t

هوش مصنوعی سخن روان



پروژه وب­سایت ضبط و تحلیل صدای ضبط شده کاربر و شناسایی لکنت زبان

تحت نظارت آزمایشگاه تجزیه و تحلیل سیگنال و گفتار دانشگاه خواجه نصیرالدین طوسی

Contents

[بخــــــش اول – مقدمه 4](#_Toc160354750)

[1‏‏-‏‏1 ویژگیها 4](#_Toc160354751)

[1‏‏-‏‏2 هدف 5](#_Toc160354752)

[بخــــش دوم – ابزار Wavesurfer 6](#_Toc160354753)

[2‏‏-‏‏1 نحوه کار با پلاگین 6](#_Toc160354755)

[2‏‏-‏‏2 نمایش سیگنال 7](#_Toc160354756)

[2‏‏-‏‏3 ضبط سیگنال صوتی 9](#_Toc160354757)

[2‏‏-‏‏4 بزرگنمایی سیگنال 12](#_Toc160354758)

[2‏‏-‏‏5 افزودن جدول زمانی به سیگنال 13](#_Toc160354759)

[2‏‏-‏‏6 افزودن تایمر پخش 15](#_Toc160354760)

[2‏‏-‏‏7 تغییر سرعت پخش سیگنال 16](#_Toc160354761)

[2‏‏-‏‏8 ایجاد نواحی بر روی سیگنال 17](#_Toc160354762)

[2‏‏-‏‏9 ایجاد حلقه برای پخش مکرر بر روی نواحی 18](#_Toc160354763)

[2‏‏-‏‏10 بارگذاری فایل صوتی ضبط شده 19](#_Toc160354764)

[بخــــش سوم – ایجاد پروژه جنگو 20](#_Toc160354765)

[3‏‏-‏‏1 مقدمه 20](#_Toc160354767)

[3‏‏-‏‏2 ایجاد ساختار پروژه 20](#_Toc160354768)

[3‏‏-‏‏3 ایجاد مدلها ClientApp Models 21](#_Toc160354769)

[3‏‏-‏‏3‏‏-‏‏1 ایجاد مدل کاربری سفارشی 21](#_Toc160354770)

[3‏‏-‏‏3‏‏-‏‏2 ایجاد کلاس Manager سفارشی 23](#_Toc160354771)

[3‏‏-‏‏4 ایجاد سیگنالها ClientApp Signlas 24](#_Toc160354772)

[3‏‏-‏‏5 پیادهسازی ارسال کد اعتبارسنجی 25](#_Toc160354773)

[3‏‏-‏‏6 ایجاد تابعهای ClientApp Views 26](#_Toc160354774)

[3‏‏-‏‏6‏‏-‏‏1 ثبتنام کاربر و ارسال کد اعتبار سنجی: 26](#_Toc160354775)

[3‏‏-‏‏6‏‏-‏‏2 بررسی کد اعتبارسنجی وارد شده توسط کاربر: 27](#_Toc160354776)

[3‏‏-‏‏6‏‏-‏‏3 بروزرسانی اطلاعات پروفایل کاربری: 28](#_Toc160354777)

[3‏‏-‏‏6‏‏-‏‏4 ورود به سامانه: 28](#_Toc160354778)

[3‏‏-‏‏6‏‏-‏‏5 خروج از سیستم: 29](#_Toc160354779)

[3‏‏-‏‏6‏‏-‏‏6 فراموشی و تغییر رمز عبور 29](#_Toc160354780)

[3‏‏-‏‏7 ایجاد مسیرهای ClientApp URL 31](#_Toc160354781)

[3‏‏-‏‏8 ایجاد مدلهای CoreApp Models 31](#_Toc160354782)

[3‏‏-‏‏9 ایجاد تابعهای CoreApp Views 33](#_Toc160354783)

[3‏‏-‏‏10 ایجاد مسیرهای CoreApp URL 38](#_Toc160354784)

[3‏‏-‏‏11 ایجاد فانکشنهای Search و Pagination 39](#_Toc160354785)

[3‏‏-‏‏12 تنظیمات کلی پروژه 40](#_Toc160354786)

# بخــــــش اول – مقدمه

در این پروژه وب سایت، قصد داریم یک سامانه‌ای برای ضبط و تحلیل صداهای ضبط شده توسط کاربران ارائه دهیم. این سامانه قابلیت شناسایی لکنت زبان در صداها را نیز دارا خواهد بود. وب‌سایت سخن روان با استفاده از هوش مصنوعی یک پلتفرم آموزشی و تمرینی است که به افراد امکان می‌دهد تا صدای خود را ضبط کرده و از طریق تحلیل صدا با تکنیک‌های هوش مصنوعی، لکنت‌های گفتاری خود را تشخیص دهند. این وب‌سایت به کاربران این امکان را می‌دهد تا به تفکیک تشخیص دهند که آیا لکنت گفتاری آن‌ها از نوع کششی یا تکرار واژه‌ها است.

برای پیاده‌سازی این سامانه، از فریمورک Django به عنوان فریمورک پایتون برای توسعه وب استفاده خواهیم کرد.

یکی از ابزارهای کلیدی که در این پروژه استفاده خواهیم کرد، پلاگین جاواسکریپت Wavesurfer است. این پلاگین قدرتمند به ما امکان می‌دهد تا امکانات ضبط، پخش و تحلیل صدا را به سادگی در وب سایتمان اضافه کنیم.

با استفاده از Django و Wavesurfer، قصد داریم یک پلتفرم کاربرپسند و قدرتمند برای ضبط و تحلیل صداها ایجاد کنیم. این پروژه نه تنها به کاربران امکان می‌دهد تا صداهای خود را ضبط کنند و پخش کنند، بلکه با تحلیل دقیق صداها، می‌تواند به شناسایی و اصلاح لکنت زبان کمک کند.

با این پروژه، امیدواریم تا تجربه کاربری منحصر به فردی را برای کاربران خود فراهم کنیم و به بهبود مهارت‌های زبانی آن‌ها کمک کنیم.

## ویژگی­ها

* بازپخش گفتار:
  + پخش صوتی به کاربران امکان می‌دهد تا به صورت مکرر به صدای ضبط‌شده خود گوش دهند و این امکان را دارند که به راحتی ضبط‌های خود را مرور و ارزیابی کنند.
* تمرکز گرایی:
  + کاربران می‌توانند هر یک از ناتوانی های تکرار و یا کشش را به عنوان ناتوانی انتخابی خود برای مشاهده گزارش نمودار انتخاب کنند و تمرکز خود را به بر روی بهبود آن بگذارند.
* قدرت بزرگنمایی:
  + با قدرت بزرگنمایی، کاربران می‌توانند به آسانی نمایش صدای ضبط‌شده را تنظیم کرده و آن را به سبک ترجیحی خود به گسترده کنند
* نرخ بازپخش:
  + نرخ پخش، کاربران را قادر می‌سازد که سرعت پخش صدای ضبط‌شده خود را به طور انعطاف پذیر با امکان تغییر از نصف سرعت عادی تا سرعت پخش 4 برابری کنترل کنند
* قابلیت حلقه:
  + توانایی حلقه به کاربران اجازه می‌دهد که قسمت‌های شناخته‌شده ناتوانی را مجدداً پخش کنند، و کاربر تشویق به تجزیه و تحلیل دقیق برای ارزیابی متمرکز می‌شود
* تعداد ناتوانی:
  + قابلیت مشاهده تعداد ناتوانی های شناخته شده در صدای ضبط‌شده را به تفکیک فراهم می‌کند و به تحلیل الگوهای گفتار ارزشمندی می‌آورد
* گزارشات:
  + کاربران روند پیشرفت خود در رفع لکنت گفتاری را در یک گراف زمانی مشاهده می کنند.

## هدف

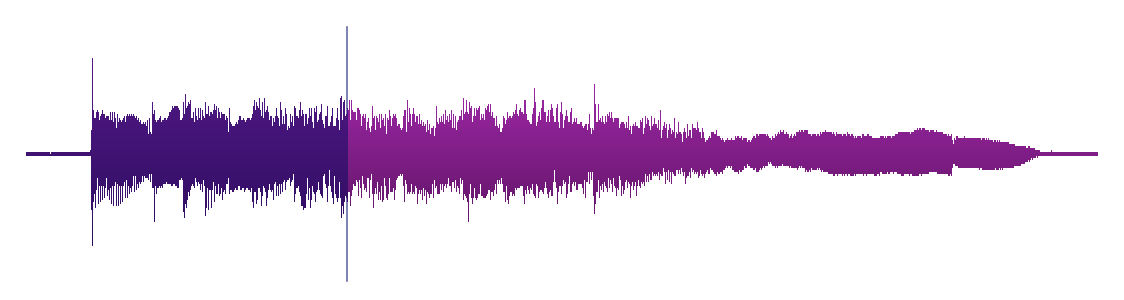
این وب‌سایت برای کمک به افراد در بهبود صدای گفتار و افزایش مهارت‌های زبانی طراحی شده است، به ویژه برای افرادی که با مشکلات لکنت گفتاری کششی یا تکراری روبرو هستند و کاربران را به تمرین مداوم و بهبود گفتارشان تشویق می‌کند. این سامانه توسط تجربه و هوش مصنوعی به افراد کمک می‌کند تا به یک صحبت روان‌تر و مهارت‌های ارتباطی بهتر دست یابند.

# بخــــش دوم – ابزار Wavesurfer



## نحوه کار با پلاگین

Wavesurfer.js یک پلاگین جاواسکریپت نمایش سیگنا­های صوتی با توانایی ایجاد شکل موج­های تعاملی و قابل تنظیم است.



برای شروع استفاده از پلاگین WaveSurfer.js ، می­بایست فایل های لازم پلاگین را از وب سایت رسمی دانلود کرده و یا آن را از طریق یک CDN (شبکه تحویل محتوا) در فایل HTML خود قرار داد.

<!DOCTYPE html>

<html>

<head>

<!-- Using Official CDN File -->

 <script src="https://unpkg.com/wavesurfer.js@7/dist/wavesurfer.min.js"></script>

<!-- Using Downloaded .js File -->

 <script src="js/wavesurfer/wavesurfer.min.js"></script>

</head>

<body>

</body>

<script type="module">

<!-- Using JavaScript Import Function from Official CDN File -->

 import WaveSurfer from 'https://unpkg.com/wavesurfer.js@7/dist/wavesurfer.esm.js'

<!-- Using JavaScript Import Function from Downloaded .js File -->

 import WaveSurfer from './js/wavesurfer/wavesurfer.esm.min.js'

</script>

</html>

## نمایش سیگنال

بعد از واردکردن فایل­های جاواسکریپت پلاگین، یک تگ <div> با شناسه "waveform" برای رندر کردن و نمایش سیگنال صوتی در فایل HTML خود ایجاد می­کنیم.

<!DOCTYPE html>

<html>

<head></head>

<body>

    <div id="waveform">

        <!-- the waveform will be rendered here -->

    </div>

</body>

<script type="module">

  import WaveSurfer from './js/wavesurfer/wavesurfer.esm.min.js'

</script>

</html>

نکته : برای استفاده از پلاگین حتما باید این div را تعریف به id مورد نظر خود را ارتباط دهیم.

حال برای نمایش سیگنال صوتی کافی­ست که متغیر WaveSurfer را به id ارتباط داده و فایل صوتی موردنظرمان را که قصد پردازش آن را داریم، با استفاده از قابلیت url موجود در پلاگین، مسیردهی کرده و فراخوانی کنیم. سایر تنظیمات، مربوط به ظاهر و شکل نمایش سیگنال می­باشد.

<!DOCTYPE html>

<html>

<head></head>

<body>

    <div id="waveform">

        <!-- the waveform will be rendered here -->

    </div>

</body>

<script type="module">

  import WaveSurfer from './js/wavesurfer/wavesurfer.esm.min.js'

*const* wavesurfer = WaveSurfer.create({

        container: '#waveform',

        waveColor: '#4F4A85',

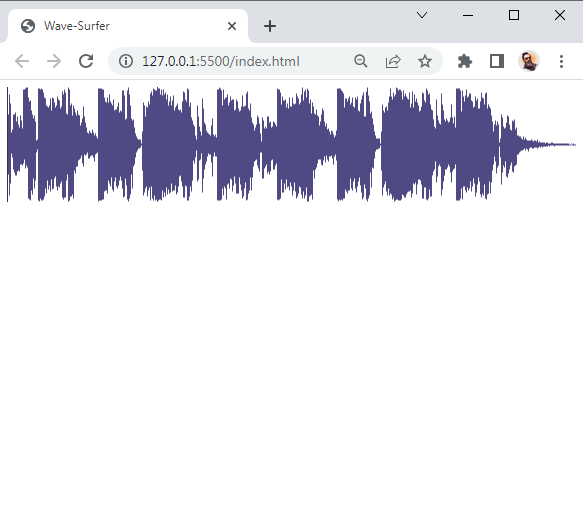
        progressColor: '#383351',

        url: './example.mp3',

    })

</script>

</html>

حال در مرورگر سیگنال صوتی، به نمایش درخواهد آمد

برای پخش سیگنال صوتی، نیاز به یک کد جاواسکریپت دیگر می­باشد که از قابلیت­های پلاگین بهره خواهیم گرفت.

<!DOCTYPE html>

<html>

<head></head>

<body>

    <div id="waveform">

        <!-- the waveform will be rendered here -->

    </div>

</body>

<script type="module">

  import WaveSurfer from './js/wavesurfer/wavesurfer.esm.min.js'

*const* wavesurfer = WaveSurfer.create({

        container: '#waveform',

        waveColor: '#4F4A85',

        progressColor: '#383351',

        url: './example.mp3',

    })

    wavesurfer.on('interaction', () *=>* {

        wavesurfer.play()

    })

</script>

</html>

## ضبط سیگنال صوتی

در ادامه، قصد داریم با استفاده از پلاگین یک سیگنال صوتی گفتاری را که توسط کاربر بیان می­شود را ضبط و سپس نمایش دهیم. برای این کار ابتدا یک تمپلیت جدید به نام recording.html ایجاد کرده و در اولین اقدام پلاگین را در فایل html خود قرار داده و در ادامه همانند مورد مشابه فوق یک تگ <div> با شناسه "waveform" برای رندر کردن و نمایش سیگنال صوتی در فایل HTML خود ایجاد می­کنیم.

<!DOCTYPE html>

<html>

<head></head>

<body>

    <div id="waveform">

        <!-- the waveform will be rendered here -->

    </div>

</body>

<script type="module">

  import WaveSurfer from './js/wavesurfer/wavesurfer.esm.min.js'

</script>

</html>

حال در این بخش، به تعریف کد HTML دکمه­های ضبط با شناسه record و پخش با شناسه play و همچنین استایل­دهی مختصر فضای نمایش سیگنال با شناسه waveform در فایل خود می­پردازیم :

نکته: می­توانیم کد HTML برای دانلود فایل سیگنال ضبط شده نیز با استفاده از تگ <a> به کد اضافه کنیم

<!DOCTYPE html>

<html>

<head></head>

<body>

    <button id="record">RECORD</button>

    <button id="play" disabled>PLAY</button>

<a id="audiolink" style="display: none">Download audio</a>

    <div id="waveform" style="border: 3px solid orangered; border-radius: 5px; padding: 50px;">

        <!-- the waveform will be rendered here -->

    </div>

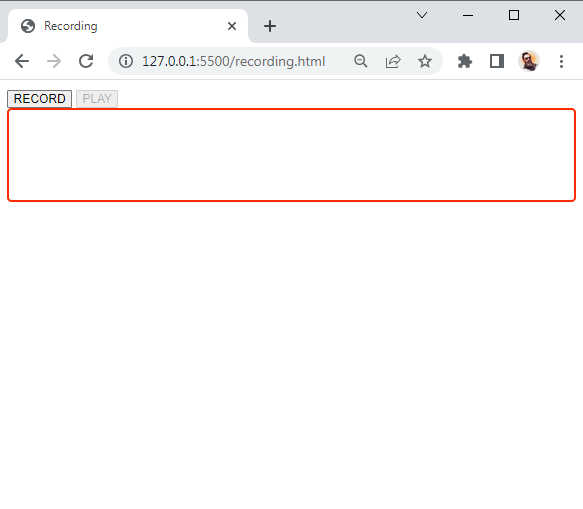
</body>

<script type="module">

  import WaveSurfer from './js/wavesurfer/wavesurfer.esm.min.js'

</script>

</html>

نمایش فایل ایجاد شده در مرورگر به شکل زیر خواهد بود:

حال در قسمت script به نوشتن کد اصلی می شتابیم؛ ابتدا کد اسکریپت تعریف متغیر WaveSurfer را مشابه بخش 1- نمایش سیگنال می­نویسیم:

<!DOCTYPE html>

<html>

<head></head>

<body>

    <button id="record">RECORD</button>

    <button id="play" disabled>PLAY</button>

    <div id="waveform" style="border: 3px solid orangered; border-radius: 5px; padding: 50px;">

        <!-- the waveform will be rendered here -->

    </div>

</body>

<script type="module">

  import WaveSurfer from './js/wavesurfer/wavesurfer.esm.min.js'

*const* wavesurfer = WaveSurfer.create({

        container: '#waveform',

        waveColor: '#4F4A85',

        progressColor: '#383351',

    })

</script>

</html>

نکته: این کد پایه اصلی بیشتر مواردی خواهدبود که از پلاگین wavesurfer بهره می­برند

نکته: با توجه به اینکه مقرر است سیگنال صوتی گفتار ضبط شده توسط کاربر پردازش و نمایش داده شود نیازی به استفاده از قابلیت url و وارد کردن سیگنال صوتی دیگری نمی­باشد

در ادامه می­بایست برای استفاده ار قابلیت ضبط پلاگین، فایل جاواسکریپت record.esm.min.js نیز علاوه بر فایل اصلی وارد کرد

    import WaveSurfer from './js/wavesurfer/wavesurfer.esm.min.js'

    import RecordPlugin from './js/wavesurfer/record.esm.min.js'

حال کدهای مربوط به RecordPlugin و دکمه­های record و play را بصورت ذیل تعریف خواهیم کرد.

    // Play/Pause

    wavesurfer.once('ready', () *=>* {

        playButton.onclick = () *=>* {

            wavesurfer.playPause()

        }

        wavesurfer.on('play', () *=>* {

            playButton.textContent = 'Pause'

        })

        wavesurfer.on('pause', () *=>* {

            playButton.textContent = 'Play'

        })

    })

    // Record/Stop

    recButton.onclick = () *=>* {

        if (wavesurfer.isPlaying()) {

            wavesurfer.pause()

        }

        if (record.isRecording()) {

            record.stopRecording()

            recButton.textContent = 'Record'

            playButton.disabled = false

            return

        }

        recButton.disabled = true

        record.startRecording().then(() *=>* {

            recButton.textContent = 'Stop'

            recButton.disabled = false

        })

    }

    // Initialize the Record plugin

*const* record = wavesurfer.registerPlugin(RecordPlugin.create())

*const* recButton = document.querySelector('#record')

*const* playButton = document.querySelector('#play')

دستور document.querySelector('#record') برای تعریف دستوری است که دکمه record می­فرستد و همچنین دستور document.querySelector('#play) برای تعریف دستور دکمه play می­باشد.

در این مرحله، دستورات مربوط به ارائه لینک دانلود برای ذخیره فایل سیگنال گفتار ضبط شده را مطرح می­کنیم:

    // Download link

*const* link = document.querySelector('#audiolink')

    record.on('stopRecording', () *=>* {

        link.href = record.getRecordedUrl()

        link.download = 'recording.wav'

        link.style.display = ''

    })

    record.on('startRecording', () *=>* {

        link.href = ''

        link.download = ''

        link.style.display = 'none'

    })

## بزرگنمایی سیگنال

برای افزودن قابلیت بزرگنمایی سیگنال گفتار ابتدا باید مقدمات آن را با اضافه کردن تگ label به فایل html خود فراهم کنیم:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Recording</title>

</head>

<body>

    <button id="record">RECORD</button>

    <button id="play" disabled>PLAY</button>

    <div id="waveform" style="border: 3px solid orangered; border-radius: 5px; padding: 50px;">

        <!-- the waveform will be rendered here -->

    </div>

    <label>

        Zoom: <input type="range" min="10" max="1000" value="100" />

    </label>

</body>

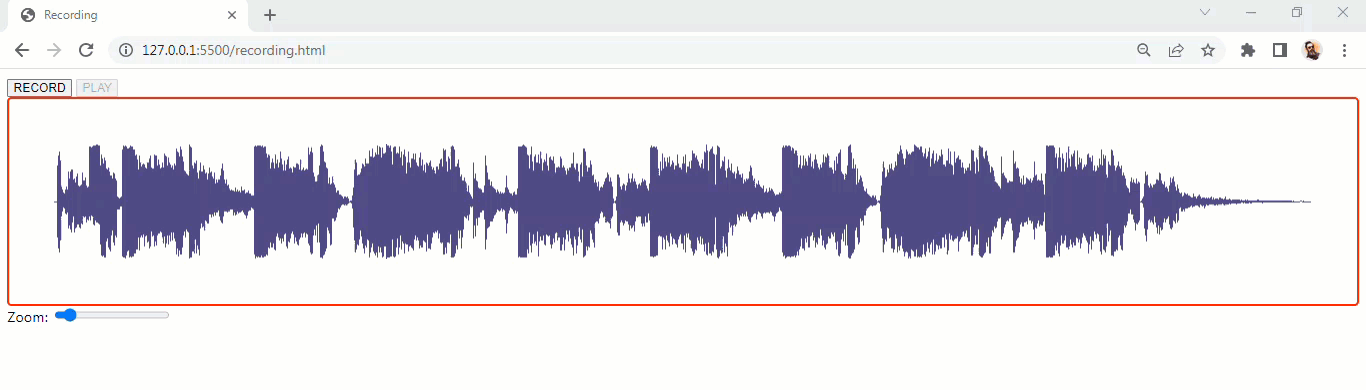
<script type="module">

   ...

</script>

</html>

حال با استفاده از کد دستوری زیر که داخل تگ <script> قرار می­گیرد، می­توان قابلیت بزرگ­نمایی یا همان zoom را به سیگنال گفتار ضبط شده و یا هر نوع سیگنال گفتاری اضافه نمود.

چگونگی نمایش آن در مرورگر به شکل ذیل خواهد بود

    // Update the zoom level on slider change

    wavesurfer.once('decode', () *=>* {

*const* slider = document.querySelector('input[type="range"]')

        slider.addEventListener('input', (*e*) *=>* {

*const* minPxPerSec = *e*.target.valueAsNumber

            wavesurfer.zoom(minPxPerSec)

        })

    })

## افزودن جدول زمانی به سیگنال

برای افزودن جدول زمانی، همانند مورد ضبط نیاز به استفاده از پلاگین دیگری به نام timeline.esm.min.js از زیرمجموعه­های پلاگین wavesurfer داریم که می­بایست قبل از هر اقدامی به فایل HTML افزوده گردد.

    import WaveSurfer from './js/wavesurfer/wavesurfer.esm.min.js'

    import RecordPlugin from './js/wavesurfer/record.esm.min.js'

    import TimelinePlugin from './js/wavesurfer/timeline.esm.min.js'

سپس کافیست که متغیر­های آن را در اسکریپت تعریف و استفاده کرد.

    // Create a timeline plugin instance with custom options

*const* topTimeline = TimelinePlugin.create({

        height: 20,

        insertPosition: 'beforebegin',

        timeInterval: 0.2,

        primaryLabelInterval: 5,

        secondaryLabelInterval: 1,

        style: {

            fontSize: '20px',

            color: '#2D5B88',

        },

    })

    // Create a second timeline

*const* bottomTimline = TimelinePlugin.create({

        height: 10,

        timeInterval: 0.1,

        primaryLabelInterval: 1,

        style: {

            fontSize: '10px',

            color: '#6A3274',

        },

    })

لازم به ذکر است که متغیرهای تعریف شده باید بصورت ذیل به متغیر اصلی wavesurfer با شاخصه waveform اضافه شوند:

    // Create an instance of WaveSurfer

*const* wavesurfer = WaveSurfer.create({

        container: '#waveform',

        waveColor: '#4F4A85',

        progressColor: '#383351',

        plugins: [topTimeline, bottomTimline],

    })

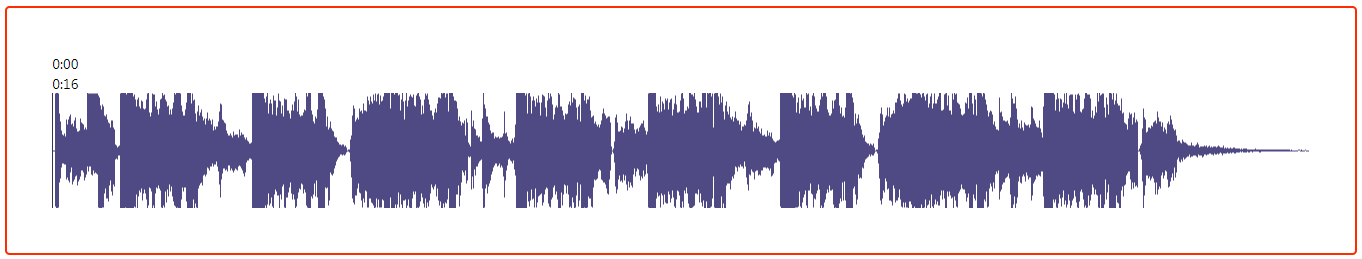
نکته: اجباری به استفاده از topTimeline و bottomTimline بصورت همزمان وجود ندارد و ما می­توانیم تنها به یک مورد به انتخاب و سلیقه خودمان بسنده کنیم.

و در نهایت، فایل نهایی بصورت زیر نمایش داده خواهد شد:

نکته: متغیرهای مربوط به timeline حتما می­بایست قبل از متغیر wavesurfer تعریف شوند چرا که از آن­ها به عنوان یکی از مبانی تعریف متغیر wavesurfer استفاده می­کنیم

## افزودن تایمر پخش

هرچند نیازی ضروری برای افزودن تایمر پخش نمی­باشد چرا که تایم­لاین یا همان جدول زمانی کارایی تایمر را نیز دارد ولی بهرحال شرح آن در این بخش خالی از لطف نخواهد بود.

برای افزودن تایمر ابتدا باید کدهای تمپلیت آن را با دو تگ <div> با شناسه­های time و duration داخل تگ با شناسه waveform به کد HTML خود اضافه کنیم

    <div id="waveform" style="border: 3px solid orangered; border-radius: 5px; padding: 50px;">

        <!-- the waveform will be rendered here -->

<div id="time">0:00</div>

        <div id="duration">0:00</div>

    </div>

و در ادامه کد اسکریپت تعریف و مدیریت این دو شناسه را داخل تگ اسکریپت اضافه می­کنیم:

    // Current time & duration

    {

*const* formatTime = (*seconds*) *=>* {

*const* minutes = Math.floor(*seconds* / 60)

*const* secondsRemainder = Math.round(*seconds*) % 60

*const* paddedSeconds = `0${secondsRemainder}`.slice(-2)

            return `${minutes}:${paddedSeconds}`

        }

*const* timeEl = document.querySelector('#time')

*const* durationEl = document.querySelector('#duration')

        wavesurfer.on('decode', (*duration*) *=>* (durationEl.textContent = formatTime(*duration*)))

        wavesurfer.on('timeupdate', (*currentTime*) *=>* (timeEl.textContent = formatTime(*currentTime*)))

    }

## تغییر سرعت پخش سیگنال

    // Set the playback speed rate

*const* speeds = [0.5, 0.75, 1, 1.5, 2]

    document.querySelector('#speed').addEventListener('input', (*e*) *=>* {

*const* speed = speeds[e.target.valueAsNumber]

        document.querySelector('#rate').textContent = speed.toFixed(2)

        wavesurfer.setPlaybackRate(speed)

        wavesurfer.play()

    })

  <div id="waveform" style="border: 3px solid orangered; border-radius: 5px; padding: 50px;">

      <div> Playback rate: <span id="rate">1.00</span>x </div>

  </div>

  <div>Playback Rate :

      <span>0.5x</span>

      <input id="speed" type="range" min="0" max="4" step="1" value="2"/>

      <span>2x</span>

  </div>

## ایجاد نواحی بر روی سیگنال

<script type="module">

  import WaveSurfer from './js/wavesurfer/wavesurfer.esm.min.js'

  import RegionsPlugin from './js/wavesurfer/regions.esm.min.js'

*const* wavesurfer = WaveSurfer.create({

    container: '#waveform',

    waveColor: '#4F4A85',

    progressColor: '#383351',

    url: './example.mp3',

  })

  // Initialize the Regions plugin

*const* wsRegions = wavesurfer.registerPlugin(RegionsPlugin.create())

  // Create some regions at specific time ranges

  wavesurfer.on('decode', () *=>* {

    // Regions

    wsRegions.addRegion({

      start: 1,

      end: 8,

      content: 'Red Label',

      color: 'rgba(234,6,6, 0.7)',

      drag: false,

      resize: true,

    })

    wsRegions.addRegion({

      start: 9,

      end: 12,

      content: 'Drag me',

      color: 'rgba(130,214,22,0.7)',

      resize: false,

    })

  })

</script>

  <div id="waveform" style="border: 3px solid orangered; border-radius: 5px; padding: 50px;">

  </div>

## ایجاد حلقه برای پخش مکرر بر روی نواحی

  <div id="waveform" style="border: 3px solid orangered; border-radius: 5px; padding: 50px;">

      <label><input type="checkbox" checked="${loop}"/>Loop regions</label>

  </div>

    // Loop a region on click

*let* loop = false

    // Toggle looping with a checkbox

    document.querySelector('input[type="checkbox"]').onclick = (*e*) *=>* {

        loop = e.target.checked

    }

    {

*let* activeRegion = null

    wsRegions.on('region-in', (*region*) *=>* {

        activeRegion = region

    })

    wsRegions.on('region-out', (*region*) *=>* {

        if (activeRegion === region) {

        if (loop) {

            region.play()

        } else {

            activeRegion = null

        }

        }

    })

    wsRegions.on('region-clicked', (*region*, *e*) *=>* {

        e.stopPropagation() // prevent triggering a click on the waveform

        activeRegion = region

        region.play()

        region.setOptions({ color: randomColor() })

    })

    // Reset the active region when the user clicks anywhere in the waveform

    ws.on('interaction', () *=>* {

        activeRegion = null

    })

    }

## بارگذاری فایل صوتی ضبط شده

*const* upButton = document.querySelector('#audiolink')

    // Upload Click Function

    upButton.onclick = () *=>* {

        progressBarContainer.style.display = 'block';

*const* formData = new FormData();

*const* recordedUrl = record.getRecordedUrl();

*const* blob = fetch(recordedUrl).then(*response* *=>* response.blob());

        blob.then(*audioBlob* *=>* {

            formData.append('audioData', audioBlob, '{{phrase.language}}-{{ user.email|cut:"@"|cut:"." }}-{{phrase.title|cut:" "|lower }}-{% now "YmdHis"%}.wav');

            formData.append('textData', '{{phrase.id}}');

*const* xhr = new XMLHttpRequest();

            xhr.open("POST", '/record/{{phrase.id}}/', true);

            xhr.setRequestHeader('X-CSRFToken', '{{ csrf\_token }}');

            // Progress event handler

            xhr.upload.addEventListener("progress", *function*(*e*) {

                if (e.lengthComputable) {

*const* percentCompleted = Math.round((e.loaded \* 100) / e.total);

                    progressBar.style.width = percentCompleted + '%';

                    progressLabel.textContent = percentCompleted + '%';

                    // Add animated progress bar if desired

                    progressBar.style.transition = 'none';

                    progressBar.style.transition = 'width 1s ease-in-out';

                    if (percentCompleted === 100) {

                        // Display the modal after progress reaches 100%

*const* modalNotification = document.getElementById('modal-notification');

*const* bootstrapModal = new bootstrap.Modal(modalNotification);

                        bootstrapModal.show();

                    }

                }

            }, false);

            xhr.onreadystatechange = *function*() {

                if (xhr.readyState === XMLHttpRequest.DONE) {

                    if (xhr.status === 200) {

                        // Redirect or display success message

                        window.location.href = '/record/{{phrase.id}}/';

                    } else {

                        // Display error message

                        console.error('Error:', xhr.statusText);

                    }

                }

            };

            xhr.send(formData);

        }, 'audio/wav');

        upButton.disabled = true

    }

# بخــــش سوم – ایجاد پروژه جنگو



## مقدمه

حال به سراغ ایجاد پروژه جنگو در فضای اختصاصی در نظر گرفته شده می­رویم.

python3 -m venv venv

قبل از هرچیزی باید لیست پکیج­های مورد نیاز برای این پروژه را نصب کنیم:

با دستور ذیل می­توان تمامی پکیج­ها و نسخه نصب شده آن­ها در محیط اختصاصی را لیست کرد:

Package Version ---------------------------- ----------- Django 4.1.4 ghasedakpack 0.1.13 jalali-pandas 0.2.2 jdatetime 3.8.2 librosa 0.10.1 pandas 1.5.0 Pillow 10.0.0 python-speech-features 0.6 requests 2.31.0 tensorflow 2.13.0

## ایجاد ساختار پروژه

پس از اطمینان از نصب جنگو، اجرای دستور ذیل، نسبت به ایجاد پروژه با نام stammer و همچنین دو اپ coreapp و clientapp اقدام می­کنیم

(venv) python manage.py startapp coreapp

(venv) python manage.py startapp clientapp

(venv) django-admin startproject stammer

ایجاد پروژه در جنگو در حقیقت شروع ساخت وب­سایت خواهد بود

## ایجاد مدل­ها ClientApp Models

احراز هویتی که جنگو ارائه می­کند برای اکثر موارد معمول به اندازه کافی خوب است، اما ممکن است نیازهایی وجود داشته باشد که توسط مدل کاربری پیش­فرض جنگو برآورده نشوند. به عنوان مثال، سیستم احراز هویت پیش­فرض جنگو به کاربران وب­سایت اجازه می­دهد که فقط با استفاده از نام کاربری و رمز عبور خود وارد شوند. اما در وب‌سایت‌های امروزی، ممکن است نیاز باشد به کاربران این امکان را داد که بتوانند با استفاده از آدرس ایمیل خود به جای نام کاربری وارد وب­سایت شوند و یا با استفاده از شماره موبایل و بعد از ارسال کد تایید بتوانند از خدمات وبسایت استفاده کنند.

توصیه می­شود از همان ابتدای شروع یک پروژه، یک مدل کاربر سفارشی تنظیم شود، حتی اگر مدل کاربر پیش­فرض برای پروژه کافی باشد. این مدل مانند مدل کاربر پیش‌فرض رفتار می‌کند، اما در صورت نیاز می‌توان آن را در آینده سفارشی کرد.

### ایجاد مدل کاربری سفارشی

پیش از ایجاد مدل کاربر سفارشی حتما می­بایست در نظر داشت که دستور migrates را اعمال نکنیم و حتما به یاد داشته باشیم قبل از اعمال اولین migartion باید مدل کاربر سفارشی را ایجاد کنیم.

برای ایجاد یک مدل کاربر از طریق ارث­بری از دو کلاس AbstractUser و یا AbstractBaseUser می­توان بهره گرفت.

در این پروژه چون فیلدهای موجود در مدل کاربری پیش­فرض، نیاز پروژه را فراهم کرده و تنها قصد داریم شماره موبایل را جایگزین فیلد نام کاربری کنیم از گزینه AbstractUser استفاده خواهیم کرد.

در حقیقت، AbstractUser اجرای کامل مدل User پیش فرض را به عنوان یک مدل قابل گسترش ارائه می­دهد، به این معنی که فیلدهای کاملی را که همراه با مدل User پیش­فرض ارائه می­شوند، به اضافه فیلدهای جدیدی که در پروژه تعریف می­شوند، قابل دستیابی خواهند بود.

به­عنوان نمونه، در مدل سفارشی زیر، فیلد username از مدل User را حذف و فیلدهای ایمیل و شماره موبایل و کد اعتبارسنجی و زمان انقضای کد اعتبار سنجی را با توجه به نیاز پروژه اضافه نمودیم.

**clientapp/models.py**

from django.db import models

from django.contrib.auth.models import AbstractUser

from django.core.validators import RegexValidator

from django.conf import settings

import uuid

from . import managers

*class* customUser(AbstractUser):

    username = None

    email = models.EmailField(*max\_length*=200, *unique*=True)

    mobile = models.CharField(*max\_length*=11, *unique*= True, *validators*=[RegexValidator(*regex*=*r*'^09\d{9}$', *message*="The mobile number should consist of 11 digits and begin with 09, just like 09XXXXXXXXX.")])

    otp = models.PositiveBigIntegerField(*blank*=True, *null*=True)

    otp\_create = models.DateTimeField(*auto\_now*=True)

    USERNAME\_FIELD = 'email'

    REQUIRED\_FIELDS= ['mobile']

    objects = managers.customManager() # Define Custom Manager

*def* \_\_str\_\_(*self*):

        return *self*.email

در مدل فوق دو گزینه otp و otpCreate را جهت ارسال کد تایید ثبت­نام کاربری و زمان انقضای کد Otp به مدل کاربری اضافه نمودیم.

در این راستا، جنگو امکان می­دهد تا با ارائه یک مقدار برای تنظیم AUTH\_USER\_MODEL که به یک مدل سفارشی اشاره می­کند، مدل کاربر پیش­فرض را لغو کرده و مدل کاربری سفارشی خود را جایگزین کنیم. بنابراین بعد از ایجاد مدل، حتما می­بایست آن را در تنظیمات پروژه به جنگو معرفی نماییم:

**tutorial/settings.py**

AUTH\_USER\_MODEL = "appName.customUser"

### ایجاد کلاس Manager سفارشی

در ادامه می­بایست به ایجاد manager سفارشی برای مدل کاربری سفارشی خود باعنوان customManager اقدام نماییم:

**clientapp/managers.py**

from django.contrib.auth.base\_user import BaseUserManager

*class* customManager(BaseUserManager):

*def* create\_user(*self*, *email*, *password*, \*\**other\_fields*):

        user = *self*.model(*email*=*email*, \*\**other\_fields*)

        user.set\_password(*password*)

        user.save()

        return user

*def* create\_superuser(*self*, *email*, *password*, \*\**other\_fields*):

*other\_fields*.setdefault('is\_superuser', True)

*other\_fields*.setdefault('is\_staff', True)

*other\_fields*.setdefault('is\_active', True)

        if *other\_fields*.get('is\_superuser') is not True:

            raise ValueError('Superuser must has is\_superuser = True')

        if *other\_fields*.get('is\_staff') is not True:

            raise ValueError('Superuser must has is\_staff = True')

        return *self*.create\_user(*email*, *password*, \*\**other\_fields*)

در ادامه مدل پروفایل کاربری را برای اضافه نمودن اطلاعات موردنیاز دیگر از کاربر ایجاد می­کنیم:

**clientapp/models.py**

*class* profile(models.Model):

    user = models.OneToOneField(settings.AUTH\_USER\_MODEL, *related\_name*='profile', *on\_delete*=models.CASCADE)

    name = models.CharField(*max\_length*=500, *null*=True, *blank*=True)

    email = models.EmailField(*null*=True, *blank*=True)

    mobile = models.CharField(*max\_length*=11)

    picture = models.ImageField(*default*='images/profile/jordan.jpg', *null*=True, *upload\_to*='uploads/profiles')

    medical\_history = models.CharField(*max\_length*=15, *choices*=treatmentChoices)

    medical\_seasons = models.CharField(*max\_length*=15, *null*=True, *blank*=True, *choices*=sessionChoices)

    microphone = models.CharField(*default*='NOK',*max\_length*=15)

    create = models.DateTimeField(*auto\_now\_add*=True)

    lastlogin = models.DateTimeField(*null*=True, *blank*=True)

    lastlogout = models.DateTimeField(*null*=True, *blank*=True)

    id = models.UUIDField(*default*=uuid.uuid4, *primary\_key*=True, *unique*=True, *editable*=False)

*class* Meta:

        ordering = ["name"]

*def* \_\_str\_\_(*self*):

        return *self*.user.email

حال می­توان دستورات makemigrations و migrate را بترتیب و پشت سرهم اجرا کرد

python manage.py makemigrations

python manage.py migrate

قبل از هر اقدامی، می­بایست نسبت به ایجاد کاربری با بالاترین سطح دسترسی ممکن (super user) اقدام کرد:

python manage.py createsuperuser

## ایجاد سیگنال­ها ClientApp Signlas

برای ارتباط برقرار کردن بهنگام ثبت­نام کاربر بین دو مدل cutomuser و profile بدلیل مکمل بودن یکدیگر، نیاز به استفاده از سیگنال­ها می­باشد

**clientapp/signals.py**

from django.db.models import signals

from django.dispatch import receiver

from clientapp.models import customUser, profile

@receiver(signals.post\_save, *sender*=customUser)

*def* createUser(*sender*, *instance*, *created*, \*\**kwargs*):

    if *created*:

        profile.objects.create(

*user* = *instance*,

*email* = *instance*.email,

*mobile* = *instance*.mobile

        )

@receiver(signals.post\_delete, *sender*=profile)

*def* deleteProfile(*sender*, *instance*, \*\**kwargs*):

    userObj = *instance*.user

    userObj.delete()

@receiver(signals.post\_save, *sender*=profile)

*def* updateProfile(*sender*, *instance*, *created*, \*\**kwargs*):

    userObj = *instance*.user

    if *created* == False:

        userObj.email = *instance*.email

        if ' ' in *instance*.name:

            userObj.last\_name = *instance*.name.split(' ', 1)[1]

        userObj.first\_name = *instance*.name.split(' ', 1)[0]

        userObj.save()

## پیاده­سازی ارسال کد اعتبارسنجی

برای ارسال کد اعتبارسنجی برای کاربر بهنگام ثبت نام، نیاز می­باشد که از یک ارائه دهنده این سرویس استفاده نمود که در این پروژه از سرویس قاصدک بدین منظور استفاده شده است:

**clinetapp/utils.py**

import random

import ghasedakpack

*def* otpCreate():

    number = random.randint(10000,99999)

    return str(number)

*def* otpSend(*phone*, *otp*):

    sms = ghasedakpack.Ghasedak("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

    sms.verification({'receptor':*phone*,'type': '1','template': 'OTP','param1':*otp*})

## ایجاد تابع­های ClientApp Views

در نهایت برای تکمیل اپ clientapp نیاز به ایجاد فانکشن­هایی در views.py خواهیم بود که بتوانند فعالیت­های مربوط به احرازهویت، ثبت­نام و ... کاربر را مدیریت کنند.

### ثبت­نام کاربر و ارسال کد اعتبار سنجی:

**clinetapp/views.py**

*def* signUp(*request*):

    if *request*.user.is\_authenticated:

        return redirect('Home')

    else:

        form = clientForms.userForm()

        if *request*.method == 'POST':

            form = clientForms.userForm(*request*.POST)

            try:

                user = customUser.objects.get(*email* =  *request*.POST['email'])

                if user.is\_active == False:

                    user.delete()

            except:

                pass

            if form.is\_valid():

                userObj = form.save(*commit*=False)

                userObj.email = userObj.email.lower()

                if ' ' in userObj.first\_name:

                    userObj.last\_name = userObj.first\_name.split(' ', 1)[1]

                userObj.first\_name = userObj.first\_name.split(' ', 1)[0]

                otp = otpCreate()

                userObj.otp = otp

                otpSend(userObj.mobile, otp)

                userObj.otp\_create = timezone.now()

                userObj.is\_active = False

*request*.session["mobile"] = userObj.mobile

                userObj.save()

                messages.info(*request*, "Verification code was sent to your pohne.\'" + str(userObj.mobile) + "\'")

                return redirect('verify')

            else:

                messages.error(*request*, 'Something went Wrong, please check the errors')

        context = {'form':form}

        return render(*request*, 'signup.html', context)

### بررسی کد اعتبارسنجی وارد شده توسط کاربر:

**clinetapp/views.py**

*def* verify(*request*):

    mobile = *request*.session.get('mobile')

    userObj = customUser.objects.get(*mobile*=mobile)

    timerSeconds = 120

    context = {'userObj': userObj, 'timerSeconds':timerSeconds }

    timeDelta = timezone.now() - userObj.otp\_create

    if *request*.method == 'POST':

        if timeDelta.total\_seconds() > timerSeconds:

                otp = otpCreate()

                otpSend(userObj.mobile, otp)

                userObj.otp = otp

                userObj.otp\_create = timezone.now()

                userObj.save()

                messages.warning(*request*, "Sorry, your time has expired. We have sent another code to your phone.\'" + str(userObj.mobile) + "\'")

                return redirect('verify')

        else:

            verifyCode = *request*.POST.get('otpCode')

            if str(userObj.otp) == str(verifyCode):

                userObj.is\_active = True

                userObj.save()

                login(*request*, userObj)

                del *request*.session['mobile']

                return redirect('updateAccount')

            else:

                messages.error(*request*, 'Wrong Code! Please check the code and try again')

                return redirect('verify')

    return render(*request*, 'verify.html', context)

### بروزرسانی اطلاعات پروفایل کاربری:

**clinetapp/views.py**

@login\_required(*login\_url*='signIn')

*def* updateAccount(*request*):

    profileObj = profile.objects.get(*user*=*request*.user)

    form = clientForms.profileForm(*instance*=profileObj)

    if *request*.method == 'POST':

        form = clientForms.profileForm(*request*.POST, *request*.FILES, *instance*=profileObj)

        if form.is\_valid():

            profileObj = form.save(*commit*=False)

            profileObj.lastlogin = timezone.now()

            profileObj.save()

            messages.success(*request*, 'Profile information was updated successfully')

            return redirect('Home')

    context = {'form':form}

    return render(*request*, 'update-account.html', context)

### ورود به سامانه:

**clinetapp/views.py**

*def* signIn(*request*):

    if *request*.user.is\_authenticated:

        return redirect('Home')

    else:

        if *request*.method == 'POST':

            usr = *request*.POST['Username'].lower()

            psw = *request*.POST['Password']

            try:

                user = customUser.objects.get(*email* =  usr)

                if user.is\_active == False:

                    messages.error(*request*, "registration not been completed.")

                else:

                    userCheck = authenticate(*request*,*username*=usr,*password*=psw)

                    if userCheck is not None:

                        login(*request*, userCheck)

                        profileObj = *request*.user.profile

                        profileObj.lastlogin = timezone.now()

                        profileObj.microphone = 'NOK'

                        profileObj.save()

                        return redirect('Home')

                    else:

                        messages.error(*request*, Please check and try again")

            except:

                pass

        return render(*request*, 'signin.html')

### خروج از سیستم:

**clinetapp/views.py**

@login\_required(*login\_url*='signIn')

*def* signOut(*request*):

    profileObj = *request*.user.profile

    profileObj.microphone = 'NOK'

    profileObj.lastlogout = timezone.now()

    profileObj.save()

    logout(*request*)

    return redirect('signIn')

### فراموشی و تغییر رمز عبور

**clinetapp/views.py**

*def* forgetPassword(*request*):

    if *request*.method == 'POST':

        mobile = *request*.POST['mobile']

        if mobile:

            try:

                userObj = customUser.objects.get(*mobile*=mobile)

                otp = otpCreate()

                userObj.otp = otp

                userObj.otp\_create = timezone.now()

                otpSend(userObj.mobile, otp)

*request*.session["mobile"] = userObj.mobile

                userObj.save()

                messages.info(*request*, "Verification code was sent to your pohne.\'" + str(userObj.mobile) + "\'")

                return redirect('verifyPassword')

            except:

                messages.error(*request*, 'The entered mobile number does not exist')

        else:

            messages.error(*request*, 'You must enter your mobile number')

    return render(*request*, 'forgot-password.html')

**clinetapp/views.py**

*def* verifyPassword(*request*):

    mobile = *request*.session.get('mobile')

    userObj = customUser.objects.get(*mobile*=mobile)

    timerSeconds = 120

    timeDelta = timezone.now() - userObj.otp\_create

    context = {'userObj': userObj, 'timerSeconds':timerSeconds }

    if *request*.method == 'POST':

        if timeDelta.total\_seconds() > timerSeconds:

                otp = otpCreate()

                otpSend(userObj.mobile, otp)

                userObj.otp = otp

                userObj.otp\_create = timezone.now()

                userObj.save()

                messages.warning(*request*, "Sorry, your time has expired. We have sent another code to your phone.\'" + str(userObj.mobile) + "\'")

                return redirect('verifyPassword')

        else:

            verifyCode = *request*.POST.get('otpCode')

            if str(userObj.otp) == str(verifyCode):

                return redirect('resetPassword')

            else:

                messages.error(*request*, 'Wrong Code! Please check the Code and Try Again')

                return redirect('verifyPassword')

    return render(*request*, 'verify-password.html', context)

**clinetapp/views.py**

*def* resetPassword(*request*):

    mobile = *request*.session['mobile']

    user = customUser.objects.get(*mobile*=mobile)

    if *request*.method == 'POST':

        form = clientForms.ResetPasswordForm(user, *request*.POST)

        if form.is\_valid():

            form.save()

            messages.success(*request*, 'Your password was changed successfully.')

            return redirect('signIn')  # Replace 'home' with the URL name for your home page

    else:

        form = clientForms.ResetPasswordForm(user)

    context = {'form': form}

    return render(*request*, 'reset-password.html', context)

## ایجاد مسیرهای ClientApp URL

در خاتمه بخش اپ clientapp می­بایست مسیرهای درخواست کاربری که منتج به فراخوانی فانکش­های فوق می­شوند را تعریف کرد:

**clinetapp/urls.py**

from django.urls import path

from . import views

urlpatterns = [

    path('signin/', views.signIn, *name*='signIn'),

    path('signout/',views.signOut, *name*='signOut'),

    path('signup/', views.signUp, *name*='signUp'),

    path('verify/', views.verify, *name*='verify'),

    path('update-account/', views.updateAccount, *name*='updateAccount'),

    path('forget-password/', views.forgetPassword, *name*='forgetPassword'),

    path('verify-password/', views.verifyPassword, *name*='verifyPassword'),

    path('reset-password/', views.resetPassword, *name*='resetPassword'),

]

## ایجاد مدل­های CoreApp Models

جهت پیاده­سازی فرایند پروژه، نیاز خواهیم داشت که مدل­هایی را بر اساس نیاز­های پروژه ایجاد کنیم:

* مدلی عبارات منتخب برای ارائه به کاربران جهت روخوانی - phrase
* مدلی برای ضبط و ذخیره سازی صدای کاربران - speech
* مدلی برای ایجاد پست­های مربوط به وبلاگ سامانه - blog
* مدلی برای لیست کردن دسته­بندی موضوعات پست­های وبلاگ - category
* مدلی برای ذخیره­سازی فایل­های مربوط به هوش­مصنوعی - ai

**coreApp/models.py**

from django.db import models

from django.conf import settings

import uuid

*class* phrase(models.Model):

    user = models.ForeignKey(settings.AUTH\_USER\_MODEL, *on\_delete*=models.CASCADE)

    title = models.CharField(*unique*=True, *max\_length*=25)

    body = models.TextField()

    level = models.CharField(*max\_length*=50, *choices*=grade)

    language = models.CharField(*max\_length*=50, *choices*=lang)

    create = models.DateTimeField(*auto\_now\_add*=True)

    id = models.UUIDField(*default*=uuid.uuid4, *unique*=True, *primary\_key*=True, *editable*=False)

*def* \_\_str\_\_(*self*):

        return *self*.title

*class* blog(models.Model):

    user = models.ForeignKey(settings.AUTH\_USER\_MODEL, *on\_delete*=models.PROTECT)

    language = models.CharField(*max\_length*=50, *choices*=lang)

    category = models.ForeignKey(category, *null*=True, *blank*=True, *on\_delete*=models.PROTECT)

    video = models.FileField(*null*=True, *blank*=True, *upload\_to*="uploads/video")

    title = models.CharField(*max\_length*=500)

    body = models.TextField(*null*=True, *blank*=True)

    create = models.DateTimeField(*auto\_now\_add*=True)

    id = models.UUIDField(*default*=uuid.uuid4, *unique*=True, *primary\_key*=True, *editable*=False)

*def* \_\_str\_\_(*self*):

        return *self*.title

*class* speech(models.Model):

    user = models.ForeignKey(settings.AUTH\_USER\_MODEL, *on\_delete*=models.CASCADE)

    phrase = models.ForeignKey(phrase, *null*=True, *on\_delete*=models.PROTECT)

    language = models.CharField(*default*='Fa', *max\_length*=15)

    audio = models.FileField(*upload\_to*='uploads/audio')

    repetition = models.CharField(*max\_length*=500, *null*=True, *blank*=True)

    prolongation = models.CharField(*max\_length*=500, *null*=True, *blank*=True)

    create = models.DateTimeField(*auto\_now\_add*=True)

    id = models.UUIDField(*default*=uuid.uuid4, *primary\_key*=True, *unique*=True, *editable*=False)

*def* \_\_str\_\_(*self*):

        return *f*'{*self*.user.email} - {*self*.phrase.title} - {*self*.create}'

برای نمایش مدل­های ایجاد شده در اینترفیس ادمین، می­بایست روند معرفی مدل و ثبت کردن در اینترفیس ادمین طی شود

*class* ai(models.Model):

    user = models.ForeignKey(settings.AUTH\_USER\_MODEL, *on\_delete*=models.PROTECT)

    ai\_type = models.CharField(*max\_length*=15, *choices*=aiChoices)

    ai\_model = models.FileField(*upload\_to*='uploads/ai')

    create = models.DateTimeField(*auto\_now\_add*=True)

    id = models.UUIDField(*default*=uuid.uuid4, *primary\_key*=True, *unique*=True, *editable*=False)

*def* \_\_str\_\_(*self*):

        return *f*'{*self*.ai\_type} - {*self*.create}'

*class* category(models.Model):

    user = models.ForeignKey(settings.AUTH\_USER\_MODEL, *on\_delete*=models.PROTECT)

    language = models.CharField(*max\_length*=50, *choices*=lang)

    title = models.CharField(*max\_length*=500)

    create = models.DateTimeField(*auto\_now\_add*=True)

    id = models.UUIDField(*default*=uuid.uuid4, *unique*=True, *primary\_key*=True, *editable*=False)

*def* \_\_str\_\_(*self*):

        return *self*.title

**coreapp/admin.py**

from django.contrib import admin

from . import models

# Register your models here.

admin.site.register(models.speech)

admin.site.register(models.phrase)

admin.site.register(models.category)

admin.site.register(models.blog)

admin.site.register(models.ai)

## ایجاد تابع­های CoreApp Views

در ادامه، به بحث فانکشن­های تعریف شده در فایل views.py مربوط به اپ coreapp جهت مدیریت درخواست­ها و تعریف فرایند­ها می­پردازیم.

بطورکلی می­توان فانکشن­ها را بصورت ذیل طبق فرایند لیست کرد:

* phraseFunc
  + جهت ارائه لیست عبارت­­­­های تعریف شده به کاربر
* recordFunc
  + جهت ضبط و پخش و بارگذاری فایل صوتی ضبط شده توسط کاربر در مدل speech
* recordDetail
  + جهت نمایش نتیجه آنالیز هوش مصنوعی بر روی آخرین صدای بارگذاری شده به کاربر
* archiveFunc
  + جهت نمایش تمامی فایل­های صوتی ضبط شده توسط کاربر به خود کاربر
* blogFunc
  + جهت نمایش تمامی پست­های وبلاگ به کاربر
* addAI
  + جهت اضافه کردن فایل جدید هوش مصنوعی به سامانه
* replaceFile
  + جهت جایگزین کردن کد سامانه برای تشخیص لکنت زبان

**coreapp/views.py**

@login\_required(*login\_url*='signIn')

*def* phraseFunc(*request*):

    if *request*.method == 'POST':

        userObj = *request*.user.profile

        userObj.microphone = 'OK'

        userObj.save()

    phrases = models.phrase.objects.filter(*language\_\_icontains*=*request*.session['lang'])

    phrase = models.micophone.objects.filter(*language\_\_icontains*=*request*.session['lang']).first()

    wallpapers = os.listdir('./static/media/images/wallpapers/')

    context = {'phrases':phrases, 'wallpapers':wallpapers }

    if *request*.user.profile.microphone =='NOK':

        return render(*request*, 'record-check.html', {'phrase':phrase} )

    else:

        return render(*request*, 'phrases.html', context)

from coreapp.prediction import featureExtraction

@login\_required(*login\_url*='signIn')

*def* recordFunc(*request*,*uid*):

    user = *request*.user

    voice = models.speech.objects.filter(*user*=user).first()

    phrase = models.phrase.objects.get(*id* = *uid*)

    context = {'user':user, 'phrase':phrase, 'voice':voice}

    if *request*.method == 'POST':

        audioData = *request*.FILES.get('audioData')

        textData = *request*.POST.get('textData')

        phrase = models.phrase.objects.get(*id*=textData)

        if audioData:

            sound = models.speech(*audio*=audioData, *phrase*=phrase, *user*=user)

            sound.save()

        else:

            messages.error(*request*, 'Somthing went wrong;')

        try:

            voice = models.speech.objects.filter(*user*=user).first()

            repetition, prolongation, images\_2a = featureExtraction(voice.audio.path)

            voice.repetition = repetition

            voice.prolongation = prolongation

            voice.save()

            messages.success(*request*, 'File was uploaded successfully')

            return JsonResponse({'message': 'Success'})

        except Exception as e:

            voice = models.speech.objects.filter(*user*=user).first()

            os.remove(voice.audio.path)

            voice.delete()

            messages.error(*request*, 'Something went wrong;')

            return JsonResponse({'message': 'Success'})

    return render(*request*, 'record.html', context)

در ­فانکشن فوق که مربوط به ضبط، بارگذاری و سپس آنالیز صدای ضبط شده کاربر می­باشد، فانکشن featureExtraction از فایل prediction.py استخراج می­شود که همان کد قابل جایگزینی می­باشد

@login\_required(*login\_url*='signIn')

*def* archiveFunc(*request*):

    profileObj = *request*.user.profile

    try :

        lang = *request*.session['lang']

    except:

        lang = 'Fa'

    audios = models.speech.objects.filter(*user*=*request*.user, *phrase\_\_language*=lang)

    phrases = models.phrase.objects.filter(*language\_\_icontains*=lang)

    selectedPhrase = 'Select'

    if *request*.method == "POST":

        selectedOption = *request*.POST.get('selectedOption')

        if selectedOption == '-':

            audios = models.speech.objects.filter(*user*=*request*.user)

        else:

            audios = models.speech.objects.filter(*user*=*request*.user, *phrase\_\_title*=selectedOption)

            selectedPhrase = selectedOption

    paginatedAudios, paginatedItems, customRange = utils.pagination(*request*, audios, 5)

    if *request*.session['lang'] == "En":

        labels = [speech.create.strftime('%Y-%m-%d %H:%M:%S') for speech in reversed(audios)]

    else:

        labels = [jdatetime.datetime.fromgregorian(*datetime*=speech.create).strftime("%d %b %Y %H:%M") for speech in reversed(audios)]

    repetition = [speech.repetition.count("-") if speech.repetition else 0 for speech in reversed(audios)]

    prolongation = [speech.prolongation.count("-") if speech.prolongation else 0 for speech in reversed(audios)]

    context  = {'profile':profileObj, 'episodes':audios, 'phrases':phrases, 'selectedPhrase':selectedPhrase,

                'paginatedEpisodes':paginatedAudios, 'paginator':paginatedItems, 'range':customRange,

                'labels': labels, 'repetition':repetition, 'prolongation':prolongation}

    return render(*request*, 'archive.html', context)

*def* recordDetail(*request*, *uid*):

    audio = models.speech.objects.get(*id* = *uid*)

    context = {'voice':audio}

    return render(*request*, 'record-detail.html', context)

*def* replaceFile(*request*):

    if *request*.method == 'POST':

        form = forms.fileUploadForm(*request*.POST, *request*.FILES)

        if form.is\_valid():

            uploadedFile = form.cleaned\_data['file']

            if str(uploadedFile) == 'prediction.py':

                filePath = './coreapp/prediction.py'

                os.remove(filePath)

            with open(filePath, 'wb') as newFile:

                for chunk in uploadedFile.chunks():

                    newFile.write(chunk)

            messages.success(*request*, 'New perdiction.py file has been replaced successfully')

            return redirect('Home')

    else:

        form = forms.fileUploadForm()

    return render(*request*, 'forms.html', {'form':form})

*def* addAI(*request*):

    user = *request*.user

    form = forms.aiForm()

    context = {'form':form}

    if *request*.method == 'POST':

        form = forms.aiForm(*request*.POST, *request*.FILES)

        if form.is\_valid():

            aiObj = form.save(*commit*=False)

            aiObj.user = user

            aiObj.save()

            messages.success(*request*, 'New ai model for ' + aiObj.ai\_type + ', was created successfully')

            return redirect('Home')

    return render(*request*, 'forms.html', context)

*def* blogFunc(*request*):

    posts, searchedText = utils.searchPosts(*request*)

    try:

        lang = *request*.session['lang']

    except:

        lang = 'Fa'

    if *request*.GET.get('id'):

        categoryID = *request*.GET.get('id')

        posts = models.blog.objects.filter(*language\_\_icontains* = lang, *category\_\_id* = categoryID)

    categories = models.category.objects.filter(*language\_\_icontains* = lang).annotate(*totalBlogs*=Count('blog'))

    posts, paginatedItems, customRange = utils.pagination(*request*, posts, 5)

    context = {'posts':posts, 'searchText':searchedText, 'paginator':paginatedItems,

               'range':customRange,  'categories':categories}

    return render(*request*,'blog.html', context)

**نکته:** برای درست اجرایی شدن ماژول librosaکه در prediction برای خواندن فایل صوتی ضبط شده استفاده شده است، حتما می­بایست از ماژول FFmpeg استفاده کرد، کافیست که از سایت رسمی آن https://www.gyan.dev/ffmpeg/builds پکیج مربوطه را (ffmpeg-git-full.7z) دانلود کرده و مسیر C:/ffmpeg/bin در در Path Environment تعریف کرد.

برای سیستم­عامل لینوکس، کافیست دستورذیل را اجرا کنیم

sudo apt update

sudo apt upgrade

sudo apt install ffmpeg

جهت اطمینان از تعریف مسیر کافیست در ترمینال دستور ذیل را اجرا کرد

ffmpeg -version

درصورت تعریف، خروجی بدین شکل خواهد بود

ffmpeg version 4.4.2-0ubuntu0.22.04.1 Copyright (c) 2000-2021 the FFmpeg developers

built with gcc 11 (Ubuntu 11.2.0-19ubuntu1)

## ایجاد مسیرهای CoreApp URL

برای اجرایی شدن فانکشن­های ایجاد شده فوق، می­بایست مسیردهی لازم را انجام داد.

**clinetapp/urls.py**

from django.urls import path

from . import views

urlpatterns = [

    path('', views.homeFunc, *name*='Home'),

    path('recording/', views.phraseFunc, *name*='Recording'),

    path('record/<str:uid>/', views.recordFunc, *name*='Record'),

    path('archive/', views.archiveFunc, *name*='archive'),

    path('record-detail/<str:uid>/', views.recordDetail, *name*='recordDetail'),

    path('blog/', views.blogFunc, *name*='Blog'),

    path('add-ai/', views.addAI, *name*='addAI'),

    path('replace-file/', views.replaceFile, *name*='replaceFile'),

]

## ایجاد فانکشن­های Search و Pagination

یک کار رایج برای برنامه های کاربردی وب جستجوی برخی از داده ها در پایگاه داده با ورودی کاربر است. در یک مورد ساده، جستجو می­تواند فیلتر کردن لیستی از داده­ها بر اساس یک دسته باشد. در موارد پیچیده تر ممکن است نیاز به جستجو با وزن دهی، دسته­بندی، برجسته کردن، چندین زبان و غیره باشد.

**coreapp/utils.py**

from django.core.paginator import Paginator, PageNotAnInteger, EmptyPage

from . import models

*def* searchPosts(*request*):

    searchedText = ''

    if *request*.GET.get('search'):

        searchedText = *request*.GET.get('search')

    try:

        lang = *request*.session['lang']

    except:

        lang = 'Fa'

    result = models.blog.objects.distinct().filter(*language\_\_icontains* = lang).filter(*title\_\_icontains* = searchedText)

    return result, searchedText

صفحه­بندی یا همان Pagination، فرآیند شکستن تکه­های بزرگ داده در چندین صفحه وب مجزا است. به جای نمایش تمام داده‌ها در یک صفحه به کاربر، می‌توان تعداد رکوردهای نمایش داده شده در هر صفحه را تعیین کرد و با شماره­گذاری صفحات، سپس داده‌های مربوط به صفحه درخواستی کاربر را بازگرداند.

*def* pagination(*request*, *querySet*, *displayCount*):

    pageNo = *request*.GET.get('page')

    paginatedList = Paginator(*querySet*, *displayCount*)

    try:

*querySet* = paginatedList.page(pageNo)

    except PageNotAnInteger:

        pageNo = 1

*querySet* = paginatedList.page(pageNo)

    except EmptyPage:

        pageNo = paginatedList.num\_pages

*querySet* = paginatedList.page(pageNo)

    leftIndex = int(pageNo) - 4

    rightIndex = int(pageNo) + 5

    if leftIndex < 1:

        leftIndex = 1

    if rightIndex > paginatedList.num\_pages:

        rightIndex = paginatedList.num\_pages + 1

    customRange = range(leftIndex, rightIndex)

    return *querySet*, paginatedList, customRange

برای داشتن کد تمیز و امکان استفاده مکرر، این دو فانکشن در فایل utils.py تعریف شده­اند­

## تنظیمات کلی پروژه

حسن ختام کار، تنظیم کردن و رعایت نکاتی می­باشد که باعث خواهند شد پروژه بدون خطا و در بهترین شکل خود اجرایی گردد و مربوط به تنظیمات settings.py و stammer/urls.py می­باشند

در ابتدا باید مسیرهای آدرس­دهی clientapp و coreapp را در ساختار پروژه تعریف کنیم:

**stammer/urls.py**

from django.contrib import admin

from django.urls import path, include

from django.conf import settings

from django.conf.urls.static import static

urlpatterns = [

    path('admin/', admin.site.urls),

    path('', include('coreapp.urls')),

    path('user/', include('clientapp.urls')),

] + static(settings.MEDIA\_URL, *document\_root*=settings.MEDIA\_ROOT) + static(settings.STATIC\_URL, *documet\_root*=settings.STATIC\_ROOT)

در ادامه موارد ذیل را به فایل settings.py اضافه نمود

**stammer/settings.py**

INSTALLED\_APPS = [

    '...',

    'coreapp',

    'clientapp',

]

from django.contrib.messages import constants as messages

MESSAGE\_TAGS = {

        messages.DEBUG: 'alert-secondary',

        messages.INFO: 'alert-info',

        messages.SUCCESS: 'alert-success',

        messages.WARNING: 'alert-warning',

        messages.ERROR: 'alert-danger',

 }

AUTH\_USER\_MODEL = 'clientapp.customUser'

SESSION\_EXPIRE\_AT\_BROWSER\_CLOSE = True     # optional, as this will log you out when browser is closed

SESSION\_COOKIE\_AGE = 30 \* 60               # Set the session timeout in seconds (e.g., 15 minutes)

SESSION\_SAVE\_EVERY\_REQUEST = True          # Will prrevent from logging you out after 300 seconds

# Static files (CSS, JavaScript, Images)

# https://docs.djangoproject.com/en/4.2/howto/static-files/

STATIC\_URL = 'static/'

STATICFILES\_DIRS = [BASE\_DIR / 'static']

MEDIA\_ROOT = BASE\_DIR / 'static/media'

MEDIA\_URL = 'media/'

STATIC\_ROOT  = BASE\_DIR / 'statics'

TEMPLATES = [

        'DIRS': [ BASE\_DIR / 'templates' ],

]

همچنین مقادیر مربوط به تایم زون را برای نمایش درست تاریخ و زمان، بهتر است بر روی Asia/Tehran تنظیم کرد

# Internationalization

# https://docs.djangoproject.com/en/4.2/topics/i18n/

LANGUAGE\_CODE = 'en-us'

TIME\_ZONE = 'Asia/Tehran'

USE\_I18N = True

USE\_TZ = False