**CS673 Software Engineering**

**Team 3 - Rhettoric**

**Software Design Document**

| Team Member | Role(s) | Signature | Date |
| --- | --- | --- | --- |
| Magnus Urosev | Team Leader | *Magnus Urosev* | 5/28 |
| Adrian Ortiz | Configuration Leader / Security Leader |  |  |
| Xi Zeng | QA Leader |  |  |
| Jack Cairns |  |  |  |
| Jianing Li |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Revision history**

| **Version** | **Author** | **Date** | **Change** |
| --- | --- | --- | --- |
| **V 1.0** | **Magnus Urosev** | **5/28** | **Initial docs** |
| **V 1.1** | **Xi Zeng** | **5/29** | **ERD initial draft** |
| **V 1.2** | **Xi Zeng** | **6/5** | **Update new version of ERD and separate to make it clear to see** |
| **V 1.3** | **Magnus Urosev** | **6/10** | **Addressing previous feedback from Dr. Zhang** |
| **V 1.4** | **Xi Zeng** | **6/10** | **Revise ERD for DBs** |

[Introduction](#_heading=h.gjdgxs)

[Software Architecture](#_heading=h.30j0zll)

[Class Diagram](#_heading=h.1fob9te)

[UI Design (if applicable)](#_heading=h.3znysh7)

[Database Design (if applicable)](#_heading=h.2et92p0)

[Security Design](#_heading=h.tyjcwt)

[Business Logic and/or Key Algorithms](#_heading=h.3dy6vkm)

[Design Patterns](#_heading=h.1t3h5sf)

[Any Additional Topics you would like to include.](#_heading=h.4d34og8)

[References](#_heading=h.2s8eyo1)

[Glossary](#_heading=h.17dp8vu)

# Introduction

In this section, give an overview of this document, and also address the design goals of your software system.

Rhettoric is a project focused on creating a survey. The project aims to provide accurate and reliable insight to a user's experience and feedback on a given course.

This is a project for the CS 673 class with Dr. Zhang. The objective is to create a survey for a course learning platform where at the end of the course, a survey is generated and distributed to the users to answer and provide valuable feedback to the course creator.

# Software Architecture

The backend stores the backend logic such as the Django and Docker files. This houses the question and survey models along with others.

Frontend has a placeholder React frontend for later implementation.

Survey\_form directories contain the implementation of the survey from the model implementations, some frontend, and much more.

We will use REST APIs to facilitate communication between the frontend and backend, enabling dynamic data exchange and manipulation of survey elements, in the case we implement a frontend.

Surveys are still in development and will contain user auth.

code

├── backend

│ ├── deploy\_script.sh

│ ├── Django

│ │ ├── Dockerfile

│ │ ├── Generate\_Keys.py

│ │ ├── \_\_init\_\_.py

│ │ ├── manage.py

│ │ ├── parse\_db\_values.py

│ │ ├── requirements.txt

│ │ └── settings

│ ├── Dockerfile

│ ├── \_\_init\_\_.py

│ └── requirements.txt

├── frontend

│ ├── App.js

│ ├── components

│ │ ├── Dashboard.js

│ │ ├── Login.js

│ │ └── Register.js

│ ├── Dockerfile

│ ├── index.js

│ └── public

│ └── index.html

├── \_\_init\_\_.py

├── nginx

│ └── nginx.conf

├── Readme.md

├── survey\_form

│ ├── db.sqlite3

│ ├── Dockerfile

│ ├── \_\_init\_\_.py

│ ├── manage.py

│ ├── requirements.txt

│ ├── survey\_app

│ │ ├── admin.py

│ │ ├── apps.py

│ │ ├── \_\_init\_\_.py

│ │ ├── migrations

│ │ ├── models.py

│ │ ├── \_\_pycache\_\_

│ │ ├── runtests.py

│ │ ├── serializers.py

│ │ ├── tests.py

│ │ ├── urls.py

│ │ └── views.py

│ ├── survey\_form\_backend

│ │ ├── asgi.py

│ │ ├── \_\_init\_\_.py

│ │ ├── \_\_pycache\_\_

│ │ ├── settings.py

│ │ ├── urls.py

│ │ └── wsgi.py

│ ├── survey\_frontend

│ │ ├── node\_modules

│ │ ├── package.json

│ │ ├── package-lock.json

│ │ ├── public

│ │ ├── README.md

│ │ └── src

│ ├── surveys

│ │ ├── admin.py

│ │ ├── apps.py

│ │ ├── \_\_init\_\_.py

│ │ ├── migrations

│ │ ├── models.py

│ │ ├── \_\_pycache\_\_

│ │ ├── serializers.py

│ │ └── tests.py

│ └── venv

│ ├── bin

│ ├── include

│ ├── lib

│ ├── lib64 -> lib

│ └── pyvenv.cfg

└── test\_github\_access.txt

# Class Diagram

In this section, you will provide a detailed description of each component (or package) and use one or multiple class diagrams to show the main classes and their relationships in each component.

This is our implementation for how someone will complete the survey and the relationship between question, response, and the survey.

# UI Design (if applicable)

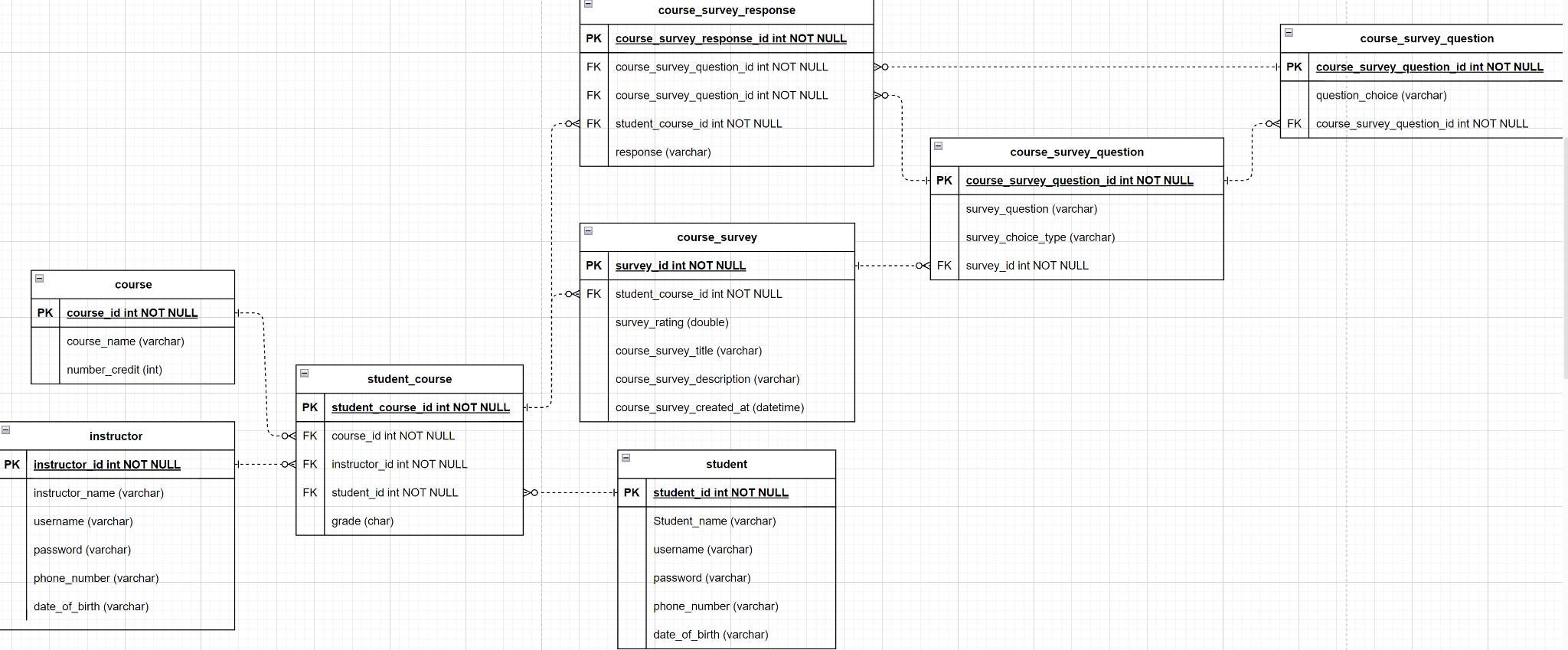
In this section, you can describe your UI design. You can include both your initial design before the implementation and the screenshots of your UI after the implementation.

N/A

# Database Design (if applicable)

In this section, you shall describe any database schema if used in your software system.

As of now we are thinking of a SQL DB but are still TBD. Django has built in code for sqllite and postgressql



# Security Design

In this section, you shall describe any security design in your software system.

Within Django we will be using JSON Web Tokens which is built into the framework. This will take care of user authentication.

From DBs design, we would separate the login page for student and instructor login like myBU portal.

# Business Logic and/or Key Algorithms

In this section, you shall describe any key algorithms used in your software system, either in terms of pseudocode or flowchart, or sequence diagrams.

Algorithm: Survey Creation

* Facilitator Input:
  + Facilitator provides the survey title and description.
  + Facilitator provides a list of questions, each with text and choice type.
* Survey Creation:
  + Create a new survey instance with the given title and description.
  + Save the survey instance to the database.
* Question Assignment:
  + For each question provided by the user:
    - Create a new question instance with the given text and choice type.
    - Assign the question to the created survey.
    - Save the question instance to the database.

# Design Patterns

In this section, you shall describe any design patterns used in your software system.

The big design pattern that we will be using is the OO factory design pattern. The idea is we can easily create questions within a survey. This is nice since we will want single instances of the same type of question and will add other attributes to them.

Example:  
  
class Question:

def create(self):

raise NotImplementedError("Subclass must implement abstract method")

class MultipleChoiceQuestion(Question):

def create(self):

return "Multiple Choice Question"

class OpenEndedQuestion(Question):

def create(self):

return "Open Ended Question"

class QuestionFactory:

@staticmethod

def get\_question(question\_type):

questions = {

"multiple\_choice": MultipleChoiceQuestion,

"open\_ended": OpenEndedQuestion

}

return questions[question\_type]()

# Usage

factory = QuestionFactory()

question = factory.get\_question("multiple\_choice")

print(question.create())

# Any Additional Topics you would like to include.

# References

* Django Documentation: Official documentation for Django [Django Documentation](https://docs.djangoproject.com/)
* Django REST Framework: Documentation for using Django REST framework for creating REST APIs [DRF Documentation](https://www.django-rest-framework.org/)
* JSON Web Tokens (JWT): Documentation on JWT, an open standard for securely transmitting information. Available at: JWT Documentation

# Glossary

* Algorithm: A step-by-step procedure for calculations, data processing, and automated reasoning tasks.
* API (Application Programming Interface): A set of functions and protocols for building and interacting with software applications.
* Authentication: The process of verifying the identity of a user or process.
* Authorization: The process of verifying what a user is allowed to do.
* Django: A high-level Python web framework that encourages rapid development and clean, pragmatic design.
* JWT (JSON Web Token): A compact, URL-safe means of representing claims to be transferred between two parties.
* REST (Representational State Transfer): An architectural style for designing networked applications.
* Token: A piece of data that represents the right to perform some operation.