9.a Course Name: Angular JS
Module Name: Server Communication using HttpClient
Create an application for Server Communication using HttpClient.
Program:
In the example used for custom services concept, add HttpModule to the app.module.ts to make use of HttpClient class.

- In the example used for custom services concept, add HttpModule to the **app.module.ts** to make use of HttpClient class.

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import {HttpClientModule} from '@angular/common/http';

import { AppComponent } from './app.component';
import { BookComponent } from './book/book.component';


@NgModule({
  imports: [BrowserModule, HttpClientModule],
  declarations: [AppComponent, BookComponent],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

- Add the following code in **book.service.ts** file

```
import { Injectable } from '@angular/core';
import { HttpClient, HttpErrorResponse, HttpHeaders } from '@angular/common/http';
import { Observable, throwError } from 'rxjs';
import { catchError, tap } from 'rxjs/operators';
import { Book } from './book';
@Injectable({
    providedIn:'root'
})
export class BookService {
  booksUrl = 'http://localhost:3020/bookList';
  constructor(private http: HttpClient) { }
  getBooks(): Observable<Book[]> {
    return this.http.get<Book[]>('http://localhost:3020/bookList').pipe(
      tap((data: any) => console.log('Data Fetched:' + JSON.stringify(data))),
      catchError(this.handleError));
  }
  addBook(book: Book): Observable<any> {
    const options = new HttpHeaders({ 'Content-Type': 'application/json' });
    return this.http.post('http://localhost:3020/addBook', book, { headers: options }).pipe(
      catchError(this.handleError));
  }
  updateBook(book: Book): Observable<any> {
    const options = new HttpHeaders({ 'Content-Type': 'application/json' });
    return this.http.put<any>('http://localhost:3020/update', book, { headers: options
}).pipe(
      tap((_: any) => console.log(`updated hero id=${book.id}`)),
      catchError(this.handleError)
    );
  }
  deleteBook(bookId: number): Observable<any> {
    const url = `${this.booksUrl}/${bookId}`;
    return this.http.delete(url).pipe(
      catchError(this.handleError));
  }
  private handleError(err: HttpErrorResponse): Observable<any> {
    let errMsg = '';
    if (err.error instanceof Error) {
// A client-side or network error occurred. Handle it accordingly.
      console.log('An error occurred:', err.error.message);
      errMsg = err.error.message;
    } else {
// The backend returned an unsuccessful response code.
// The response body may contain clues as to what went wrong,
      console.log(`Backend returned code ${err.status}`);
      errMsg = err.error.status;
    }
    return throwError(()=>errMsg);
  }
}
```

- Write the code given below in **book.component.ts**

```
import { Component, OnInit } from '@angular/core';
import { BookService } from './book.service';
import { Book } from './book';
@Component({
  selector: 'app-book',
  templateUrl: './book.component.html',
  styleUrls: ['./book.component.css']
})
```

```
export class BookComponent implements OnInit {
  title = 'Demo on HttpClientModule';
  books!: Book[];
  errorMessage!: string;
  ADD_BOOK!: boolean;
  UPDATE_BOOK!: boolean;
  DELETE_BOOK!: boolean;
  constructor(private bookService: BookService) { }
  getBooks() {
    this.bookService.getBooks().subscribe({
      next:  books => this.books = books,
      error:error => this.errorMessage = <any>error
    })
  }
  addBook(bookId: string, name: string): void {
    let id=parseInt(bookId)
    this.bookService.addBook({id, name })
      .subscribe({next:(book: any) => this.books.push(book)});
  }
  updateBook(bookId: string, name: string): void {
    let id=parseInt(bookId)
    this.bookService.updateBook({ id, name })
      .subscribe({next:(book: any) => this.books = book});
  }
  deleteBook(bookId: string): void {
    let id=parseInt(bookId)
    this.bookService.deleteBook(id)
      .subscribe({next:(book: any) => this.books = book});
  }
  ngOnInit() {
    this.getBooks();
  }
}
```

- Write the code given below in **book.component.html**

```
<h2>{{ title }}</h2>
<h2>My Books</h2>

<ul class="books">
<li *ngFor="let book of books">
<span class="badge">{{ book.id }}</span> {{ book.name }}
</li>
</ul>
```

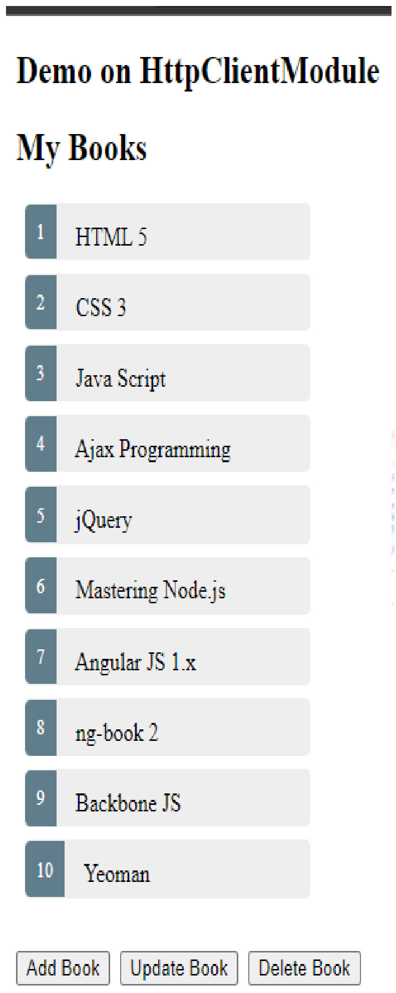
```

<button class="btn btn-primary" (click)="ADD_BOOK = true">Add Book</button>&nbsp;
<button class="btn btn-primary" (click)="UPDATE_BOOK = true">Update Book</button>&nbsp;
<button class="btn btn-primary" (click)="DELETE_BOOK = true">Delete Book</button>
<br />

<div *ngIf="ADD_BOOK">
<table>
<tr>
<td>Enter Id of the book:</td>
<td>
<input type="number" #id />
</td>
</tr>
<br />

<tr>
<td>Enter Name of the Book:</td>
<td>
<input type="text" #name />
<br />
</td>
</tr>
<br />

<tr>
<td>
<button class="btn btn-primary" (click)="addBook(id.value, name.value); ADD_BOOK = false">
        Add Record
</button>
</td>
</tr>
</table>

<br />
</div>

<div *ngIf="UPDATE_BOOK">
<table>
<tr>
<td>Enter Id of the book:</td>
<td>
```

```
<input type="number" #id />
</td>
</tr>
<br />

<tr>
<td>Enter Name of the Book:</td>
<td>
<input type="text" #name />
<br />
</td>
</tr>
<br />

<tr>
<td>
<button class="btn btn-primary" (click)="updateBook(id.value, name.value);
UPDATE_BOOK = false">
          Update Record
</button>
</td>
</tr>
</table>
</div>
<br />

<div *ngIf="DELETE_BOOK">
<table>
<tr>
<td>Enter Id of the book:</td>
<td>
<input type="number" #id />
</td>
</tr>
<br />

<tr>
<td>
<button class="btn btn-primary" (click)="deleteBook(id.value); DELETE_BOOK =
false">
          Delete Record
</button>
</td>
</tr>
</table>
</div>
```

```
95.

96. <div class="error" *ngIf="errorMessage">{{ errorMessage }}</div>
```

- Save the files and check the **output** in the browser

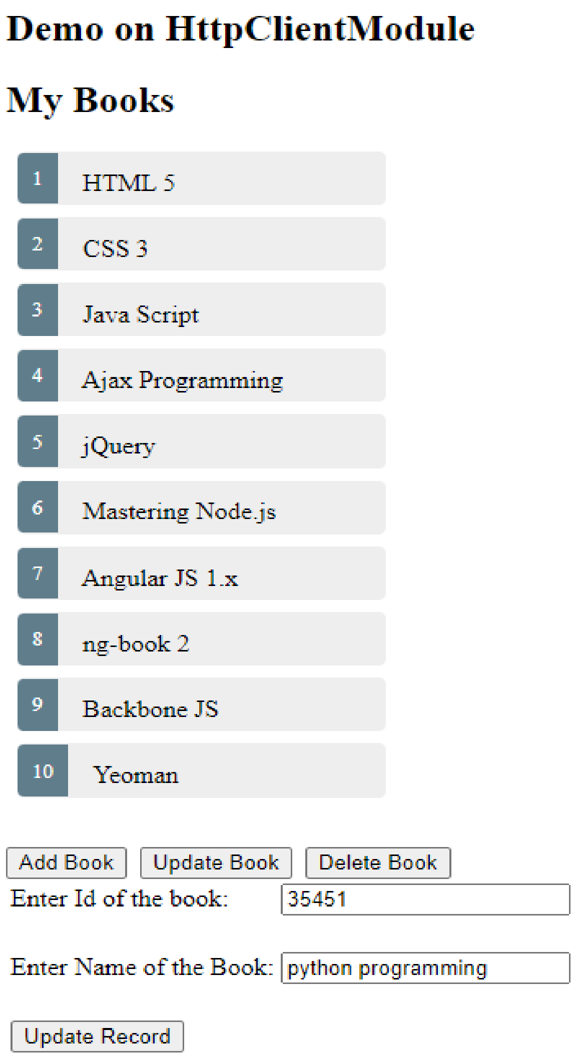
Output:



Demo on HttpClientModule

My Books

```
1  HTML 5
2  CSS 3
3  Java Script
4  Ajax Programming
5  jQuery
6  Mastering Node.js
7  Angular JS 1.x
8  ng-book 2
9  Backbone JS
10 Yeoman
```



Demo on HttpClientModule

My Books

```
1  HTML 5
2  CSS 3
3  Java Script
4  Ajax Programming
5  jQuery
6  Mastering Node.js
7  Angular JS 1.x
8  ng-book 2
9  Backbone JS
10 Yeoman
```

Enter Id of the book: 35451

Enter Name of the Book: python programming

9.b Course Name: Angular JS
Module Name: Communicating with different backend services using Angular HttpClient Create a custom service called ProductService in which Http class is used to fetch data stored in the JSON files
Program:

```
var MyApp = angular.module("MyApp", []);
```

```
Myapp.controller('MyController', function($scope, TaxFactory) {
/*calling service*/
TaxFactory.cal();
}])

// Services getting registered using Module Factory.

Myapp.factory('TaxFactory', function() {
// creating empty serviceobject
var calService = {};

// object with some bussiness logic
calService.cal = function() {
    ......
}

// returning object that can be used by the controller.
return calService;

}]);
```

We can go with any of the service/factory.

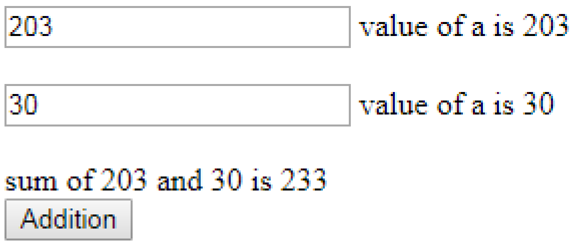
```
var MyApp = angular.module("MyApp", []);
Myapp.controller('MyController', function($scope, TaxService) {
/*calling service*/
TaxService.cal();
}])

// Services getting registered using Module Factory.

Myapp.service('TaxService', function() {

this.cal = function() {
    ......
};

..........
}]);
```

Myapp.config

```
var MyApp = angular.module("MyApp", []);
Myapp.controller('MyController', function($scope, TaxService) {
    TaxService.cal();
});
```

```
MyApp.provider('TaxService', function() {
this.cal = function() {
    ......
};
..........
return Taxservice
}]);

Myapp.config(["TaxServiceProvider", function(TaxServiceProvider) {
    TaxServiceProvider.config();
}]);
```

Output:



```
203     value of a is 203

30      value of a is 30

sum of 203 and 30 is 233
```

10.a Course Name: Angular JS
Module Name: Routing Basics, Router Links Create multiple components and add routing to provide navigation between them.
Program:
1. Consider the example used for the HttpClient concept.
2. Create another component with the name dashboard using the following command

```
D:\MyApp>ng generate component dashboard
```

3.Open dashboard.component.ts and add the following code

```
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { Book } from '../book/book';
import { BookService } from '../book/book.service';
@Component({
  selector: 'app-dashboard',
  templateUrl: './dashboard.component.html',
  styleUrls: ['./dashboard.component.css']
})
export class DashboardComponent implements OnInit {
  books: Book[] = [];
  constructor(
    private router: Router,
    private bookService: BookService) { }
  ngOnInit(): void {
    this.bookService.getBooks()
      .subscribe({next:books => this.books = books.slice(1, 5)});
  }
  gotoDetail(book: Book): void {
    this.router.navigate(['/detail', book.id]);
  }
}
```

4.Open dashboard.component.html and add the following code

```
<h3>Top Books</h3>
<div class="grid grid-pad">
<div *ngFor="let book of books" (click)="gotoDetail(book)" class="col-1-4">
<div class="module book">
<h4>{{ book.name }}</h4>
</div>
</div>
</div>
```

5.Open dashboard.component.css and add the following code

```
[class*="col-"] {
  float: left;
}
*,
*:after,
*:before {
  -webkit-box-sizing: border-box;
  -moz-box-sizing: border-box;
  box-sizing: border-box;
}
```

```
h3 {
  text-align: center;
  margin-bottom: 0;
}
[class*="col-"] {
  padding-right: 20px;
  padding-bottom: 20px;
}
[class*="col-"]:last-of-type {
  padding-right: 0;
}
.grid {
  margin: 0;
}
.col-1-4 {
  width: 25%;
}
.module {
  padding: 20px;
  text-align: center;
  color: #eee;
  max-height: 120px;
  min-width: 120px;
  background-color: #607d8b;
  border-radius: 2px;
}
h4 {
  position: relative;
}
.module:hover {
  background-color: #eee;
  cursor: pointer;
  color: #607d8b;
}
.grid-pad {
  padding: 10px 0;
}
.grid-pad > [class*="col-"]:last-of-type {
  padding-right: 20px;
}
@media (max-width: 600px) {
  .module {
    font-size: 10px;
    max-height: 75px;
  }
}
```

```
@media (max-width: 1024px) {
  .grid {
    margin: 0;
  }
  .module {
    min-width: 60px;
  }
}
```

6.Create another component called book-detail using the following command

D:\MyApp>ng generate component bookDetail

Open book.service.ts and add getbook() method as shown below to fetch specific book details

```
import { Injectable } from '@angular/core';
import { HttpClient, HttpErrorResponse, HttpHeaders, HttpResponse } from
'@angular/common/http';
import { Observable, throwError } from 'rxjs';
import { catchError, tap, map} from 'rxjs/operators';
import { Book } from './book';
@Injectable({
    providedIn:'root'
})
export class BookService {
  booksUrl = 'http://localhost:3020/bookList';
  private txtUrl = './assets/sample.txt';
  constructor(private http: HttpClient) { }
  getBooks(): Observable<Book[]> {
    return this.http.get<any>(this.booksUrl, {observe:'response'}).pipe(
      tap((data: any) => console.log('Data Fetched:' + JSON.stringify(data))),
     catchError(this.handleError));
  }
  getBook(id: any) {
    return this.getBooks().pipe(
      map((books) => books.find((book) => book.id == id))
    );
  }
  addBook(book: Book): Observable<any> {
    const options = new HttpHeaders({ 'Content-Type': 'application/json' });
    return this.http.post('http://localhost:3020/addBook', book, { headers: options }).pipe(
      catchError(this.handleError));
  }
  updateBook(book: Book): Observable<any> {
    const options = new HttpHeaders({ 'Content-Type': 'application/json' });
    return this.http.put<any>('http://localhost:3020/update', book, { headers: options }).pipe(
      tap((_: any) => console.log(`updated hero id=${book.id}`)),
      catchError(this.handleError)
    );
  }
  deleteBook(bookId: number): Observable<any> {
    const url = `${this.booksUrl}/${bookId}`;
    return this.http.delete(url).pipe(
      catchError(this.handleError));
  }
  private handleError(err: HttpErrorResponse): Observable<any> {
    let errMsg = '';
    if (err.error instanceof Error) {
      // A client-side or network error occurred. Handle it accordingly.
      console.log('An error occurred:', err.error.message);
      errMsg = err.error.message;
    } else {
      // The backend returned an unsuccessful response code.
      // The response body may contain clues as to what went wrong,
      console.log(`Backend returned code ${err.status}`);
      errMsg = err.error.status;
    }
    return throwError(()=>errMsg);
  }
}
```

8.Open book-detail.component.ts and add the following code

```
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute } from '@angular/router';
import { Book } from '../book/book';
import { BookService } from '../book/book.service';
@Component({
  selector: 'app-book-detail',
  templateUrl: './book-detail.component.html',
  styleUrls: ['./book-detail.component.css'],
})
export class BookDetailComponent implements OnInit {
  book!: Book;
  error!: any;
  constructor(
    private bookService: BookService,
    private route: ActivatedRoute
  ) { }
  ngOnInit() {
      this.route.paramsMap.subscribe(params => {
      this.bookService.getBook(params.get('id')).subscribe((book) => {
        this.book = book ?? this.book;
      });
    });
  }
  goBack() {
    window.history.back();
  }
}
```

9.Open book-detail.component.html and add the following code

```
<div *ngIf="book">
<h2>{{ book.name }} details!</h2>
<div><label>id: </label>{{ book.id }}</div>
<div>
<label>name: </label><input [(ngModel)]="book.name" placeholder="name" />
</div>
<button (click)="goBack()">Back</button>
</div>
```

10.Open book-detail.component.css and add the following code

```
label {
  display: inline-block;
  width: 3em;
  margin: 0.5em 0;
  color: #607d8b;
  font-weight: bold;
}
input {
  height: 2em;
  font-size: 1em;
  padding-left: 0.4em;
}
button {
  margin-top: 20px;
  font-family: Arial;
  background-color: #eee;
  border: none;
  padding: 5px 10px;
  border-radius: 4px;
  cursor: pointer;
  cursor: hand;
}
button:hover {
  background-color: #cfd8dc;
}
button:disabled {
  background-color: #eee;
  color: #ccc;
  cursor: auto;
}
```

11.Generate PageNotFound component using the following CLI command

```
D:\MyApp>ng g c PageNotFound
```

12.Add below code to page-not-found.component.html:

```
<div>
```

```
<h1>404 Error</h1>
<h1>Page Not Found</h1>
</div>
```

13.Add the below code to app-routing.module.ts:

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { BookComponent } from './book/book.component';
import { DashboardComponent } from './dashboard/dashboard.component';
import { BookDetailComponent } from './book-detail/book-detail.component';
import { PageNotFoundComponent } from './page-not-found/page-not-found.component';
const appRoutes: Routes = [
   { path: 'dashboard', component: DashboardComponent },
   { path: '', redirectTo: '/dashboard', pathMatch: 'full' },
   { path: 'books', component: BookComponent },
   { path: 'detail/:id', component: BookDetailComponent },
   { path: '**', component: PageNotFoundComponent },
];
@NgModule({
   imports: [
      RouterModule.forRoot(appRoutes)
   ],
   exports: [
      RouterModule
   ]
})
export class AppRoutingModule { }
```

14.Write the below-given code in app.module.ts

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { HttpClientModule } from '@angular/common/http';
import { FormsModule } from '@angular/forms';
import { AppComponent } from './app.component';
import { BookComponent } from './book/book.component';
import { DashboardComponent } from './dashboard/dashboard.component';
import { BookDetailComponent } from './book-detail/book-detail.component';
import { AppRoutingModule } from './app-routing.module';
import { PageNotFoundComponent } from './page-not-found/page-not-found.component';
@NgModule({
  imports: [BrowserModule, HttpClientModule, FormsModule, AppRoutingModule],
  declarations: [AppComponent, BookComponent, DashboardComponent,
BookDetailComponent, PageNotFoundComponent],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

15.Write the below-given code in app.component.ts

```
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  styleUrls: ['./app.component.css'],
  templateUrl: './app.component.html'
})
export class AppComponent {
  title = 'Tour of Books';
}
```

16.Write the below-given code in app.component.html

```
<h1>{{title}}</h1>
<nav>
<a [routerLink]='["/dashboard"]' routerLinkActive="active">Dashboard</a>
<a [routerLink]='["/books"]' routerLinkActive="active">Books</a>
</nav>
<router-outlet></router-outlet>
```

17.Open app.component.css and add the following code

```
/* Master Styles */
h1 {
  color: #369;
  font-family: Arial, Helvetica, sans-serif;
  font-size: 250%;
}
h2, h3 {
  color: #444;
  font-family: Arial, Helvetica, sans-serif;
  font-weight: lighter;
}
body {
  margin: 2em;
}
body, input[text], button {
  color: #888;
  font-family: Cambria, Georgia;
}
a {
  cursor: pointer;
  cursor: hand;
}
button {
  font-family: Arial;
  background-color: #eee;
  border: none;
  padding: 5px 10px;
  border-radius: 4px;
  cursor: pointer;
  cursor: hand;
}
button:hover {
  background-color: #cfd8dc;
}
button:disabled {
  background-color: #eee;
  color: #aaa;
  cursor: auto;
}
/* Navigation link styles */
nav a {
  padding: 5px 10px;
  text-decoration: none;
  margin-right: 10px;
  margin-top: 10px;
  display: inline-block;
  background-color: #eee;
  border-radius: 4px;
}
nav a:visited, a:link {
  color: #607D8B;
}
nav a:hover {
  color: #039be5;
  background-color: #CFD8DC;
}
nav a.active {
  color: #039be5;
}
/* everywhere else */
* {
  font-family: Arial, Helvetica, sans-serif;
}
```

18.Open styles.css under the src folder and add the following code

```
/* You can add global styles to this file, and also import other style files */
body{
    padding:10px;
}
```

19. Open book.component.ts file in book folder and add the following code

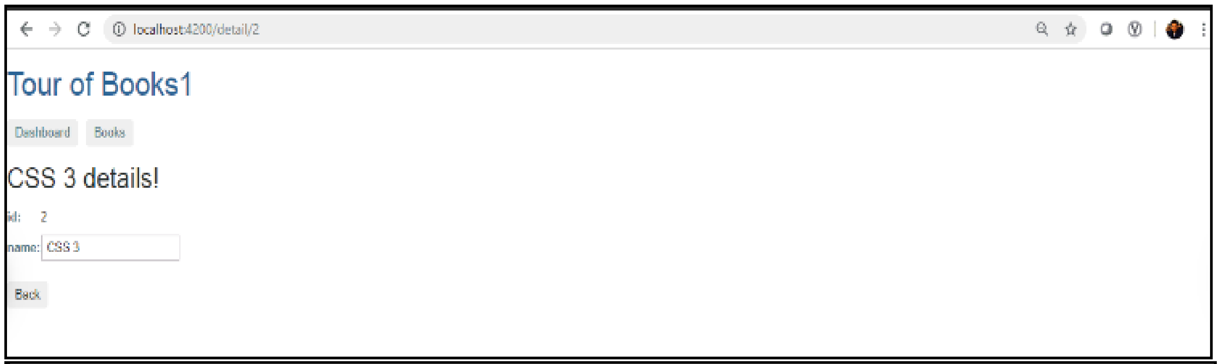
```
import { Component, OnInit } from '@angular/core';
import { Book } from './book';
import { BookService } from './book.service';
@Component({
  selector: 'app-book',
  templateUrl: './book.component.html',
  styleUrls: ['./book.component.css']
})
export class BookComponent implements OnInit {
  books!: Book[];
  errorMessage!: string;
  constructor(private bookService: BookService) { }
  getBooks() {
    this.bookService.getBooks().subscribe({
       next:  books => this.books = books,
       error:error => this.errorMessage = <any>error
     })
   }

  ngOnInit(): void {
    this.getBooks();
  }
}
```

20.Open book.component.html and update with below code.

```
<h2>My Books</h2>
<ul class="books">
<li *ngFor="let book of books">
<span class="badge">{{ book.id }}</span> {{ book.name }}
</li>
</ul>
<div class="error" *ngIf="errorMessage">{{ errorMessage }}</div>
```

21.Save the files and check the output in the browser.

Output:



localhost:4200/detail/2

Tour of Books1

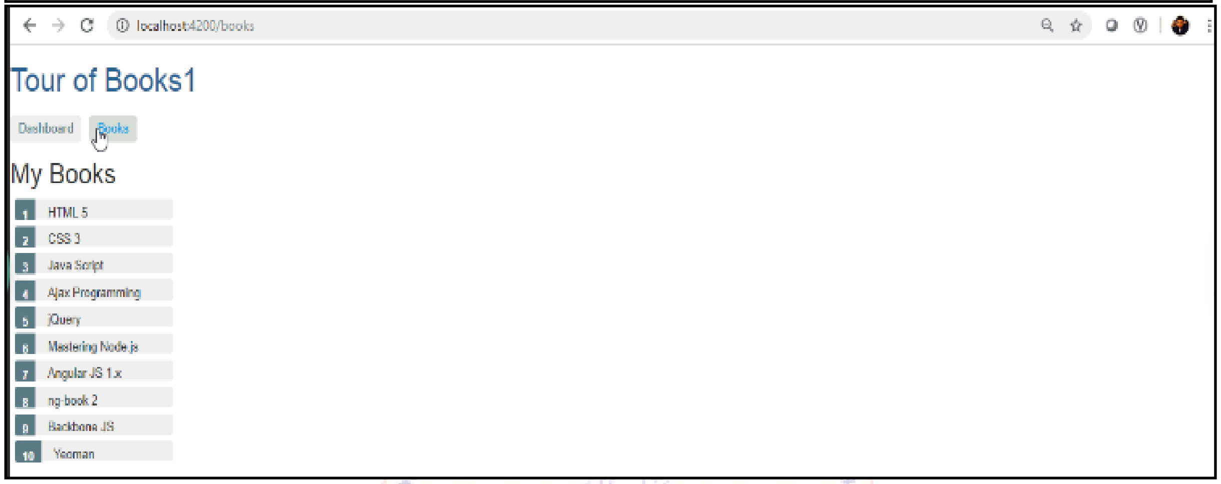
Dashboard Books

CSS 3 details!

id: 2

name: CSS 3

Back



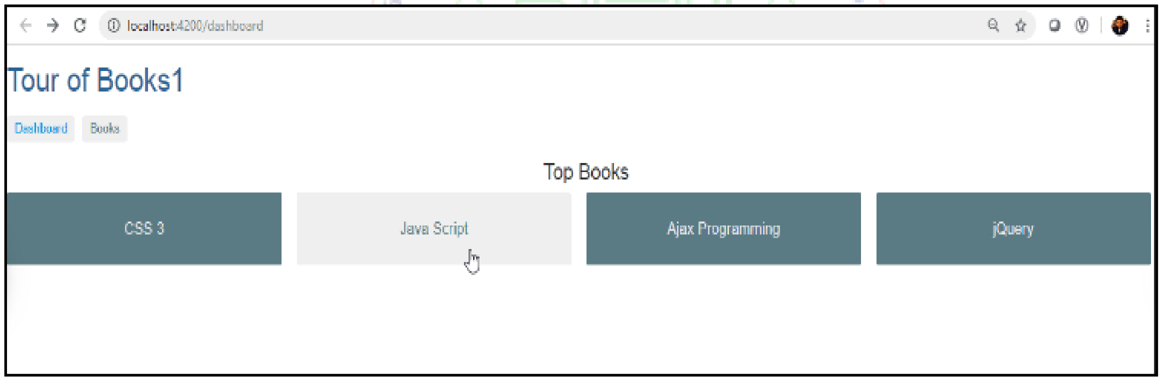
localhost:4200/books

Tour of Books1

Dashboard Books

My Books

1 HTML 5
2 CSS 3
3 Java Script
4 Ajax Programming
5 jQuery
6 Mastering Node.js
7 Angular JS 1.x
8 ng-book 2
9 Backbone JS
10 Yeoman



localhost:4200/dashboard

Tour of Books1

Dashboard Books

Top Books

CSS 3 Java Script Ajax Programming jQuery

10.b Course Name: Angular JS
Module Name: Route Guards
Considering the same example used for routing, add route guard to BooksComponent.

Only after logging in, the user should be able to access BooksComponent. If the user tries to give the URL of Bookscomponent in another tab or window, or if the user tries.

Program:

1.Create a component named LoginComponent and add the following code to the login.component.html file:

```
<h3 style="position: relative; left: 60px">Login Form</h3>
<div *ngIf="invalidCredentialMsg" style="color: red">
  {{ invalidCredentialMsg }}
</div>
<br />
<div style="position: relative; left: 20px">
<form [formGroup]="loginForm" (ngSubmit)="onFormSubmit()">
<p>User Name <input formControlName="username" /></p>
<p>
      Password
<input
        type="password"
        formControlName="password"
        style="position: relative; left: 10px"
      />
</p>
<p><button type="submit">Submit</button></p>
</form>
</div>
```

2.Add the following code to the login.component.ts file:

```
import { Component } from '@angular/core';
import { FormBuilder, FormGroup } from '@angular/forms';
import { Router } from '@angular/router';
import { LoginService } from './login.service';
@Component({
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css'],
})
export class LoginComponent {
  invalidCredentialMsg!: string;
  loginForm!: FormGroup;
  constructor(
    private loginService: LoginService,
    private router: Router,
    private formbuilder: FormBuilder
  ) {
    this.loginForm = this.formbuilder.group({
      username: [],
      password: [],
    });
  }
  onFormSubmit(): void {
    const uname = this.loginForm.value.username;
    const pwd = this.loginForm.value.password;
    this.loginService
      .isUserAuthenticated(uname, pwd)
      .subscribe({next:(authenticated) => {
        if (authenticated) {
          this.router.navigate(['/books']);
        } else {
          this.invalidCredentialMsg = 'Invalid Credentials. Try again.';
        }
      }});
  }
}
```

3.Create user.ts file under login folder and add the following code to user.ts file:

```
export class User {
  constructor(public userId: number, public username: string, public password: string) { }
}
```

4.Add the following code to the login.service.ts file inside login folder:

```
import { Injectable } from '@angular/core';
import { Observable, of } from 'rxjs';
import { map } from 'rxjs/operators';
import { User } from './user';
const USERS = [
    new User(1, 'user1', 'user1'),
    new User(2, 'user2', 'user2')
];
const usersObservable = of(USERS);
@Injectable({
    providedIn: 'root'
})
export class LoginService {
    private isloggedIn = false;
    getAllUsers(): Observable<User[]> {
        return usersObservable;
    }
    isUserAuthenticated(username: string, password: string): Observable<boolean> {
        return this.getAllUsers().pipe(
            map(users => {
                const Authenticateduser = users.find(user => (user.username === username) &&
(user.password === password));
                if (Authenticateduser) {
                    this.isloggedIn = true;
                } else {
```

```
                this.isloggedIn = false;
            }
            return this.isloggedIn;
        })
    );
  }
  isUserLoggedIn(): boolean {
    return this.isloggedIn;
  }
}
```

5.Create another service class called login-guard.service inside login folder and add the following code:

```
import { Injectable } from '@angular/core';
import { CanActivate, Router } from '@angular/router';
import { LoginService } from './login.service';
@Injectable({
  providedIn: 'root'
})
export class LoginGuardService implements CanActivate {
  constructor(private loginService: LoginService, private router: Router) { }
  canActivate(): boolean {
    if (this.loginService.isUserLoggedIn()) {
      return true;
    }
    this.router.navigate(['/login']);
    return false;
  }
}
```

6.Add the following code in app.module.ts:

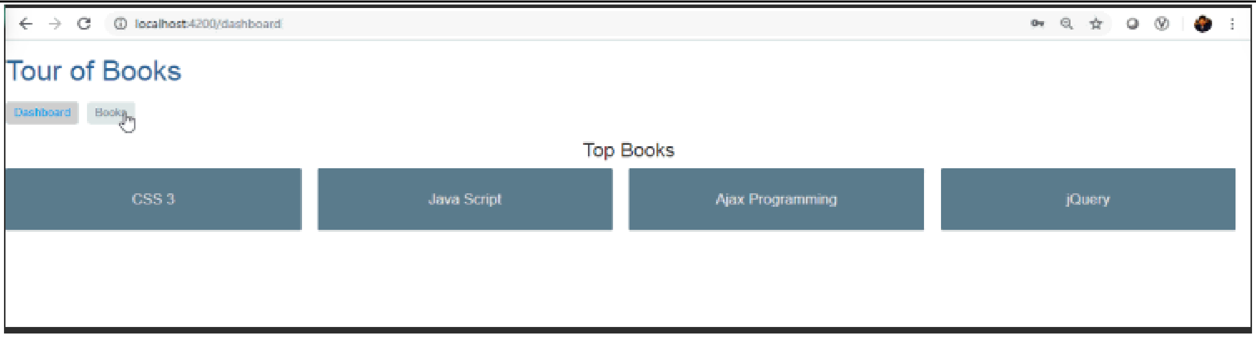
```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { HttpClientModule } from '@angular/common/http';
import { FormsModule, ReactiveFormsModule } from '@angular/forms';
import { AppComponent } from './app.component';
import { BookComponent } from './book/book.component';
import { DashboardComponent } from './dashboard/dashboard.component';
import { BookDetailComponent } from './book-detail/book-detail.component';
import { AppRoutingModule } from './app-routing.module';
import { PageNotFoundComponent } from './page-not-found/page-not-found.component';
import { LoginComponent } from './login/login.component';
@NgModule({
  imports: [BrowserModule, HttpClientModule, ReactiveFormsModule,FormsModule,
AppRoutingModule],
  declarations: [AppComponent, LoginComponent, BookComponent, DashboardComponent,
BookDetailComponent, PageNotFoundComponent],
  providers: [],
 bootstrap: [AppComponent]
})
export class AppModule { }
```

7.Add the following code to app-routing.module.ts:

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { BookComponent } from './book/book.component';
import { DashboardComponent } from './dashboard/dashboard.component';
import { BookDetailComponent } from './book-detail/book-detail.component';
import { PageNotFoundComponent } from './page-not-found/page-not-found.component';
import { LoginGuardService } from './login/login-guard.service';
import { LoginComponent } from './login/login.component';
const appRoutes: Routes = [
   { path: 'dashboard', component: DashboardComponent },
   { path: '', redirectTo: '/dashboard', pathMatch: 'full' },
   { path: 'books', component: BookComponent , canActivate:[LoginGuardService] },
   { path: 'detail/:id', component: BookDetailComponent },
   {path: 'login',component:LoginComponent},
   { path: '**', component: PageNotFoundComponent },
];
@NgModule({
   imports: [
      RouterModule.forRoot(appRoutes)
   ],
   exports: [
      RouterModule
   ]
})
export class AppRoutingModule { }
```

8.Save the files and check the output in the browser

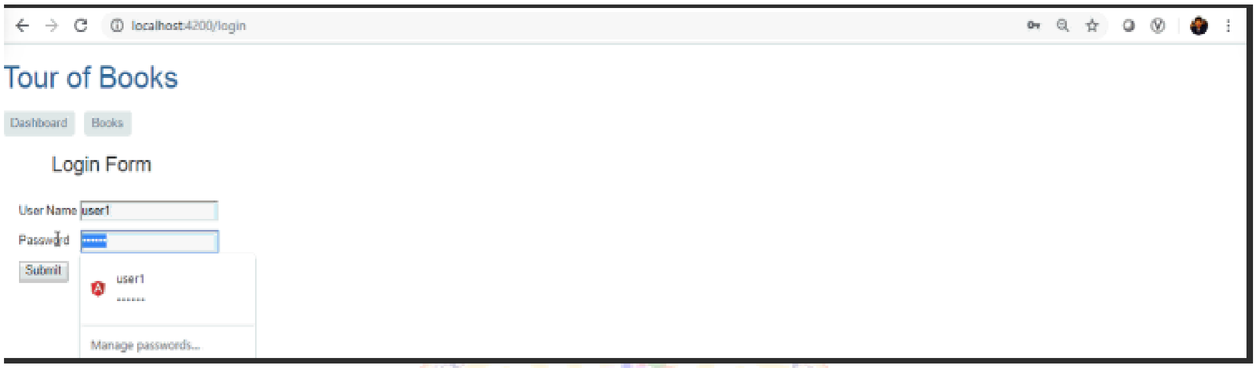
Output:



localhost:4200/dashboard

Tour of Books

Top Books

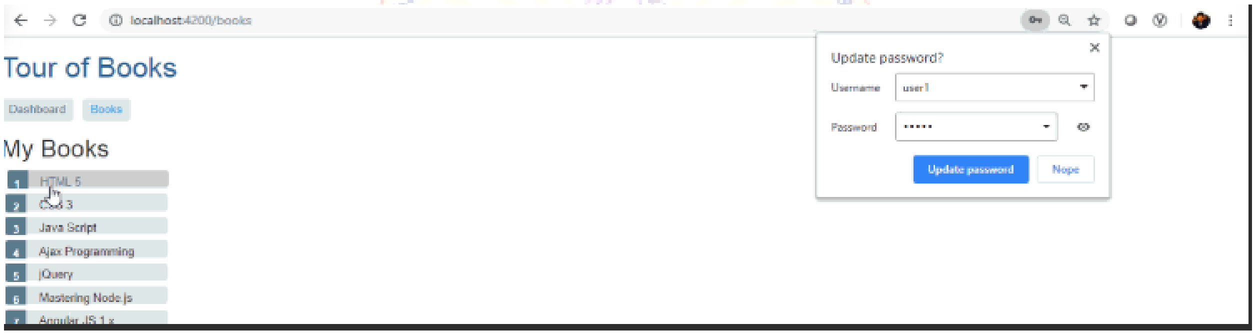


localhost:4200/login

Tour of Books

Login Form

User Name user1
Password ••••••



localhost:4200/books

Tour of Books

My Books

1 HTML 5
2 CSS 3
3 Java Script
4 Ajax Programming
5 jQuery
6 Mastering Node.js
7 Angular JS 1.x

Update password?
Username user1
Password •••••

10.c Course Name: Angular JS
Module Name: Asynchronous Routing
Apply lazy loading to BookComponent. If lazy loading is not added to the demo, it has loaded in 1.14 s. Observe the load time at the bottom of the browser console. Press F12 in the browser and click the Network tab and check the Load time.
Program:
1. Write the code given below in the book-routing.module.ts file inside book folder.

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { BookComponent } from './book.component';
import { LoginGuardService } from '../login/login-guard.service';
const bookRoutes: Routes = [
    {
        path: '',
        component: BookComponent,
        canActivate: [LoginGuardService]
    }
];
@NgModule({
    imports: [RouterModule.forChild(bookRoutes)],
    exports: [RouterModule]
})
export class BookRoutingModule { }
```

2.Create the book.module.ts file inside book folder and add the following code

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { BookComponent } from './book.component';
import { BookRoutingModule } from './book-routing.module';
@NgModule({
  imports: [CommonModule, BookRoutingModule],
  declarations: [BookComponent]
})
export class BookModule { }
```

3.Add the following code to the app-routing.module.ts file

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { BookDetailComponent } from './book-detail/book-detail.component';
import { BookComponent } from './book/book.component';
import { DashboardComponent } from './dashboard/dashboard.component';
import { LoginGuardService } from './login/login-guard.service';
import { LoginComponent } from './login/login.component';
import { PageNotFoundComponent } from './page-not-found/page-not-found.component';
```

```
const appRoutes: Routes = [
   { path: '', redirectTo: '/login', pathMatch: 'full' },
   { path: 'login', component: LoginComponent },
   { path: 'books', loadChildren: () => import('./book/book.module').then(m => m.BookModule)
},
   { path: 'dashboard', component: DashboardComponent },
   { path: 'detail/:id', component: BookDetailComponent} ,
   { path: '**', component: PageNotFoundComponent }
];
@NgModule({
   imports: [
      RouterModule.forRoot(appRoutes)
   ],
   exports: [
      RouterModule
   ]
})
export class AppRoutingModule { }
```

4.Add the following code to the app.module.ts file

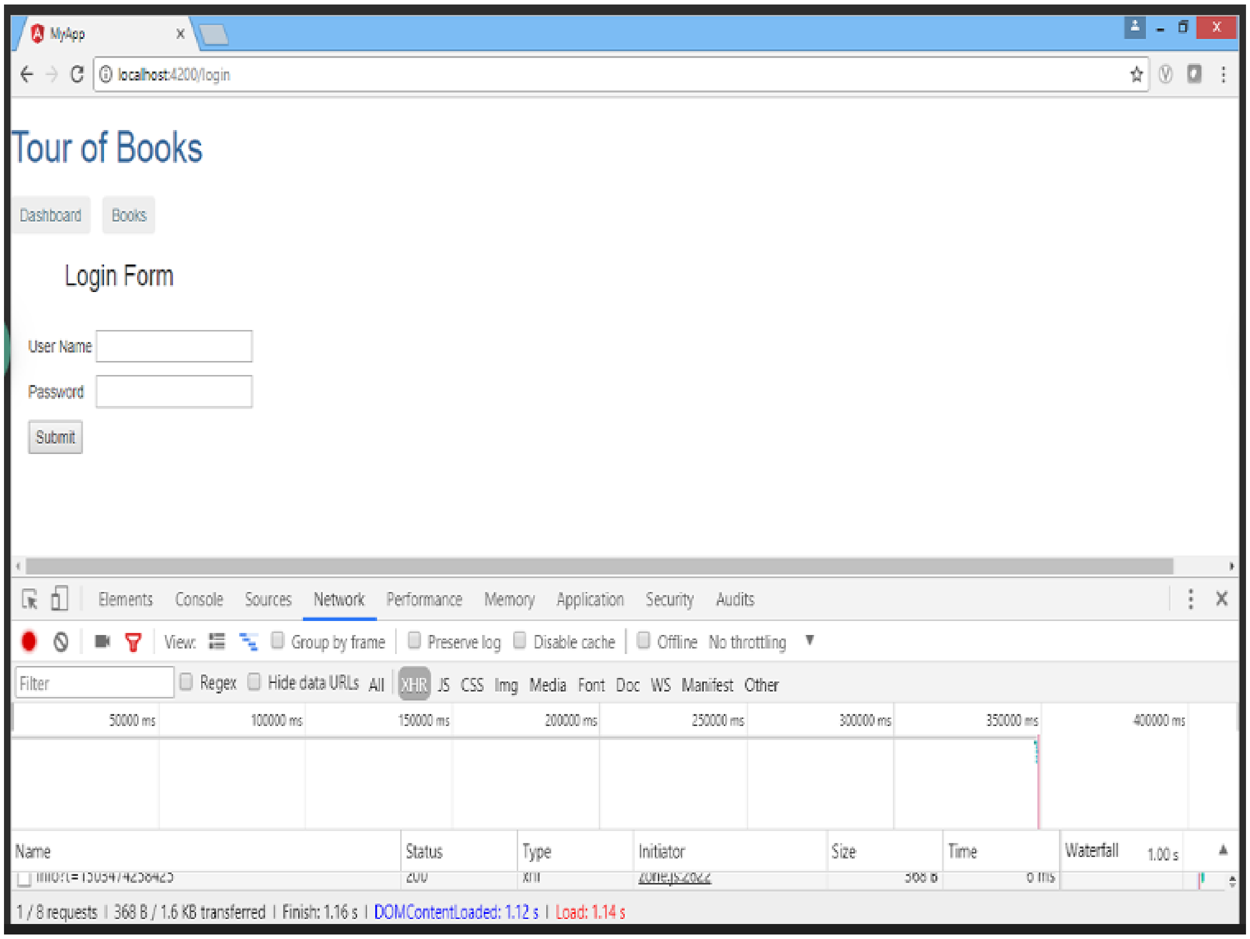
```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { HttpClientModule } from '@angular/common/http';
import { FormsModule, ReactiveFormsModule } from '@angular/forms';
import { AppComponent } from './app.component';
import { BookComponent } from './book/book.component';
import { DashboardComponent } from './dashboard/dashboard.component';
import { BookDetailComponent } from './book-detail/book-detail.component';
import { AppRoutingModule } from './app-routing.module';
import { PageNotFoundComponent } from './page-not-found/page-not-found.component';
import { LoginComponent } from './login/login.component';
@NgModule({
  imports: [BrowserModule, HttpClientModule, ReactiveFormsModule,FormsModule,
AppRoutingModule],
  declarations: [AppComponent, LoginComponent, DashboardComponent,
BookDetailComponent, PageNotFoundComponent],
  providers: [],
  bootstrap: [AppComponent]
})
```

export class AppModule { }

5.Save the files and check the output in the browser

<u>Output</u>:

If lazy loading is not added to the demo, it has loaded in 1.14 s. Observe the load time at the bottom of the browser console. Press F12 in the browser and click the Network tab and check the Load time



Login Form

User Name

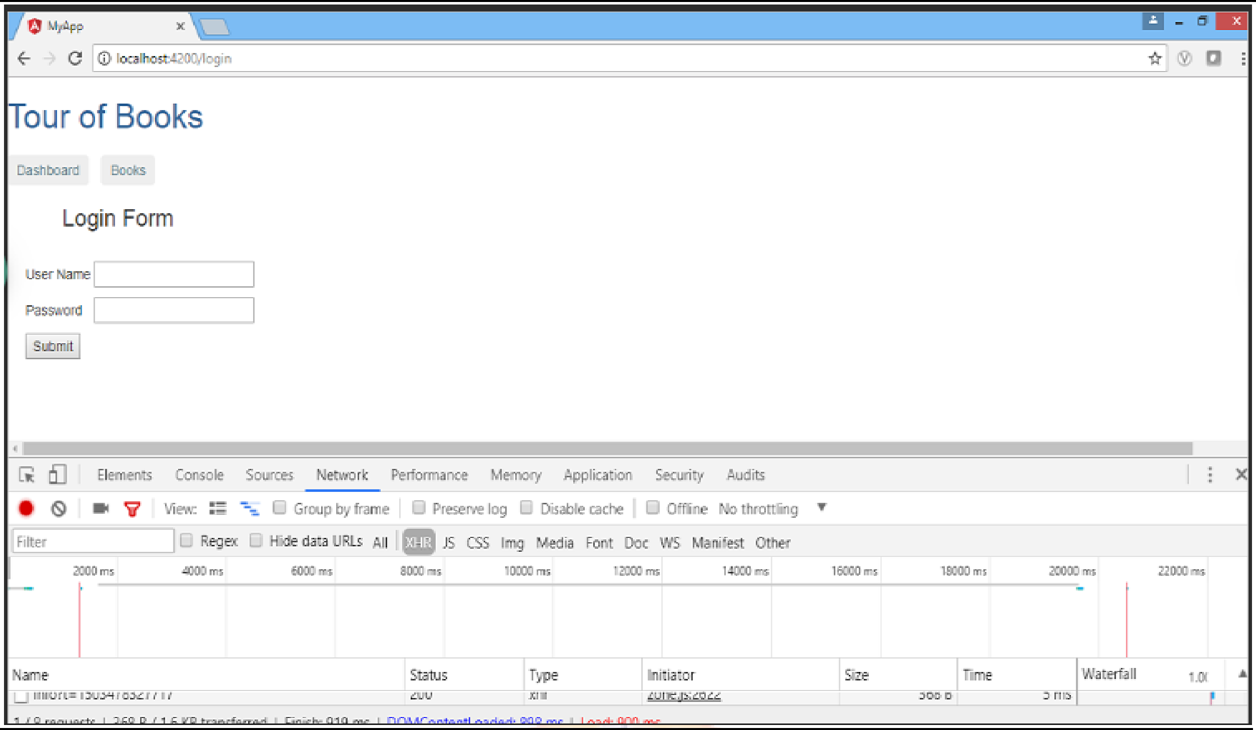
Password

Submit

Name	Status	Type	Initiator	Size	Time
info?t=1505474258425	200	xhr	zone.js:2022	368 B	6 ms

1 / 8 requests | 368 B / 1.6 KB transferred | Finish: 1.16 s | DOMContentLoaded: 1.12 s | Load: 1.14 s

If lazy loading is added to the demo, it has loaded in 900 ms. As BookComponent will be loaded after login, the load time is reduced initially.


MyApp
localhost:4200/login
Elements Console Sources Network Performance Memory Application Security Audits
View: Group by frame Preserve log Disable cache Offline No throttling
Filter Regex Hide data URLs All XHR JS CSS Img Media Font Doc WS Manifest Other
2000 ms 4000 ms 6000 ms 8000 ms 10000 ms 12000 ms 14000 ms 16000 ms 18000 ms 20000 ms 22000 ms

Tour of Books

Dashboard Books

Login Form

User Name

Password

Submit

Name	Status	Type	Initiator	Size	Time	Waterfall
info?t=1503478327717	200	xhr	zone.js:2622	368 B	5 ms	

1 / 8 requests | 368 B / 1.6 KB transferred | Finish: 919 ms | DOMContentLoaded: 898 ms | Load: 900 ms

10.d Course Name: Angular JS
Module Name: Nested Routes Implement Child Routes to a submodule.
Program:
1.Open the code inside the app-routing.module.ts file under the app folder

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { WelcomeComponent } from './welcome/welcome.component';
import { LoginComponent } from './login/login.component';
// Routing configuration
const appRoutes: Routes = [
  { path: 'welcome', component: WelcomeComponent },
  { path: 'login', component: LoginComponent },
  { path: 'products', loadChildren: () =>
import('./products/products.module').then(m=>m.ProductsModule) },
  { path: '', redirectTo: '/welcome', pathMatch: 'full' }
];
@NgModule({
  imports: [
    RouterModule.forRoot(appRoutes)
  ],
  exports: [
    RouterModule
  ]
})
export class AppRoutingModule { }
```

2.Understand the code in the app.module.ts file to explore the addition routing module to the root module

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { ReactiveFormsModule } from '@angular/forms';
import { HttpClientModule } from '@angular/common/http';
import { AppComponent } from './app.component';
import { AppRoutingModule } from './app-routing.module';
import { WelcomeComponent } from './welcome/welcome.component';
import { LoginComponent } from './login/login.component';
@NgModule({
    imports: [BrowserModule, HttpClientModule, ReactiveFormsModule, AppRoutingModule],
    declarations: [AppComponent, WelcomeComponent, LoginComponent],
    providers: [],
    bootstrap: [AppComponent]
})
export class AppModule { }
```

3.To understand the code inside the product-routing.module.ts file under the products folder.

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { ProductListComponent } from './product-list/product-list.component';
import { ProductDetailComponent } from './product-detail/product-detail.component';
import { CartComponent } from './cart/cart.component';
import { AuthGuardService } from './auth-guard.service';
// Child routes for products page
const routes: Routes = [
  {
    path: '',
    children: [
      { path: '', component: ProductListComponent },
      { path: 'cart', component: CartComponent },
      { path: ':id', component: ProductDetailComponent }
    ],
    canActivate: [AuthGuardService]
  }];
@NgModule({
  imports: [RouterModule.forChild(routes)],
  exports: [RouterModule]
})
export class ProductsRoutingModule { }
```

4.Understand the code in the products.module.ts file, explore the addition of products routing module to it. This module has been lazily loaded in the app-routing.module.ts.

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { FormsModule } from '@angular/forms';
import { ProductsRoutingModule } from './products-routing.module';
import { ProductListComponent } from './product-list/product-list.component';
import { ProductDetailComponent } from './product-detail/product-detail.component';
import { CartComponent } from './cart/cart.component';
import { OrderByPipe } from './product-list/orderby.pipe';
import { RatingComponent } from './rating.component';
import { ProductService } from './product.service';
import { AuthGuardService } from './auth-guard.service';
@NgModule({
  imports: [
    CommonModule,
    FormsModule,
    ProductsRoutingModule
  ],
  declarations: [ProductListComponent, ProductDetailComponent, CartComponent,
OrderByPipe, RatingComponent],
  providers: [ProductService, AuthGuardService]
})
```

```
export class ProductsModule { }
```

5.Explore the code present on the product-list.component.html page to understand the implementation of routing

```
<nav class='navbar navbar-default navbar-fixed-top navbarpos'>
        <div class='container-fluid'>
                <a class='navbar-brand txtcolor'>{{pageTitle}} <span
                        class="glyphicon glyphicon-shopping-cart txtcolor"></span></a>
                <div class="input-group pull-right col-md-3 searchboxpos">
                        <input type="text" class="form-control" placeholder="Search" name="q"
[(ngModel)]="listFilter"
                                (change)="searchtext()">
                        <div class="input-group-btn">
                                <button class="btn btn-default">
                                        <em class="glyphicon glyphicon-search"></em>
                                </button>
                        </div>
                </div>
                <!-- Displays the selected Items in the cart along with price -->
                <div class="pull-right txtcolor cartpos">
                        <span class="glyphicon glyphicon-shopping-cart"></span><a
[routerLink]="['cart']"
                                class="txtcolor">{{selectedItems}}&nbsp;items</a>
                        <span>, {{total | currency:'USD':'symbol':'1.2-2'}} </span>
                </div>
        </div>
</nav>
<br />
<br />
<!-- Displays a carousel with the given images -->
<div class="container" class="carouselpos">
        <div id="carousel-example-generic" class="carousel slide carouselheight" data-
ride="carousel" data-interval="3000">
                <ol class="carousel-indicators">
                        <li data-target="#carousel-example-generic" data-slide-to="0"
class="active"></li>
                        <li data-target="#carousel-example-generic" data-slide-to="1"></li>
                        <li data-target="#carousel-example-generic" data-slide-to="2"></li>
                </ol>
                <div class="carousel-inner">
                        <div class="item active">
                                <img src="assets/imgs/carousel_smart_phone.jpg" alt="First slide"
style="min-width:100%;height:350px;">
                        </div>
                        <div class="item carouselimgpos">
```

```
                                        <img src="assets/imgs/carousel1.jpg" alt="Second slide"
style="min-width:100%;height:350px;">
                              </div>
                              <div class="item">
                                        <img src="assets/imgs/tablet_blue_stylus.jpg" alt="Third slide"
style="min-width:100%;height:350px;">
                              </div>
                    </div>
                    <a class="left carousel-control" href="#carousel-example-generic" role="button"
data-slide="prev">
                              <span class="glyphicon glyphicon-chevron-left"></span>
                    </a>
                    <a class="right carousel-control" href="#carousel-example-generic"
role="button" data-slide="next">
                              <span class="glyphicon glyphicon-chevron-right"></span>
                    </a>
          </div>
          <!-- Displays two tabs called Tablets and Mobiles -->
          <div class='panel with-nav-tabs panel-primary noborder'>
                    <div class='panel-heading noborder bgcolor'>
                              <ul class="nav nav-tabs noborder">
                                        <li class="active tabpos nav-item">
                                                  <a class="nav-link active" (click)="tabselect('tablet')" data-
toggle="tab">

                                                            <i class="fa fa-tablet fa-3x" aria-
hidden="true"></i>
                                                            <div>Tablets</div>
                                                  </a>
                                        </li>
                                        <li class="tabpos nav-item">
                                                  <a class="nav-link active" (click)="tabselect('mobile')"
data-toggle="tab"><i
                                                                      class="fa fa-mobile fa-3x" aria-
hidden="true"></i>
                                                            <div>Mobiles</div>
                                                  </a>
                                        </li>
                              </ul>
                    </div>
                    <div class='panel-body'>
                              <div class="tab-content">
                                        <!-- Filtering dropdown -->
                                        <div class="tab-pane fade in active" id="tabprimary">
                                                  <div class="btn-group">
                                                            <button type="button" class="btn btn-
default">Filter</button>
```

```
                                        <button type="button" class="btn btn-default
dropdown-toggle" data-toggle="dropdown">
                                              <span class="caret"></span><span
class="sr-only">Toggle Dropdown</span>
                                        </button>
                                        <ul class="dropdown-menu multi-column columns-
3 noclose">
                                              <div class="row vdivide">
                                                    <div class="col-md-4">
                                                          <ul class="multi-column-
dropdown noclose">

      <h4>Manufacturer</h4>
                                                                <li *ngFor="let
manufac of manufacturers">
                                                                      <input
type="checkbox" [ngModel]="manufac.checked"

      (change)="filter(manufac)">

      <label>{{manufac.id}} </label>
                                                                </li>
                                                          </ul>
                                                    </div>
                                                    <div class="col-md-4">
                                                          <ul class="multi-column-
dropdown noclose">
                                                                <h4>OS</h4>
                                                                <li *ngFor="let
ostypes of os">
                                                                      <input
type="checkbox" [ngModel]="ostypes.checked"

      (change)="filter(ostypes)">
                                                                      <label>
{{ostypes.id}}</label>
                                                                </li>
                                                          </ul>
                                                    </div>
                                                    <div class="col-md-4">
                                                          <ul class="multi-column-
dropdown noclose">
                                                                <h4>Price
Range</h4>
                                                                <li *ngFor="let price
of price_range">
```

```
                                                                                                <input
type="checkbox" [ngModel]="price.checked" (change)=filter(price)>
                                                                                                <label>{{
price.id}} </label>
                                                                                        </li>
                                                                                </ul>
                                                                        </div>
                                                                </div>
                                                        </ul>
                                                </div>
                                                <span *ngIf="chkmanosprice.length> 0">
{{products.length}} results</span>
                                                <!-- sort dropdown -->
                                                <div class="pull-right">
                                                        <span>Sort By </span>
                                                        <select [ngModel]="sortoption" #sortBy
(change)="onChange(sortBy.value)">
                                                                <option
value="popularity">Popularity</option>
                                                                <option value="pricelh">Price - Low to
High</option>
                                                                <option value="pricehl">Price - High to
Low</option>
                                                        </select>
                                                </div>
                                                <!-- Displays the products data -->
                                                <div *ngIf='products && products.length'>
                                                        <div class="row" *ngFor='let product of products |
orderBy:sortoption ; let i = index'
                                                                [hidden]="(i%4)>0">
                                                                <div class="col-xs-3">
                                                                        <span class="thumbnail text-
center">
                                                                                <div>
                                                                                        <img
[src]='product.imageUrl' [title]='product.productName'

        [style.width.px]='imageWidth' [style.height.px]='imageHeight'

        [style.margin.px]='imageMargin' alt='Product Image'>
                                                                                </div>
                                                                                <div class="caption">
                                                                                        <div>
                                                                                                <a
[routerLink]="[product.productId]">
```

```
          {{product.productName}} </a>
                                                                                </div>
                                                                                <div>{{ product.price
| currency:'USD':'symbol':'1.2-2'}}</div>
                                                                                <div></div>
                                                                                <app-rating
class="ratingcolor" [rate]='product.rating'></app-rating>
                                                                                <div>
                                                                                        <button
(click)="addCart(product.productId)" class="btn btn-primary">Add to

          Cart</button>
                                                                                </div>
                                                                        </div>
                                                                </span>
                                                        </div>
                                                        <div class="col-xs-3">
                                                                <div *ngIf="products[i+1]"
class="thumbnail text-center">
                                                                        <div>
                                                                                <img
[src]='products[i+1].imageUrl' [title]='products[i+1].productName'

          [style.width.px]='imageWidth' [style.height.px]='imageHeight'

          [style.margin.px]='imageMargin'>
                                                                        </div>
                                                                        <div class="caption">
                                                                                <div>
                                                                                        <a
[routerLink]="[products[i+1].productId]">

          {{products[i+1].productName}} </a>
                                                                                </div>
                                                                                <div>{{
products[i+1].price | currency:'USD':'symbol':'1.2-2'}}
                                                                                </div>
                                                                                <div></div>
                                                                                <app-rating
class="ratingcolor" [rate]='products[i+1].rating'></app-rating>
                                                                                <div></div>
                                                                                <div>
                                                                                        <button
(click)="addCart(products[i+1].productId)"
```

```
            class="btn btn-primary">Add to Cart</button>
                                                                    </div>
                                                              </div>
                                                        </div>
                                                  </div>
                                                  <div class="col-xs-3">
                                                        <div *ngIf="products[i+2]"
class="thumbnail text-center">
                                                              <div>
                                                                    <img
[src]='products[i+2].imageUrl' [title]='products[i+2].productName'

            [style.width.px]='imageWidth' [style.height.px]='imageHeight'

            [style.margin.px]='imageMargin' alt='Product'>
                                                              </div>
                                                              <div class="caption">
                                                                    <div>
                                                                          <a
[routerLink]="[products[i+2].productId]">

            {{products[i+2].productName}} </a>
                                                                    </div>
                                                                    <div>{{
products[i+2].price | currency:'USD':'symbol':'1.2-2'}}
                                                                    </div>
                                                                    <div></div>
                                                                    <app-rating
class="ratingcolor" [rate]='products[i+2].rating'></app-rating>
                                                                    <div></div>
                                                                    <div>
                                                                          <button
(click)="addCart(products[i+2].productId)"

            class="btn btn-primary">Add to Cart</button>
                                                                    </div>
                                                              </div>
                                                        </div>
                                                  </div>
                                                  <div class="col-xs-3">
                                                        <div *ngIf="products[i+3]"
class="thumbnail text-center">
                                                              <div>
                                                                    <img
[src]='products[i+3].imageUrl' [title]='products[i+3].productName'
```

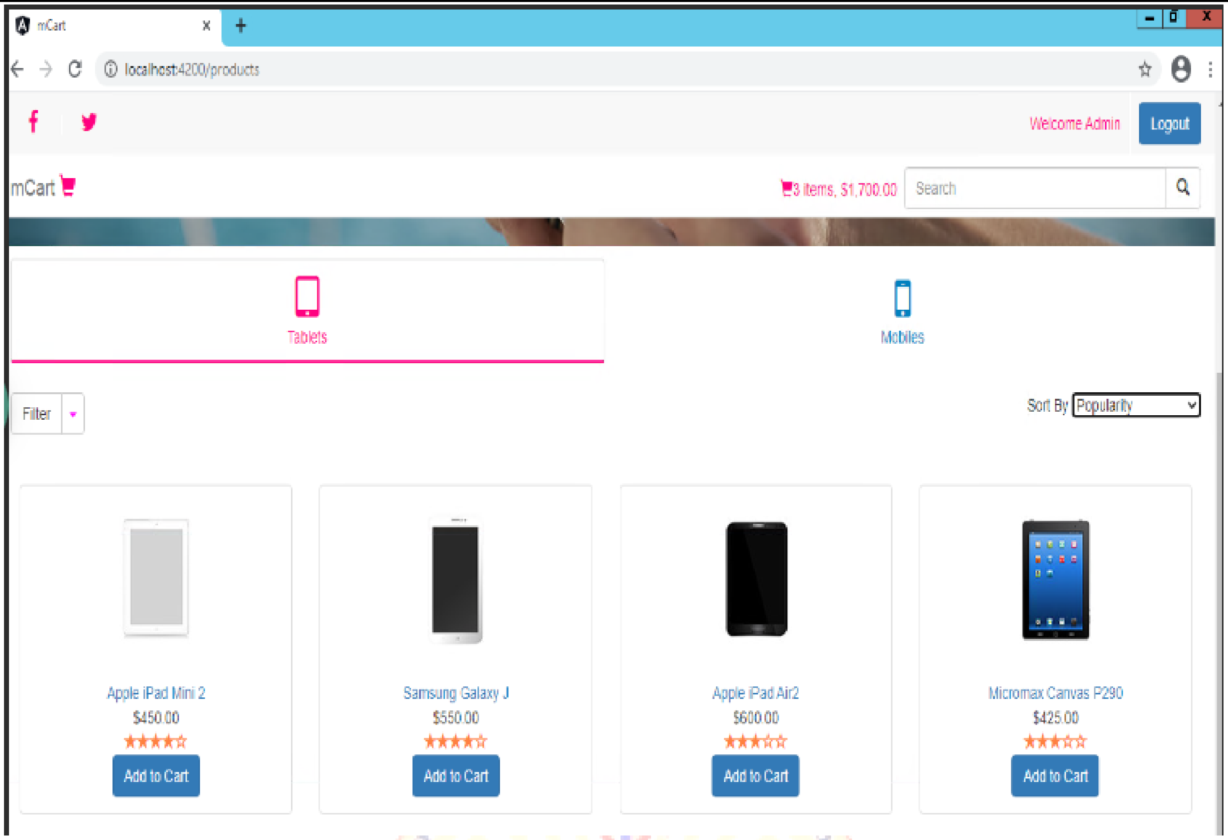
```
        [style.width.px]='imageWidth' [style.height.px]='imageHeight'

        [style.margin.px]='imageMargin'>
                                                                        </div>
                                                                        <div class="caption">
                                                                                <div>
                                                                                        <a
[routerLink]="[products[i+3].productId]">

        {{products[i+3].productName}} </a>
                                                                                </div>
                                                                                <div>{{
products[i+3].price | currency:'USD':'symbol':'1.2-2'}}
                                                                                </div>
                                                                                <div></div>
                                                                                <app-rating
class="ratingcolor" [rate]='products[i+3].rating'></app-rating>
                                                                                <div></div>
                                                                                <div>
                                                                                        <button
(click)="addCart(products[i+3].productId)"

        class="btn btn-primary">Add to Cart</button>
                                                                                </div>
                                                                        </div>
                                                                </div>
                                                        </div>
                                                </div>
                                        </div>
                                        <div *ngIf='!products || products.length==0'>
                                                <div class="row">
                                                        <div class="alert alert-warning">
                                                                There are no products available for
the selected criteria!
                                                        </div>
                                                </div>
                                        </div>
                                </div>
                                <br /><br />
                        </div>
                </div>
        </div>
</div>
```

6.Open the product-detail.component.ts file under the product-detail folder and understand the code below:

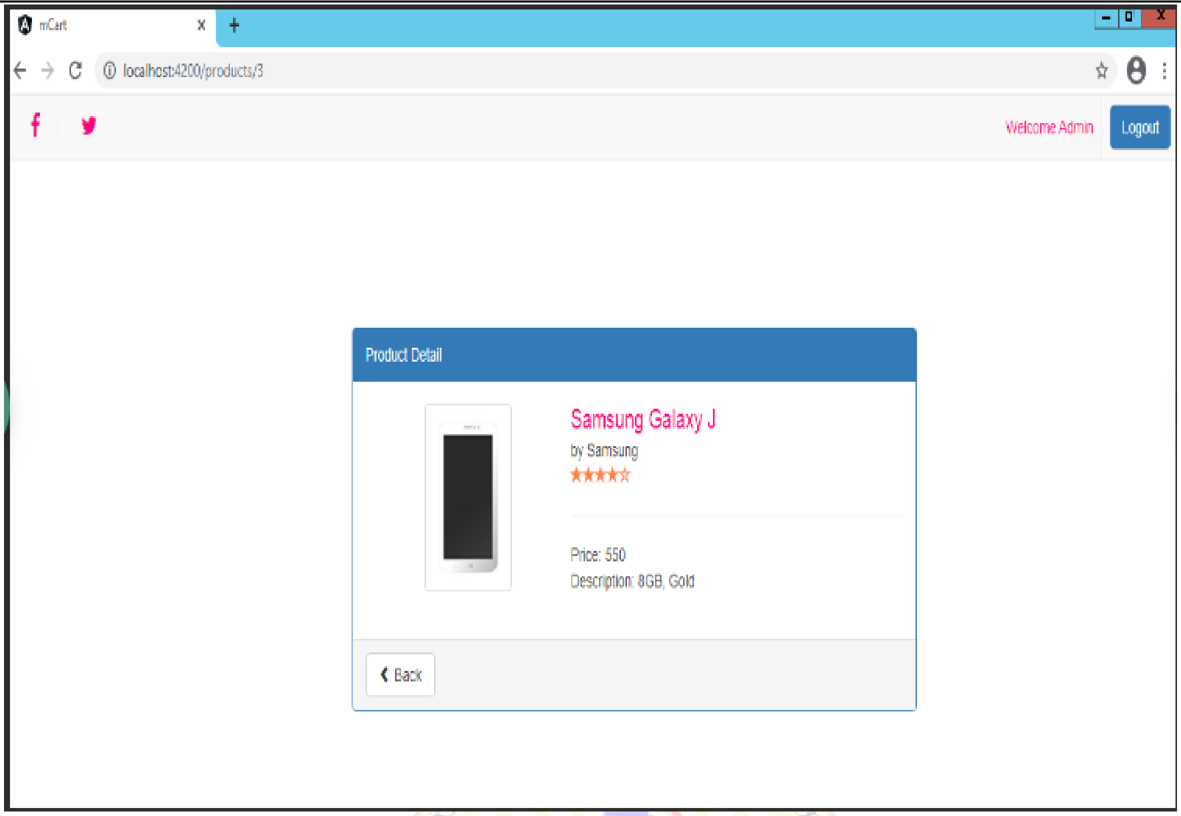
```
import { Component } from '@angular/core';
import { ActivatedRoute, Router } from '@angular/router';
import { Product } from '../product';
import { ProductService } from '../product.service';
@Component({
   templateUrl: 'product-detail.component.html',
   styleUrls: ['product-detail.component.css']
})
export class ProductDetailComponent {
   pageTitle = 'Product Detail';
   product: Product;
   imageWidth = 100;
   imageMargin = 2;
   errorMessage: string = '';
   id = 0;
   // Fetches the route parameter form the url which is selected product id
   constructor(private route: ActivatedRoute,
      private router: Router, public productService: ProductService) {
      this.id = +this.route.snapshot.paramMap.get('id')!;
      this.product = this.productService.products.filter((product: any) => product.productId ===
this.id)[0];
   }
   // Invoked when back button is clicked
   // Navigates to products page
   onBack(): void {
      this.router.navigate(['/products']);
   }
```

Output:

Below is the screenshot for ProductList Component. Observe the URL in the address bar. /products is the path to render ProductsListComponent



Below is the output of ProductDetailComponent. Observe the URL in the address bar. The URL is product/3 where 3 is the product id of the product selected



Samsung Galaxy J
by Samsung

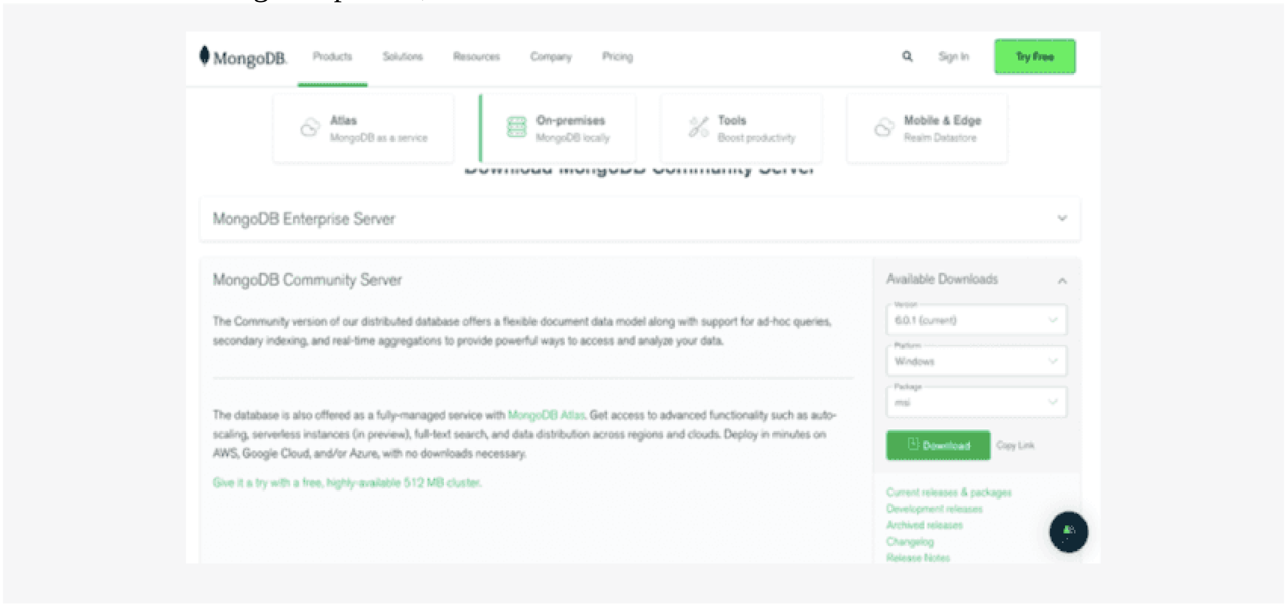
Price: 550
Description: 8GB, Gold

11.a Course Name: MongoDB Essentials - A Complete MongoDB Guide Module Name: Installing MongoDB on the local computer, Create MongoDB Atlas Cluster Install MongoDB and configure ATLAS

Program:

Step 1: Go to the Official MongoDB website
Step 2: Navigate to Products > Community Edition
Step 3: Select the appropriate installer file from the dropdown menus on the Community Edition page.

- In the version dropdown, select the latest version, 6.0.1(current)
- In the Platform dropdown, select Windows
- In the Package dropdown, select msi



MongoDB Community Server

The Community version of our distributed database offers a flexible document data model along with support for ad-hoc queries, secondary indexing, and real-time aggregations to provide powerful ways to access and analyze your data.

The database is also offered as a fully-managed service with MongoDB Atlas. Get access to advanced functionality such as auto-scaling, serverless instances (in preview), full-text search, and data distribution across regions and clouds. Deploy in minutes on AWS, Google Cloud, and/or Azure, with no downloads necessary.

Give it a try with a free, highly-available 512 MB cluster.

Version: 6.0.1 (current)
Platform: Windows
Package: msi

Step 4: Click the green "Download" button. Wait for 2-5 minutes for the file to download. (Depending on your internet speed)

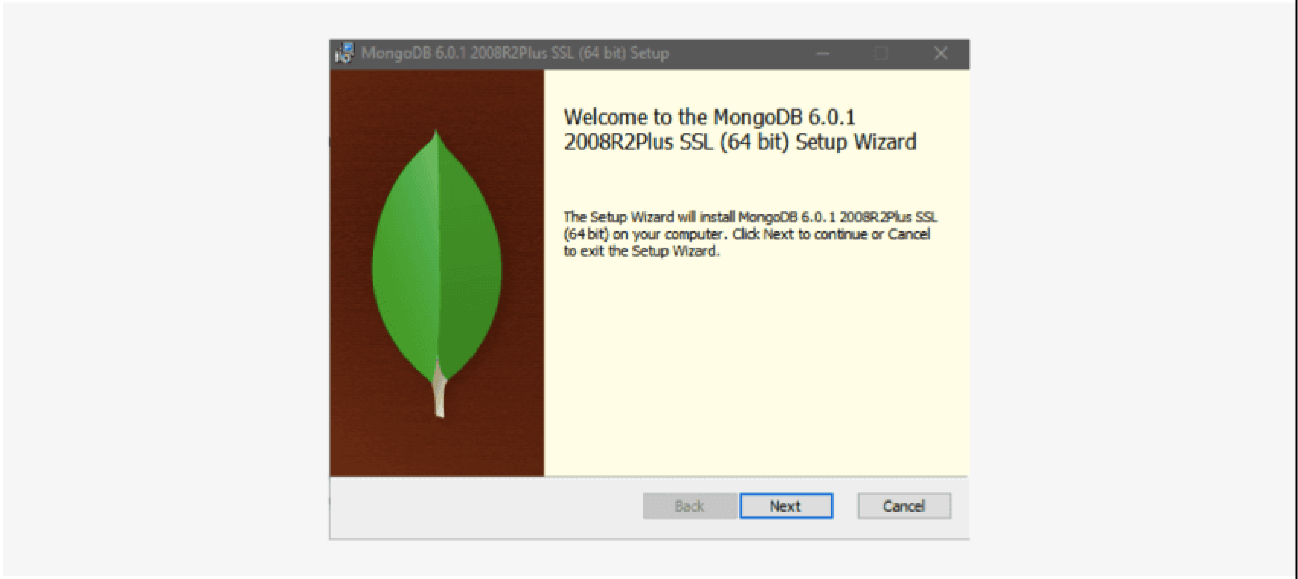
Installation of MongoDB on Windows 10 :

After the installer file has been downloaded, it's time to run the installer file.

Procedure

Step: 1: Go to the downloaded directory in your system (by default, it should be in the `Downloads` directory).

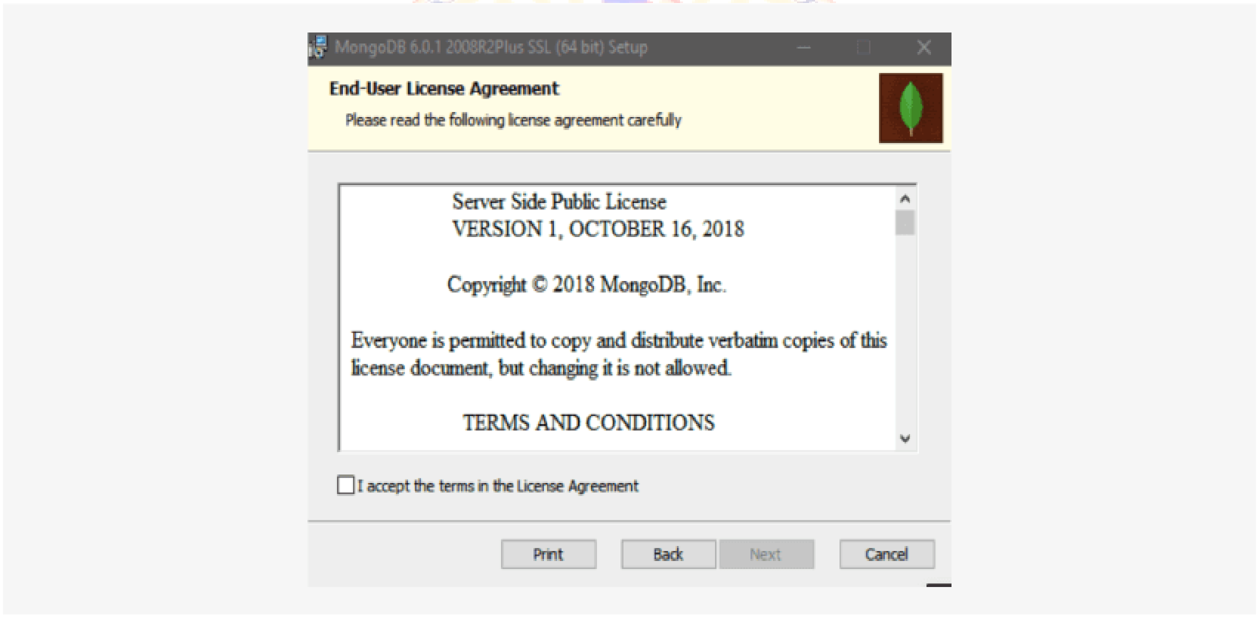
Step 2: Double-click on the .msi file. It will open the MongoDB setup windows.


MongoDB 6.0.1 2008R2Plus SSL (64 bit) Setup

Welcome to the MongoDB 6.0.1 2008R2Plus SSL (64 bit) Setup Wizard

The Setup Wizard will install MongoDB 6.0.1 2008R2Plus SSL (64 bit) on your computer. Click Next to continue or Cancel to exit the Setup Wizard.

Step 3: It will open the MongoDB Community Edition installation wizard. This setup wizard guides you through the installation of MongoDB in your system. To continue the process, click "Next."


MongoDB 6.0.1 2008R2Plus SSL (64 bit) Setup

End-User License Agreement

Please read the following license agreement carefully

Server Side Public License
VERSION 1, OCTOBER 16, 2018

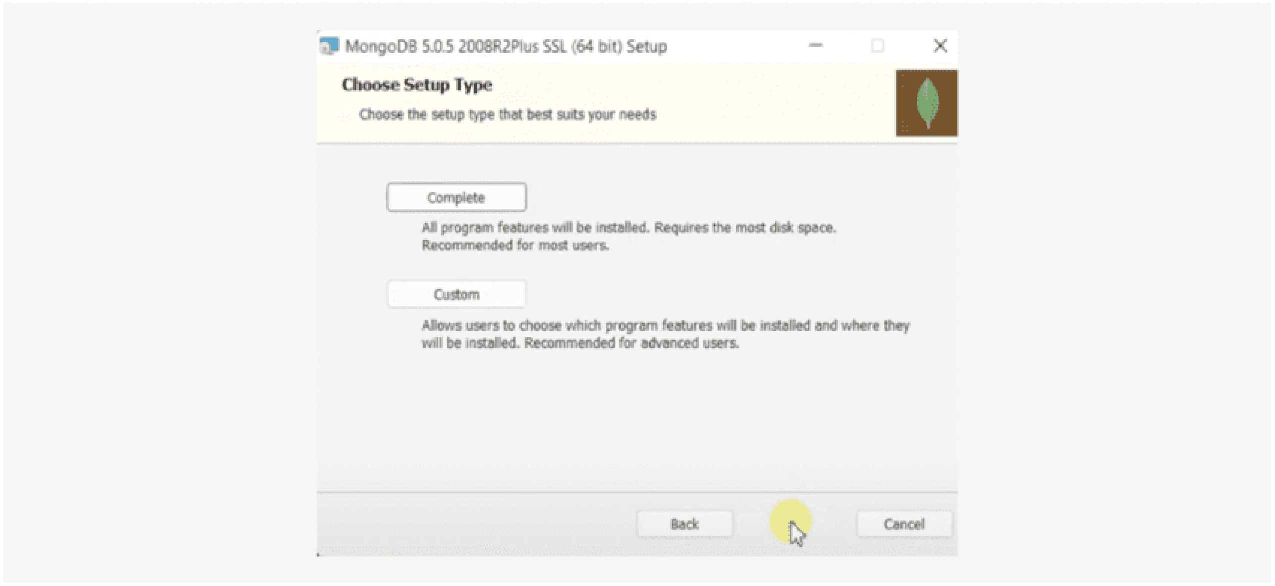
Copyright © 2018 MongoDB, Inc.

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

TERMS AND CONDITIONS

I accept the terms in the License Agreement

Step 4: Read the End-User License Agreement, accept the terms and conditions, and then click the "Next" button to continue.


MongoDB 5.0.5 2008R2Plus SSL (64 bit) Setup

Choose Setup Type

Choose the setup type that best suits your needs

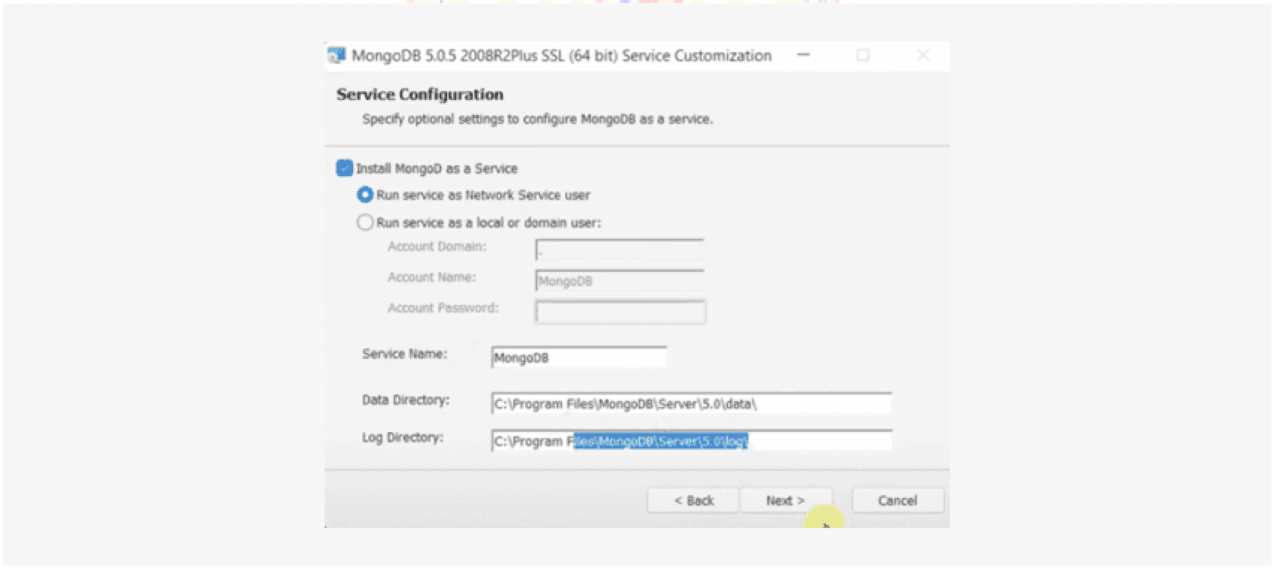
Complete

All program features will be installed. Requires the most disk space. Recommended for most users.

Custom

Allows users to choose which program features will be installed and where they will be installed. Recommended for advanced users.

Step 5: Next, you can choose either the Complete setup or Custom setup type to proceed. But for a beginner, we'd recommend using the Complete setup option. It installs MongoDB in the default location. Select the Complete setup, and click "Next."


MongoDB 5.0.5 2008R2Plus SSL (64 bit) Service Customization

Service Configuration

Specify optional settings to configure MongoDB as a service.

[✓] Install MongoD as a Service

● Run service as Network Service user

○ Run service as a local or domain user:

Account Domain: .

Account Name: MongoDB

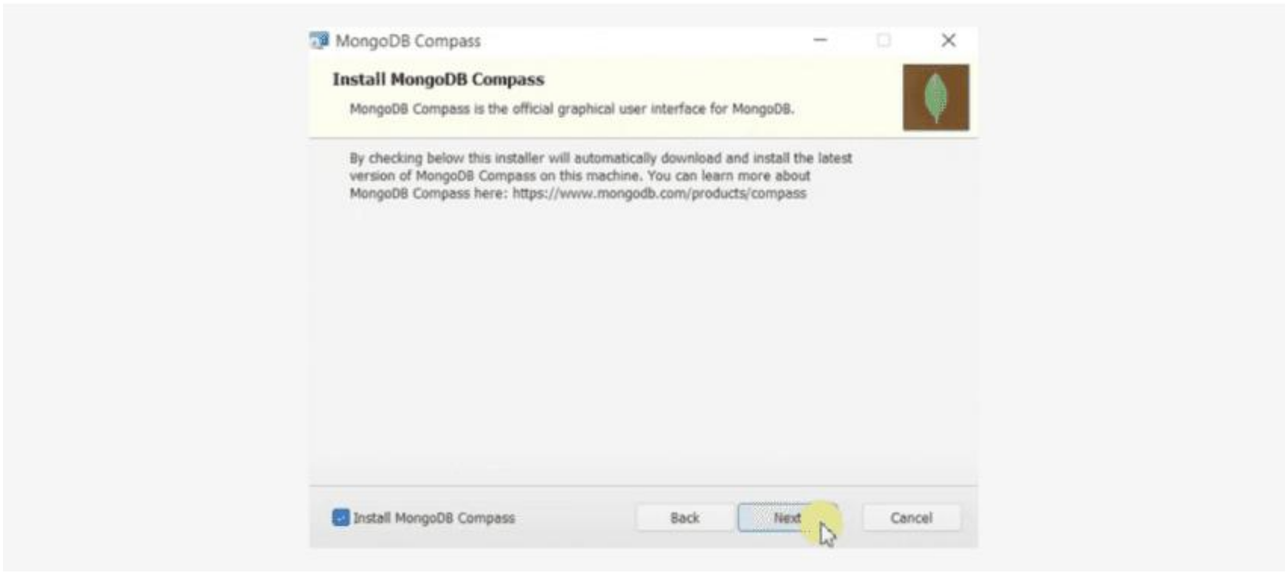
Account Password:

Service Name: MongoDB

Data Directory: C:\Program Files\MongoDB\Server\5.0\data\

Log Directory: C:\Program Files\MongoDB\Server\5.0\log\

Step 6: Select the "Install MongoD as a Service" option on the next page. Keep all other parameters as default. Click on the "Next" button.


MongoDB Compass

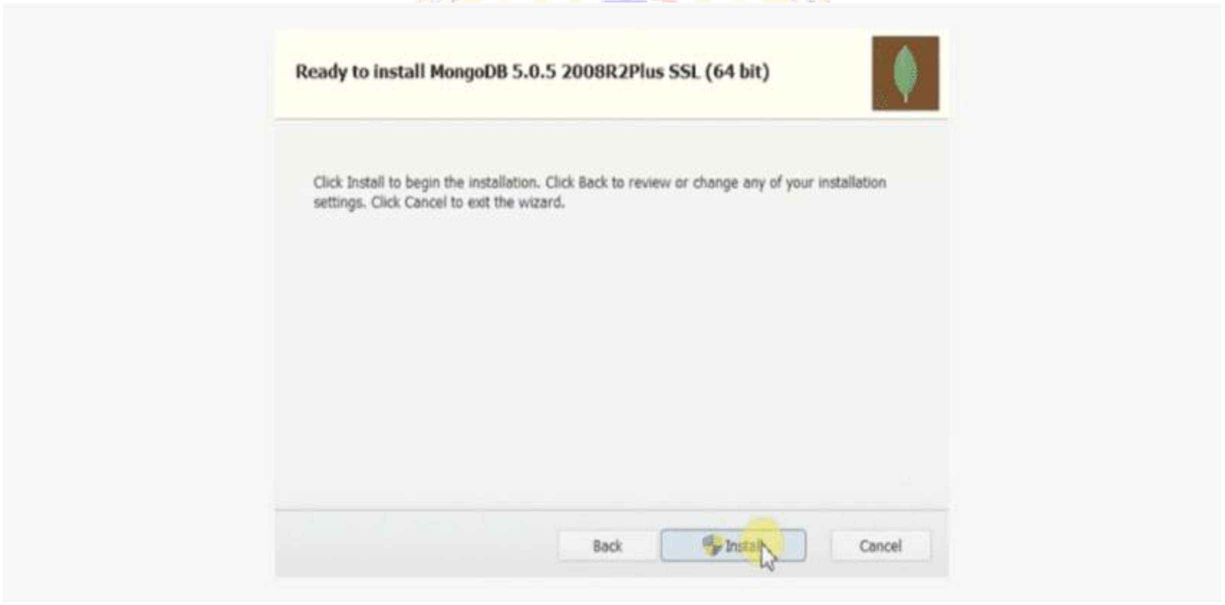
Install MongoDB Compass

MongoDB Compass is the official graphical user interface for MongoDB.

By checking below this installer will automatically download and install the latest version of MongoDB Compass on this machine. You can learn more about MongoDB Compass here: https://www.mongodb.com/products/compass

✓ Install MongoDB Compass

Step 7: In the next step, you will get an option to install MongoDB compass. Uncheck it if you don't want MongoDB compass to be installed on your device, and then click the "Next" button.


Ready to install MongoDB 5.0.5 2008R2Plus SSL (64 bit)

Click Install to begin the installation. Click Back to review or change any of your installation settings. Click Cancel to exit the wizard.

Step 8: In the "Ready to install MongoDB" page, click the "Install" button, give administrator access, and wait for the installation to finish. Once installation is complete, you can click on the "Finish" button to finalize your installation.

Configuring Atlas:

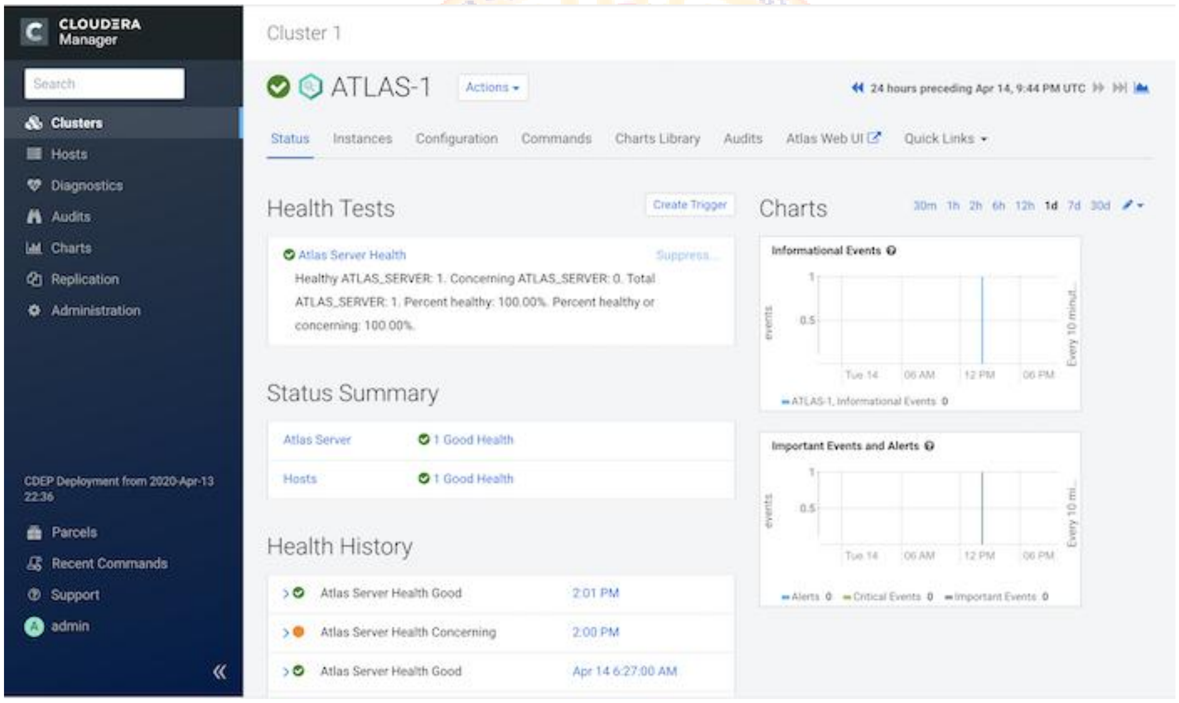
Cloudera Manager manages Atlas as a service, including monitoring its health and collecting its logs. Use Cloudera Manager to configure additional health checks, set up extractors, and set other Atlas properties.

Typically, you would need to make changes to Atlas configuration for the following reasons:

- Expanding resources for the Atlas server. For example in CDP Data Center, the initial memory allocation for Atlas is 2 MB.
- Changing Atlas security settings to add security levels or update parameters for Knox, LDAP, TLS, and Kerberos.
- Updating Atlas for other environment changes such as log directory locations.

To start or stop metadata collection from a service, make the configuration change in the properties for that service.

Atlas collects metadata for services in a given cluster managed by Cloudera Manager. If Cloudera Manager manages more than one cluster, you may have multiple instances of Atlas, one in each cluster. When you check the health of Atlas or configure its properties, make sure you are working in the correct cluster.



Health Tests

Atlas Server Health — Suppress...

Healthy ATLAS_SERVER: 1. Concerning ATLAS_SERVER: 0. Total ATLAS_SERVER: 1. Percent healthy: 100.00%. Percent healthy or concerning: 100.00%.

Status Summary

Atlas Server	1 Good Health
Hosts	1 Good Health

Health History

Atlas Server Health Good	2:01 PM
Atlas Server Health Concerning	2:00 PM
Atlas Server Health Good	Apr 14 6:27:00 AM

11.b Course Name: MongoDB Essentials - A Complete MongoDB Guide
Module Name: Introduction to the CRUD Operations Write MongoDB queries to perform CRUD operations on document using insert(), find(), update(), remove().
Program:

MongoDB provides two different create operations that you can use to insert documents into a collection:
db.collection.insertOne()
db.collection.insertMany()

insertOne()

As the namesake, insertOne() allows you to insert one document into the collection. For this example, we're going to work with a collection called RecordsDB. We can insert a single entry into our collection by calling the insertOne() method on RecordsDB. We then provide the information we want to insert in the form of key-value pairs, establishing the schema.

```
db.RecordsDB.insertOne({

    name: "Marsh",

    age: "6 years",

    species: "Dog",

    ownerAddress: "380 W. Fir Ave",

    chipped: true

})
```

If the create operation is successful, a new document is created. The function will return an object where "acknowledged" is "true" and "insertID" is the newly created "ObjectId."

```
> db.RecordsDB.insertOne({

... name: "Marsh",

... age: "6 years",

... species: "Dog",

... ownerAddress: "380 W. Fir Ave",

... chipped: true
... })

{

        "acknowledged" : true,

        "insertedId" : ObjectId("5fd989674e6b9ceb8665c57d")

}
```

insertMany()

It’s possible to insert multiple items at one time by calling the *insertMany()* method on the desired collection. In this case, we pass multiple items into our chosen collection (*RecordsDB*) and separate them by commas. Within the parentheses, we use brackets to indicate that we are passing in a list of multiple entries. This is commonly referred to as a nested method.

```
db.RecordsDB.insertMany([{

    name: "Marsh",

    age: "6 years",

    species: "Dog",

    ownerAddress: "380 W. Fir Ave",

    chipped: true},

      {name: "Kitana",

      age: "4 years",

      species: "Cat",

      ownerAddress: "521 E. Cortland",

      chipped: true}])


db.RecordsDB.insertMany([{ name: "Marsh", age: "6 years", species: "Dog",

ownerAddress: "380 W. Fir Ave", chipped: true}, {name: "Kitana", age: "4 years",
species: "Cat", ownerAddress: "521 E. Cortland", chipped: true}])

{

        "acknowledged" : true,

        "insertedIds" : [

                ObjectId("5fd98ea9ce6e8850d88270b4"),

                ObjectId("5fd98ea9ce6e8850d88270b5")

        ]

}
```

find()

In order to get all the documents from a collection, we can simply use the *find()* method on our chosen collection. Executing just the *find()* method with no arguments will return all records currently in the collection.

db.RecordsDB.find()

Output:

```
{ "_id" : ObjectId("5fd98ea9ce6e8850d88270b5"), "name" : "Kitana", "age" : "4 years", "species" : "Cat", "ownerAddress" : "521 E. Cortland", "chipped" : true }

{ "_id" : ObjectId("5fd993a2ce6e8850d88270b7"), "name" : "Marsh", "age" : "6 years", "species" : "Dog", "ownerAddress" : "380 W. Fir Ave", "chipped" : true }

{ "_id" : ObjectId("5fd993f3ce6e8850d88270b8"), "name" : "Loo", "age" : "3 years", "species" : "Dog", "ownerAddress" : "380 W. Fir Ave", "chipped" : true }

{ "_id" : ObjectId("5fd994efce6e8850d88270ba"), "name" : "Kevin", "age" : "8 years", "species" : "Dog", "ownerAddress" : "900 W. Wood Way", "chipped" : true }
```

findOne()

In order to get one document that satisfies the search criteria, we can simply use the *findOne()* method on our chosen collection. If multiple documents satisfy the query, this method returns the first document according to the natural order which reflects the order of documents on the disk. If no documents satisfy the search criteria, the function returns null. The function takes the following form of syntax.

db.{collection}.findOne({query}, {projection})

Update Operations:

For MongoDB CRUD, there are three different methods of updating documents:

- db.collection.updateOne()
- db.collection.updateMany()
- db.collection.replaceOne()

updateOne():

db.RecordsDB.updateOne({name: "Marsh"}, {$set:{ownerAddress: "451 W. Coffee St. A204"}})

Output:

{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }

updateMany():

db.RecordsDB.updateMany({species:"Dog"}, {$set: {age: "5"}})

Output:

{ "acknowledged" : true, "matchedCount" : 3, "modifiedCount" : 3 }

Delete Operations:

MongoDB has two different methods of deleting records from a collection:

- db.collection.deleteOne()
- db.collection.deleteMany()

deleteOne():

deleteOne() is used to remove a document from a specified collection on the MongoDB server. A filter criteria is used to specify the item to delete. It deletes the first record that matches the provided filter.

db.RecordsDB.deleteOne({name:"Maki"})

Output:

```
{ "acknowledged" : true, "deletedCount" : 1 }
```

deleteMany():

db.RecordsDB.deleteMany({species:"Dog"})

Output:

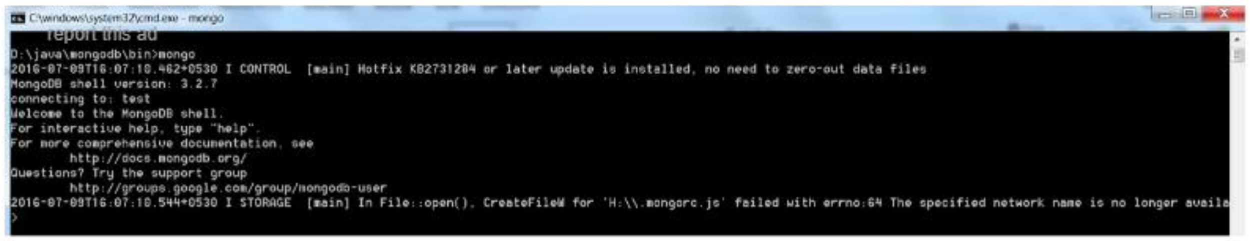
```
{ "acknowledged" : true, "deletedCount" : 2 }
```

12.a Course Name: MongoDB Essentials - A Complete MongoDB GuideModule Name: Create and Delete Databases and Collections
Write MongoDB queries to Create and drop databases and collections.
program:

Creating a Database

Open the command prompt and navigate to the **/bin** folder of the MongoDB using the cd command and execute the command mongod there. This will initiate the MongoDB server. We have to keep this command prompt window alive, as this is running MongoDB. To stop the MongoDB server, simply enter exit and press Enter.

Now, Open another command prompt and navigate to the **/bin** folder of the MongoDB again and execute the command mongo. This will open up the client to run the MongoDB commands.

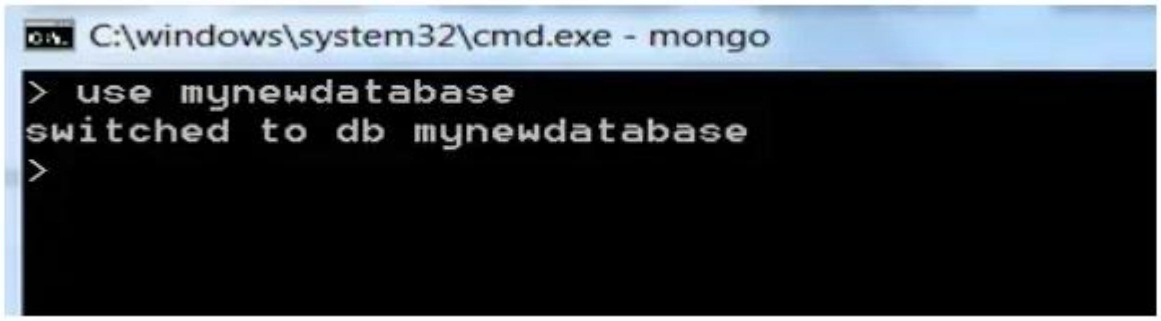


```
C:\windows\system32\cmd.exe - mongo
D:\java\mongodb\bin>mongo
2016-07-09T16:07:10.462+0530 I CONTROL  [main] Hotfix KB2731284 or later update is installed, no need to zero-out data files
MongoDB shell version: 3.2.7
connecting to: test
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        http://docs.mongodb.org/
Questions? Try the support group
        http://groups.google.com/group/mongodb-user
2016-07-09T16:07:10.544+0530 I STORAGE  [main] In File::open(), CreateFileW for 'H:\\.mongorc.js' failed with errno:64 The specified network name is no longer availa
>
```

In the command prompt window in which we ran the mongo command, after successfully connecting to the mongodb, just type the command the following :

```
use database_name
```

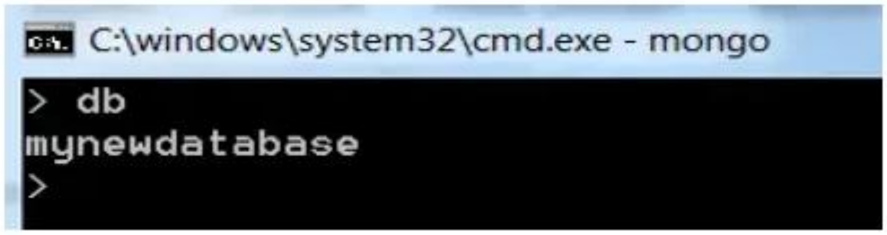
This will create a new database with the name **database_name** if there is no database already present with the same name. If a database already exists with the mentioned name, then it just connects to that database.



```
C:\windows\system32\cmd.exe - mongo
> use mynewdatabase
switched to db mynewdatabase
>
```

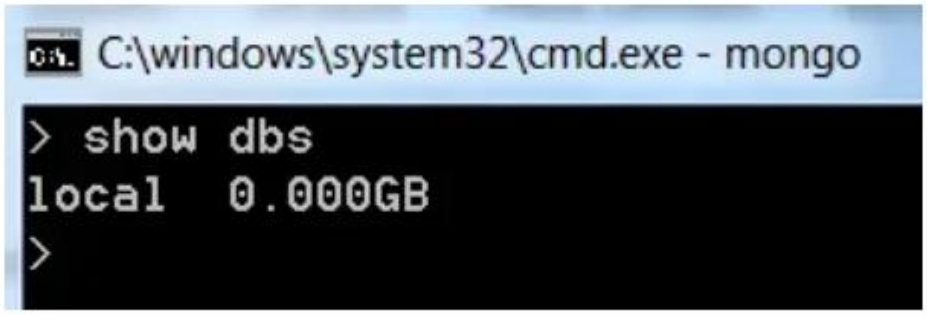
In the above picture, it creates a new database called **mynewdatabase** and will also connect to the same.

To check the current connected database, type in db in the command prompt window, and you will get the name of the current database as result.



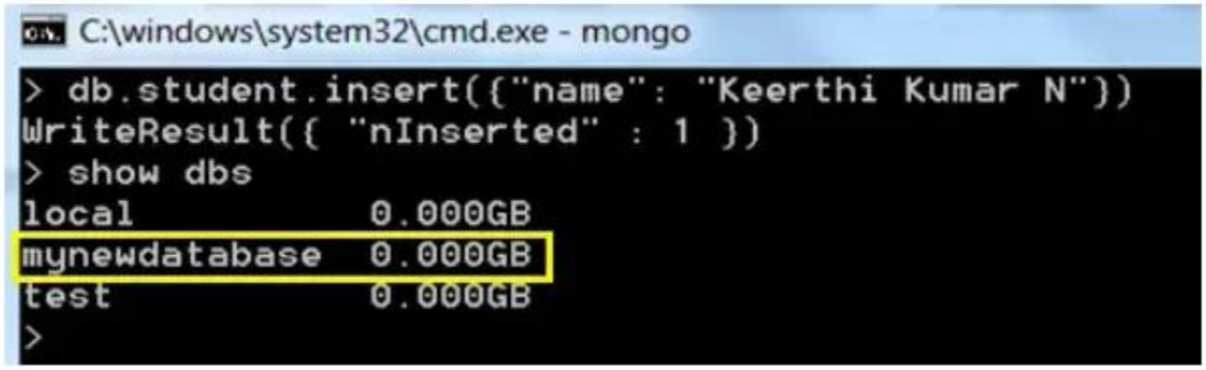
```
C:\windows\system32\cmd.exe - mongo
> db
mynewdatabase
>
```

To see the list of all the databases in MongoDB, use command show dbs



```
C:\windows\system32\cmd.exe - mongo
> show dbs
local   0.000GB
>
```

Please note that the newly created dstabase **mynewdatabase** has not been listed after running the above command. This is because, no records have been inserted into that database yet. Just insert one record and then run the command again as shown below:



```
C:\windows\system32\cmd.exe - mongo
> db.student.insert({"name": "Keerthi Kumar N"})
WriteResult({ "nInserted" : 1 })
> show dbs
local          0.000GB
mynewdatabase  0.000GB
test           0.000GB
>
```

To Insert data, run the following command. Dont worry about it, we will learn this in detail in next lessons.

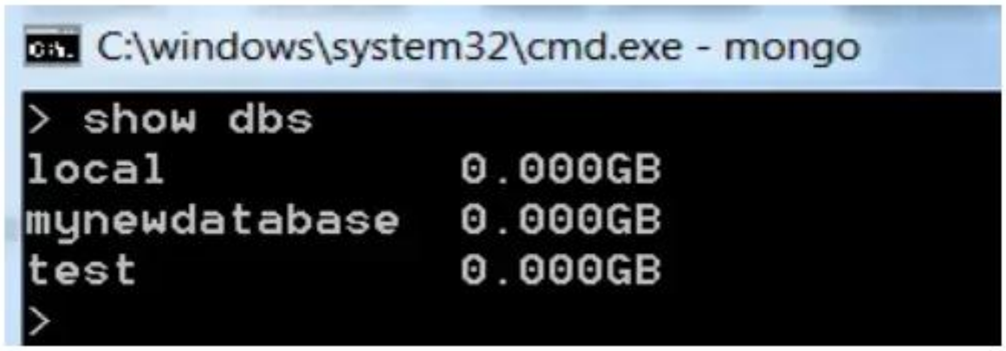
```
db.student.insert({name : "Viraj" })
```

Copy

NOTE: In MongoDB, test will be the default database. If no database is created, then all the data will be stored in the test database.

Drop a Database

First check the list of databases available as shown below, using the show dbs command.

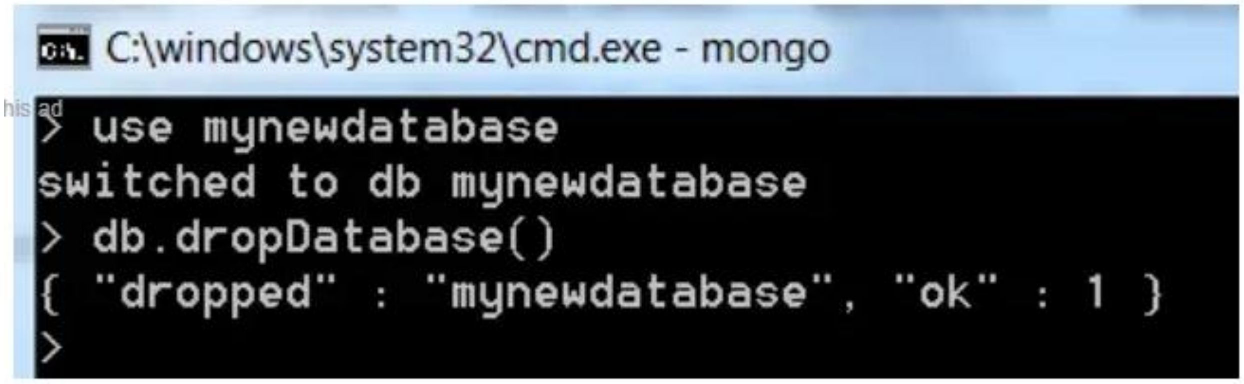


C:\windows\system32\cmd.exe - mongo

```
> show dbs
local            0.000GB
mynewdatabase    0.000GB
test             0.000GB
>
```

If the newly created database **mynewdatabase** has to be dropped(deleted). Run the below command to delete the database. Before deleting the database, connect to the required database which is to be deleted.

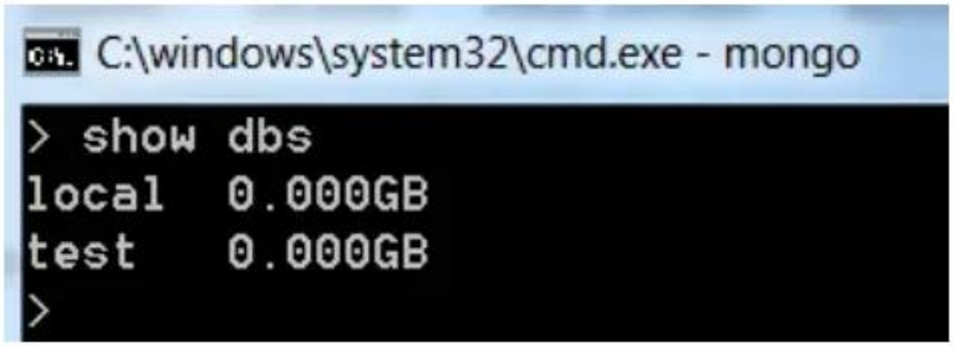
```
db.dropDatabase()
```



C:\windows\system32\cmd.exe - mongo

```
> use mynewdatabase
switched to db mynewdatabase
> db.dropDatabase()
{ "dropped" : "mynewdatabase", "ok" : 1 }
>
```

Now again check the list of databases, to verify whether the database is deleted or not.


C:\windows\system32\cmd.exe - mongo

```
> show dbs
local   0.000GB
test    0.000GB
>
```

Note that the database **mynewdatabase** has been deleted and hence, it will not be listed in the list of the databases.

Creating a Collection

In MongoDB a collection is automatically created when it is referenced in any command. For example, if you run an insert command :

```
db.student.insert({

        name:"Viraj"

})
```

Field	Type	Description
capped	boolean	(Optional) To create a capped collection, where in we specify the the maximum size or document counts to prevent it from growing beyond a set maximum value.

size	number	(Optional) To specify the maximum size in bytes for a capped collection. If a collection is capped and reaches its maximum size limit, MongoDB then removes older documents from it to make space for new.
max	number	(Optional) This can be used to specify the maximum number of documents allowed in a capped collection.
validator	document	(Optional) Validates any document inserted or updated against provided validation rules.
validationLevel	string	(Optional) This is used to define how strictly validation rules must be applied to existing documents during an update. Available values are : off No validation for inserts or updates. strict This is the default value for this option. This instructs MongoDB to apply validation rules to all inserts and updates. moderate This is used when we want to apply validation rules to inserts and updates on only the existing valid documents and not on the existing invalid documents.
validationAction	string	(Optional) This can be used to set the action to be taken upon validation i.e. if any document fails the set validation then whether to show error or just warn about the violations. Default value is error.

Above command will create a collection named **student** if it doesn't exist already in the database. But we can explicitly create a collection using the createCollection() command. The

syntax of createCollection method is:

```
db.createCollection(name, options)
```

In the above command, **name** will be the name of the collection and **options** is a document which can be used to specify configurations for the collection.

options parameter is optional, and you can simply create a collection by just providing a name for the collection. But if you want to configure your collection, you can use various options available to do so. Following are the available configuration options for creating a collection in MongoDB:

MongoDB: Creating a Capped Collection

We can create a capped collection using the following command.

```
db.createCollection("student",{ capped :true, size :5242880, max :5000})
```

This will create a collection named student, with maximum size of 5 megabytes and maximum of 5000 documents.

MongoDB: Drop a Collection

Any collection in a database in MongoDB can be dropped easily using the following command:

```
db.collection_name.drop()
```

drop() method will return true is the collection is dropped successfully, else it will return false.

12.b Course Name: MongoDB Essentials - A Complete MongoDB Guide

Module Name: Introduction to MongoDB Queries Write MongoDB queries to work with records using find(), limit(), sort(), createIndex(), aggregate()

Program:

1.Using **find()** to retrieve records:

- To retrieve all records in a collection, use an empty object as the query parameter:

db.collection.find({})

- To retrieve all records where a specific field matches a certain value, you can specify a query object as the parameter:

db.collection.find({field: value})

2.Using **limit()** to limit the number of records returned:

- To limit the number of records returned by a query, use the **limit()** method. For example, to retrieve the first 10 records in a collection, you can use:

db.collection.find().limit(10)

3.Using **sort()** to sort records:

- To sort records in ascending or descending order, use the **sort()** method. For example, to sort records by a field in ascending order:

db.collection.find().sort({field: 1})

- To sort records by a field in descending order:

db.collection.find().sort({field: -1})

4.Using **createIndex()** to create indexes:

- To create an index on a collection, use the **createIndex()** method. For example, to create an ascending index on a field:

db.collection.createIndex({field: 1})

5.Using **aggregate()** to perform complex operations:

- The **aggregate()** method allows you to perform complex operations on the data in a collection. For example, to group records by a field and calculate the sum of another field:

db.collection.aggregate([{ $group: { _id: "$field", total: { $sum: "$otherField" } } }])

- This query groups records by the value of the **field** field, and calculates the sum of the **otherField** field for each group. The results are returned as an array of objects with **_id** and **total** fields.

8.a Course Name: Angular JS

Module Name: Custom Validators in Template Driven forms Create a custom validator for the email field in the course registration form.

Program: