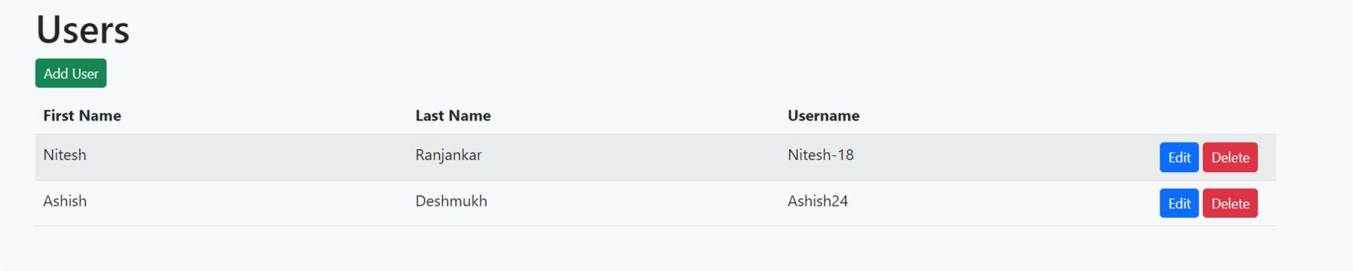
Create four API using Node.JS, ExpressJS and MongoDB for CURD Operations on assignment

Users-routing.module.ts:

```
import { Injectable } from '@angular/core';
import { Router } from '@angular/router';
import { HttpClient } from '@angular/common/http';
import { BehaviorSubject, Observable } from 'rxjs';
import { map } from 'rxjs/operators';

import { environment } from '@environments/environment';
import { User } from '@app/_models';

@Injectable({ providedIn: 'root' })
export class AccountService {
    private userSubject: BehaviorSubject<User | null>;
    public user: Observable<User | null>;

    constructor(
        private router: Router,
        private http: HttpClient
    ) {
        this.userSubject = new BehaviorSubject(JSON.parse(localStorage.getItem('user')!));
        this.user = this.userSubject.asObservable();
    }

    public get userValue() {
        return this.userSubject.value;
    }

    login(username: string, password: string) {
        return this.http.post<User>(`${environment.apiUrl}/users/authenticate`, { username, password })
            .pipe(map(user => {
                // store user details and jwt token in local storage to keep user logged in between page refreshes
```

```
            localStorage.setItem('user', JSON.stringify(user));
            this.userSubject.next(user);
            return user;
        }));
    }

    logout() {
        // remove user from local storage and set current user to null
        localStorage.removeItem('user');
        this.userSubject.next(null);
        this.router.navigate(['/account/login']);
    }

    register(user: User) {
        return this.http.post(`${environment.apiUrl}/users/register`, user);
    }

    getAll() {
        return this.http.get<User[]>(`${environment.apiUrl}/users`);
    }

    getById(id: string) {
        return this.http.get<User>(`${environment.apiUrl}/users/${id}`);
    }
update(id: string, params: any) {
        return this.http.put(`${environment.apiUrl}/users/${id}`, params)
            .pipe(map(x => {
                // update stored user if the logged in user updated their own record
                if (id == this.userValue?.id) {
                    // update local storage
                    const user = { ...this.userValue, ...params };
                    localStorage.setItem('user', JSON.stringify(user));

                    // publish updated user to subscribers
                    this.userSubject.next(user);
```

```
            }
            return x;
        }));
    }

    delete(id: string) {
        return this.http.delete(`${environment.apiUrl}/users/${id}`)
            .pipe(map(x => {
                // auto logout if the logged in user deleted their own record
                if (id == this.userValue?.id) {
                    this.logout();
                }
                return x;
            }));
    }
}
```



app-routing.module.ts:

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';

import { HomeComponent } from './home';
import { AuthGuard } from './_helpers';

const accountModule = () => import('./account/account.module').then(x => x.AccountModule);
const usersModule = () => import('./users/users.module').then(x => x.UsersModule);
```

```
const routes: Routes = [
    { path: '', component: HomeComponent, canActivate: [AuthGuard] },
    { path: 'users', loadChildren: usersModule, canActivate: [AuthGuard] },
    { path: 'account', loadChildren: accountModule },

    // otherwise redirect to home
    { path: '**', redirectTo: '' }
];

@NgModule({
    imports: [RouterModule.forRoot(routes)],
    exports: [RouterModule]
})
export class AppRoutingModule { }
```