```python
# Source: Oppia - https://github.com/oppia/oppia - an online learning tool that enables anyone to easily create and share interactive activities

"""Provides email services."""

from __future__ import absolute_import  # pylint: disable=import-only-modules
from __future__ import unicode_literals  # pylint: disable=import-only-modules

import feconf
import python_utils

from google.appengine.api import mail


def get_incoming_email_address(reply_to_id):
    """Gets the incoming email address. The client is responsible for recording
    any audit logs.
    Args:
        reply_to_id: str. The unique id of the sender.
    Returns:
        str. The email address of the sender.
    """
    return 'reply+%s@%s' % (reply_to_id, feconf.INCOMING_EMAILS_DOMAIN_NAME)


def _is_email_valid(email_address):
    """Determines whether an email address is invalid.
    Args:
        email_address: str. Email address to check.
    Returns:
        bool. Whether specified email address is valid.
    """
    if not isinstance(email_address, python_utils.BASESTRING):
        return False

    stripped_address = email_address.strip()
    if not stripped_address:
        return False

    return True


def _is_sender_email_valid(sender_email):
    """Gets the sender_email address and validates it is of the form
    'SENDER_NAME <SENDER_EMAIL_ADDRESS>'.
    Args:
        sender_email: str. The email address of the sender.
    Returns:
        bool. Whether the sender_email is valid.
    """
    if not _is_email_valid(sender_email):
        return False

    split_sender_email = sender_email.split(' ')
    if len(split_sender_email) < 2:
        return False

    email_address = split_sender_email[-1]
```

```
    if not email_address.startswith('<') or not email_address.endswith('>'):
        return False

    return True


def send_mail(
        sender_email, recipient_email, subject, plaintext_body, html_body,
        bcc_admin=False, reply_to_id=None):
    """Sends an email. The client is responsible for recording any audit logs.
    In general this function should only be called from
    email_manager._send_email().
    Args:
        sender_email: str. The email address of the sender. This should be in
            the form 'SENDER_NAME <SENDER_EMAIL_ADDRESS>'.
        recipient_email: str. The email address of the recipient.
        subject: str. The subject line of the email.
        plaintext_body: str. The plaintext body of the email.
        html_body: str. The HTML body of the email. Must fit in a datastore
            entity.
        bcc_admin: bool. Whether to bcc feconf.ADMIN_EMAIL_ADDRESS on the email.
        reply_to_id: str or None. The unique reply-to id used in reply-to email
            sent to recipient.
    Raises:
        ValueError: If 'sender_email' or 'recipient_email' is invalid, according
            to App Engine.
        Exception: If the configuration in feconf.py forbids emails from being
            sent.
    """
    if not feconf.CAN_SEND_EMAILS:
        raise Exception('This app cannot send emails.')

    if not _is_email_valid(recipient_email):
        raise ValueError(
            'Malformed recipient email address: %s' % recipient_email)

    if not _is_sender_email_valid(sender_email):
        raise ValueError(
            'Malformed sender email address: %s' % sender_email)

    msg = mail.EmailMessage(
        sender=sender_email, to=recipient_email,
        subject=subject, body=plaintext_body, html=html_body)
    if bcc_admin:
        msg.bcc = [feconf.ADMIN_EMAIL_ADDRESS]
    if reply_to_id:
        msg.reply_to = get_incoming_email_address(reply_to_id)

    # Send message.
    msg.send()


def send_bulk_mail(
        sender_email, recipient_emails, subject, plaintext_body, html_body):
```

```
112     """Sends an email. The client is responsible for recording any audit
    logs.
113     In general this function should only be called from
114     email_manager._send_email().
115     Args:
116         sender_email: str. The email address of the sender. This should be in
117             the form 'SENDER_NAME <SENDER_EMAIL_ADDRESS>'.
118         recipient_emails: list(str). The list of recipients' email addresses.
119         subject: str. The subject line of the email.
120         plaintext_body: str. The plaintext body of the email.
121         html_body: str. The HTML body of the email. Must fit in a datastore
122             entity.
123     Raises:
124         ValueError: If 'sender_email' or 'recipient_email' is invalid,
    according
125             to App Engine.
126         Exception: If the configuration in feconf.py forbids emails from
    being
127             sent.
128     """
129     if not feconf.CAN_SEND_EMAILS:
130         raise Exception('This app cannot send emails.')
131 
132     for recipient_email in recipient_emails:
133         if not _is_email_valid(recipient_email):
134             raise ValueError(
135                 'Malformed recipient email address: %s' % recipient_email)
136 
137     if not _is_sender_email_valid(sender_email):
138         raise ValueError(
139             'Malformed sender email address: %s' % sender_email)
140 
141     for recipient_email in recipient_emails:
142         mail.send_mail(
143             sender_email, recipient_email, subject, plaintext_body,
144             html=html_body)
```