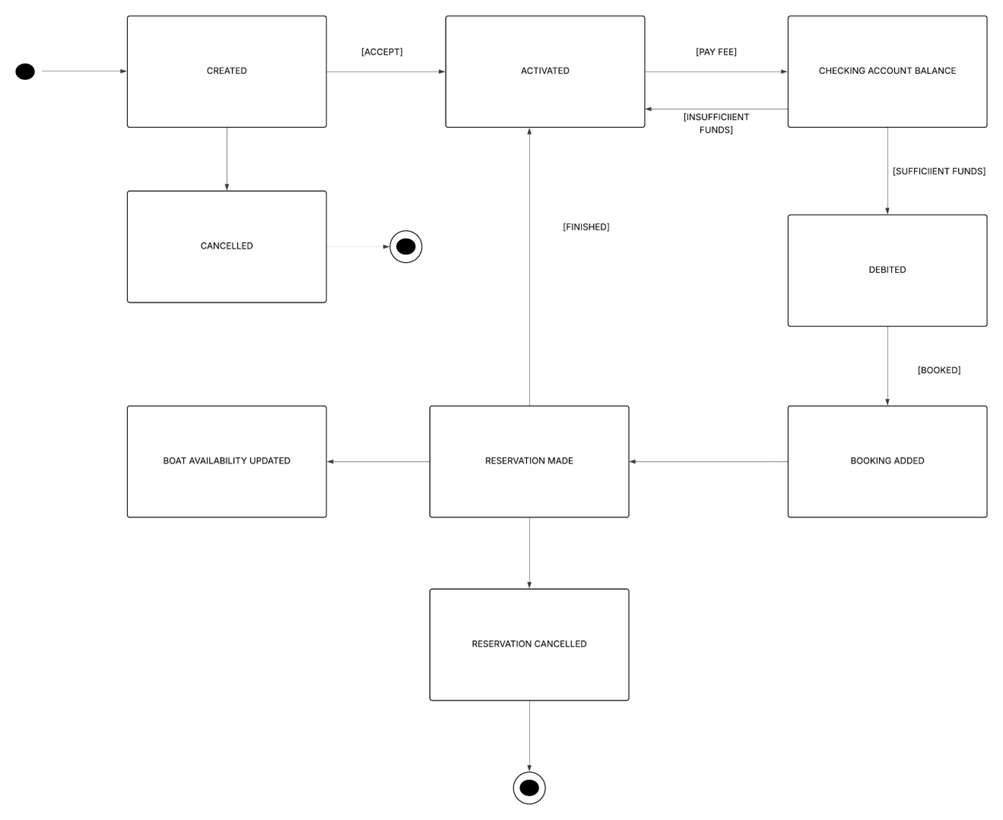
**Assignment 8**

**STUDENT: THABANG MOTAMPANE**

**STUDENT NUMBER: 251699544**

1. State Transition Diagram



State transition diagram documentation:

The above state transition diagram highlights the account registration, payment process and booking process for the boat rental system. When an account is created for a customer, the customer has the option of cancelling or accepting the account. If the account is accepted, it is first activated. The customer then makes a payment fee which causes the account object to enter a check balance state. This checks that sufficient funds are available. If sufficient funds are available, the account is debited. After debiting the account, a booking and reservation are made. If there are insufficient funds are to complete the payment, the account returns to the activated state. A reservation may also be cancelled, therefore ending the process.

2. Activity Diagram

A diagram of a process flow

AI-generated content may be incorrect.

Activity diagram documentation:

The activity diagram above highlights how all the actors will interact will the system. The customer first logs on to the system. The customer then confirms the booking and makes the payment. The system administrator receives the payment confirmation and approves the booking accordingly. The sales assistant then creates the booking into the system and reserves the boat for the customer. The customer’s booking details are then stored on the system and emailed to the client. The system then updates the boat’s availability. When the client returns the boat, the service assistant takes the boat in for maintenance and manages the boat inventory. The owner can generate reports on the rental history of the boats.

3. Integration with Prior Work:

The above diagrams align with the system’s functional requirements and user stories.

* Users (customers and staff) can register, log in, and manage their profiles.
* Customers can rent boats and staff can manage boats/bookings.
* Users can browse/filter boats by type, size, location, availability, etc.
* Users can select a boat, choose a rental period, and confirm the booking.
* Users can view, update, or cancel their bookings within allowed policies.
* The staff/admin will be able to approve paid bookings.
* Staff/Admin can add or update boat info (type, size, capacity, location, pricing, images)
* The system can update boat availability to prevent double-bookings and manage boat schedules.
* Users can pay for bookings using credit/debit cards or online wallets.
* The system can generate receipts and send confirmations via email/SMS.
* The owner can generate reports on boat rental history and payments.

4. Reflection:

Creating activity diagrams and state transition diagrams is not an easy task, especially if your functional requirements are not clear. If the functional requirements aren’t clear, it’s hard to map out activities and workflows accurately. You might not know what decisions or branches are needed. Trying to include everything can also make the diagram unreadable, so the balancing of activities can also be very tricky.

Overusing decision nodes or swim-lanes can create visual clutter. Not clearly separating responsibilities between actors/systems can also obscure who does what.

When it comes to state transition diagrams, its very important to keep the flow consistent and understandable. Missing intermediate or error states (like "Pending Payment" or "Cancelled") can cause gaps in understanding system behaviour. Some transitions depend on multiple conditions/events. Mapping this cleanly can be tricky. If multiple states are active in parallel (e.g., a boat can be in “Cleaning” and “Unavailable”), visualizing that in a simple state diagram can also be tough.

If stories are not detailed enough, you won’t know what states or activities to represent. You might skip certain transitions or activities just because they weren’t captured. Several user stories can touch the same state or activity (e.g., "Booking Cancelled" could result from multiple user stories) Ensuring consistency and avoiding redundancy across diagrams becomes a challenge.