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

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

**Version:** 0.1

**Modifier:** David Leatherwood

**Date:** 4/5/2020

**Description of Change:** Editing document

**Version:** 0.2

**Modifier:** Alec Sears

**Date:** 4/5/2020

**Description of Change:** Revised the introduction. Revised Process View.

# **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 using Kruchten’s 4+1 View Model of Architecture[[1]](#footnote-0)

1. **Logical** – Described in detail in Logical View; this view contains class diagrams and behavior diagrams to demonstrate how the proposed architecture covers the required functionality provided to end users.
2. **Process** – Described in Process View; explains the concurrent threads that make up the program.
3. **Physical** – This view describes the mapping of hardware to software. In our case, this mapping is out of our control but the tools and frameworks we use restrict compatibility to Microsoft Windows systems and the class of hardware it supports.
4. **Development** – Described in Development View; specifies the supporting architecture for the product, typically in terms of components and packages. This view is concerned with the abstract design of the architecture from the developer’s perspective, accounting for the given development tools and resources.
5. **Use Case** – Described in Use Case View; this is the +1 view. This is the practical use case expressed as an architectural view to align each other view with consumer expectations.

# **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:

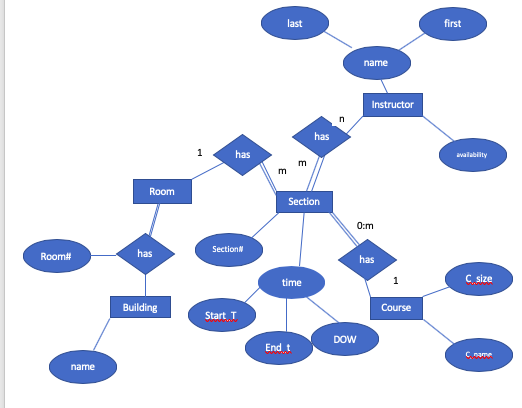
1. Rules must apply to reduce conflict in teachers scheduling. Example, no teacher should have a class at the same time as another teacher in the same room.
2. The client should be able to edit, delete, and update the faculties information. The clients should also be able to retrieve information for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

## 5.1 UI Threads

The UI threads run in a black box to us. Our UI framework, WPF, manages two threads that we know of. A rendering thread, and an I/O thread. These work in tandem (and isolation) to prevent the interface from blocking, but further work is needed on our part to prevent background tasks from blocking. Additionally, each independent window we create would need its own set of threads, still managed by WPF but must be explicitly provisioned.

## 5.2 Data Lookup Thread

Certain input fields in the application will provide a real-time lookup capability. In order to facilitate simultaneous user input and data search, we will create a thread on demand.

## 5.3 Rule Processing Thread

When the user calls for a conflict check, we will need to handle this in a background thread. The work will take too long to force the user to wait on. We will process the conflict rules in the background on a new thread, and report progress to the UI.

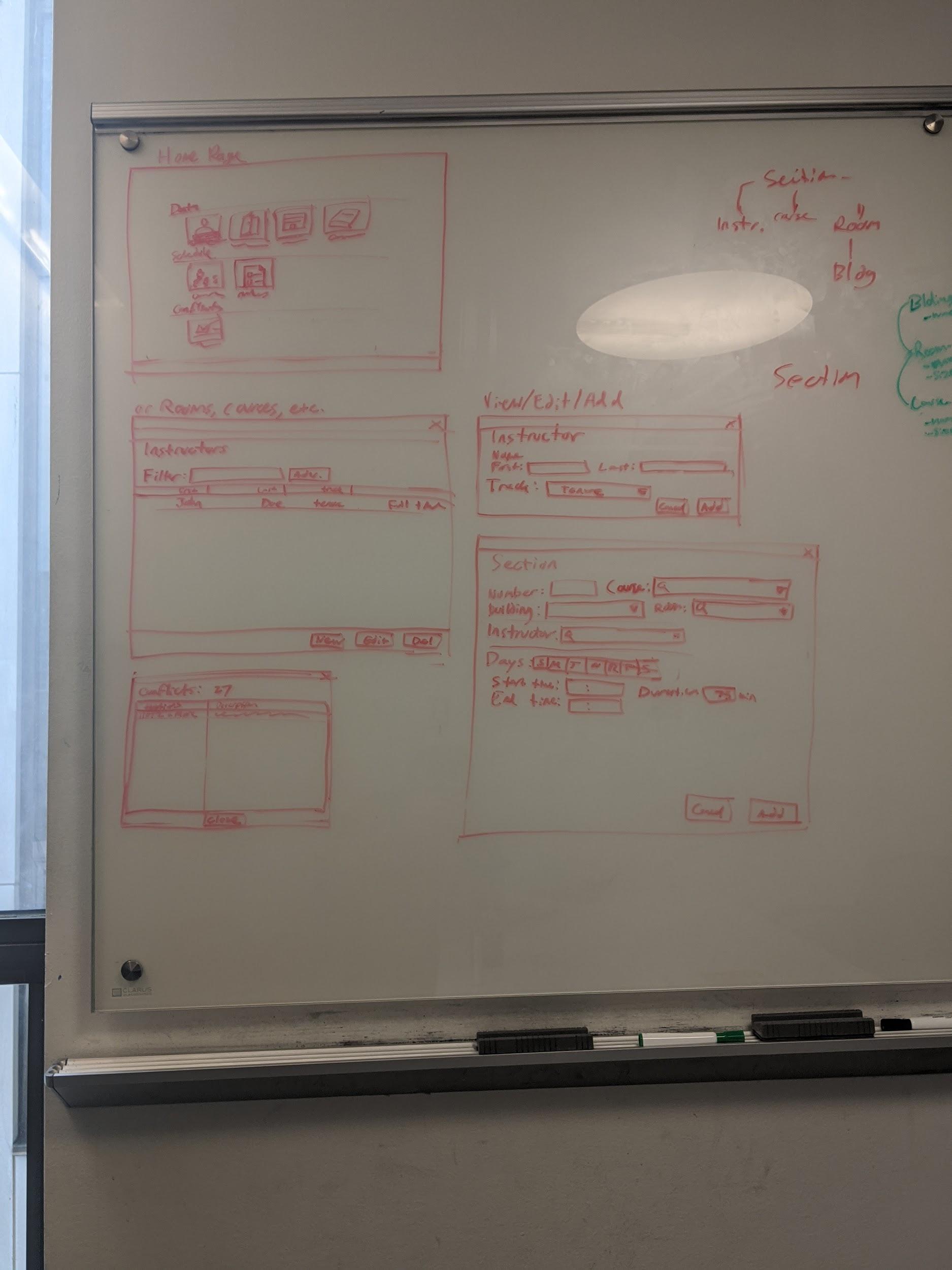
## 5.4 Database Thread

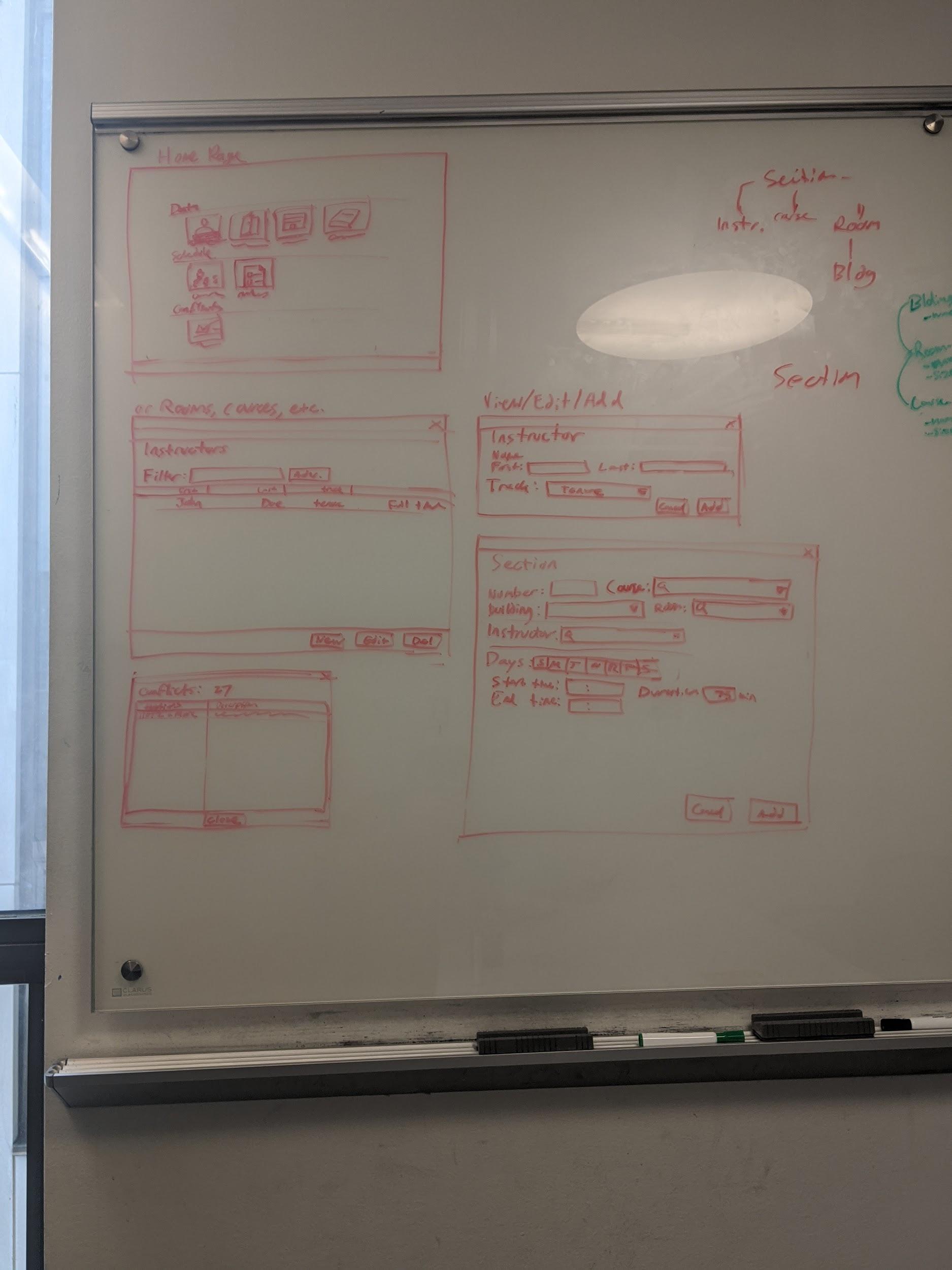
Database access should not block the UI. There will be one thread to handle the connection to a database. Multi-threaded database access is tricky and unnecessary for a single user application, so one thread will be created that is dedicated to handling database actions. Database frameworks may provide built in threading. Exact details TBD.

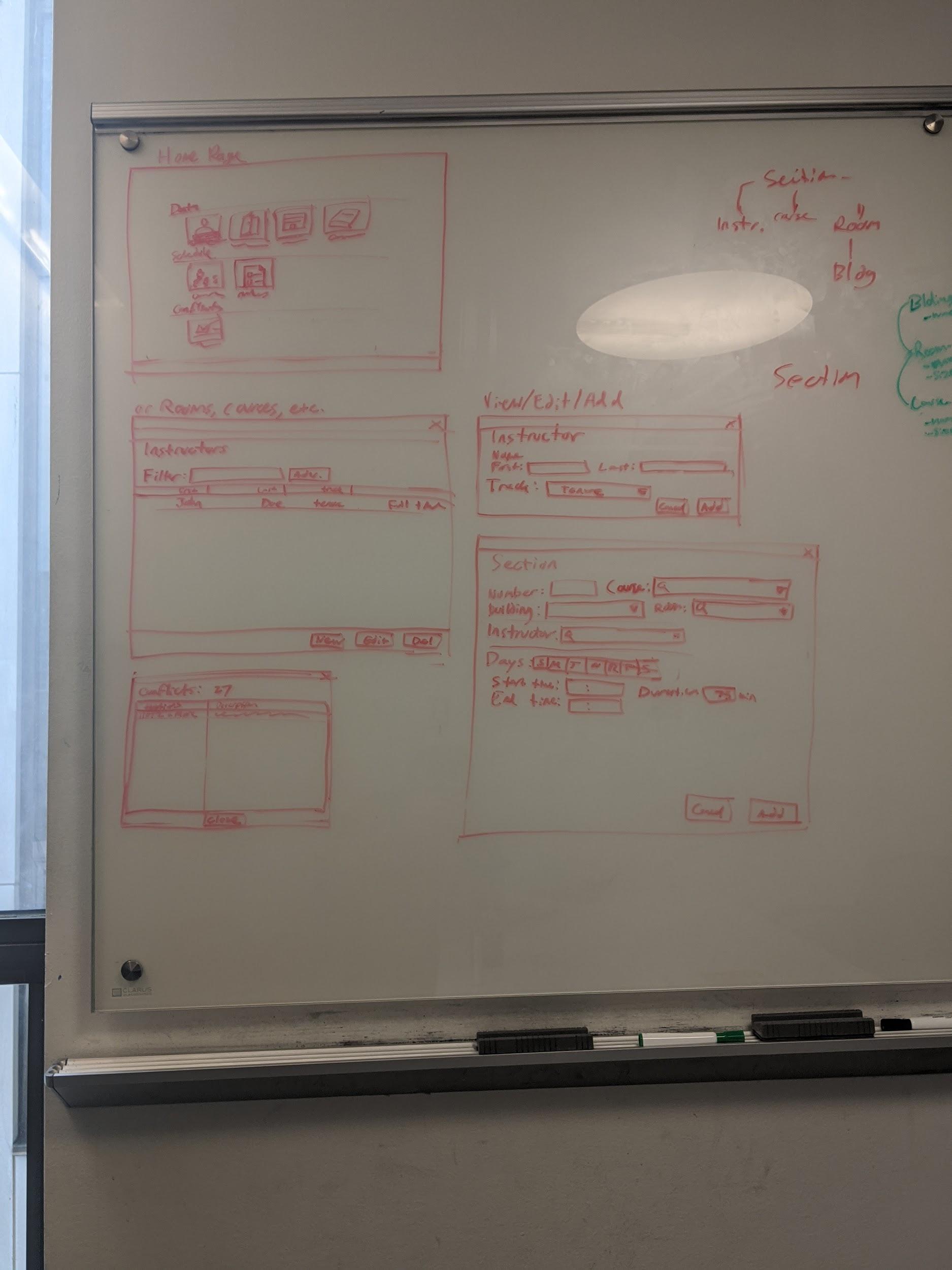
# **6 DEVELOPMENT VIEW**

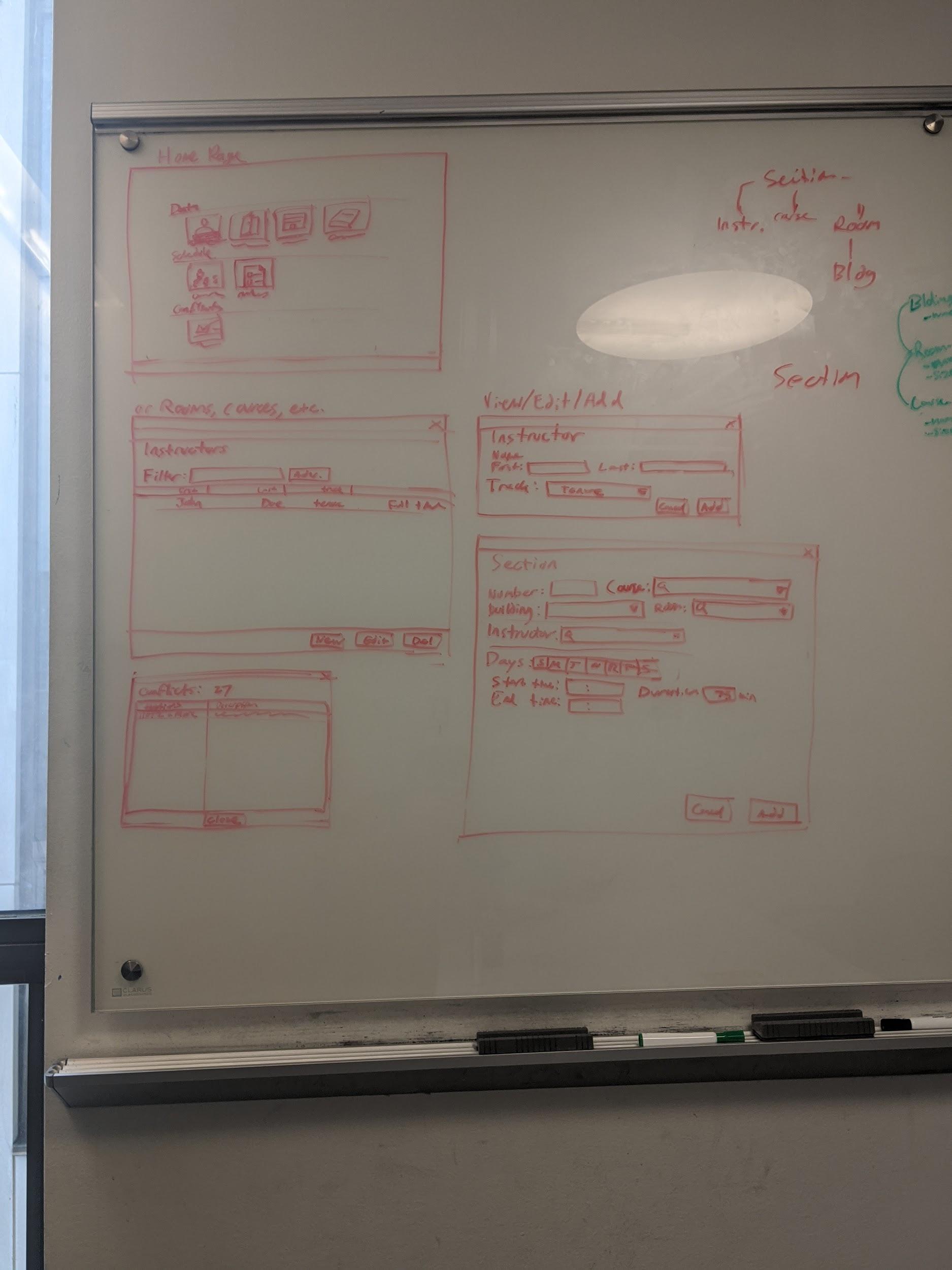
.[TBH]

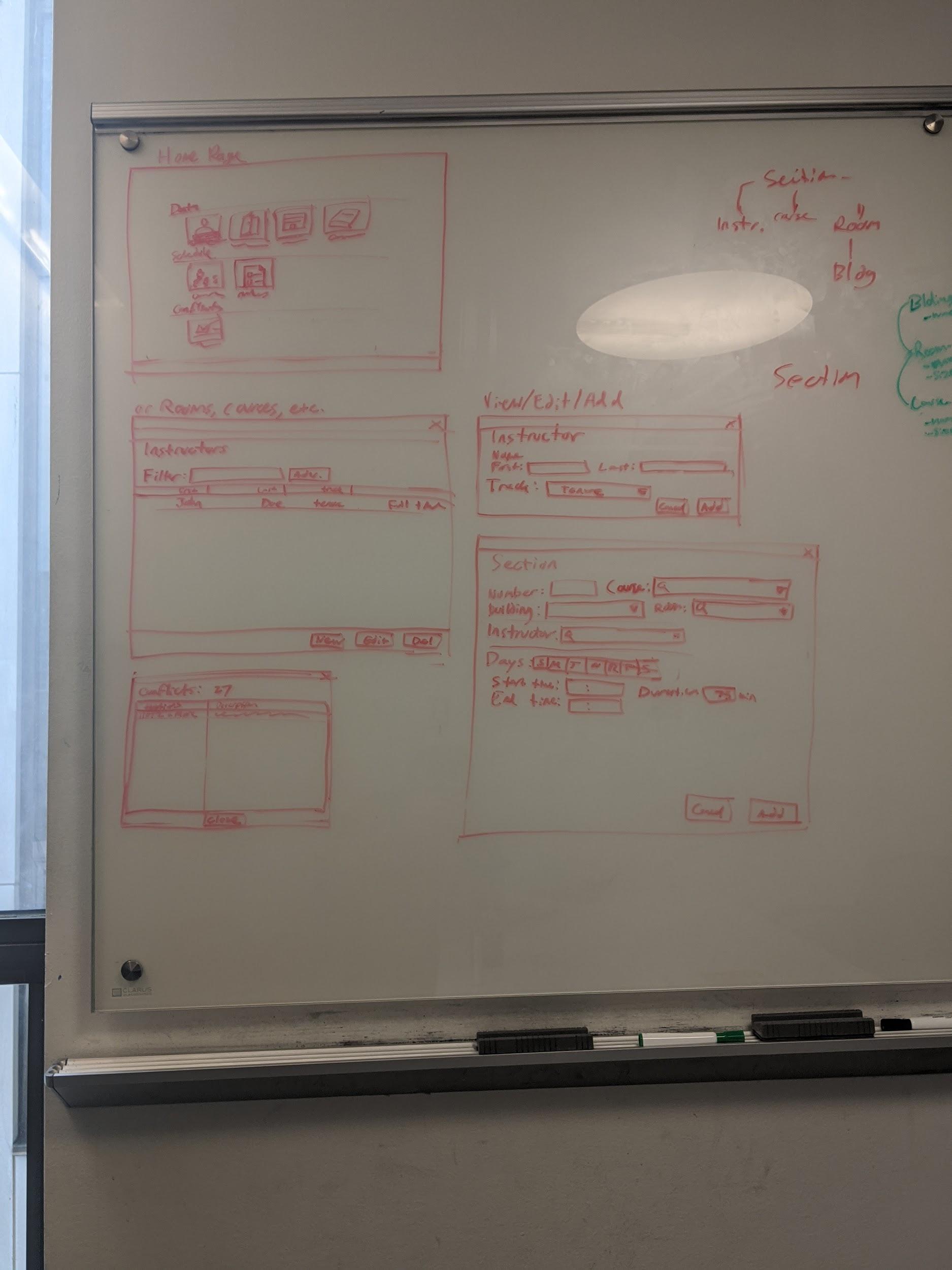
# **7 PROTOTYPES**











# **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

1. <https://kbingham-umkc.github.io/CS451-Software-Engineering-Capstone/SE/pl/architecture/fourplusone.pdf> [↑](#footnote-ref-0)