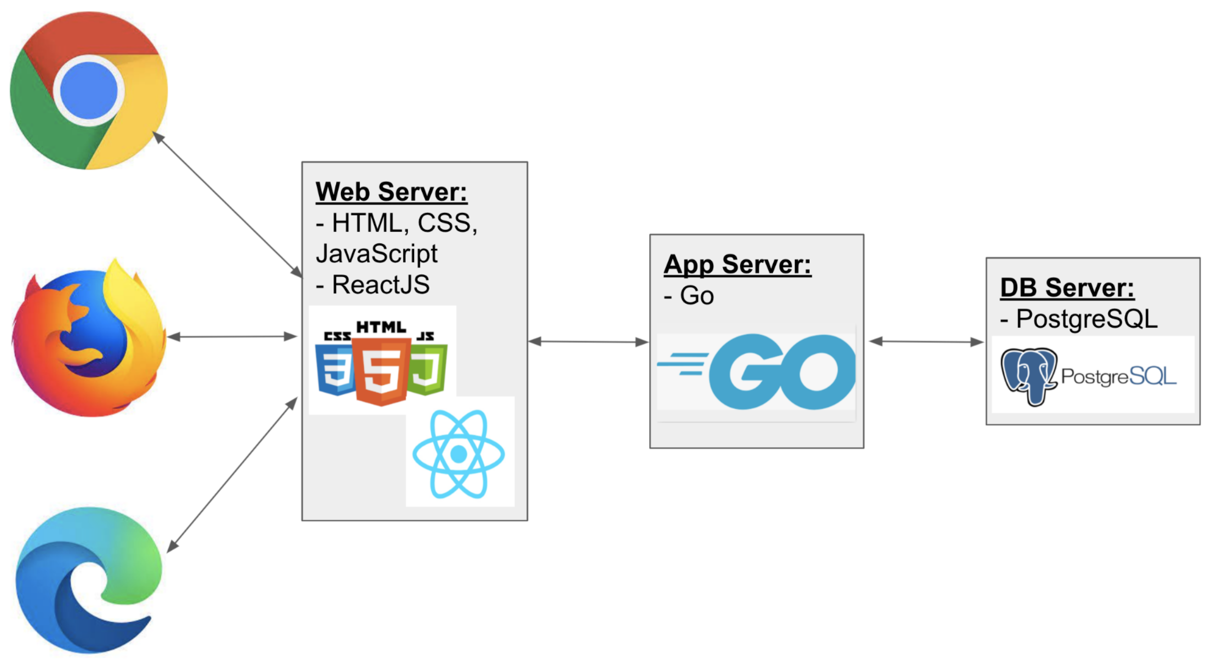

We chose to use the ReactJS framework for our web server because it is a very common framework for web development. Also, based on our initial sketches, we will benefit greatly from being able to create reusable UI components that can change data without reloading the page. ReactJS is also fast and simple to use. Next, we chose Go for our backend server because it is a fast, modern, static programming language and two of our members have experience with it. Furthermore, Go's performance is comparable to C and C++ (with no virtual machines and direct compilation to machine code) and is good for building microservices. Lastly, for our database server, we chose PostgreSQL because it is open source, free, and seems to fit our project requirements (for ACID transactions, a table like structure, and support for complex queries).

Technologies Summary:
Web server: ReactJS (HTML+CSS+JS)

Application server:

1. Go
   https://golang.org/
   https://tour.golang.org/welcome/1
2. Gorm (Help us connect with database, like JDBC in java)
   https://goswagger.io/
   https://github.com/go-gorm/gorm
3. Go Swagger 2.0
   we write a .yaml file (similar to JSON) like the following, then Go Swagger will help us generate all pre required files. We only need to write handlers for our api.
   https://goswagger.io/

```yaml
---
swagger: '2.0'
info:
  version: 1.0.0
  title: Greeting Server
paths:
  /hello:
    get:
      produces:
        - text/plain
      parameters:
        - name: name
          required: false
          type: string
          in: query
          description: defaults to World if not given
      operationId: getGreeting
      responses:
        200:
          description: returns a greeting
          schema:
              type: string
              description: contains the actual greeting as plain text
```

4. OpenAPI specification 2.0 (Some syntaxes)
   https://github.com/OAI/OpenAPI-Specification/blob/main/versions/2.0.md

Database server: PostgreSQL