Muhammed Uwais Adam
Student Number: EDUV136908

SOFTWARE ENGINEERING

PROJECT 1

Project Number: 1

THE PROBLEM

You have been recruited by Eduvos to develop a system that will help track and manage occupancy of their parking garage and allow clients to find and reserve available parking spaces at Eduvos Office with ease.

The following issues concern management:
- Inefficient usage of parking space
- Congestion inside the garage, encountered whilst drivers try to find a vacant spot

Make a reservation

To make a reservation, the client must register with Eduvos via their website. During the registration, the client must provide their name, a valid email address, a licence plate number, a date for reservation, a duration for parking and a credit card number.

These details are checked and then stored in a database. If the details are incorrect, the user is asked to resubmit the information. The client can also update their reservation 24 hours prior to their booked reservation date. If the vehicle is borrowed or rented, the client must provide the licence plate number when making the parking reservation.

If the given registration number is different from the one in the client's profile, a temporary number associated with the client is to be created. This number will stay valid for the duration of the reservation.

Reservation options
Two types of reservations are available:
- Daily reservations
- Monthly reservations

The following devices must be installed inside the parking garage:

- Two licence plate readers – one at the garage entrance which will read the licence plate of the cars driving in and the other on the exit pathway which will read those leaving (using digital cameras).
- Sensors – these will be installed on every parking spot and will sense if the space is occupied or vacant.
- A debit machine for clients charged extra.

Payment

Registered users must pay their parking fee during registration, by using either a credit or debit card. Unregistered users must manually pay their parking fee with cash (deposited into a debit machine) or a credit or debit card.

Payment is made online via the Eduvos website. After the payment is made a receipt and reservation details are emailed to the client. If the parking duration is exceeded, the client will be charged extra per extra hour spent there (paid via a debit machine).

1. LIST THE ACTORS

ACTORS THAT PARTICIPATE IN THE PARKING GARAGE SYSTEM.

- Client
- Eduvos Website
- License Plate Reader
- Sensor's
- Payment Processor

2. DETERMINE THE USE CASES

USE CASE ANALYSIS IN THE PARKING GARAGE SYSTEM.

- Reserve a Parking
- Find Available Parking Spots
- Pay for Parking
- Update Parking Reservation
- Pay for Exceeded Parking Time
- Send Parking Notifications

Author: Muhammed Uwais Adam

Version: Version Number 1

3. ELABORATE SCENARIOS FOR RESERVATIONS

Primary scenario: Client Makes a reservation for a parking

- A client uses the Eduvos website to fill in his/her details and register to make a reservation for a parking spot.
- The systems sends a booking confirmation email.

Alternative: The client gives incorrect details.

- The system informs the user to resubmit the information.

Primary scenario: The client updates their booking 24 hours prior

- The system updates booking reservation.
- The system sends a updated booking confirmation email.

Alternative: The client does not update their booking 24 hours prior

- The system terminates the reservation.
- The system informs the client that the booking has been terminated.
- The systems also send the client a option to keep current reservation before terminating with a time limit of 20 minutes to respond.
- After 20 minutes of informing the client, the reservation is terminated.

Primary scenario: The client arrives in the registered registration number

- The system signs the client in.
- The systems starts a timer.
- The system corresponds with sensors to find a available parking spot

Alternative: The client arrives with a different registration number

- The client updates his/her registration in a temporary section on the system.
- The system issues a temporary number to the registration.

Primary scenario: The Client registers and the payment is successful

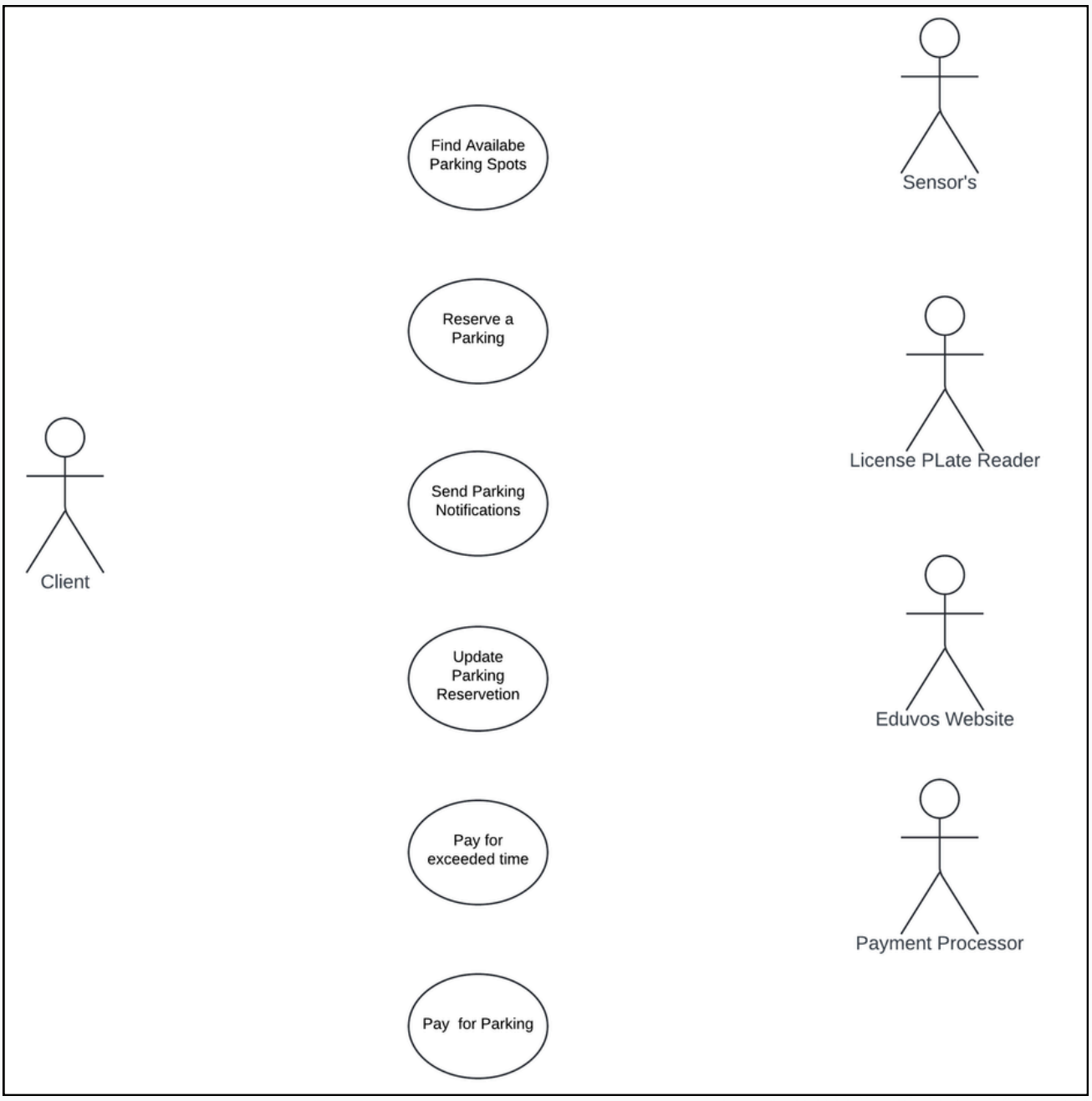
- The system emails a receipt and reservation details.
- The system stores clients details to a database

Alternative: The Client registers and the payment is un-successful

- The correct details of client is stored in a database.
- The systems informs customer of payment failure.
- The system sends a payment link.

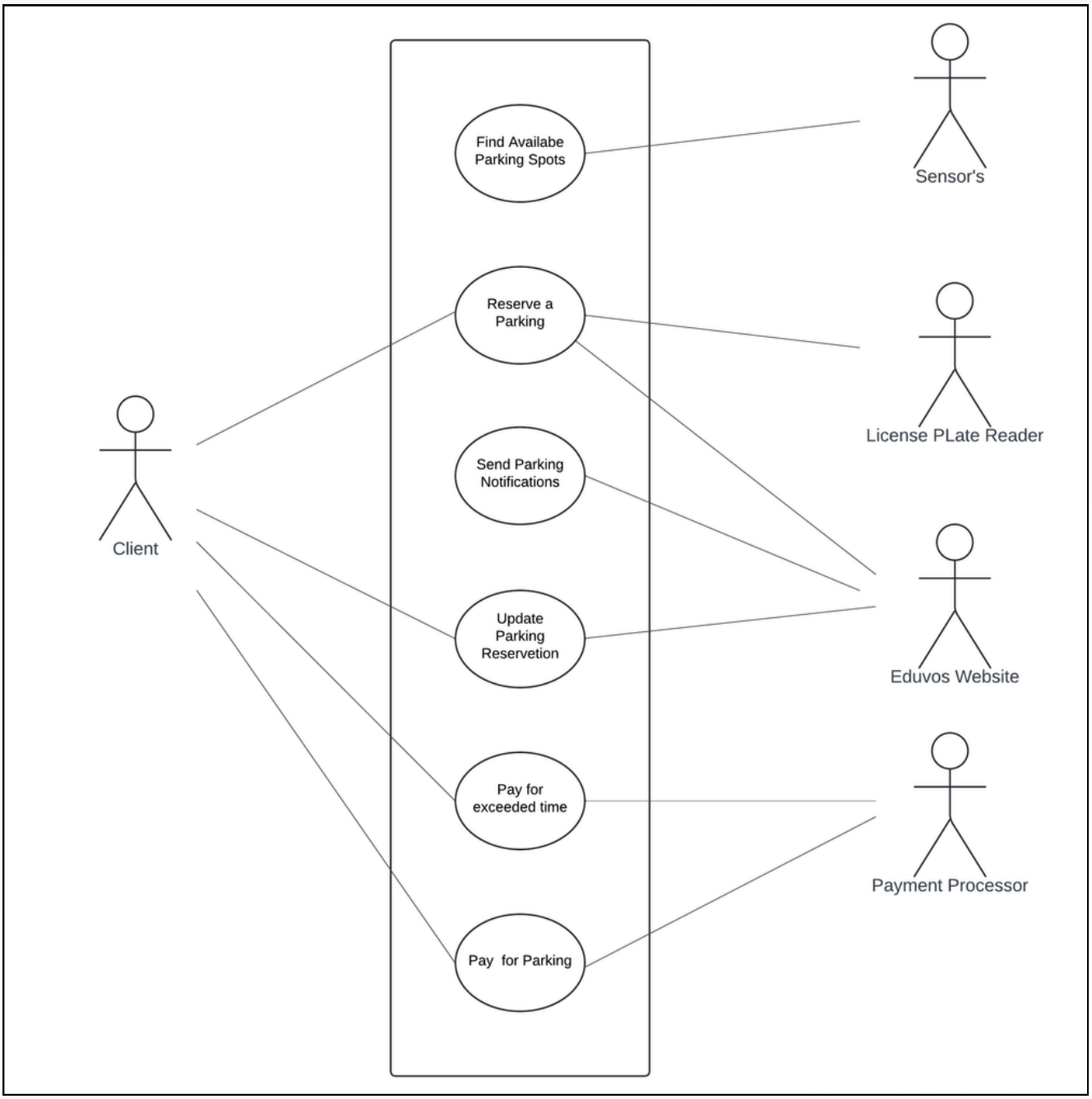
4. USE CASE DIAGRAM FOR PARKING GARAGE SYSTEM

DOCUMENTATION



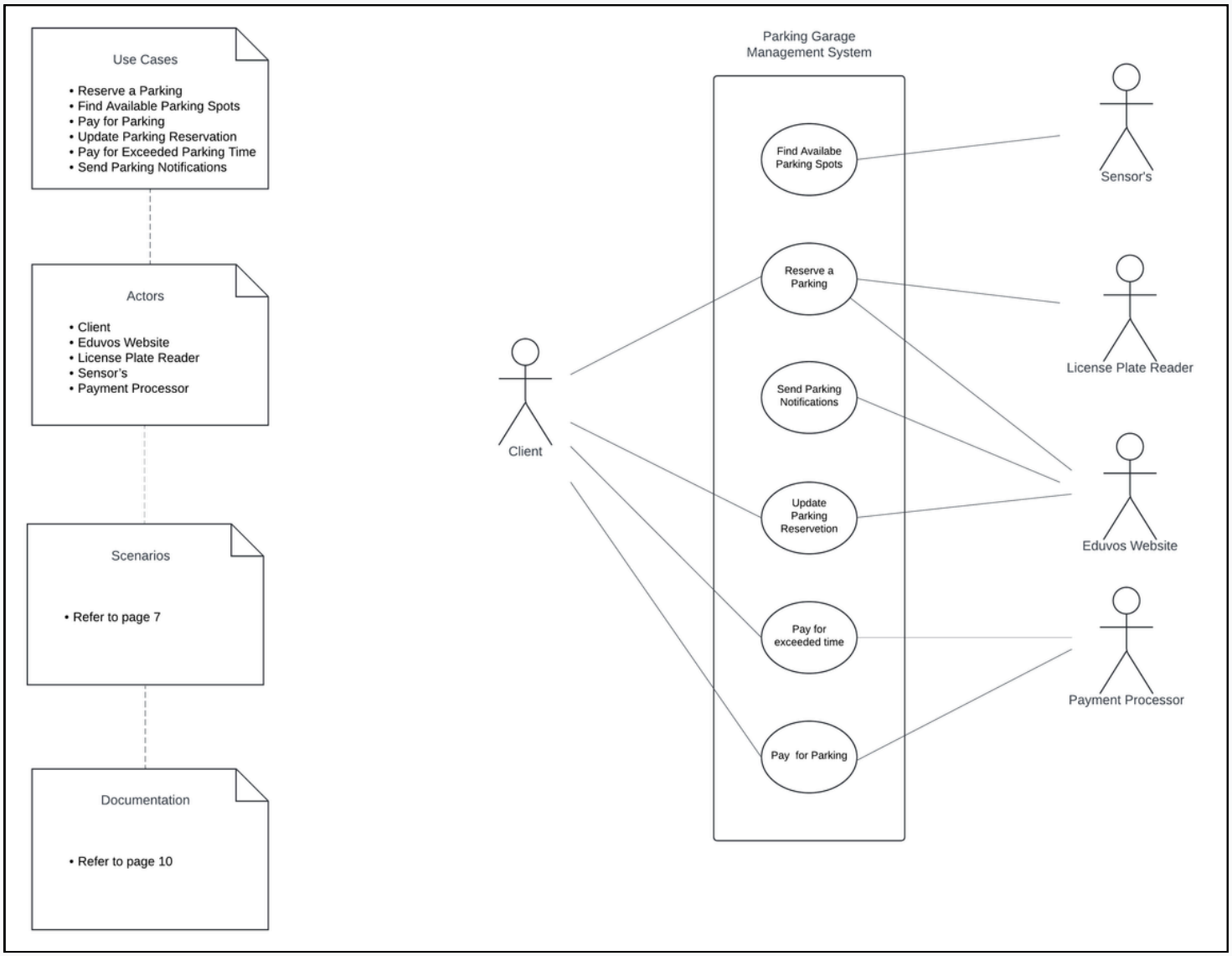
4. USE CASE DIAGRAM FOR PARKING GARAGE SYSTEM

DOCUMENTATION



4.1. USE CASE DIAGRAM FOR PARKING GARAGE SYSTEM

FINAL USE CASE DIAGRAM

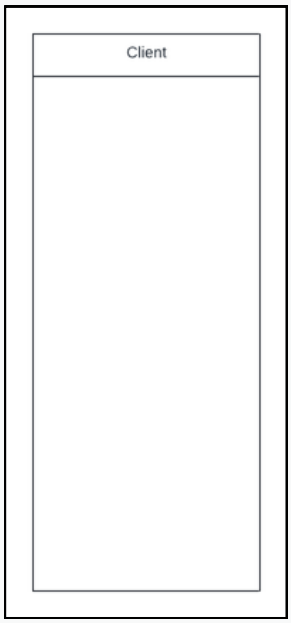


- The sensors installed in the garage find available Parking Spots.
- Client reserves a parking.
- The License Plate Reader takes details of number plate.
- Client updates parking reservation
- The system/Eduvos website captures reserve parking details
- The system/Eduvos website sends parking notifications
- The system/Eduvos website updates parking reservations
- Client Pays for parking
- Payment Processors capture payments made

5. ACTIVITY DIAGRAM FOR PARKING GARAGE SYSTEM

DOCUMENTATION

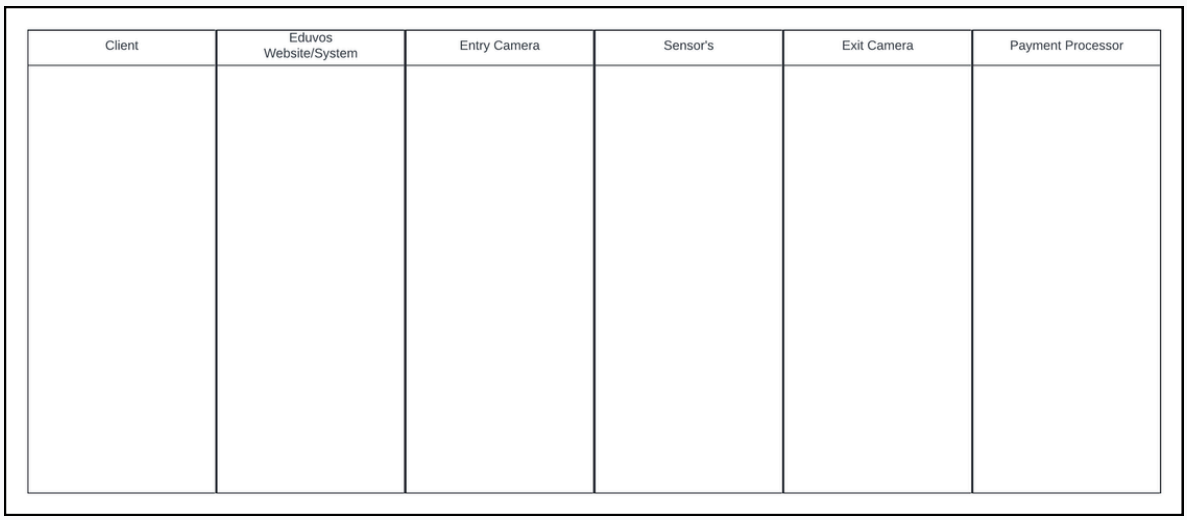
SHOWING SWIMLANES



Activity partitions (also known as swimlanes in earlier versions of UML) are demarcated areas that show which actors are responsible for the activities that take place in each ‘lane’.

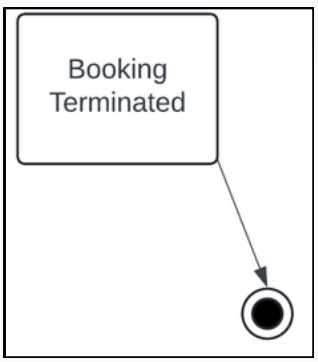
The name of the actor is at the top of the partition.

Partitions are the actors in a use case.

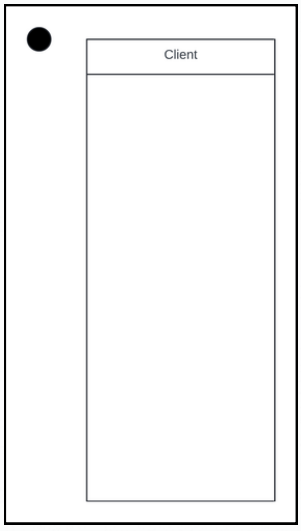


SHOWING INITIAL AND END ICONS

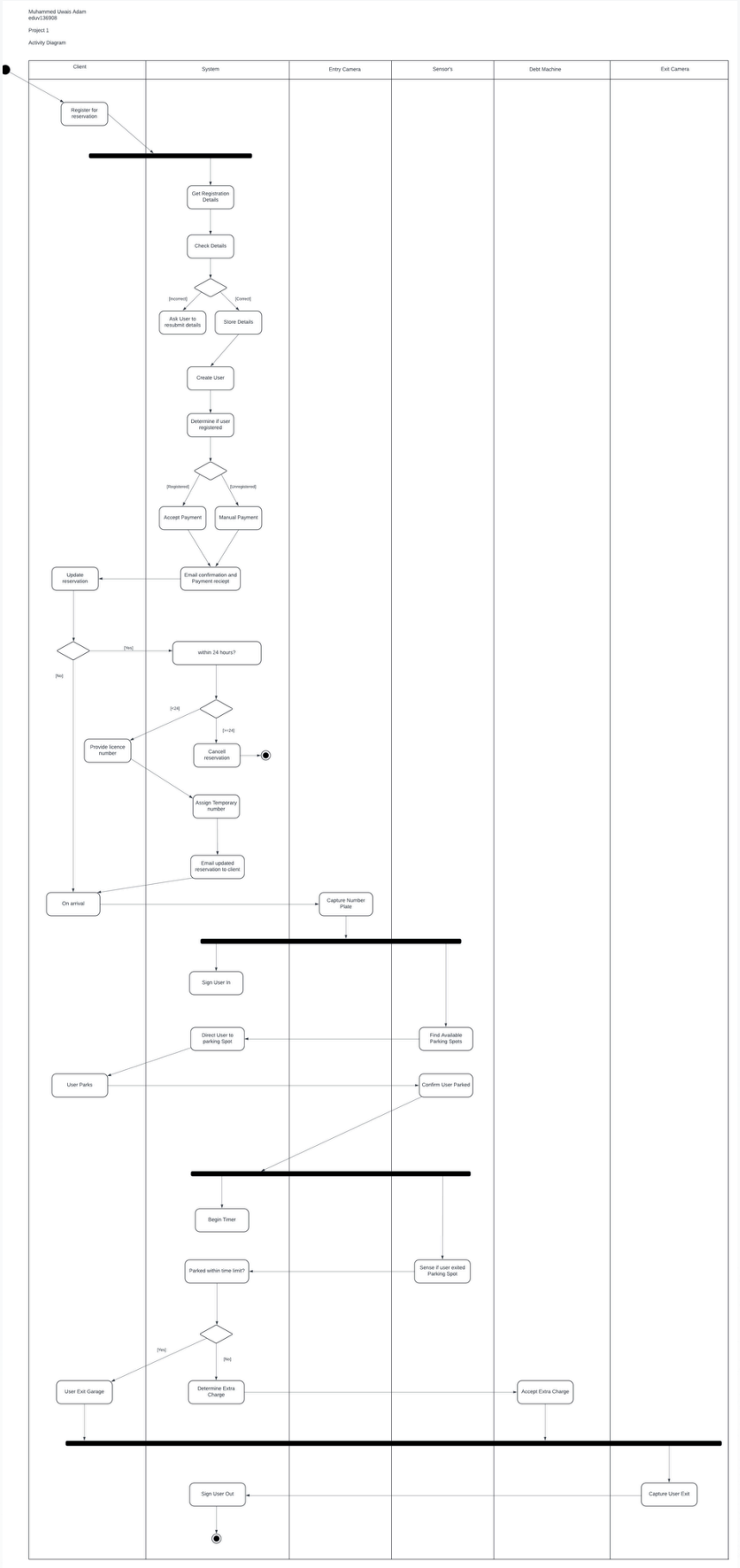
End Icon indicated as 2 circles with a white filler outer and a black filled inner .



Initial Icon indicated as a filled black circle.



5. ACTIVITY DIAGRAM FOR PARKING GARAGE SYSTEM



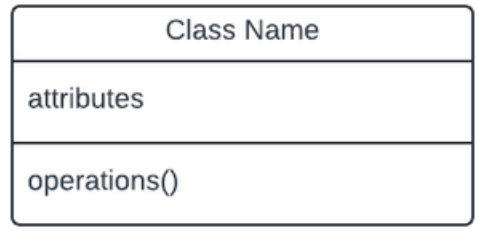
PLEASE SEE FINAL ACTIVITY DIAGRAM ATTACHED

6. CLASSES AND THE CLASS DIAGRAM FOR PARKING SYSTEM

CLASS DIAGRAM DOCUMENTATION

How classes and their relationships are defined in the diagram:

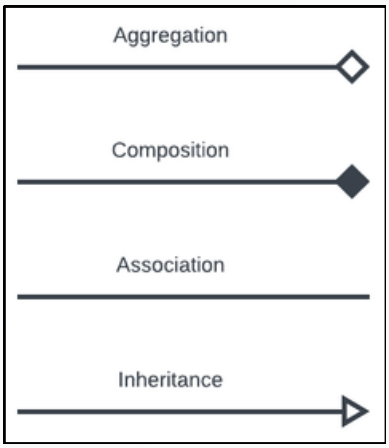
In the Parking System class diagram, a class is represented by a rectangular shape. The rectangle is divided into sections to clearly show the class name at the top, followed by a list of its attributes (properties or data members) in the middle, and finally its behaviors (operations or methods) at the bottom.



CLASS DIAGRAM DOCUMENTATION

How classes and their relationships are defined in the diagram:

- Aggregation: This is a "has-a" relationship where one class (the whole) is made up of or contains other classes (the parts), but the parts can exist independently of the whole.
- Composition: This is a stronger form of aggregation, also a "has-a" relationship. The parts are entirely dependent on the whole and cannot exist without it.
- Association: This is a general relationship between two classes, indicating some kind of connection or interaction.
- Inheritance: This is an "is-a" relationship where one class (the child class) inherits the attributes and methods of another class (parent class).

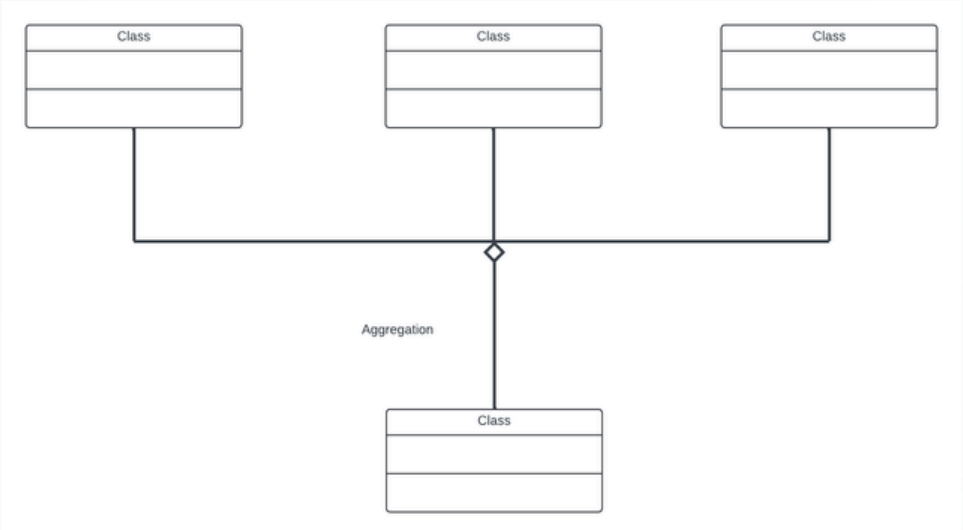


CLASS DIAGRAM DOCUMENTATION

How classes and their relationships are defined in the diagram:

This is a simplified representation using a single diamond symbol attached to the aggregating class makes the diagram more visually appealing, easier to understand, and more scalable for larger systems.

It communicates the concept of aggregation in this diagram but can be any other relationship, while minimizing visual complexity.



6. CLASSES AND THE CLASS DIAGRAM FOR PARKING SYSTEM

CLASS DIAGRAM DOCUMENTATION

A Parking Garage has an (aggregated) Entry Camera Class, Sensors, and an Exit Camera Class.
The cameras are fitted with a Number Plate Reader (NPR). Both cameras capture the client's number plate, while the sensors determine which parking spots are available.

A parking garage also has an (aggregated) debt machine to charge clients for overtime and unregistered users.
A parking garage has an (aggregated) system. The Parking Garage class was added so that the system is not limited to only one garage, allowing Eduvos to expand their parking garage or use the same system across all campuses.

The system is associated with a GUI class. The GUI class captures client information and updates to reservations. The system is composed of a database; without this database, the system cannot exist meaningfully.

The System class gets the client's information, checks it, and sends it to the database if correct. The System class creates a User on the system, representing the client. The System class then starts a timer, determines how long the user parked overtime, and finally calculates the cost.

The database class stores all details provided by the system and updates any details or users where applicable.

The System class has a User.

The User class stands between the Reservation class and other classes. The User has a Vehicle, and if the user has a borrowed vehicle, the Borrowed Vehicle class inherits its data types, etc., from its parent class, Vehicle. The client has a Reservation, so a Reservation class was created.

The Reservation class gets all the details from the system and updates the reservation if there is an update. The Reservation class has a Payment class. The Payment class accepts user payments and sends a receipt to the Reservation class, which then emails the client with the reservation details and receipt.

The Reservation class is directly associated with the Borrowed Vehicle class so it can assign a temporary number if the user updates the reservation with a borrowed vehicle.

There are two types of reservations: Monthly and Daily. Both inherit operations and attributes from their parent class, Reservation.

Since there is no stated difference between the reservations, they were added for future purposes, should Eduvos decide to add different rules between the two reservation types.

6. CLASSES AND THE CLASS DIAGRAM FOR PARKING SYSTEM

A Parking Garage has an (aggregated) Entry Camera Class, Sensors, and an Exit Camera Class.
The cameras are fitted with a Number Plate Reader (NPR). Both cameras capture the client's number plate, while the sensors determine which parking spots are available.

A parking garage also has an (aggregated) debt machine to charge clients for overtime and unregistered users.
A parking garage has an (aggregated) system.

The Parking Garage class was added so that the system is not limited to only one garage, allowing Eduvos to expand their parking garage or use the same system across all campuses.
The system is associated with a GUI class. The GUI class captures client information and updates to reservations.

The system is composed of a database; without this database, the system cannot exist meaningfully. The System class gets the client's information, checks it, and sends it to the database if correct. The System class creates a User on the system, representing the client. The System class then starts a timer, determines how long the user parked overtime, and finally calculates the cost.

The database class stores all details provided by the system and updates any details or users where applicable.

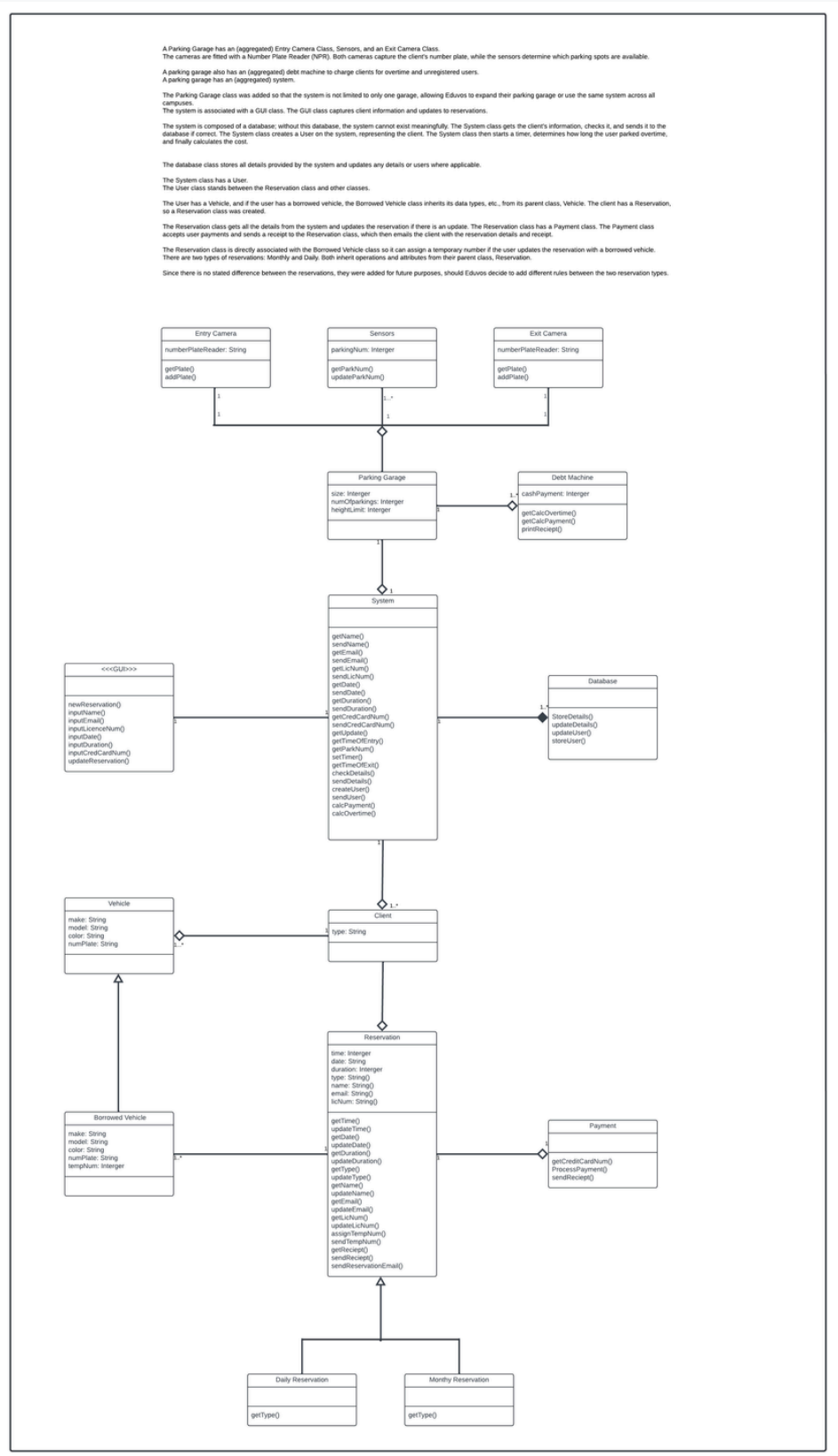
The System class has a User.
The User class stands between the Reservation class and other classes.

The User has a Vehicle, and if the user has a borrowed vehicle, the Borrowed Vehicle class inherits its data types, etc., from its parent class, Vehicle. The client has a Reservation, so a Reservation class was created.

The Reservation class gets all the details from the system and updates the reservation if there is an update. The Reservation class has a Payment class. The Payment class accepts user payments and sends a receipt to the Reservation class, which then emails the client with the reservation details and receipt.

The Reservation class is directly associated with the Borrowed Vehicle class so it can assign a temporary number if the user updates the reservation with a borrowed vehicle.
There are two types of reservations: Monthly and Daily. Both inherit operations and attributes from their parent class, Reservation.

Since there is no stated difference between the reservations, they were added for future purposes, should Eduvos decide to add different rules between the two reservation types.



PLEASE SEE FINAL CLASS DIAGRAM ATTACHED

7. COMMUNICATION DIAGRAM FOR DISPLAYING A BOOKING LIST

COMMUNICATION DIAGRAM DOCUMENTATION

The communication diagram starts with the :System object that will get() the booking details from the database. As mentioned in the scenario that only correct details are stored in the database hence why we get the booking details from the database.

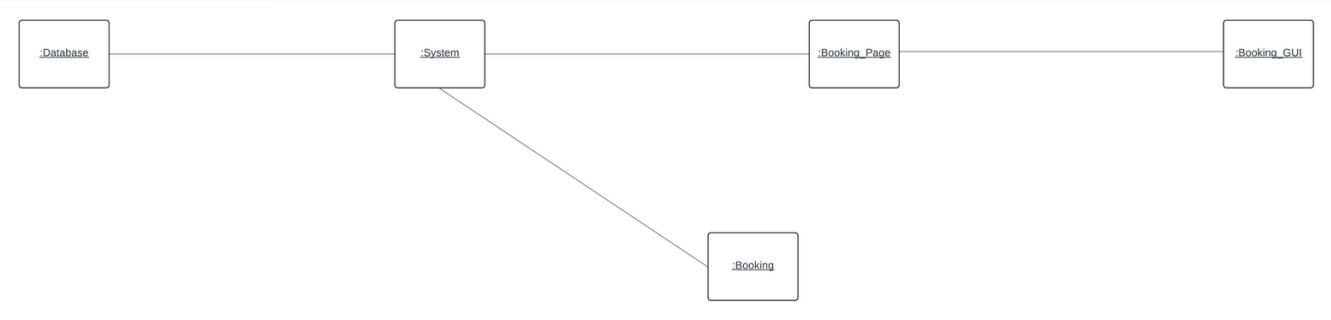
The :System object then <<<creates>>> a :Booking object which contains booking details.
The :System can create multiple booking objects at a time.

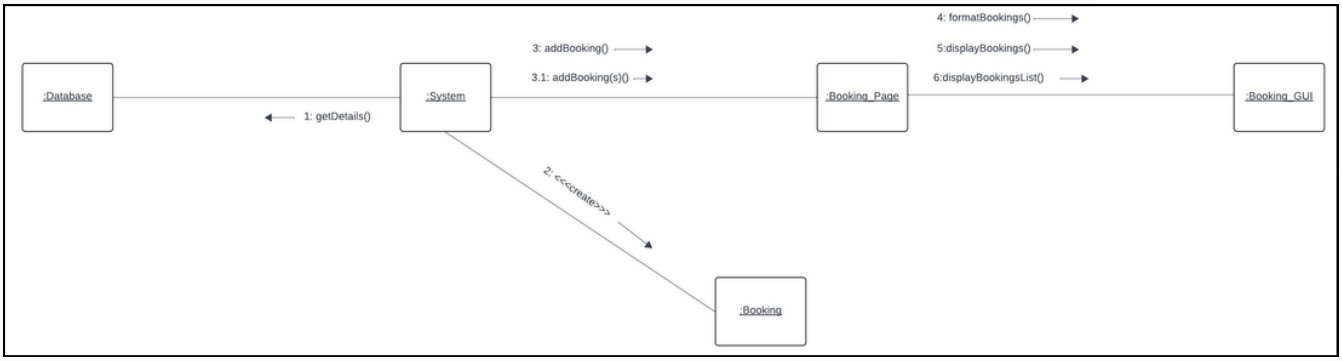
The :System object then adds() the booking and all bookings to the :Booking_Page object.

The :Booking_Page object then formats the bookings, displays the bookings and displays a booking list to the :Booking_GUI object.

The Administrator can now view all bookings in a list format via the GUI.

Since the question does not specify how a administrator interacts with the GUI, I left the object out for better readability.





Communication Diagram Final.

8. STATE MACHINE DIAGRAM FOR AN OBJECT OF THE ACCOUNT CLASS

STATE MACHINE DIAGRAM DOCUMENTATION

The Account object is created, The Account object can be one of two states: Unregistered and Registered.

If unregistered the state then goes to debt machine payment to indicate this Account will be paid physically. Unregistered Accounts get their reservation instantly.
From Debt Machine Unregistered Accounts goes into the state Reservation Confirmed.

If the Account is registered, it goes into a reservation pending state where payments is accepted. From reservation pending to reservation payment.

If the payment is successful the reservation is confirmed, and if not it is cancelled.

The reservation confirmed state goes to a booking pending state, if a account updates its booking within 24 hours the booking is confirmed.

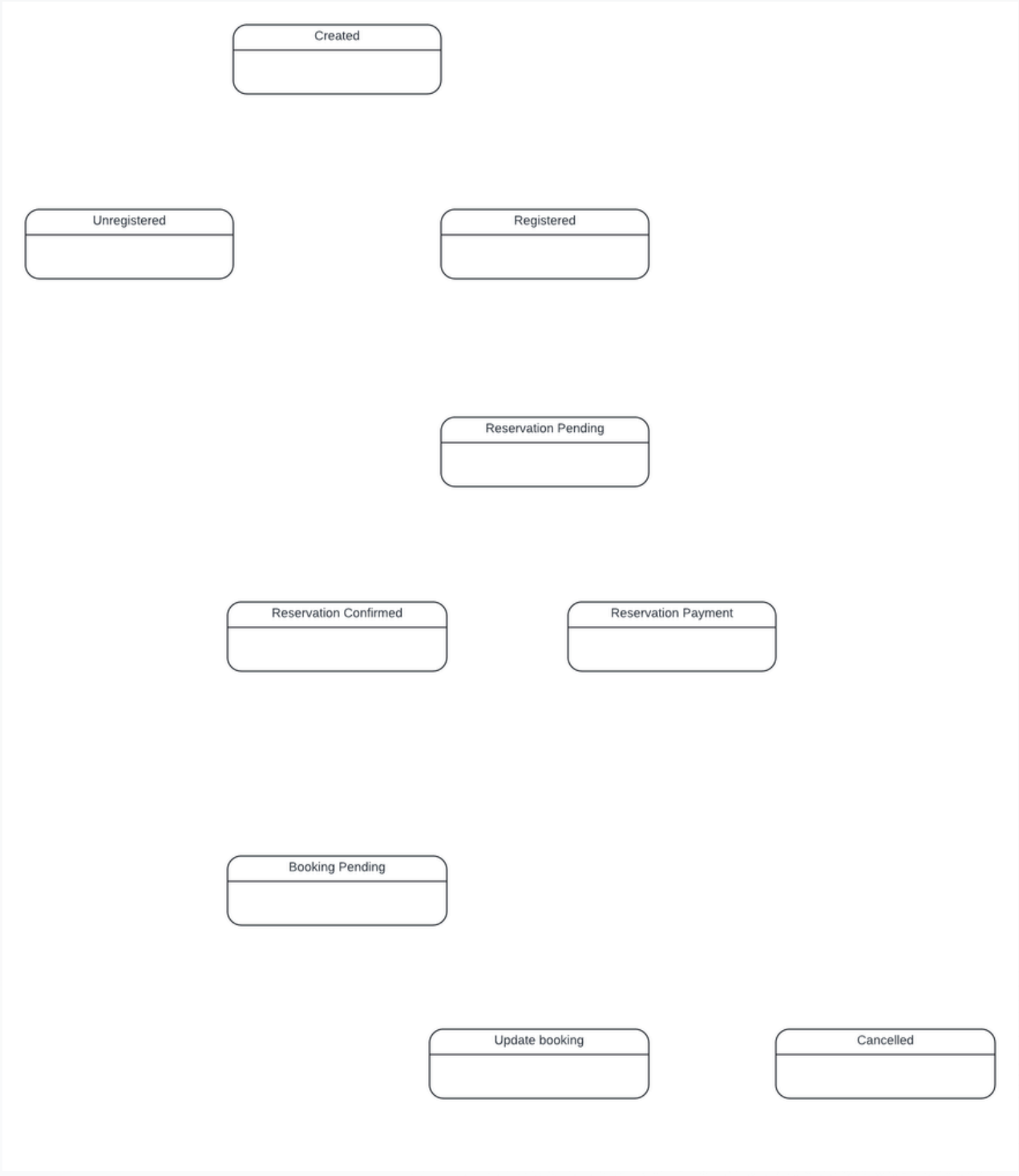
If a account updates booking over 24 hours the booking is cancelled.

If there are no updates and the updates to the booking was successful the booking is confirmed and the process ends.

8. STATE MACHINE DIAGRAM FOR AN OBJECT OF THE ACCOUNT CLASS

STATE MACHINE DIAGRAM DOCUMENTATION

Showing States

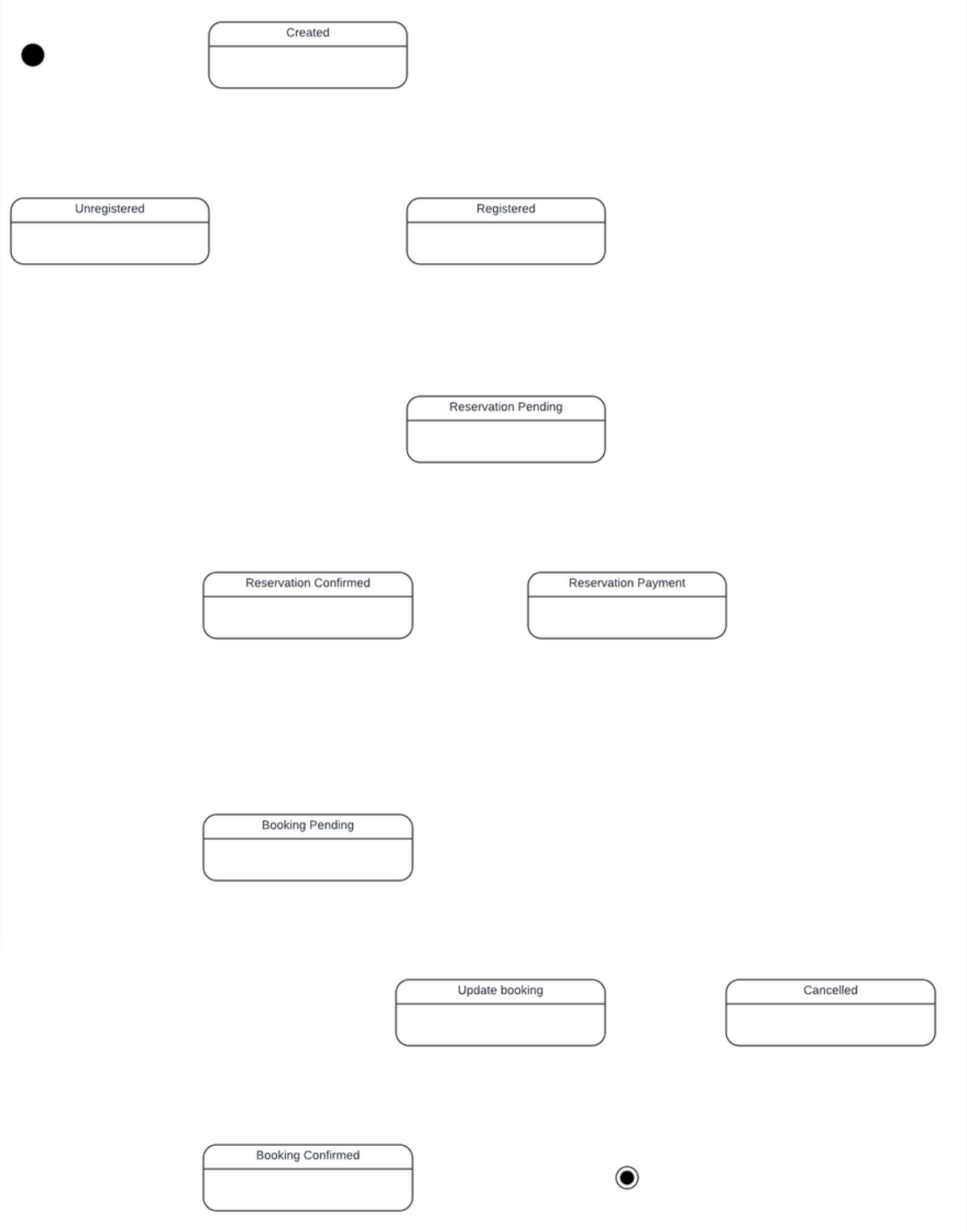


- Create a State element for a new state

8. STATE MACHINE DIAGRAM FOR AN OBJECT OF THE ACCOUNT CLASS

STATE MACHINE DIAGRAM DOCUMENTATION

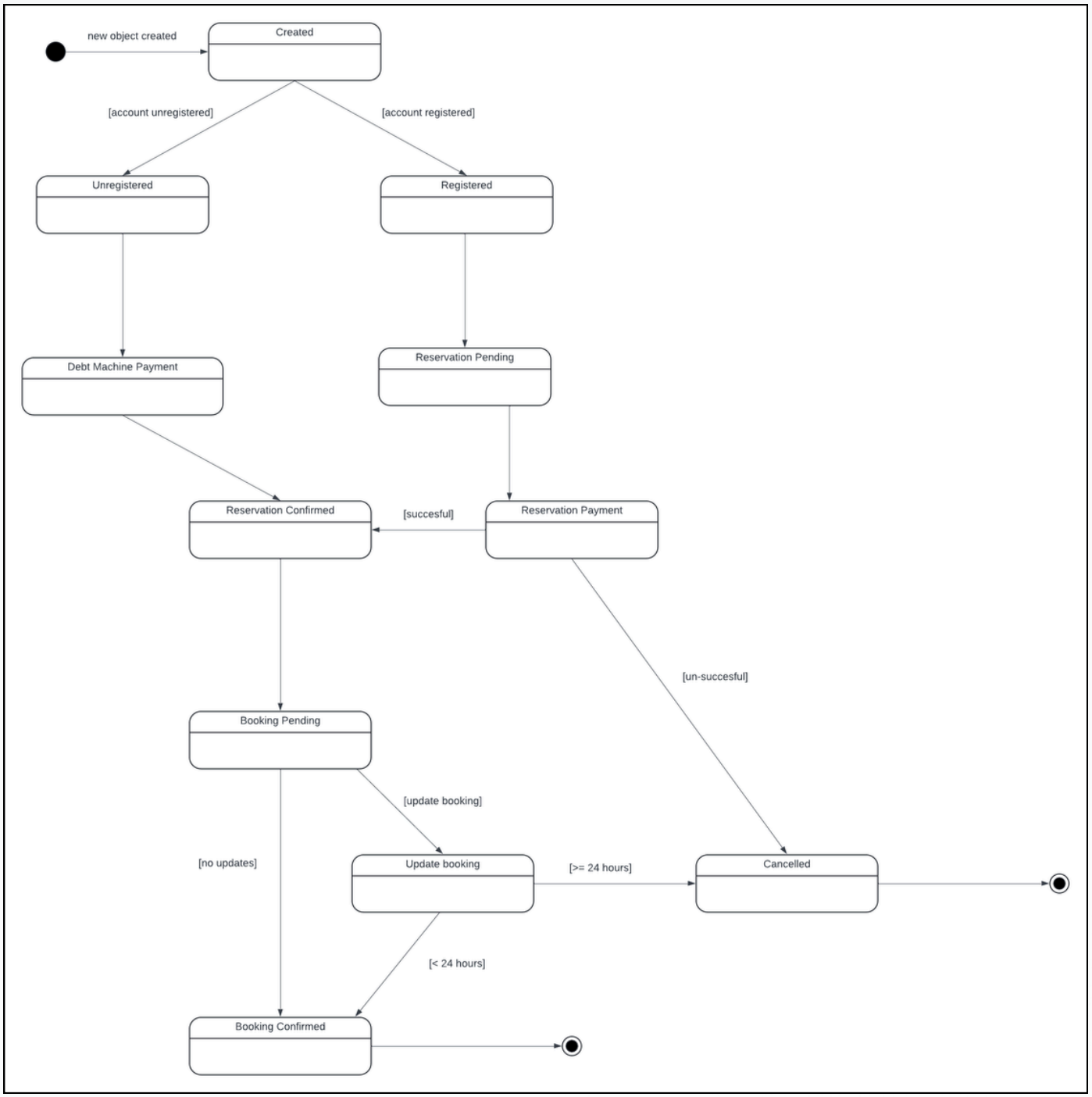
Showing States



- Add a New Initial and Final State elements.

8. STATE MACHINE DIAGRAM FOR AN OBJECT OF THE ACCOUNT CLASS

STATE MACHINE DIAGRAM DOCUMENTATION



PLEASE SEE FINAL STATE MACHINE DIAGRAM ATTACHED

9. SOFTWARE DEVELOPMENT METHODOLOGIES

1.AGILE SOFTWARE DEVELOPMENT METHODOLOGY

Agile Software Development methodology is one of the best software development approaches that is used to design a disciplined software management process which also allows some frequent alteration in the development project. This is a type of software development methodology that is one conceptual framework for undertaking various software engineering projects. Agile Development is used to minimize risk by developing software in short time boxes which are called iterations that generally last for one week to one month.

2.DEVOPS METHODOLOGY

Among all the software development approaches, DevOps is a well-known phrase that is receiving a lot of attention because to the unwavering advantages it provides to its clients. The beginning of DevOps is not the same as the isolated processes of Development and Operations. For the duration of the full life-cycle, these two departments work as a unified team. This operates simultaneously for every firm. Development and operational teams may work on security, quality assurance, and other activities simultaneously thanks to the continuous integration and delivery approach.

3.RAPID APPLICATION DEVELOPMENT (RAD)

Outperforming other software development approaches in terms of development speed and quality, Rapid Application Development (RAD) is a highly efficient methodology. Its architecture makes it simple to utilize the full potential of software development. Accelerating the overall software development process is the primary goal of the rapid application development technique. Because it permits active user input in the development process, the aim is easily attainable.

4.SCRUM

We employ this kind of development approach for businesses where the needs are constantly changing and quick adjustments are simple to implement. Scrum software development starts with a quick planning session, moves into a meeting, and ends with a final review. Businesses may use this strategy, which enables several iterations in one go, to expedite software development. It is among the greatest approaches for software development since it can quickly get even the slowest-moving projects back on track.

9/10. OUR SOLUTION

5.FEATURE DRIVEN DEVELOPMENT (FDD)

Of all the software techniques, feature-driven development is the most iterative and is designed for usage by big teams working with object-oriented technologies on a project. Organizations making the switch from a phase-based to an iterative strategy might benefit from this kind of model. An FDD approach is another name for a feature-driven methodology.

RAPID APPLICATION DEVELOPMENT (RAD)

WHY RAD FOR EDUVOS?

- User-Centric
- Prototyping
- Timeboxing
- Flexibility

- The Rapid Application Development Model assists in lowering the risk and effort requirements for software developers.
- This architecture facilitates timely project assessments for the clients.
- Customer input is encouraged by this process, which always gives room for improvement in software development projects.
- There's a chance that natural prototyping leads to fewer faults.
- Every RAD step provides the client with the functionality that is most important.

REFERENCES

- Software Engineering - (2020-2021) (Textbook)
- https://www.tatvasoft.com/blog/top-12-software-development-methodologies-and-its-advantages-disadvantages/
- https://www.uptech.team/blog/software-development-methodologies
- https://www.indeed.com/career-advice/career-development/software-development-methodologies