**דרישות לפרויקט גמר בקורס נושאים מתקדמים באפליקציות אינטרנטיות**

**כללי:** עליכם לממש מערכת אינטרנטית תוך שילוב הטכנולוגיות המופיעות בסעיף הדרישות. המערכת יכולה להיות חנות אינטרנטית, רשת חברתית, אך תוכלו להציע רעיון אחר כל עוד הוא תומך במסמך הדרישות.

**קבוצות:** ניתן לעבוד בקבוצות של 3- 4 סטודנטים בלבד. עם זאת, כל אחד מחברי הקבוצה חייב להכיר את כל הקוד שנכתב ועליו להיות מסוגל לענות על שאלות מכל חלקי הקוד במהלך ההגנה. כמו כן, על הסטודנטים להשתמש, בזמן פיתוח הפרויקט, במערכת לניהול גרסאות (**Git**) ובמועד ההגנה להציג את **ההתנהלות (וההשקעה)** של **כל אחד** מחברי הקבוצה. ההגנה תתקיים במהלך השיעור האחרון של הקורס בסמסטר ו/או במועדים נוספים שיפורסמו בהמשך.

**דרישות:**

1. המערכת תתוכנן ותמומש באמצעות MEAN Stack – כלומר :
   1. תשתית המערכת תבוסס Node.JS בצד השרת
   2. שימוש ב Express.JS web framework בצד השרת
   3. אחסון ושליפת הנתונים מתוך Mongo DB
   4. תכנון בצד הלקוח מבוסס MVW על בסיס Angular
2. המערכת תתמוך ב HTML5 וב CSS3
3. האפליקציה תתבסס על עקרונות ה Single Page Application – משמע תתבצע טעינה של הדף הראשי פעם אחת בלבד וכל העדכונים הנוספים יעבדו על בסיס AJAX ו/או WebSockets
4. תהיה הפרדה ברורה בין ה View, ה Controller וה Model.
5. על המערכת לתמוך בלפחות 3 מודולים שונים של Angular. (למשל בנושא לדוגמא של מערכת הודעות: מודול אחד על התצוגה, מודול שני – ממשק מנהל, מודול שלישי – סטטיסטיקות על הודעות)

רשימת Groups,Checklists,Statistics

1. על שניים מהמודולים, המערכת צריכה לתמוך ב Create (יצירת אובייקט חדש), Update (עדכון תוכן האובייקט – למשל עדכון פרטי המוצר הנמכר בחנות), Delete (מחיקת אובייקט – למשל מחיקת ספק), List (רשימת אובייקטים: למשל רשימת הלקוחות), Search (חיפוש אובייקט על פי קטגוריות שונות – למשל מצא את הלקוחות הגרים באיזור המרכז).

Groups,Checklists – פילטרים לפי שם, תיאור

1. המערכת תכיל לפחות 3 סוגי מסמכים ב MongoDB אשר קשורים ביניהם באמצעות reference

Groups,Checklists,

1. המערכת תתמוך בלפחות 2 שאילתות חיפוש המאפשרות ללקוח להגדיר פרמטרים (לפחות 3 פרמטרים) לחיפוש באמצעות ממשק המשתמש (כמו למשל חיפוש טלויזיה בזאפ על פי: גודל מסך, רזולוציה, משקל וכד')

\SmartList\backend\routes\groups.js

router.get("", (req, res, next) => {

return Group.find((filter!=""?JSON.parse(filter):null)).count();

1. המערכת תתמוך בלפחות שאילתה אחת המבצעת פעולת Group By ב MongoDB

SmartList\src\app\components\two-ways-bindings\two-ways-bindings.component.ts

constructor(public **groupsService**:GroupsService, public socketService:SocketService ) {

this.counterChange = new EventEmitter();

ngOnInit() {

//Load default value - current actual count value

this.**groupsService**.getGroups(this.groupsPerPage, this.currentPage);

this.groupsSub = this.groupsService

.getGroupUpdateListener()

.subscribe((groupData: {groups: Group[], groupCount: number}) => {

this.GroupsCount = groupData.groupCount;

});

}

SmartList\src\app\services\groups.service.ts

SmartList\src\app\services\groups-hll.service.ts

SmartList\backend\routes\groups.checklists.js

router.get("", (req, res, next) => {

Group.aggregate([

{

$lookup:

{

from: "checklists",

localField: "\_id",

foreignField: "group",

as: "Checklists"

}

},

{

$project: {

\_id: 0,

title: 1,

ChecklistsCount: { $size: '$Checklists' },

}

\SmartList\src\app\services\groups-hll.service.ts

getGroupChecklistsCountTop10() {

return this.http.get<{

}

1. המערכת תתמוך בלפחות שאילתה אחת המבצעת פעולת Map-Reduce ב MongoDB
2. על המערכת להכיל לפחות מרכיב אחד המממש Two-Way-Binding בין השרת לבין ה Browser תוך שימוש ב WebSockets כך שגם השרת יכול ליזום עדכונים מול ה Browser. למשל במסך ה Admin יוצג שדה באתר המראה את מספר הלקוחות המחוברים באותו רגע (כל משתמש חדש שמתווסף יגרום לשרת לדווח על כך לכל המשתמשים שהם Admin המחוברים באמצעות WebSockets ללא שימוש ב Polling, וכל אחד מה Admins יוכל לבחור לעדכן את הנתון הזה, שיגרום לעדכון מיידי הן בשרת והן אצל שאר ה Admin-ים המחוברים באותו הרגע. דוגמא אחרת – אתר חדשות שמראה Live הודעות שהגיעו באותו הרגע ללא צורך ב Refresh/polling מה Browser, וה Admin יכול גם לערוך חדשות קיימות מה שיגרום עדכון מיידי בשרת, ב DB ואצל יתר הלקוחות שמחוברים כרגע לאתר). את רכיב זה יש לממש כ Custom Binding ב Angular.

SmartList\server.js

var io = require('socket.io').listen(server);

global.getIO = io;

  client.on('AddGroup', function(GroupCount) {

console.log('server event: get message AddGroup ' + GroupCount);

client.broadcast.emit('GroupAdded', GroupCount);

});

SmartList\backend\routes\groups.js

//Add new

router.post(

/////////////////

//Socket smits all client with current count of groups

///////////////

//execute cocket emmiter for notifying all observerable.

Group.estimatedDocumentCount((err,count)=>{

console.log("Group.count" + count+1);

global.getIO.sockets.emit('GroupAdded', count+1);

});

SmartList\src\app\services\socket.service.ts

@Injectable({ providedIn: "root" })

export class SocketService {

private url = 'http://localhost:3000';

public getMessages = () => {

console.log("getMessages");

return Observable.create((observer) => {

SocketService.socket.on('GroupAdded', (message) => {

observer.next(message);

});

});

}

SmartList\src\app\components\groups\group-list\group-list.component.html

<two-ways-bindings [(counter)]="GroupsCount" [groupsPerPage]="groupsPerPage" [currentPage]="currentPage" ></two-ways-bindings>

SmartList\src\app\components\two-ways-bindings\two-ways-bindings.component.ts

constructor(public groupsService:GroupsService, public socketService:SocketService ) {

this.counterChange = new EventEmitter();

ngOnInit() {

//register getMessages events of socket

this.socketService.getMessages()

.subscribe((message: string) => {

console.log("subscribe");

this.GroupsCount = parseInt(message);

});

}

1. על המערכת להכיל לפחות מרכיב אחד העושה שימוש באלגוריתם CMS (Count Min Sketch) המאפשר ספירה סטטיסטית של תופעות (למשל על מנת לזהות כמה אנשים גלשו במערכת עד כה אשר מכילים תכונות לפי הגולש האחרון ומהי התכונה הבולטת ביותר כמשותף) או באלגוריתם HLL (Hyper Log Log) המבצע Distinct Count (למשל מספר המילים החדשות שכתבו בצ'ט עד כה, שמות חדשים שנראו עד כה). את התוצאה של תצוגת התוצר של אחד מהאלגוריתמים יש לפרמט באמצעות שימוש ב Custom Pipe ב Angular.

**HLL (Hyper Log Log)**

SmartList\backend\routes\groups.HLL.js

router.get("", (req, res, next) => {

var hll = require('hll');

/////////////////////

//Count distinct Checklist description's words

////////////////////

.

.

.

res.status(200).json(h.estimate() );

SmartList\src\app\Pipes\HLLPipe.ts

transform(value: string): string {

let newStr: string = "Count Distinct of Checklist's words in Description:" + value + ", (by Checklist's Description)";

return newStr;

\SmartList\src\app\components\groups\groups-hll\groups-hll.component.ts

ngOnInit() {

this.groupsService.getGroupHLL().subscribe(groupData => {

console.log(JSON.stringify(groupData) );

this.GroupsCount =parseInt(JSON.stringify(groupData) );

});

1. על המערכת להכיל לפחותCustom Directive אחד המציג נתונים בגרף תוך שימוש בחבילה D3.js
2. על המערכת להתבסס על נתונים שנשלפו מאתרים/מערכות אמיתיות באמצעות Scraping שייכתבו לתוך MongoDB – המכיל ביום ההגשה/הגנה לפחות 100 אובייקטים שנוצרו אוטומאטית כתוצאה מתהליך ה Scraping (ולא ידנית. ניתן להוסיף ידנית בנוסף ל 100 האלה)

\SmartList\backend\dataFromScraper\bakingmad2.json

\SmartList\backend\models\initDB.js

///////////////////////////////////////////////////

// resetdb.resetDB();

///////////////////////////////////////////////////

1. על המערכת לתמוך בתכונה שתאפשר חיפוש יעיל על הנתונים שנסרקו ב Scraper ושהוכנסו ל MongoDB באמצעות אלגוריתם Aho-Corasick שיבצע חיפוש מקביל על מספר מונחים שהלקוח יוכל להזין בחיפוש
2. באחד מדפי המערכת תוצג מפה מבוססת Google Maps/Bing Maps ובה מסומנות כתובות שנקראו מבסיס הנתונים (למשל רשימת סניפים של רשת החנויות)

SmartList\backend\routes\users.js

//Add new

router.post(

const newuser = new User({

username: req.body.email,

email: req.body.email,

address: req.body.address,

**location: [req.body.longitude, req.body.latitude ]**

});

**בהצלחה!**