

- **Eclipse IDE**

- **Installation:**

<https://eclipseide.org/>

- **Set – Up**

File-Switch workspace

- **Importing files**

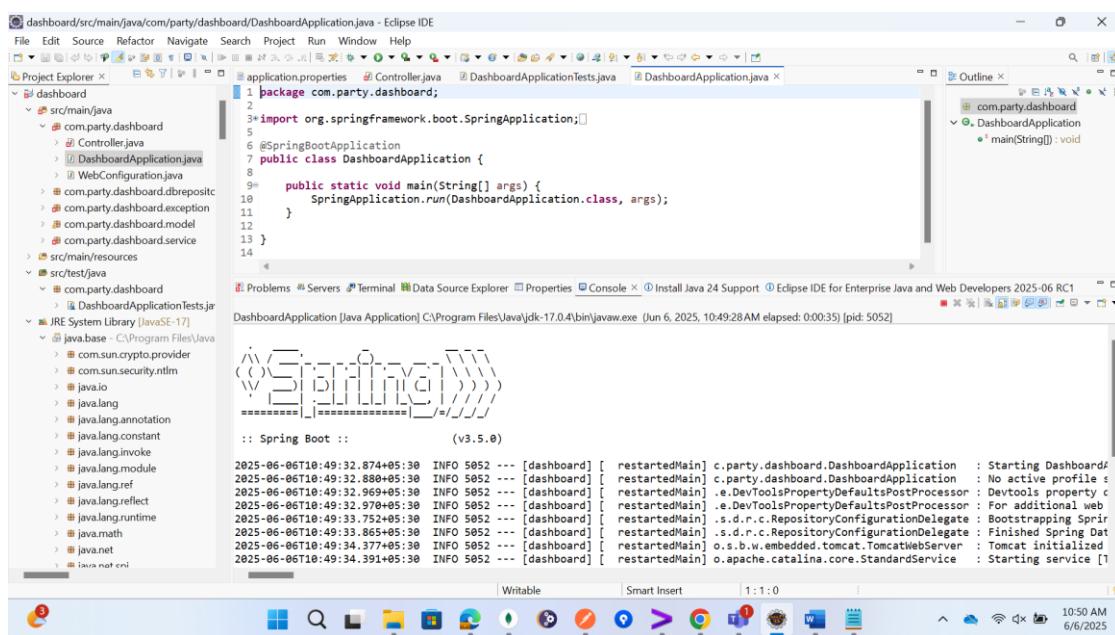
Import project-Existing projects into workspace

- **Main class**

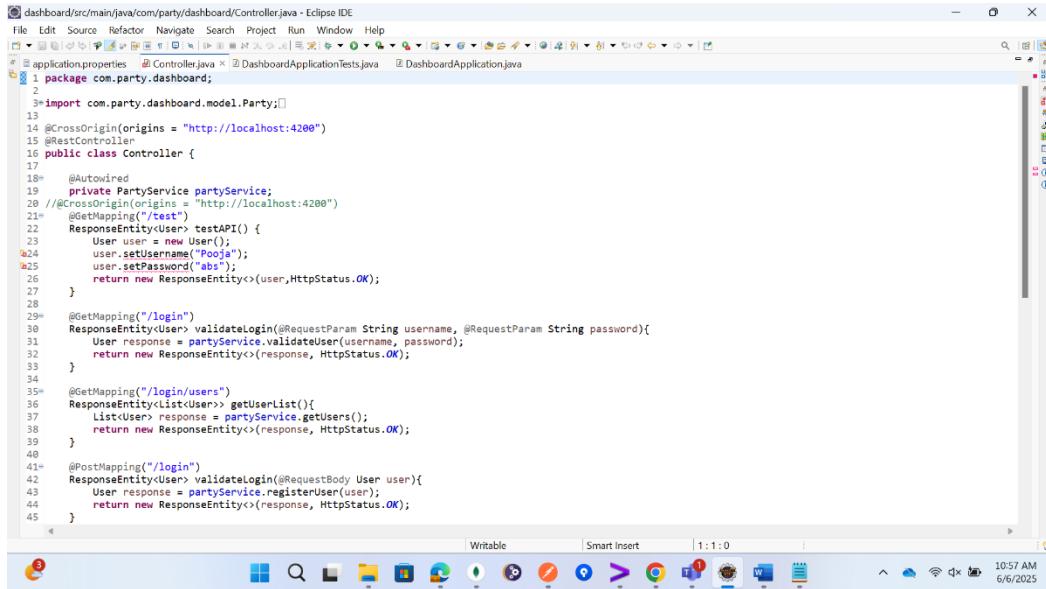
Dashboardapplication.java

```
package com.party.dashboard;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class DashboardApplication {
    public static void main(String[] args) {
        SpringApplication.run(DashboardApplication.class, args);
    }
}
```

}



- Controller.java



The screenshot shows the Eclipse IDE interface with the title bar "dashboard/src/main/java/com/party/dashboard/Controller.java - Eclipse IDE". The code editor displays the Controller.java file, which contains Java code for a Spring Boot REST controller. The code includes annotations like @RestController, @GetMapping, and @PostMapping, and imports from com.party.dashboard.model.Party and com.party.dashboard.service.PartyService. The code handles various API endpoints such as /test, /login, and /login/users. The bottom status bar shows the date and time as 10:57 AM 6/6/2025.

```
1 package com.party.dashboard;
2
3 import com.party.dashboard.model.Party;
4
5 import org.springframework.http.HttpHeaders;
6 import org.springframework.http.HttpStatus;
7 import org.springframework.http.ResponseEntity;
8 import org.springframework.web.bind.annotation.*;
9
10 import com.party.dashboard.model.User;
11 import com.party.dashboard.model.UserResponse;
12 import com.party.dashboard.service.PartyService;
13
14 @CrossOrigin(origins = "http://localhost:4200")
15 @RestController
16 public class Controller {
17
18     @Autowired
19     private PartyService partyService;
20
21     @GetMapping("/test")
22     ResponseEntity<User> testAPI() {
23         User user = new User();
24         user.setUsername("Pooya");
25         user.setPassword("abs");
26
27         return new ResponseEntity<>(user,HttpStatus.OK);
28     }
29
30     @GetMapping("/login")
31     ResponseEntity<User> validateLogin(@RequestParam String username, @RequestParam String password){
32         User response = partyService.validateUser(username, password);
33
34         return new ResponseEntity<>(response, HttpStatus.OK);
35     }
36     @GetMapping("/login/users")
37     ResponseEntity<List<User>> getUserList(){
38         List<User> response = partyService.getUsers();
39
40         return new ResponseEntity<>(response, HttpStatus.OK);
41     }
42     @PostMapping("/login")
43     ResponseEntity<User> validateLogin(@RequestBody User user){
44         User response = partyService.registerUser(user);
45
46         return new ResponseEntity<>(response, HttpStatus.OK);
47     }
}
```

- DashboardApplicationTests.java
package com.party.dashboard;

```
import org.junit.jupiter.api.Test;
import org.springframework.boot.test.context.SpringBootTest;
```

```
@SpringBootTest
class DashboardApplicationTests {
```

```
    @Test
    void contextLoads() {
    }
}
▪ Technologies Used in Your Spring Boot Project
```

1. Backend Framework

-Spring Boot (v3.5.0)

<https://start.spring.io/>

2. Database

Mongodb

<https://www.mongodb.com/try/download/community>

3. Programming Language

[Download Java JDK 17.0.4 \(64-bit\) from FileHorse.com](#)

4. IDE

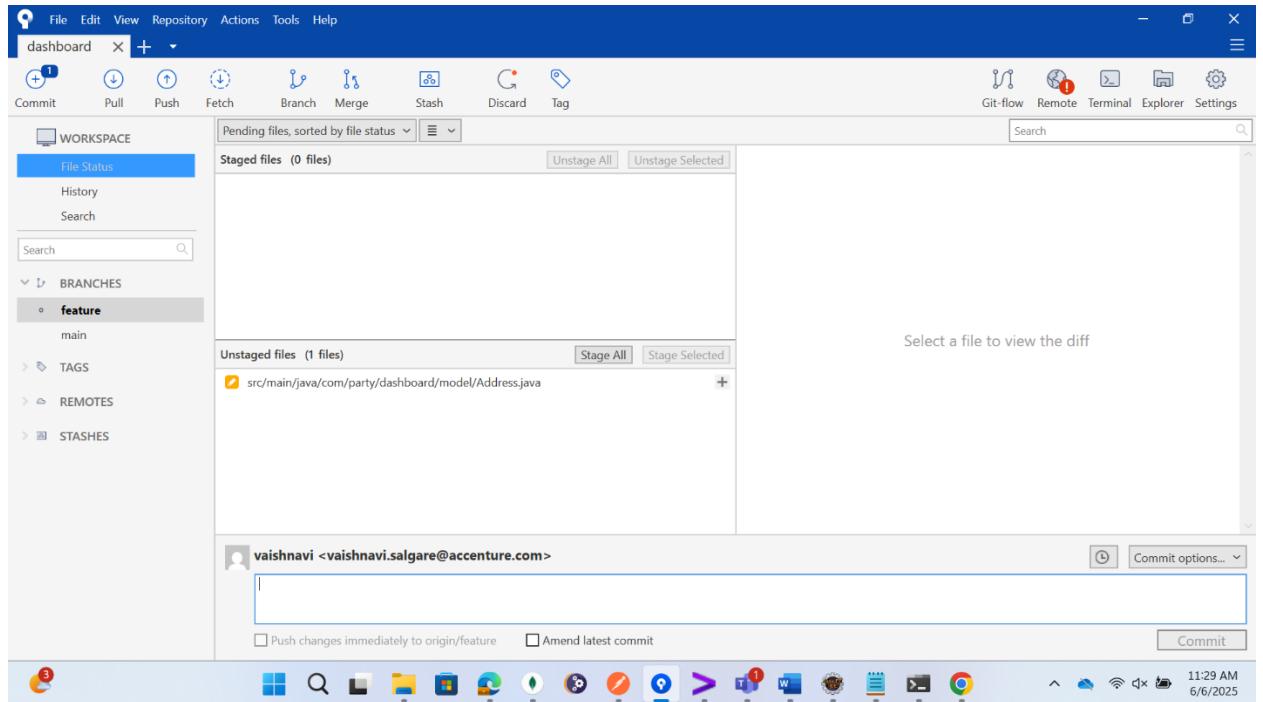
Eclipse IDE for Enterprise Java

Link: <https://eclipseide.org/>

5. sourceTree

<https://www.sourcetreeapp.com/>

- Initialize a repo with .gitignore
- Stage files → Write message → Click Commit
- Push to GitHub as backup or for team collaboration
- Pull to GitHub
- merge to GitHub



■ Tools

1. Git & GitHub

<https://git-scm.com/downloads>

2. Postman

-Testing tool

<https://www.postman.com/downloads/>

The screenshot shows the Postman application interface. On the left, the sidebar has sections for 'My Workspace' (Collections, Environments, Flows, History), 'Overview', 'Getting started', and 'No environment'. The main area shows a POST request to 'http://localhost:4200'. The 'Body' tab is selected, showing a JSON payload:

```
1 {  
2   "name": "vaishnavi salgare",  
3   "email": "vaishnavi.salgare@accenture.com"  
4 }
```

The 'Response' section below shows a placeholder image of a rocket launching and the message 'Could not send request'.

3. Mongodb

Mongodb installer:

<https://www.mongodb.com/try/download/community>

Mongodb GUI:<https://www.mongodb.com/try/download/compass>

The screenshot shows the MongoDB Compass application. The left sidebar has 'Compass' and 'Connections (0)' sections. The main area features a magnifying glass icon over a cloud, with the text 'Welcome to MongoDB Compass' and 'To get started, connect to an existing server or + Add new connection'. A green button at the bottom right says 'CREATE FREE CLUSTER'.

▪ Created New Database

Database name: party

Inserted records from party.java :

The screenshot shows the MongoDB Compass interface. In the left sidebar under 'CONNECTIONS', there are two entries: 'local' and 'testDB'. Under 'testDB', the 'party' collection is selected. The main pane displays a single document in the 'Documents' tab. The document structure is as follows:

```
{  
  "_id": {  
    "$oid": "6842b0c3025d232dd1bf9068"  
  },  
  "firstName": "Alice",  
  "lastName": "bonus",  
  "dateOfBirth": {  
    "$date": "2000-07-14T00:00:00.000Z"  
  },  
  "address": {  
    "country": "India",  
    "state": "Maharashtra",  
    "city": "Pune"  
  },  
  "ContactChannel": {  
    "phoneNumber": 7972345,  
    "emailAddress": "abc@gmail.com"  
  }  
}
```

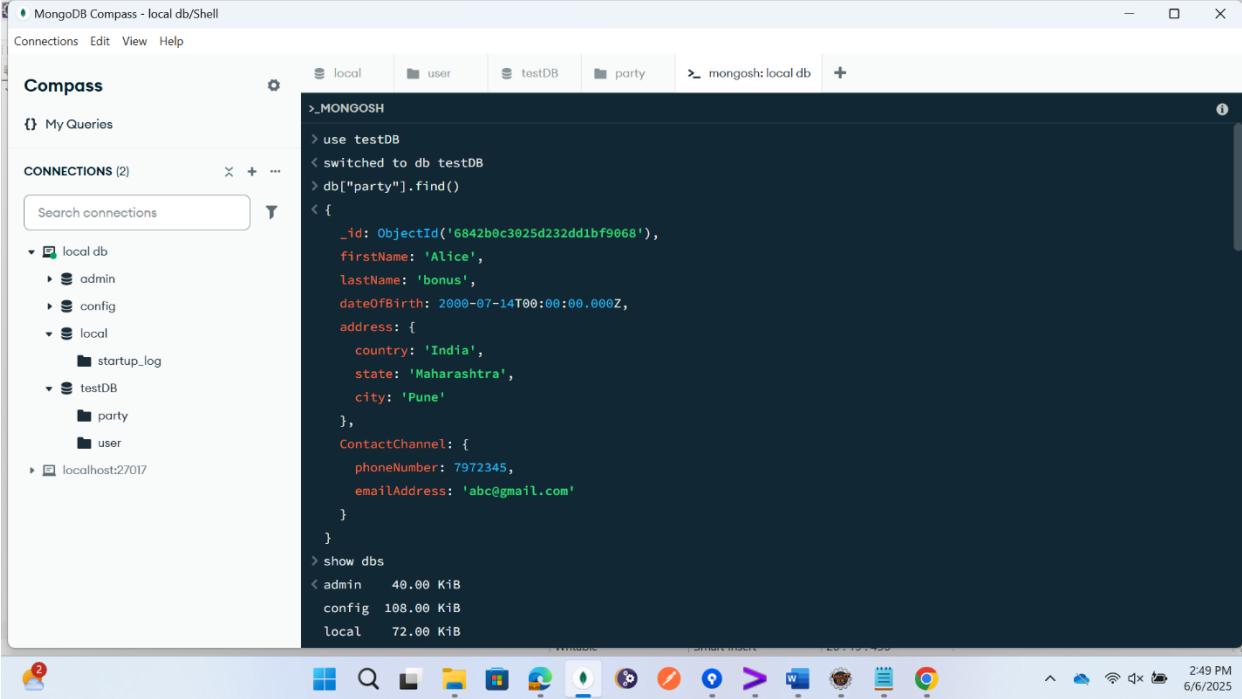
Below the document, there are buttons for 'ADD DATA', 'EXPORT DATA', 'UPDATE', and 'DELETE'. The status bar at the bottom right shows the time as 2:48 PM and the date as 6/6/2025.

The screenshot shows the Eclipse IDE interface. The 'Project Explorer' view on the left lists several Java packages and files. The 'Party.java' file is open in the code editor. The code defines a 'Party' class with attributes like id, firstName, lastName, dateOfBirth, address, contactChannel, genderIdentity, and partyId. The 'Outline' view on the right shows the class hierarchy and its members. The status bar at the bottom right shows the time as 2:52 PM and the date as 6/6/2025.

```
1 package com.party.dashboard.model;  
2  
3 import org.springframework.data.mongodb.core.mapping.Document;  
4  
5 @Data  
6 @Document(collection="party")  
7 public class Party {  
8     @Id  
9     String id;  
10    String firstName;  
11    String lastName;  
12    String dateOfBirth;  
13    Address address;  
14    ContactChannel contactChannel;  
15    String genderIdentity;  
16    String partyId;  
17  
18 }
```

- **MongoDB Shell:**

- `db["party"].find()`

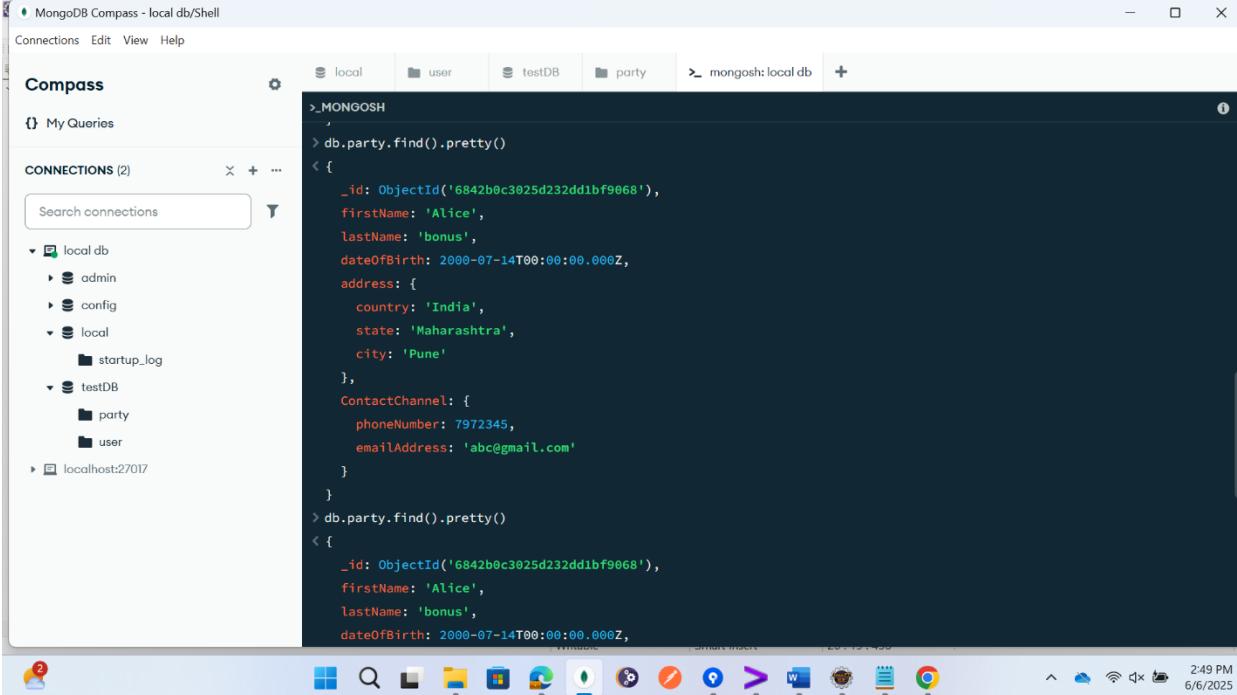


The screenshot shows the MongoDB Compass interface. On the left, the 'Connections' sidebar lists 'local db', 'testDB', and 'party'. The main panel displays the MongoDB shell command history and its output. The command `db["party"].find()` is run, and the response shows a single document from the 'party' collection:

```
>_MONGOSH
> use testDB
< switched to db testDB
> db["party"].find()
< [
  {
    "_id": ObjectId('6842b0c3025d232dd1bf9068'),
    "firstName": 'Alice',
    "lastName": 'bonus',
    "dateOfBirth": 2000-07-14T00:00:00.000Z,
    "address": {
      "country": 'India',
      "state": 'Maharashtra',
      "city": 'Pune'
    },
    "ContactChannel": {
      "phoneNumber": 7972345,
      "emailAddress": 'abc@gmail.com'
    }
  }
]
> show dbs
< admin   40.00 KiB
config  108.00 KiB
local    72.00 KiB
```

- `db.party.find().pretty():`

- Proper formatted JSON



The screenshot shows the MongoDB Compass interface. The 'Connections' sidebar lists 'local db', 'testDB', and 'party'. The main panel displays the MongoDB shell command history and its output. The command `db.party.find().pretty()` is run twice, and the response shows the same document from the 'party' collection, but it is displayed in a more readable, pretty-printed JSON format:

```
>_MONGOSH
> db.party.find().pretty()
< [
  {
    "_id": ObjectId('6842b0c3025d232dd1bf9068'),
    "firstName": "Alice",
    "lastName": "bonus",
    "dateOfBirth": 2000-07-14T00:00:00.000Z,
    "address": {
      "country": "India",
      "state": "Maharashtra",
      "city": "Pune"
    },
    "ContactChannel": {
      "phoneNumber": 7972345,
      "emailAddress": "abc@gmail.com"
    }
  }
]
> db.party.find().pretty()
< [
  {
    "_id": ObjectId('6842b0c3025d232dd1bf9068'),
    "firstName": "Alice",
    "lastName": "bonus",
    "dateOfBirth": 2000-07-14T00:00:00.000Z,
```

❖ API Testing (Postman):

1. HTTP Request:

- **GET Request:** To retrieve or fetch data
- **POST Request:** To create and update data
- **PUT Request:** To update data
- **DELETE Request:** For deleting data

The screenshot shows the Postman application interface. On the left, the 'My Workspace' sidebar lists collections, environments, flows, and history. The 'mycollection' collection is expanded, showing several test cases: 'GET validate user', 'POST save users', 'GET test party', 'POST party', 'GET test partyl', 'GET put party', and others. The main workspace displays a 'validate user' test case. The 'Params' tab is selected, showing query parameters: 'username' with value 'test' and 'password' with value 'test123'. The 'Send' button is visible at the top right of the request configuration area.

This screenshot shows another view of the Postman interface. The 'mycollection' collection is selected in the sidebar. A 'test partyId' test case is being edited. The 'Body' tab is selected, showing a JSON payload: { "id": { "\$oid": "6842b21b025d232dd1bf906c" }, "firstName": "Alice", "lastName": "bonus123", "dateOfBirth": { "\$date": "2000-07-14T00:00:00.000Z" }, "address": { "country": "India", "state": "Maharashtra", "city": "Pune" }, "ContactChannel": { "phoneNumber": 7972345, "emailAddress": "abc@gmail.com" } }. The 'JSON' dropdown is selected under the body type. The 'Send' button is visible at the top right.

The screenshot shows the Postman application interface. On the left, the sidebar displays 'My Workspace' with various collections and environments. The main area shows a POST request to 'http://localhost:8081/login'. The 'Body' tab is selected, showing the following JSON payload:

```
1 "username": "abc",
2 "password": "abc"
```

The 'Response' tab below contains a cartoon illustration of a rocket launching.

The screenshot shows the MongoDB Compass application interface. The left sidebar shows connections and databases. The main area is focused on the 'party' collection within the 'testDB' database. The 'Documents' tab is selected, displaying two documents. The first document's details are visible:

```
_id: ObjectId('6842b0c3025d232dd1bf9068')
firstName : "Alice"
lastName : "bonus"
dateOfBirth : 2000-07-14T00:00:00.000+00:00
address : Object
ContactChannel : Object
```

The second document's details are partially visible:

```
_id: ObjectId('6842b21b025d232dd1bf906c')
firstName : "Alice"
lastName : "bonus123"
dateOfBirth : 2000-07-14T00:00:00.000+00:00
address : Object
country : "India"
state : "Maharashtra"
city : "Pune"
ContactChannel : Object
partyId : "1"
genderIdentity : "F"
```

The screenshot shows the MongoDB Compass interface connected to the database 'testDB.party'. The left sidebar displays connections, with 'local db' expanded to show 'admin', 'config', 'local', and 'testDB' (which contains 'party' and 'user'). The main area shows a query result for documents where 'lastName' is 'bonus'. One document is highlighted, showing fields like '_id', 'firstName', 'lastName', 'dateOfBirth', 'address', and 'ContactChannel'. The top right has buttons for 'Generate query', 'Explain', 'Reset', 'Find', and 'Options'. The bottom status bar shows connection details and system icons.

MongoDB Compass - localhost:27017/testDB.party

Connections Edit View Collection Help

Compass

Welcome testDB party +

localhost:27017 > testDB > party

Documents 2 Aggregations Schema Indexes 1 Validation

Open MongoDB shell

My Queries

CONNECTIONS (2)

Search connections

local db

- admin
- config
- local
- testDB
 - party
 - user

localhost:27017

- admin
- config
- local
- testDB
 - party
 - user

ADD DATA EXPORT DATA UPDATE DELETE 25 1-1 of 1

Generate query Explain Reset Find Options

_id: ObjectId('6842b0c3025d232dd1bf9068')
firstName: "Alice"
lastName: "bonus"
dateOfBirth: 2000-07-14T00:00:00.000+00:00
address: Object
ContactChannel: Object

3 111, 101 99 of 304 CHARACTERS

100/70 WINDOWS (C:\KLF) 017/6

12:52 PM 6/10/2025

The screenshot shows the Postman application interface. On the left, the sidebar displays 'My Workspace' with sections for 'Collections', 'Environments', 'Flows', and 'History'. The main workspace shows a collection named 'mycollection' containing several requests:

- Process a VALID transaction
- Attempt an INVALID transaction
- mycollection**
 - GET test API
 - GET validate user
 - POST save users**
 - GET parties
 - POST party**
 - GET party/partId**
 - PUT put party
 - DELETE Delete Party
- New Collection
- REST API basics: CRUD, test & variable

The 'GET party/partId' request is currently selected. The request details panel shows:
Method: GET
URL: http://localhost:8080/party/2
Params: none
Headers: (6)
Body: raw (selected)
Cookies: (1)
Test Results: 200 OK (17 ms, 564 B)
Body content:

```
{ "id": "684812d16a2bcc2a53037d76", "firstName": "Demo", "lastName": "Demo", "dateOfBirth": "2000-07-14T00:00:00.000+00:00", "address": { "country": "India", "state": "Maharashtra", "city": "Pune" }, "contactChannel": { "phoneNumber": "12345", "emailAddress": "abc@gmail.com" } }
```

- Delete:

The screenshot shows the Postman interface. On the left, the 'My Workspace' sidebar is open, showing various collections and environments. The 'mycollection' collection is expanded, revealing several test cases: 'GET test API', 'GET validate user', 'POST save users', 'GET test party', 'GET test partyId', 'GET put party', and 'DEL Delete'. The 'DEL Delete' test case is currently selected. In the main workspace, a DELETE request is being made to the URL `http://localhost:8080/party/1`. The 'Params' tab is active, showing a single parameter 'Key' with the value 'Value'. The response status is 204 No Content, indicating a successful delete operation.

Task:

Adding occupation field in Party model class then Save API in occupation field add data in DB Apply same for get and update.

The screenshot shows the Eclipse IDE interface. The code editor displays the `Party.java` file from the `com.party.dashboard.model` package. A new field, `private String occupation;`, has been added to the `Party` class. The Outline view on the right shows the updated class structure, including the new `occupation` field. The bottom status bar indicates the application was started at 2:37:06 PM on June 10, 2025.

The screenshot shows the Postman application interface. On the left, the 'My Workspace' sidebar lists collections, environments, flows, and history. The main workspace displays a POST request to 'mycollection / party'. The request URL is 'http://localhost:8080/party'. The 'Body' tab is selected, showing the following JSON payload:

```
4
5   "lastName": "Demo",
6   "dateOfBirth": "2000-07-14T00:00:00.000+00:00",
7   "address": {
8     "country": "India",
9     "state": "Maharashtra",
10    "city": "Pune"
11  },
12  "contactChannel": [
13    {
14      "phoneNumber": "12345",
15      "emailAddress": "abc@gmail.com"
16    }
17  ],
18  "genderIdentity": "F",
19  "partyId": "2",
20  "occupation": "Doctor"
```

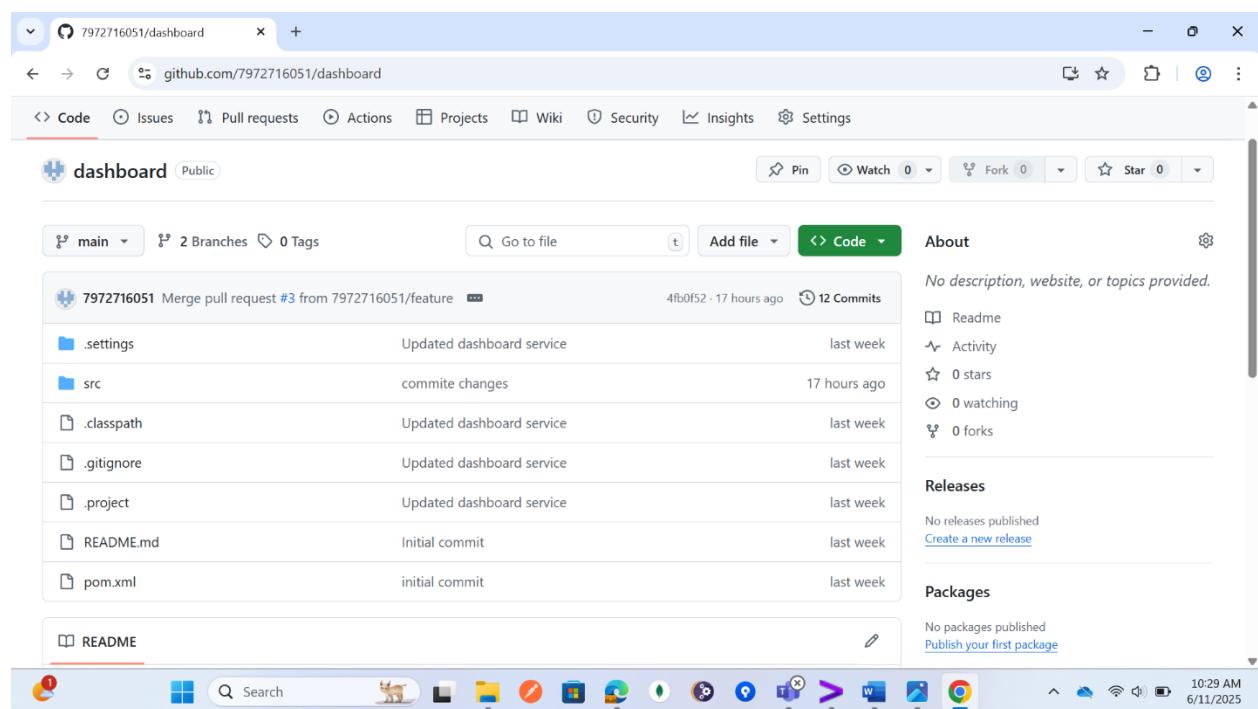
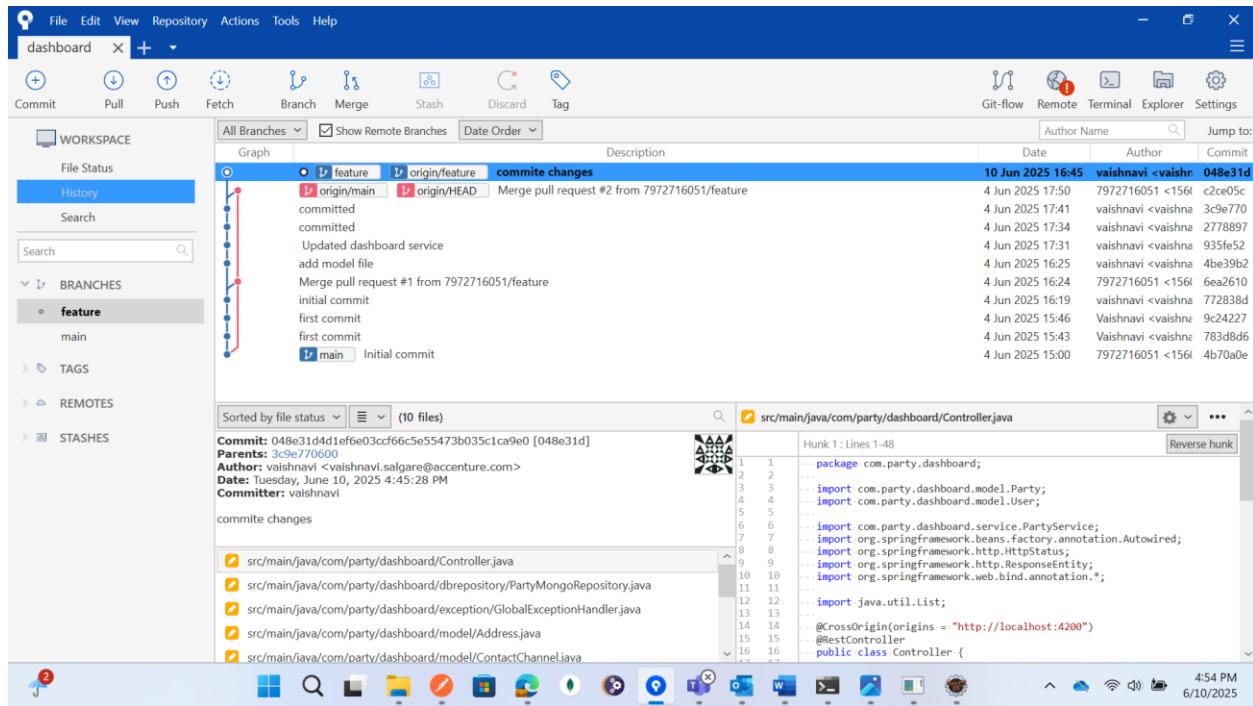
The response status is 200 OK with a duration of 93 ms and a size of 564 B. Below the JSON preview, there are tabs for Body, Cookies, Headers, Test Results, and Visualize.

The screenshot shows the MongoDB Compass application interface. The left sidebar shows connections: 'local db' (with 'admin', 'config', 'local', 'testDB' (containing 'party' and 'user')) and 'localhost:27017' (with 'admin', 'config', 'local', 'testDB' (containing 'party' and 'user')). The main workspace shows the 'testDB' database and the 'party' collection. The 'Documents' tab is selected, displaying a single document with the following data:

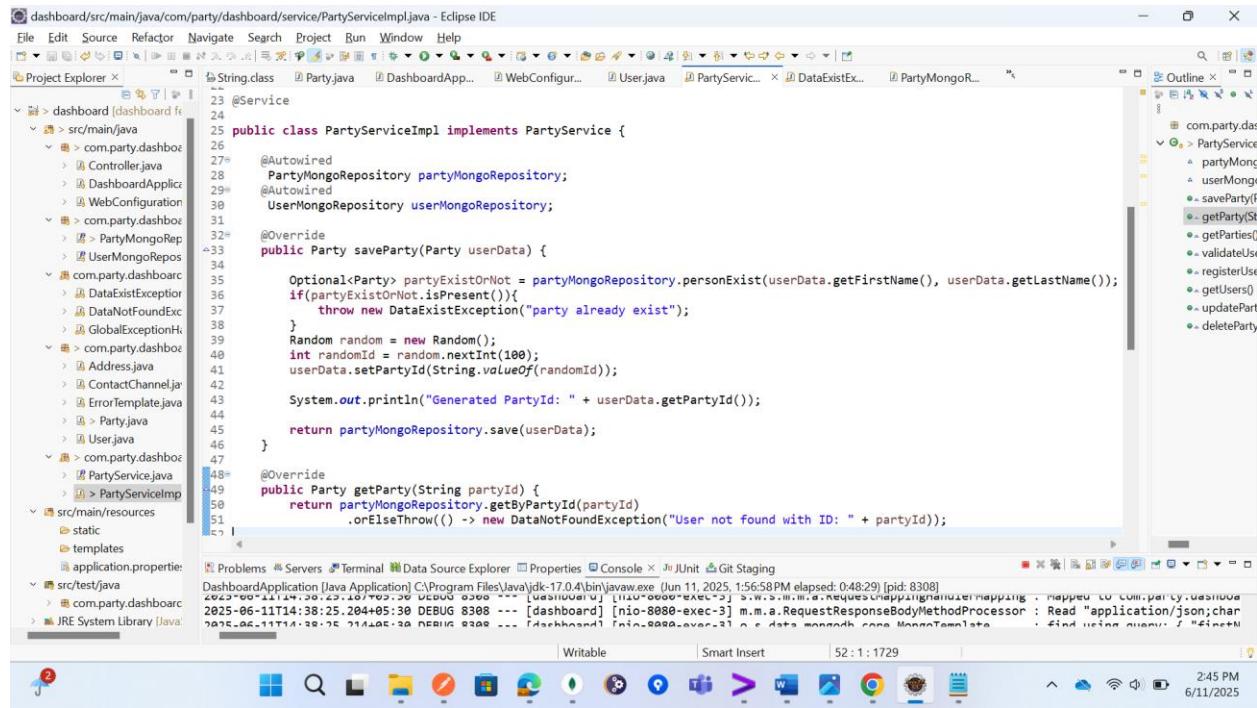
```
_id: ObjectId('6842b21b025d232dd1bf906c')
firstName: "Alice"
lastName: "bonus123"
dateOfBirth: 2000-07-14T00:00:00.000+00:00
address: Object
  ContactChannel: Object
    partyId: "1"
    genderIdentity: "F"
    occupation: "Doctor"
```

Below the document, there are buttons for 'ADD DATA', 'EXPORT DATA', 'UPDATE', and 'DELETE'. The status bar at the bottom indicates 25 documents, 1-1 of 1, and the date/time 6/10/2025 3:11 PM.

- Push , Pull and merged files into Git:



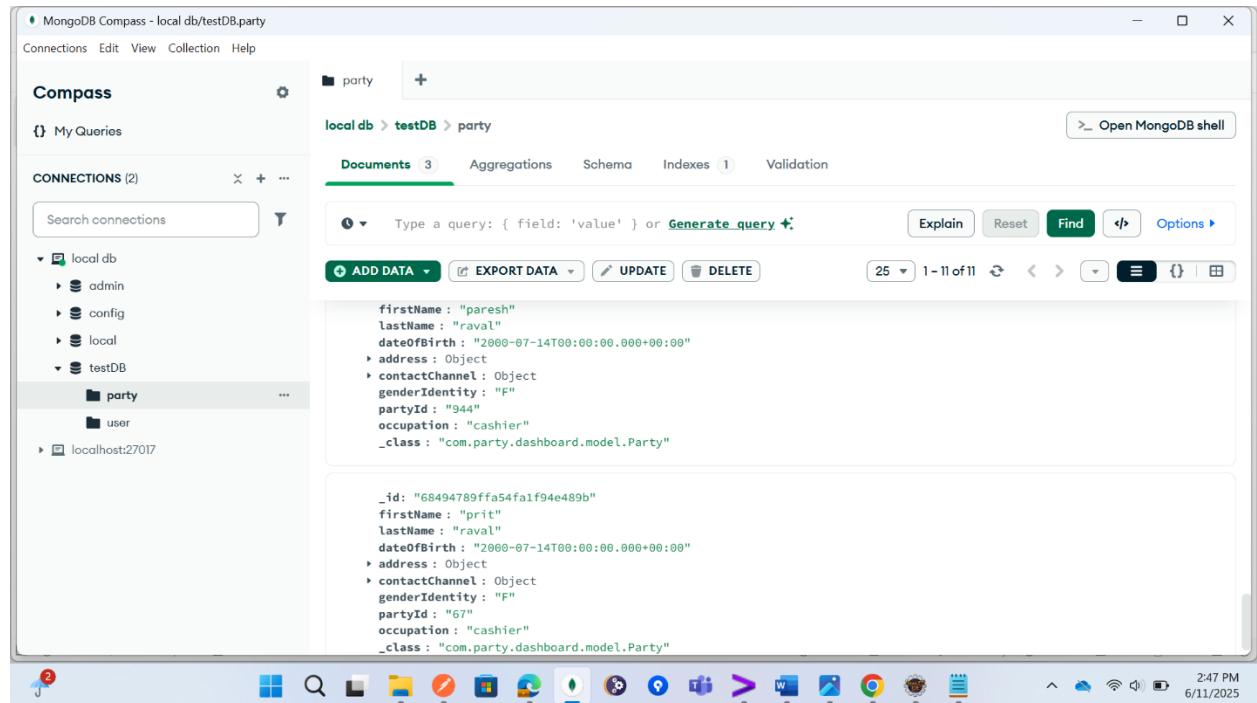
- Auto generated PartyId for users:



```

1  package com.party.dashboard.service;
2
3  import com.party.dashboard.model.Party;
4  import com.party.dashboard.model.User;
5  import com.party.dashboard.repository.PartyMongoRepository;
6  import com.party.dashboard.repository.UserMongoRepository;
7  import org.springframework.beans.factory.annotation.Autowired;
8  import org.springframework.stereotype.Service;
9
10 import static org.junit.Assert.*;
11
12 /**
13  * Service implementation for managing parties.
14  */
15 @Service
16 public class PartyServiceImpl implements PartyService {
17
18     @Autowired
19     private PartyMongoRepository partyMongoRepository;
20
21     @Autowired
22     private UserMongoRepository userMongoRepository;
23
24     @Override
25     public Party saveParty(Party userData) {
26
27         Optional<Party> partyExistOrNot = partyMongoRepository.personExist(userData.getFirstName(), userData.getLastName());
28
29         if(partyExistOrNot.isPresent()){
30             throw new DataExistException("party already exist");
31         }
32
33         Random random = new Random();
34
35         int randomId = random.nextInt(100);
36
37         userData.setPartyId(String.valueOf(randomId));
38
39         System.out.println("Generated PartyId: " + userData.getPartyId());
40
41         return partyMongoRepository.save(userData);
42     }
43
44     @Override
45     public Party getParty(String partyId) {
46
47         return partyMongoRepository.getPartyById(partyId)
48             .orElseThrow(() -> new DataNotFoundException("User not found with ID: " + partyId));
49     }
50
51 }

```



MongoDB Compass - local db/testDB.party

Connections Edit View Collection Help

Compass

My Queries

Connections (2)

Search connections

- local db
 - admin
 - config
 - local
 - testDB
 - party
 - user
- localhost:27017

local db > testDB > party

Documents 3 Aggregations Schema Indexes Validation

Type a query: { field: 'value' } or [Generate query](#) [+](#)

[EXPLAIN](#) [RESET](#) [FIND](#) [OPTIONS](#)

[ADD DATA](#) [EXPORT DATA](#) [UPDATE](#) [DELETE](#)

25 1-11 of 11

| | | |
|--------------------------------------------|-----------------------------------------------|-----------------------------------------------|
| firstName : "paresh" | lastName : "raval" | dateOfBirth : "2000-07-14T00:00:00.000+00:00" |
| address : Object | contactChannel : Object | genderIdentity : "F" |
| partyId : "944" | partyId : "67" | occupation : "cashier" |
| _class : "com.party.dashboard.model.Party" | _id: "68494789ffa54fa1f94e489b" | firstName : "prit" |
| | lastName : "raval" | lastName : "raval" |
| | dateOfBirth : "2000-07-14T00:00:00.000+00:00" | dateOfBirth : "2000-07-14T00:00:00.000+00:00" |
| | address : Object | address : Object |
| | contactChannel : Object | contactChannel : Object |
| | genderIdentity : "F" | genderIdentity : "F" |
| | partyId : "67" | partyId : "67" |
| | occupation : "cashier" | occupation : "cashier" |
| | _class : "com.party.dashboard.model.Party" | _class : "com.party.dashboard.model.Party" |

Home Workspaces API Network

Search Postman Ctrl K

Invite Upgrade

My Workspace

New Import POST save [CONFLIK mycol mycol POST par GET party PUT put pi]

HTTP mycollection / party/partyId

Save Share

Collections Environments Flows History

+

End-to-End Tests mycollection

- GET test API
- GET validate user
- POST save users
- GET parties
- POST party
- GET party/partyId
- PUT put party
- DEL Delete Party

New Collection REST API basics: CRUD, test & variable

mycollection

GET test API

GET validate user

POST save users

GET parties

POST party

GET party/partyId

PUT put party

DEL Delete Party

New Collection

REST API basics: CRUD, test & variable

HTTP mycollection / party/partyId

GET http://localhost:8080/party/96160

Params Authorization Headers (6) Body Scripts Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON

Body Cookies Headers (8) Test Results 200 OK 47 ms 567 B Save Response

{} JSON Preview Visualize

```
2 "id": "68494253ffa54fa1f94e4897",
3 "firstName": "pinki",
4 "lastName": "s",
5 "dateOfBirth": "2000-07-14T00:00:00.000+00:00",
6 "address": {
7     "country": "India",
8     "state": "Maharashtra",
9     "city": "Mumbai"
10 },
11 "contactChannel": {
12     "phoneNumber": "12345",
13     "emailAddress": "a@gmail.com"
14 },
15 "genderIdentity": "F",
16 "partyId": "96160",
```

Postbot Runner Start Proxy Cookies Vault Trash

Online Find and replace Console

2:48 PM 6/11/2025

- **Frontend Technologies:**

Angular:

- **To install Angular:**

<https://nodejs.org/>

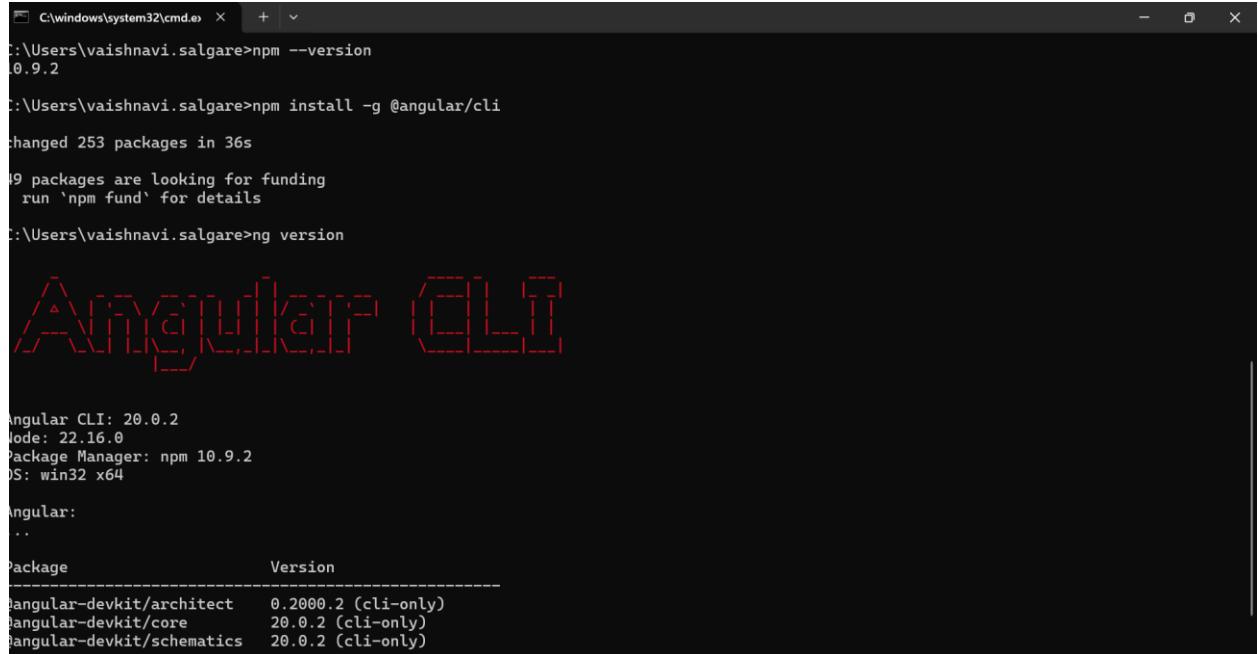
npm install -g @angular/cli@16, #for angular install globally

ng new appname,

ng version,

ng generate component(ng g c component-name) --skip-tests,

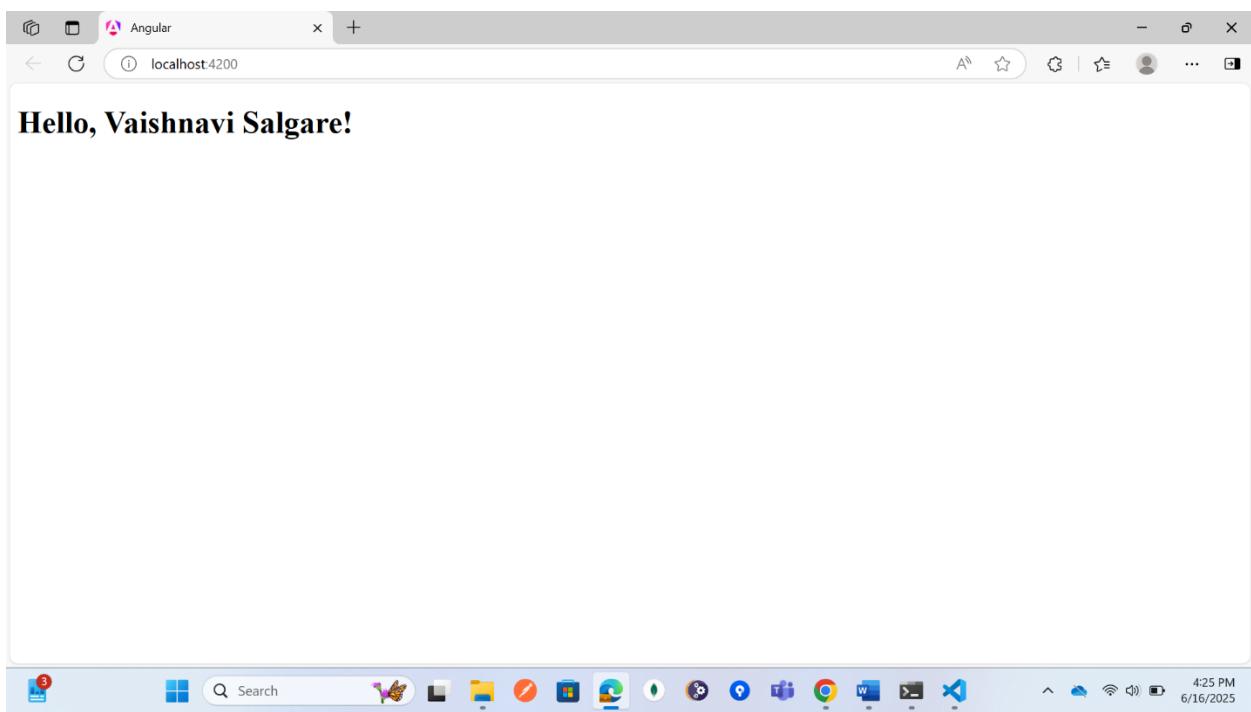
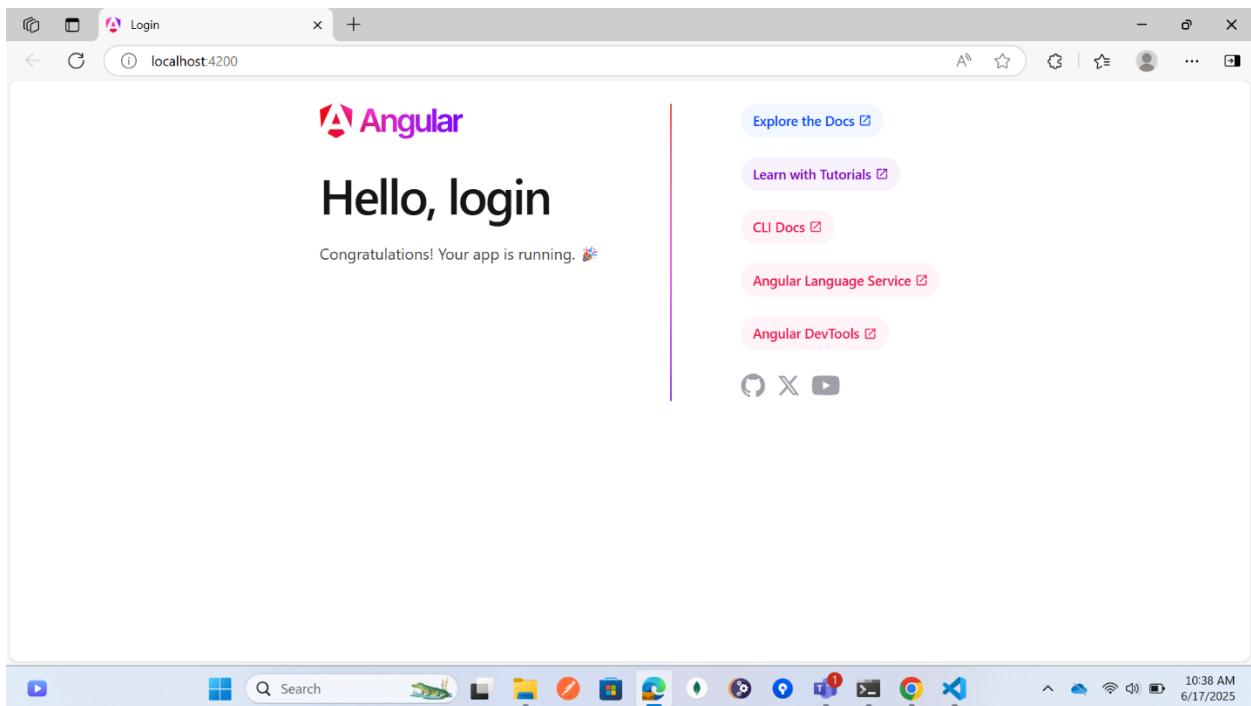
ng generate service(ng g s service-name)



```
C:\Windows\system32\cmd.exe + ^ 
C:\Users\vaishnavi.salgare>npm --version
10.9.2
C:\Users\vaishnavi.salgare>npm install -g @angular/cli
changed 253 packages in 36s
9 packages are looking for funding
  run `npm fund` for details
C:\Users\vaishnavi.salgare>ng version

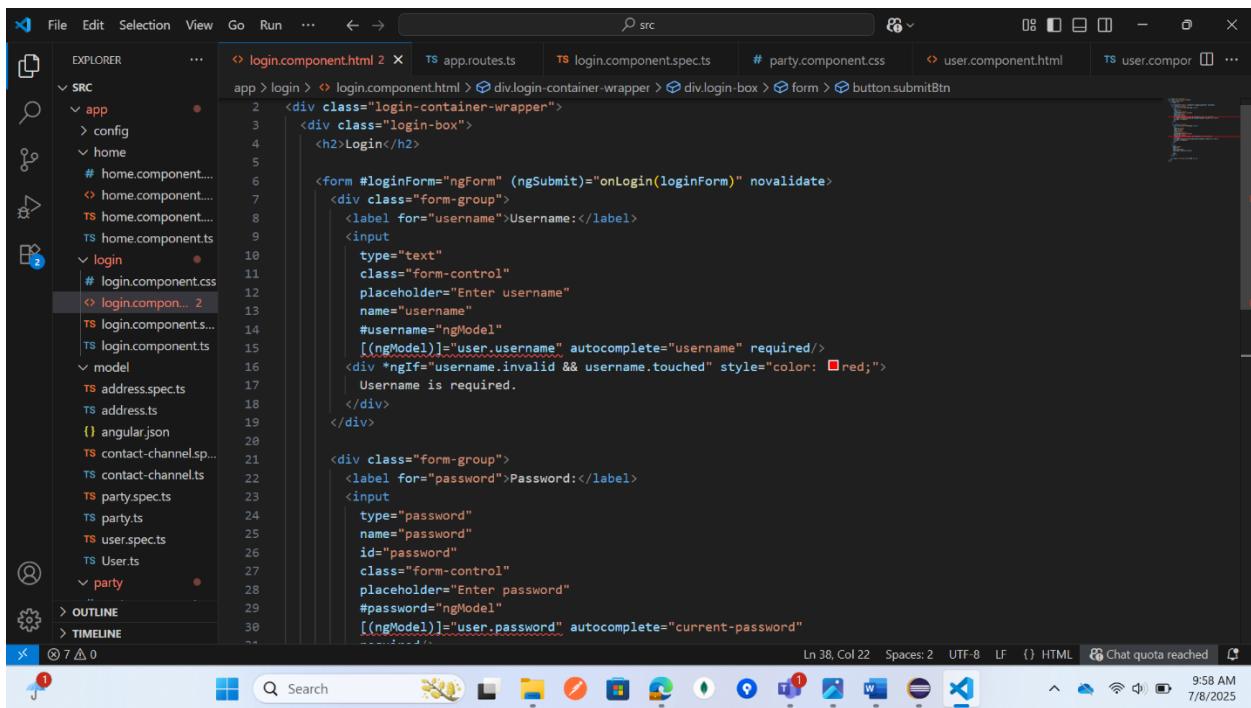
Angular CLI: 20.0.2
Node: 22.16.0
Package Manager: npm 10.9.2
OS: win32 x64

Angular:
...
Package          Version
-----
@angular-devkit/architect    0.2000.2 (cli-only)
@angular-devkit/core         20.0.2 (cli-only)
@angular-devkit/schematics   20.0.2 (cli-only)
```



- **Creating Frontend For the same:**

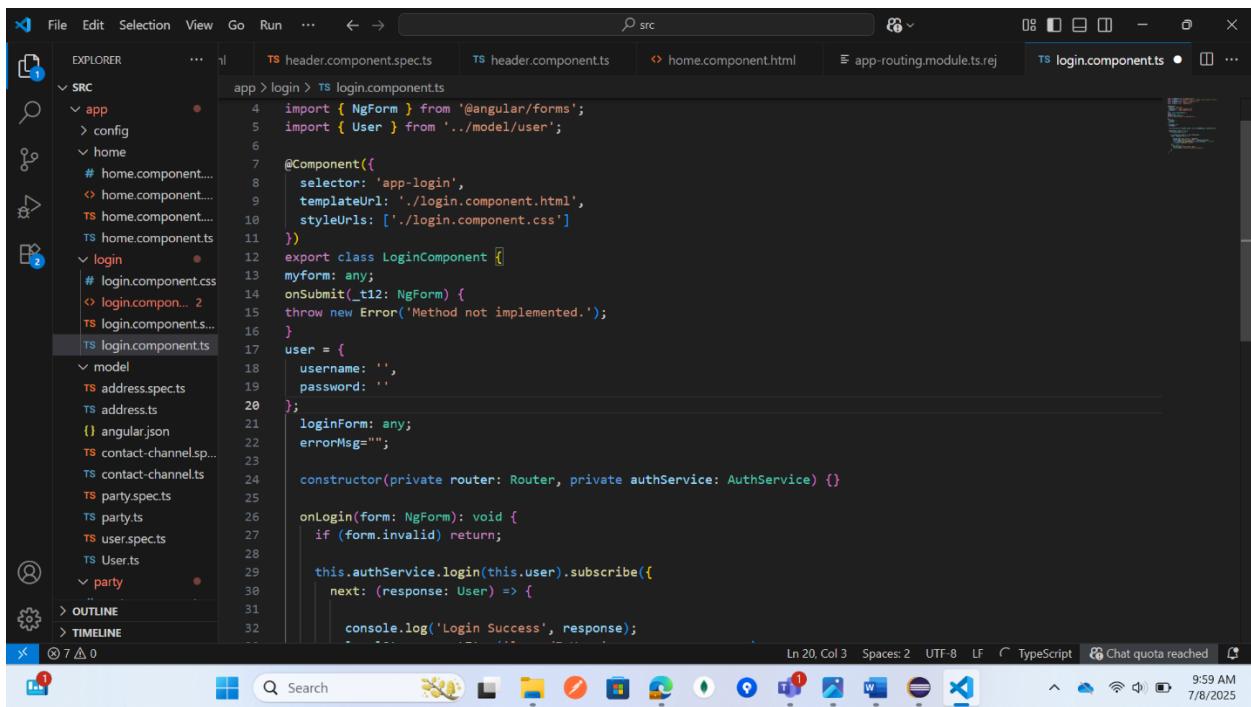
Login.component.html



```
2  <div class="login-container-wrapper">
3    <div class="login-box">
4      <h2>Login</h2>
5
6      <form #loginForm="ngForm" (ngSubmit)="onLogin(loginForm)" novalidate>
7        <div class="form-group">
8          <label for="username">Username:</label>
9          <input
10            type="text"
11            class="form-control"
12            placeholder="Enter username"
13            name="username"
14            #username="ngModel"
15            [(ngModel)]="user.username" autocomplete="username" required>
16          <div *ngIf="username.invalid && username.touched" style="color: red;">
17            Username is required.
18          </div>
19        </div>
20
21        <div class="form-group">
22          <label for="password">Password:</label>
23          <input
24            type="password"
25            name="password"
26            id="password"
27            class="form-control"
28            placeholder="Enter password"
29            #password="ngModel"
30            [(ngModel)]="user.password" autocomplete="current-password">
31        </div>
32    </form>

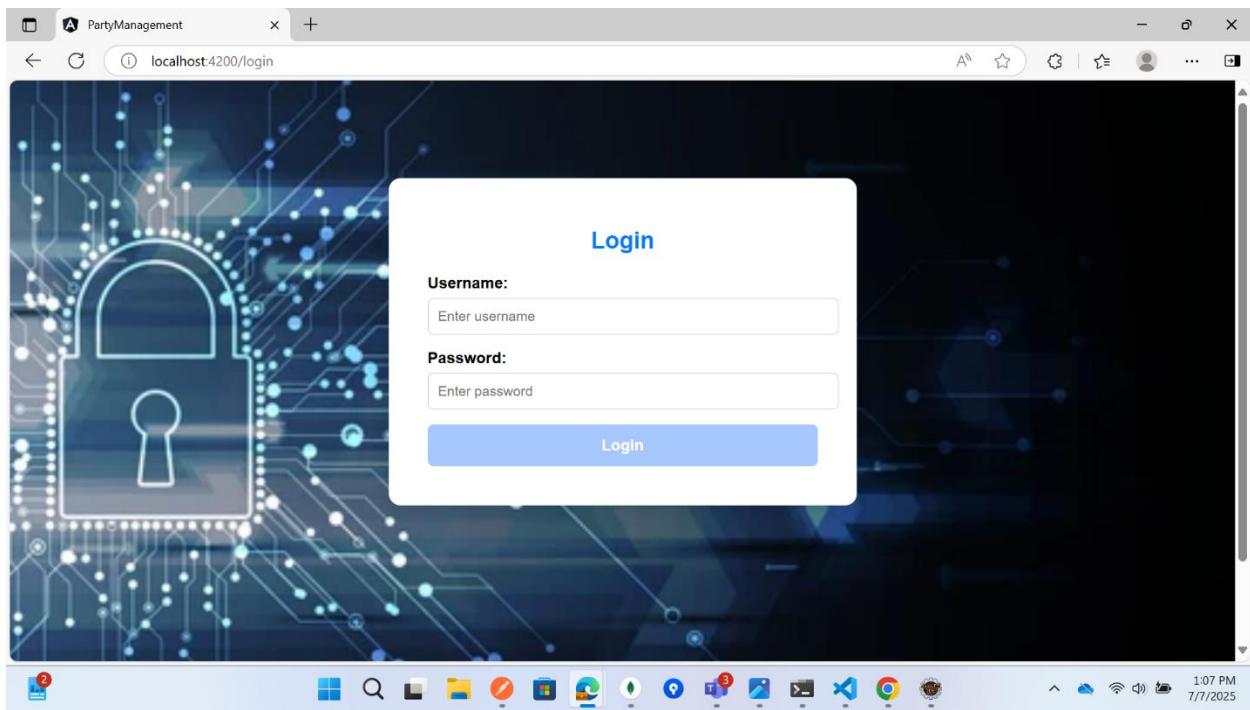
```

Login.component.ts



```
4  import { NgForm } from '@angular/forms';
5  import { User } from '../model/user';
6
7  @Component({
8    selector: 'app-login',
9    templateUrl: './login.component.html',
10   styleUrls: ['./login.component.css']
11 })
12 export class LoginComponent {
13   myForm: any;
14   onSubmit(_t12: NgForm) {
15     throw new Error('Method not implemented.');
16   }
17   user = {
18     username: '',
19     password: ''
20   };
21   loginForm: any;
22   errorMsg="";
23
24   constructor(private router: Router, private authService: AuthService) {}
25
26   onLogin(form: NgForm): void {
27     if (form.invalid) return;
28
29     this.authService.login(this.user).subscribe({
30       next: (response: User) => {
31         console.log('Login Success', response);
32       }
33     });
34   }
35 }

```



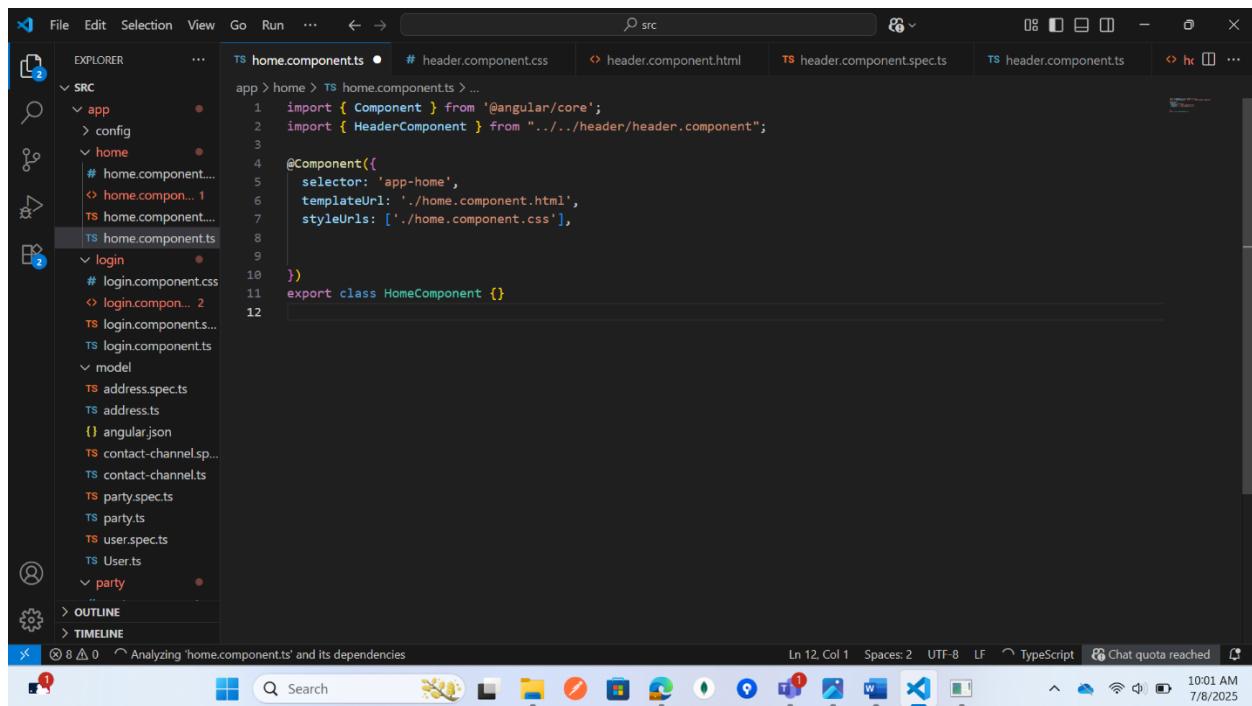
Home.component.html

A screenshot of the Visual Studio Code editor. The left sidebar shows the project structure under "SRC" with files like "header.component.spec.ts", "header.components.ts", "home.component.html", "login.component.ts", "model.ts", "party.ts", and "User.ts". The main editor tab is "home.component.html" containing the following code:

```
<h2>Welcome to Party Management</h2>
<app-header></app-header>
<div class="main-container">
  <router-outlet> </router-outlet>
</div>
```

The status bar at the bottom indicates "Analyzing 'home.component.ts' and its dependencies".

Home.component.ts

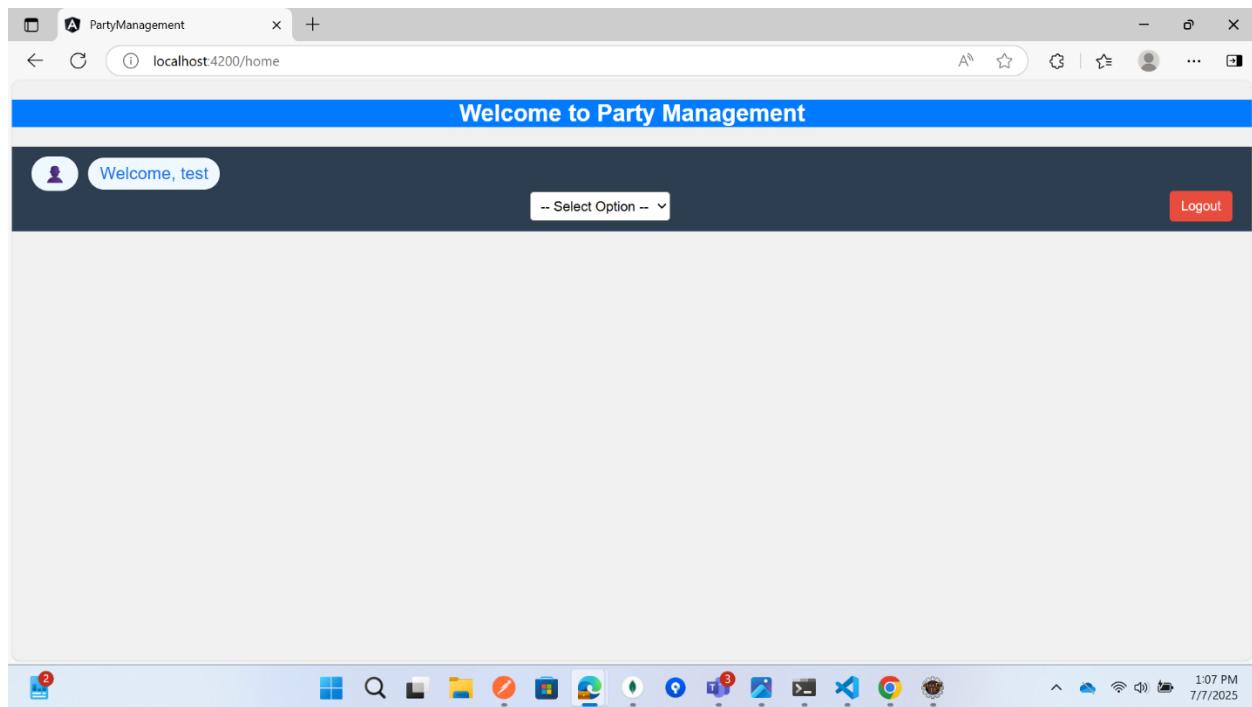


```
File Edit Selection View Go Run ... < > src File Explorer ... SRC app config home # home.component... < home.compon... 1 TS home.component... TS home.components... TS home.components.ts login # login.component.css < login.compon... 2 TS login.components... TS login.component.ts model TS address.spec.ts TS address.ts {} angular.json TS contact-channel.sp... TS contact-channel.ts TS party.spec.ts TS party.ts TS user.spec.ts TS User.ts party < ... > OUTLINE > TIMELINE
```

```
TS home.component.ts • # header.component.css header.component.html TS header.component.spec.ts TS header.component.ts
```

```
app > home > TS home.component.ts > ...
1 import { Component } from '@angular/core';
2 import { HeaderComponent } from "../../header/header.component";
3
4 @Component({
5   selector: 'app-home',
6   templateUrl: './home.component.html',
7   styleUrls: ['./home.component.css'],
8
9 })
10 export class HomeComponent {}
```

Ln 12, Col 1 Spaces: 2 UTF-8 LF TypeScript Chat quota reached 10:01 AM 7/8/2025



Header.component.html

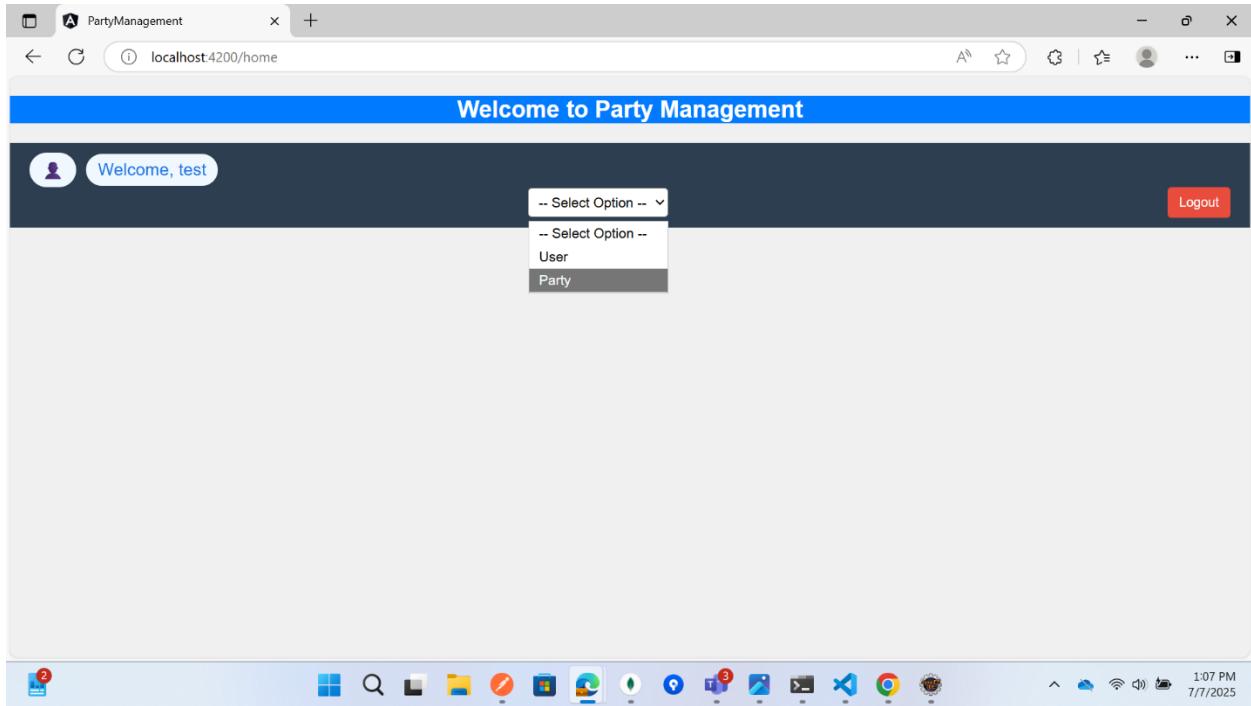
The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer (left):** Shows the project structure under the `SRC` folder, including `app`, `header`, `party`, `popup`, `services`, and various component and service files.
- Editor (center):** Displays the `header.component.html` file content. The code defines a header component with a welcome message, a logo, a navigation bar with dropdown options, and an authentication section with a logout button.
- Bottom Status Bar:** Shows the current line (Ln 12, Col 11), spaces used (Spaces: 2), encoding (UTF-8), line endings (LF), and the file type (HTML). It also indicates a "Chat quota reached" notification.

Header.component.ts

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer (left):** Shows the project structure under the `SRC` folder, including `app`, `header`, `party`, `popup`, `services`, and various component and service files.
- Editor (center):** Displays the `header.component.ts` file content. The code defines a `HeaderComponent` class that implements `OnInit`. It uses Angular's `Router` to handle navigation based on user selection and logout events.
- Bottom Status Bar:** Shows the current line (Ln 3, Col 1), spaces used (Spaces: 2), encoding (UTF-8), line endings (LF), and the file type (TypeScript). It also indicates a "Chat quota reached" notification.



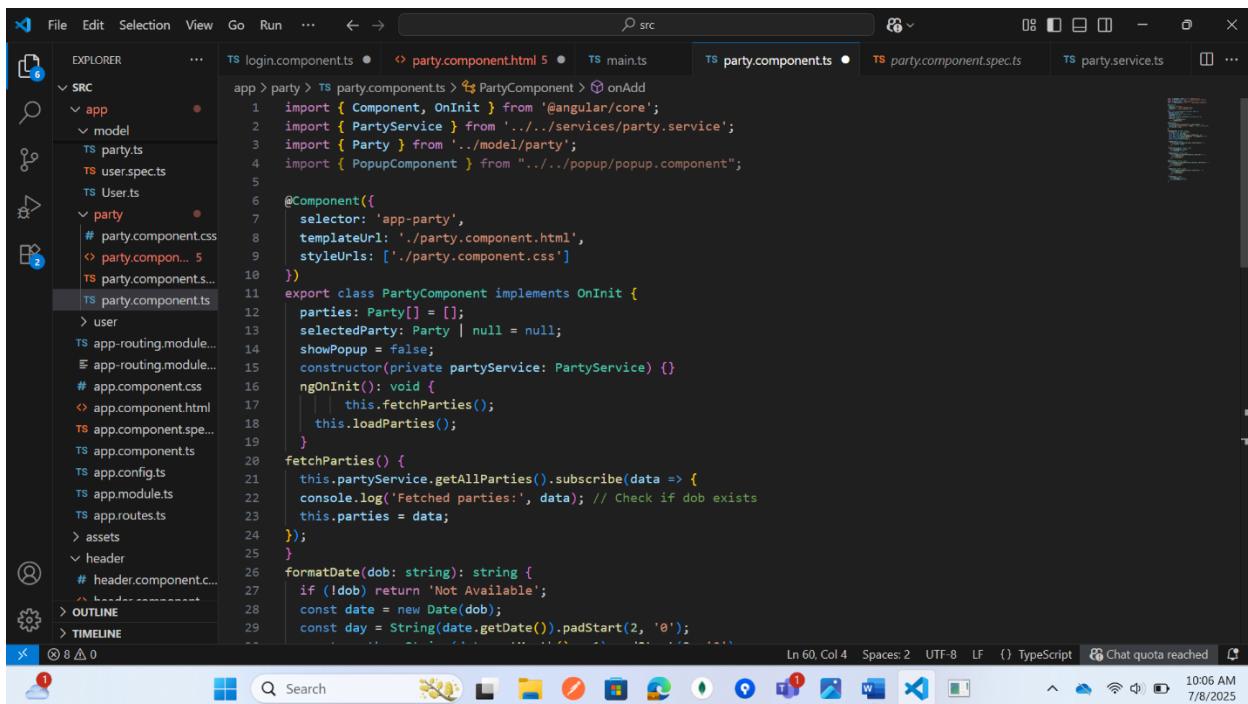
Part.component.html

```
<h2>Party List</h2>


| Party ID       | Party ID            | First Name            | Last Name            | DOB                                          | Gender                     | Occupation             | Address                                                                          | Contact                                                                           | Action |
|----------------|---------------------|-----------------------|----------------------|----------------------------------------------|----------------------------|------------------------|----------------------------------------------------------------------------------|-----------------------------------------------------------------------------------|--------|
| {{ party.id }} | {{ party.partyId }} | {{ party.firstName }} | {{ party.lastName }} | {{ party.dateOfBirth   date: 'dd/MM/yyyy' }} | {{ party.genderIdentity }} | {{ party.occupation }} | {{ party.address.city }}, {{ party.address.state }}, {{ party.address.country }} | {{ party.contactChannel.phoneNumber }}<br>{{ party.contactChannel.emailAddress }} |        |

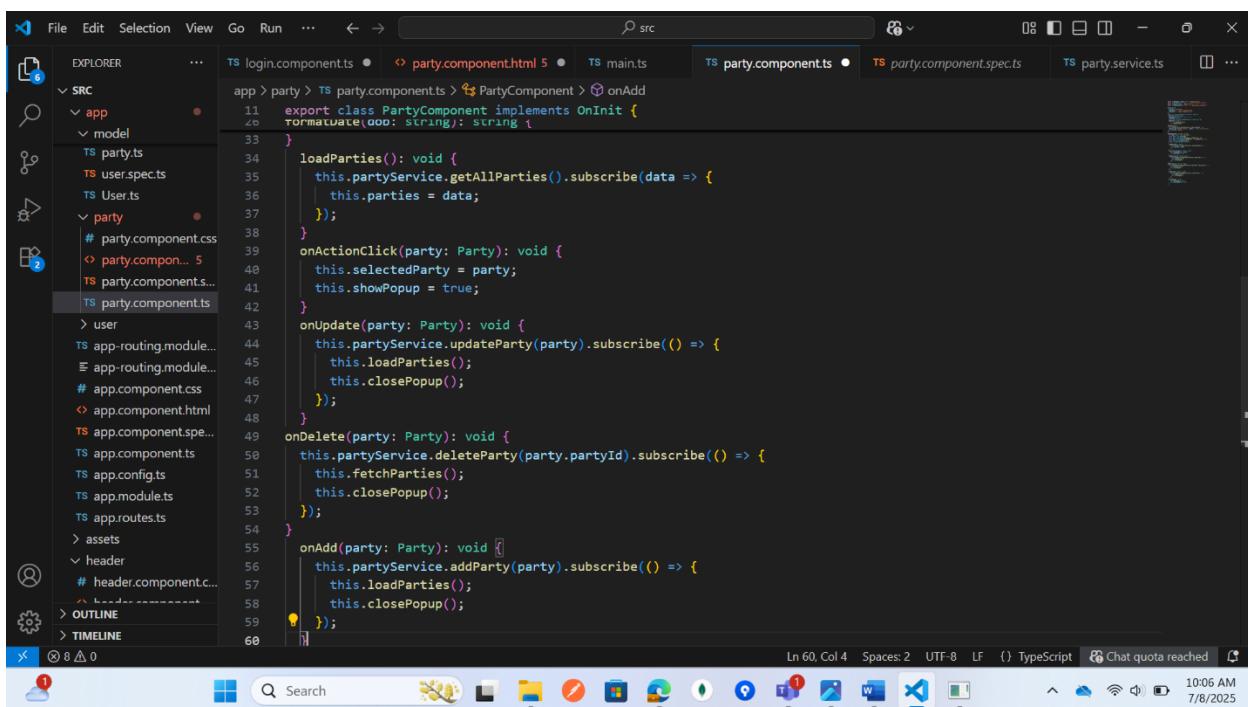

```

Part.component.ts



```
app > party > TS party.component.ts > PartyComponent > onAdd
1 import { Component, OnInit } from '@angular/core';
2 import { PartyService } from '../../../../../services/party.service';
3 import { Party } from '../../../../../model/party';
4 import { PopupComponent } from '../../../../../popup/popup.component';

5
6 @Component({
7   selector: 'app-party',
8   templateUrl: './party.component.html',
9   styleUrls: ['./party.component.css']
10 })
11 export class PartyComponent implements OnInit {
12   parties: Party[] = [];
13   selectedParty: Party | null = null;
14   showPopup = false;
15   constructor(private partyService: PartyService) {}
16   ngOnInit(): void {
17     this.fetchParties();
18     this.loadParties();
19   }
20   fetchParties() {
21     this.partyService.getAllParties().subscribe(data => {
22       console.log('Fetched parties:', data); // Check if dob exists
23       this.parties = data;
24     });
25   }
26   formatDate(dob: string): string {
27     if (!dob) return 'Not Available';
28     const date = new Date(dob);
29     const day = String(date.getDate()).padStart(2, '0');
30     const month = String(date.getMonth() + 1).padStart(2, '0');
31     const year = date.getFullYear();
32     return `${day}/${month}/${year}`;
33   }
34   loadParties(): void {
35     this.partyService.getAllParties().subscribe(data => {
36       this.parties = data;
37     });
38   }
39   onActionClick(party: Party): void {
40     this.selectedParty = party;
41     this.showPopup = true;
42   }
43   onUpdate(party: Party): void {
44     this.partyService.updateParty(party).subscribe(() => {
45       this.loadParties();
46       this.closePopup();
47     });
48   }
49   onDelete(party: Party): void {
50     this.partyService.deleteParty(party.partyId).subscribe(() => {
51       this.fetchParties();
52       this.closePopup();
53     });
54   }
55   onAdd(party: Party): void {
56     this.partyService.addParty(party).subscribe(() => {
57       this.loadParties();
58       this.closePopup();
59     });
60 }
```



```
app > party > TS party.component.ts > PartyComponent > onAdd
11 export class PartyComponent implements OnInit {
12   constructor(private partyService: PartyService) {}
13   ngOnInit(): void {
14     this.fetchParties();
15   }
16   fetchParties(): void {
17     this.partyService.getAllParties().subscribe(data => {
18       this.parties = data;
19     });
20   }
21   onActionClick(party: Party): void {
22     this.selectedParty = party;
23     this.showPopup = true;
24   }
25   onUpdate(party: Party): void {
26     this.partyService.updateParty(party).subscribe(() => {
27       this.loadParties();
28       this.closePopup();
29     });
30   }
31   onDelete(party: Party): void {
32     this.partyService.deleteParty(party.partyId).subscribe(() => {
33       this.fetchParties();
34       this.closePopup();
35     });
36   }
37   onAdd(party: Party): void {
38     this.partyService.addParty(party).subscribe(() => {
39       this.loadParties();
40       this.closePopup();
41     });
42 }
```

Party.service.ts

```

services > TS party.service.ts > PartyService > deleteParty
1 import { Injectable } from '@angular/core';
2 import { HttpClient } from '@angular/common/http';
3 import { Observable } from 'rxjs';
4 import { Party } from '../app/model/party';
5
6 @Injectable({
7   providedIn: 'root'
8 })
9 export class PartyService {
10   private baseUrl = 'http://localhost:8080/party';
11
12   constructor(private http: HttpClient) {}
13
14   // GET all parties
15   getAllParties(): Observable<Party[]> {
16     return this.http.get<Party[]>(this.baseUrl);
17   }
18   updateParty(party: Party): Observable<any> {
19     return this.http.put(`${this.baseUrl}`, party); // ✅ No ID in URL
20   }
21   addParty(party: Party): Observable<any> {
22     return this.http.post(this.baseUrl, party);
23   }
24   deleteParty(partyId: any): Observable<any> {
25     return this.http.delete(`${this.baseUrl}/${partyId}`);
26   }
27 }

```

| Party ID | First Name | Last Name | DOB | Gender | Occupation | Address | Contact | Action |
|----------|------------|-----------|------------|--------|--------------|------------------------------|------------------------------|--------------------------|
| 3 | vijaya | sagare | 11/11/1111 | Female | Student | latur, Madhyapradesh, India | 000 PH1@gmail.com | <button>Actions</button> |
| 2 | omkar | patil | 12/12/1212 | Male | st | Mumbai, Madhyapradesh, India | 1212121212 abc1@gmail.com | <button>Actions</button> |
| 3 | vaishnavi | salgare | 06/12/2004 | Female | intern | Pune, Maharashtra, India | 7979797979 VS1@gmail.com | <button>Actions</button> |
| 4 | John | Deo | 11/01/2011 | Male | Developer | Mumbai, Maharashtra, India | 1234567809 JJ11@gmail.com | <button>Actions</button> |
| 5 | Leo | D | 12/02/1900 | Male | Business | NewYork, New York, NewYork | 9999999999 NY01@gmail.com | <button>Actions</button> |
| 6 | Los | Angeles | 12/02/1996 | Male | entrepreneur | Delhi, Delhi, India | 3216540987 DD@gmail.com | <button>Actions</button> |

User.component.html

The screenshot shows the VS Code interface with the 'user.component.html' file open in the editor. The code displays an HTML template for a user list:

```
<app> user > <user.component.html> <div.container> <table border="1" cellpadding="8" cellspacing="0">
  <thead>
    <tr>
      <th>Username</th>
      <th>Password</th>
    </tr>
  </thead>
  <tbody>
    <tr *ngFor="let user of users">
      <td>{{ user.username }}</td>
      <td>{{ user.password }}</td>
    </tr>
  </tbody>
</table>
```

User.component.ts

The screenshot shows the VS Code interface with the 'user.component.ts' file open in the editor. The code defines a UserComponent class that implements OnInit:

```
export class UserComponent implements OnInit {
  users: User[] = [];
  newUser: User = {
    username: '',
    password: ''
  };
  constructor(private userService: UserService) {}
  ngOnInit(): void {
    this.loadUsers();
  }
  loadUsers(): void {
    this.userService.getAllUsers().subscribe({
      next: (data) => (this.users = data),
      error: (err) => console.error('Failed to load users', err)
    });
  }
  saveUser(): void {
    this.userService.createUser(this.newUser).subscribe({
      next: () => {
        this.newUser = { username: '', password: '' };
        this.loadUsers();
      },
      error: (err) => console.error('Failed to save user', err)
    });
  }
  deleteUser(id: number): void {
    if (confirm('Are you sure to delete this user?')) {
      this.userService.deleteUser(id).subscribe({
        next: () => this.loadUsers(),
        error: (err) => console.error('Failed to delete user', err)
      });
    }
  }
}
```

User.service.ts

The screenshot shows the Visual Studio Code interface. The left sidebar displays the file structure under 'SRC'. The main editor area shows the code for 'user.service.ts'.

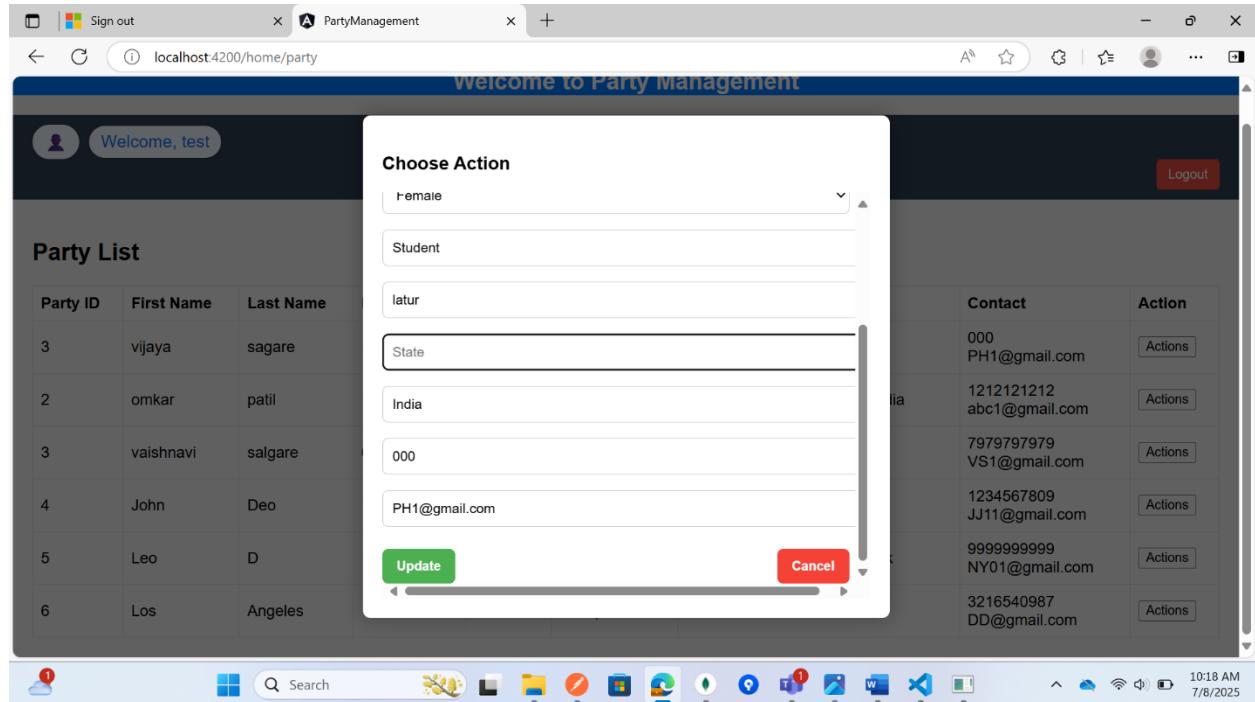
```
services > user.service.ts > UserService > getAllUsers
2   import { HttpClient } from '@angular/common/http';
3   import { Observable } from 'rxjs';
4
5   import { User } from '../app/model/user';
6
7   @Injectable({
8     providedIn: 'root'
9   })
10  export class UserService {
11    private apiUrl = 'http://localhost:8080/login/users'; // Spring Boot backend
12
13    constructor(private http: HttpClient) {}
14
15    getAllUsers(): Observable<User[]> {
16      return this.http.get<User[]>(this.apiUrl);
17    }
18
19    createUser(user: User): Observable<User> {
20      return this.http.post<User>(this.apiUrl, user);
21    }
22
23    deleteUser(id: number): Observable<void> {
24      return this.http.delete<void>(`${this.apiUrl}/${id}`);
25    }
26
27    validateUser(username: string, password: string): Observable<User> {
28      return this.http.post<User>(`${this.apiUrl}/validate`, { username, password });
29    }
30 }
```

The status bar at the bottom indicates 'Ln 15, Col 38 Spaces: 2 UTF-8 LF () TypeScript Chat quota reached' and the system tray shows the date and time as '10:11 AM 7/8/2025'.

The screenshot shows a Microsoft Edge browser window titled 'PartyManagement'. The address bar shows 'localhost:4200/home/user'. The page title is 'Welcome to Party Management'. The top navigation bar includes a user icon, a 'Welcome, test' message, a dropdown menu set to 'User', and a 'Logout' button. The main content area is titled 'User List' and displays a table of user data:

| Username | Password |
|----------|----------|
| test | test |
| sample | sample |
| abc | abc |

The system tray at the bottom shows the date and time as '10:07 PM 7/7/2025'.



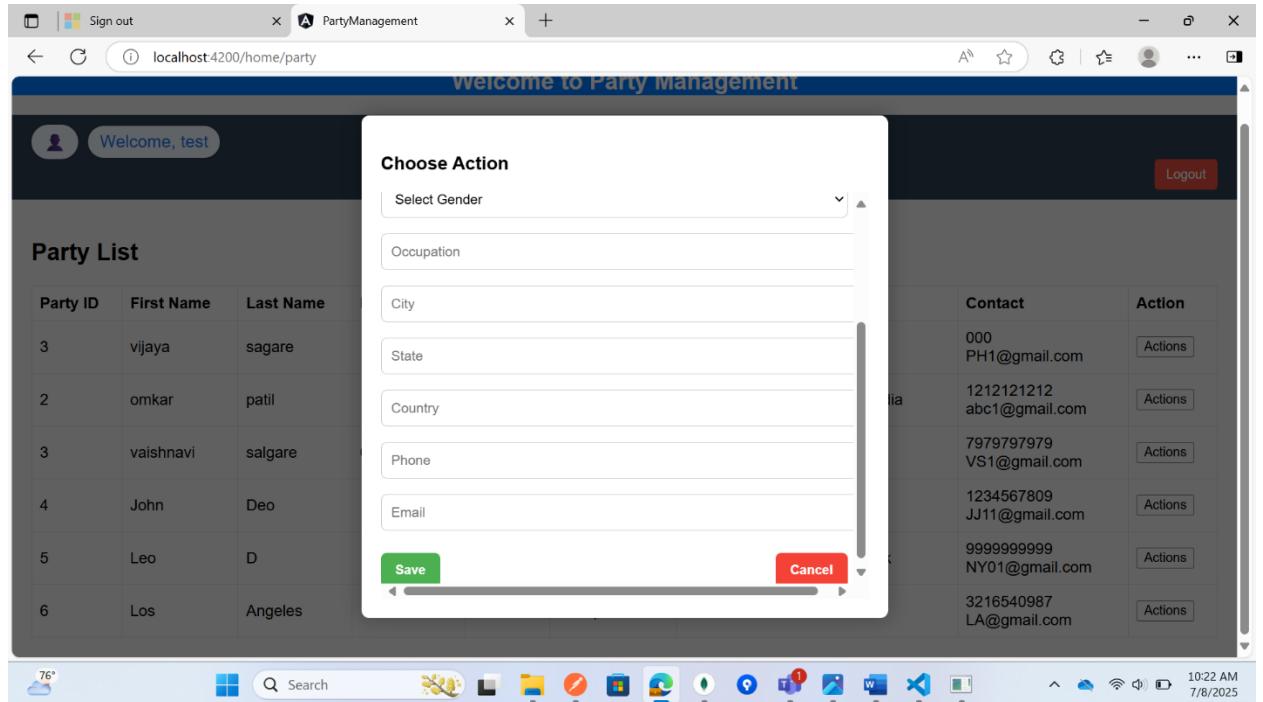
The screenshot shows the MongoDB Compass interface connected to the "local db/testDB.party" database. On the left, the "Connections" sidebar lists "local db", "localhost:27017", and "testDB". The "testDB" connection is expanded, showing the "party" collection. The main pane displays the "Documents" tab with 6 results. A search bar at the top allows querying documents. Below the search bar are buttons for "ADD DATA", "EXPORT DATA", "UPDATE", and "DELETE". The results list shows two document snippets:

```
firstName : "Leo"
lastName : "D"
dateOfBirth : "1980-02-12"
address : Object
contactChannel : Object
genderIdentity : "Male"
partyId : "5"
occupation : "Business"
__class : "com.party.dashboard.model.Party"
```



```
_id: "686b77211da2680d3ee8ed7d"
firstName : "Los"
lastName : "Angeles"
dateOfBirth : "1996-02-12"
address : Object
contactChannel : Object
genderIdentity : "Male"
partyId : "6"
occupation : "entrepreneur"
__class : "com.party.dashboard.model.Party"
```

- To Add New Party:



- **Angular Technologies:**

1. Two-Way Data Binding
2. Dependency Injection (DI)
3. Routing & Navigation
4. Services & HTTP Client
5. Pipes