Bluestaq Online Assessment: Project Outline

Victor Gong

Brainstorming

Elevator

- Start with MVP - carry single person to whatever floor
    - Expand to guest generation, optimization, control system
- Draw Finite State Machine for elevator operation or flow chart
- During elevator's ascent or descent, if guest presses button on floor in same direction of travel, stop to pick up guest; otherwise add to queue
- Customizable number of floors; N < 10 has 1 elevator running, > 10 has 2 (one up one down), in general, we add one extra elevator every time the number of floors double
- Time delay for door open, door close
- Max capacity per carriage
- Users can place guests and make requests from any floor
- Testing software - randomly spawn 10,000 batches of 10,000 guests to assess efficiency of multi-carriage system

General Workflow:

- Open → Fill Guests → Close → Transit → Empty Guests

Includes both user inputs and virtual guests that take elevator from random floors

Plan

MVP: Single-Carriage Elevator

Motivation: Develop a simple system that contains the fundamental elements of an elevator
Goal: Create an instant-moving elevator that the user can interact with via console

Elevator States:

- Current Floor
- Direction: Up | Stationary | Down (enum: 1, 0, -1)
- Pick-up requests (sorted set)
- Drop-off requests (sorted set)

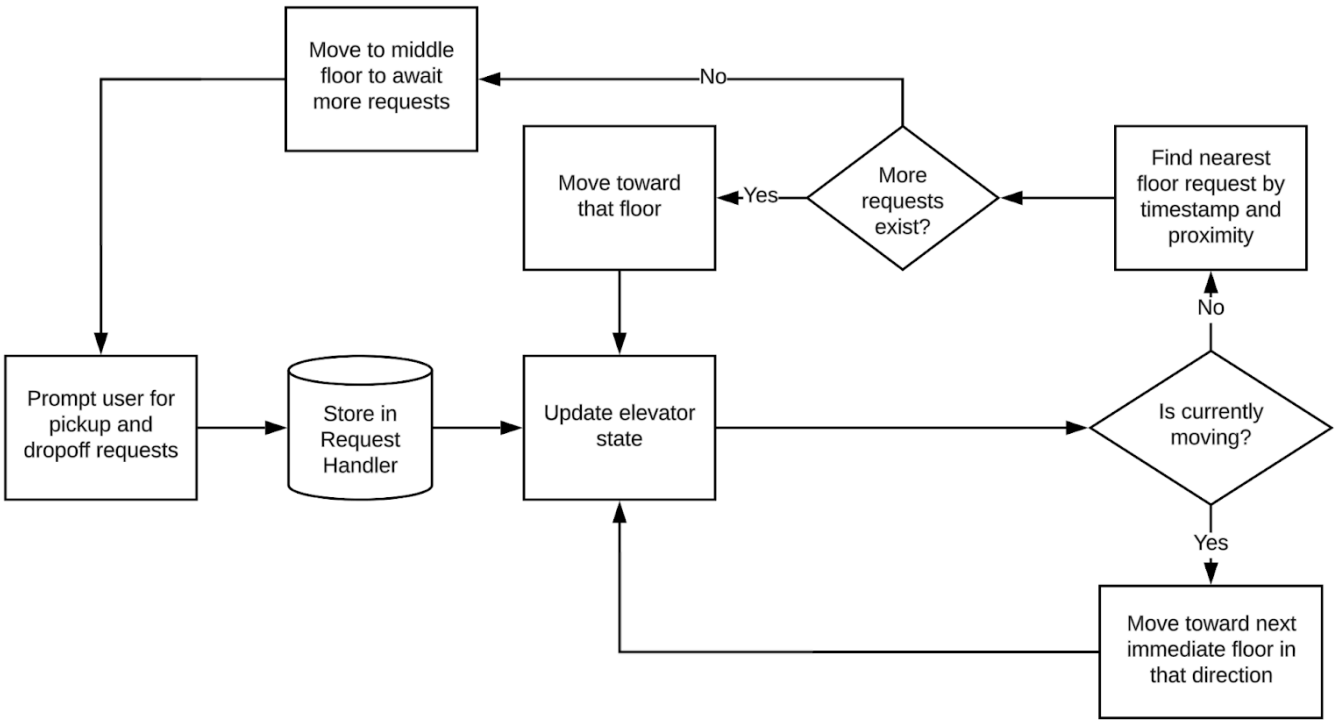
User Actions:

- Since only one elevator, all requests go to one tracking system for now
- Calls elevator on floor (from floor, outside elevator)
    - Add to pick-up requests
- Enters which floor(s) they want to go to (from inside elevator)

    - Add to drop-off requests
- User can input more requests via console every time elevator completes all previous requests and reaches a steady state

Decision Making:

- Update decisions after every state change (e.g. added/completed request)
- If moving up / down
    - Continue moving, completing pick-up and drop-off requests until no more in that direction
    - Reverse transit direction to sweep other side
    - If finished all requests, return to middle floor (N/2)
- If stationary
    - Address request with earliest timestamp
    - Do nothing if no requests
- Timestamps represented by discrete actions (e.g. adding a new pick-up request takes 1 unit of time)



Multi-Carriage System

Motivation: Make elevator system more efficient and accessible for tall buildings
Goal: Expand single-carriage elevator into multiple carriages controlled by centralized dispatcher

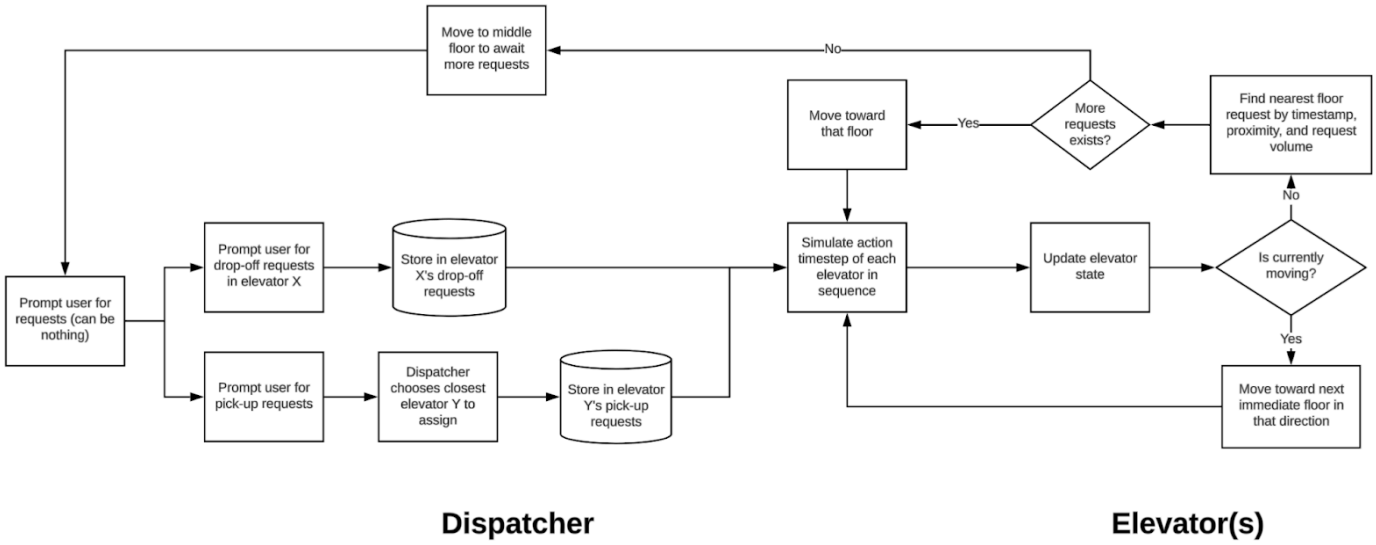
Changes

- Expand to 2+ elevators operating at once, based on floor number (1 + log2(floors / 10))
- Central dispatcher system that assigns elevator to guests on floor when request made

Methods

- Dispatcher handles user prompting
    - Iterates through each elevator, calling update() to step to next action
        - update() returns whether or not the elevator's state changed
        - If elevator settles into steady state, we prompt the user for further requests

- Dispatcher takes all user requests and distributes tasks to each elevator carriage based on proximity
    - Drop-off request: pressed from within specific elevator → forced
    - Pick-up request: pressed from floor, greedy strategy to find first elevator
        - Considers distance, transit direction, and number of requests
        - Takes closest elevator traveling towards floor, with minimum # of requests
    - **Two prompt methods in dispatcher: promptDropoff(elevator) and promptPickup(), handles separately



Virtual Guests & Carrying Capacity

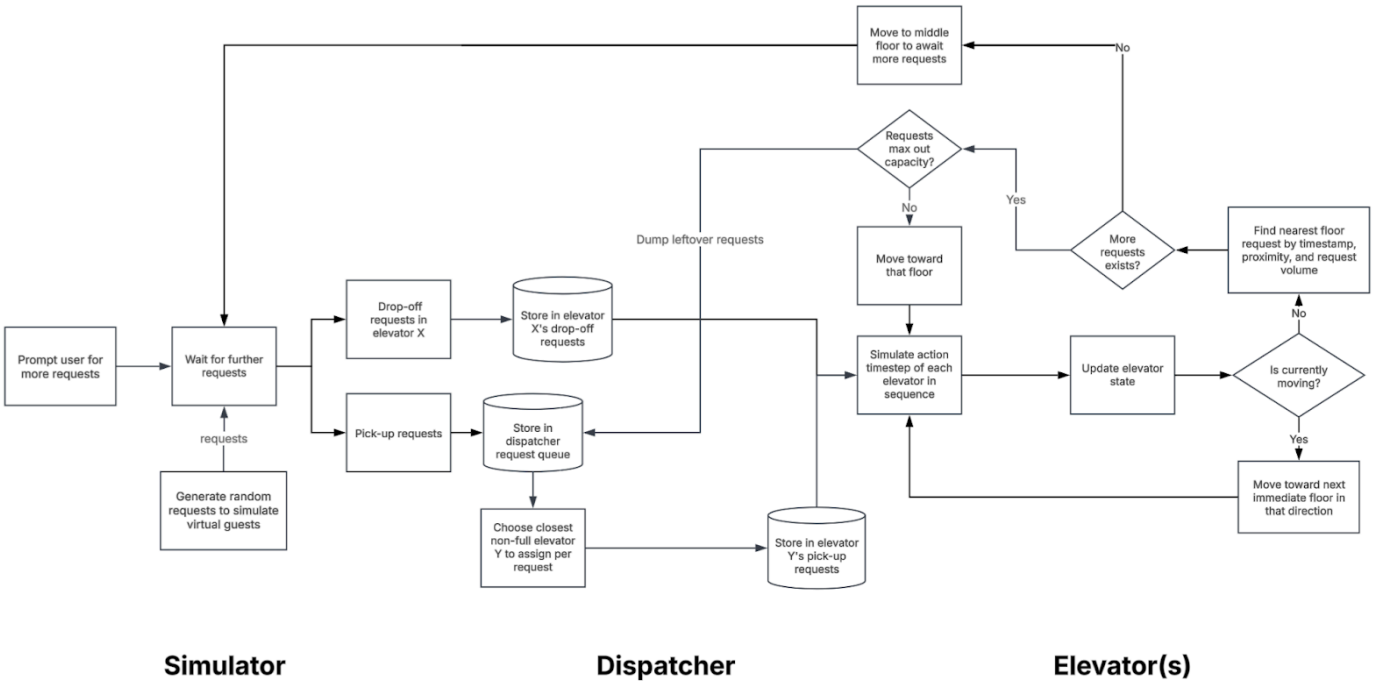
Motivation: Develop way to stress-test efficiency of elevator system and enrich simulation
Goal: Add virtual guests that make requests to the elevator, instead of manual user input. Also introduce carrying capacity where capping number of people an elevator can hold at once

Carrying Capacity

- Add carrying capacity to elevators → full elevators cannot take more requests
- All requests now go into dispatcher queue, then addressed one by one
- Additional request parameter: numGuests
    - Update carrying capacity of elevator
        - Completed pick-up, add numGuests
        - Completed drop-off, subtract numGuests
    - Elevator is smart: if taking on pick-up request overflows the capacity, it dumps the leftover guests back into Dispatcher queue
        - E.g. pick-up 5 guests, drop-off 3 guests both at floor 2, but elevator capacity = 9 / 10. Dumps partial pick-up request of 1 guest at floor 2 back into Dispatcher queue
- Bidirectional communication between elevators and dispatcher

Virtual guests

- Automatically generate pick-up and drop-off requests across different floors for testing
    - Random requests at floors (with random but reasonable # of guests)
- User can still make requests, toggleable in the settings
    - Checks that total drop-off guests doesn't exceed actual elevator capacity



Future Features

- Smarter scheduling algorithm for elevator
  - Make elevator look into the future
    - If pick-up request will exceed capacity, don't even attempt to go to that floor
    - Directly dump request back to dispatcher, continue on without that baggage
  - Dispatcher avoids overflow entirely through capacity planning
    - Have a reserved capacity variable for every elevator, such that the dispatcher won't assign fully-reserved elevators more requests
- Multithreading for concurrent elevator updates within Dispatcher
- Visual Interface & Live Continuous Simulation
  - Use timestamps for something other than the elevator's decision-making processing
  - Visualize process as UI instead of text on console
- Map passenger flow through different floors and test elevator efficiency with guest traffic
- User dashboard for making pickup and dropoff requests through buttons
- Integrate this system into a physical elevator model and see how it performs! 🙂

GitHub repository available here: https://github.com/ViceBitz/BluestaqOA2025
Original Google Doc: Bluestaq OA

<Flowcharts made with Lucidcharts>