Existing System

Data Models

Feel free to make a copy of this, or work directly within the box and treat this as a worksheet.

Assignments Resource

```
GET /api/assignments/<AssignmentId>
POST /api/assignments

Assignment {
   ● assignmentId: AssignmentId
   ● courseId: CourseId
   ● question: Question
}
```

Submissions Resource

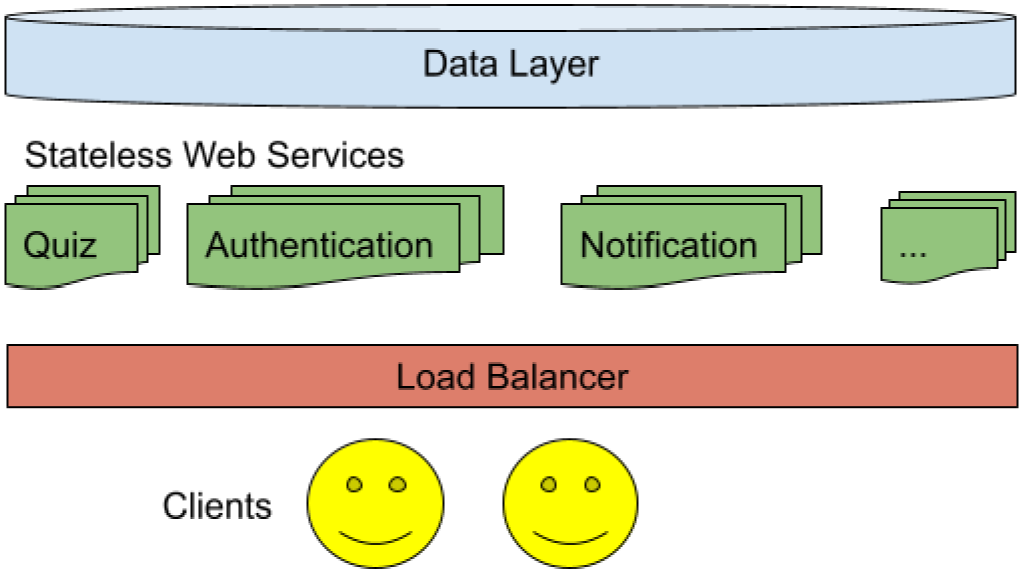
```
GET /api/submissions/<SubmissionId>
PUT /api/submissions/<SubmissionId>
POST /api/submissions

Submission {
   ● submissionId: SubmissionId
   ● assignmentId: AssignmentId
   ● submitterId: UserId
   ● answer: Answer
   ● submitted: Boolean
}
```

Grades Resource

```
GET /api/grades/<submissionId>
```

Service Structure



Feature Request: Staff Graded Assignments

Extend the current system to support staff graded assignments. For staff graded assignments, after a learner has submitted an assignment, a staff member will evaluate the assignment. They provide a grade and potentially written feedback.

1. How would you augment the models above to support this?
2. How will you calculate the submission state: started, waiting for grading, graded?

```
Assignment {
    ● assignmentId: AssignmentId
    ● courseId: CourseId
    ● question: Question
    ● instructor : Instructor
}


Question {

getQuestionText();

}
```

```
AutoGradedQuestion extends Question{

}

StaffGradedQuestion extends Question {

}



Enum SubmissionState{
        Started,
        waitingForGrading,
        Graded
}

Answer {

getAnswer();

}

AutoGradedAnswer extends Answer{
}

StaffGradedAnswer extends  Answer{

}


Submission {
   ● submissionId: SubmissionId
   ● assignmentId: AssignmentId
   ● submitterId: UserId
   ● answer: Answer
   ● submissionState :  SubmissionState
}


Person

Instructor extends Person{
}

GET /api/submissions/<SubmissionId>
POST /api/submissions/grade/<SubmissionId>

{
  grade : float
}
```

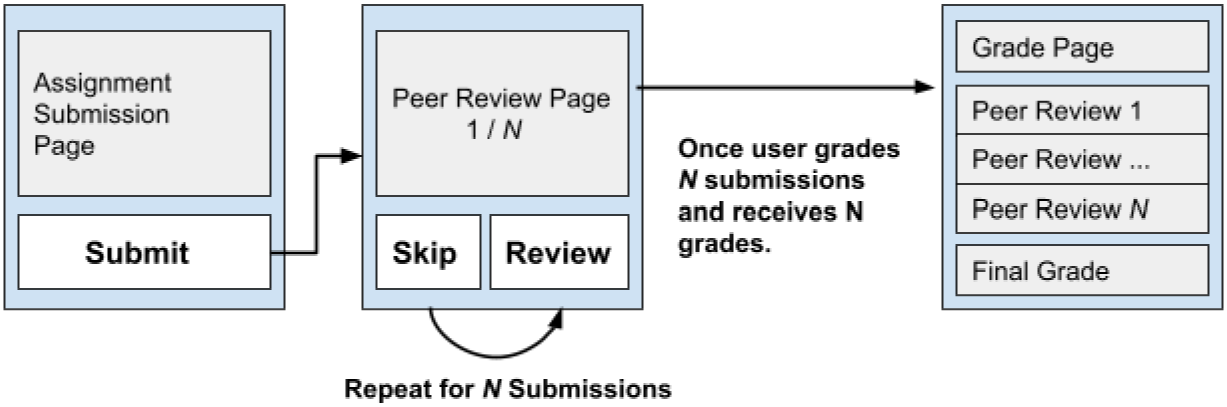
—-------------------

Feature Request: Peer Graded Assignments

Extend the current system to support peer graded assignment. For peer assignments, after a learner has submitted an assignment, they are required to review N of their peers' submissions to the same assignment. After they have given N reviews and either received N reviews or a week has elapsed since they submitted, they can get their grade. The final grade is the average of peer reviews. Learners are allowed to skip reviews (e.g., if the submission is not in their native language) and sometimes they forget to finish. See wireframes for the overall flow.

1. How would you augment the models / APIs above to support this?
2. How will you calculate the submission state: started, needs to give reviews, waiting to receive reviews, graded?

Wire frame Designs



```
Enum SubmissionState{
     Started,
     needsToGiveReviews,
        waitingForGrading,
        Graded
}

Peer extends Person {

}

Submission {
   ● submissionId: SubmissionId
   ● assignmentId: AssignmentId
   ● submitterId: UserId
   ● answer: Answer
   ● submissionState :  SubmissionState
   ● ReviewedBy : [Peer]
}

PeerReview {
        AssignmentId : String
# 200 k
        ReviewWaitingAssignment : minHeap<reviewCount, -timeLeft,  SubmissionID>

        }
}
```

0, -7,
0, 0

Corn job, reading this PeerReview, and removing submission from below whose review count >=5 or timeleft is 0.

```
PeerGraded extends Grade {

        Submission id  :
        Reviews : [{ “gradeBy” : Peer
           Grade : float
        }
        ]

}
```

Hashing : submission id / hash

So the submission should go a bucket

Each peer, we will create topic in kafka queue, and once his submission is done, we will pull top 20 <waiting review submission> from the min heap and place them in the queue for that user.