**2/11/19**

**COMP-3700**

**HW3**

**Notes:** 1.) All questions have been copied down to this document for readability.

2.) State changes cannot be modeled correctly in ArgoUML, so they have been modeled with action states (professor’s permission was received).

3.) Exceptions for use cases have been combined with the regular scenarios for all 5 sequence diagrams (professor’s permission was received).

4.) Some diagrams may need to be zoomed in to see all details.

1. Consider a computer email system.
   1. List three actors. Explain the relevance of each actor.

*User* – This actor is the one who uses the email system for daily activities. It is responsible for getting an email, sending the email, and managing the email account.

*Email server* – This robo-actor is the one who maintains the email system. It is responsible for validating a user’s password and updating databases with information (emails, passwords, phone numbers, names, etc.).

*Administrator* – This actor is the one who performs maintenance on the email system. It is responsible for performing server maintenance and emailing the current users to inform them about downtime.

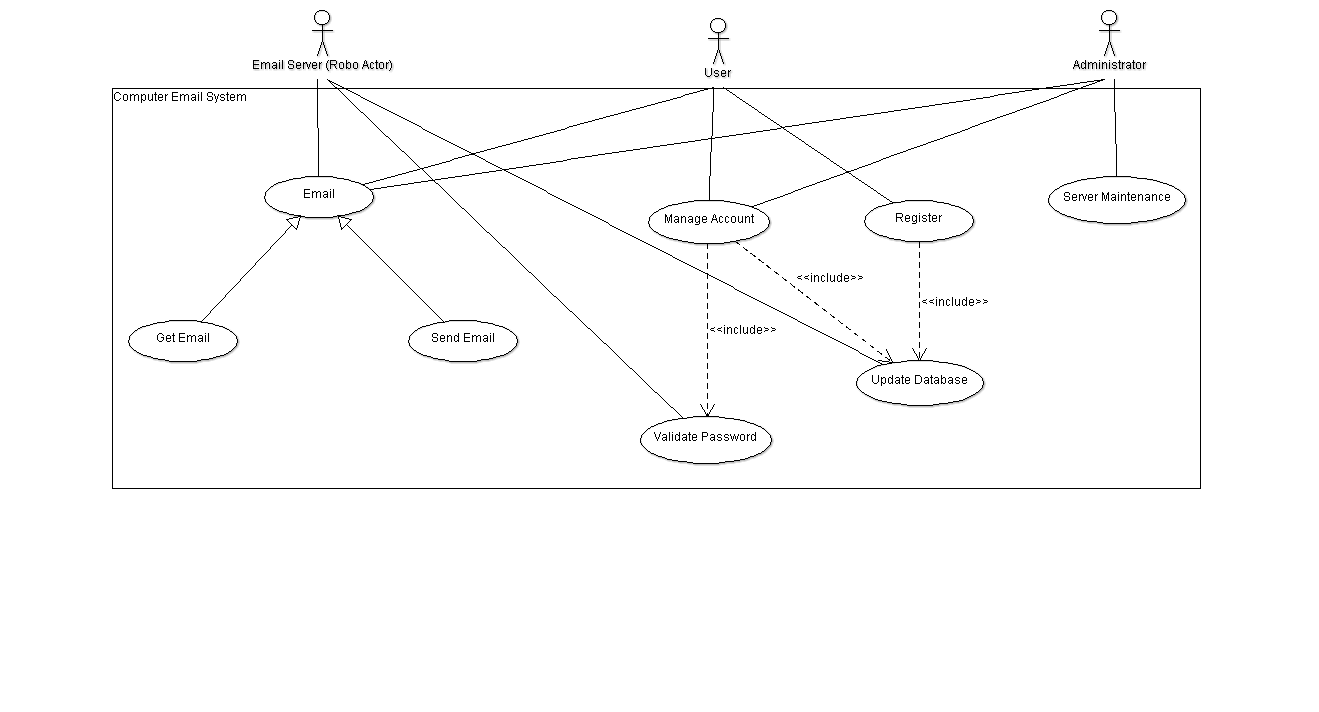
* 1. One use case is to get email. List four additional use cases at a comparable level of abstraction. Summarize the purpose of each use case with a sentence.

*Send email* – The composed email is sent from a sender to one (or many) receiver(s).

*Manage account* – the user manages the account information associated with its account, such as changing the password, the phone number / backup email (either one or the other is required), or the name on the account.

*Server maintenance* – The administrator shuts down the email service to perform required maintenance when errors arise.

*Registration* – A new user registers an email account with the system.

* 1. Prepare a use case diagram for a computer email system.
  2. Prepare a normal scenario for each use case. Remember that a scenario is an example, and need not exercise all functionality of the use case.

**Scenarios for get email use case:**

**Regular scenario:**

John Doe logs in.

System establishes secure communications.

System updates interface with all new emails since last login and displays options for managing account info and composing a new email.

John Doe logs out.

System establishes insecure communication.

System returns to log in screen.

**Server is down:**

John Doe logs in.

System notifies John Doe that the server is down and is currently undergoing maintenance.

System returns to log in screen.

**Incorrect password:**

John Doe logs in.

System notifies John Doe that the password is incorrect.

System returns to log in screen.

**Scenarios for send email use case:**

**Regular scenario:**

John Doe logs in.

System establishes secure communications.

System updates interface with all new emails since last login and displays options for managing account info and composing a new email.

John Doe selects the second option and composes an email for recipient(s).

System saves draft of email.

John Doe sends email to recipient(s).

System sends email to appropriate email address(es).

System displays confirmation to John Doe that email sent.

John Doe logs out.

System establishes insecure communication.

System returns to log in screen.

**Non-existent recipient:**

John Doe logs in.

System establishes secure communications.

System updates interface with all new emails since last login and displays options for managing account info and composing a new email.

John Doe selects the second option and composes an email for recipient(s).

System saves draft of email.

John Doe sends email to recipient(s).

System notifies John Doe that the email could not be sent to one or more non-existent recipients.

System returns error code for the problem.

John Doe logs out.

System establishes insecure communication.

System returns to log in screen.

**Email is canceled:**

John Doe logs in.

System establishes secure communications.

System updates interface with all new emails since last login and displays options for managing account info and composing a new email.

John Doe selects the second option and composes an email for recipient(s).

System saves draft of email.

John Doe cancels sending the email.

System notifies John Doe that the email was canceled and that a draft has been saved.

John Doe logs out.

System establishes insecure communication.

System returns to log in screen.

**Scenarios for manage account use case:**

**Regular scenario:**

John Doe logs in.

System establishes secure communications.

System updates interface with all new emails since last login and displays options for managing account info and composing a new email.

John Doe clicks button for managing account info.

System connects John Doe to account updating interface.

John Doe changes name on account and phone number associated with account.

System requests password for the account.

System validates password.

System updates database with new name and phone number for the account.

System displays “Account information updated” screen.

System returns to main interface.

John Doe logs out.

System establishes insecure communication.

System returns to log in screen.

**Incorrect password:**

John Doe logs in.

System establishes secure communications.

System updates interface with all new emails since last login and displays options for managing account info and composing a new email.

John Doe clicks button for managing account info.

System connects John Doe to account updating interface.

John Doe changes name on account and phone number associated with account.

System requests password for the account.

System validates password.

System notifies John Doe that the password was incorrect, and the information has not been updated.

System returns to main interface.

John Doe logs out.

System establishes insecure communication.

System returns to log in screen.

**Changes are canceled:**

John Doe logs in.

System establishes secure communications.

System updates interface with all new emails since last login and displays options for managing account info and composing a new email.

John Doe clicks button for managing account info.

System connects John Doe to account updating interface.

John Doe changes name on account and phone number associated with account.

System requests password for the account.

John Doe cancels and does not enter the password to the account.

System returns to main interface.

John Doe logs out.

System establishes insecure communication.

System returns to log in screen.

**Scenarios for server maintenance use case:**

**Regular scenario:**

Admin logs in.

System establishes secure communications.

Admin sends email to all existing users notifying them of server downtime for maintenance.

Admin disables system.

System asks if the admin is sure.

Admin confirms and the system is disabled.

Admin performs system maintenance.

Admin enables system.

Admin sends email to all existing users notifying them of what changes have been made to the server.

Admin logs out.

System establishes insecure communication.

System returns to log in screen.

**Incorrect password:**

Admin logs in.

System notifies admin that the password is incorrect.

System returns to log in screen.

**Canceled:**

Admin logs in.

System establishes secure communications.

Admin sends email to all existing users notifying them of server downtime for maintenance.

Admin disables system.

System asks if the Admin is sure.

Admin cancels disabling the system.

Admin logs out.

System establishes insecure communication.

System returns to log in screen.

**Server cannot be enabled:**

Admin logs in.

System establishes secure communications.

Admin sends email to all existing users notifying them of server downtime for maintenance.

Admin disables system.

System asks if the admin is sure.

Admin confirms and the system is disabled.

Admin performs system maintenance.

Admin enables system.

System notifies admin that there are errors and the system cannot be enabled.

Admin performs system maintenance.

Admin enables system.

Admin sends email to all existing users notifying them of what changes have been made to the server.

Admin logs out.

System establishes insecure communication.

System returns to log in screen.

**Nothing is fixed:**

Admin logs in.

System establishes secure communications.

Admin sends email to all existing users notifying them of server downtime for maintenance.

Admin disables system.

System asks if the admin is sure.

Admin confirms and the system is disabled.

Admin performs system maintenance.

Admin enables system.

Admin sends email to all existing users notifying them of what changes have been made to the server.

Admin receives email from John Doe that the system errors have not been fixed.

Admin responds with mass email to all existing users notifying them of further server downtime for maintenance.

Admin disables system.

System asks if the admin is sure.

Admin confirms and the system is disabled.

Admin enables system.

Admin sends email to all existing users notifying them of what changes have been made to the server.

Admin logs out.

System establishes insecure communication.

System returns to log in screen.

**Scenarios for registration use case:**

**Regular scenario:**

John Doe enters name, backup email or phone number, desired email, and password into the registration interface.

System checks the database for email address, finds no match.

System registers info into database, sends registration code to attached backup email or phone number.

John Doe checks backup email or phone number for registration code.

John Doe confirms the registration code.

System fully saves the information to the database, login is now capable.

**Email already taken:**

John Doe enters name, backup email or phone number, desired email, and password into the registration interface.

System checks the database for email address, finds match.

System notifies John Doe that email is already taken.

John Doe changes desired email and resubmits information into the registration interface.

System checks the database for email address, finds no match.

System registers info into database, sends registration code to attached backup email or phone number.

John Doe checks backup email or phone number for registration code.

John Doe confirms the registration code.

System fully saves the information to the database, login is now capable.

**Password not strong enough:**

John Doe enters name, backup email or phone number, desired email, and password into the registration interface.

System notifies John Doe that the password is not strong enough.

John Doe changes desired password and resubmits information into the registration interface.

System checks the database for email address, finds no match.

System registers info into database, sends registration code to attached backup email or phone number.

John Doe checks backup email or phone number for registration code.

John Doe confirms the registration code.

System fully saves the information to the database, login is now capable.

**Registration code not received:**

John Doe enters name, backup email or phone number, desired email, and password into the registration interface.

System checks the database for email address, finds no match.

System registers info into database, sends registration code to attached backup email or phone number.

John Doe checks backup email or phone number for registration code but cannot find it.

John Doe requests registration code be sent again.

System sends new registration code to attached backup email or phone number.

John Doe checks backup email or phone number for registration code.

John Doe confirms the registration code.

System fully saves the information to the database, login is now capable.

* 1. Prepare an exception scenario for each use case.

**Exceptions for getting mail use case:**

*Server is Down:* The user cannot receive emails due to a server malfunction and/or server maintenance.

*Incorrect Password:* The user cannot log in due to an incorrect password.

**Exceptions for sending mail use case:**

*Non-existent Recipient:*The email is sent to recipient(s), but one or more of the emails do not exist, generating an error code from the server.

*Canceled:* The email is canceled before being sent. A draft of the email is saved on the server.

*Server is Down:* The user cannot send emails due to a server malfunction and/or server maintenance.

*Incorrect Password:* The user cannot log in due to an incorrect password.

**Exceptions for managing account use case:**

*The password entered is incorrect:*The user provides the wrong password and the account info is not changed.

*Canceled:* The manage account option is canceled before changes are made.

**Exceptions for server maintenance use case:**

*Incorrect Password:* The admin cannot log in due to an incorrect password.

*Canceled:* The admin cancels on shutting the server down.

*Server Cannot be Enabled:* Options are changed incorrectly, and the server cannot be enabled.

*Errors are Not Fixed:* The server is enabled successfully, but the existing errors were not fixed in the process.

**Exceptions for registration use case:**

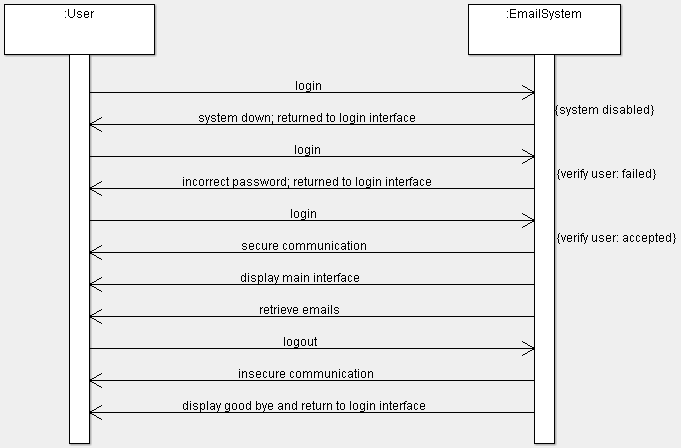
*Email is Already Taken:* The registration is unsuccessful due to the email already being registered.

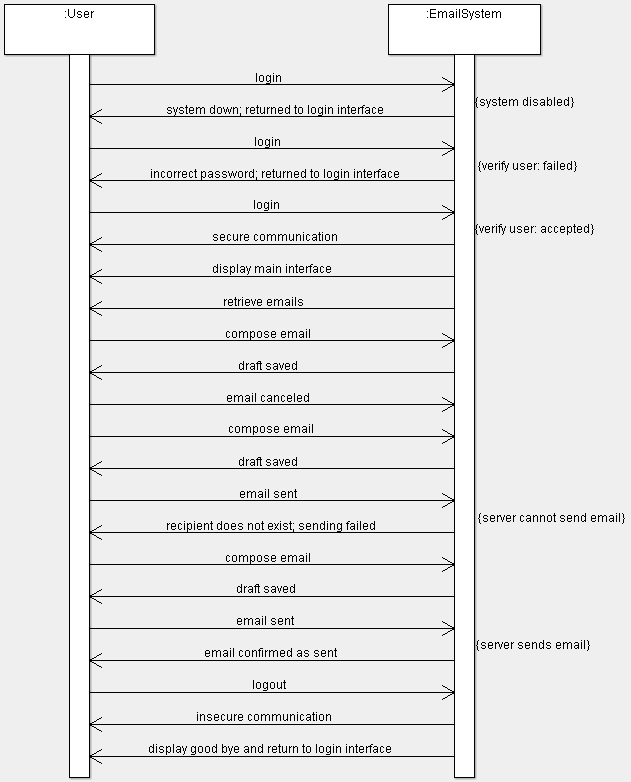
*Password is Not Strong Enough:* The password lacks the proper strength to be safe and must be revised for account safety.

*Phone Number / Backup Email Does Not Receive Authentication Code:* The attached phone number or backup email does not receive the authentication code. A new code must be sent or a different phone number or backup email must be attached until the code is successfully received.

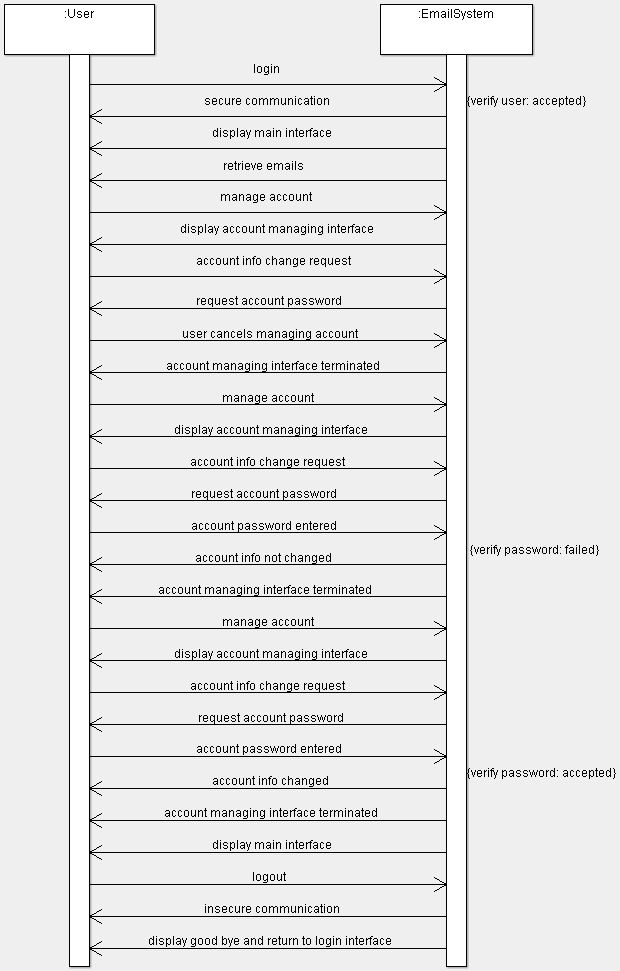
* 1. Prepare a sequence diagram corresponding to each scenario in (d).

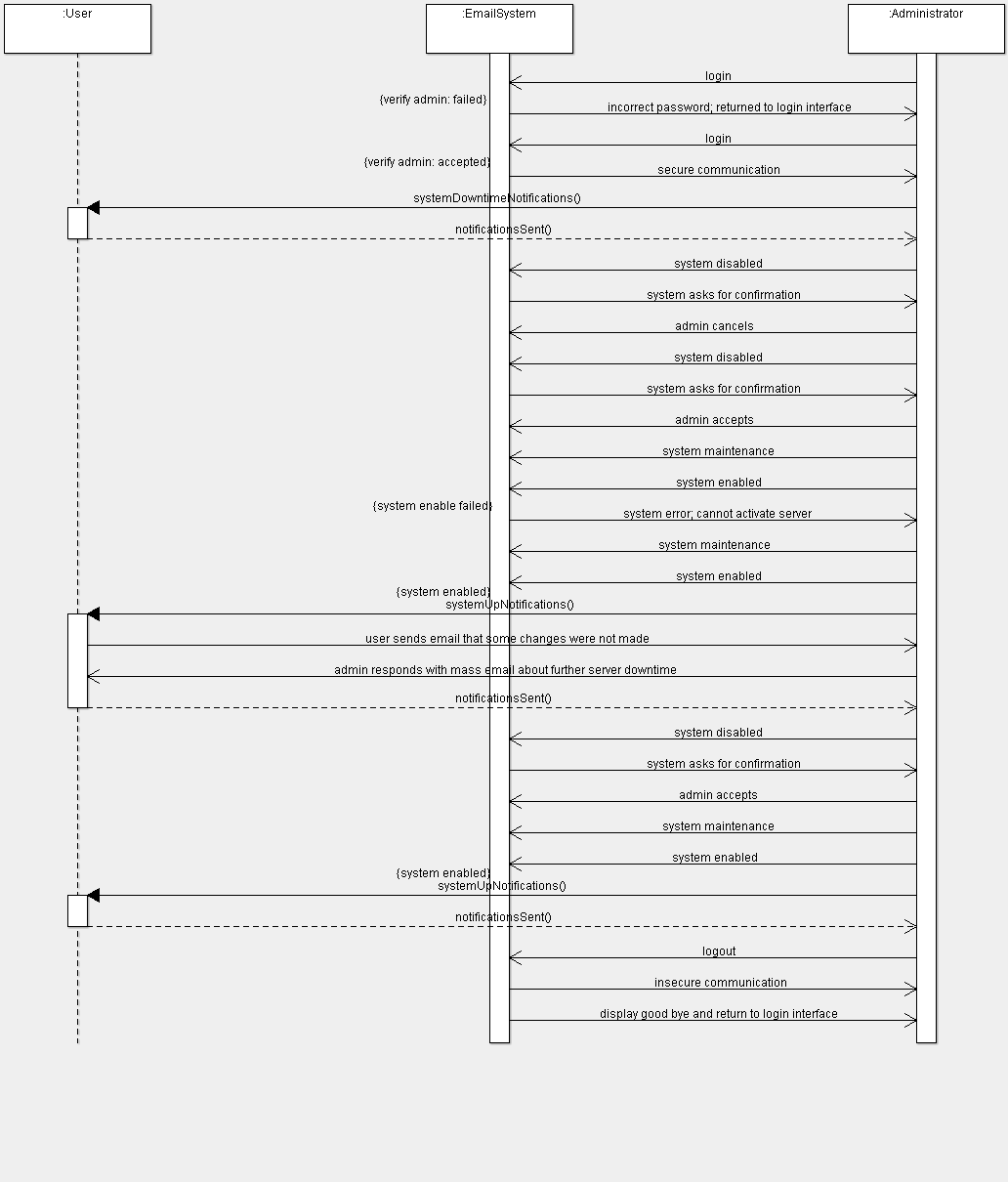
**Get email use case:**



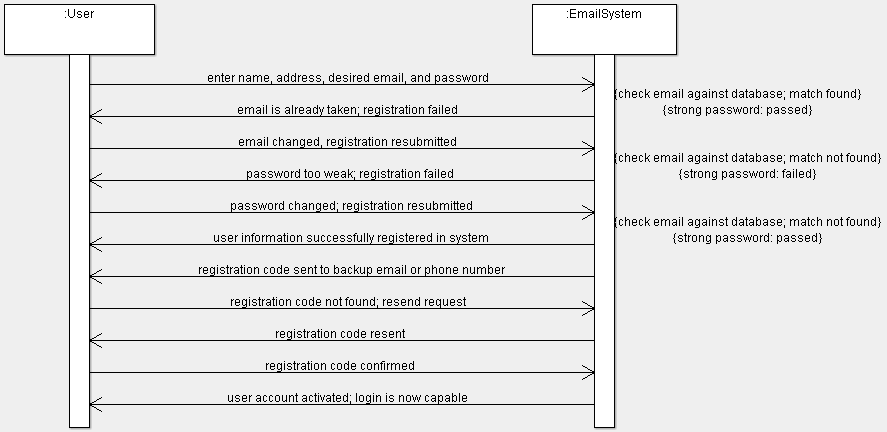
**Send email use case:**

**Manage account use case:**

****

**** **System administration use case:**

**Registration use case:**

****

1. Consider an online airline reservation system. You may want to check airline Web sites to give you ideas.
   1. List two actors. Explain the relevance of each actor.

*Passenger* – This actor is the one that signs up to the website to reserve flights. It is responsible for making flight reservations, making payments, checking the flight status, and checking for available flights.

*Bank* – This robo-actor is the one that is responsible for validating payments and card information.

* 1. One use case is to make a flight reservation. List four additional use cases at a comparable level of abstraction. Summarize the purpose of each use case with a sentence.

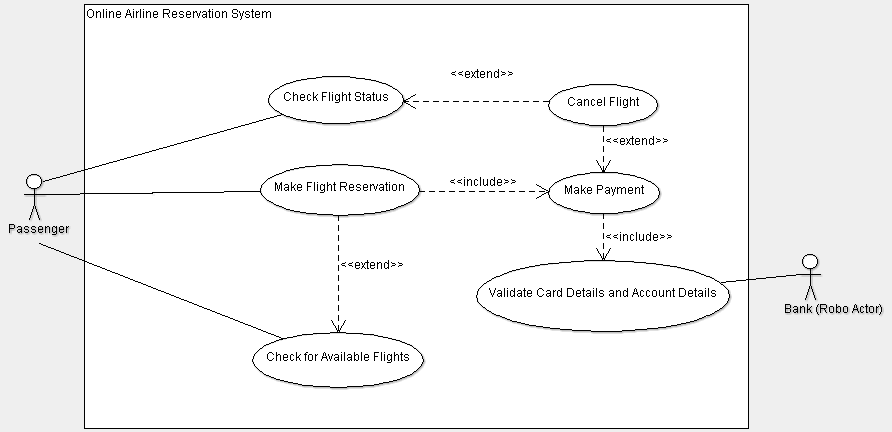
*Checking flight status* **–** The status of the flight is observed through the website, allowing the passenger to cancel or view when the plane will arrive.

*Making payments* **–** The passenger makes a payment for a flight reservation, which can be canceled and refunded before the flight date.

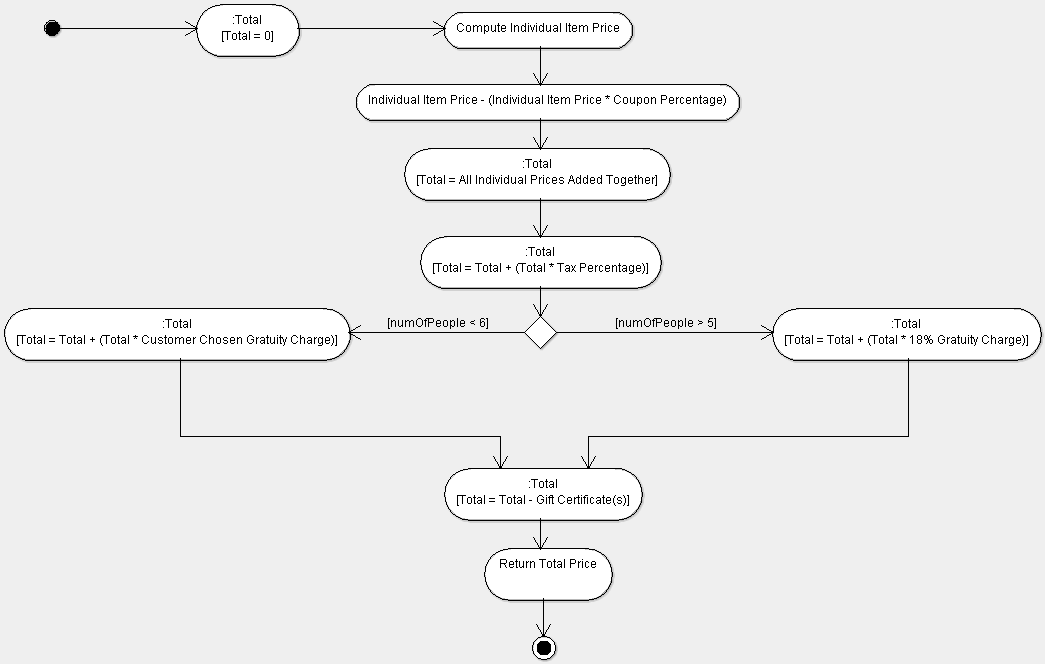
*Checking for available flights* **–** The passenger checks for available flights to find an appropriate one to pay for.

*Validate payments* **–** The bank validates the card information and available funds before the transaction is approved.

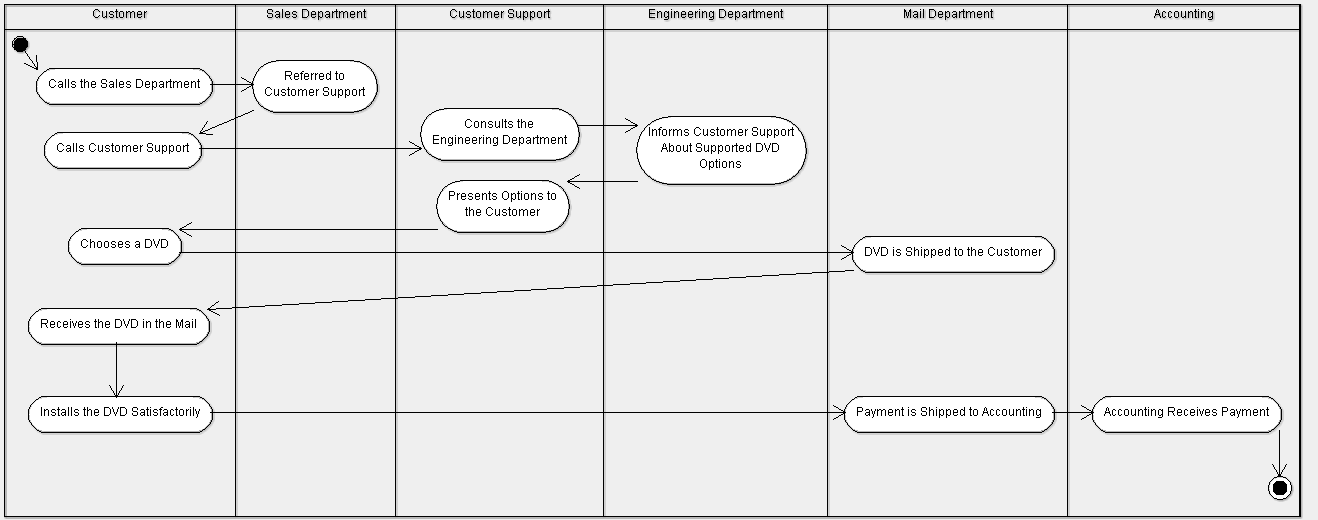
* 1. Prepare a use case diagram for an online airline reservation system.



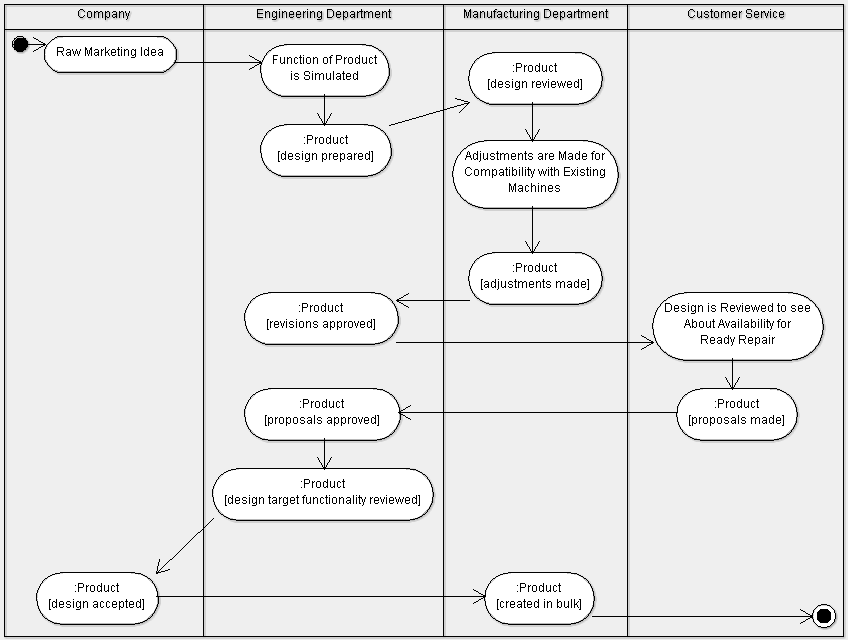
1. Prepare an activity diagram for computing a restaurant bill. There should be a charge for each delivered item. The total amount should be subject to tax and a service charge of 18% for groups of six or more. For smaller groups, there should be a blank entry for a gratuity according to the customer’s discretion. Any coupons or gift certificates submitted by the customer should be subtracted.



1. A customer decides to upgrade her PC and purchase a DVD player. She begins by calling the sales department of the PC vendor and they tell her to talk to customer support. She then calls customer support and they put her on hold while talking to engineering. Finally, customer support tells the customer about several supported DVD options. The customer chooses a DVD and it is shipped by the mail department. The customer receives the DVD, installs it satisfactorily, and then mails her payment to accounting. Construct an activity diagram for this process. Use swimlanes to show the various interactions.



1. A company is manufacturing a new product and must coordinate several departments. The product starts out as a raw marketing idea that goes to engineering. Engineering simulates the function of the product and prepares a design. Manufacturing reviews the design and adjusts it to conform to existing machinery. Engineering approves the revisions and customer service then looks at the design – a good design must enable ready repair. Engineering approves the customer service proposals and ensures that the resulting design still meets the target functionality. Construct an activity diagram for this process. Use swimlanes to show the various interactions. Show the changes in the state of the design as the activity diagram proceeds.



1. Develop and draw a UML class diagram to specify the metamodel for activity diagrams. Activity diagrams include concepts such as activities, precedence relations among activities, control flow, data (object flow), and various control objects such as the decision, merge, fork, join, final, and initial nodes.

