wie speichere ich Daten aus Flask-WTF-Formular nach dem Absenden in der Datenbank und aktualisiere den aktuell eingeloggten User, und dabei soll verhindert werden das eine email gespeichert wird, die bereits verwendet wurde und zusätzlich das ein eingeloggte Benutzer seine eigene aktuelle email weiterhin verwenden darf?

Du brauchst drei Schritte:

1. Formulardaten auslesen
2. E-Mail-Eindeutigkeit prüfen (aber eigene E-Mail erlauben)
3. `current_user` aktualisieren und speichern

Hier ein kompaktes Beispiel, das genau zu deinem Setup (Flask-WTF, Flask-Login, SQLAlchemy) passt.

```
from flask_login import login_required, current_user
from sqlalchemy import select
from forms import ProfileForm
from db import db, User

@app.route("/profil", methods=["GET", "POST"])
@login_required
def profil():
    # Formular mit aktuellen User-Daten vorbelegen
    form = ProfileForm(obj=current_user)

    if form.validate_on_submit():
        # 1. Prüfen, ob es einen anderen User mit dieser E-Mail gibt
        existing = db.session.execute(
            select(User).filter_by(email=form.email.data)
        ).scalar_one_or_none()

        # a) Es gibt einen User mit dieser E-Mail UND es ist NICHT der aktuell eingeloggt
        if existing and existing.userId != current_user.userId:
            flash("Diese E-Mail-Adresse wird bereits von einem anderen Konto verwendet.",
            return render_template("profil.html", form=form)

        # b) Entweder es gibt keinen anderen User mit der E-Mail
        #    oder es ist der aktuelle User selbst → Update ist erlaubt
        current_user.firstName = form.firstName.data
```

```
        current_user.name = form.name.data
        current_user.email = form.email.data
        current_user.phone = form.phone.data or None

        db.session.commit()
        flash("Profil erfolgreich gespeichert.", "success")

        # Optional: zurück zum passenden Dashboard
        if current_user.role == "helfer":
            return redirect(url_for("helfer"))
        else:
            return redirect(url_for("kunde"))

    # GET oder ungültiges Formular → Seite anzeigen
    return render_template("profil.html", form=form)
```

Was hier passiert:

- `form = ProfileForm(obj=current_user)` füllt das Formular beim GET mit den aktuellen Daten.
- `form.validate_on_submit()` prüft bei POST, ob alle Validatoren ok sind.
- `select(User).filter_by(email=form.email.data)` sucht nach einem Benutzer mit dieser E-Mail.
- `if existing and existing.userId != current_user.userId:` stellt sicher, dass:
  - andere Nutzer diese E-Mail **nicht** haben dürfen,
  - der eingeloggte Nutzer seine eigene E-Mail weiterverwenden darf, auch wenn sie sich nicht ändert.
- Wenn alles okay ist, werden die Felder von `current_user` aktualisiert und mit `db.session.commit()` gespeichert.

Damit erfüllst du alle Anforderungen: Daten werden aus dem Flask-WTF-Formular übernommen, der aktuell eingeloggte User wird aktualisiert, doppelte E-Mails werden verhindert und die eigene E-Mail bleibt erlaubt.

⁂

1. Bildschirmfoto-2025-10-28-um-20.49.30.jpg
2. https://www.youtube.com/watch?v=sFhybwAjDqw
3. https://www.digitalocean.com/community/tutorials/how-to-add-authentication-to-your-app-with-flask-login
4. https://flask-sqlalchemy.readthedocs.io/en/stable/quickstart/
5. Bildschirmfoto-2025-10-28-um-21.47.51.jpg
6. https://flask-security-too.readthedocs.io/en/stable/customizing.html
7. https://laravel.io/forum/04-18-2017-updating-another-user-and-ignoring-its-nameemail-when-validating
8. https://stackoverflow.com/questions/44703577/using-same-form-to-create-and-update-but-different-validation-logic
9. https://stackoverflow.com/questions/9911467/sqlalchemy-update-if-unique-key-exists

10. https://flask-security-too.readthedocs.io/en/stable/api.html
11. https://laracasts.com/discuss/channels/laravel/email-unique-but-not-when-its-users-own-email-and-user-tries-to-update-his-profile
12. https://stackoverflow.com/questions/5685831/unique-validator-in-wtforms-with-sqlalchemy-models