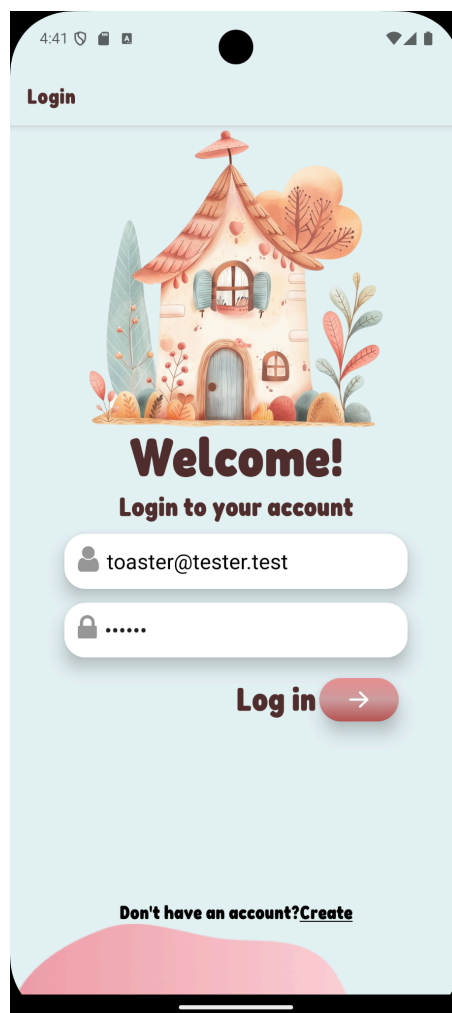


מסמך ניתוח FamilyNest

רשימת מסכים: 

מסכי התחברות והרשמה:

- מסך התחברות **LoginScreen**:



- מסך הרשמת משתמש חדש **SignUpScreen**:
 ○ שליחת הזמנה לא פונקציונלי כרגע

4:47

← **SignUp**

Create your FamilyNest account!

Account Information:

Family Name User name


Email-address

Create Password

Enter password

Re-Enter password

Create your parent profile:



Gender First name

Birthdate Passkey

2-1-2025

Send partner invitation: ○

Sign Up


4:50

← **SignUp**

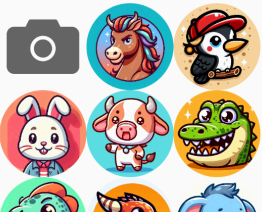
Create your FamilyNest account!

Account Information:

Test test123



Select Avatar



Confirm

Send partner invitation: ✓

partner@gmail.com

Sign Up

4:51

← **SignUp**

Create your FamilyNest account!

Account Information:

Test test123


test@gmail.com

Passwords not matching

.....

.....

Create your parent profile:



Male Test

Birthdate

2-1-1975

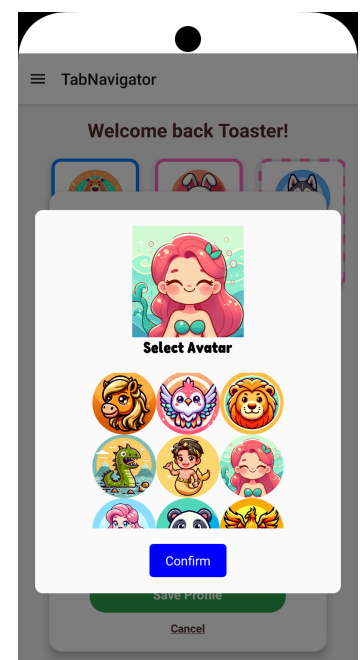
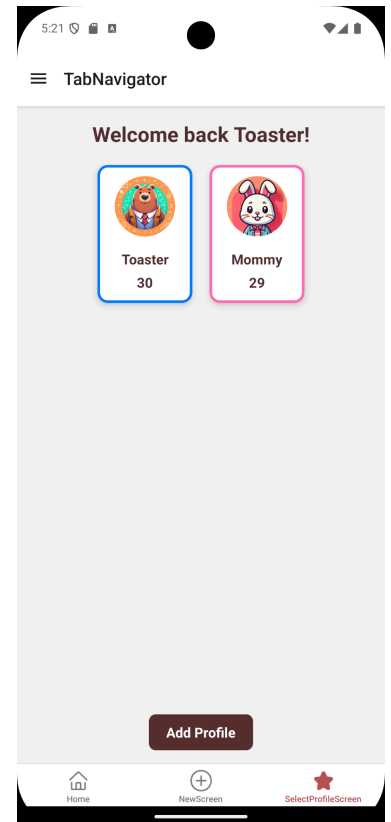
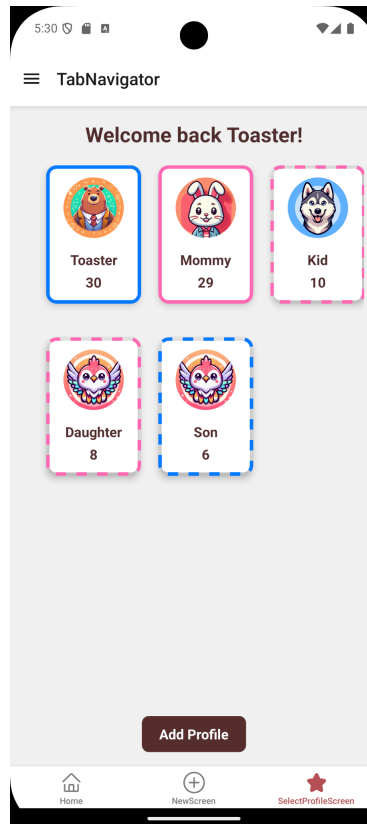
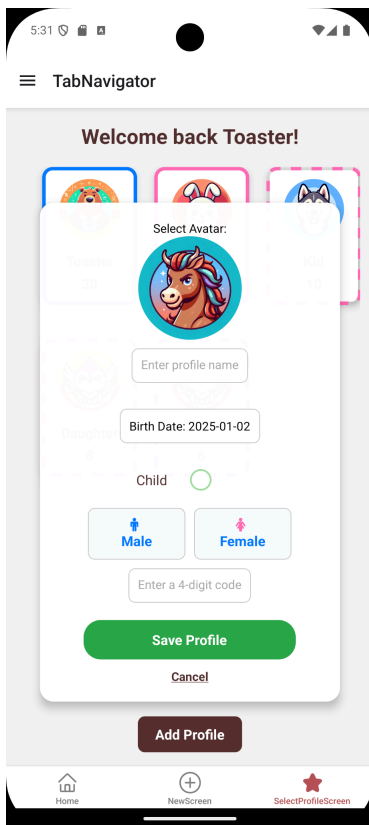
Send partner invitation: ✓

partner@gmail.com

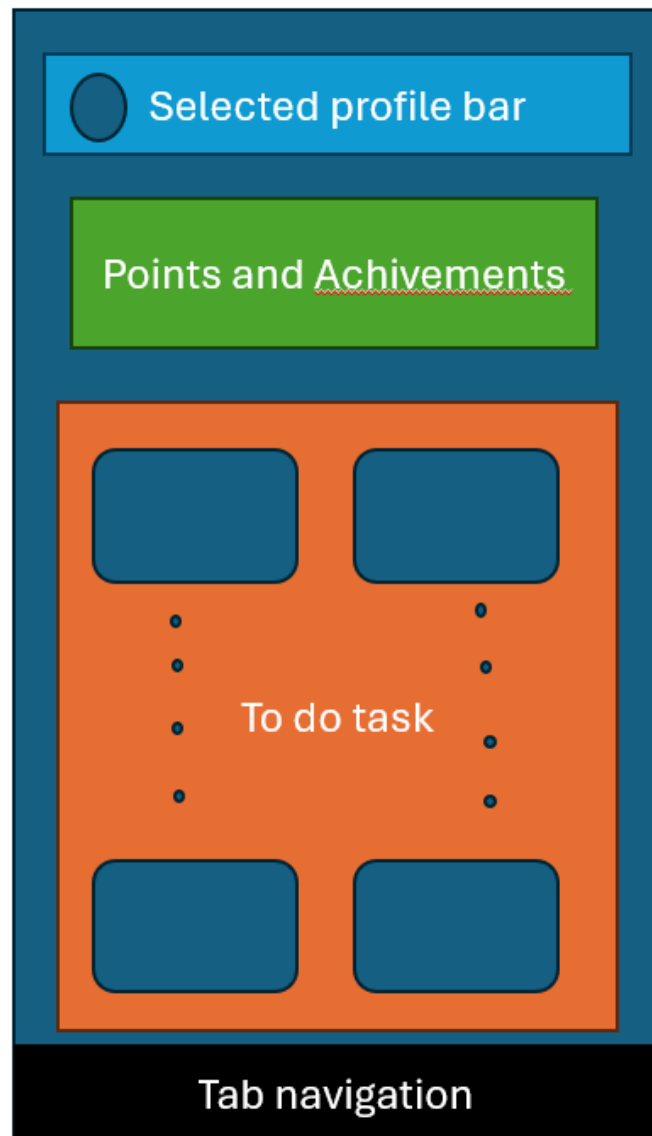
Sign Up

מסכי עבודה:

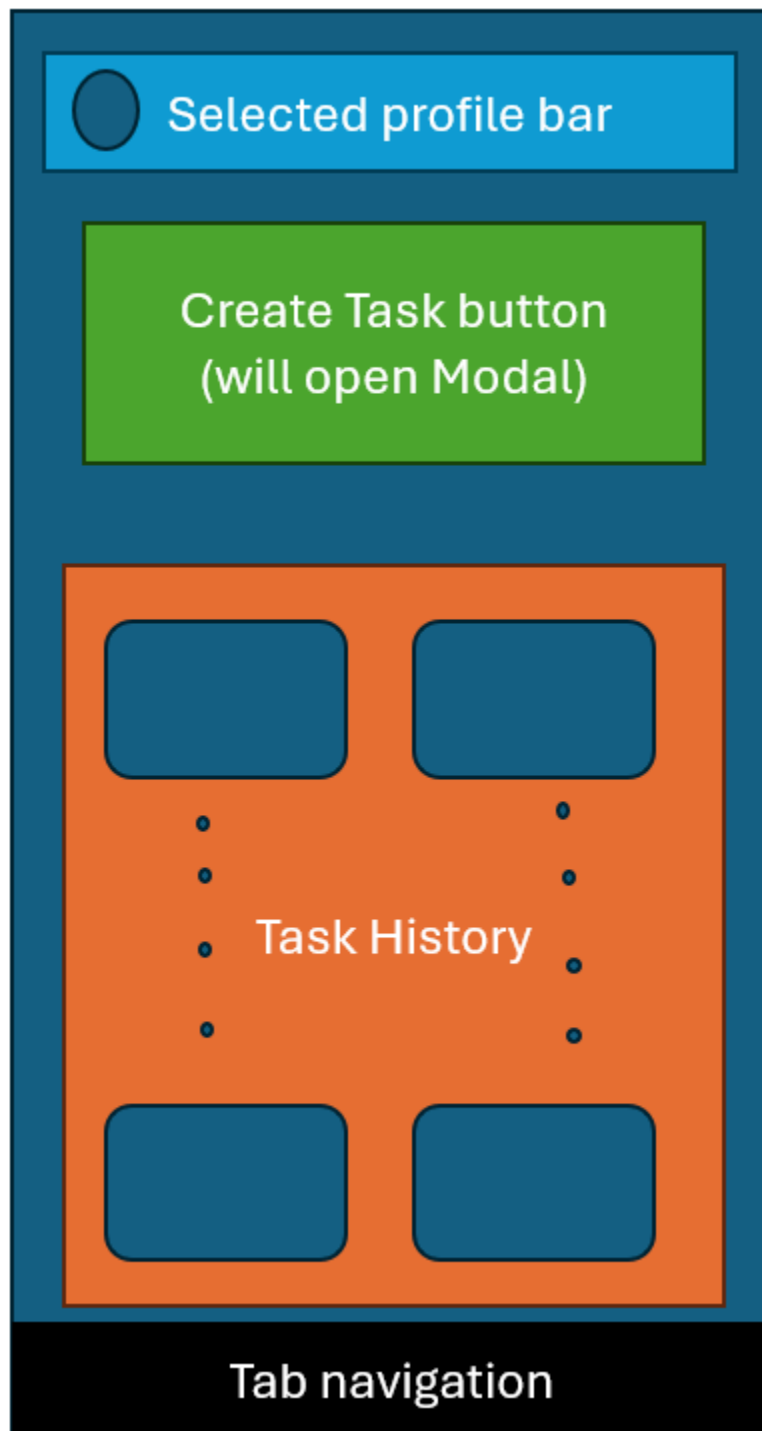
- מסך בחירת פרופיל או יצירת פרופיל **SelectProfileScreen**:
 - נחיתה במסך הזה כאשר יש צורך לבחור פרופיל או שהמשתמש חדש ואין פרופילים קיימים.
 - כחולים ורודים לפי המין. קווים מקווקוים מסמנים ילד.



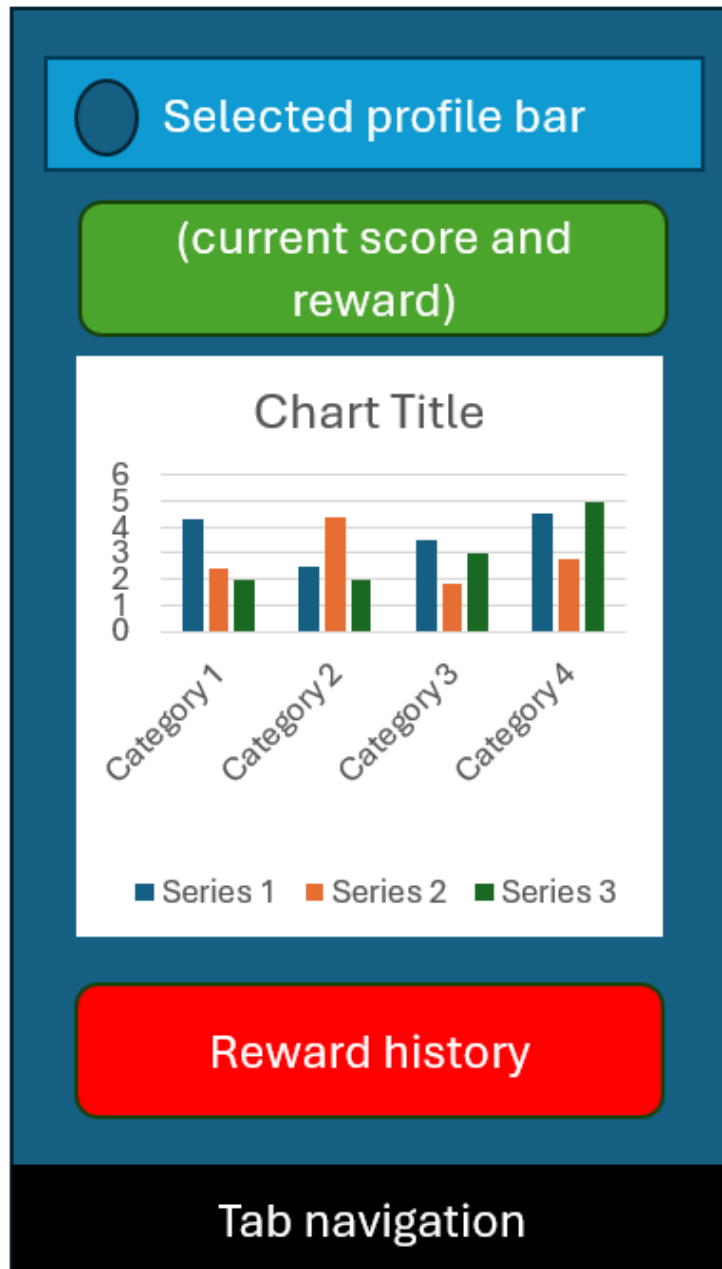
- מסך ראשי **Home**: (בשלבי פיתוח ראשונים)
 - יכלול הצגה של תמונת המצב עבור ילדים או הורים בהתאמה.



- מסך יצירת מטלות **TasksScreen**:



- מסך ניקוד ופרסים **RewardScreen**:



זאת לא רשימה ועיצוב סופי של המסכים ויכולה להשתנות.

מעברים בין מסכים:

Login->Home/SelectProfileScreen: מעבר ממסך למסך יצירת ובחירת פרופיל או למסך הבית בהתאם למצב.

SignupScreen->SelectProfileScreen: מעבר ממסך הרשמה למסך יצירה ובחירת פרופילים.

מעבר בין מסכי העבודה יתבצע בעזרת **TabNavigation**.

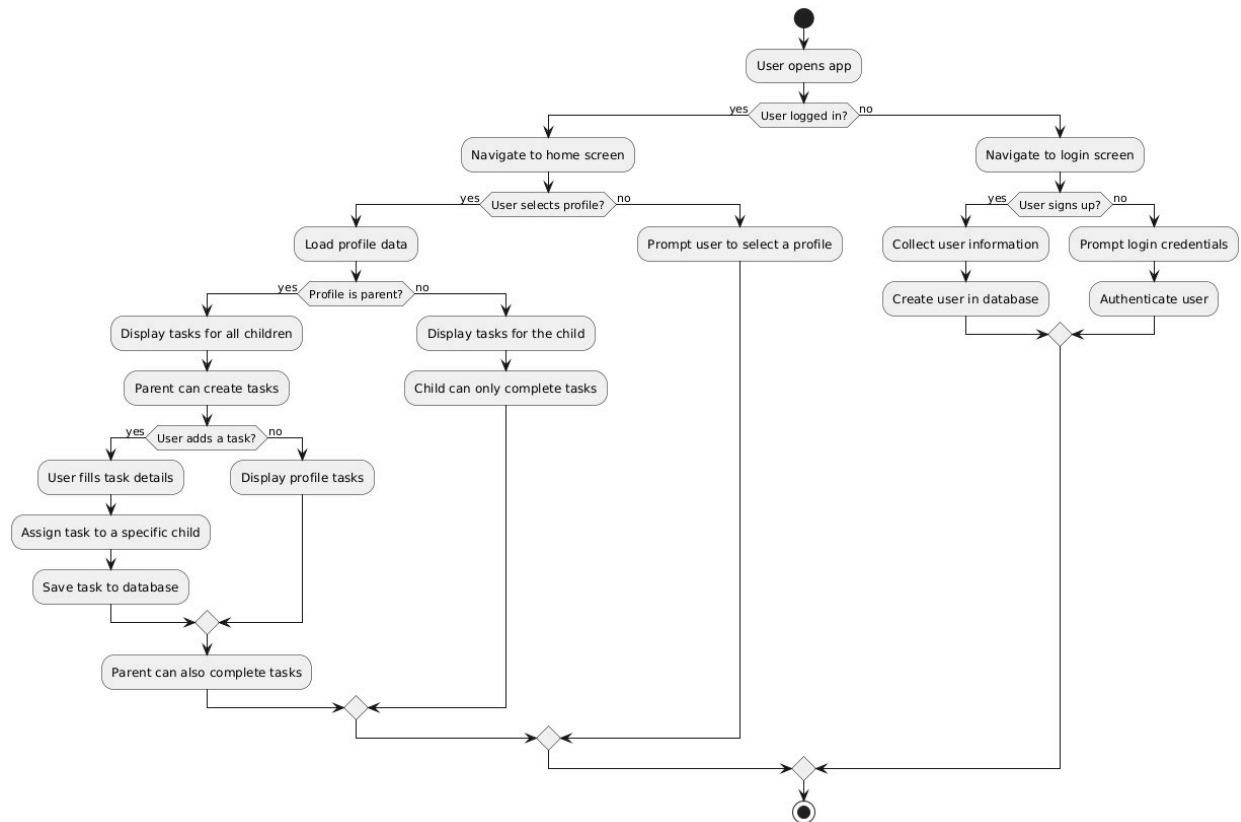
תבנית העיצוב MVC:

נשתמש ב-MVC מכיוון שהיא פשוטה וקל להטמיע אותה בפרויקטים שמבוססים על **React Native**, ובמיוחד כאשר מדובר במערכת שבה יש הפרדה ברורה בין הלוגיקה העסקית (Model), ממשק המשתמש (View), ושלב התיווך (Controller) שמנהל את האינטראקציה בין השניים. כל רכיב במערכת מופרד באופן ברור, כך שכל אחד מהם יתפקד באופן עצמאי. ה-**Model** יטפל בניהול הנתונים, ה-**View** יציג אותם, וה-**Controller** ינהל את העברת המידע בין השניים.

יתרונות ארכיטקטורת MVC:

1. **פשטות**: קל להבנה וליישום, במיוחד לפרויקטים קטנים ובינוניים.
 2. **תחזוקה קלה**: הפרדה ברורה בין המודולים מקלה על תחזוקה ושדרוגים עתידיים.
 3. **גמישות**: מאפשר שינוי בלוגיקה או בממשק ללא השפעה על רכיבים אחרים.
- הבחירה ב-MVC תאפשר לאפליקציה להיות מודולרית, קלה לתחזוקה ותסייע בשדרוגים עתידיים.

Activity Diagram:



הסבר לדיאגרמה:

מבנה הדיאגרמה:

הדיאגרמה מתארת את זרימת העבודה באפליקציה מרגע פתיחתה ועד פעולות המשתמש בממשק, תוך התייחסות לתרחישים שונים כמו התחברות, יצירת פרופיל, ניהול משימות והשלמתן.

פירוט השלבים:

1. פתיחת האפליקציה:

- המשתמש פותח את האפליקציה.
- אם המשתמש מחובר (Logged in), האפליקציה מעבירה אותו למסך הבית.
- במידה שלא, המשתמש מועבר למסך התחברות.

2. תהליך התחברות:

- המשתמש יכול לבחור להתחבר או להירשם:

- **רישום משתמש חדש:** המשתמש מזין פרטים, ואלו נשמרים במסד הנתונים.
- **התחברות קיימת:** המשתמש מזין פרטי התחברות, והאפליקציה מאמתת את הנתונים.

3. **בחירת פרופיל:**

- לאחר ההתחברות, המשתמש מתבקש לבחור פרופיל.
- אם לא נבחר פרופיל, תופיע הודעה המנחה אותו לבחור פרופיל.

4. **פעולות לפי סוג הפרופיל:**

- **הורה:**
 - הורה יכול להציג את כל המשימות של ילדיו.
 - להורה יש אפשרות ליצור משימות חדשות, למלא את פרטי המשימה ולשייך אותה לילד מסוים.
 - המשימות נשמרות במסד הנתונים.
 - בנוסף, הורה יכול להשלים משימות.
- **ילד:**
 - ילד רואה את המשימות שהוקצו לו בלבד.
 - לילד יש אפשרות לסמן משימות כהושלמו.

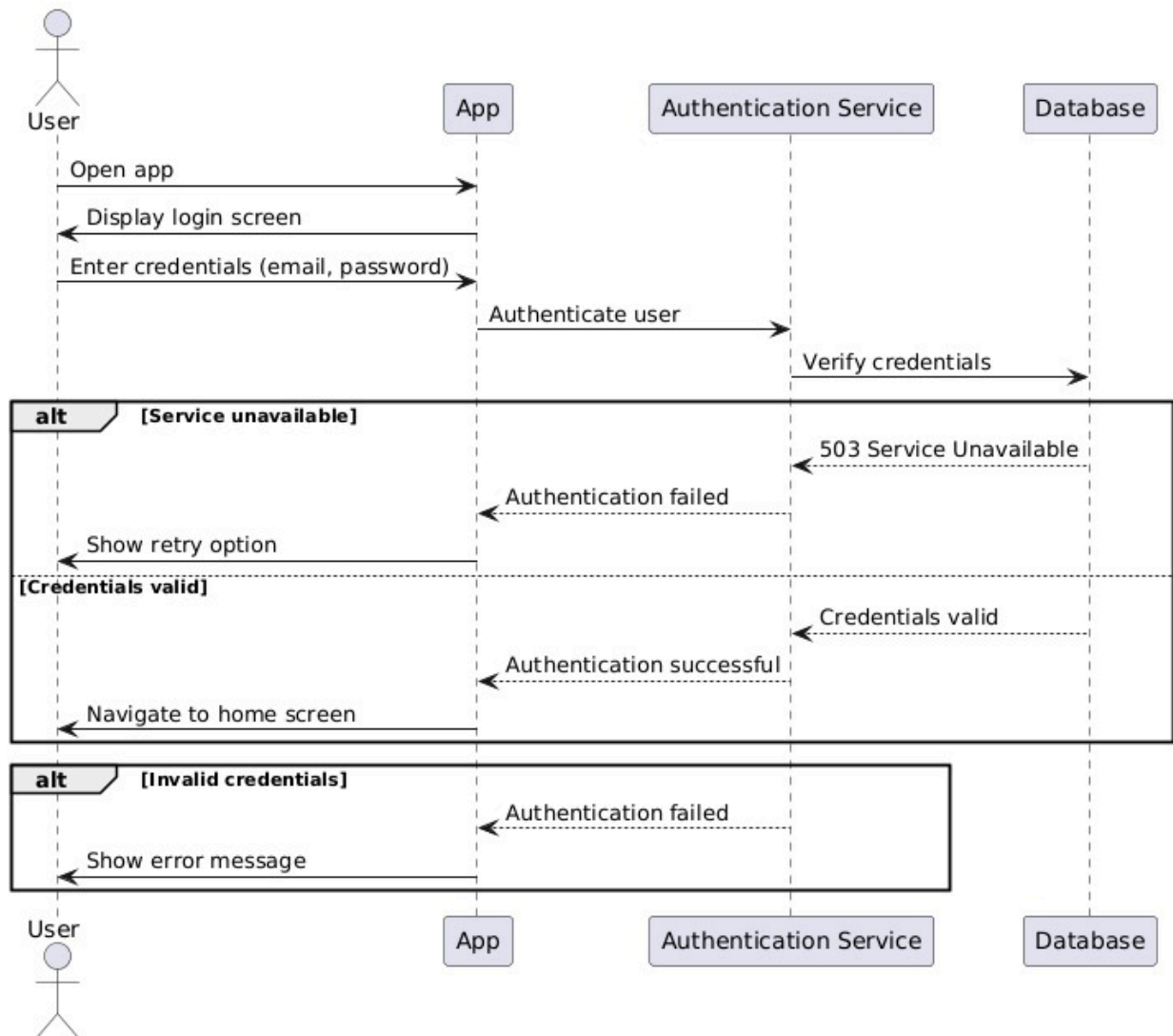
5. **אינטראקציות במסך הבית:**

- כל פרופיל (הורה או ילד) רואה את המידע הרלוונטי בהתאם לסוגו.
- במידה שהמשתמש בוחר ליצור משימה, תהליך זה מנוהל במסכי המשימה.

מהות הדיאגרמה:

הדיאגרמה מציגה את האינטראקציות המרכזיות של המשתמש עם האפליקציה, החל מתהליך ההתחברות, דרך ניהול הפרופילים ועד לניהול משימות. היא ממחישה כיצד מתבצעים התהליכים בהתאם לסוג המשתמש (הורה או ילד) ומספקת מבט על התהליכים המרכזיים שמנוהלים באפליקציה.

:Sequence Diagram



הסבר לדיאגרמה:

מבנה הדיאגרמה:

הדיאגרמה מתארת את זרימת תהליך ההתחברות באפליקציה, משלב פתיחת האפליקציה ועד התחברות מוצלחת או הצגת הודעת שגיאה למשתמש.

פירוט השלבים:

1. פתיחת האפליקציה:

○ המשתמש פותח את האפליקציה, ומוצג לו מסך ההתחברות.

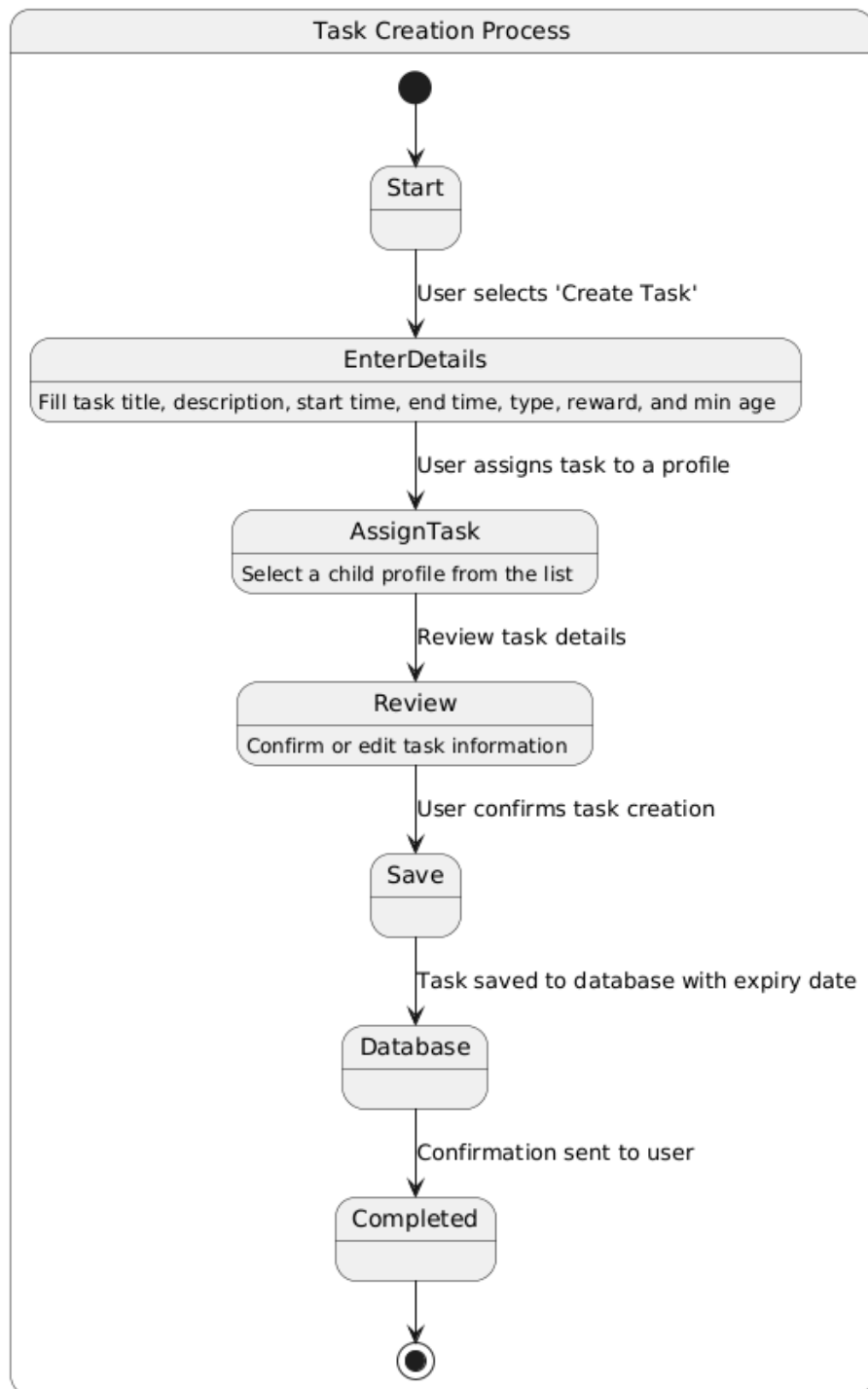
2. הזנת פרטי התחברות:

- המשתמש מזין את כתובת הדוא"ל והסיסמה שלו באפליקציה.
- 3. **אימות פרטי המשתמש:**
 - האפליקציה שולחת את פרטי ההתחברות לשירות האימות (Authentication Service).
 - שירות האימות מבצע בדיקה מול מסד הנתונים.
- 4. **תרחיש 1: שירות לא זמין:**
 - אם מסד הנתונים מחזיר שגיאה מסוג **Service Unavailable 503**:
 - שירות האימות מדווח על כישלון לאפליקציה.
 - האפליקציה מציגה למשתמש אפשרות לנסות שוב.
- 5. **תרחיש 2: פרטים נכונים:**
 - אם מסד הנתונים מאשר את תקינות הפרטים:
 - שירות האימות שולח הודעת הצלחה לאפליקציה.
 - האפליקציה מעבירה את המשתמש למסך הבית.
- 6. **תרחיש 3: פרטים שגויים:**
 - אם הפרטים אינם תואמים את הרשומות במסד הנתונים:
 - שירות האימות שולח הודעת כישלון לאפליקציה.
 - האפליקציה מציגה הודעת שגיאה למשתמש.

מהות הדיאגרמה:

הדיאגרמה ממחישה את תהליך ההתחברות, תוך התמקדות באינטראקציה בין המשתמש, האפליקציה, שירות האימות ומסד הנתונים. היא מדגישה את התרחישים המרכזיים שיכולים להתרחש (כמו שירות לא זמין או פרטי התחברות שגויים) ואת התגובות של המערכת בהתאם.

:State Machine Diagram



הסבר לדיאגרמה:

מבנה הדיאגרמה:

הדיאגרמה מתארת את תהליך יצירת המשימה באפליקציה, מרגע בחירת המשתמש באפשרות "צור משימה" ועד לשמירת המשימה במאגר הנתונים ואישורה.

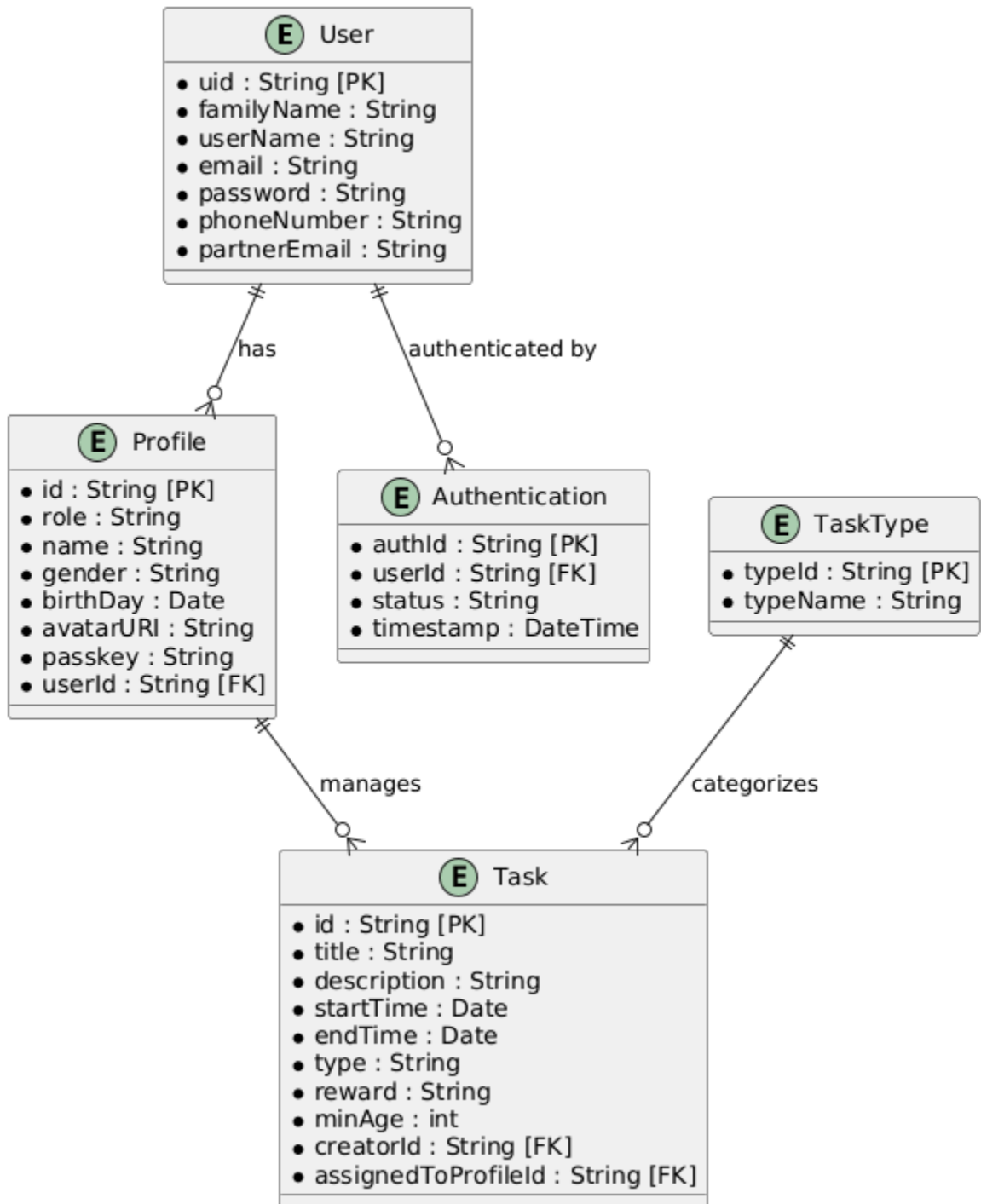
פירוט השלבים:

1. התחלת תהליך:
 - התהליך מתחיל כאשר המשתמש בוחר באפשרות "צור משימה" באפליקציה.
2. הזנת פרטי המשימה:
 - המשתמש ממלא את פרטי המשימה, כולל:
 - כותרת.
 - תיאור.
 - זמן התחלה וסיום.
 - סוג המשימה.
 - תגמול.
 - גיל מינימלי לביצוע המשימה.
3. שיבוץ משימה לפרופיל:
 - לאחר מילוי הפרטים, המשתמש משייך את המשימה לפרופיל מסוים (למשל, ילד ספציפי) מתוך רשימת הפרופילים הזמינים.
4. סקירת פרטי המשימה:
 - המשתמש סוקר את פרטי המשימה שיצר.
 - יש אפשרות לערוך את המידע במקרה הצורך.
5. שמירת המשימה:
 - המשתמש מאשר את יצירת המשימה.
 - המשימה נשמרת במאגר הנתונים עם תאריך תפוגה (לדוגמה, משימה שפגה לאחר שבוע).
6. אישור וסיום:
 - המערכת שולחת אישור על יצירת המשימה למשתמש.
 - התהליך מסתיים.

מהות הדיאגרמה:

הדיאגרמה מציגה את כל שלבי יצירת המשימה באפליקציה, תוך שימת דגש על אינטראקציות המשתמש עם המערכת ועל שלבי הבקרה לפני שמירת המשימה. היא ממחישה את הזרימה ההגיונית שמבטיחה שהמשימה מוגדרת, משויכת, ומאושרת כהלכה.

:(ERD (Entity-Relationship Diagram



הסבר לדיאגרמה:

מבנה הדיאגרמה:

הדיאגרמה מתארת את הקשרים בין ישויות מרכזיות באפליקציה באמצעות דיאגרמת ERD (Entity-Relationship Diagram). היא מציגה את המבנה הנתוני ואת היחסים בין משתמשים, פרופילים, משימות ותהליכי אימות.

פירוט הישויות והקשרים:

1. ישות User (משתמש):

- מזהה ייחודי (**uid**) לכל משתמש.
- פרטים אישיים כמו שם משפחה, שם משתמש, דוא"ל, סיסמה, מספר טלפון ודוא"ל בן/בת זוג.
- משתמש יכול להיות הורה או ילד.
- לכל משתמש יכולים להיות מספר פרופילים.

2. ישות Profile (פרופיל):

- מזהה ייחודי (**id**) לכל פרופיל.
- תכונות כוללות:
 - תפקיד (**role**) - הורה או ילד.
 - שם, מין, תאריך לידה, תמונת פרופיל (**avatarURI**), וקוד אישי (**passkey**).
- משויך למשתמש יחיד (**userId**).

3. ישות Task (משימה):

- מזהה ייחודי (**id**) לכל משימה.
- פרטי המשימה כוללים:
 - כותרת, תיאור, זמן התחלה וסיום, סוג משימה, תגמול, וגיל מינימלי לביצוע.
- כל משימה נוצרה על ידי משתמש מסוים (**creatorId**) ומשויכת לפרופיל (**assignedToProfileId**).

4. ישות Authentication (אימות):

- מזהה ייחודי (**authId**) לכל תהליך אימות.
- פרטים כוללים:
 - מזהה משתמש (**userId**), סטטוס האימות (**status**), ותאריך ושעה.

5. ישות TaskType (סוגי משימות):

- מזהה ייחודי (**typeId**) ותיאור סוג המשימה (**typeName**).
- משימות מקוטלגות לפי סוג.

הקשרים בין הישויות:

1. **משתמשים ופרופילים:**
 - כל משתמש יכול להחזיק מספר פרופילים.
2. **פרופילים ומשימות:**
 - כל פרופיל מנהל מספר משימות.
3. **משתמשים ותהליכי אימות:**
 - כל משתמש עובר תהליך אימות.
4. **משימות וסוגי משימות:**
 - כל משימה מסווגת תחת סוג מסוים.

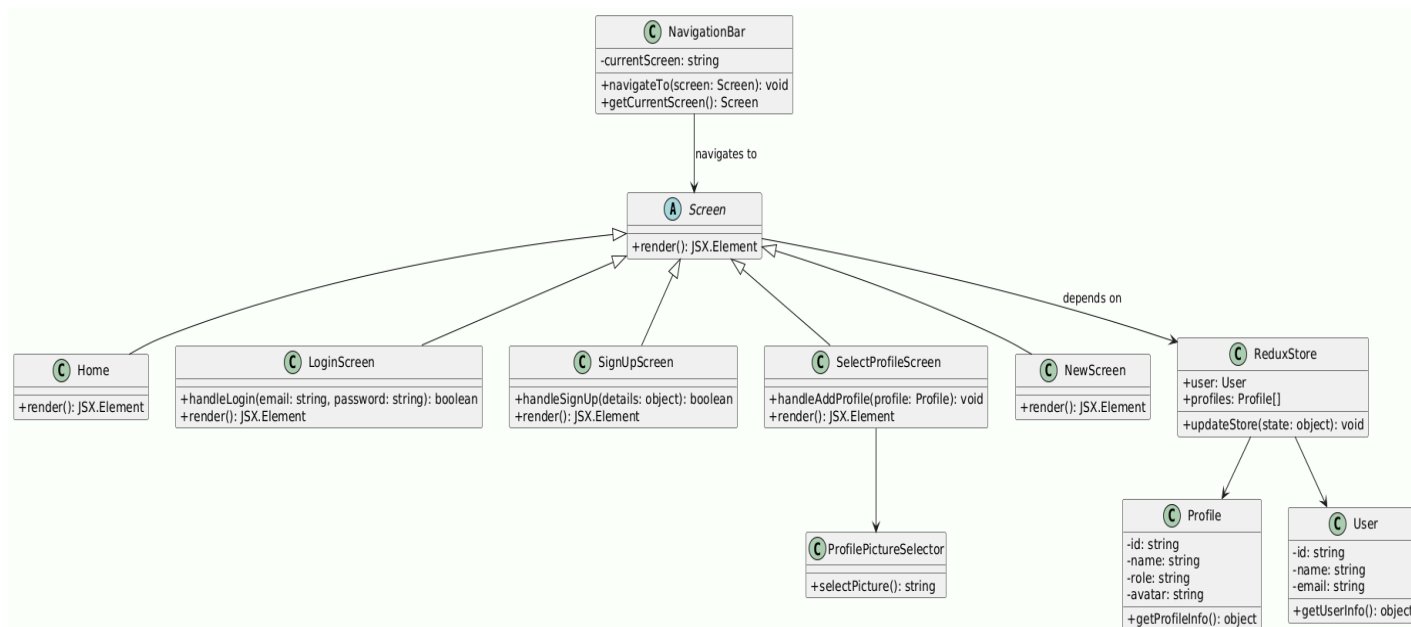
נורמליזציה:

הדיאגרמה עומדת בעקרונות נורמליזציה ב-3NF מכיוון שכל עמודה תלויה ישירות במפתח הראשי, אין תלות בין עמודות שאינן מפתחות, והמידע מופרד לישויות נפרדות למניעת כפילויות. לדוגמה, **TaskType** הופרדה מ-**Task** לניהול מרכזי של סוגי משימות, ו-**Authentication** נפרדת מ-**User** לניהול עצמאי של אימותים. הקשרים בין הישויות מוגדרים באמצעות מפתחות זרים, מה שמבטיח עקביות ויעילות במבנה הנתונים.

מהות הדיאגרמה:

הדיאגרמה מציגה את המבנה הנתוני של האפליקציה בצורה נורמלית (3NF), תוך הדגשת הקשרים בין הישויות. היא משמשת ככלי עזר להבנת הארגון והקשרים בין המשתמשים, הפרופילים, המשימות ותהליכי האימות, ומבטיחה עיצוב מבנה נתונים יעיל ומדויק.

: Class Diagram



הסבר :

(סרגל ניווט):

- תפקיד: משמש כמרכז הניווט באפליקציה.

- מאפיינים:

- `currentScreen`: משתנה ששומר את המסך הפעיל כרגע.

- מתודות:

- `navigateTo(screen: Screen)`:

- [משנה את המסך הפעיל למסך המבוקש.

- `getCurrentScreen()`:

מחזירה את המסך הפעיל הנוכחי.

`Screen` (מחלקת אבסטרקטית):

- תפקיד: מהווה מחלקת בסיס לכל המסכים באפליקציה.

- מתודות:

- `render()`: מתודה שכל מחלקות הבת מממשות כדי להציג את תוכן המסך.

- Home (מסך הבית):
 - ירושה: יורש מ-Screen.
 - תפקיד: מייצג את מסך הבית של האפליקציה.
 - מתודות:
 - render(): מציג את התוכן של מסך הבית.
- LoginScreen (מסך התחברות):
 - תפקיד: מאפשר למשתמשים להתחבר.
 - מתודות:
 - handleLogin(email: string, password: string): מבצע בדיקת פרטי התחברות.
 - render(): מציג את מסך ההתחברות.
- SignUpScreen (מסך הרשמה):
 - תפקיד: מאפשר למשתמשים להירשם.
 - מתודות:
 - handleSignUp(details: object): מטפל בלוגיקת ההרשמה.
 - render(): מציג את מסך ההרשמה.
- SelectProfileScreen (מסך בחירת פרופיל):
 - תפקיד: משמש לניהול והוספת פרופילים.
 - מתודות:
 - handleAddProfile(profile: Profile): מוסיף פרופיל חדש.
 - render(): מציג את מסך בחירת הפרופיל.
- NewScreen (מסך נוסף):
 - תפקיד: מייצג מסך נוסף או placeholder.
 - מתודות:
 - render(): מציג את המסך.
- ProfilePictureSelector (בחירת תמונת פרופיל):
 - תפקיד: רכיב שמאפשר לבחור תמונת פרופיל.
 - מתודות:
 - selectPicture(): מחזירה את התמונה שנבחרה.
- ReduxStore (ניהול המצב הגלובלי):
 - תפקיד: מנהל את כל הנתונים הגלובליים באפליקציה.
 - מאפיינים:
 - user: שומר את פרטי המשתמש הנוכחי.
 - profiles: רשימת הפרופילים הקיימים.
 - מתודות:
 - updateStore(state: object): מעדכנת את המצב הגלובלי.
- Profile (פרופיל):
 - תפקיד: מייצג פרופיל של משתמש.
 - מאפיינים:

- id, name, role, avatar: פרטי הפרופיל.
- מתודות:
- getProfileInfo(): מחזירה את פרטי הפרופיל.
- User (משתמש):
- תפקיד: מייצג את המשתמש במערכת.
- מאפיינים:
- id, name, email: פרטי המשתמש.
- מתודות:
- getUserInfo(): מחזירה את פרטי המשתמש.

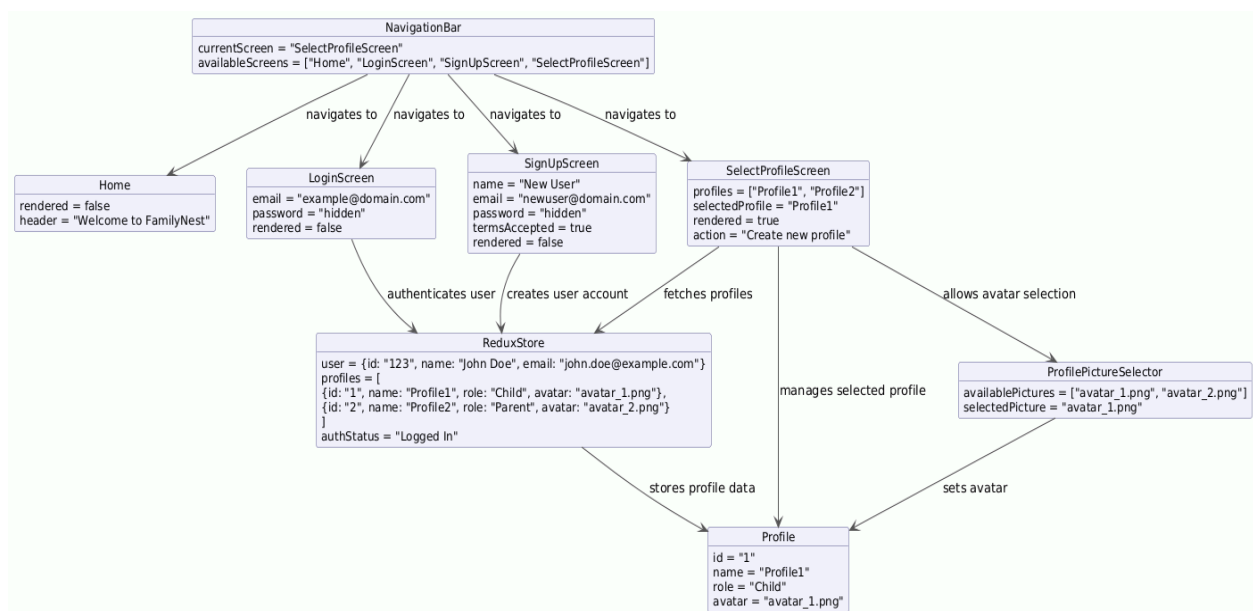
קשרים בין מחלקות:

1. Screen → NavigationBar: סרגל הניווט אחראי על מעבר בין מסכים.
2. Screen → מסכים ספציפיים: כל המסכים (כגון Home, LoginScreen) יורשים מ-Screen.
3. ProfilePictureSelector → SelectProfileScreen: מסך בחירת פרופיל משתמש ברכיב בחירת תמונה.
4. ReduxStore → User/Profile: ReduxStore מנהל את פרטי המשתמשים והפרופילים.

מהות הדיאגרמה:

הדיאגרמה מראה מבנה אפליקציה מודולרי, שבו כל מחלקה ממלאת תפקיד ברור ומוגדר. היא מדגישה את הקשרים בין הרכיבים בצורה שמסבירה כיצד האפליקציה פועלת מבחינה לוגית.

Object Diagram :



הסבר :

:NavBar

- אחראי על מעבר בין המסכים באפליקציה.
- משתנה **currentScreen** מציין את המסך הנוכחי.

- משתנה `availableScreens` מכיל את רשימת המסכים הזמינים.

2. `:Home`

- מסך הבית של האפליקציה.
- המאפיינים `rendered` ו-`header` מייצגים האם המסך מוצג ומה הכותרת שלו.

3. `:LoginScreen`

- מסך ההתחברות.
- מכיל נתוני התחברות (אימייל וסיסמה).
- מאמת את המשתמש מול ה-`ReduxStore`.

4. `:SignUpScreen`

- מסך ההרשמה.
- מאפיינים ליצירת משתמש חדש (`name, email, password`).
- יוצר חשבון משתמש חדש ב-`ReduxStore`.

5. `:SelectProfileScreen`

- מסך בחירת הפרופיל.
- מאפיינים `profiles` ו-`selectedProfile` מנהלים את רשימת הפרופילים והפרופיל שנבחר.

6. `:ReduxStore`

- מאחסן נתונים גלובליים:
 - פרטי משתמש.
 - רשימת פרופילים.
 - סטטוס אימות.

7. `:Profile`

- ייצוג של פרופיל יחיד.
- מאפיינים `role, name, id`, ו-`avatar`.

8. `:ProfilePictureSelector`

- מנהל את בחירת תמונות הפרופיל.
- מאפיינים `availablePictures` ו-`selectedPicture` מגדירים את התמונות הזמינות והנבחרת.

הקשרים המרכזיים:

- **NavigationBar**: מנווט בין כל המסכים.
- **ReduxStore**: משמש כמאגר נתונים גלובלי.
- **SelectProfileScreen → ProfilePictureSelector**: מנהל בחירת תמונה.
- **LoginScreen ו-SignUpScreen**: מאמתים ומשתמשים בנתונים מה-`ReduxStore`.

מהות הדיאגרמה:

הדיאגרמה ממחישה בצורה תמציתית את הזרימה והקשרים בין האובייקטים באפליקציה, תוך התמקדות בניהול נתונים, ניווט, ובחירת פרופילים.