Study Sphere

Diego Castro, Vibha Joshi, Jun Kusano, Siraaj Sandhu, Henry Van Cleave

App Description

- Homework-help website designed to help students learn and practice course material
- Questions are organized based on course content
- Users can bookmark questions to save for later
- Students can join courses where they can collaborate with their peers
- Users can make a profile for free

What makes Study Sphere different from existing websites?

- Promotes collaboration
  - Live chat
  - Creates communities of students taking the same course
- Encourages students to learn rather than cheat
- Provides personalized experience for users
  - Students can save questions, join classes, etc
- Free
  - Accessible to anyone

Features

- Navigation & Theme:
  - Light theme with purple accents
  - Persistent navbar linking all pages
- Authentication:
  - Guest mode allows limited browsing; login required for interactions
  - Login and registration pages with session setup and redirection
- Core Features:
  - Home Page: Search classes, view popular classes/questions
  - Class Page: Ask new questions, view/search questions, live chat for discussions
  - Question Submission: Submit questions with images
  - Question Page: Bookmark questions, submit and rank answers
  - Profile Page: Manage bookmarks, classes, and account settings
- Design:
  - Focused on user engagement and accessibility with an intuitive light-themed interface

Tools We Used

- Repository:
  - Github - store, share, an collaborate our code (5)
- Project Tracker:
  - Github Project Board - Used to effectively plan and track out work throughout the project (4)
- Database:
  - PostgreSQL - Used to organize, store, and manage data from users (5)
- IDE:
  - Visual studio - For developing and debugging our code (5)
- UI Tools:
  - Lucidchart (1), HTML(5), HandleBars (5), Bootstrap/CSS (4) - Creates cleaner and nicer visuals for the user
- Languages: JavaScript (5), SQL (4)

Tools We Used

Application Server:
- NodeJS - Server creation, database connectivity (5)

Deployment environment:
- Render - Deploys a running prototype of an application to users (4)

External API:
- AWS - Used to store images for answers and questions (4)

Testing Tool:
- Mocha, Chai - Used for testing code (4)

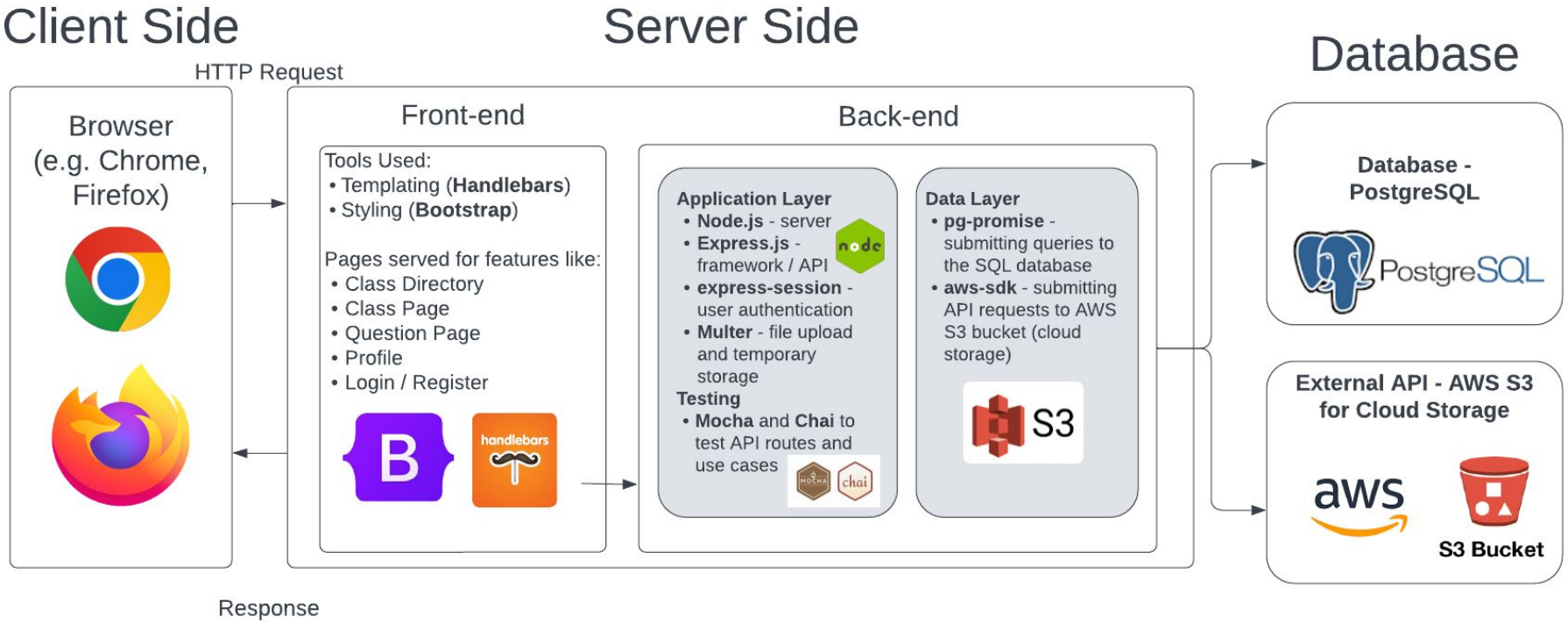
Framework:
- Express in Node - Simplifies the process for API development (5)

- Methodology:
  - Agile (4)

Communication

- Intrateam
  - Discord
  - Text
  - Normal team meetings on wednesdays
  - Pull requests
- With TA
  - Zoom for weekly meetings
  - Every monday at 5:15

Architecture Diagram



Challenges

- Primary challenge: Supporting different forms of media
  - How to incorporate different forms of media, including image, text, and video in user posts?
    - **AWS Simple-Storage-Service (S3) & Buckets**
    - **aws-sdk** node module
- Often encountered need to support similar action across many different pages
  - Don't want to repeat same functionality (**Don't Repeat Yourself - DRY**)
  - Created our own APIs in these cases
    - e.g. **"search" API** accepts a get request with a search key and topic, queries an arbitrary table in the DB, and sends a JSON response containing results
    - **Easier to test using Postman, etc.**
  - Others: **"vote" API** for liking/disliking answers, **"image" API** for fetching images from AWS S3

Future Scope/Enhancements

- SSO (single sign on)
  - Make service easily generalizable to multiple institutions/schools
- Verifying faculty to enhance credibility of answers
- video/multimedia + arbitrary file upload
  - AWS-S3 makes this relatively easy to incorporate
- Step-by-step answers
  - Have users correctly answer “hints” in answer posts to slowly reveal an entire answer - answer keys

Demo

Thank you!