# *Final Presentation:*

*Topic: An Afterschool Software.*

*Duration: October 2023-January 2024*

*Worked by: Learda Dushi, Fiona Begvarfaj, Alesia Palloshi*

*Professor: I.Hakrama*

*Software Engineering Bachelor 2*

UNIVERSITETI®
METROPOLITAN
TIRANA

***Brief description***

"Princi Vogel After School" is a comprehensive and user-friendly mobile application designed to enhance the collaboration between teachers and parents in supporting their children's education. This app serves as a virtual bridge between the school and home environment, streamlining communication and involvement in students' academic endeavors.

Teachers can utilize the Princi Vogel After School app to seamlessly share information about upcoming assignments, class projects, and lesson plans. They can upload resources, such as study materials, project guidelines, and important announcements, ensuring that parents are well-informed about their children's academic activities.

Parents, on the other hand, benefit from real-time updates on their child's progress, upcoming tasks, and overall school performance. The app provides a platform for parents to actively engage in their child's education by allowing them to monitor completed assignments, view project submissions, and communicate with teachers regarding any concerns or questions.

Key features of Princi Vogel After School include a centralized calendar for tracking important dates, a secure messaging system for direct communication between teachers and parents, and a user-friendly interface for easy navigation. The app aims to foster a collaborative learning environment, promoting the holistic development of students with the active participation of both educators and parents.

Princi Vogel After School is designed to simplify the school-to-home connection, empowering parents to play a more active role in their child's educational journey and ensuring that teachers have an effective tool to facilitate transparent communication and support academic success.

User Characteristics:

Princi Vogel After School has 4 main types of users:

Administrators

Administrative Technicians

Facilities Manager

Parents

Administrators:

Role: Oversee and manage the overall functioning of the school.

Characteristics:

Require access to comprehensive data and analytics for school-wide performance.

Need the ability to monitor and manage user accounts for both teachers and students.

Seek features for communication and collaboration among teachers, students, and parents.

Prioritize security and data privacy to ensure compliance with educational standards.

Administrative Technicians:

Role: Support the technical infrastructure and IT needs of the school.

Characteristics:

Demand reliable and scalable technical support to ensure smooth operation of the app.

Require integration capabilities with existing school management systems and databases.

Look for features that simplify software updates, troubleshooting, and maintenance.

Need access to robust data backup and recovery options.

Facilities Manager:

Role: Responsible for the physical facilities and resources within the school.

Characteristics:

Seek features that aid in managing and scheduling facility usage for after-school activities.

Need a platform for reporting and tracking maintenance issues within the school premises.

Look for communication tools to coordinate with teachers, administrators, and parents regarding facility-related matters.

Prioritize ease of use and accessibility to ensure all staff can efficiently utilize the app.

Parents:

Role: Actively involved in their child's education and well-being.

Characteristics:

Require a user-friendly interface for easy navigation and understanding of their child's academic progress.

Seek real-time notifications for upcoming assignments, events, and general school updates.

Look for features that facilitate direct communication with teachers and administrators.

Value the ability to access their child's completed projects, assignments, and overall academic performance.

Product Requirements.

Princi Vogel After School will only be available for Android and IOS phone systems.

The system will be easy to use so the user won't have any problems during the whole time he/she is still a student or an employee.

Accessibility

Princi Vogel After School can be used everywhere as long as there is internet connection and the device has sufficient power supply.

Responsiveness

The application is thought to be extremely responsive and fast too..

Flexibility

The application is thought to adapt the updates very easy but also deal with errors/problems as quick as possible

Effectiveness

This application will be easy to use and understand and as user-friendly as possible.

In case the user deals with an error, the messages will be easy to understand and also associated with a step-by-step procedure in order to get rid of it.Efficiency:

The application is going to be very efficient and 0 time-consuming, meaning that each user will fulfill his tasks super-fast and in the best case with no errors/warnings at all.

The interface will also be very easy and user-friendly with no complicated buttons or actions.

Performance Requirements

Since Princi Vogel After School is going to be installed in a computer/laptop/mobile, the data will be delivered in real-time.

The bigger the space, the more will be the users therefore the better the performance.

Space Requirements

The app will be able to handle at least 50 users at a time, and the database system should handle at least 40-50 users at any given time.

Maximum user load: 50.

Per-user memory requirement: 2-4 GB RAM.

Availability

Internet access is necessary.

The application will be available 24/7.

Maintanence

The app will be updated, when necessary, in order to process all the operations in real time.

In case the system will crash, the app should not display anything to the user except an error message “Something interesting is happening! Please come back to check what has happened!”

If the crash happens the system should start as soon as possible.

Integrity

The personal information must be available only for the actual user.

All users should provide personal credentials to log in into the system and should be authenticated before accessing their own profiles.

Security

When a new user is logged in, a randomly generated password must be sent to it by the admin email.

Software Designs

Successful log in

The user will manually fill the email input box.

The user will manually fill the password input box.

The app will check inputs with database records, if they match the user is successfully logged in.

In the users screen will be displayed the home page of his/her own page.

Failing to log in

The user will manually fill the email input box.

The user will manually fill the password input box.

The app checks the input with the database records, inputs doesn't match.

A message will be displayed to user, asking to enter the wrong input again.

User remains at the log in page.

Facilities Managers Scenarios

Checking Attandace

Facilities Manager is firstly logged in.

In front of him will be shown a button "check daily report" by default.

When clicked it will open a form with necessary fields to fill (Date, name, surname, class, etc).

Check deadline of projects

The facilities manager is firstly logged in.

In the left column of sections, he will click on “Projects” section.

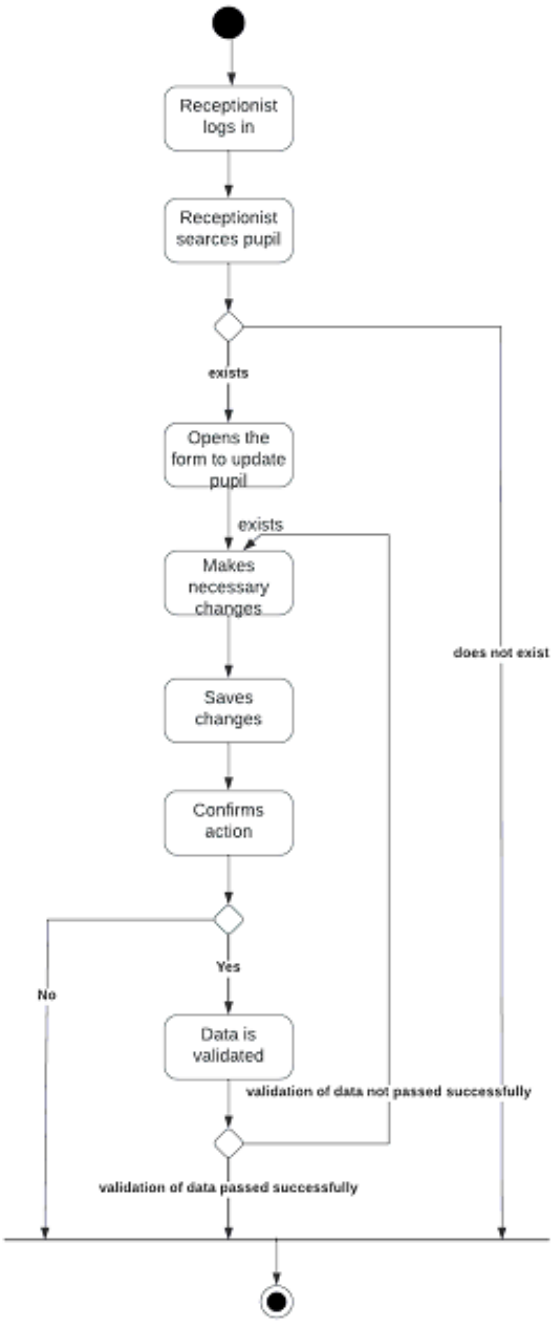
After clicking a table with the name of project, its code, deadline and details columns will be shown on the screen.

At the search bar at the top of the page checks for project by entering its name or code and then check the deadline.

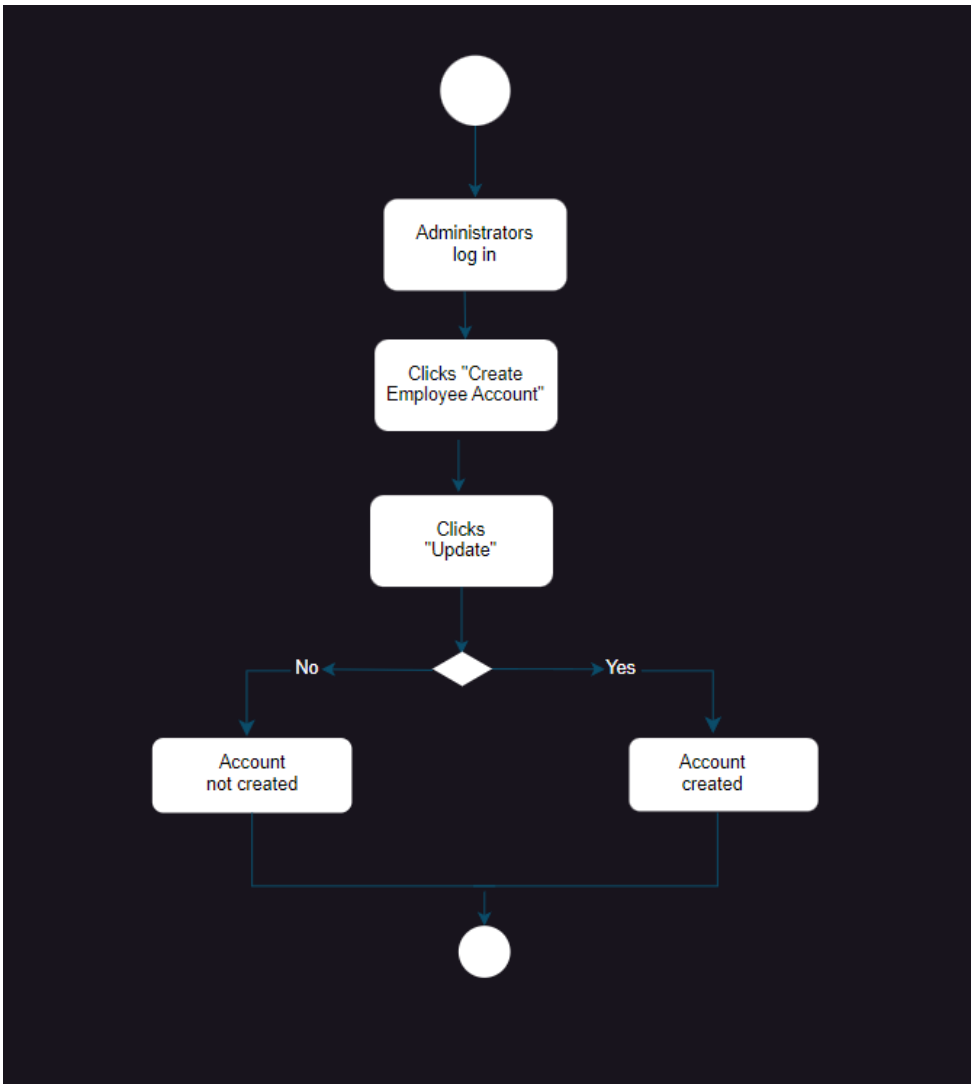
## ***ACTIVITY DIAGRAMS***

Activity diagrams in UML provide a concise and visual representation of dynamic aspects within a system, emphasizing the sequencing of activities, actions, and decisions. This

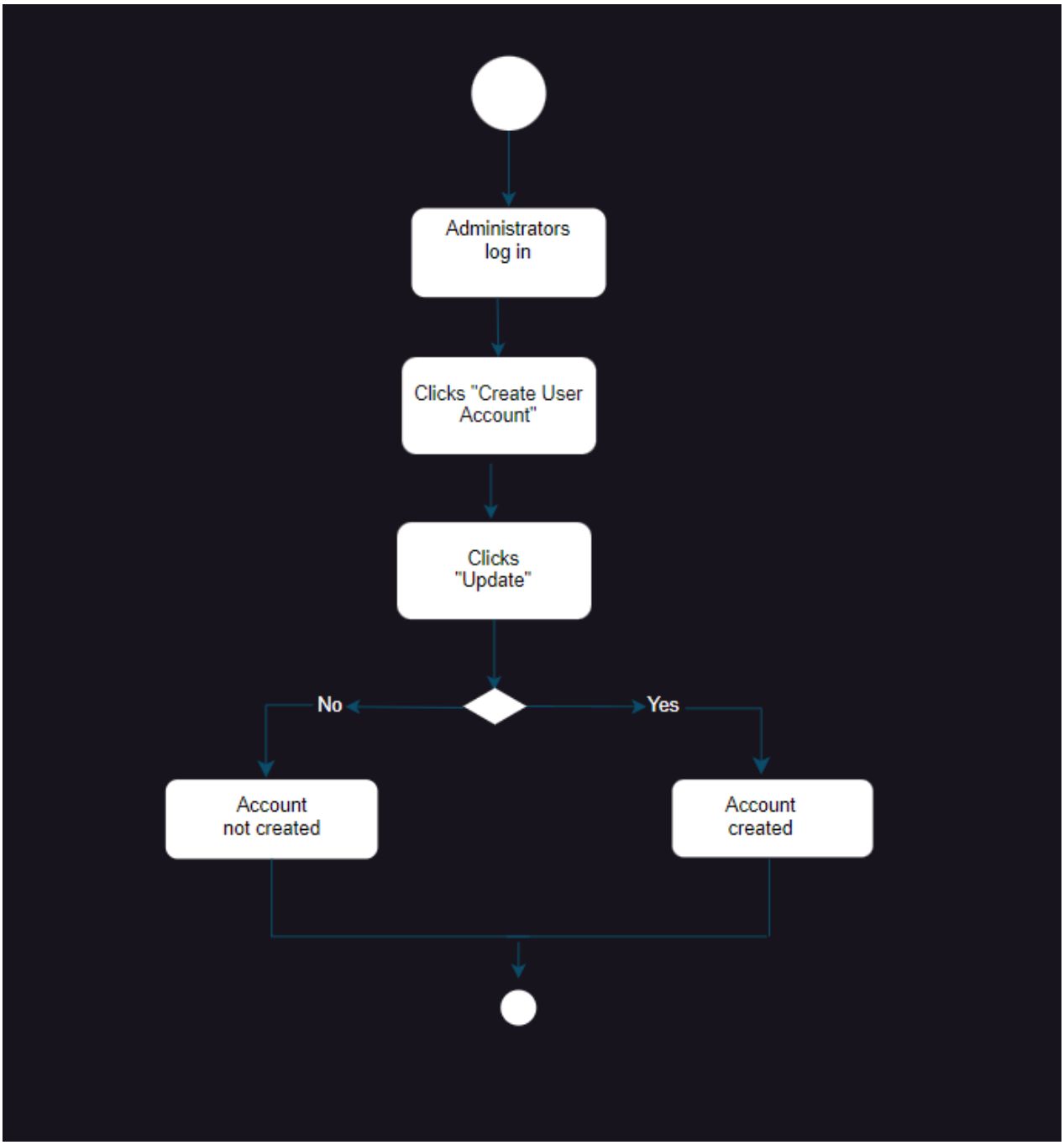
diagram serves as a valuable tool for optimizing account management processes, ensuring security, and streamlining administrative tasks associated with user accounts.



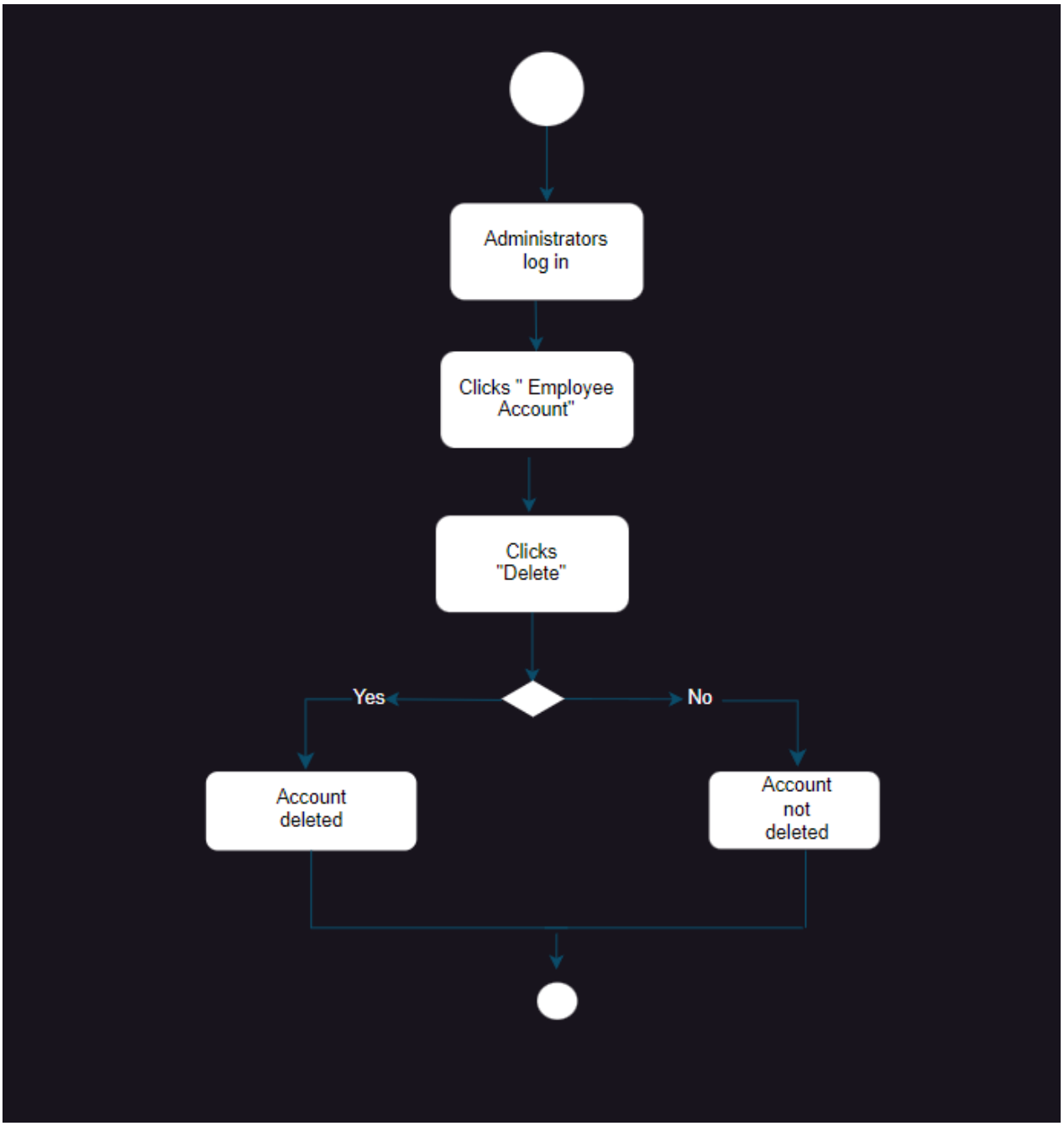
*-CREATE EMPLOYEE ACCOUNT*



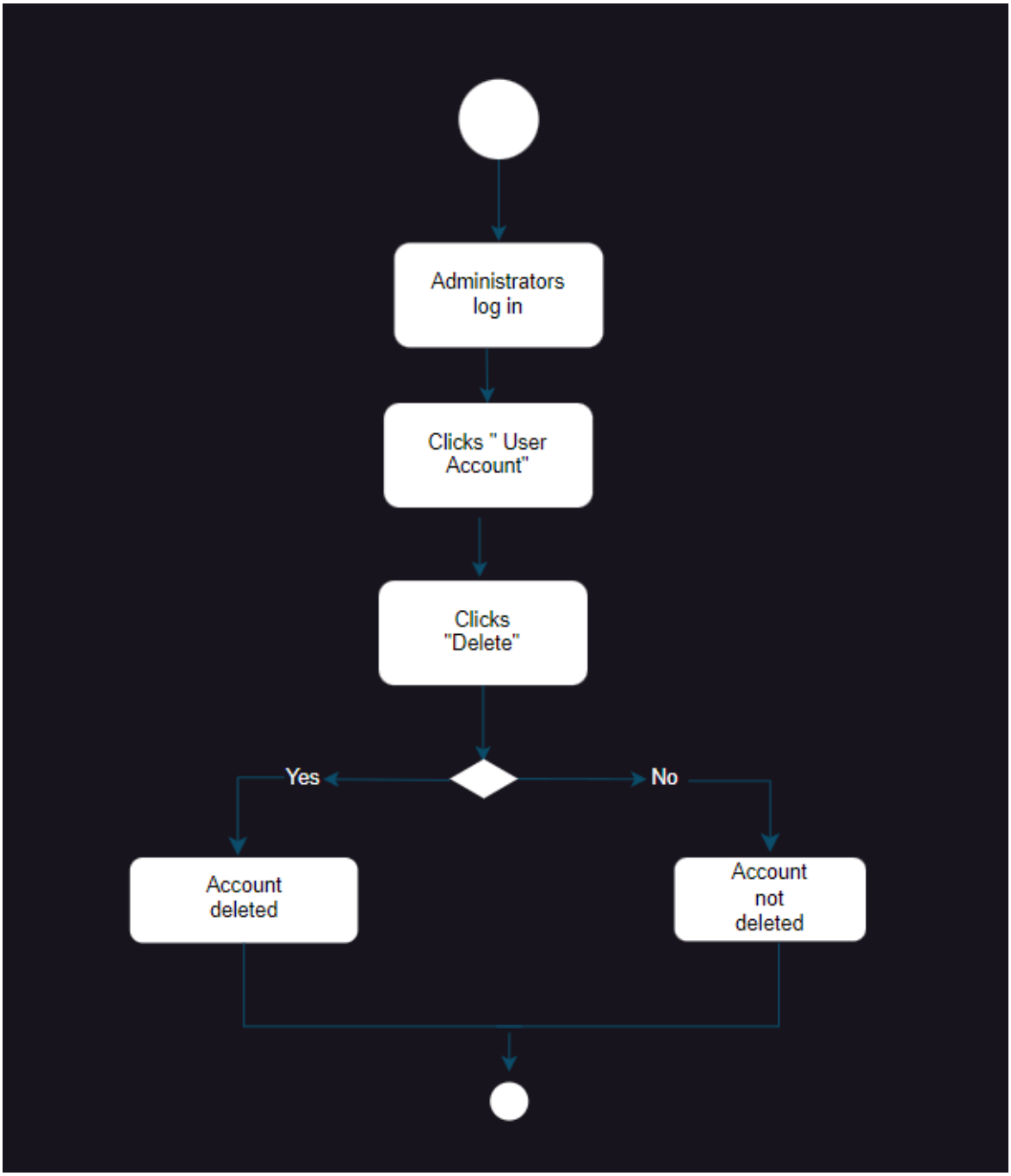

*-CREATE USER ACCOUNT*

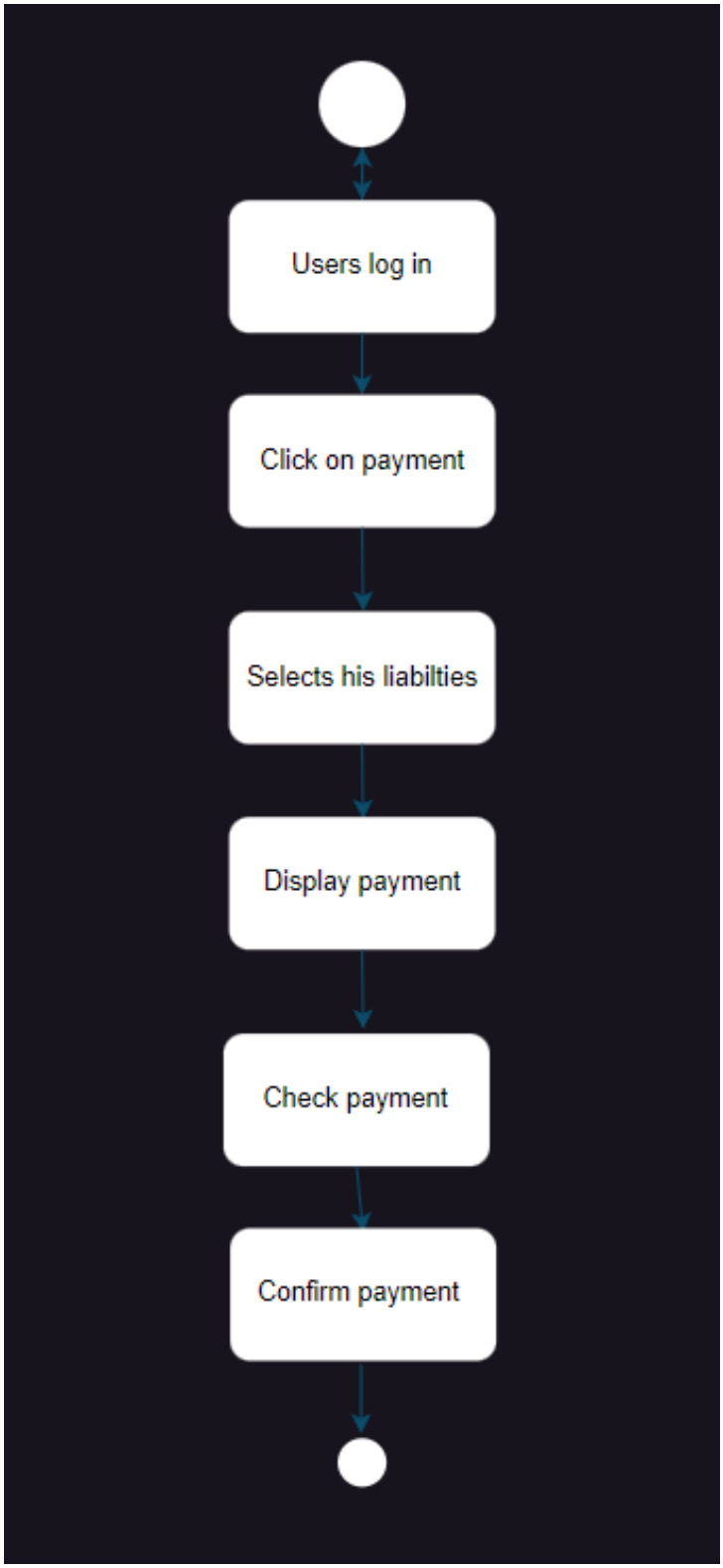



*-DELETE EMPLOYEE ACCOUNT*



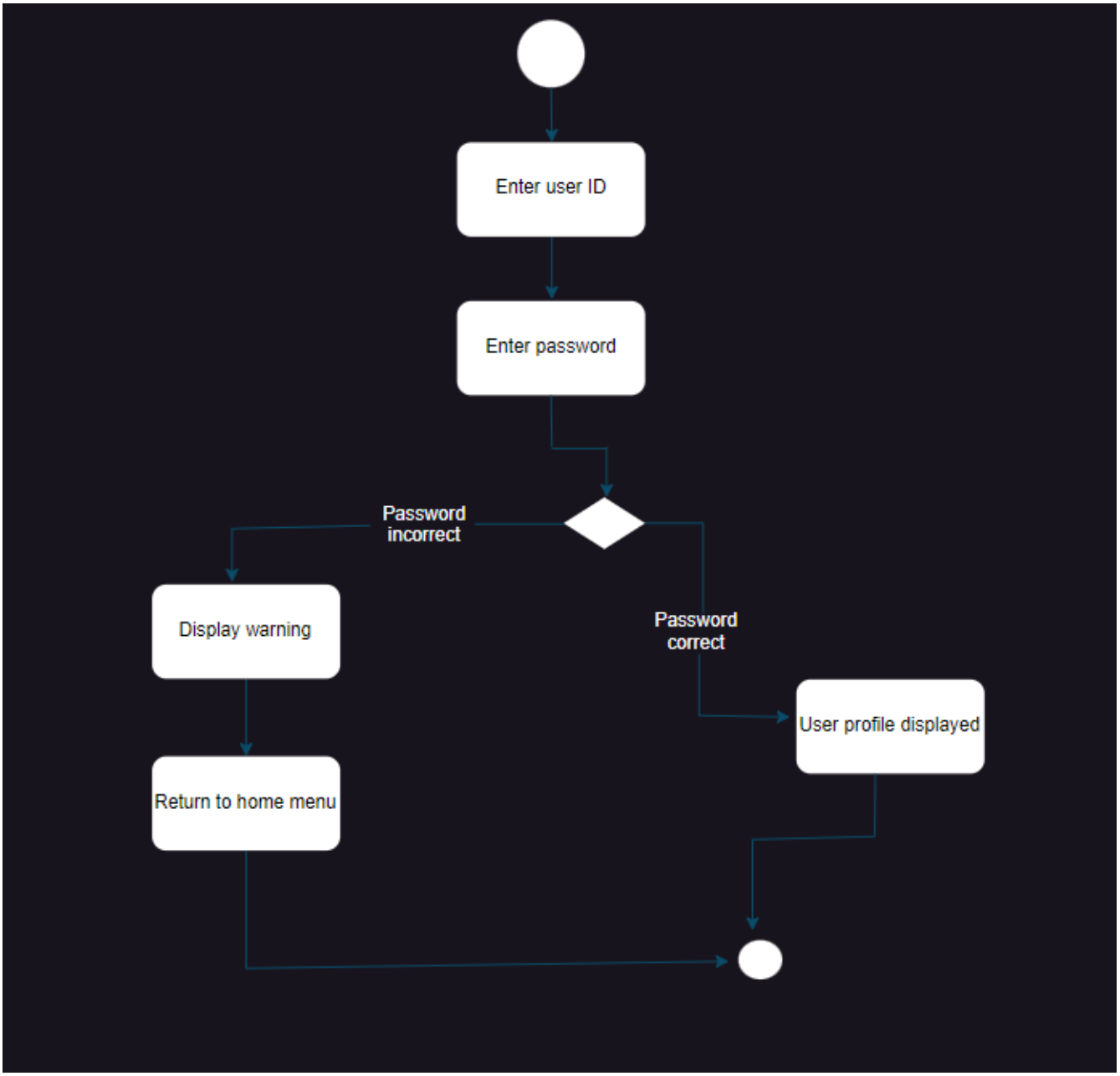

*-DELETE USER ACCOUNT*


Administrators log in
Clicks " User Account"
Clicks "Delete"
Yes
No
Account deleted
Account not deleted

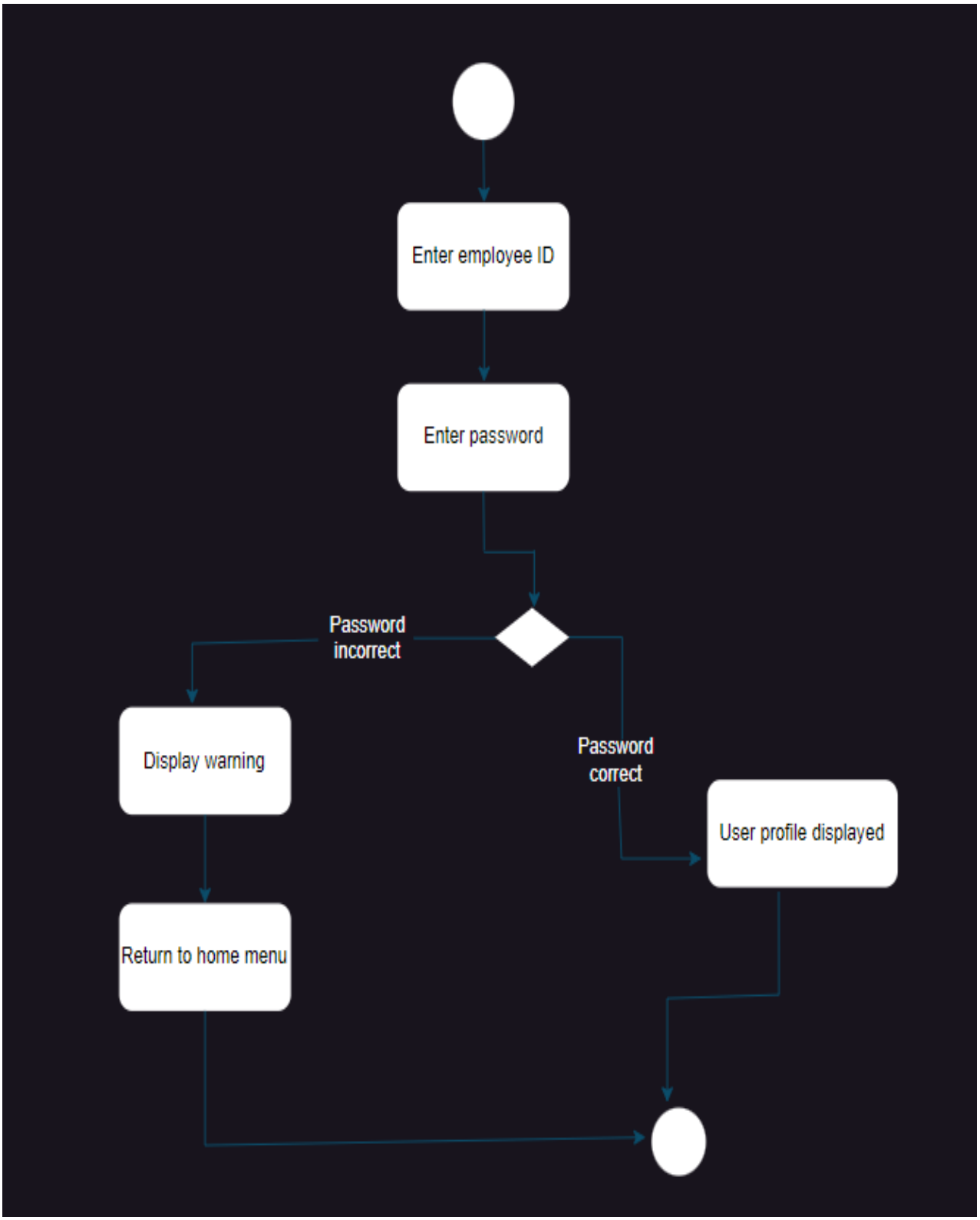



*-FINANCES*

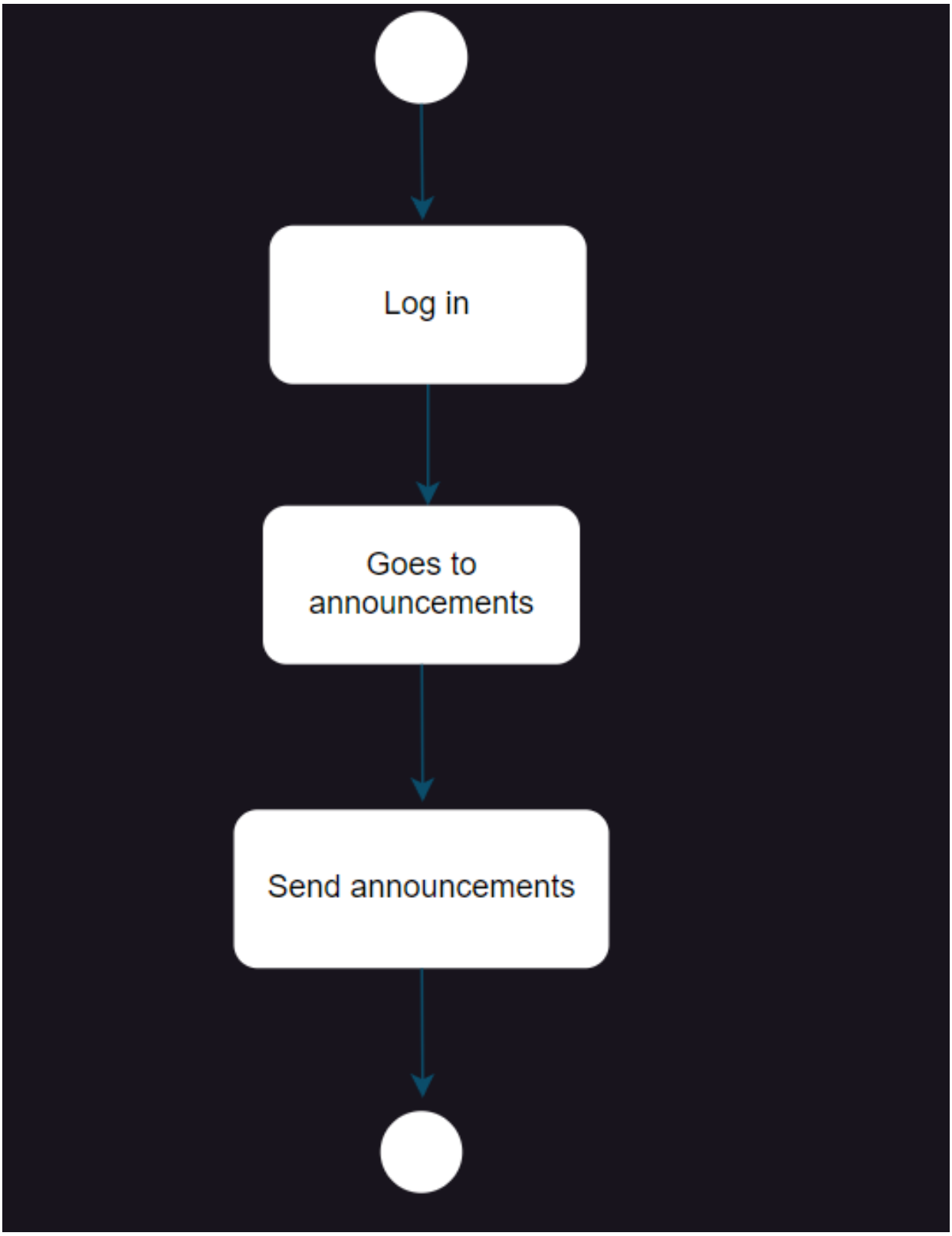
*-USER LOG IN*


Enter user ID
Enter password
Password incorrect
Display warning
Return to home menu
Password correct
User profile displayed

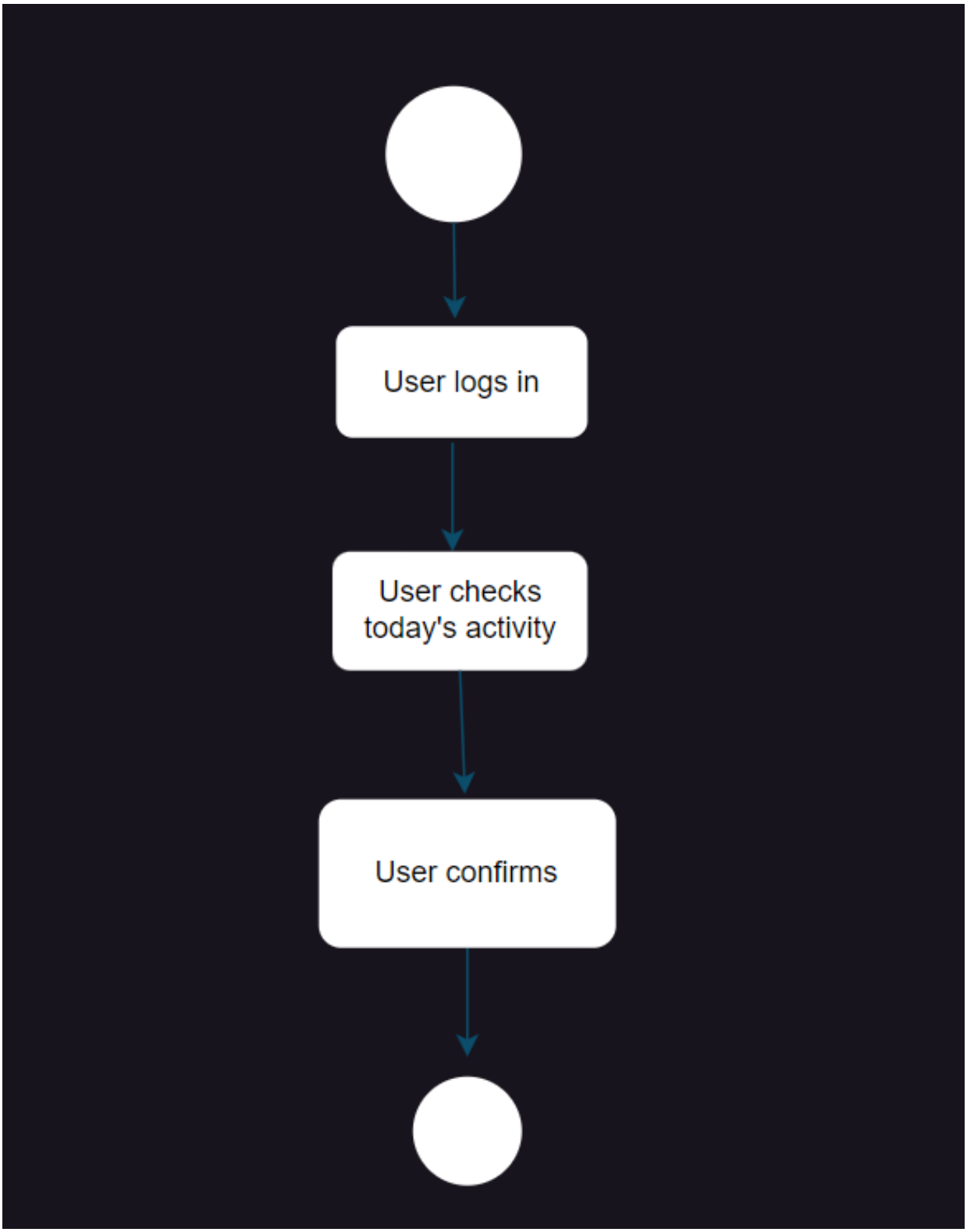
*-EMPLOYEE LOG IN*



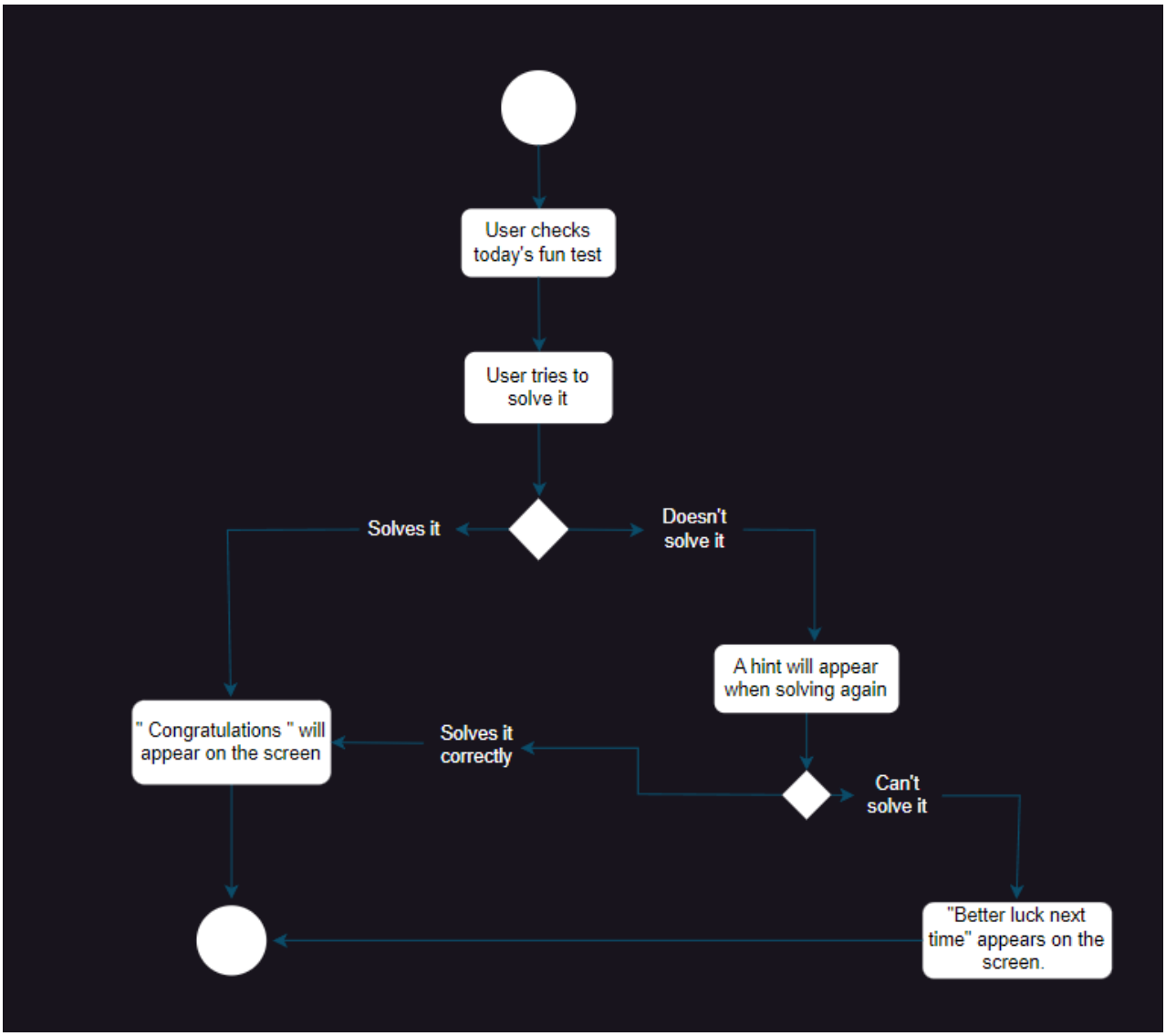

*-ANNOUNCEMENTS*



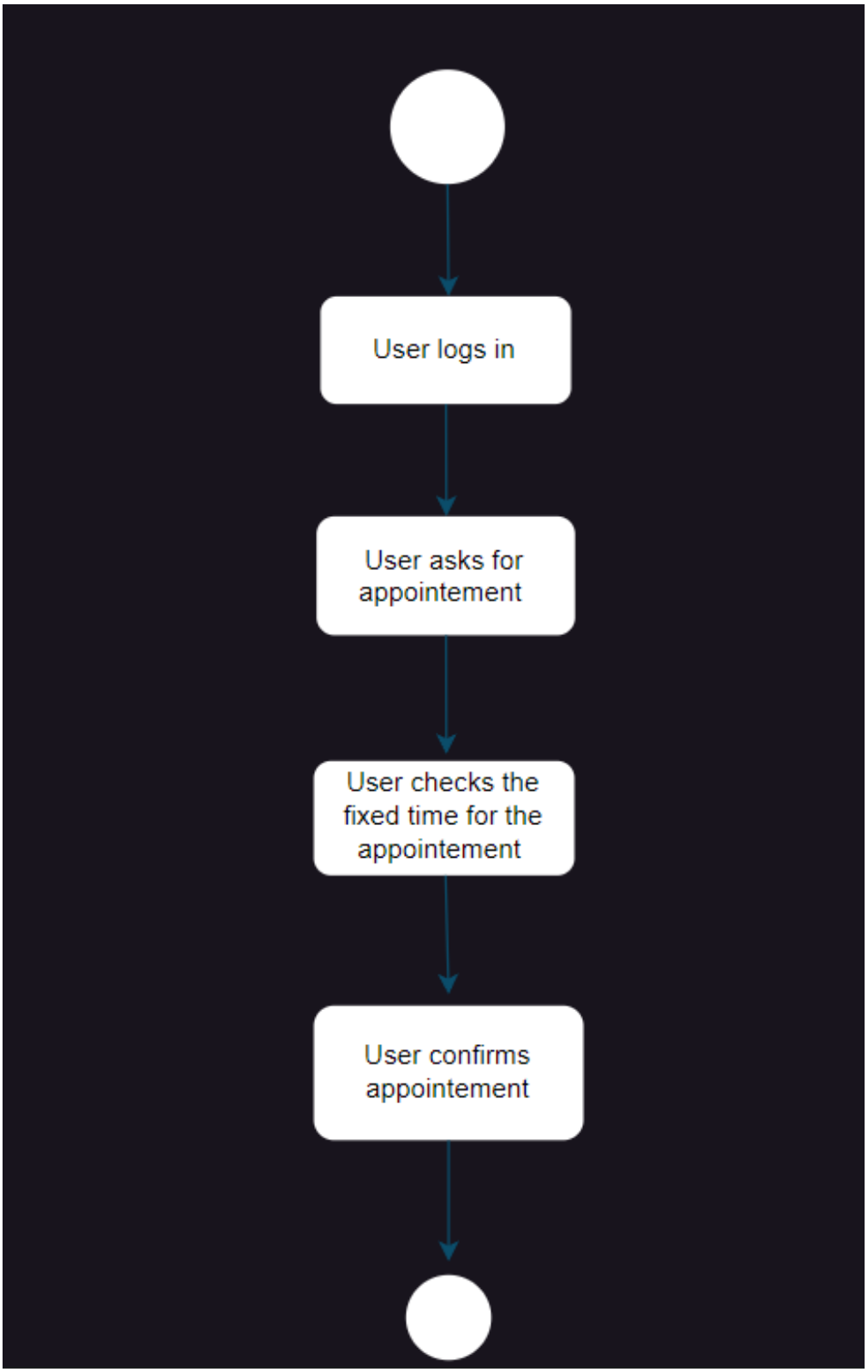

*-ACTIVITY*



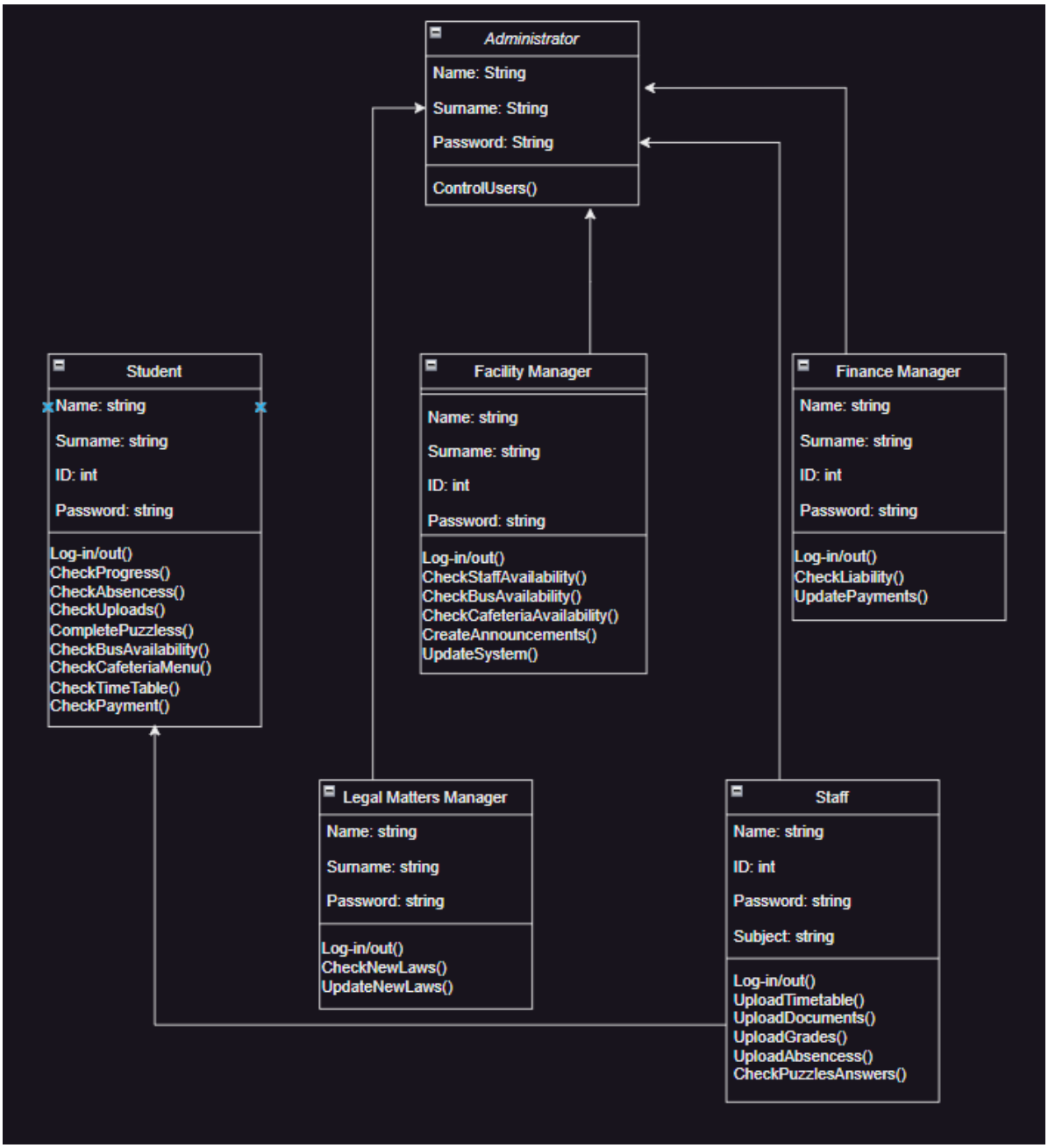

*-QUIZZ TIME*



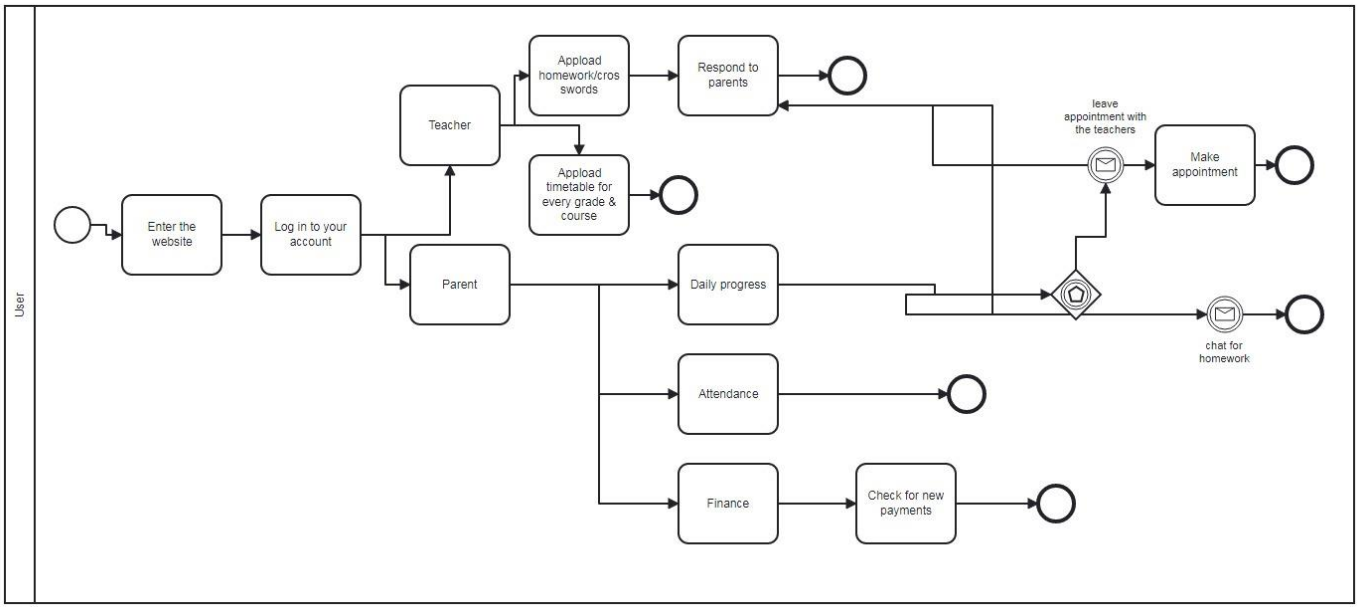

*-APPOINTEMENTS*


User logs in
User asks for appointement
User checks the fixed time for the appointement
User confirms appointement

***Class diagram:*** A class diagram is a type of diagram in the Unified Modeling Language (UML) that represents the structure and relationships of a system's classes and their interactions. It is a visual representation of the static view of a system, focusing on the classes, their attributes, methods, and the associations between them.

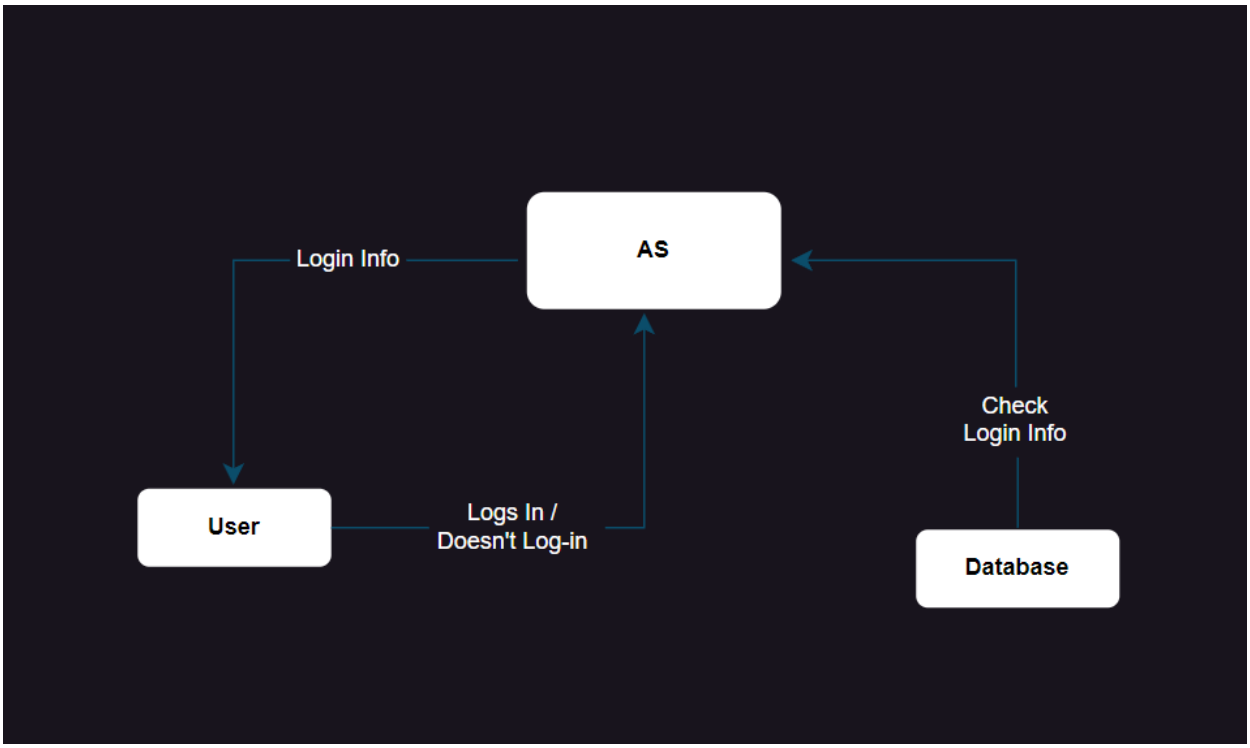


***BPMN:***

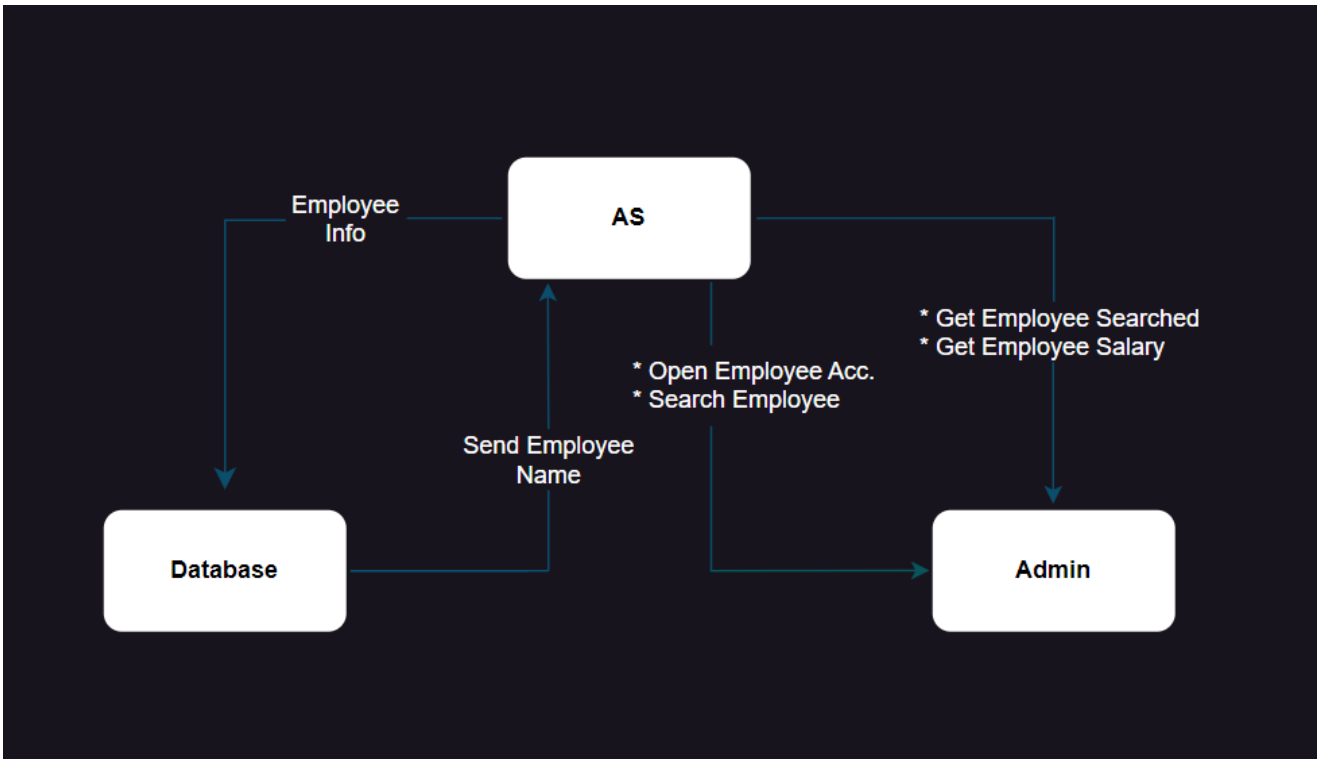

User
Enter the website
Log in to your account
Teacher
Parent
Apload homework/cros swords
Apload timetable for every grade & course
Respond to parents
Daily progress
Attendance
Finance
Check for new payments
leave appointment with the teachers
Make appointment
chat for homework

***Communication diagrams:*** Communication diagrams are particularly useful for visualizing the flow of messages and interactions between objects during the execution of a specific use case or scenario. They provide a dynamic perspective on how objects collaborate to achieve a particular goal or functionality within a system. These diagrams are valuable during the design and communication phases of software development, allowing stakeholders to understand the sequence of interactions and the runtime behavior of a system.

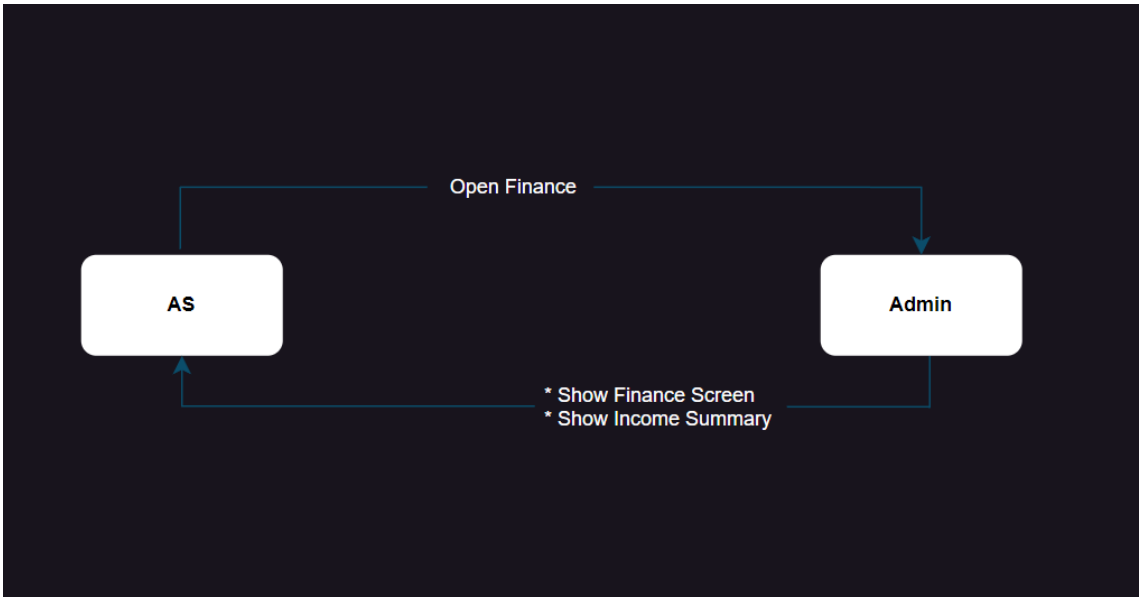
*-LOG IN*



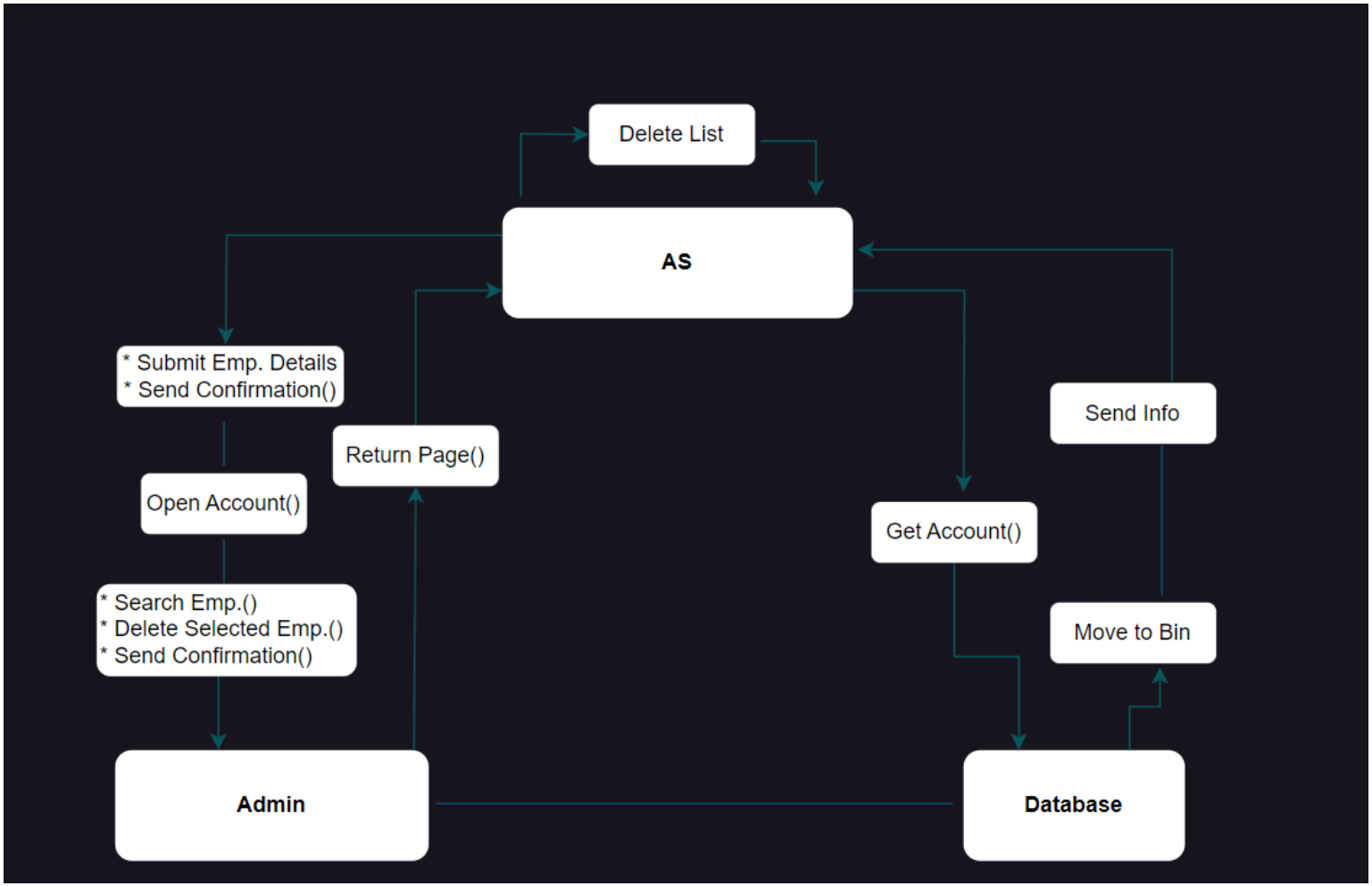
*-EMPLOYEE INFORMATION*



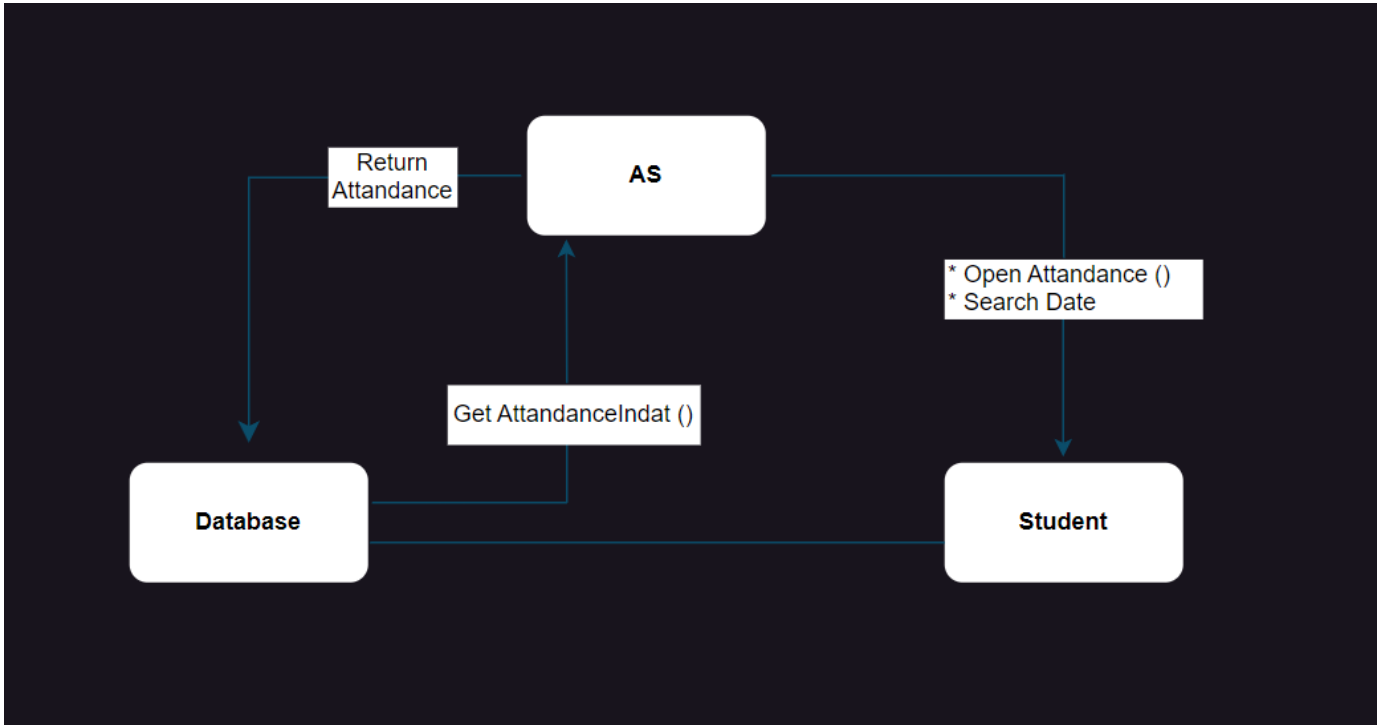

*-FINANCE*



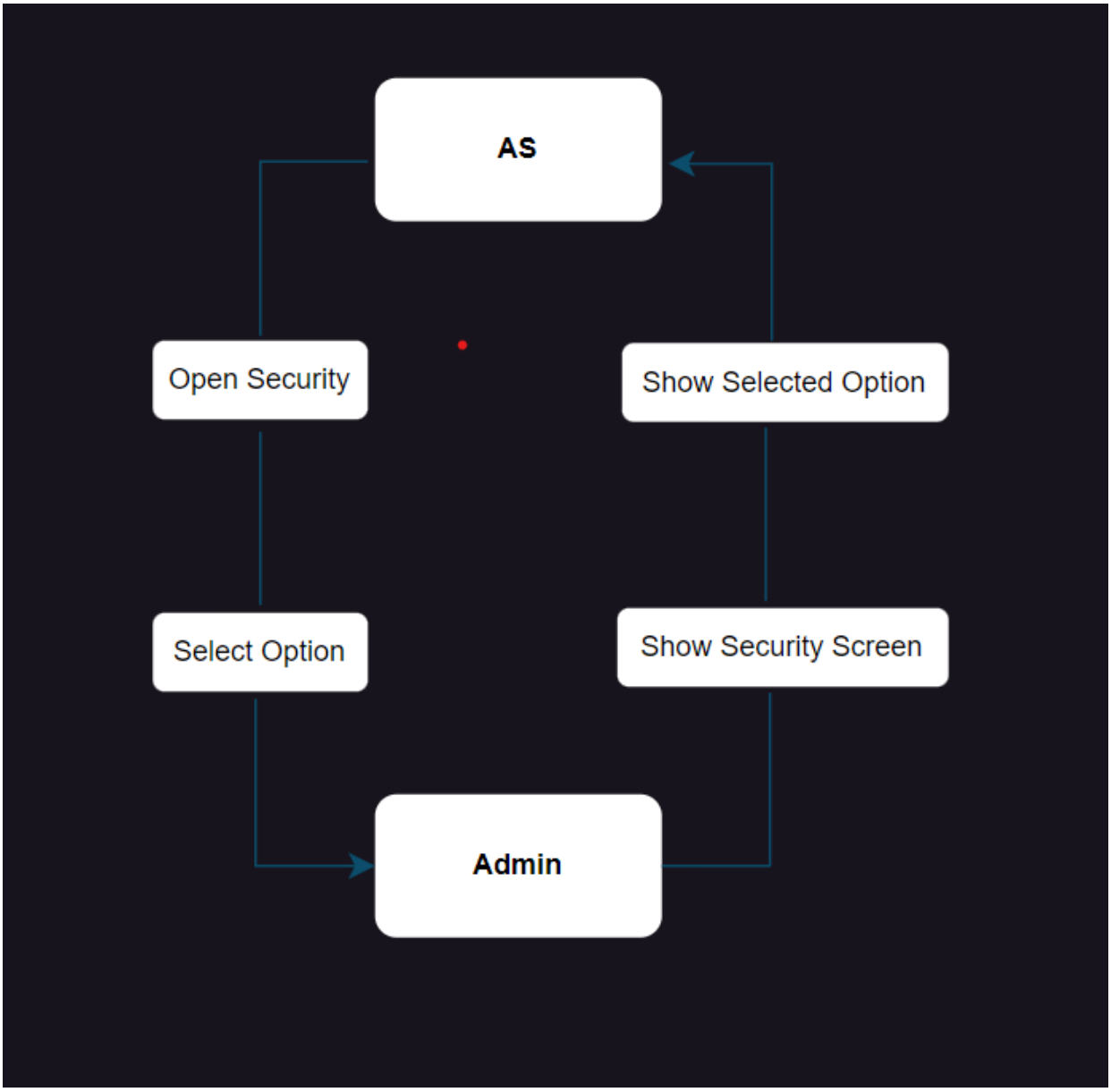

*-CREATE|DELETE EMPLOYEE ACCOUNT*


Delete List
AS
* Submit Emp. Details
* Send Confirmation()
Return Page()
Send Info
Open Account()
Get Account()
* Search Emp.()
* Delete Selected Emp.()
* Send Confirmation()
Move to Bin
Admin
Database

*-CHECK ATTENDANCE*

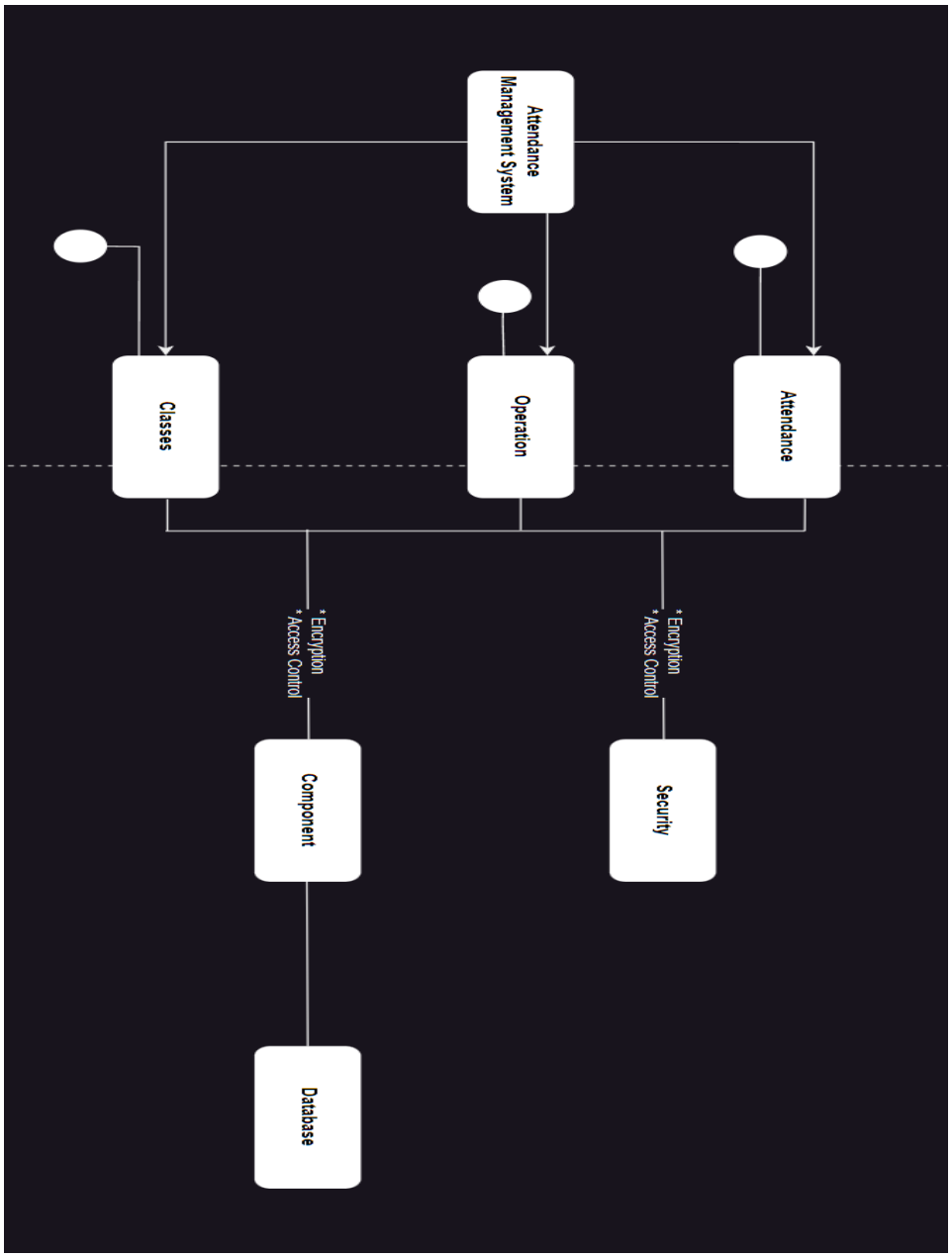

Return
Attandance
AS
* Open Attandance ()
* Search Date
Get AttandanceIndat ()
Database
Student

*-SECURITY*


AS
Open Security
Show Selected Option
Select Option
Show Security Screen
Admin

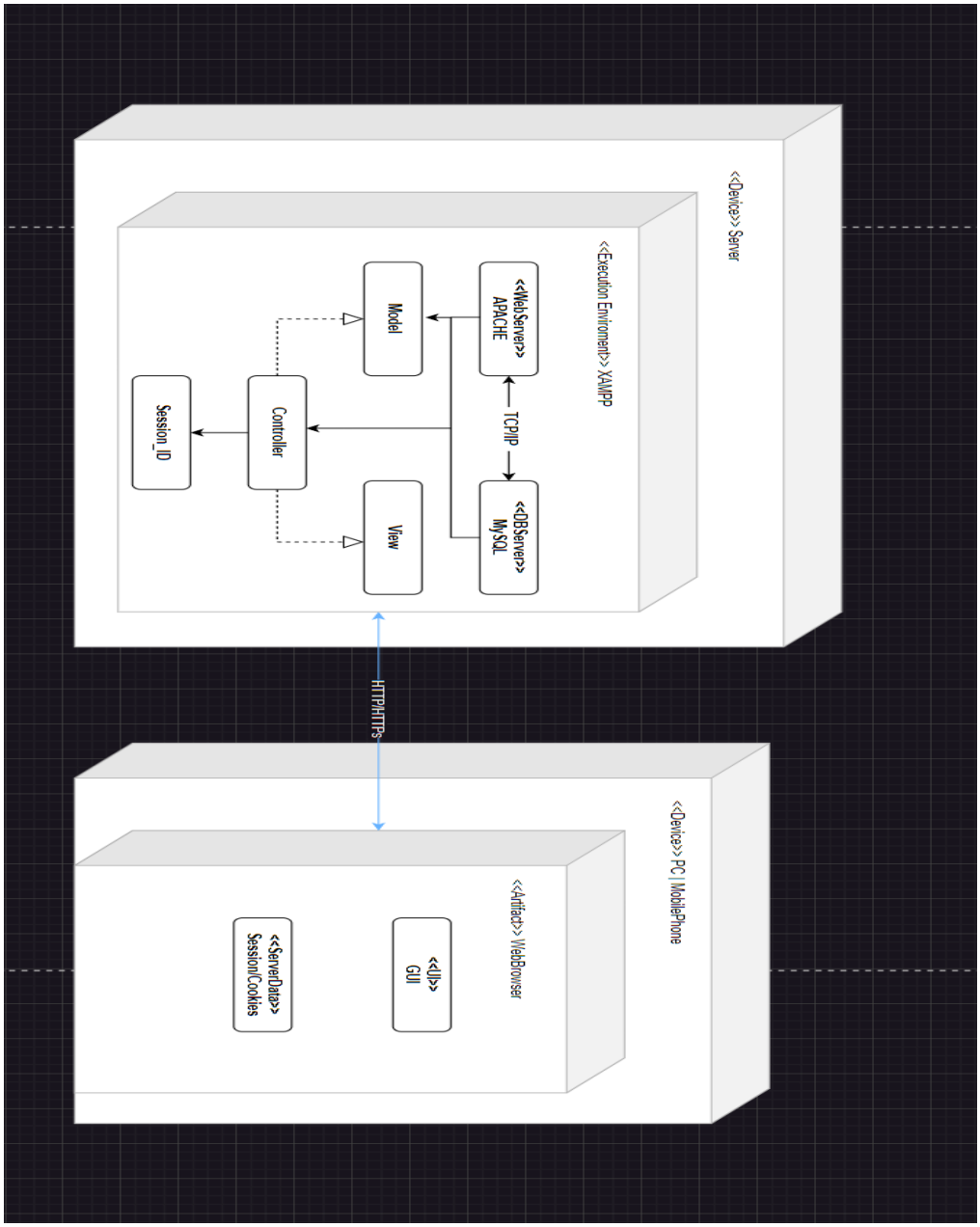
*-COMPONENT DIAGRAM:*

A component diagram is a type of diagram in the Unified Modeling Language (UML) that illustrates the high-level structure of a system and the dependencies between its components. Components in this context represent modular, replaceable, and deployable parts of a software system.

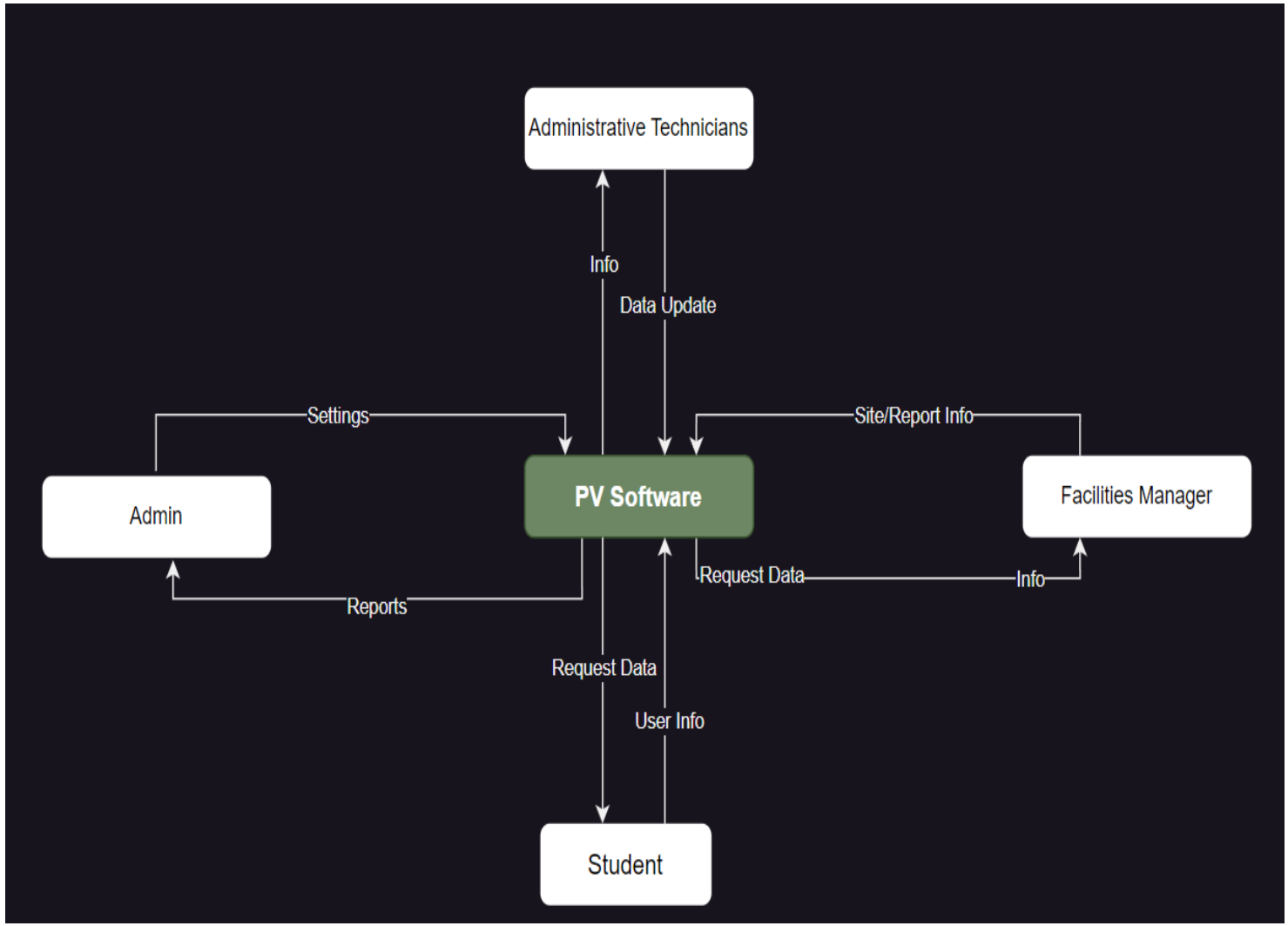



*-DEPLOYEMENT DIAGRAMS:* A deployment diagram is a type of diagram in the Unified Modeling Language (UML) that depicts the physical arrangement of hardware components and software artifacts in a system. It provides a visual representation of how a software

system is deployed across different nodes (hardware devices) and how those nodes are interconnected.

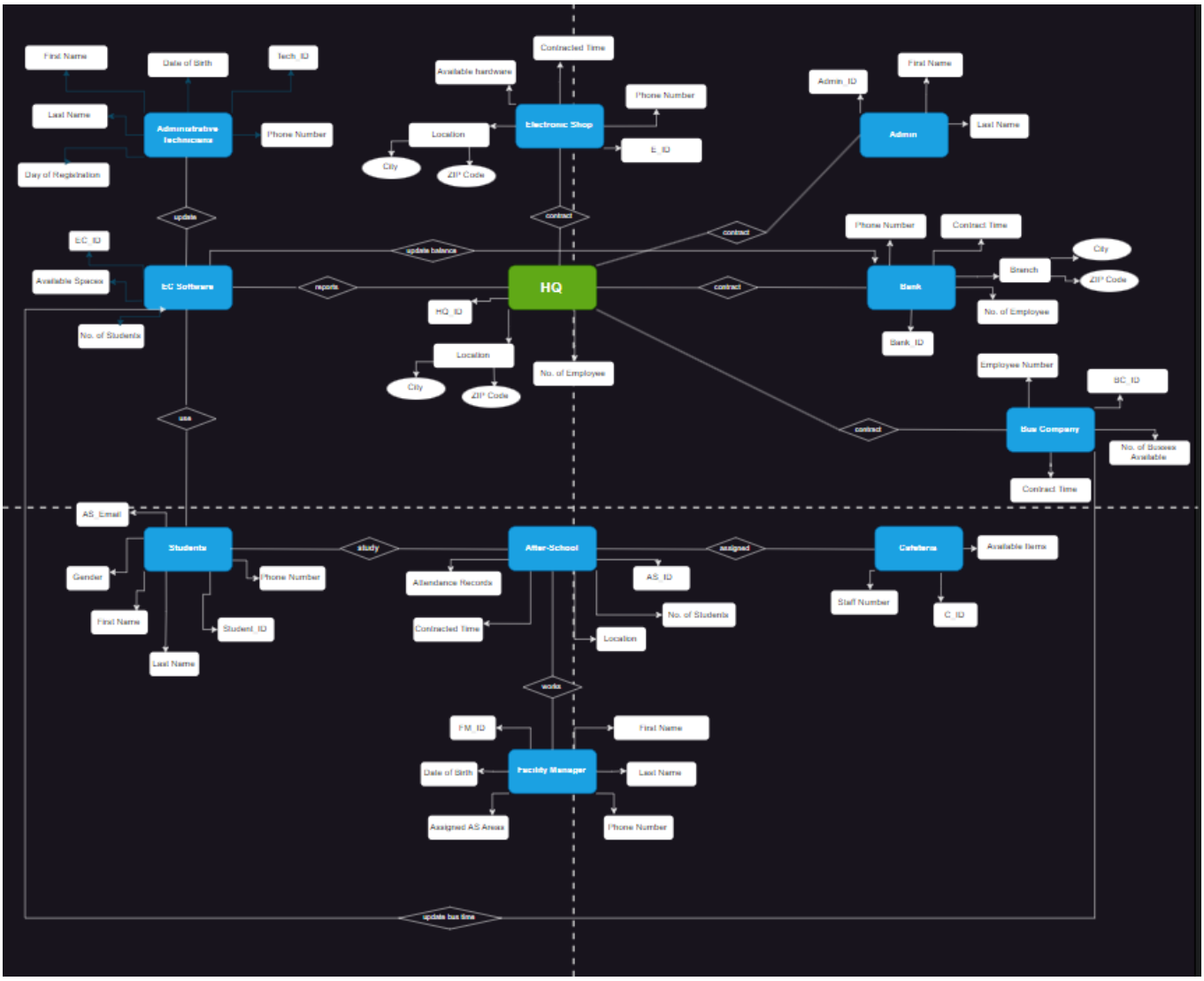


DFD: A Data Flow Diagram (DFD) is a visual representation that illustrates the flow of data within a system and the processes that manipulate or transform the data. DFDs are commonly used in system analysis and design to model and understand how data moves through different stages in a system.


Administrative Technicians
Info
Data Update
Settings
Site/Report Info
Admin
PV Software
Facilities Manager
Request Data
Info
Reports
Request Data
User Info
Student

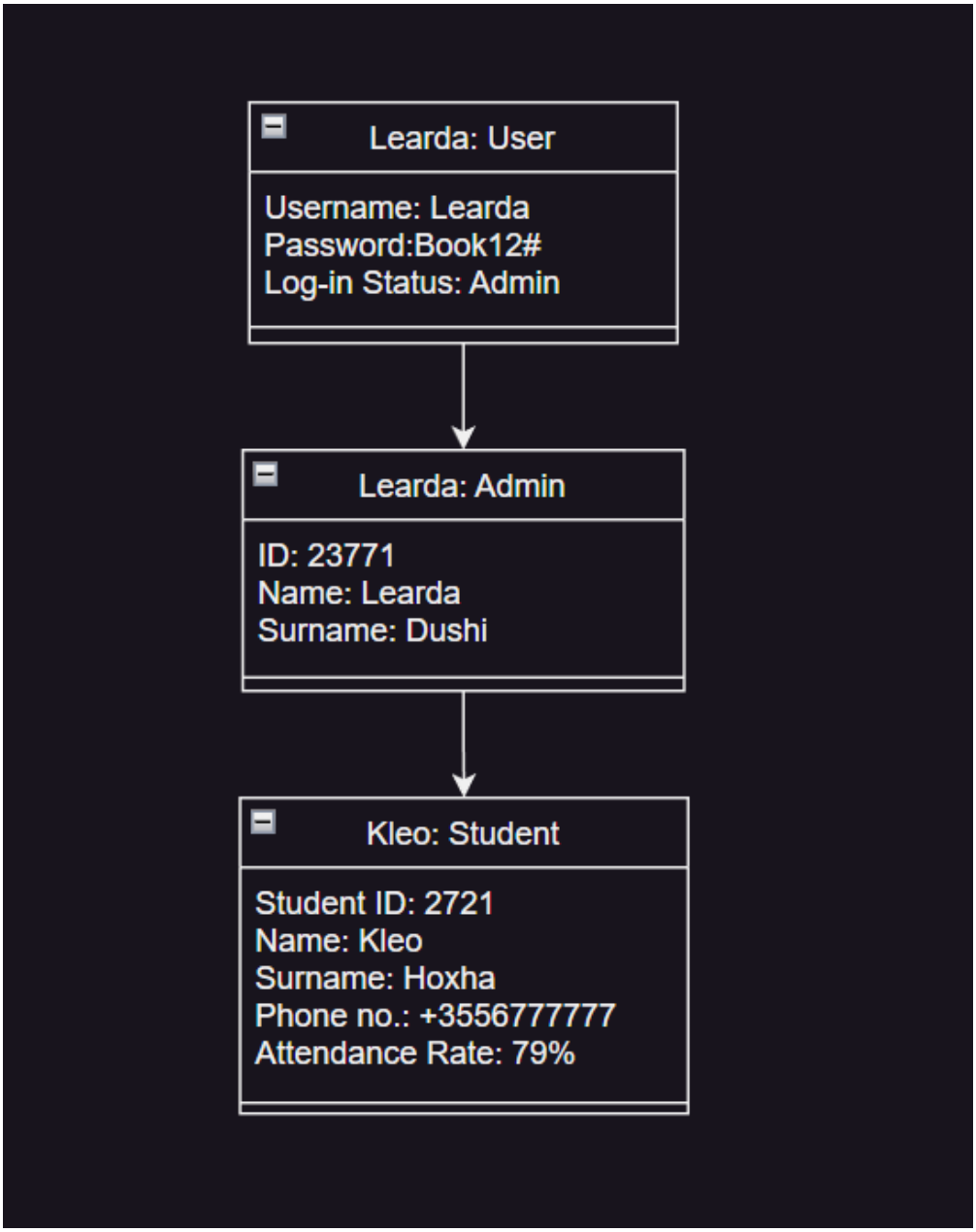
ERD:

An Entity-Relationship Diagram (ERD) is a visual representation of the entities (objects or concepts) within a system or application and the relationships between them. ERDs are commonly used in database design to model the structure of a database, illustrating how data is organized and how entities interact with each other.



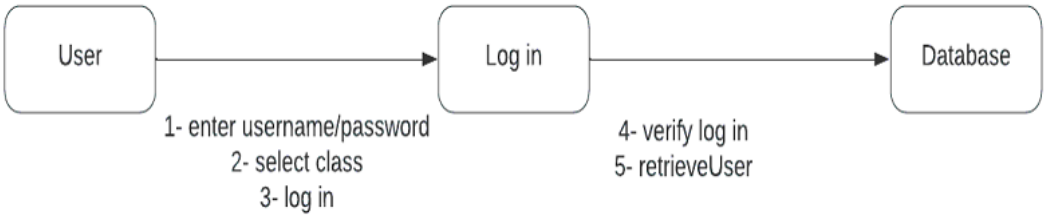

Object diagram:

An object diagram is a structural diagram in the Unified Modeling Language (UML) that provides a snapshot of a system at a specific point in time, focusing on instances of classes and their relationships. It visualizes how objects collaborate and interact with each other to achieve a specific scenario or functionality. Object diagrams are particularly useful for illustrating real-world examples and instances of a system's design.

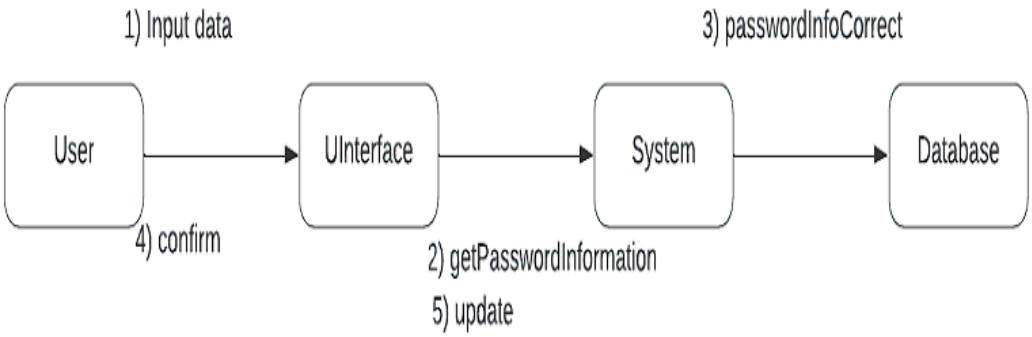



Collaboration diagram:

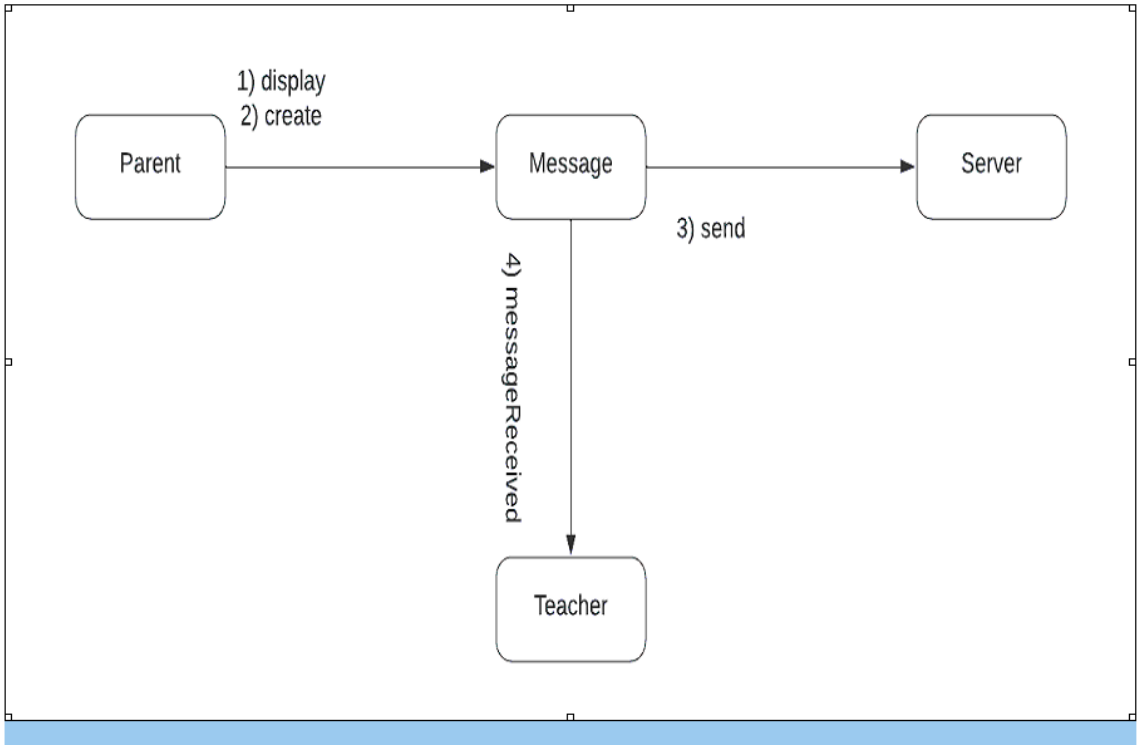
## Collaboration diagram (Log In)



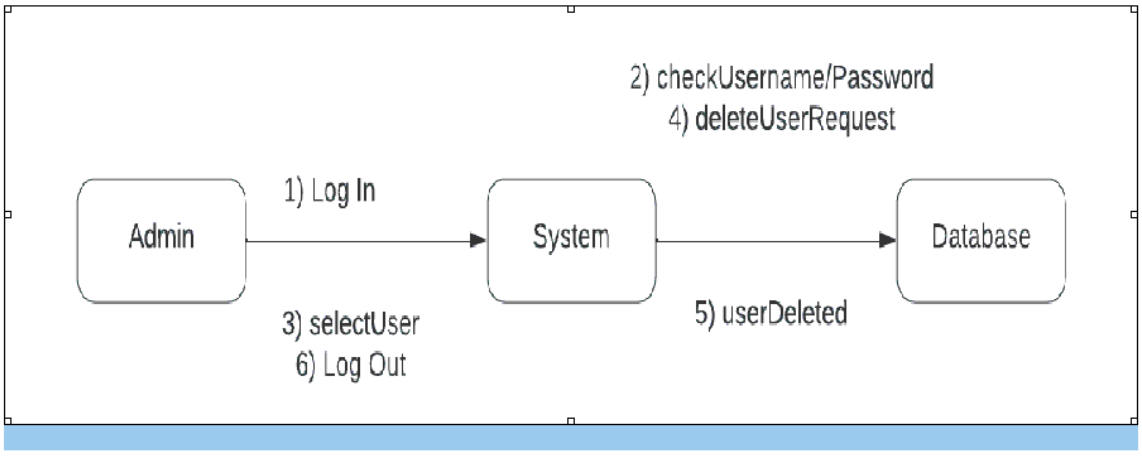

## Change password

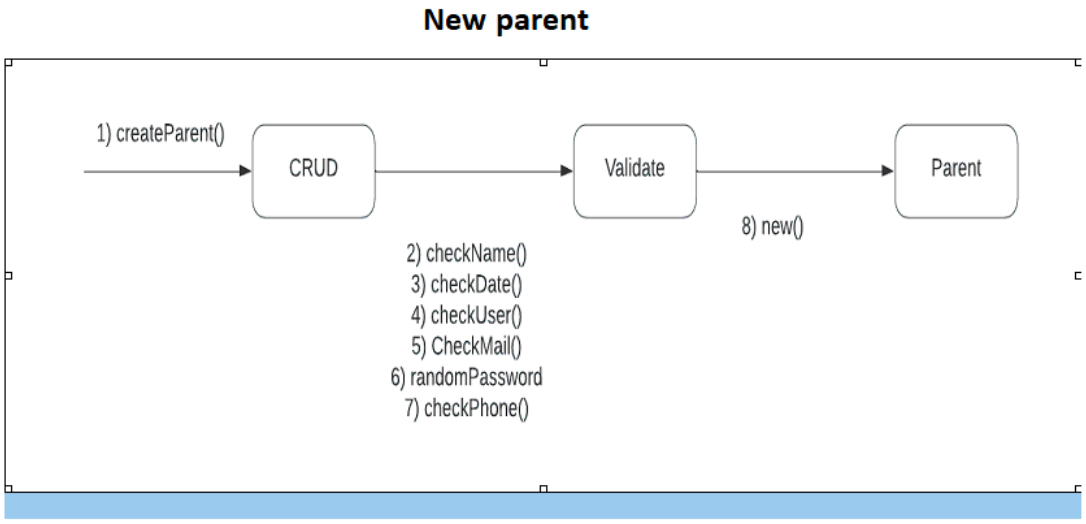


**Contact teacher**

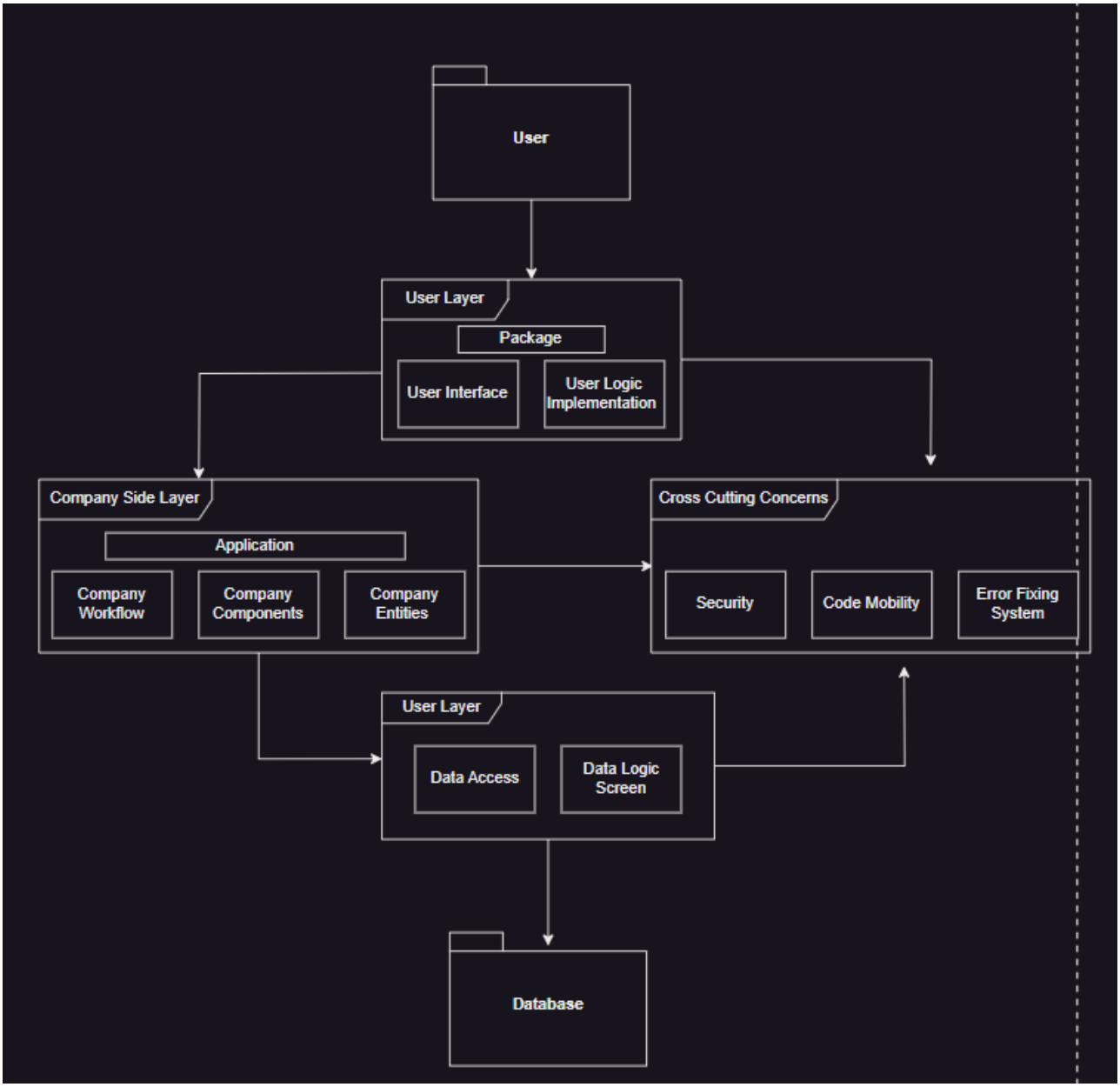



**Delete user**



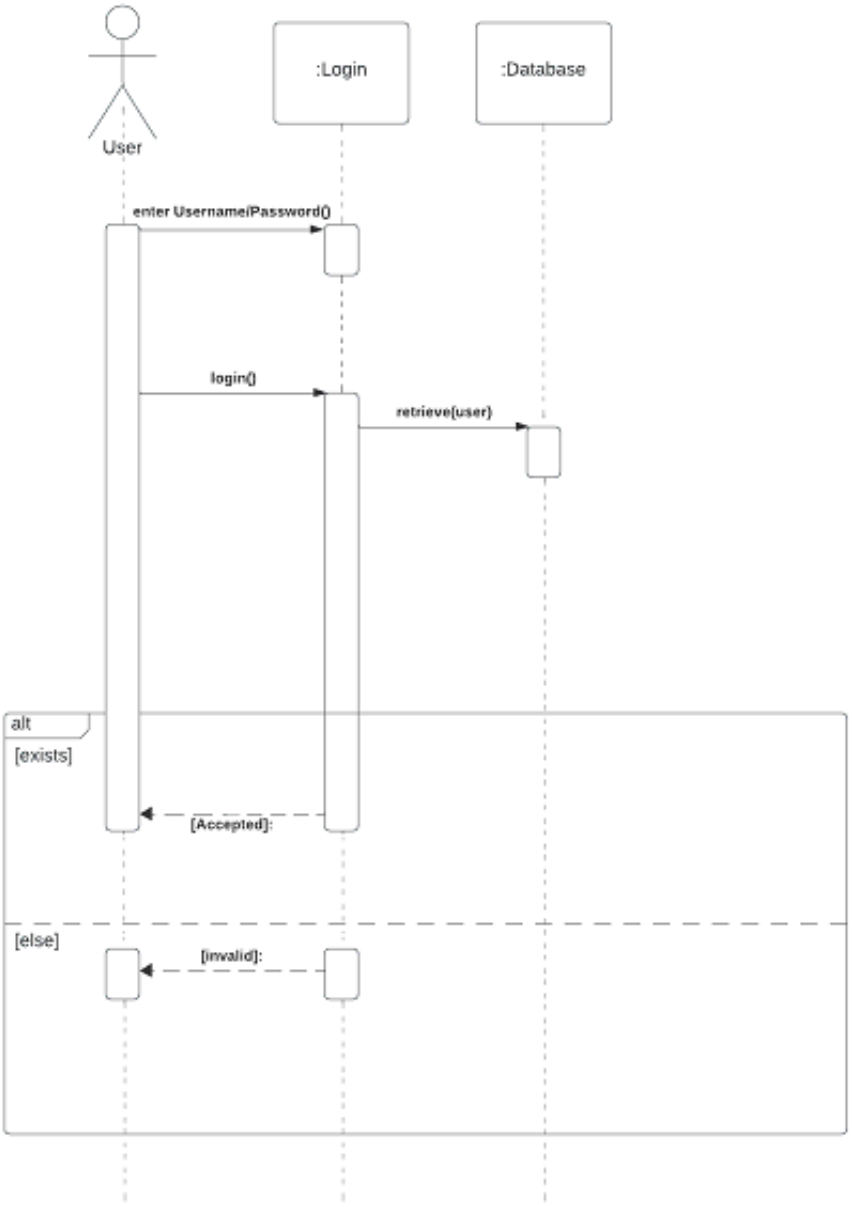



PACKAGE DIAGRAM: A package diagram is a type of diagram in the Unified Modeling Language (UML) that organizes and represents the elements of a system or software application into hierarchical structures known as packages.

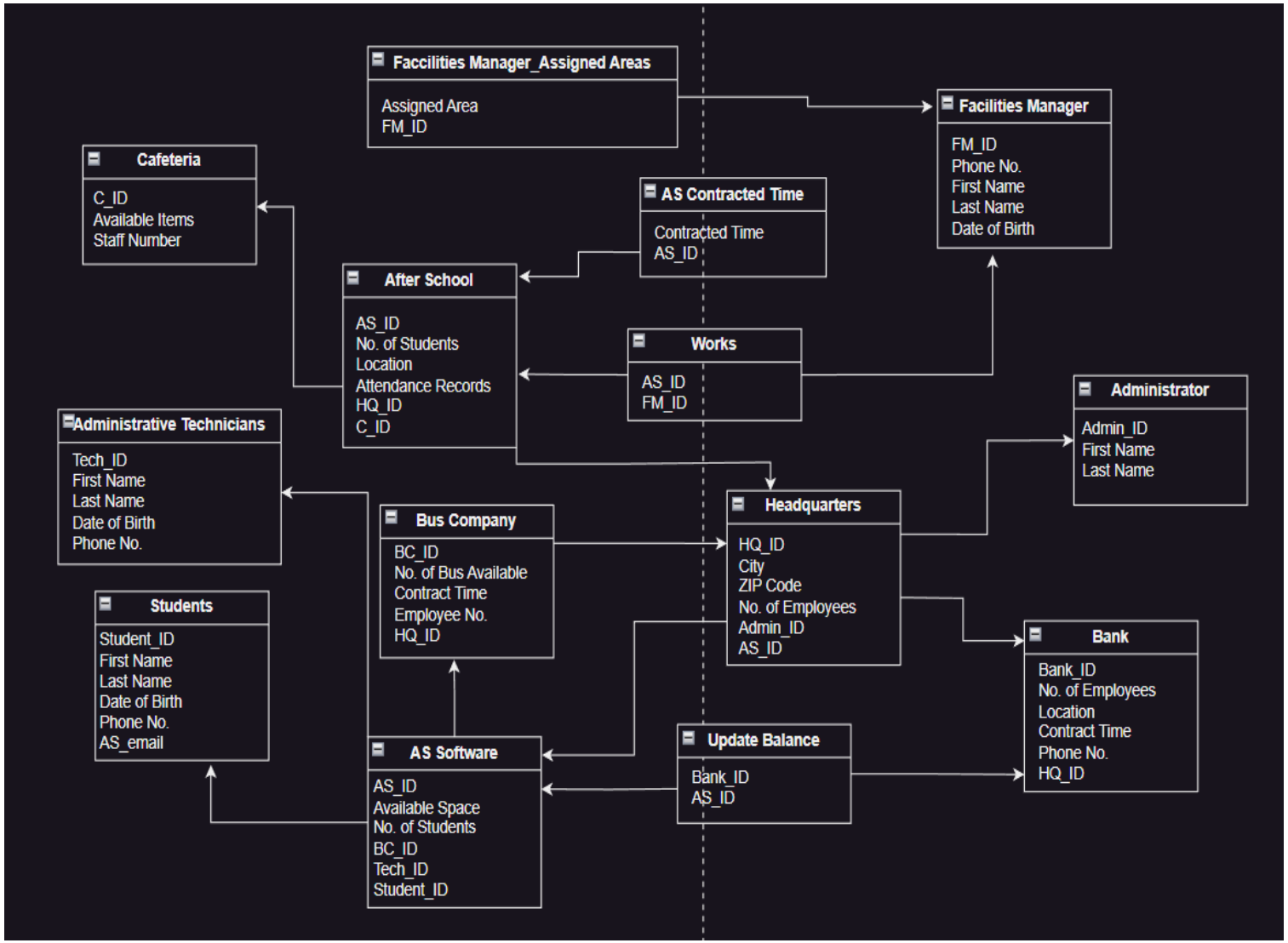



SEQUENCE DIAGRAM:

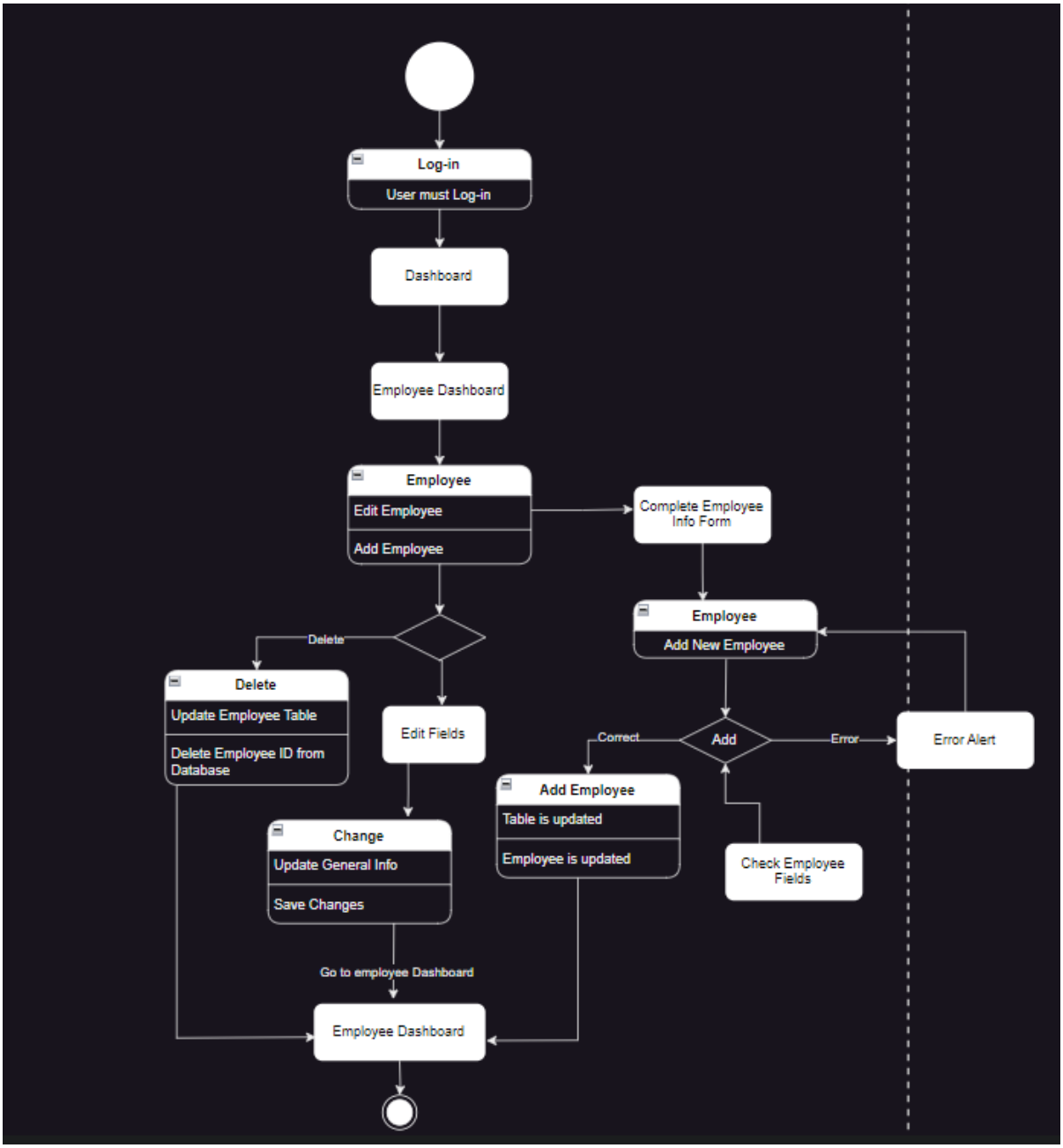
## Sequence diagram



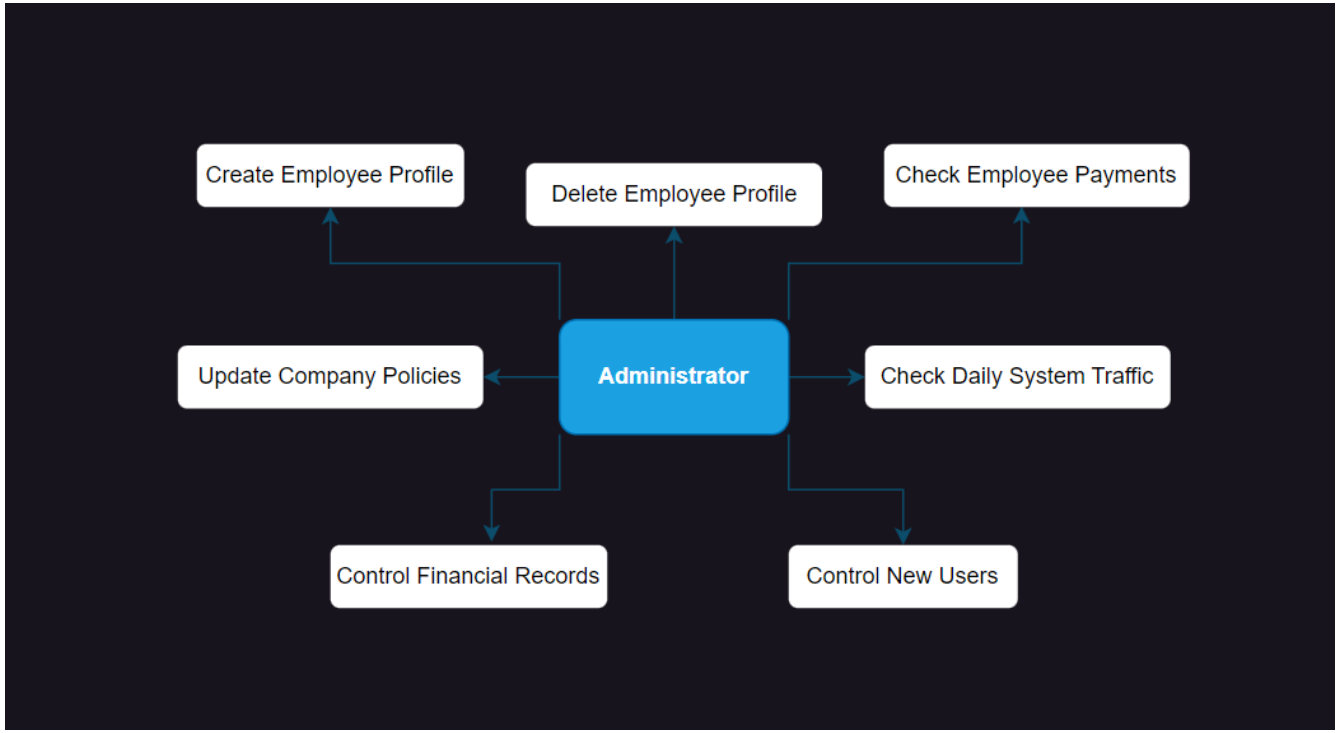

RATIONAL DIAGRAM:

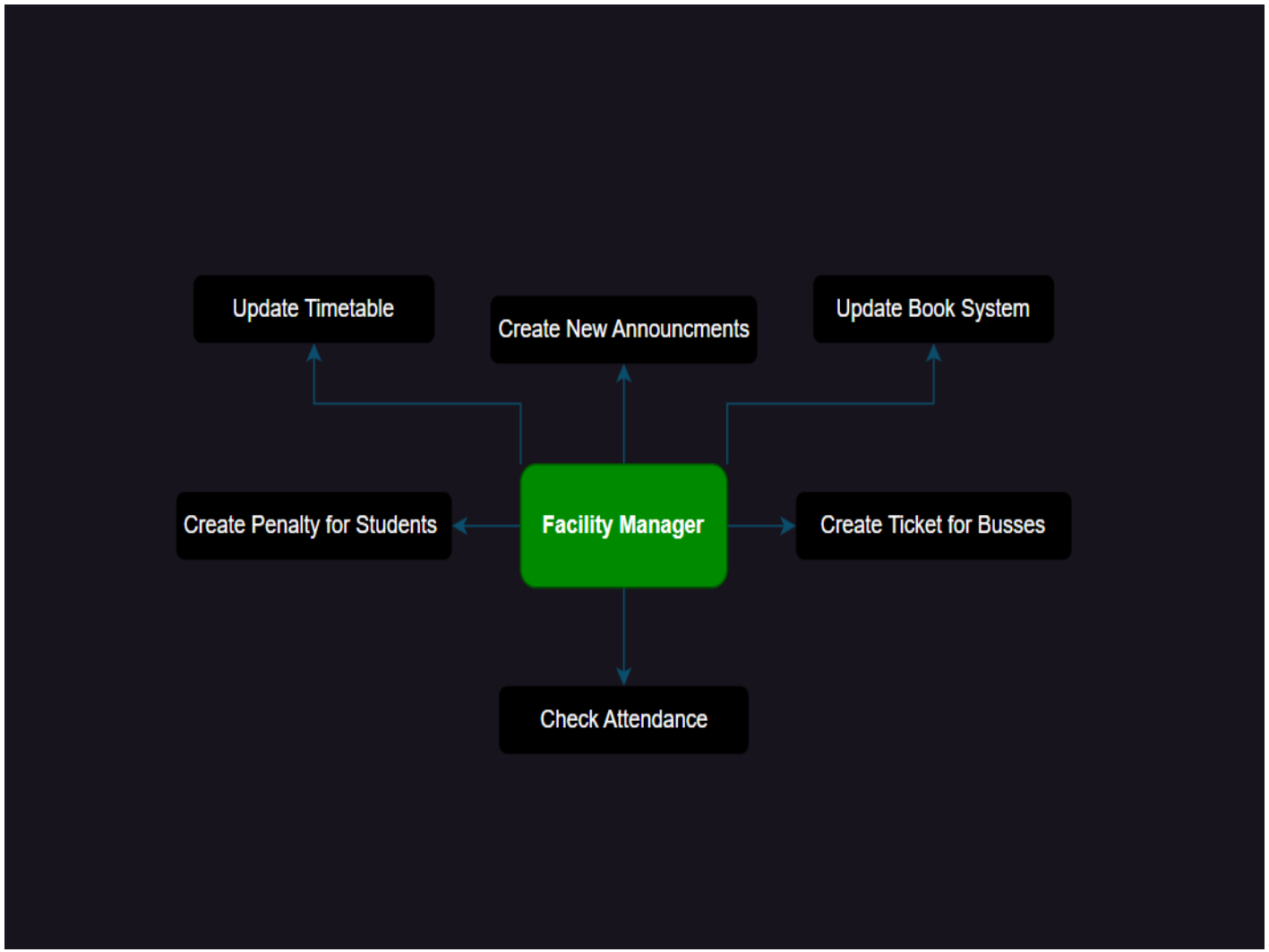

Faccilities Manager_Assigned Areas
Assigned Area
FM_ID
Facilities Manager
FM_ID
Phone No.
First Name
Last Name
Date of Birth
Cafeteria
C_ID
Available Items
Staff Number
AS Contracted Time
Contracted Time
AS_ID
After School
AS_ID
No. of Students
Location
Attendance Records
HQ_ID
C_ID
Works
AS_ID
FM_ID
Administrative Technicians
Tech_ID
First Name
Last Name
Date of Birth
Phone No.
Administrator
Admin_ID
First Name
Last Name
Bus Company
BC_ID
No. of Bus Available
Contract Time
Employee No.
HQ_ID
Headquarters
HQ_ID
City
ZIP Code
No. of Employees
Admin_ID
AS_ID
Students
Student_ID
First Name
Last Name
Date of Birth
Phone No.
AS_email
Bank
Bank_ID
No. of Employees
Location
Contract Time
Phone No.
HQ_ID
AS Software
AS_ID
Available Space
No. of Students
BC_ID
Tech_ID
Student_ID
Update Balance
Bank_ID
AS_ID

STATE DIAGRAMS: A state diagram, also known as a state machine diagram, is a type of behavioral diagram in the Unified Modeling Language (UML) that models the dynamic behavior of a system by representing its states, transitions between states, and the events that trigger those transitions.

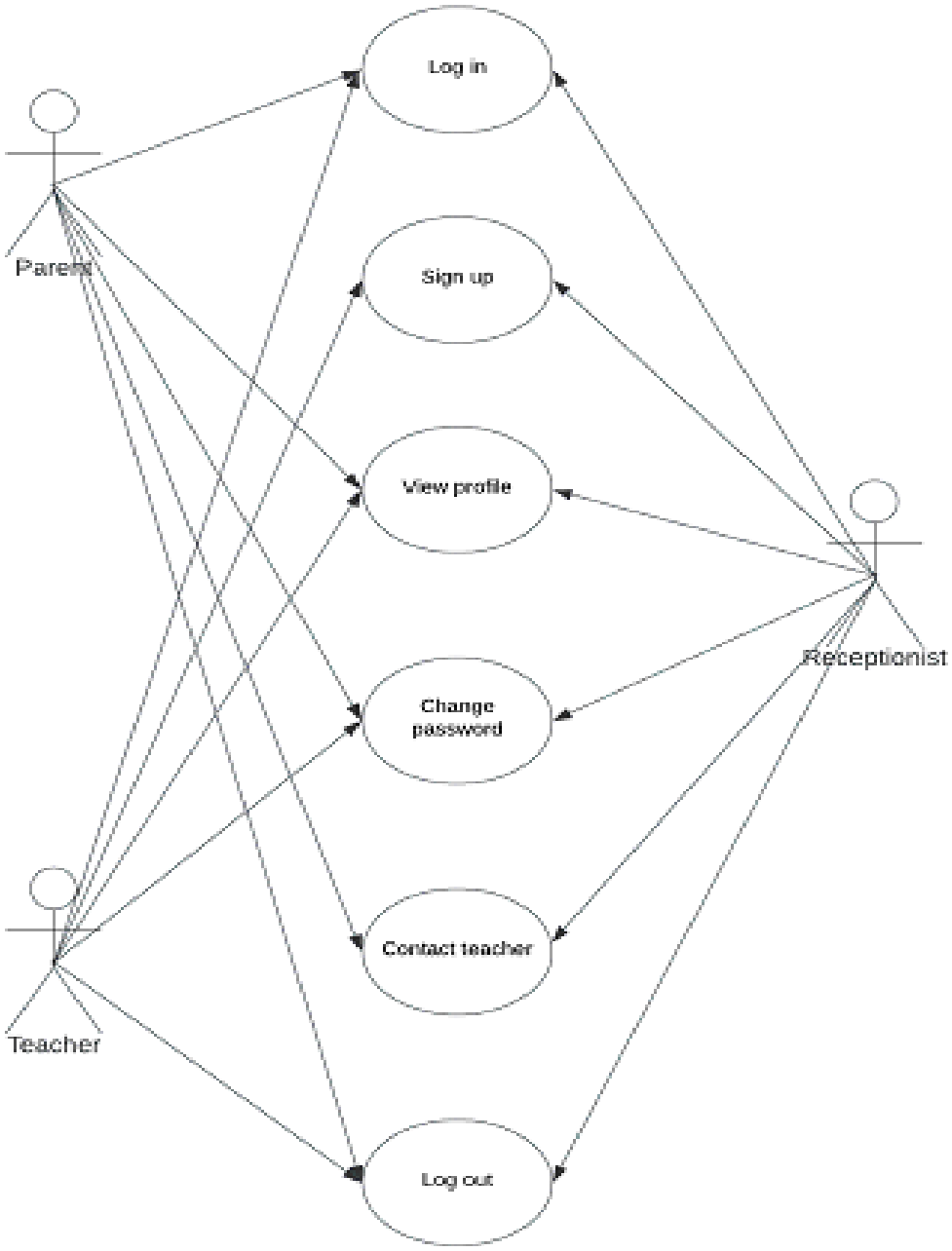


USE CASE DIAGRAMS: A use case diagram is a type of diagram in the Unified Modeling Language (UML) that provides a high-level visual representation of the interactions between actors (external entities such as users or other systems) and a system. Use case diagrams are primarily used to capture and communicate the functional requirements of a system from a user's perspective.

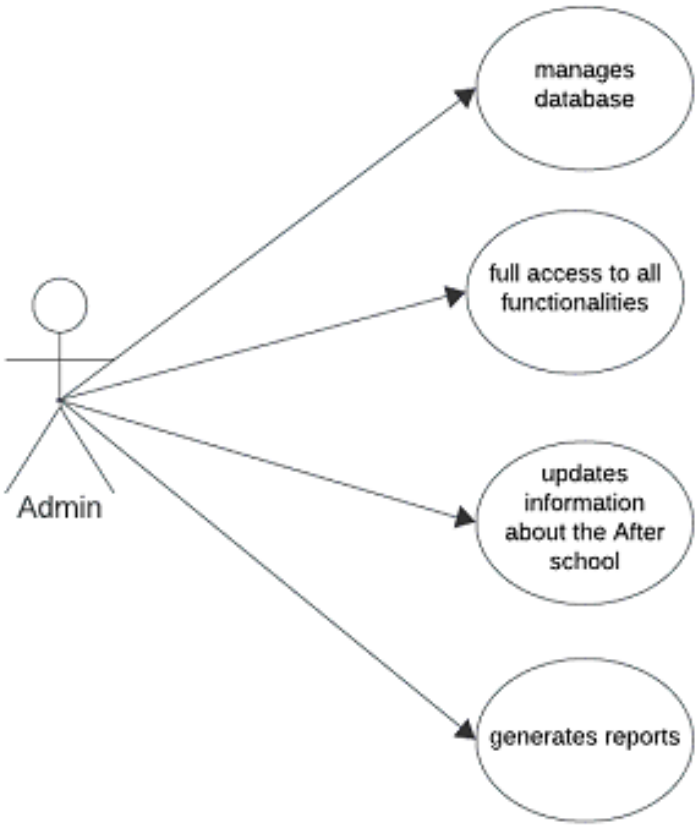

Create Employee Profile
Delete Employee Profile
Check Employee Payments
Update Company Policies
Administrator
Check Daily System Traffic
Control Financial Records
Control New Users

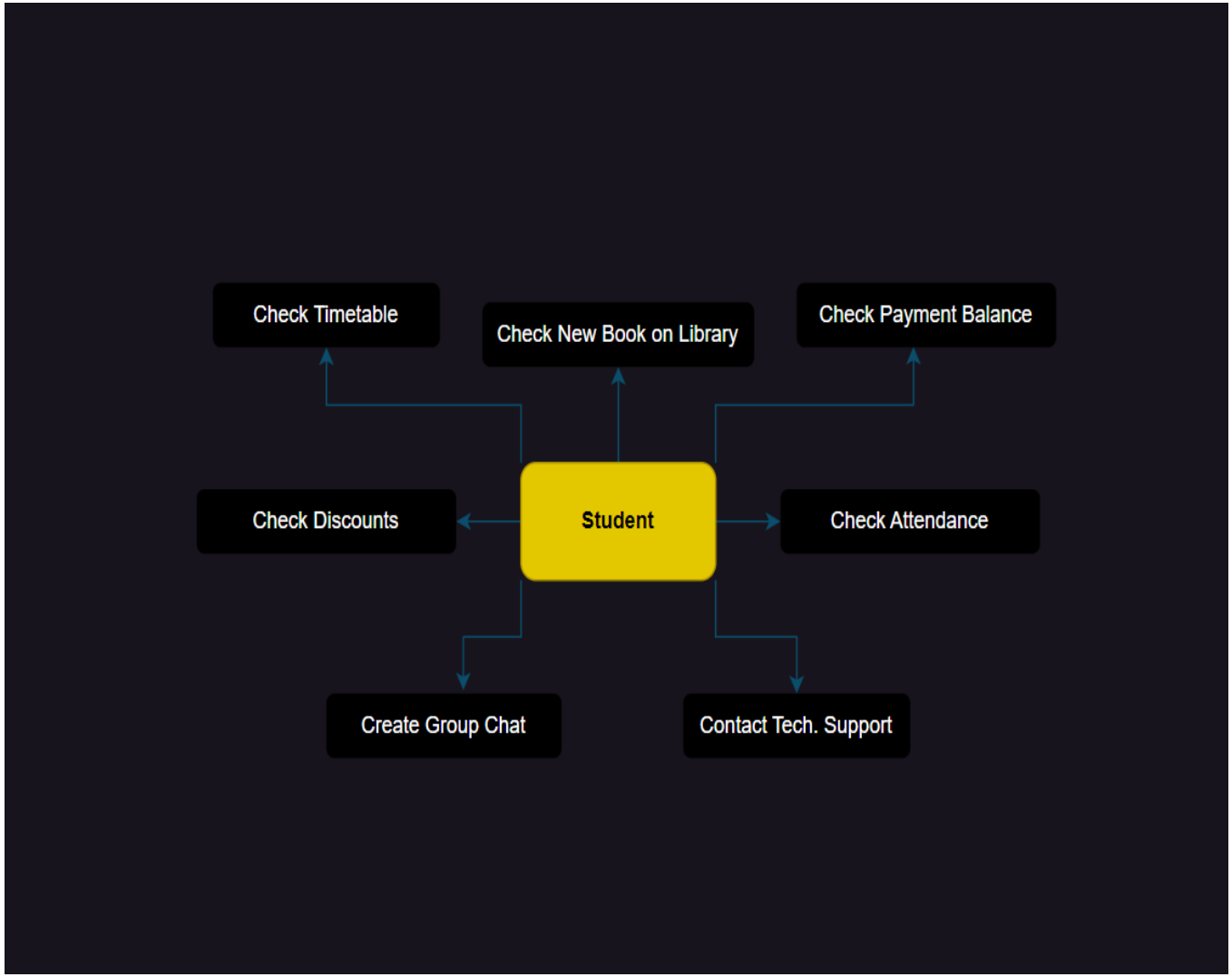

Update Timetable
Create New Announcments
Update Book System
Create Penalty for Students
Facility Manager
Create Ticket for Busses
Check Attendance

# Use Case diagram



## Use Case diagram




Check Timetable
Check New Book on Library
Check Payment Balance
Check Discounts
Student
Check Attendance
Create Group Chat
Contact Tech. Support