

Web and Mobile Programming

ICP7

Wiki report

ICP Group:6

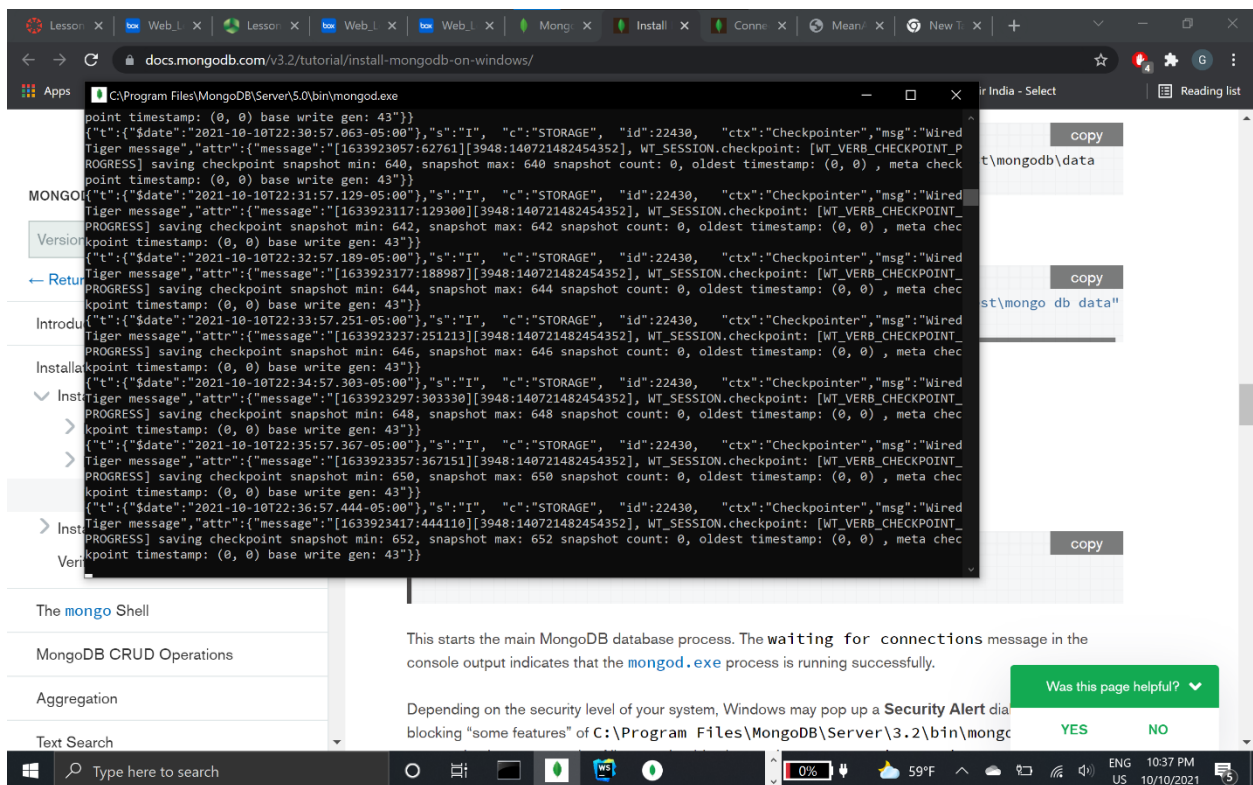
Karthik Yanagandula

Gayathri Garikapati

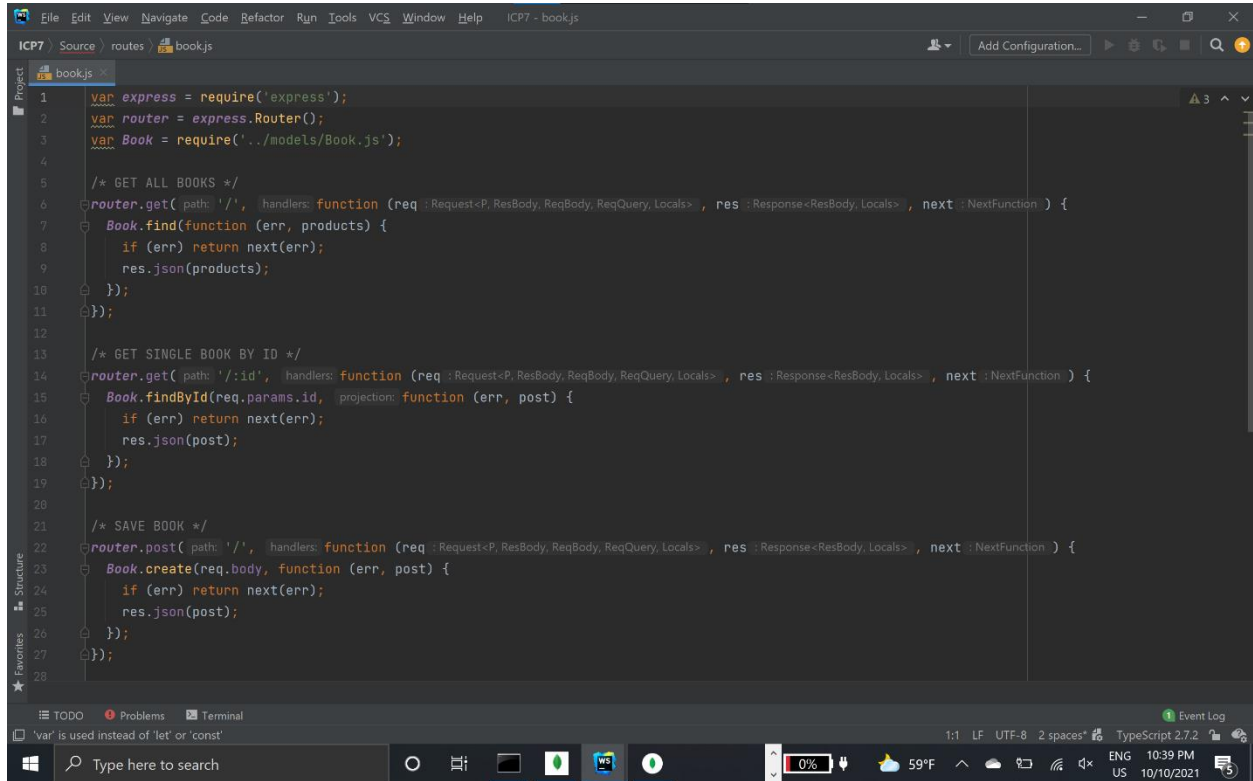
As part of this ICP task we have learned the CRUD(Create, Read, Update and Delete) Operations and the basics of node.js, Express, Mongodb. Established the connection so that front-end and back-end can communicate with each other. Initial requirements for this ICP task is to install mongodb and we have used the documentation provided in lesson plan as reference for installation.

For this ICP we have used local connection instead of using with cloud and have started and connected to the mongodb using below commands:

"C:\Program Files\MongoDB\Server\3.2\bin\mongod.exe": command to start mongodb
"C:\Program Files\MongoDB\Server\3.2\bin\mongo.exe: command to connect to mongodb

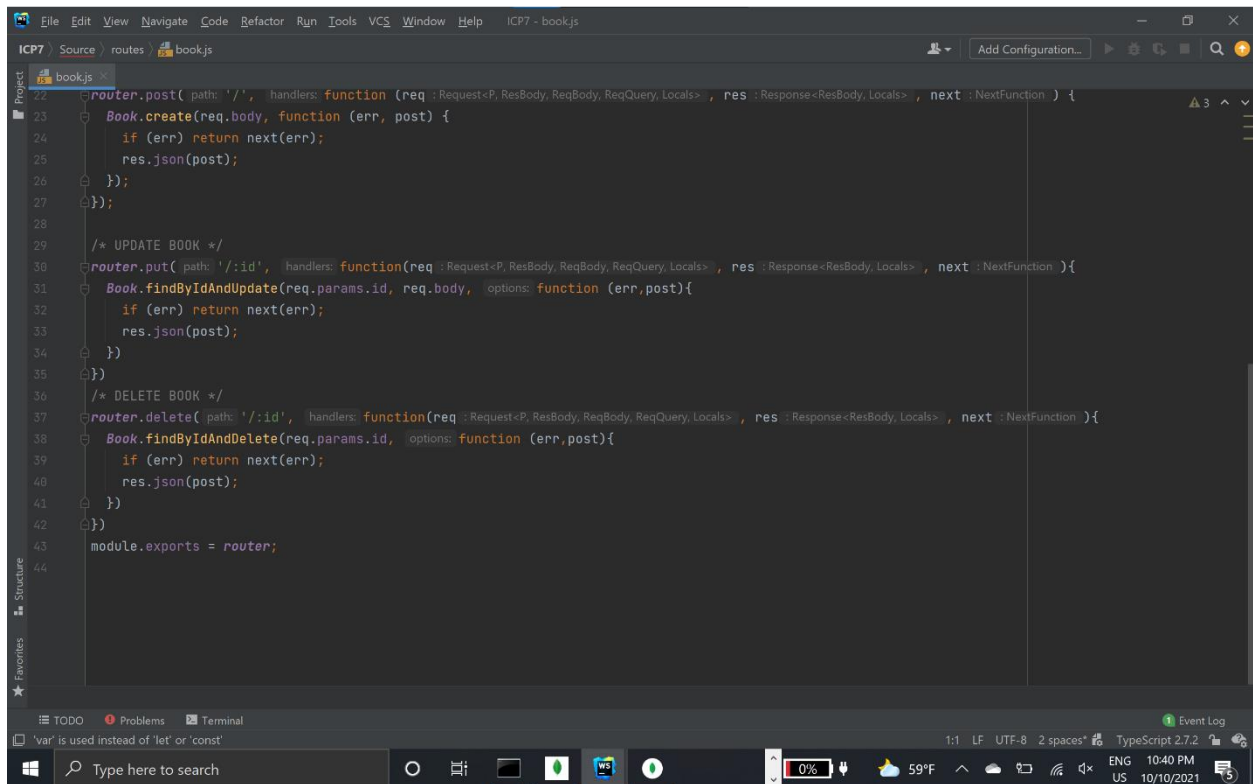


With book.js containing the code to save, get, update and delete book, the routes for the application are done.



```
1 var express = require('express');
2 var router = express.Router();
3 var Book = require('../models/Book.js');
4
5 /* GET ALL BOOKS */
6 router.get( path: '/', handlers: function (req : Request<P, ResBody, ReqBody, ReqQuery, Locals> , res : Response<ResBody, Locals> , next : NextFunction ) {
7   Book.find(function (err, products) {
8     if (err) return next(err);
9     res.json(products);
10  });
11 });
12
13 /* GET SINGLE BOOK BY ID */
14 router.get( path:('/:id', handlers: function (req : Request<P, ResBody, ReqBody, ReqQuery, Locals> , res : Response<ResBody, Locals> , next : NextFunction ) {
15   Book.findById(req.params.id, projection: function (err, post) {
16     if (err) return next(err);
17     res.json(post);
18   });
19 });
20
21 /* SAVE BOOK */
22 router.post( path: '/', handlers: function (req : Request<P, ResBody, ReqBody, ReqQuery, Locals> , res : Response<ResBody, Locals> , next : NextFunction ) {
23   Book.create(req.body, function (err, post) {
24     if (err) return next(err);
25     res.json(post);
26   });
27 });
28
```

The screenshot shows a code editor with the file 'book.js' open. The code defines an Express.js application with three routes: a GET route for all books, a GET route for a single book by ID, and a POST route to save a new book. The routes use the 'Book' model from '../models/Book.js'. The IDE interface includes a sidebar with 'Project', 'Structure', and 'Favorites' views, and a bottom panel with 'TODO', 'Problems', and 'Terminal' tabs. A status bar at the bottom shows the file encoding as UTF-8 and the TypeScript version as 2.7.2.



```
22 router.post( path: '/', handlers: function (req : Request<P, ResBody, ReqBody, ReqQuery, Locals> , res : Response<ResBody, Locals> , next : NextFunction ) {
23   Book.create(req.body, function (err, post) {
24     if (err) return next(err);
25     res.json(post);
26   });
27 });
28
29 /* UPDATE BOOK */
30 router.put( path: '/:id', handlers: function(req : Request<P, ResBody, ReqBody, ReqQuery, Locals> , res : Response<ResBody, Locals> , next : NextFunction ){
31   Book.findByIdAndUpdate(req.params.id, req.body, {options: function (err,post){
32     if (err) return next(err);
33     res.json(post);
34   }}
35 );
36
37 /* DELETE BOOK */
38 router.delete( path: '/:id', handlers: function(req : Request<P, ResBody, ReqBody, ReqQuery, Locals> , res : Response<ResBody, Locals> , next : NextFunction ){
39   Book.findByIdAndDelete(req.params.id, {options: function (err,post){
40     if (err) return next(err);
41     res.json(post);
42   }}
43 );
44 module.exports = router;
```

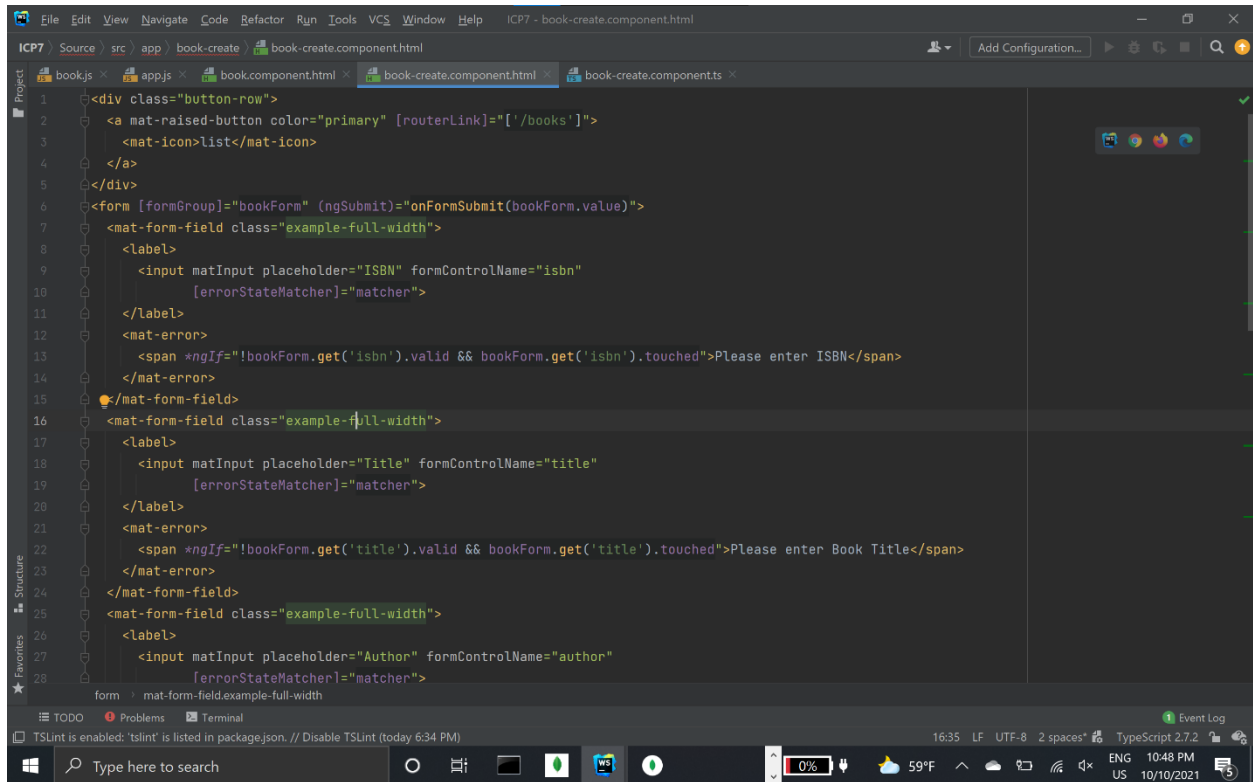
Below is the code for app.js using which we can check the MongoDB Connection and ensure that the database is connected with our application.

```
ICP7 - app.js
1 var createError = require('http-errors');
2 var express = require('express');
3 var path = require('path');
4 var favicon = require('serve-favicon');
5 var logger = require('morgan');
6
7 var mongoose = require('mongoose');
8 mongoose.connect('mongodb://localhost/mean-angular6')
9   .then(() => console.log('connection successful'))
10  .catch((err) => console.error(err));
11
12 var apiRouter = require('./routes/book');
13
14 var app = express();
15
16 app.use(logger('dev'));
17 app.use(express.json());
18 app.use(express.urlencoded({ extended: false }));
19 app.use(express.static(path.join(__dirname, 'dist/mean-angular6')));
20 app.use('/books', express.static(path.join(__dirname, 'dist/mean-angular6')));
21 app.use('/book-details/:id', express.static(path.join(__dirname, 'dist/mean-angular6')));
22 app.use('/book-create', express.static(path.join(__dirname, 'dist/mean-angular6')));
23 app.use('/book-edit/:id', express.static(path.join(__dirname, 'dist/mean-angular6')));
24 app.use('/api', apiRouter);
25
26 // catch 404 and forward to error handler
27 app.use(function (req: Request<RouteParameters<string>, any, any, ParsedQs, Record<string, any>>, res: Response<any, Record<string, any>>, next: NextFunction) {
28   next(createError(404));
29 });
```

Below is the HTML code for initial book table

```
ICP7 - book.component.html
7 <!-- Title Column -->
8 <ng-container matColumnDef="isbn">
9   <th mat-header-cell *matHeaderCellDef> ISBN</th>
10  <td mat-cell *matCellDef="let element" class="isbn-col"> {{element.isbn}} </td>
11 </ng-container>
12
13 <!-- Title Column -->
14 <ng-container matColumnDef="title">
15   <th mat-header-cell *matHeaderCellDef> Title</th>
16   <td mat-cell *matCellDef="let element"> {{element.title}} </td>
17 </ng-container>
18
19 <!-- Author Column -->
20 <ng-container matColumnDef="author">
21   <th mat-header-cell *matHeaderCellDef> Author</th>
22   <td mat-cell *matCellDef="let element"> {{element.author}} </td>
23 </ng-container>
24
25 <tr mat-header-row *matHeaderRowDef="displayedColumns"></tr>
26 <tr mat-row *matRowDef="let row; columns: displayedColumns;" [routerLink]="['/book-details/', row._id]"></tr>
27 </table>
28 </div>
29
30 <div class="button-row">
31   <a mat-raised-button color="primary" [routerLink]="['/book-create']">
32     <mat-icon>add</mat-icon>
33   </a>
34 </div>
```

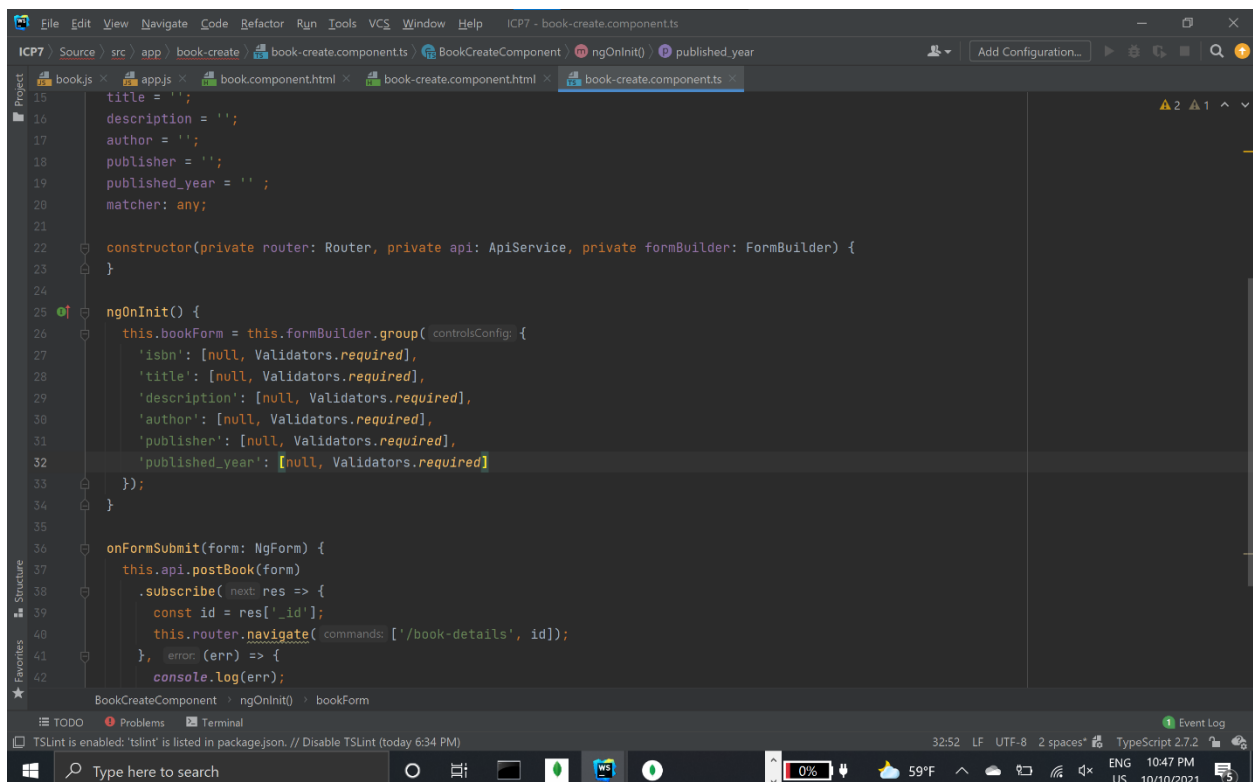
Below is the HTML code for book creation



The screenshot shows a Visual Studio Code editor window with the file `book-create.component.html` open. The code is an Angular HTML template for a book creation form. It features a button at the top with the text "list" and a router link to `/books`. Below the button is a form with three input fields: "ISBN", "Title", and "Author". Each input field is wrapped in a `<mat-form-field>` component and includes a label, an input element, and an error message. The error messages are displayed using `<mat-error>` and `` elements, with the text "Please enter ISBN" and "Please enter Book Title". The form is submitted using the `onFormSubmit` method. The status bar at the bottom shows the time as 16:35 and the date as 10/10/2021.

```
1 <div class="button-row">
2   <a mat-raised-button color="primary" [routerLink]="['/books']">
3     <mat-icon>list</mat-icon>
4   </a>
5 </div>
6 <form [formGroup]="bookForm" (ngSubmit)="onFormSubmit(bookForm.value)">
7   <mat-form-field class="example-full-width">
8     <label>
9       <input matInput placeholder="ISBN" formControlName="isbn"
10         [errorStateMatcher]="matcher">
11     </label>
12     <mat-error>
13       <span *ngIf="!bookForm.get('isbn').valid && bookForm.get('isbn').touched">Please enter ISBN</span>
14     </mat-error>
15   </mat-form-field>
16   <mat-form-field class="example-full-width">
17     <label>
18       <input matInput placeholder="Title" formControlName="title"
19         [errorStateMatcher]="matcher">
20     </label>
21     <mat-error>
22       <span *ngIf="!bookForm.get('title').valid && bookForm.get('title').touched">Please enter Book Title</span>
23     </mat-error>
24   </mat-form-field>
25   <mat-form-field class="example-full-width">
26     <label>
27       <input matInput placeholder="Author" formControlName="author"
28         [errorStateMatcher]="matcher">
29     </label>
30   </mat-form-field>
31 </form>
```

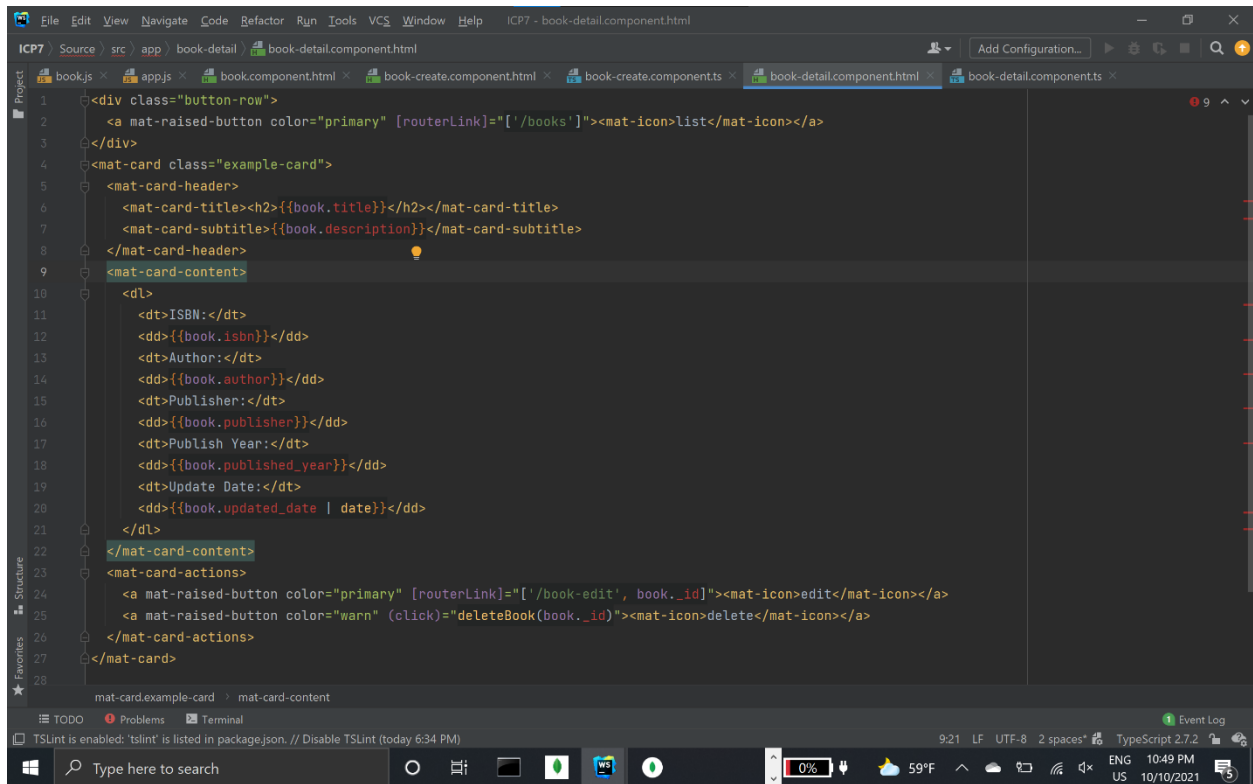
Below is the code for typescript file for book create



The screenshot shows a Visual Studio Code editor window with the file `book-create.component.ts` open. The code is a TypeScript class for the `BookCreateComponent`. It includes a constructor that takes `Router`, `ApiService`, and `FormBuilder` as arguments. The `ngOnInit` method initializes the `bookForm` using `FormBuilder.group` with a configuration object that defines the required fields: `isbn`, `title`, `description`, `author`, `publisher`, and `published_year`. The `onFormSubmit` method calls `api.postBook` with the form data, subscribes to the response, and navigates to the `/book-details` route with the book ID. The status bar at the bottom shows the time as 32:52 and the date as 10/10/2021.

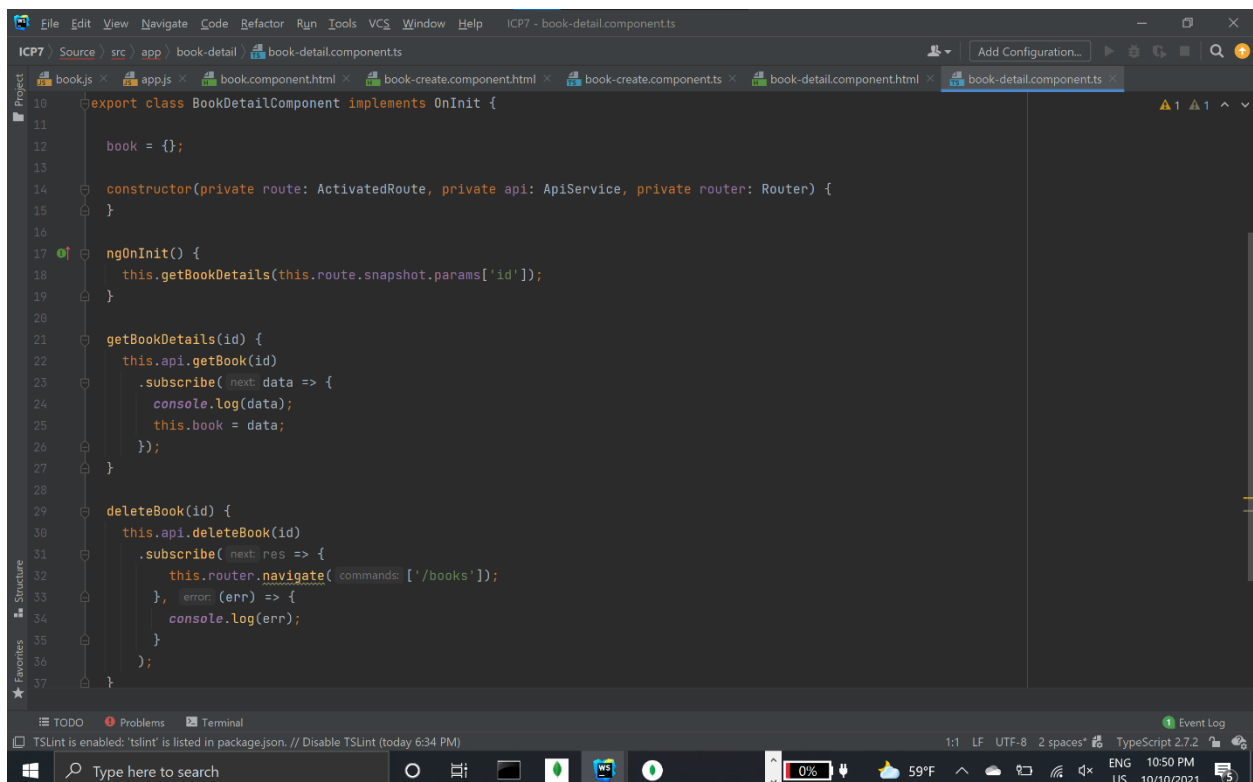
```
15 title = '';
16 description = '';
17 author = '';
18 publisher = '';
19 published_year = '';
20 matcher: any;
21
22 constructor(private router: Router, private api: ApiService, private formBuilder: FormBuilder) {
23 }
24
25 ngOnInit() {
26   this.bookForm = this.formBuilder.group(controlsConfig: {
27     'isbn': [null, Validators.required],
28     'title': [null, Validators.required],
29     'description': [null, Validators.required],
30     'author': [null, Validators.required],
31     'publisher': [null, Validators.required],
32     'published_year': [null, Validators.required]
33   });
34 }
35
36 onFormSubmit(form: NgForm) {
37   this.api.postBook(form)
38     .subscribe(next res => {
39       const id = res['_id'];
40       this.router.navigate(commands: ['/book-details', id]);
41     }, error (err) => {
42       console.log(err);
43     });
44 }
```

Below is the html code for book details and deletion of book



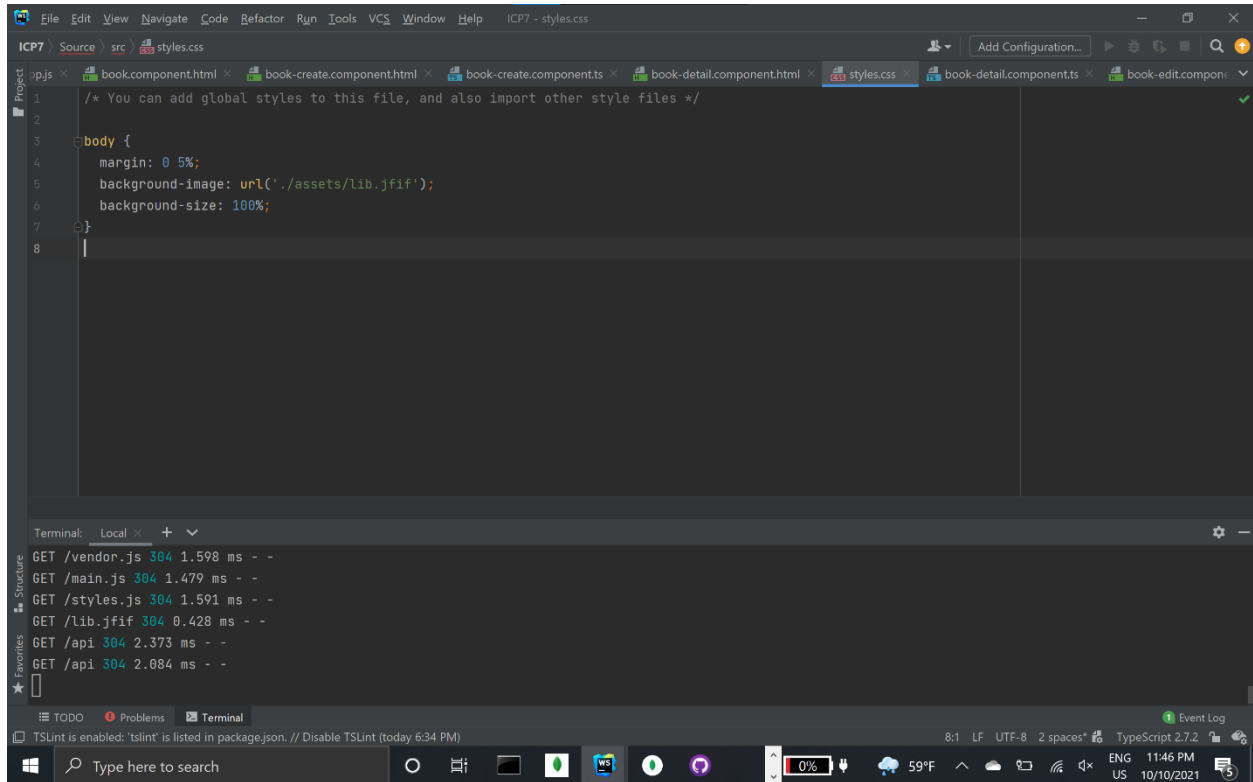
```
1 <div class="button-row">
2   <a mat-raised-button color="primary" [routerLink]="['/books']"><mat-icon>list</mat-icon></a>
3 </div>
4 <mat-card class="example-card">
5   <mat-card-header>
6     <mat-card-title><h2>{{book.title}}</h2></mat-card-title>
7     <mat-card-subtitle>{{book.description}}</mat-card-subtitle>
8   </mat-card-header>
9   <mat-card-content>
10    <dl>
11      <dt>ISBN:</dt>
12      <dd>{{book.isbn}}</dd>
13      <dt>Author:</dt>
14      <dd>{{book.author}}</dd>
15      <dt>Publisher:</dt>
16      <dd>{{book.publisher}}</dd>
17      <dt>Publish Year:</dt>
18      <dd>{{book.published_year}}</dd>
19      <dt>Update Date:</dt>
20      <dd>{{book.updated_date | date}}</dd>
21    </dl>
22  </mat-card-content>
23  <mat-card-actions>
24    <a mat-raised-button color="primary" [routerLink]="['/book-edit', book._id]"><mat-icon>edit</mat-icon></a>
25    <a mat-raised-button color="warn" (click)="deleteBook(book._id)"><mat-icon>delete</mat-icon></a>
26  </mat-card-actions>
27 </mat-card>
28
```

Typescript code for book details and deletion of book



```
10 export class BookDetailComponent implements OnInit {
11
12   book = {};
13
14   constructor(private route: ActivatedRoute, private api: ApiService, private router: Router) {
15   }
16
17   ngOnInit() {
18     this.getBookDetails(this.route.snapshot.params['id']);
19   }
20
21   getBookDetails(id) {
22     this.api.getBook(id)
23       .subscribe( next: data => {
24         console.log(data);
25         this.book = data;
26       });
27   }
28
29   deleteBook(id) {
30     this.api.deleteBook(id)
31       .subscribe( next: res => {
32         this.router.navigate( ['../books'] );
33       }, error: (err) => {
34         console.log(err);
35       });
36   }
37 }
```

Below is the css code for adding the background image.



The screenshot shows the Visual Studio Code editor interface. The main editor window displays a CSS file named `styles.css` with the following content:

```
1 /* You can add global styles to this file, and also import other style files */
2
3 body {
4   margin: 0 5%;
5   background-image: url('../assets/lib.jfif');
6   background-size: 100%;
7 }
8
```

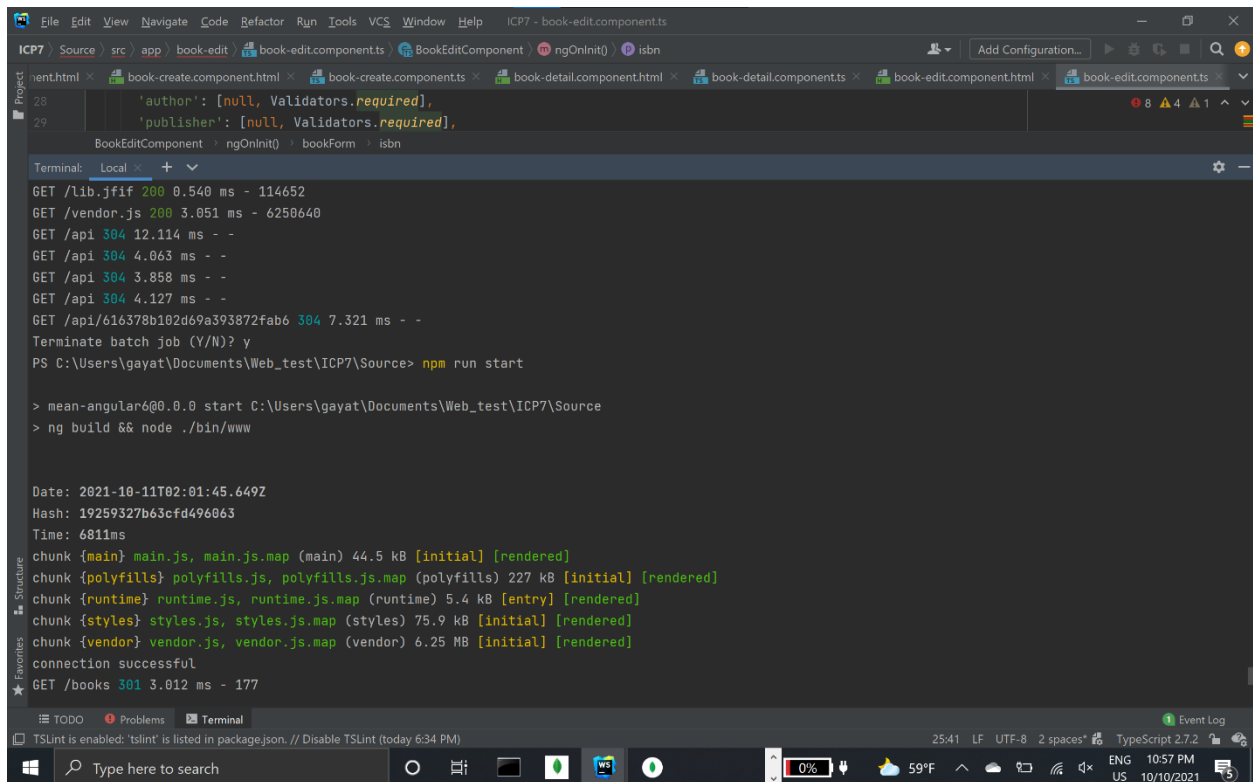
Below the editor, the Terminal window is open, showing the output of a command. The terminal displays the following network requests and their durations:

```
GET /vendor.js 304 1.598 ms - -
GET /main.js 304 1.479 ms - -
GET /styles.js 304 1.591 ms - -
GET /lib.jfif 304 0.428 ms - -
GET /api 304 2.373 ms - -
GET /api 304 2.084 ms - -
```

The status bar at the bottom indicates the file encoding is UTF-8, the line length is 80, and the tab width is 2 spaces. The system tray shows the date and time as 10/10/2021, 11:46 PM.

OUTPUT:

By executing the command "npm run start" we have established connection and after the execution displays "Connection successful".



```
ICP7 - book-edit.component.ts
ICP7 Source src app book-edit book-edit.component.ts BookEditComponent ngOnInit isbn
book-edit.component.html
'author': [null, Validators.required],
'publisher': [null, Validators.required],
BookEditComponent ngOnInit bookForm isbn

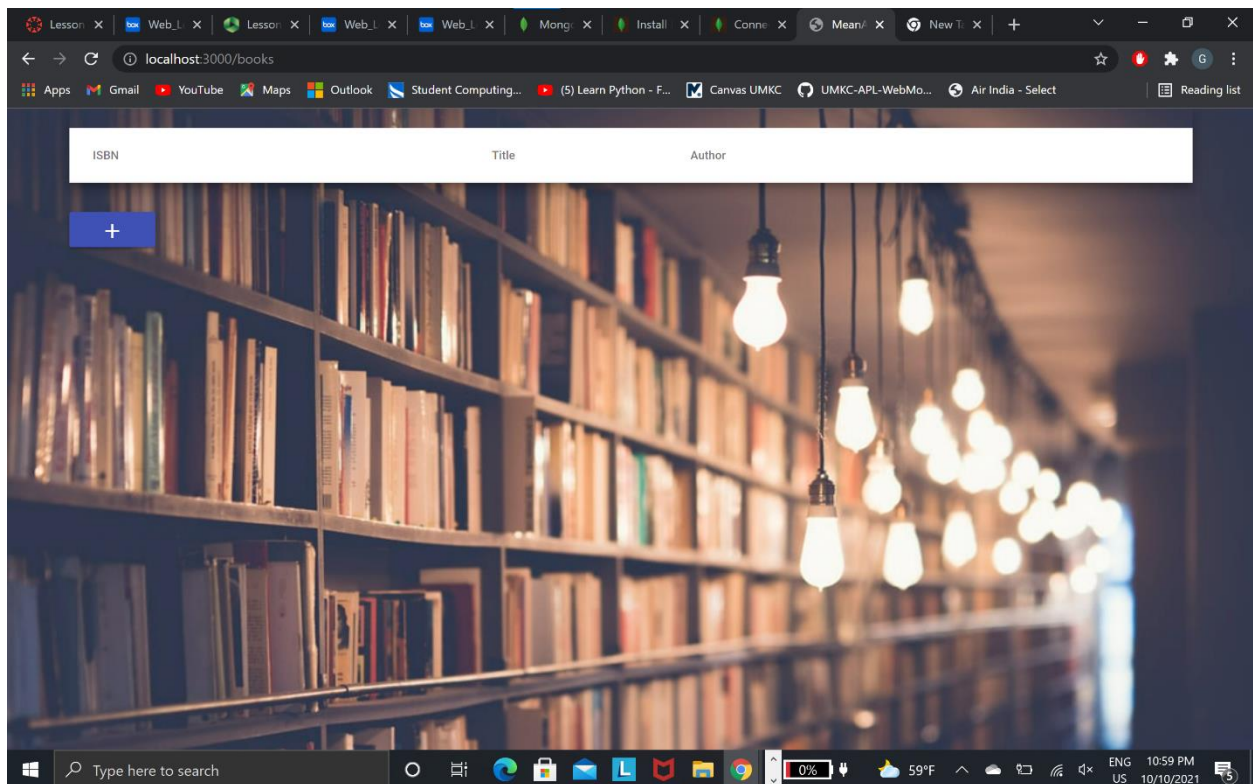
Terminal: Local
GET /lib.js 200 0.540 ms - 114652
GET /vendor.js 200 3.051 ms - 6250640
GET /api 304 12.114 ms - -
GET /api 304 4.063 ms - -
GET /api 304 3.858 ms - -
GET /api 304 4.127 ms - -
GET /api/616378b102d69a393872fab6 304 7.321 ms - -
Terminate batch job (Y/N)? y
PS C:\Users\gayat\Documents\Web_test\ICP7\Source> npm run start

> mean-angular6@0.0.0 start C:\Users\gayat\Documents\Web_test\ICP7\Source
> ng build && node ./bin/www

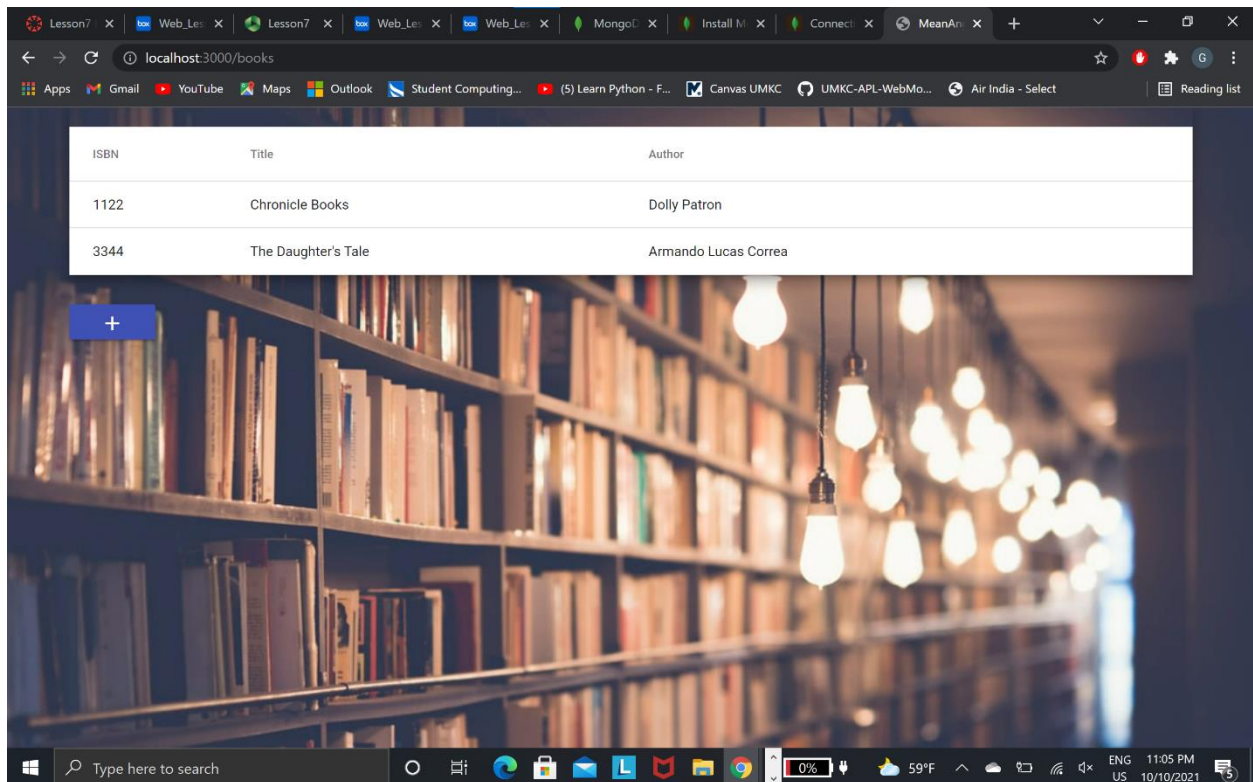
Date: 2021-10-11T02:01:45.649Z
Hash: 19259327b63cfd496063
Time: 6811ms
chunk {main} main.js, main.js.map (main) 44.5 kB [initial] [rendered]
chunk {polyfills} polyfills.js, polyfills.js.map (polyfills) 227 kB [initial] [rendered]
chunk {runtime} runtime.js, runtime.js.map (runtime) 5.4 kB [entry] [rendered]
chunk {styles} styles.js, styles.js.map (styles) 75.9 kB [initial] [rendered]
chunk {vendor} vendor.js, vendor.js.map (vendor) 6.25 MB [initial] [rendered]
connection successful
GET /books 301 3.012 ms - 177

TODO Problems Terminal
TSLint is enabled: 'tslint' is listed in package.json. // Disable TSLint (today 6:34 PM)
```

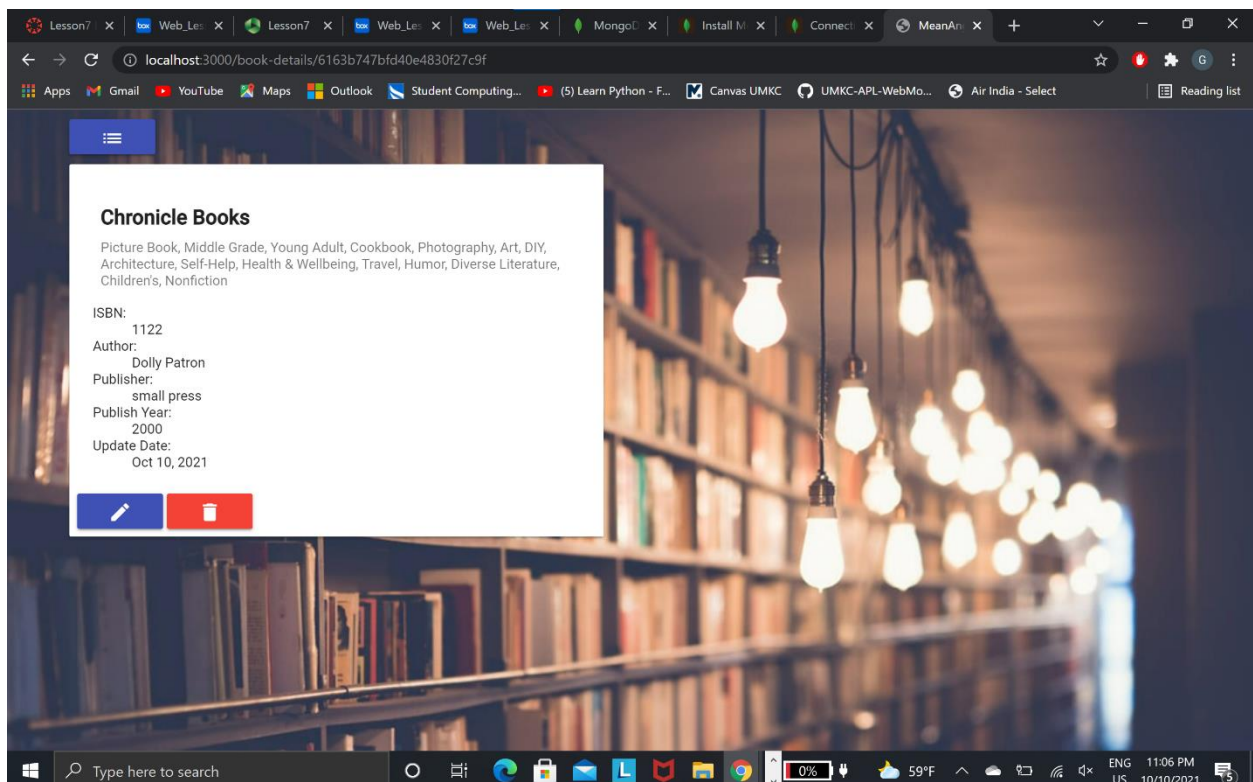
Below is the initial screen displayed when the application is executed.



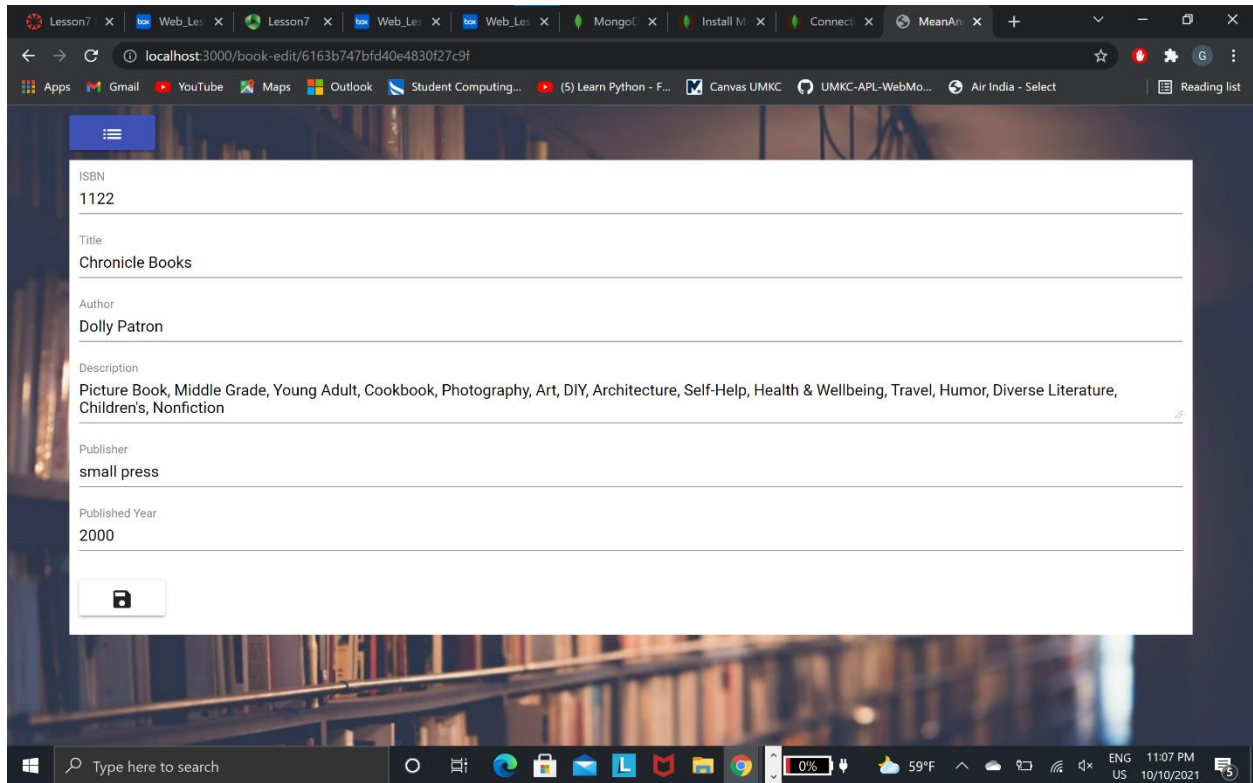
After adding the books



Below displays the books information.



Below displays the update function of the application. Here, we have updated the publisher for Chroniclebooks.



Lesson7 x Web_Les x Lesson7 x Web_Les x Web_Les x MongoD x Install M x Connect x MeanAn x +

localhost:3000/book-edit/6163b747bfd40e4830f27c9f

Apps Gmail YouTube Maps Outlook Student Computing... (5) Learn Python - F... Canvas UMKC UMKC-APL-WebMo... Air India - Select Reading list

ISBN
1122


Title
Chronicle Books

Author
Dolly Patron

Description
Picture Book, Middle Grade, Young Adult, Cookbook, Photography, Art, DIY, Architecture, Self-Help, Health & Wellbeing, Travel, Humor, Diverse Literature, Children's, Nonfiction

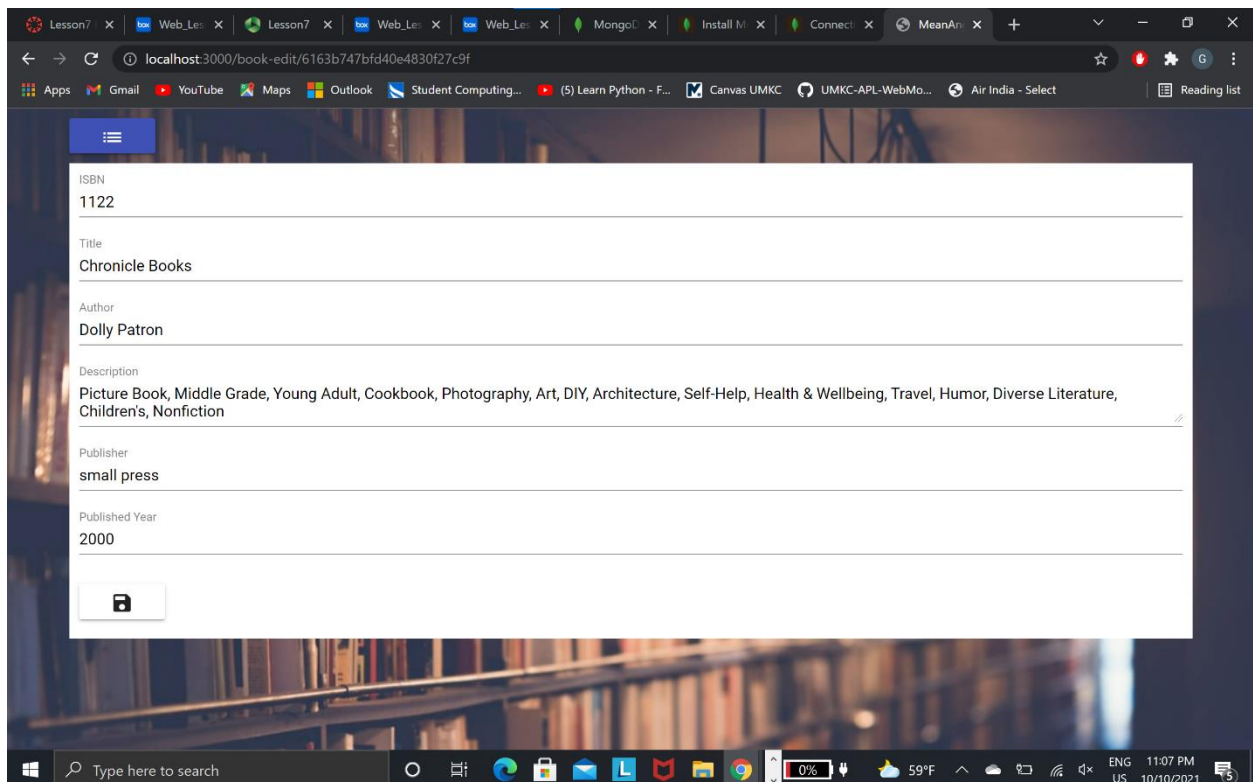
Publisher
small press

Published Year
2000



Type here to search

0% 59°F ENG US 11:07 PM 10/10/2021



Lesson7 x Web_Les x Lesson7 x Web_Les x Web_Les x MongoD x Install M x Connect x MeanAn x +

localhost:3000/book-edit/6163b747bfd40e4830f27c9f

Apps Gmail YouTube Maps Outlook Student Computing... (5) Learn Python - F... Canvas UMKC UMKC-APL-WebMo... Air India - Select Reading list

ISBN
1122


Title
Chronicle Books

Author
Dolly Patron

Description
Picture Book, Middle Grade, Young Adult, Cookbook, Photography, Art, DIY, Architecture, Self-Help, Health & Wellbeing, Travel, Humor, Diverse Literature, Children's, Nonfiction

Publisher
small press

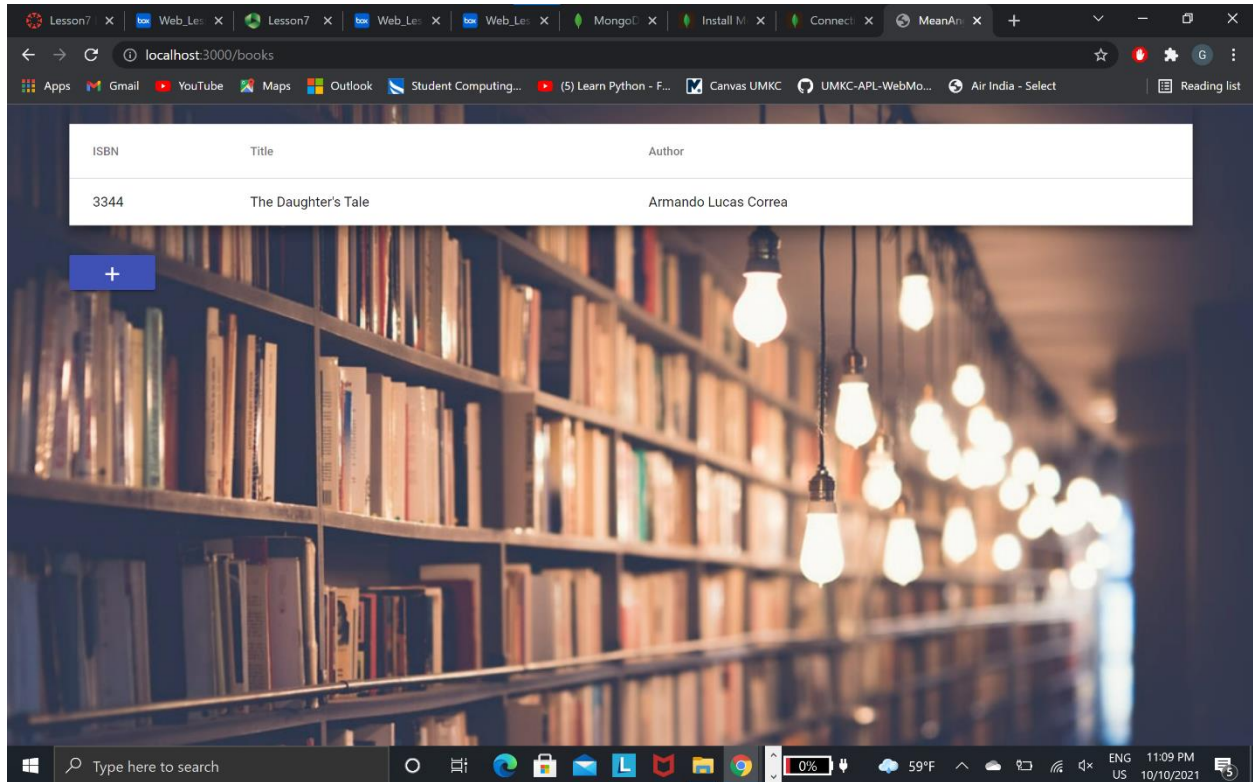
Published Year
2000



Type here to search

0% 59°F ENG US 11:07 PM 10/10/2021

Deleted the book Chroniclebooks



Video link: <https://youtu.be/AYAGZugibMo>

