# ParallelElevator

This project is an simulation of a multithreaded elevator operating system. It is written in c++11 using std::thread, mutex, and condition_variable to provide synchronization between threads.

The project is build to work with Dr. Eric Rees' elevator API that can be used to ping new information about the state of the building, the next input, etc

# Algorithm

When a person is recieved from the next input API the are sent to the scheduler thread which processes the data in the input queue and pushes it to the output queue. The way that the system decides which elevator to assign a person to is found in the function calculateScore() and uses the following algorithm

Score = distance_penalty + start_direction_penalty + end_direction_penalty + passenger_penalty

Where: distance_penalty = the distance from the elevator's current floor to the user's start floor

start_direction_penalty = if the elevator is not heading toward the person we add 1/2 its max range

end_direction_penalty = if the elevator is not heading toward the person's destination add 1/2 its max range

passenger_penalty = a score of +5 per person on the elevator, this represents load/unloading time for each person and helps to load balance the elevators.

# Contributors

This project was created by Josh Josey and Jaden Hicks.

Jaden handled the building.h and building.cpp creation

Josh handled the api_control.h and api_control.cpp creation

The code in main.cpp was written collaboratively as we met in person to complete that section before going off on our own to test and refine the project.