# **Lineage 2 Quest Master**

*“In the realm of* ***Aden****, brave adventurers rise through the ranks by capturing quests and earning powerful items. Each quest holds a reward and a challenge tailored to one of three epic classes:* ***Mystic Muse****,* ***Ghost Hunter****, and* ***Doombringer****. As players grow, they gain experience and level up — building their path toward glory.” - Lineage 2*

## Entities

You are provided with empty entity Java classes and enums. You must complete them following the structure below. Field names, types, and constraints must match. **Do not rename, remove, or add any extra fields.**

### Player

* id – UUID, primary key
* username – String, column not-null, column unique
* password – String, column not-null
* nickname – String, column not-null
* role – Enum (PlayerRole), nullable
* playerClass – Enum (PlayerClass), nullable
* xp – double
* createdOn – LocalDateTime, column not-null
* updatedOn – LocalDateTime, column not-null

### Quest

* id – UUID, primary key
* title – String, column not-null, column unique
* description – String, column not-null
* xp – double, column not-null
* bannerUrl – String, column not-null, column unique
* eligibleClass – Enum (PlayerClass), column not-null
* rewardItem – Item, column not-null– an item could be associated with multiple quests
* capturer – Player, nullable – a player could capture multiple quests
* createdOn – LocalDateTime, column not-null
* updatedOn – LocalDateTime, column not-null
* createdBy – String, column not-null
* updatedBy – String, column not-null

### Item

* id – UUID, primary key
* name – String, column not-null, column unique
* type – Enum (ItemType), column not-null
* xpBonusMultiplier – double
* iconUrl – column not-null, column unique
* createdOn – LocalDateTime, column not-null
* updatedOn – LocalDateTime, column not-null
* createdBy – String, column not-null
* updatedBy – String, column not-null

### PlayerClass

The class of a player. Each enum value **must have a display name**:

* MYSTIC\_MUSE("Mystic Muse")
* GHOST\_HUNTER("Ghost Hunter")
* DOOMBRINGER("Doombringer")

### PlayerRole

The role of a player/user.

* ADVENTURER
* QUEST\_MASTER

### ItemType

The type of an item. Each enum value **must have a display name**:

* WEAPON("Weapon")
* ARMOR("Armor")
* OTHER("Other")

## Page Requirements

You must implement the following **pages and endpoints**:

### Index Page / (Public landing page - already implemented for you)

* View: **index.html**

### Register Page /register

* View: **register.html**
* The page has a form for user registration, that must obey the following validations: username, password, nickname (1–12 characters)
  + Display invalid input red label when the submitted value is incorrect
  + Display the constraint yellow validation message always
* On success, redirect to Login Page

### Login Page /login

* View: **login.html**
* Activates session and redirects to Home Page after login

### Role Selection Page /roles

* View: **role-select.html**
* User must select role once (either **QUEST\_MASTER** or **ADVENTURER**)
* If **ADVENTURER**, the system assigns a **random** class (**MYSTIC\_MUSE**, **GHOST\_HUNTER**, or **DOOMBRINGER**)
* After successful role and class assignment, redirect to **/home**
* If the role is already set, redirect to /home directly. Do not allow the user to change their role/class once they are set.

### Home Page /home Important: *If the logged in user has NO role selected, redirect to Role Selection Page. For this you may need to extend the logic in the given interceptor.*

Based on the user role:

#### If **QUEST\_MASTER**, the user must see the following:

* View: **quest-master-home.html**
* **Open Quest** button to sends request to **/quests** page
* **New Item** button to send request to **/items** page
* Display system statistics on the view:
  + Total adventurers, total Quests, and total Items
* Display adventurer list (**ordered** **by** level descending). Each adventurer entry must display:
  + Class Icon, Nickname, Class Name, Level, XP progress bar
  + **Note:** You should think of a way to display the progress bar with the correct width, there is a way using thymeleaf to add CSS class dynamically. Analyze the bar given in the view and determine how the width is being used.

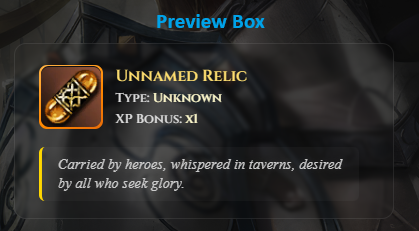
***Level and progress calculation:***

*int* ***level*** *= (int)(playerXp / 10) + 1;  
double* ***progress*** *= Math.max(1, ((playerXp % 10) / 10.0) \* 100);*

#### If **ADVENTURER**, the user must see the following:

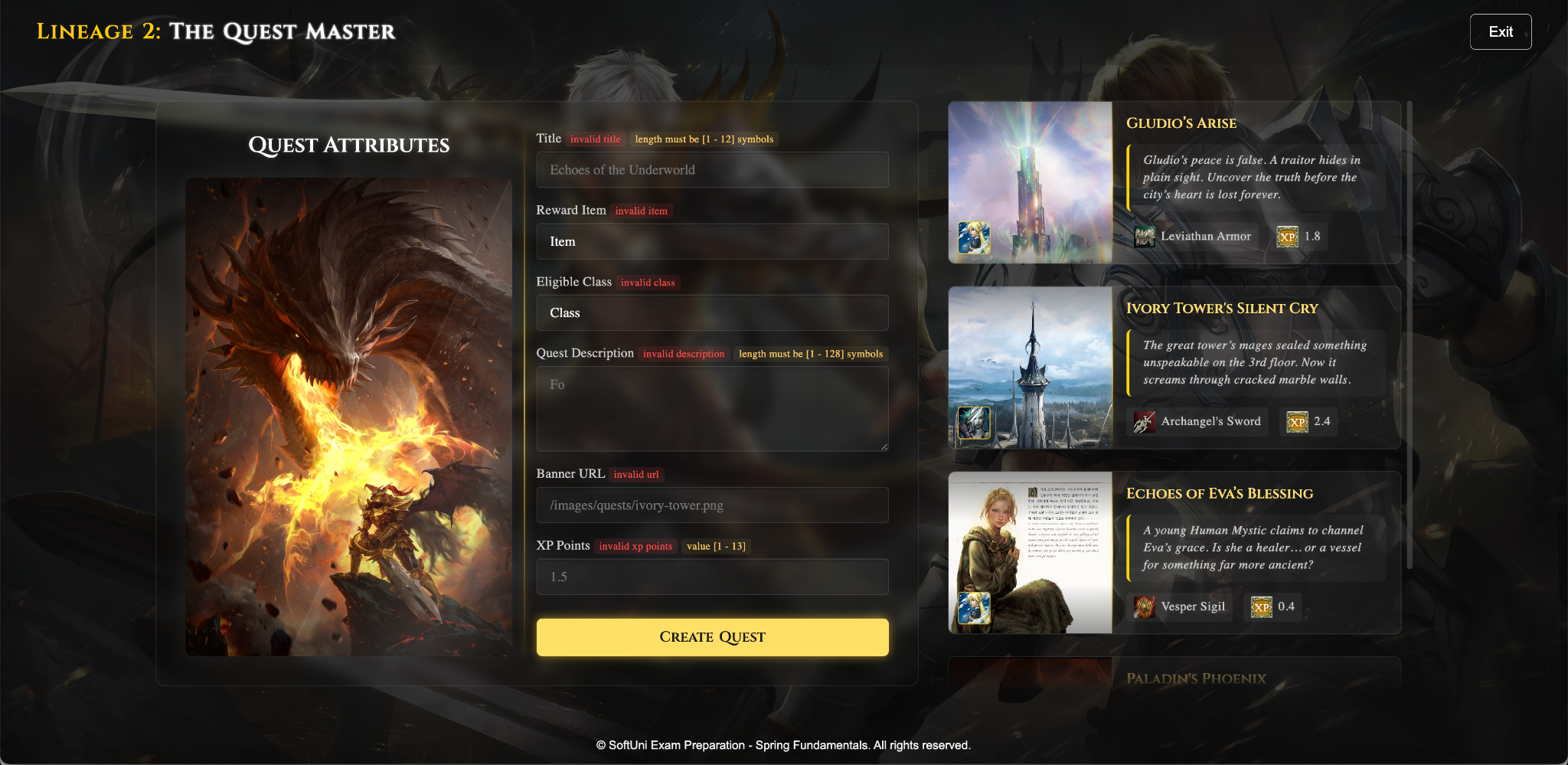
* View: **adventurer-home.html**
* Display player details: nickname, class, level, XP, progress bar
  + **Note:** You should think of a way to display the progress bar with the correct width, there is a way using thymeleaf to add CSS class dynamically. Analyze the bar given in the view and determine how the width is being used.
* Display quests list (**eligible for their class only**)
  + Ordered by creation date (desc)
  + Shows: banner, title, XP, item reward, class icon, and button:
    - Capture if not yet taken
    - Completed if already captured by a player
* Display **earned** Item list (all items this player earned, while capturing quests):
  + Shows icon, name, type, XP multiplier in format **“x{multiplier} XP”**
  + Shows the quest it was earned from

### Items Page /items

* View: **items.html**
* This page has a form for creating items, that must obey the following validation constraints:
  + Name (6–26 chars), XP Multiplier (1–3)
  + All fields required
  + Display invalid input red label when the submitted value is incorrect
  + Display the constraint yellow validation message always
* Every new item must be unique (by name and icon URL)
* After creation, redirect to Items Page again, the newly created item must be displayed in the list with items.
* The page must display a list of all created items (ordered by **createdOn** in descending order, then ordered by **xpBonusMultiplier** in descending order). Each item has an Icon, Name, Type, Multiplier, Creator nickname, Creation date. **XP Bonus Multiplier must be formatted EXACTLY 1 symbol after the decimal sign.**
* Button “Back to Dashboard” sends the user to **Home page**
* **Do not change the HTML/CSS/JS for the preview box!**

### Quests Page /quests

* View: quests.html
* This page supports a Create Quest form, that must obey the following validation constraints
  + Title (6–26 chars), Description (110–128 chars), XP (1.0–13.0)
  + All fields required
  + Display invalid input red label when the submitted value is incorrect
  + Display the constraint yellow validation message always
* Every new quest must be unique (by title and banner URL)
* List of created quests (ordered by creation date and XP desc). Each Quest entry displays title, description, banner image, eligible class icon (mystic-muse-icon.png, ghost-hunter-icon.png, doombringer-icon.png), reward item (icon + item name), xp
* After creation, redirect to the Quest Page again, the newly created quest must be present in the list with quests.



### Capture Quest Logic

* An adventurer can capture any quest eligible for their class
* Capturing:
  + Sets the current player as capturer
  + Player earns: **quest.xpPoints \* rewardItem.xpBonusMultiplier**
  + Every quest a player captured means that the player earned the item from that quest.
* After capturing the quest redirect to the home page.
* Business constraints:
  + Once a quest is captured, it becomes unavailable to others, and cannot be captured a second time.

## Visual Details

* All lists support empty placeholders (e.g. “No quests available for your class”). **You should** **properly display the correct html component when a list of objects** (items, quests, adventurers) is empty. These components are commented out in the html documents, please check carefully.
* Whenever you should display a class icon dynamically, you should use one of the following assets provided in the project:
  + mystic-muse-icon.png, ghost-hunter-icon.png, doombringer-icon.png

## Security

* Session-based login (store **user\_id** in session)
* Access control:
  + Guests: only /, /register, /login
  + Logged users: the rest of the endpoints
* Role checks must be applied (e.g. only adventurers can capture quests, only quest masters can create new items/quests)

## Validation & Exception Handling

Every form in the application must include **proper server-side validation**. If a user submits a form with invalid data (e.g. missing fields, invalid lengths, out-of-range numbers), the form should be redisplayed with:

* Appropriate **error messages** shown in red next to the specific fields that failed validation.

You must also enforce all **business constraints** and throw **custom exceptions** when these rules are violated. Make sure your system is **resilient to invalid input**. It is normal to encounter White Label Error Page when an exception is thrown.