**Important APIs**

**Logic**

bool Logic::executeUserInput(string input) {

parserPtr = new Parser(input);

Parser::commandType command = getCommand();

Event userEvent = getEvent();

executeCommand(command, userEvent);

deleteParserPtr();

return true;

}

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::ADDFULLDAY:

break;

case Parser::ADD:

break;

case Parser::ADDSTART:

break;

case Parser::ADDMULFULLDAYS:

break;

case Parser::ADDMULDAYS:

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;

}

case Parser::SHOWDAY:

break;

case Parser::SHOWMTH:

break;

case Parser::SHOWYR:

break;

case Parser::SHOWUSER:

break;

case Parser::ERROR\_:

break;

default:

break;

}

}