19. Linkedin

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. 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

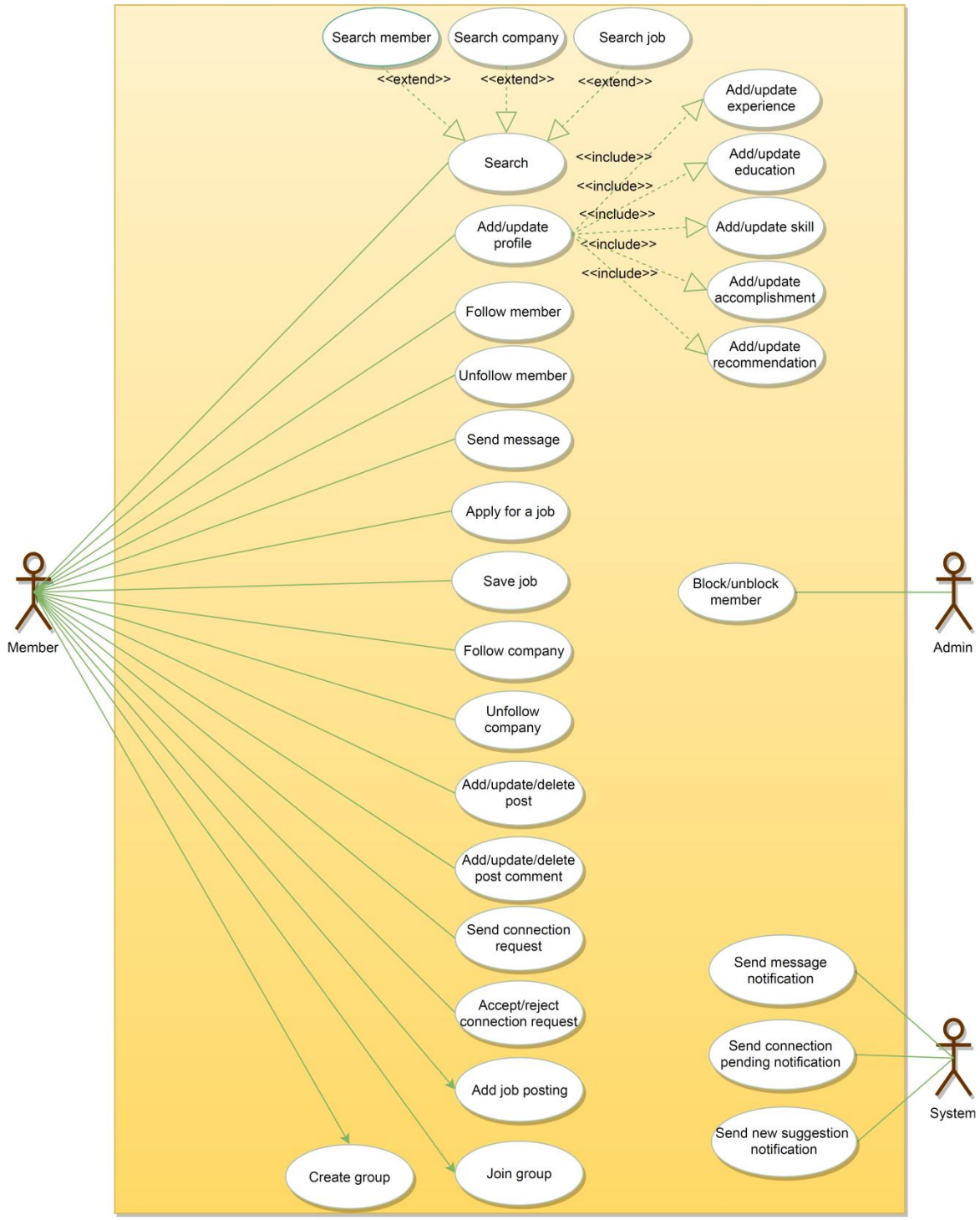
We have three main Actors in our system:

- **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.

Here is the use case diagram of LinkedIn:

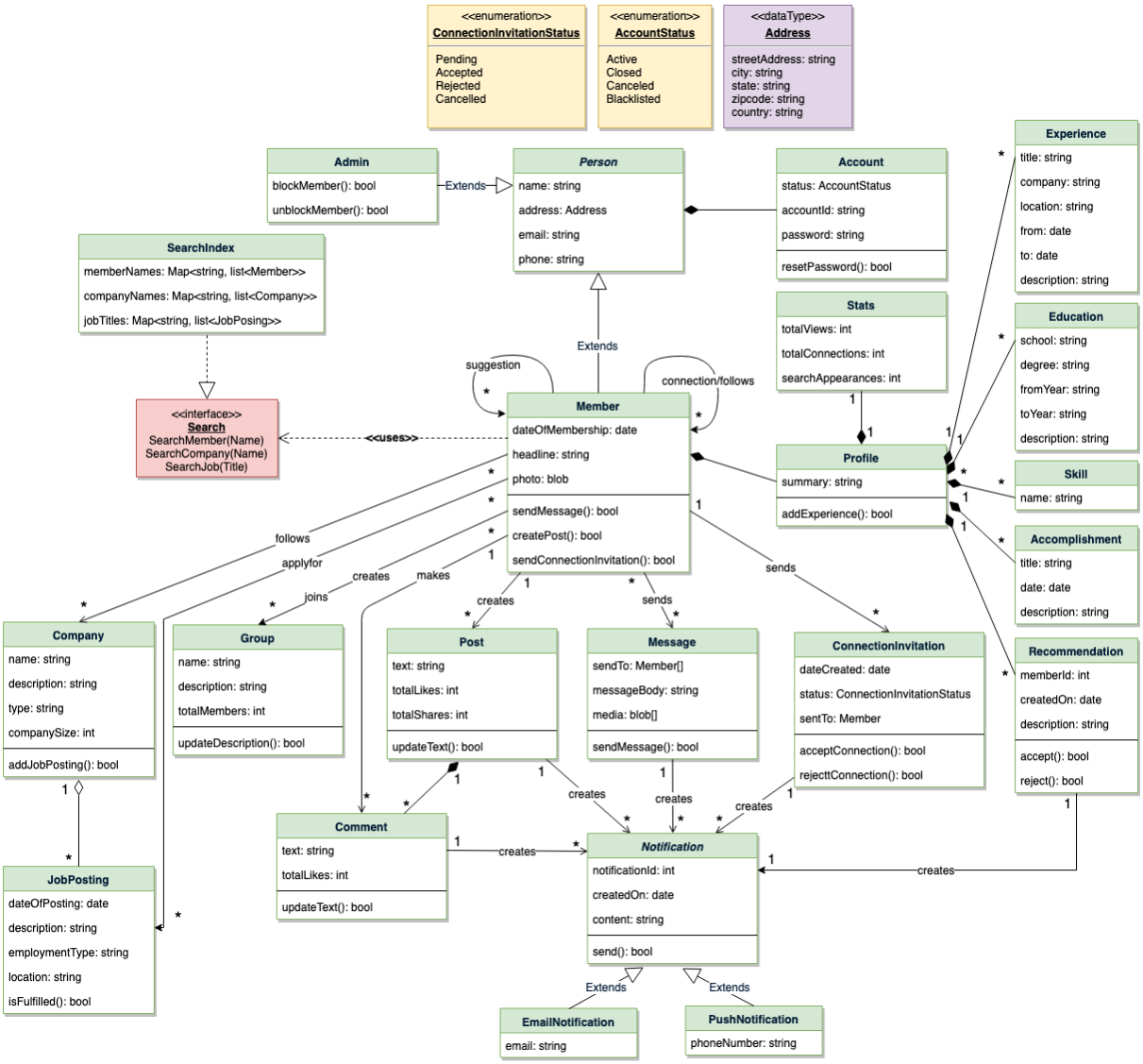


Use Case Diagram for LinkedIn

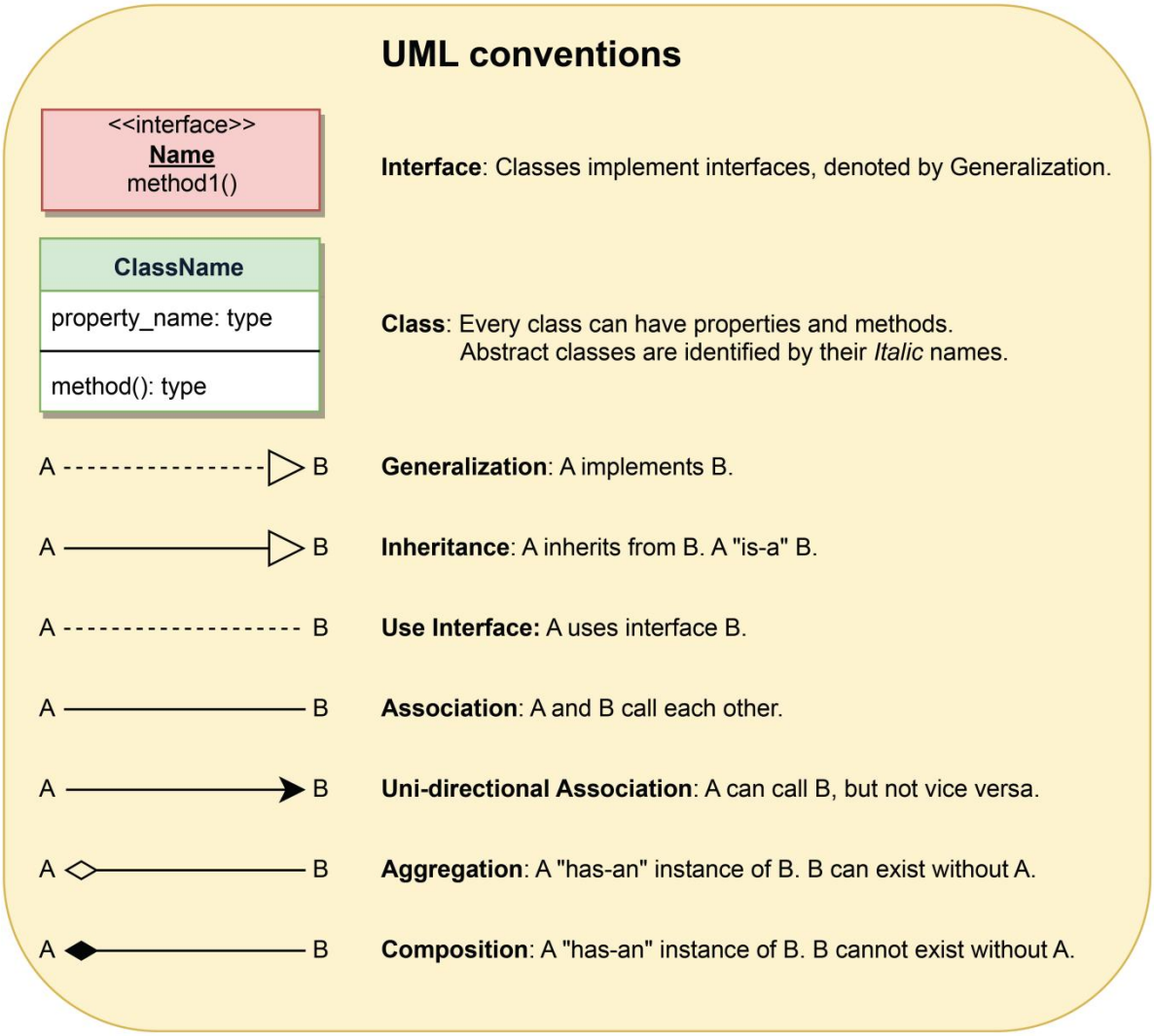
Class Diagram

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.



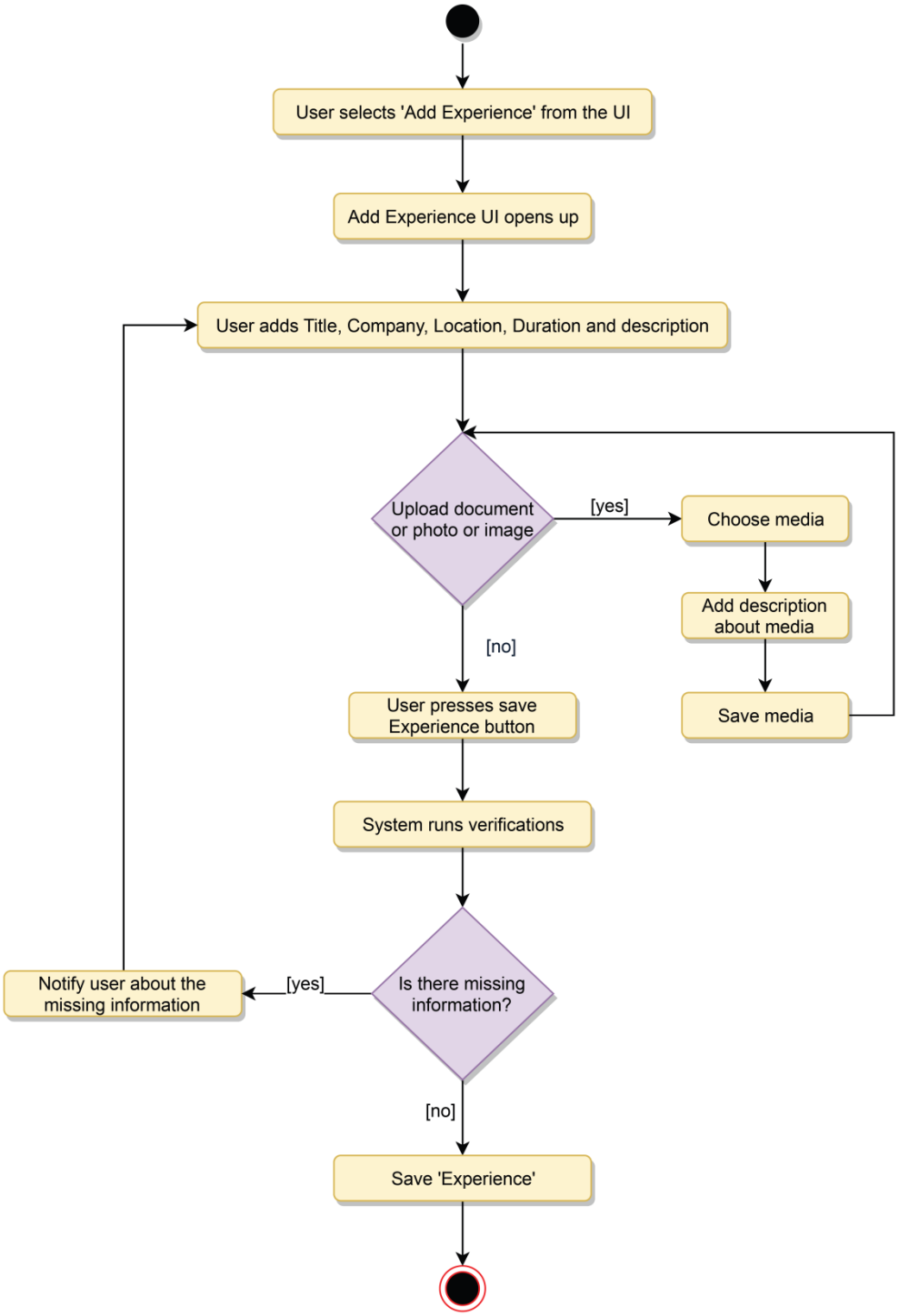
Class Diagram for LinkedIn



UML for LinkedIn

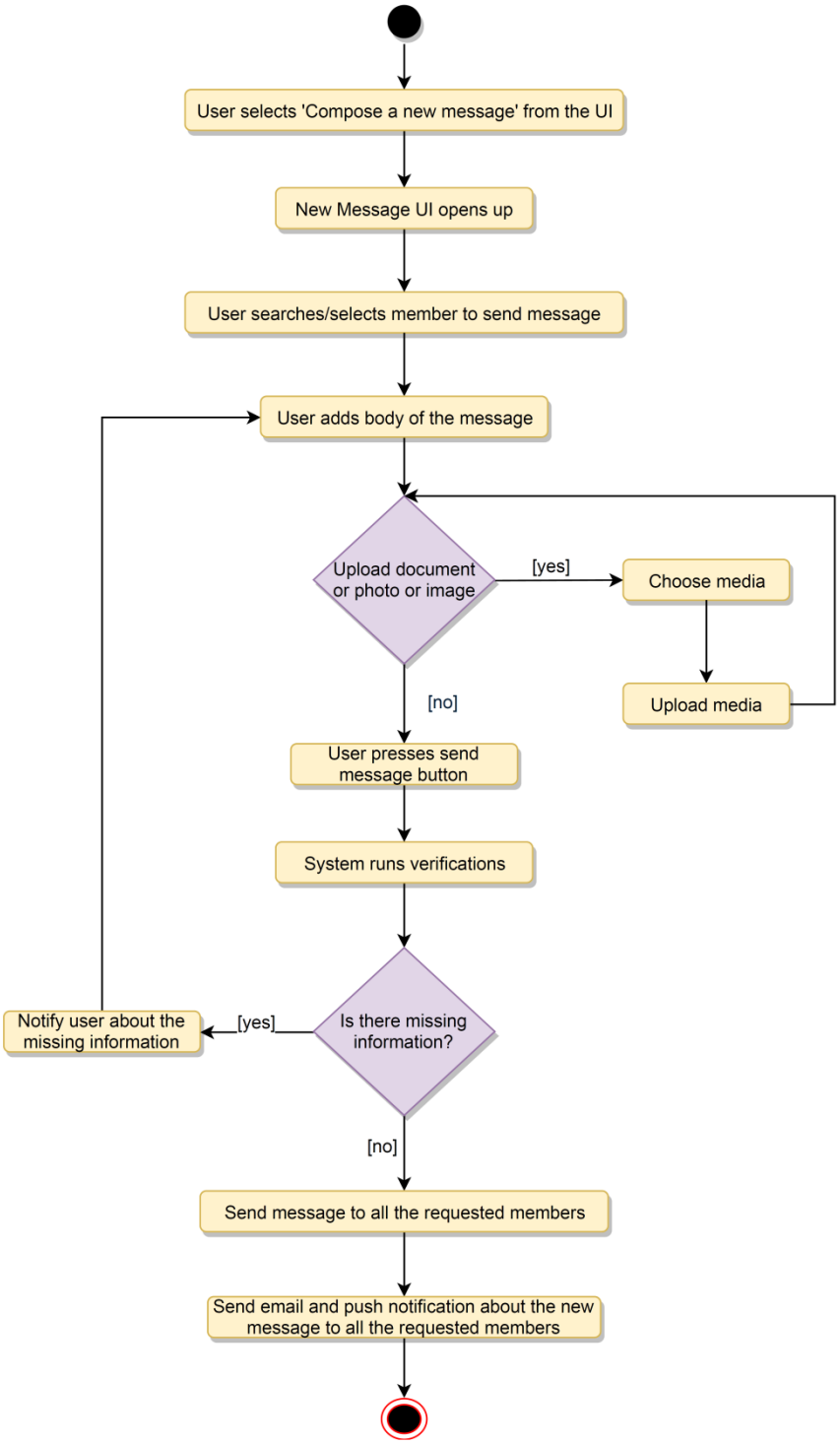
Activity Diagrams

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



Activity Diagram for LinkedIn Add Experience to Profile

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:



Activity Diagram for LinkedIn Send Message

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:

```
import java.util.*;
import java.time.LocalDate;


enum ConnectionInvitationStatus {
   PENDING, ACCEPTED, CONFIRMED, REJECTED, CANCELED;
}


enum AccountStatus {
   ACTIVE, BLOCKED, BANNED, COMPROMISED, ARCHIVED, UNKNOWN;
}


class Address {
   private String street;
   private String city;
   private String state;
   private String zipCode;
   private String country;


   public Address(String street, String city, String state, String zipCode, String country) {
      this.street = street;
      this.city = city;
      this.state = state;
```

```
        this.zipCode = zipCode;
        this.country = country;
    }
}

abstract class Person {
    private String name;
    private Address address;
    private String email;
    private String phone;
    private Account account;

    public Person(String name, Address address, String email, String phone, Account account) {
        this.name = name;
        this.address = address;
        this.email = email;
        this.phone = phone;
        this.account = account;
    }
}

class Account {
    private String id;
    private String password;
    private AccountStatus status;
```

```
    public Account(String id, String password, AccountStatus status) {
        this.id = id;
        this.password = password;
        this.status = status;
    }

    public void resetPassword() {
        // Reset password logic
    }
}

class Member extends Person {
    private LocalDate dateOfMembership;
    private String headline;
    private List<String> photos;
    private List<Member> memberSuggestions;
    private List<Member> memberFollows;
    private List<Member> memberConnections;
    private List<Company> companyFollows;
    private List<Group> groupFollows;
    private Profile profile;

    public Member(String name, Address address, String email, String phone, Account account) {
        super(name, address, email, phone, account);
        this.dateOfMembership = LocalDate.now();
        this.headline = "";
```

```
        this.photos = new ArrayList<>();
        this.memberSuggestions = new ArrayList<>();
        this.memberFollows = new ArrayList<>();
        this.memberConnections = new ArrayList<>();
        this.companyFollows = new ArrayList<>();
        this.groupFollows = new ArrayList<>();
        this.profile = new Profile();
    }
}

class Admin extends Person {
    public Admin(String name, Address address, String email, String phone, Account account) {
        super(name, address, email, phone, account);
    }

    public void blockUser(Member member) {}
    public void unblockUser(Member member) {}
}

class Profile {
    private String summary;
    private List<Experience> experiences;
    private List<String> educations;
    private List<String> skills;
    private List<String> accomplishments;
    private List<String> recommendations;
```

```
    public Profile() {
        this.experiences = new ArrayList<>();
        this.educations = new ArrayList<>();
        this.skills = new ArrayList<>();
        this.accomplishments = new ArrayList<>();
        this.recommendations = new ArrayList<>();
    }
}

class Experience {
    private String title;
    private String company;
    private String location;
    private LocalDate from;
    private LocalDate to;
    private String description;
}

class Company {
    private String name;
    private String description;
    private String type;
    private String companySize;
    private List<JobPosting> activeJobPostings;
}
```

```
class JobPosting {
    private LocalDate dateOfPosting;
    private String description;
    private String employmentType;
    private String location;
    private boolean isFulfilled;
}

class Group {
    private String name;
    private String description;
    private int totalMembers;
    private List<Member> members;
}

class Post {
    private String text;
    private int totalLikes;
    private int totalShares;
    private Member owner;
}

class Message {
    private Member sentTo;
    private String messageBody;
```

```
    private String media;
}

interface Search {
    Member searchMember(String name);
    Company searchCompany(String name);
    JobPosting searchJob(String title);
}

class SearchIndex implements Search {
    private Map<String, Member> memberNames;
    private Map<String, Company> companyNames;
    private Map<String, JobPosting> jobTitles;

    public SearchIndex() {
        this.memberNames = new HashMap<>();
        this.companyNames = new HashMap<>();
        this.jobTitles = new HashMap<>();
    }

    public void addMember(Member member) {
        this.memberNames.put(member.toString(), member); // Placeholder for actual getter
    }

    public void addCompany(Company company) {}
    public void addJobPosting(JobPosting jobPosting) {}
```

```
    public Member searchMember(String name) { return memberNames.get(name); }

    public Company searchCompany(String name) { return companyNames.get(name); }

    public JobPosting searchJob(String title) { return jobTitles.get(title); }

}
```