**Implementation**

The front end is written in ReactJS in JavaScript and the backend is written in Flask web server in python. Both parts of the code share data between each other via RESTful APIs.

The implementation of this project heavily utilises the GET and the POST HTTP requests.

The Frontend End

Folder structure

|-src

| |-App.js

| |-components

| | |-BaseTemplate.jsx

| | |-BaseTemplateTeacher.jsx

| | |-Card.jsx

| | |-Nav.jsx

| | |-NavTeacher.jsx

| |-index.css

| |-index.js

| |-logo.svg

| |-pages

| | |-StudentDashboard.jsx

| | |-StudentLogin.jsx

| | |-StudentReports.jsx

| | |-StudentSubmission.jsx

| | |-TeacherCreate.jsx

| | |-TeacherDashboard.jsx

| | |-TeacherEntry.jsx

| | |-TeacherLogin.jsx

**Student Login**

We have used functional react components to build the UI in pieces. By doing this we can divide the entire code in small sections that will make maintaining the code easy and simple.

Axios is a package that helps us make calls to API endpoints from the frontend.

The React router dom is another package that helps with Client side navigation on the SPA app.

There are 2 text inputs which hold the username and password respectively. A button is added to take these inputs and send this data to the server to authenticate the user. If the server replies back with a positive response that means the user has successfully authenticated else we show an error message.

Axios handles the call to the backend in this scenario. It is a post request to “<http://localhost:5000>” where the web server is listening.

The useState hooks are used to store the password and username as and when the user types it.

Code:

**Studentlogin.jsx**

import { useState } from "react";

import { useHistory } from "react-router-dom";

import axios from "axios";

function StudentLogin(params) {

const [username, setUsername] = useState("");

const [password, setPassword] = useState("");

let history = useHistory();

function login(e) {

e.preventDefault();

axios

.post("http://localhost:5000/loginStudent", { username, password })

.then((res) => {

console.log(res.data);

if (res.data.message === "auth successful") {

localStorage.setItem("classid", res.data.user.classid);

localStorage.setItem("sid", res.data.user.sid);

localStorage.setItem("name", res.data.user.name);

history.push("/student");

} else {

alert("something went wrong");

}

});

}

return (

<section className="h-screen bg-gray-800 flex flex-col justify-center items-center ">

<div className="text-3xl bg-white p-6 rounded-xl ">

<div className="mb-">

{" "}

<span className="font-bold">Smart</span>{" "}

<span className="text-yellow-500 font-bold">Classroom</span>{" "}

</div>

<div className="text-lg mb-5 ">Login </div>

<form className="flex flex-col text-lg w-screen max-w-xl">

<input

type="text"

onChange={(e) => setUsername(e.target.value)}

placeholder="Username"

className="border-2 rounded-md p-3 my-2"

/>

<input

type="password"

onChange={(e) => setPassword(e.target.value)}

placeholder="Password"

className="border-2 rounded-md p-3 my-2 mb-10"

/>

<button

onClick={(e) => login(e)}

type="submit"

className="bg-yellow-300 text-white text-lg font-bold px-6 py-6 rounded-2xl"

>

Login

</button>

</form>

</div>

</section>

);

}

export default StudentLogin;

**Student Dashboard**

import BaseTemplate from "../components/BaseTemplate";

import greetingTime from "greeting-time";

import Card from "../components/Card";

import date from "date-and-time";

import axios from "axios";

import { useEffect, useState } from "react";

function Home(params) {

useEffect(() => {

axios

.post("http://localhost:5000/getNotification", {

class: localStorage.getItem("classid"),

})

.then((res) => {

setnotifications(() => {

return res.data.map((item) => item);

});

});

}, []);

const [notifications, setnotifications] = useState([]);

const notifiComp = notifications.map((item) => {

return (

<*Card*

priority={item.priority}

title={item.title}

teacher={item.teacher}

></*Card*>

);

});

return (

<*BaseTemplate*>

<section className="lg:col-span-3 border">

<section className=" m-16 mt-14 flex flex-col">

<div className=" flex justify-between">

<div className="flex flex-col">

<span className="text-3xl">

{greetingTime(new *Date*())} {localStorage.getItem("name")},

</span>

<span className="">Get to speed with your day</span>

</div>

<span className="opacity-70">

{date.format(new *Date*(*Date*.now()), "ddd, DD MMMM, YYYY")}

</span>

</div>

<div className="mt-14"></div>

<div className="font-medium">Annoucements</div>

<div className="mt-7"></div>

<div className="grid grid-cols-1 lg:grid-cols-2 gap-3">

{notifiComp}

</div>

</section>

</section>

<section className="grid grid-rows-2 m-16 lg:m-0">

<div className="lg:border lg:border-l-0">

<div className="m-14 mx-5">

Calendar of Events

<img

src="https://via.placeholder.com/200"

className=" w-full mt-3 rounded-xl"

alt=""

/>

</div>

</div>

<div className="lg:border lg:border-t-0 lg:border-l-0">

<div className="m-14 mt-5 mx-5">

Time Table

<img

src="https://via.placeholder.com/200"

className=" w-full mt-3 rounded-xl"

alt=""

/>

</div>

</div>

</section>

</*BaseTemplate*>

);

}

export default Home;

**The Backend**

The backend is written in flask python. The database used is sqlite3. The flask server utilises ORM or object relational mapping instead of native SQL. This is done because ORMs have a clear advantage over the classic SQL methods. The port on which the backend runs by default is 5000

**\_\_init\_\_.py**

from flask import Flask

import os

from flask\_sqlalchemy import SQLAlchemy

from flask\_cors import CORS

from flask\_marshmallow import Marshmallow

currentDirectory = os.path.dirname(os.path.abspath(\_\_file\_\_))

app = Flask(\_\_name\_\_)

CORS(app)

app.config["SQLALCHEMY\_DATABASE\_URI"] = "sqlite:///test.db"

db = SQLAlchemy(app)

ma = Marshmallow(app)

from backend import routes

from backend import models

This file initialises the web server. It takes care of circular imports by modularizing the code base. It loads the routes and models into the app and initialises the sqlite database. It also takes care of CORS and SQLAlchemy

Models.py

from backend import db , ma

class Student(*db*.*Model*):

sid = db.Column(db.Integer, primary\_key=True)

username = db.Column(db.String(80), unique=True, nullable=False)

password = db.Column(db.String(120),nullable=False)

usn = db.Column(db.String(80), unique=True)

name = db.Column(db.String(80))

classid = db.Column(db.String(30))

def \_\_repr\_\_(self):

return '<Student %r>' % self.username

class StudentSchema(*ma*.*SQLAlchemySchema*):

class Meta:

model = Student

sid = ma.auto\_field()

username = ma.auto\_field()

password = ma.auto\_field()

usn = ma.auto\_field()

name = ma.auto\_field()

classid = ma.auto\_field()

class Teacher(*db*.*Model*):

tid = db.Column(db.Integer, primary\_key=True)

username = db.Column(db.String(80), unique=True, nullable=False)

password = db.Column(db.String(120),nullable=False)

usn = db.Column(db.String(80), unique=True)

name = db.Column(db.String(80))

classHandled = db.Column(db.String(30)) *# array stored in string*

def \_\_repr\_\_(self):

return '<Teacher %r>' % self.username

class TeacherSchema(*ma*.*SQLAlchemySchema*):

class Meta:

model = Teacher

tid = ma.auto\_field()

username = ma.auto\_field()

password = ma.auto\_field()

usn = ma.auto\_field()

name = ma.auto\_field()

class Notification(*db*.*Model*):

nid= db.Column(db.Integer, primary\_key=True)

priority = db.Column(db.String(20),default="low")

title = db.Column(db.String(120))

desc = db.Column(db.String(256))

classid = db.Column(db.String(30))

tid = db.Column(db.Integer, db.ForeignKey("teacher.tid"))

class NotificationSchema(*ma*.*SQLAlchemySchema*):

class Meta:

model = Notification

nid= ma.auto\_field()

priority = ma.auto\_field()

title = ma.auto\_field()

desc = ma.auto\_field()

classid = ma.auto\_field()

tid = ma.auto\_field()

class SubmissionRequest(*db*.*Model*):

srid = db.Column(db.Integer, primary\_key=True)

tid = db.Column(db.Integer, db.ForeignKey("teacher.tid"))

title = db.Column(db.String(120))

deadline = db.Column(db.String(100))

desc = db.Column(db.String(256))

classid = db.Column(db.String(100))

class SubmissionRequestSchema(*ma*.*SQLAlchemySchema*):

class Meta:

model = SubmissionRequest

srid = ma.auto\_field()

tid = ma.auto\_field()

title = ma.auto\_field()

deadline = ma.auto\_field()

desc = ma.auto\_field()

classid = ma.auto\_field()

class Submission(*db*.*Model*):

subid = db.Column(db.Integer, primary\_key=True)

sid = db.Column(db.Integer , db.ForeignKey("student.sid"))

srid = db.Column(db.Integer , db.ForeignKey("submission\_request.srid"))

filepath = db.Column(db.String(128))

*type* = db.Column(db.String(10))

class SubmissionSchema(*ma*.*SQLAlchemySchema*):

class Meta:

model = Submission

subid = ma.auto\_field()

sid = ma.auto\_field()

srid = ma.auto\_field()

filepath = ma.auto\_field()

*type* = ma.auto\_field()

class Report(*db*.*Model*):

rid = db.Column(db.Integer , primary\_key =True)

*type* = db.Column(db.String(30))

total = db.Column(db.Integer)

marksObtained = db.Column(db.Integer)

sid = db.Column(db.Integer , db.ForeignKey("student.sid"))

class ReportSchema(*ma*.*SQLAlchemySchema*):

class Meta:

model = Report

rid = ma.auto\_field()

*type* = ma.auto\_field()

total = ma.auto\_field()

marksObtained = ma.auto\_field()

sid = ma.auto\_field()

Each table is written in the form of class and has a corresponding schema to convert database objects to python dictionaries that will then be converted into json via the jsonify function.

Route.py

1. Student login route

@app.route("/loginStudent", methods = ["POST"])

def loginStudent():

request\_data = request.get\_json()

username = request\_data["username"]

password = request\_data["password"]

user = Student.query.filter\_by(username=username).first()

print(user)

if not user == None and user.password == password:

return jsonify({"message": "auth successful", "user": studentSchema.dump(user) })

else:

return jsonify({"message": "auth unsuccessful"})

1. Teacher Login route

@app.route("/loginTeacher", methods = ["POST"])

def loginTeacher():

request\_data = request.get\_json()

username = request\_data["username"]

password = request\_data["password"]

user = Teacher.query.filter\_by(username=username).first()

print(user)

if not user == None and user.password == password:

return jsonify({"message": "auth successful", "user": teacherSchema.dump(user) })

else:

return jsonify({"message": "auth unsuccessful"})

1. getNotification route

@app.route("/getNotification" , methods=["POST"])

def getNotification():

*# parameters required*

*# class of the student*

request\_data = request.get\_json()

classOfStudent = request\_data["class"]

allNotification = Notification.query.filter\_by(classid = classOfStudent).all()

print(allNotification)

res = []

for i in allNotification:

res.append({"title":i.title, "teacher": Teacher.query.filter\_by(tid = i.tid).first().name , "priority": i.priority })

return jsonify(res)

1. getSubmissionRequest

@app.route("/getSubmissionRequest" , methods=["POST"])

def getSubmissionRequest():

*# parameters required*

*# class of the student*

request\_data = request.get\_json()

classOfStudent = request\_data["class"]

allSubmissionRequest = SubmissionRequest.query.filter\_by(classid = classOfStudent).all()

*#print(allSubmissionRequest)*

res = []

for i in allSubmissionRequest:

res.append({"title":i.title,

"assignedTeacher": Teacher.query.filter\_by(tid = i.tid).first().name ,

"deadline": i.deadline,

"description" : i.desc,

"submissionID" : i.srid,

"teacherPicture":"https://via.placeholder.com/50"

})

return jsonify(res)

1. getTeacherSubmissionDetails

@app.route("/getTeacherSubmissionDetails" , methods=["POST"])

def getTeacherSubmissionDetails():

*# parameters required*

*# srid*

request\_data = request.get\_json()

srid = request\_data["srid"]

allSubmission = Submission.query.filter\_by(srid=srid).all()

res = []

for i in allSubmission:

res.append({

"studentName": Student.query.filter\_by(sid = i.sid).first().name ,

"studentUSN": Student.query.filter\_by(sid = i.sid).first().usn ,

"class": Student.query.filter\_by(sid = i.sid).first().classid ,

"deadline": SubmissionRequest.query.filter\_by(srid= i.srid).first().deadline,

"filepath" : i.filepath

})

return jsonify(res)

1. createNotification

@app.route("/createNotification" , methods=["POST"])

def createNotification():

*# parameters required*

*# class, priority, title, tid*

request\_data = request.get\_json()

targetClass = request\_data["class"]

tid = request\_data["tid"]

priority = request\_data["priority"]

title = request\_data["title"]

notification = Notification(priority=priority, tid=tid, title=title, classid=targetClass)

teacher = Teacher.query.filter\_by(tid = tid).first()

db.session.add(notification)

db.session.commit()

return jsonify({"message": "added successfully" , "statuscode": 200, "insertedNotification": notification.nid, "teacher": teacher.name})

1. createSubmissionRequest

@app.route("/createSubmissionRequest" , methods=["POST"])

def createSubmissionRequest():

*# parameters required*

*# class, priority, title, tid*

request\_data = request.get\_json()

targetClass = request\_data["class"]

tid = request\_data["tid"]

desc = request\_data["desc"]

deadline = request\_data["deadline"]

title = request\_data["title"]

submissionRequest = SubmissionRequest(

tid=tid,

title=title,

deadline=deadline,

desc=desc,

classid=targetClass

)

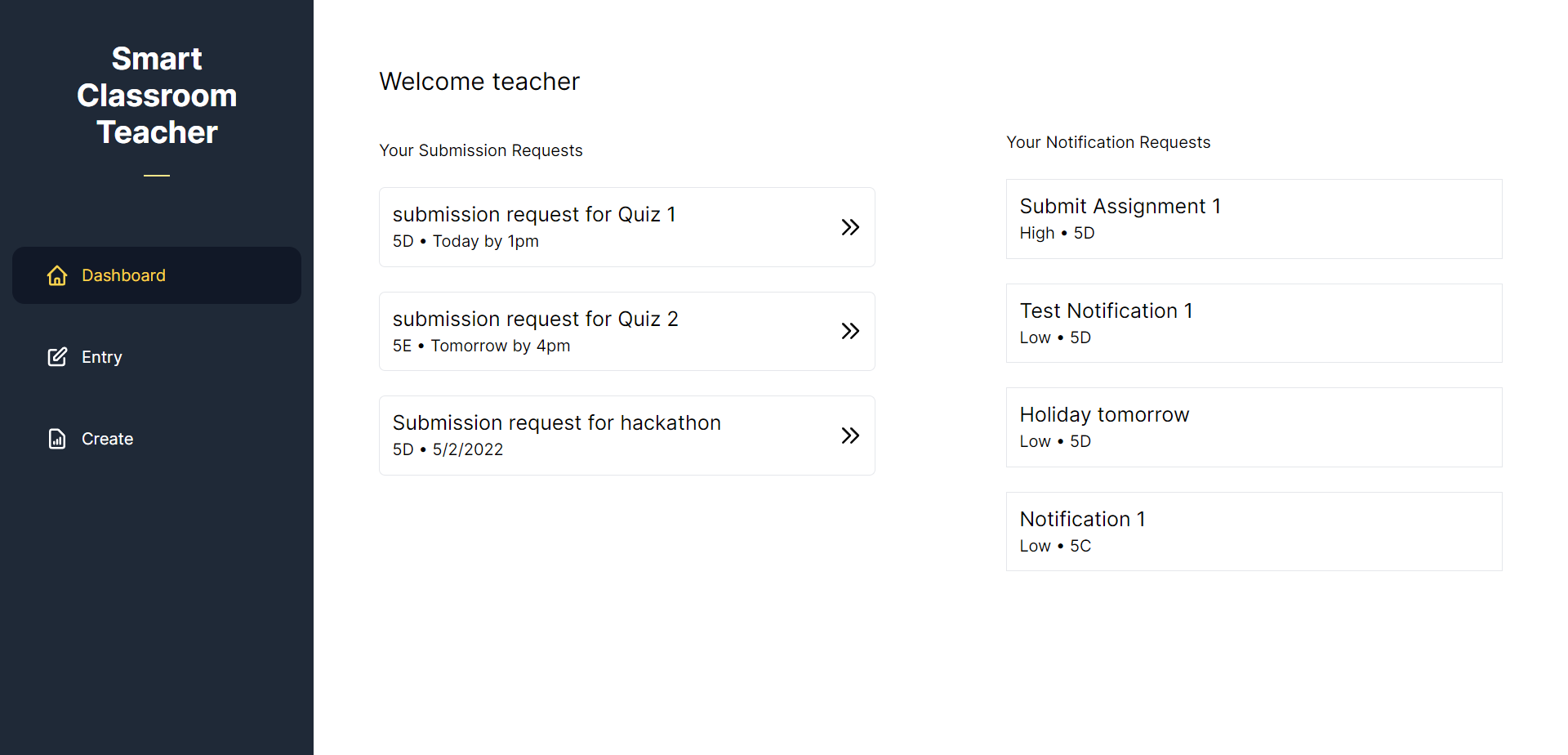
db.session.add(submissionRequest)

db.session.commit()

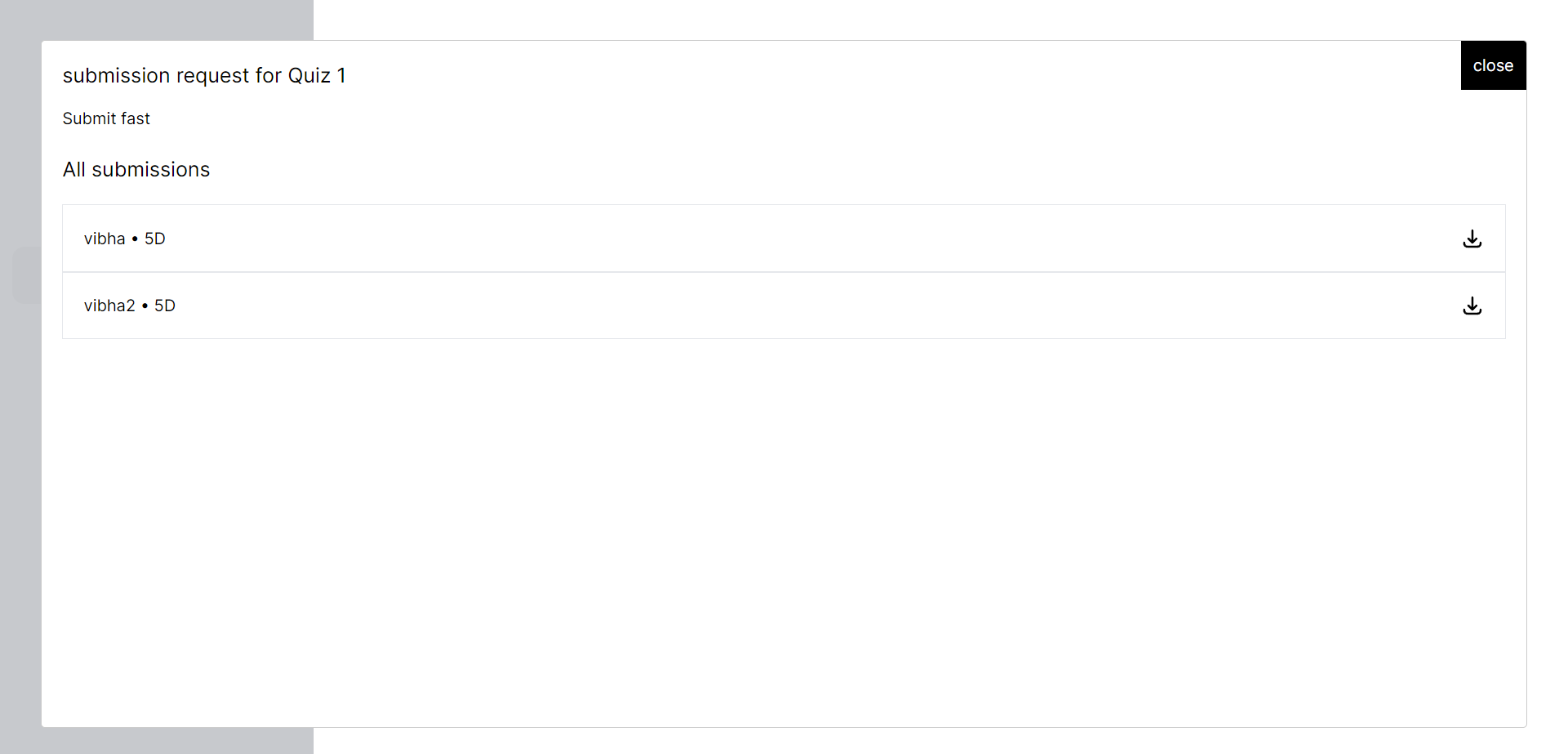
return jsonify({"message": "added successfully" , "statuscode": 200, })

Output

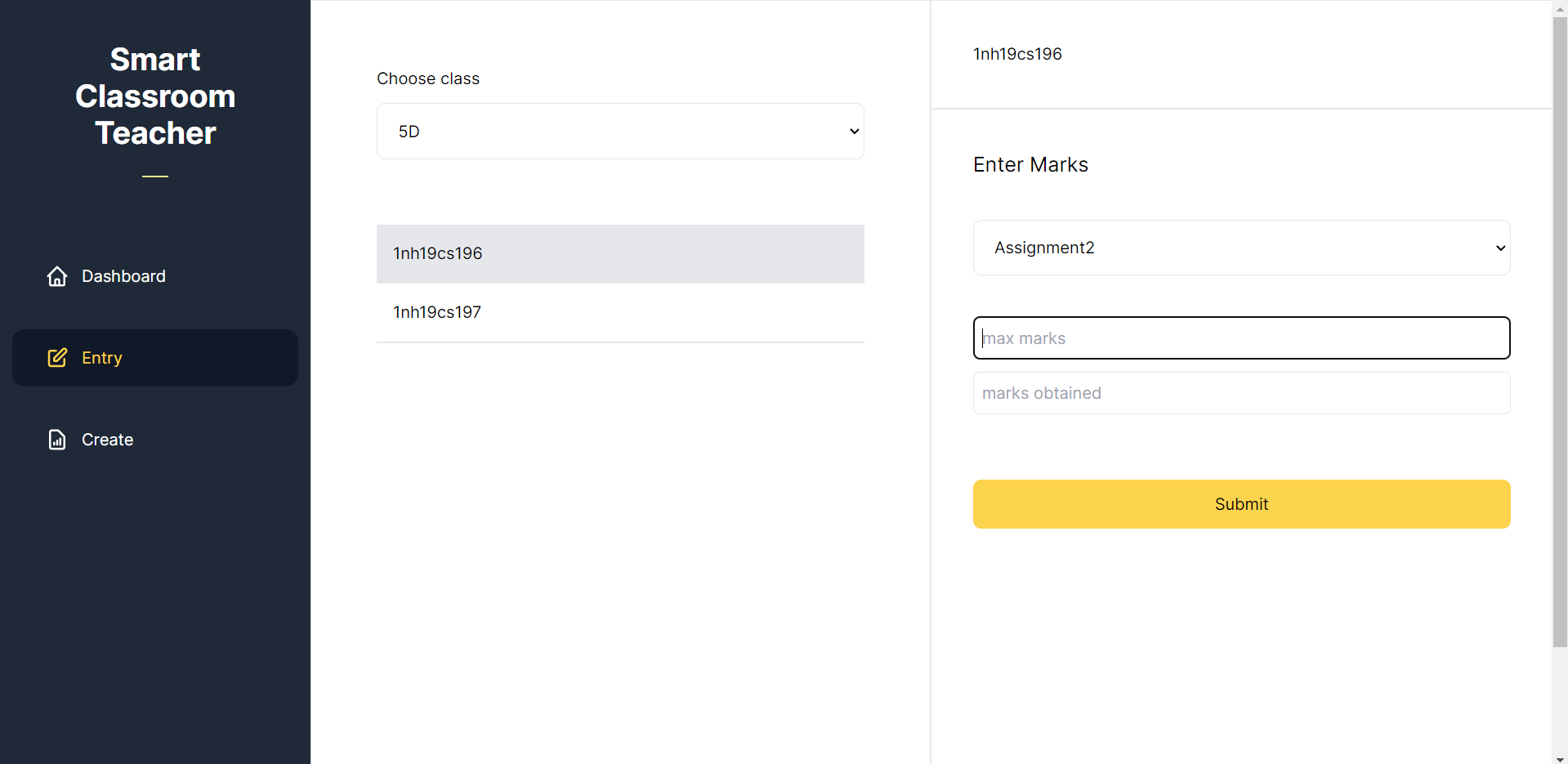
**Teacher Dashboard**



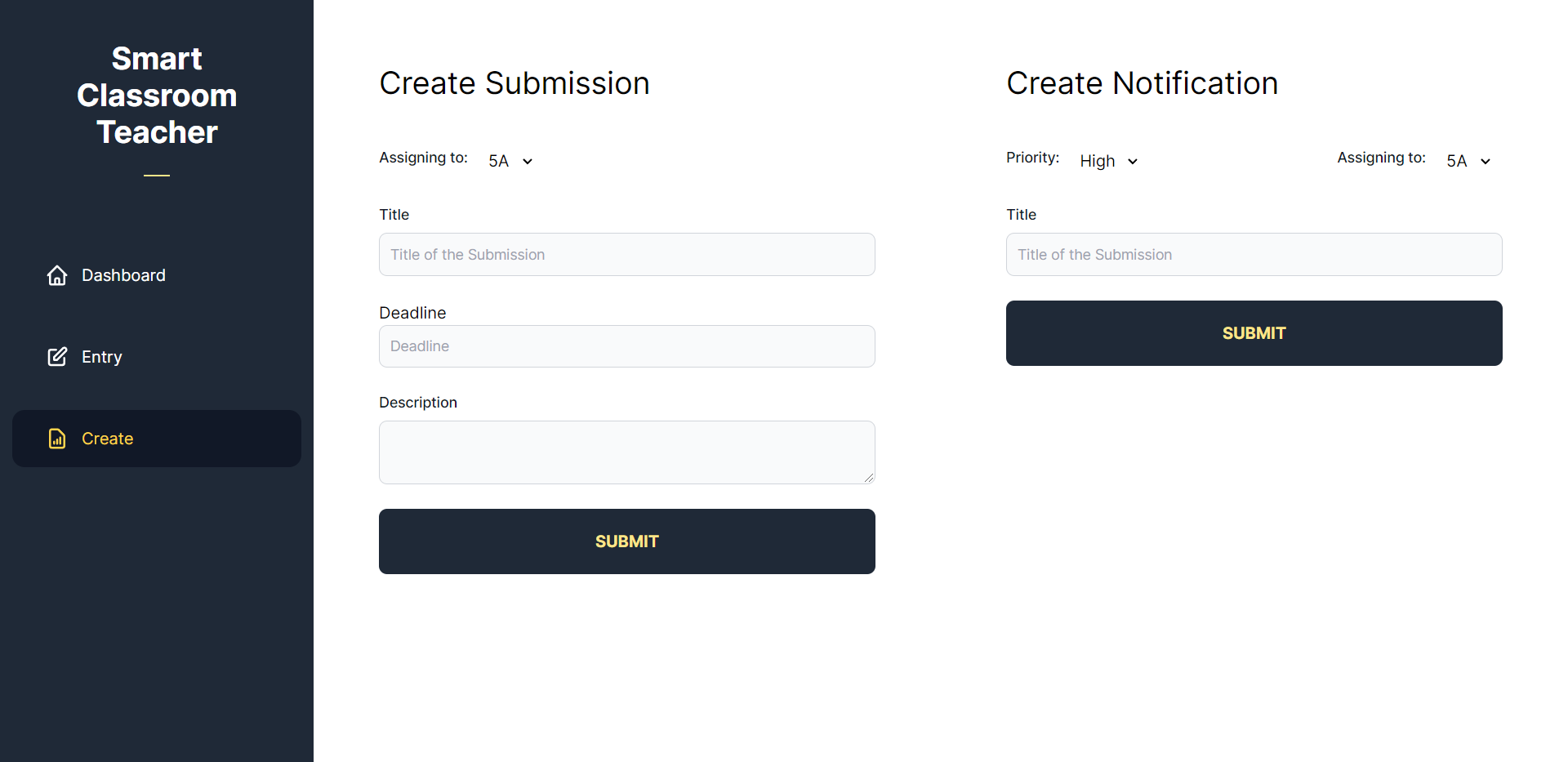
**Submission to all Submission Requests**



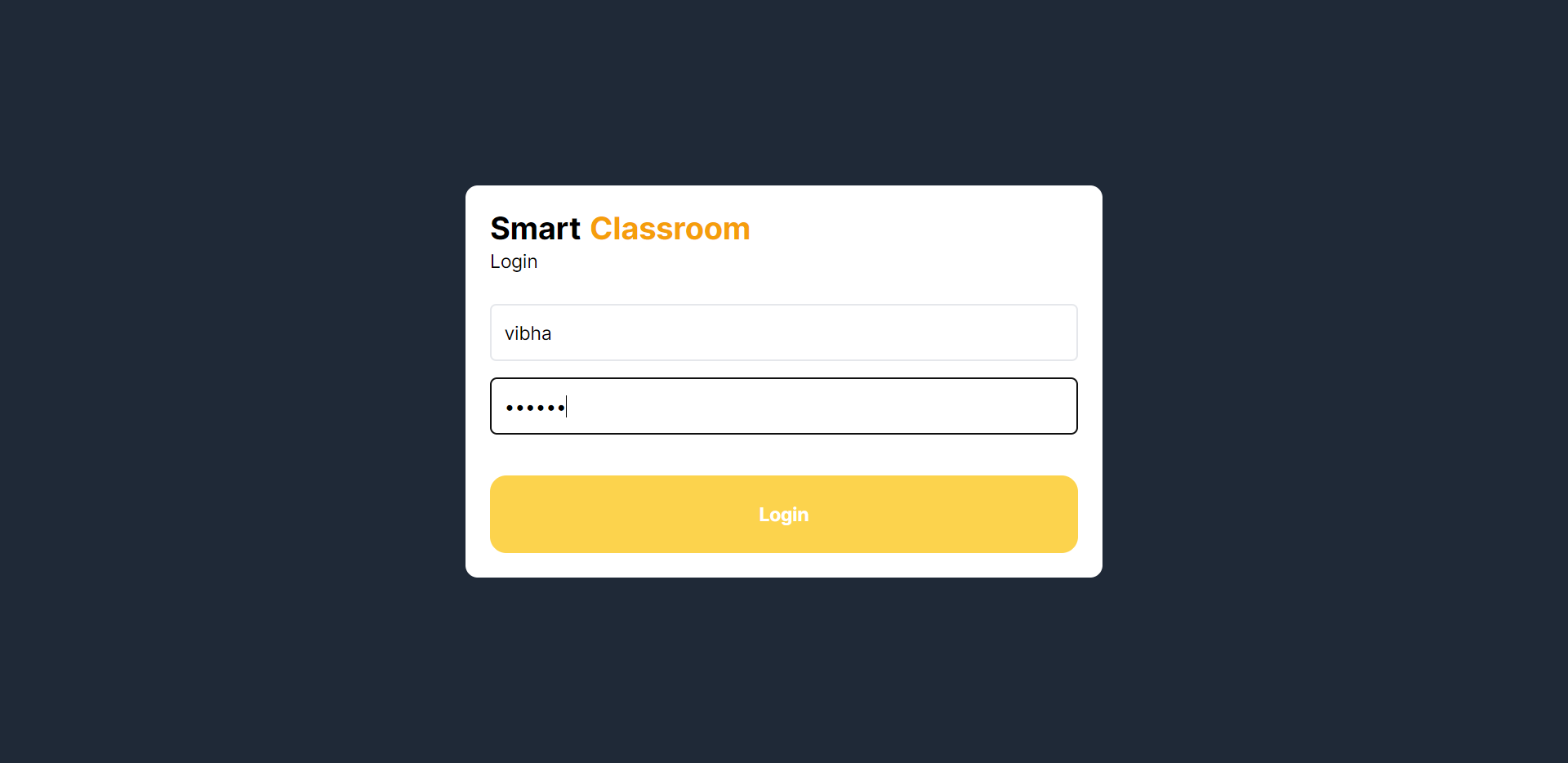
**Teacher Entry Page**



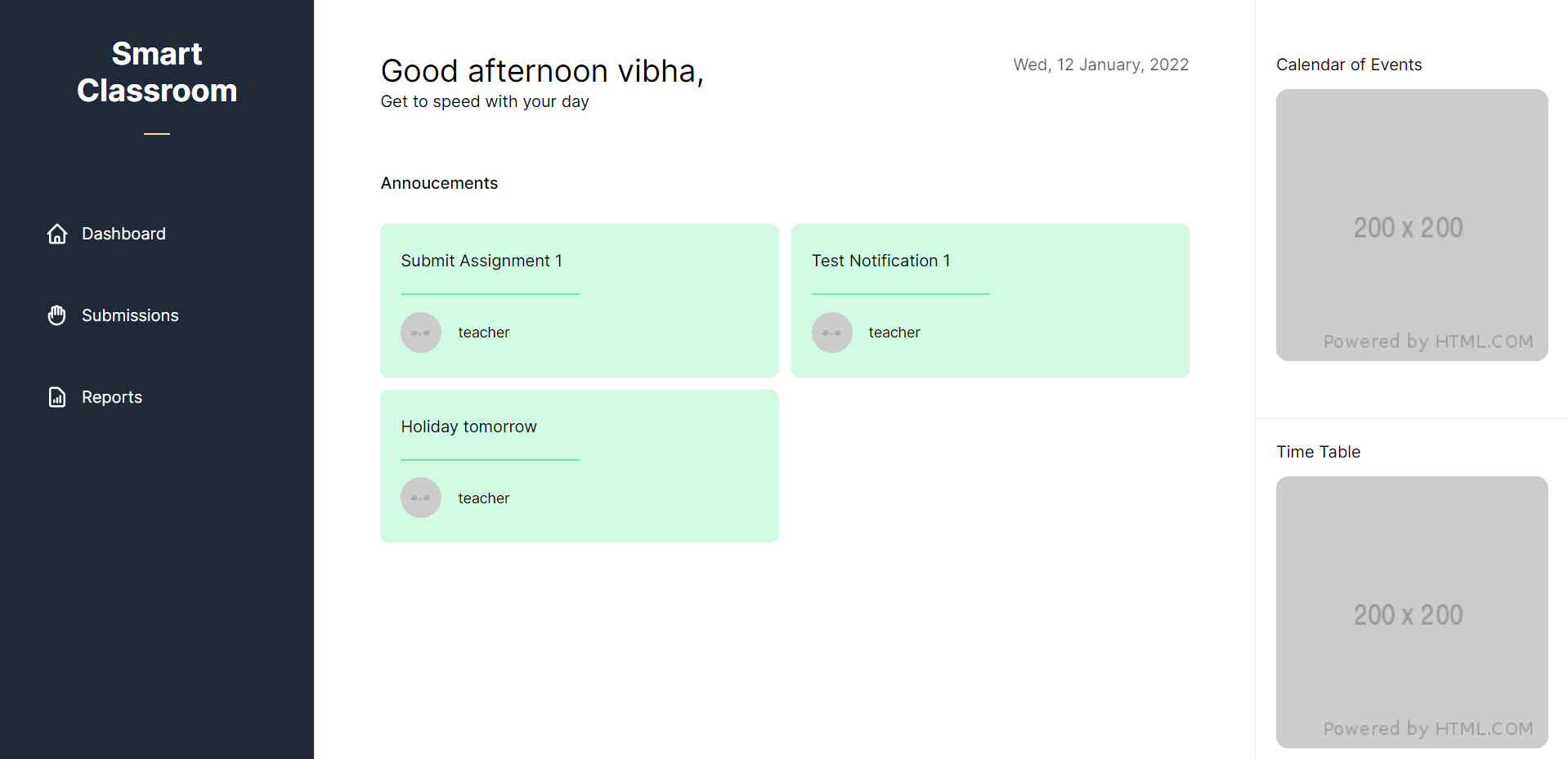
**Teacher Create Page**



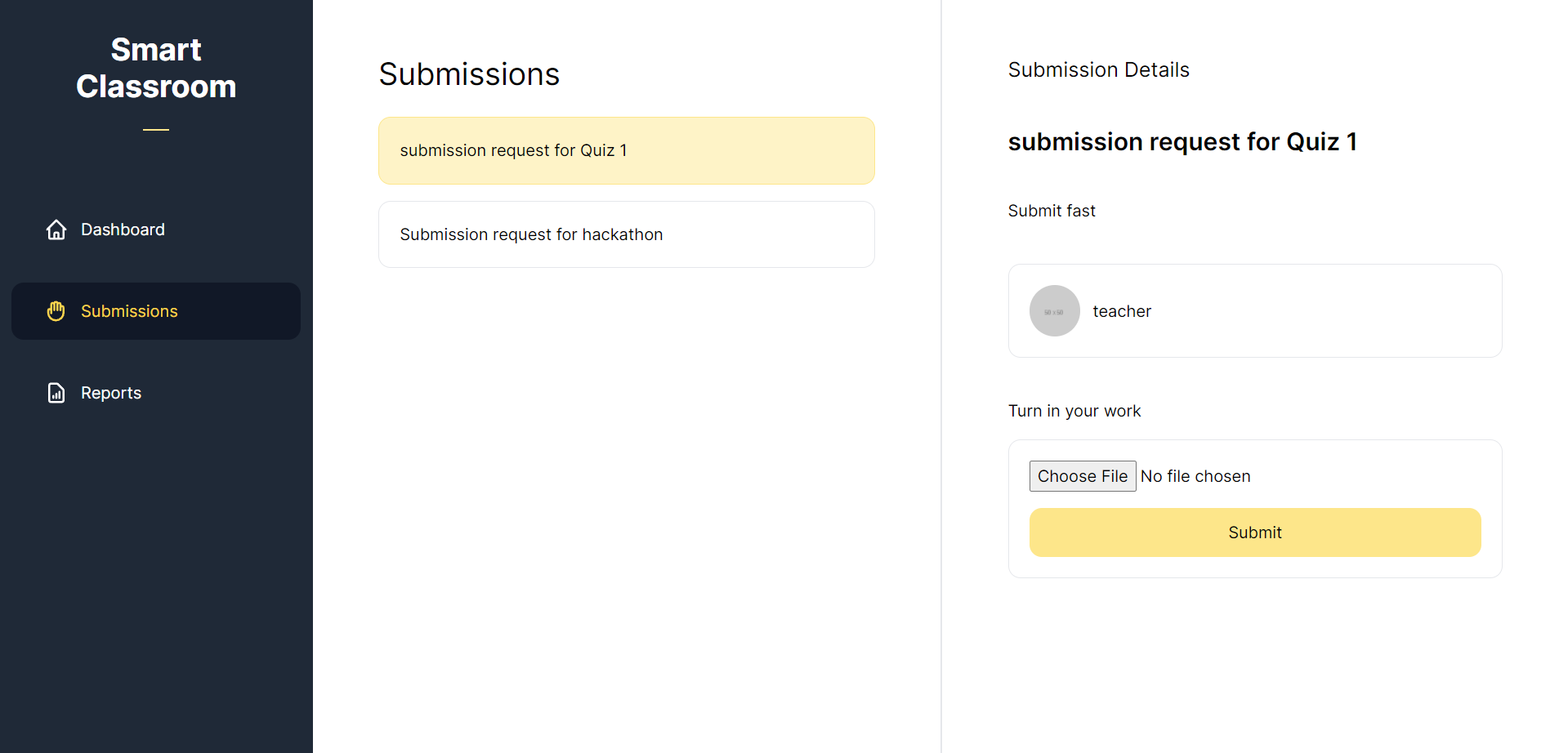
**Student Login Page**



**Student Dashboard**



**Student Submissions Page**



**Student Report Page**

