Design LinkedIn

Let's design LinkedIn.

We'll cover the following

LinkedIn is a social network for professionals. The main goal of the site is to enable its members to connect with people they know and trust professionally, as well as to find new opportunities to grow their careers.

A LinkedIn member’s profile page, which emphasizes their skills, employment history, and education, has professional network news feeds with customizable modules.

LinkedIn is very similar to Facebook in terms of its layout and design. These features are more specialized because they cater to professionals, but in general, if you know how to use Facebook or any other similar social network, LinkedIn is somewhat comparable.

System Requirements

We will focus on the following set of requirements while designing LinkedIn:

1. Each member should be able to add information about their basic profile, experiences, education, skills, and accomplishments.
2. Any user of our system should be able to search for other members or companies by their name.
3. Members should be able to send or accept connection requests from other members.
4. Any member will be able to request a recommendation from other members.
5. The system should be able to show basic stats about a profile, like the number of profile views, the total number of connections, and the total number of search appearances of the profile.
6. Members should be able to create new posts to share with their connections.
7. Members should be able to add comments to posts, as well as like or share a post or comment.
8. Any member should be able to send messages to other members.
9. The system should send a notification to a member whenever there is a new message, connection invitation or a comment on their post.
10. Members will be able to create a page for a Company and add job postings.
11. Members should be able to create groups and join any group they like.

12. Members should be able to follow other members or companies.

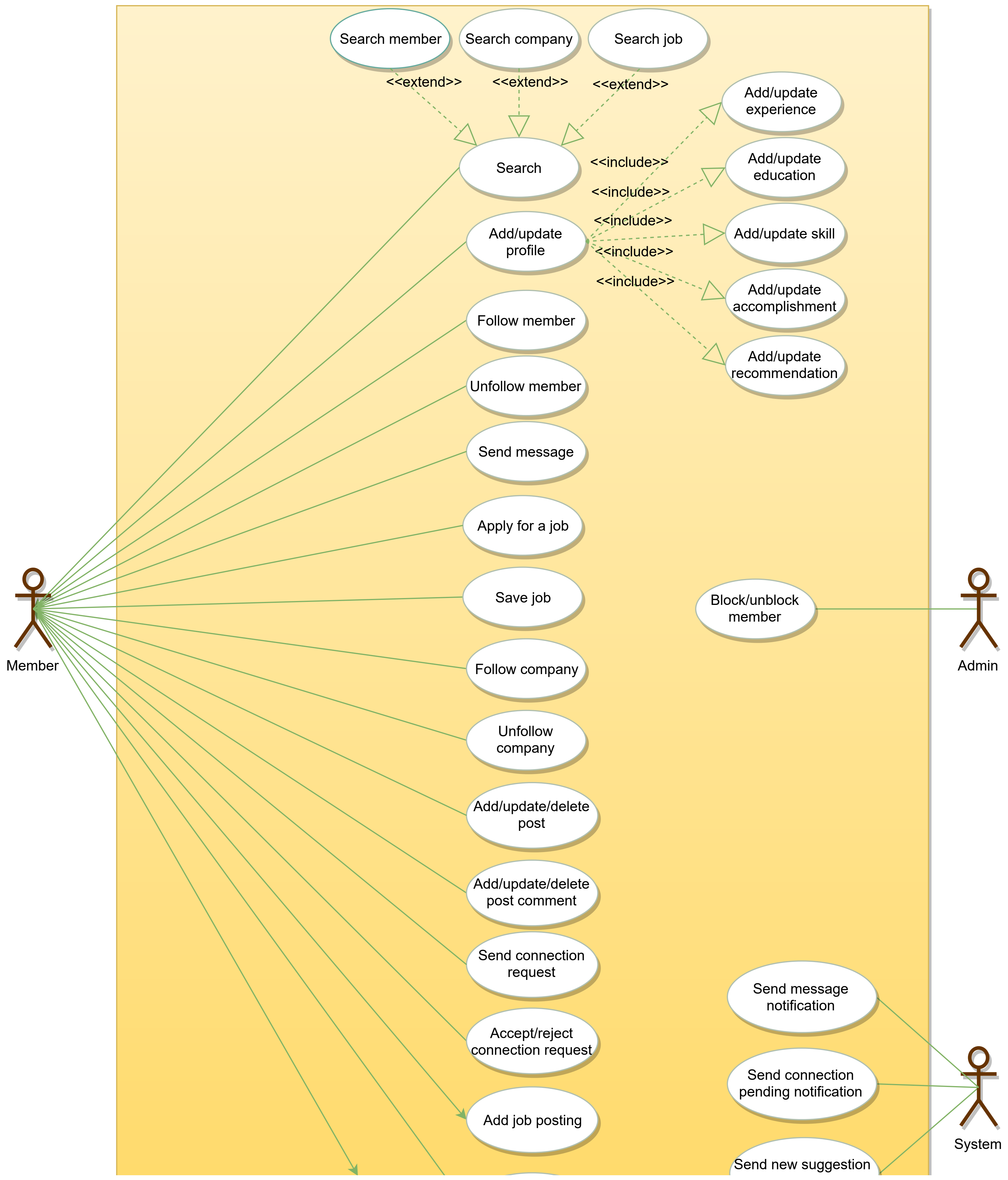
Use case diagram

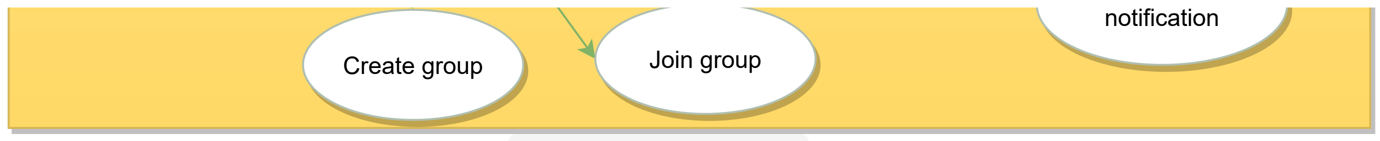
We have three main Actors in our system:

- **Member:** All members can search for other members, companies or jobs, as well as send requests for connection, create posts, etc.
- **Admin:** Mainly responsible for admin functions such as blocking and unblocking a member, etc.
- **System:** Mainly responsible for sending notifications for new messages, connections invites, etc.

Here are the top use cases of our system:

- **Add/update profile:** Any member should be able to create their profile to reflect their experiences, education, skills, and accomplishments.
- **Search:** Members can search other members, companies or jobs. Members can send a connection request to other members.
- **Follow or Unfollow member or company:** Any member can follow or unfollow any other member or a company.
- **Send message:** Any member can send a message to any of their connections.
- **Create post:** Any member can create a post to share with their connections, as well as like other posts or add comments to any post.
- **Send notifications:** The system will be able to send notifications for new messages, connection invites, etc.



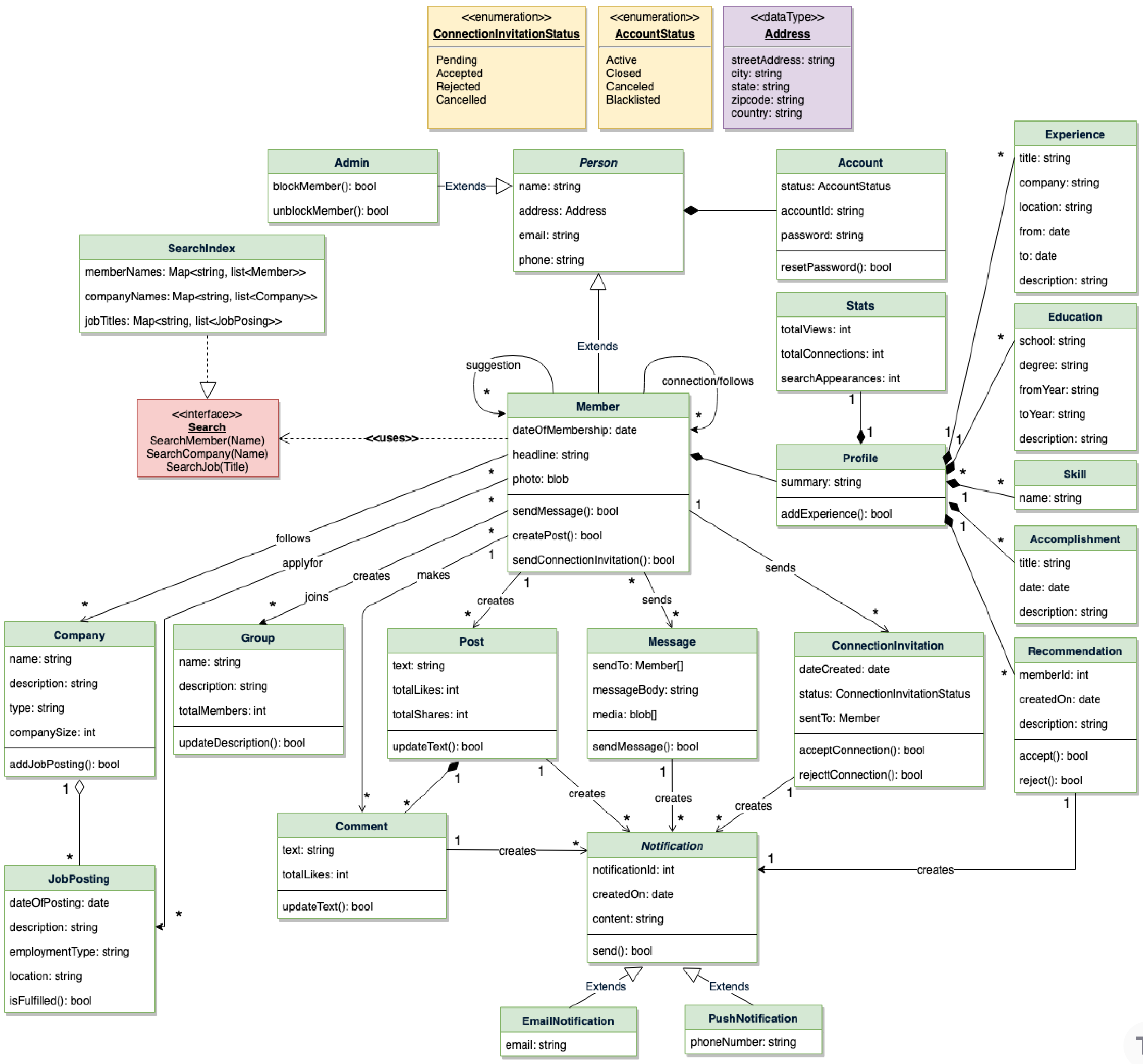


Use case diagram

Class diagram

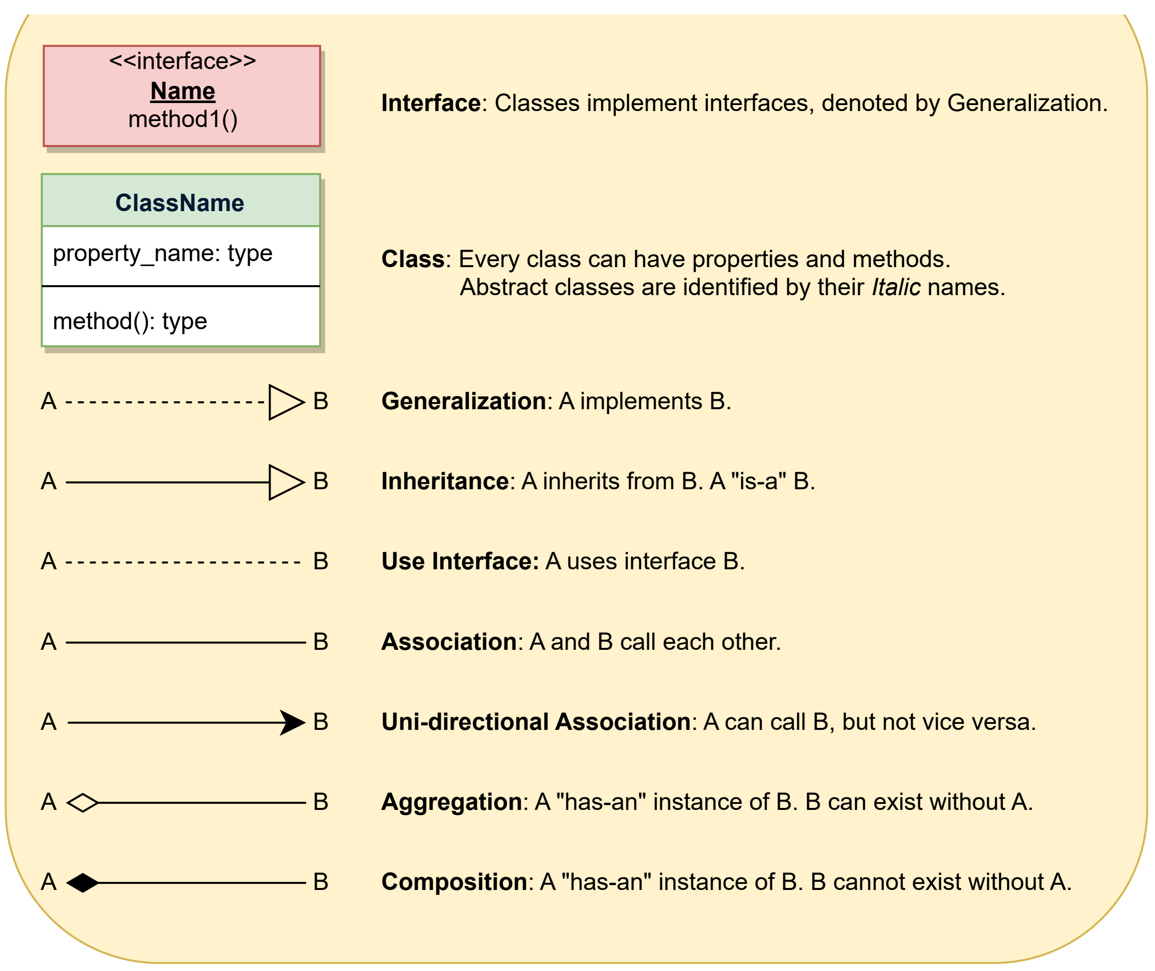
Here are the main classes of the LinkedIn system:

- **Member:** This will be the main component of our system. Each member will have a profile which includes their Experiences, Education, Skills, Accomplishments, and Recommendations. Members will be connected to other members and they can follow companies and members. Members will also have suggestions to make connections with other members.
- **Search:** Our system will support searching for other members and companies by their names, and jobs by their titles.
- **Message:** Members can send messages to other members with text and media.
- **Post:** Members can create posts containing text and media.
- **Comment:** Members can add comments to posts as well as like them.
- **Group:** Members can create and join groups.
- **Company:** Company will store all the information about a company's page.
- **JobPosting:** Companies can create a job posting. This class will handle all information about a job.
- **Notification:** Will take care of sending notifications to members.



Class diagram

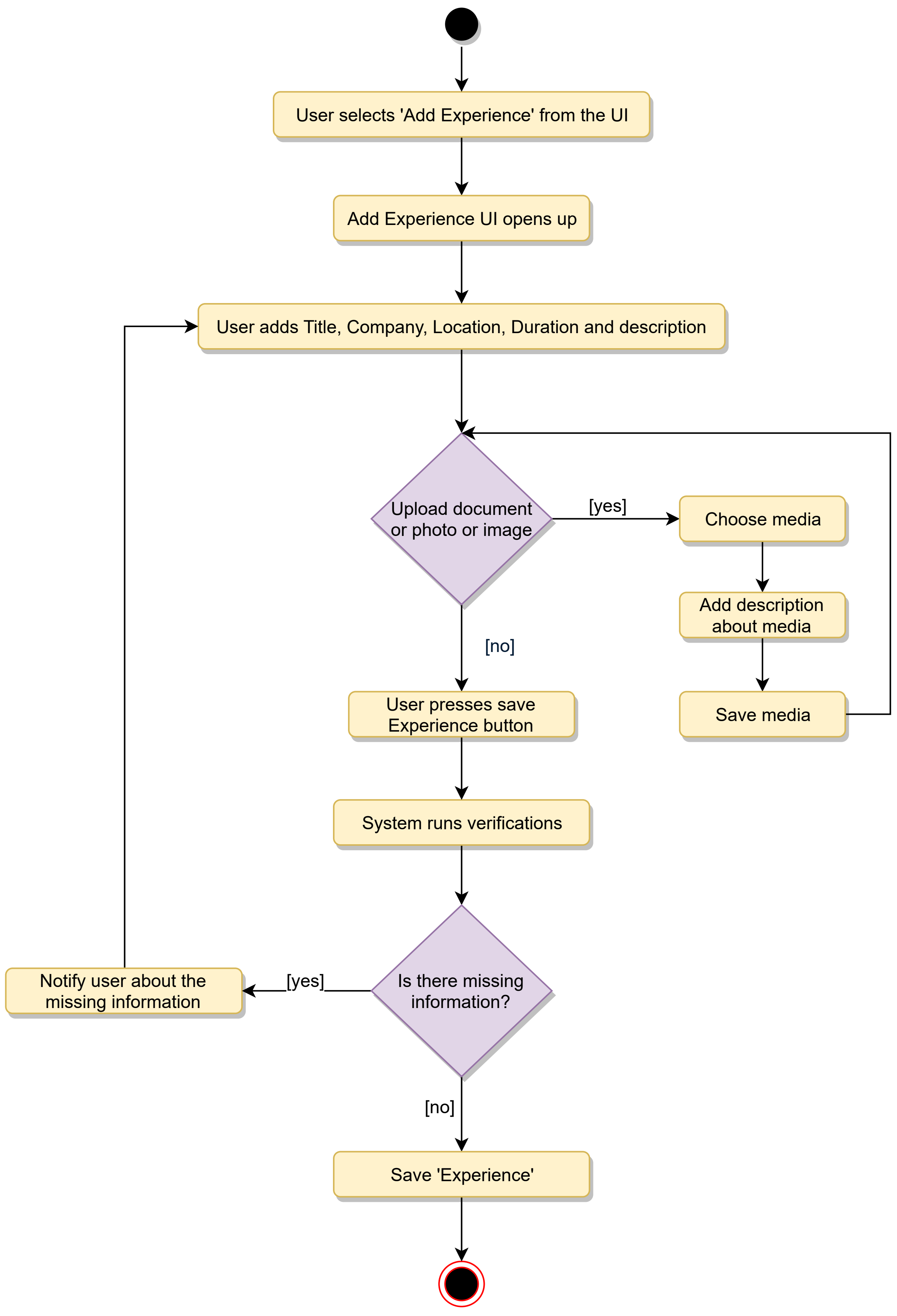
UML conventions



<<interface>>
Name
method1()
Interface: Classes implement interfaces, denoted by Generalization.
ClassName
property_name: type
method(): type
Class: Every class can have properties and methods. Abstract classes are identified by their *Italic* names.
A B
Generalization: A implements B.
A B
Inheritance: A inherits from B. A "is-a" B.
A B
Use Interface: A uses interface B.
A B
Association: A and B call each other.
A B
Uni-directional Association: A can call B, but not vice versa.
A B
Aggregation: A "has-an" instance of B. B can exist without A.
A B
Composition: A "has-an" instance of B. B cannot exist without A.

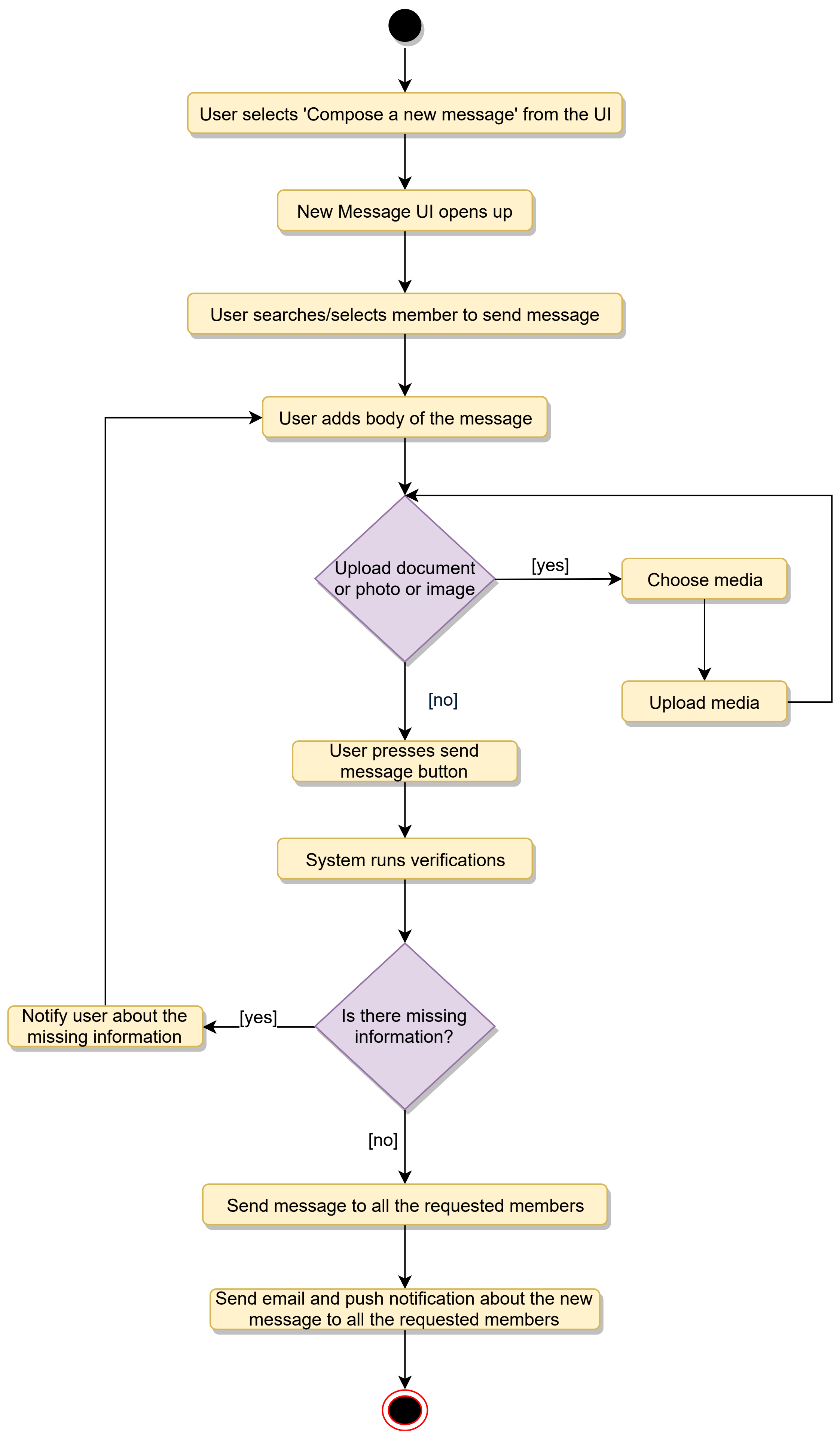
Activity diagrams

Add experience to profile: Any LinkedIn member can perform this activity. Here are the steps to add experience to a member profile:



User selects 'Add Experience' from the UI
Add Experience UI opens up
User adds Title, Company, Location, Duration and description
Upload document or photo or image
[yes]
Choose media
Add description about media
Save media
[no]
User presses save Experience button
System runs verifications
Is there missing information?
[yes]
Notify user about the missing information
[no]
Save 'Experience'

Send message: Any Member can perform this activity. After sending a message, the system needs to send a notification to all the requested members. Here are the steps for sending a message:



User selects 'Compose a new message' from the UI
New Message UI opens up
User searches/selects member to send message
User adds body of the message
Upload document or photo or image
[yes]
Choose media
Upload media
[no]
User presses send message button
System runs verifications
Is there missing information?
[yes]
Notify user about the missing information
[no]
Send message to all the requested members
Send email and push notification about the new message to all the requested members

Code

Here is the high-level definition for the classes described above:

Enums, data types, and constants: Here are the required enums, data types, and constants:

Java Python

```
class ConnectionInvitationStatus(Enum):
  PENDING, ACCEPTED, CONFIRMED, REJECTED, CANCELED = 1, 2, 3, 4, 5


class AccountStatus(Enum):
  ACTIVE, BLOCKED, BANNED, COMPROMISED, ARCHIVED, UNKNOWN = 1, 2, 3, 4, 5, 6


class Address:
  def __init__(self, street, city, state, zip_code, country):
    self.__street_address = street
    self.__city = city
    self.__state = state
```

```
        self.__zip_code = zip_code
        self.__country = country
```

Account, Person, Member, and Admin: These classes represent the different people that interact with our system:

Java Python

```
    # For simplicity, we are not defining getter and setter functions. The reader can
    # assume that all class attributes are private and accessed through their respective
    # public getter methods and modified only through their public methods function.


    class Account:
      def __init__(self, id, password, status=AccountStatus.Active):
        self.__id = id
        self.__password = password
        self.__status = status

      def reset_password(self):
        None


    class Person(ABC):
      def __init__(self, name, address, email, phone, account):
        self.__name = name
        self.__address = address
        self.__email = email
        self.__phone = phone
        self.__account = account


    class Member(Person):
      def __init__(self):
        self.__date_of_membership = datetime.date.today()
```

Profile, Experience, etc: A member's profile will have their job experiences, educations, skills, etc:

Java Python

```
    class Profile:
      def __init__(self, summary, experiences, educations, skills, accomplishments, recommendations):
        self.__summary = summary
        self.__experiences = experiences
        self.__educations = educations
        self.__skills = skills
        self.__accomplishments = accomplishments
        self.__recommendations = recommendations
        self.__stats = []

      def add_experience(self, experience):
        None

      def add_education(self, education):
        None

      def add_skill(self, skill):
        None

      def add_accomplishment(self, accomplishment):
        None

      def add_recommendation(self, recommendation):
        None


    class Experience:
```

Company and JobPosting: Companies can have multiple job postings:

```
class Company:
  def __init__(self, name, description, type, company_size):
    self.__name = name
    self.__description = description
    self.__type = type
    self.__company_size = company_size

    self.__active_job_postings = []


class JobPosting:
  def __init__(self, description, employment_type, location, is_fulfilled):
    self.__date_of_posting = datetime.date.today()
    self.__description = description
    self.__employment_type = employment_type
    self.__location = location
    self.__is_fulfilled = is_fulfilled
```

Group, Post, and Message: Members can create posts, send messages, and join groups:

```
class Group:
  def __init__(self, name, description):
    self.__name = name
    self.__description = description
    self.__total_members = 0
    self.__members = []

  def add_member(self, member):
    None

  def update_description(self, description):
    None


class Post:
  def __init__(self, text, owner):
    self.__text = text
    self.__total_likes = 0
    self.__total_shares = 0
    self.__owner = owner


class Message:
  def __init__(self, sent_to, message_body, media):
    self.__sent_to = sent_to
    self.__message_body = message_body
    self.__media = media
```

Search interface and SearchIndex: SearchIndex will implement the Search interface to facilitate searching for members, companies and job postings:

```
from abc import ABC, abstractmethod

class Search:
  def search_member(self, name):
    None

  def search_company(self, name):
    None

  def search_job(self, title):
    None


class SearchIndex(Search):
  def __init__(self):
    self.__member_names = {}
    self.__company_names = {}
    self.__job_titles = {}

  def add_member(self, member):
    if member.get_name() in self.__member_names:
      self.__member_names.get(member.get_name()).add(member)
    else:
      self.__member_names[member.get_name()] = member

  def add_company(self, company):
```

Back

Design a Car Rental System

Next

Design Cr

✓ Mark as Completed