**Important APIs**

**Logic**

bool executeUserInput(string input): Called by UI with the exact string entered by user. Creates Parser object to determine the correct action to take, then calls another Logic method executeCommand to actually execute the action. It returns true by default, and will be able to return false when future error cases are implemented.

bool Logic::executeUserInput(string input) {

parserPtr = new Parser(input);

Parser::commandType command = getCommand();

Event userEvent = getEvent();

executeCommand(command, userEvent);

deleteParserPtr();

return true;

}

void executeCommand(Parser::commandType command, Event userEvent): Calls APIs from EventStorage to execute the desired command based on the input parameters (e.g. if command = ADDFLOAT, it will call EventStorage to add the floating event denoted by userEvent). Since there are many possible commands, a switch case is implemented. Only relevant cases are shown below.

void Logic::executeCommand(Parser::commandType command, Event userEvent) {

string eventName = parserPtr->getNameOfEvent();//can be index in integer form (e.g. "1") or actual name of event (e.g. "lunch")

int index, id;

vector<Event> tempEvents;

switch (command) {

case Parser::ADDFLOAT: {

display.setFloatingEvents(eventStore.addEvent(userEvent));

display.setFeedbackStrings(userEvent.getName() + ADDED\_MESSAGE);

break;

}

case Parser::DELETE\_: {

if (isNumber(eventName)) {

index = std::stoi(eventName);

id = display.getID(index);

eventName = display.getEventName(index);

} else {

id = INVALID\_NUMBER;

}

tempEvents = eventStore.deleteEvent(id, eventName);

bool isFloat = tempEvents[0].getIsFloating();

if (isFloat) {

display.setFloatingEvents(tempEvents);

} else {

display.setNormalEvents(tempEvents);

}

display.setFeedbackStrings(eventName + DELETED\_MESSAGE);

break;

}

case Parser::EDIT: {

Event tempEvent = parserPtr->getEvent();

if (isNumber(eventName)) {

index = std::stoi(eventName);

id = display.getID(index);

eventName = display.getEventName(index);

} else {

id = INVALID\_NUMBER;

}

tempEvents = eventStore.editEvent(id, eventName, tempEvent);

bool isFloat = tempEvents[0].getIsFloating();

if (isFloat) {

display.setFloatingEvents(tempEvents);

} else {

display.setNormalEvents(tempEvents);

}

display.setFeedbackStrings(userEvent.getName() + EDITED\_MESSAGE);

break;

}

default:

break;

}

}

**Parser**

**EventStorage**