GradRooAte

Architecture/Design Document

**Table of Contents**

[**1 INTRODUCTION**](#_jhavr81lc2xh) **3**

[**2 DESIGN GOALS and CONSTRAINTS**](#_4ck5t9vgwybd) **3**

[**3 SYSTEM BEHAVIOR**](#_onke6gruhvc5) **4**

[**4 LOGICAL VIEW**](#_5le2v42yahop) **4**

[4.1 High-Level Design(Architecture)](#_usuo4y2vmb9b) 4

[4.2 Mid-Level Design](#_h732e8uptc3d) 4

[4.3 Detailed Class Design](#_1x9ud6qkg3m1) 4

[**5 PROCESS VIEW**](#_c3wo1gina9yi) **4**

[**6 DEVELOPMENT VIEW**](#_13264nokevwc) **4**

[**7 PROTOTYPES**](#_hfadi8h008pr) **5**

[**8 USE CASE VIEW**](#_yhsywet54vkr) **8**

Change History

**Version:** 0.1

**Modifier:** <Kendal Sutton>

**Date:** 3/29/2020

**Description of Change:** Editing document/Rough Draft

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Change History

**Version:** 0.1

**Modifier:** David Leatherwood

**Date:** 4/5/2020

**Description of Change:** Editing document

# **1 INTRODUCTION**

**Architecture and Design**

For this document, it will describe the architecture and design for the Faculty Scheduling application. This is being developed by the team, GradRooAte, for the University of Missouri-Kansas City. This Faculty Scheduling application will help organize faculties class schedules at UMKC.

For further definition, the architecture of the Faculty Scheduling application is described from 4 different perspectives [1995 Krutchen]:

1. Logical – the logical view typically includes class diagrams and behavior diagrams to make clear that the proposed architecture covers the required functionality. Concerned with functionality provided to end users. Think analysis model (abstraction from problem domain).
2. Process – the process view describes concurrent threads and processes.
3. Development – the development view takes into account the development environment. Once you have identified all the needed software elements you can begin to map them to deployable modules, development teams, etc. Those aspects of the architecture that are driven by practical concerns of the software development environment, are addressed by the development view. Architecture from the programmer’s perspective. Think design model (abstraction of actual or anticipated implementation).
4. The Use Case view is the 5th or +1 view. It’s conceptually separate from the other views because it shows how the other views work together to validate the efficacy and completeness of the architecture.

# **2 DESIGN GOALS and CONSTRAINTS**

We will base our design on the stakeholder. The main goal is to create a user-friendly GUI that inputs rules, classes, professors, rooms and other items for a semester scheduling. Constraint and requirements will also be included in the designs. Those are:

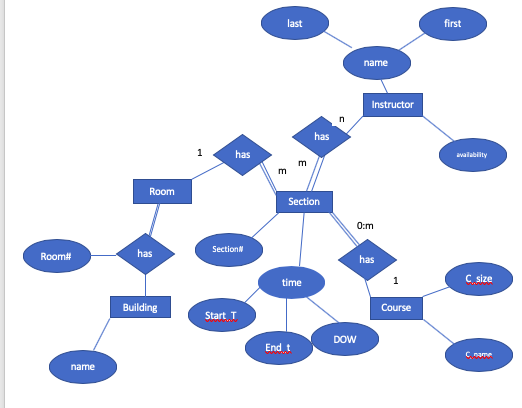
1. Rules must apply to reduce conflict in teachers scheduling. Example, no teacher should have a class at the same time as another teacher in the same room.
2. The client should be able to edit, delete, and update the faculties information. The clients should also be able to retrieve information for the semesters.

# **3 SYSTEM BEHAVIOR**

Structure and behavior of a system defines the system behavior. In section 7, the prototypes show the design of the programs. That determines what the user will view and in further development.

# **4 LOGICAL VIEW**

The logical view is concerned with the functionality that the system provides to end-users. UML diagram is posted below:



## **4.1 High-Level Design(Architecture)**

[TBD]

## **4.2 Mid-Level Design**

[TBD]

## **4.3 Detailed Class Design**

[TBD]

# **5 PROCESS VIEW**

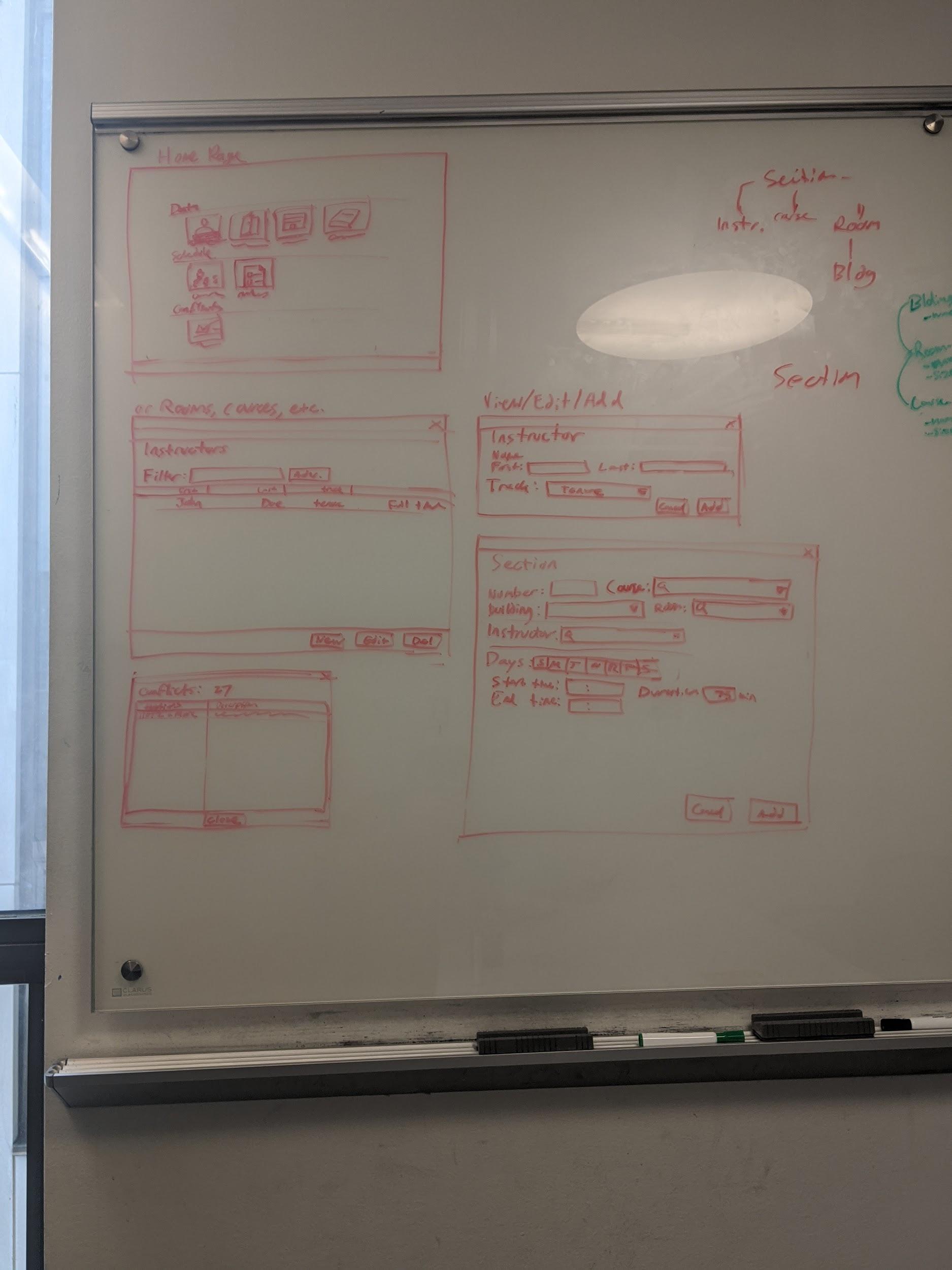
The process view describes concurrent threads and processes. The following threads will be used and all will utilize black-box:

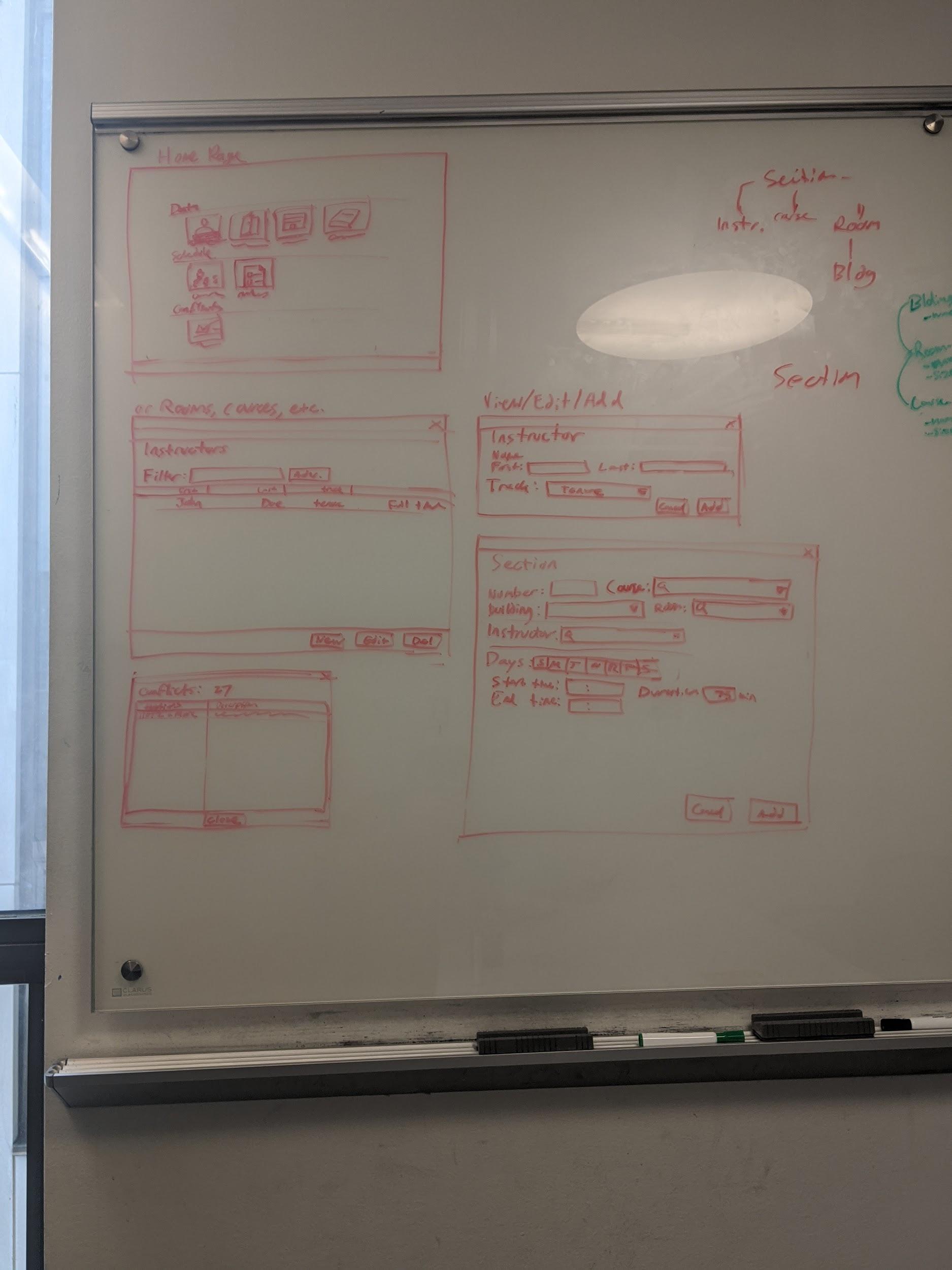
1. UI Thread - will have multiple threads so the UI doesn’t stall
2. Processing Thread - determine and check conflicts
3. I/O Thread - monitors the completion of CPU’s not in use

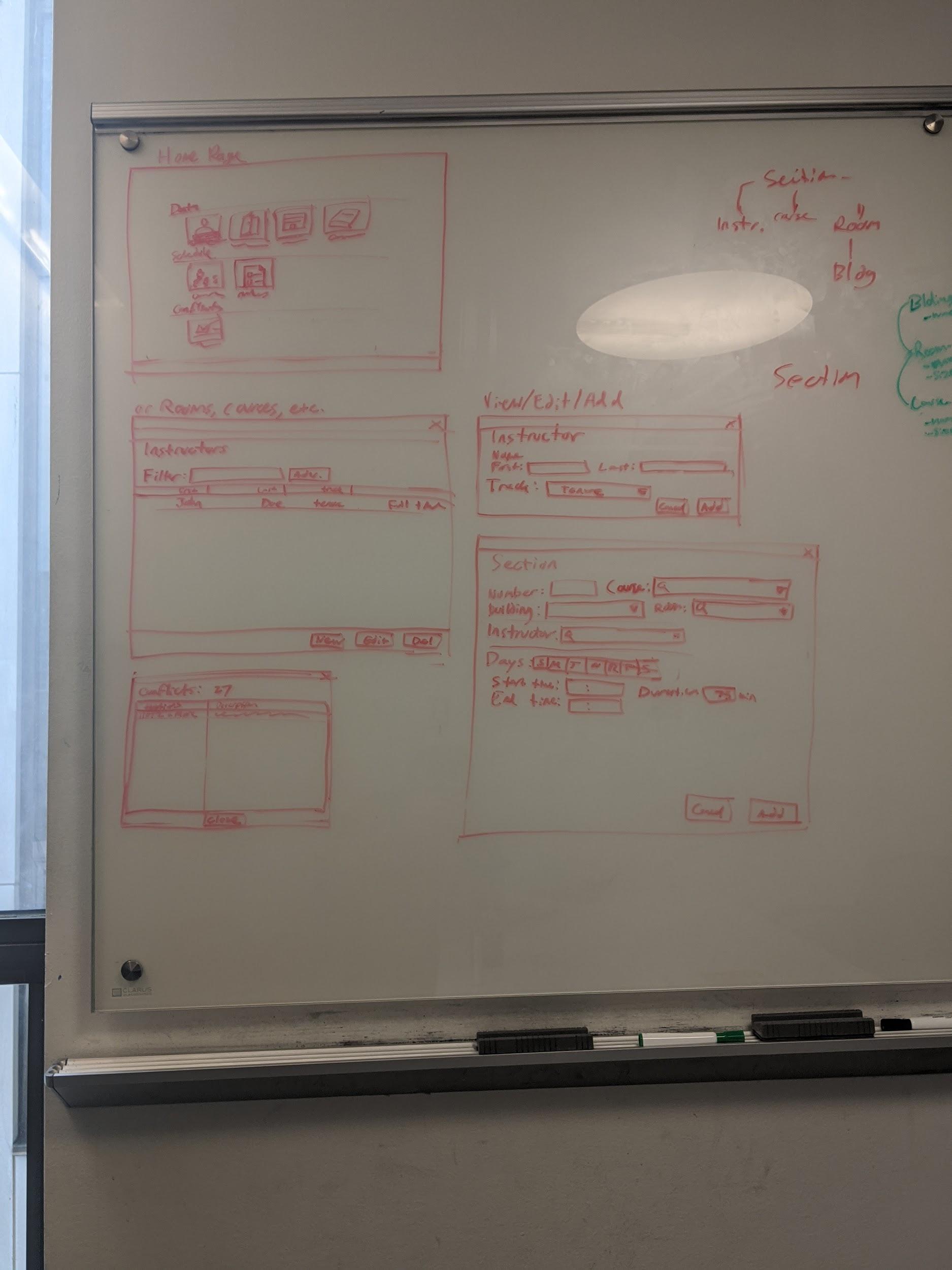
# **6 DEVELOPMENT VIEW**

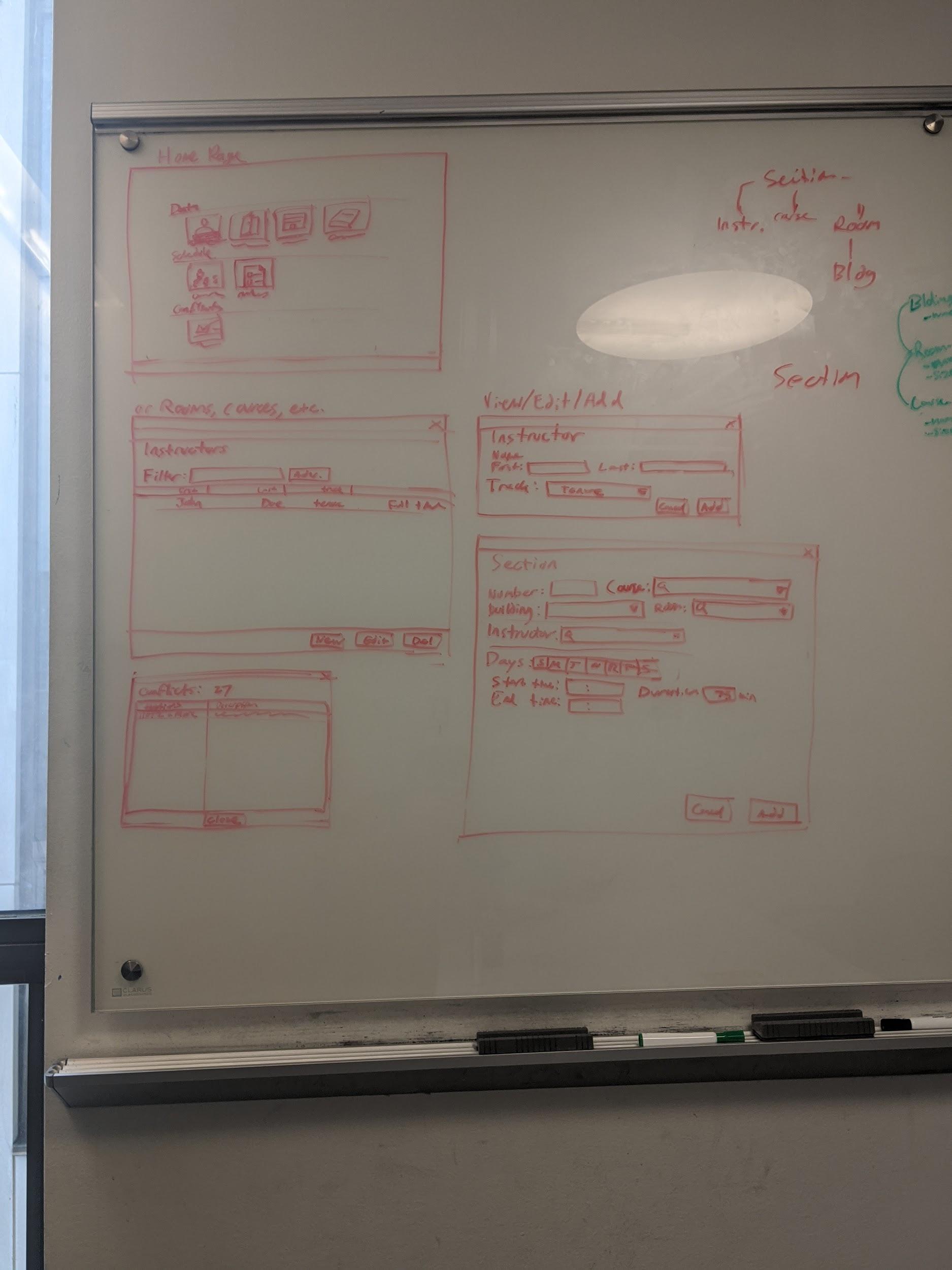
.[TBH]

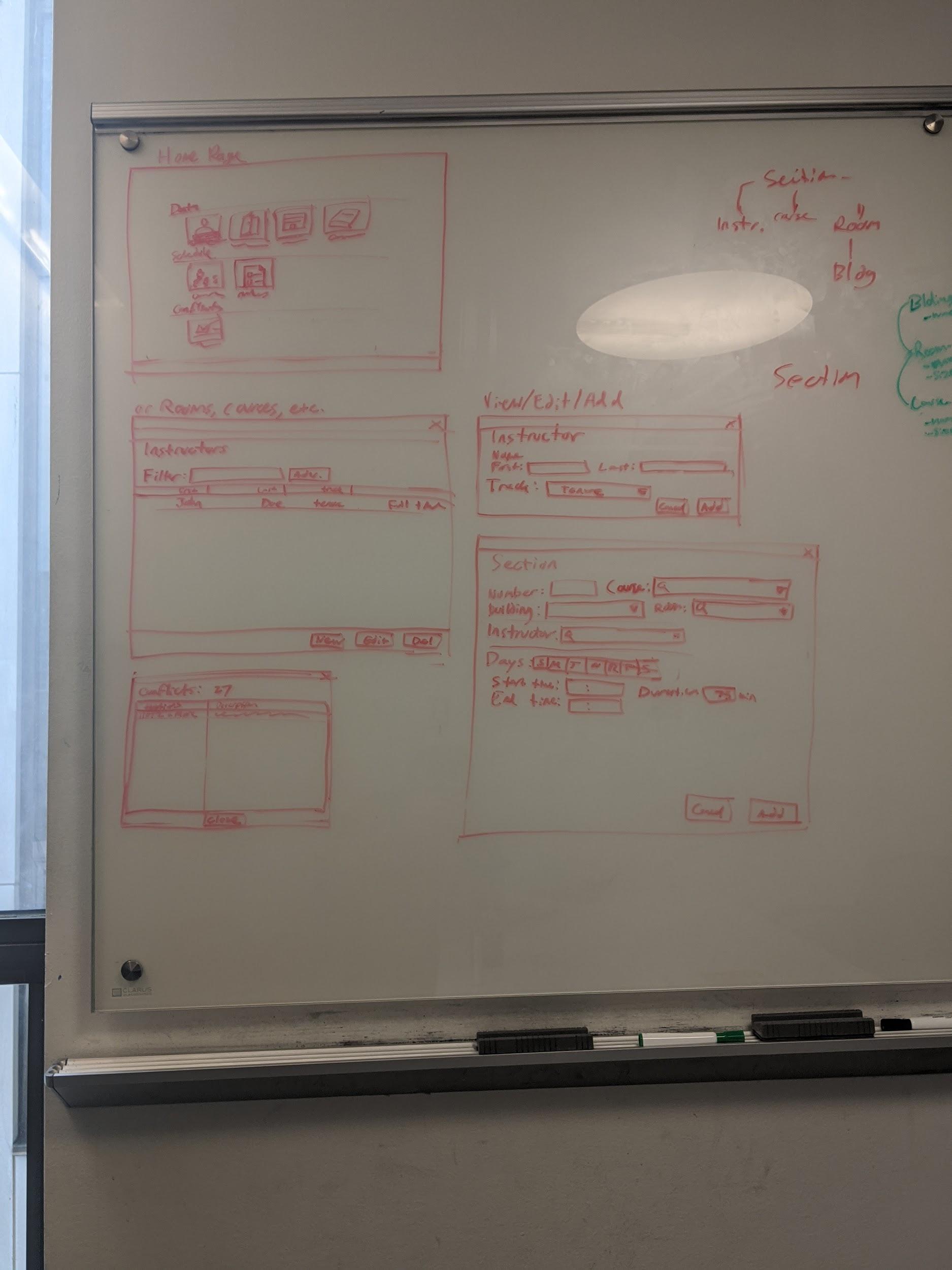
# **7 PROTOTYPES**











# **8 USE CASE VIEW**

Manage rules for schedules -

* Add Rules
* Delete Rules
* Edit Rules

Schedule Checking -

* Input a schedule
* Required information for each item
  + Professor
  + Class Name
  + Building Name
  + Room Number
  + Class/Room Size
  + Etc.
* System will show conflicts in schedule against user programed rules