**System:** myproject

**Reviewer:** Ruth Cohen

**Date:** 07.04.24

קוד רוויו הוא הזדמנות נהדרת ללמוד ולהתפתח כמתכנת. אני שמחה ללוות אותך בתהליך ולספק משוב בונה שיעזור לך לצמוח.

## **עקרונות כלליים לכתיבת קוד איכותי**

* קוד נקי וקריא הוא הבסיס לכל תהליך הפיתוח. הקפידי על שמות משתנים בהירים ותיאוריים, פרמטרים מובנים, פיסוק ומעבר שורה נכון, סידור לוגי של רכיבי הקוד ועוד. קוד שקל לקרוא הוא קוד שקל להבין, לתחזק ולשפר.
* תיעוד נרחב והסברים הם קריטיים. תעדי כל פונקציה, מה היא מקבלת ומה היא מחזירה, באילו מצבים היא פועלת ומהי ההתנהגות הצפויה. תיעוד ברור יעזור לך ולאחרים להבין את הקוד ולמנוע באגים.
* פצלי את הקוד לפונקציות קטנות, כל אחת עם מטרה מוגדרת היטב. פונקציות קצרות הן הרבה יותר קריאות, ניתנות לשימוש חוזר וקלות יותר לבדיקה ותחזוקה.
* שמרי על עקביות מוחלטת בסגנון הקוד. כך הקורא יכול להבין מהר יותר את הקוד באזורים שונים של האפליקציה. לדוגמה מיקום פתיחת מקשי מפתחות, רווחים, שמות ועוד.
* אמצי טכניקות מודרניות שמקלות על הקוד כמו List Comprehension, Unpacking Operators וכו'. זה יחסוך שורות קוד רבות.

שימי ❤  
מבנה הקוד רוויו מתבצע בפורמט הבא:

**Issue/Concern:** : הבעיה בנושא/חלק מסוים בקוד בו אני רואה שיטה מיושנת, קוד לא אופטימלי, ארכיטקטורה פשוטה ועוד

**Suggestion/Feedback**: הצעת פתרון וחלופה למצב הקוד הקיים **Example** : צילום מסך לדוגמא גנרית לפתרון האפשרי - מעל הדוגמא מצוין סעיף לאיזה חלק בקוד שלך הפתרון רלוונטי **Impact**: מה התועלת בפתרון והחלופה המוצעת

**Priority:** מה חשיבות תיקון במיזם שלך, מתחלק לשלוש רמות: min - חשיבות נמוכה mid- חשיבות בינונית, max - חשיבות גבוהה

**code review:**

.1

**Issue/Concern:**

שם הפרויקט אינו מתאר את מה שהפרויקט עושה ואין לו משמעות מיוחדת. מה זה myproject?

**Suggestion/Feedback:**

תני לפרוייקט שם שמתאר את המהות או המשמעות שלו או שם אחר.

**Example:**

אם בונים API שנועד לניהול תגובות של משתמשים אז נקרא לשם הפרויקט comments-api

אם בונים אפליקציה לניהול תזונה וספורט ורוצים לתת לה שם ייחודי ומשמעותי, אפשר לקרוא לה myFitness או myWellness

**Impact:**

שם שמתאר את הפרויקט בצורה נכונה מאפשר לאנשים מבחוץ להבין על מה הפרויקט מדבר, נותן לו משמעות מיוחדת ומבדל אותו משאר הפרויקטים.

**Priority**: max

2.

**Issue/Concern:**

**אין בראנץ' ייעודי לפיתוח**

וכן לא ברור מה ההבדל master לmain, אם מדובר בשני בראנצ'ים ראשיים תמחקי אחד מהם ותפתחי בראנץ' לפיתוח. הגדירי בראנץ' ראשי אחד.

**Suggestion/Feedback:**

**תפתחי בראנץ' חדש לפיתוח ותעבדי רק איתו כל עוד את מפתחת דברים חדשים, לאחר שהקוד שפיתחת מתאים לסביבת פרודקשן תפתחי פול רקווסט ותעבירי אותו לברנץ' הראשי.**

**Example:**

דוגמאות לשמות של בראנץ' פיתוח:

dev development,

**דוגמאות לשמות של בראנץ' ראשי לפרודקשן:**

main, prod, production

**Impact:**

**מונעים קוד שאינו נבדק מלהיכנס לסביבת פרודקשן, הוא קודם עובר בבראנץ' פיתוח ולאחר שאושר מעבירים לפרודקשן. וכן שמירה על קוד נקי ומסודר בפרודקשן.**

**Priority**: mid

3.

**Issue/Concern:**

**חסר קובץ requirements.txt עבור חבילות שהותקנו על ידי pip**

**כמו שעבור חבילות npm יש לך קובץ package.json כך עבור חבילות של פייתון יש ליצור את הקובץ הנ"ל, קובץ זה מכיל את שמות החבילות.**

**:Suggestion/Feedback**

**באופן כללי כדאי תמיד ליצור סביבה וירטואלית כאשר מפתחים בפיתון, הסביבה הווירטואלית מתוארת על ידי תיקיה שמתקינים בפרויקט, בד"כ קוראים לה venv, וזו תכיל את חבילות הפרויקט. בנוסף הקובץ המתעד את שמות החבילות יקרא requirements.txt**

**הפעילי את הפקודות הבאות: (מותאמות לbash של הטרמינל בvs code)**

1. **צרי סביבה וירטואלית:**

**python -m venv venv**

1. **הפעילי אותה:**

**source venv/Scripts/activate**

1. **כעת התקיני את החבילות שהותקנו באמצעות pip מחדש. הן אוטומטית יותקנו בתוך התיקיה venv**
2. **תעדי את שמות החבילות בקובץ requirements.txt:**

**pip freeze > requirements.txt**

**הערה לגבי פקודה 2: יכול להיות שקובץ הactivate שלך נמצא במיקום שונה בתיקית הvenv במידה והפקודה לא עבדה לך, בדקי את המיקום המדויק והפעילי אותה עבורו.**

source [PATH\_TO\_ACTIVATE\_]

**כעת מי שיוריד את הפרויקט יוכל להתקין את החבילות האלו בקלות עם הפקודה הבאה: pip install requirements.txt**

**(מומלץ להוסיף פקודה זאת לקובץ הREADME.md תחת הוראות התקנה)**

**Impact:**

**מקל על מי שרוצה להבין באיזה חבילות השתמשת או להוריד את הפרויקט ולהפעיל אותו עם החבילות.**

**Priority**: max

4.

**Issue/Concern:**

**חסר קובץ .ignore - ללא קובץ זה מועלים קבצים מיותרים לגיט ומעמיסים על הקוד, כפי שניתן לראות תיקיית החבילות עלתה לגיטהאב למרות שלא אמורים להעלות אותה.**

**Suggestion/Feedback:**

צרי קובץ חדש בשם .gitignoreבתיקיה הראשית של הפרויקט, הקובץ הזה יכיל את כל הדברים שלא צריך להעלות לגיטהאב. שזה כולל חבילות, קבצים רגישים המכילים סיסמאות, תיקיה המייצגת סביבה וירטואלית, קבצי logs וכו'.

**Example:**



תוכן קובץ סטנדרטי שכזה בפיתון יראה כך:

# Python compiled bytecode

\*.pyc

# Python bytecode cache directories

\_\_pycache\_\_/

# Virtual environment directory

venv/

# Environment variable files

.env

**Impact:**

חבילת הפרויקט קלה יותר, כי אם היא תכיל את כל החבילות יהיה מאוד קשה לנייד אותה ולהעלות אותה לענן, כמו כן, חיסכון בקוד מיותר בגיטהאב, ובכך מונע מאנשים שמורידים את הפרויקט להוריד דברים מיותרים או דברים שראוי שיתקינו בעצמם. וכן מונע חשיפה של קבצי סיסמאות בגיטהאב.

**Priority**: max

1. **Issue/Concern:**

**חסר קובץ** README.md

**Suggestion/Feedback:**

צרי קובץ בשם README.md, בקובץ זה תתארי את הפרויקט. הוסיפי כמה שיותר תיאור רלוונטי אך מתומצת (תיאור פרויקט, איך להתקין, איך להשתמש, רישיון וכו')

**Example:**

קובץREADME המתאר פרויקט סטנדרטי בפיתון:

**# Project Name**

**## Description**

Welcome to [Project Name]! This project aims to [briefly describe the purpose or functionality of the project]. This README provides essential information for understanding the project.

**## Installation**

To install [Project Name], follow these steps:

1. Clone the repository: `git clone https://github.com/yourusername/project-name.git`

2. Navigate to the project directory: `cd project-name`

3. [Optional] Set up a virtual environment: `python -m venv venv` (optional but recommended)

4. Activate the virtual environment:

   - On Windows: `venv\Scripts\activate`

   - On macOS and Linux: `source venv/bin/activate`

5. Install dependencies: `pip install -r requirements.txt`

6. [Any additional installation steps, such as configuring environment variables]

**## Usage**

To use [Project Name], follow these steps:

1. [Provide instructions on how to use the project. Include examples if applicable.]

2. [Include any additional usage instructions or command-line options]

**## Built With**

- [Python](https://www.python.org/) - The programming language used.

- [Library/Framework Name](Link to library/framework) - Description of the library/framework used.

**## License**

This project is licensed under the MIT License - see the [LICENSE](LICENSE) file for details.

**Impact:**

**מראה על פרויקט מקצועי, נותן אפשרות לאנשים מבחוץ להבין מה רעיון הפרויקט וכיצד להתקין אותו.**

**Priority: mid**

**Issue/Concern:**

הקובץ tests.py ריק

**Suggestion/Feedback:**

מומלץ להוסיף טסטים לבדיקת תקינות הקוד.

**Example:**

from django.test import TestCase

from .models import YourModel

from .forms import YourForm

class YourModelTestCase(TestCase):

    def test\_model\_creation(self):

        obj = YourModel.objects.create(name='Test Object')

        self.assertEqual(obj.name, 'Test Object')

class YourFormTestCase(TestCase):

    def test\_form\_valid\_data(self):

        form\_data = {'field1': 'value1', 'field2': 'value2'}

        form = YourForm(data=form\_data)

        self.assertTrue(form.is\_valid())

def test\_form\_invalid\_data(self):

        form\_data = {'field1': '', 'field2': 'value2'}

        form = YourForm(data=form\_data)

        self.assertFalse(form.is\_valid())

**הקלאס הראשון בודק אם מודל נוצר בצורה נכונה והקלאס השני בודק** form validation **עם פונקציה אחת באמצעות מידע תקין ועם פונקציה נוספת באמצעות מידע לא תקין.**

**Impact:**

טסטים יכולים לעזור לתפוס באגים, מבטיחים יציבות לקוד, שומרים על קוד איכותי, מודולרי וניתן להרחבה, מפשטים תהליכי CD/CI ועוד...

**Priority**: mid

**Issue/Concern:**

Import duplications and redundency

**Suggestion/Feedback:**

יש כפילות כאשר מייבאים את החבילות, כמו כן יש מודולים שלא נעשה בהם שימוש ולכן צריך למחוק אותם.

**Example:**

דוגמאות מתוך views.py:

השורות הבאות:

from django.contrib.auth import authenticate, login

from django.contrib.auth import get\_user\_model

from django.contrib.auth import logout

יכולות להצטמצם לשורה אחת:

from django.contrib.auth import authenticate, login, get\_user\_model, logout

כמו כן, יש למחוק מודלים שלא נעשה בהם שימוש, כמו:

from django.http import HttpResponseBadRequest, HttpResponseForbidden

from django.http import HttpResponse

from .models import ReminderOption

**Impact:**

חסכון בשורות קוד, אין כפילויות, הקוד קריא ונקי יותר.

**Priority**:mid



**Issue/Concern:**

בתיקית הtemplates אין הפרדה לקבצים מסודרים עבור קוד מסוג js וקוד css, הכל נמצא ביחד עם הHTML

**Suggestion/Feedback:**

יש להפריד את הקוד של js והקוד של css מקבצי הHTML ולכתוב אותם בקבצים משלהם ואז לקרוא להם מתוך קבצי הHTML.

**Example:**

appointments/

├── static/

│ ├── styles.css

│ └── scripts.js

└── templates/

├── login.html

└── owner\_dashboard.html …

**Impact:**

קוד קריא ומודולרי יותר, ניתן לתחזוקה והרחבה ועם ביצועים גבוהים יותר. (בפרויקטים גדולים זה גם מקל על עבודת צוות כי יש יותר הפרדה בקוד וכל אחד יכול לעבוד על חלק אחר).

**Priority**: mid



**Issue/Concern:**

אין תיעוד והגדרת קוד ע"י דוקומנטציה.

**Suggestion/Feedback:**

יש לתעד פונקציות ומודולים. תיעוד של פונקציה יכלול: תיאור הפונקציה, פרמטרים, מה היא מחזירה ואיזה שגיאות היא עלולה להעלות. היא יכולה לכלול עוד במידת הצורך.

תיעוד של מודול צריך לכלול תיאור לתפקיד שלו. ניתן להוסיף דברים נוספים כשמדובר בפרויקטים גדולים יותר כמו dependencies, usage, key components וכו'.

בשביל תיעוד בפיתון יש להשתמש בDocstrings , ראי: https://www.geeksforgeeks.org/python-docstrings/

**Example:**

תיעוד לפונקציה :register

def register(request):

    """

    Register a new user.

    Handles the registration process for new users. Validates the registration form data

    and creates a new user account if the form is valid. Only allows one owner registration.

    Parameters:

        request (HttpRequest): The HTTP request object containing the form data submitted by the user.

    Returns:

        HttpResponseRedirect: Redirects to the login page upon successful registration.

        HttpResponse: Renders the registration form if the request method is GET.

        HttpResponse: Re-renders the registration form with validation errors if the form is invalid.

    Raises:

        Http404: If the requested user does not exist.

        ValueError: If the form data is invalid or incomplete.

    """

    # Function implementation...

תיעוד למודול views.py

התיעוד ימצא בראש הקובץ מעל הimports

"""

Views Module

This module contains Django views for handling HTTP requests related to user authentication, appointment management,

and owner dashboard functionalities.

"""

# Rest of your code...

**Impact:**

תיעוד פונקציות יכולה לעזור לשפר את הקריאות וההבנה עבור מפתחים שעשויים לעבוד על הקוד או לתחזק.

**Priority: mid**

**לסיכום:**

ממליצה על כמה אספקטים וכלים לעבוד איתם:

* ללמוד על סביבה וירטואלית בפיתון.
* ללמוד על Docstrings.
* ללמוד איזה קבצים כדאי לכלול ב.gitignore
* ללמוד כיצד ליצור קובץ README איכותי בפרויקט פיתון.

מעבר לזה הפרויקט נראה טוב והמון בהצלחה 🙂