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:

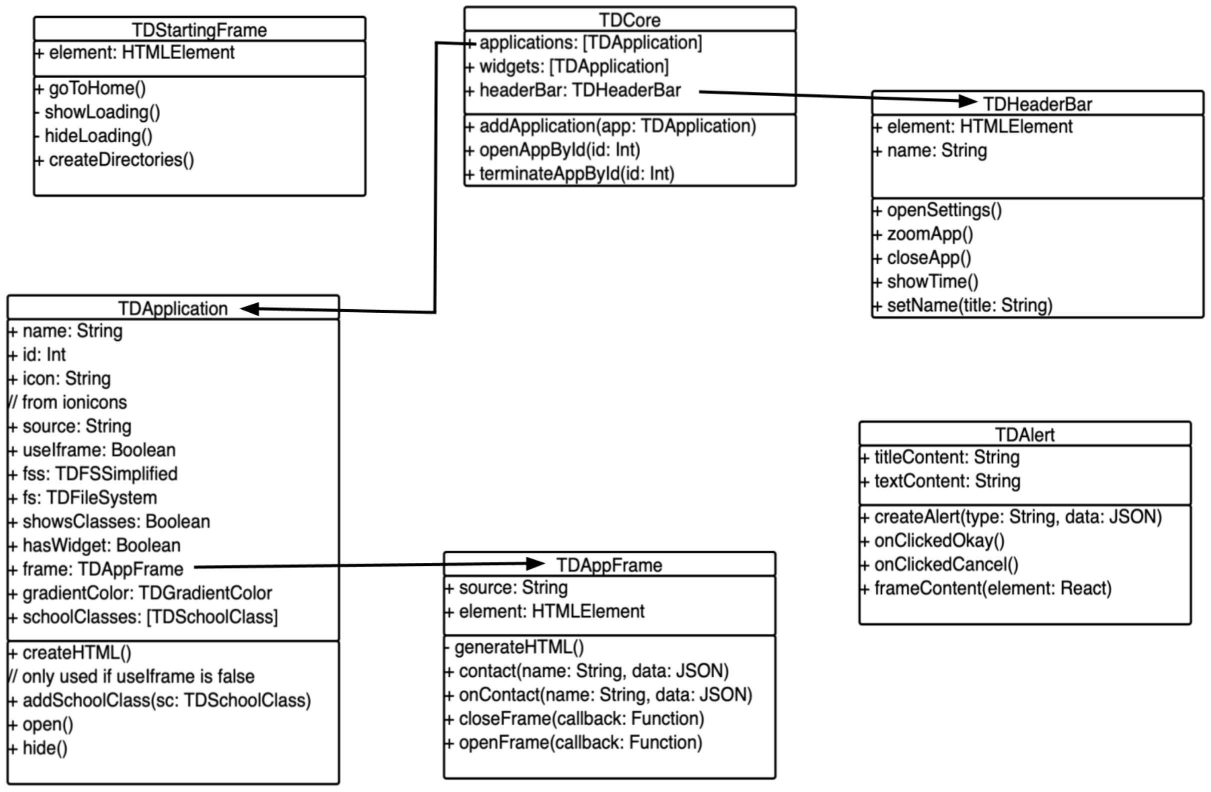
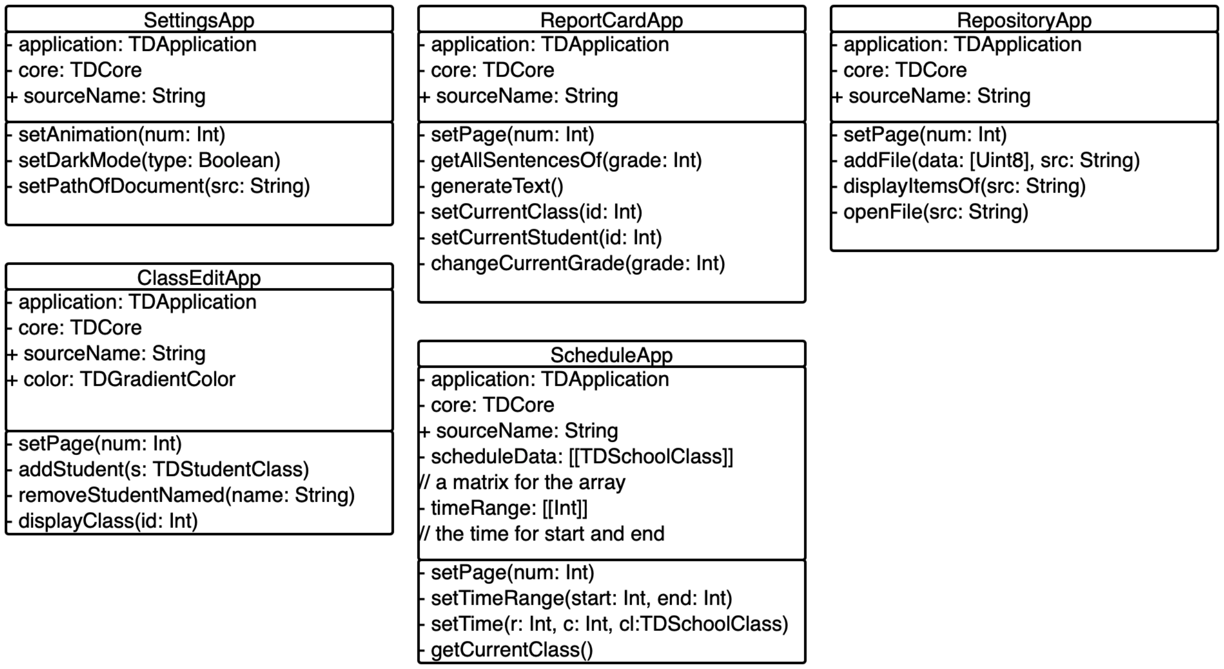


Figure 1: Core GUI Application Structure UML design



SettingsApp
- application: TDApplication
- core: TDCore
+ sourceName: String
- setAnimation(num: Int)
- setDarkMode(type: Boolean)
- setPathOfDocument(src: String)
ReportCardApp
- application: TDApplication
- core: TDCore
+ sourceName: String
- setPage(num: Int)
- getAllSentencesOf(grade: Int)
- generateText()
- setCurrentClass(id: Int)
- setCurrentStudent(id: Int)
- changeCurrentGrade(grade: Int)
RepositoryApp
- application: TDApplication
- core: TDCore
+ sourceName: String
- setPage(num: Int)
- addFile(data: [Uint8], src: String)
- displayItemsOf(src: String)
- openFile(src: String)
ClassEditApp
- application: TDApplication
- core: TDCore
+ sourceName: String
+ color: TDGradientColor
- setPage(num: Int)
- addStudent(s: TDStudentClass)
- removeStudentNamed(name: String)
- displayClass(id: Int)
ScheduleApp
- application: TDApplication
- core: TDCore
+ sourceName: String
- scheduleData: [[TDSchoolClass]]
// a matrix for the array
- timeRange: [[Int]]
// the time for start and end
- setPage(num: Int)
- setTimeRange(start: Int, end: Int)
- setTime(r: Int, c: Int, cl:TDSchoolClass)
- getCurrentClass()

Figure 2: Main Applications UML design

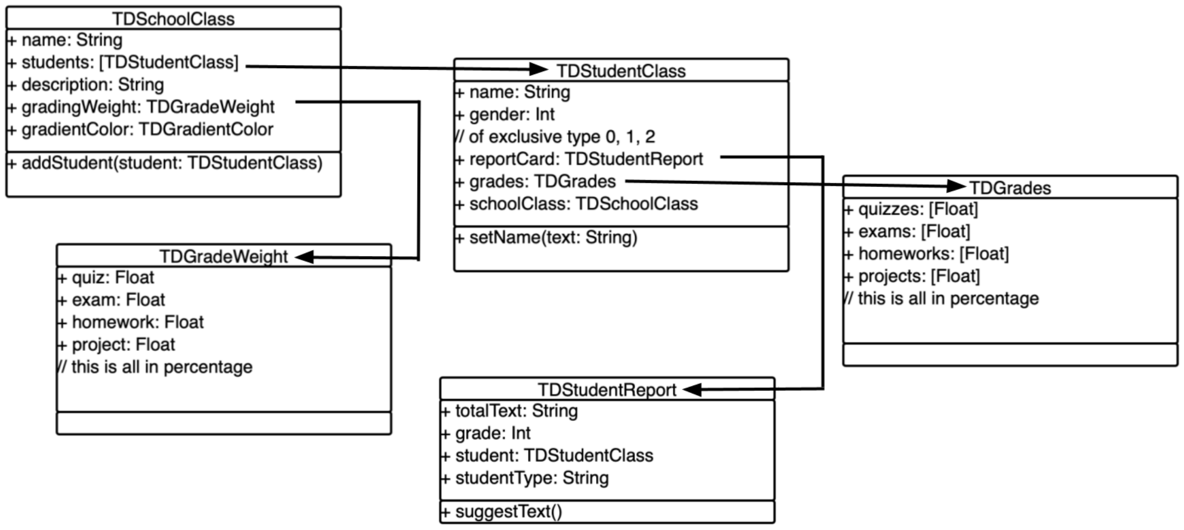


Figure 3: School based Classes UML design

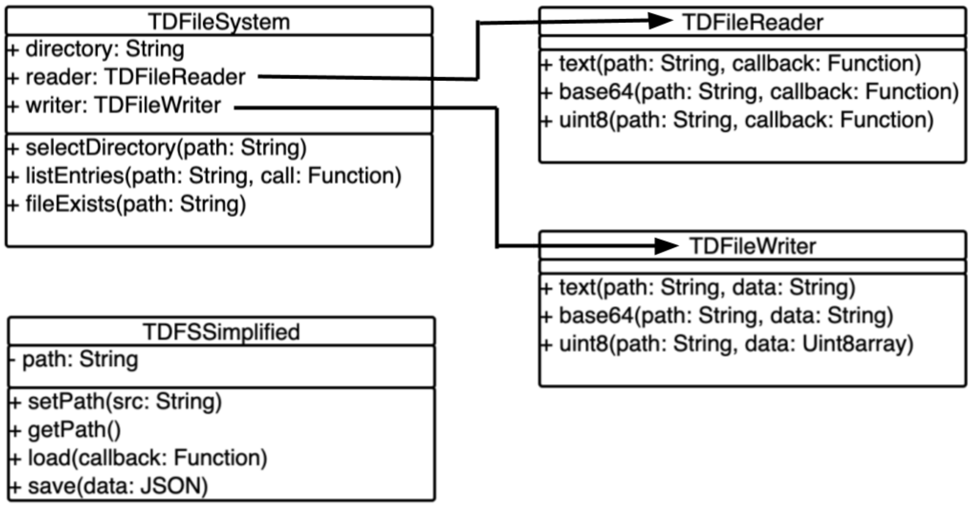

TDFileSystem
+ directory: String
+ reader: TDFileReader
+ writer: TDFileWriter
+ selectDirectory(path: String)
+ listEntries(path: String, call: Function)
+ fileExists(path: String)
TDFileReader
+ text(path: String, callback: Function)
+ base64(path: String, callback: Function)
+ uint8(path: String, callback: Function)
TDFileWriter
+ text(path: String, data: String)
+ base64(path: String, data: String)
+ uint8(path: String, data: Uint8array)
TDFSSimplified
- path: String
+ setPath(src: String)
+ getPath()
+ load(callback: Function)
+ save(data: JSON)

Figure 4: File System Classes UML design

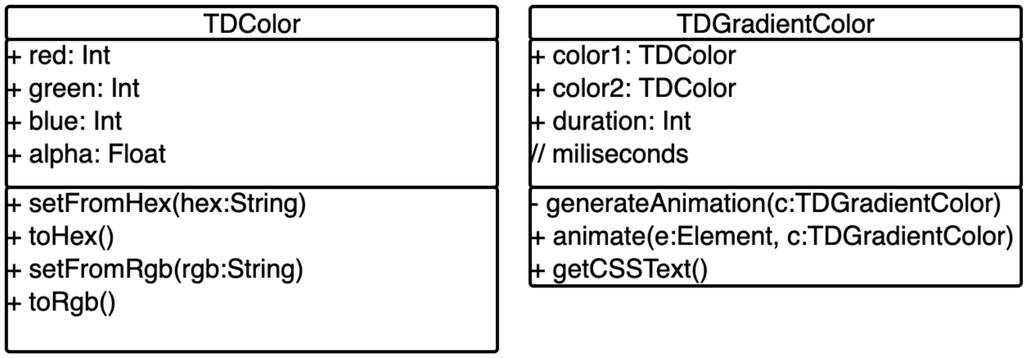


Figure 5: Coloring System UML design

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. TDApplication: 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. TDAlert: This is the class for alerting the user of a specific event
2. Main Applications: (All inherited from TDApplication)
    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. TDFSSimplifed: 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:

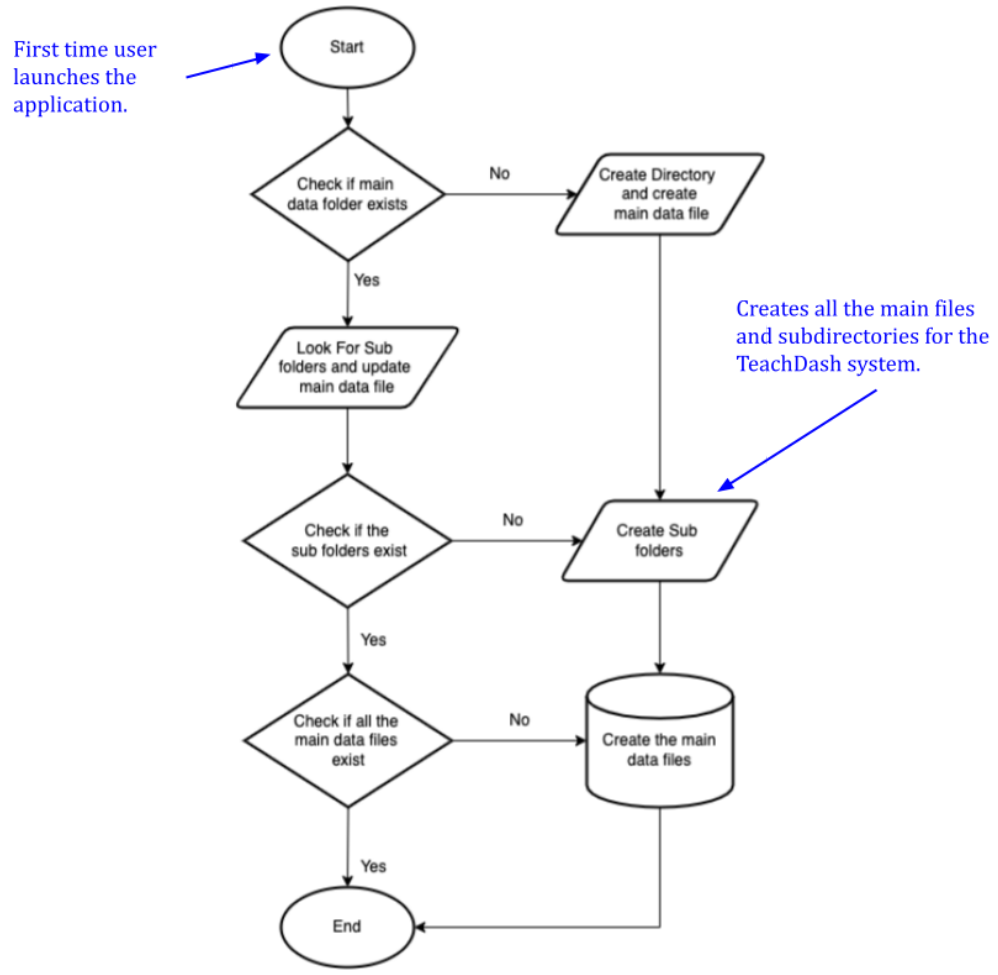

First time user launches the application.
Start
Check if main data folder exists
No
Create Directory and create main data file
Yes
Look For Sub folders and update main data file
Creates all the main files and subdirectories for the TeachDash system.
Check if the sub folders exist
No
Create Sub folders
Yes
Check if all the main data files exist
No
Create the main data files
Yes
End

Figure 6: App first launched flowchart diagram

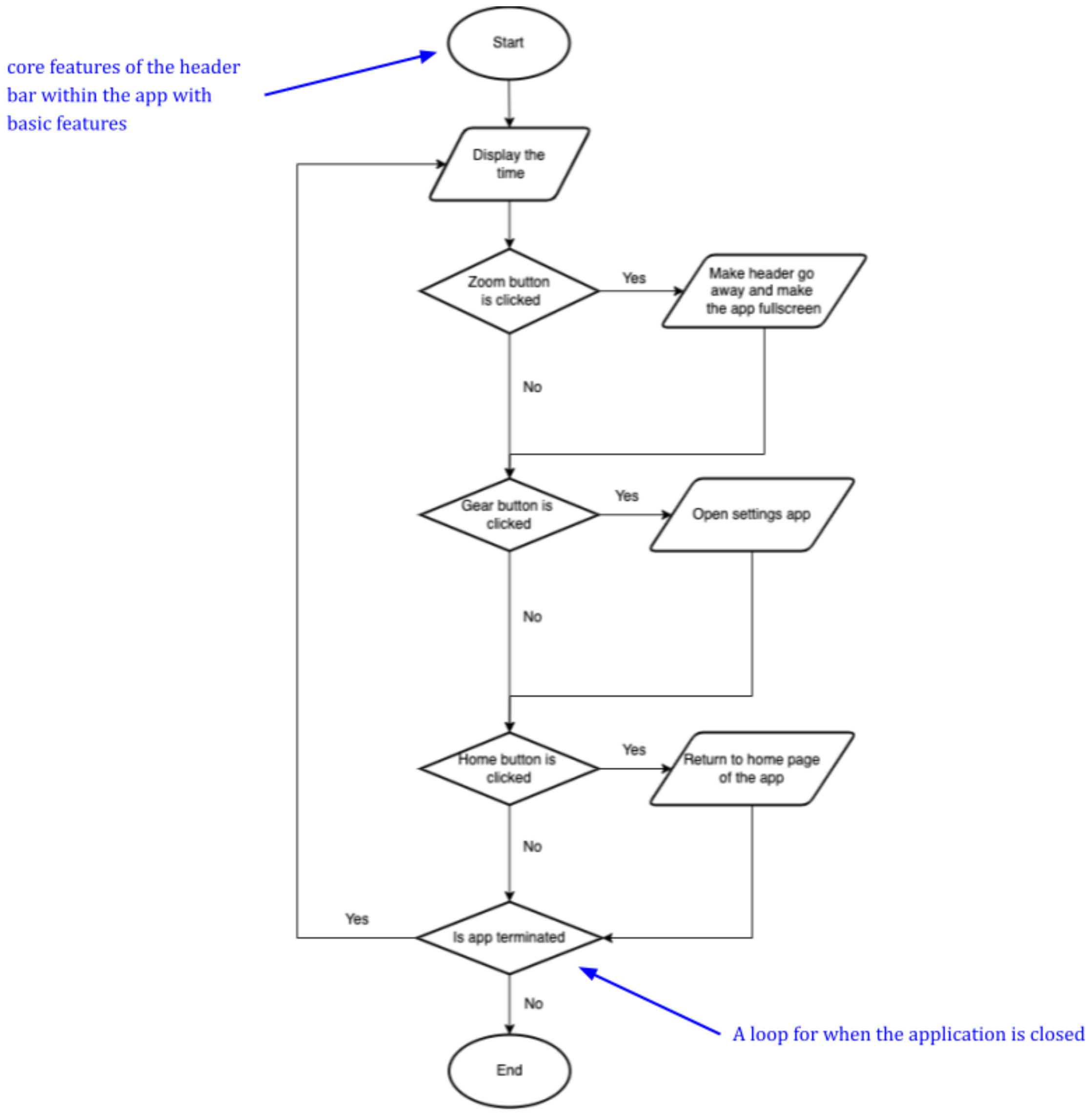

core features of the header bar within the app with basic features
Start
Display the time
Zoom button is clicked
Yes
Make header go away and make the app fullscreen
No
Gear button is clicked
Yes
Open settings app
No
Home button is clicked
Yes
Return to home page of the app
No
Yes
Is app terminated
No
A loop for when the application is closed
End

Figure 7: Header Bar Buttons flowchart diagram

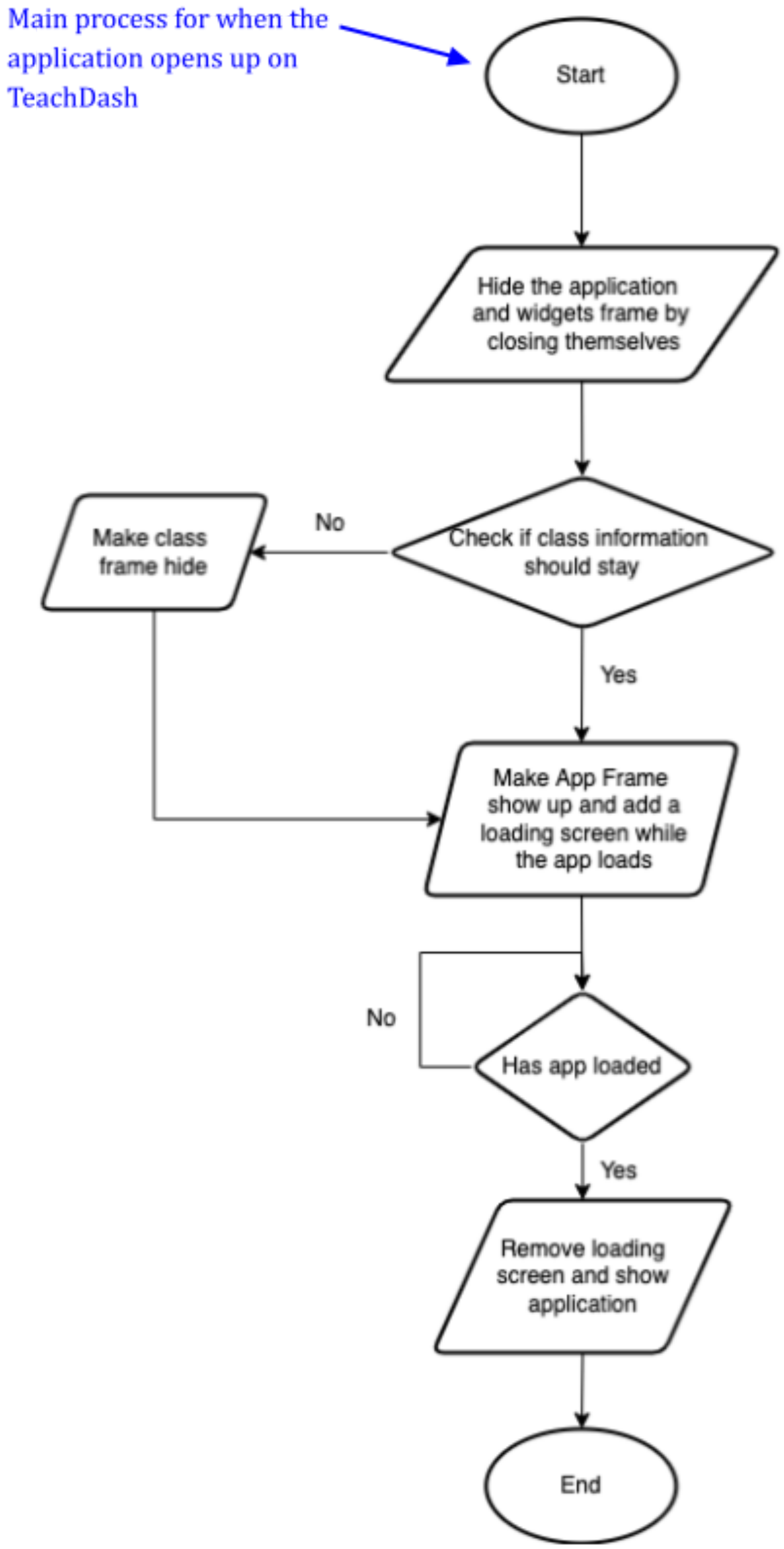

Main process for when the application opens up on TeachDash
Start
Hide the application and widgets frame by closing themselves
Check if class information should stay
No
Make class frame hide
Yes
Make App Frame show up and add a loading screen while the app loads
No
Has app loaded
Yes
Remove loading screen and show application
End

Figure 8: Changing Application Setup flowchart diagram

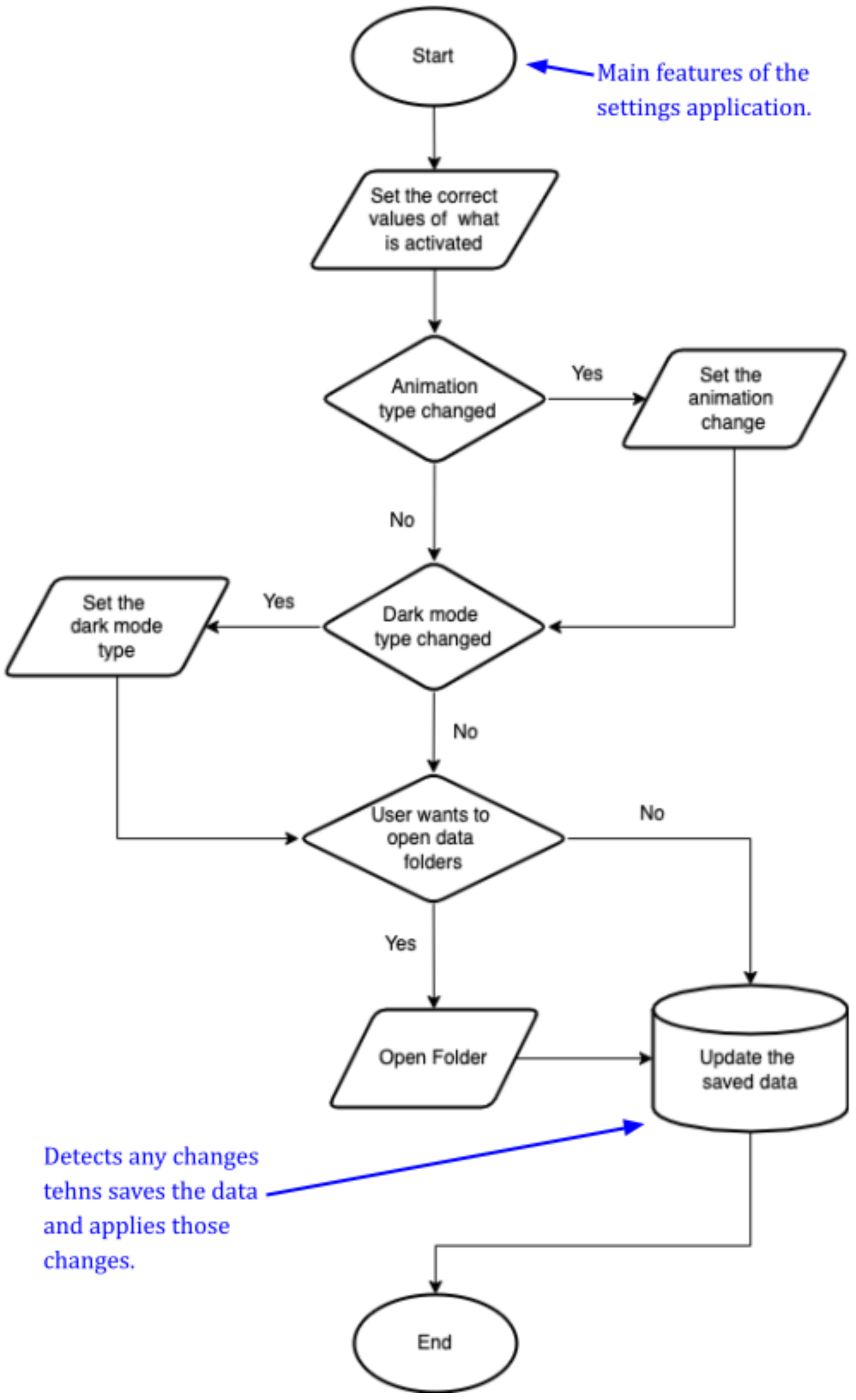
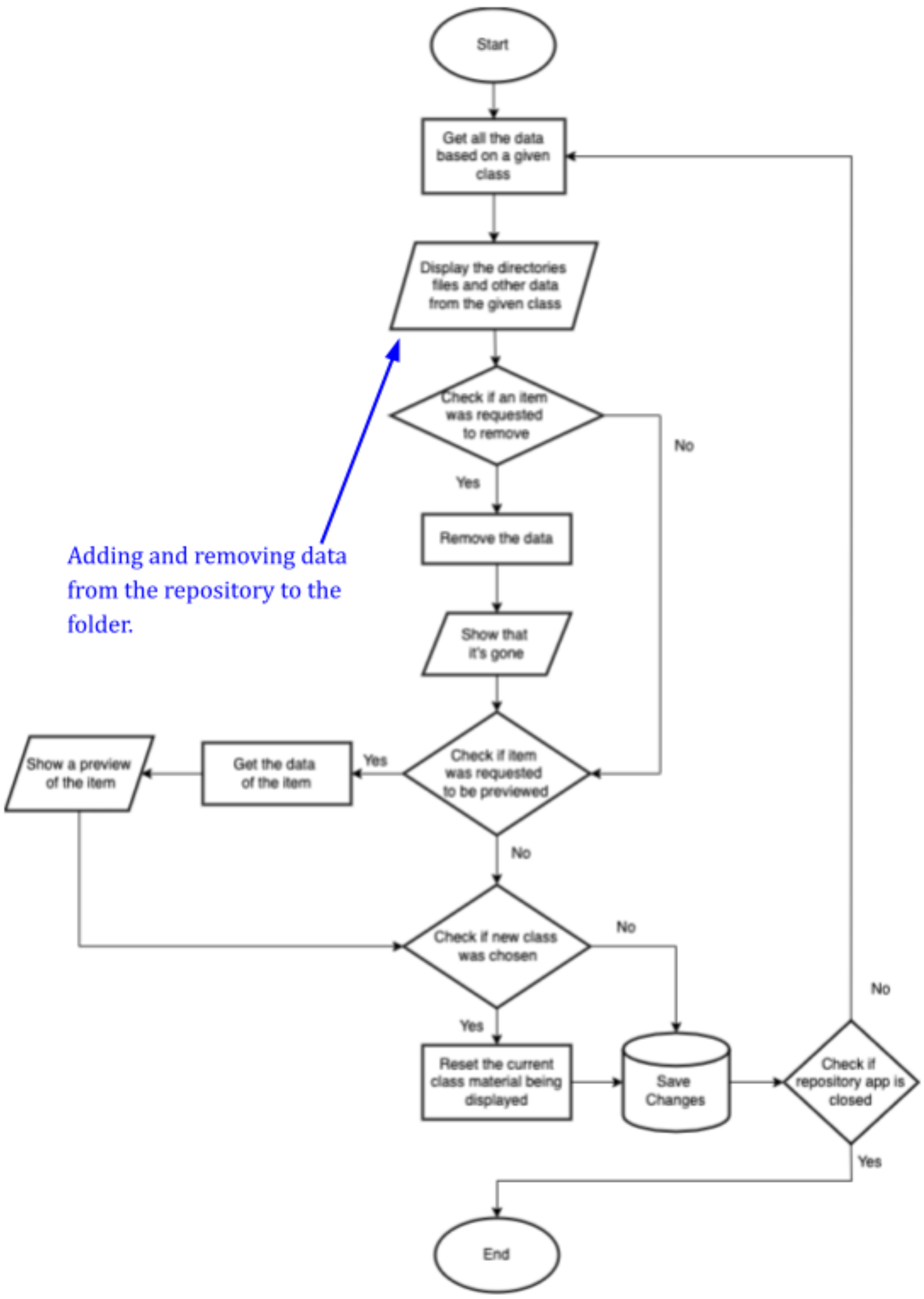

Start
Main features of the settings application.
Set the correct values of what is activated
Animation type changed
Yes
Set the animation change
No
Set the dark mode type
Yes
Dark mode type changed
No
User wants to open data folders
No
Yes
Open Folder
Update the saved data
Detects any changes tehns saves the data and applies those changes.
End

Figure 9: Settings Application flowchart diagram


Start
Get all the data based on a given class
Display the directories files and other data from the given class
Check if an item was requested to remove
No
Yes
Remove the data
Show that it's gone
Check if item was requested to be previewed
Yes
Get the data of the item
Show a preview of the item
No
Check if new class was chosen
No
Yes
Reset the current class material being displayed
Save Changes
Check if repository app is closed
No
Yes
End

Adding and removing data from the repository to the folder.

Figure 10: Repository Application flowchart diagram

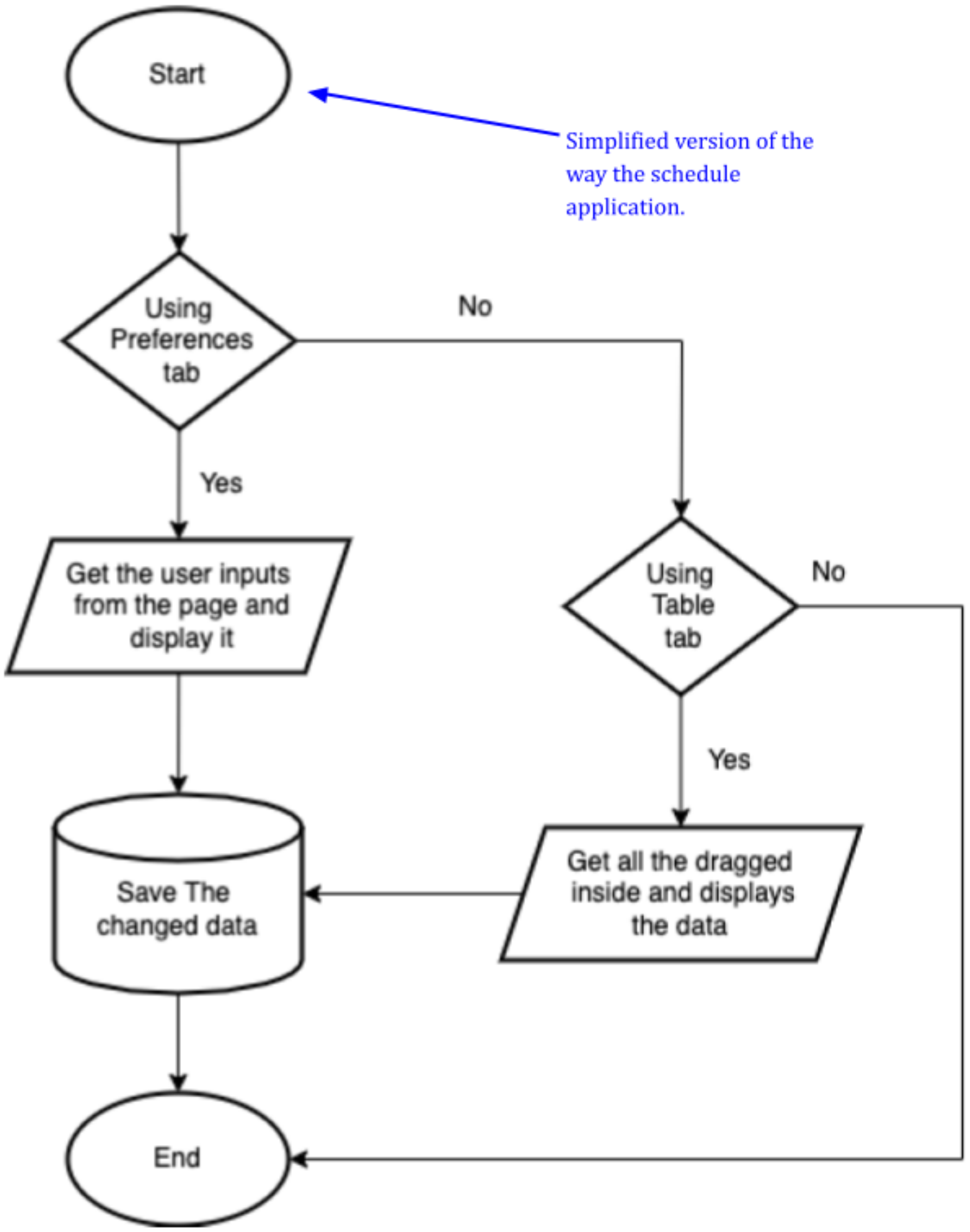

Start
Simplified version of the way the schedule application.
Using Preferences tab
No
Yes
Get the user inputs from the page and display it
Using Table tab
No
Yes
Save The changed data
Get all the dragged inside and displays the data
End

Figure 11: Schedule Application flowchart diagram

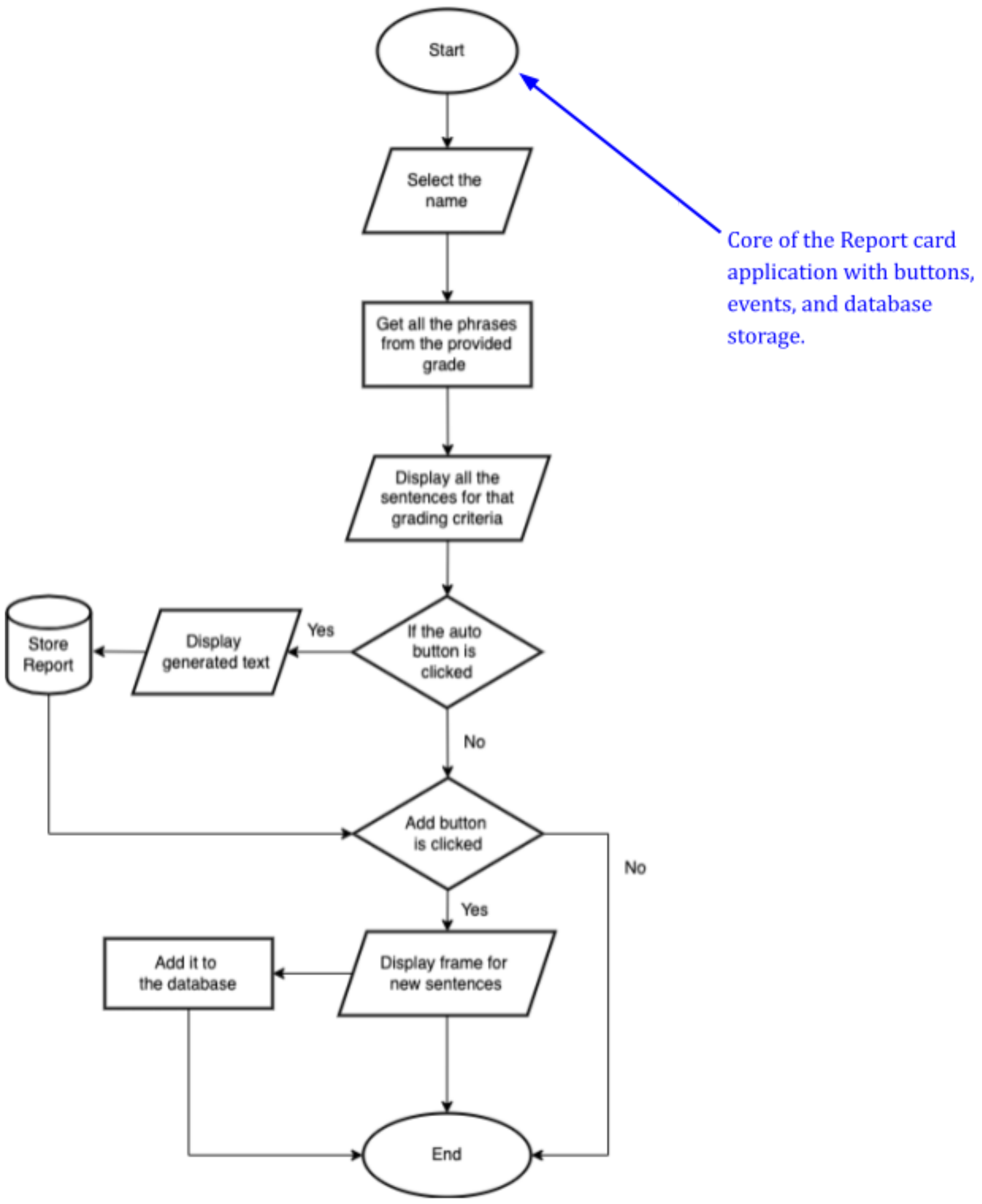

Start
Select the name
Get all the phrases from the provided grade
Display all the sentences for that grading criteria
If the auto button is clicked
Yes
Display generated text
Store Report
No
Add button is clicked
No
Yes
Display frame for new sentences
Add it to the database
End
Core of the Report card application with buttons, events, and database storage.

Figure 12: Report Card Application flowchart diagram

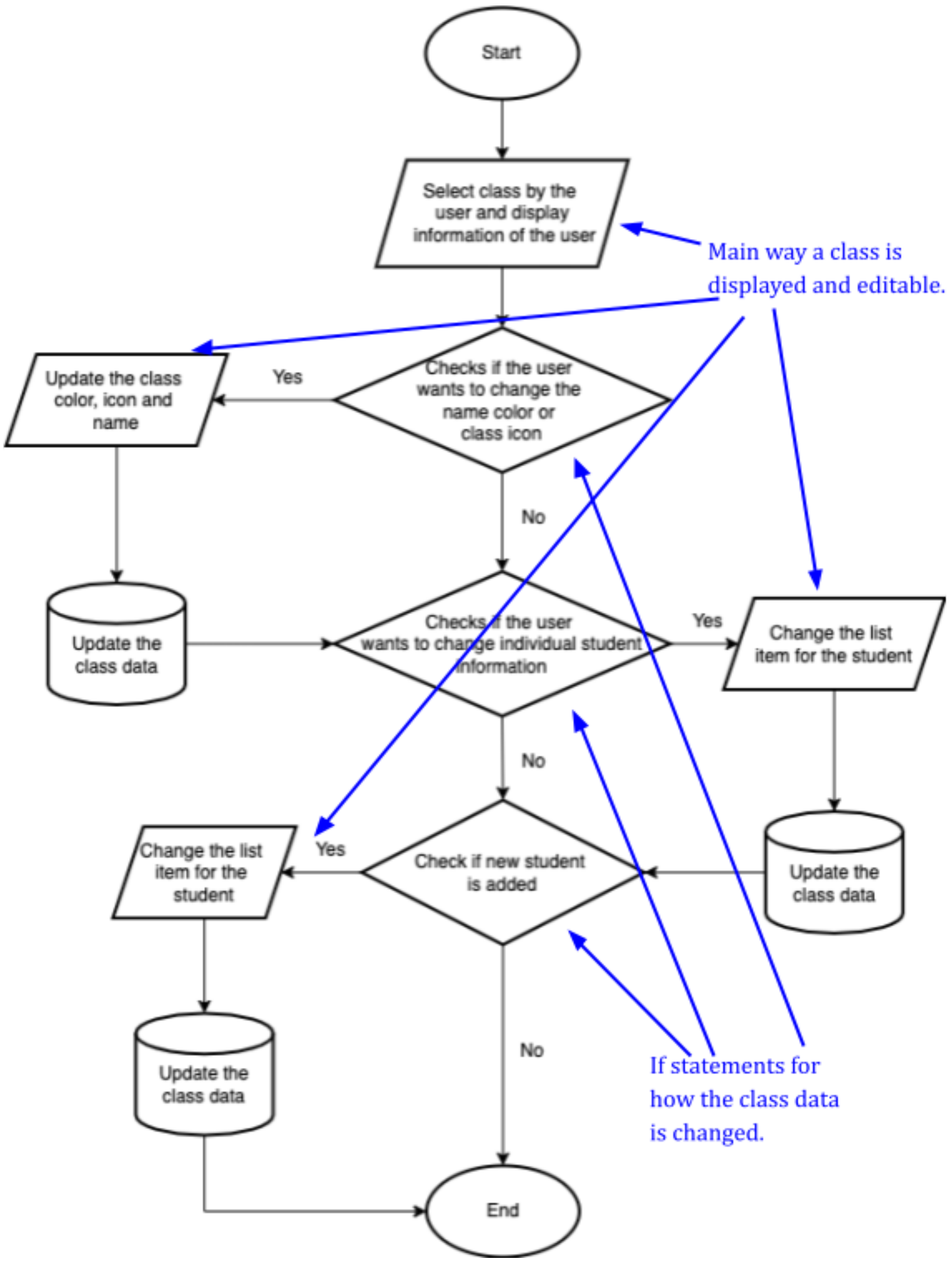

Start
Select class by the user and display information of the user
Main way a class is displayed and editable.
Checks if the user wants to change the name color or class icon
Yes
Update the class color, icon and name
No
Update the class data
Checks if the user wants to change individual student information
Yes
Change the list item for the student
Update the class data
No
Check if new student is added
Yes
Change the list item for the student
Update the class data
No
If statements for how the class data is changed.
End

Figure 13: Class Editor Application flowchart diagram

Graphical Visualization:

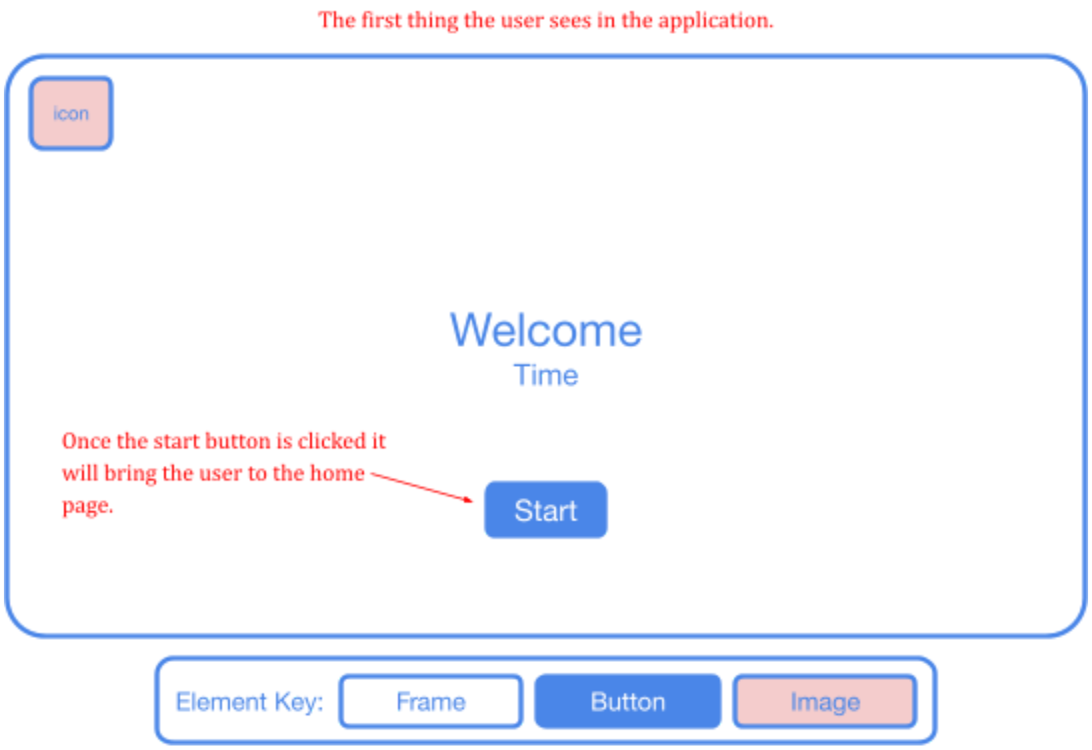

The first thing the user sees in the application.
icon
Welcome
Time
Once the start button is clicked it will bring the user to the home page.
Start
Element Key:
Frame
Button
Image

Figure 14: Welcome Screen Graphical Visualization

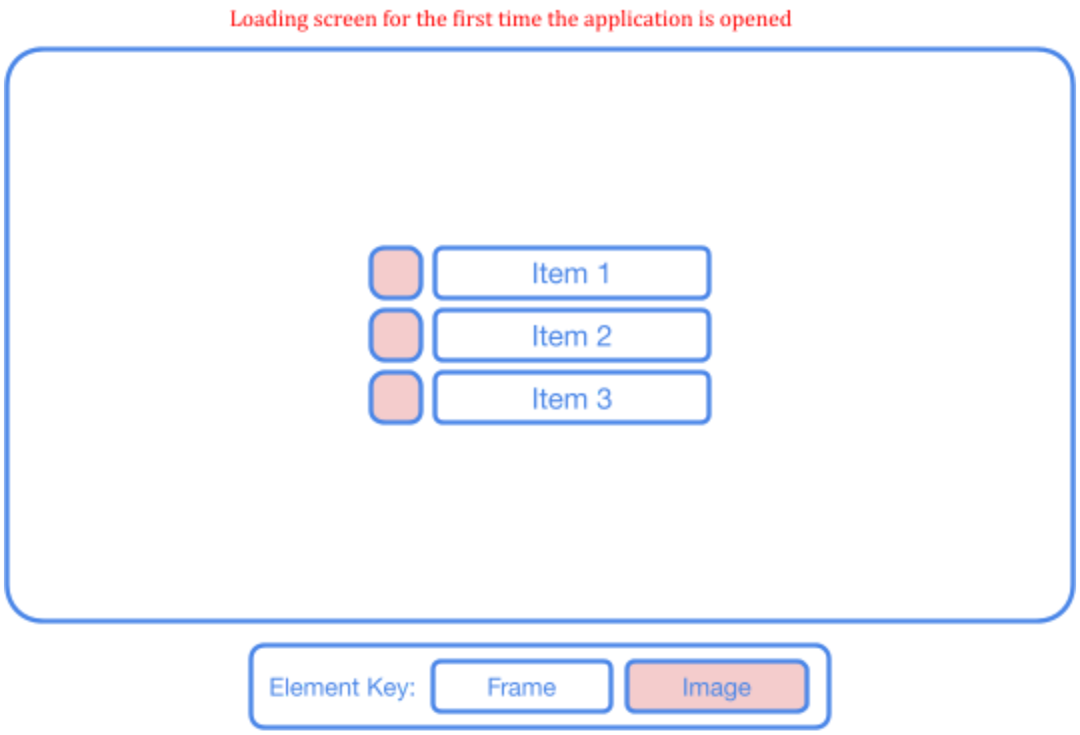


Figure 15: Loading Screen Graphical Visualization

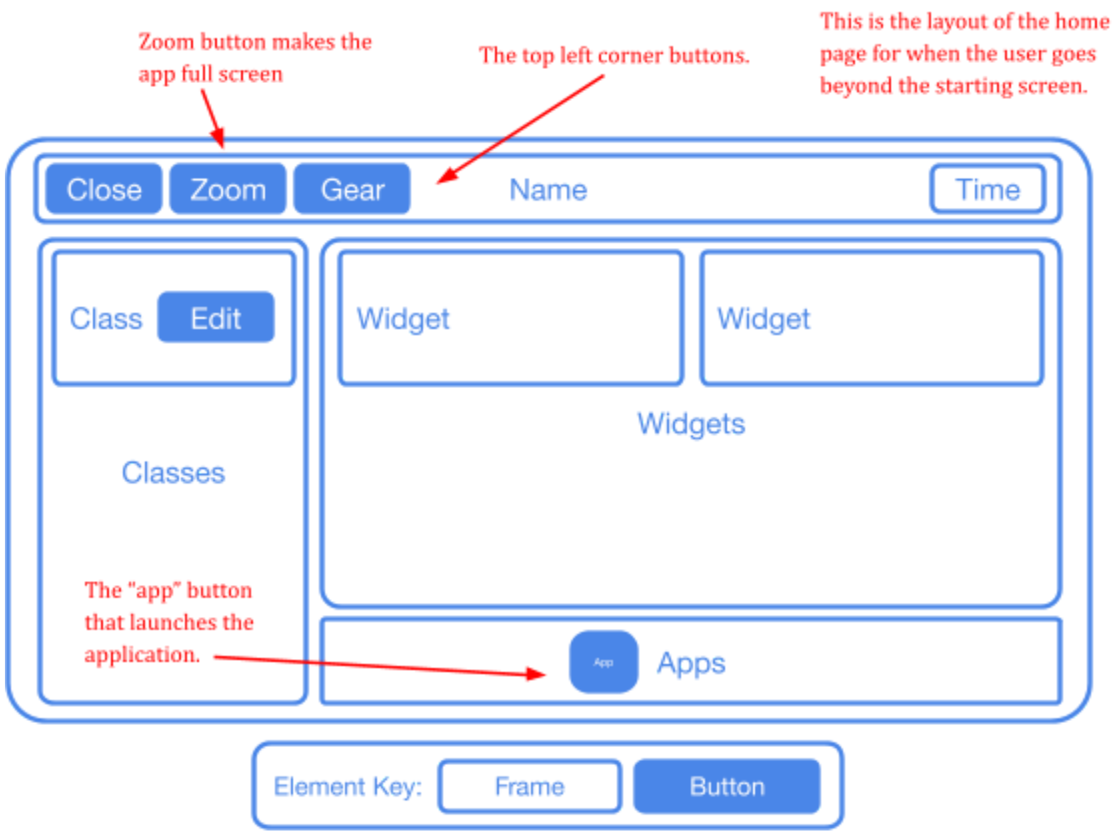

Zoom button makes the app full screen
The top left corner buttons.
This is the layout of the home page for when the user goes beyond the starting screen.
Close
Zoom
Gear
Name
Time
Class
Edit
Classes
Widget
Widget
Widgets
The "app" button that launches the application.
App
Apps
Element Key:
Frame
Button

Figure 16: Home Page Graphical Visualization

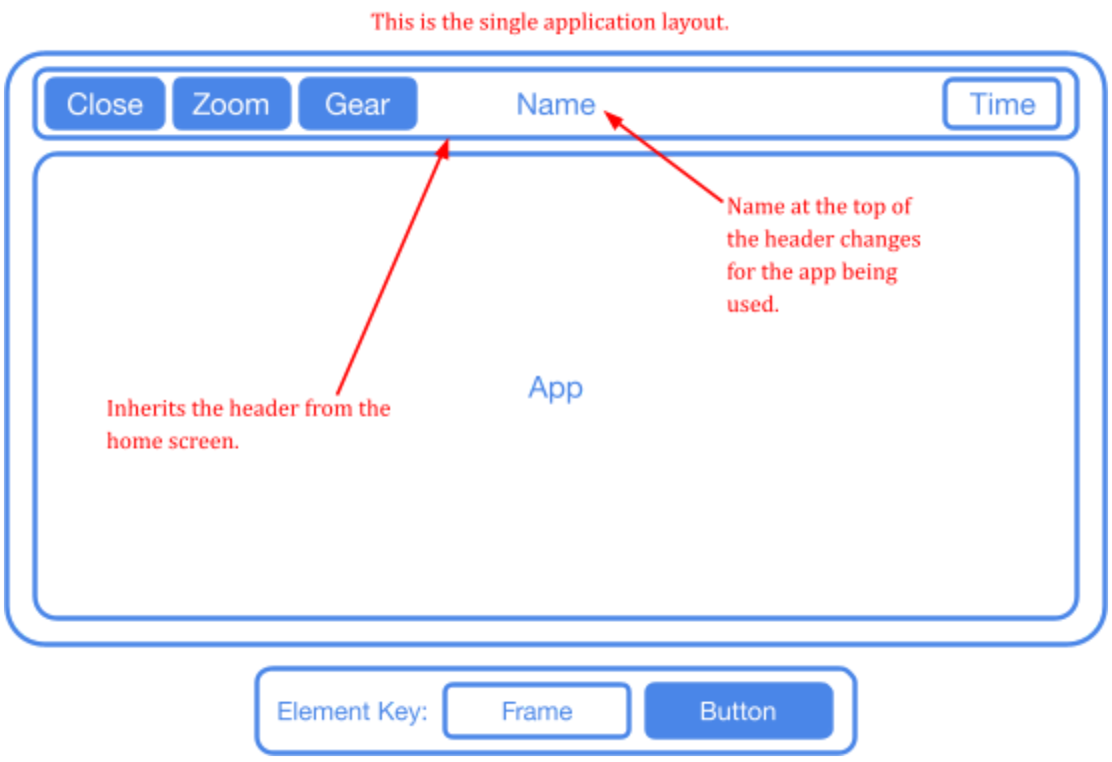

This is the single application layout.
Close
Zoom
Gear
Name
Time
Name at the top of the header changes for the app being used.
App
Inherits the header from the home screen.
Element Key:
Frame
Button

Figure 17: Single Application Layout Graphical Visualization

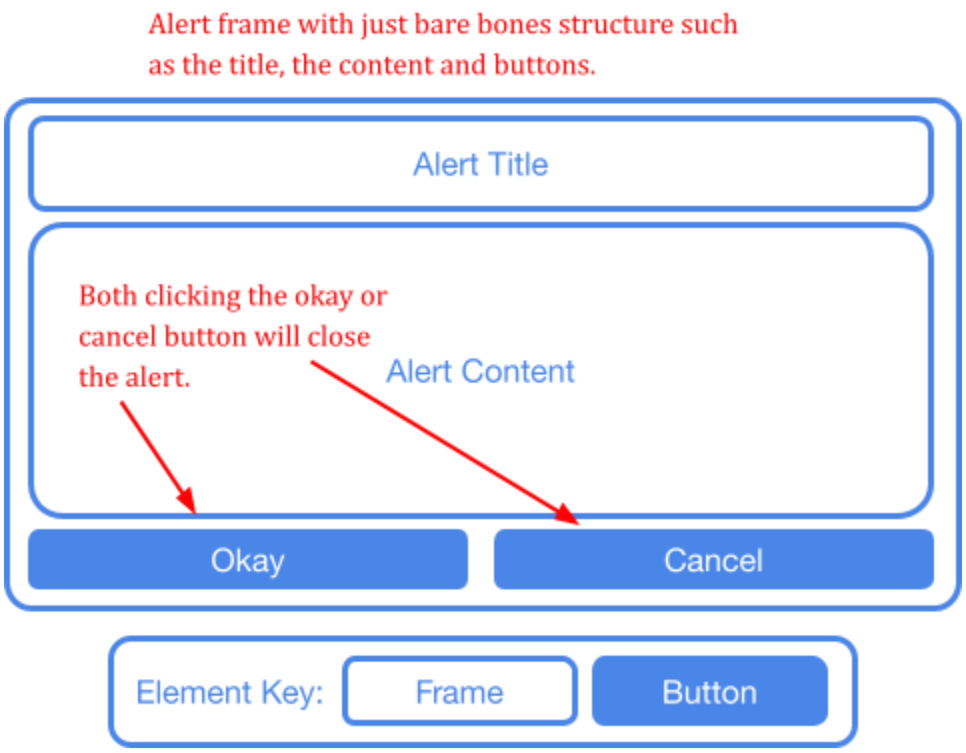

Alert frame with just bare bones structure such as the title, the content and buttons.
Alert Title
Both clicking the okay or cancel button will close the alert.
Alert Content
Okay
Cancel
Element Key:
Frame
Button

Figure 18: Alert System Graphical Visualization

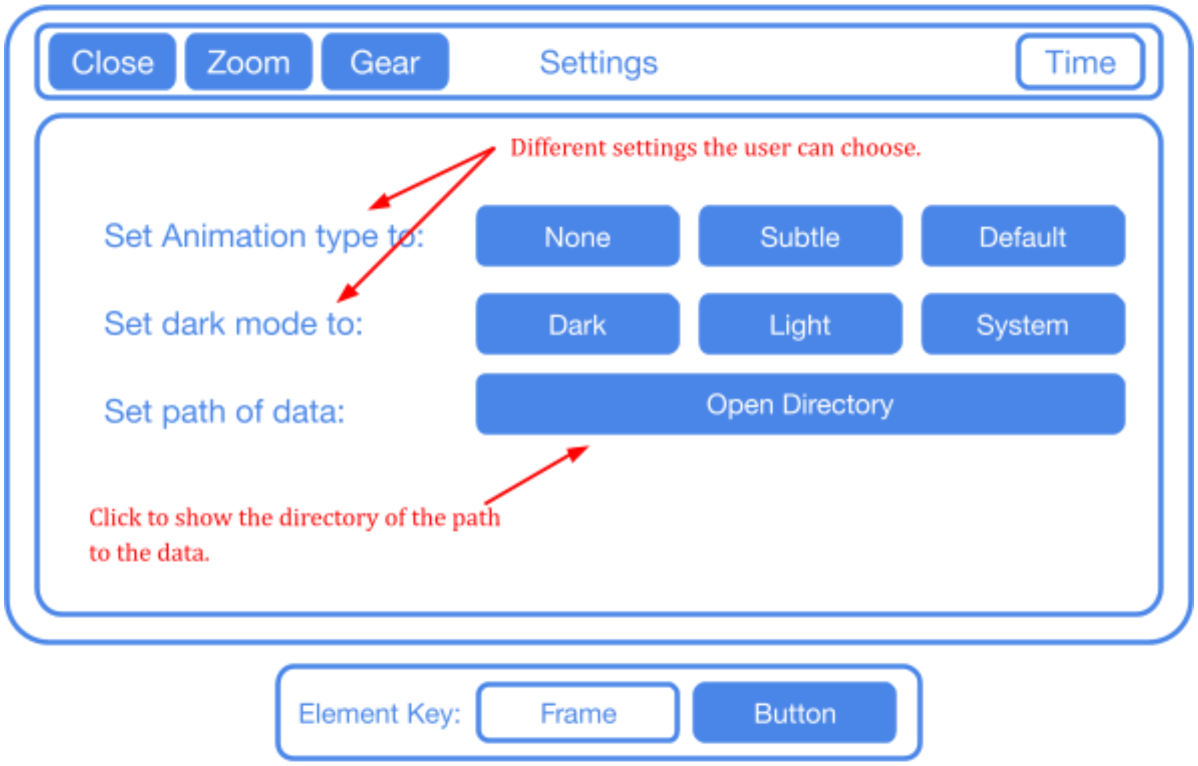


Figure 19: Settings Application Graphical Visualization

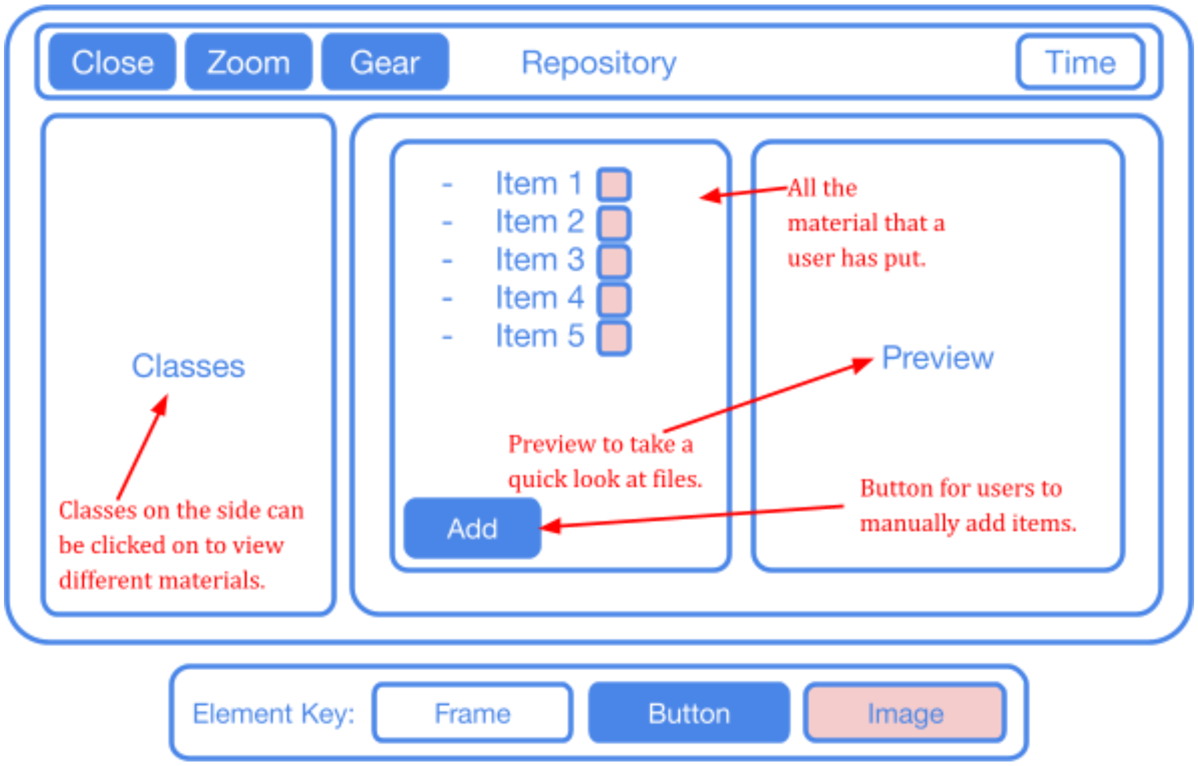

Close
Zoom
Gear
Repository
Time
Classes
Classes on the side can be clicked on to view different materials.
- Item 1
- Item 2
- Item 3
- Item 4
- Item 5
All the material that a user has put.
Preview
Preview to take a quick look at files.
Add
Button for users to manually add items.
Element Key:
Frame
Button
Image

Figure 20: Repository Application Graphical Visualization

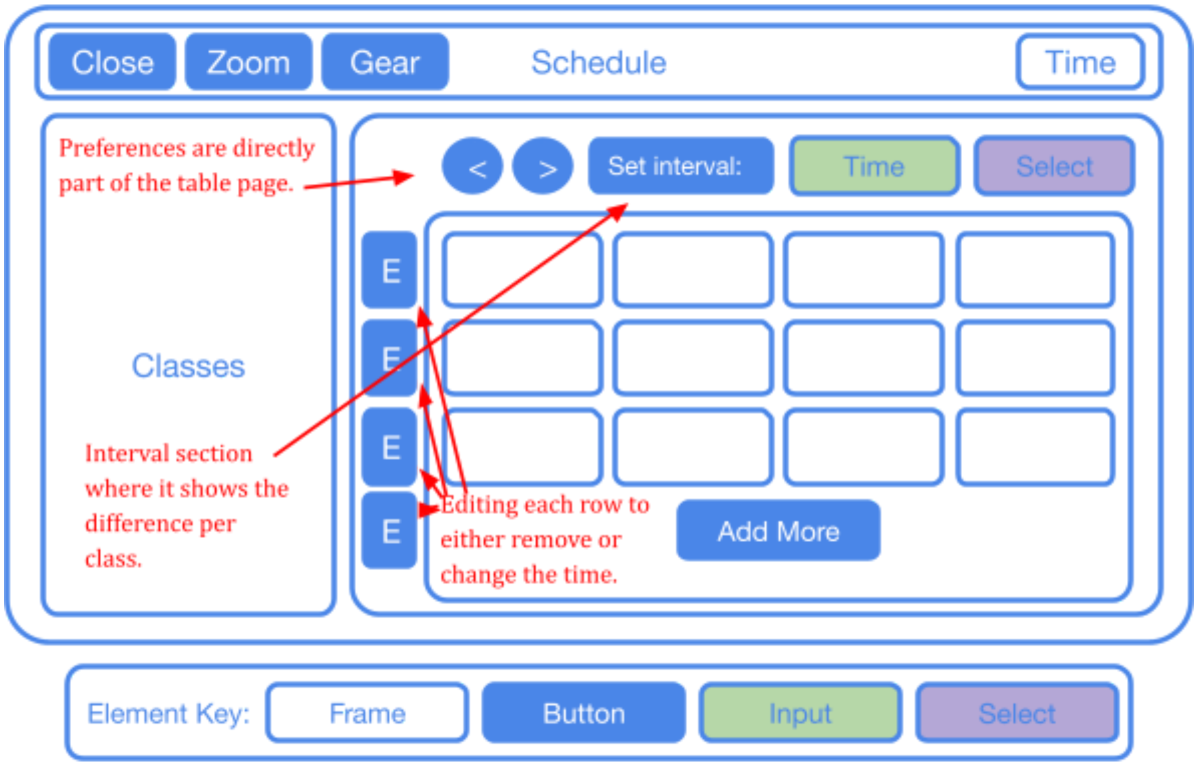

Close
Zoom
Gear
Schedule
Time
Preferences are directly part of the table page.
<
>
Set interval:
Time
Select
Classes
Interval section where it shows the difference per class.
E
E
E
E
Editing each row to either remove or change the time.
Add More
Element Key:
Frame
Button
Input
Select

Figure 21: Schedule Application Graphical Visualization

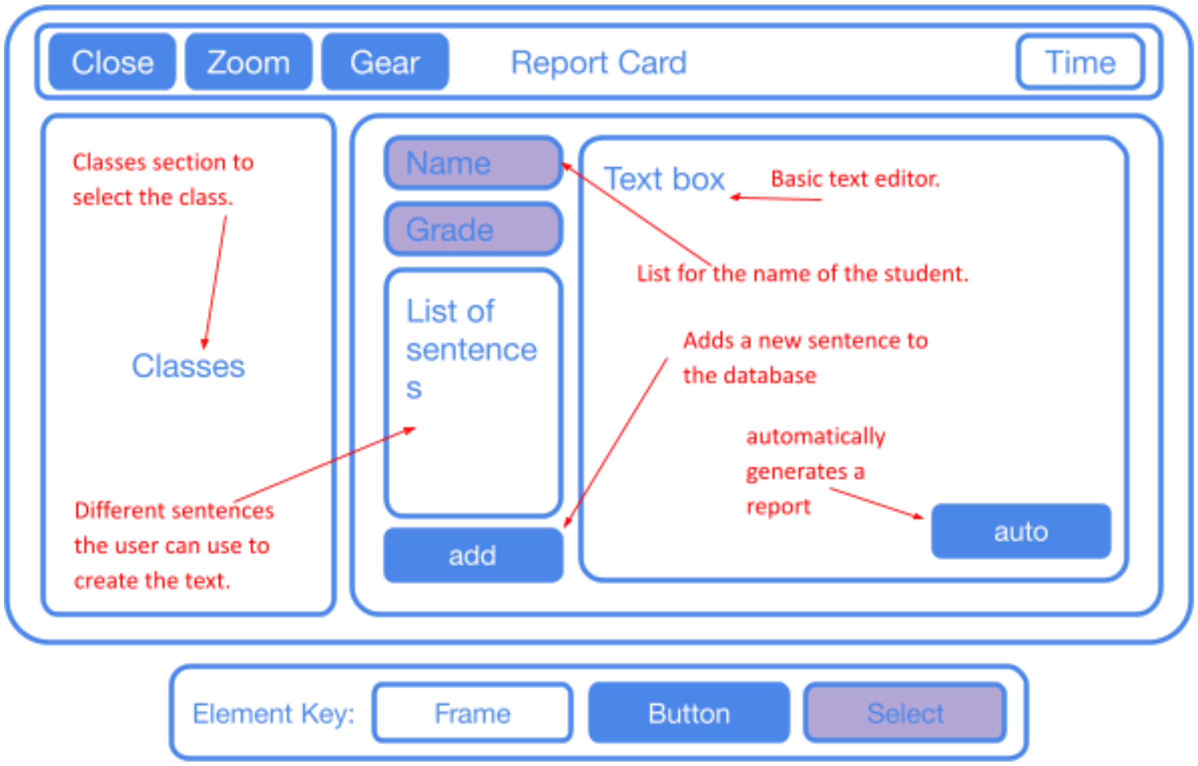

Close
Zoom
Gear
Report Card
Time
Classes section to select the class.
Classes
Different sentences the user can use to create the text.
Name
Grade
List of sentences
add
Text box
Basic text editor.
List for the name of the student.
Adds a new sentence to the database
automatically generates a report
auto
Element Key:
Frame
Button
Select

Figure 22: Report Card Application Graphical Visualization

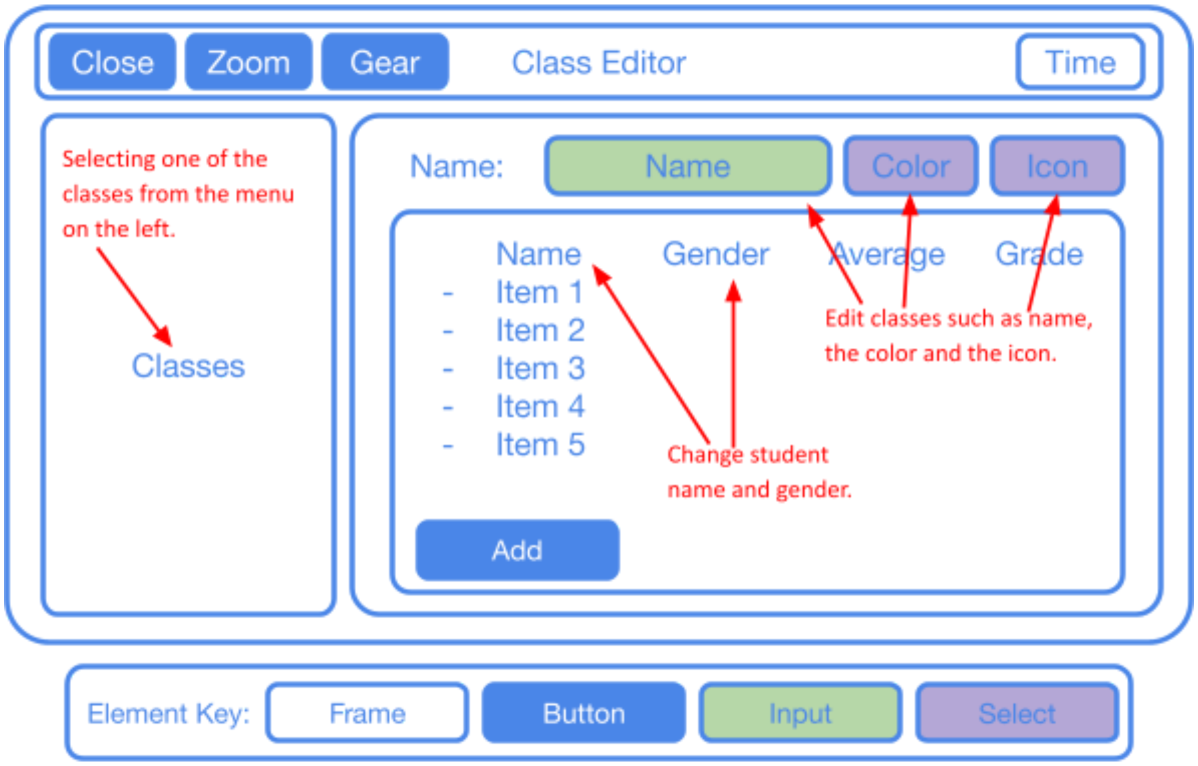


Figure 23: Class Editor Application Graphical Visualization

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 seperated 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.

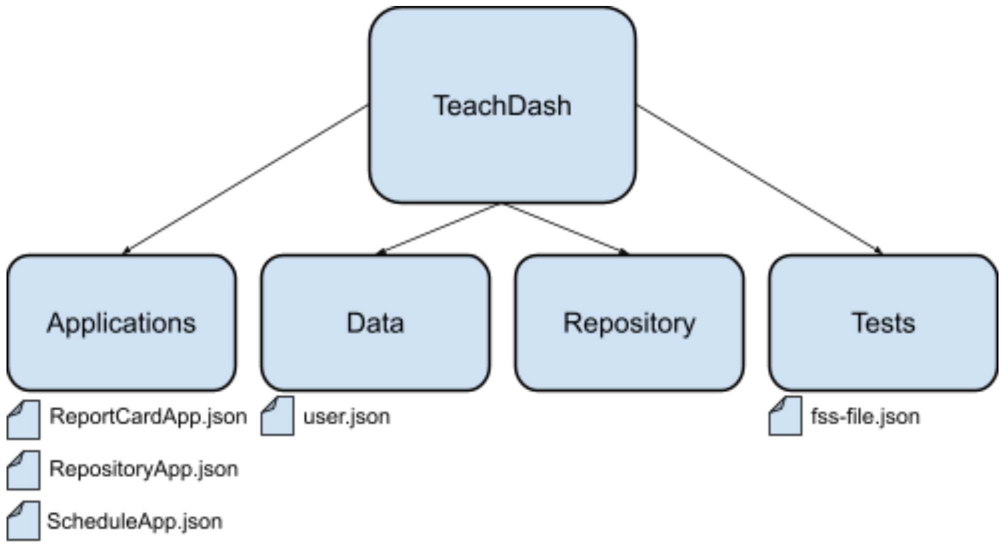


Figure 24: File General Saving Data

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