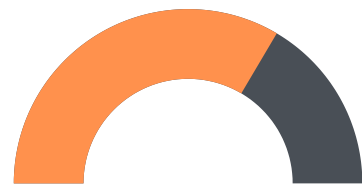


EasyPortal

Business Management System

Created by Timotius Mogot and Morgan Rohan

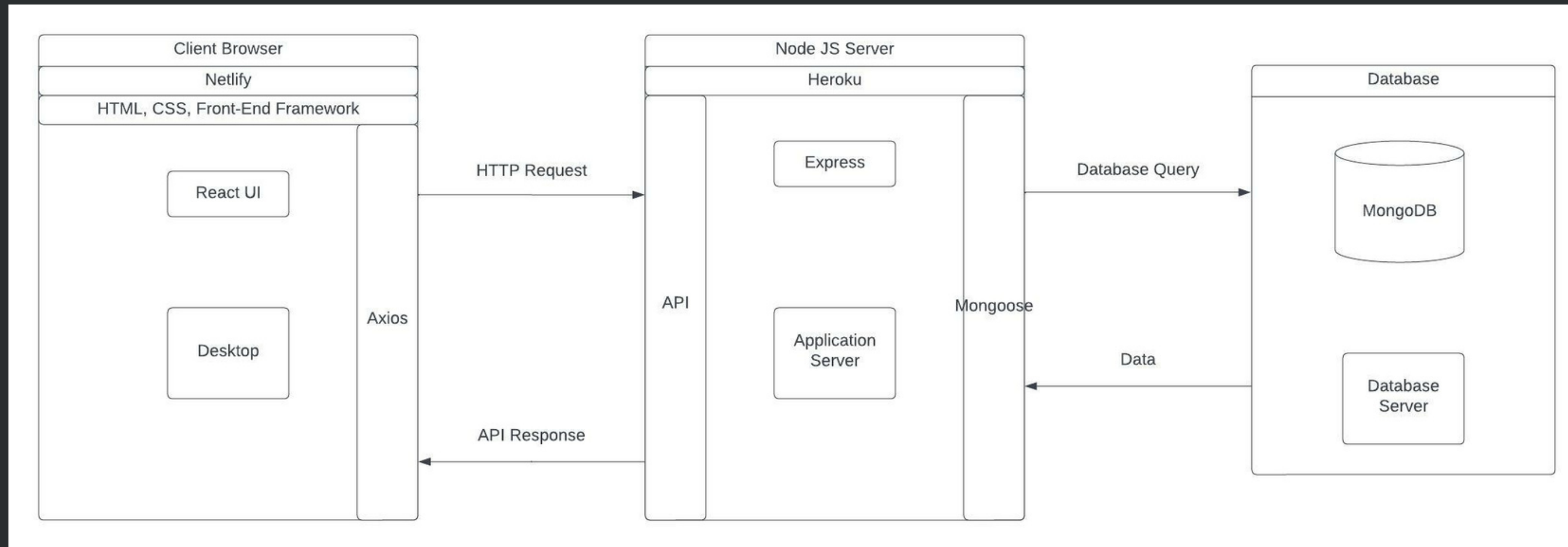
EasyPortal



Purpose

The purpose of this web app is to manage, calculate and display employees' payroll and scheduling activities.

EasyPortal seeks to solve how these issues can impact smaller businesses, where these responsibilities often fall to only one or two people and can be very time consuming.



Also:
-Firebase for Authentication

Testing:
-Postman
-Jest
-Cypress

Tech Stack & Application Architecture

Trello Workspaces Recent Starred Templates Create Search 8 ? MR

This board is set to public. You can change its visibility at any time. [Learn more here](#)

Full Stack App Project ☆ Public Board Filter Power-Ups Automation Share ...

Client

- Styling of User Interface MR 1 22 Nov
- Contact Form Page MR 21 Nov
- Pay Calculation Quiz Page MR 21 Nov
- + Add a card

Issues

- Set Up Config / API MR TM 1 22 Nov
- + Add a card

Testing

- Set Up Express Server TM 13 Nov
- Set Up MongoDB DB TM 13 Nov
- Deploy Server to Heroku TM 13 Nov
- Jest Testing 19 Nov
- Auth Functionality TM 16 Nov
- + Add a card

Completed

- Part A: Review documents for compiled README MR TM 5 Nov
- Part A: R0 - Create repo for README file MR TM 4/4
- Part A: R1 - Application description 4/4
- Part A: R2 - Dataflow diagram MR TM 25 Oct 7/7
- Part A: R3 - Application Architecture Diagram MR TM 25 Oct 6/6
- + Add a card

+ Add another list

Task Delegation

Some issues faced throughout the application build:

- Initial front end build was difficult to interact and wouldn't render correctly
- Interaction between Firebase and MongoDB
- API connections and pulling employee data to front end

Learnings:

- Team communication
- MERN structure

Challenges

```
App.js — FrontEnd
JS Rosters.js JS App.js X JS rosterServices.js JS UserFunctions.js M JS api.js JS Register.js
EasyPortal-Client > src > JS App.js > App
99 // Use ternery to operate loading page and main page
100 return [
101   <StateContext.Provider value={{ store, dispatch }}>
102     <Container maxWidth='lg'>
103       <LogInBar />
104       <Nav title="EasyPortal"
105         sections={sections}>
106       </Nav>
107     </Container>
108     <Routes>
109       {!loggedInUser ?(
110         <>
111           <Route path="/" element={<SimpleHome />} />
112           <Route path="/about" element={<About />} />
113           <Route path="/register" element={<Register />} />
114           <Route path="/login" element={<LogIn />} />
115           <Route path="/thankyou" element={<ThankYouPage />} />
116           <Route path="*" element={<NotFound />} />
117         </>
118       ):(
119         <>
120           <Route path="/" element={<SimpleHome />} />
121           <Route path="/about" element={<About />} />
122           <Route path="/rosters" element={<Rosters />} />
123           <Route path="/rosters/new" element={<NewRoster />} />
124           <Route path="/rosters/:id" element={<RosterDetails />} />
125           <Route path="/rosters/update/:id" element={<NewRoster />} />
126           <Route path="/register" element={<Register />} />
127           <Route path="/login" element={<LogIn />} />
128           <Route path="/thankyou" element={<ThankYouPage />} />
129           <Route path="/dashboard" element={<EmployeeDashboard />} />
130           <Route path="*" element={<NotFound />} />
131         </>
132       )}
133     </Routes>
134   </StateContext.Provider>
135 ];
136 }
```


```
LogIn.js — FrontEnd
JS Rosters.js JS LogIn.js X JS rosterServices.js JS UserFunctions.js M JS api.js JS Register.js
EasyPortal-Client > src > components > JS LogIn.js > ...
24
25 function handleSubmit(event) {
26   event.preventDefault();
27
28   loginUser(formState)
29     .then((data) => {
30     let displayName = data.displayName;
31     let token = data.token;
32     sessionStorage.setItem("token", token);
33     sessionStorage.setItem("user", displayName);
34     dispatch({ type: "setLoggedInUser", data: displayName });
35     dispatch({ type: "setAdminUser", data: displayName});
36     dispatch({ type: "setToken", data: token });
37     navigate("/dashboard");
38   })
39   .catch((error) => console.log(error));
40 }
41 return (
42   <div>
43     <label>Email:</label>
44     <input
45       type="email"
46       name="email"
47       value={formState.email}
48       onChange={handleChange}
49     ></input>
50     <label>Password:</label>
51     <input
52       type="password"
53       name="password"
54       value={formState.password}
55       onChange={handleChange}
56     ></input>
57     <Button onClick={handleSubmit}>Login</Button>
58   </div>
59 );
60 }
```

Functions & Application Dataflow

User Stories and Client Interaction

Darren

Darren wants an easy way to roster and manage employee shifts to get back some work-life balance



Brief description

Darren is a 32 year old man and owns a gym with 15 employees. He likes to be outside and hiking, trying new recipes but also likes to game in his spare time.

Persona Main Job (Main goals)

- Darren's main job is to manage the gym (as the business owner) but he's also a certified personal trainer and wants more time to do sessions instead of business admin

Personality

- Confident
- Open-minded
- Upbeat

Interests

- Nature
- Nutrition
- Tech
- Gaming

Apps used by Darren

- Instagram
- Twitter
- Discord

Gains

- Able to easily schedule and make changes to the roster
- More time for PT sessions
- Less complications with employees (staff not turning up for shifts, confused over pay or leave)

Pains

- Employees not sure of their schedule
- Spending more time on admin and rostering than in the gym with clients

User Stories Revisited

- Updated to reflect scope of application within the available timeline
- Updated to reflect working state of the application

Client Interaction

- Aware of current functionality
- Discussed revised delivery date and reduced features

Thank you!

Timotius Mogot and Morgan Rohan