

Criterion B: Design

Description:

The design section of the documentation will provide UML diagrams, flowcharts, graphical visualizations of the product, a test plan, and the database model. The Unified Modeling Language diagram presents all the classes that will be used for creating the product and the different variables and methods used for the individual classes. The flow charts will show how different processes of the software work and the different components of the application. The graphical visualizations show how the product will be designed with buttons, frames and text. There is also the test plan of all the success criteria and how it will be tested with the expected input and output. Finally, there is the database section which presents the model to how data will be stored both with individual files and individual data.

UML Tables:

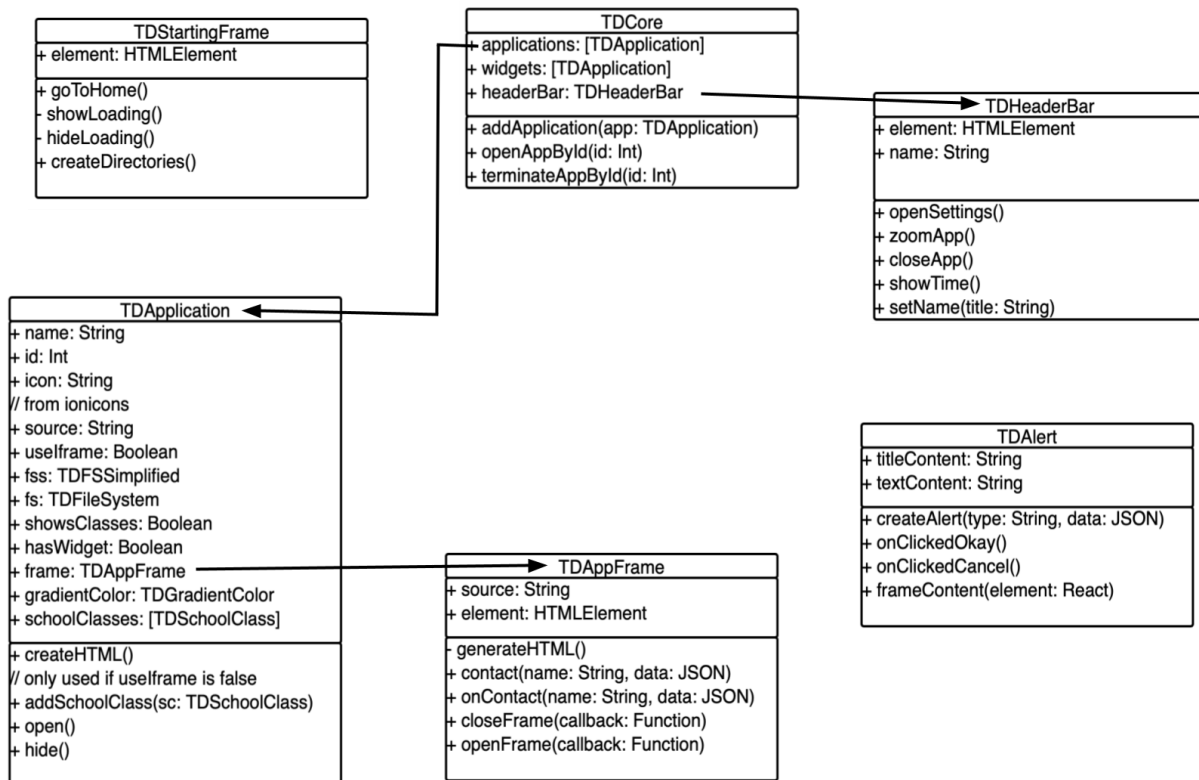


Figure 1: Core GUI Application Structure UML design

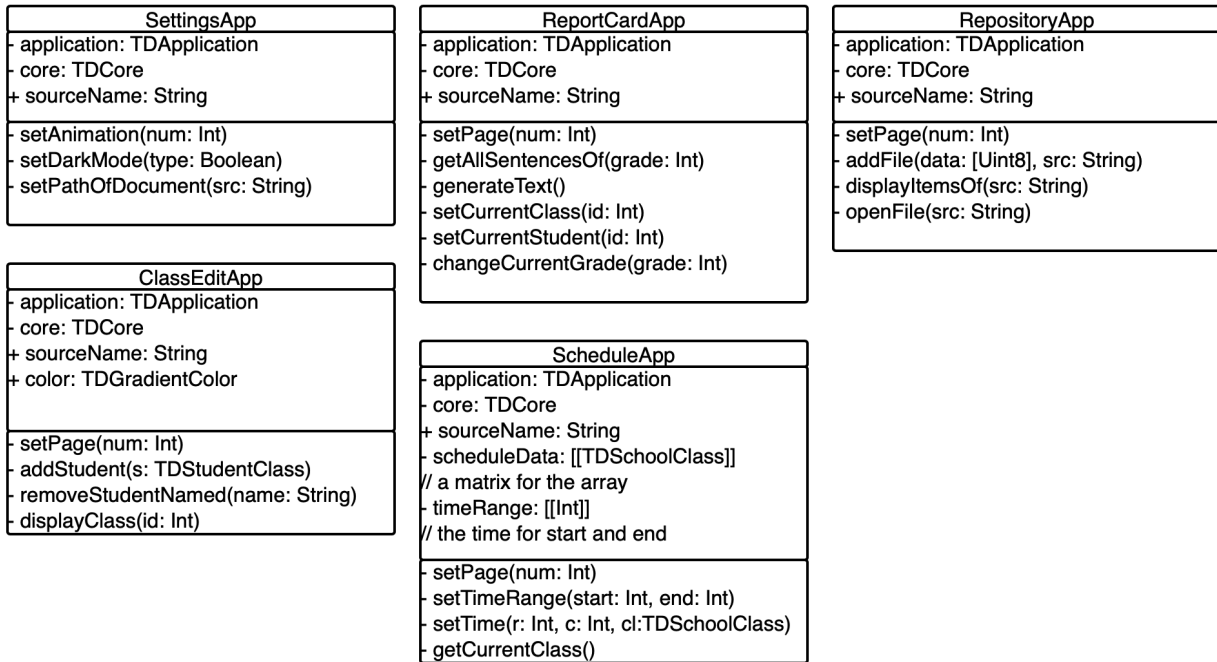


Figure 2: Main Applications UML design

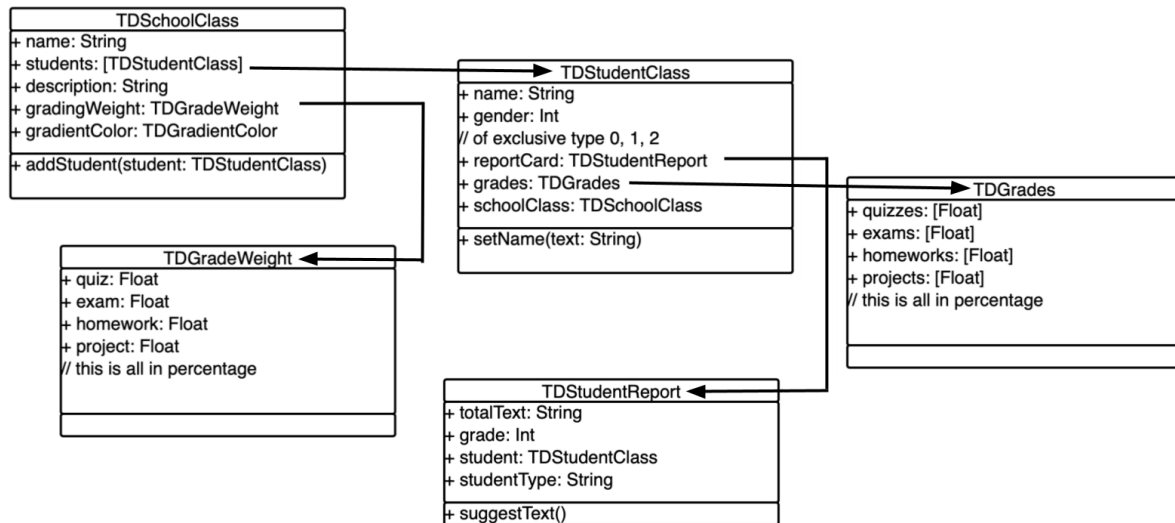


Figure 3: School based Classes UML design

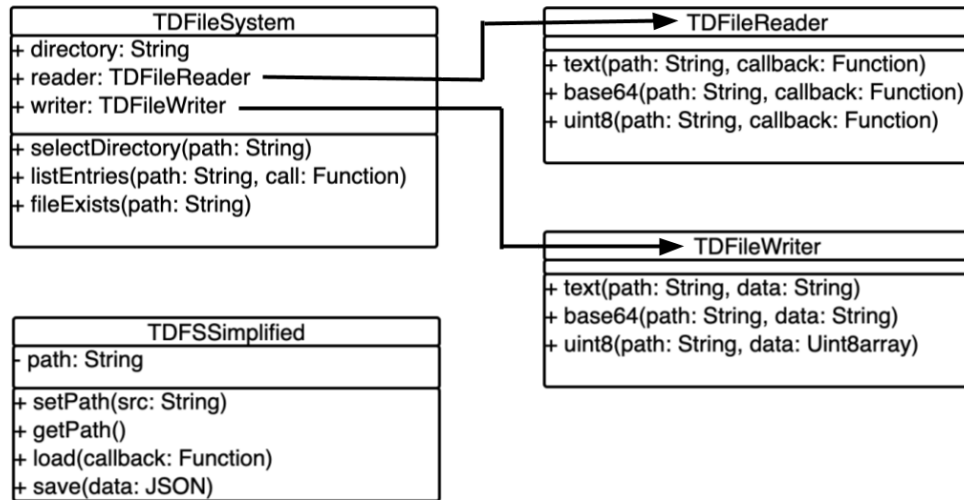


Figure 4: File System Classes UML design

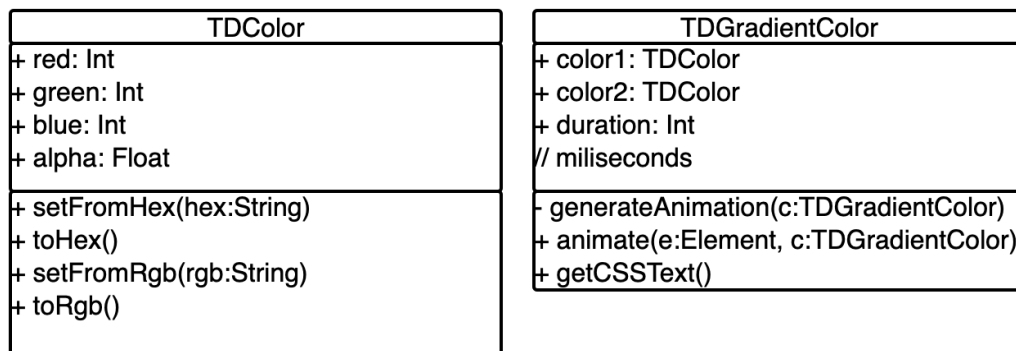


Figure 5: Coloring System UML design

Classes definition:

1. Core GUI Application Structure:
 - a. TDCore: This is where all the general core system data and graphical side of the app is located.
 - b. TDApplcation: This is the main application class that contains all the data for a sub-application in the TeachDash application.
 - c. TDHeaderBar: This is the class for the main header of the layout of the app where there are some controls for the user to interact with the sub apps.
 - d. TDAppFrame: this is the class for the actual GUI side of a TeachDash application with different methods for the core to interact with the application.
 - e. TDStartingFrame: This is the frame for when the application first loads and there is the information

- f. TAlert: This is the class for alerting the user of a specific event
- 2. Main Applications: (All inherited from TDAApplication)
 - a. SettingsApp: This is the settings app of TeachDash which has the main controls.
 - b. ReportCardApp: This is the application that allows teachers to make report cards.
 - c. RepositoryApp: This is the application that acts as a filesystem for the teacher's data.
 - d. ClassEditApp: This is the application that allows teachers to edit the classes that they have.
 - e. ScheduleApp: This is the application that allows teachers to have a schedule for what class they have and when.
- 3. School based Classes:
 - a. TDStudentClass: This is the class that contains general information on the students in the class.
 - b. TDSchoolClass: This is the class that contains the main information on a school class with general information.
 - c. TDStudentReport: This is the class that contains information about the way a student should be presented on a report card.
 - d. TDGrades: This is the class for all the grades a student has.
 - e. TDGradeWeight: This is a class that contains information on how much a grade should be weighted by percent (such as tests being 50%, quiz 20%, etc).
- 4. File System Classes
 - a. TDFileSystem: This is a class that has all the core file or directory methods
 - b. TDFileReader: This is a simplified version of the class FileReader to make it simpler to get file information
 - c. TDFileWriter: This is a class similar to the TDFileReader class but for writing
 - d. TDFSSimplified: This is a class that makes loading and saving JSON data incredibly simplified without having to look through the file or knowing where it is.
- 5. Coloring System:
 - a. TDColor: This is a class just for a single color that can be used by the element and in data form
 - b. TDGradientColor: This is a class to simplify making gradients on the graphic user interface of the application

Flowcharts:

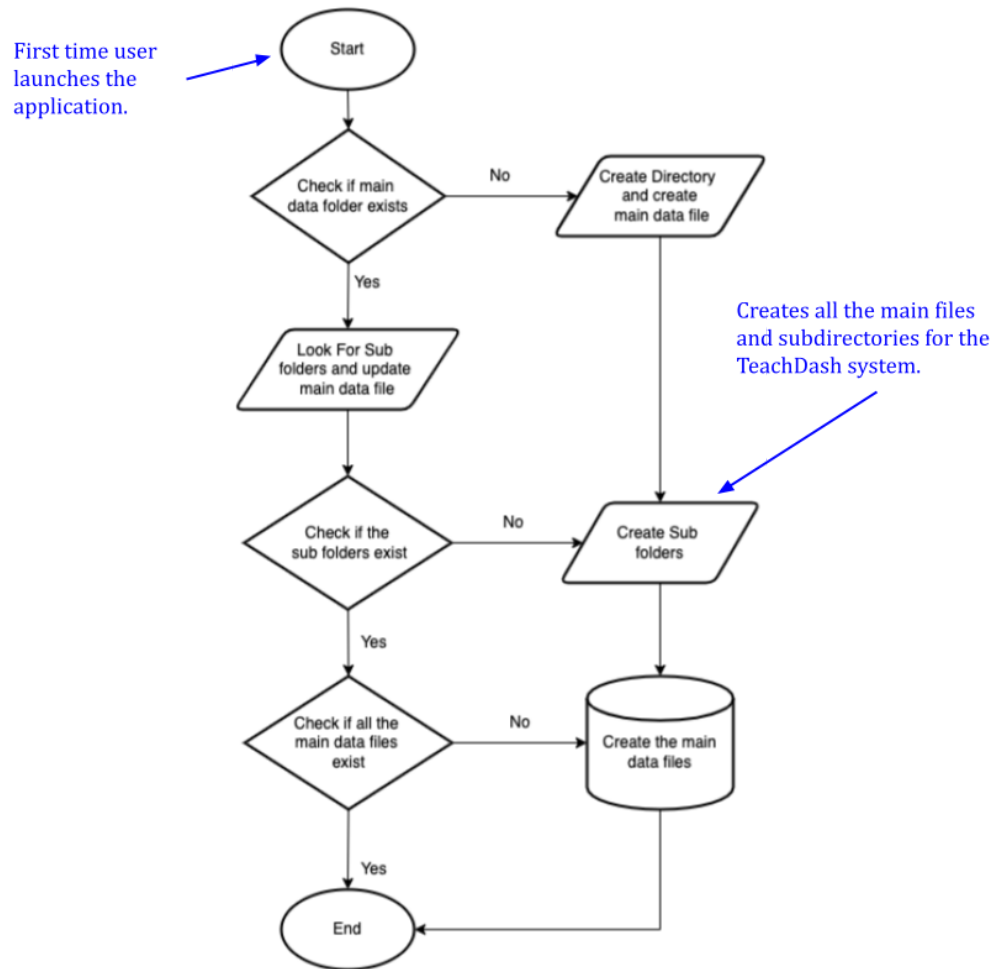


Figure 6: App first launched flowchart diagram

core features of the header bar within the app with basic features

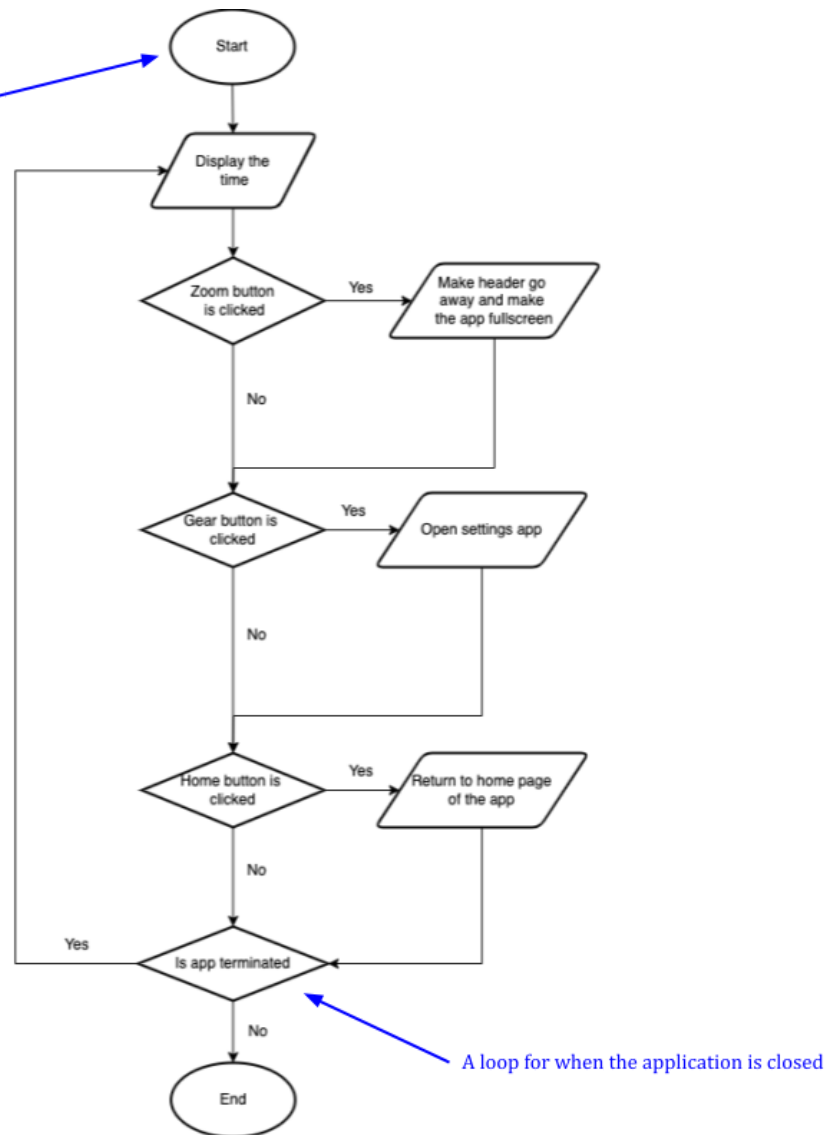


Figure 7: Header Bar Buttons flowchart diagram

Main process for when the application opens up on TeachDash

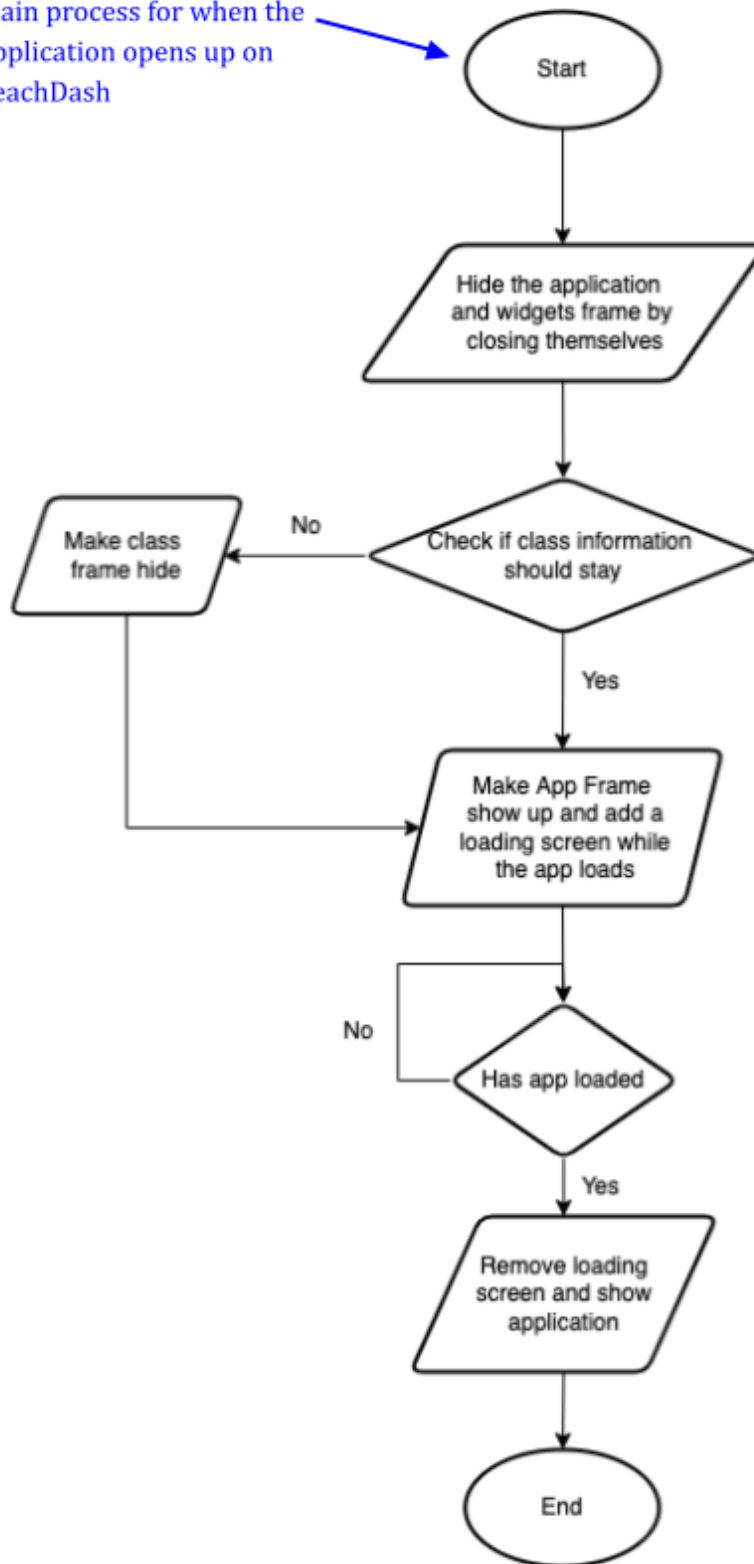


Figure 8: Changing Application Setup flowchart diagram

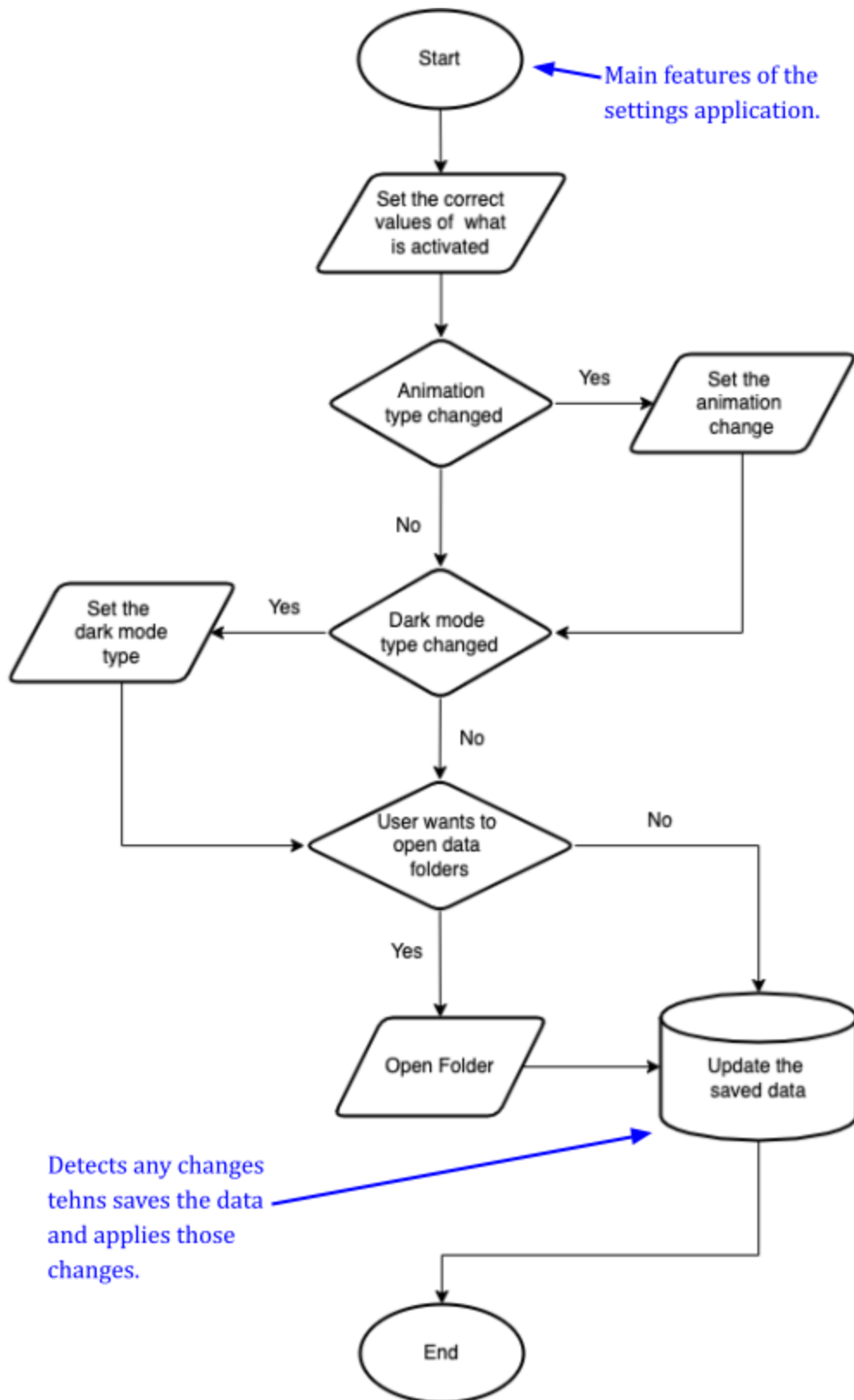


Figure 9: Settings Application flowchart diagram

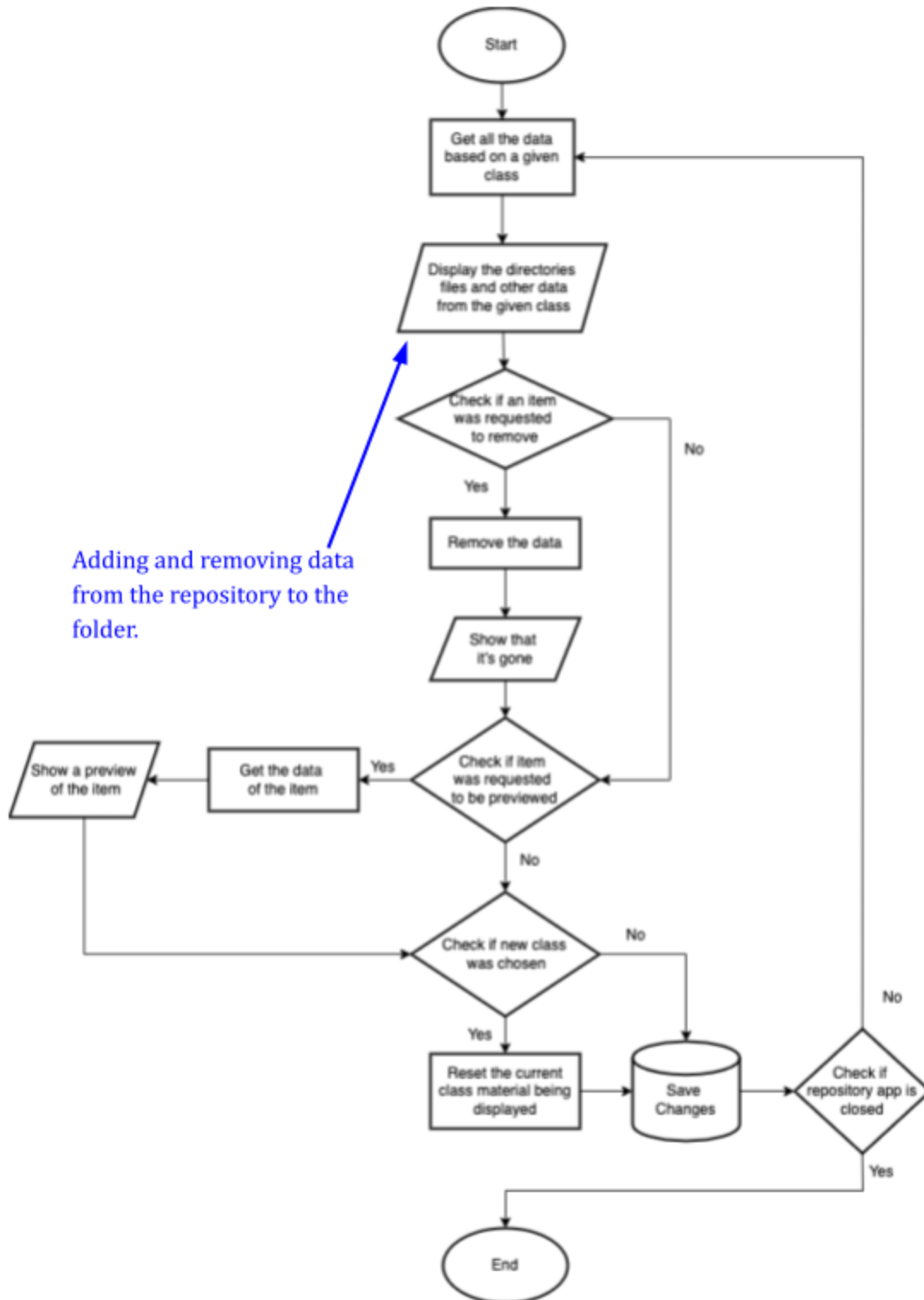


Figure 10: Repository Application flowchart diagram

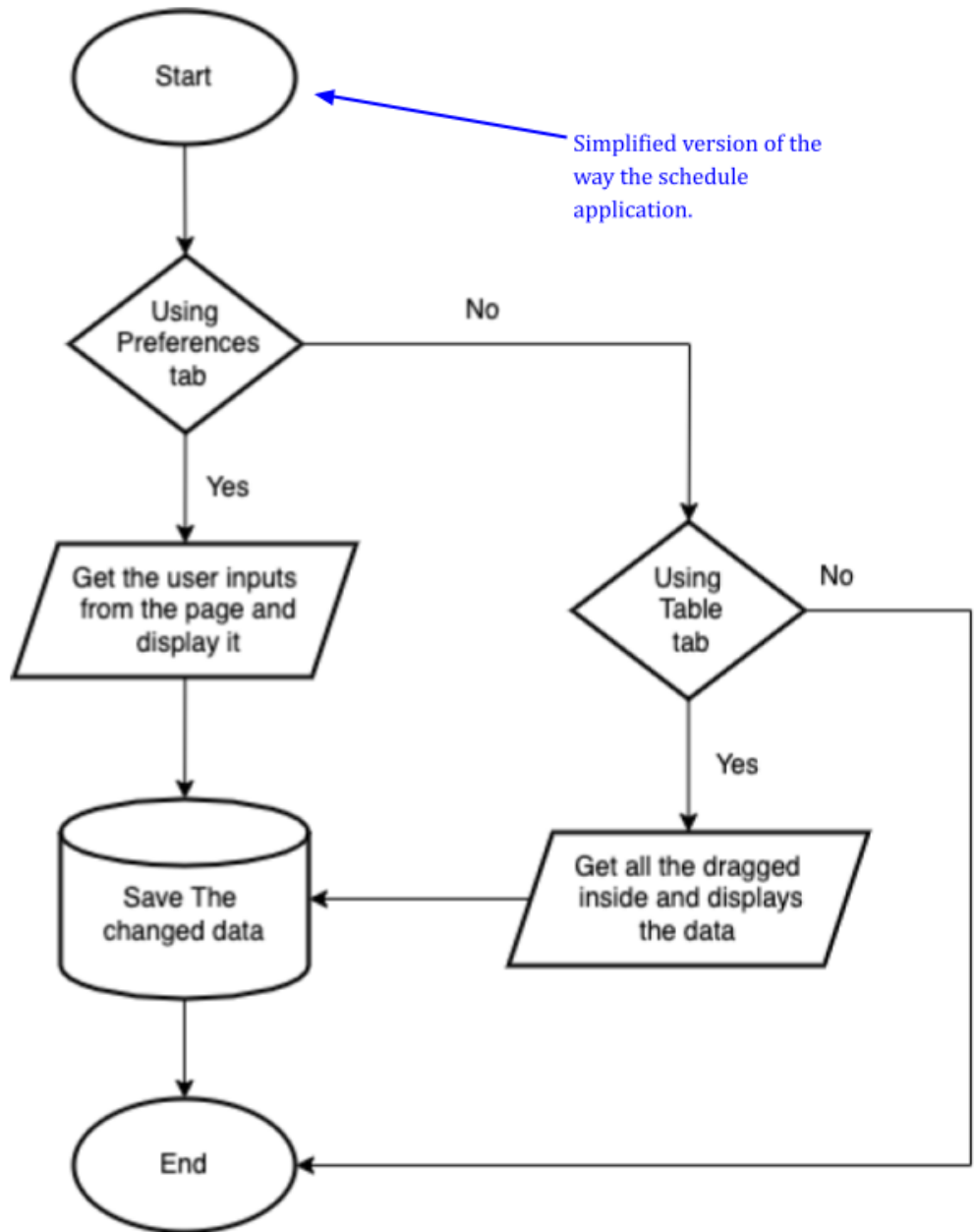


Figure 11: Schedule Application flowchart diagram

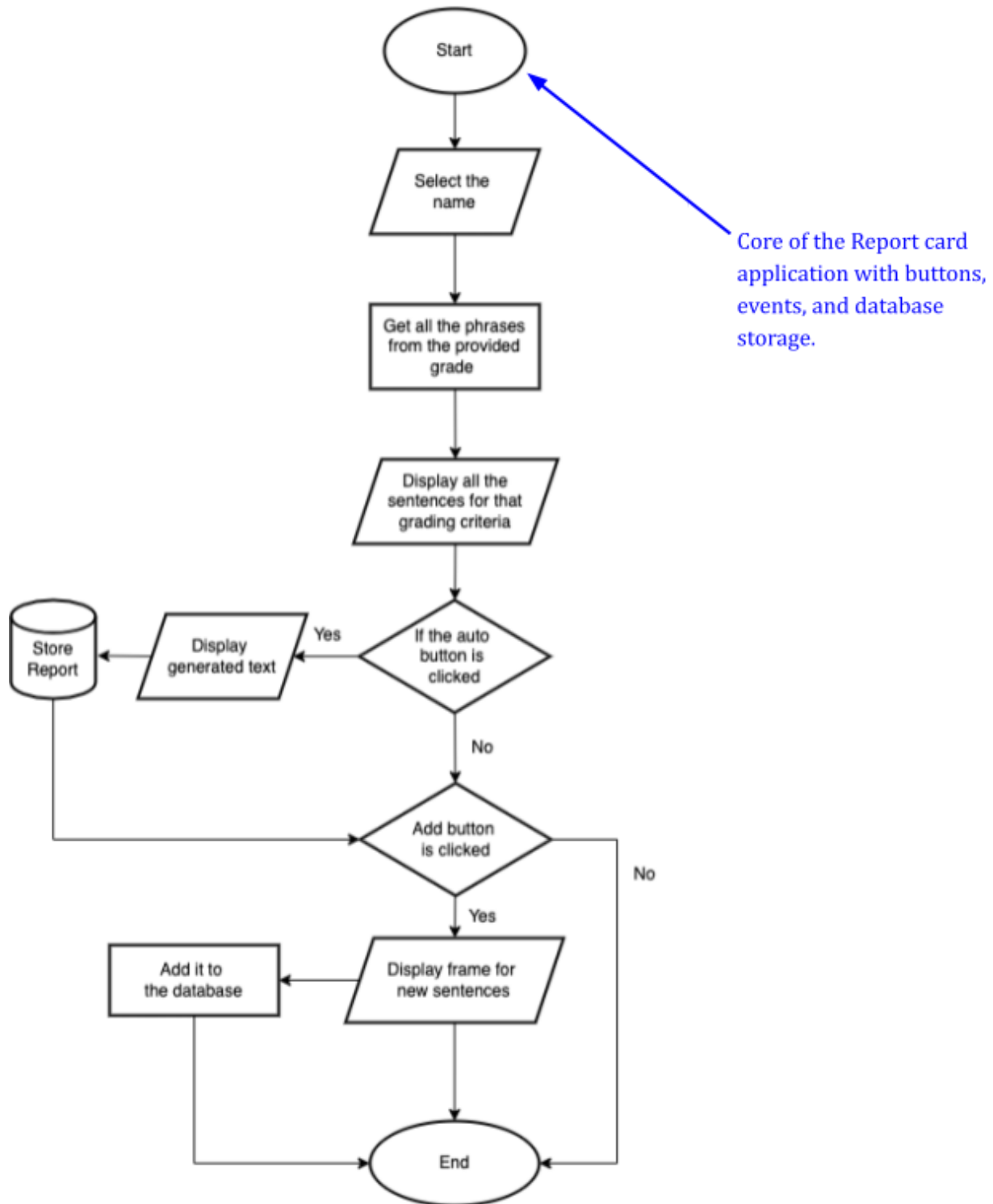


Figure 12: Report Card Application flowchart diagram

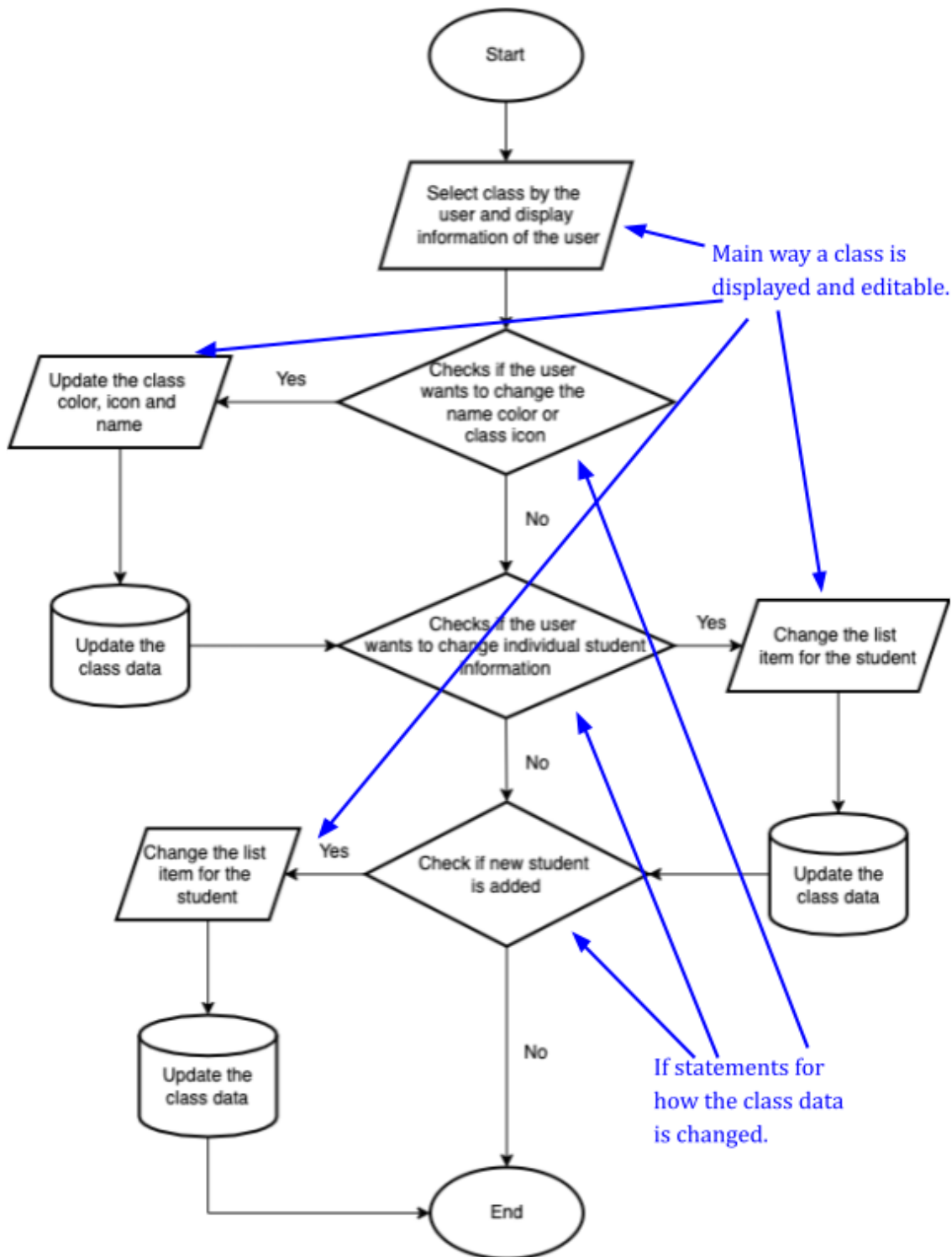


Figure 13: Class Editor Application flowchart diagram

Graphical Visualization:

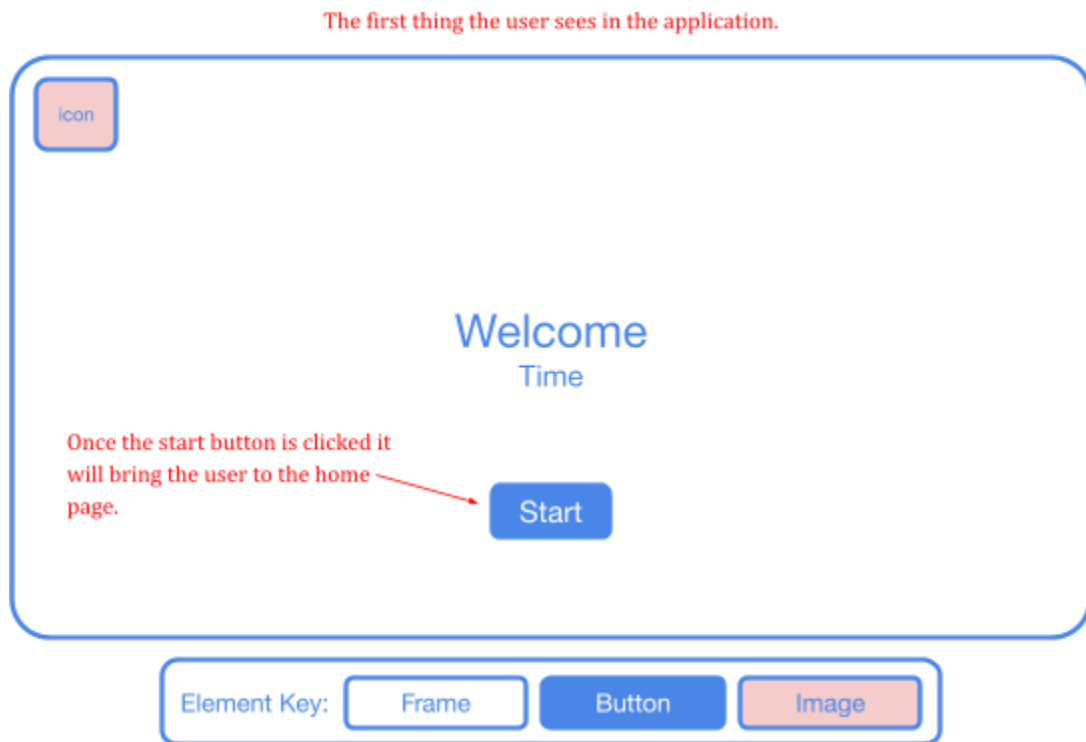


Figure 14: Welcome Screen Graphical Visualization

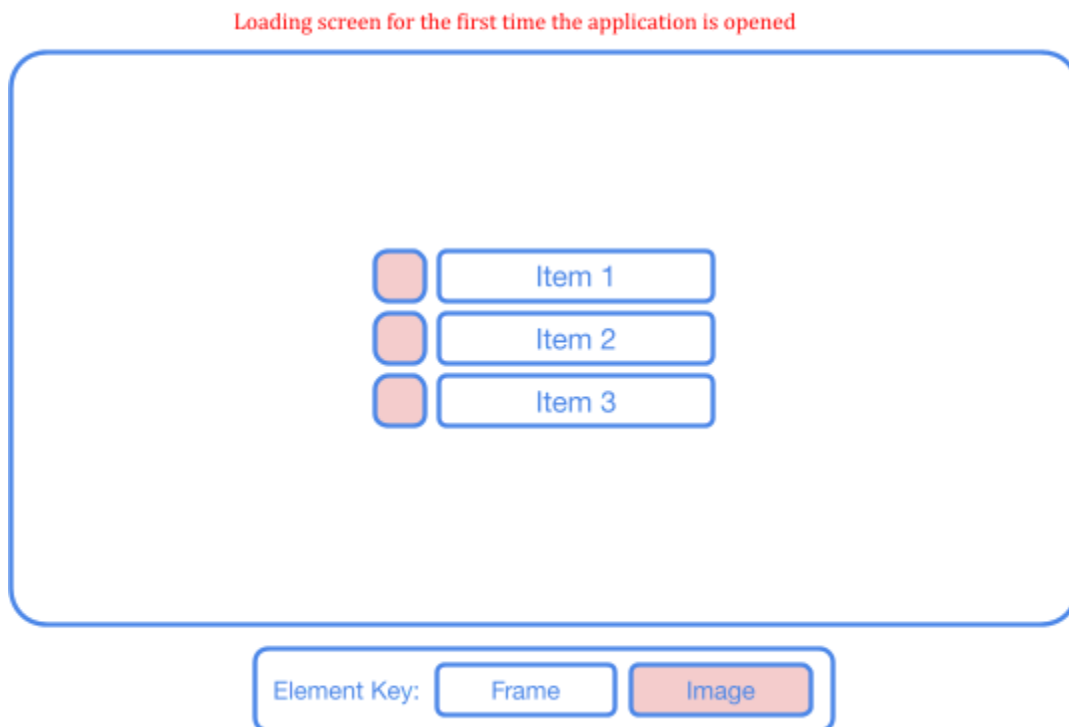


Figure 15: Loading Screen Graphical Visualization

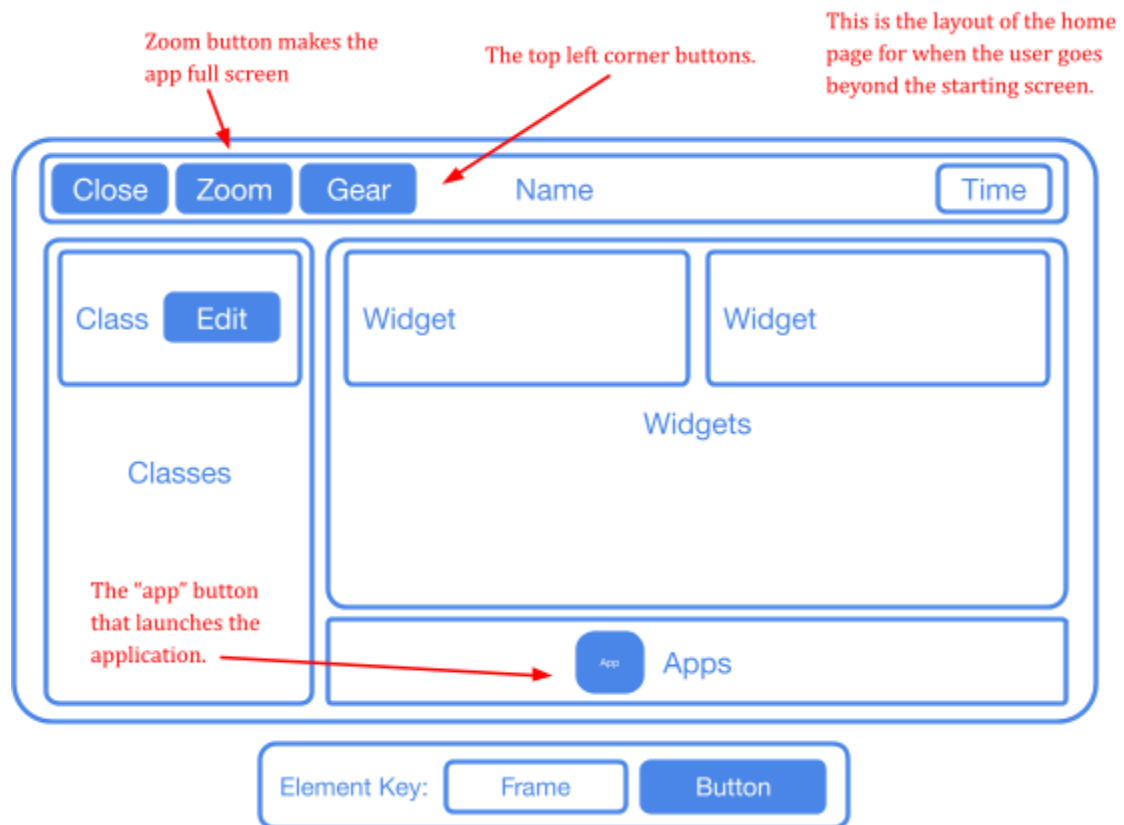


Figure 16: Home Page Graphical Visualization

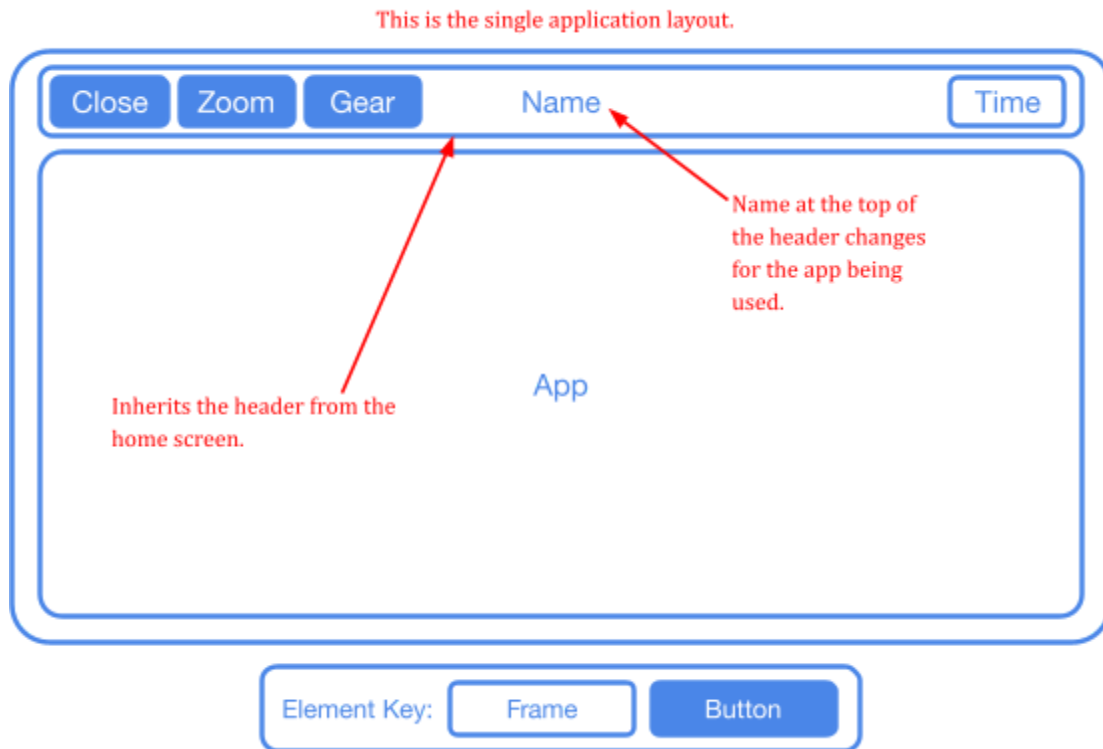


Figure 17: Single Application Layout Graphical Visualization

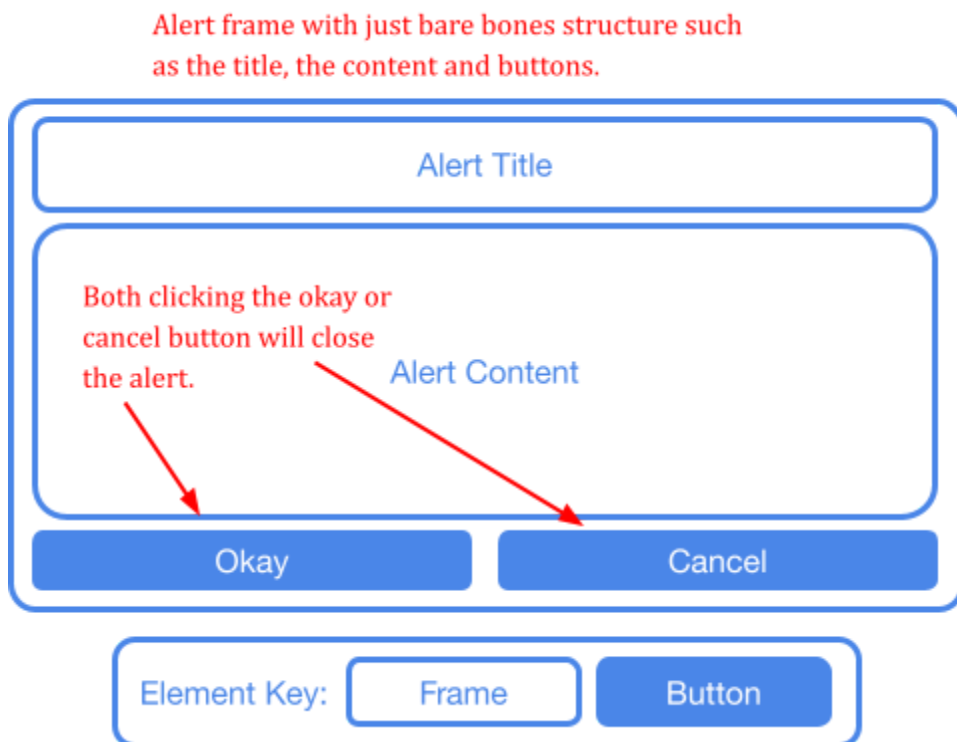


Figure 18: Alert System Graphical Visualization

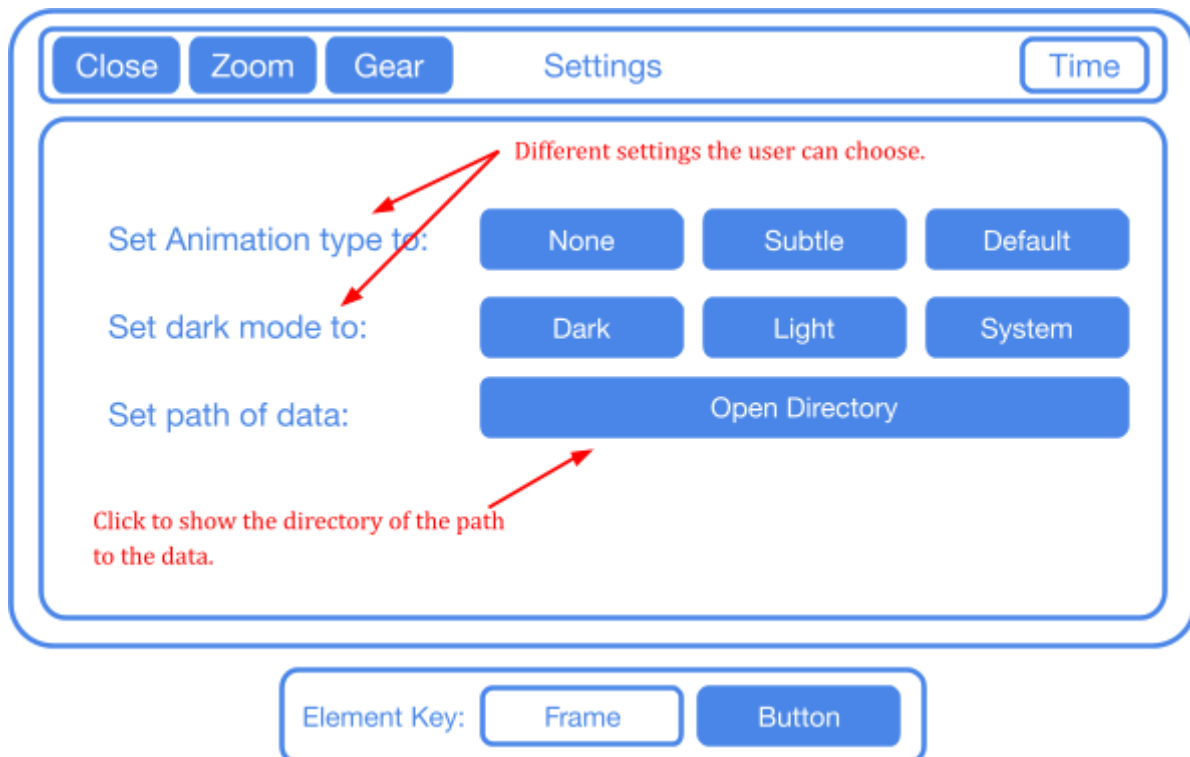


Figure 19: Settings Application Graphical Visualization

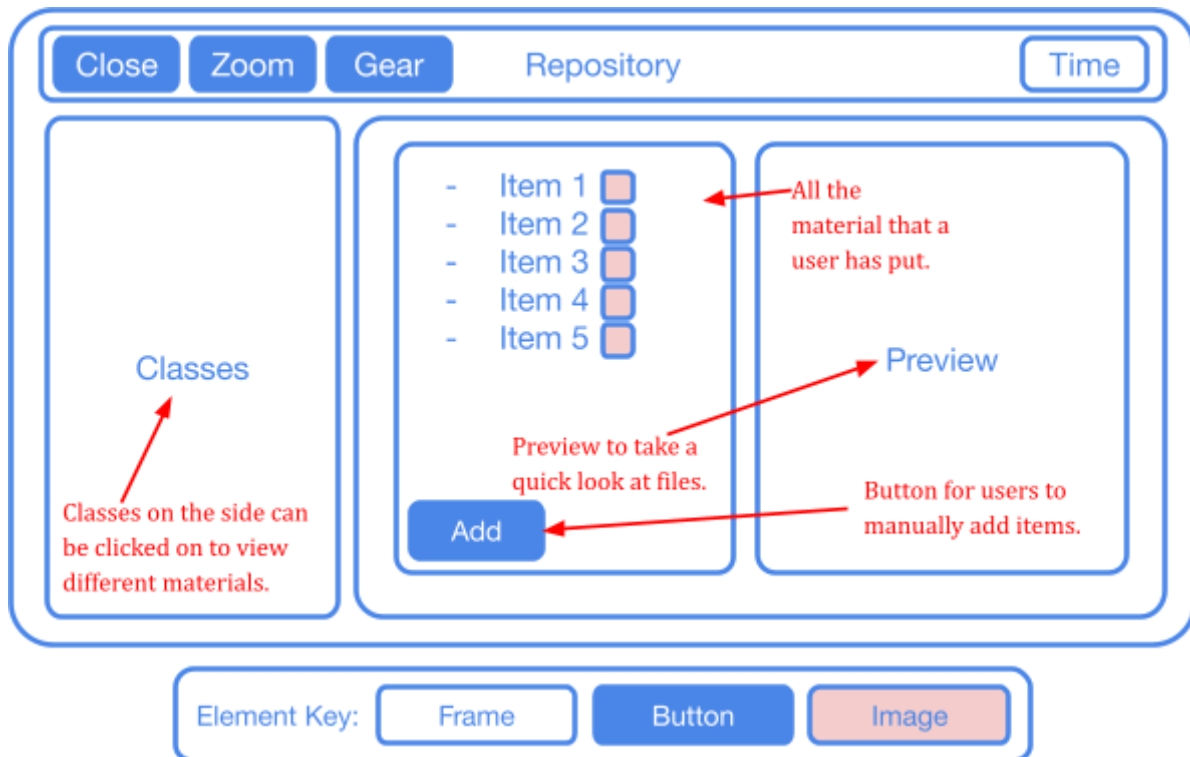


Figure 20: Repository Application Graphical Visualization

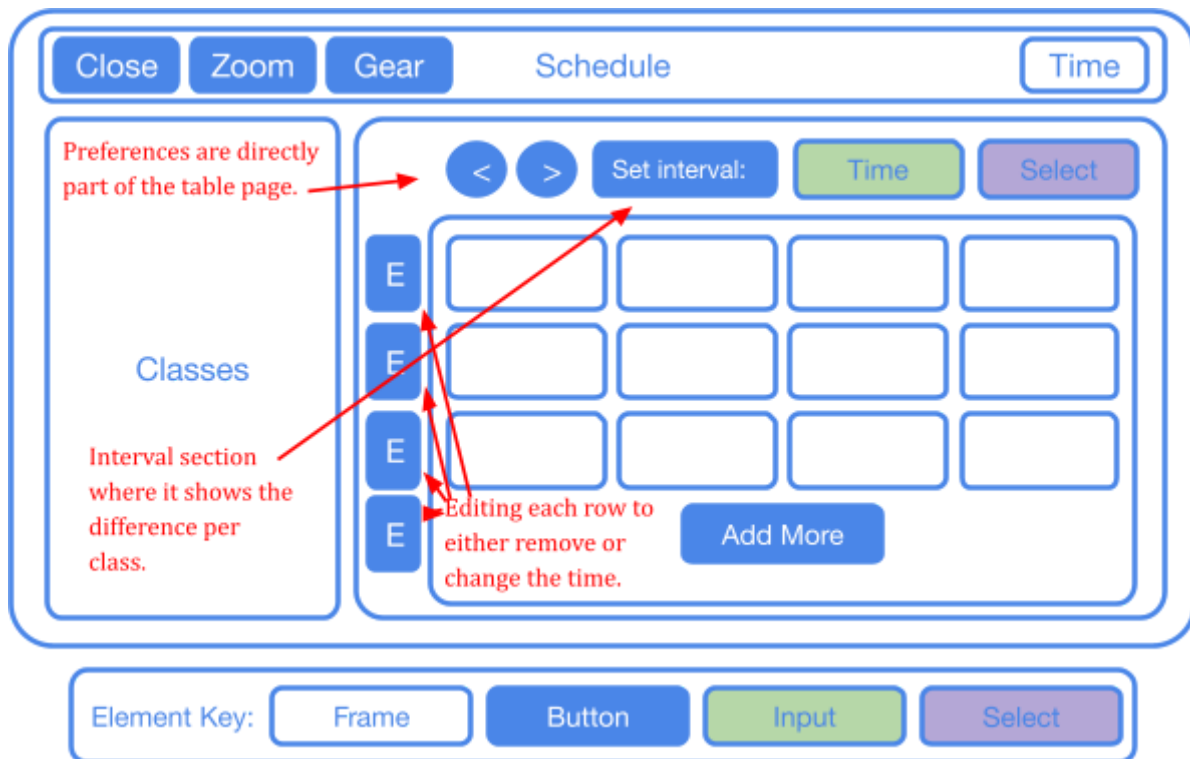


Figure 21: Schedule Application Graphical Visualization

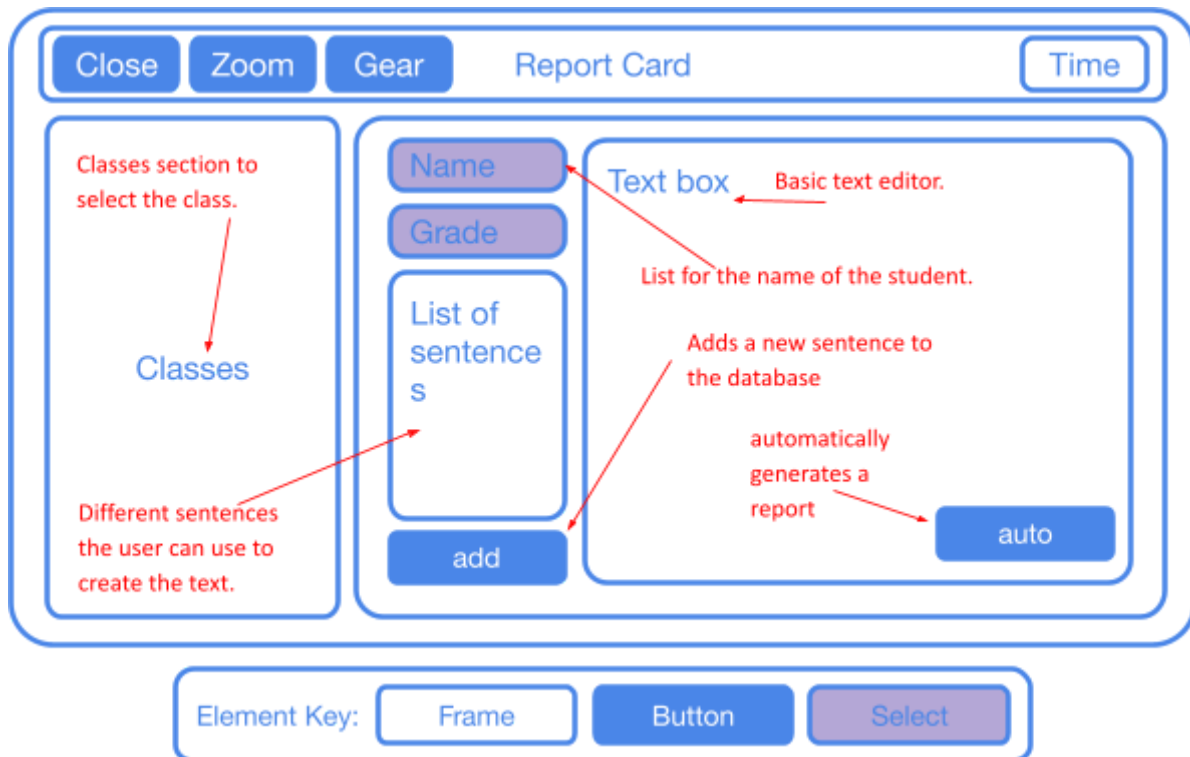


Figure 22: Report Card Application Graphical Visualization

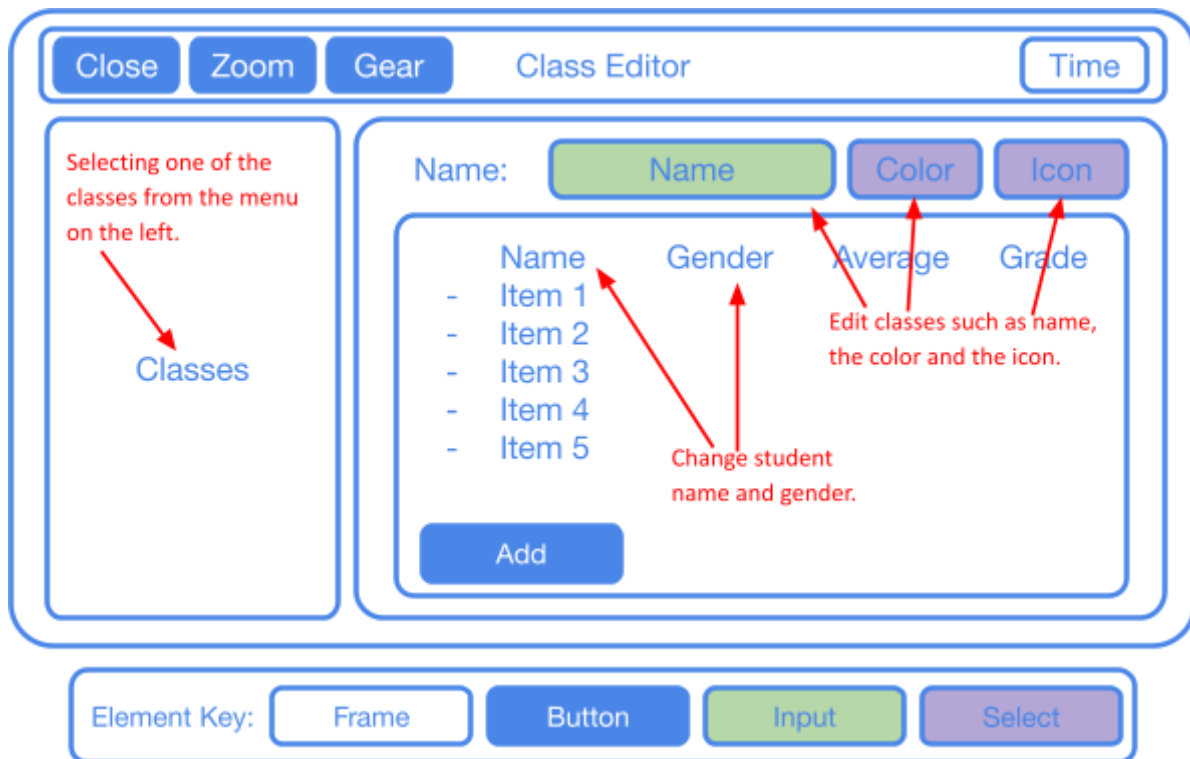


Figure 23: Class Editor Application Graphical Visualization

Test Plan:

Success Criteria:	Description:	Input Information:	Expected Output:
1. The application detects if the user opened the application for the first time.	This will test if the user has opened the application by detecting the variable value of firstTimeOpened in the user data or detects if the file exists.	The two main inputs are either the existence of the file or the value of the firstTimeOpened variable.	If the user data file doesn't exist, then it will create the file with the firstTimeOpened variable value to true and then it will make it false after the application stored it.
2. An operating system layout with a header and dark / light mode.	This criteria will be tested for going through the settings application and clicking on the layout buttons.	To test it, the user will click on the buttons in the settings app through the application and it will then change the value of colorMode variable from the user data.	If the user clicks on the system, then the variable colorMode will be empty, if light or dark is clicked then it will store the value "body-light" or "body-dark."
3. Ability to toggle the animation system.	This will be tested to allow the user to enable or disable the animation of the user interface where it will load pages much faster and be significantly less smooth so it can go much faster.	To test it, the user will click on the button on and off for the animation system and it will change the user data as such.	When the user clicks on the button, it will toggle the allowsAnimation variable where it changes from true or false.
4. A shortcut to edit each class from the home page.	This criteria will be met by having 3 buttons under each school class from the home page to the other sub-applications.	It will be tested by manually clicking it through the homepage of the application.	Once it is clicked, the sub-application should open up with the dedicated school class that was opened from.
5. Being able to create and edit different classes.	This will allow for the user to be able to easily edit school classes, create new ones and control the student information.	This will be done by clicking on the class editor and have the option for the user to click on the different user interface to edit the student information.	If the user interacts with any of the components of the sub-application it should modify the user data and then the system writes the user data as it saves it.

6. Ability to drag and drop files into a repository of each individual class with the ability to preview them.	This feature will simply have the ability for the user to organize the repository sub-application by dragging files into the application.	To test this, I will have several dummy files in which I drag them into the middle of the application.	Once the file is dropped, the software should duplicate that file to the user data path and display the newly added file.
7. A scheduling system in a drag and drop format that is separated in multiple weeks.	Define the matrix for the schedule system in a drag and drop format by taking the school classes and dropping them into the frame.	To test this I will make sure that the navigation section of the sub-application has the ability to click on the school classes and drag it into a box.	Once the class is dropped, the system should identify where the school class was dropped and its corresponding location to the matrix.
8. Having a text editor for writing report cards.	This just has a basic text editor when writing report cards throughout the software.	When going to the report card sub-application, the user can type on their keyboard after clicking on the text box.	When the user starts typing different letters and symbols, the new added text will render on the application and should be stored for the user data.
9. A data based for the different sentences to add them.	The application will have a portion of data that stores different sentences that can be used for the report card section of the application.	To use sentences for the report card sub-application, the user will have an entire section in which they can double click on the sentences to use them.	Once the user double clicks on a sentence, the text will appear on the report card editor and will change based on the student's name.
10. The report card section can learn from the user's input and write.	This portion of the product will use the text that the user wrote in the report card section and learn from that data to use it properly.	To test this I will write multiple sentences in the report card application text editor and click on "learn from text."	Once I click on the button to learn the text data, the software will extract the different sentences and will add them to the sentences database.

Table 1: Test Plan

Files General Saving:

This diagram shows the file structure of the TeachDash application in which all the data is written with files. The Applications folder contains data for the individual sub applications in JSON format. The repository is a folder which possesses all the different folders that are created in the application and can manage the different classes.

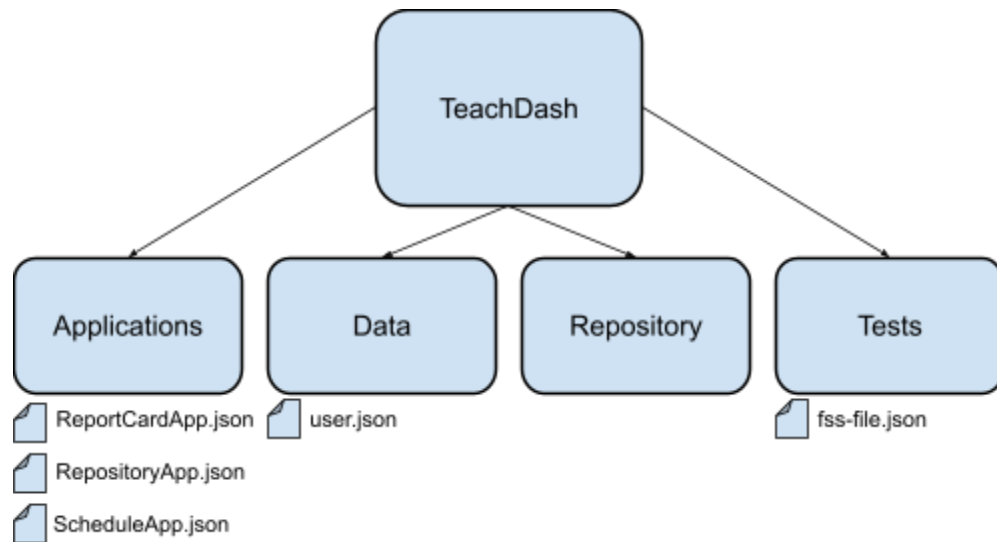


Figure 24: File General Saving Data

User Data Format:

This table presents the different information for the user data that is stored by the application.

Property Name:	Property Type:	Property Description:
allowsAnimation	Boolean	A variable to check if animations can be used by the application (default is true)
allowsNotifications	Boolean	A variable to checks if notifications can be used by the application (default is true)
appsVisitedById	[String]	a list of all the applications visited by their ids to see if the application should be reloaded.
colorMode	String	this tells if the system is using the color mode chosen by the user or the default one (of exclusive type "dark", "light" or "")

firstTimeOpen	Boolean	a variable to see if the user has opened the app before to see if the starting screen needs to show the loading or not.
schoolClasses	TDSchoolClass	An array of all the different school classes that the user created and has a different place.
studentClassIds	Integer	A pointer variable to get the school class ids to define a new school class id so that it's not using a system.

Table 2: User Data Format Data