1. Overview General

Funcționalitatea **Forgot Password** permite utilizatorilor care și-au uitat parola să o reseteze. Acest proces implică de obicei solicitarea unei noi parole și actualizarea acesteia în baza de date.

2. Flow-ul Codului pentru Forgot Password

Să vedem cum funcționează fiecare parte din cod:

Frontend (Angular)

În frontend, interfața colectează noua parolă de la utilizator și o trimite la server pentru a fi actualizată.

- **forgot-password.component.html**: Este fișierul care definește structura formularului pentru resetarea parolei. Conține câmpuri pentru introducerea adresei de email (sau numelui de utilizator), a noii parole și pentru confirmarea acesteia.

html

Copy code

```
<form [formGroup]="forgotPasswordForm">
  <div class="forgot-password-form-flex">
    <mat-card>
      <mat-card-title class="title mb-3">Forgot Password</mat-card-title>
      <mat-card-content class="d-flex flex-column gap-2">
        <mat-form-field>
          <input type="text" matInput placeholder="Email" formControlName="username">
        </mat-form-field>
        <mat-form-field>
          <input type="password" matInput placeholder="New Password"
formControlName="newPassword">
        </mat-form-field>
        <mat-form-field>
          <input type="password" matInput placeholder="Confirm New Password"
formControlName="confirmNewPassword">
        </mat-form-field>
        <div *ngIf="showMessage" class="text-danger">
          Passwords are not the same.
```

```
      </div>
    </mat-card-content>
    <mat-card-actions>
      <div class="button-flex-container">
        <button mat-raised-button color="primary" (click)="resetPassword()">Reset Password</button>
      </div>
    </mat-card-actions>
  </mat-card>
 </div>
</form>
```

- **forgot-password.component.ts**: Acesta este fișierul TypeScript care gestionează logica componentului de resetare a parolei.
    - forgotPasswordForm: Este un FormGroup care definește structura formularului și regulile de validare.
    - resetPassword(): Această metodă este apelată atunci când utilizatorul apasă pe butonul "Reset Password". Verifică dacă parolele introduse se potrivesc și trimite cererea de resetare a parolei către server.

typescript

Copy code

```
import { Component } from '@angular/core';
import { FormControl, FormGroup, Validators } from '@angular/forms';
import { AuthenticationService } from '../../../services/authentication.service';
import { MatSnackBar } from '@angular/material/snack-bar';
import { Router } from '@angular/router';

@Component({
 selector: 'app-forgot-password',
 templateUrl: './forgot-password.component.html',
 styleUrls: ['./forgot-password.component.scss']
})
```

```
export class ForgotPasswordComponent {

  protected showMessage: boolean = false;

  constructor(private authenticationService: AuthenticationService, private snackBar: MatSnackBar,
private router: Router) {}

  protected forgotPasswordForm: FormGroup = new FormGroup({
    username: new FormControl('', Validators.required),
    newPassword: new FormControl('', Validators.required),
    confirmNewPassword: new FormControl('', Validators.required)
  });

  public resetPassword() {
    this.forgotPasswordForm.markAllAsTouched();

    if (this.forgotPasswordForm.invalid) {
      return;
    }

    let username = this.forgotPasswordForm.value.username;
    let newPassword = this.forgotPasswordForm.value.newPassword;
    let confirmNewPassword = this.forgotPasswordForm.value.confirmNewPassword;

    if (newPassword !== confirmNewPassword) {
      this.showMessage = true;
      return;
    }
```

```
  let credentialsModel = {
    username: username,
    password: newPassword,
    newPassword: confirmNewPassword,
  };

  this.authenticationService.updatePassword(credentialsModel).subscribe({
    next: (res: any) => {
      this.router.navigate(['/login']);
      this.openSuccessSnackBar(res.message);
    },
    error: (err: any) => {
      console.log(err);
      this.openFailureSnackBar(err.error);
    }
  });
}

public openSuccessSnackBar(message: string) {
  this.snackBar.open(message, "OK", {
    duration: 3000,
    panelClass: ['green-snackbar', 'register-snackbar'],
  });
}

public openFailureSnackBar(message: string) {
  this.snackBar.open(message, "Try again!", {
    duration: 3000,
    panelClass: ['red-snackbar', 'register-snackbar'],
```

```
  });
 }
}
```

Backend (ASP.NET Core)

Partea de backend primește cererea de resetare a parolei, verifică dacă utilizatorul există, și actualizează parola în baza de date.

- **AuthenticationController.cs**: Acest controller conține logica pentru gestionarea cererilor de autentificare și resetare a parolei.
    - Metoda UpdatePassword() primește datele de la frontend, caută utilizatorul în baza de date și actualizează parola acestuia.

csharp

Copy code

```
[Route("api/[controller]/[action]")]
[ApiController]
public class AuthenticationController : ControllerBase
{
  private AppContext _appContext;

  public AuthenticationController(AppContext appContext)
  {
    _appContext = appContext;
  }

  [HttpPost(Name = "UpdatePassword")]
  public ActionResult UpdatePassword(UpdateCredentialsModel credentialsModel)
  {
    try
    {
      User user = _appContext.Users.FirstOrDefault(x => x.Username == credentialsModel.Username);
```

```
            if (user == null)
            {
                throw new ArgumentException("Cannot find user!");
            }

            user.Password = credentialsModel.Password;
            _appContext.SaveChanges();
        }
        catch (Exception e)
        {
            return StatusCode(StatusCodes.Status500InternalServerError, e.Message);
        }

        return Ok(new
        {
            Message = "Password was successfully changed! You can log in now."
        });
    }
}
```

- **UpdateCredentialsModel.cs**: Acesta este modelul care definește structura datelor necesare pentru a actualiza parola utilizatorului. Extinde CredentialsModel, adăugând un câmp pentru noua parolă.

```
csharp
Copy code
public class UpdateCredentialsModel : CredentialsModel
{
    public string? NewPassword { get; set; }
}
```

- **User.cs**: Acesta este modelul de date care reprezintă un utilizator în baza de date, similar cu cel utilizat pentru înregistrare.

csharp

Copy code

```
public class User
{
    public int Id { get; set; }
    public string Username { get; set; }
    public string Password { get; set; }
}
```

Rezumat

1. **Frontend**:
    - Utilizatorul introduce adresa de email/numele de utilizator, noua parolă și confirmarea parolei.
    - Când utilizatorul apasă pe "Reset Password", datele sunt validate și trimise la server pentru actualizare.
2. **Backend**:
    - Serverul primește cererea de resetare a parolei.
    - Verifică dacă utilizatorul există în baza de date.
    - Dacă utilizatorul există, parola este actualizată și modificările sunt salvate în baza de date.
    - Serverul returnează un mesaj de succes sau de eroare către frontend.
3. **Feedback pentru Utilizator**:
    - Utilizatorul primește un mesaj de confirmare a resetării parolei sau un mesaj de eroare în caz de probleme.