Scheduling System for Math Department Resources

Matthew Mooney

Brandon Rozzi

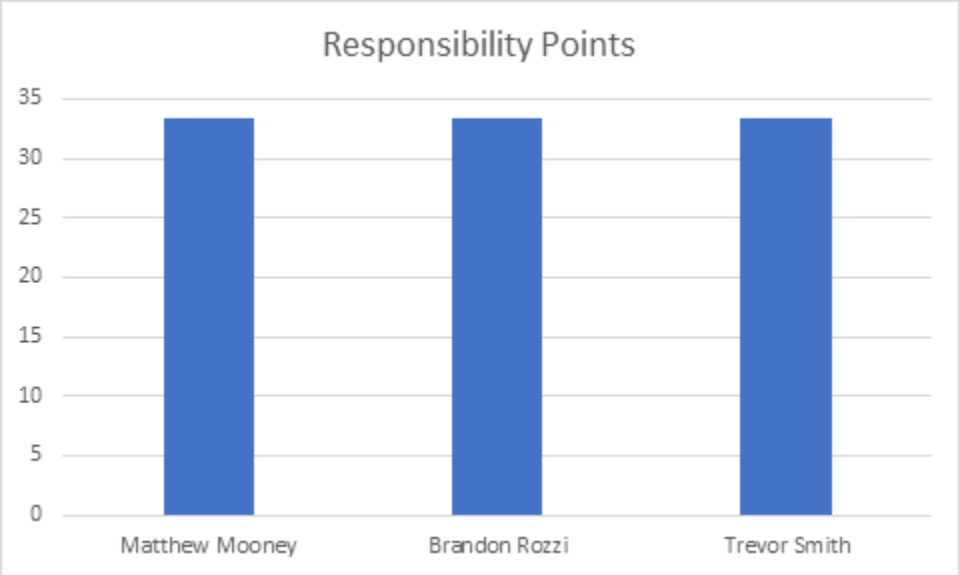
Trevor Smith

Individual Contribution Breakdown

Table 1: Responsibility Matrix

Allocation by Team Member		**Team Member Name**		
		Matthew Mooney	Brandon Rozzi	Trevor Smith
Responsibility levels	Sec 1: Interaction Diagrams	33%	33%	33%
	Sec 2: Class Diagram and Interface Specification	33%	33%	33%
	Sec 3: System Architecture and System Design	33%	33%	33%
	Sec.4: Algorithms and Data Structures (if applicable)	33%	33%	33%
	Sec.5: User Interface Design and Implementation	33%	33%	33%
	Sec.6: Design of Tests	33%	33%	33%
	Sec.7: Project Management and Plan of Work	33%	33%	33%

Graph 1: Responsibility Chart



Contents

Interaction Diagrams

Approval

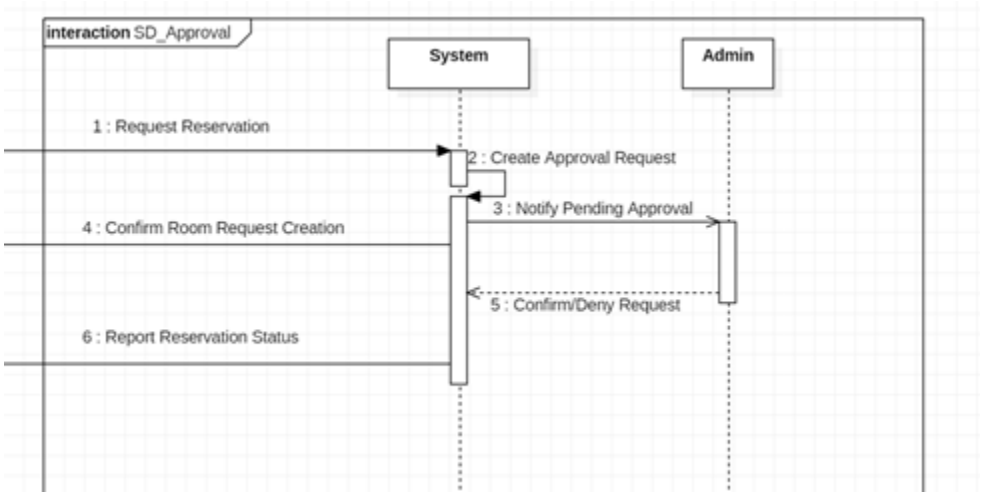


Figure 1: Approval Sequence Diagram

The Approval Use Case has two main contributors, and one outside actor who starts the flow of events. The design principle of the System actor is the Expert Doer Principle. The System actor must be notified of an outside user wanting to reserve something, create the request, and then notify the Admin of the pending request. Since the System is tasked with both communication and computation responsibilities, using Expert Doer Principle for the System makes sense. The Admin's design principle is High Cohesion since the Admin shouldn't need to perform any computations. The Admin should only need to communicate with the system that the request has been approved or denied.

Room Scheduling

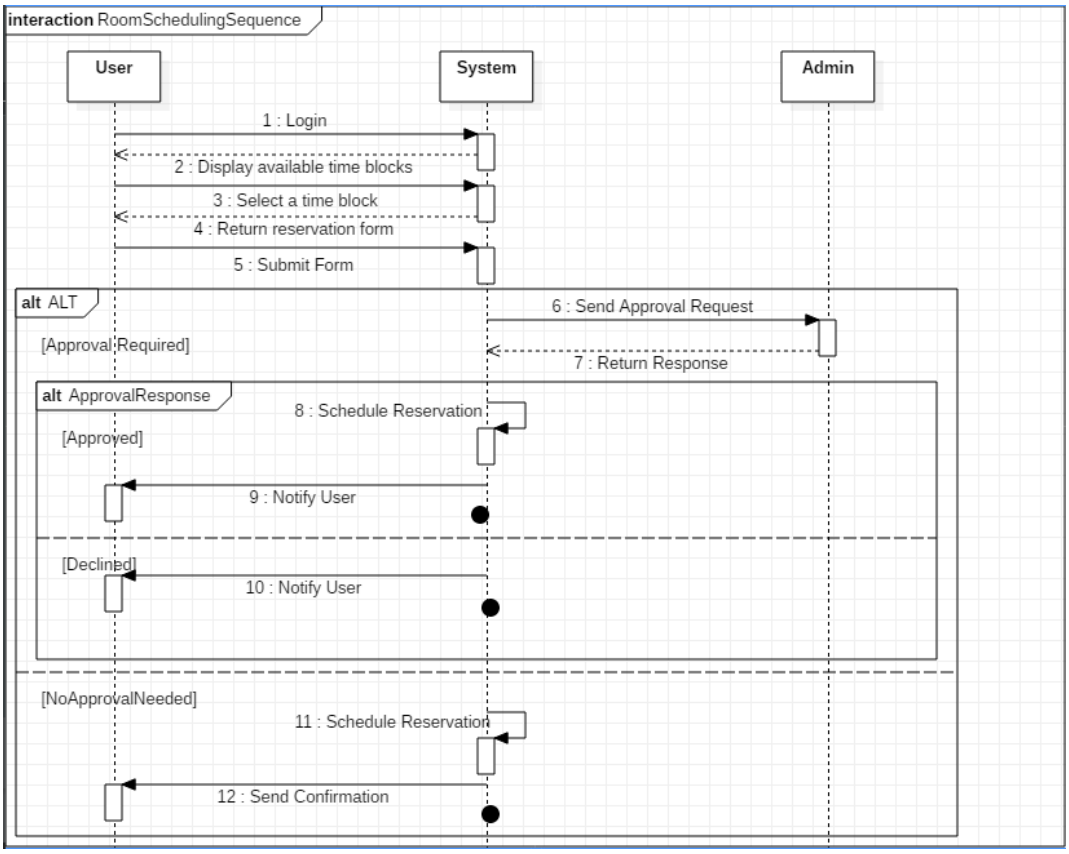


Figure 2: Room Scheduling Sequence Diagram

The room scheduling use case has two main contributors and an outside actor who starts the flow of events. The System actor must be notified of an outside user wanting to reserve a room, create the request, send the request, schedule the event and notify the end user. Since the System is tasked with both communication and computation responsibilities, using Expert Doer Principle for the System makes sense. The Admin's design principle is High Cohesion since the Admin shouldn't need to perform any computations. The Admin should only need to communicate with the system that the request has been approved or denied.

Adding/Editing Tutors

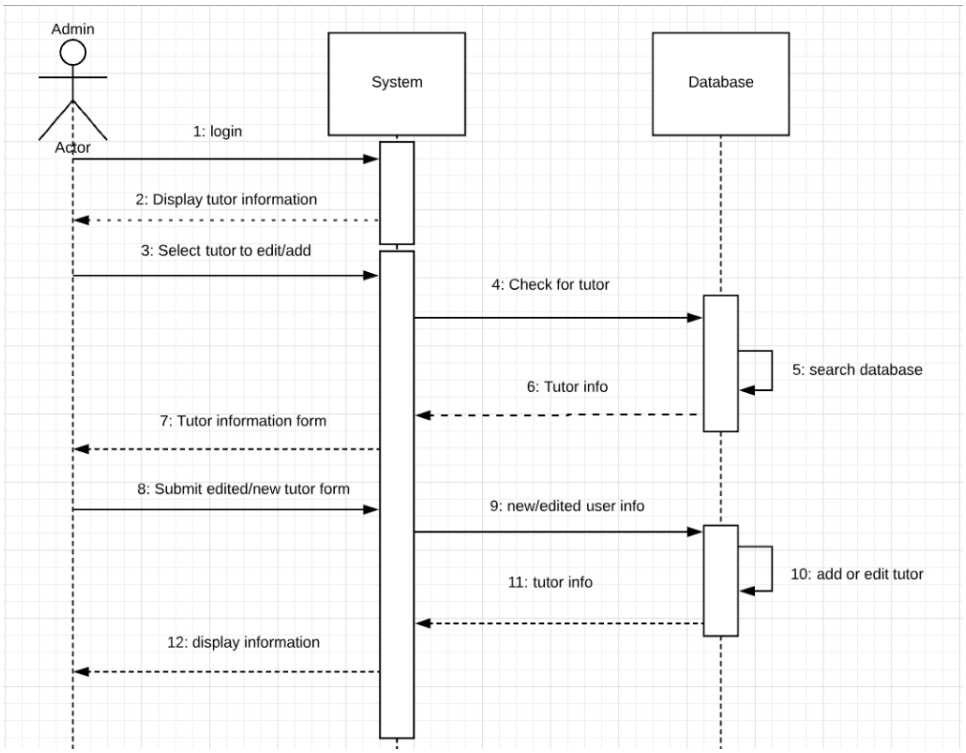


Figure 3: Adding/Editing Tutors Sequence Diagram

The Adding/Editing Tutors Use Case has one main contributor who starts the flow of events. The admin actor must login and select a tutor to edit/add, give information about the tutor, and submit the form. The design principle of the System actor is the Expert Doer Principle. Since the system actor does not need to do any computations then the admins design principle is of high cohesion. The admin only needs to communicate with the system to add or edit a tutor.

Class Diagram and Interface Specification

Room Scheduling System

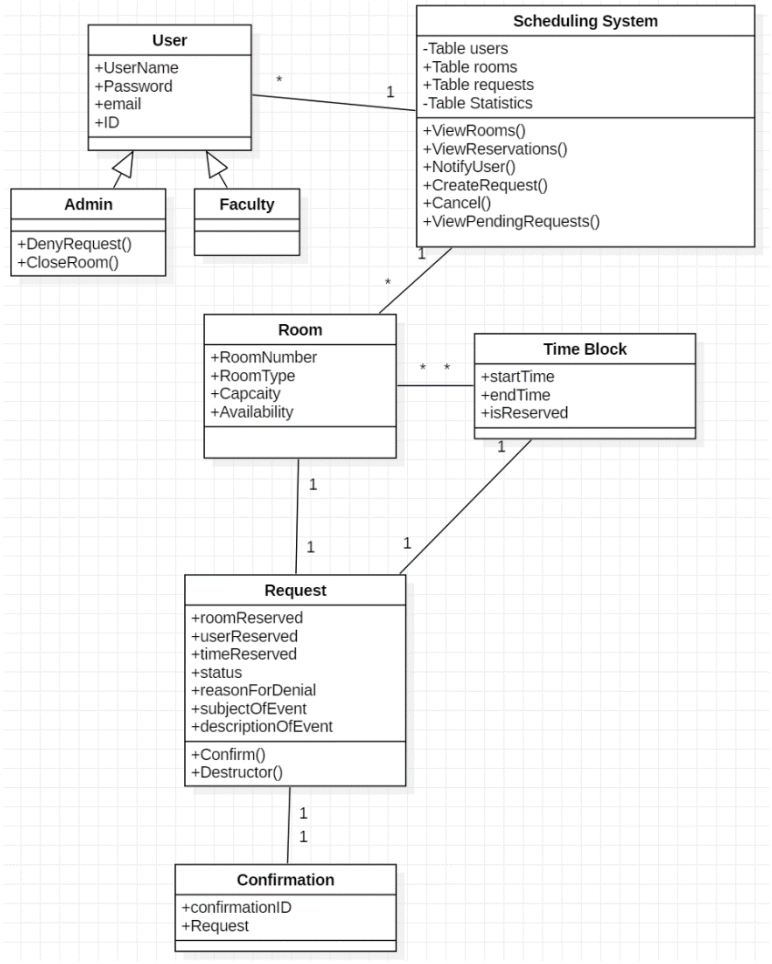


Figure 4: Room Scheduling System Class Diagram

Data Types and Operations

Scheduling System

- Table users;
- Table rooms;
- Table requests;
- Table Statistics;
- Room[] ViewRooms() - Returns an array of all rooms.
- Request[] ViewReservations() - returns an array of all requests made by the user.
- Void NotifyUser(Confirmation) - notifies user of their request being approved/denied and why.
- Void CreateRequest(user, room, TimeBlock) - Creates request.
- Void Cancel(Request) - cancels the request.
- Void ViewPendingRequests() - shows the admin all requests that are pending.

User

- Private String UserName - The user's login identification
- Private String Password - The user's login password
- Private String email - the user's email address
- Private String ID – the users ID, typically their Y number.

Admin (extend User)

- Void DenyRequest(Request) - Denies the selected request, informing the user it was denied and for what reason.
- Void CloseRoom(Room, DATETIME, DATETIME) - Reserves the specified room from the two inputted times. (reserves the room so that it is "closed" to others)

Faculty (extend User)

-

Room

- int RoomNumber - The number of the room, first digit details floor it is on.
- String RoomType – The type of room, whether it is a lab or a conference room
- Int Capacity – total number of people that the room can hold.
- TimeBlock[] Availability – An array of the rooms availability, time slots are removed when the room is in use.

Request

- Room roomReserved – The room that the Request is for.
- User userReserved - The user who requested the room.
- TimeBlock timeReserved - The TimeBlock that the request is for.
- Reoccuring – integer 0-6 for what day of the week it is reoccurring, negative value means not reoccurring.
- Int status - The current status of the request. 0 = pending, -1 = denied, 1 = approved.
- String reasonForDenial - The reason the admin gave for denying the room.
- String subjectOfEvent - The name of the event the reservation is for.
- String descriptionOfEvent - A detailed description of the event.

Confirmation

- Usigned int confirmationId - The identification of the confirmation
- Request* Request - The request linked to the confirmation.

Time Block

- DATETIME startTime – Start time of the reservation of the room.
- DATETIME endTime - End time of the reservation of the room.

Statistic

- Room analyzedRoom – The room being analyzed by the statistic
- Date DayofWeek – The day being analyzed by the stat
- TimeBlock highestUtil – The timeblock with the highest level of utilization
- TimeBlock lowestUtil – The timeblock with the lowest level of utilization
- Map<TimeBlock, float> avgVisitsHourly – A map of average visits during a given timeblock

Traceability Matrix

Domain Concept	Classes	Explanation
User	User	A user represents the concepts of the person who will be utilizing the service.
User	Faculty	An extended user who is automatically approved for room requests.
Admin	Admin	Admin is an extension of a user who has management abilities.
Room	Room	A room is the representation of the concept of a physical room that will be reserved.
Reservation	Request	A request represents a request for a reservation, a request contains all details of a reservation. An approved request represents a scheduled reservation.
Reservation	Confirmation	When a request is created, approved, declined, or cancelled, an associated confirmation must be generated to notify a user.
Room SS	Scheduling System	The room scheduling system represents the system which will be responsible for completing operations and communicating with the user.
Stats	Statistic	A statistic contains analytical data that the admin can use to understand room utilization.

Tutor Scheduling System

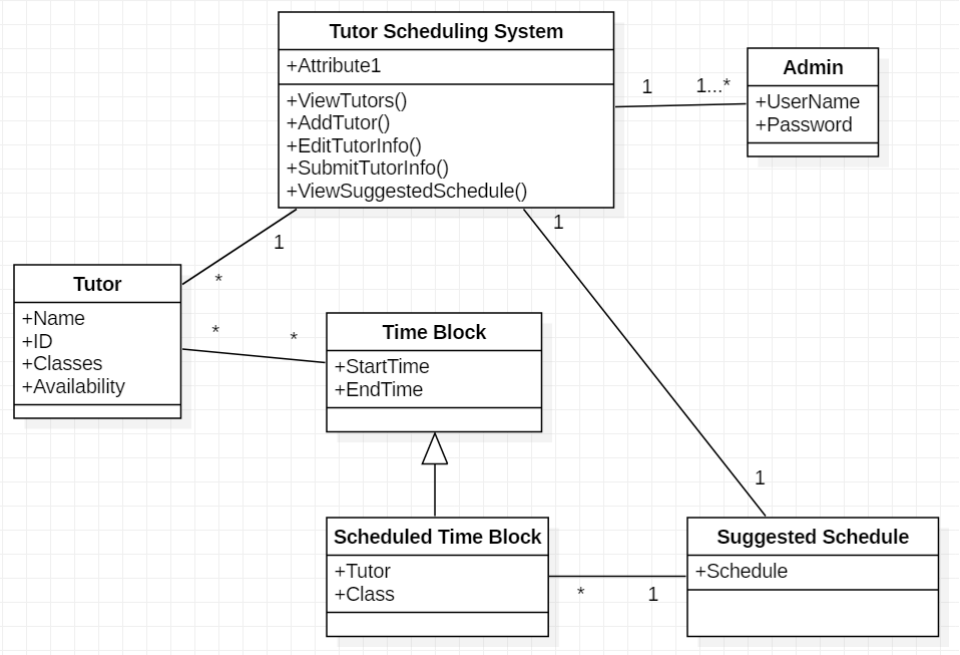


Figure 5: Tutor Scheduling System Class Diagram

Data Types and Operations

Tutor Scheduling System

- Table Tutors
- tutor[] ViewTutors() - return a list of tutors
- form AddTutor() - add a tutor
- form EditTutorInfo(tutor) - edit information about a tutor
- void SubmitTutorInfo() – submits a tutor form and adds the info to the tutor table
- SuggestedSchedule ViewSuggestedSchedule() – Returns a suggested schedule for the admin to view

Tutor

- String name – Tutors full name.
- String ID – Tutors YSU ID.
- String Classes – The specific classes a tutor can teach.
- TimeBlock[] availability - An array of the hours a tutor can work.

Admin

- String Username - Admins login id.
- String Password - Admins password to log in.

Time Block

- DATETIME startTime - The time a tutor can start work.

- DATETIME endTime - The time a tutor has to leave.

Scheduled Time Block

- Tutor Tutor
- String Class

Suggested Schedule

- ScheduleTimeBlock[] Schedule

Traceability Matrix

Domain Concept	Class	Explanation
Admin	Admin	The admin class represents the administrator who will interact with the system.
Tutor SS	Tutor Scheduling System	This represents the system which the admin will complete the operations and communicate to the admin.
Tutor	Tutor	Represents a tutor who works at the MAC.
Tutor	Time Block	A tutor has a time block that represents a period in which they are available.
Suggested Schedule	SuggestedSchedule	A suggested schedule is a data type that the tutor scheduling system will generate for the admin to use in scheduling tutors.
Suggested Schedule	ScheduledTimeBlock	A scheduledtimbeblock is an extended timeBlock that is used to represent a period in which the tutor is scheduled to work.

System Architecture and System Design

Architectural Styles

Client-Server:

- Service must be accessible from anywhere.
- Some level of security required.
- Server needed for any and all decision making.
- Allows for user to pull information at runtime so the information shown is always up to date.
- Can easily change design to an n-tiered approach if necessary.
- Client-Server architecture is easily scalable.

Identifying Subsystems

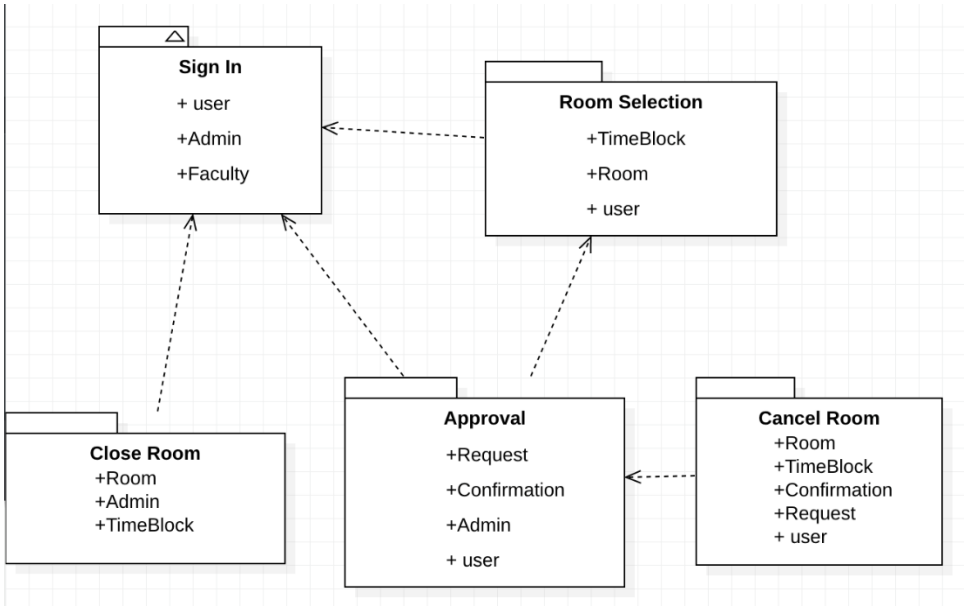


Figure 6: Room Scheduling Package Diagram

Sign In Package

- Subsystem to authenticate type of user.
- User Class - Public
- Admin Class – Public
- TimeBlock Class – Public

Room Selection Package

- Subsystem for a user or faculty member to select a room to reserve.
- Room Selection Package is dependent on the sign in package on whether a user or faculty member has signed in.
- User Class - Public
- TimeBlock Class - Public
- Room – Public

Approval Package

- Subsystem to get the admin approval and confirm a room reservation.
- Dependent on the sign in system. Only admin has access.
- Dependent on the Room Selection Package. Approval Needs a user, room, and timeblock for the Room Selection Package.
- Request class - Public
- Confirmation Class – Public
- Admin Class – Public
- User Class – Public

Cancel Room Package

- Subsystem for a user or faculty member to cancel a room reservation.
- Cancel Room Package is dependent on Approval. An accepted approval needs to happen before a user can cancel a reservation.
- Room – Public
- TimeBlock – Public
- Confirmation – Public
- Request – Public
- User- Public

Close Room Package

- Subsystem for the admin to close a room so it cannot be reserved
- Close Room is depending on whether an Admin, user, or faculty member signs into the system.
- Room Class – Public
- Admin Class- Public
- TimeBlock Class – Public

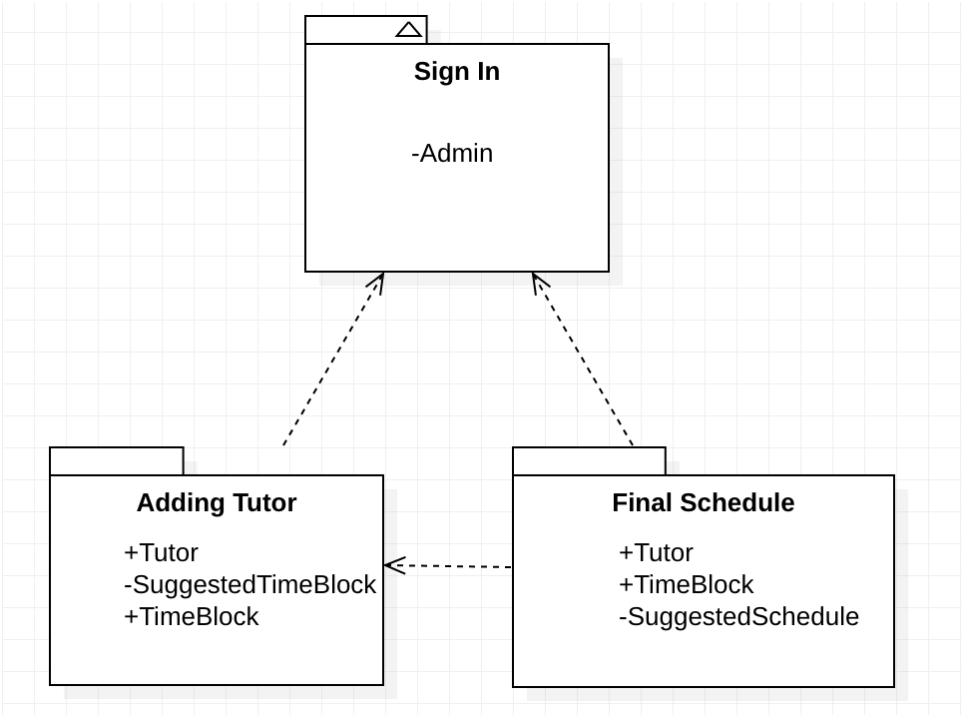


Figure 7: Suggested Schedule Package Diagram

Sign In Package

- Subsystem to authenticate the admin's credentials.
- Admin – Private

Adding Tutor

- Subsystem for the admin to add a tutor for the suggested schedule
- Dependent on the sign in Package for an admin to log in.
- Tutor Class – Public
- SegestedTimeBlock Class – Private
- TimeBlock – Public

Final Schedule

- Subsystem that gives the admin a suggested work schedule for the tutors.
- Dependent on sign in Package for an admin to log in.
- Dependent on Adding Tutor Package. Tutor Need to be added for a suggested schedule to be made
- Tutor Class – Public
- TimeBlock Class – Public
- SuggestedSchedule Class - Public

Subsystems to Hardware

Client Machine:

- Client Browser will pull html files containing dynamic content
- Client Browser will execute JavaScript to send an http request to the API and acquire the dynamic content
- Client Browser will display the content

Server:

- AWS S3 will store the html and JavaScript which will be pulled by the user's browser
- RESTful API (AWS API Gateway) will catch http requests and trigger server functions (Lambda).
- AWS Lambda will communicate with the database, complete operations and return dynamic content
- Database (DynamoDB) will store data to which will be pulled and edited by Lambda

Persistent Data Storage

AWS S3:

- Will be responsible for storing flat html and JavaScript files

DynamoDB:

NoSql database which will store class data

Network Protocol

HTTP:

- User's Browser will use the HTTP protocol to request webpages from S3
- HTTP requests will be sent to the API to request dynamic content and or perform operations.

Global Control Flow

- The system is event driven. Events occur when a webpage is loaded and when a user performs an operation. These events result in API calls which executes a Lambda function.
- The Room Scheduling System will require real time updates. This will operate in a periodic fashion, updating multiple times a minute as well as on event triggers.
- The system will operate on multiple threads. If two users perform the same operation at the same time, multiple independent Lambda functions are initiated. This allows for infinite scaling. The threads are independent of each other, with no synchronization.

Hardware Requirements

- Client internet connection with a minimum bandwidth of 512 Kbps.
- Windows XP SP 2 or higher
- 128MB Minimum of RAM
- Pentium 4 Processor or higher

Algorithms and Data Structures

Algorithms

The tutor scheduling system requires a mathematical algorithm to create the suggested schedule from the list of tutors that work for the MAC. The algorithm the team has currently designed is a simple algorithm that schedules tutors in order from least available to most available. In theory, this algorithm will not only fill the schedule with tutors but also distribute hours somewhat fairly between tutors. As the tutors are being stored in a database, the algorithm needs to sort the database by the number of time blocks a tutor is available in ascending order. Once it has the list organized by the least available tutor, the system simply needs to go through each time block and get the first tutor it finds in the database that can work that time block and schedule them. This process needs to be repeated for each possible class.

User Interface Design and Implementation

As both the room scheduling system and tutor scheduling system are web-based the team decided to create the user interface using Bootstrap. As Bootstrap also contains libraries for Javascript, it works well with our backend systems which also run on Javascript. Our initial mockups are functionally similar to the end result, with only a few minor differences made to make the website even easier to use.

Ease-of-use was a key concept at the forefront of the User Interface design. Our goal was to allow the user to request a room in as few clicks as possible. Currently, once a user has logged in, they can request a room in as little as three clicks. This number changes based on the users need, such as changing the date of the request or requesting multiple reservations at a time, but the overall ease-of-use remains at an ideal level.

The tutor scheduling system was also designed with ease-of-use in mind. The team wanted to ensure that the administrator for the system would be able to figure out the system quickly and then use it efficiently. This was accomplished by using forms to add or edit tutors. The form has labeled fields

for all the information the system needs about an individual tutor and simply needs to be filled out. When the administrator needs to edit a tutor, the fields of the form are automatically filled with the information that was previously entered so the administrator only needs to input the changes needed.

Design of Tests

The functionality of the room scheduling system mostly depends on inserting, retrieving, and editing various tables in a database. As such, the tests for each individual component will be similar with the only differences being what data is being affected in the database. These tests can be broken down into three different categories: Inserting, Retrieving, and Editing. However, retrieving data will be done in both the inserting and retrieving tests for each type of data so it will not require a test.

- Inserting Data Test
    - The test will insert a test record into the system for each type of data (user information, room requests, etc.)
    - It will then check to see if the test data is available inside the database by checking the relevant table for the information. If the test finds a row inside the table that has all the test data, then the test has passed.
    - After the test has passed, the test information will be deleted from the database.
- Editing Data Test
    - This test will retrieve the last record in the database and will save the information.
    - The test will then edit the information in the last record to be the desired test information.
    - To check whether the test has passed or failed, the data of the last record will be retrieved again and checked against the test data. If they are the same, the test has passed.
    - Finally, once the test has passed or failed, the data in the last table will be restored to its original value so that it does not affect the actual database.

The functionality of the tutor scheduling system also depends on the tests above, as such it will use similar methods of testing. On top of the above tests, it also needs a test for the creation of the suggested schedule to test the algorithm.

- Suggested Schedule Creation
    - This test will need a test database so that the result of the algorithm can be tested in a controlled environment. The database will contain enough information to fill a schedule.
    - The test will run the algorithm for creating a schedule against the test database.
    - The result of the algorithm will then be compared against the expected data to ensure that the algorithm is working as expected.
    - If the expected data and the algorithms results are the same, then the test has passed.

While the tests described above ensure that our database is working correctly, the functionality of the website itself also needs to be tested. The team plans to do real user tests of the system as well to ensure that it is working as expected. Between the automated testing of the database and the manual testing of the webpages, the system should have a high-degree of test coverage.

Project Management

Merging Contribution Conflicts

- The team worked in an online document simultaneously while discussing design elements.
- Each team member worked together on the class diagrams and package diagrams for both the scheduling system and suggested tutoring schedule, so all team members will be on the same page when implementing the system begins.

Project Coordination and Progress Report

- Implementation has not begun.
- The team plans to begin implementation in the next few days to keep on schedule.

Plan of Work

We are currently building our prototype which we will use to pivot to our first demo. The demo is our top priority now that we have most of the design for the project completed. While we are not on track for the prototype's original deadline, we do not believe it should affect our demo's deadline since the majority of the prototype should be able to be used for demo purposes. We are currently on track with our reports and should have no issues getting each of them completed by their due dates.

Math Scheduling System			DURATION (days)
START DATE	END DATE	DESCRIPTION	
10/5/18	10/9/18	Figure out Web Hosting	4
10/5/18	10/19/18	Databases Complete	14
10/5/18	10/31/18	Complete Prototype	26
10/16/18	11/8/18	Full Report #2	22
10/5/18	11/12/18	Finish Demo #1	37
11/8/18	12/4/18	Full Report #3	26
10/5/18	11/12/18	Finish Demo #1	37
11/30/18	12/7/18	Complete Project	7

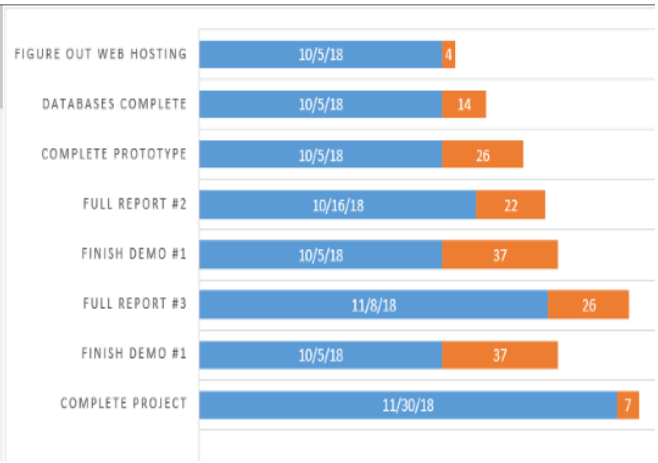


Figure 7: Gannt Chart for Scheduling System

Breakdown of Responsibilities

Team Member	Assigned Task
Brandon Rozzi	Design the tutor management modules
Matthew Mooney	Design the approval module
Trevor Smith	Design the room scheduling module

References