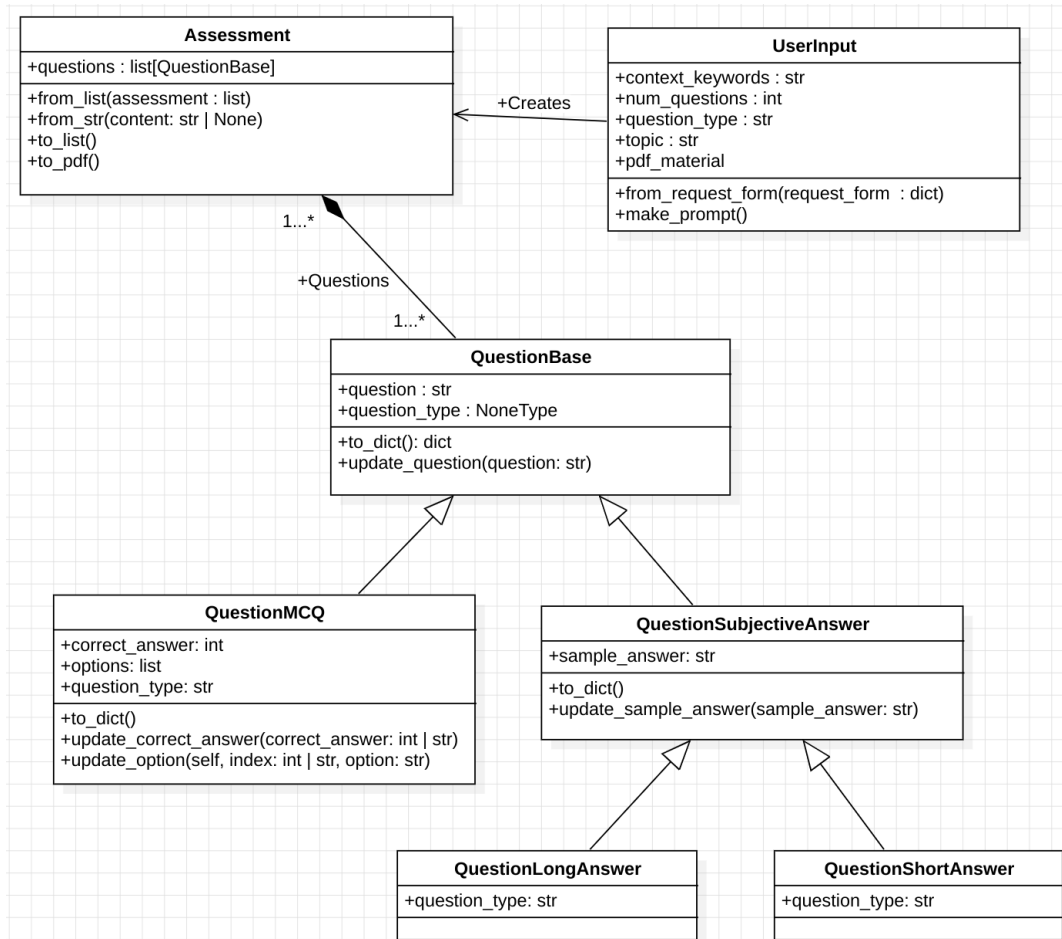# Product Design

Team 20: Archisha Panda, Gargi Shroff, Ankith Pai, Prakhar Jain
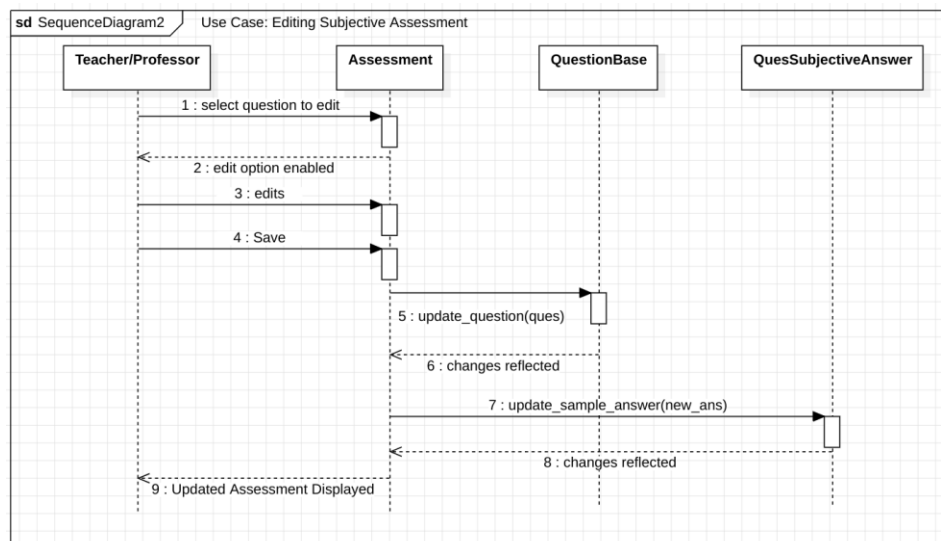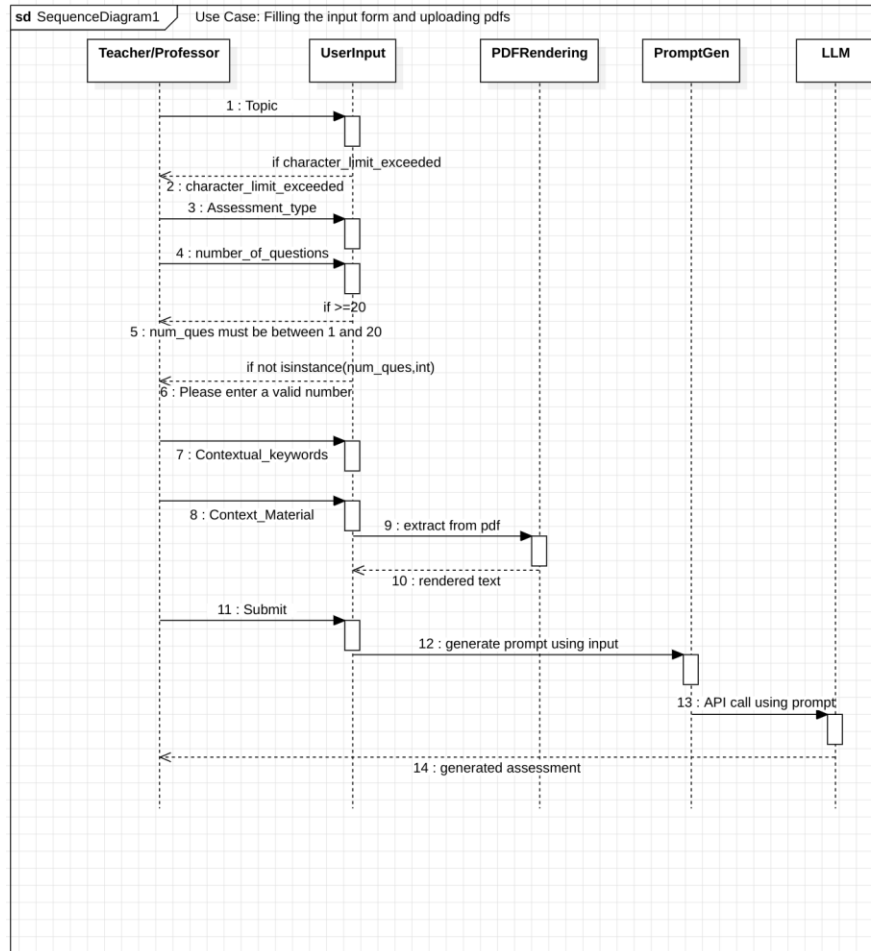
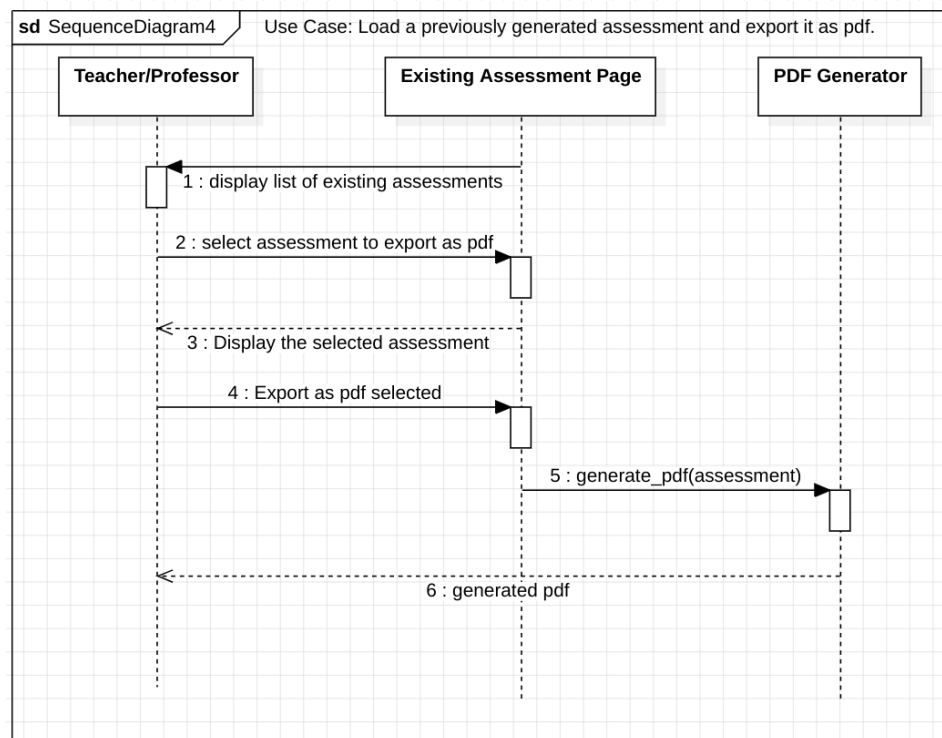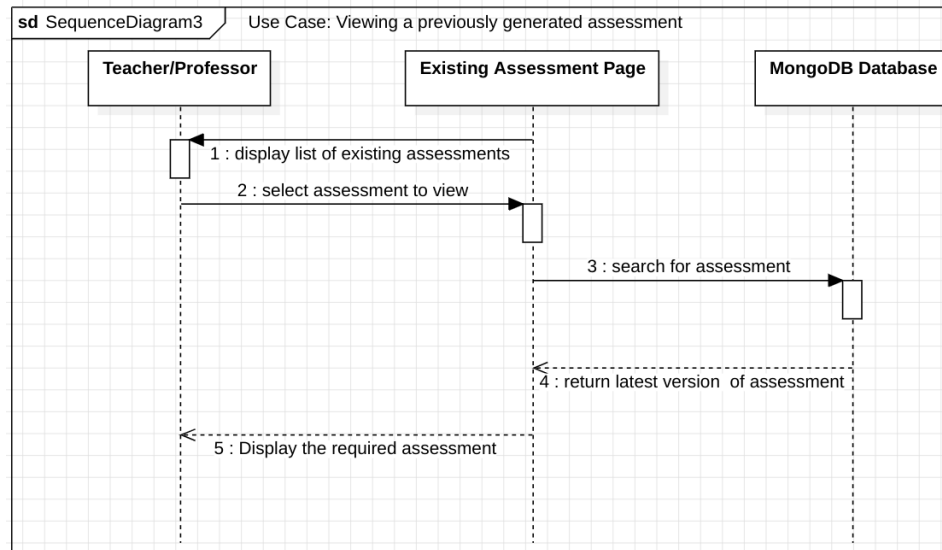### Design Model

`UserInput`	This is a class that represents the input entered by the user sent from the frontend for page 1. Class state • `topic:` Topic of the assessment to be generated • `question_type:` Type of questions in the assessment • `num_questions:` Number of questions to generate • `context_keywords:` Any optional context keywords needed for the LLM • `pdfs:` List of PDF contexts uploaded Class behavior • `from_request_form:` Takes the input dictionary and instantiates the class from it • `make_prompt:` Constructs the prompt string to be forwarded to the LLM
`QuestionBase`	This is a base class used to represent a question Class state • `question:` Stores the question string • `question_type:` Type of the question Class behavior • `update_question:` This method updates the stored question • `to_dict:` Returns a dict representation of the question
`QuestionSubjectiveAnswer`	This is a subclass of `QuestionBase` for representing a subjective question, inherits all state and behavior, and additionally implements the following. *Class state* • `sample_answer:` Stores a sample answer Class behavior • `update_sample_answer:` This method updates the stored question
`QuestionShortAnswer`	This is a subclass of `QuestionSubjectiveAnswer` for representing a subjective short answer type question. Inherits all state and behaviour, and does not implement anything more

`QuestionLongAnswer`	This is a subclass of `QuestionSubjectiveAnswer` for representing a subjective long answer type question. Inherits all state and behaviour, and does not implement anything more
`QuestionMCQ`	This is a subclass of `QuestionBase` for representing an MCQ question, inherits all state and behavior, and additionally implements the following. *Class state* • `options:` Stores a list of all options that are pickable • `correct_answer:` Stores the index of the correct option Class behavior • `update_option:` This method updates a particular option at the given index. • `update_correct_answer:` This method updates the correct answer
`Assessment`	This is a class that represents a generated assessment. *Class state* • `questions:` Stores the list of questions Class behavior • `from_list`: Constructor to get assessment from list representation • `from_str`: Constructor to get assessment from str (LLM response) • `to_list`: Returns list representation of the assessment • `to_pdf`: Returns a generated PDF for the assessment

.

Assessment

+questions : list[QuestionBase]

+from_list(assessment : list)
+from_str(content: str | None)
+to_list()
+to_pdf()

UserInput

+context_keywords : str
+num_questions : int
+question_type : str
+topic : str
+pdf_material

+from_request_form(request_form : dict)
+make_prompt()

+Creates

1...*

+Questions

1...*

QuestionBase

+question : str
+question_type : NoneType

+to_dict(): dict
+update_question(question: str)

QuestionMCQ

+correct_answer: int
+options: list
+question_type: str

+to_dict()
+update_correct_answer(correct_answer: int | str)
+update_option(self, index: int | str, option: str)

QuestionSubjectiveAnswer

+sample_answer: str

+to_dict()
+update_sample_answer(sample_answer: str)

QuestionLongAnswer

+question_type: str

QuestionShortAnswer

+question_type: str

Sequence Diagram(s)

sd SequenceDiagram1 — Use Case: Filling the input form and uploading pdfs

| Teacher/Professor | UserInput | PDFRendering | PromptGen | LLM |

1 : Topic

if character_limit_exceeded
2 : character_limit_exceeded

3 : Assessment_type

4 : number_of_questions

if >=20
5 : num_ques must be between 1 and 20

if not isinstance(num_ques,int)
6 : Please enter a valid number

7 : Contextual_keywords

8 : Context_Material

9 : extract from pdf

10 : rendered text

11 : Submit

12 : generate prompt using input

13 : API call using prompt

14 : generated assessment

sd SequenceDiagram2 — Use Case: Editing Subjective Assessment

| Teacher/Professor | Assessment | QuestionBase | QuesSubjectiveAnswer |

1 : select question to edit

2 : edit option enabled

3 : edits

4 : Save

5 : update_question(ques)

6 : changes reflected

7 : update_sample_answer(new_ans)

8 : changes reflected

9 : Updated Assessment Displayed

sd SequenceDiagram3 ⟍ Use Case: Viewing a previously generated assessment

| Teacher/Professor | Existing Assessment Page | MongoDB Database |

1 : display list of existing assessments

2 : select assessment to view

3 : search for assessment

4 : return latest version of assessment

5 : Display the required assessment

sd SequenceDiagram4 ⟍ Use Case: Load a previously generated assessment and export it as pdf.

| Teacher/Professor | Existing Assessment Page | PDF Generator |

1 : display list of existing assessments

2 : select assessment to export as pdf

3 : Display the selected assessment

4 : Export as pdf selected

5 : generate_pdf(assessment)

6 : generated pdf

Design Rationale

- Removed the `from_dict` classmethod, from all question classes as it had uneeded redundancy. This function was responsible for initializing the class given the dict representation of the assessment. Now, the constructor function `__init__` does what this function used to do.
- Added a `QuestionSubjectiveAnswer` class that `QuestionLongAnswer` and `QuestionShortAnswer` inherit from. This removes code duplication in these two classes, by moving their common functionality to the super class.

- Added `update_*` methods that update various attributes of question classes. This is done so that input validation can be enforced.