



## **Sunnydale School District Management System by SheQL**

**Yuyao Ding  
Benita Issac  
Liana Pakingan  
Louisa Stumpf**

# Contents

<b>Executive Summary.....</b>	<b>3</b>
<b>1. Introduction.....</b>	<b>3</b>
1.1 Background.....	3
1.2 Problem Statement.....	3
1.3 Objective of the Application.....	3
1.4 Overview.....	4
<b>2. Application Overview.....</b>	<b>4</b>
<b>3. Database Design.....</b>	<b>5</b>
3.1 Operational Database.....	5
3.1.1 Purpose and Structure.....	5
3.2 Analytical Database.....	6
3.2.1 Purpose and Structure.....	6
3.2.2 ETL (Extract, Transform, Load) Process.....	7
<b>4. Data Sources.....</b>	<b>8</b>
4.1 Description of Data Used.....	8
4.2 How the Data Was Created.....	8
4.3 Volume and Variety of Data.....	8
<b>5. Functionality and Specifications of Application.....</b>	<b>9</b>
5.1 Login Page.....	9
5.1.1 Login.....	9
5.2 Guardian Portal.....	9
5.2.2 Student Grades.....	10
5.2.3 Attendance.....	10
5.2.4 Contact Teacher.....	10
5.3 Student Portal.....	10
5.3.1 Dashboard.....	10
5.3.2 Grades.....	10
5.3.3 Profile.....	11
5.4 Teacher Portal.....	11
5.4.1 Dashboard.....	11
5.4.2 Grade Management.....	12
5.4.3 Attendance.....	14
5.4.4 Communication.....	15
5.4.5 Analytics.....	16
5.5 Administrator Portal.....	16
5.5.1 Dashboard.....	16
5.5.2 Executive Dashboard.....	16
<b>7. Challenges and Lessons Learned.....</b>	<b>16</b>
<b>8. Conclusion.....</b>	<b>17</b>
8.1 Summary of Accomplishments.....	17
8.2 Benefits to School District.....	17
8.3 Possible Future Features.....	17
<b>9. Appendices.....</b>	<b>18</b>
9.1 Files Used.....	18
9.2 Operational Database Queries.....	19
9.3 Analytical Database Queries.....	20

# Executive Summary

This report outlines the development of a school district management application created for a database technologies course. The application supports students, guardians, teachers, and administrators by providing a centralized platform to manage academic data, including courses, grades, attendance, and user roles.

The system includes two databases: an operational database for real-time updates and user interactions, and an analytical database for reporting and analysis. The operational database uses a denormalized relational structure to improve performance. The analytical database follows a star schema, centered on student attendance and performance, enabling insights across schools, teachers, and time.

A custom Python-based ETL pipeline moves data between systems, handling cleaning, transformation, and loading efficiently. The application uses simulated data—over 120,000 attendance records and 90,000 grades—to demonstrate functionality. Screenshots, schema diagrams, and queries are included in the appendix.

This project demonstrates key concepts in database design, ETL processes, and data-driven application development.

## 1. Introduction

### 1.1 Background

In today's data-driven educational environment, the effective management and analysis of student data is critical to running schools efficiently. Schools routinely collect information such as grades, attendance records, and course schedules, but this data is often fragmented across multiple systems. In school districts, where multiple schools are managed under a single administrative body, these challenges are amplified. Without a centralized platform, it becomes difficult to monitor academic performance across campuses, provide timely support to students, or make data-informed decisions at the district level. A connected, user-friendly system is needed to help all stakeholders access and manage academic data in one place.

### 1.2 Problem Statement

Although a variety of educational software tools exist, most are designed with a narrow focus. Supporting only a single user group such as students or teachers. Few offer comprehensive features that address the needs of all key stakeholders within a school district. As a result, gaps in communication persist, delays in responding to student needs occur, and opportunities for data-driven decision-making are missed. The lack of a unified, centralized platform leads to operational inefficiencies and limits the district's ability to support student success at scale.

### 1.3 Objective of the Application

The objective of our application is to provide a centralized platform that enables students, guardians, teachers, and district administrators to efficiently track and manage academic performance. The system is designed to support both day-to-day operations and long-term

strategic planning by combining an operational database for real-time data management with an analytical database for performance analysis and insights.

## 1.4 Overview

This report outlines the design and development of the Sunnydale School District Management Application created as part of our team project. It begins with a detailed description of the application's purpose and functionality. Next, it presents the database structure, including both the entity-relationship (ER) diagram and the relational and analytical schemas. The report also discusses the sources and types of data used, key queries that demonstrate the system's capabilities, and sample user scenarios supported by screenshots of the interface. Finally, the paper concludes with reflections on the project's challenges, key takeaways, and potential future enhancements to the system. Additional technical materials such as full query listings, schema diagrams, and extended screenshots are included in the appendix.

## 2. Application Overview

The school district management application is a database-driven platform designed to help students, guardians, teachers, and administrators efficiently manage academic data across multiple schools within a district. Built with a graphical user interface (GUI), an operational database, and an analytical database, the system supports both real-time functionality and data-driven reporting.

The application serves as a centralized hub for school operations. It allows users to log in based on their role and access relevant features — for example, teachers can manage grades and attendance, students can view their academic records, guardians can monitor their child's progress, and administrators can oversee district-wide performance. Each user's experience is tailored to their role through role-based access control and personalized views.

The GUI provides an intuitive interface for interacting with the underlying data. Common operations include:

- Adding, viewing, and modifying student, teacher, guardian, and administrator profiles
- Enrolling students in courses and assigning teachers
- Recording attendance and entering grades

The operational database supports day-to-day activities by storing real-time data that updates with each user interaction. In contrast, the analytical database is optimized for querying and reporting. It uses a star schema structure to track academic performance across students, schools, and courses, enabling administrators to generate insights such as average grades by school or attendance trends by teacher.

Together, the system's front end and databases demonstrate an integrated approach to managing educational data. The application not only simplifies routine academic management but also lays the groundwork for smarter decision-making at the district level through analytics and structured reporting.

## 3. Database Design

### 3.1 Operational Database

#### 3.1.1 Purpose and Structure

The operational database is designed to support the day-to-day functions of the application. It stores real-time data that is frequently updated as users interact with the system. This database ensures that students, teachers, guardians, and administrators have immediate access to the most current information relevant to their roles. Its primary goal is to provide fast, reliable, and consistent performance for routine operations such as entering grades, assigning courses, tracking attendance, and managing user records.

The operational database uses a relational model but does not fully adhere to normalization rules. Instead, the design prioritizes simplicity, reduced query complexity, and faster development. While some redundancy is present, this tradeoff was intentional: fully normalizing the database would have significantly increased the number of joins required in queries, which could impact both performance and maintainability, especially as more tables and relationships were added.

Tables are still structured with primary and foreign keys to preserve basic referential integrity and ensure data is consistent across key relationships.

Table Name	Description	Records
user	Stores login credentials and role information for all users, including students, teachers, guardians, and administrators.	8221
student	Contains student records with attributes such as name, date of birth, grade level, and school.	3000
guardian	Includes guardian details and links to students through a separate relationship table, allowing for multiple guardians per student.	5000
teacher	Holds teacher information, including name, salary, and assigned school.	200
admin	Contains administrator records, including employment information and associated school(s) or district.	10
school	Represents schools within the district, with attributes like name, type (e.g., high school, middle school), and location.	10
district	Lists district-level data and links to its assigned administrators.	1
course	Stores course identifiers and course names.	8
teaches	Links teachers to the courses they instruct.	3500
takes	Tracks course enrollments by students.	90000
attendance	Records student attendance by date and teacher.	123000

grade_details	Stores grades for each student-course combination.	90000
guardian_student_relationship	Tracks the relationships between students and their respective guardians	4523

Table 1: Operational Database Tables

While the lack of full normalization introduces some data redundancy, it also simplifies the structure, making it more accessible for new developers and easier to implement in a time-constrained project. This tradeoff was acceptable for the project's scope and objectives.

## Entity Relationship Diagram

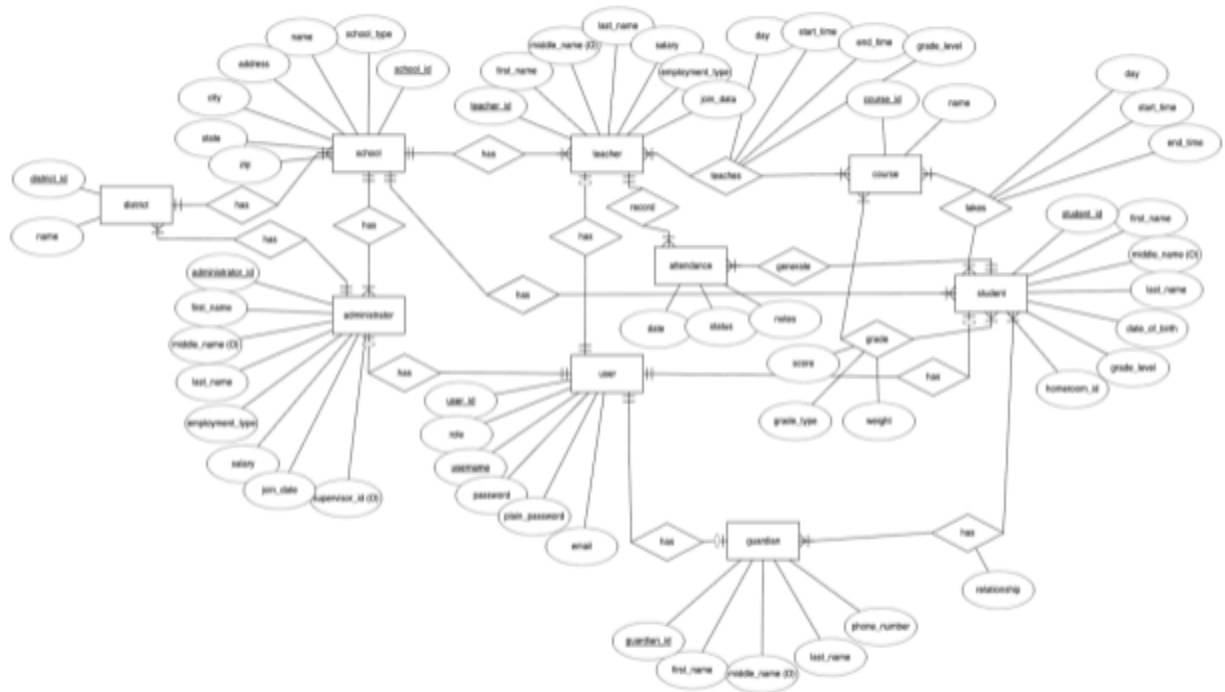


Figure 1: Entity Relationship Diagram

## 3.2 Analytical Database

### 3.2.1 Purpose and Structure

To support advanced reporting and district-wide insights, our application includes a dedicated analytical database built on a star schema architecture. In a star schema, central fact tables hold measurable events, while surrounding dimension tables provide descriptive context. This denormalized structure significantly reduces the complexity of joins and enhances query performance, particularly for business intelligence and educational analysis tools. Our analytical schema is centered around two key fact tables accompanied by five dimension tables all listed below.

Table Name	Type	Description
student_performance_fact	Fact	Stores student test scores and assessment results, supporting the evaluation of academic outcomes across courses, teachers, and time periods.
student_attendance_fact	Fact	Records daily student attendance, enabling trend analysis across time, teachers, schools, and student demographics.
student_dim	Dimension	Contains one row per student, with attributes such as name, grade level, gender, and race/ethnicity.
teacher_dim	Dimension	Stores teacher-specific details, including full name and primary subject area.
course_dim	Dimension	Describes course-related data such as course ID, name, and academic subject.
school_dim	Dimension	Provides contextual information about each school, including its name, district, state, and location.
date_dim	Dimension	Captures granular time data—day, week, month, and year—to support flexible temporal analysis.

Table 2: Analytical Database Tables

### 3.2.2 ETL (Extract, Transform, Load) Process

The ETL pipeline, developed in Python using SQLAlchemy, automates the process of transforming operational data into a format suitable for analytical use. It extracts raw data from the operational database, applies necessary transformations, and loads it into the analytical schema. The pipeline uses an `etl_control` table to track the timestamp of the last successful execution. This enables the system to extract only new or modified records, improving efficiency and supporting near-real-time updates. For initial data loads and testing, the system can truncate all fact and dimension tables in a dependency-safe order. Foreign key constraints are temporarily disabled to allow clean reloading, then restored upon completion. The data from the operational database is transformed to suit the needs of the analytical database. Student demographic values and state abbreviations are standardized. Attendance and performance records are joined with related dimensions to create fully enriched fact rows. Duplicate or orphaned records are automatically identified and excluded or logged. Bulk inserts are used to load data efficiently. Dimension tables are populated first to ensure referential integrity, followed by the fact tables. After loading, foreign key constraints are reactivated to enforce data consistency. Each load is validated with row counts, and a summary log is recorded. Any errors or anomalies are logged in the `etl_control` table, providing transparency and traceability. In benchmark testing, the ETL process handled over 6.8 million records in roughly 12 minutes. This performance demonstrates the pipeline's capacity to scale with increased data volume,

making it suitable for supporting real-time dashboards, longitudinal studies, and district-wide reporting initiatives.

## Star Schema

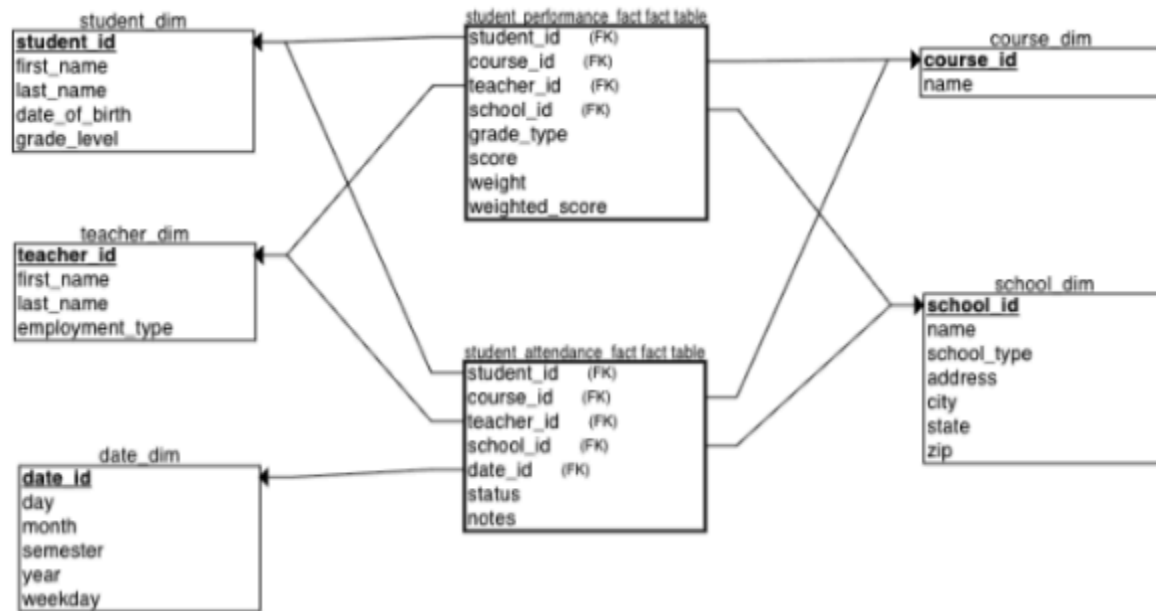


Figure 2: Star Schema

## 4. Data Sources

### 4.1 Description of Data Used

The data used in this project represents the fundamental components of a typical school district system. It includes information about users (students, teachers, guardians, and administrators), schools, districts, courses, course enrollments, attendance records, and grades. This data is used to simulate realistic scenarios for testing and demonstrating the application's features, including academic tracking, attendance monitoring, and course performance reporting.

### 4.2 How the Data Was Created

All data used in the project was synthetically generated using a custom Python script (data.py). The script was designed to simulate a realistic school district environment by producing varied and interrelated records for all core entities. Relationships such as student-guardian links, course enrollments, and attendance logs were programmatically defined to match the database schema and ensure consistency and referential integrity across tables.

### 4.3 Volume and Variety of Data

The dataset was designed to reflect the scale of a mid-sized school district. It includes a diverse



mix of entity types and relationships to support meaningful analytical queries. Table 1 summarizes the volume of data stored in each major table of the operational database. This synthetic dataset allowed for robust testing of both the operational and analytical features of the application without exposing any real-world sensitive information.

## 5. Functionality and Specifications of Application

### 5.1 Login Page

#### 5.1.1 Login

To initialize the management system, the Jupyter Notebook **main.ipynb** should be run. Upon launching the application, a user is shown the login screen, where they are prompted to enter their username and password to access their account. No 'Create Account' feature is available as all accounts are created by the district upon the enrollment of the student and employment of the teacher. Logistically, guardian accounts are also only created and associated with their respective students upon the creation of student accounts. There are four roles assigned to users based on their status in the district that determine which portal they will be redirected to upon successfully logging in:



Figure 3: Login Screen

**Student:** Users with the 'student' role are students enrolled in Sunnydale school district. Students have the ability to view their course schedule, check their grades, and see personal information regarding their homeroom teacher and guardian.

**Teacher:** Users with the 'teacher' role are teachers employed by Sunnydale school district. Teachers have the ability to see the schedule of courses they teach, manage grades, mark attendance, retrieve guardian contact information, and see analytics of student performance in their courses.

**Guardian:** Users with the 'guardian' role are guardians of students who are enrolled in Sunnydale school district. Guardians have the ability to check their students' grades, view attendance records, and find contact information for their students' homeroom teachers.

**District Admin:** Users with the 'admin' role are district administrators of Sunnydale school district. District administrators have access to school-level and district-level analytics regarding attendance and exam grades, allowing for a comprehensive overview of student performance in the district as a whole.

### 5.2 Guardian Portal

To login to the guardian portal, the user **gua\_m.lawrence** with password **2!b>GIN8j&X** can be used.

#### 5.2.1 Dashboard

Provides an overview of the students associated with the logged-in guardian. After selecting a

student, the dashboard automatically loads that student's academic information, including grades, attendance, and teacher details, redirecting the guardian to the student's grades.

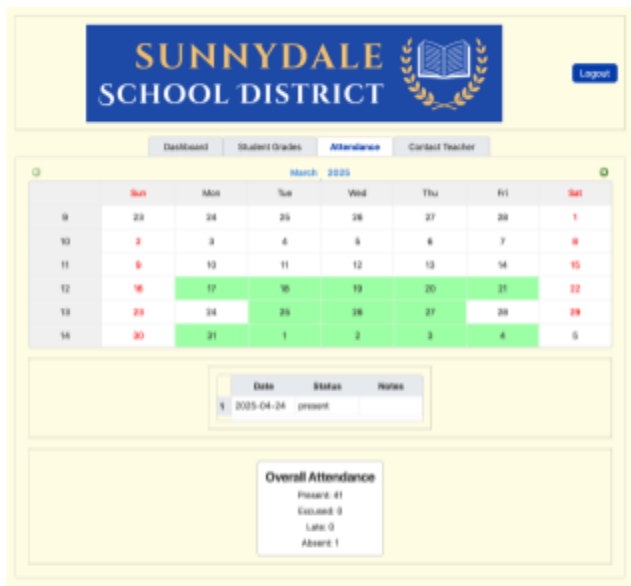


Figure 4: Attendance View

### 5.2.2 Student Grades

Displays a summary of the selected student's grades across all enrolled classes. This section gives guardians a clear view of academic performance at a glance.

### 5.2.3 Attendance

Presents the attendance record of the selected student in a calendar format. Days marked **present** appear in green, while **absences** are shown in red. A summary snapshot below the calendar highlights overall attendance trends.

### 5.2.4 Contact Teacher

Shows information about the student's homeroom and primary contact teacher.

While in-app messaging is not yet available, guardians can view the teacher's email address for direct communication.

## 5.3 Student Portal

To login to the student portal, the user **stu\_jphillip58** with password **%9(n{siK8?G]** can be used. A successful login by a user with a 'student' role will initialize the Student Portal of the management system, which consists of three different pages (Dashboard, Grades, Profile). Each page is accessible through its respectively named navigation tab.

### 5.3.1 Dashboard

The dashboard page is the first page that a student user sees upon entering the Student Portal. The page consists of a student's personal information, such as their full name and student ID, and a schedule table of all courses they are enrolled in. A calendar is also provided with the current date marked. From the dashboard, a student can navigate to view their grades or profile through the navigation tabs.

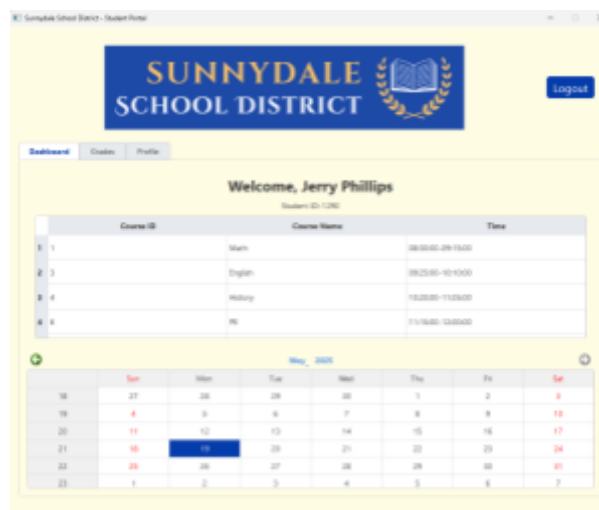
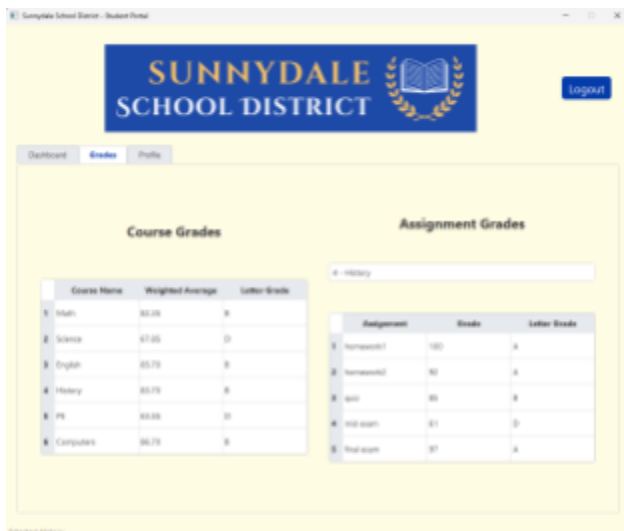


Figure 5: Student Portal Dashboard

### 5.3.2 Grades

On the grades page, students can view their overall letter grade for each course and their assignment grades by course. The 'Assignment Grades' table can be changed to display different course grades by selecting the desired course in the dropdown menu.



Course Name	Weighted Average	Letter Grade
Math	82.25	B
Science	87.25	D
English	85.75	B
History	85.75	B
PE	89.00	D
Computers	86.75	B

Assignment	Score	Letter Grade
Homework 1	100	A
Homework 2	90	A
Test	85	B
Mid-term	81	D
Final exam	87	A

Figure 6: Student Portal Grades

- Homeroom teacher information includes the teacher's full name and email address.
- Student information includes the currently logged in student's full name, date of birth, and grade level.
- Guardian information includes the guardian of the student's full name and phone number.

### 5.3.3 Profile

On the profile page, students can view information regarding their homeroom teacher, themselves, and their main guardian. This information immediately loads into the profile upon logging in.



Homeroom Teacher Info
Eric Yu tea.eric.yu@wms-zhang.com

Student Info
Jerry Phillips DOB: 2014-07-04 Grade Level: 5

Guardian Info
Timothy Phillips (984) 252-5700

Figure 7: Student Portal Profile

## 5.4 Teacher Portal

To login to the teacher portal, the user **tea\_dana.cannon** with password **d8j]6!w8:\*ne** can be used. A successful login by a user with a 'teacher' role will initialize the Teacher Portal of the management system, which consists of five different pages (Dashboard, Grade Management, Attendance, Communication, Analytics). Each page is accessible through its respectively named navigation tab. **Note:** Fullscreen may be needed to account for UI resolution differences.



Course ID	Course Name	Grade Level	Class Times	Days
1	Math	5th	11:00AM - 12:00PM	MTWTFSS
2	Science	5th	12:00PM - 1:00PM	MTWTFSS

Mon	Tue	Wed	Thu	Fri	Sat	Sun
27	28	29	30	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

### 5.4.1 Dashboard

The dashboard page is the first page that a teacher user sees upon entering the Teacher Portal. The page consists of a teacher's personal information, such as their full name and teacher ID, and a schedule table of all courses they teach. A calendar is also provided

with the current date marked. From the dashboard, a teacher can navigate to other pages of the portal using either the navigation tabs at the top or the buttons at the bottom of the screen. The two buttons, 'Go to Grade Management' and 'Mark Attendance', redirect the user to the grade management or attendance pages when clicked, respectively.

#### 5.4.2 Grade Management

On the grade management page, teachers can view and manage grades of students in any of the courses they teach.

**Viewing grades:** To view grades, teachers select one of their courses in the 'Select course' dropdown menu. Once a course is selected, the 'Select student' dropdown menu will populate with the names of all students in the selected course. Selecting a student will populate the table with all their assignment/exam grades and display their calculated weighted letter grade/percentage in the course.



Figure 9: Selecting a course

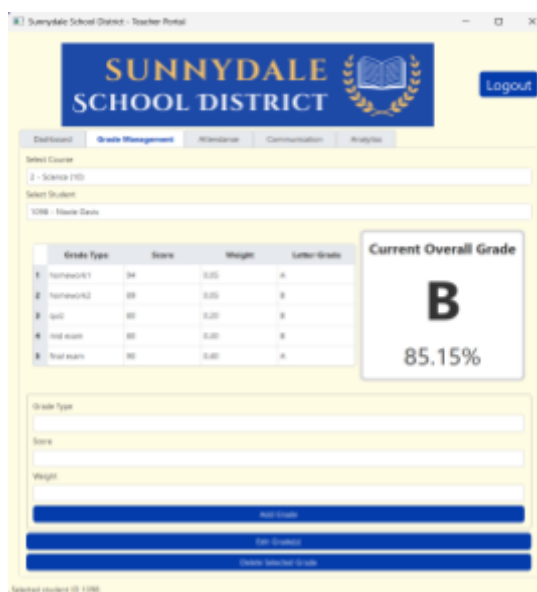


Figure 10: Selecting a student

**Adding grades:** To add a grade, teachers enter information for the grade record, such as the grade type, the student's score, and the weight of the entered assignment. Currently, only five grade types are supported as usable names for grade additions: homework1, homework2, quiz, mid exam, and final exam. A key detail to note is that because only five grade types are supported, if a user tries to enter a grade record with the name of a grade type that already exists the entry will fail. Therefore, a teacher may need to delete a grade record before adding a new one of the same name. If all information has been entered correctly (i.e. the grade type is one of the above listed names and does not exist, the score/weights are valid integer entries), the graded assignment will be added to the system and immediately reflected on the page. The student's grade table will repopulate and their calculated grade percentage will adjust to account for the new grade addition. A student's letter grade may also change depending on the effect of the added grade.

Figure 11: Entering grade details

Figure 12: Grade screen with newly updated table and grade

**Deleting grades:** Grade deletion is conducted by selecting the row of the particular grade that is to be deleted in the grade table. A teacher can either click on the number of the row or the name of the grade itself in the 'Grade Type' column. Once one grade is selected, a teacher can select the 'Delete Selected Grade' button located at the bottom of the page. A confirmation window will open asking the teacher to confirm that they would like to proceed with the deletion of the selected grade. If the teacher proceeds with the grade deletion, the grade will be removed from the system. The student's grade table will repopulate without the deleted grade and their calculated grade percentage will reflect the removal. A student's overall letter grade may also change depending on the effect of the deleted grade.

Figure 13: Selecting a grade for deletion

Figure 14: Grade screen with updated grade table and grade

**Edit grades:** Teachers are able to edit grades that have already been saved into the system. One or more grade scores can be edited at a time. To edit the score of a graded

assignment/exam, the teacher can click on the cell of the score to be modified. Once the cell of the chosen score is selected, the teacher can enter a new score (an integer between 0 and 100), pressing 'Enter' when finished to ensure that the cell keeps the new value. Once the teacher makes all the changes to the scores they wanted, they can press the 'Edit grade(s)' button to save the new score values. Once pressed, a window will pop up confirming that the grades have been edited successfully, meaning that the grade records have been updated in the system. The student's grade table will reflect the newly updated grades, with letter grades changing if the new scores warrant them to. A student's overall letter grade may also change depending on the effect of the edited grades.

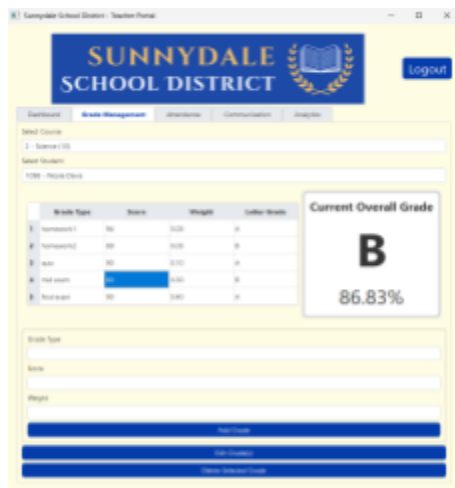


Figure 15: Selecting the score of the grade to edit

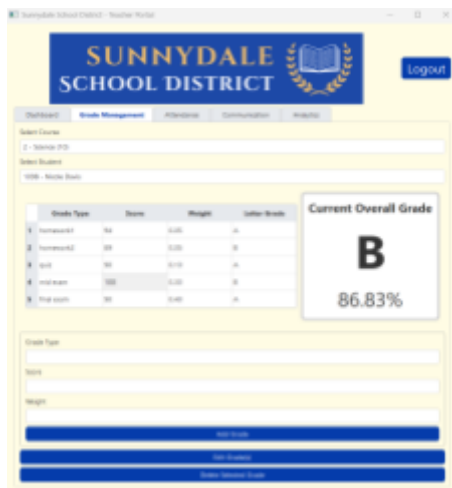


Figure 16: Editing the score in the cell of the

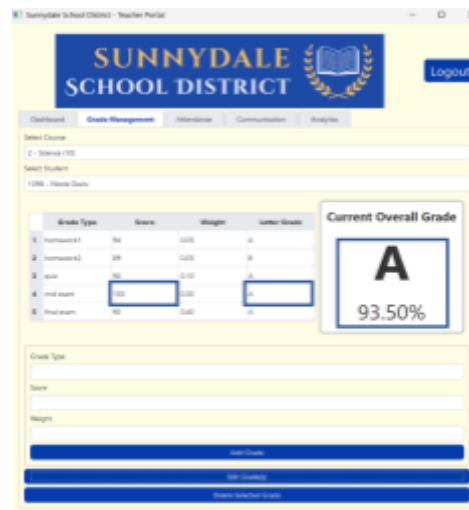


Figure 17: Grade screen with updated grade table and

### 5.4.3 Attendance

On the attendance page, teachers can view historical attendance information for their courses by day and take the current day's attendance.

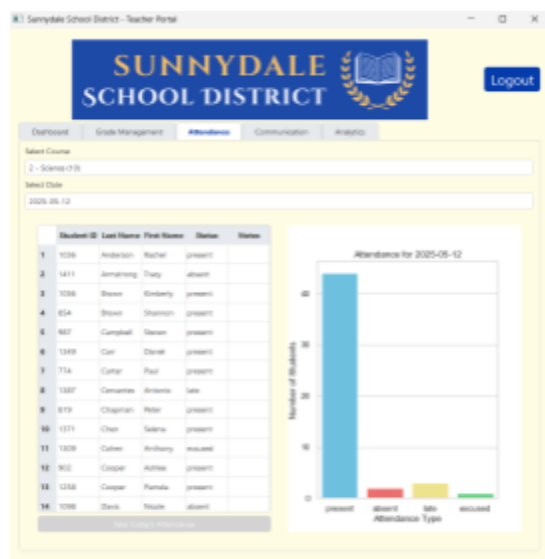


Figure 18: Attendance screen displaying a record from 5/12

**Viewing attendance:** Attendance records are sorted by date. To view the attendance record of a particular date, the teacher can select one of their courses from the 'Select Course' dropdown menu. Once a course is selected, a list of dates will populate in the 'Select Date' dropdown menu. When a date is selected, the attendance record table will populate with all the attendance records of every student in all the teacher's courses for the chosen date. Information regarding whether a student was present, absent, late, or excused for the chosen day is available along with any notes detailing the attendance status. In addition to the table, the teacher is also presented with a bar chart visualizing that day's distribution of attendance for all students in all courses.

**Marking attendance:** Teachers are able to mark attendance if it has not yet been taken for the current day. The 'Take today's attendance' button at the bottom of the page will be clickable if the teacher has yet to take attendance. If a teacher is wanting to take attendance for the day, they can click the button and the page will be reformatted into having a new attendance table. This new attendance table is similar to the viewing table with the exception that it has two checkboxes for each student to mark them absent or late. Checking either box will result in the student's status being one or the other, with neither box being checked resulting in a default 'present' status being assigned. In addition to marking attendance status, the teacher can also enter notes for each student by selecting their respective cell in the 'Notes' column and typing in the information, pressing 'Enter' when finished to ensure that the cell keeps the note. Once the teacher has recorded everything they need, they can press the 'Take today's attendance' button to record the current day's attendance in the system. After its successful recording, the new attendance record will populate the initial attendance table and a new chart will be displayed.

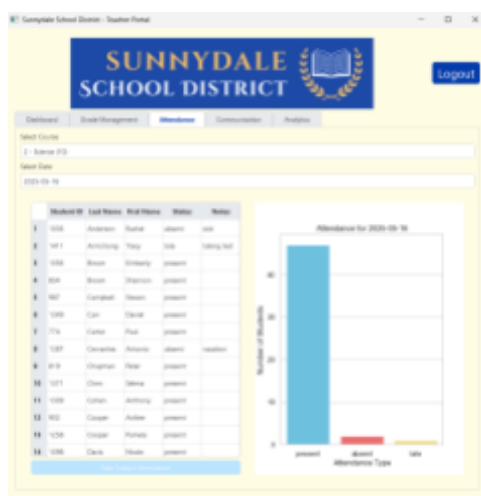


Figure 19: Pressing the 'Take Today's Attendance' button



Figure 20: Filling in the attendance table

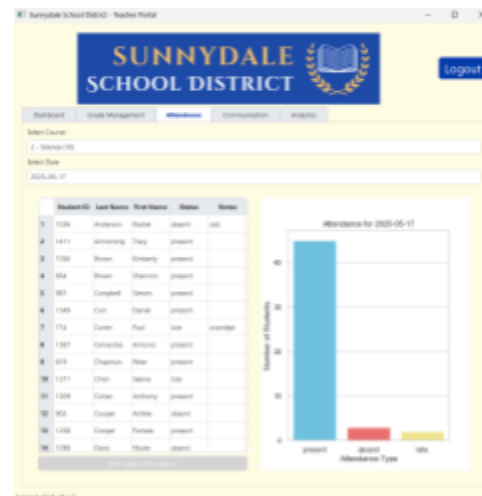


Figure 21: Attendance screen with new updated table/chart

## 5.4.4 Communication

On the communication page, teachers can find guardian contact information for each student. In the future, we hope to implement in-application messaging functionality on top of the ability to view contact information.

**Viewing guardians:** Guardian information is listed by student. All students from all the teacher's courses will be listed in the 'Select Student' dropdown menu. Teachers can select any student from the dropdown to see their guardian(s). Every guardian is listed with their name, phone number, and email address.



Figure 22: Communication screen showing contact



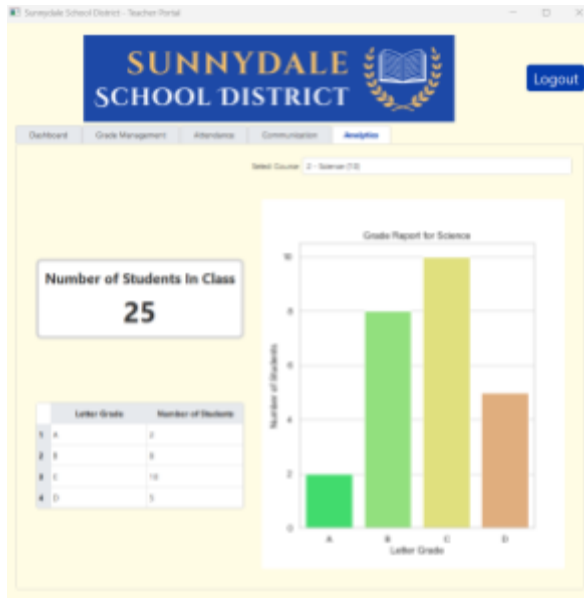


Figure 23: Letter grade table and chart of distribution

#### 5.4.5 Analytics

On the analytics page, teachers can visualize their students' academic performance by course.

**Viewing grade breakdowns by course:** Teachers can access each of their courses' grade reports by selecting the course from the 'Select Course' dropdown menu. The number of students in the course will be displayed along with a table showing the letter grade breakdown for the course. A bar chart of the letter grade distribution is also viewable on the page and dynamically changes to whichever course is selected at a given time.

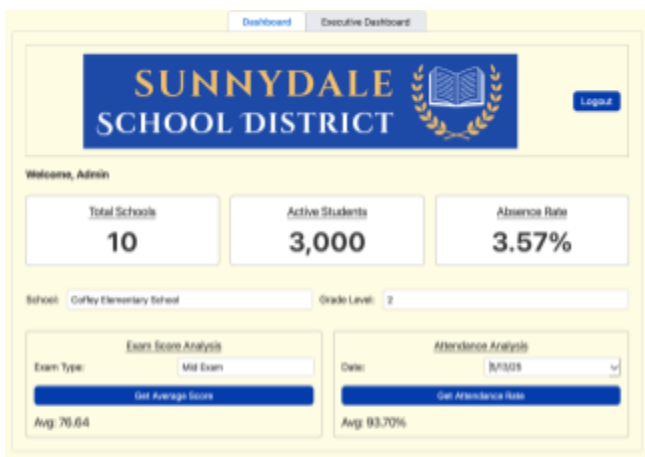


Figure 24: Admin Dashboard

### 5.5 Administrator Portal

To login to the admin portal, the user **dis\_mwilkins** with the password **V;ew\$ES6** can be used.

#### 5.5.1 Dashboard

The administrator dashboard provides a centralized view of key performance indicators (KPIs), including the number of active students, total schools in the district, and overall absence rates. It also allows users to drill down into attendance rates by school, grade level, and specific date. Additionally, the dashboard offers insights into academic performance by displaying midterm and final exam results for selected schools and grade levels.

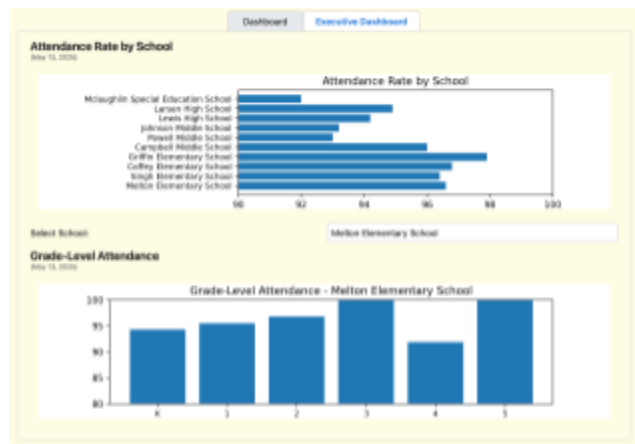


Figure 25: Admin Executive Dashboard

#### 5.5.2 Executive Dashboard

The executive dashboard displays district-wide attendance rates using interactive graphs. A dropdown menu allows users to filter and view attendance rates by grade level within a selected school, enabling more detailed analysis.



## 7. Challenges and Lessons Learned

Developing our school district application presented both technical and collaborative challenges that deepened our understanding of database systems. To avoid performance issues with excessive joins, we decided not to fully normalize our operational schema. While this improved query speed, it required careful data management to maintain consistency and integrity. Designing an effective star schema and building an ETL pipeline from scratch was a major learning experience. We used Python and SQLAlchemy to transform and load data from the operational database into the analytical database. This process taught us how to manage incremental loads, maintain referential integrity, and validate large data transfers. A significant challenge was poor performance in our executive dashboard caused by running complex analytical queries live in the app. We resolved this by generating query results in advance with a Python script and saving them to an `attendance_snapshot` file. The dashboard now loads precomputed data, resulting in faster and more responsive performance. We wrote a Python script to generate all data for testing. Creating realistic relationships and ensuring data quality across tables gave us hands-on experience in data modeling and validation. As a team project, it required consistent collaboration, version control, and task division. We learned the importance of clear communication and aligning on schema design early to avoid integration issues. Overall, the project gave us practical experience in designing, building, and analyzing complex database systems, reinforcing concepts learned throughout the semester.

## 8. Conclusion

### 8.1 Summary of Accomplishments

Over the course of the semester, our team successfully designed and implemented a fully functional school district management application. The system includes a graphical user interface, an operational relational database for real-time data management, and a star-schema-based analytical database for reporting and insights. We developed and executed a Python-based ETL pipeline to move and transform data between the two environments. The application supports core district operations such as managing users (students, teachers, guardians, and administrators), tracking course enrollment, attendance, and grades, and enabling data-driven analysis through dashboards and prebuilt queries.

### 8.2 Benefits to School District

This application provides a centralized platform for school district stakeholders to access and manage critical academic and administrative data. Administrators can monitor student performance and attendance trends across schools. Teachers benefit from streamlined grade and attendance tracking. Guardians gain visibility into their children's academic progress, and students can easily view their course and grade history. The analytical database supports strategic decision-making, allowing for efficient resource allocation, performance benchmarking, and identification of areas needing intervention.

### 8.3 Possible Future Features

There are several potential enhancements that could extend the functionality of the application:

- **Automated Data Syncing:** Real-time or scheduled ETL updates to keep analytical data continuously refreshed.
- **Role-Based Dashboards:** Customized views for students, guardians, teachers, and administrators.
- **Mobile Access:** A responsive or native mobile version for broader accessibility.
- **Attendance Anomaly Detection:** Use of analytics to flag unusual attendance patterns.
- **Integration with External Systems:** Import/export functionality with student information systems (SIS) or state databases.

These features would further strengthen the platform's value to school districts by improving usability, scalability, and decision-making capabilities.

## 9. Appendices

### 9.1 Files Used

No.	Name	Type	Description
1	sheql.ini	Config File	Database Config File (Operational)
2	sheql2.ini	Config File	Database Config File (Analytical)
3	cache.ipynb	Jupyter Notebook	
4	main.ipynb	Jupyter Notebook	Main Application Initialization
5	main.py	Python File	Main Application (Logins + Portals)
6	attendance_snapshot.py	Python File	
7	data201.py	Python	Helper Functions
8	district_dashboard_app.py	Python	District Admin Portal
9	merged_district_dashboard.py	Python	District Admin Portal
10	districtAdmin.ui	Qt5 UI File	District Admin Portal UI
11	tabbed_district_dashboard_fixed.ui	Qt5 UI File	District Admin Portal UI
12	tabbed_district_dashboard_structured2.ui	Qt5 UI File	District Admin Portal UI
13	guardian_homepage_window.py	Python	Guardian Portal
14	guardian.ui	Qt5 UI File	Guardian Portal UI
15	login.ui	Qt5 UI File	Login Window UI
16	student_homepage_window.py	Python	Student Portal

17	student.ui	Qt5 UI File	Student Portal UI
18	teacher_homepage_window.py	Python	Teacher Portal
19	teacher.ui	Qt5 UI File	Teacher Portal UI
20	db_config.ini	Config File	Operational Database Connection
21	wh_config.ini	Config File	Analytical Warehouse Connection
22	ETL.py	Python	ETL Script
23	create_StarSchema.py	Python	Script to create Analytical Warehouse

## 9.2 Operational Database Queries

All queries were written in the form of stored procedures. The four Jupyter notebooks used to populate the database with the procedures for each role can be found in the Queries folder.

Procedure Name	Role	Usage	Used in
teacher_course_schedule	Teacher	Getting course schedule	Teacher: Dashboard
teacher_information	Teacher	Get teacher name/ID	Teacher: Dashboard
teacher_one_class_all_students	Teacher	Get students from one course	Teacher: Grade Management
teacher_one_student_one_class_grades	Teacher	Get grades for all students in one course	Teacher: Grade Management
teacher_one_student_weighted_grade	Teacher	Get one student's weighted grade	Teacher: Grade Management
teacher_update_grade	Teacher	Update student's grade entries	Teacher: Grade Management
teacher_delete_grade	Teacher	Delete one student grade entry	Teacher: Grade Management
teacher_add_grade	Teacher	Add one student grade entry	Teacher: Grade Management
teacher_class_attendance_dates	Teacher	Get a class's attendance record dates	Teacher: Attendance
teacher_class_attendance_by_date	Teacher	Get class attendance records by date	Teacher: Attendance
teacher_add_attendance	Teacher	Add an attendance record for today's date	Teacher: Attendance
teacher_attendance_counts_by_date	Teacher	Get attendance type counts by date	Teacher: Attendance
teacher_all_students	Teacher	Get all students from all courses	Teacher: Communication
teacher_one_student_all_guardians	Teacher	Get all guardians of one student	Teacher: Communication
teacher_one_class_student_count	Teacher	Get the count of all students in one class	Teacher: Analytics
teacher_one_class_grade_counts	Teacher	Get letter grade counts for one class	Teacher: Analytics
get_student_course_schedule	Student	Get course schedule of one student	Student: Dashboard
get_individual_student	Student	Get student name/student ID	Student: Dashboard, Profile

get_student_grades	Student	Get student's overall course grades	Student: Grades
get_student_assignment_grades	Student	Get student's course grades	Student: Grades
get_student_guardian	Student	Get student's guardian	Student: Profile
get_student_hometeacher	Student	Get student's homeroom teacher	Student: Profile
get_guardian_info	Guardian	Get guardian's name	Guardian: Dashboard
get_guardian_student_grades	Guardian	Get overall course grades of their student	Guardian: Student Grades
get_guardian_student_attendance	Guardian	Get selected student's attendance records	Guardian: Attendance
get_student_attendance_snapshot	Guardian	Get selected student's attendance counts	Guardian: Attendance
get_guardian_student_attendance_dates	Guardian	Get selected student's attendance dates	Guardian: Attendance
get_guardian_student_teacher	Guardian	Get student's homeroom teacher contact	Guardian: Contact Teacher

### 9.3 Analytical Database Queries

Procedure Name	Usage
absence_rate	Get absence as percentage
avg_score	Get average of weighted scores
count_schools_by_type	Get a count of schools by school type
fulltime_teachers	Get count of full time teachers
get_active_students	Get count of distinct active students
get_all_schools	Get all schools names
get_avg_score	Compute the average score for a specific school, grade level, and assignment
get_grade_levels	Load appropriate grade levels once a school is selected
get_school_attendance	Get school attendance rate at each school in district
get_school_grade_attendance	Get attendance by grade level for a specific school
get_total_schools	Get total schools in the district
total_students	Get total students in the district