CSE 30246

12 December 2022

StudyBuddy Final Report

Authored By:

Jonathan Nguyen, Brooke Mackey, Thomas Mercurio, Andrea Turner

I. Introduction

Our website, StudyBuddy, successfully helps students find and review study spaces across Notre Dame's campus. Our team compiled a database of study spaces around campus, and our website allows users to filter through these spaces to find a study spot according to their specific preferences. StudyBuddy also provides customized study space recommendations to users who are logged in, based on their previous habits or intended study activity. StudyBuddy can be utilized by students to find a central study location for a group to meet. Our mission was to create a website that would help students discover new study spaces that they may not have visited before. In allowing students to upload new study spaces to an intermediate database, our website benefits from the fact that each student is familiar with a unique subset of study space locations. Our aim is to relieve students of the monotony of visiting the same study spaces over and over again and introduce them to new study spaces, including ones that may fit their needs even better.

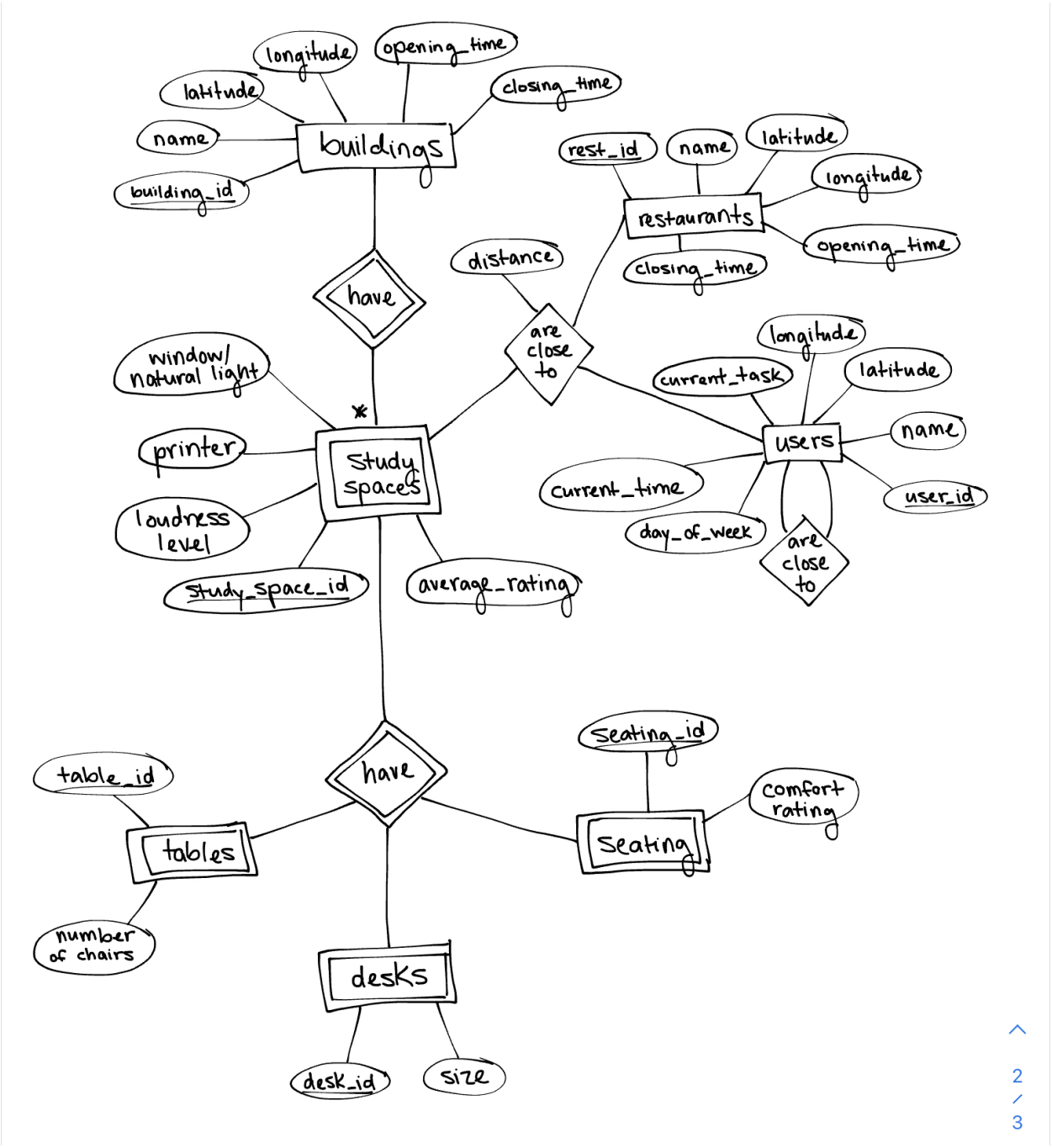
II. Usefulness

Our website is useful to all Notre Dame students who study in public places. With the combination of options regarding noise level, location, and group size, our website filters the hundreds of study spot locations on Notre Dame's campus to the best available options for the user. By eliminating the anxiety students may feel when choosing from hundreds of study spaces, StudyBuddy effectively encourages students to try out new study locations. We do not believe there is a website with this functionality available currently beyond a system used in 2020. This system existed during COVID to allow students to scan QR codes in the library, thereby marking their study location. This

idea of being able to check the availability of a study space is a feature we intend to develop for our website later. However, our website is much more robust, with many pieces of information about each study spot and other advanced functionality that makes for a positive user experience. Our website provides a useful tool for students who are looking to branch out and find a new perfect study location.

III. Our Data

To acquire data, our group manually entered data items into the database, and this proved to be one of the most taxing aspects of the project. Manual entry required a significant portion of our time, as we needed to first record the data while visiting each location and then utilize SQL to upload the data. We used an Excel sheet to record our data during the collection process, because this allowed us to emulate the structure of the final database. Our Excel sheet included the following columns for each study location: ID (int), building (str), location (str), floor (int), description (str), tables (bool), table seat comfort (int), non-table seat comfort (int), couch comfort (int), max group size (int), window/natural light rating (int), printer (bool), loudness rating (int), rating (int), outlets rating (int), max capacity (int), and amenities/notes (str). The other source of data we utilized is Google Maps. We used this latitude and longitude data to calculate a study spots' proximity to a user's current location, as well as that of any other students they are meeting. The following E.R. diagram is our original diagram from the beginning of the project. We assumed that the entity sets were buildings, study spaces, users, tables, desks, and seating. We also assumed that buildings have many study spaces, and that study spaces are a weak entity set, meaning that they are defined by their building.



From this E.R. diagram, we created the following relational schema:

users(<u>user_id</u>, name, latitude, longitude, current_time, day_of_week, current_task)

user_id → name, latitude, longitude

closeToPeople(user_id, user_friend_id, distance)

user_id, friend_id → distance

restaurants(rest_id, name, latitude, longitude, opening_time, closing_time)

rest_id → name, latitude, longitude

buildings(building_id, name, latitude, longitude, opening_time, closing_time)

building_id → name, latitude, longitude

studySpaces(study_space_id, building_id_fk, natural_light, printer, loudness, average_rating)

study_space_id → building_id_fk

closeToRestaurant(user_id_fk, study_space_id_fk, distance)

user_id_fk, study_space_id_fk → distance

tables(table_id, study_space_id_fk, number_of_chairs)

table_id → study_space_id_fk

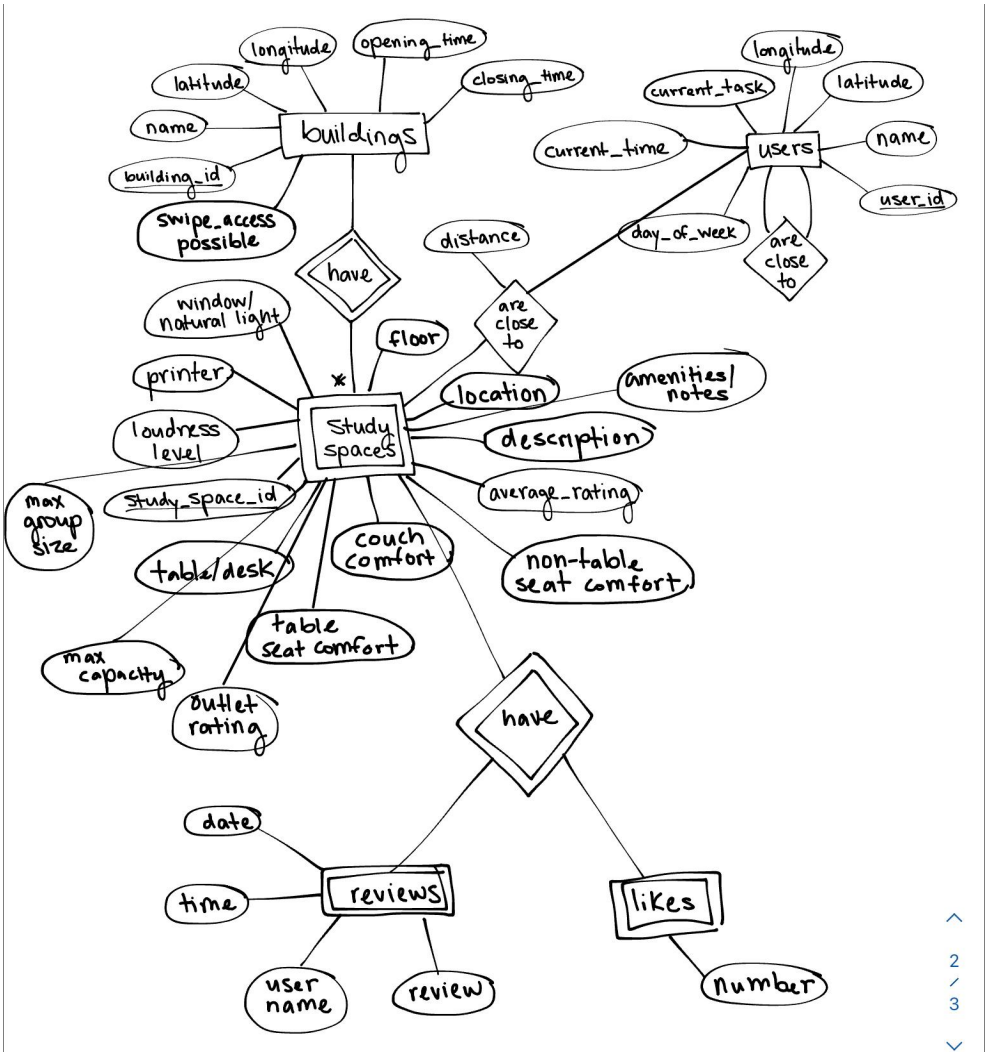
desks(desk_id, study_space_id_fk, size)

desk_id → study_space_id_fk

seating(seating_id, study_space_id_fk, comfort_rating)

seating_id → study_space_id_fk

After working on our project, we made some modifications to our original E.R. diagram and relational schema. We decided not to use any data involving restaurants, so we removed this from our E.R. diagram. We also removed tables, desks, and seating as entities and included them as attributes of study spaces instead. We added more attributes to our study space entity and a weak entity relationship between study spaces, reviews, and likes. We made these weak entity sets associated with study spaces because they are defined by their relationship to their respective study space.



These changes in our E.R. diagram are reflected in our relational schema as follows:

users(<u>user_id</u>, name, latitude, longitude, current_time, day_of_week, current_task)

user_id → name, latitude, longitude

closeToPeople(<u>user_id</u>, user_friend_id, distance)

user_id, friend_id → distance

buildings(building_id, name, latitude, longitude, opening_time, closing_time, swipe_access_possible)

building_id → name, latitude, longitude

studySpaces(study_space_id, building_id_fk, natural_light, printer, loudness, average_rating, floor, location, description, amenities_notes, non_table_seat_comfort, couch_comfort, table_seat_comfort, outlet_rating, table_desk, max_capacity, max_group_size)

study_space_id → building_id_fk

reviews(review_id, study_space_id_fk, date, time, user_name, review)

review_id → user_name, study_space_id_fk

likes(like_id, number, study_space_id_fk)

like_id → study_space_id_fk

IV. Functionality

The basic functionality of our website is to provide study spot location recommendations to users based on their preferences. Filtering options in the "Search" page include proximity to a printer, amount of natural light, noise level, types of seating options available, if there is a table/desk space, and more. Based on the requirements of the user, our website displays study spot options, including a picture of the spot, its name, and a rating based on previous users' reviews of the spot. A user can click

on a study spot to open its main “Location” page, which includes more details about the space, its location on a map, building hours, more photos, and the option to like and review the space, as well as read others’ reviews. The “Explore” page presents all of the study spaces in our database by building, and it also allows users to click on a space to read the detailed “Location” page. Users can utilize the “Suggest” page if they are logged in. The “Suggest” page gives users the option to receive custom study space recommendations either based on their previous website history or on their desired study activity. Our “Collaborate” page is intended to be used by groups looking to find a central meeting location. It prompts a user to enter their location, as well as the locations of their friends, and it uses the Google maps API to find a study location central to all group members’ locations. Finally, the “Upload” page allows users to upload new study spaces to our database. Submitted study spaces are sent to an intermediate database, so we can approve the space before adding it to our main database.

V. SQL Queries for a Basic Function

Our most basic function, the “Search” feature, allows users to search for study spots based on their preferences. The search page prompts the user to fill out information to filter their search results, like the building where they would like to study, their preferred seat comfort, level of natural light, outlet availability, or noise level, if they need a whiteboard, table or printer, and the number of people they would prefer to work with or be in the area. When the user hits the submit button, their filtering selections are collected and used to query the study_spots table in our database. We utilized the Axios library to make HTTP requests on the client side and Express on the server side to listen to and execute these HTTP requests on the back-end. While our original plan was to utilize the provided MySQL

databases on the DB8 student machines, we eventually migrated to Bitnami to deploy StudyBuddy, and Bitnami provided a MySQL database that could be more easily linked to our app. Thus, HTTP requests created by our search feature and executed by the Axios client were fielded by the Express server, which then queried the MySQL database through Bitnami. The exact SQL query is as follows:

SELECT * FROM study_spots WHERE ${building} AND ${outlets} AND ${loudness} AND ${naturalLight} AND ${group} AND ${capacity}

Where each item surrounded by a "${}", is a variable connected to the user's filtering preferences and concatenated with SQL syntax to equate it to a specific row in the table. For example, if a user indicated that their group size would be 8 people, then the "group" variable in the query would appear as "max_group_size >= 8". The results of this query are then sent back to where the Axios request was made in Search page's Javascript code, and finally, the data is utilized to display each location in the results section along with its picture.

VI. Advanced Functions

The first advanced function of StudyBuddy is the study spot recommendation feature on the "Suggest" page. On this page, users can find study spot recommendations based on their previous likes and reviews with the "History" button or a recommendation for the specific type of work they are doing based on previous user's reviews with the "Work" button. These recommendation features will be incredibly useful for users, as this feature streamlines the StudyBuddy experience. For users who

may be in a rush or do not want to manually enter their study spot preferences in the Search page, the History recommendation algorithm will automatically aggregate the user's stored data to find spots that they will enjoy. First, on the back-end, a user's likes and reviews as well as the data from every study spot they liked or reviewed is queried from the database. Then, based on their review rating, the study spot's qualities are weighted and then averaged, and for study spots the user liked, the qualities are simply averaged. Finally, the user's preferences are more heavily weighted based on their reviews, which comprise 65% of their preferences, and likes comprise 35% of their preferences. This algorithm provides a unique set of study spots for each user based on their favorite study spot qualities. For users who want to find new study spots specifically catered to their type of work, the Work recommendation feature will assist them in finding a study spot that best suits that type of work. Each time a user enters a review, they are presented with a prompt to enter the type of work they did at that location. Thus, the recommendations based on work collects data from the reviews table and the study spots that have been reviewed and then aggregates and averages the study spot's qualities. This data is then used to find similar study spots to those where previous users have done a specific type of work, and these study spots are reported on the recommendation page. In both cases, the collection and analysis of the data was significantly more advanced than our other features, as the front end needed to relay information to the back-end and then wait for the final aggregate data to display the results. Ensuring the accuracy and correct timing of each feature's data dependencies was integral to making the recommendation feature a reality.

Our second advanced function is found on our "Collaborate" page and consists of a group work recommendation feature. The page prompts the user for his or her current location, and allows

them to enter the usernames of the people they will be working with (or any name for those who do not have an account) as well as their locations. The user then enters what time the group will be meeting, with an option for "Now" which fills the current time, and if they need a whiteboard, printer, TV, or computer. After submitting, a series of API calls are made, with the advanced function calculating a "score" for each study spot that is returned from the original filtered API call. The score is calculated through a formula that takes into account the walking distances of those farther away from the locations, a weighted average of the difference between the spot's values for various fields and the preferred values based on the users' past reviews, and the overall rating of the study spot. The "advanced" nature comes from the use of the Google Maps distance matrix API and aggregating all users' past reviews, combining that into a formula that sorts the recommendations for users. The user is then presented with a list of the top ten meeting spots, with a note of the distance from each of the origin locations of members of the group, as well as notes stating if swipe access is needed at this time or the requirements they hoped for are there.

VII. Technical Challenges

One technical challenge we encountered was in our use of the Google Maps API. We researched the API, procuring a key and testing out the functionality with a Python script. When we added the calls into our JavaScript code, however, we continually ran into an error that said our request was "blocked by CORS Policy: No Access-Control-Allow-Origin header is present on the requested resource." After many hours of learning about CORS policy and trying various solution ideas on StackOverflow, we realized that the issue was that our API calls were coming from the browser, which

is why it was not allowed by CORS policy. Keeping that code in our rendered JavaScript code would allow any user on the browser to go in and change the code, making whatever API calls he or she would like. We needed to wrap the Google Maps API calls inside our own API calls, and once we did that, our problem was solved.

VIII. Final Plan and Design Specification Changes

For the most part, our project went according to plan. As discussed earlier, we made some modifications to our original E.R. diagram and relational schema plan. This is largely because, as our vision for our project evolved, we realized that we either needed to add or take away from our E.R. diagram. For example, we realized that the number of tables in a study location would not be utilized in our StudyBuddy application, so we removed it from our E.R. diagram and relational schema. On the other hand, we realized that many additional attributes, such as a rating of the availability of outlets, would be very helpful in our application, so we added these. The overall functionality of our website matches our original plan, and we successfully implemented the basic and advanced functions that we intended. As mentioned previously, the only aspect that we did not implement as intended is that we did not integrate data from the restaurant locations on campus. However, this is a feature we could easily add in the future, because we already have the Google Maps API integrated into our website. We also did not originally intend on creating an "Explore" page, but we felt that this would be a helpful addition to our website by allowing users to look through study space options grouped by building.

IX. Division of Labor

Our final division of labor was mostly similar to our original plan, but it also differed. Jon did lead the front-end development as planned. He also handled a lot of back-end work, including setting up the server and some of the basic functions. Andrea manually collected most of the data by visiting all of the study spots across campus, and Brooke predominantly managed the input of this data into our database using SQL. Andrea worked on front-end development alongside Jon, while Brooke and Thomas handled back-end development for the advanced functions. Brooke mainly handled the recommendation advanced function, while Thomas mainly worked on the collaboration advanced function.