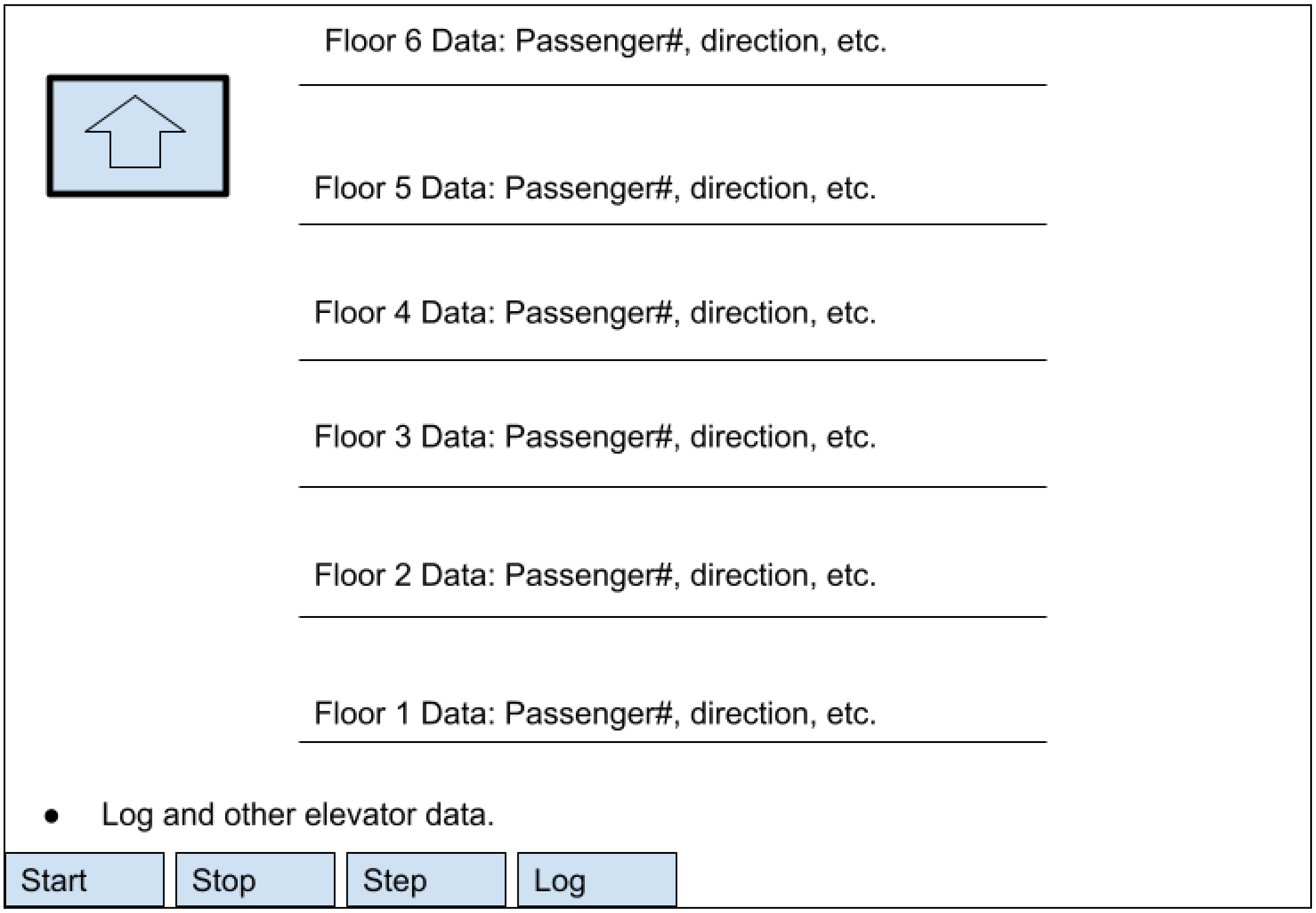
Elevator Simulation Design Document, v2

Team Name: holyC
Team Members: Auraud Zarafshar, Maddox Krape

1. **GUI Mockup**

   This can be hand-drawn or designed in slides or drawIO - but paste a picture here. It should be clear how you will convey all of the required information and controls to the users.


Floor 6 Data: Passenger#, direction, etc.
Floor 5 Data: Passenger#, direction, etc.
Floor 4 Data: Passenger#, direction, etc.
Floor 3 Data: Passenger#, direction, etc.
Floor 2 Data: Passenger#, direction, etc.
Floor 1 Data: Passenger#, direction, etc.
Log and other elevator data.
Start
Stop
Step
Log

2. **Identify the ownership of each class:**

   Maddox classes: Building, Floor, Passengers, Elevator

   Auraud classes: GUI, CallMgr, SimController

3. **Identify (first pass) all of the externally visible methods (public and protected) *other than Getters/Setters* that you think you will need for each class.**

   This should be done as one section for each class and should look like this:

   Class: Building
   // Building() Constructor for building; initiates all variables
   Public Building(int numFloors, int numElevators,String logfile);
   // configElevators()
   // addPassengersToQueue()
   //checkPassengerQueue()
   //updateElevator()

   Class: Floor
   // Floor() Constructor
   // public Floor(int qSize)
   //getter/setters

   Class: Passengers
   // Passengers() constructor
   // add/remove passenger(direction)
   //getters/setters

   // high-level comment about what methodOne does (1 sentence)
   <visibility> <type> methodOne(<type> param1, <type> param2, …)
   // high-level comment about what methodOne does (1 sentence)
   <visibility> <type> methodTwo(<type> param1, <type> param2, …)

   Class: CallMgr:
     public CallManager(Floor[] floors, int numFloors);
     Passengers prioritizePassengerCalls(int floor)
   // Getters and Setters

   Class: Controller
   stepSim() void

4. **Project Management, Work Flow, Conflict Resolution, and Milestones**

   We will take advantage of the shared repository focusing on our dedicated classes. We will communicate regularly in and out of class to clarify and problem-solve.
   If there's a problem, we will communicate, and if stuck/can't resolve it, talk to the teacher.

Conflict Resolution:

- Try to only work on our own specified classes, as multiple people working on the same class cause merge errors.

Milestones:

- **Design Document v1: Nov 14, 2022**
- **All getters and setters**
- **Logging**
- **GUI**
- **On track for completion**