study spotter

Outline

Problem Statement

Audience and Goal

Our Solution

Demo

Technical Implementation

Next Steps

Reflection

Q&A

The Problem [Alan]

- Finding a productive place to study is crucial to student's academic success
- However, libraries can often be crowded, coffee shops loud, and the outdoors unpredictable
- There is a clear need for an easy and convenient way to find places on campus to study
- Our goal: Create a tool that painlessly finds on campus spots to work using crowdsourced data

Our Solution [Jenny]: Study Spotter!

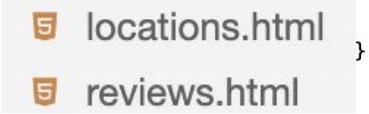
- App to store study spots on college campus and their information
  - More efficient for students to study and discover new places
- App to review study spots
  - Allows students to voice their opinions, see what others thinks about certain location, and find the study spot best for them

Demo !

Technical Components [Jenny] - Set-up & Tech Stack

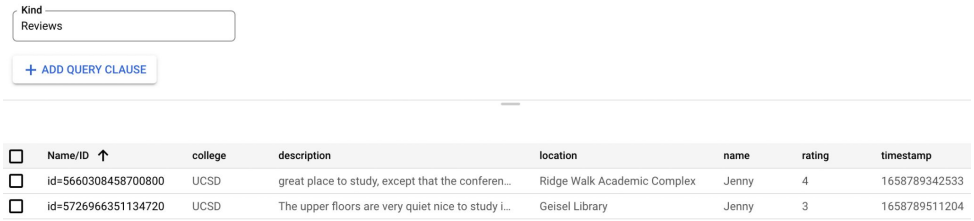
- Frontend: JavaScript, HTML, CSS
- Backend: Java, Google App Engine, DataStore
- SPA Routing
- Template HTML for the college's list of study spots and reviews page → filled in dynamically with data from DataStore

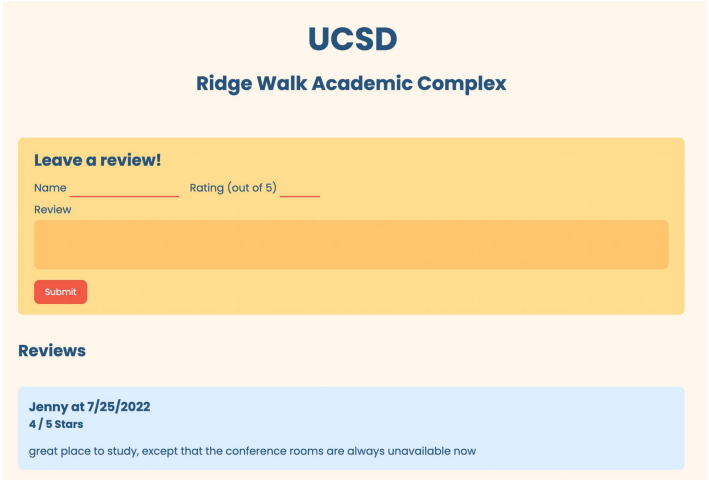
```
const routes = {
    404: {
        template: "/templates/404.html",
        title: "404 | " + pageTitle,
        description: "Page not found"
    },
    "/": {
        template: "/templates/index.html",
        title: "Home | " + pageTitle,
        description: "this is the homepage"
    },
    reviews: {
        template: "/templates/reviews.html",
        title: "Reviews | " + pageTitle,
        description: "Reviews page"
    },
    locations: {
        template: "/templates/locations.html",
        title: "Pick a Study Spot| " + pageTitle,
        description: "College page"
    }
```



Technical Components [Jenny] - Storing & Displaying User Reviews

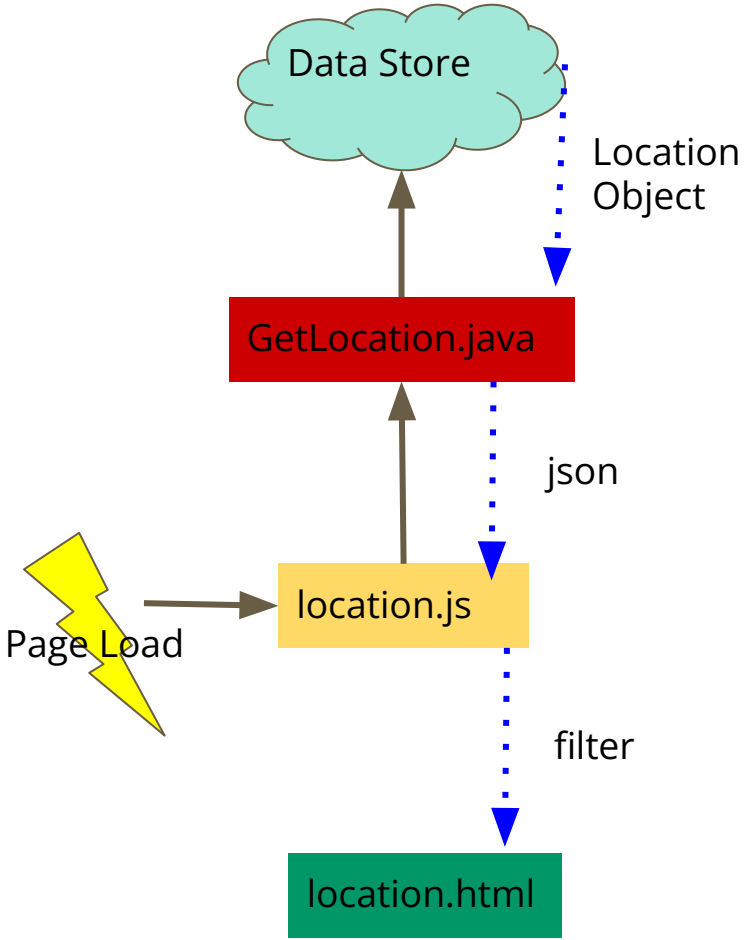
- **Storing**: Java servlet architecture + DataStore to store reviews
- **Displaying**: Fetch from DataStore + filter by location





Technical Components [Alan]

- One of the main challenges of this project: making it dynamic
  - This means our data is not hardcoded in the HTML file
- How did we do this?
  - By connecting our application to the datastore
- Steps between Data Store and App (exp.):
  1. location.js calls GetLocations.java on page load
  2. GetLocation.java returns a json to location.js
  3. location.js checks if that data entry is valid
     a. In this case, if the college matches the name
  4. If it does, it creates a new HTML card element to be displayed

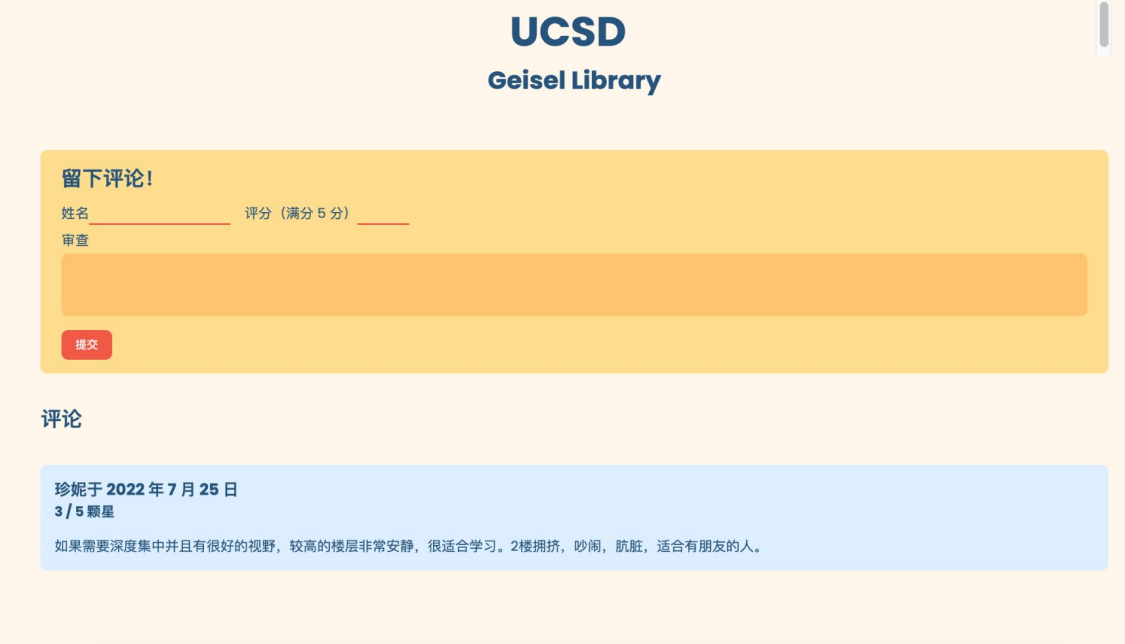


Technical Components [Sahiti]

New feature:

Google Translation to all pages of the website

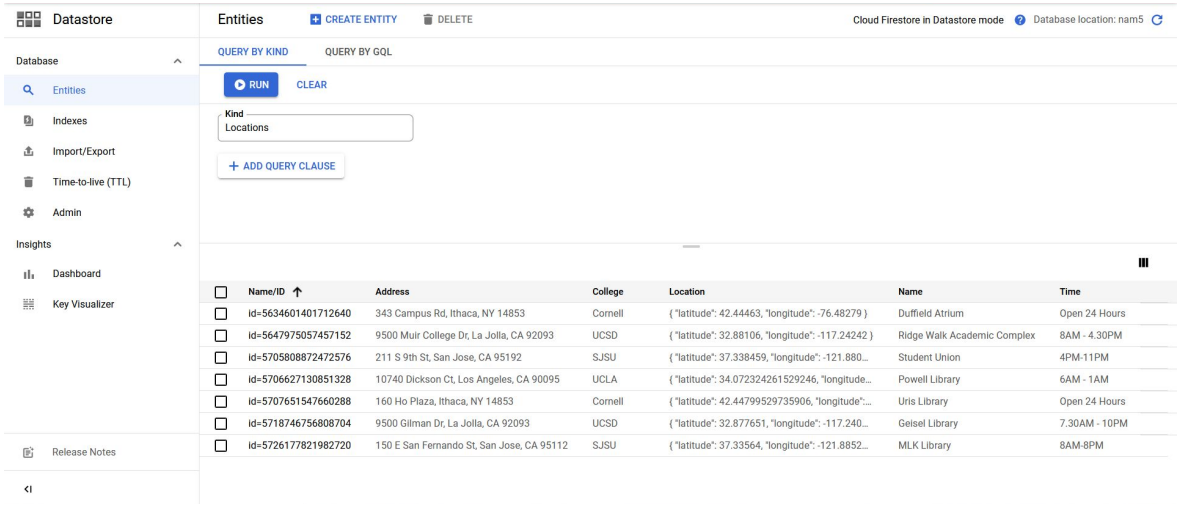
- More accessible info to all
- Offers translation in all languages



Technical Components [Stephanie] - Location Datastores

Use cloud datastore

- More dynamic
- Allows for more complex queries, e.g. filtering by college



Technical Components [Stephanie] - Landing Page

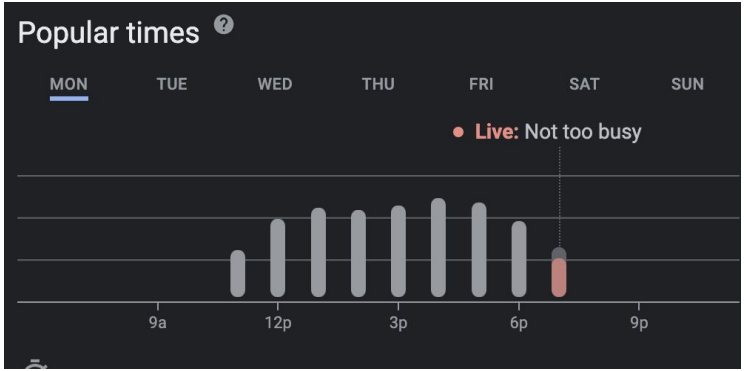
CSS + HTML

```html
<html lang="en-US">
  <head>
    <meta charset="UTF-8">
    <title>Landing Page</title>
    <link rel="stylesheet" href="templates/home-style.css">
  </head>

  <body>
    <div id="content">
        <div style="position:relative; left:200px;">
            <div align="right" class="fade-in-image">
                <img src="templates/images/logo.png" width="500" />
            </div>
            <div style="position:relative; left:80px;" class="fade-in-image">
                <p>Find the perfect study spot today!</p>
            </div>
        </div>
    </div>
</body>
```

```
        font-size: 20px;
        color: #000000;
    }

    body {
        background-image: url('images/bg.png');
        background-attachment: fixed;
        background-size: cover;
    }

    .fade-in-image {
        animation: fadeIn 8s;
        -webkit-animation: fadeIn 8s;
        -moz-animation: fadeIn 8s;
        -o-animation: fadeIn 8s;
        -ms-animation: fadeIn 8s;
    }
    @keyframes fadeIn {
      0% {opacity:0;}
      100% {opacity:1;}
    }

    @-moz-keyframes fadeIn {
```

Future features [Sahiti]

- Images of study spots
- Star-based ranking
- Live graphs with up to date metrics (# of people, how busy, etc)
- Recommendations
    - Favoriting option that could influence the recommendation system



Reflection [Stephanie]

What went well

- Product looks good!

Challenges

- Splitting up tasks evenly/modularly
- Bugs along the way

Things we learned

- Using libraries, interacting with datastores
- Git, html, css, javascript, java servlets

Q&A