

Fullstack Kanban Task

Proje adı : Kanban Board

Versiyon: V1

Yazılımcı Adı: Raouf Satto

İçindekiler:

Özet.....	3
Proje Hakkında.....	4
1.Giriş.....	5
2.Genel Kısımlar.....	6
2.1.Back end kısmı.....	6
2.2.Front end kısmı.....	14
3.Proje Resimleri.....	15
3.1.Büyük Ekran.....	15
3.2.Orta Ekran.....	17
3.3.Mobil Ekran.....	24

Özet

Kanban Nedir:

LeanKanban Üniversitesi'nin tanımlamasına göre Kanban, profesyonel hizmetlerde işleri/çalışmaları organize etme ve yönetme metodudur.

Sonuçları iyileştirmek için üzerinde çalışan iş sayısını sınırlandırmak gibi Lean(Yalın) kavramları kullanır. Kanban sistemi, üzerinde çalışılan işlerin sınırlandırılması ve uygun iş alma kapasitesine ulaşıldığında gerekli sinyalin verilmesi üzerine kurulu bir çekme mekanizmasını işaret eder. Yine LeanKanban Üniversitesi'nin tanımlamasına göre Kanban metodu, dünya çapında başarılı Kanban girişimlerinde gözlemlenen prensip ve alıştırmalar dizisidir. Tahta ,kalem ve kartlar yerine programlama üzerinde bu Projede Kanban metodu kodlarla getirdik ve bu yazma boyunca Kanban Board proje nasıl kodlanabilir öğreneceğiz.

Proje Hakkında

Bu Projede Fullstack Javascript yazılımcı olarak Front end ve back end taraflarından Kanban Board projeyi kodlarla anlatacağım ve tüm soruları: nasıl yapılır, neden bu dil programlama kullandım, bu fonksiyon ne yarar etc bu dokümantasyonda cevap verilecektir.

Baştan İyi Okumalar dilerim

Rauf 27/12/2021

1.Giriş

Yapılan proje Node.js ve Express.js kullanılarak ve MongoDB veri tabanı kullanılarak hazırlanmıştır. Sitemizde ayrıca HTML5, SCSS ve React kullanılmış ve mobil uyumlu hale getirilmiştir.

Toplam sitede 1 kullanıcı bulunmaktadır. Bu kullanıcı site içerisindeki Boardları görebilir, silebilir ve Kullanıcı kartlarını inceleyebilir.

Kanban Board ekleme işlemi yaptıktan sonra 2.Sayfaya geçecek bu Sayfada 4 tane liste görünecek bu listeler: Backlog, To do, in Progress ve Done.

Kullanıcı bu Sayfada bir Task ekleme ve silme etkisine sahiptir.Ayrıca Listelere kartları listeler arasında taşınabilir ve kartların sıralaması aynı liste içerisinde değiştirilebilir. (Sürükle bırak yöntemi kullanark yapabilir).

2.Genel Kısımlar

2.1.Back end Kısımı:

bu kısımda Api Klasörü oluşturduk ve Node.js kullanarak 4 tane Klasörü oluşturduk bunları:

- 1) **Controllers:** bu Klasörde kullanılan tüm proje kontrolleri (logic code) bulunur. Task controller ve Board controller dosyası içinde bulunur bu dosyalarda Task veya Board için ekleme,silme , güncelleme ve listeleme fonksiyonları bulunur, bu fonksiyonlar ve mongoose üzerinde veriler MongoDB'e kayıt eder. Bu fonksiyonde async ve await prensipleri kullandım.

boardController.js dosyasında bulunacak fonksiyonları:

createBoard: bu fonksiyon üzerinde Mongoose Modeli schema(BoardModel) kullanarak database Bir Board oluşturulacaktır.

getAllBoards: bu fonksiyon üzerinde Mongoose Modeli schema(BoardModel) kullanarak databaseteki mevcut Boardları getirecektir.

getOneBoard: bu fonksiyon üzerinde Mongoose Modeli schema(BoardModel) kullanarak databaseteki Board slugı üzerinde istediğimiz Boardu getirecektir.

deleteBoard: bu fonksiyon üzerinde Mongoose Modeli schema(BoardModel) kullanarak databaseteki Board slugı üzerinde istediğimiz Boardu silmeden önce bu Boardla bağlı tüm taskları (kanban kartları) silecek ve sildikten sonra bu Boardu da silinecektir.

taskController.js dosyasında bulunacak fonksiyonları:

getAllTasks: bu fonksiyon üzerinde Mongoose Modeli schema(TaskModel) kullanarak databaseteki istediğimiz boarda göre mevcut taskları (kanban kartları) getirecektir.

createTask: bu fonksiyon üzerinde Mongoose Modeli schema(TaskModel) kullanarak database Bir Task oluşturulacaktır.

getOneTask: bu fonksiyon üzerinde Mongoose Modeli schema(TaskModel) kullanarak databaseteki Task slugı üzerinde istediğimiz Task ı getirecektir.

deleteTask: bu fonksiyon üzerinde Mongoose Modeli schema(TaskModel) kullanarak databaseteki Task slugı üzerinde istediğimiz Task silinecektir.

updateTask: bu fonksiyon üzerinde Mongoose Modeli schema(TaskModel) kullanarak databaseteki Task slugı üzerinde istediğimiz Task güncellenecektir.

.

- 2) Models: bu Klasörde kullanılan Mongoose(MongoDb framework) Modeli schema kullanarak Task ve Board için Modellerini oluşturacaktır ve ona göre verilerimiz databaseye kayıt edecek.

boardModel.js bu dosyada board name ve slugı verileri databaseye saklanacağız.

taskModel.js bu dosyada task title ,body ,task statusö boardId ve slugı verileri databaseye saklanacağız.

Not: Modelde Slugify kütüphane kullandım sebebi Task title ve Board name değeri slug değişkeni aynı değeri alacak ve routerde sorgulamada id yerinde kullanacağız.

Ör:

```
BoardSchema.pre("validate", function (next) {  
  this.slug = slugify(this.name, { lower: true, strict: true });  
  next();  
});
```

- 3) Middleware: bu Klasörde kullanıcı tarafından eklenecek taskı sadece Backlog, To do, in Progress ve Done dan status değeri seçsin,onlardan başka yasaklanma için kullandım.
- 4) Routes: bu Klasörde controllers klasörde kullanılan fonksiyonları Api router ile bağlanmak için kullanırız.

Ör:

```
// http://localhost:5000/api/boards  
router  
  .route("/")  
  .get(boardController.getAllBoards)  
  .post(boardController.createBoard);
```

.env bu dosyada database linki ve port numarası saklanacağız ve github a atarken github hesabımıza gönderilmez.

Ör:

MONGO_URL=mongodb://localhost:27017/RastTaskProject

PORT=5000

Ve son olarak **app.js** dosya,bu dosyada birazdan önceki anlattığımız klasörleri bu dosyada çağırarak ve mongoDb ile bağlayarak projemiz back end tarafında çalıştıracak.

Ör:

```
//connect DB
mongoose
  .connect(process.env.MONGO_URL, {
    useNewUrlParser: true,
    useUnifiedTopology: true,
  })
  .then(() => {
    console.log("connected to db sccessfully");
  })
  .catch((err) => {
    console.log(err);
  });
```

```
// Routes
app.use("/api/boards", boardRoutes);
app.use("/api/tasks", taskRoutes);
```

ve Api bilgileri front end tarafa göndermek için CORS kütüphane kullandım ve app.js çağırarak çalıştırırız.

```
const cors = require("cors");
const app = express();

app.use(cors());
```

package.json bu dosyada projedeki kullandığımız tüm kütüphaneleri bulunur.

Ör:

```
"dependencies": {
  "cors": "^2.8.5",
  "dotenv": "^10.0.0",
  "express": "^4.17.2",
  "mongoose": "^6.1.2",
  "slugify": "^1.6.4"
},
"devDependencies": {
  "nodemon": "^2.0.15"
}
```


2.2.Front end Kısmı:

bu kısımda Client Klasörü oluşturduk ve HTML5 ,SCSS ve React kullanarak oluşturduğumuz Apiyi front kısmı kodlayacağız.

Back endten gelen verileri CORS hataları önlemek için **package.json** dosyasında proxy kullanarak api port numarası yazarak veriler aktaracak.

```
"proxy": "http://localhost:5000/api",
```

Bizim projemiz App.js dosyasında başlıyor bu dosyada react-router-dom kütüphaneyi kullanarak Routelerimiz çağıracağız.

Ör:

```
import { BrowserRouter as Router, Switch, Route } from "react-router-dom";
```

```
function App() {  
  return (  
    <div className="app">  
      <Router>  
        <Switch>  
          <Route exact path="/">  
            <Home />  
          </Route>  
  
          <Route path="/:slug">  
            <Board />  
          </Route>  
        </Switch>  
      </Router>  
    </div>  
  );  
}
```

Ve terminalda npm start yazarak **localhost:3000/** bilgisayar tarayıcısı açılır ve Home komponent çağıracağız, **Home.jsx** dosyasında 3 komponent bulunur.

```
<div className="home">  
  <div className="homeBoardsAdd">  
    <div className="header">
```

```

        <Header title="Kanban Board" />
      </div>
      <BoardForm />
    </div>
    <div className="homeBoards">
      <Boards />
    </div>
  </div>

```

birinci olan **Header.jsx** komponent bu komponentte projen adı verini yazarız ve ikinci komponent **BoardForm.jsx** komponent bulunur, bu komponentte **submitHandler** fonksiyonu üzerinde useRef ve axios kütüphane kullanarak Board adı verini alınır ve Board adı en az 3 harftan oluşturması lazım bu yüzden inputa bir validation yapmak için **checkTitleValidation** fonksiyonu kodladık.

```

// board name input validation
const checkTitleValidation = (event) => {
  const val = event.target.value.trim();
  let valids = { ...titleValid };
  valids.touched = true;
  if (val.length <= 0) {
    valids.isValid = false;
    valids.errMsg = "Card title is Required";
  } else if (val.split(" ").length < 2) {
    valids.isValid = false;
    valids.errMsg = "Card title must be more than 2 words";
  } else {
    valids.isValid = true;
    valids.errMsg = "";
  }
  setTitleValid({ ...valids });
};

```

üçüncü komponent **Boards.jsx** bu komponentte db deki mevcut boardları **fetchBoards** fonksiyonu üzerinde axios ve useState(),useEffect react hooks ile kullanarak tüm boardların verileri aktaracağız ve **BoardCard.jsx** komponent üzerinde her board verileri tarayıcıda sunuacaktır ve Boardlar yoksa **NoCard.jsx** komponent sunacak bu komponent sadece ufak bir message (boardlar henüz yok ,ekleyebilirsiniz) diyor sunacak .

```

return (
  <div className="boards">
    {boards ? (
      boards.map((board) => {
        return (
          <BoardCard
            key={board._id}

```

```

        board={board}
        remove={() => deleteBoard(board.slug)}
      />
    );
  })
) : (
  // if no cards will show
  <NoCard title="there are no Boards yet" />

)
</div>
);

```

BoardCard.jsx komponentte delete butonu bulunur basınca istediğimiz board ,board slugı **deleteBoard** fonksiyona gönderilir ve bu fonksiyon üzerinde db deki boardu silinir.

```

// TO DELETE BOARD WITH ITS TASKS USING BOARD SLUG
const deleteBoard = async (slug) => {
  try {
    await axios.delete(`/boards/${slug}`);
    await fetchBoards();
  } catch (error) {
    console.log(error);
  }
};

```

Ve BoardCard title e bassak useParams() üzerinde bastığımız board slugı yakalayacağız ve bu Board ile alakalı tasklarını sunacak ve bu anda **Board.jsx** komponent çalışacak ve bu komponentte **fetchTasks** fonksiyon üzerinde taskları apiden axios get methodu ile sunacak.

```

// get tasks from api
const fetchTasks = async () => {
  try {
    const tasks = await axios.get(`/tasks/${slug}`);
    setCols(tasks.data.data);
    setBoardId(tasks.data.boardId);
  } catch (error) {
    console.log(error);
  }
};
useEffect(() => {
  fetchTasks();
}, []);

```

Objectteki mevcut sütunlar aktaracak ve her sütun **Col.jsx** komponenti olarak bilgilerini sunacak.

```

    {Object.entries(cols).length > 0 ? (
      Object.entries(cols).map(([_id, col], index) => {
        // send every object as a col in api [todo, done, backlog, inProgress] to col
        components
          return <Col key={_id} col={col} index={index}
            _id={_id} />;
      })
    ) : (
      <NoCard title="Obsess No Board with this name " />
    )}
  )}

```

Bu **Col.jsx** bu komponent de içindeki mevcut elementleri **item.jsx** komponent şekilde sunulacak.

```

    <div className="drop" style={{
      // to change color during drag
      backgroundColor: snapshot.isDraggingOver
        ? "#ad9393"
        : "#463b3b",}}
      ref={provider.innerRef}>
      {col.items.map((item, index) => {
        // to send single item card
        return <Item item={item} index={index}
          key={index} />;}})
      {provider.placeholder}
    </div>

```

Ve listeler arasında yeri değiştirmek için sürüküle bırak şekilde react-beautiful-dnd kütüphanesi üzerinde onDragEnd fonksiyonu kodladım bu fonksiyon 3 parametre alacak onlar: result, col, setCol ve 2 şart koyuyoruz 1.olan bir sütundan bir kart sürüküleyip başka sütuna aktarmak için çalışacak ve 2.olan aynı sütundaki task yerini değiştirebiliriz.

```

const onDragEnd = (result, cols, setCols) => {
  // avoid drag the div outside the div parent
  const { source, destination } = result;
  if (!destination) return;

  // to enable index left his place and drag drop it in the another
  column
  if (source.droppableId !== destination.droppableId) {
    // list name [Todo,inProgress.....]
    const sourceCol = cols[source.droppableId];
    const destCol = cols[destination.droppableId];
    // the items which have same list name
    const sourceItems = [...sourceCol.items];
    const destItems = [...destCol.items];

```

```

const [removed] = sourceItems.splice(source.index, 1);
destItems.splice(destination.index, 0, removed);

setCols({
  ...cols,
  [source.droppableId]: {
    ...sourceCol,
    items: sourceItems,
  },
  [destination.droppableId]: {
    ...destCol,
    items: destItems,
  },
});
}
// to enable index left his place and drag drap it in the same column
else {
  const columns = cols[source.droppableId];
  const _items = [...columns.items];
  const [removed] = _items.splice(source.index, 1);
  _items.splice(destination.index, 0, removed);
  setCols({
    ...cols,
    [source.droppableId]: {
      ...columns,
      items: _items,
    },
  });
}
}
};

```

Ve **Board.jsx** komponentte **Col.jsx** component ile 2 komponent de sunulacak **Model.jsx** ve **Header.jsx** ,**Header.jsx** komponentte board Adi title olarak sunulacak ve Modall komponenti için boardId göndereceğiz bu komponentte task forumu doldurarak yeni bir task oluşturulacak.

```

<div className="header">
  <Header title={` ${slug} Board `} />
  <div className="modal">
    <Modall slug={boardId} />
  </div>
</div>

```

Modall.jsx komponentte formu doldurup ve submit butona basarak **submitHandler** fonksiyonu çalışacak.

```

const submitHandler = async (e) => {
  e.preventDefault();
  const newPost = {

```

```

        title: title.current.value,
        body: body.current.value,
        status: status.current.value,
        color: color.current.value,
        boardId: boardId.current.value.toString(),
    };
    try {
        await axios.post("/tasks", newPost);
        window.location.reload();
    } catch (err) {
        console.log("err", err);
    }
};

```

Ve Board ekleme buton gibi validation için **checkTitleValidation** ve **checkTitleValidation** fonksiyonu kodladık.

Ör:

```

// body input validation
const checkBodyValidation = (event) => {
    const val = event.target.value.trim();
    let valids = { ...bodyValid };
    valids.touched = true;
    if (val.length <= 0) {
        valids.isValid = false;
        valids.errMsg = "Card body is Required";
    } else if (val.split(" ").length < 6) {
        valids.isValid = false;
        valids.errMsg = "Card body must be more than 6 words";
    } else {
        valids.isValid = true;
        valids.errMsg = "";
    }
    setBodyValid({ ...valids });
};

```

3.Proje Resimleri:

3.1 Büyük ekran:



cro-team Board

+ Add Card

Done

close the door
close the door close the door

28 seconds agoDelete

in Progress

Go office
Go office Go office

2 minutes agoDelete

Todo

sleep early
ad me this

58 seconds agoDelete

sleep
sleep at home

just nowDelete

Backlog

work night
work night

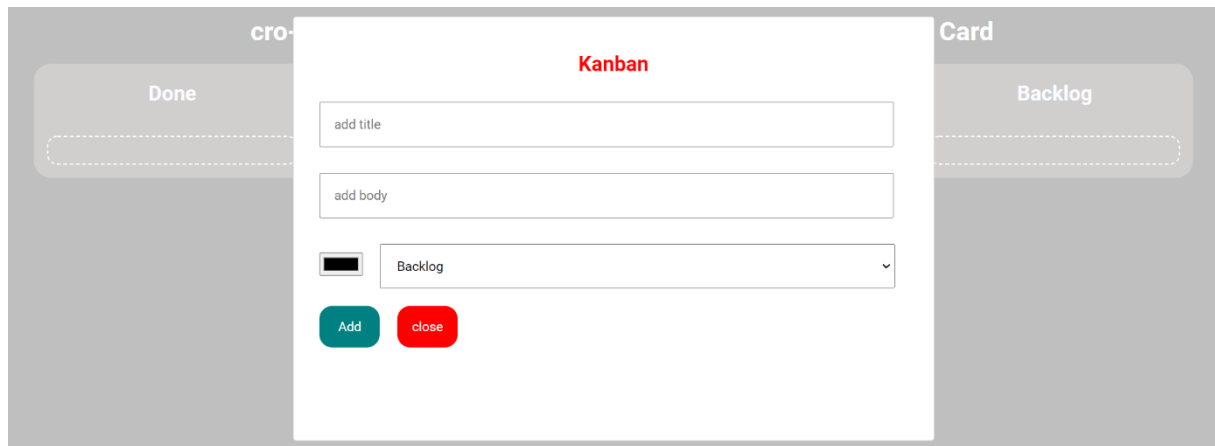
1 minute agoDelete

flower Board

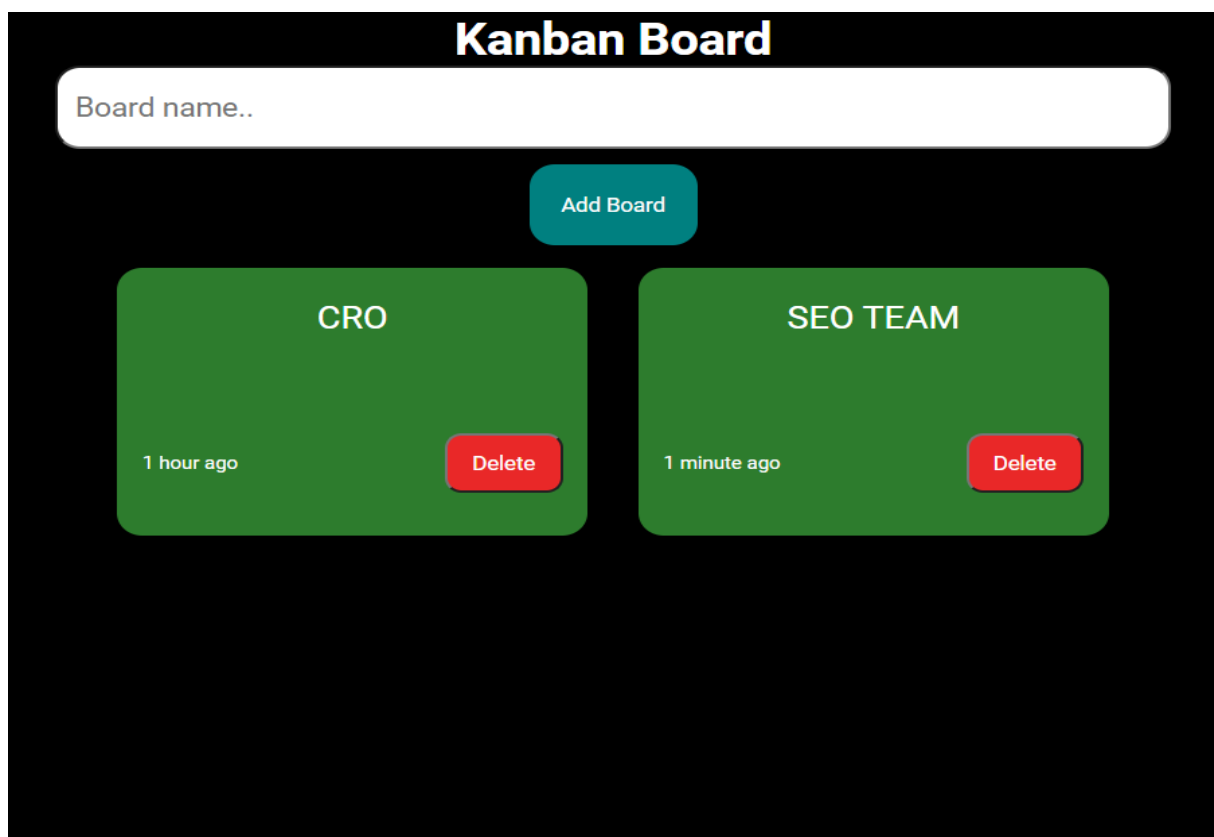
+ Add Card

Obsss No Board with this name

16



3.2 Orta ekran:



Kanban Board

Board name..

Add Board

there are no Boards yet

cro-team Board

+ Add Card

Done

in Progress

Todo

Backlog

cro-team Board

+ Add Card

Done

close the door
close the door close the door

28 seconds ago

Delete

in Progress

Go ofice
Go ofice Go ofice

2 minutes ago

Delete

Todo

sleep early
ad me this

58 seconds ago

Delete

sleep
sleep at home

just now

Delete

Backlog

work night
work night

1 minute ago

Delete

Kanban Board

Board name..

Add Board

there are no Boards yet

cro-


Card

ss

Kanban

add title

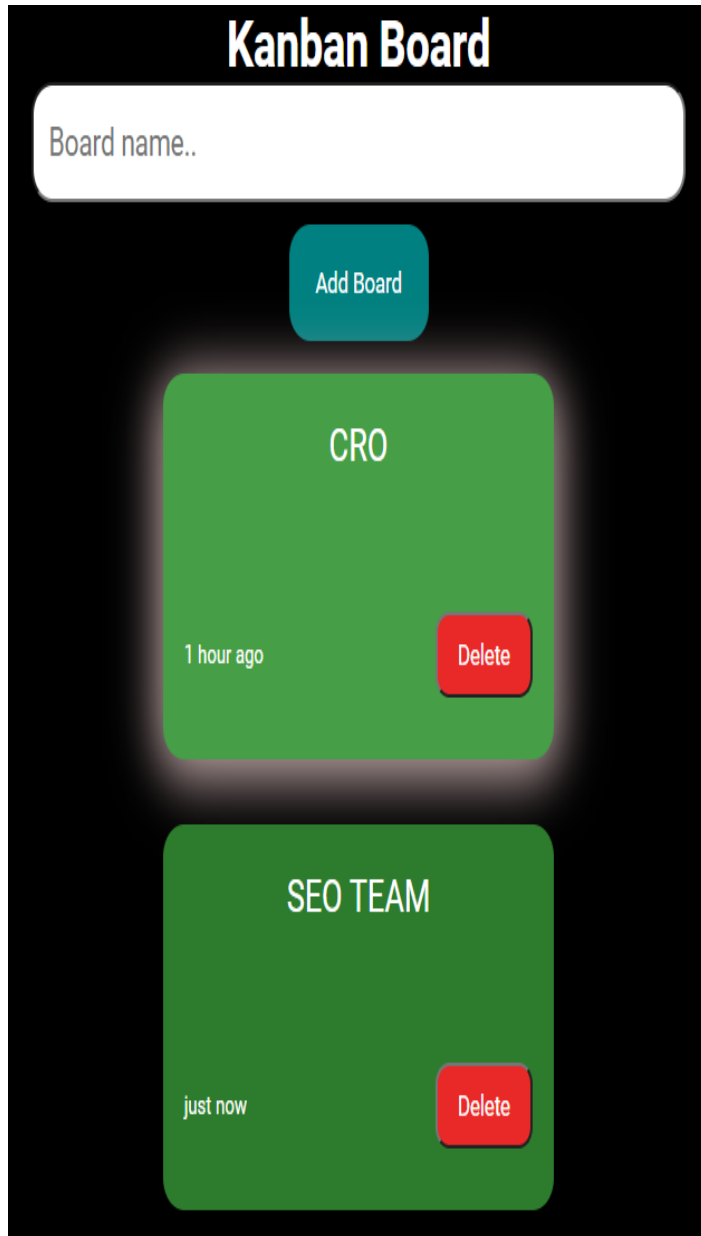
add body



Backlog

Addclose

3.3 Mobil ekran:



cro-team Board

+ Add Card

Done

close the door
close the door close the door

28 seconds agoDelete

in Progress

Go office
Go office Go office

2 minutes agoDelete

Todo

sleep early
ad me this

58 seconds agoDelete

sleep
sleep at home

just nowDelete

Backlog

work night
work night

1 minute agoDelete

cro-team Board

+ Add Card

Done



in Progress



Todo



Backlog



cro-team Board

+ Add Card

Kanban

add title

add body



Backlog



Add

close