```
import threading
import typing

import flask
import flask_mail # type: ignore

TESTING = False
DEBUG_PRINT_EMAIL = False

mail_lock = threading.Lock()



class MailKeeper:
    """Singleton to maintain SMTP mail client."""

    instance = None

    @classmethod
    def get_instance(cls) -> typing.Optional['MailKeeper']:
        """Get a shared instance of this singleton.
        @return: Shared instance of the MailKeeper singleton or None if mail
            is disabled.
        @rtype: MailKeeper
        """
        return cls.instance

    @classmethod
    def init_mail(cls, app: flask.Flask) -> None:
        """Initialize the global MailKeeper singleton for the given app.
        @param app: The Flask app to initialize with.
        @type app: flask.Flask
        """
        cls.instance = MailKeeper(app)
```

```
    def _init_(self, app: flask.Flask):
        """Create a new MailKeeper.
        @param app: The Flask app to create mailing capabilites for.
        @type app: flask.Flask
        """
        self.__mail = flask_mail.Mail(app)
        self.__from_addr = app.config['MAIL_SEND_FROM']

    def get_mail_instance(self) -> flask_mail.Mail:
        """Get the underlying Flask-Mail client.
        @return: Flask-Mail SMTP client / facade.
        @rtype: flask_mail.Mail
        """
        return self.__mail

    def get_from_addr(self) -> str:
        """Get the address from which mail will be sent.
        @return: Email address
        """
        return self.__from_addr

def init_mail(app: flask.Flask):
    """Initialize the MailKeeper singleton to enable SMTP capabilities.
    Creates the system-wide MailKeeper singleton which allows the application to
    send email. This should be called once at the initialization of the Flask
    application.
    @param app: The flask application to initialize the MailKeeper with.
    @type app: flask.Flask
    """
    MailKeeper.init_mail(app)

def disable_mail() -> None:
    """Disable mail sending."""
    MailKeeper.instance = None
```

```
def get_mail_keeper() -> typing.Optional[MailKeeper]:
    """Get the current mail keeper or None if mail disabled."""
    return MailKeeper.get_instance()


def send_msg(email: str, subject: str, message: str) -> None:
    """Send an email.
    Sends an email through the current MailKeeper or takes no action if a
    mail keeper is not available.
    @param email: The email address to which the message should be sent.
    @param subject: The subject line.
    @param message: The message to send.
    """
    with mail_lock:

        mail_keeper = get_mail_keeper()
        mail_keeper_realized: MailKeeper

        if mail_keeper:
            mail_keeper_realized = mail_keeper # type: ignore
            flask_message = flask_mail.Message(
                subject,
                sender=mail_keeper.get_from_addr(),
                recipients=[email.replace(' ', '')],
                body=message
            )
            mail_keeper_realized.get_mail_instance().send(flask_message)

        else:
            if DEBUG_PRINT_EMAIL:
                print(message)
```