**Nguyễn Minh Nhựt – 518H0545**

**Exercise 1:**

Do you know that it costs a lot of money to get a “Certiﬁed Java Programmer” certiﬁcate? It could cost you thousands of euros. Let’s imagine we will develop a browser-based training system to help people prepare for such a certiﬁcation exam.

* A user can request a quiz for the system.
* The system picks a set of questions from its database, and compose them together to make a quiz.
* It rates the user’s answers, and gives hints if the user requests it.

In addition to users, we also have tutors who provide questions and hints. And also examinators who must certify questions to make sure they are not too trivial, and that they are essential.

Quests:

* *Make a use case diagram to model this system*.
* Work out some of your use cases.
* Since we don’t have real stake holders here, you are free to ﬁll in details you think is essential for this example.

Java Certification Online

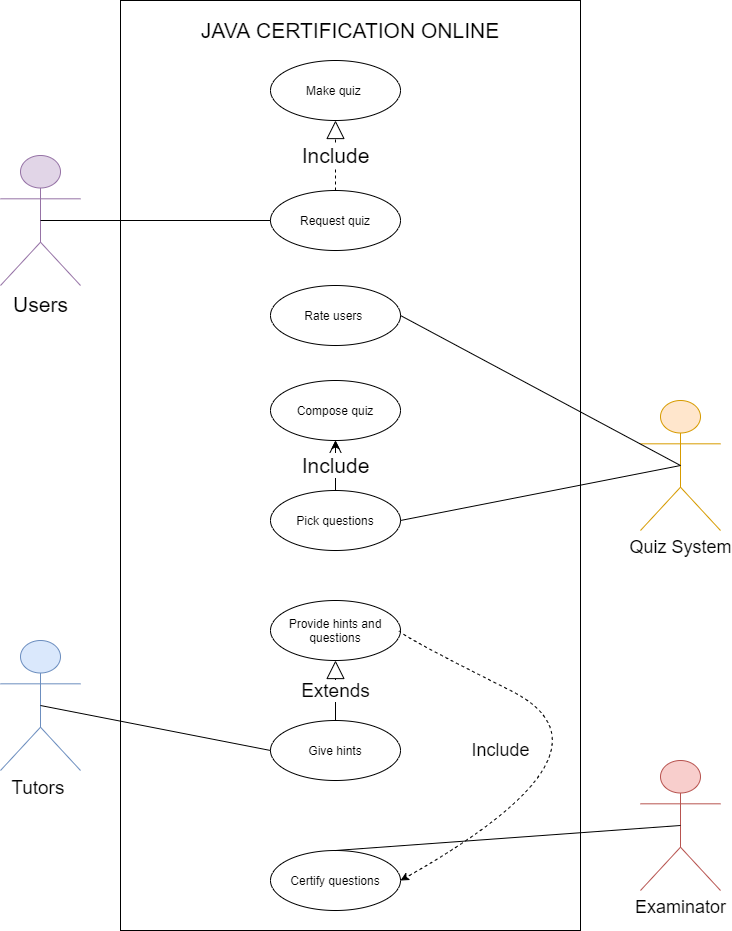
List of actors:

* User: can be programmers, students, etc. Anyone wants to be certified with Java certification
* Quiz System
* Tutor
* Examinator

List of use cases:

|  |  |
| --- | --- |
| **Use case** | **Description** |
| Request quiz | User can request quiz from system |
| Make quiz | User can make quiz after request for it |
| Pick set of questions | System can pick a set of random questions when hears a quiz request from user |
| Compose quiz | System will use the set of questions to compose a final quiz |
| Rate user | After finishing the quiz, system will be able to mark and then rate the user |
| Gives hints | When there is a difficult question and the user cannot solve it easily, the system can give out hints |
| Provide questions | Tutors are people who make those questions, come up the idea, the heart of each question |
| Provide hints | Tutors will be responsible for making hints for each question they have made lately |
| Certify questions | Since nothing is perfect, examinators will check up and certify the questions from tutors |

Use case diagram:



**Exercise 2:**

Suppose we want to develop software for an alarm clock.

* The clock shows the time of day.
* Using buttons, the user can set the hours and minutes ﬁelds individually, and choose between 12 and 24-hour display.
* It is possible to set one or two alarms.
* When an alarm ﬁres, it will sound some noise -> clock.
* The user can turn it off, or choose to ’snooze’.
* If the user does not respond at all, the alarm will turn off itself **after 2 minutes**->clock. ’Snoozing’ means to turn off the sound, but the alarm will ﬁre again after some minutes of delay. This “snoozing time” is pre-adjustable.
* Identify the top-level functional requirement for the clock, and model it with a use case diagram

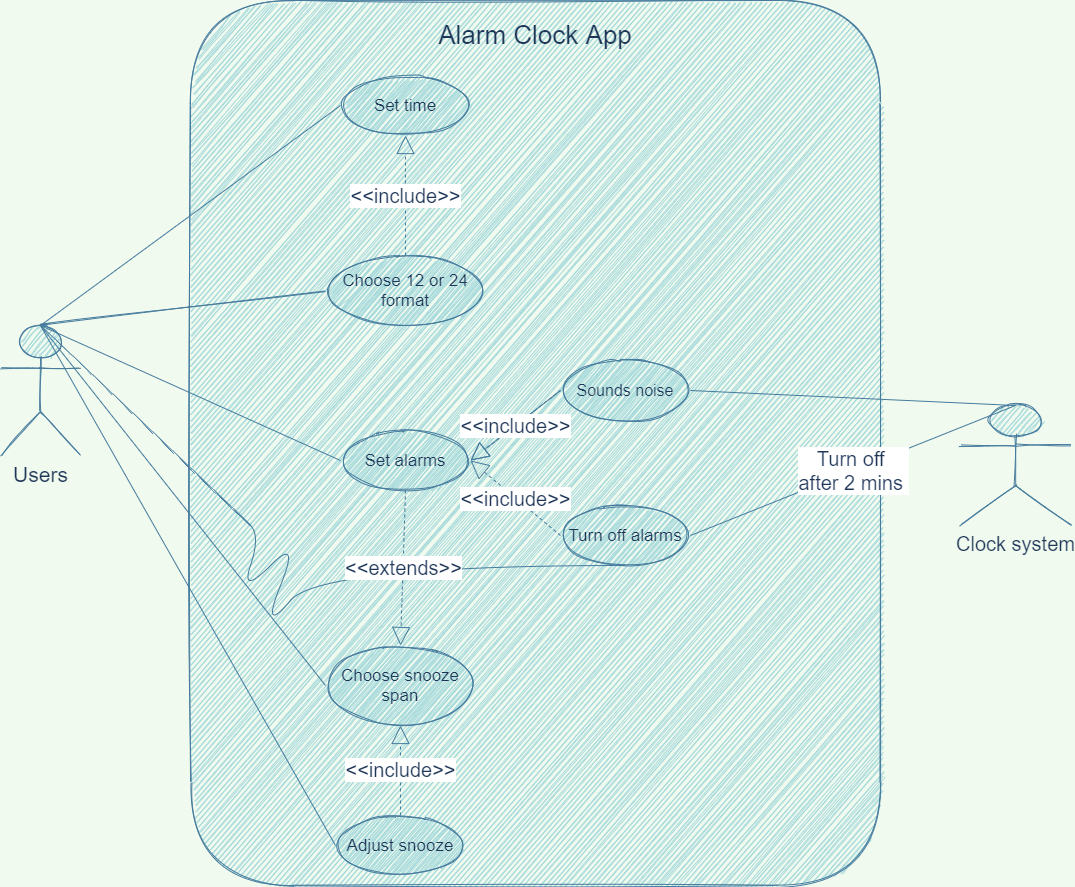
**Functional requirements:**

**Process – oriented**

* Using buttons, the user can set the hours and minutes ﬁelds individually (**Set time**), and choose between 12 and 24-hour display (**Choose 12 or 24 format**).
* It is possible to set one or two alarms (**Set alarms**).
* When an alarm ﬁres, it will sound some noise -> clock (**Sounds noise**).
* The user can turn it off (**Turn off alarm**), or choose to ’snooze’ (**Choose snooze span**).
* If the user does not respond at all, the alarm will turn off itself **after 2 minutes**->clock. ’Snoozing’ means to turn off the sound, but the alarm will ﬁre again after some minutes of delay. This “snoozing time” is pre-adjustable.

**Information – oriented**

* Alarm clock app will be able to store time display format
* Alarm clock app will be able to store snooze span if there is any
* Alarm clock app will be able to store alarm noise format

****

**Exercise 3:**

Using your knowledge of how an ATM is used, **develop a set of use cases** that could serve as a basis for understanding the requirements for an ATM system.

List of actors:

* Operator
* User(Customer)
* Banker

List of use cases for ATM system:

|  |  |
| --- | --- |
| **Use case** | **Description** |
| System startup | The system is started up when the operator turns the switch on |
| System shutdown | The system is shutdown when the operator turns the switch off |
| Session | Banking session is started when a customer inputs an ATM card into the card reader slot of the machine |
| Cash Withdrawal Transaction | A cash withdrawal transaction is started from within a session when the customer choose withdrawal transaction |
| Deposit Transaction | A deposit transaction is started from within a session when the customer chooses deposit transaction from the list |
| Transfer Transaction | A transfer transaction is started from within a session when the customer chooses transfer transaction from the list |
| Balance Inquiry Transaction | A balance inquiry transaction is started from within a session when the customer chooses balance inquiry transaction from the list |
| Invalid PIN | An invalid PIN error will be raised from within a transaction when banker reports to the system that the inserted PIN is invalid then disapprove the transaction |
| Failed Transaction | A failed transaction extension is started from a session when a transaction use case fails to perform correctly |