

Status Report 1

Group 6: Connor, Kate, Rachel, and Yumna

RoomWrangler

21 March 2024

1. Introduction

a. Highlights

i. What was the plan for this iteration?

Our main goal was for the product to have basic functionality for a user to log in and reserve a room on our website. To achieve this, our plan focused heavily on setting up our coding environments, fundamentals for our database, and implementing the login functionality. We also developed a very basic pathway for the user to see rooms displayed on the home page, select a room, and launch a modal to create a reservation. We also planned to narrow down what features and needs we wanted to prioritize in our project.

ii. Highlight what the team accomplished.

We set up the framework for our application, including the functionality of room reserving. We added the ability to log into our application with the Google Login API, and restricted access to users with a Davidson email domain. We added the ability to upload a room to the database from the website, including information about the room's features and an image of the room. We added the ability to display all rooms in the database on the home page, and a layout for additional room information when the user clicks on an individual room from the home page. We added the ability to reserve a room from this additional-information page.

Meanwhile, we also met with the registrar to learn more about where we can collect data from, if there are any special rules about certain spaces and whether they had any specific needs they would want to see in such a system. We were able to get insights for their current process and helpful suggestions about how an automated system could look like.

b. Changes

i. Summarize any major changes since the proposal. Include each change's date, motivation, description, and implications. If there were none, note that there were no changes.

1. Customers (2/19): Rather than the Center for Teaching and Learning, we chose the Registrar's Office as a customer to be in contact with (in addition to students and faculty). After contact with the Center for Teaching and Learning, we discovered that AT room reservations are done through the Registrar's Office or EMS (for which the Registrar's Office is responsible). This change meant that we switched gears and set up a meeting with our new customer.
2. Data (2/28): In our proposal, we had planned on importing data about all of Watson for this iteration. As we sat down and planned this sprint after meeting with the Registrar's Office and thinking deeper about the work necessary to complete more pressing goals for this sprint, we decided to put this task on hold until our next iteration. This decision was especially due to the expected (and real) complexity of setting up databases. We knew that our database might not be set up fully until near the end of the sprint, not leaving time to actually fill the database with data. This means that for this iteration, our product will not yet have real-life applicable use. However, our product is functional and ready for real data.

2. Customer-Iteration Goals

a. Describe the product backlog of this iteration.

Based on conversations with students and the Registrar, we determined that this product should support:

- i. Ability to specify a start time and end time and search for a room that is available during that window
- ii. Ability to add desired room features (such as whiteboards, projectors, and computers) as search criteria
- iii. Ability to filter rooms by building
- iv. Ability to specify the type of reservation (such as AT session, Office Hours, or personal use) and grant different priorities based on reservation type
- v. Ability to see existing reservations for individual rooms

We plan to refine the product backlog further during future meetings with T&I and our student customers.

b. Describe the customer's desired overall experience.

- i. Students and professors
 - 1. Want a centralized system for reserving an academic room
 - 2. Want to be able to quickly reserve a room without needing to wait on unnecessary external approvals
 - 3. Want to be able to see available rooms without having to physically check if a room is occupied (knocking, opening doors, listening in)
- ii. Registrar
 - 1. Want to be able to give ATs more autonomy when scheduling a room
 - 2. Would like our app to inform users if a room is available at a specific time instead of having to contact Registrar

c. Describe the sprint backlog of this iteration.

- i. Setup GitHub repo
- ii. Follow tutorial
 - 1. Setup coding environment
 - 2. Setup database environment
 - 3. Create login functionality
 - 4. Ability to upload rooms to database and see them displayed on webpage
- iii. Create basic workflow for reserving a room
- iv. Gather and import data on rooms in Watson

Explanation: Our sprint backlog was based on the sprint goal of implementing basic functionality for a user to log in and reserve a room. The backlog included 5 general tasks: Set up database and basic interactions; create a simple UI to upload rooms; implement login and authentication with Google API; create functionality to select a room information about it, including a Reserve button; create functionality to reserve a room, subsequently uploading this reservation to our database. Other tasks included meeting with the registrar, setting up our coding environment for each member, and

following tutorials to help us understand the technology better. To set up the coding environment, we followed a tutorial on setting up an Airbnb-esque website. We adjusted certain aspects of this tutorial to fit our needs.

d. Explain why you have selected the work items in the sprint backlog for this sprint (or iteration).

For the specific items in our sprint backlog, we wanted to focus on the core functionality of our application: the ability to reserve a room. We made sure to include all work items that were absolutely necessary for this goal.

We limited ourselves to the specific work items above since we knew a lot of our decisions were to be made after meeting with the registrar and realizing the extent of work that needed to be done. For example, we were not sure if group members would have to collect data about each room physically, or if it existed somewhere else already and could be used from there. The registrar showed us it existed in EMS and we could try using data from there. We still would have to figure out how to scrape data from EMS, so we decided not to include that in our first iteration. Meanwhile, just understanding and working with the tutorials to get started with our project was time consuming (given the tutorial is around ~8 hours long) so it made sense to limit how much real-world functionality we could see by the end of sprint 1.

e. Use Cases:

- i. Write a use case for each main user goal for a primary or secondary customer. Show each use case's title, user goal, and full basic flow. Choose meaningful titles. Each use case should have at least one specific alternative flow and one bounded alternative flow.**

Name: Login

Goal: A user logs in to the system using their Davidson Google account

Actors: User, RoomWrangler system

Basic Flow: <ol style="list-style-type: none">1. System displays a “Log In with Google” button2. User clicks the “Log In with Google” button3. System redirects to Google Login {Redirects to Google Login}4. Google Login prompts user to enter their email address5. User enters email address {Verifies a valid Davidson email}	Alternative Flows: <i>Specific Alternative Flow:</i> At {System finds user’s account} , if account does not exist, the system creates a new account associated with the Davidson Google account used in the login process. Resume basic flow at {User is logged in} <i>Bounded Alternative Flow:</i> Currently, no bounded alternative flow has
---	---

<p>address}</p> <ol style="list-style-type: none"> System redirects to Davidson College sign in Davidson College sign in prompts user to enter email address User enters email address {Verifies email address exists within Davidson College} System prompts user to enter password User enters password {Verifies password is correct} Davidson College sign in system sends a Duo push to the user User authenticates using Duo {System finds user's account} System returns to home page with the user logged in {User is logged in} 	<p>been implemented. In the future, we would like to make it such that if the connection to the google API has been lost at any point between {Redirects to Google Login} and {System finds user's account}, the user is redirected to the home page and given an appropriate error message prompting them to attempt another log in.</p>
---	---

Name: Upload Room

Goal: A user creates a new room to be stored in the system

Actors: User; RoomWrangler system

<p>Basic Flow:</p> <p>{System displays home page}</p> <ol style="list-style-type: none"> User selects "Upload a Room" System opens page to select room features User selects room features User clicks the "Next" button System opens page to select the building the room is in User selects the correct building User clicks the "Next" button System opens a page to input room information User inputs correct information User clicks the "Next" button System opens a page to upload or find a picture of the room User enters a picture of the room User clicks the "Create" button 	<p>Alternative Flows:</p> <p><i>Specific Alternative Flow:</i></p> <p>At {System verifies valid input in required fields} if any required fields are missing information, system displays a warning message and does not add room to database. Resume the basic flow at {Home page}</p> <p><i>Bounded Alternative Flow:</i></p> <p>Currently, no bounded alternative flow has been implemented. At some point in the future, however, we would like to have added functionality so that if at any point between {System verifies valid input in required fields} and {Home Page}, if connection to the database has been lost and the new room isn't properly rendered on the page, the user receives an appropriate message to inform</p>
--	--

<p>{System verifies valid input in required fields}</p> <p>14. Room is added to database</p> <p>{Home page}</p> <p>15. System returns to home page with newly added room.</p>	<p>him or her that the database is down.</p>
---	--

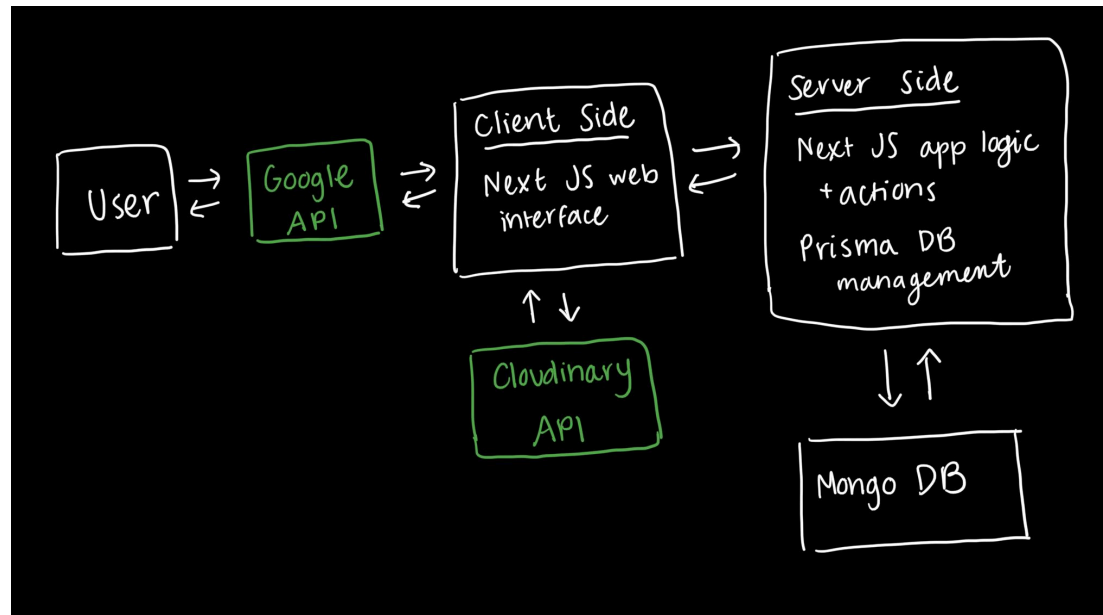
Name: Reserve a Room

Goal: A user makes a room reservation

Actors: User; RoomWrangler system

<p>Basic Flow:</p> <p>{Room page}</p> <ol style="list-style-type: none"> 1. User clicks the “Reserve” button 2. System prompts user to enter start and end time of reservation 3. User enters start and end time and clicks the “Reserve” button <p>{Verify required fields are filled}</p> <ol style="list-style-type: none"> 4. System displays “Reservation created” <p>{Store reservation in database}</p> <ol style="list-style-type: none"> 5. User receives message informing him or her of a successful reservation. <p>{Redirected to Homepage}</p>	<p>Alternative Flows:</p> <p><i>Specific Alternative Flow:</i></p> <p>At {Verify required fields are filled}, if required fields are not filled, prompt user to enter start and end times.</p> <p>Resume basic flow at {Verify required fields are filled}</p> <p><i>Bounded Alternative Flow:</i></p> <p>Currently, no bounded flow has been implemented. There would be the potential, however, for two people to try to reserve a room for the same time at once. If someone else reserves a room at your desired time at any point between {Store reservation in database} and {Redirected to Homepage}, then the user should be informed that his or her time is taken and prompted to attempt a new reservation at a different time.</p>
---	--

3. System Description



- a. Draw a block diagram to show how the proposed system will interact with external services, databases, etc. Clearly mark the boundaries of the system. Use the above diagram to introduce the system. What are the main elements of the proposed system?

The main elements of the system that we are responsible for implementing are the client side and server side components. We are building our client side web components with Next.js. Our server side components include Next.js hooks for handling logic between our web components, which allows us to keep our components loosely coupled. We are also using prisma to handle interactions with our data layer, which is hosted in MongoDB.

4. Current Status

- a. Summarize the current implementation status of your system

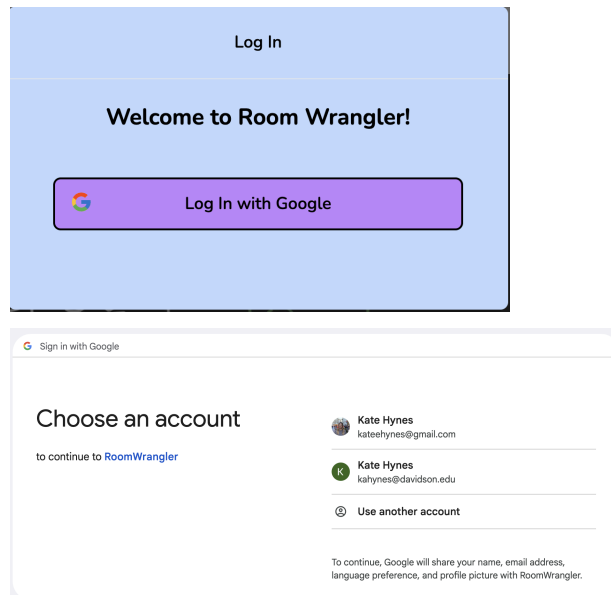
We have a modal prompting the user to login and authenticate with Google API. We automatically create accounts if the user doesn't exist in our database, and we only support emails with @davidson.edu. We have the database set up, although we are currently still adjusting the schema. After logging in, the user lands on our application home page. This home page includes a navigation bar component that includes a button which launches a modal to upload rooms, including cloudinary API to upload images. On the home page, the user can see a layout of all the rooms currently in our database. From this page, the user can click on an individual room to open a page with more information about that room, including a button to launch a modal for creating a reservation.

- b. Screenshots

For this section, we included the major elements of our UI as separate parts. Since the behavior of each of our hook, action, and route components is extremely similar, we only gave in depth explanations of one hook, one action, and one route.

- i. Add the screenshot(s) of the system's working part(s). Explain the screenshot, i.e., explain

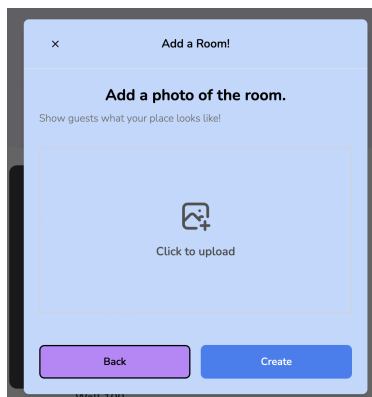
1. Login Modal



- a. What you are trying to show in the screenshot
 - i. This web component is responsible for the login functionality.
- b. Which part in the system description diagram the part belongs to
 - i. Google API & Client-Side Components
- c. What is the behavior of the system
 - i. You can see that there is a button labeled “Log In with Google” that when clicked leads to the Google Login. It does this by accessing the Google Login API.
- d. Why is this system important

- i. If a user reserves a room, the system must know *who* the reservation is for. We could simply have users input a name at the time of reservation, instead. However, the login functionality ensures that only Davidson students and faculty members can reserve rooms. It also sets the stage for later features such as viewing upcoming reservations and types of profiles for users with varied reservation permissions.

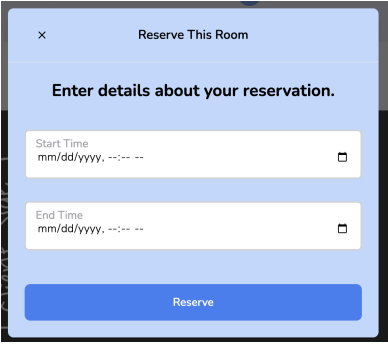
2. Upload Room Modal



- a. What you are trying to show in the screenshot
 - i. This screenshot shows the last screen of our modal that allows a user to upload a room to the database. This screen prompts the user to upload an image, a service which is provided by Cloudinary API.
- b. Which part in the system description diagram the part belongs to
 - i. Client-Side Components & Cloudinary API
- c. What is the behavior of the system
 - i. This component prompts the user to enter information about the room, including the room's building and features, and an image of the room.
- d. Why is this system important
 - i. This component allows users to quickly and easily upload information about a room to the database.

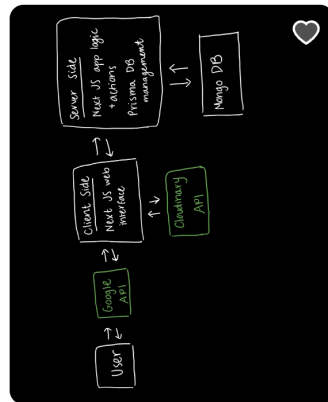
This is especially useful for our group as we attempt to build our reservation system, and could benefit system administrators who need to import information about rooms to our database. Including an image for the room will allow users to get a better understanding of the room that they are trying to reserve.

3. Reserve Modal

A screenshot of a mobile application modal titled "Reserve This Room". The modal has a light blue background and a white border. At the top, there is a close button (X) and the title "Reserve This Room". Below the title, the instruction "Enter details about your reservation." is displayed. There are two input fields: "Start Time" with a placeholder "mm/dd/yyyy, --:-- --" and "End Time" with a placeholder "mm/dd/yyyy, --:-- --". Both fields have a calendar icon on the right. At the bottom, there is a blue button labeled "Reserve".

- a. What you are trying to show in the screenshot
 - i. This screenshot shows our modal that allows a user to create a reservation for a room in our database.
- b. Which part in the system description diagram the part belongs to
 - i. Client-Side Components
- c. What is the behavior of the system
 - i. This component prompts the user to enter a start and end time for the reservation after the user clicks a button to launch the modal. This component is currently housed on an individual room's Room Information page, which populated the room field on the data entry created for the reservation.
- d. Why is this system important
 - i. This component allows users to quickly and easily reserve a room, which is the core functionality of our project.

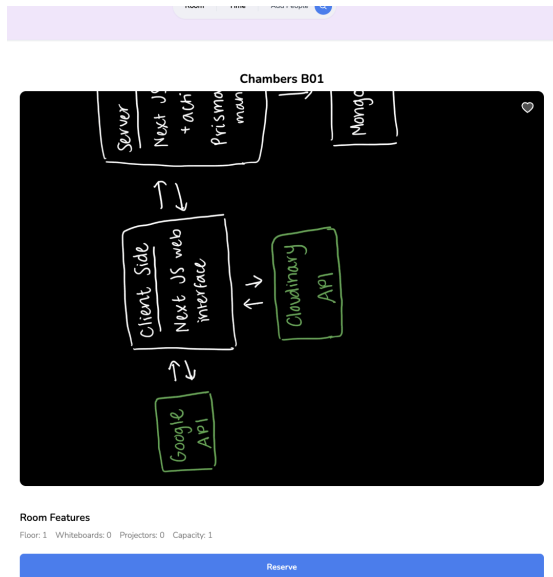
4. Room Card



Chambers B01

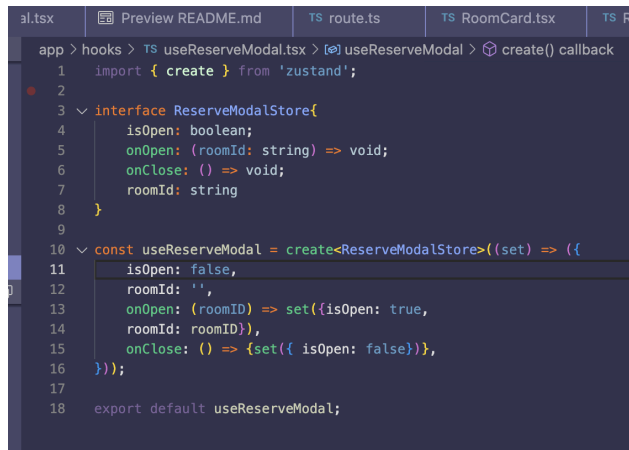
- a. What you are trying to show in the screenshot
 - i. This screenshot shows the component that displays on our application's home page to represent an individual room in our database.
- b. Which part in the system description diagram the part belongs to
 - i. Client-Side Components
- c. What is the behavior of the system
 - i. Currently, this component will render for each room in our database. The user can click on this component to open up a Room Information page, which will display more information about the room, including its features and a button to launch the Reserve modal.
- d. Why is this system important
 - i. This component allows users to quickly and easily see available rooms and to open up a page where they can view more information and reserve the room. This component will eventually be used to show rooms that are available and match the user's search criteria in a more condensed layout.

5. Room Information



- a. What you are trying to show in the screenshot
 - i. This screenshot shows the Room Information page, which includes the room's features and the Reserve button to launch the Reserve modal.
- b. Which part in the system description diagram the part belongs to
 - i. Client-Side Components
- c. What is the behavior of the system
 - i. This component displays more information about the given room after the user selects a Room Card.
- d. Why is this system important
 - i. This component allows users to view more information about an individual room, and houses the Reserve functionality for an individual room.

6. Hooks



```
al.tsx | Preview README.md | TS route.ts | TS RoomCard.tsx | TS R
app > hooks > TS useReserveModal.tsx > useReserveModal > create() callback
1 import { create } from 'zustand';
2
3 interface ReserveModalStore {
4   isOpen: boolean;
5   onOpen: (roomId: string) => void;
6   onClose: () => void;
7   roomId: string
8 }
9
10 const useReserveModal = create<ReserveModalStore>((set) => ({
11   isOpen: false,
12   roomId: '',
13   onOpen: (roomId) => set({isOpen: true,
14     roomId: roomId}),
15   onClose: () => set({isOpen: false}),
16 });
17
18 export default useReserveModal;
```

- a. What you are trying to show in the screenshot
 - i. This screenshot shows one of our hooks: the useReserveModal hook.
- b. Which part in the system description diagram the part belongs to
 - i. Client-Side Components
- c. What is the behavior of the system
 - i. Importing a hook for a component allows other components in our application to consume that component.
- d. Why is this system important
 - i. Using hooks allows our system to retain a modular design, and keeps coupling between components loose.

7. Routes

```

app > api > rooms > TS route.ts > POST
16 // left category here but it's not actually getting
17 const body = await request.json();
18 const {
19   building,
20   number,
21   buildingAndNumber,
22   floor,
23   imageSrc,
24   category,
25   capacity,
26   whiteboards,
27   computers,
28   projectors,
29 } = body;
30
31 Object.keys(body).forEach((value: any) => {
32   if (!body[value]) {
33     NextResponse.error();
34   }
35 });
36
37
38 const room = await prisma.room.create({
39   data: {
40     building,
41     number,
42     buildingAndNumber,
43     floor,
44     imageSrc,
45     capacity,
46     whiteboards,
47     computers,
48     projectors,
49   }
50 });

```

- a. What you are trying to show in the screenshot
 - i. This screenshot shows one of our routes, the reservation route.
- b. Which part in the system description diagram the part belongs to
 - i. Server-Side Components
- c. What is the behavior of the system
 - i. Our route components allow us to communicate between our application and database. Our web components can make a request to a route component, and the route components use prisma, a database toolkit that directly interacts with the database, to process and send the request to the database.
- d. Why is this system important
 - i. The routes allow for communication between our application and database.

8. Actions

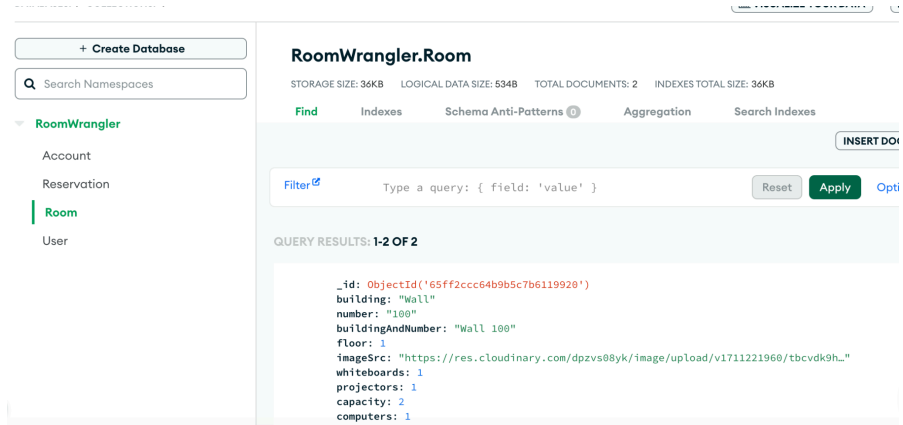
```

app > actions > TS getRoomById.ts > *o IParams > roomId
1  import prisma from '@app/libs/prismadb';
2
3  interface IParams {
4    roomId?: string;
5  }
6
7  export default async function getRoomById(
8    params: IParams
9  ) {
10   try {
11     const { roomId } = params;
12
13     const room = await prisma.room.findUnique({
14       where: {
15         id: roomId
16       },
17     });
18     if (!room) {
19       return null;
20     }
21
22     return room;
23   }
24   catch (error: any) {
25     throw new Error(error);
26   }
27 }

```

- a. What you are trying to show in the screenshot
 - i. This screenshot shows one of our actions: getRoomById.
- b. Which part in the system description diagram the part belongs to
 - i. Server-Side Components
- c. What is the behavior of the system
 - i. The action components use prisma to request data from the database and process the data before sending it to the web components.
- d. Why is this system important
 - i. The action components allow us to get data from the database into our application.

9. MongoDB



- a. What you are trying to show in the screenshot
 - i. This screenshot shows one of our data models: the Room model. It shows the way information related to our Room object is stored for an example room.
- b. Which part in the system description diagram the part belongs to,
 - i. MongoDB
- c. What is the behavior of the system
 - i. This part of our system is responsible for storing data about Users, Accounts, Rooms, and Reservations that is then pulled in by our server-side components and passed to our client-side components. Our group members can modify entries directly in the database.
- d. Why this system important.
 - i. This part of our system stores all of our data about rooms and reservations, which is the core functionality of our application.

c. Tests

- i. **List the tests you performed for this iteration's implemented parts.**
 1. We performed acceptance testing on the login functionality.

- a. Attempted logging in with a non-Davidson account to ensure that user was prevented from accessing the application
 - b. Attempted logging in on an incognito browser with a Davidson account to ensure user was still able to log in with Davidson account through Google API
 2. We performed acceptance testing on uploading a room to the database functionality.
 - a. Attempted uploading rooms with all fields filled out in the modal, and ensured that a Room entry was successfully created with all the correct information.
 - b. Attempted uploading rooms without filling out required fields in the modal, and ensured that the warning message appeared and creation of this room was prevented until the user successfully filled out all required fields.
 3. We performed acceptance testing for creating new buildings when uploading a room.
 - a. Upon completion of the room uploading process, if a new building is selected that hasn't previously been created, it is automatically created and uploaded to the database by our software.
 4. We performed acceptance testing on making reservations
 - a. Attempted clicking "reserve" on a room and ensured that a page popped up to input start and end time of reservation.
 - b. On the reservation page, attempted to finalize reservation without inputting start and end time and ensured that the system would not create the reservation.
 - c. Attempted to create a reservation with all required fields filled and ensured that the reservation was stored in the database.
- ii. From the acceptance tests in the proposal, explain your plan to test them technically. In other words, how would you automatically test your acceptance tests? What are the inputs to the tests and outputs from the tests? How would you determine whether the test was passed or failed?

1. We conducted the acceptance tests from the proposal that apply to our goals for this iteration. In this iteration, we did this simply by acting as a user, performing the actions, and recording the results. If the system reacted in the way that we had initially declared we wanted it to, then it passed the test. We did not automate any of our acceptance tests for this iteration.

5. Project Management

- a. **Continue to maintain the Change Log. Add any new changes to the project, tracking the date and description of each change. Use the table below:**

Date	Description
2/13	Had initially wanted our system to work for all academic buildings. As we gained a clearer picture of what can be accomplished by four college students within a semester, we decided to limit the scope of our project to only one academic building, with potential for adding more if the project continues after the semester is over.
2/28	Decided we want ATs to schedule through us. Initially, we were not sure if we wanted our product to handle AT scheduling. When we met with the Registrar's Office, they told us that they would like this.
3/21	Decided to discuss possibilities for putting our product into production with T&I. Initially, we weren't sure if we wanted RoomWrangler to be pushed to production, but now we think we may have time to incorporate that into our plans and can pursue this aspect.

6. Review and Retrospective

- a. **What went well?**

We accomplished the goals that we set out for this sprint. We also were flexible about times and methods for group meetings: we met sometimes in person and sometimes over zoom, and we were able to coordinate schedules well to accommodate these meetings. Our communication about these meetings went smoothly.

b. What didn't go well?

Setting up the coding environment and learning about the technology was more intensive and time consuming than we had expected. Additionally, the tutorial we followed to set up our website is, of course, not perfectly tailored to us. We ran into parts of the tutorial that were not necessary for our project or that we needed to modify for our needs. Although this was expected, it was still an obstacle to deal with throughout this iteration.

c. For the goals that were not met, what were the issues?

- i. Database still not yet finalized, as we're still ironing out exactly what details make the most sense based on what the customers want.
- ii. We did not add all rooms in Watson to our database as our database is still not yet finalized.
- iii. Did not implement search for a room functionality, as we decided to prioritize other features/ set up details at the time.

d. How do you plan to overcome the issues?

- i. As T&I gives us more input on what fields are necessary, we will continue to update them. Thankfully, the combination of Prisma and MongoDB make it fairly straightforward to modify the database as we continue development, so this shouldn't be a terribly difficult obstacle to overcome.
- ii. This goes hand in hand with the first point. Due to our changing database, we didn't want to add all the data in yet in case we had to make significant changes later. We plan to do this as soon as our database schema has been finalized.
- iii. Search functionality will be our main priority going forward as we are now done addressing other higher priority features.

e. What do you plan to do differently in the next iteration?

We will better gauge the amount of time tasks will require. Luckily, we are done with the initial setup, tutorial, and getting used to the technology. So, the time commitments for the rest of the tasks should be more predictable. We will also each be responsible for proactively practicing and/or learning about the technologies that we are utilizing if needed: if any team member is unsure about any of the technologies that they will be using, they should learn/practice with it or ask other team members about it to ensure that they can contribute in a timely manner and that learning technology is not obstacle for us going forward.

7. Team Management

a. What were the team roles for this iteration?

Kate - Developer

Connor - Developer

Rachel - Product Owner, Developer

Yumna - Scrum Master, Developer

b. What did each team member contribute?

Kate: Database setup, following tutorial, planning reserve workflow and developing custom components

Connor: Set up coding environment, integrating APIs, following tutorial for basic foundation and developing custom components

Rachel: Met with Registrar; Drafted Product Backlog; Conducted testing

Yumna: Met with Registrar; Drafted Sprint Backlog; Conducted testing

c. What were the challenges regarding team management, e.g., regular meeting, etc.?

We did not specifically designate certain tasks to certain people or come up with deadlines for any tasks. This resulted in some uncertainties about generally who was doing what and when. Work tasks also ended up unevenly distributed. It was also difficult to coordinate schedules for meetings, but we managed to make that work.

d. What are the plans to overcome the challenges?

Going forward, we will assign certain tasks to certain people ahead of time. We will also come up with soft and hard deadlines for tasks.

e. If you were the third party who knows very well about your team, what suggestions would you give to your team?

- Consider each person's areas of experience and knowledge when sprint planning.
- Ask another member for help if you are struggling with a task.
- Stay on top of the work!

8. Goals for Next Iteration

a. Write the next iteration's product log.

- i. Search for a room based on certain criteria including time window availability and features
- ii. Creating a calendar view to see availability for each room.
- iii. Create a reservation system that is functional for Watson.

b. Write the next iteration's sprint log.

- i. Add ability to search for a room by reservation time window and filter by room features
- ii. Finalize data models
- iii. Setup meeting with T&I
- iv. Setup additional meetings with student(s) who will be our student customers
- v. Create a Calendar component
 - 1. Create functional calendar with no data based on design
 - 2. Get calendar to pull in Reservation data from in-app created reservations and any external data
- vi. Collect data about Watson rooms and standing reservations

c. Other than the issues discussed in Section 6, i.e., Review and Retrospective, what potential challenges do you see in the next iteration?

Some potential challenges in this iteration are unexpected challenges that may come with developing without heavy guidance. We followed a tutorial for most of our setup in sprint 1. Now we will be working on our own for the rest of the project. Although the tutorial greatly helped us to get acquainted with the technologies, we still are entering new territory in which we will be figuring out how to use the technologies to best suit our needs and achieve our goals. We will also be writing a lot more code from scratch, which brings potential for more bugs.

d. Briefly explain how your team would overcome each of the mentioned challenges.

We will be sure to make a plan before beginning to code, for example assigning tasks to team members. We will also conduct unit testing frequently to fix bugs as early as possible and

determine whether it can be solved within our current sprint or may need to be deferred to a later sprint. We are also prioritizing our tasks/ features we want to see beforehand, so that it is easier to drop them if needed in case time runs out.