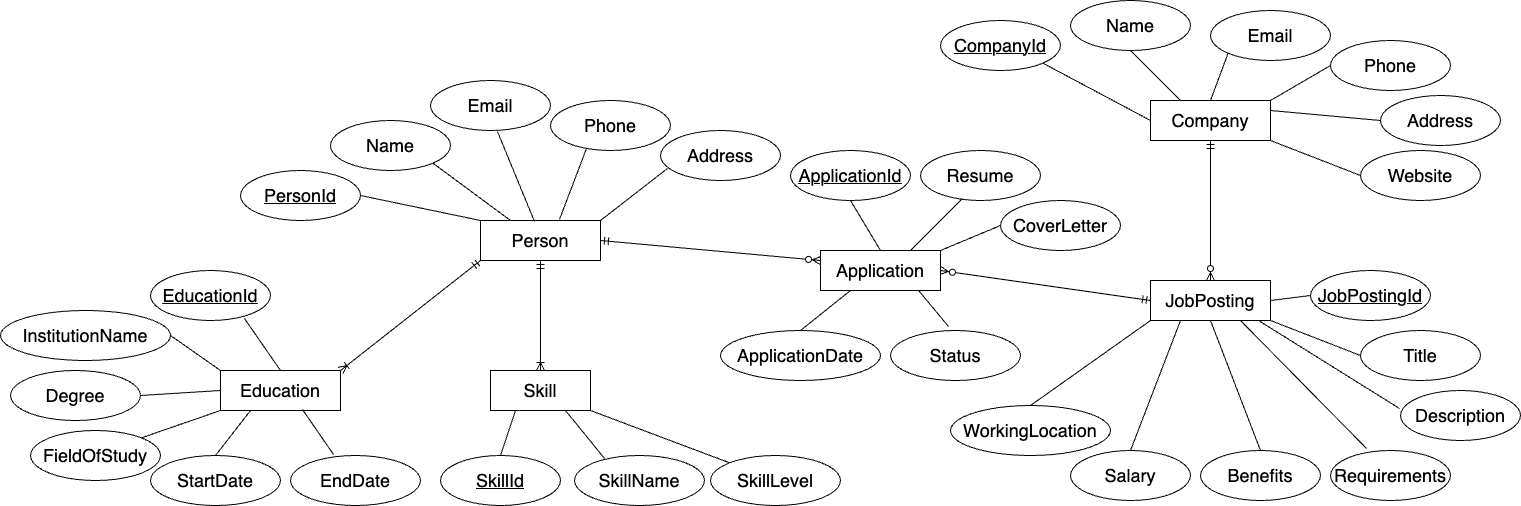
**TEST - COMPUTER PROJECT**

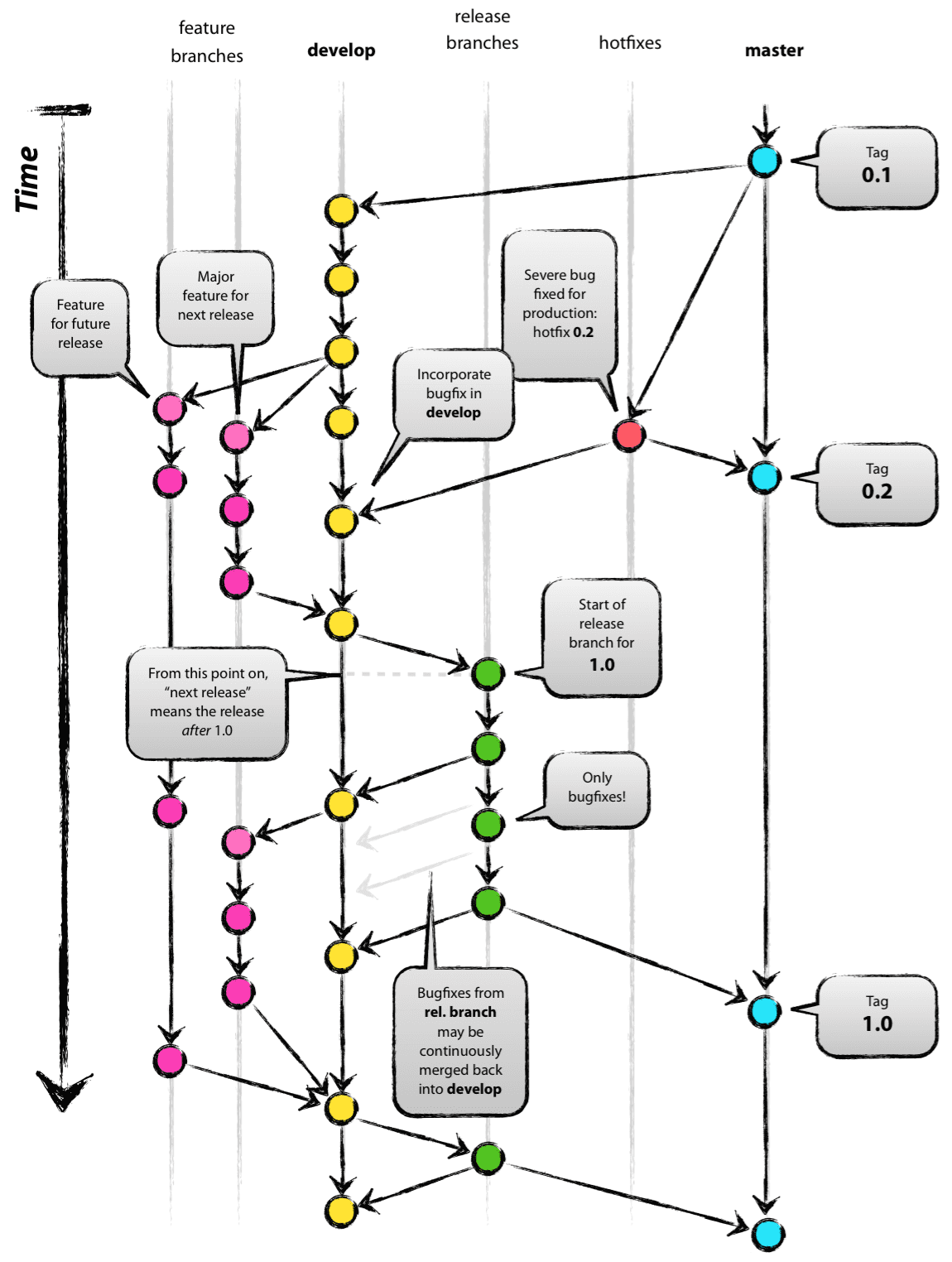
Full Name: Tran Hung Class: T2207M

**Question No.1:**

ER Model for system:  


**Question No.2:**

One popular git strategy is Git Flow or GitHub Flow. In this strategy we have:



1. ***Main Branches (2 main branches):***

* **Main/Master Branch**: This is the production-ready branch that always contains the stable version of the code. It is the source of truth for what is currently in production.
* **Develop Branch**: This branch contains the latest code with completed and tested features that are ready to be included in the next release. It is a working branch where integration of features and testing occurs.

1. ***Supporting Branches:***

* **Feature Branches**: These are used for developing new features. Each feature branch is typically created from the develop branch and is named descriptively for the feature being worked on (e.g., feature/new-login-system).
* **Release Branches**: When the develop branch has enough completed features for a new release, a release branch is created (e.g., release/v1.0). This branch is used for final testing, bug fixing, and preparing the release.
* **Hotfix Branches**: These branches are used for urgent bug fixes to the code in production. They are created from the main branch (e.g., hotfix/critical-bug-fix).

1. ***Other Branches:***

* **Bugfix Branches**: Similar to feature branches, these are used for fixing bugs. They may be created from either the develop branch or a release branch depending on when the bug was found (e.g., bugfix/fix-login-error).
* **Experimental Branches**: These branches are used for trying out new ideas or experimental features that are not yet ready to be part of the develop branch.

### ***\* Steps to Develop a New Module:***

1. ***Step 1: Preparation***

* **Requirements Gathering**: Understand the requirements and objectives of the new module.
* **Design**: Create a design for the module, including architectural diagrams and technical specifications.
* **Setup Environment**: Ensure that the development environment is properly set up and configured.

1. ***Step 2: Branching***

* **Create Feature Branch**: Create a new feature branch from the develop branch for the module (e.g., feature/new-module).

1. ***Step3: Development***

* **Coding**: Implement the module according to the design specifications.
* **Unit Testing**: Write and run unit tests to ensure individual components of the module work correctly.

1. ***Step 4: Integration***

* **Merge to Develop**: Once the module is complete and tested, merge the feature branch into the develop branch.
* **Integration Testing**: Perform integration testing to ensure the new module works well with the existing codebase.

1. ***Step 5: Pre-Release***

* **Create Release Branch**: If the module is part of a larger release, create a release branch from develop (e.g., release/v1.0).
* **Final Testing**: Conduct final testing, including regression testing and user acceptance testing.

1. ***Step 6: Release***

* **Merge to Main**: Once the release branch is stable and all issues are resolved, merge it into the main branch.
* **Tag Release**: Tag the main branch with the release version (e.g., v1.0).
* **Deploy**: Deploy the new module to the production environment.

1. ***Step 7: Post-Release***

* **Hotfixes**: If any critical issues are found post-release, create hotfix branches from the main branch to address them immediately.
* **Documentation**: Update any relevant documentation to reflect the changes made by the new module.

**Question No.3:  
  
*3.1. Types of Testing Strategies***

***3.1.1. Unit Testing***

* **Definition**: Testing individual components or functions of the software in isolation.
* **Purpose**: To ensure that each part of the code works as expected independently.
* **Tools**: JUnit, NUnit, pytest, etc.

***3.1.2. Integration Testing***

* **Definition**: Testing the interaction between integrated units or components.
* **Purpose**: To verify that different modules or services work together as intended.
* **Tools**: JUnit, NUnit, pytest, Postman, etc.

***3.1.3. System Testing***

* **Definition**: Testing the complete and integrated software system.
* **Purpose**: To evaluate the system’s compliance with specified requirements.
* **Tools**: Selenium, TestComplete, QTP, etc.

***3.1.4. Acceptance Testing***

* **Definition**: Testing conducted to determine if the system meets the acceptance criteria.
* **Purpose**: To ensure the system is ready for delivery to the end user.
* **Tools**: Cucumber, FitNesse, etc.

***3.1.5. Regression Testing***

* **Definition**: Re-running tests on the modified software to ensure that existing functionalities still work.
* **Purpose**: To confirm that changes or additions haven't broken existing functionality.
* **Tools**: Selenium, TestComplete, QTP, etc.

***3.1.6. Performance Testing***

* **Definition**: Testing to determine the performance of the system under load.
* **Purpose**: To ensure the system meets performance criteria such as speed, scalability, and stability.
* **Tools**: JMeter, LoadRunner, Gatling, etc.

***3.1.7. Security Testing***

* **Definition**: Testing to identify vulnerabilities, threats, and risks in the system.
* **Purpose**: To ensure that the system is secure from attacks.
* **Tools**: OWASP ZAP, Burp Suite, Nessus, etc.

***3.1.8. Usability Testing***

* **Definition**: Testing the system’s user interface and user experience.
* **Purpose**: To ensure the system is user-friendly and intuitive.
* **Tools**: UserTesting, Lookback, etc.

***3.1.9. Compatibility Testing***

* **Definition**: Testing the system's compatibility with different environments.
* **Purpose**: To ensure the system works well across various browsers, devices, operating systems, etc.
* **Tools**: BrowserStack, Sauce Labs, etc.

***3.1.10. Alpha and Beta Testing***

* **Definition**: Alpha testing is conducted by internal teams, while beta testing is conducted by a limited number of actual users.
* **Purpose**: To find bugs that were not discovered during earlier testing phases and to get feedback from real users.
* **Tools**: Custom tools for collecting feedback, bug tracking systems.

**3.2. Unit Testing vs Integration Test:**

|  | **Unit Testing** | **Integration Test** |
| --- | --- | --- |
| Purpose | Unit testing is used to verify that individual units of code (such as functions, methods, or classes) work as intended. It isolates each part of the program and shows that the individual parts are correct. It is typically automated and conducted by developers. | Integration testing is used to test the interaction between integrated units or components. The goal is to identify issues related to the interactions between different pieces of code, such as APIs, databases, and other services. It helps ensure that integrated modules work together as expected. |
| Tool | JUnit (Java), NUnit (.NET), pytest (Python), Jasmine (JavaScript). | JUnit, NUnit, pytest, Postman, SoapUI. |

**3.3. Test new module**

***3.3.1 Review Requirements and Design Documents:***

* Understand the purpose, functionalities, and expected behavior of the module.

***3.3.2. Unit Tests (if available):***

* If unit tests exist for the module, execute them to verify the core functionalities work as intended.

***3.3.3. API Testing (if applicable):***

* If the module exposes APIs, test them to ensure they function correctly and integrate seamlessly with other parts of the system.

***3.3.4. Integration Testing:***

* Create integration tests to simulate how the module interacts with other modules or the main application.
* Focus on data exchange, error handling, and overall system behavior.

***3.3.5. Functional Testing:***

* Develop test cases based on the module's functionalities and user stories.
* Test the module from a user's perspective to ensure it works as expected.

***3.3.6. Non-Functional Testing (if applicable):***

* Depending on the module's purpose, consider performance, usability, or security testing.

***3.3.7. Defect Reporting and Regression Testing:***

* Log any issues encountered during testing.
* Once fixes are implemented, re-run relevant tests to ensure the defects are resolved and no regressions are introduced.

***3.3.8. Final Review and Sign-off:***

* Review all test results and ensure the module meets quality standards.
* Provide final sign-off before the module is released to production.