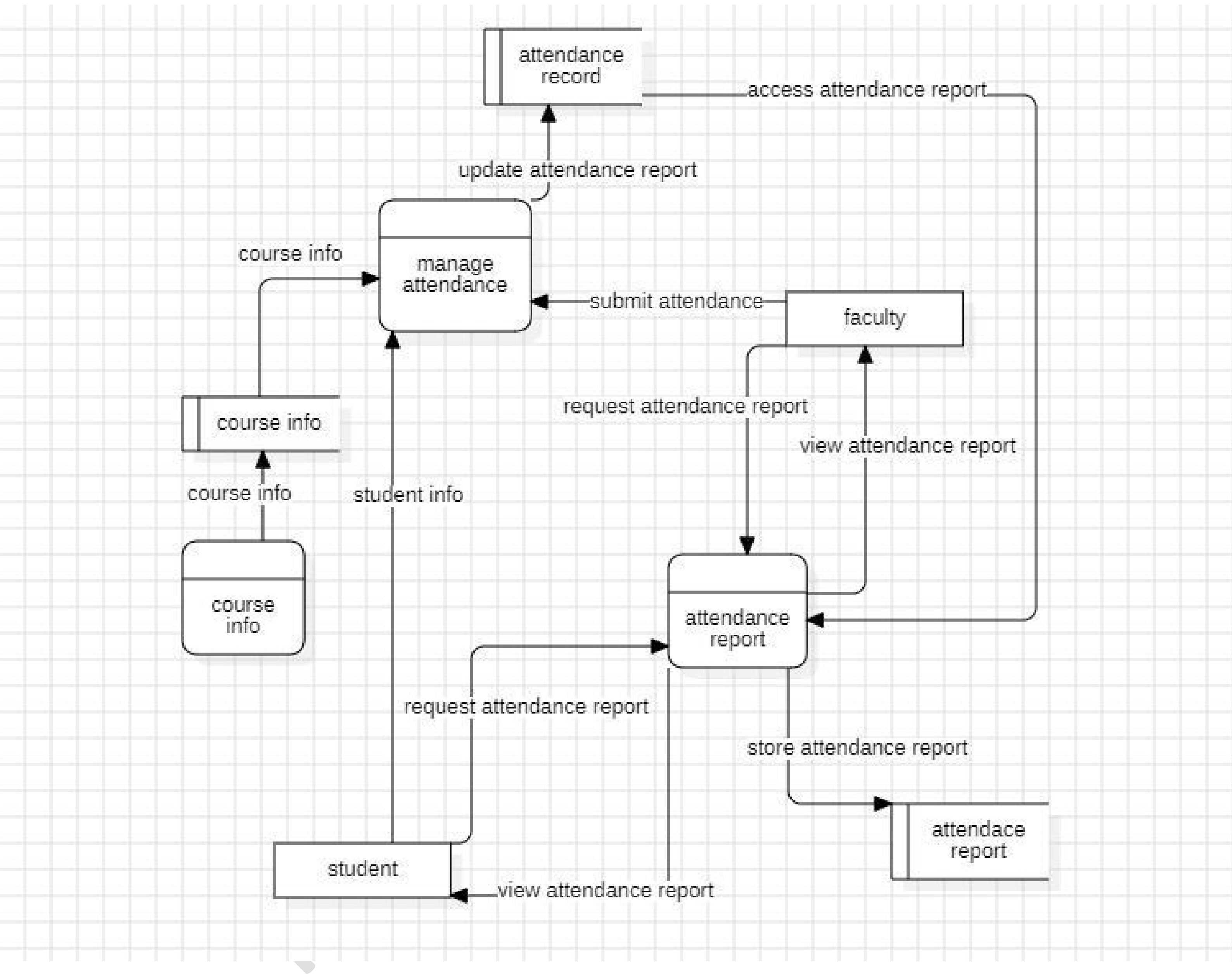


Fig: Level 2 DFD of College Management System's attendance process

Lab 15: Introduction to ZIRA tool

Objective

To gain an understanding of the Zira tool, its features, functionalities, and its use in project management, particularly in software development teams.

Introduction

Zira is a project management and workflow automation tool designed to streamline collaboration and productivity within teams. It is widely used in software engineering for task tracking, team coordination, and automating repetitive workflows. Zira integrates with development platforms and tools such as GitHub, allowing developers to manage issues, pull requests, and project milestones effectively.

Key features of Zira include:

- Workflow automation using triggers and actions.
- Customizable project boards.
- Integration with version control systems.
- Advanced reporting and analytics.

Materials and Tools Required

- A computer with an active internet connection.
- Access to the Zira tool (free or licensed version).
- A GitHub account (optional, for integration purposes).
- A browser or Zira desktop application.

When to Use Zira

Zira should be used when:

- Managing complex projects with multiple team members.
- Automating repetitive tasks in workflows to save time.
- Collaborating across departments or remote teams.
- Monitoring project progress and generating detailed reports.
- Tracking tasks, issues, and pull requests in software development.

Why Use Zira

Zira is used because it:

- Enhances team productivity through streamlined workflows.
- Provides a centralized platform for task management and communication.
- Automates routine tasks, reducing manual errors.
- Integrates seamlessly with popular tools like GitHub, enabling better collaboration.
- Offers powerful analytics and reporting to make data-driven decisions.

Advantages of Zira

- **User-Friendly Interface:** Easy to navigate and use.
- **Automation:** Saves time by automating repetitive processes.
- **Integration:** Works with tools like GitHub and Slack for a unified experience.
- **Customization:** Highly customizable boards and workflows to fit team needs.
- **Improved Collaboration:** Facilitates better communication and task delegation.
- **Scalability:** Suitable for teams of all sizes.

Applications of Zira

- **Software Development:** Managing issues, pull requests, and project milestones.
- **Agile Project Management:** Using Kanban and Scrum methodologies.
- **Marketing Campaigns:** Tracking tasks and deadlines.
- **Human Resources:** Onboarding workflows and employee task management.
- **Education:** Managing projects and assignments for academic purposes.

Lab 16: Creating a bug and resolving it using zira tool

Objectives

- To understand the process of creating a bug in Jira.
- To learn how to assign, prioritize, and resolve bugs using Jira.
- To gain hands-on experience with Jira's interface and features.

Introduction

In software engineering, bug tracking and resolution are critical aspects of the software development lifecycle. Tools like Jira help teams efficiently manage, track, and resolve bugs. This lab report demonstrates the process of creating a bug and resolving it using Jira, a popular project management and issue-tracking tool.

Tools and Technologies Used

- Jira Software : For bug tracking and project management.
- Web Browser : To access Jira's web interface.
- Sample Project : A dummy software project for demonstration purposes.

Steps to Create and Resolve a Bug in Jira

Step 1: Log in to Jira

✓ Open your web browser and navigate to your Jira instance.
✓ Log in with your credentials.

Step 2: Create a New Project
- If you don’t have a project, create one by clicking on "Create Project."
- Choose a project template (e.g., Software Development).

Step 3: Create a Bug Issue
- Navigate to the project board.
- Click on "Create" to add a new issue.
- Select "Bug" as the issue type.
- Fill in the required details:
  - Summary : Brief description of the bug.
  - Description : Detailed explanation of the bug.
  - Priority : Set the priority (e.g., High, Medium, Low).
  - Assignee : Assign the bug to a team member.

Step 4: Assign and Prioritize the Bug
- Assign the bug to a developer.
- Set the priority based on the severity of the bug.

Step 5: Reproduce and Analyze the Bug
- The assigned developer reproduces the bug in the development environment.
- They analyze the root cause of the bug.

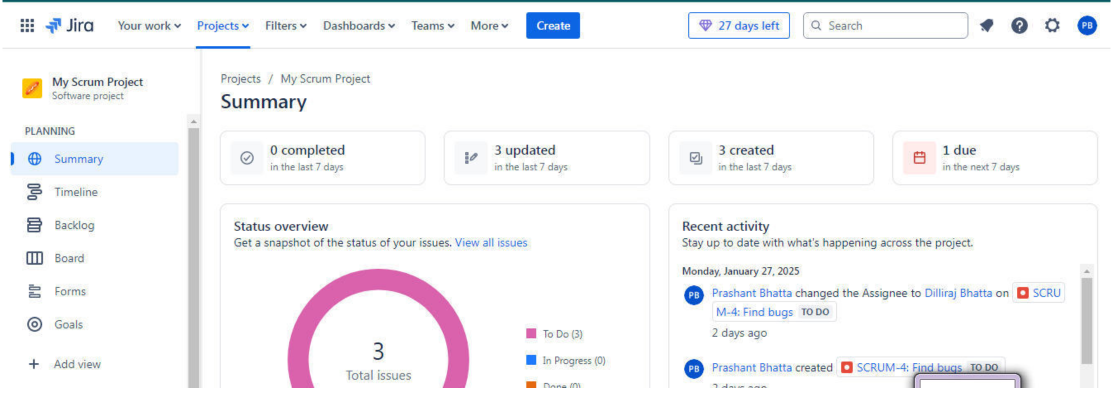
Step 6: Resolve the Bug
- Once the bug is fixed, the developer updates the issue status to "Resolved."
- Add comments describing the solution.

Step 7: Verify the Fix
- A tester or QA engineer verifies the fix.
- If the bug is resolved, the issue status is updated to "Closed."

Screenshots

Below are the screenshots of the process:

1. Jira Dashboard :



Projects / My Scrum Project

Summary

0 completed in the last 7 days

3 updated in the last 7 days

3 created in the last 7 days

1 due in the next 7 days

Status overview

Get a snapshot of the status of your issues. View all issues

3 Total issues

To Do (3)

In Progress (0)

Done (0)

Recent activity

Stay up to date with what's happening across the project.

Monday, January 27, 2025

Prashant Bhatta changed the Assignee to Dilliraj Bhatta on SCRUM-4: Find bugs TO DO

2 days ago

Prashant Bhatta created SCRUM-4: Find bugs TO DO

Fig : Jira Dashboard

2. Bug Details :

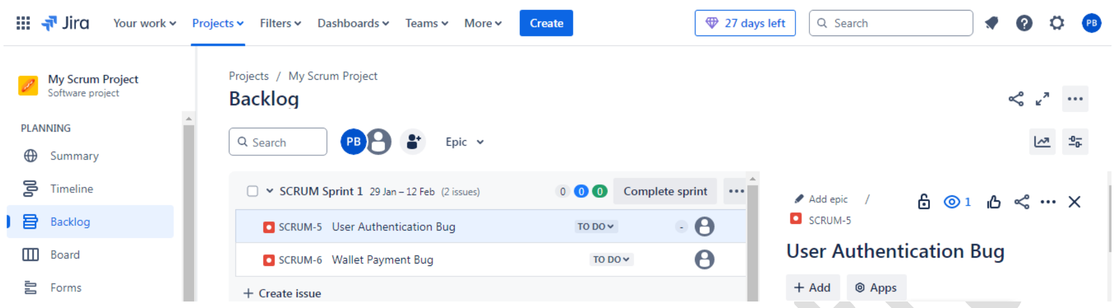


Fig: Bug Details

3. Bug Resolution :

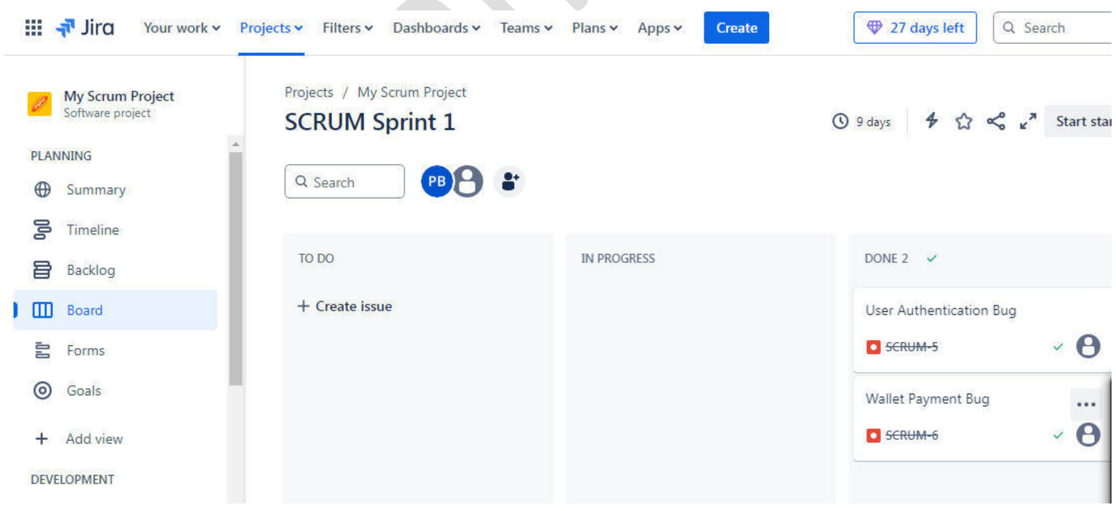


Fig : Bug Resolution

Lab 17: Creating a E-commerce project using Zira tool

Objectives

- To create an e-commerce project in Jira.
- To define epics and user stories for the project.
- To create and resolve bugs related to the user stories.
- To gain hands-on experience with Jira's features for project management.

Introduction

In software engineering, project management tools like Jira are essential for organizing and tracking tasks, especially in complex projects like e-commerce systems. This lab report demonstrates the creation of an e-commerce project in Jira, including epics, user stories, and bug tracking. The epics include Login , Validation , Payment , Payment Validation , and Delivery .

Tools and Technologies Used

- Jira Software : For project management and issue tracking.
- Web Browser : To access Jira's web interface.
- Sample E-Commerce Project : A dummy project for demonstration purposes.

Steps to Create an E-Commerce Project in Jira

Step 1: Log in to Jira

1. - Open your web browser and navigate to your Jira instance.
2. - Log in with your credentials.

Step 2: Create a New Project

3. - Click on "Create Project."
4. - Choose a project template (e.g., Scrum).
5. - Name the project "E-Commerce Platform."

Step 3: Define Epics

6. - Navigate to the "Backlog" or "Board" view.
7. - Create the following epics:

◆ Login : User authentication and authorization.
◆ Validation : Input validation for forms and data.
◆ Payment : Payment gateway integration.
◆ Payment Validation : Validation of payment details.
◆ Delivery : Order tracking and delivery management.

Step 4: Create User Stories for Each Epic

8. - For each epic, create user stories to break down the tasks. Below are examples:

Epic 1: Login
Story 1 : As a user, I want to log in using my email and password so that I can access my account.
Story 2 : As a user, I want to reset my password if I forget it.
Bug : Login fails for users with special characters in their email.

Epic 2: Validation
Story 1 : As a user, I want my email to be validated during registration so that I can provide a valid email address.
Story 2 : As a user, I want my password to meet complexity requirements so that my account is secure.
Bug : Validation error message is not user-friendly.

Epic 3: Payment
Story 1 : As a user, I want to pay using my credit card so that I can complete my purchase.
Story 2 : As a user, I want to see a confirmation page after payment.
Bug : Payment fails for international cards.

Epic 4: Payment Validation
Story 1 : As a user, I want my credit card details to be validated before payment.
Story 2 : As a user, I want to be notified if my payment is declined.
Bug : CVV validation is not working.

Epic 5: Delivery
Story 1 : As a user, I want to track my order so that I know its status.
Story 2 : As a user, I want to receive notifications about my delivery.
Bug : Delivery status is not updating in real-time.

Step 5: Assign and Prioritize Tasks

9. - Assign each user story to a team member.

10. - Set priorities for the stories (e.g., High, Medium, Low).

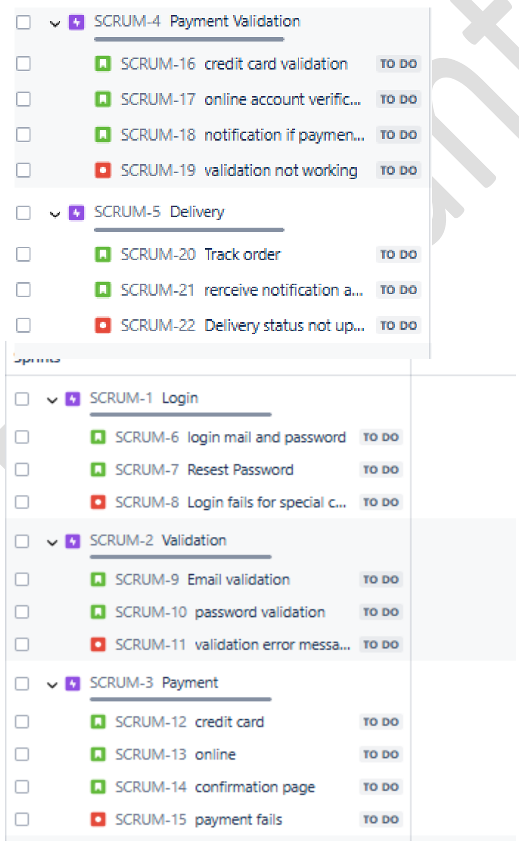
Step 6: Create and Resolve Bugs

- For each bug, create a separate issue in Jira.
- Assign the bug to a developer.
- Once resolved, update the status to "Resolved" and verify the fix.

Step 7: Track Progress

- Use the Jira board to track the progress of epics, stories, and bugs.
- Move tasks across columns (e.g., To Do, In Progress, Done).

Screenshots :



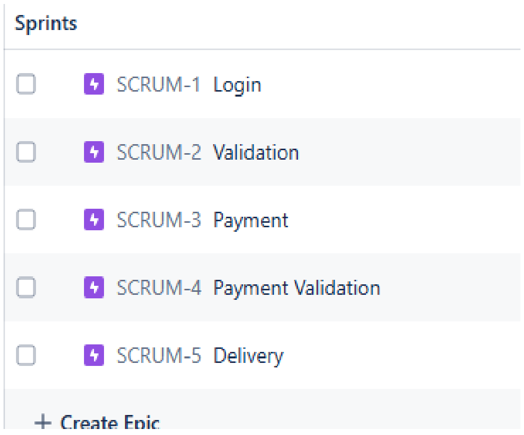
SCRUM-4 Payment Validation
- SCRUM-16 credit card validation TO DO
- SCRUM-17 online account verific... TO DO
- SCRUM-18 notification if paymen... TO DO
- SCRUM-19 validation not working TO DO

SCRUM-5 Delivery
- SCRUM-20 Track order TO DO
- SCRUM-21 rerceive notification a... TO DO
- SCRUM-22 Delivery status not up... TO DO

SCRUM-1 Login
- SCRUM-6 login mail and password TO DO
- SCRUM-7 Resest Password TO DO
- SCRUM-8 Login fails for special c... TO DO

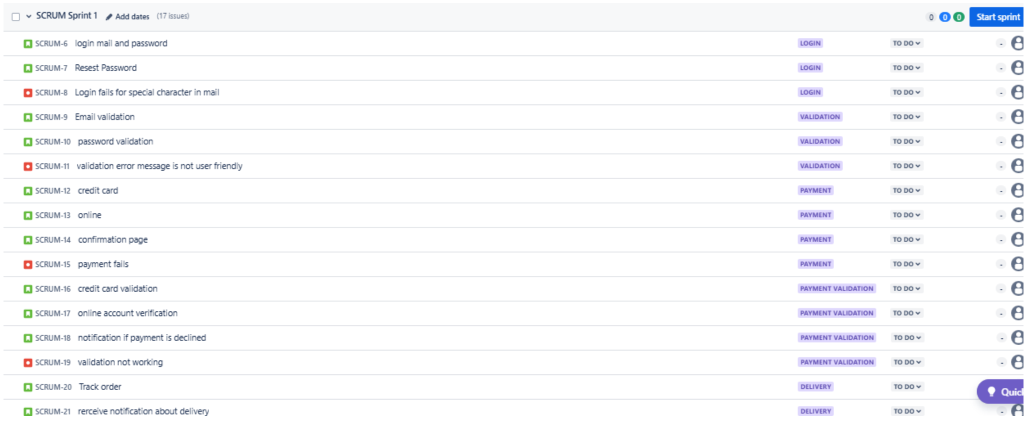
SCRUM-2 Validation
- SCRUM-9 Email validation TO DO
- SCRUM-10 password validation TO DO
- SCRUM-11 validation error messa... TO DO

SCRUM-3 Payment
- SCRUM-12 credit card TO DO
- SCRUM-13 online TO DO
- SCRUM-14 confirmation page TO DO
- SCRUM-15 payment fails TO DO



Sprints
- SCRUM-1 Login
- SCRUM-2 Validation
- SCRUM-3 Payment
- SCRUM-4 Payment Validation
- SCRUM-5 Delivery

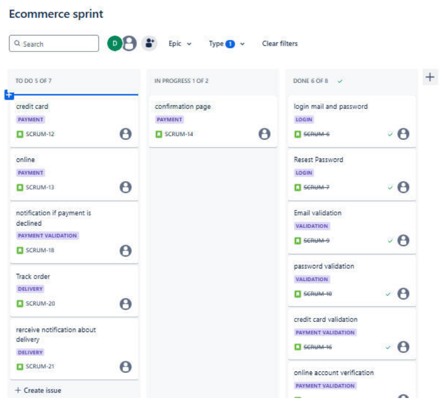
Fig: creation of Epics require for project



SCRUM Sprint 1 · Add dates (17 issues) · Start sprint

🟩 SCRUM-6	login mail and password	LOGIN	TO DO
🟩 SCRUM-7	Resest Password	LOGIN	TO DO
🟥 SCRUM-8	Login fails for special character in mail	LOGIN	TO DO
🟩 SCRUM-9	Email validation	VALIDATION	TO DO
🟩 SCRUM-10	password validation	VALIDATION	TO DO
🟥 SCRUM-11	validation error message is not user friendly	VALIDATION	TO DO
🟩 SCRUM-12	credit card	PAYMENT	TO DO
🟩 SCRUM-13	online	PAYMENT	TO DO
🟩 SCRUM-14	confirmation page	PAYMENT	TO DO
🟥 SCRUM-15	payment fails	PAYMENT	TO DO
🟩 SCRUM-16	credit card validation	PAYMENT VALIDATION	TO DO
🟩 SCRUM-17	online account verification	PAYMENT VALIDATION	TO DO
🟩 SCRUM-18	notification if payment is declined	PAYMENT VALIDATION	TO DO
🟥 SCRUM-19	validation not working	PAYMENT VALIDATION	TO DO
🟩 SCRUM-20	Track order	DELIVERY	TO DO
🟩 SCRUM-21	rerceive notification about delivery	DELIVERY	TO DO

Fig : adding stories and bugs to epics



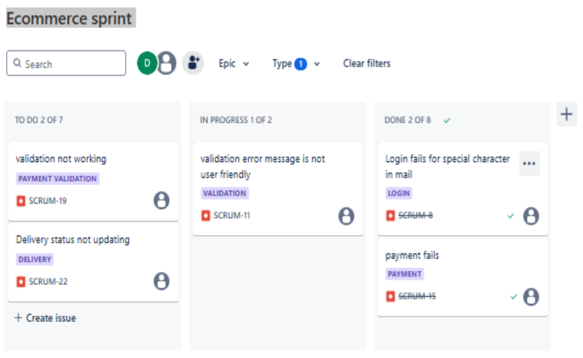
Ecommerce sprint

Search · Epic · Type 1 · Clear filters

TO DO 5 OF 7
- credit card — PAYMENT — SCRUM-12
- online — PAYMENT — SCRUM-13
- notification if payment is declined — PAYMENT VALIDATION — SCRUM-18
- Track order — DELIVERY — SCRUM-20
- rerceive notification about delivery — DELIVERY — SCRUM-21
- + Create issue

IN PROGRESS 1 OF 2
- confirmation page — PAYMENT — SCRUM-14

DONE 6 OF 8 ✓
- login mail and password — LOGIN — ~~SCRUM-6~~ ✓
- Resest Password — LOGIN — ~~SCRUM-7~~ ✓
- Email validation — VALIDATION — ~~SCRUM-9~~ ✓
- password validation — VALIDATION — ~~SCRUM-10~~ ✓
- credit card validation — PAYMENT VALIDATION — ~~SCRUM-16~~ ✓
- online account verification — PAYMENT VALIDATION



Ecommerce sprint

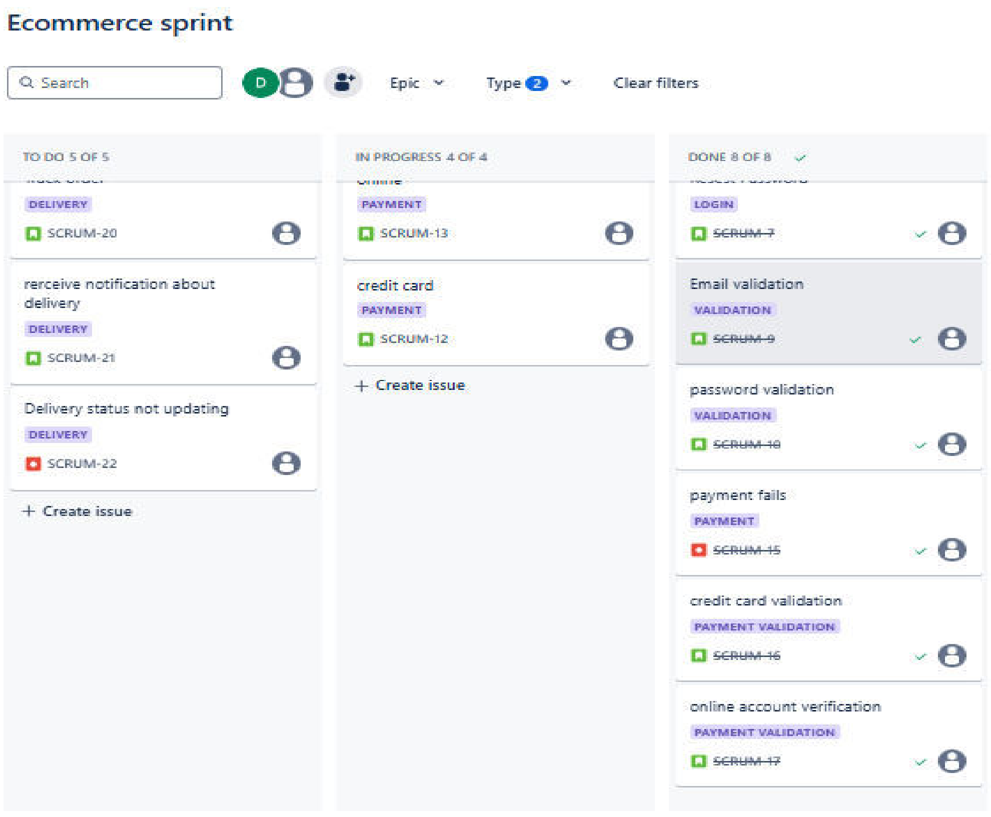
Search · Epic · Type 1 · Clear filters

TO DO 2 OF 7
- validation not working — PAYMENT VALIDATION — SCRUM-19
- Delivery status not updating — DELIVERY — SCRUM-22
- + Create issue

IN PROGRESS 1 OF 2
- validation error message is not user friendly — VALIDATION — SCRUM-11

DONE 2 OF 8 ✓
- Login fails for special character in mail — LOGIN — ~~SCRUM-8~~ ✓
- payment fails — PAYMENT — ~~SCRUM-15~~ ✓

Fig: stories and bugs in sprint



Ecommerce sprint

Search · Epic · Type 2 · Clear filters

TO DO 5 OF 5
- DELIVERY — SCRUM-20
- rerceive notification about delivery — DELIVERY — SCRUM-21
- Delivery status not updating — DELIVERY — SCRUM-22
- + Create issue

IN PROGRESS 4 OF 4
- PAYMENT — SCRUM-13
- credit card — PAYMENT — SCRUM-12
- + Create issue

DONE 8 OF 8 ✓
- LOGIN — ~~SCRUM-7~~ ✓
- Email validation — VALIDATION — ~~SCRUM-9~~ ✓
- password validation — VALIDATION — ~~SCRUM-10~~ ✓
- payment fails — PAYMENT — ~~SCRUM-15~~ ✓
- credit card validation — PAYMENT VALIDATION — ~~SCRUM-16~~ ✓
- online account verification — PAYMENT VALIDATION — ~~SCRUM-17~~ ✓

Fig: Complete Ecommerce project