Gamified Learning Progress Tracker’s HLD

1. Use Case diagram:

A use case diagram is a visual representation of the interactions between users (actors) and a system, focusing on the functionalities the system provides. It helps to illustrate the high-level functionality of a system and the different ways users can interact with it.

Use case diagrams are typically high-level and don't show detailed steps within each use case.

Here's use case diagram components are:

a. Actors:

Actors represent entities outside the system that interact with it, such as users, external systems, or devices. They are depicted as stick figures or other symbols on the diagram.

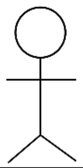


Fig: Actor

b. Use Cases:

These are the specific tasks or actions that the system needs to perform. Each use case represents a particular way the system is used. They're shown as ovals on the diagram.

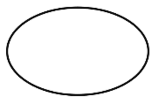


Fig: Use case

c. Relationships:

Relationships between actors and use cases, and between different use cases, are depicted using lines and arrows.

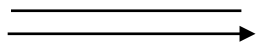


Fig: Relationship

d. System boundary:

This is like the border of the diagram and it defines what's inside our system and what's outside. It's like drawing a line around our system to show its limits.

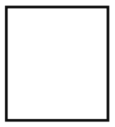


Fig: System boundary

Benefits of using Use Case Diagrams:

- **Visualization:** Provides a clear and concise overview of the system's functionality.
- **Communication:** Improves communication and understanding between stakeholders (developers, users, analysts).
- **Analysis:** Helps identify missing functionalities, inconsistencies, and potential problems.
- **Requirement gathering:** Aids in capturing and documenting user requirements.

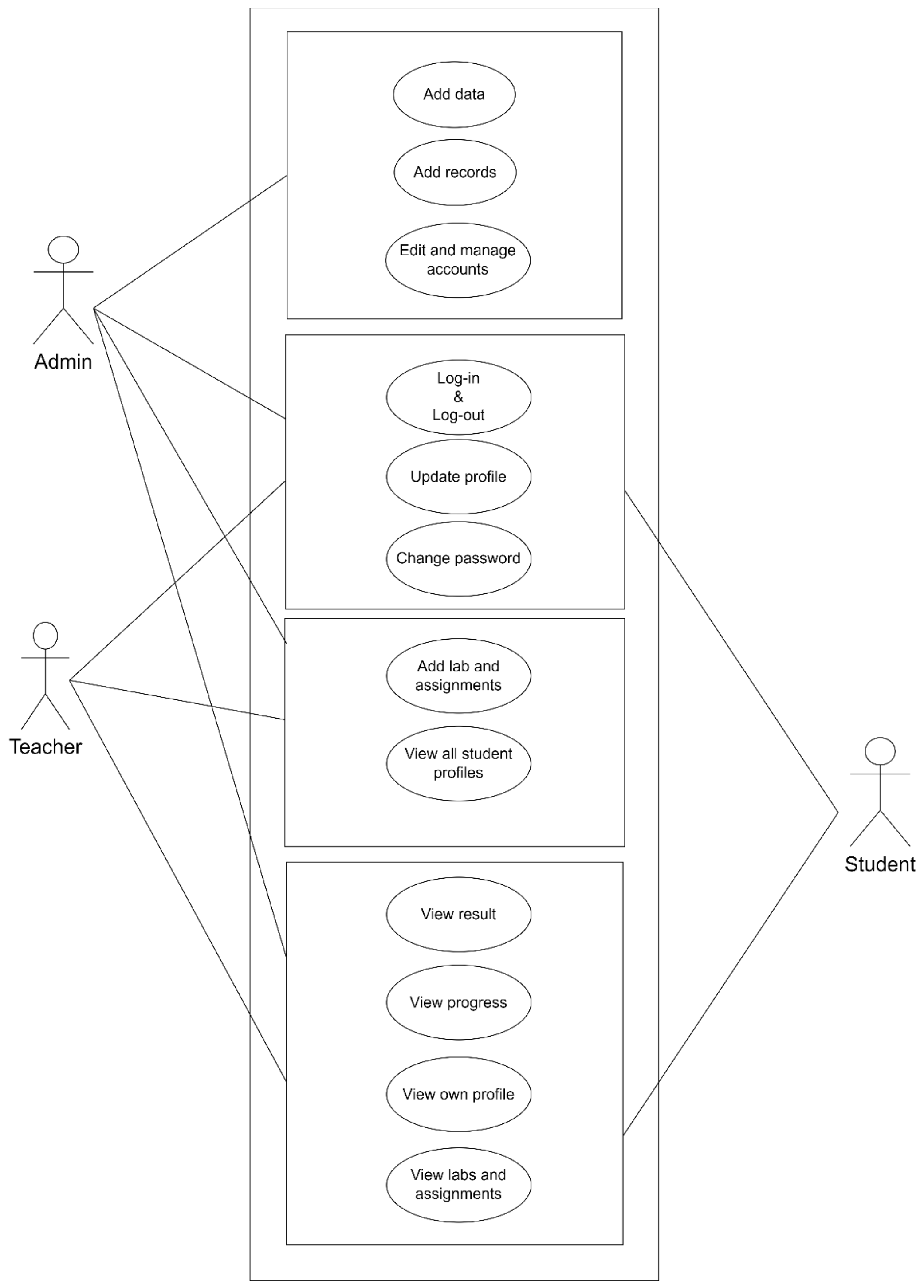


Fig : Use case diagram of Gamified Learning Progress Tracker

2. Class Diagram:

A class diagram is a type of UML (Unified Modeling Language) diagram that visually represents the classes in a system and the relationships between them. It graphical notation used to construct and visualize object oriented systems. It is designed for the system which is going to use oop methods.

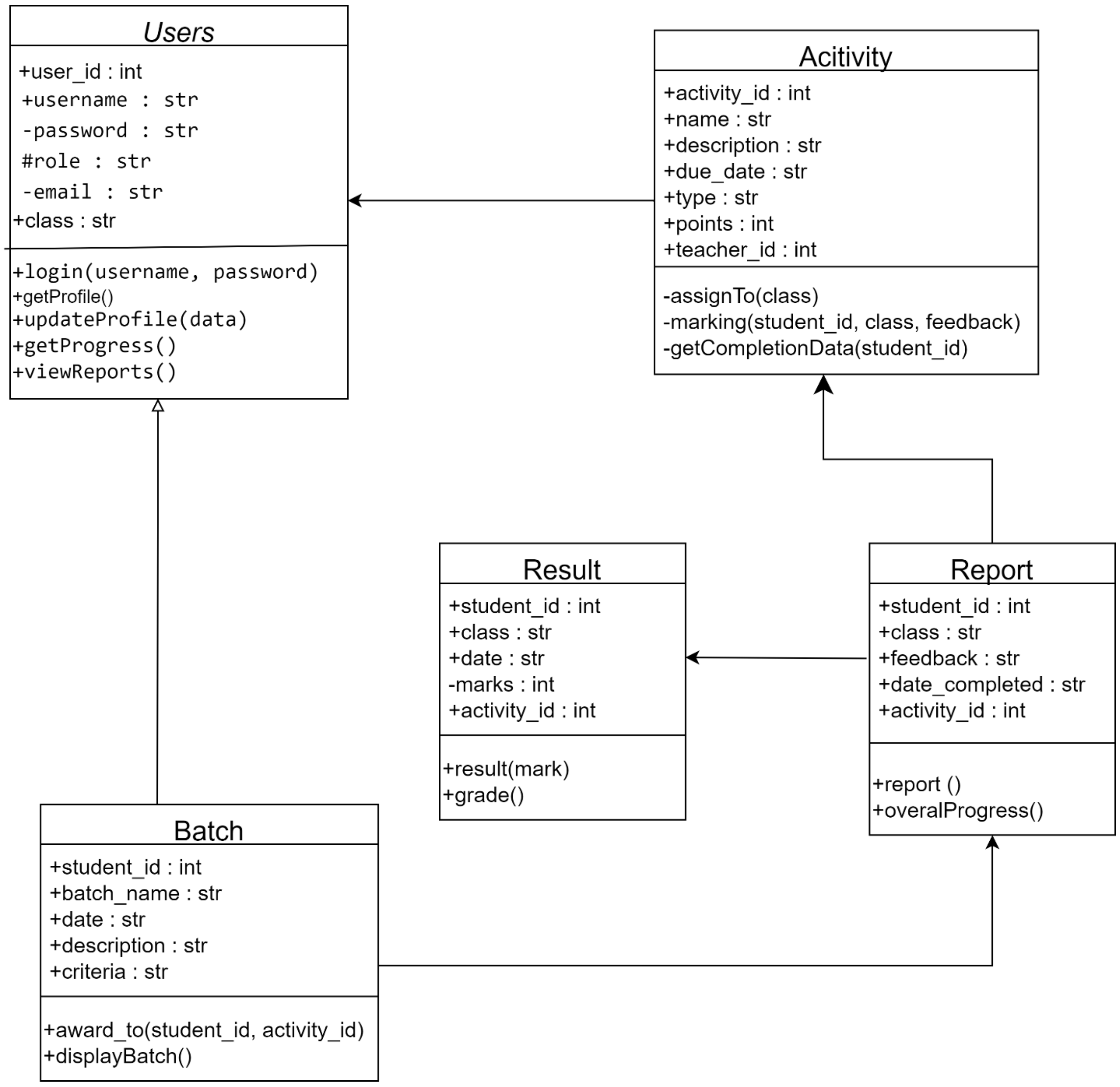


Fig : Class diagram of this system

Users:

- Attributes: id, username, password, role, name, email, class.
- Methods: login(username, password), getProfile(), updateProfile(data), getProgress(), viewReports().

Activity:

- Attributes: id, name, description, due_date, type, points, teacher_id.
- Methods: assignTo(class), marking (student_id, class, feedback), getCompletionData(student_id).

Report:

- Attributes: id, student_id, activity_id, feedback, date_completed.
- Methods: report(), overallprogress().

Result:

- Attributes: student_id, date, class, marks, activity_id .
- Methods: result(marks), grade().

Badge:

- Attributes: student_id, batch_name, date, description, criteria.
- Methods: award_to(student_id, activity_id), displayBatch().

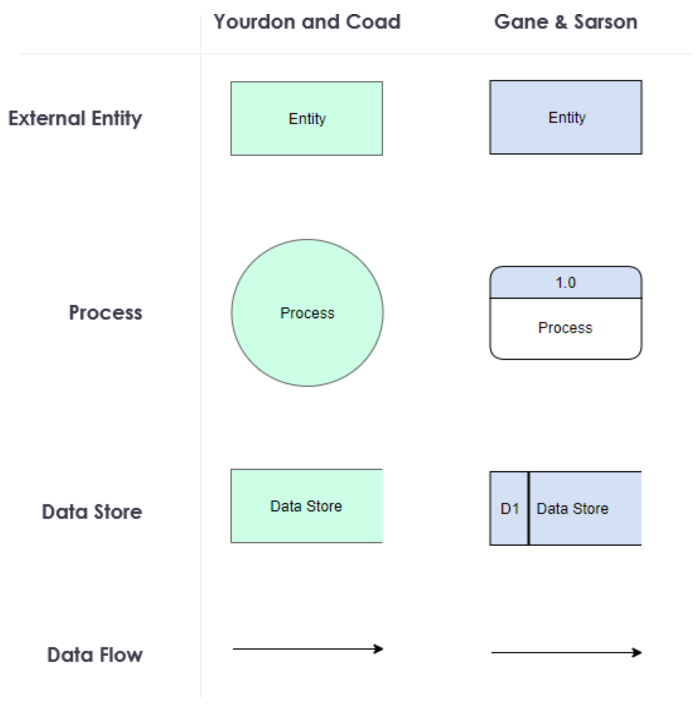
Why use class diagrams?

- **Visualization and communication:** Class diagrams provide a clear and concise way to visualize the overall structure of the system, making it easier to understand and communicate design ideas to others.
- **Identify potential issues:** By visualizing relationships and data flow, we can identify potential problems or inconsistencies early in the development process, saving time and effort later.
- **Document design:** Class diagrams serve as documentation for our system's structure, making it easier to understand and maintain in the future.

3. Data flow diagram (dfd):

A Data Flow Diagram (DFD) is like a map that shows how data moves through a system.

Components of dfd are :



- **Processes:** These are the actions or tasks that happen in the system. Each process takes in inputs, does something with them, and produces outputs.
- **Data Stores:** These are like storage areas for the data. They can be databases, files, or even just a place to keep information.
- **Data Flows:** Arrows show how data moves between processes, data stores, and external entities. It's like following the path of information as it travels through the system.
- **External Entities:** These are sources or destinations of data that are outside the system. They could be users, other systems, or anything that interacts with the system.

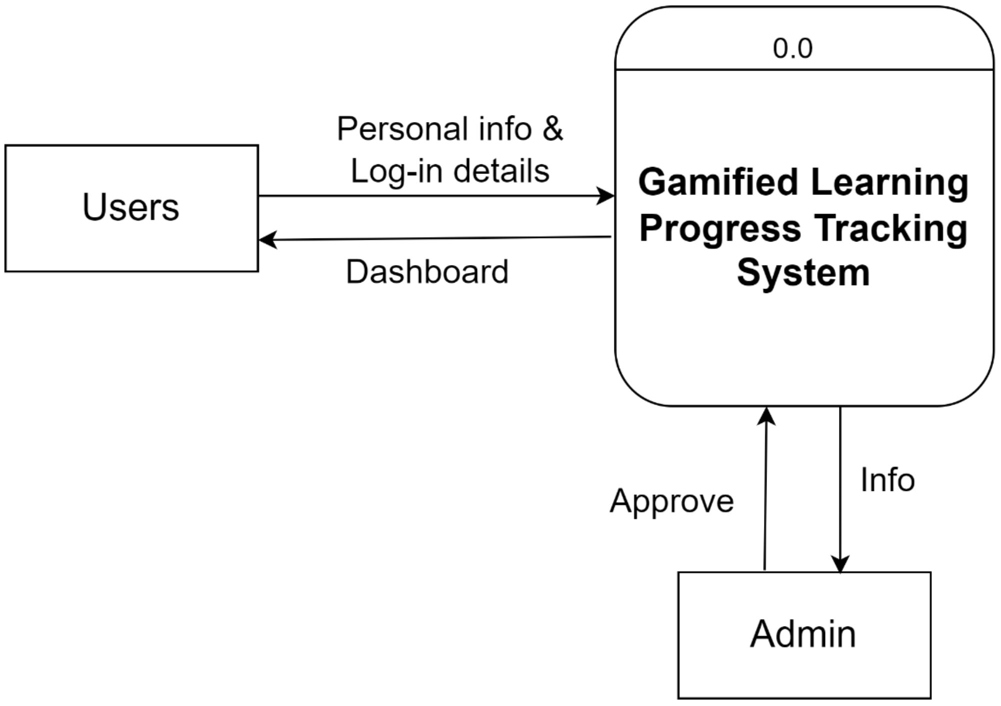

0.0
Gamified Learning Progress Tracking System
Users
Personal info & Log-in details
Dashboard
Approve
Info
Admin

Fig : Level 0 DFD (Context level diagram)

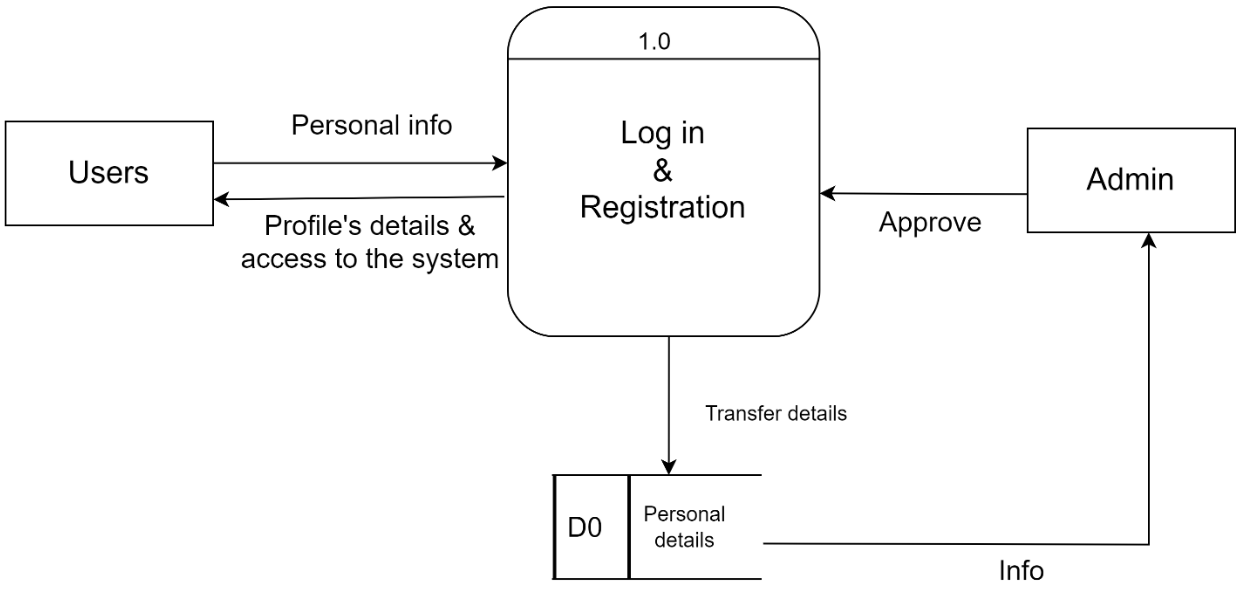


Fig : Level 1 dfd

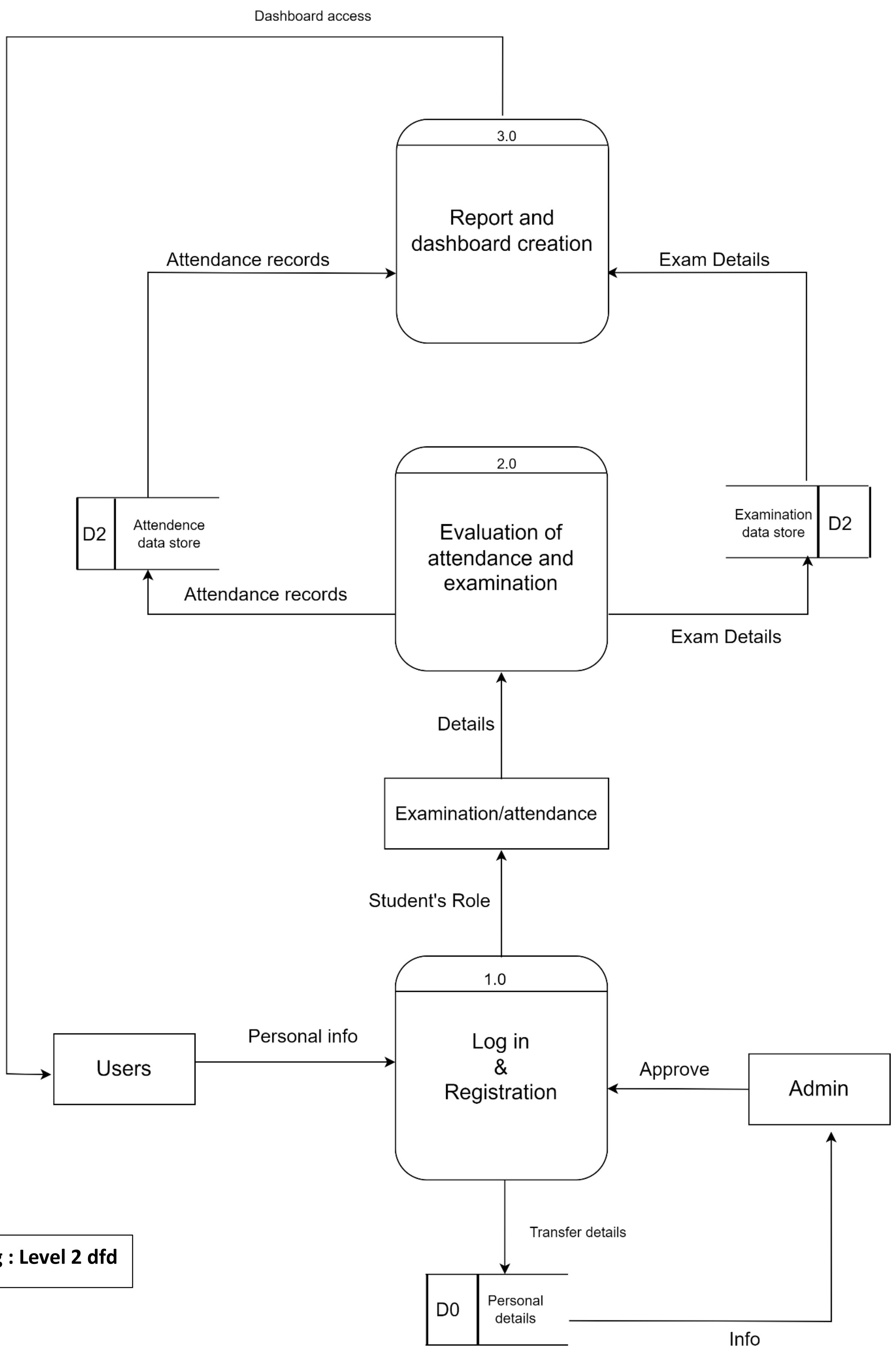


Fig : Level 2 dfd

Reasons to draw dfd :

- Clarity: They visually map data flow, making complex systems easy to understand for both technical and non-technical audiences.
- Analysis: By identifying data flows and transformations, DFDs help uncover inefficiencies and improve process flow.
- Communication: They act as a shared language, facilitating discussions and agreements between stakeholders about system design.
- Documentation: DFDs serve as valuable documentation, aiding system maintenance and understanding for future reference.

Conclusion:

We've looked at three diagrams: Use Case Diagrams, Class Diagrams, and Data Flow Diagrams (DFDs). Each shows a different part of how a system works.

- Use Case Diagram: It helps us see what users need and how they'll use the system.
- Class Diagram: This one shows the inside of the system, like its building blocks and how they connect.
- DFD: This diagram is all about how data moves around in the system, from where it starts to where it ends up.

These diagrams work best together. We can use them to see the whole picture of the system – what it does, how it's built, and how data flows which works well and makes users happy.