

```
1 # Source: Oppia - https://github.com/oppia/oppia - an online learning tool
  # that enables anyone to easily create and share interactive activities
2
3 """Tests for the GAE mail API wrapper."""
4
5 from __future__ import absolute_import # pylint: disable=import-only-modules
6 from __future__ import unicode_literals # pylint: disable=import-only-
  modules
7
8 from core.platform.email import gae_email_services
9 from core.tests import test_utils
10 import feconf
11
12
13 class EmailTests(test_utils.GenericTestBase):
14     """Tests for sending emails."""
15     RECIPIENT_EMAIL = 'user@example.com'
16     RECIPIENT_USERNAME = 'user'
17     SENDER_EMAIL = 'Sender <sender@example.com>'
18
19     def test_sending_email(self):
20         # Emails are not sent if the CAN_SEND_EMAILS setting is not turned
  on.
21         email_exception = self.assertRaisesRegexp(
22             Exception, 'This app cannot send emails.')
23         messages = self.mail_stub.get_sent_messages(
24             to=feconf.ADMIN_EMAIL_ADDRESS)
25         self.assertEqual(0, len(messages))
26         with self.swap(feconf, 'CAN_SEND_EMAILS', False), email_exception:
27             gae_email_services.send_mail(
28                 self.SENDER_EMAIL, feconf.ADMIN_EMAIL_ADDRESS,
29                 'subject', 'body', 'html', bcc_admin=False)
30         messages = self.mail_stub.get_sent_messages(
31             to=feconf.ADMIN_EMAIL_ADDRESS)
32         self.assertEqual(0, len(messages))
33
34         messages = self.mail_stub.get_sent_messages(
35             to=feconf.ADMIN_EMAIL_ADDRESS)
36         self.assertEqual(0, len(messages))
37         with self.swap(feconf, 'CAN_SEND_EMAILS', True):
38             gae_email_services.send_mail(
39                 self.SENDER_EMAIL, feconf.ADMIN_EMAIL_ADDRESS,
40                 'subject', 'body', 'html', bcc_admin=False)
41         messages = self.mail_stub.get_sent_messages(
42             to=feconf.ADMIN_EMAIL_ADDRESS)
43         self.assertEqual(1, len(messages))
44
45     def test_email_bcc_is_sent_if_bcc_admin_is_true(self):
46         # Tests that email has ADMIN_EMAIL_ADDRESS as bcc if bcc_admin is
  True.
47         messages = self.mail_stub.get_sent_messages(to=self.RECIPIENT_EMAIL)
48         self.assertEqual(0, len(messages))
49         with self.swap(feconf, 'CAN_SEND_EMAILS', True):
50             gae_email_services.send_mail(
51                 self.SENDER_EMAIL, self.RECIPIENT_EMAIL,
52                 'subject', 'body', 'html', bcc_admin=True)
53         messages = self.mail_stub.get_sent_messages(to=self.RECIPIENT_EMAIL)
54         self.assertEqual(1, len(messages))
55         self.assertEqual(messages[0].bcc, feconf.ADMIN_EMAIL_ADDRESS)
56
```

```
57 def test_email_bcc_not_sent_if_bcc_admin_is_false(self):
58     # Tests that email has no bcc attribute if bcc_admin is False.
59     messages = self.mail_stub.get_sent_messages(to=self.RECIPIENT_EMAIL)
60     self.assertEqual(0, len(messages))
61     with self.swap(feconf, 'CAN_SEND_EMAILS', True):
62         gae_email_services.send_mail(
63             self.SENDER_EMAIL, self.RECIPIENT_EMAIL,
64             'subject', 'body', 'html', bcc_admin=False)
65     messages = self.mail_stub.get_sent_messages(to=self.RECIPIENT_EMAIL)
66     self.assertEqual(1, len(messages))
67     self.assertFalse(hasattr(messages[0], 'bcc'))
68
69 def test_sending_email_with_reply_to_id_adds_reply_to_email(self):
70     # Tests that email has reply_to_address if reply_to_id is passed.
71     messages = self.mail_stub.get_sent_messages(to=self.RECIPIENT_EMAIL)
72     self.assertEqual(0, len(messages))
73
74     reply_to_id = 'reply_to_id'
75     with self.swap(feconf, 'CAN_SEND_EMAILS', True):
76         gae_email_services.send_mail(
77             self.SENDER_EMAIL, self.RECIPIENT_EMAIL,
78             'subject', 'body', 'html', reply_to_id=reply_to_id)
79     messages = self.mail_stub.get_sent_messages(
80         to=self.RECIPIENT_EMAIL)
81     expected_reply_to_address = 'reply+reply_to_id@example.com'
82     self.assertEqual(messages[0].reply_to, expected_reply_to_address)
83
84 def test_sending_email_without_reply_to_id_not_add_reply_to_email(self):
85     """Tests that email does not have reply_to_address if reply_to_id is
86     not passed.
87     """
88     messages = self.mail_stub.get_sent_messages(to=self.RECIPIENT_EMAIL)
89     self.assertEqual(0, len(messages))
90     with self.swap(feconf, 'CAN_SEND_EMAILS', True):
91         gae_email_services.send_mail(
92             self.SENDER_EMAIL, self.RECIPIENT_EMAIL,
93             'subject', 'body', 'html')
94     messages = self.mail_stub.get_sent_messages(
95         to=self.RECIPIENT_EMAIL)
96     self.assertFalse(hasattr(messages[0], 'reply_to'))
97
98 def test_email_not_sent_if_sender_address_is_malformed(self):
99     # Tests that email is not send if sender email address is malformed.
100
101     # Case when malformed_sender_email is None.
102     messages = self.mail_stub.get_sent_messages(to=self.RECIPIENT_EMAIL)
103     self.assertEqual(0, len(messages))
104     malformed_sender_email = None
105     email_exception = self.assertRaisesRegex(
106         ValueError, 'Malformed sender email address: %s'
107         % malformed_sender_email)
108     with self.swap(feconf, 'CAN_SEND_EMAILS', True), email_exception:
109         gae_email_services.send_mail(
110             malformed_sender_email, self.RECIPIENT_EMAIL,
111             'subject', 'body', 'html')
112     messages = self.mail_stub.get_sent_messages(to=self.RECIPIENT_EMAIL)
113     self.assertEqual(0, len(messages))
114
115     # Case when malformed_sender_email is an empty string.
116     messages = self.mail_stub.get_sent_messages(to=self.RECIPIENT_EMAIL)
```

```
117 self.assertEqual(0, len(messages))
118 malformed_sender_email = ''
119 email_exception = self.assertRaisesRegex(
120     ValueError, 'Malformed sender email address: %s'
121     % malformed_sender_email)
122 with self.swap(feconf, 'CAN_SEND_EMAILS', True), email_exception:
123     gae_email_services.send_mail(
124         malformed_sender_email, self.RECIPIENT_EMAIL,
125         'subject', 'body', 'html')
126 messages = self.mail_stub.get_sent_messages(to=self.RECIPIENT_EMAIL)
127 self.assertEqual(0, len(messages))
128
129 # Case when malformed_sender_email does not have name of the sender,
130 # only the email address.
131 messages = self.mail_stub.get_sent_messages(to=self.RECIPIENT_EMAIL)
132 self.assertEqual(0, len(messages))
133 malformed_sender_email = '<malformed_sender_email@example.com>'
134 email_exception = self.assertRaisesRegex(
135     ValueError, 'Malformed sender email address: %s'
136     % malformed_sender_email)
137 with self.swap(feconf, 'CAN_SEND_EMAILS', True), email_exception:
138     gae_email_services.send_mail(
139         malformed_sender_email, self.RECIPIENT_EMAIL,
140         'subject', 'body', 'html')
141 messages = self.mail_stub.get_sent_messages(to=self.RECIPIENT_EMAIL)
142 self.assertEqual(0, len(messages))
143
144 # Case when malformed_sender_email does not have email address of the
145 # sender and only the name.
146 messages = self.mail_stub.get_sent_messages(to=self.RECIPIENT_EMAIL)
147 self.assertEqual(0, len(messages))
148 malformed_sender_email = 'MalformedSender'
149 email_exception = self.assertRaisesRegex(
150     ValueError, 'Malformed sender email address: %s'
151     % malformed_sender_email)
152 with self.swap(feconf, 'CAN_SEND_EMAILS', True), email_exception:
153     gae_email_services.send_mail(
154         malformed_sender_email, self.RECIPIENT_EMAIL,
155         'subject', 'body', 'html')
156 messages = self.mail_stub.get_sent_messages(to=self.RECIPIENT_EMAIL)
157 self.assertEqual(0, len(messages))
158
159 # Case when malformed_sender_email does not have email address of the
160 # form '<sender_email_address>'.
161 messages = self.mail_stub.get_sent_messages(to=self.RECIPIENT_EMAIL)
162 self.assertEqual(0, len(messages))
163 malformed_sender_email = 'MalformedSender malformed_sender@gmail.com'
164 email_exception = self.assertRaisesRegex(
165     ValueError, 'Malformed sender email address: %s'
166     % malformed_sender_email)
167 with self.swap(feconf, 'CAN_SEND_EMAILS', True), email_exception:
168     gae_email_services.send_mail(
169         malformed_sender_email, self.RECIPIENT_EMAIL,
170         'subject', 'body', 'html')
171 messages = self.mail_stub.get_sent_messages(to=self.RECIPIENT_EMAIL)
172 self.assertEqual(0, len(messages))
173
174 def test_email_not_sent_if_recipient_address_is_malformed(self):
175     # Tests that email is not sent if recipient email address is
malformed.
```

```

176
177     # Case when malformed_recipient_email is None.
178     malformed_recipient_email = None
179     email_exception = self.assertRaisesRegex(
180         ValueError, 'Malformed recipient email address: %s'
181         % malformed_recipient_email)
182     with self.swap(feconf, 'CAN_SEND_EMAILS', True), email_exception:
183         gae_email_services.send_mail(
184             self.SENDER_EMAIL, malformed_recipient_email,
185             'subject', 'body', 'html')
186
187     # Case when malformed_recipient_email is an empty string.
188     malformed_recipient_email = ''
189     email_exception = self.assertRaisesRegex(
190         ValueError, 'Malformed recipient email address: %s'
191         % malformed_recipient_email)
192     with self.swap(feconf, 'CAN_SEND_EMAILS', True), email_exception:
193         gae_email_services.send_mail(
194             self.SENDER_EMAIL, malformed_recipient_email,
195             'subject', 'body', 'html')
196
197
198 class BulkEmailsTests(test_utils.GenericTestBase):
199     SENDER_EMAIL = 'Sender <sender@example.com>'
200     SENDER_USERNAME = 'sender'
201     RECIPIENT_A_EMAIL = 'a@example.com'
202     RECIPIENT_A_USERNAME = 'usera'
203     RECIPIENT_B_EMAIL = 'b@example.com'
204     RECIPIENT_B_USERNAME = 'userb'
205     RECIPIENT_EMAILS = [RECIPIENT_A_EMAIL, RECIPIENT_B_EMAIL]
206
207     def test_correct_bulk_emails_sent(self):
208         message_a = self.mail_stub.get_sent_messages(
209             to=self.RECIPIENT_EMAILS[0])
210         self.assertEqual(len(message_a), 0)
211         message_b = self.mail_stub.get_sent_messages(
212             to=self.RECIPIENT_EMAILS[1])
213         self.assertEqual(len(message_b), 0)
214         with self.swap(feconf, 'CAN_SEND_EMAILS', True):
215             gae_email_services.send_bulk_mail(
216                 self.SENDER_EMAIL, self.RECIPIENT_EMAILS,
217                 'subject', 'body', 'html')
218         message_a = self.mail_stub.get_sent_messages(
219             to=self.RECIPIENT_EMAILS[0])
220         self.assertEqual(len(message_a), 1)
221         message_b = self.mail_stub.get_sent_messages(
222             to=self.RECIPIENT_EMAILS[1])
223         self.assertEqual(len(message_b), 1)
224
225     def test_bulk_emails_not_sent_if_can_send_emails_is_false(self):
226         # Emails are not sent if the CAN_SEND_EMAILS setting is not turned
227         on.
228         message_a = self.mail_stub.get_sent_messages(
229             to=self.RECIPIENT_EMAILS[0])
230         self.assertEqual(len(message_a), 0)
231         message_b = self.mail_stub.get_sent_messages(
232             to=self.RECIPIENT_EMAILS[1])
233         self.assertEqual(len(message_b), 0)
234         email_exception = self.assertRaisesRegex(
235             Exception, 'This app cannot send emails.')

```

```
235     with email_exception:
236         gae_email_services.send_bulk_mail(
237             self.SENDER_EMAIL, self.RECIPIENT_EMAILS,
238             'subject', 'body', 'html')
239     message_a = self.mail_stub.get_sent_messages(
240         to=self.RECIPIENT_EMAILS[0])
241     self.assertEqual(len(message_a), 0)
242     message_b = self.mail_stub.get_sent_messages(
243         to=self.RECIPIENT_EMAILS[1])
244     self.assertEqual(len(message_b), 0)
245
246     def test_bulk_mails_not_sent_if_sender_email_is_malformed(self):
247         """Tests that bulk emails are not sent if sender email address is
248         malformed.
249         """
250
251         # Case when malformed_sender_email is None.
252         message_a = self.mail_stub.get_sent_messages(
253             to=self.RECIPIENT_EMAILS[0])
254         self.assertEqual(0, len(message_a))
255         message_b = self.mail_stub.get_sent_messages(
256             to=self.RECIPIENT_EMAILS[1])
257         self.assertEqual(0, len(message_b))
258         malformed_sender_email = None
259         email_exception = self.assertRaisesRegex(
260             ValueError, 'Malformed sender email address: %s'
261             % malformed_sender_email)
262         with self.swap(feconf, 'CAN_SEND_EMAILS', True), email_exception:
263             gae_email_services.send_bulk_mail(
264                 malformed_sender_email, self.RECIPIENT_EMAILS,
265                 'subject', 'body', 'html')
266         message_a = self.mail_stub.get_sent_messages(
267             to=self.RECIPIENT_EMAILS[0])
268         self.assertEqual(0, len(message_a))
269         message_b = self.mail_stub.get_sent_messages(
270             to=self.RECIPIENT_EMAILS[1])
271         self.assertEqual(0, len(message_b))
272
273         # Case when malformed_sender_email is an empty string.
274         message_a = self.mail_stub.get_sent_messages(
275             to=self.RECIPIENT_EMAILS[0])
276         self.assertEqual(0, len(message_a))
277         message_b = self.mail_stub.get_sent_messages(
278             to=self.RECIPIENT_EMAILS[1])
279         self.assertEqual(0, len(message_b))
280         malformed_sender_email = ''
281         email_exception = self.assertRaisesRegex(
282             ValueError, 'Malformed sender email address: %s'
283             % malformed_sender_email)
284         with self.swap(feconf, 'CAN_SEND_EMAILS', True), email_exception:
285             gae_email_services.send_bulk_mail(
286                 malformed_sender_email, self.RECIPIENT_EMAILS,
287                 'subject', 'body', 'html')
288         message_a = self.mail_stub.get_sent_messages(
289             to=self.RECIPIENT_EMAILS[0])
290         self.assertEqual(0, len(message_a))
291         message_b = self.mail_stub.get_sent_messages(
292             to=self.RECIPIENT_EMAILS[1])
293         self.assertEqual(0, len(message_b))
294
```



```

295 # Case when malformed_sender_email does not have name of the sender
296 # and only the email address.
297 message_a = self.mail_stub.get_sent_messages(
298     to=self.RECIPIENT_EMAILS[0])
299 self.assertEqual(0, len(message_a))
300 message_b = self.mail_stub.get_sent_messages(
301     to=self.RECIPIENT_EMAILS[1])
302 self.assertEqual(0, len(message_b))
303 malformed_sender_email = '<malformed_sender_email@example.com>'
304 email_exception = self.assertRaisesRegex(
305     ValueError, 'Malformed sender email address: %s'
306     % malformed_sender_email)
307 with self.swap(feconf, 'CAN_SEND_EMAILS', True), email_exception:
308     gae_email_services.send_bulk_mail(
309         malformed_sender_email, self.RECIPIENT_EMAILS,
310         'subject', 'body', 'html')
311 message_a = self.mail_stub.get_sent_messages(
312     to=self.RECIPIENT_EMAILS[0])
313 self.assertEqual(0, len(message_a))
314 message_b = self.mail_stub.get_sent_messages(
315     to=self.RECIPIENT_EMAILS[1])
316 self.assertEqual(0, len(message_b))
317
318 # Case when malformed_sender_email does not have email address of the
319 # sender and only the name.
320 message_a = self.mail_stub.get_sent_messages(
321     to=self.RECIPIENT_EMAILS[0])
322 self.assertEqual(0, len(message_a))
323 message_b = self.mail_stub.get_sent_messages(
324     to=self.RECIPIENT_EMAILS[1])
325 self.assertEqual(0, len(message_b))
326 malformed_sender_email = 'MalformedSender'
327 email_exception = self.assertRaisesRegex(
328     ValueError, 'Malformed sender email address: %s'
329     % malformed_sender_email)
330 with self.swap(feconf, 'CAN_SEND_EMAILS', True), email_exception:
331     gae_email_services.send_bulk_mail(
332         malformed_sender_email, self.RECIPIENT_EMAILS,
333         'subject', 'body', 'html')
334 message_a = self.mail_stub.get_sent_messages(
335     to=self.RECIPIENT_EMAILS[0])
336 self.assertEqual(0, len(message_a))
337 message_b = self.mail_stub.get_sent_messages(
338     to=self.RECIPIENT_EMAILS[1])
339 self.assertEqual(0, len(message_b))
340
341 # Case when malformed_sender_email does not have email address of the
342 # form '<sender_email_address>'.
343 message_a = self.mail_stub.get_sent_messages(
344     to=self.RECIPIENT_EMAILS[0])
345 self.assertEqual(0, len(message_a))
346 message_b = self.mail_stub.get_sent_messages(
347     to=self.RECIPIENT_EMAILS[1])
348 self.assertEqual(0, len(message_b))
349 malformed_sender_email = 'Malformed Sender
malformed_sender@example.com'
350 email_exception = self.assertRaisesRegex(
351     ValueError, 'Malformed sender email address: %s'
352     % malformed_sender_email)
353 with self.swap(feconf, 'CAN_SEND_EMAILS', True), email_exception:

```

```
354         gae_email_services.send_bulk_mail(
355             malformed_sender_email, self.RECIPIENT_EMAILS,
356             'subject', 'body', 'html')
357     message_a = self.mail_stub.get_sent_messages(
358         to=self.RECIPIENT_EMAILS[0])
359     self.assertEqual(0, len(message_a))
360     message_b = self.mail_stub.get_sent_messages(
361         to=self.RECIPIENT_EMAILS[1])
362     self.assertEqual(0, len(message_b))
363
364     def test_bulk_mails_not_sent_if_recipient_email_is_malformed(self):
365         """Tests that bulk emails are not sent if recipient email address is
366         malformed.
367         """
368
369         # Case when both recipient email address strings are None.
370         malformed_recipient_emails = [None, None]
371         email_exception = self.assertRaisesRegex(
372             ValueError, 'Malformed recipient email address: %s'
373             % malformed_recipient_emails[0])
374         with self.swap(feconf, 'CAN_SEND_EMAILS', True), email_exception:
375             gae_email_services.send_bulk_mail(
376                 self.SENDER_EMAIL, malformed_recipient_emails,
377                 'subject', 'body', 'html')
378         messages_a = self.mail_stub.get_sent_messages(
379             to=malformed_recipient_emails[0])
380         self.assertEqual(len(messages_a), 0)
381         messages_b = self.mail_stub.get_sent_messages(
382             to=malformed_recipient_emails[1])
383         self.assertEqual(len(messages_b), 0)
384
385         # Case when both recipient email address strings are empty.
386         malformed_recipient_emails = ['', '']
387         email_exception = self.assertRaisesRegex(
388             ValueError, 'Malformed recipient email address: %s'
389             % malformed_recipient_emails[0])
390         with self.swap(feconf, 'CAN_SEND_EMAILS', True), email_exception:
391             gae_email_services.send_bulk_mail(
392                 self.SENDER_EMAIL, malformed_recipient_emails,
393                 'subject', 'body', 'html')
394         messages_a = self.mail_stub.get_sent_messages(
395             to=malformed_recipient_emails[0])
396         self.assertEqual(len(messages_a), 0)
397         messages_b = self.mail_stub.get_sent_messages(
398             to=malformed_recipient_emails[1])
399         self.assertEqual(len(messages_b), 0)
400
401         # Case when one of the recipient email address strings is None.
402         malformed_recipient_emails = [self.RECIPIENT_A_EMAIL, None]
403         with self.swap(feconf, 'CAN_SEND_EMAILS', True), email_exception:
404             gae_email_services.send_bulk_mail(
405                 self.SENDER_EMAIL, malformed_recipient_emails,
406                 'subject', 'body', 'html')
407         messages_a = self.mail_stub.get_sent_messages(
408             to=malformed_recipient_emails[0])
409         self.assertEqual(len(messages_a), 0)
410         messages_b = self.mail_stub.get_sent_messages(
411             to=malformed_recipient_emails[1])
412         self.assertEqual(len(messages_b), 0)
413
```

```
414 # Case when one of the recipient email address strings is empty.
415 malformed_recipient_emails = [self.RECIPIENT_A_EMAIL, '']
416 with self.swap(feconf, 'CAN_SEND_EMAILS', True), email_exception:
417     gae_email_services.send_bulk_mail(
418         self.SENDER_EMAIL, malformed_recipient_emails,
419         'subject', 'body', 'html')
420 messages_a = self.mail_stub.get_sent_messages(
421     to=malformed_recipient_emails[0])
422 self.assertEqual(len(messages_a), 0)
423 messages_b = self.mail_stub.get_sent_messages(
424     to=malformed_recipient_emails[1])
425 self.assertEqual(len(messages_b), 0)
426
```