**המחלקה למדעי המחשב**

**פרויקט גמר**

**Advanced Shift Management**

**מערכת לניהול משמרות**



**מנחה אקדמי:** פרופסור אסתר דוד

**תאריך הגשה: 23.2.22**

**מגישים:**

1. מגן חג'בי - 312191323
2. בר חג'בי - 312191331
3. שיאור בסט – 208054390
4. נתנאל ברזילי

**תוכן עניינים**

[1. מסמך דרישות 2](#_Toc48559209)

[1.1. הצגת הבעיה: 2](#_Toc48559210)

[1.2. רקע תיאורטי: 2](#_Toc48559211)

[1.3. דרישות פונקציונליות: 3](#_Toc48559212)

[1.4. דרישות לא פונקציונליות : 3](#_Toc48559213)

[1.5. לוח זמנים לביצוע הפרויקט וחלוקת העבודה: 3](#_Toc48559214)

[1.6. יתרונות המערכת : 4](#_Toc48559215)

[2. מסמך עיצוב 6](#_Toc48559216)

[2.1. דיון על פתרונות אפשריים לבעיה והנמקה לפתרון הנבחר 6](#_Toc48559218)

[2.2. דיאגרמות 9](#_Toc48559219)

[3. דוח מסכם 11](#_Toc48559220)

[3.1. מהות הפרויקט 11](#_Toc48559222)

[3.2. תיאור הבעיות שהתעוררו במשך העבודה ופתרונן 11](#_Toc48559223)

[3.3. הוראות התקנה של המערכת 13](#_Toc48559224)

[3.4. הוראות למשתמשי המערכת 13](#_Toc48559225)

[4. מבנה הקוד 28](#_Toc48559226)

[5. Spring Boot 35](#_Toc48559227)

[6. Databases 38](#_Toc48559228)

# מסמך דרישות

## הצגת הבעיה:

מערכת ASM[[1]](#footnote-1) הינה מערכת לניהול משמרות. מערכת זו מחשבת בצורה אופטימלית לוח משמרות חודשי, תוך מיקסום בקשות העובדים. כלומר, המערכת תחשב את לוח המשמרות כך שכמה שיותר בקשות למשמרות יאושרו.

המערכת פותחה בתצורת Web אשר הינה רספונסיבית למחשבים נייחים, ולמכשירים ניידים כאחד.

המערכת פותחה מתוך צורך השעה אשר התפתח בארץ לאור משבר הקורונה והעבודה בתוך "קפסולות" ומתוך רצון לתת מענה טכנולוגי לתכנון אופטימלי.

להלן מספר סיבות לצורך של חברה במערכת כזו :

* תכנון אופטימלי של משמרות – מתן מענה בקשות מקסימלי ביחס לדרישות העובדים.
* מענה איכותי ופשוט לדרישות העובדים.
* ממשק נוח ופשוט למנהל ולעובד כאחד.
* לרוב, שיבוץ המשמרות כיום הוא מאתגר. יש להתחשב באילוצים השונים ולבנות לוח שיספק כמה שיותר בקשות של העובדים. בעקבות כך, במקרים רבים אחראי המשמרות משקיע זמן עבודה יקר, שיכול להגיע ליותר מעשר שעות חודשיות, על בניית לוח משמרות. באמצעות מערכת ASM ניתן לחסוך שעות רבות של עבודה, שינוצלו לדברים חשובים יותר מבניית לוח משמרות.

לאור האמור לעיל, מצאנו לנכון לפתח מערכת שתספק מענה לבניית לוח משמרות למקומות עבודה, כגון: בתי חולים, משרדים, בנקים ועוד.

## רקע תיאורטי:

בהנחיית פרופסור אסתר דוד יצרנו מערכת מתקדמת אשר מנהלת משמרות בצורה אופטימלית, אך גם משלבת אלמנטים של נוחות ופשטות. פיתוח המערכת מאפשר מחד גיסא למנהלים לנתח בצורה פשוטה את דרישות העובדים ולעבד את הבקשות בצורה מיטבית תוך מתן תחושה של "יחס אישי", ומאידך לספק פלטפורמה מעולה להעלות את אילוצי/דרישות העובד למנהל כוח האדם. למנהל ישנן אפשרויות ייחודיות רק לו, שיפורטו להלן בפרקים הרלוונטים.

## דרישות פונקציונליות:

* + 1. המערכת תאפשר קביעת משמרות לכל עובד.
    2. המערכת תאפשר צפיה במשמרות מחודשים קודמים (היסטוריה).
    3. המערכת תאפשר לעובד לשמור את המשמרות אליהן הוא רוצה להירשם, וכן לשנות את בחירתו בכל זמן נתון כל עוד המנהל לא ביצע חישוב לוח משמרות.
    4. המערכת תאפשר למנהל לרשום משתמשים חדשים.
    5. המערכת תאפשר למנהל לצפות בכלל המשתמשים והרשאותיהם.
    6. המערכת תאפשר למנהל לבצע מחיקת משתמשים.
    7. כאשר ההרשמה למשמרות פתוחה, המערכת תציג למנהל בצורה גרפית את אחוז העובדים שמילאו בקשות למשמרות.
    8. המערכת מאפשרת למשתמש לבצע שינוי סיסמא.
    9. המערכת תאפשר לעובד לצפות בכרטיס העובד שלו.

## דרישות לא פונקציונליות :

* + 1. זמני התגובה של המערכת צריכים להיות טובים עד כדי 2 שניות בין מעברי דפים ועדכונים. הפעולה של חישוב המשמרות יכולה לחרוג מזמן זה, אך לא תעלה על 5 שניות.
    2. המערכת תשתמש בשפות JavaScript, HTML, CSS וSpring Boot, וכן בשפת שאילתות של SQL (DB-MySQL).
    3. המערכת צריכה להיות רספונסיבית. כלומר, להתאים את עצמה לכל גודל מסך (נייח, נייד, טאבלט ועוד).
    4. המערכת נדרשת לממשק משתמש קל ונוח.
    5. המערכת נדרשת לעשות שימוש באלגוריתם אופטימלי לשיבוץ משמרות.
    6. המשתמש יוכל לשמור את פרטי ההתחברות שלו בדפדפן, וכך להימנע מהקלדת פרטיו בכל כניסה.

## לוח זמנים לביצוע הפרויקט וחלוקת העבודה:

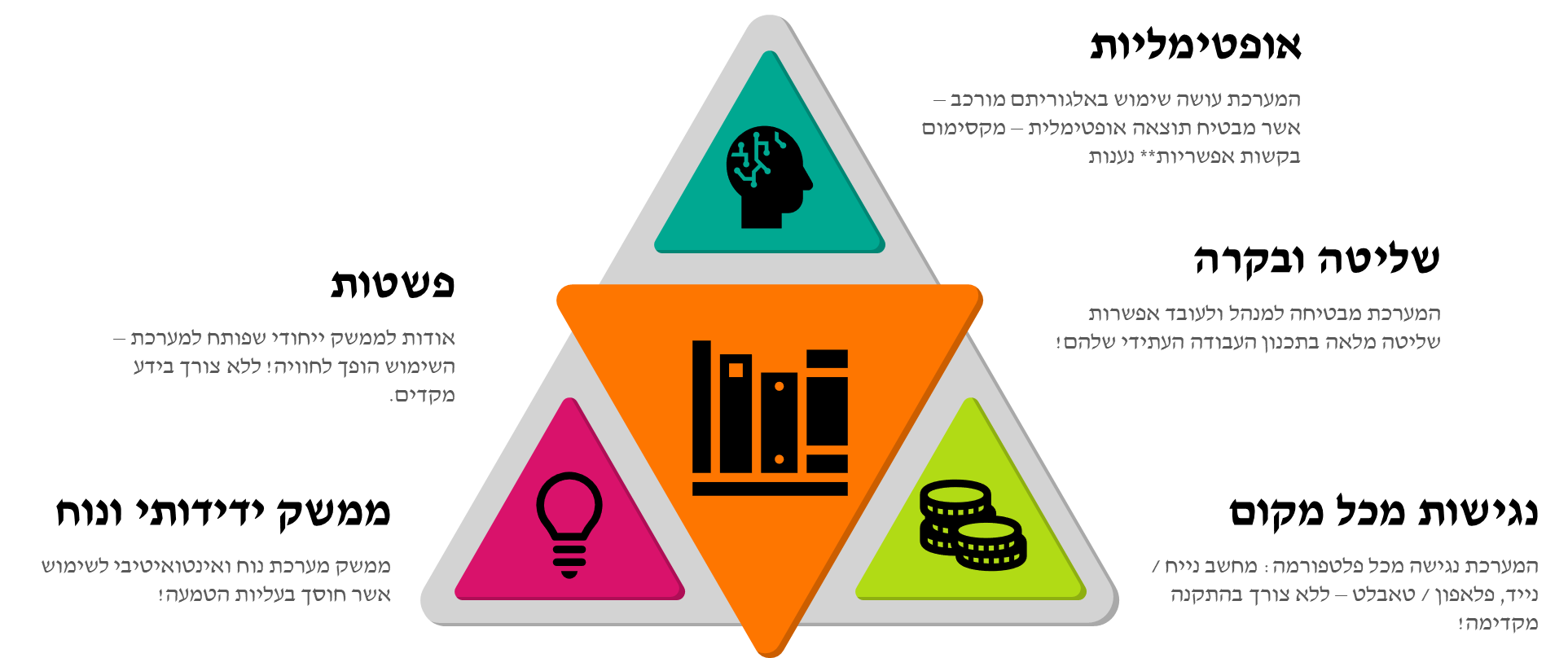
תקופה זו ניתן לחלק לשלושה חלקים: תכנון, למידה וביצוע:

* **תכנון** – חלק זה לקח ארבעה חודשים. תהליך אשר כלל התייעצויות רבות עם פרופסור אתי דוד ומומחה התעשיה מר שי גבעתי, אשר הביאו אותנו לתובנות רבות והשפיעו מאוד על ביצוע הפרוייקט.
* **למידה** – חלק זה למעשה אינו חלק רשמי משלבי הפרוייקט אך לאור האתגרים שהביאה עימה "הקורונה" בחרנו לראות זאת כהזדמנות: הזדמנות ללמידה מקצועיות אישית, הזדמנות לנסות ולטעות ולהתחיל מחדש, הזדמנות לגבש לעצמנו את תפיסת "האני מאמין המקצועי שלי". בשלב זה למדנו טכנולוגיות שאינן כלולות בסילבוסים הלימודיים, כגון: Spring boot, Thymeleaf, Hibernate ועוד. פירוט נוסף על טכנולוגיות אלו יובא בהמשך, בפרקים הרלוונטים.
* **ביצוע** – בשלב זה בנינו בפועל את המערכת. בעקבות תהליכי הלמידה שנקטנו, תכנון נכון, ועמידה קפדנית בזמנים הצלחנו לסיים את מרבית הפרוייקט עוד לפני תקופת המבחנים ובצורה יעילה.

חלוקת העבודה הייתה:

* **מחקר אודות שפת תכנות** – אורין שמעוני וברק סגל.
* **מחקר אודות מימוש אלגוריתם** – נוי חדד.
* **מחקר ממשק לקוח** – אליחי בן אברהם.
* **צד שרת** – אורין שמעוני וברק סגל (Pair Programing).
* **צד לקוח** – נוי חדד ואליחי בן אברהם.
* **מסמכים ומצגת** – חלוקה של כולם, בהתאם לחלק היחסי שלו בפרויקט.
* **דיאגרמות תהליכים** – נוי חדד.
* **אינטגרציה** – אורין שמעוני וברק סגל.

## יתרונות המערכת :



# מסמך עיצוב



## דיון על פתרונות אפשריים לבעיה והנמקה לפתרון הנבחר

ככל שבית עסק/חברה/ארגון מעסיק/ה עובדים רבים יותר, כך הופך עניין שיבוץ העובדים למטלה מפרכת, קשה, שלוקחת זמן רב. בנוסף, השיבוץ הסופי לרוב לא אופטימלי, וחלק גדול מהעובדים אינו מרוצה ממנו.

ישנם מספר פתרונות שאינם טכנולוגייה לבעיה זו**:**

**שיבוץ ע"י אחד העובדים** – ישנם מקומות בהן החברה מפקידה בידי אחד העובדים את המטלה לעשות שיבוץ שלו ושל שאר העובדים בכל פרק זמן שנקבע מראש. בדרך כלל העובד בוחר להשתמש באקסל או בקובץ אחר ולעיתים אף בכתיבה ידנית על דף. אך אחת הבעיות בשיטה זו היא עניין המוסריות, יתכן שלעובד תיהיה נטיה לתת חשיבות גבוה יותר לאילוצים ולצרכים שלו מאשר לשאר העובדים. בנוסף, לעיתים ישנם אחראי משמרות שנותנים משמרות נוחות לחבריהם, למשפחתם ולכל המרבה במחיר.

**שיבוץ ע"י מנהל** – ישנם מקומות בהן המנהל או דרג בכיר אחר, דואג לשיבוץ עובדיו. פתרון זה אמנם פותר חלק מהבעיות המוסריות שהצגנו לעיל, אך גם פתרון זה אינו חף מטעויות. מנהל, בכיר ככל שיהיה, הוא בן אנוש, ולכן הוא אינו חסין מטעויות. בזמן השיבוץ ישנם פרמטרים רבים שיש לקחת בחשבון, כמו: אילוצי עובדים, עדיפות עובדים, הגבלת מספר שעות, מניעה ממשמרות חופפות ועוד. התחשבות בכל הפרמטרים מקשה על בניית לוח משמות שיספק את כל הפרמטרים ביחד וליצור שיבוץ אופטימלי.

גם מערכת ASM אינה פותרת את כל האתגרים שהצגנו, שהרי בסופו של דבר יש התנגשויות בין בקשות של עובדים. אך לכל הפחות, את הפתרון המוסרי היא מספקת, כיוון שמדובר בחישוב לוגי ללא התחשבות באמוציות אנושיות או נתינת עדיפות ל"עצמי" או ל"חברי".

גם אם נמצא אדם ישר שיבנה את לוח המשמרות, עדיין ישנם יתרונות למערכת ASM על חישוב לוח משמרות אנושי. לדוגמה, היא חוסכת זמן רב של בדיקת אילוצים ובניית לוח משמרות. על ידי ASM ניתן לחסוך בזבוז כוח אדם ותשלום על שיבוץ משמרות, שמהווה הוצאה לא הכרחית.

**פתרון נוסף** שקיים בחלק מהחברות הינו **שיבוץ עצמי,** בו מפיצים לוח שנה של טווח זמן השיבוץ וכל עובד משבץ את עצמו במספר מוגבל של משמרות, ואחר סבב אחד של שיבוץ של כלל העובדים מתבצע עוד סבב עד שכל המשמרות אוישו או לחילופין לא נעשה שינוי מהסבב הקודם.

כל הפתרונות שהצגנו לא היוו פתרון אופטימלי למיקסום בקשות העובדים, מה שהוביל אותנו בתור קבוצה לנסות למצוא דרך לשיבוץ העובדים בצורה האופטימלית והנגישה ביותר, תוך כדי חיסכון בכוח אדם והוצאות מיותרות.

ההתקדמות הטכנולוגית הגיעה אל מערכות שיבוץ העובדים בדמות אפליקצית שיבוץ מקוונת. אפליקצית שיבוץ מקוונת היא למעשה אפליקציית נוכחות הזמינה ברשת האינטרנט בכל רגע נתון מכל מקום בעולם. כך שלאחר פתיחתההרשמה לשיבוץ משמרות על ידי המנהל, כל עובד יכול לסמן בלוח שנה שנמצא באפליקציה את המשמרות שבהן הוא רוצה לעבוד.

באמצעות מילוי טופס מקוון של בקשת משמרות, אנו חוסכים משאבים לארגון ובנוסף השיבוץ הסופי שיווצר יהווה פתרון אופטימלי מבחינת אישור מקסימלי של בקשות עובדים (מקסימום בקשות של עובדים למשמרות יאושרו).

המערכת מבוססת על אלגוריתם שלוקח בחשבון את כל הבקשות של העובדים, ומכריע באופן אופטימלי בין כל התנגשות של משמרות בין עובדים שונים. דבר זה מוביל לכך שלאחר פירסום של השיבוץ יהיה מינימום אנשים שלא יהיו מרוצים, מה שכל פתרון שיבוץ אנושי אחר לא יכול להבטיח.

**בנוסף יש למערכת יתרונות נוספים:**

**ארגון וסדר** – העובד רואה מול העיניים את המשמרות באופן גרפי, מה שמאשפר לו לבחור את המשמרות באופן שקול תוך כדי לקיחה בחשבון של עוד פרמטרי זמן, כמו תאריך, שעות עוד. דבר זה חוסך טעויות מצד העובד שהוא אינו שם לב אליהן. לדוגמה, כשעובד רואה את לוח השנה בצורה וויזואלית זה עוזר לו לזכור אירועים עתידיים שימנעו ממנו להגיע למשמרת.

**אתיקה** – כאשר מדובר על שיבוץ ממוחשב ללא התערבות מצד בן אדם, כל העובדים יכולים להיות סמוכים ובטוחים שהשיבוץ נעשה משיקולים של אופטימליזציה ולא מתוך שיקולים אישיים.

**גמישות** – המערכת מאפשרת לאחראי משמרות לבחוק כמה ימים יהיו במשמרת. ניתן לבחור בין משמרת אחת ביום לארבע משמרות ביום. כך המערכת מתאימה את עצמה לארגונים שונים בהתאם לדרישה.

**ידידותית למשתמש** – המערכת נבנתה באופן שבו היא תהיה ברורה ונגישה לשימוש למשתמש הממוצע. היא אינה, מצריכה ידע קודם, ניסיון טכנולוגי או קורס הטמעה בכדי להתחיל להשתמש בה. בנוסף המערכת מתאימה את עצמה למכשירים השונים (רספונסיבית). לכן החלטנו שהמערכת בנוסף להיותה אופטימלית מבחינת אישור בקשות, היא תהיה גם ידידותית ולא תגזול מזמנם של עובדים יותר ממספר מועט של דקות.

לאחר דיון בין חברי הקבוצה ולאחר התייעצות עם מנהלי תחומים שאחראים על עניין השיבוץ בארגון בו הם עובדים (כגון: ביה"ח ברזילי), החלטנו לבנות מערכת נגישה לכלל העובדים וכן למנהלים, שיוכלו באופן פשוט, זריז ואופטימלי לבנות לוח משמרות. כאשר המנהל פותח את ההרשמה למשמרות, הוא יכול לעקוב אחרי אחוז העובדים שמילאו את המשמרות שלהם. כל עובד יכול למלא משמרות מכל מקום בעולם (ומכל מכשיר). כאשר המנהל רוצה לחשב לוח משמרות, הוא יכול בלחיצת כפתור להפעיל את החישוב ללוח משמרות ובכך ליצור לוח משמרות לחודש שבחר. כמו כן המנהל יכול לצפות בפרטים של עובדיו, וכן כל העובדים יכולים לצפות בשיבוצים של חודשים קודמים.

## דיאגרמות

**דיאגרמה מערכת ASM – עובד רגיל**

נחיתה

התחברות

הרשמה למשמרות

לוח משמרות

מידע על מנהלי המערכת

התנתקות

בחירת חודש

מידע אישי

שינוי סיסמה

**דיאגרמה מערכת ASM – מנהל**

נחיתה

התחברות

לוח משמרות

הרשמה למשמרות

מידע על מנהלי המערכת

דף מנהל

סגירת הרשמה

צפייה בפרטי העובדים

פתיחת הרשמה

הוספת עובד חדש

מחיקת עובד קיים

שינוי פרטי אחראי משמרות

שינוי פרטי חברה

התנתקות

חישוב לוח משמרות

מידע אישי

שינוי סיסמה

# דוח מסכם



## מהות הפרויקט

בפרוייקט ASM, בנינו מערכת לניהול חלוקת משמרות בין העובדים, המיועדת לשימוש במחשבים, פלאפונים, טאבלטים וכל מכשיר שיודע להציג קבצי HTML עם CSS וJavaScript. המטרה לבנות מערכת אחת שתתאים את עצמה למכשיר שממנו ניגשים אליה (רספונסיבית). בנוסף, המערכת צריכה להיות פשוטה לשימוש, כך שגם משתמש חדש יבין בקלות איך משתמשים במערכת ללא הסבר מראש.

חלק חשוב במערכת, הוא שביעות הרצון של העובדים. ASM מחשבת משמרות, כך שהיא מקבלת את המספר המקסימלי של בקשות למשמרות שניתן לאשר. כלומר, היא לוקחת את כל הבקשות למשמרות, וע"י האילוצים שיש, היא מוצאת את מספר הבקשות המקסימלי שניתן לאשר. כך אנו מעלים את שביעות הרצון של העובדים מהמשמרות שהם מקבלים.

דגש נוסף הוא הנושא של האבטחה. השתמשנו בSpring Security בכדי לאבטח את המערכת ובכך השגנו סיפריית אבטחה חזקה המנהלת את בקרת הגישה, ואימות הזהות של המשתמשים השונים.

## תיאור הבעיות שהתעוררו במשך העבודה ופתרונן

במהלך הביצוע של הפרויקט נתקלנו באתגרים והתלבטויות שונות כיצד לבצע את הפרויקט:

* **טכנולוגיה** – בתחילת הפרויקט התלבטנו כיצד לבנות את המערכת. האם להשתמש ב"java" ללא spring boot או להשתמש בשפה אחרת שאנחנו מכירים מהלימודים שלנו במכללה. ההתלבטות הייתה האם להיכנס לטכנולוגיה שאנו לא מכירים ובכך להאריך את משך בניית הפרויקט, או להשתמש באמצעים שאנו מכירים ובכך לחסוך את ימי הלימוד שנצטרך להשקיע עוד לפני שנתחיל את הפרויקט. בסופו של דבר, בעצתו של מר שי גבעתי, החלטנו ללכת על spring boot, שהיא טכנולוגיה הנמצאת בשימוש גם כיום, ובכך להשיג יתרון למציאת עבודה בעתיד.

הבחירה שלנו להשתמש ב-spring boot דרשה מאיתנו ללמוד טכנולוגיות חדשות שלא נלמדה במכללה, כפי שנפרט בפרקים הבאים. במהלך החודשים הראשונים לא התקדמנו בבניית המערכת, אלא השקענו זמן בלמידת האמצעים הדרושים להשלמת הפרויקט. הלמידה ללא מרצה שינחה אותך הייתה מייגעת אך נתנה לנו כלים להתמודד עם טכנולוגיות חדשות, דבר חשוב בתחום ההיטק בימינו שבו הטכנולוגיות מתחלפות בקצב מהיר.

* **בחירת שם** – העלנו מספר אופציות לבחירת שם שיסגיר את מטרתה של המערכת ולבסוף החלטנו על ASM (Advanced Shifts Management).
* **עיצוב** – התלבטנו האם לקחת תבנית עיצוב קיימת, או לעצב את המערכת בעצמנו. בעקבות ההחלטה שלנו ללכת על טכנולוגיה שאנו לא מכירים, ולמידת הטכנולוגיה לקחה נתח נכבד מהזמן שלנו, החלטנו לקחת תבנית עיצוב מוכנה, ועליה בצענו את ההתאמות שנדרשו בכדי להתאים אותו למערכת שלנו.
* **אלגוריתם** – רצינו אלגוריתם שיהיה אופטימלי מבחינת סיפוק בקשות העובדים, שכמה שיותר בקשות למשמרות יאושרו. בהתחלה חשבנו לכתוב את האלגוריתם בעצמנו או לממש את אחד האלגרותמים מהמאמרים שמצאנו על "Nurse Scheduling Problem". אך גם כאן, בעקבות הזמן הממושך שלקח לנו ללמוד את הטכנולוגיה, והידיעה שאחד מחברי הצוות חוזר לשירות צבאי באוגוסט, השתמשנו בסיפרייה של Google בשם "OR-Tool".
* **לוח שנה** – התלבטנו האם להשתמש בלוח שנה קיים או לבנות אחד בעצמנו. בסופו של דבר החלטנו לקחת עיצוב (CSS) ללוח שנה ולבנות אותו בעצמנו. התמודדנו עם אתגרים שונים במהלך בניית הלוח שנה, כגון: כיצד להתאים את הלוח שנה גם להרשמה למשמרות וגם לתצוגת משמרות שחושבו, איך לחשב כמה ימים יש בכל חודש, כיצד להציג משתמשים בתצוגת משמרות ועוד.
* **אתגרים והתלבטויות נוספות** שהתעוררו תוך כדי התקדמות הבנייה של המערכת. לדוגמה:
  + **האם לאפשר למחוק חשבון של מנהל?**

כיוון שההתקנה ההתחלתית של המערכת מתבצעת על ידי הכנסת משתמש מנהל (כיוון שההרשמה לאתר סגורה, ורק מנהל יכול לרשום עובדים חדשים), בחרנו למנוע מחיקה של מנהל כדי לא לאפשר מצב שבו המערכת נשארה ללא מנהל.

* + **האם לאפשר לעובדים להירשם לבד או שרק למנהל יהיו הרשאות לרשום עובד חדש?**

בסופו של דבר החלטנו שרק מנהל יוכל לרשום משתמשים.

* + **האם לאפשר לעובד לשנות את הפרטים האישיים שלו?**

כיוון שההרשמה מתבצעת על ידי המנהל, בחרנו שלא לאפשר לעובד לשנות את הפרטים האישיים שלו, אלא רק לשנות סיסמה.

* + **במידה והמנהל סוגר את ההרשמה, האם לשמור את הטפסים של המשמרות שהעובדים מילאו או למחוק אותם?**

כאן בחרנו לשמור את המידע הזה, ובמידה והמנהל יפתח מחדש את ההרשמה לאותו חודש, הטפסים שנשמרו יוצגו לעובדים.

## הוראות התקנה של המערכת

1. תחילה יש לעדכן מספר הגדרות:

* פרטי התחברות למסד הנתונים (שרת, פורט, שם משתמש וסיסמה).
* הפורט שבו תרוץ המערכת (ברירת מחדל: 8080).
* עדכון הנתיב של תיקיית ההתקנה של פייתון ושל קובץ הפייתון שמריץ את הסיפריה של OR-Tool לחישוב המשמרות (**יש לוודא שמותקן פייתון**). את העדכון יש לבצע במחלקה PythonOrTools.java.

1. לאחר מכן יש לייצא את הפרויקט שלנו כקובץ הרצה. סוג הקובץ שנייצא תלוי במערכת שבה נרצה להריץ את המערכת. לדוגמה:
2. עבור שרת Tomcat נוסיף בקובץ ה"pom.xml" את התלות של "tomcat server" ונייצא קובץ מסוג ".war".
3. עבור פלטפורמה של Windows (שמריץ Java) נייצא קובץ מסוג ".jar". לאחר לחיצה יצירת הקובץ, נלחץ עליו והמערכת תתחיל לפעול.

כמובן שישנן פלטפורמות נוספות שיודעות להריץ אפליקציית web של spring boot בווריאציות שונות (כגון: AWS של אמזון, Google Cloud ועוד) ולכל אחת מהן צריך לבצע התאמות שונות בהתאם לדרישות.

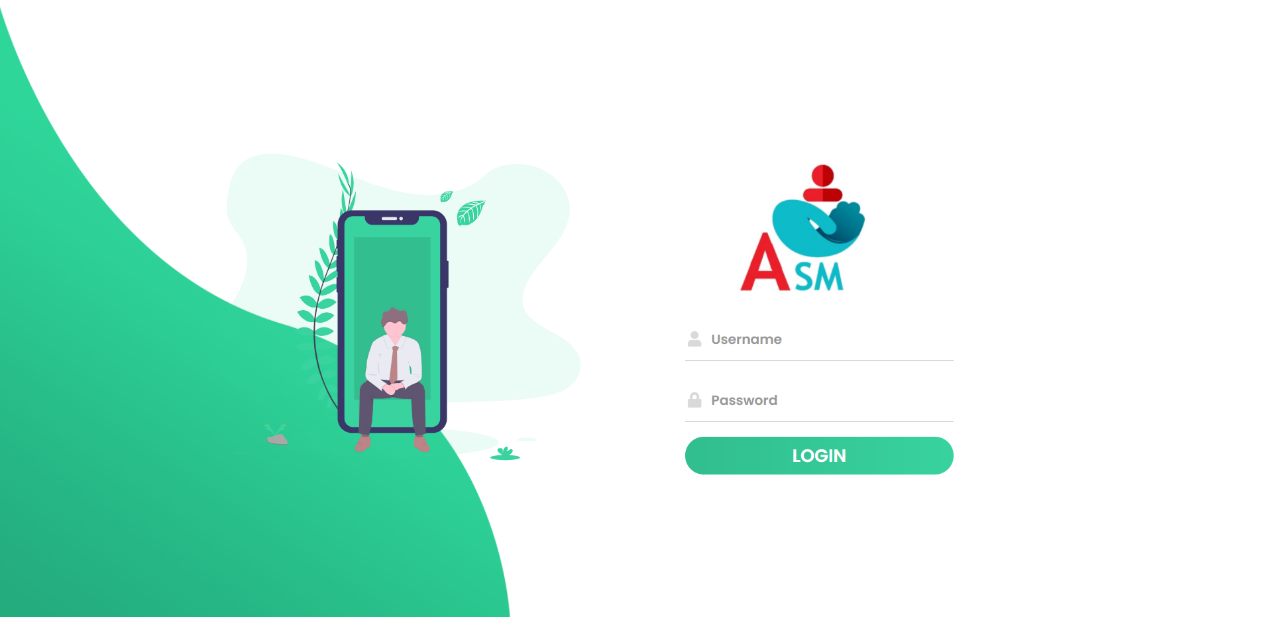
1. **הגדרת משתמש מנהל** – כיוון שההרשמה למערכת מתאפשרת רק על ידי מנהל, יש לרשום משתמש חדש באופן ידני ולהגדיר לו הרשאות מנהל.
2. המערכת מוכנה לשימוש ☺

## הוראות למשתמשי המערכת

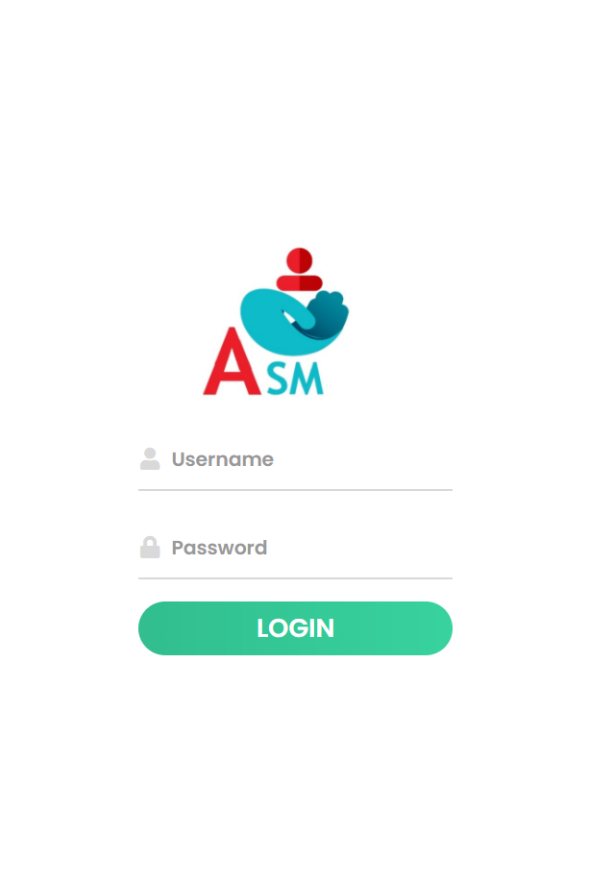
1. **התחברות:**

תחילה יש לבצע התחברות למערכת ע"י שם משתמש (אימייל) וסיסמה.

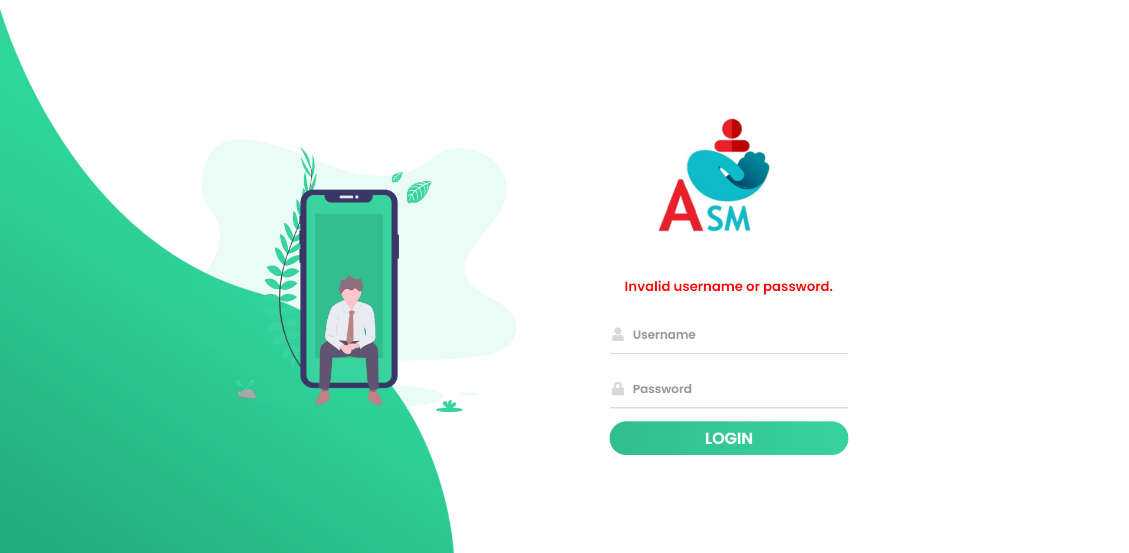
תצוגת מחשב:

****

תצוגת נייד:

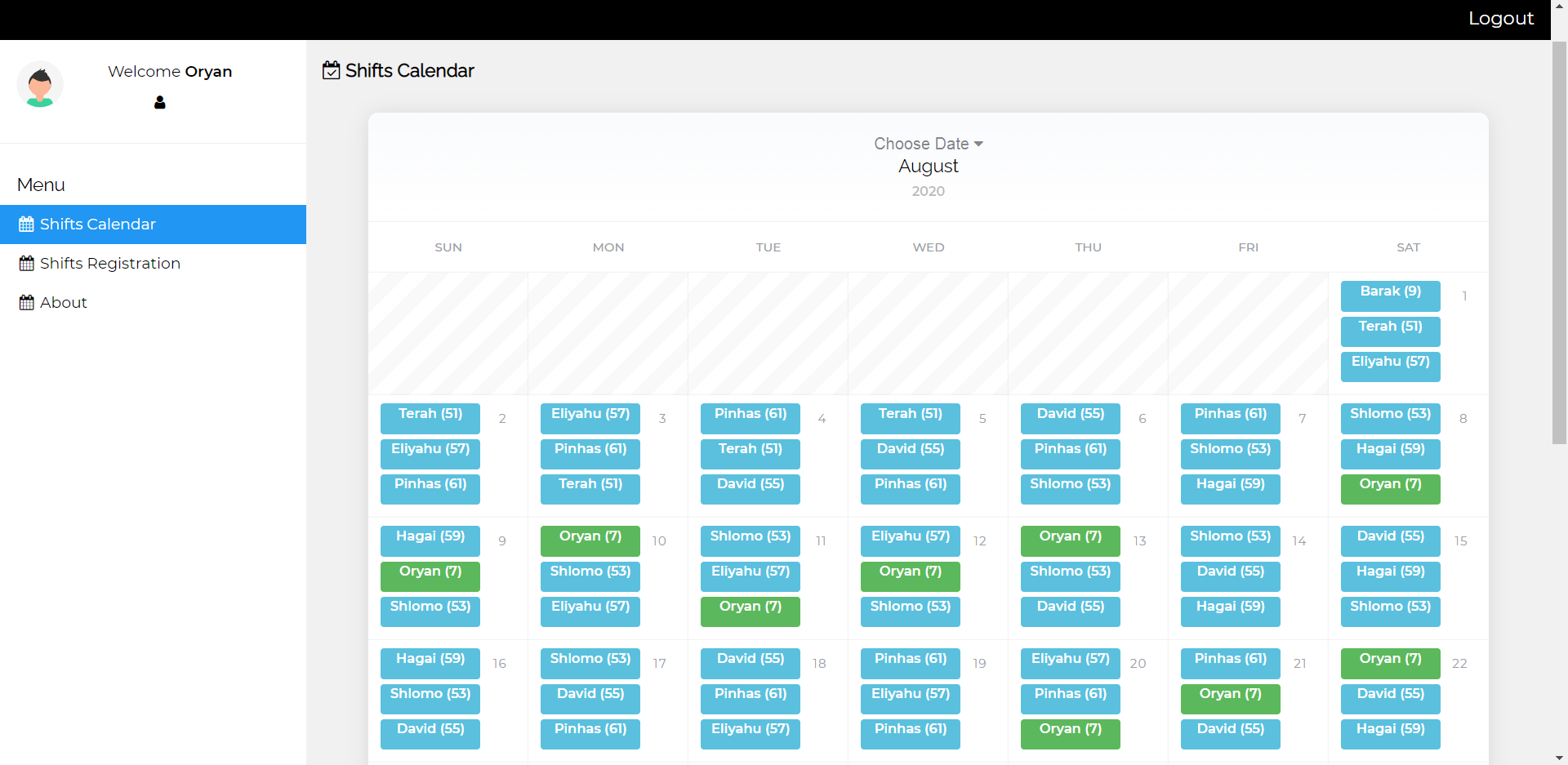


במידה ואחד מהפרטים שהוזנו היו שגויים, תוצג הודעה למשתמש:



לאחר התחברות מוצלחת המשתמש יועבר לתצוגת לוח משמרות. ישנם הבדלים בין תצוגת המנהל לתצוגת עובד רגיל, כפי שנפרט בהמשך.

1. **תצוגת עובד רגיל:**
2. **Shifts Calendar** – לאחר ההתחברות המשתמש יגיע ללוח המשמרות:

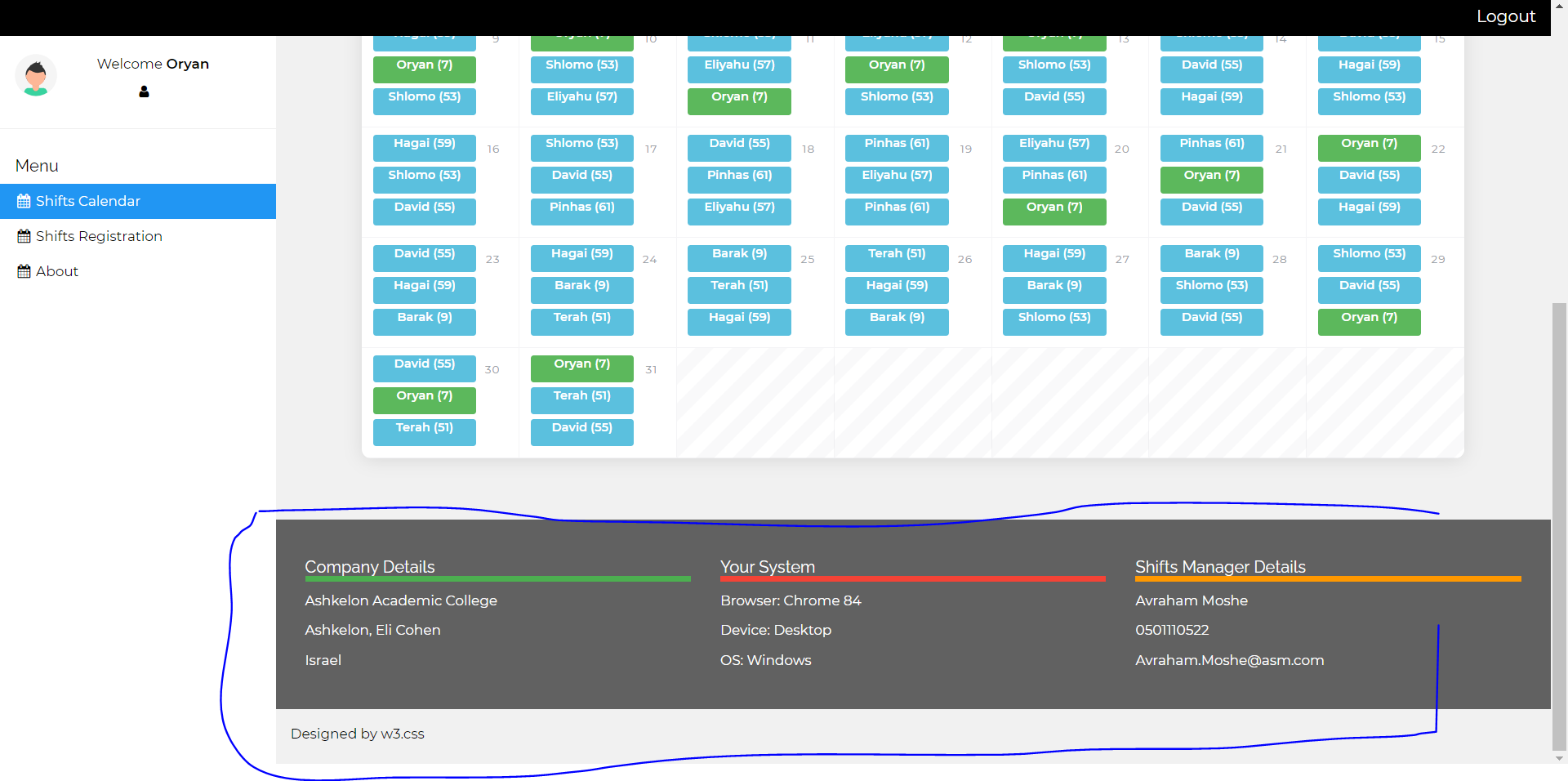


בחלקו השמאלי של כל אחד מהעמודים ישנו **תפריט** ובו מוצגים:

* + שמו הפרטי של העובד.
  + כפתור בצורת איש שמוביל לדף כרטיס עובד ושינוי סיסמה.
  + Shifts Calendar – לוח משמרות (הדף הנוכחי).
  + Shifts Registration – הרשמה למשמרות.
  + About – דף עם נתונים על מנהלי האתר (בדוגמה שלנו שמנו את הפרטים האישיים של מגישי הפרויקט).

בחלקו הימני העליון של האתר יש **כפתור "Logout"** להתנתקות מהמערכת וחזרה לעמוד ההתחברות.

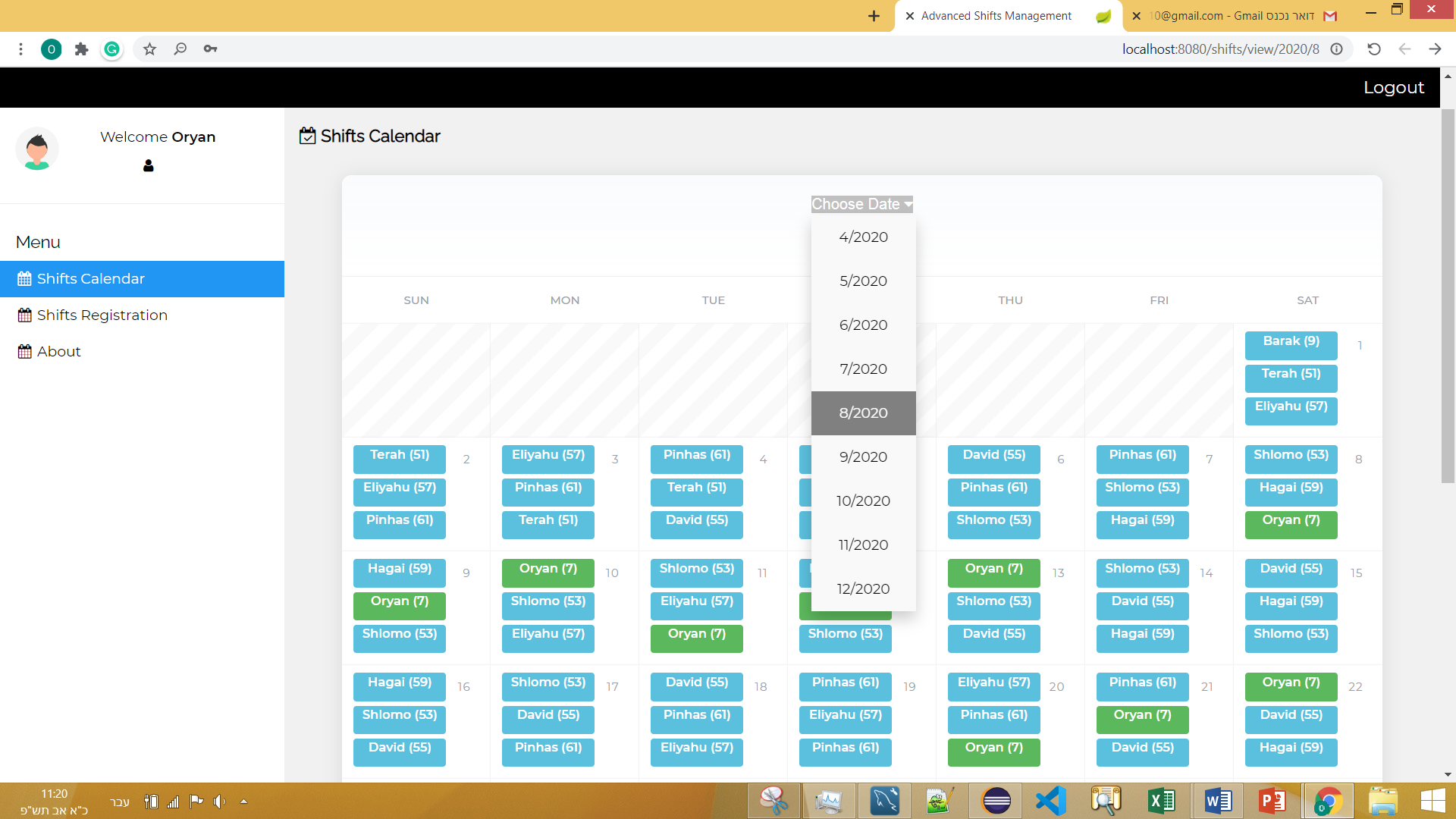
**בחלקו התחתון (footer)** של כל אחד מהעמודים יוצג המידע הבא:



* פרטי החברה: שם החברה וכתובת.
* פרטי המערכת של המשתמש: דפדפן, מכשיר ומערכת הפעלה.
* פרטי מנהל המשמרות: שם מלא, מספר טלפון ומייל.

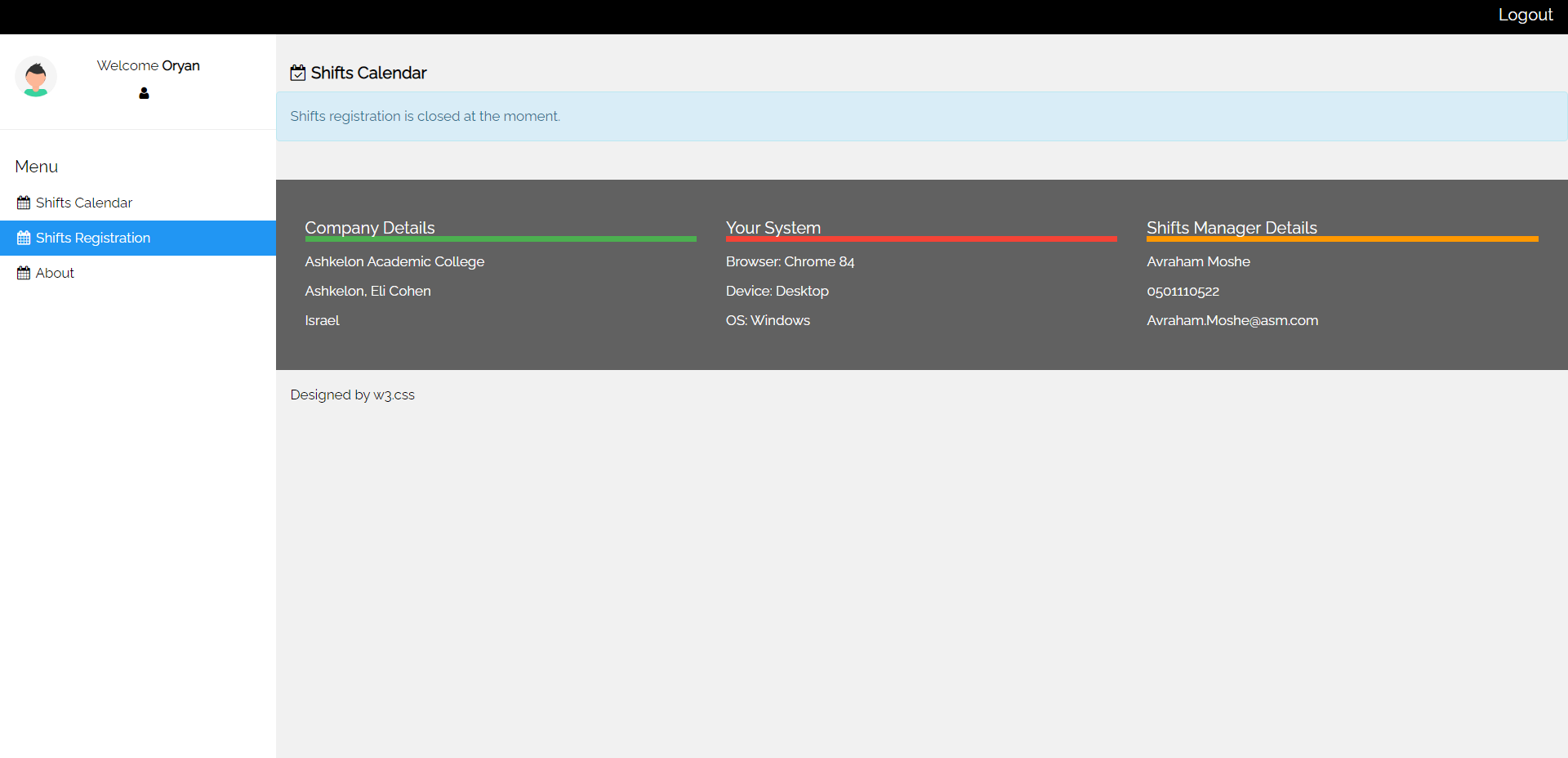
**בחלון המרכזי** ישנו לוח שנה ובו מוצגים המשמרות של החודש הנוכחי, כאשר מסומנות לעובד המשמרות שלו בצבע ירוק. בכל מלבן של משמרת מוצגים: השם הפרטי של העובד והמספר המזהה שלו (שאותו הוא מקבל בהרשמה למערכת, ויכול לצפות בו בדף שבו נמצא ה"כרטיס עובד" שלו, כפי שיוסבר בהמשך). כך גם אם יש שני עובדים בעלי אותו שם, הם יוכלו לזהות את עצמם בעזרת הסימון הירוק או בעזרת המספר המזהה שלהם.

בנוסף ישנה אפשרות לצפות בלוחות משמרות של חודשים קודמים, או עתידיים, ע"י מעבר עם העכבר מעל הכפתור "Choose Date" שנמצא בחלקו העליון של הלוח שנה:

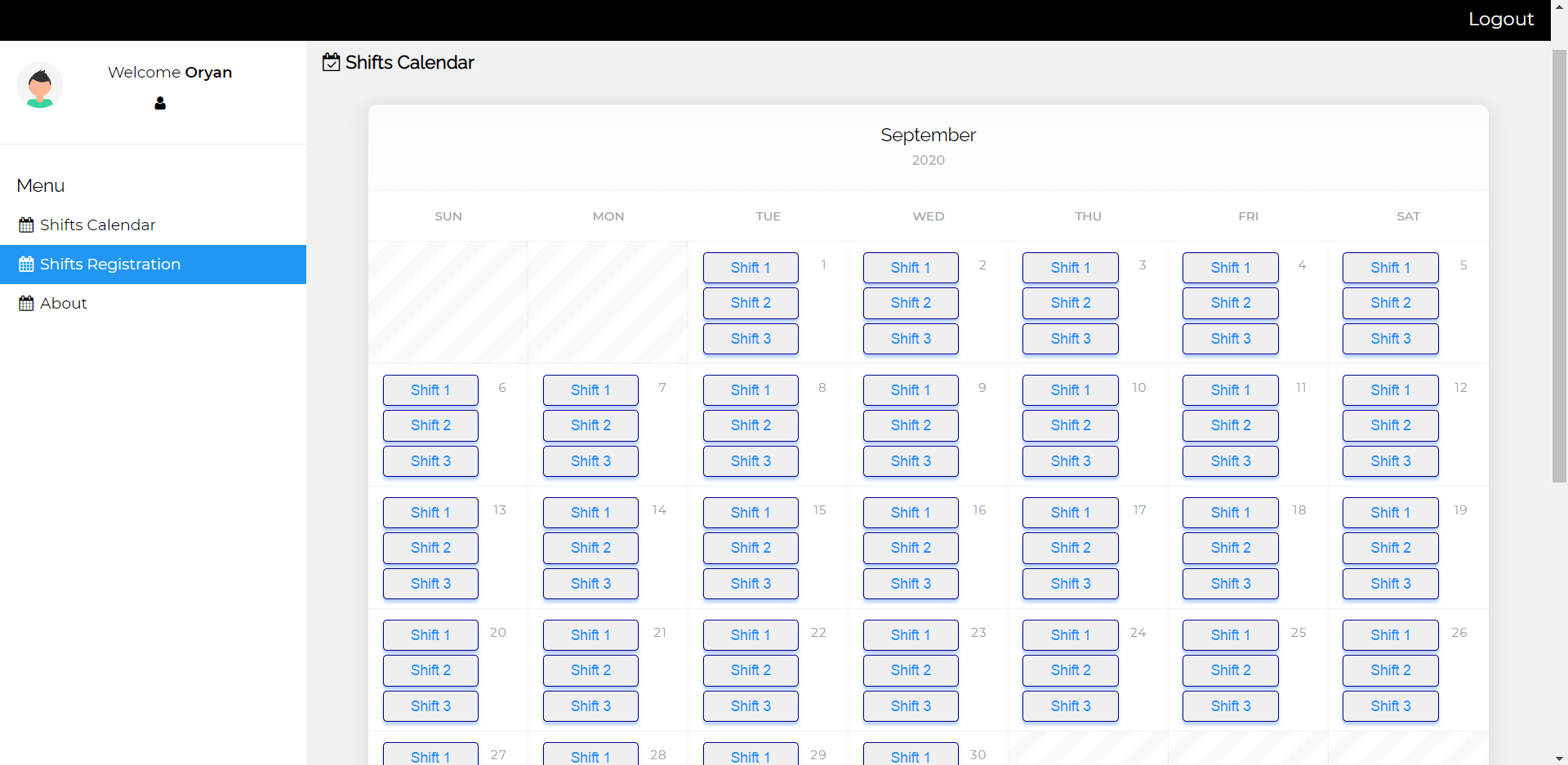


1. **Shifts Registration** – בעמוד זה ניתן להירשם למשמרות. ישנן שתי אפשרויות:

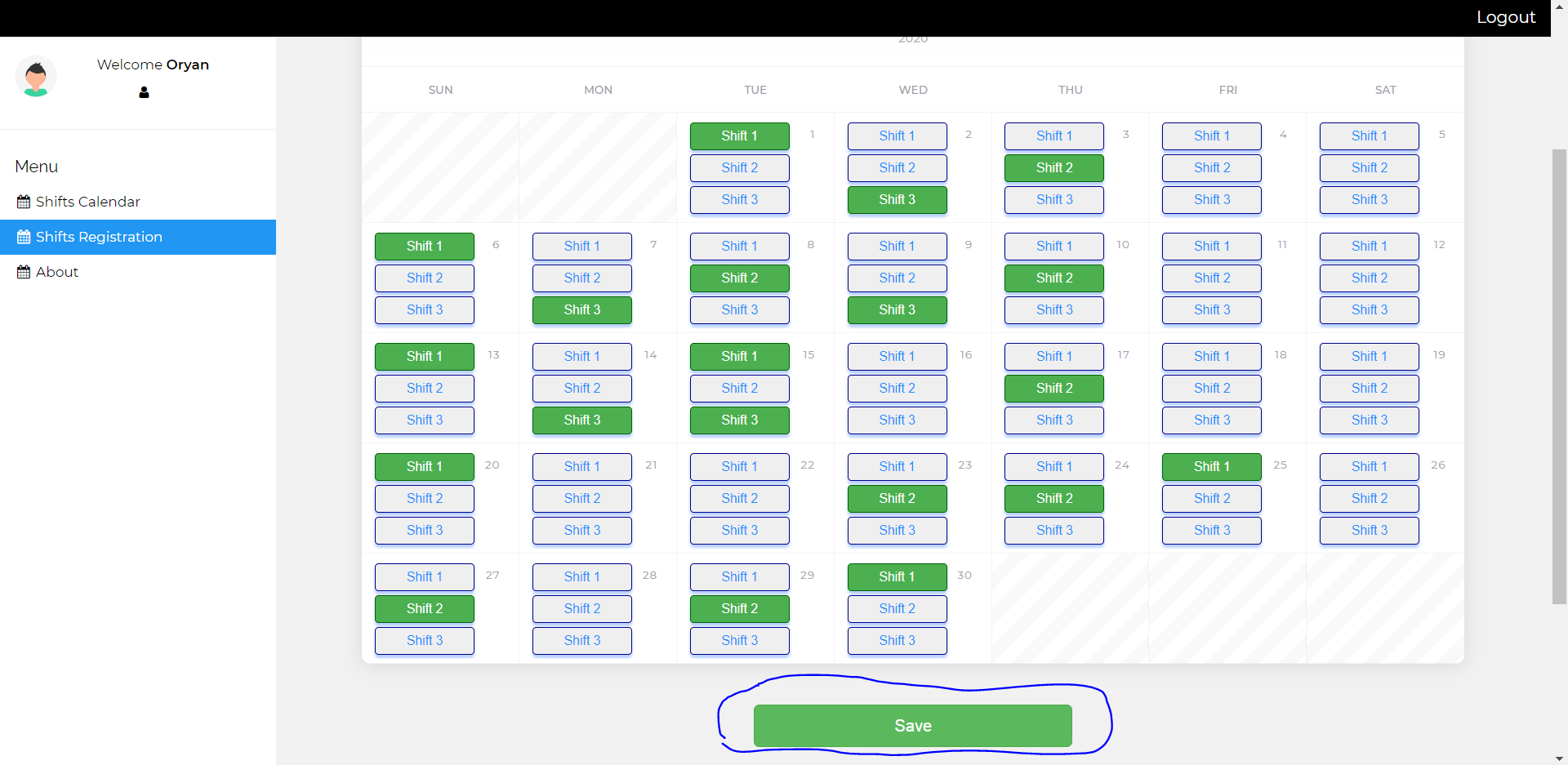
* עמוד ההרשמה **סגור** (המנהל לא פתח הרשמה למשמרות):



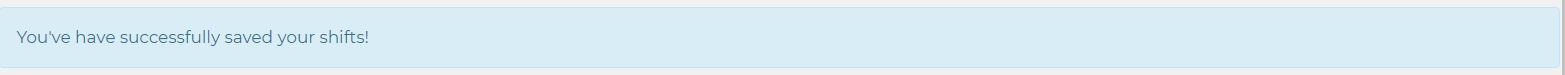
* עמוד ההרשמה **פתוח**:



המשתמש בוחר את התאריכים ואת המשמרות שהוא מעוניין בהם ע"י לחיצה על הכפתור המתאים בלוח שנה. לאחר שהמשתמש סיים לבחור את המשמרות שלו עליו ללחוץ על "Save":



לאחר שהמשתמש לחץ על "Save" תוצג לו הודעה:



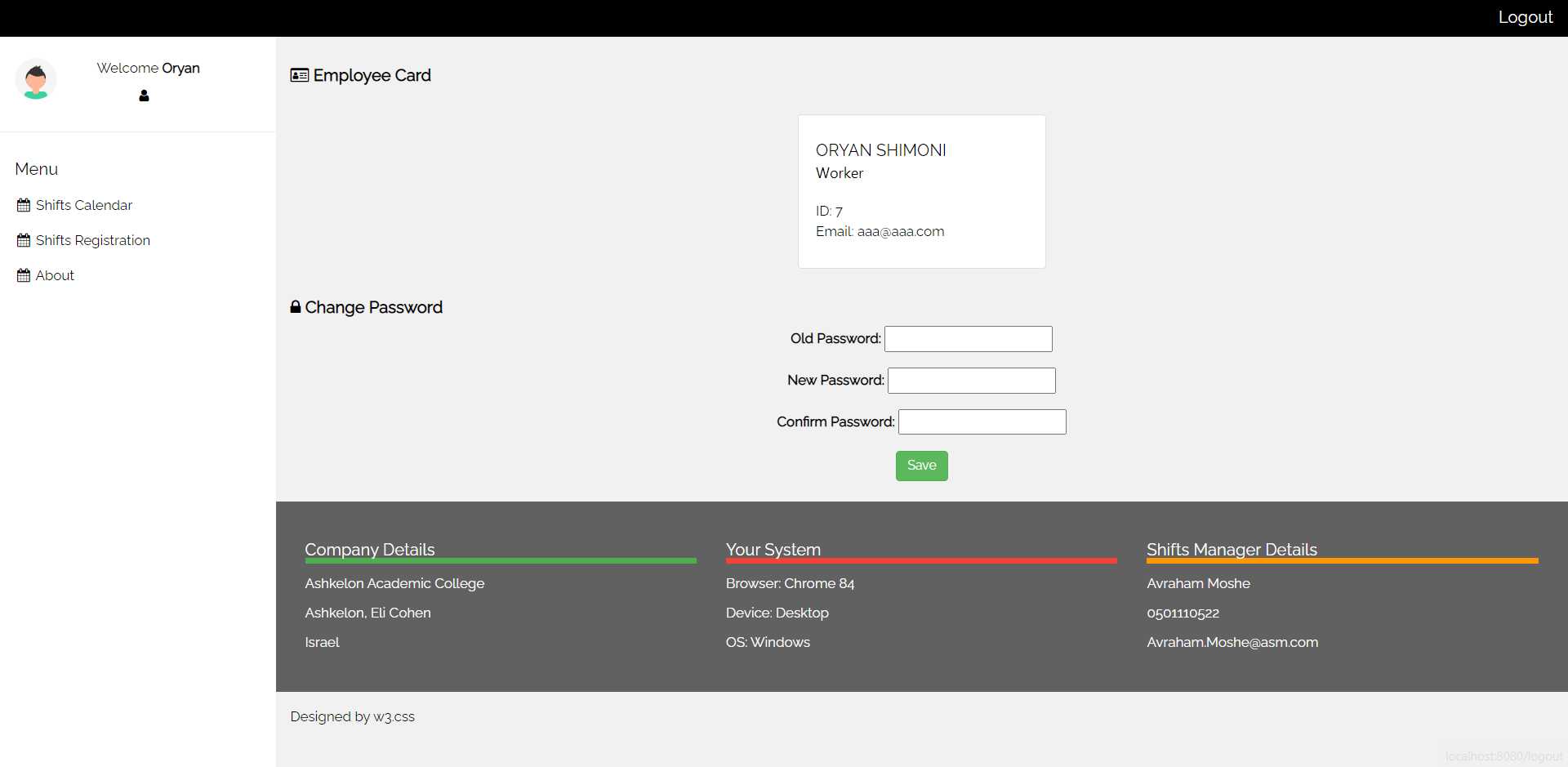
המשתמש יכול לחזור לעמוד זה ולצפות\לערוך את בחירת המשמרות שלו. אולם לאחר שהמנהל יפעיל את חישוב המשמרות (יוסבר בהמשך) או יסגור את ההרשמה, לא תתאפשר צפייה\עריכה של המשמרות.

1. **About** – בדף זה מוצג מידע על מנהלי המערכת:

עבור כל אחד מהמפתחים מוצג המידע הבא:

* שם מלא.
* עיר מגורים.
* מספר פלאפון.
* כתובת מייל.

1. **Info** – עמוד מידע אישי ושינוי סיסמה:

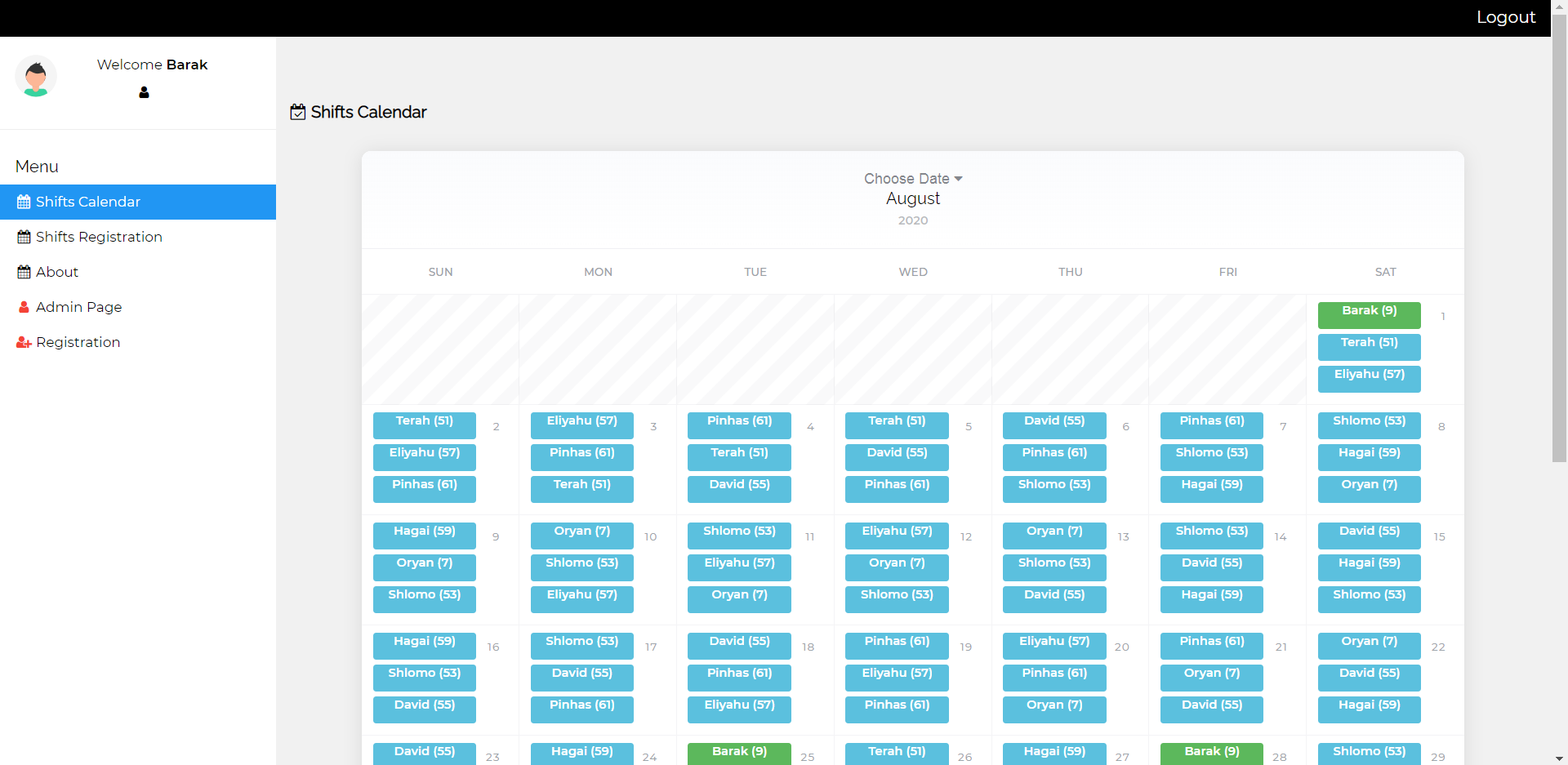


בדף זה מוצגים:

* **כרטיס עובד:** בכרטיס זה מוצגים הפרטים הבאים:
  + שם מלא.
  + תפקיד (Worker או Admin).
  + ID – מספר מזהה של המשתמש (בכדי להבדיל בין עובדים בעלי אותו שם).
  + כתובת מייל.
* **שינוי סיסמה:** בכדי לשנות סיסמה יש למלא שלושה שדות:
  + סיסמה נוכחית (Old Password).
  + סיסמה חדשה (New Password).
  + אימות סיסמה חדשה (Confirm Password), בכדי לוודא שאין טעות בהקלדה של הסיסמה.

1. **תצוגת מנהל:**

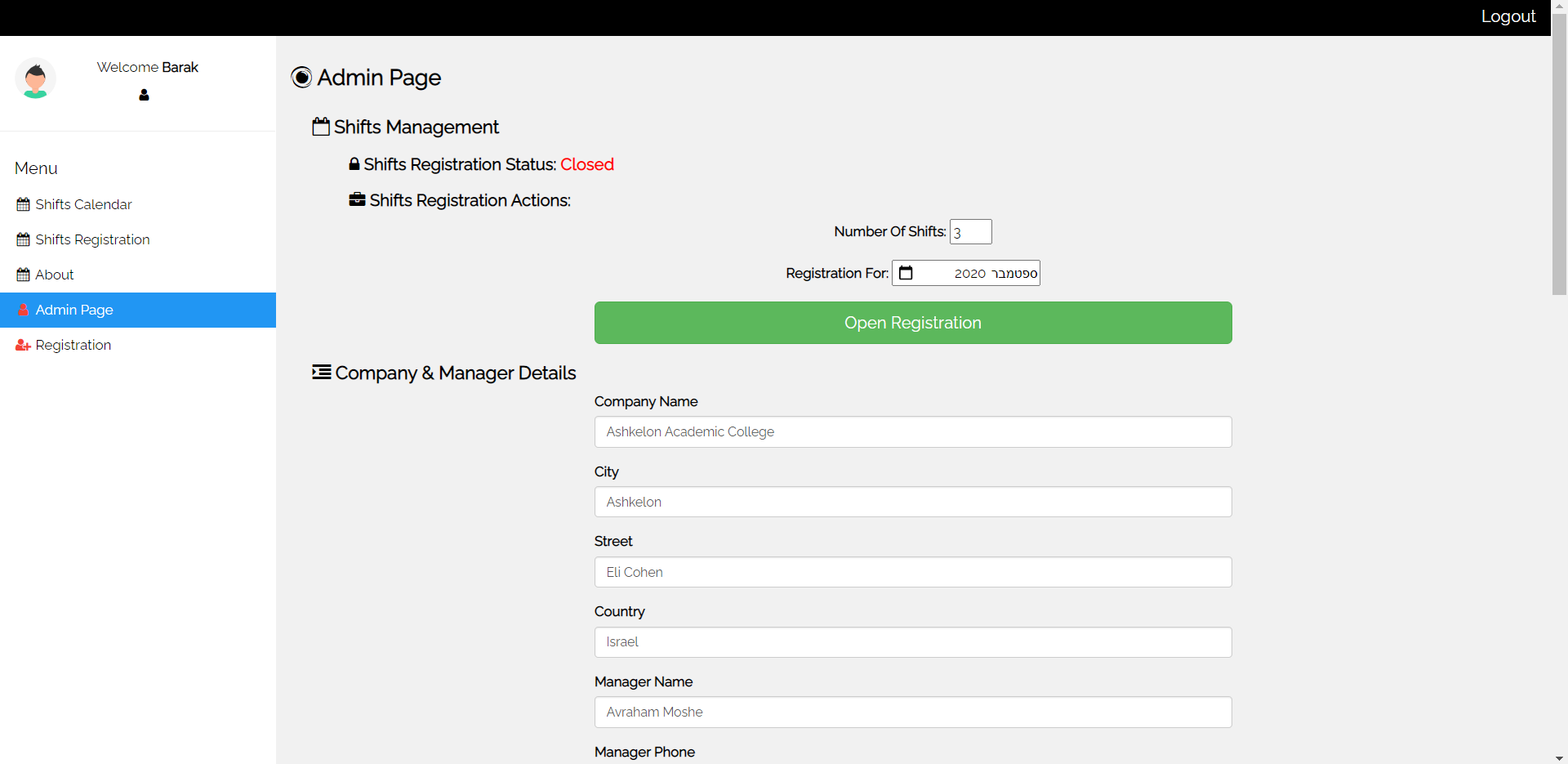
בנוסף לעמודים הרגילים שיש לכל משתמש, למנהל יש שני עמודים נוספים:



כפי שניתן לראות בצד שמאל בתפריט (בצבע אדום):

* **Admin Page** – דף ניהול.
* **Registration** – דף הרשמה לעובד חדש.

1. **Admin Page** –



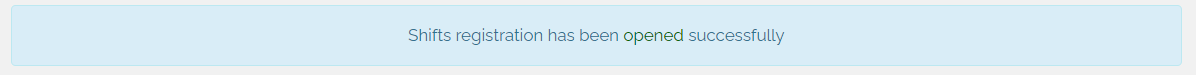
בדף זה המנהל יכול לבצע מספר פעולות:

* **Shifts Management** – מתחלק לשני חלקים:
  + **הרשמה סגורה:** כפי שניתן לראות בתמונה לעיל, למנהל יש אפשרות לראות את סטטוס ההרשמה (Closed) וכן לפתוח את ההרשמה.

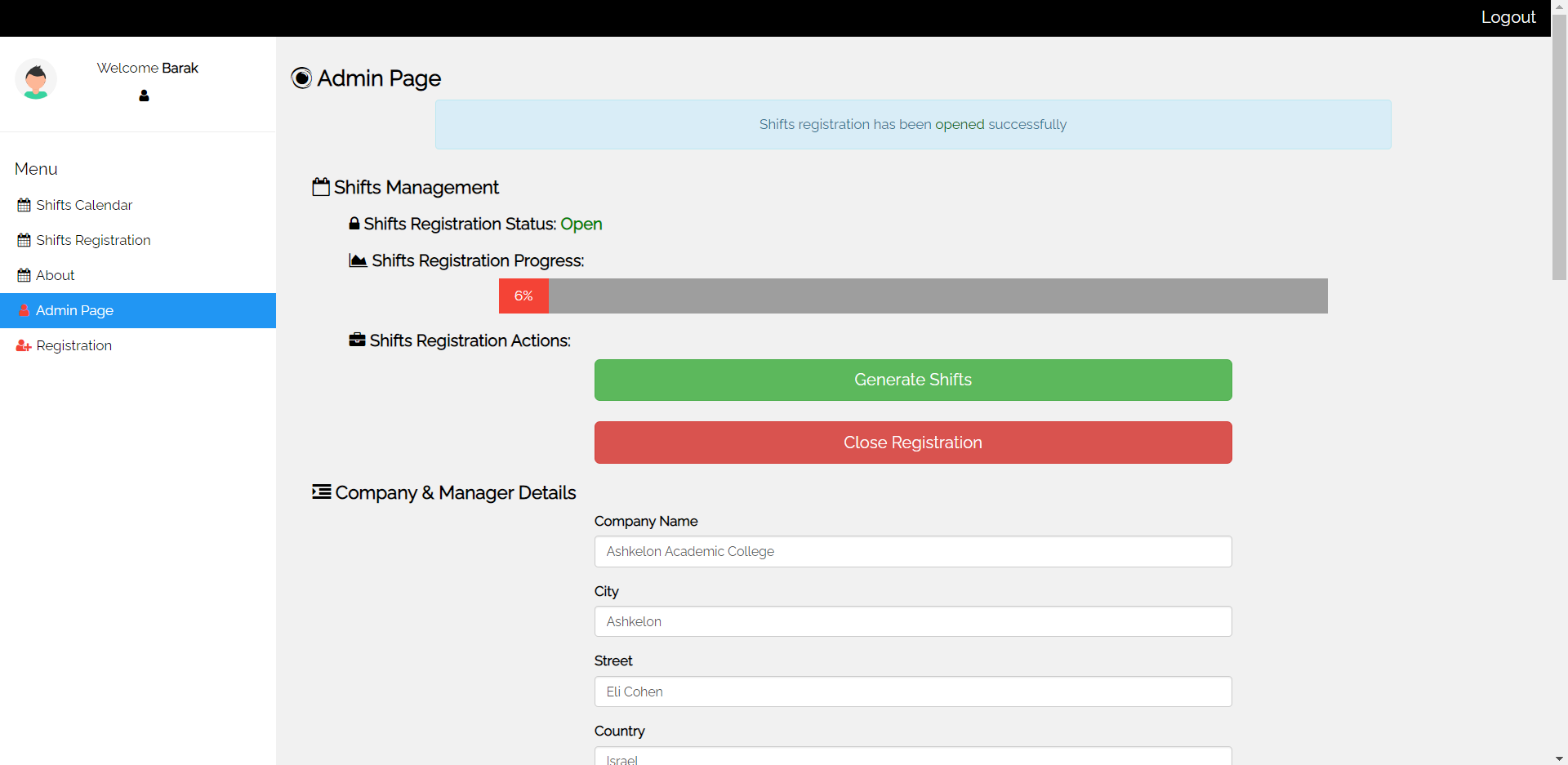
במידה והמנהל ירצה לפתוח את ההרשמה, עליו לבחור:

* + - מספר משמרות ביום (בין 1 ל-4).
    - חודש **עתידי** – כיוון שאין טעם לפתוח הרשמה לחודשים קודמים או לחודש הנוכחי, המערכת מחייבת לבחור בחודש עתידי.

בלחיצה על הכפתור "Open Registration" ההרשמה תיפתח, ותוצג למנהל ההודעה:



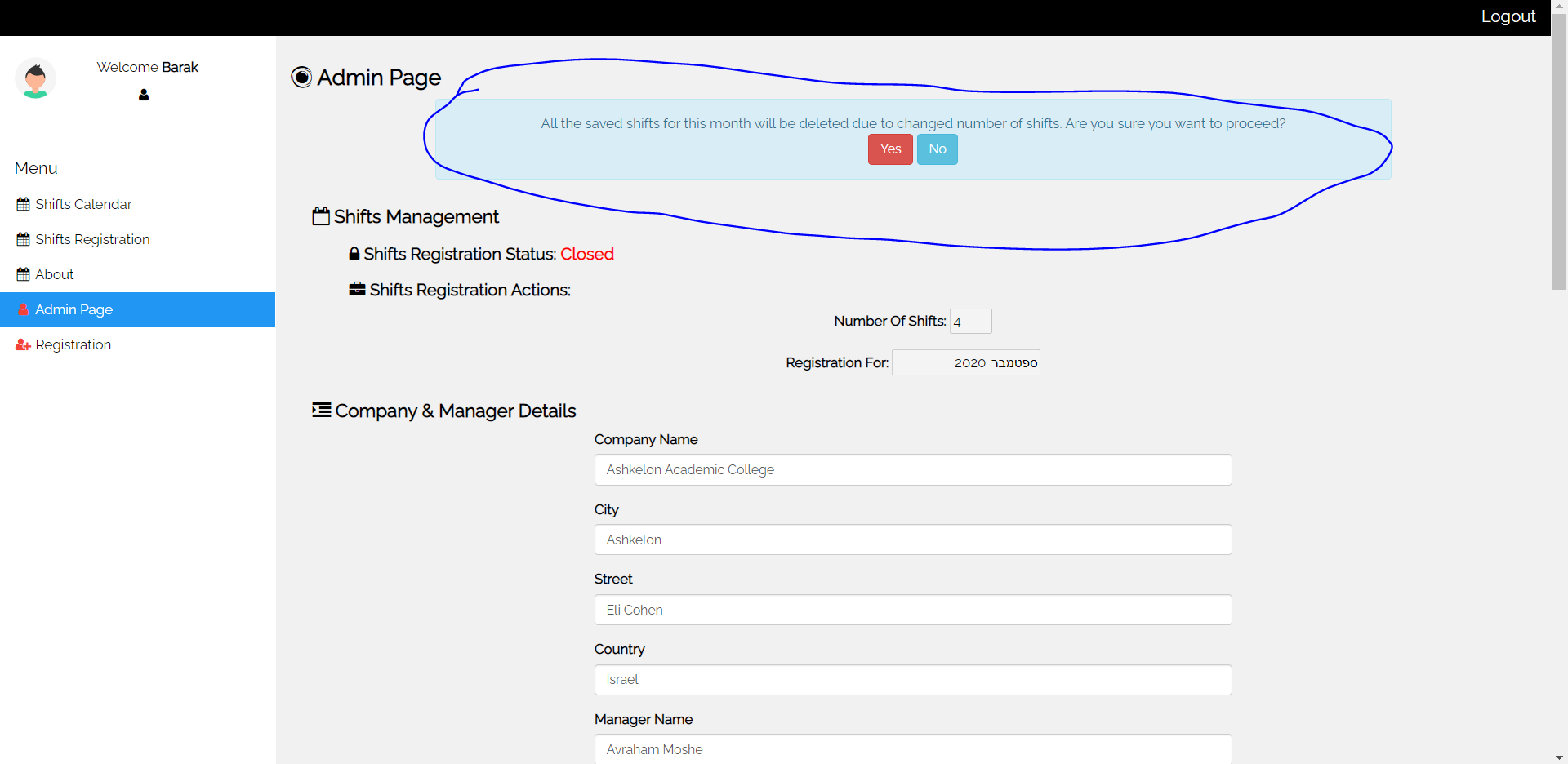
* **הרשמה פתוחה:**



כפי שניתן לראות, התצוגה של "Shifts Management" השתנתה:

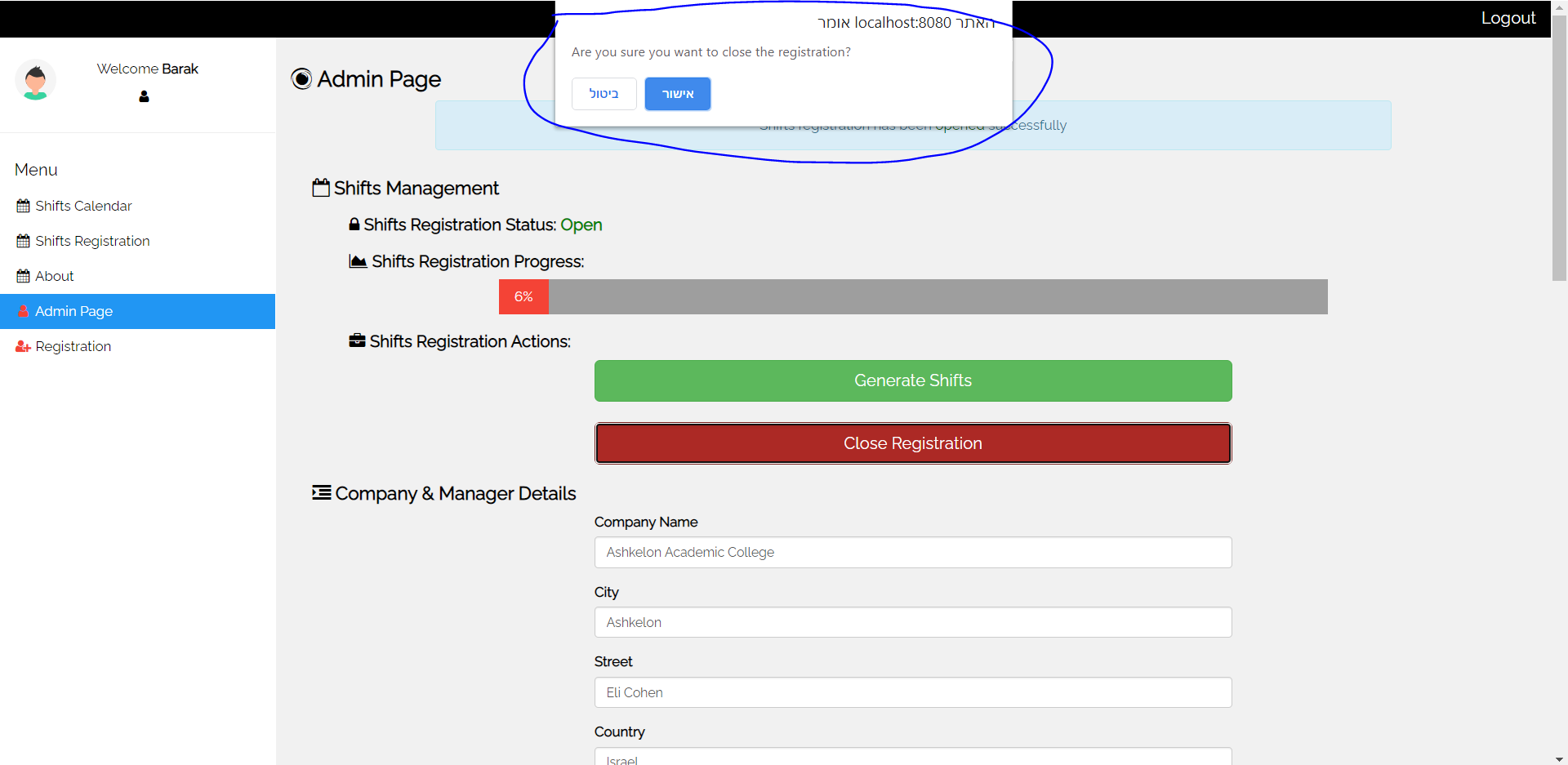
* + - כעת הסטטוס של ההרשמה הוא "Open".
    - ישנו גרף שאומר למנהל כמה אחוז נרשמו (צבעו משתנה בהתאם לאחוזים: עד 25% צבעו אדום, בין 25%-75% צבעו צהוב ומעל 75% צבעו ירוק.
    - פעולות:
    - **Generate Shifts** – בלחיצה על כפתור זה, המערכת תייצר לוח משמרות כאשר היא ממקסמת את סיפוק הבקשות של העובדים. בנוסף ההרשמה אוטומטית תיסגר וכעת ניתן לצפות בלוח המשמרות אילו משמרות קיבל כל עובד (במידה וכבר קיים לוח משמרות לחודש זה, הלוח הישן ימחק ובמקומות יהיה את הלוח משמרות החדש).

**מקרה חריג:** במידה וכבר הייתה הרשמה לחודש מסויים (ולכן יש מידע שמור על משמרות בחודש זה) וכעת כשהמנהל פותח את ההרשמה הוא משנה את מספר המשמרות (לדוגמה: אם כבר הייתה הרשמה עם שלוש משמרות ביום, וכעת המנהל פותח את ההרשמה עם ארבע משמרות ביום), תופיע הודעה שתזהיר את המנהל שכל המידע השמור יימחק:



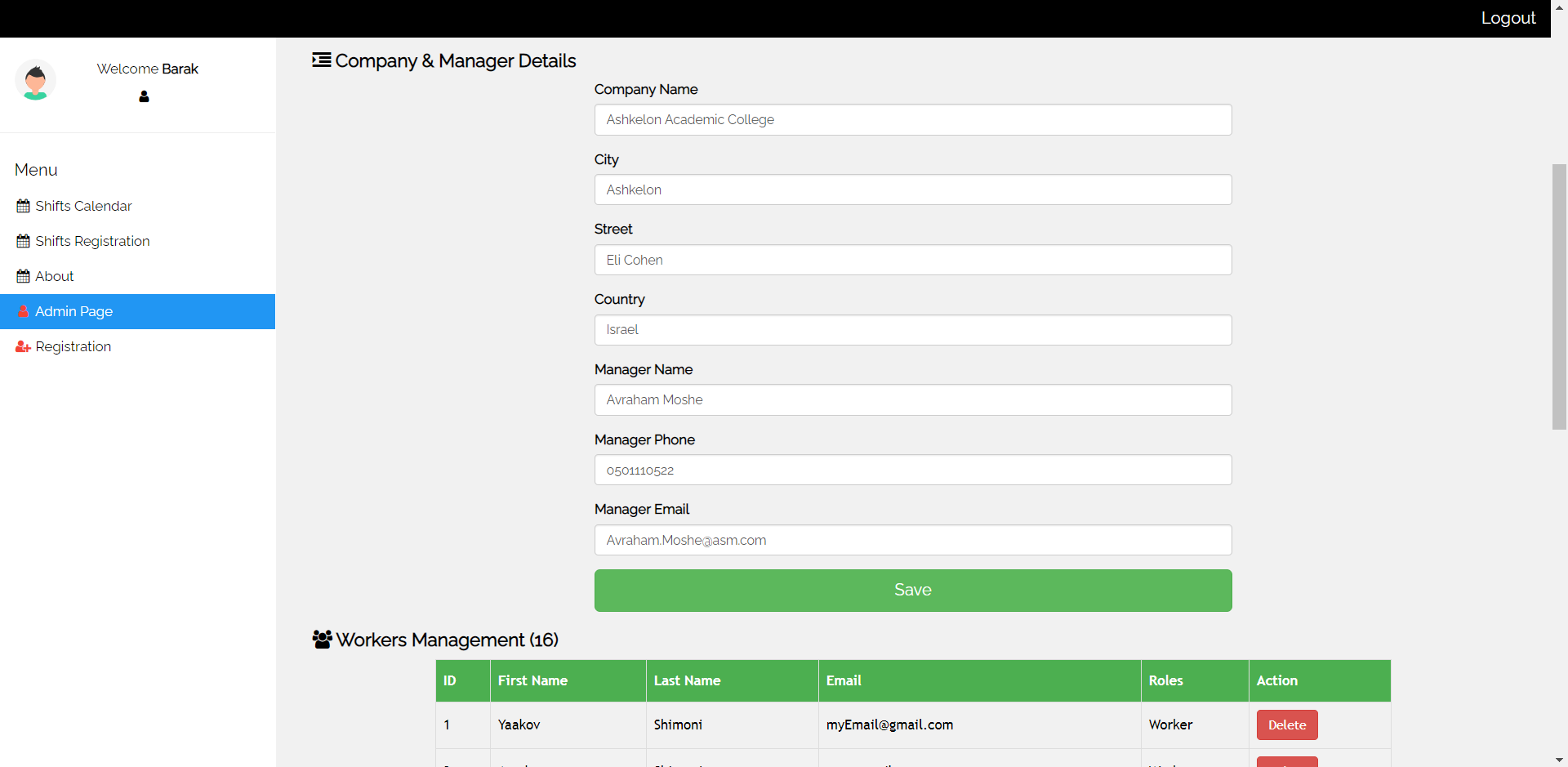
רק במידה והמנהל ילחץ "Yes" ההרשמה תפתח עם ההגדרות החדשות, והמידע הישן שהיה שמור יימחק.

* + - **Close Registration** – במידה והמנהל ירצה לסגור את ההרשמה, יופיע חלון (alert) שמוודא שהוא אכן רוצה לעשות זאת:



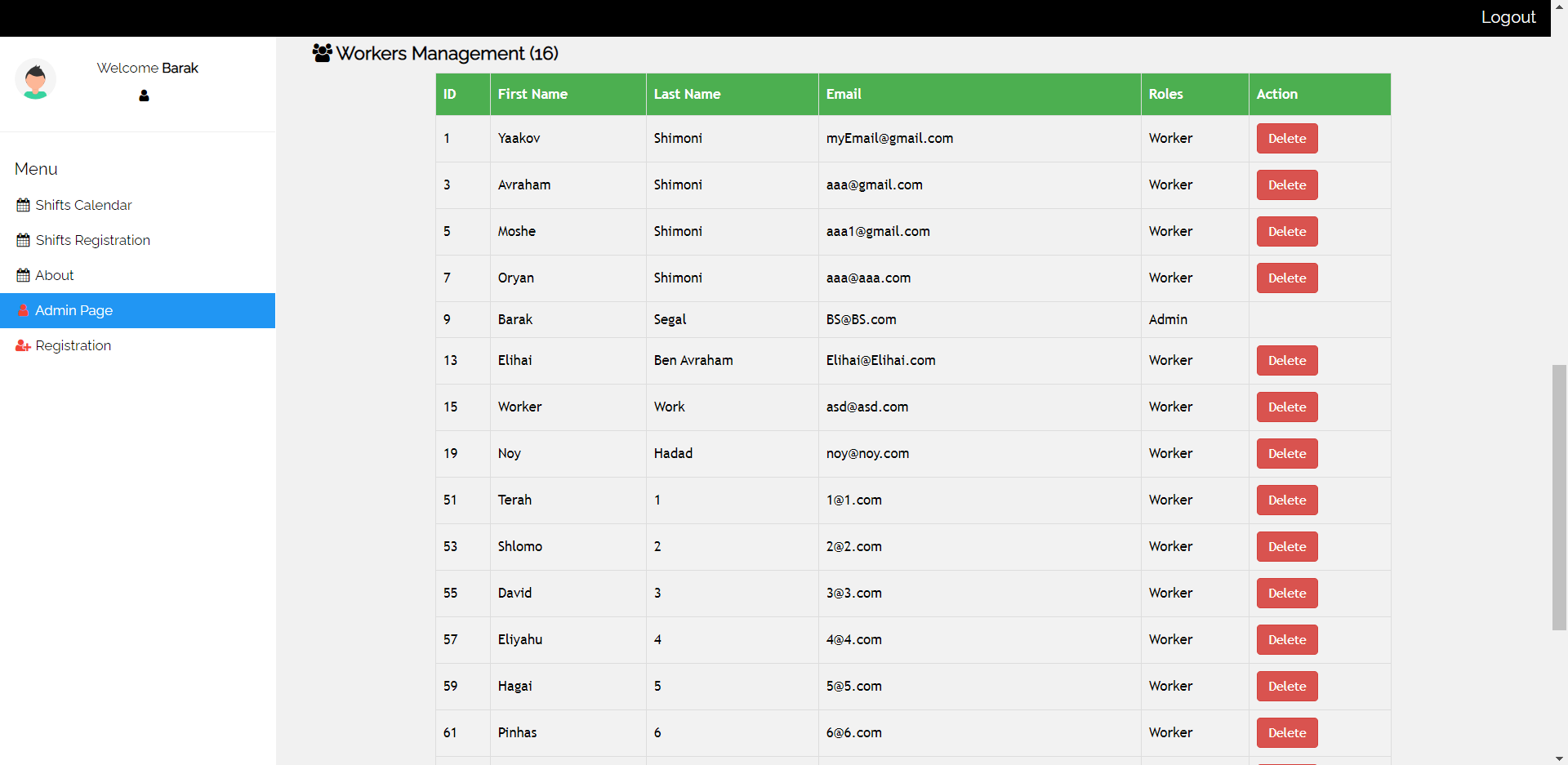
רק לאחר שהמנהל יאשר – ההרשמה תיסגר ולא תתאפשר הרשמה למשמרות.

* **Company & Manager Details** –



בחלק זה המנהל יכול לערוך את:

* + פרטי החברה: שם החברה, עיר, רחוב ומדינה.
  + פרטי מנהל משמרות: שם מלא, מספר טלפון וכתובת מייל.
* **Workers Management** –

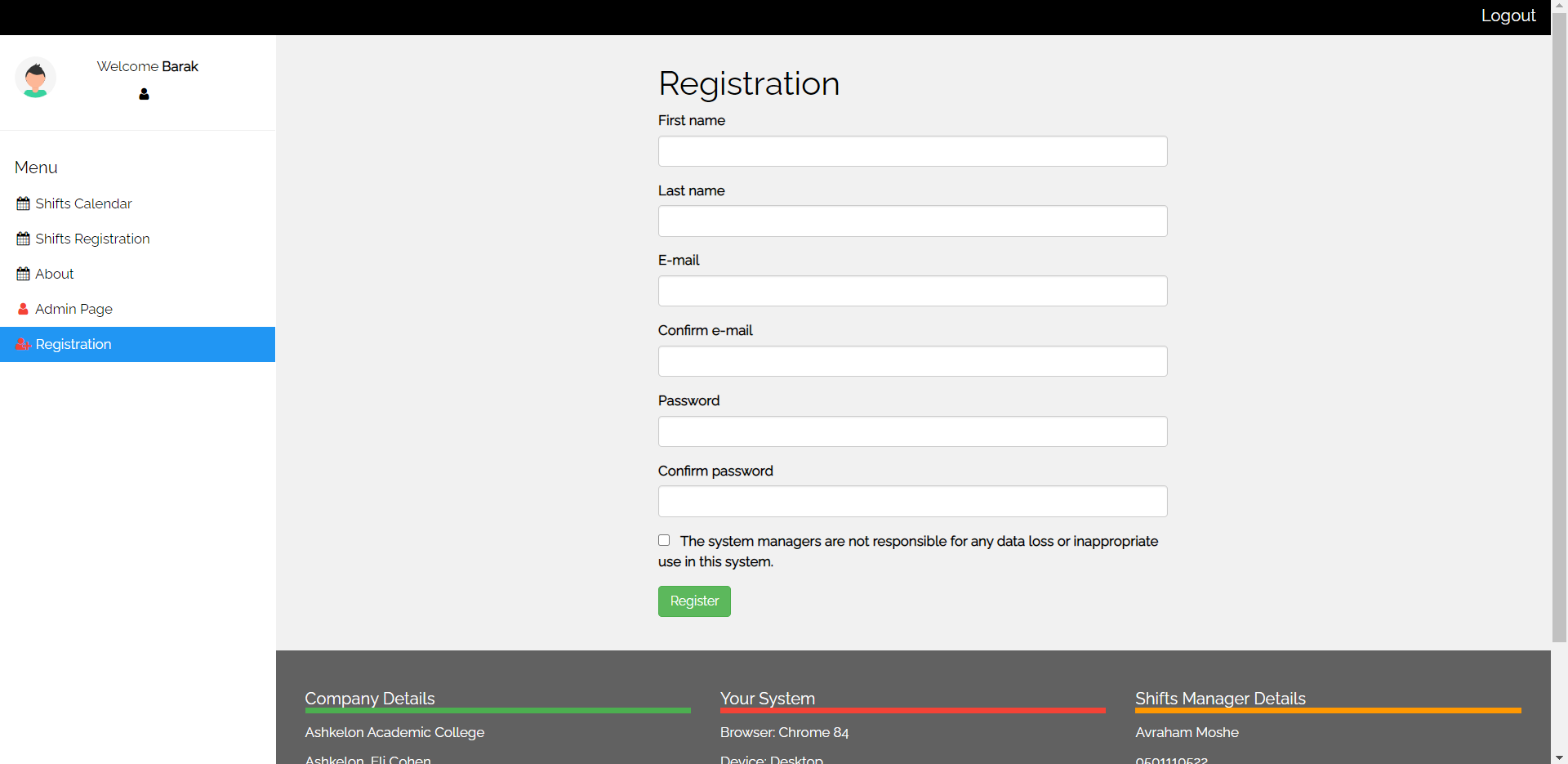


כפי שניתן לראות בתמונה לעיל, בחלק זה המנהל יכול לצפות בפרטים של המשתמשים וכן למחוק משתמשים.

הערות:

* + - * בשורה של הכותרת מופיע בסוגריים מספר העובדים.
      * בשורה שבה מופיעים פרטים של משתמש מנהל – אין כפתור של "Delete". מחיקת מנהל חייבת להתבצע באופן ידני. המערכת בנוי כך בכדי להגן על מנהלי המערכת.

1. **Registration** –



בדף זה המנהל יכול לרשום עובדים חדשים. יש למלא את הפרטים הבאים:

* + - שם פרטי.
    - שם משפחה.
    - כתובת מייל תקינה – כתובה באנגלית ומספרים, יש סטרינג לפני ה-"@" וכן לאחריו ויש לפחות נקודה אחת לאחר ה-"@". לדוגמה:

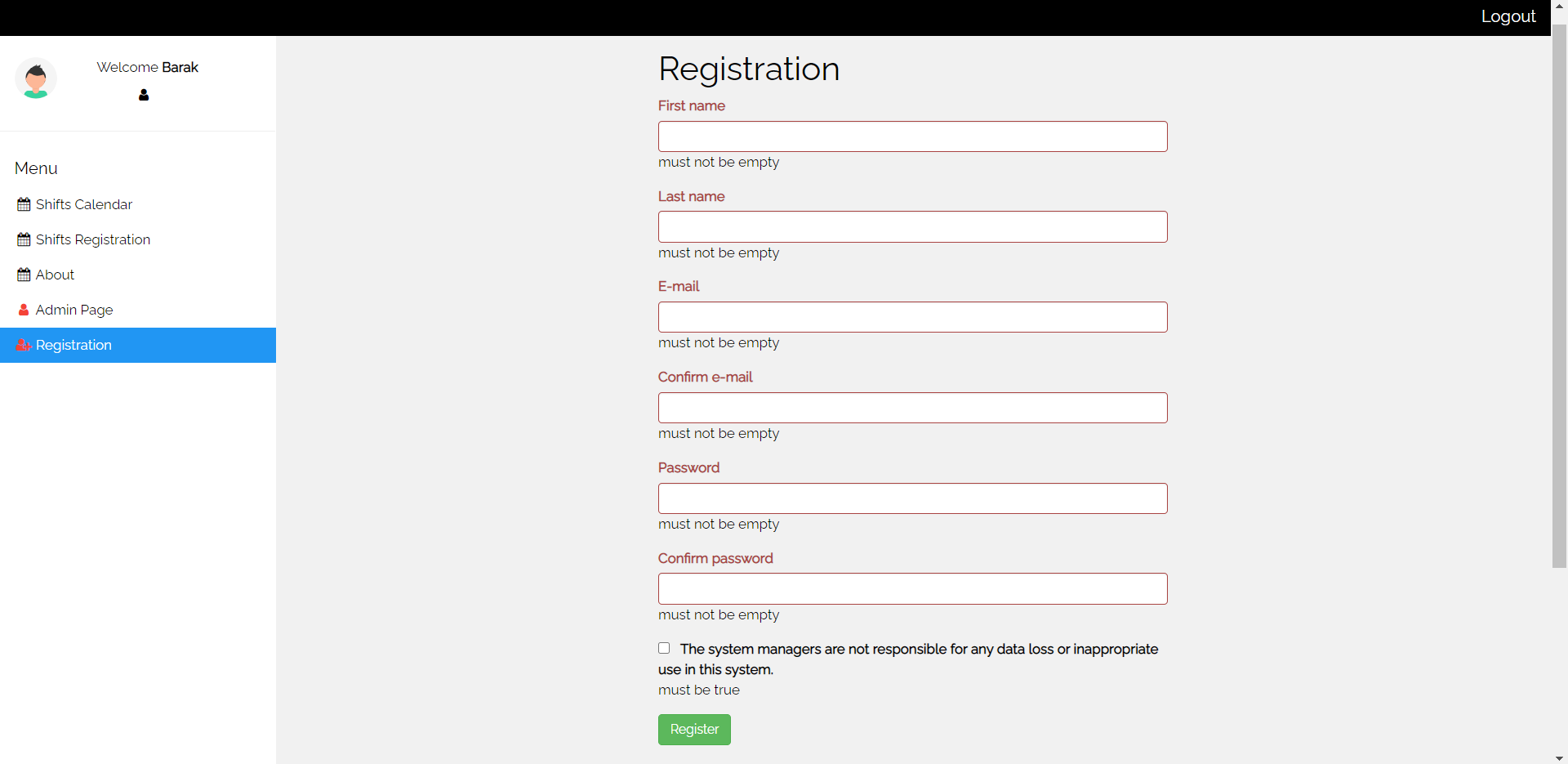
[String@String.String](mailto:String@String.String) או [String@String.String.com](mailto:String@String.String.com) וכדומה.

* + - סיסמה.
    - לאשר שהוא מסכים לתנאים (תמונה לעיל).

בלחיצה על "Register" המערכת תאמת את הנתונים הבאים:

* כל השדות מולאו (אין שדות ריקים).
* לא קיים שם משתמש עם כתובת מייל זהה.
* תקינות של כתובת המייל שהוזנה.
* השדות של "E-mail" ו-"Confirm e-mail" זהים.
* השדות של "Password" ו-"Confirm password" זהים.
* המשתמש אישר והסכים לתנאים.

במידה ואחד מהתנאים לעיל לא התקיים, תוצג הודעת שגיאה מתאימה. לדוג':

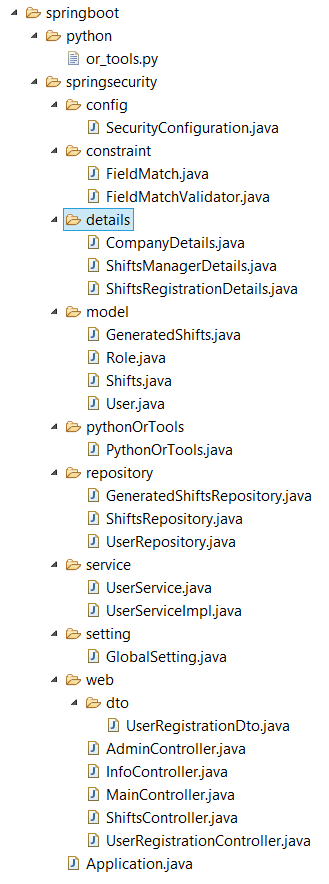


# מבנה הקוד

הקוד מכיל קובץ "pom.xml" המכיל קונפיגורציות, הגדרות ותלויות של Spring boot.

בנוסף, הקוד מחולק למספר תיקיות:

1. תיקיית "**springboot**":



בתיקייה זו ישנה מחלקה אחת:

**Application.java** – תפקידה של מחלקה זו הוא רק להריץ את ה-main ולהפעיל את האפליקציה.

**בנוסף, בתיקייה זו יש מספר תתי תיקיות:**

* **Python** –

**or\_tools.py:** קובץ הפייתון שמייצר את לוח המשמרות. קובץ זה משתמש בסיפריית OR-Tool של גוגל.

בתהליך חישוב המשמרות המערכת ממקסמת סיפוק בקשות. כלומר, היא מאשרת כמה שיותר בקשות של עובדים (לדוג' אם היו 100 בקשות למשמרות סה"כ מכל העובדים, היא תשבץ את העובדים במשמרות כך שכמה שיותר עובדים ישובצו במשמרות שהם שביקשו).

במידה ויש משמרות שאף עובד לא ביקש – המערכת תחלק אותן בין העובדים.

* **Springsecurity** – מכילה כמעט את כל הקבצים של ה"Back end":
* **Config** –

**SecurityConfiguration.java** – קובץ הקונפיגורציות של האבטחה (לאילו עמודים יכול כל אחד לגשת, הצפנת סיסמאות וכו').

* **Constraint** – מכילה אילוצי טפסים. במקרה שלנו מדובר באילוצים של טופס ההרשמה למשתמש חדש (אימייל תקין, אימות סיסמה ומייל וכו'). ישנם שני קבצים שאחראים על אימות זה:

**FieldMatch.java** – ממשק אבסטרקטי של המתודות שנדרשות למימוש.

**FieldMatchValidator.java** – מחלקה שמוודאת שהאילוצים בטופס סופקו כראוי.

* **Details** – מכילה מחלקות עם הפרטים על המערכת:

**CompanyDetails.java** – מחלקה המכילה את הפרטים על החברה. מכילה את השדות: שם החברה, מדינה, עיר ורחוב.

**ShiftsManagerDetails.java** – מחלקה המכילה את הפרטים על מנהל המשמרות. מכילה את השדות: שם מלא, כתובת מייל ומספר טלפון.

**ShiftsRegistrationDetails.java –** מחלקה המכילה את הפרטים על ההרשמה למשמרות (בשימוש רק כאשר יש הרשמה פתוחה). מכילה את השדות: האם ההרשמה פתוחה (Boolean), שנה, חודש, מספר משמרות.

* **Model** – על פי עיקרון הMVC (יוסבר בהמשך), תיקייה זו מכילה את המחלקות שאחראיות על המידע שנשמר. ממחלקות אלה יוצרים אובייקטים שנשמרים במסד הנתונים. המחלקות בתיקיה זו:

**GeneratedShifts.java** – עבור כל חודש שמיוצר עבורו לוח משמרות, נוצר אוביקט ממחלקה זו שמכיל את כל המידע הנדרש. מחלקה זו מכילה את השדות: id, חודש, שנה, מספר משמרות ביום, והמשמרות שנשמרו.

**Role.java** – מחלקה זו אחראית על התפקידים במערכת. במקרה שלנו יש שני תפקידים: Admin ו-Worker (עובד רגיל). עבור כל תפקיד שיצרנו ישנו אובייקט שמכיל את המידע עליו. השדות במחלקה זו: id ושם.

**Shifts.java** – מחלקה זו שומרת את המידע של המשמרות שמולאו על ידי משתמשים בהרשמה. כאשר ההרשמה פתוחה, עבור כל משתמש שהזין את המשמרות שהוא מעוניין בהן, נוצר אובייקט ממחלקה זו ששומר את המשמרות שהוא בחר עם כל המידע הנדרש. מחלקה זו מכילה את השדות: id, מייל, שנה, חודש ומשמרות שהמשתמש בחר.

**User.java** – עבור כל משתמש שנרשם למערכת – נוצר אובייקט ממחלקה זו ששומר את כל המידע עליו. מחלקה זו מכילה את השדות: id, שם פרטי, שם משפחה, כתובת מייל, סיסמה ותפקיד (Admin או Worker).

* **PythonOrTools** – תיקייה זו מכילה את המחלקה:

**PythonOrTools.java** – אחראית על התקשורת בין התהליך של קובץ הפייתון (שמחשב את המשמרות כפי שהסברנו לעיל) לבין תוכנית ה-Java שלנו.

* **Repository** – מכילה ממשקים שמנהלים את העברות הנתונים בין השרת למסד הנתונים:

**GeneratedShiftsRepository.java** – מחלקה שאחראית על העברת אובייקטים מסוג "GeneratedShifts" (לוחות משמרות, כפי שהסברנו לעיל).

**ShiftsRepository.java** – מחלקה שאחראית על העברת אובייקטים מסוג "Shifts" (הרשמה למשמרות של משתמשים).

**UserRepository.java** – מחלקה שאחראית על העברת אובייקטים של משתמשים.

* **Service** – מכילה מחלקה וממשק שתפקידם לתווך בין UserRegistratiobDto (יוסבר בהמשך) לבין User.java.

**UserService.java** – ממשק עם רשימה של מתודות למימוש.

**UserServiceImpl.java** – מחלקה שמממשת את הממשק הנ"ל. מחלקה זו אחראית בעצם על התיווך בין בדיקת משתמש חדש שנרשם (משתמש זה נבדק ע"י המחלקות בתיקייה של constraint לעיל) ליצירת משתמש חדש במידה ועבר את הבדיקות.

* **Setting** –

**GlobalSetting.java** – מחלקה זו מכילה הגדרות כלליות של המערכת. מכילה את השדות: שם האתר, חודש עדכני ושנה עדכנית.

* **Web** – תיקייה זו אחראית על החלק של הControllerים (לפי החלוקה של MVC).

**תת תיקייה dto** – ר"ת של: "Data Transfer Object" - שזה בעצם אובייקט שמחזיק מידע שמועבר בין שני תהליכים. תיקייה זו מכילה מחלקה אחת:

**UserRegistrationDto.java** – אחראית לתיווך בין יצירת משתמש חדש לבין בדיקת כלל האילוצים הנדרשים בהרשמה.

בנוסף תיקיית Web מכילה מחלקות נוספות שאחראיות על הפונקציונליות בדפים השונים באתר (כלומר, כשמשתמש מבצע פעולה או מבקש לגשת לעמוד מסויים, אלו המחלקות שאחראיות לספק לו את השירות הנדרש):

**AdminController.java** – אחראי על הפעולות שמבוצעות בעמוד של המנהל.

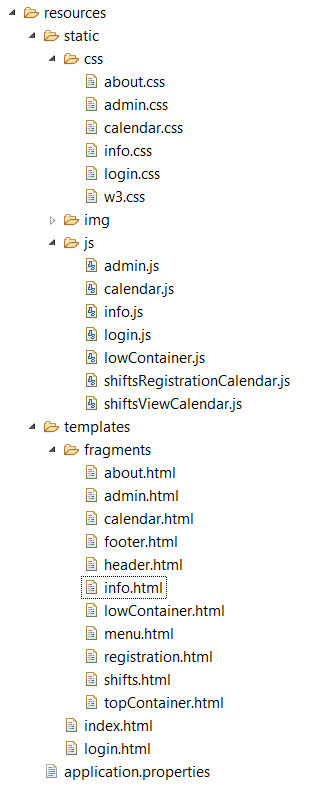
**InfoController.java** – אחראי על הפעולות שמבוצעות בדף של המידע.

**ShiftsController.java** –אחראי על הפעולות שמבוצעות בדפים של "Shifts Calendar" ושל "Shifts Registration".

**UserRegistrationController.java** – אחראי על הפעולות בטופס ההרשמה של משתמש חדש.

**MainController.java** –אחראי על הפעולות בשאר הדפים (התחברות ו-about).

1. תיקיית "**resources**":



בתיקיה זו יש קובץ אחד של הגדרות וכן תיקיות עם הקבצים של הצד לקוח (Front end). תיקייה זו מכילה את כל הקבצים של התצוגה למשתמש (View לפי החלוקה של MVC):

* **Application.properties –** קובץ הגדרות של המערכת. מכיל את פרטי ההתחברות למסד הנתונים, הפורט שבו תרוץ האפליקציה, הגדרות של Hibernate (יוסבר בהמשך) והגדרות כלליות של המערכת.
* **Static –** מכילה את כל הקבצים של הצד לקוח (Front end). מתחלקת לשלוש תתי תיקיות:

CSS – מכילה את קבצי העיצוב:

* + about.css – עיצוב של דף המידע על מנהלי המערכת.
  + admin.css – עיצוב של דף המנהל.
  + calendar.css – עיצוב של לוח שנה.
  + info.css – עיצוב של עמוד פרטי המשתמש.
  + login.css – עיצוב של עמוד ההתחברות.
  + w3.css – סיפרייה חיצונית של עיצוב.

Img – מכילה תמונות שהמערכת משתמשת בהן (תמונות של מנהלי המערכת שמוצגים בעמוד של About וכו').

Js – מכילה את קבצי הJavaScript:

* + admin.js – מכיל קוד שרץ כאשר נכנסים לעמוד המנהל.
  + calendar.js – מכיל קוד שרץ כאשר פותחים לוח שנה.
  + info.js – קוד שאחראי על וולידציה של שינוי סיסמה.
  + login.js – מכיל קוד שרץ כשנכנסים לעמוד ההתחברות.
  + lowContainer.js – קוד שבודק את נתוני המערכת של המשתמש.
  + shiftsRegistrationCalendar.js – קוד שרץ כשנכנסים להרשמה למשמרות.
  + shiftsViewCalendar.js – קוד שרץ שנכנסים לעמוד של צפייה בלוח משמרות.
* **Templates** – מכיל את קבצי הHTML של המערכת. קבצי הHTML הם דינאמיים (משתנים בהתאם לבקשות של המשתמש) ומנוהלים על ידי Thymeleaf (יוסבר בהמשך):

**login.html** – קובץ HTML של ההתחברות.

**index.html** – קובץ HTML שמוצג למשתמש בכל הדפים (חוץ מעמוד ההתחברות), כאשר בכל עמוד הוא משתמש בקבצי HTML נוספים מהתיקייה של **fragments** בהתאם לבקשת המשתשמש:

* + about.html – במידה והמשתמש ביקש את דף המידע על מנהלי המערכת, קובץ הindex יכיל דף זה.
  + admin.html – במידה והמשתמש ביקש את דף המנהל, קובץ הindex יכיל דף זה (רק למנהל יש גישה לדף זה).
  + calendar.html – במידה והמשתמש ביקש דף עם לוח שנה (תצוגת לוח משמרות או הרשמה למשמרות), קובץ הindex יכיל דף זה.
  + footer.html – קובץ זה מופיע בכל העמודים, ולכן תמיד index יכיל אותו. קובץ זה מציג את הזכויות יוצרים.
  + header.html – מכיל את כל המידע הנצרך לתגית ה<head>. כיוון שההגדרות בתגית ה<head> נצרכות בכל דפי המערכת, קובץ הindex תמיד יכיל אותו.
  + info.html – במידה והמשתמש ביקש את דף המידע האישי, קובץ הindex יכיל דף זה.
  + lowContainer.html – קובץ זה מופיע בכל העמודים, ולכן תמיד index יכיל אותו. זהו הקובץ שמציג את החלק התחתון של האתר (עם הפרטים על החברה, מנהל המשמרות והמערכת של המשתמש).
  + menu.html – קובץ זה מופיע בכל העמודים, ולכן תמיד index יכיל אותו. זהו הקובץ שאחראי על התצוגה של התפריט.
  + registration.html – במידה והמשתמש ביקש את דף ההרשמה לעובד חדש, קובץ הindex יכיל דף זה (רק למנהל יש גישה לדף זה).
  + shifts.html – במידה והמשתמש ביקש את הדף של תצוגת לוח משמרות או הרשמה למשמרות, קובץ הindex יכיל דף זה.
  + topContainer.html – קובץ זה מופיע בכל העמודים, ולכן תמיד index יכיל אותו. זהו הקובץ שמציג את החלק העליון של האתר (הפס השחור עם כפתור ההתנתקות).

# Spring Boot

Spring היא שלד תוכנה בקוד פתוח ומנגנון להיפוך שליטה[[2]](#footnote-2) (Inversion of Control) עבור הפלטפורמה של Java[[3]](#footnote-3). בעזרת סיפריה זו ניתן ליצור אפליקציות Java שונות. Spring Boot היא הרחבה של סיפריית Spring שמפחיתה משמעותית את הקונפיגורציות הנדרשות בשביל לייצר אפליקציה של Spring.

אנו השתמשנו במספר סיפריות שמשתלבות עם Spring:

1. **Spring Security** – סיפריה שמספקת שירותי אבטחה. בעזרת סיפריה זו ביצענו בקרת גישה ואימות זהות של המשתמשים. כמו כן, בעזרת סיפריה זו ביצענו הצפנה של הסיסמאות של המשתמשים ע"י פונקציית הצפנה חד סטרית. בנוסף סיפרייה זו מאפשרת להוסיף קטעי קוד בתוך קבצי הHTML בכדי לקבל פרטים על המשתמש שמחובר (כמו: מה רמת ההרשאה שלו, מה האימייל שלו וכו').
2. **Hibernate** – ספריה שממפה אובייקטים למסד נתונים טבלאי. לדוגמה, אם יצרתי אובייקט מסוג User ואני מעוניין לשמור אותו במסד נתונים טבלאי, ניתן לעשות זאת באמצעות שימוש בפונקציות של Hibernate. בעזרת ספריה זו ניתן לבצע שאילתות בצורה פשוטה, ולקבל את תוצאות השאילתא ישירות לתוך אובייקטים שאנו מגדירים.



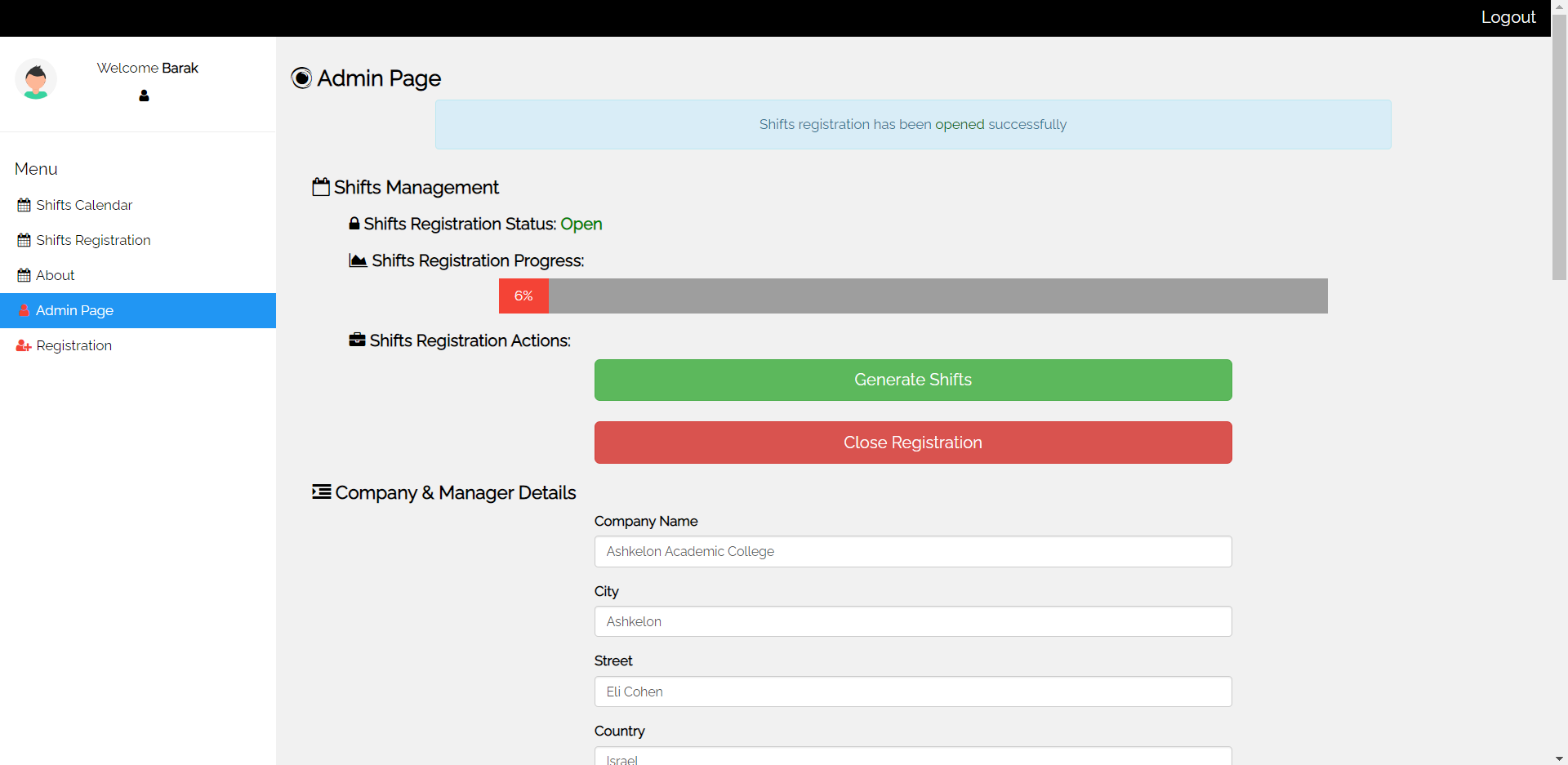
סיפריה זו אינה גובה תקורת ביצועים גובהה, ובעזרתה חסכנו את הצורך בטיפול ידני משאילתות על בסיס הנתונים, והמרות בין אובייקטים.

המיפוי בין מחלקות של Java לטבלאות של בסיס נתונים יכול להתבצע באמצעות XML או על ידי שימוש באנוטציות Java. אנו השתמשנו באנוטציות של Java[[4]](#footnote-4).

1. **Thymeleaf** – מנוע מודרני בשפת Java הפועל בצד שרת עבור פיתוח Web וכן עבור פיתוח סביבות עצמאיות (standalone). סיפרייה זו מאפשרת לכתוב קטעי קוד בתוך קובץ HTML סטטי, וכן להכניס לדף HTML מידע ששמור בצד של השרת, ובכך להפוך את קבצי הHTML לדינאמיים. לדוגמה:



בקטע קוד לעיל, הגדרנו תגית div שמקבלת משתנה שנקרא "pageId" (משתנה זה הוא בצד של השרת). תגית זאת שולטת במה שמוצג בחלק שמסומן באדום בתמונה דלהלן:



ע"י המשתנה "pageId" אנו מקבלים נתיב שמייבא את תגית הdiv שאנו רוצים ומוסיף אותה לקובץ הHTML שלנו. למשל, בתמונה לעיל, המנהל ניגש לעמוד הניהול. כלומר, המשתנה "pageId" יקבל את הערך "admin", ומציג את עמוד המנהל (בהנחה ויש לו גישה לשם). בנוסף, ע"י המשתנה "pageId", המערכת יודעת איזה חלק בתפריט לצבוע בכחול (כפי שניתן לראות בתמונה לעיל בתפריט בצד שמאל).

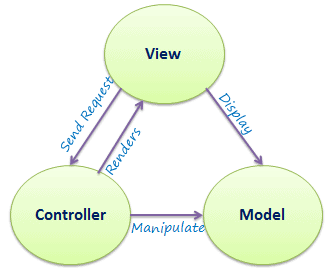
1. **Spring MVC** – סביבת עבודה שמספקת ארכיטקטורת Model-View-Controller (MVC), המאפשרת לפתח אפליקציות Web גמישות לשינויים ע"י הפרדה בין החלקים השונים של המערכת:

* **Model** – מכיל את המידע (data). במקרה שלנו מדובר במידע נשמר בשרת MySQL ומנוהל ע"י Hibernate.
* **View** – התצוגה למשתמש. במקרה שלנו מדובר בדפי הHTML שמוצגים למשתמש (קבצי HTML מכילים גם קבצי CSS ו-JavaScript).
* **Controller** – מקשר בין שני החלקים הנ"ל. כלומר, אחראי על עיבוד בקשות המשתמש, בניית עמוד מתאים לתצוגה ושליחתו למשתמש. במקרה שלנו, ישנם מספר קבצי Controller שאחראים על כך (קבצי java):

MainCntroller, AdminController, InfoController, ShiftsController, UserRegistrationController.

הפירוט על כל אחד מהקבצים הנ"ל נמצא בחלק "מבנה הקוד".

**תמונה להמחשת ארכיטקטורת MVC:**



# Databases

**SQL**‏ (**S**tructured **Q**uery **L**anguage) היא שפת מחשב הצהרתית לטיפול ועיבוד מידע בבסיסי נתונים יחסיים. כלומר, היא מתארת רק מה לבצע ולא כיצד לבצע, ומתמקדת במהות ולא בטכניקה. מכיוון שכך, ניתן להשתמש באותה הצהרת SQL בבסיסי נתונים שונים, הפועלים בדרכים שונות לחלוטין. אי תלותה של SQL במבנה הנתונים, בנוסף על עוצמתה של השפה, המאפשרת לאחזר חתכים מורכבים בפשטות רבה, דבר המביא להגדלת הפריון של מתכנתי השפה, והתפשטותה למשתמשים שאינם מתכנתים, הפכו אותה לשפה המקובלת ביותר כיום בניהול בסיסי נתונים. השפה מאפשרת שליפת נתונים ועדכונם ויצירת טבלאות ועדכונן.

בניגוד לשפות תכנות עיליות, סדר הביצוע של משפט SQL איננו מתבצע לפי סדר הכתיבה. בדרך כלל פסוקית SELECT תתבצע לקראת הסוף, כאשר הפסוקית הראשונה שתתבצע תהיה פסוקית FROM. כל שלב בעיבוד מייצר טבלה וירטואלית זמנית אשר מהווה את הקלט לשלב הבא.

מסדי נתונים יחסיים הם כיום הדרך המקובלת לאגירת מידע רב. שפת SQL היא הדרך המקובלת לגשת למסדי נתונים יחסיים. וגופים שונים כמו גופי ממשל, חברות מסחריות, וארגונים ללא כוונות רווח עושים בו שימוש רב, אם במסדי נתונים גדולים ואם במסדי נתונים מקומיים.

שפת SQL פותחה במקורה על ידי , אך מאז הועתקה על ידי חברות רבות. מערכות עיקריות לניהול בסיסי נתונים בשפה הן: Microsoft SQL Server, Oracle, MySQL, Microsoft Access, SQLite, PostgreSQL.

עבור בניית מאגר הנתונים של המערכת שלנו, השתמשנו במערכת MySQL.

****

**MySQL** הוא מסד נתונים יחסי ורב משתמשים מבוסס שפת שאילתות מובנית SQL‏ (**S**tructured **Q**uery **L**anguage). התוכנה פותחה במקור על ידי החברה השוודית MySQL AB, אך כיום היא בבעלות חברת . בנוסף, התוכנה היא תוכנה חינמית, אשר נכתבה בשפות C++ ו-C, ויכולה לפעול על מספר רב של מערכות הפעלה, כגון: , , ,  ועוד.

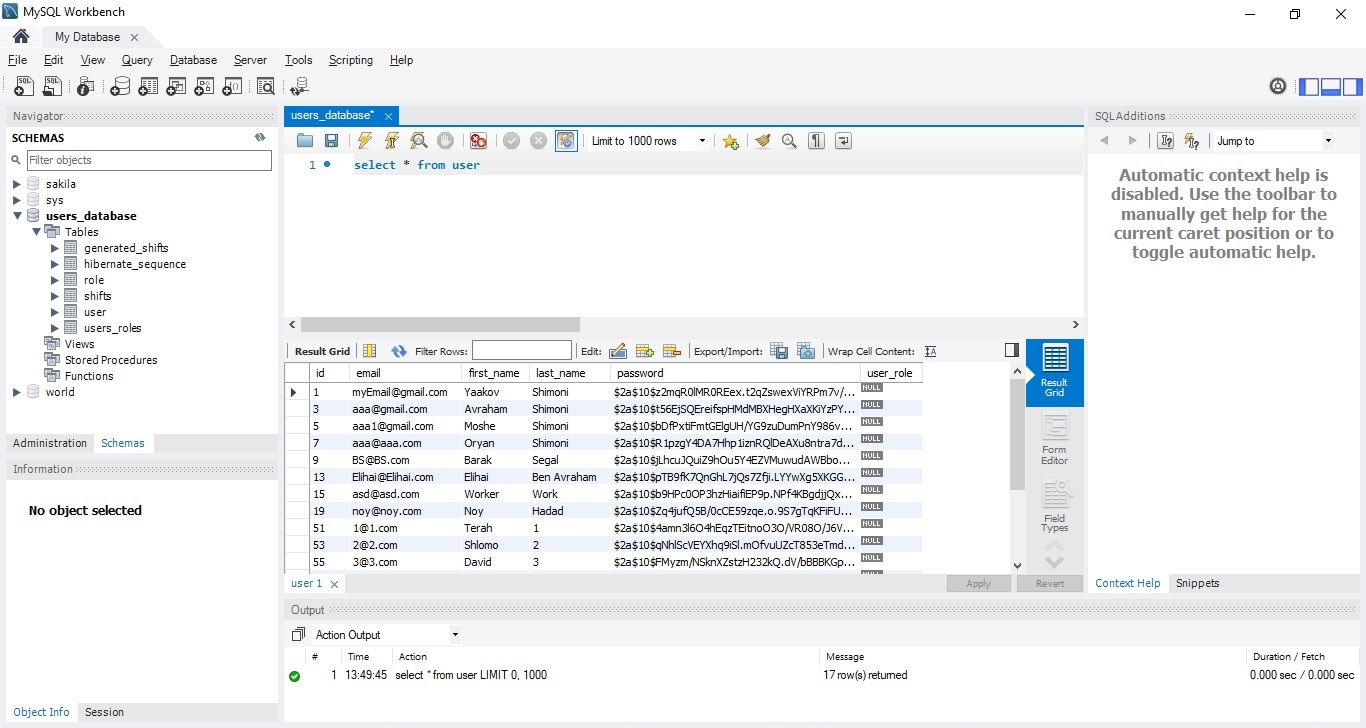
ישנם סוגים רבים של שרתי בסיסי נתונים, אך MySQL הוא אחד ממסדי הנתונים הפופולריים ביותר ברשת, במיוחד עם שפת PHP (למרות שניתן בהחלט להשתמש בו עם כל שפת תכנות אחרת).

הנתונים בשפת SQL מאוחסנים בטבלאות, וכך גם הנתונים ב-MySQL. מכיוון ש- SQL היא שפה המאפשרת גישה למידע השמור במסדי נתונים, אנחנו יכולים לבקש ממאגר הנתונים ב-MySQL מידע ספציפי באמצעות שאילתות, ולקבל בתוצאה את השורות העונות על אותן שאילתות. בשפת SQL ניתן להציג בעזרת שאילתות מידע ממסד הנתונים, אך ניתן גם לעשות שימוש אחר במידע זה כגון: לעדכן אותו באמצעות הפקודה UPDATE, למחוק אותו באמצעות הפקודה DELETE, להכניס מידע חדש באמצעות הפקודה INSERT וכו'.

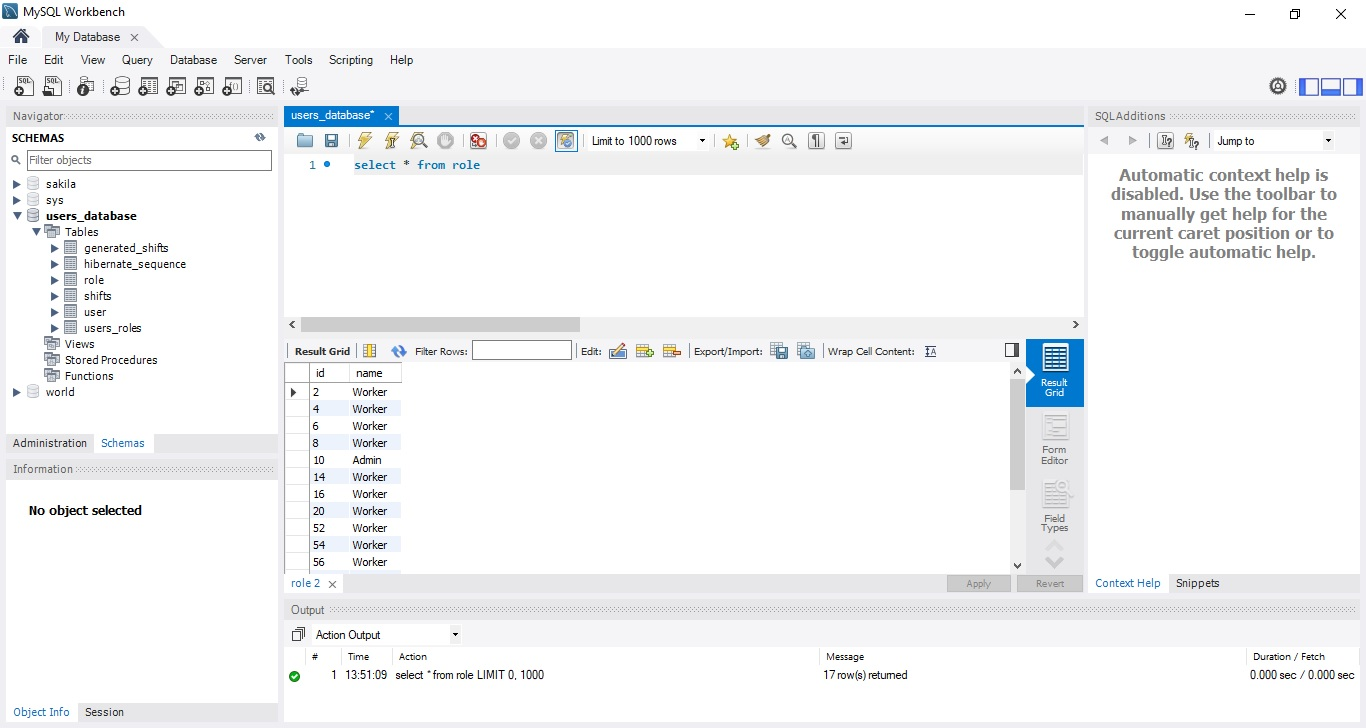
סביבת העבודה בה עבדנו עם MySQL נקראת MySQL Workbench. בעזרת תוכנה זוב ביצענו שאילתות, צפינו בטבלאות בצורה גרפית וביצענו פעולות נוספות, כמו שינוי הגדרות של טבלאות וכד'.

**מסד הנתונים בפרויקט שלנו מכיל 5 טבלאות, והן:**

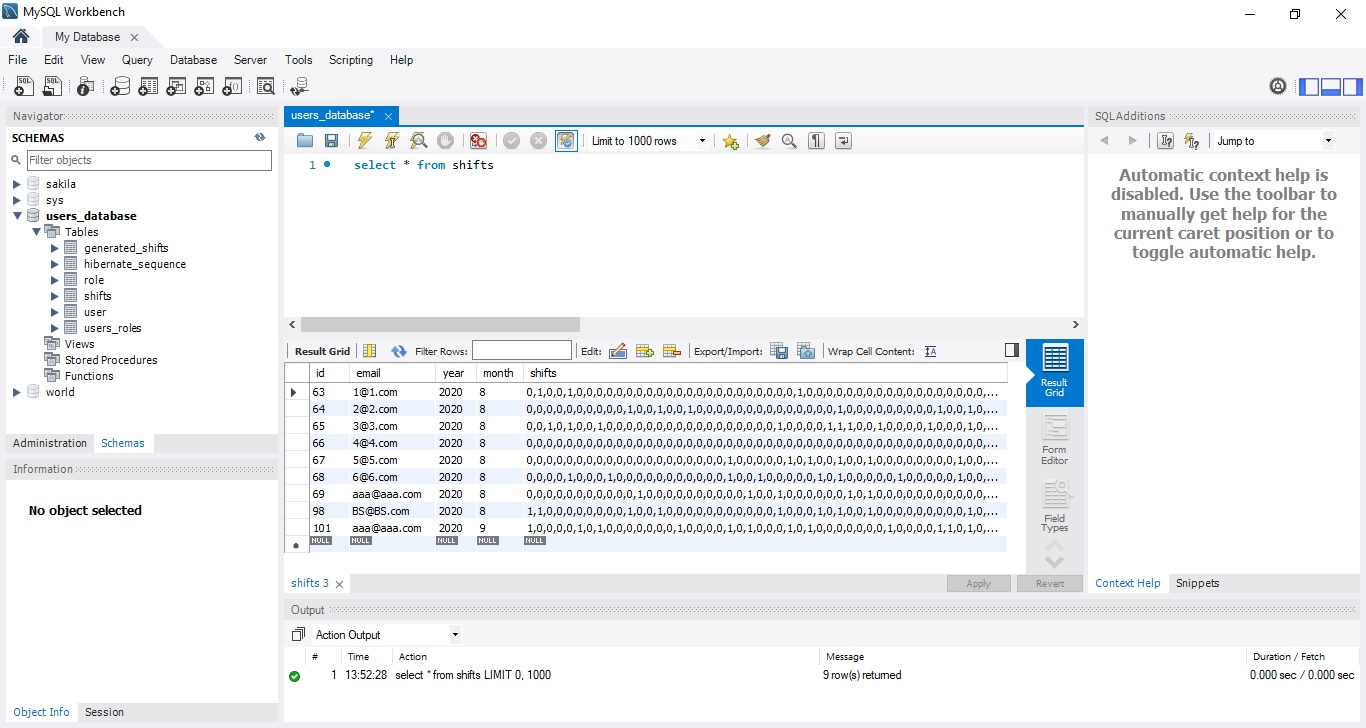
**user –** הטבלה user היא טבלה המכילה את הנתונים הבסיסיים על כל המשתמשים הרשומים למערכת. בטבלה זו, מאוחסנים הערכים הבאים: המפתח הראשי **id** (מס' הסידורי של משתמש מסוים במערכת), **email** (כתובת המייל שלו), **first\_name** (השם הפרטי שלו), **last\_name** (שם המשפחה שלו), **password** (סיסמת ההתחברות שלו למערכת, אשר מוצפנת ב-MySQL ולא ניתנת לפיענוח), ו-**user\_role** (תפקיד – מנהל או עובד). להלן שאילתה ב-MySQL המחזירה את כל הנתונים של טבלה זו:



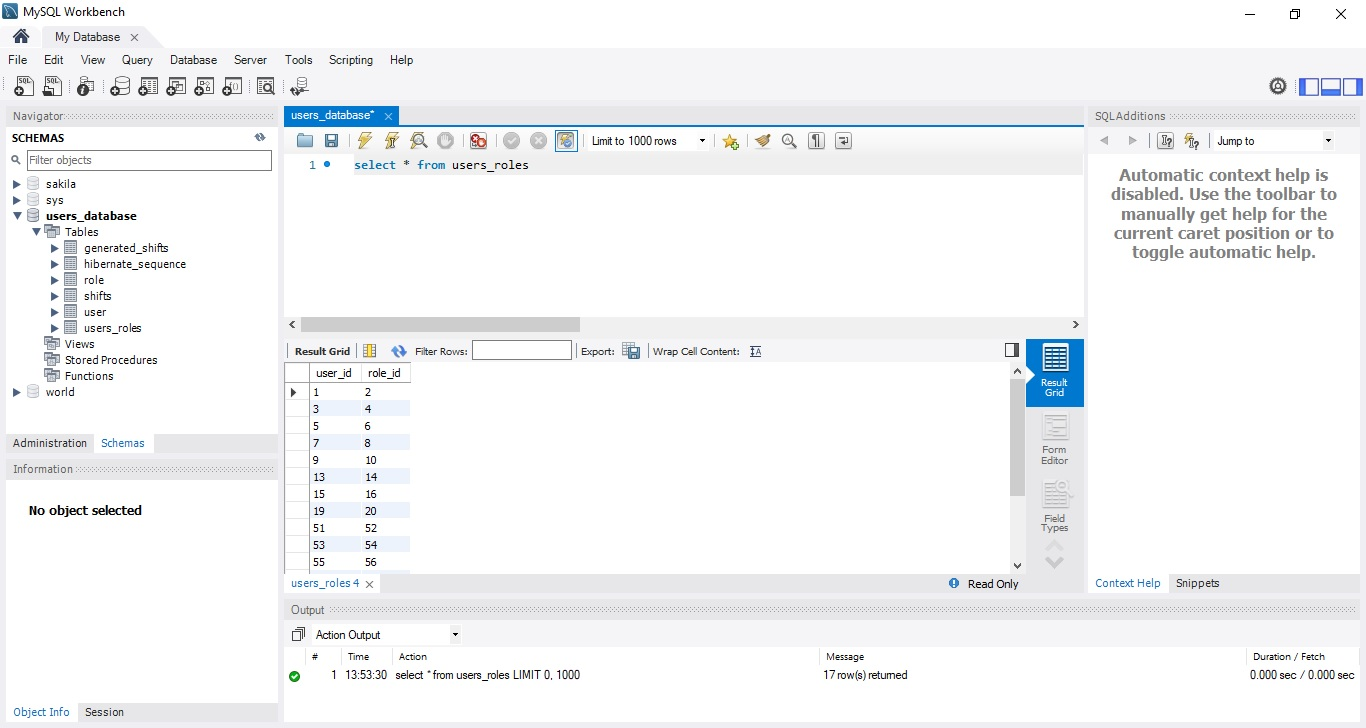
**role** – הטבלה role היא טבלה המכילה נתונים על התפקיד של המשתמש במערכת, אם הוא משתמש רגיל או מנהל המערכת. בטבלה זו, מאוחסנים הערכים הבאים: המפתח הראשי **id** ו-**name** (תפקיד המשתמש במערכת). להלן שאילתה ב-MySQL המחזירה את כל הנתונים של טבלה זו:



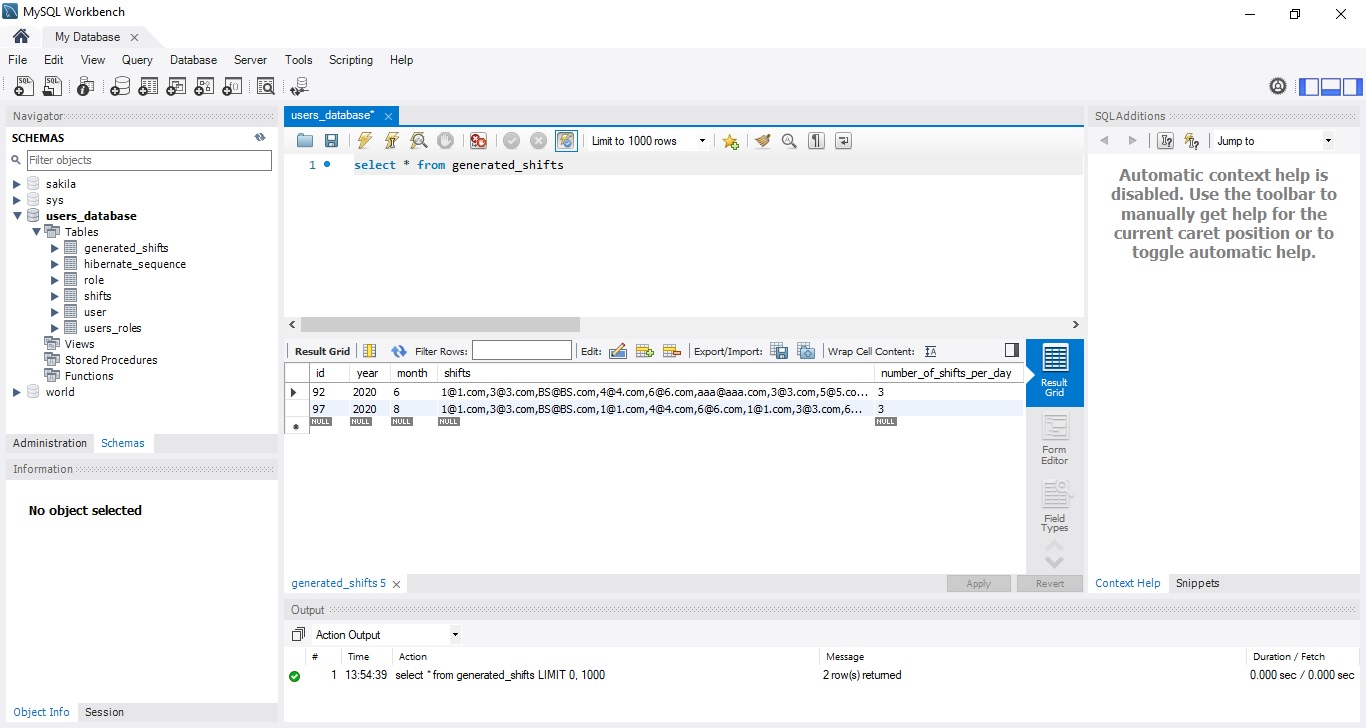
**shifts** – הטבלה shifts היא טבלה המכילה נתונים על משתמשים הרשומים למשמרות מסוימות במערכת. בטבלה זו, מאוחסנים הערכים הבאים: המפתח הראשי **id**, **email** (כתובת המייל של משתמש הרשום למשמרת), **year** (השנה לה הוא נרשם למשמרת), **month** (החודש לו הוא נרשם למשמרת) ו-**shifts** (המשמרות שהמשתמש נרשם אליהן, אשר מקוטלגות בין 0 ל-1). להלן שאילתה ב-MySQL המחזירה את כל הנתונים של טבלה זו:



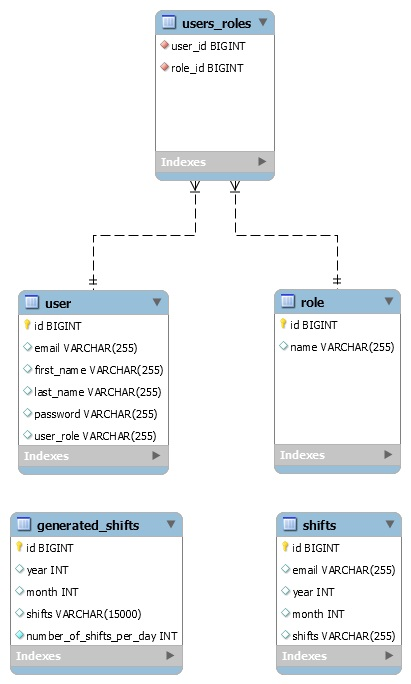
**users\_roles** – הטבלה users\_roles היא טבלה המחזיקה את המפתחות הראשיים של הטבלאות user ו-role. בטבלה זו, מאוחסנים הערכים הבאים: **user\_id** (מס' הסידורי של משתמש מסוים במערכת) ו-**role\_id**. להלן שאילתה ב-MySQL המחזירה את כל הנתונים של טבלה זו:



**generated\_shifts** – הטבלה generated\_shifts היא טבלה המכילה נתונים על המשמרות שנוצרו במערכת. בטבלה זו, מאוחסנים הערכים הבאים: המפתח הראשי **id**, **year** (השנה של המשמרות שנוצרו), **month** (החודש של המשמרות שנוצרו), **shifts** (כתובות המייל של כל המשתמשים הרשומים למשמרת בחודש הנ"ל) ו-**number\_of\_shifts\_per\_day** (מספר המשמרות בכל יום). להלן שאילתה ב-MySQL המחזירה את כל הנתונים של טבלה זו:



להלן דיאגרמת הטבלאות של המערכת שלנו במסד הנתונים:



חשוב לציין, בכל הטבלאות יש עמוד של **id**. עמודה זו נצרכת לעבודה עם Hibernate. אמנם היא לא חובה, אך היא מאפשרת גמישות בכתיבת הקוד אם משתמשים בה. בעזרת העמודה id, ניתן לשמור אובייקטים שמכילים אובייקטים ממחלקות אחרות במסד הנתונים, על ידי קישור לid המתאים (כפי שניתן לראות בטבלה users\_roles שמכילה אובייקטים מ-user ומ-role). עמודת ה-id מוזנת באופן אוטומטי על ידי Hibernate, על ידי שימוש באנוטציה: "@GeneratedValue", כפי שניתן לראות בתמונה להלן:



1. Advanced Shifts Management. [↑](#footnote-ref-1)
2. סגנון של עיצוב תוכנה שבו קוד הניתן לשימוש חוזר שולט בביצוע של קוד המיועד לפתרון של בעיות ספציפיות. המשמעות היא שהקוד שניתן לשימוש חוזר והקוד המיועד לפתרון של בעיה ספציפית מפותחים בנפרד, ולבסוף מאוחדים לקבלת יישום אחד כולל, המשלב את חלקי הקוד השונים (מתוך ויקיפדיה). [↑](#footnote-ref-2)
3. ויקיפדיה. [↑](#footnote-ref-3)
4. אנוטציות של Java היא סוג של מנגנון תחבירי להוספת metadata לקוד המקור (האנוטציות מסומנות ב"@").. ניתן להוסיף אנוטציות למחלקות, מתודות, משתנים, פרמטרים ו-packages. לדוגמה, אחת האנוטציות המוכרות של Java היא "Override@". אנוטציה זו בודקת שהמתודה המסומנת על ידה דורסת מתודה אחרת. אם המתודה אינה דורסת מתודה אחרת – היא גורמת לשגיאת קומפילציה. [↑](#footnote-ref-4)