פרויקט סמינר הישן

מגישה : אורה טולדנו

DiversiTech

הכשרה והשתלמות
מכון בית יעקב למורות

מה"ט
המכון הממשלתי להכשרה טכנולוגית

הסמינר הישן

שם המוסד החינוכי: מכון בית יעקב למורות- סמינר הישן

שם הסטודנט: אורה טולדנו

שם החברה המארחת: diversitech טכנולוגיה

שם הלקוח הסופי: הסמינר הישן

שם הפרויקט: הסמינר הישן

תיאור הפרויקט: אתר לניהול האגף האדמיניסטרטיבי של המכון

שם המנחה האישי: תהילה אשלג

שם המנחה האקדמי: תהילה אשלג

תאריך הגשה :

תוכן העניינים:

1. מבוא
2. תיאור חברת Diversitek
3. תיאור הלקוח הסופי – הסמינר הישן
4. תיאור הפרויקט
5. מטרות הפרויקט
6. תהליך העבודה:
    1. שלב א: הכרת החברה והפרויקט.
    2. שלב ב: אפיון ודרישות.
    3. שלב ג: תכנון.
    4. שלב ד: ביצוע ופיתוח.
    5. שלב ה: בדיקות ואבטחת איכות.
    6. שלב ו: הטמעה ותמיכה.
7. סיכום ומסקנות.
8. נספחים.

מבוא :

תיאור כללי של הפרויקט : אתר לניהול האגף האדמיניסטרטיבי של המכון.

כיום בית העסק מתנהל באמצעות קבצי excel.לניהול האדמיניסטטיבי של המכון .בפרויקט זה מתוכנן לבצע המרה של צורת העבודה לדיגיטלית .הפרויקט נוגע בחלקים מסוימים של שירותי המשרד והם : קורסים,תלמידים ודווח נוכחות .

מטרות המערכת :

1. תכלול תשתית אימות והרשאות לזיהוי המשתמש וסמכויותיו המערכת מאפשרת למזכירות לגשת למידע ופעולות באתר לפי הרשאות שקיבלו ממנהל האתר.
2. תכלול תשתית DB (Postgres SQL DB) - לשמירת המידע במרוכז.
3. תכלול תשתית ענן -המערכת מתוכננת להיות פרוסה בענן.
4. תהיה ידידותית למשתמש ,אינפורמטיבית ואינטראקטיבית למכללה
5. תהיה חכמה מהירה וחדשנית -תפותח בארכיטקטורת MS , בשפות : TS , Java ובטכנולוגיות : Angular , Java spring boot

חברת Diversitek

שם החברה : דייברסיטק טכנולוגיה

תחום עיסוק:

דייברסיטק טכנולוגיה הינו בית תוכנה המספק פתרונות טכנולוגיים מגוונים ללקוחות קצה ,ביניהם עסקים קטנים ובינוניים.

מוצרים ושירותים:

פיתוח תוכנות לניהול לידים ולקוחות ,

בניית אתרי תדמית משוכללים ,

פתרונות מותאמים אישית לפי דרישות הלקוח ,

שירותי תמיכה טכנית והטמעה

לקוחות החברה:

עסקים קטנים ובינוניים ממגוון תחומים ,חברות טכנולוגיה ,מוסדות חינוך

מבנה ארגוני:

מנכ"ל :אביגיל מיכלסון

CTO : שוקי גור

PMO : חנה ברגמן

צוותי פיתוח ותמיכה

תיאור תפקיד הסטודנטית בחברה:

הסטודנטיות בפרקטיקום משתלבות בצוותי הפיתוח של דייברסיטק טכנולוגיה ,ועובדות על פרויקטים טכנולוגיים אמיתיים עבור לקוחות החברה .במסגרת הפרקטיקום ,הסטודנטיות לוקחות חלק בכל שלבי הפיתוח ,החל מהאיפיון והתכנון ,דרך הפיתוח והבדיקות ,ועד להטמעה ותמיכה טכנית .

תיאור לקוח הקצה - הסמינר הישן ירושלים

תיאור המכון באופן כללי:

מכון ההכשרה והשתלמויות של סמינר הישן בירושלים מציע לבוגרות הסמינר לימודי המשך, הכוללים קורסים פרונטליים ומקוונים להרחבת אופקים, וכן השתלמויות המוכרות לגמולים, ל"אופק חדש" ול"עוז לתמורה".[26] במסגרת המכון ניתן ללמוד לקראת תואר שווה ערך: "אקוויוולנט לתואר בוגר" ("דרגה מס' 1") ו"אקוויוולנט לתואר מוסמך" ("דרגה מס' 2").[27]

בנוסף, מתקיימות במכון תוכניות לנשות חינוך ותיקות וגמלאיות.[26] (מתוך ויקיפדיה)

לפרטים נוספים על המכון: תיאור סמינר הישן - ויקיפדיה חרדית

האגף העיקרי שמולו הפרויקט מתנהל הוא:

האגף האדמיניסטרטיבי של המכון.

אנשי קשר:

חני לוין chlevin@mbj.org.il

חני פוליקמן ch-f@mbj.org.il

שולמית ברלין shulamitberlin@gmail.com

תיאור הפרויקט

תיאור כללי: הקמת מערכת full-stack כמפורט בשקופית מספר 4.

תפקיד הסטודנט :מפתחת תשתיות full-stack

מטרות ויעדים : לספק פלטפורמה נגישה לתפעול האגף באופן דיגיטלי.

מוצרים צפויים :

1. DB מרכזי לשמירת כל נתוני התלמידים והקורסים.
2. מסכים ליצירה, עדכון ומחיקה של תלמידים וקורסים.
   - 2.1 מסך לצפיית מערכת שעות של קורס בתצוגת לוח שנה + אפשרות להוסיף שיעור למערכת השעות הקיימת. וכן אופציות עריכת פרטי שיעור קיימים.
   - 2.2 מסך לצפייה בהיסטוריית נוכחות של תלמיד בתצוגת לוח שנה.
   - 2.3 מסך לצפייה בפרטי סטודנטים.
3. מסך לעדכון ודווח נוכחות
4. רכיב זיהוי פנים
5. מנגנון הפעלת פעולות אוטומטיות במערכת לפי הגדרת זמנים ותדירות של הצוות.
6. מסך הגדרת הגדרות מערכת
7. מסך הגדרת הרשאות

מטרות הפרויקט:

מטרות עיקריות:
שיפור נגישות המידע, הגברת מעורבות המשתמשים וייעול המשימות הניהוליות.

תרומה לחברת Diversitech:
לספק פרויקט מוצלח ורווחי, להציג את יכולת החברה בפיתוח אתרים ולשפר את תיק העבודות של החברה.

תרומה לסמינר הישן:
מעבר מניהול ידני לדיגטלי של כל נושאי התלמידות והקורסים. חיסכון באנשי צוות לניהול האגף. הרחבת פעולות המכון באמצעות המערכת שתמהר ותקצר תהליכים.

תוכן עניינים - תהליך העבודה:

שלב 1: היכרות עם החברה והפרויקט

שלב 2: אפיון ודרישות

שלב 3: תכנון

שלב 4: ביצוע ופיתוח

שלב 5: בדיקות ואבטחת איכות

שלב 6: הטמעה ותמיכה

שלב 1 : הכרת החברה והפרויקט

ראשית הגעתינו לחברה התקיים כנס הסברה להצגת הלקוחות והפרויקטים בחברה.

בכנס זה חולקנו לצוותות לפי פרויקטים ולאחר הכרות עם ראש הצוות התחילה העבודה במתודולוגיית אדג'ייל (Adgile).

שלב 2 : אפיון ודרישות

לאחר שיחות מול הלקוח, הופק אפיון עבור הפרויקט ע״י ה-CTO של Diversitech.

אנו קיבלנו את האפיון במסמכי word ב – google drive

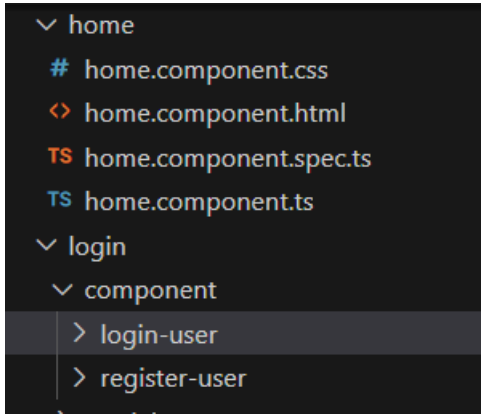
שלב 3 : תכנון

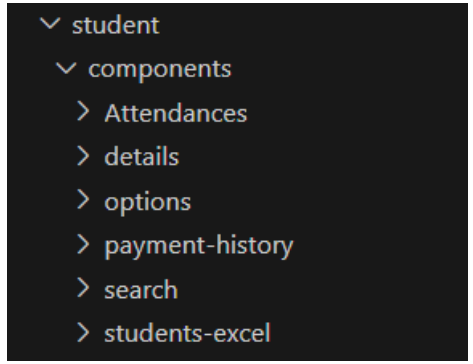
1. בתור התחלה ישבתי עם מפתחות נוספות לתכנן את המבנה הארכיטקטי של הפרויקט בצד קליינט.

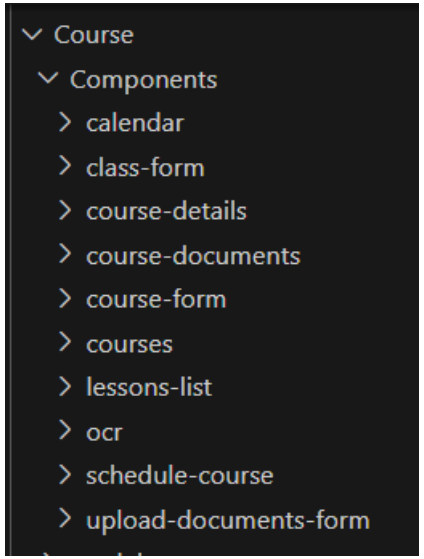
אציין שהשתמשנו בטכנולוגיית אנגולר לכתיבת צד קליינט ועל כן התכנון כלל:

- הבנת דרישות הלקוח מתוך מסמכי האיפיון לעומק.
- חלוקת מסכי האפליקציה למודולים וקומפוננטות ,כשהמטרה שכל מפתחת תהיה אחראית על feature אחד.

פירוט החלוקה שביצענו:







הראש צוות תכנן את ארכיטקטורת צד הסרבר באמצעות סרוויסים שונים.

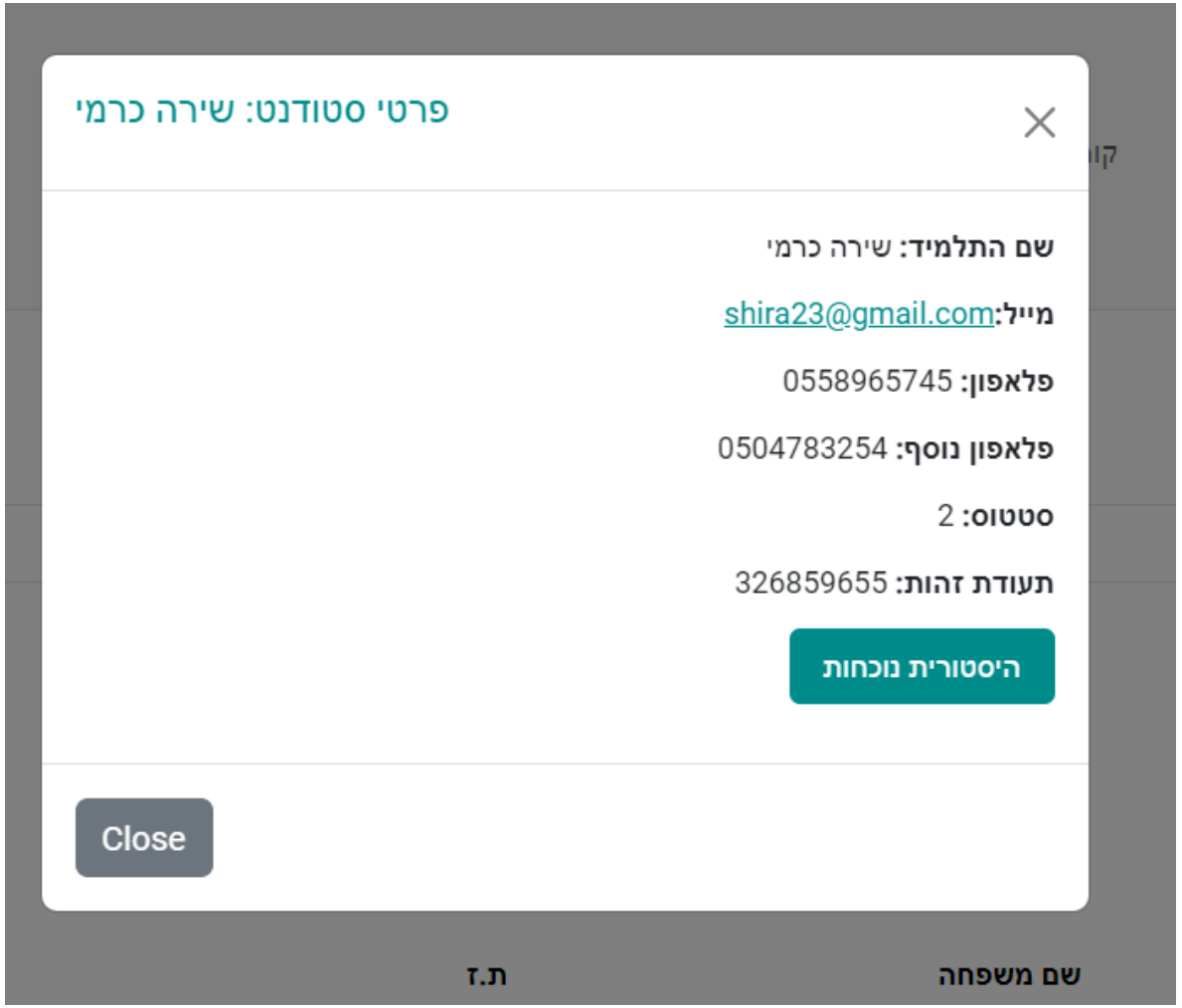
שלב ד' : ביצוע ופיתוח

במהלך הפיתוח פיתחתי באזורים שונים בכל רחבי האפליקציה ,בשל כך הטכנולוגיות והביצועים שאפרט להלן הינן מגוונות הן מבחינת סוגיהן והן מבחינת רמתן :

#	החלק הסופי במוצר:	הטכנולוגיה	השפה	תיאור המשימה
1	מסך פרטי תלמיד	Angular	TS	במשימה זו פיתחתי מודל ב- Angular שמציג פרטי סטודנט בצורה מסודרת בחלון מודל. המודל כולל פרטים כגון שם, מייל, פלאפון, סטטוס ותעודת זהות של הסטודנט. כמו כן, ניתן לנווט להיסטוריית נוכחות של הסטודנט דרך קישור בתוך המודל. **עיקרי המשימה:** **רכיב Angular:** רכיב DetailsComponent המציג את פרטי הסטודנט במודל. קבלת פרטי הסטודנט מהשירות StudentService באמצעות המתודה getStudentDetails. **HTML:** שימוש ב- ngIfכדי לוודא שהפרטים יוצגו רק כאשר קיימים פרטי סטודנט. הצגת פרטי הסטודנט בשדות הרלוונטיים, כולל קישורים למייל והיסטוריית נוכחות. **Bootstrap:** שימוש ב- Bootstrap להפקת חלון המודל ולעיצובו. **תכנות:** הגדרת תלויות ורכיבים (Input, Output, ViewChild)לניהול התצוגה והאינטראקציה עם המודל. ניהול הפתיחה והסגירה של המודל באמצעות Bootstrap Modal.

צילומי מסך של המשימה

```
ngAfterViewInit(): void {
  if (this.studentModal) {
    const modalElement = this.studentModal.nativeElement;
    this.modalInstance = new Modal(modalElement);
    modalElement.addEventListener('hidden.bs.modal', () => {
      this.modalClose.emit();
    });
  }
}

getStudentDetails(id: string): void {
  this.studentService.getStudentById(id).subscribe(student => {
    this.currentStudent = student;
    this.showModal();
  });
}

navigateToAttendance(): void {
  this.router.navigate([`${this.studentUrl}/attendance`]);

}

private showModal(): void {
  if (this.modalInstance) {
    this.modalInstance.show();
  }
}
```



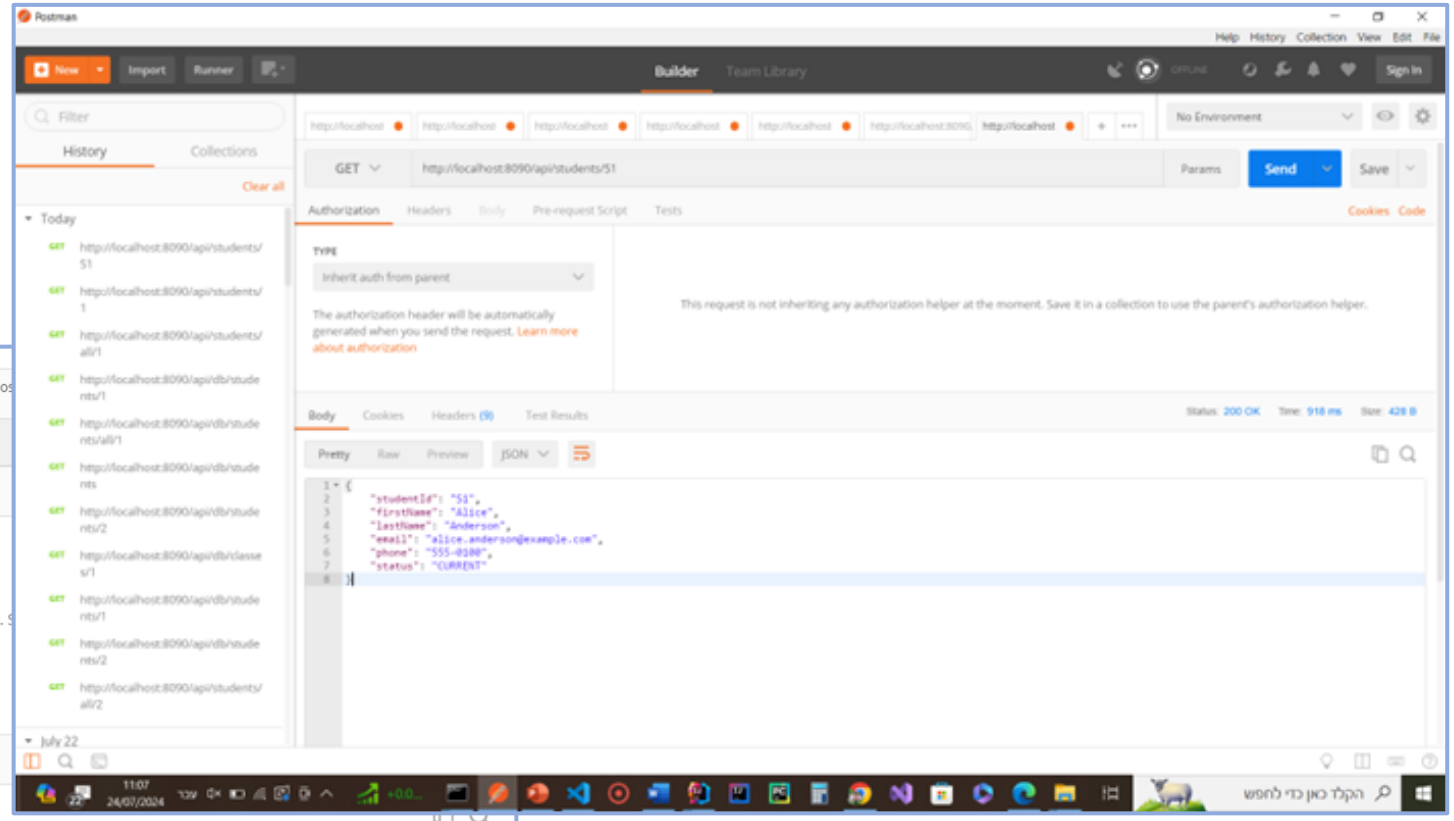
צילומי מסך של המשימה

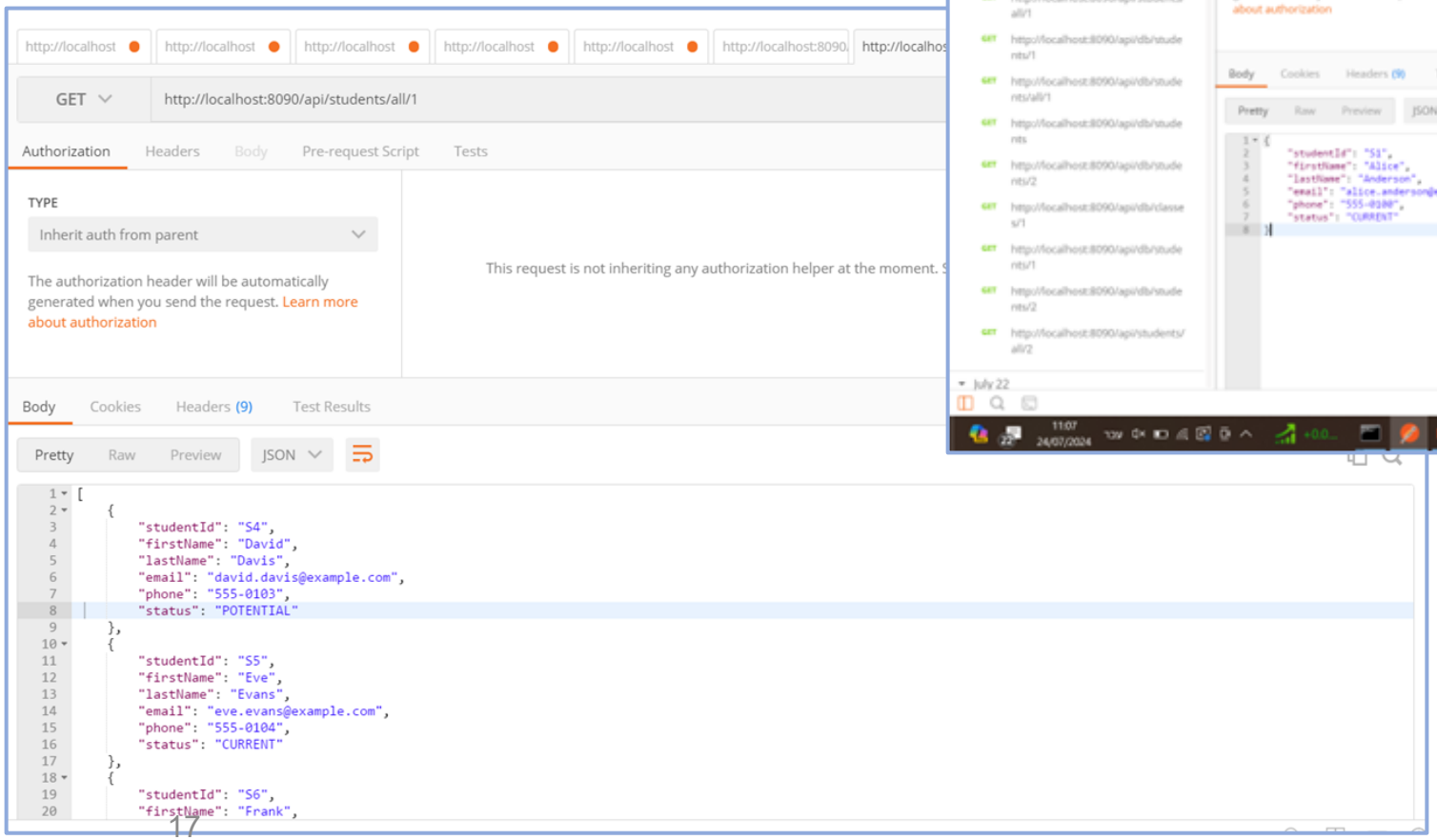
```
@Component({
  selector: 'app-details',
  templateUrl: './details.component.html',
  styleUrls: ['./details.component.css']
})
export class DetailsComponent implements OnChanges, AfterViewInit {
  @Input() currentStudent: Student | null = null;
  @Output() modalClose = new EventEmitter<void>();
  @ViewChild('studentModal', { static: true }) studentModal!: ElementRef;
  private studentUrl = `${environment.baseURL}`;
  private modalInstance: Modal | null = null;

  constructor(private studentService: StudentService,private router: Router) {

  }

  ngOnChanges(changes: SimpleChanges): void {
    if (changes['currentStudent'] && this.currentStudent) {
      this.showModal();
    }
  }
```

```
  <button
    type="button"
    class="btn-close"
    data-bs-dismiss="modal"
    aria-label="Close"
  ></button>
</div>
<div class="modal-body">
  <div *ngIf="currentStudent">
    <p><strong>שם התלמיד:</strong> {{ currentStudent.firstName }} {{ currentStudent.lastName }}</p>
    <p><strong>מייל:</strong> <a  class="linkMail" [href]="'mailto:' + currentStudent.email">{{ currentStudent.email }}</a></p>
    <p><strong>פלאפון:</strong> {{ currentStudent.phone1 }}</p>
    <p><strong>פלאפון נוסף:</strong> {{ currentStudent.phone2 }}</p>
    <p><strong>סטטוס:</strong> {{ currentStudent.status }}</p>
    <p><strong>תעודת זהות:</strong> {{ currentStudent.studentId }}</p>

    <p>
      <a [routerLink]="['attendance', currentStudent.studentId]" class="router-link" data-bs-dismiss="modal">
        היסטורית נוכחות
      </a>
    </p>
  </div>
</div>
<div class="modal-footer">
  <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">
    Close
  </button>
```

#	החלק הסופי במוצר:	הטכנולוגיה	השפה	תיאור המשימה
2	**: החלק המעבד את נתוני המסד של פרטי התלמיד, בצד הסרבר**	Java spring boot	java	במשימה זו פיתחתי שירות REST באמצעות Spring Bootלניהול מידע על סטודנטים. השירות כולל פעולות שליפה, הוספה, עדכון ומחיקה של פרטי סטודנט באמצעות RestTemplate. השירות מאפשר עבודה מול בסיס נתונים חיצוני ומספק נקודות קצה לממשק לקוח. **עיקרי המשימה:** **שירות סטודנטים**: פיתוח מחלקת Service בשם StudentService המתקשרת עם בסיס הנתונים החיצוני באמצעות RestTemplate. הגדרת נתיבי ה- URL ומסלולי ה- API בקובץ הקונפיגורציה. **אנוטציות ו-MVC:** שימוש באנוטציות @ Service, @Autowired ו-@ Value כדי להגדיר שירותים ולהזריק תלויות. שימוש ב-@ RestController, @RequestMapping ו- @ CrossOrigin במחלקת StudentController לניהול הבקשות ל-API. **קונפיגורציה**: יצירת מחלקת קונפיגורציה AppConfig להגדרת RestTemplate והגדרות . CORS**מחלקת ישות**: יצירת מחלקת Entity בשם Student הכוללת פרטים כמו מזהה, שם פרטי, שם משפחה, מייל, טלפונים, חוב וסטטוס.

צילומי מסך של המשימה





צילומי מסך של המשימה

```java
private final StudentService studentService;

@Autowired
public StudentController(StudentService studentService) { this.studentService = studentService; }

@GetMapping("/all/{studentId}")
public ResponseEntity<List<Student>> getAllStudents(@PathVariable int stude
    return studentService.getAllStudents(studentId);
}
@GetMapping("/{studentId}")
public ResponseEntity<Student> getStudentById(@PathVariable String student
    return studentService.getStudentById(studentId);
}
@GetMapping("/byCourse/{courseId}")
public ResponseEntity<List<Student>> getStudentsByCourseId(@PathVariable in
    System.out.println("courseId:" + courseId);
    return studentService.getStudentsByCourseId(courseId);
}
@PostMapping()
public void addStudent(@RequestBody Student student) {
    System.out.println("student: "+student.toString());
    studentService.addStudent(student);
}
@PutMapping("/{studentId}")
public void updateStudent(@PathVariable String studentId, @RequestBody Stu
    studentService.updateStudent(studentId, student);
}
@DeleteMapping("/{id}")
public void deleteStudent(@PathVariable String id) { studentService.delete
```

```java
@Value("${external-apis.db-connector.paths.getStudentsByCourseId}")
private String studentsByCourseIdPath;
@Value("${external-apis.db-connector.paths.getStudentById}")
private String studentByIdPath;
@Value("${external-apis.db-connector.paths.addStudent}")
private String addStudentPath;
@Value("${external-apis.db-connector.paths.updateStudent}")
private String updateStudentPath;
@Value("${external-apis.db-connector.paths.deleteStudent}")
private String deleteStudentPath;

public ResponseEntity<List<Student>> getAllStudents(int studentId) {
    String url = getUrlValueFromConfig(dbConnectorHost, allStudentPath, List.of(String.valueOf(studentId)));
    HttpHeaders headers = new HttpHeaders();
    HttpEntity<List<Student>> entity = new HttpEntity<>(headers);
    ResponseEntity<List<Student>> response = restTemplate.exchange(url, HttpMethod.GET, entity, new ParameterizedTypeRefer
    });
    return response;
}
public ResponseEntity<List<Student>> getStudentsByCourseId(int courseId) {
    String url = UriComponentsBuilder.fromHttpUrl(dbConnectorHost)
            .path(studentsByCourseIdPath)
            .buildAndExpand(courseId)
            .toUriString();

    HttpHeaders headers = new HttpHeaders();
    HttpEntity<Student> entity = new HttpEntity<>(headers);
    ResponseEntity<List<Student>> response = restTemplate.exchange(url, HttpMethod.GET, entity, new ParameterizedTypeRefer
    return response;
}
```

צילומי מסך של המשימה

```java
import ...

@Configuration
public class AppConfig implements WebMvcConfigurer {

    @Bean
    public RestTemplate restTemplate() { return new RestTemplate(); }

    @Bean
    public StudentService studentService(RestTemplate restTemplate) { return new StudentService(restTemplate); }

    @Override
    public void addCorsMappings(CorsRegistry registry) {
        registry.addMapping( pathPattern: "/**")
                .allowedOrigins("http://localhost:4200")
                .allowedMethods("GET", "POST", "PUT", "DELETE", "OPTIONS")
                .allowedHeaders("*")
                .allowCredentials(true);
    }
}
```

#	החלק הסופי במוצר:	הטכנולוגיה	השפה	תיאור המשימה
3	מוצר פעיל בענן render - ייסדתי תשתית פריסה בענן לזמן הפיתוח.	Docker render	--	1. ביצעתי מחקר מקדים במסגרת תכנון הפריסה. המטרה הייתה למצוא תשתית מרכזית שנוכל להסתנכרן בה במשך העבודה. בדקתי את כל האפשרויות בעננים של החברות הגדולות: Amazon -AWS - Microsoft - Azure cloud ,Google - GCP cloud, לבסוף בשל מגבלות פיננסיות החלטתי ללכת על פתרון של ענן זול יותר- render 2.חקרתי את התהליכים הנדרשים לפריסה כמו יצירת dockerfile וכו' ,התהליכים מפורטים במסכים הבאים (המסכים הבאים הוצגו לצוות בשביל למידת הנושא ע"י מסירת session)

צילומי מסך של המשימה

איך מעלים פרויקט java spring boot

ל-render.

צילומי מסך של המשימה

1. ליצור docker image מהפרויקט

מכיוון שהענן שאנו משתמשות בו לא תומך בשפת java צריך לארוז את הפרויקט ל-docker image

נעשה זאת כך:

1. להוריד docker desktop -למי שאין.
2. להוסיף dockerfile לפרויקט -

הקובץ הזה צריך להקרא Dockerfile

בלי סיומת , הקובץ צריך להמצא בתיקיית

השורש של הפרויקט (בתיקייה בה נמצא הקובץ pom.xml)

דוגמא לdockerfile

```
FROM maven:3.8.4-openjdk-17 AS build
WORKDIR /app
COPY pom.xml .
COPY src ./src
RUN mvn clean package -DskipTests

#שלב 2: יצירת התמונה להרצה
FROM openjdk:17-jdk-slim
WORKDIR /app
COPY --from=build /app/target/*.jar app.jar
ENTRYPOINT ["java", "-jar", "app.jar"]
```

צילומי מסך של המשימה

לפני הפקודות הבאות צריך לפתוח את ה-docker desktop (שיהיה במצב running)

3. לבנות dockerfile ע"י הפקודה docker build
דוגמא:

```
docker build -t user-service .
```

לאחר מכן תוכלו לראות את הimage ב-docker desktop בקטגוריית images.

4.השלב הזה הוא רק כדי לבדוק אם עובד הdocker -image.
להריץ את הdocker-image ע"י הפקודה :

```
docker run -p 8080:8080 user-service
```

צילומי מסך של המשימה

2.להעלות את הפרויקט לrender

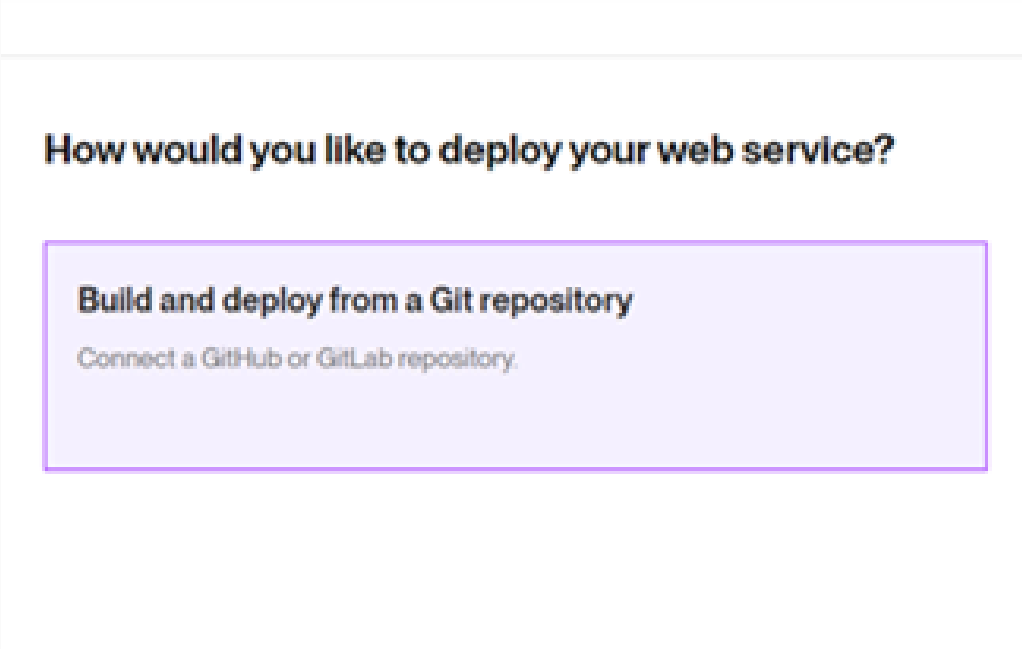
דבר ראשון צריך להעלות את הפרויקט עם ה-dockerfile לגיט.

ואז נעלה לrender.

1.פותחים את render בגוגל ופותחים חשבון.

2.בוחרים new web service מתוך כל המוצרי

3.בוחרים להעלות מתוך repository



צילומי מסך של המשימה

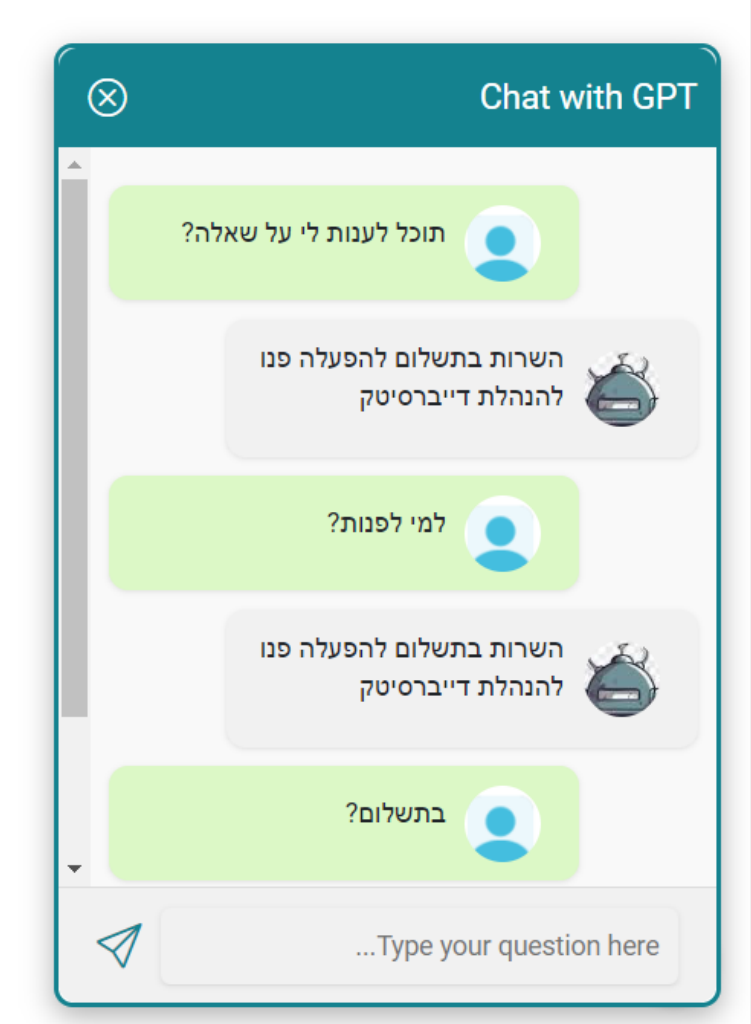
4. להתחבר לפרויקט ע"י url לrepo בgitub.

5.להשלים פרטים על הפרויקט:

name: השם של הפרויקט-איך שאת רוצה
language: docker
branch:איזה שצריך..
region:עדיף סינגפור כי הכי קרוב
root directory:התיקייה בה נמצא הקובץ הpom.xml .
build command:הפקודה docker build שכתבתם.
start command:הפקודה docker run שכתבתם.

לבחור כמובן את התוכנית free.
ואז הוא יתחיל להריץ את הפרויקט ,זה יכול לקחת די הרבה זמן וכאשר הוא גומר- מצב live
,תוכלו להעתיק את הקישור מהפינ
ולפתוח את הפרויקט בקלות בכל ז

https://old-seminar-1-front-eu8g.onrender.com

#	החלק הסופי במוצר:	הטכנולוגיה	השפה	תיאור המשימה
4	chatBot 	HashiCorp Vault, OpenAI-AP Flask SocketIO, Flask	python	**התקנת הספריות הנדרשות**: תחילה התקנתי את הספריות hvac (לתקשורת עם HashiCorp Vault) , flask (ליצירת השרת), flask-socketio לתמיכה בתקשורת בזמן אמת עם הלקוח), ו- openai (לשימוש ב- API של OpenAI). **ייבוא הספריות**: ייבאת את הספריות הנדרשות בקוד שלך, כולל ספריות נוספות כמו logging לצורך ניהול והצגת לוגים, ו- datetime לעבודה עם תאריכים ושעות. **הגדרת אפליקציית Flask**: יצרתי אובייקט Flask, שהפך את הקוד שלי לשרת אינטרנטי שמאפשר לטפל בבקשות מהלקוח. **הגדרת SocketIO** : השתמשתי ב- Flask-SocketIOכדי להוסיף תמיכה בתקשורת בזמן אמת בין הלקוח לשרת, מה שמאפשר לך לטפל בהודעות נכנסות ולהגיב אליהן בזמן אמת. **הגדרת לוגים**: קבעת את רמת הלוגים של השרת כדי להקל על מעקב אחרי פעולות והודעות שגיאה. **שליפת מפתח ה- API**: הגדרת פונקציה שמשתמשת ב- HVAC (הלקוח של Vault) HashiCorp לשלוף את מפתח ה- API של OpenAIמאחסון הסודות של Vault. זה מאפשר לך להבטיח שהמפתח נשמר בצורה בטוחה ומאובטחת.

#	החלק הסופי במוצר:	הטכנולוגיה	השפה	תיאור המשימה
4	chatBot	HashiCorp Vault, OpenAI-API, Flask SocketIO, Flask	python	**המרת שאילתות**: פיתחת פונקציה הממירה שאילתות בשפה טבעית לשאילתות SQLבאמצעות המודל של OpenAI. הפונקציה שולחת את השאילתה ל- API של OpenAI עם פרומפט שמתאר את הפעולה הנדרשת. **ניהול שגיאות**: הוספת טיפול בשגיאות במקרה של חריגה מהמכסה שנקבעה על ידי OpenAI, והצגת הודעה מתאימה במקרה כזה. **ביצוע שאילתות**: פיתחת פונקציה המבצעת את השאילתות שנוצרות ומחזירה תוצאות מדוגמות (למשל, רשימת קורסים) לצורך דוגמה. בשלב זה, הפונקציה מתמקדת בנתונים בדוגמה, אך ניתן לשדרג אותה להתחברות לבסיס נתונים אמיתי. **קבלת תגובות מ- ChatGPT :** יצרתי פונקציה המתקבלת מהמודל של OpenAI תגובה מותאמת אישית המבוססת על המידע שהתקבל מהשאילתות. **הגדרת טיפול בהודעות**: כתבת פונקציה המאזינה להודעות מהלקוח ומביאה את התשובה המתאימה על פי השאילתה שהתקבלה. הפונקציה משדרת את התגובה ללקוח דרך SocketIO. **הפעלה של השרת**: הפעלת את השרת כדי שהאפליקציה תהיה זמינה ותגיב לבקשות בזמן אמת.

צילומי מסך של המשימה

```python
def natural_language_to_sql(prompt):
    conversion_prompt = f"Convert the following natural language query to an SQL query: '{prompt}'"
    try:
        # קריאה ל-OpenAI עם המפתח שנקרא מ-Vault
        response = openai.ChatCompletion.create(
            model="gpt-3.5-turbo",
            messages=[
                {"role": "system", "content": "You are an assistant that converts natural language questions into SQL queries."},
                {"role": "user", "content": conversion_prompt},
            ],
            max_tokens=150,
            api_key=fetch_api_key()  # שימוש במפתח ה-API מ-Vault
        )
        sql_query = response['choices'][0]['message']['content'].strip()
        return sql_query
    except openai.error.RateLimitError:
        logging.error("Exceeded quota :שגיאת מערכת")
        return "להפעלה נא פנו להנהלת דיברסיטק"
```

```python
def fetch_api_key():
    client = hvac.Client(url='https://valut-service.onrender.com', token='root')
    secret = client.secrets.kv.v2.read_secret_version(path='open_ai_api_key')
    return secret['data']['data']['open_ai']
```

צילומי מסך של המשימה

```python
def ask_chatgpt(prompt):
    sql_query = natural_language_to_sql(prompt)
    if(sql_query== "להפעלה נא פנו להנהלת דיברסיטק"):
        return "להפעלה נא פנו להנהלת דיברסיטק"
    results = query_db(sql_query)
    formatted_data = '\n'.join([str(res) for res in results])
    chatgpt_prompt = f"User asked: {prompt}\nHere is the relevant data:\n{formatted_data}\nProvide a response to the user bas
    chatgpt_response = get_chatgpt_response(chatgpt_prompt)
    return chatgpt_response
```

```python
def get_chatgpt_response(prompt):
    response = openai.Completion.create(
        engine="gpt-3.5-turbo",
        prompt=prompt,
        max_tokens=150
    )
    return response.choices[0].text.strip()
```

```python
@socketio.on('message')
def handle_message(message):
    logging.info(f'Received message: {message}')
    response = ask_chatgpt(message) + " " + message + "!!!!!!!!!! "
    emit( event: 'response', *args: response)

if __name__ == '__main__':
    socketio.run(app, debug=True, allow_unsafe_werkzeug=True)
```

#	החלק הסופי במוצר:	הטכנולוגיה	השפה	תיאור המשימה
5	מקום אחסון נתונים רגישים בענן render .	render, HashiCorp Vault	--	1. התקנת HashiCorp Vault : שימוש ב- Dockerfile להתקנת Vault בענן render. קביעת תצורת ה- Dockerfile כך שיתאים לצרכים של הפרויקט. 2. הגדרת תצורת Vault : יצירת קובץ תצורה עבור Vaultעם הפרטים הנדרשים. שימוש בפקודת vault server -config=<config-file>להרצת השרת. 3. שמירת מפתחות OpenAI בקוד Python שימוש ב- hvac להתחברות ל-Vault 4. יצירת קוד Python שמתחבר ל- Vault ומשיג את מפתחות ה openAi – api.

צילומי מסך של המשימה

api_key_openai

Secret | Metadata | Paths | Version History

JSON | Delete | Destroy | Copy | Version 1 | Create new version +

Key	Value	Version 1 created Aug 06, 2024 11:00 AM
open_ai	■■■■■■■■■■■■	

valut-service-ok19.onrender.com/ui/vault/auth?with=token

valut-service-ok19.onrender.com/ui/vault/secrets/secret/kv/api_key_openai/details?version=

Sign in to Vault

Method

Token

Token

••••

Sign in

Contact your administrator for login credentials.

שלב 5 : בדיקות אבטחה ואיכות

כאשר סיימתי לפתח, הרצתי את האתר בסביבות שונות :

1. לוקאלית
2. על הענן render

בדקתי שהמסכים עובדים כראוי, במידה והמשימה הייתה על צד קליינט, ובמידה והמשימה הייתה על צד סרבר, בדקתי שה apis מגיבים כראוי באמצעות postman.

שלב 6 : הטמעה ותמיכה

בסיום כל ספרינט נערך דמו ללקוח, בסביבת ענן.

אנשי הקשר של הלקוח קבלו את הלינק לאתר להכרות עם המערכת ונתינת פידבק לשינויים נדרשים בספרינטים הבאים.

בעתיד הצוותות הבאים יפעילו את המערכת בענן חזק יותר ויטמיעו את המערכת באגף לשימוש יומיומי.

סיכום ומסקנות :

סיכום כללי :
הפרויקט נחל הצלחה בזכות עבודה משותפת של הצוות ,שימוש בטכנולוגיות מתקדמות וארכיטקטורת מיקרו-סרוויסים מבוזרת .המערכת מאפשרת ניהול קורסים ,שיעורים ומסמכים בצורה יעילה ונוחה ,תוך שמירה על חוויית משתמש גבוהה .הפרויקט הושק בהצלחה לענן וממשיך להיות מתוחזק ומשודרג באופן שוטף.

מסקנות אישיות:
כמפורט בפרק שמונה- צברתי ידע רב וחדש בטכנולוגיות חדשניות וחכמות, למדתי נהלי עבודה במתודולוגיית אדג'ייל(Agile), התמודדתי עם אתגרים בפיתוח ופתרתי באגים שונים במהלך הפיתוח.

תרומה לפרויקט ולחברה:
מעבר למשימות שלי החשובות בפיתוח , הכוונתי את הצוות לנראות נכונה של האתר, בדגש להנפיק מערכת עם חוית משתמש גבוהה וידידותית יותר

נספחים :

הסבר מעמיק על הפונקציה getUrlValueFromConfig :

```
while (matcher.find()) {
String replacement = uniqueVals.size() > index ? uniqueVals.get(index) : matcher.g
matcher.appendReplacement(sb, replacement);
index++;
}
```

`matcher.find()`: מוצא את ההתרחשות הבאה של הביטוי הרגולרי ב-URL.

`String replacement = uniqueVals.size() > index ? uniqueVals.get(index) : matcher.group(0);`

קובע את מחרוזת ההחלפה:

אם `index` נמצא בתוך גבולות `uniqueVals`, הוא מקבל את הערך באינדקס הנוכחי.

אחרת, שומר על המקום השמור המקורי (למשל, `{param}`) אם אין ערך תואם זמין.

`matcher.appendReplacement(sb, replacement);`: מוסיף את החלק של ה-URL עד להתאמה ואת מחרוזת ההחלפה ל-`StringBuffer`.

`++index`: מגדיל את האינדקס לשימוש בערך הבא מרשימת `uniqueVals` עבור המקום השמור הבא.

שילוב כתובת הבסיס והנתיב

```
Copy code  java
String url = baseUrl + path;
```

מחבר את `baseUrl` ואת `path` ליצירת מחרוזת כתובת ה-URL המלאה.

קומפילציה של ביטוי רגולרי

```
Copy code  java
Pattern pattern = Pattern.compile("\\{([^}]+)\\}");
```

מקמפל את הביטוי הרגולרי `\\{([^}]+)\\}` לאובייקט `Pattern`.

ביטוי זה תואם כל תת-מחרוזת שמתחילה ב-`{`, seguida בכמה תווים שאינם `}`, ומסתיימת ב-`}`. לדוגמה, `{param1}`.

יצירת Matcher

```
Copy code  java
Matcher matcher = pattern.matcher(url);
```

יוצר אובייקט `Matcher` שיבצע חיפושים עבור התופסות של הביטוי הרגולרי בכתובת ה-URL.

אתחול StringBuffer

```
Copy code  java
StringBuffer sb = new StringBuffer();
```

מאתחל אובייקט `StringBuffer`, שישמש לבניית מחרוזת התוצאה הסופית. `StringBuffer` משמש למניפולציה יעילה של מחרוזות.

נספחים :

מצורפים הנספחים הבאים :

- נספח א' טופס הצעה לפרויקט בתעשייה-פרקטיקום .
- אישור מעסיק לפרקטיקום.