**Section 5: Class Diagram (Person C)**

**Overview**

The class design models the structural backbone of the AI-powered course generation platform, defining relationships among key entities such as Course, Lesson, Quiz, Flashcard, User, and their interactions. The system adopts an object-oriented design approach, using **inheritance**, **aggregation**, and a **Factory Pattern** to streamline object creation, ensure modularity, and promote scalability.

**5.1 Key Classes and Descriptions**

| **Class Name** | **Description** | **Key Attributes** | **Key Methods** |
| --- | --- | --- | --- |
| User | Represents all users (Creators and Learners). | userID, name, email, password, role | login(), logout(), viewDashboard() |
| Creator (extends User) | Subclass representing content creators who upload and edit materials. | approvalStatus | uploadDocument(), editLesson() |
| Learner (extends User) | Subclass representing learners who study and take quizzes. | progress, enrolledCourses | takeQuiz(), reviewFlashcards() |
| Course | Aggregates lessons, quizzes, and flashcards. | courseID, title, description, difficulty, lessons[], quizzes[], flashcards[] | addLesson(), addQuiz(), publishCourse() |
| Lesson | AI-generated learning module derived from uploaded content. | lessonID, title, content, keyConcepts[] | generateSummary(), getKeywords() |
| Quiz | Represents quizzes linked to lessons. | quizID, type, questions[], score | generateQuiz(), evaluateQuiz() |
| Flashcard | Represents a quick-review unit containing question-answer pairs. | cardID, question, answer, topic | displayCard(), editCard() |
| AIContentFactory | ImplementsFactory Patternfor creating AI-generated learning artifacts (Lesson, Quiz, Flashcard). | N/A | createContent(contentType): Content |
| RepositoryManager | Manages GitHub integration for versioning and team collaboration. | repoURL, versionHistory[] | commitChanges(), pullUpdates() |
| Dashboard | Aggregates analytics and progress tracking data for learners and creators. | userID, progressStats, courseStats | generateReport(), displayOverview() |

**5.2 Relationships**

1. **Inheritance**
   * Creator and Learner both inherit from User (User → Creator, User → Learner).
   * This allows sharing of authentication and profile features while supporting specialized actions based on their role.
2. **Aggregation**
   * Course aggregates multiple Lessons, Quizzes, and Flashcards.
   * Deleting a course does not delete the original user or AI factory; thus, it’s a *weak aggregation* relationship.
3. **Association**
   * User interacts with courses (1...\*) and dashboard analytics.
   * Lessons are linked with quizzes and flashcards through logical associations based on learning topics.
4. **Factory Pattern Implementation**
   * The **AIContentFactory** acts as a centralized object creator following the Factory Method Design Pattern.
   * It dynamically produces subclasses (Lesson, Quiz, Flashcard) using createContent(contentType) depending on input type.

**Example:**

AIContentFactory factory = **new** AIContentFactory();

Lesson newLesson = (Lesson) factory.createContent("Lesson");

Quiz newQuiz = (Quiz) factory.createContent("Quiz");

**5.3 Textual Representation of Diagram**

+-------------------+

| User |

+-------------------+

| -userID |

| -name |

| -email |

| -password |

| -role |

+-------------------+

| +login() |

| +logout() |

| +viewDashboard() |

+-------------------+

/ \

/ \

+-----------+ +-----------+

| Creator | | Learner |

+-----------+ +-----------+

| -approvalStatus| -progress |

| +uploadDoc() | +takeQuiz()|

| +editLesson() | +reviewFlashcards()|

+----------------+-------------+

|

|

+---------------+

| Course |

+---------------+

| -courseID |

| -title |

| -description |

| -difficulty |

+---------------+

| +addLesson() |

| +addQuiz() |

| +publishCourse()|

+---------------+

| o

| ● Aggregation

+----------------+---------------------+

| | |

+--------+ +--------+ +-------------+

| Lesson | | Quiz | | Flashcard |

+--------+ +--------+ +-------------+

| -lessonID| | -quizID | | -cardID |

| -content | | -type | | -question |

| -keywords| | -score | | -answer |

+----------+ +---------+ +-------------+

| +generateSummary() | +generateQuiz() | +displayCard() |

+--------------------------------------------------------+

+--------------------------+

| AIContentFactory |

+--------------------------+

| +createContent(type) |

+--------------------------+

+--------------------------+

| Dashboard |

+--------------------------+

| -progressStats |

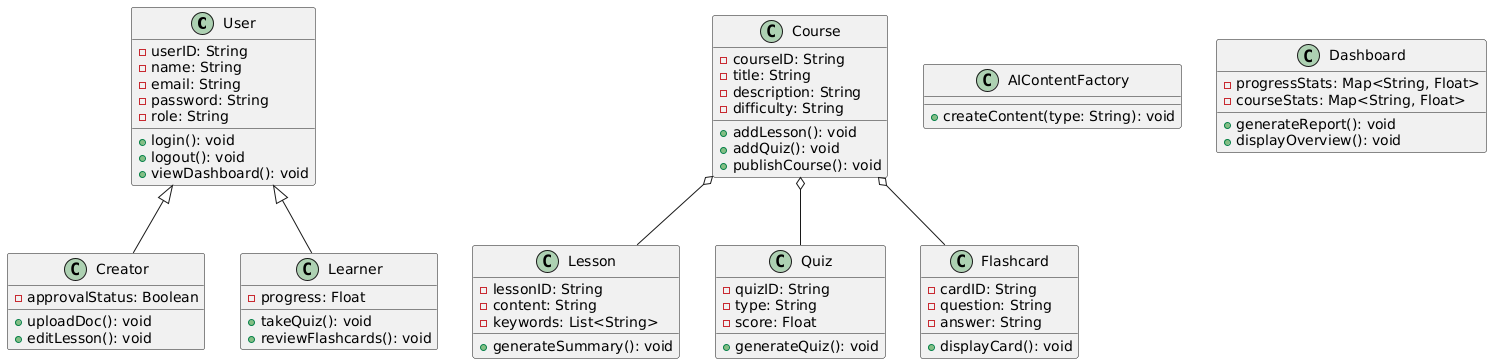
| -courseStats |

+--------------------------+

| +generateReport() |

| +displayOverview() |

+--------------------------+



**5.4 Explanation of Factory Pattern Integration**

The AIContentFactory adheres to the **Factory Method** principles by encapsulating content creation logic to produce different course components based on type input. This supports scalability and reusability: new types (e.g., “Assessment”, “InteractiveExercise”) can be introduced without altering existing class structures.

**5.5 Relationships Summary**

| **Relationship Type** | **Example Classes** | **Description** |
| --- | --- | --- |
| Inheritance | User → Creator, Learner | Common attributes and behaviors shared among user types |
| Aggregation | Course → Lesson, Quiz, Flashcard | Course owns references to learning components |
| Association | User → Dashboard | One-to-one analytics interaction |
| Factory Pattern | AIContentFactory → Lesson/Quiz/Flashcard | Centralized content creation logic |

**5.6 System Benefits from Design**

* Promotes **modularity** and **maintainability** through loosely coupled classes.
* **Extensible** architecture allows easy addition of new content types or analytics modules.
* **Multi-role integration** through inheritance supports natural role-based functionality.
* **Factory Pattern** ensures clean separation between creation logic and application logic.