```
import logging


import attr
from flask import Blueprint, flash, redirect, request, url_for
from flask.views import MethodView
from flask_babelplus import gettext as _
from flask_login import current_user, login_required
from pluggy import HookimplMarker


@attr.s(frozen=True, cmp=False, hash=False, repr=True)
class UserSettings(MethodView):
    form = attr.ib(factory=settings_form_factory)
    settings_update_handler = attr.ib(factory=settings_update_handler)


    decorators = [login_required]


    def get(self):
        return self.render()


    def post(self):
        if self.form.validate_on_submit():
            try:
                self.settings_update_handler.apply_changeset(
                    current_user, self.form.as_change()
                )
            except StopValidation as e:
                self.form.populate_errors(e.reasons)
                return self.render()
            except PersistenceError:
                logger.exception("Error while updating user settings")
                flash(_("Error while updating user settings"), "danger")
                return self.redirect()

            flash(_("Settings updated."), "success")
            return self.redirect()
        return self.render()

    def render(self):
        return render_template("user/general_settings.html", form=self.form)

    def redirect(self):
        return redirect(url_for("user.settings"))


@attr.s(frozen=True, hash=False, cmp=False, repr=True)
class ChangePassword(MethodView):
    form = attr.ib(factory=change_password_form_factory)
    password_update_handler = attr.ib(factory=password_update_handler)
    decorators = [login_required]

    def get(self):
        return self.render()

    def post(self):
        if self.form.validate_on_submit():
            try:
                self.password_update_handler.apply_changeset(
                    current_user, self.form.as_change()
            )
        except StopValidation as e:
            self.form.populate_errors(e.reasons)
            return self.render()
        except PersistenceError:
            logger.exception("Error while changing password")
            flash(_("Error while changing password"), "danger")
            return self.redirect()

        flash(_("Password updated."), "success")
        return self.redirect()
    return self.render()

def render(self):
    return render_template("user/change_password.html", form=self.form)

def redirect(self):
    return redirect(url_for("user.change_password"))


@attr.s(frozen=True, cmp=False, hash=False, repr=True)
class ChangeEmail(MethodView):
    form = attr.ib(factory=change_email_form_factory)
    update_email_handler = attr.ib(factory=email_update_handler)
    decorators = [login_required]

    def get(self):
        return self.render()
```

```
def post(self):
    if self.form.validate_on_submit():
        try:
            self.update_email_handler.apply_changeset(
                current_user, self.form.as_change()
            )
        except StopValidation as e:
            self.form.populate_errors(e.reasons)
            return self.render()
        except PersistenceError:
            logger.exception("Error while updating email")
            flash(_("Error while updating email"), "danger")
            return self.redirect()

        flash(_("Email address updated."), "success")
        return self.redirect()
    return self.render()

def render(self):
    return render_template("user/change_email.html", form=self.form)

def redirect(self):
    return redirect(url_for("user.change_email"))
```