

Purdue Course Finder

Design Document

Team 9 - CS 307 Fall 2022

Alex Kobus

Alex Plump

Tommy Lane

Peter Zong

Index

• Purpose	3
○ Functional Requirements	3
○ Non-Functional Requirements	6
• Design Outline	6
○ High-Level Overview	6
○ Components	7
• Design Issues	8
○ Functional Issues	8
○ Non-Functional Issues	9
• Design Details	10
○ Class Diagrams	10
○ Sequence Diagrams	12
○ Activity Diagrams	15
○ UI Mockups	16

Purpose

Many students see their schedule at the beginning of the year and must go searching for their classes. Making this process harder, Google Maps often doesn't recognize building acronyms, making it even more difficult to find classroom locations. Our product will eliminate that issue by providing a map of Purdue which outlines buildings, classrooms, the courses in those classrooms, the time slots of those courses, and more. This map will also be searchable using the names or acronyms of buildings, or by using course numbers or titles. This will help students to find exactly where their classes are, find more information about the classrooms and buildings, and see the locations of other sections for their courses.

Currently, the only method of looking up classes is going to the Purdue course look-up page. Although this provides users with information about courses and who teaches them, the location is an abstraction. For example, room BHEE 129 is a well-known room for many upperclassmen, but for new students, this might be a challenge to navigate to. Our software provides a map and building layout so students can visually see exactly where their classes are and don't wander around on the first few days of the semester.

Functional Requirements

1. Usage

As a User,

- a. I would like to see a tutorial that explains how to use the software.
- b. I would like to see an option to select a specific semester to view the courses offered that semester.

2. Map

As a User,

- a. I would like to see a birds-eye view map of Purdue.
- b. I would like to search the map for a building's location.
- c. I would like to see a sidebar that shows a list of all classes and buildings.
- d. I would like to see campus buildings highlighted on the map.
- e. I would like to see campus buildings labeled on the map
- f. I would like to be able to click a highlighted building on the map to see more information about it, the rooms in it, and the classes in it.
- g. I would like to filter the buildings that I see highlighted on the map by name.

- h. I would like to filter the buildings that I see highlighted on the map by class location.
- i. I would like to filter the buildings that I see highlighted on the map by section location.

3. Account

As a User,

- a. I would like a dedicated page for signing up.
- b. I would like a dedicated page for logging in.
- c. I would like to create an account.
- d. I would like to delete an account.
- e. I would like to change my account email.
- f. I would like to change my account password.
- g. I would like the login data to be encrypted.

4. General Filter

As a Student,

- a. I would like to filter the map and sidebar to show only relevant courses and course locations.
- b. I would like to filter the map and sidebar to show only relevant sections and section locations.
- c. I would like to filter the map and sidebar to show only relevant buildings.
- d. I would like to filter the map and sidebar to show only relevant classrooms.
- e. I would like to filter the map and sidebar to show only relevant meeting times.
- f. I would like the option to manually input parameters to search for a course/building in case the filters return no results.

5. Filter by Favorites

As a User,

- a. I would like to create a list of my favorite buildings and classrooms.
- b. I would like to create a list of saved (favorite) classes and sections.
- c. I would like to filter the map by my favorite buildings and rooms.
- d. I would like to filter the map by my favorite classes and sections.

6. Personal Schedule

As a Student,

- a. I would like to create a personal schedule for all my classes in a week.
- b. I would like to see the estimated time it takes to walk between my classes
- c. I would like to see the estimated time it takes to bike between my classes

7. Courses and Classrooms

As a User,

- a. I would like to see current Purdue courses, and their section times, professors, and locations.
- b. I would like the sidebar to show classrooms in a building after selecting that building.
- c. I would like to select a classroom from the sidebar to see more information about it and the classes in it.

As a Student,

- d. I would like to see a schedule page that shows all class meetings in a selected room each week.
- e. I would like to see a schedule page that shows all meetings of a selected class in each week.

As a Professor,

- f. I would like to see the number of seats in a classroom.
- g. I would like to see the number of students registered in a section.
- h. I would like to see statistics that show how often a classroom or building is used. (if time allows)

8. Miscellaneous

As a User,

- a. I would like to see a suggestion page for future iterations (if time allows).

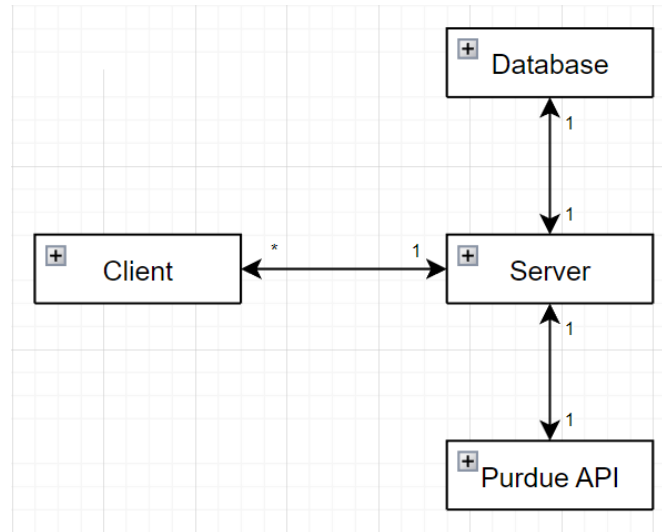
Non-Functional Requirements

1. As a User, I would like the application to be easy to use.
2. As a User, I would like the application to be fast.
3. As a User, I would like the application to be secure.
4. As a Developer, I would like to store Purdue course info from Purdue's API in a SQL database.
5. As a Developer, I would like to save backend logs of what pages are accessed most frequently.
6. As a Developer, I would like to save encrypted account login information in a database.
7. As a Developer, I would like to employ a front-end using a React.JS web application.
8. As a Developer, I would like to employ a back-end using Spring Boot.
9. As a Developer, I would like to restrict schedule creation specifically to users that are logged in.
10. As a Developer, I would like to optimize the web application, so that response times are lower than 1 second.
11. As a Developer, I would like to optimize the web application, so that page loading times are lower than 500 ms.
12. As a Developer, I would like the application to be hosted on AWS.
13. As a Developer, I would like to create the application in a way that allows it to be adapted to include other universities.
14. As a Developer, I would like to have developer roles on accounts for testing and any account resolutions (like resetting passwords). (if time allows)

Design Outline

High-Level Overview

Our project is a web application that will display information about Purdue courses in a more visual and intuitive way than is currently available. Many students cannot understand the complex building naming and alternatively find classrooms within those buildings. Our solution is to show a birds-eye map of Purdue, and display building information on that map that can be interacted with. We will use an API to pull data from the Purdue Course Registry to populate a database. Using a client-server model, we will send this information to users upon request.



Components

1. Client

- React.JS Frontend.
- User accessed webpage that presents users with the option to login or be anonymous.
- Client will display a UI map that users can move around on.
- Client will send requests to update map through filtering and searching.
- Data the client displays will come from the server.

2. Server

- Spring Boot Backend.
- Server will handle requests from the Client.
- Server will query the database in order to access information upon login requests.
- Server will utilize API to access course information upon user request to update the UI.
- Store logs on what pages are accessed by users to record traffic.

3. Database

- A relational SQL database that will hold account information in a table.
- Information includes user's account username, password, email, phone, and name.

- c. Database may be used as a cache feature for highly requested queries.
- d. Information will be provided to and gathered from the server specifically.

4. API

- a. An API that accesses Purdue's course registration for a given Location, i.e., West Lafayette, Fort Wayne, etc.
- b. Ability to access course information by semester.
- c. Labels data types to create relationships between courses, sections, rooms, buildings, and instructors.
- d. This will be consumed by the server.

Design Issues

Functional Issues

1. Collecting User Information
 - a. Option 1: Store information about Users regarding email, IP address, and pages accessed
 - b. Option 2: Store information about Users regarding email.
 - c. Option 3: Store no information about Users.
 - d. Solution: Store information about Users regarding email, IP address, and pages accessed
 - e. Reasoning: Although it is an ethical practice to avoid collecting excessive information about users, the decision was made to collect the basic information for user accounts as well as some of the logs associated with users (and unauthenticated IP addresses). This was done with intent to prevent harm which justifies further collection. If a user has a problem with their account being compromised or otherwise describes a problem, having logs of what happened as well as the user associated with the logs will help with diagnosing and repairing problems.
2. Is a user account system necessary?
 - a. Option 1: Yes
 - b. Option 2: No
 - c. Solution: Yes
 - d. Reasoning: We would like users to have the ability to store favorites. These allow the users to access the website repeatedly and immediately have quick access to certain buildings or sections (that likely comprise their personal schedules). The

best way to do this would seem to be a user account system which would allow the users to access this information from any computer on campus by remembering their login information.

3. How should buildings and classrooms be shown?
 - a. Option 1: Building overlays on a map
 - b. Option 2: A sidebar
 - c. Option 3: Both
 - d. Solution: Both
 - e. Reasoning: Although the idea of showing building information (such as the abbreviated name) on a map seems helpful, there is more that would need to be seen, and often simply staring at a map is not the best way to find a building. Therefore, in addition to the map, a sidebar will be implemented which will give a sorted list of buildings with an ability to find specific classrooms.

Non-Functional Issues

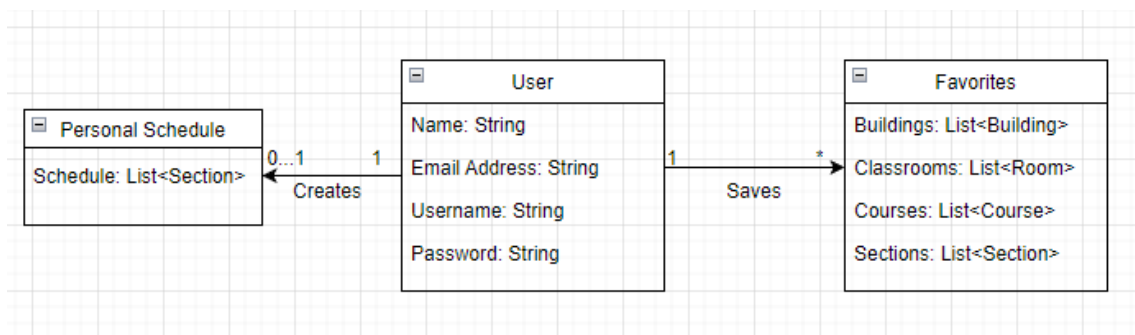
1. What framework should we create our front-end in?
 - a. Option 1: React
 - b. Option 2: Flutter
 - c. Solution: React
 - d. Reasoning: Not only does our team have more experience with React, but React is more mature than Flutter as a web framework with lots of support and information available should we need it.
2. How will we host our backend?
 - a. Option 1: AWS (or another cloud provider)
 - b. Option 2: Private Server
 - c. Solution: AWS (or another cloud provider)
 - d. Reasoning: Although using a computer owned by a team member for hosting our backend would likely be the simplest solution, we believe that cloud hosting would be a better solution. The application will be more reliably managed and our team has had some exposure to AWS services in the past.

3. What type of database should we use?
 - a. Option 1: SQL
 - b. Option 2: NoSQL
 - c. Solution: SQL
 - d. Reasoning: NoSQL databases seem to be getting more popular so it was a discussion we needed to have. Ultimately SQL was decided upon because it was familiar to our team, and we saw no major benefit of going with NoSQL. There are benefits to NoSQL such as the lack of a rigid schema system but our application will have no use for this as we know what kind of data we will be storing beforehand.

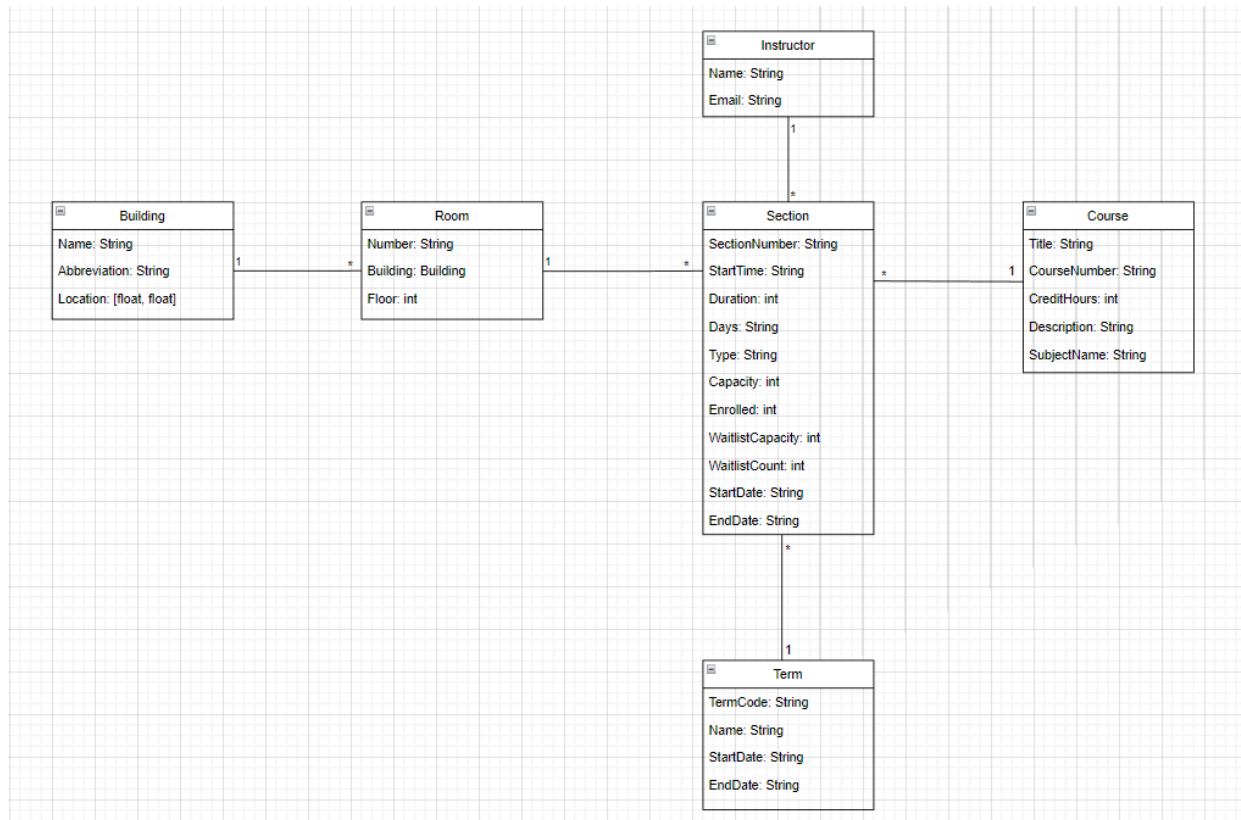
4. How will the database be used to store course information?
 - a. Option 1: Copy all information from the API at once into the database
 - b. Option 2: Move the information as needed
 - c. Option 3: Call the API on every request, don't use a database for this
 - d. Solution: Move the information as needed
 - e. Reasoning: Our discussion began because we wanted a way to avoid hitting the API repeatedly and forming an overreliance on it. One of the first ideas we had was to simply move all information from the API at once with a script. This seemed to in some ways go against one of our intentions of not overloading the API, so we opted for the slower method of using the database as more of a long-term caching solution to avoid relying on the API more than necessary.

Design Details

Class Diagrams



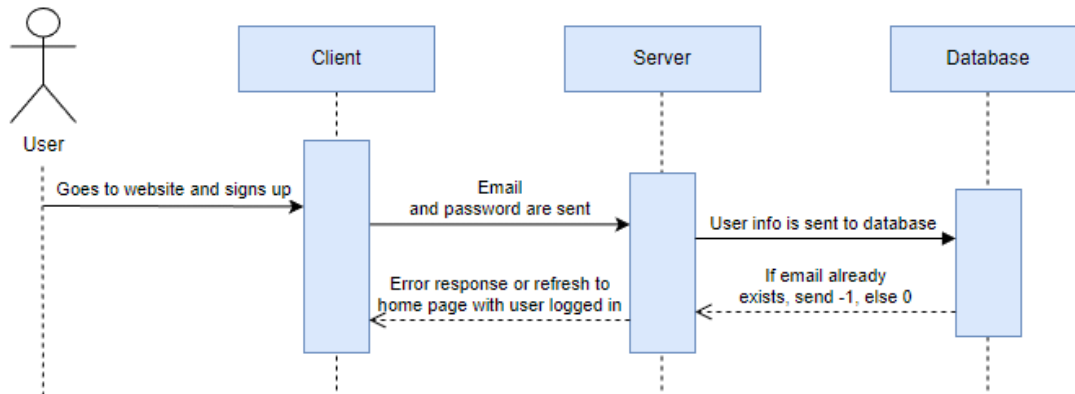
- The account that students and professors can create will contain information about their name, username, and email address. These accounts also contain optional functions that allow users to favorite things such as buildings and courses. Users can select the courses they are enrolled in and filter the map with the respective buildings and sections.



- Each course offers a different number of sections in each term depending on a few factors, including how many students are enrolled. Sections have a variety of fields describing when lecture time or labs start, what days the section takes place, how long meetings are, how many students are enrolled in a section, and waitlist information. Sequentially, every course at Purdue has multiple sections proportional to however many students are in the course. Each of these sections has a corresponding instructor, usually a TA, who holds a name and email address.
- The buildings at Purdue all have different abbreviations commonly used to describe them as well as a longitude/latitude location. Each building has several rooms within it which each can be described by the building they are within, the room number, and the floor the room is on within the building. Also, each room will correspond to some number of sections which will take place within the room.

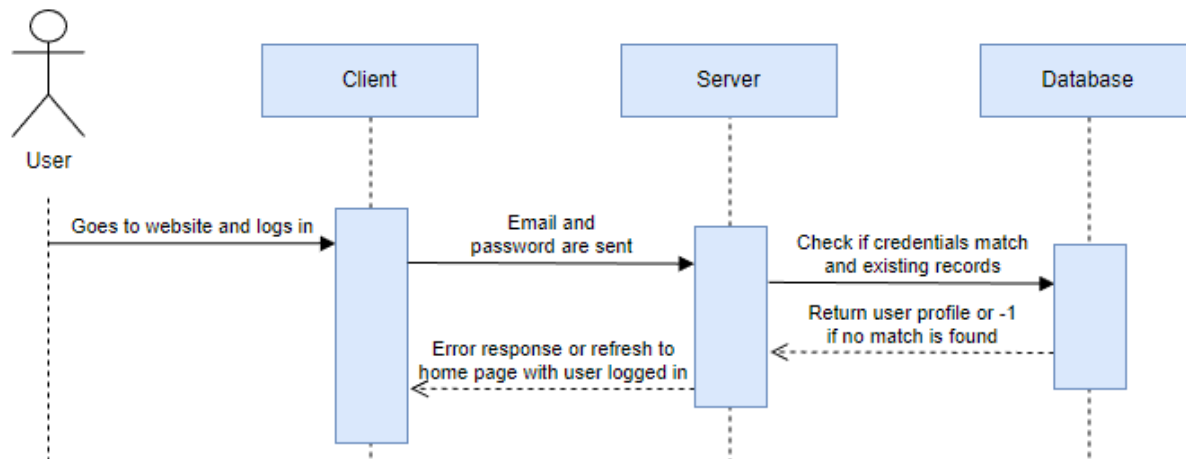
Sequence Diagrams

1. Sequence of events when user signs up



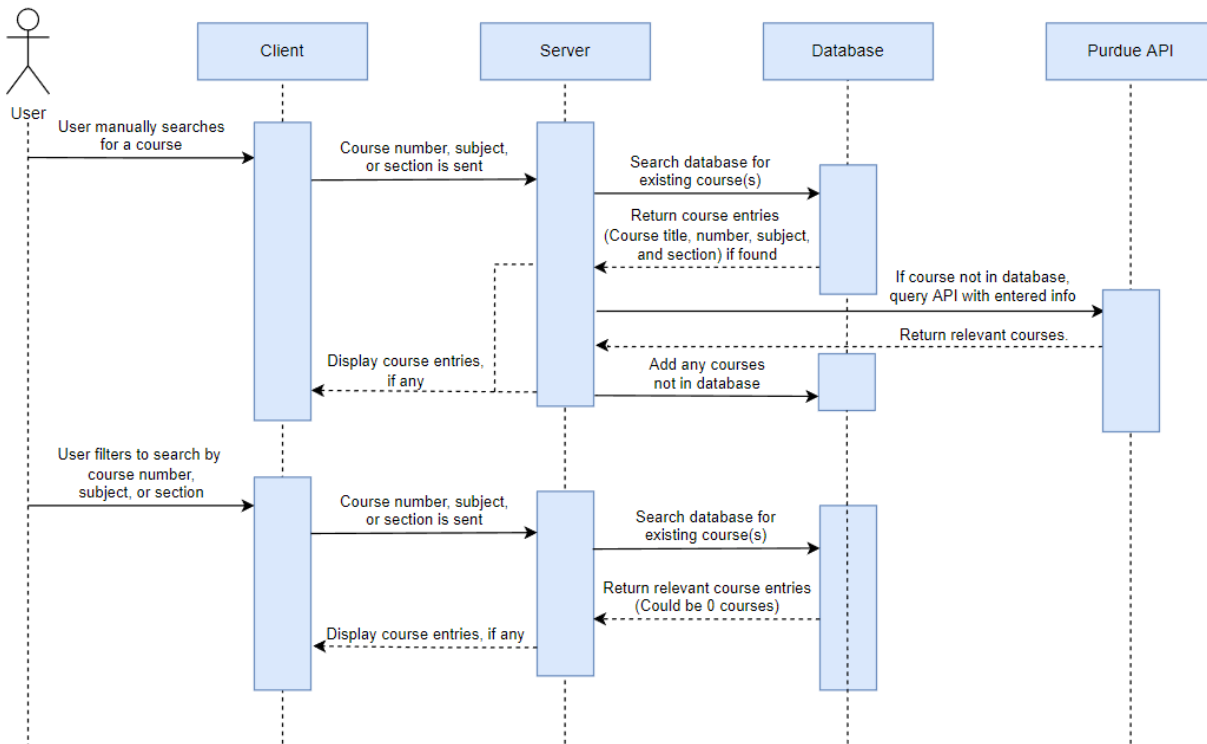
When a user signs up for an account, the email and password are sent to the server, which checks if the email already exists or not in the database. If it doesn't exist, a new account is created and the user is redirected to the home page being logged in. If the email already exists, the server sends back a message telling the user that the email is already in use.

2. Sequence of events when user logs in



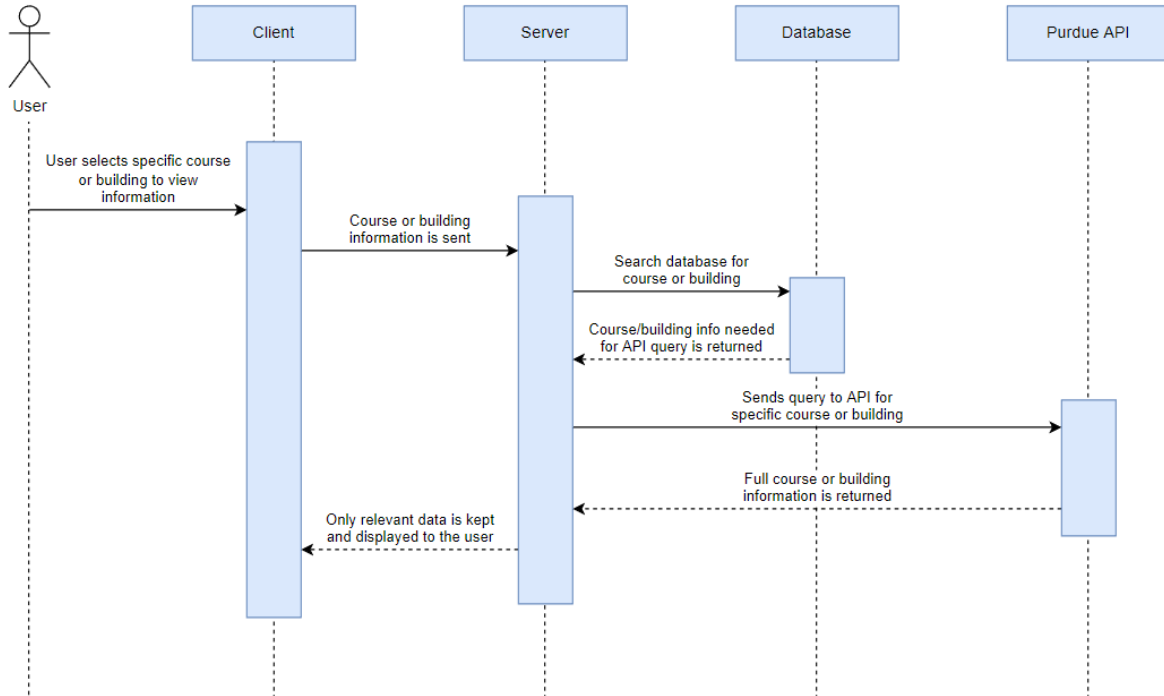
When a user logs in, the email and password are sent to the server, which checks if the email and password match an existing record in the database. If the email-password combination is found, then the user profile data is sent back to the server and the user is redirected to the home page being logged in. Otherwise, the server sends back a message telling the user that the email/password is incorrect and to try again.

3. Sequence of events when user searches for a course either manually or through filters



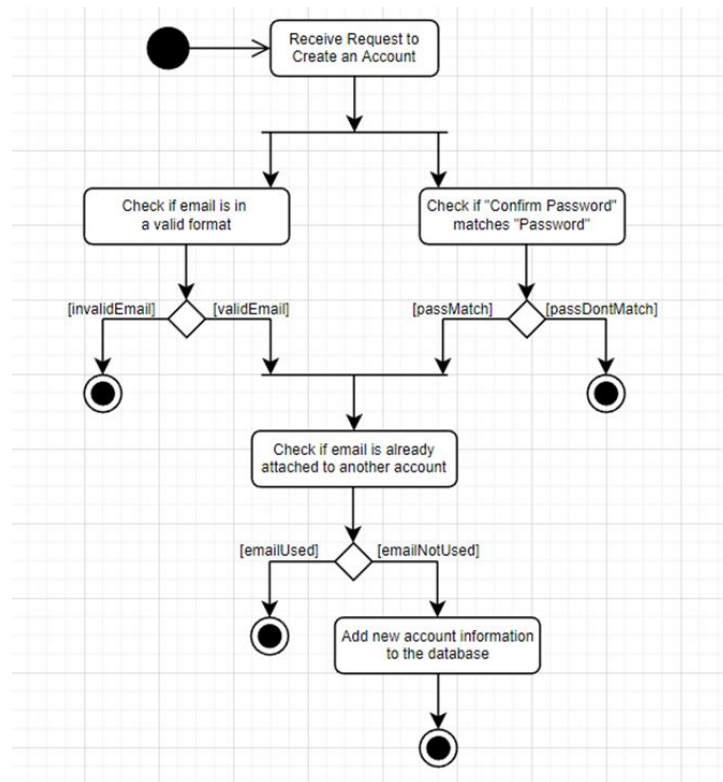
When a user searches for a course, they have the option to either use filters to narrow down their search or manually enter course info if the course isn't found. When the filters are applied, the filter data is sent to the server, which filters the database for relevant courses. If any courses are returned, they are then displayed to the user. If the course the user is looking for doesn't show up, they have the option to manually add course details and then hit search, which first checks the database to see if the course exists, and if not, queries the Purdue API with the course parameters. Relevant courses are then returned to the server, and any courses not already in the database are then added to it. Finally, these relevant courses (if any), are displayed to the user.

4. Sequence of events when user selects a building or course to view more information



After a user creates an account, on the map they can see all the buildings. When the user wants to see what a specific building looks like, they can click that building on the map. This will send a request to the server to access information pertaining to the layout of the building. The server must access the Database for information on floor plans. On the user's end this will display the first-floor floor plan by default. Next, they can switch floors to see the basement or any higher floor. On any classroom, when clicked, the server must ping the Purdue API in order to access the information about the courses that take place in a specific room selected by the user.

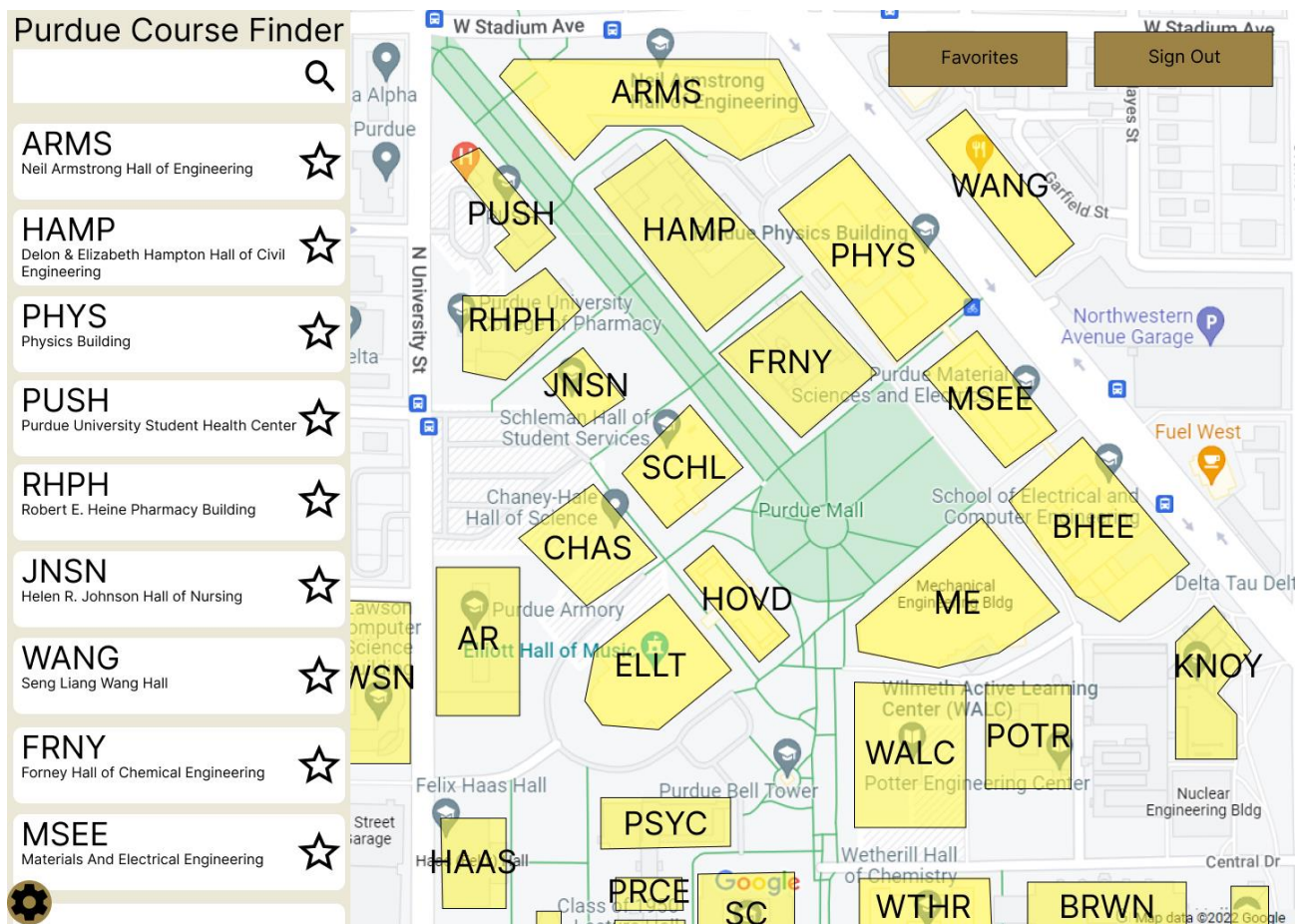
Activity Diagrams



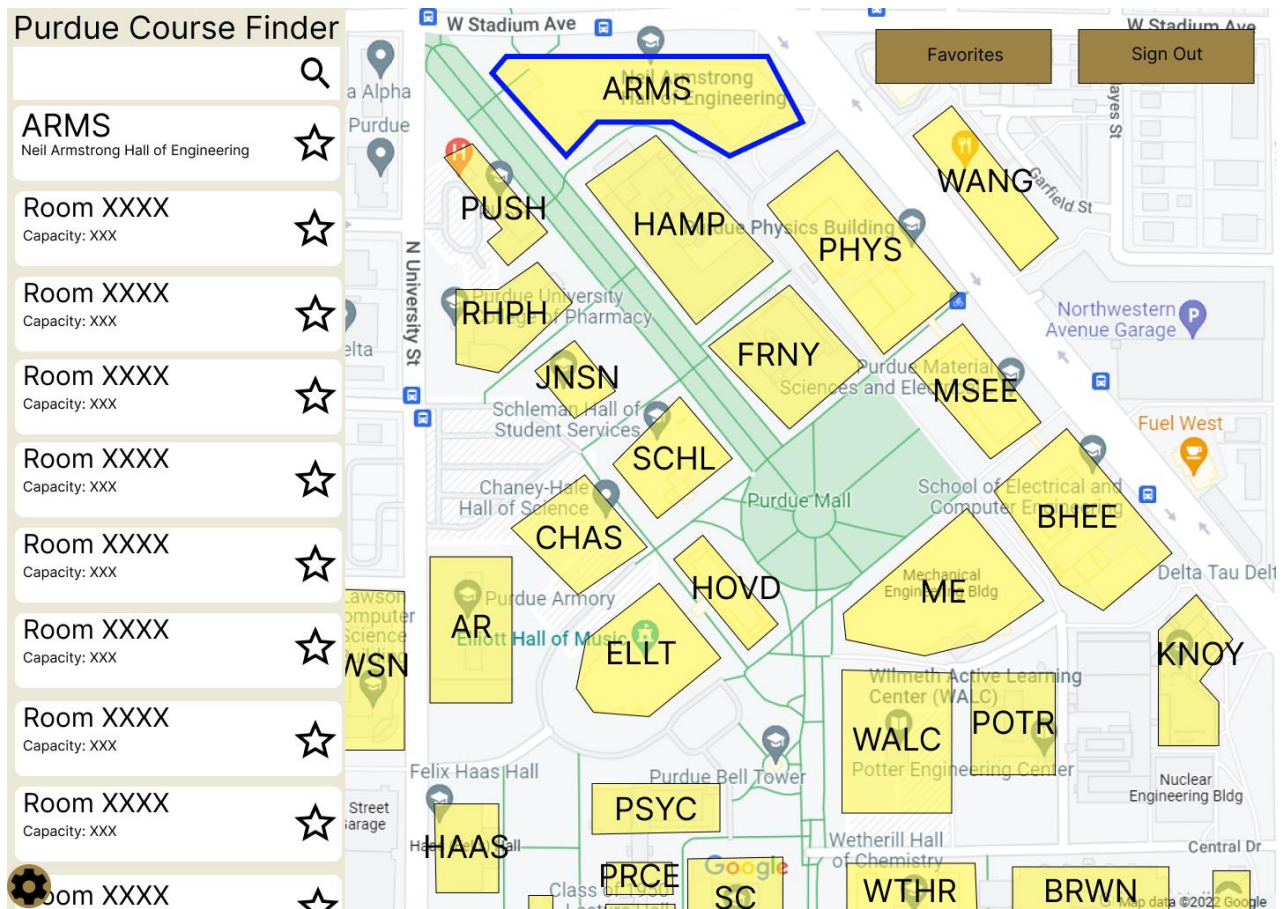
This Activity Diagram shows the sequence of events that occur when a user creates an account in our application. When the application receives the request to create an account from the user, it first checks that the provided email address and password are in a valid format, and that the value entered in the “Confirm Password” textbox matches the value entered into the “Password” textbox. If the email address and password are valid, the application’s backend checks if the provided email address already has an account. If the email address is already connected to an account, an error is shown to the user. If the email address is not connected to another account, a new account is created with the credentials provided, and the user is logged into the application with their new account.

UI Mockups

1. Main Page: This is the main page of our application. It shows the map of Purdue's campus with building locations highlighted and labeled. The sidebar on this page shows a list of the Purdue buildings shown in this part of the map. The search bar can be used to find buildings, or to filter the highlighted buildings by class or section locations. The star icons on the sidebar allow users to quickly favorite those items. The star next to a favorite item would be filled in and would allow a user to click it to unfavorite the item. The gear icon in the bottom left of the page leads to a settings page. The "Favorites" button in the upper right leads to a page where the user can see and edit lists of their favorited buildings, rooms, courses, and sections. This button will only appear if the user is signed in to an account. The "Sign Out" button in the upper right logs the user out of their current account. When a user is not signed in, this button will say "Log In" and lead to the log in page.



- Building Selected: This mockup shows the page when a building is selected from the map. That building will have a special highlight to show that it is currently selected, and the other buildings will remain labeled and highlighted as normal. The sidebar will have information about the selected building at the top. That is followed by a list of classrooms in the building. Clicking on one of these rooms will lead to the schedule page that shows all sections in that room.



- Search for Course: This mockup shows the page when a course is searched for. The only buildings that are highlighted and labeled on the map are buildings that contain sections for the selected course. The sidebar will have information about the selected course on top. That is followed by a list of all the offered sections of that course for the selected semester. Clicking on the course at the top of the list would lead to the schedule page showing all sections of the course. Clicking on a specific section would lead to a page showing more information about that section.

Purdue Course Finder

CS 35400

Operating Systems

☆

CS 35400-LE1

Room: PHYS 203
Instructor: Douglas Comer

☆

CS 35400-LE2

Room: WTHR 104
Instructor: Chunyi Peng

☆

CS 35400-P07

Room: HAAS 257
Instructor: Shreyas Kharbanda

☆

CS 35400-P05

Room: HAAS 257
Instructor: Shreyas Kharbanda

☆

CS 35400-P01

Room: HAAS 257
Instructor: Frances O'Leary

☆

CS 35400-P08

Room: HAAS 257
Instructor: Andrew Smith

☆

CS 35400-P04

Room: HAAS 257
Instructor: Andrew Smith

☆

CS 35400-P09

Room: HAAS 257
Instructor: Haotian Deng

☆

CS 35400-P03

☆

W Stadium Ave

Garfield St

N University St

Central Dr

Neil Armstrong Hall of Engineering

PURDUE

Purdue Physics Building

PHYS

Purdue University College of Pharmacy

Purdue Material Sciences and Electrical...

Schleman Hall of Student Services

Chaney-Hale Hall of Science

Purdue Armory

Elliott Hall of Music

Felix Haas Hall

HAAS

Purdue Bell Tower

Wetherill Hall of Chemistry

WTHR

Class of 1950

Purdue Mall

School of Electrical and Computer Engineering

Mechanical Engineering Bldg

Wilmeth Active Learning Center (WALC)

Potter Engineering Center

Nuclear Engineering Bldg

Delta Tau Delta

Fuel West

Northwestern Avenue Garage

Map data ©2022 Google

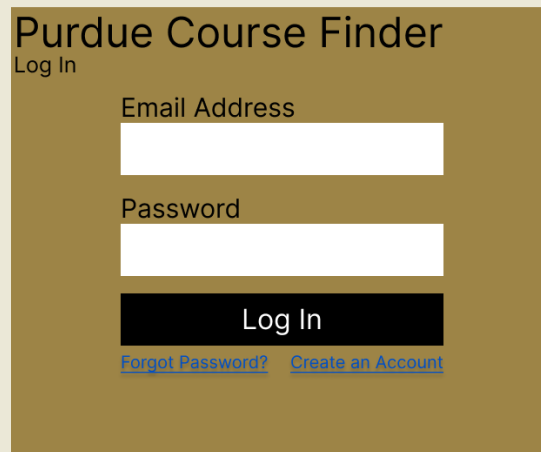
Favorites

Sign Out

4. Schedule Page: This mockup shows the page that shows the schedule of all sections of a selected course. A similar page will be used to show the schedule of all favorited sections, or of all sections in a specific room. For the course schedule, the page's title is "Class Schedule - [Course Name]", as seen below. For the room schedule, the page's title would be "Room Schedule - [Room Number]". For the favorited sections schedule, the page's title would be "Favorited Sections". Clicking on a section from this page will lead to a page showing more information about that section. That page would also allow the user to favorite or unfavorite that section.

← Class Schedule - CS 35400							
	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
6a							
7a							
8a							
9a							
10a				CS 35400-P01 9:30-10:20 HAAS 257 Instructor: O'Leary, Frances		CS 35400-P02 9:30-10:20 HAAS 257 Instructor: Das, Pratyush	
11a					CS 35400-P09 10:30-11:20 HAAS 257 Instructor: Deng, Haotian	CS 35400-P06 10:30-11:20 HAAS 257 Instructor: Satish Kumar, ...	
12p					CS 35400-P03 11:30-12:20 HAAS 257 Instructor: Lu, Edwin		
1p			CS 35400-P07 12:30-1:20 HAAS 257 Instructor: Kharbanda, Sh...				
2p		CS 35400-LE1 1:30-2:20 PHYS 203 Instructor: Comer, Douglas	CS 35400-P05 1:30-2:20 HAAS 257 Instructor: Kharbanda, Sh...	CS 35400-LE1 1:30-2:20 PHYS 203 Instructor: Comer, Douglas		CS 35400-LE1 1:30-2:20 PHYS 203 Instructor: Comer, Douglas	
3p				CS 35400-P04 2:30-3:20 HAAS 257 Instructor: Smith, Andrew			
4p		CS 35400-LE2 3:30-4:20 WTHR 104 Instructor: Peng, Chunyi		CS 35400-LE2 3:30-4:20 WTHR 104 Instructor: Peng, Chunyi		CS 35400-LE2 3:30-4:20 WTHR 104 Instructor: Peng, Chunyi	
5p					CS 35400-P10 4:30-5:20 HAAS 257 Instructor: Kim, Myeongsu		
6p							
7p							
8p							
9p							

5. Log In Page: This is a mockup of the page that allows users to sign into the web application. Each account must have an email address and a password, and each account must have a unique email address. The “Forgot Password?” link will allow users to reset their account’s password by sending them a password reset email. The “Create an Account” link will lead to the sign up page.



The mockup shows a login form titled "Purdue Course Finder" with a "Log In" link. It includes input fields for "Email Address" and "Password", a "Log In" button, and links for "Forgot Password?" and "Create an Account".

Purdue Course Finder
Log In

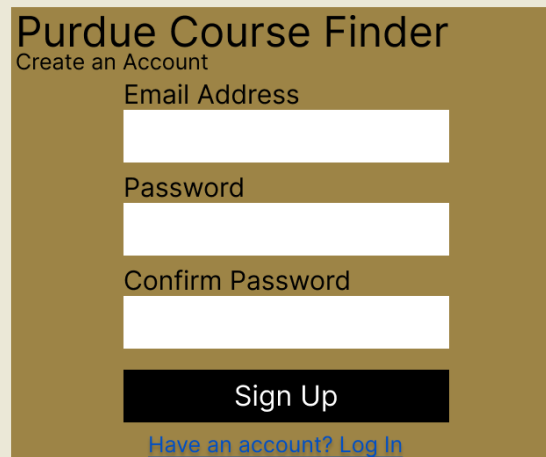
Email Address

Password

Log In

[Forgot Password?](#) [Create an Account](#)

6. Sign Up Page: This is a mockup of the page that allows users to create an account for our web application. Each account must have an email address and a password, and each account must have a unique email address. If someone tries to create an account with an email address that already has an account, an error will be shown that “An account with that email already exists”. If the values entered in the “Confirm Password” and “Password” boxes are not the same, an error will be shown that states that they must match. The “Have an account? Log In” link leads to the log in Page.



The mockup shows a sign-up form titled "Purdue Course Finder" with the subtitle "Create an Account". It contains three input fields: "Email Address", "Password", and "Confirm Password". Below these fields is a black "Sign Up" button. At the bottom, there is a blue link that says "Have an account? Log In".

Purdue Course Finder

Create an Account

Email Address

Password

Confirm Password

Sign Up

[Have an account? Log In](#)