Software Engineering Project

Yellow Team

Tyler Kelly, Dylan Knepper, Aimane Mahtar, Caitlin Morales, Githendu Mukiri, Andrew Peterson, Mohmedsiddik Rana, Alisha Rizvi, and Jeremy Rojas

Problem We’re Solving

- Boston Code Camp needs help recording counts of attendance at their panels
- 3 counts done for each session
- Minimal Memorization

Overview of Our Solution

- Create a web based app
  - Simple and efficient
  - Create sessions
  - Lookup sessions
  - Input attendance

- Use JavaScript, HTML, and Firebase

Problems We Faced

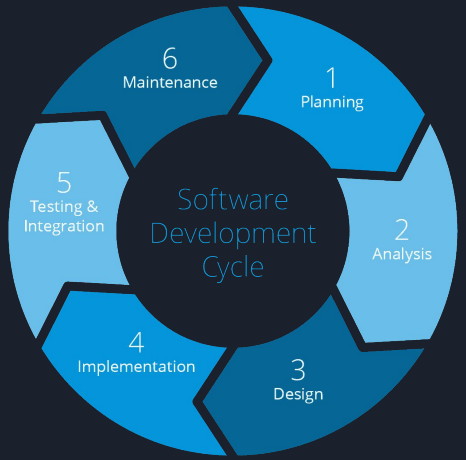
- Rushed development cycle
- Understanding and use of facades
- Underestimate time required for releases

Rushed Development

- Previous projects were centered around quick turnover
- Projects were not required to be maintainable
- Emphasis was put on what the program did rather then how it did it
- We were encouraged to jump right into coding instead of planning out robust code
- Led to a scatterbrained approach where solutions were not properly planned out before implementation began

Stay On Track

- Weekly meetings to make sure we were all on the same page
- Ask in depth questions of the client so we knew all of the parameters before continuing
- Focused on design documentation before starting to code
- Split into sub groups and worked independently
- Reconvened and reviewed what each group might have missed
- Only discussed the release currently being worked on



Requirements

- Every project begins with Requirements
  - What the system must do to fulfill the user/client wants and needs.
- Requirements Interviewing
- Use Cases
- 3 weeks to finalize requirements

Requirements

Functional Requirements

- List approved sessions
- Record session attendance 3 times per session
- Save and store attendance data
- Ability to view data

Requirements

Non-functional Requirements

- Prevent Bad Input
- Organize data for easy searching
- Remote viewing of data
- Searchability/Filtering of session data
- Secure data entry permissions for appropriate positions
- Ability to edit and update entered data

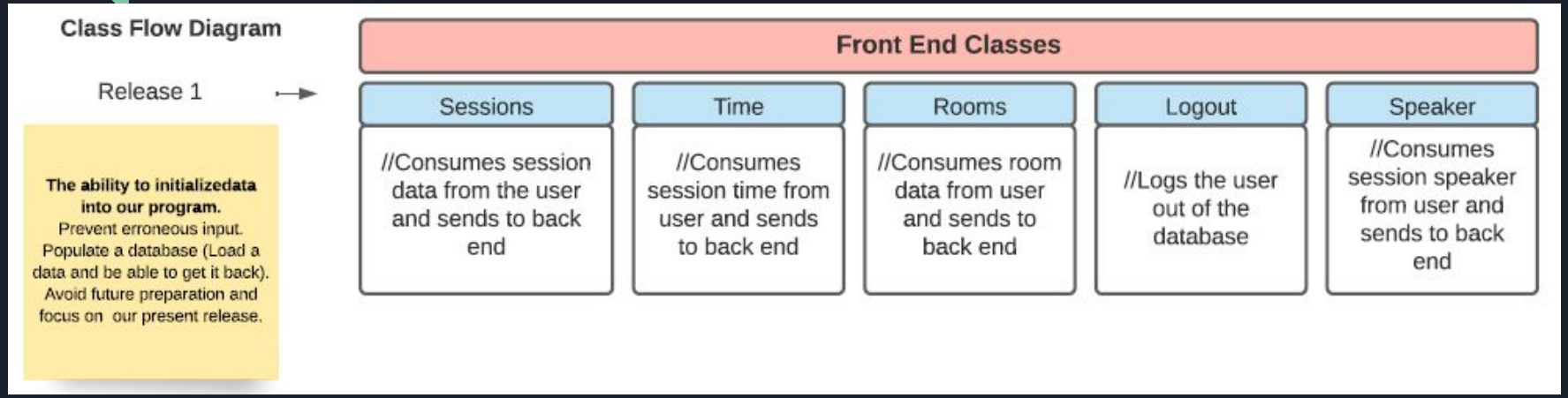
Use Cases

- Event Coordinator
  - One user
  - Initializes data to be used for sessions
  - Initializes sessions and schedule
- Counters
  - Multiple users
  - Adds attendance count to each session
- Data Extraction
  - One user
  - Pulls data from database and shares with sponsors

Release 1

- Meant for Event Coordinators
- Focuses on initializing event data
  - Rooms, speakers, schedule
- Establishes the backend and business logic layers
- Let user set up data for Release 2

Architecture
Initial Design



Initial design was a general idea of how our classes would work.

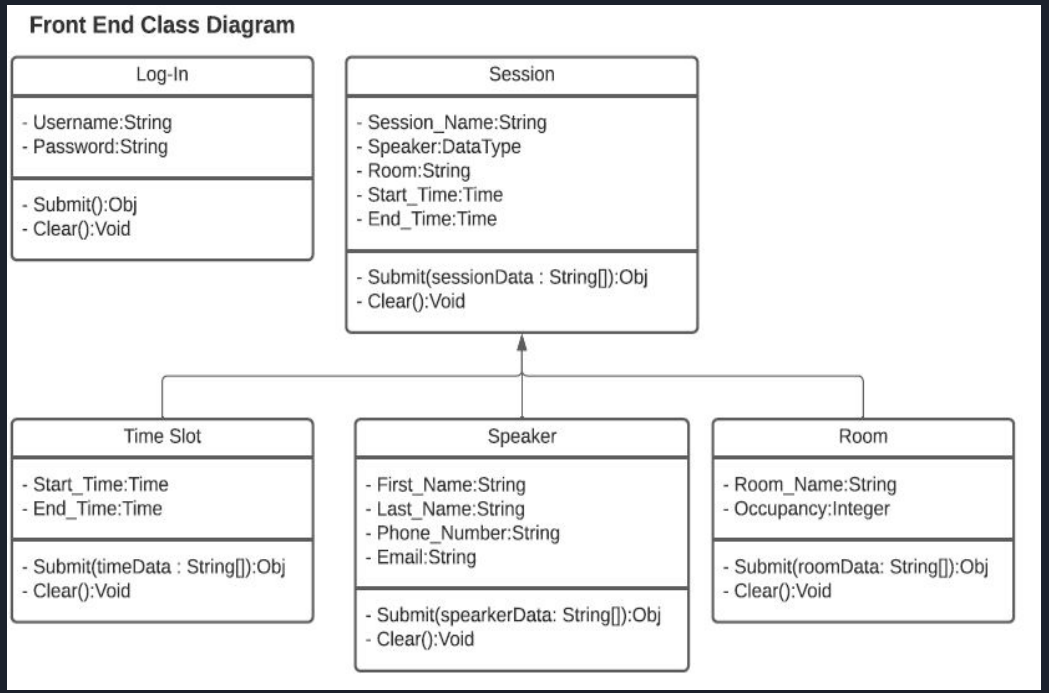
- Our initial software diagram lacked discipline
  - We learned that our software architecture would need to follow a more ***SOLID*** set of principles before we could think about coding.

Architecture

Designing our program with S.O.L.I.D Principles

Applying design principles to our software design

- Each one of our classes is designed with one responsibility
    - Time Slot, Speaker, and Room are responsible for one piece of information.
- Sessions are designed to be extended without modifying source code
- A Session object can be replaceable with instances of its subtypes.
- Splitting our session creation into separate interfaces over creating one general-purpose interface
- Software modules have loose relations.



Environments

- Web Based
  - Democratization of hardware
- JavaScript
  - CSS & HTML Integration for web development
  - NodeJS
    - NodeJS is cross-platform
    - Allows the javascript to be run on the server-side
      - Realtime Database
  - Familiarity
- Github
  - Version Control
  - Task Allocation

Environments

Firebase

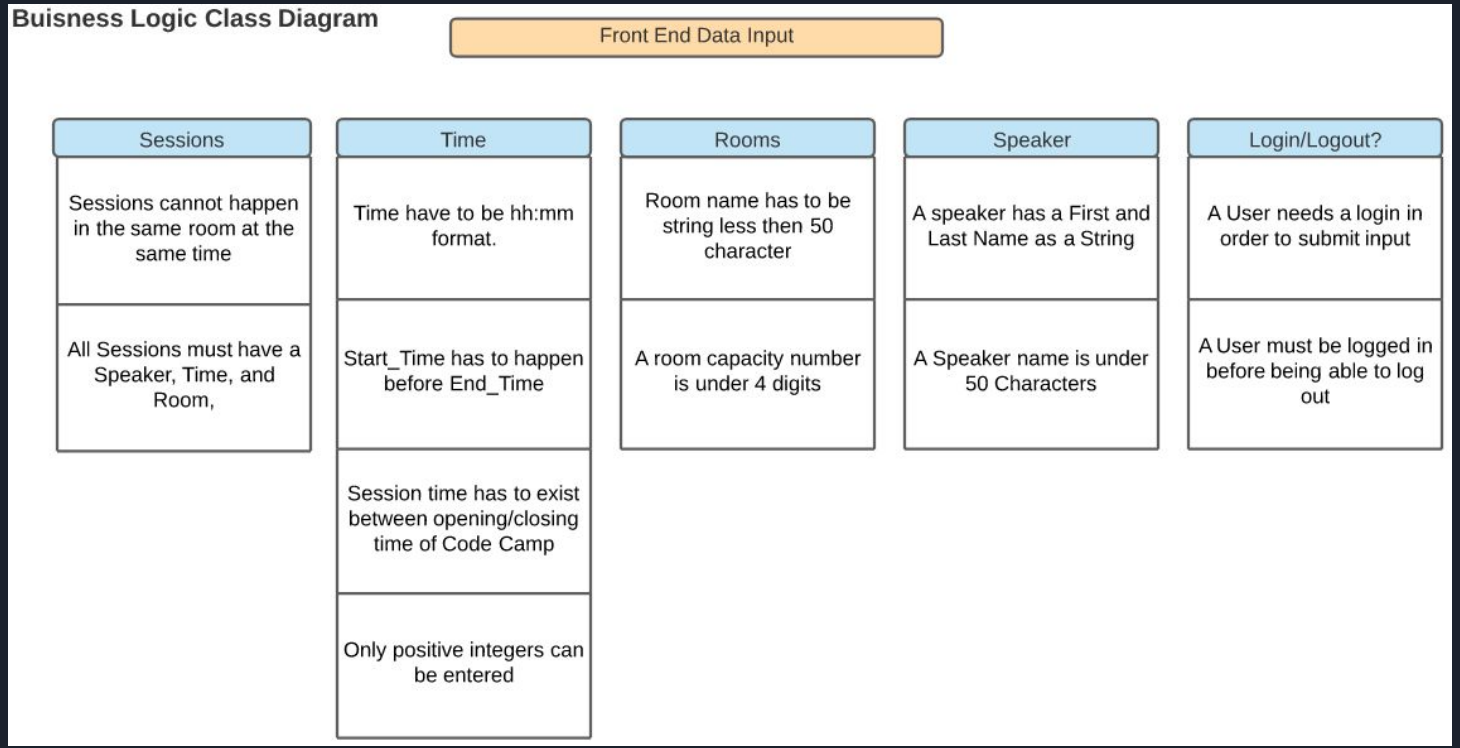
- Firebase
  - Firebase has zero SQL
    - Sessions are collection of documents
  - Dynamic Schema allows for faster data storage and retrieval
  - Security
    - Server-side rules
- Facade Issues
  - Array Structure
  - Client to Server Hierarchy

Business Logic

- Our business logic was centralized on the front end
  - Abided by nonfunctional requirements (aka business rules)
  - Helped us structure our front end UI
  - Organize data to be passed into DB facade/controller

- Sessions
  - Speaker, Time and Room
- Time
  - HH:MM format, start time before end time, positive ints only
- Rooms
  - No duplicate entries, room # < 10 digits, positive ints only
- Speaker
  - First/Last name string input, Speaker name < 50 chars
- Login/Out
  - User requires login to submit, User must be logged in to log out

Business Logic



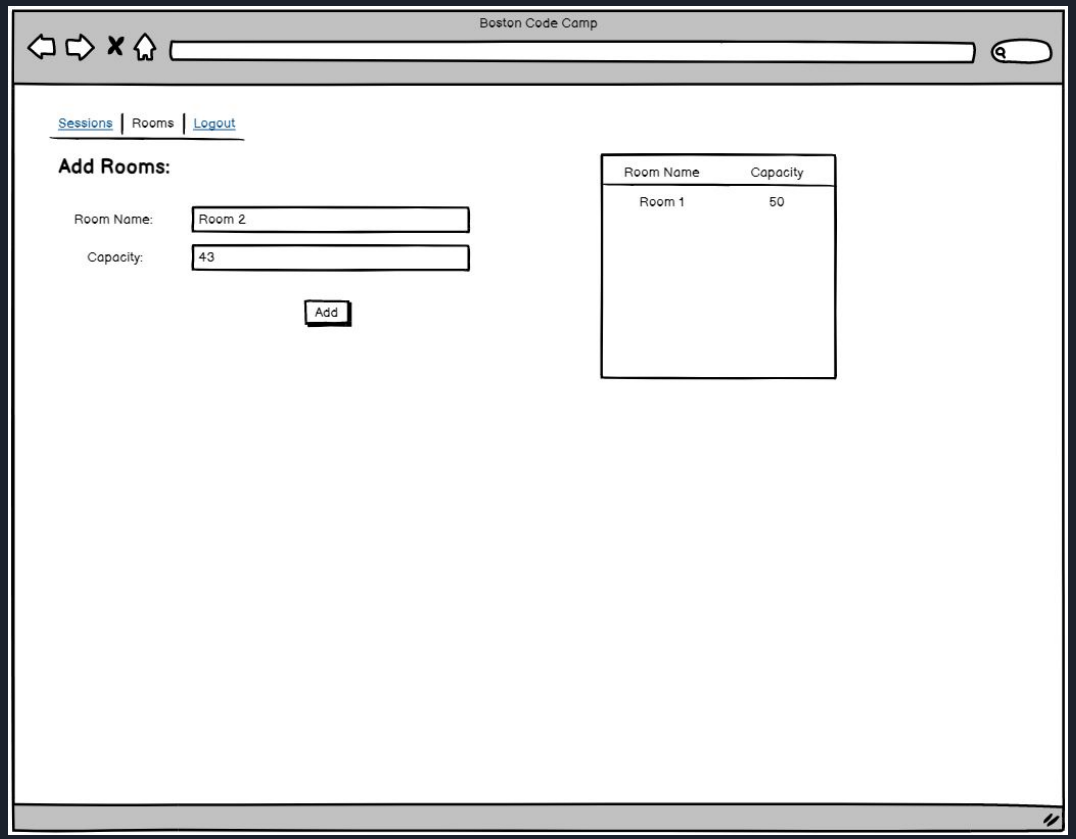
Front End

- HTML
- CSS
- Javascript,
- jQuery
- Bootstrap 4
- SCSS

Front End

- Separate pages for Sessions, Rooms, Speakers, and Time Slots
- Sessions page will have dropdowns for all Inputs, Predefined from other pages
- All pages will have links to the others at the top

UI Diagram


Boston Code Camp
Sessions | Rooms | Logout
Add Rooms:
Room Name:
Room 2
Capacity:
43
Add
Room Name
Capacity
Room 1
50

Back End

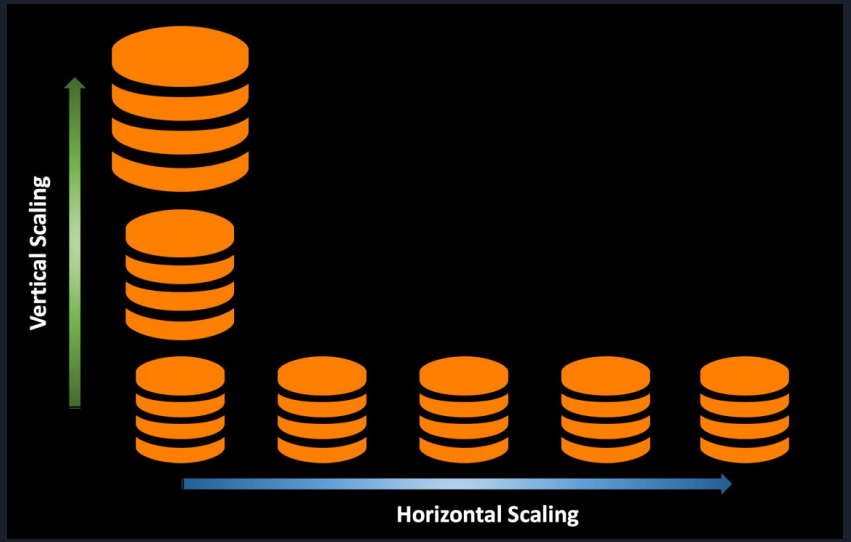
- Microsoft Azure
- MySQL
- MongoDB
- Many Options
- Lets try something new?

Google Firebase

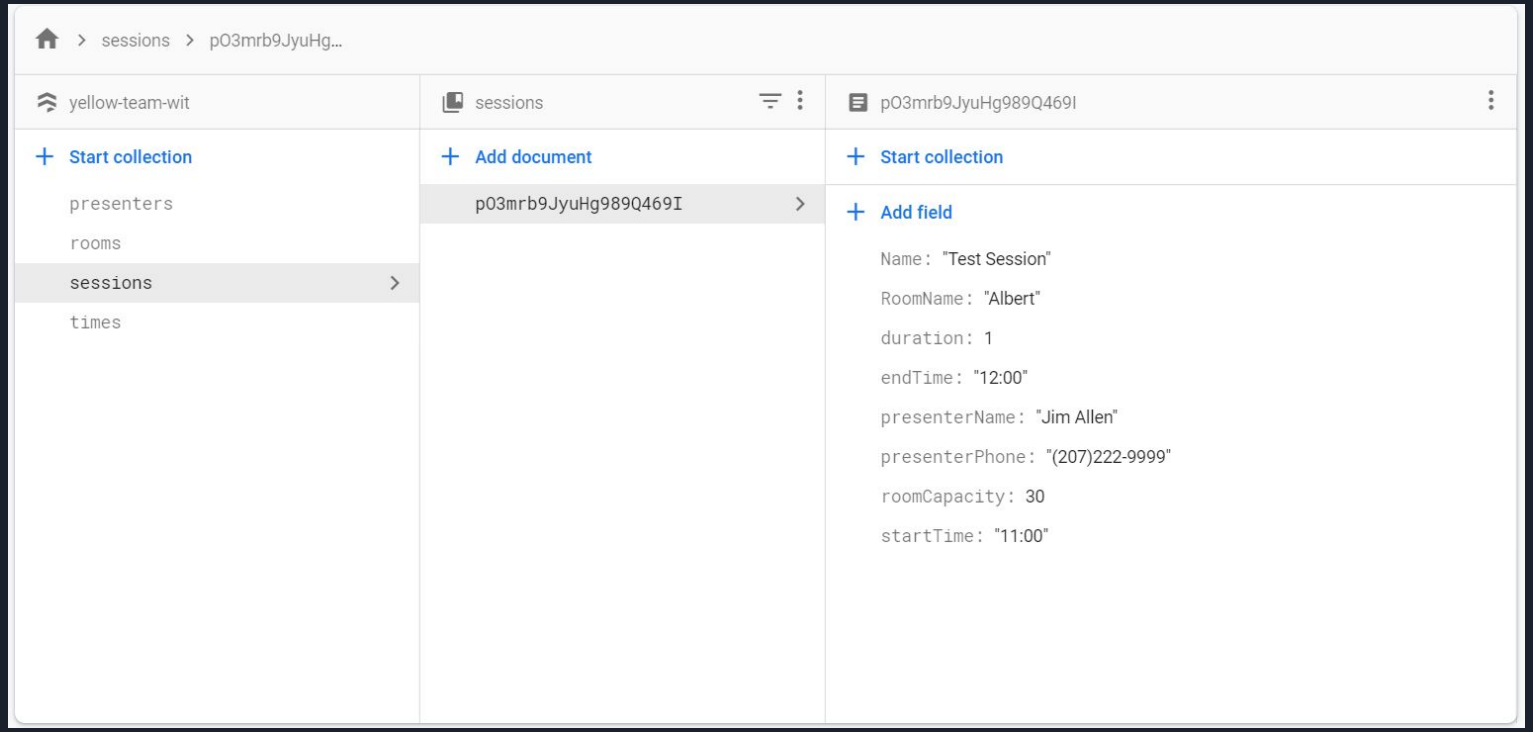
- Firestore
- Realtime Database
- Storage
- Hosting
- Cloud Functions
- Machine Learning

Horizontal Expansion

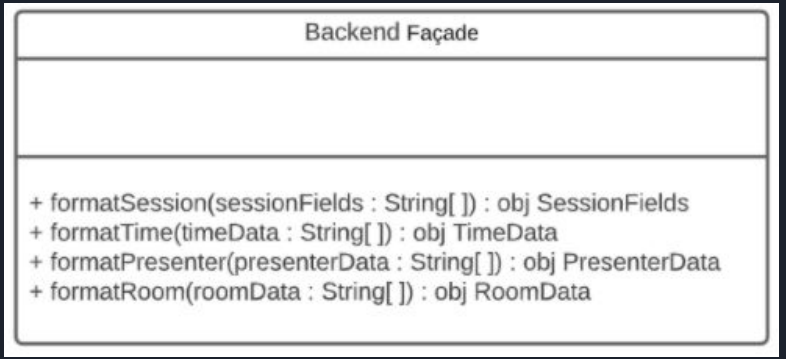
- Expand “forever”
- Variable
- Less Cost



Back End


sessions
pO3mrb9JyuHg...
yellow-team-wit
Start collection
presenters
rooms
sessions
times
sessions
Add document
pO3mrb9JyuHg989Q469I
pO3mrb9JyuHg989Q469I
Start collection
Add field
Name: "Test Session"
RoomName: "Albert"
duration: 1
endTime: "12:00"
presenterName: "Jim Allen"
presenterPhone: "(207)222-9999"
roomCapacity: 30
startTime: "11:00"

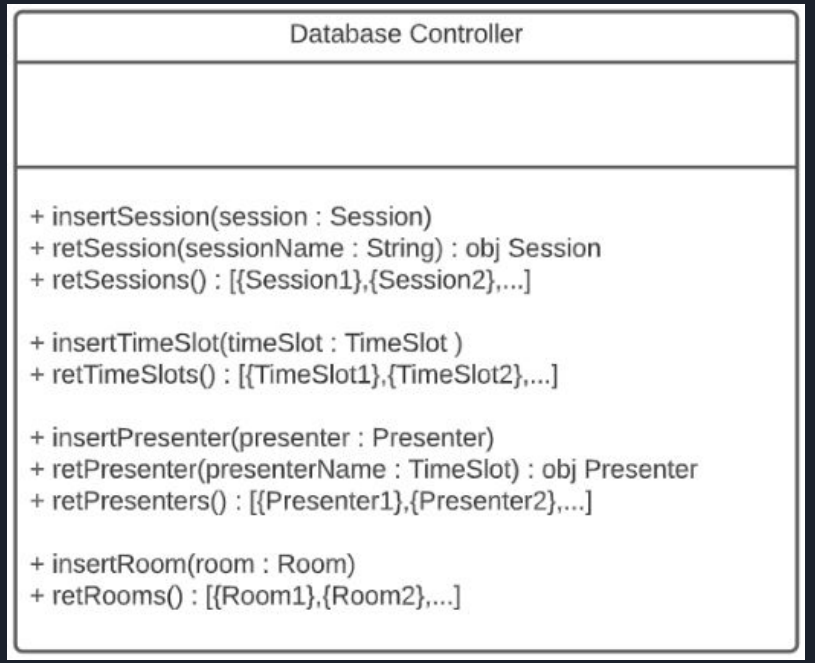
Backend Façade



- Connected to Business Logic
- Implements another security layer
- Creates objects for sessions, time, presenter, and rooms

Database Controller

- Connection to Database
- Retrieves Data
- Inserts Data
- Implements another layer of security



Business Logic -> Database Facade -> Database Controller -> Database

Final Thoughts

- What we would do differently:
  - Better preparation when asking client questions
  - Have a better ticketing system
  - Better time management & communication

- Takeaway:
  - Use of Agile & S.O.L.I.D principles

Questions?

References

- File:Node.js logo 2015.svg. (n.d.). Retrieved April 11, 2021, from https://commons.wikimedia.org/wiki/File:Node.js_logo_2015.svg
- File:image00.png. (n.d.). Retrieved April 11, 2021, from https://1.bp.blogspot.com/-YIfQT6q8ZM4/Vzyq5z1B8HI/AAAAAAAAAAc/UmWSSMLKtKgtH7CACEIUp12zXkrPK5UoACLcB/s1600/image00.png
- File:GitHub-Logo.png. (n.d.). Retrieved April 11, 2021, from https://logos-world.net/wp-content/uploads/2020/11/GitHub-Logo.png
- File:JavaScript-logo.png. (n.d.). Retrieved April 11, 2021, from https://commons.wikimedia.org/wiki/File:JavaScript-logo.png
- Author Jasper van der Hoek May 16. (2020, February 27). What is the agile development Cycle? A quick intro to agile development. Retrieved April 11, 2021, from https://www.mendix.com/blog/pursuing-a-full-agile-software-lifecycle/