# Question 1

Transport Layer: TCP

Application Layer: SMTP

Email: [johnsmith@hotrides.com](mailto:johnsmith@hotrides.com)

Subject: “Check my new ride!!!”

Color: Blue silver



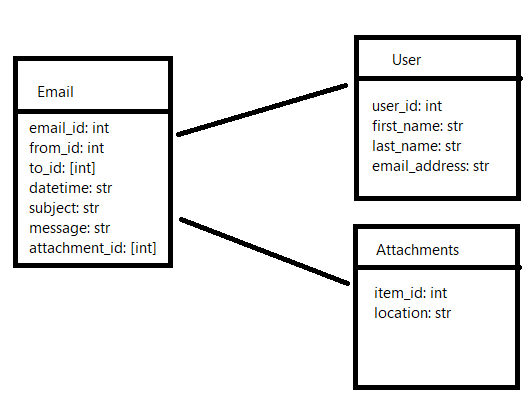
Password: “bestThief”

# Question 2

Snippet from “email\_data\_modeling.py”

from typing import List, Optional  
  
  
class User:  
 def \_\_init\_\_(  
 self, user\_id: int, first\_name: str, last\_name: str, email\_address: str  
 ):  
 self.\_user\_id = user\_id  
 self.\_first\_name = first\_name  
 self.\_last\_name = last\_name  
 self.\_email\_address = email\_address  
  
  
class Attachment:  
 def \_\_init\_\_(self, item\_id: int, path: str):  
 self.\_item\_id = item\_id  
 self.\_path = path  
  
  
class Email:  
 def \_\_init\_\_(  
 self, from\_: User, to: List[User], date\_time\_: str  
 , subject: Optional[str] = None  
 , message: Optional[str] = None  
 , attachment: List[Optional[Attachment]] = None  
 ):  
 self.\_from = from\_  
 self.\_to = to  
 self.\_date\_time\_ = date\_time\_  
 self.\_subject = subject  
 self.\_message = message  
 self.\_attachment = attachment

# Question 3



Users table has following column:

1. user\_id: assigned to each user
2. first\_name
3. last\_name
4. email\_address

Attachments table has the following column:

1. item\_id: assigned to each item attachment
2. location: path to where the item is stored.

Email has the following column:

1. email\_id: assigned to each email
2. from\_id: the user\_id of the sender, user\_id is used to fetch information from Users Table
3. to\_id: an array of user\_id of the receivers, which is used to fetch information from User Table
4. datetime: string representation of when the email is sent
5. subject: string representing the subject of the email
6. message: string representing the message of the email
7. attachment\_id: an array of item\_id, which is used to fetch information from Attachment Table

# Question 4

Snippet from “email\_parsing.py”

from typing import Optional  
  
import yaml  
  
  
def find\_sender\_email(string: Optional[str] = None, file\_location: Optional[str] = None) -> (str, str):  
 if not string and not file\_location:  
 return None  
  
 from\_field = "From" # Sender  
  
 data = yaml.safe\_load(string) if string else None  
 if not data:  
 with open(fr"{file\_location}") as f:  
 data = yaml.safe\_load(f)  
  
 sender\_data = data[from\_field].split(" ")  
 sender\_email = sender\_data[-1]  
 if sender\_email[0] == "<":  
 sender\_email = sender\_email[1:-1]  
 return " ".join(sender\_data[:-1]), sender\_email

# Question 5

Snippet from “test\_email\_parsing.py”

import unittest  
  
from email\_parsing import find\_sender\_email  
  
RAW\_STRING = "Received: from x.y.test by example.netvia TCP with ESMTP id ABC12345" \  
 "for <mary@example.net>; 21 Nov 1997 10:05:43 -0600\n" \  
 "Received: from machine.example by x.y.test; 21 Nov 1997 10:01:22 -0600\n" \  
 "From: John Doe <jdoe@machine.example>\n" \  
 "To: Mary Smith <mary@example.net>\n" \  
 "Subject: Saying Hello\n" \  
 "Date: Fri, 21 Nov 1997 09:55:06 -0600\n" \  
 "Message-ID: <1234@local.machine.example>\n"  
  
RAW\_STRING\_NO\_BRACKET = "From: John Doe jdoe@machine.example\n" \  
 "To: Mary Smith <mary@example.net>\n"  
  
RAW\_STRING\_NO\_NAME = "From: jdoe@machine.example\n" \  
 "To: Mary Smith <mary@example.net>\n"  
  
RAW\_STRING\_MANY\_NAME = "From: John Tim Apple Doe jdoe@machine.example\n" \  
 "To: Mary Smith <mary@example.net>\n"  
  
  
class TestEmailParser(unittest.TestCase):  
 def setUp(self):  
 self.email\_original = RAW\_STRING  
 self.email\_no\_bracket = RAW\_STRING\_NO\_BRACKET  
 self.email\_no\_name = RAW\_STRING\_NO\_NAME  
 self.email\_many\_name = RAW\_STRING\_MANY\_NAME  
  
 def test\_original(self):  
 self.assertEqual(find\_sender\_email(self.email\_original), ('John Doe', 'jdoe@machine.example'))  
  
 def test\_no\_bracket(self):  
 self.assertEqual(find\_sender\_email(self.email\_no\_bracket), ('John Doe', 'jdoe@machine.example'))  
  
 def test\_no\_name(self):  
 self.assertEqual(find\_sender\_email(self.email\_no\_name), ('', 'jdoe@machine.example'))  
  
 def test\_many\_name(self):  
 self.assertEqual(find\_sender\_email(self.email\_many\_name), ('John Tim Apple Doe', 'jdoe@machine.example'))  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 unittest.main()

# Question 6

The user interacts with the web interface to read email intercepted in the database. When the user enters the interface, the user will be given a list of all the emails sorted by date and time the email was sent. There will be options to allow the user to sort the email based on the database criteria such as “from” and “if there are attachments”. For filtering, each column will have a drop-down menu to filter the desired object. A search bar will be included to help with the filtering. A header in the search bar before the string input by the user will help narrow down the column the user wishes to search.