***Software Engineering***

***Software Requirements Specification***

***(SRS) Document***

**<Schooly>**

**<September 13, 2022>**

**<1.0.3>**

**<By: Andrew van Es, Evan Corsar, Nishant Sharma>**

**<Keep all information secure and safe. Be reliable and ensure ease of use.>**

**Table of Contents**

[***1 Introduction***](#_heading=h.gjdgxs) ***4***

[**1.1 Purpose**](#_heading=h.30j0zll) **4**

[**1.2 Document conventions**](#_heading=h.1fob9te) **4**

[**1.3 Definitions, Acronyms, and Abbreviations**](#_heading=h.3znysh7) **4**

[**1.4 Intended audience**](#_heading=h.2et92p0) **4**

[1.4.1 General Description](#_heading=h.tyjcwt) 5

[1.4.2 Functional Requirements](#_heading=h.3dy6vkm) 5

[1.4.3 Technical Requirements](#_heading=h.1t3h5sf) 5

[1.4.4 Non-Functional Requirements](#_heading=h.4d34og8) 5

[**1.5 Project Scope**](#_heading=h.2s8eyo1) **5**

[**1.6 Technology Challenges**](#_heading=h.17dp8vu) **5**

[**1.7 References**](#_heading=h.3rdcrjn) **5**

[***2 General Description***](#_heading=h.26in1rg) ***6***

[**2.1 Product perspective**](#_heading=h.lnxbz9) **6**

[**2.2 Product features**](#_heading=h.35nkun2) **6**

[**2.3 User class and characteristics**](#_heading=h.1ksv4uv) **6**

[**2.4 Operating environment**](#_heading=h.44sinio) **6**

[**2.5 Constraints**](#_heading=h.2jxsxqh) **6**

[**2.6 Assumptions and dependencies**](#_heading=h.z337ya) **7**

[***3 Functional Requirements***](#_heading=h.3j2qqm3) ***7***

[**3.1 Primary**](#_heading=h.1y810tw) **7**

[**3.2 Secondary**](#_heading=h.4i7ojhp) **7**

[***4 Technical Requirements***](#_heading=h.2xcytpi) ***7***

[**4.1 Operating System & Compatibility**](#_heading=h.1ci93xb) **7**

[**4.2 Interface requirements**](#_heading=h.3whwml4) **8**

[4.2.1 User Interfaces](#_heading=h.2bn6wsx) 8

[4.2.2 Hardware Interfaces](#_heading=h.qsh70q) 8

[4.2.3 Communications Interfaces](#_heading=h.3as4poj) 8

[4.2.4 Software Interfaces](#_heading=h.1pxezwc) 8

[***5 Non-Functional Requirements***](#_heading=h.49x2ik5) ***9***

[**5.1 Performance requirements**](#_heading=h.2p2csry) **9**

[**5.2 Safety requirements**](#_heading=h.147n2zr) **9**

[**5.3 Security requirements**](#_heading=h.3o7alnk) **9**

[**5.4 Software quality attributes**](#_heading=h.23ckvvd) **9**

[5.4.1 Availability](#_heading=h.ihv636) 9

[5.4.2 Correctness](#_heading=h.32hioqz) 9

[5.4.3 Maintainability](#_heading=h.1hmsyys) 9

[5.4.4 Reusability](#_heading=h.41mghml) 10

[5.4.5 Portability](#_heading=h.2grqrue) 10

[**5.5 Process Requirements**](#_heading=h.vx1227) **10**

[5.5.1 Development Process Used](#_heading=h.3fwokq0) 10

[5.5.2 Time Constraints](#_heading=h.1v1yuxt) 10

[5.5.3 Cost and Delivery Date](#_heading=h.4f1mdlm) 10

[**5.6 Other requirements**](#_heading=h.2u6wntf) **10**

[**6 Software Architecture**](#_heading=h.hp74rk6xy2c1) **12**

[6.1 MVC Diagram:](#_heading=h.ipjtemyhptfj) 12

[6.2 Database Schema](#_heading=h.wsj1a1jubnbm) 13

[**7 Software Design**](#_heading=h.gy5n6abdp3f5) **14**

[7.1 State Diagram](#_heading=h.2re68hfuy3v8) 14

[7.1.1 Admin State Diagram](#_heading=h.fll5vzdk3b40) 14

[7.1.2 Student State Diagram](#_heading=h.46c6j5fki3jg) 15

[7.1.3 Teacher State Diagram](#_heading=h.a8173vney2za) 16

[7.2 UML diagram](#_heading=h.jarzhwrh3a12) 17

[**8 Appendix**](#_heading=h.g1yvdadkltp2) **21**

[8.1 Use-Case Model](#_heading=h.u3m6i3wgfwrn) 21

# Introduction

## Purpose

Schooly is designed to help students and instructors stay organized with their various classes. Whether that be to keep track of assignments due or stay on top of grading assignments, Schooly allows students and instructors to stay ahead with their respective courses.

## **Document conventions**

The purpose of this Software Requirements Document (SRD) is to describe the various client-views and developer-view required for Schooly. Client-orientated requirements describe the various views that a student, instructor, and administrator may observe. Developer-orientated requirements describe the system from a technical perspective. Requirements include in-depth descriptions of functionality, data and data storage, and other implementations of features.

## Definitions, Acronyms, and Abbreviations

| **Term** | **Definition. Acronym, Abbreviation** |
| --- | --- |
| DBMS | An abbreviation for Database Management System. |
| MS | An abbreviation for Microsoft. Microsoft is a large software company which produces the software that will be used to implement ATPS. |
| UI | An abbreviation for User Interface. |

## **Intended audience**

TODO: This section is intended to describe which part of the SRS document is intended for which reader. Include a list of all stakeholders of the project, developers, project managers, and users for better clarity.

### **General Description**

This section is intended for all readers to understand the basics and non-technical specifications of the project.

### **Functional Requirements**

This section is intended for developers, project managers, and other more advanced stakeholders to understand the complex and more technical terms and requirements for this project.

### **Technical Requirements**

This section is like the Functional Requirements. This is more intended for the developers and teams working on this project to understand the needs and issues that may occur during the development.

### **Non-Functional Requirements**

This section is intended for all users that need an understanding of time constraints, future plans for this project, and other legal requirements.

## **Project Scope**

Specify how the software goals align with the overall business goals and outline the benefits of the project to the business.

Schooly is designed to keep students and teachers on track and organized. Allowing students and teachers access to to-do lists, view assignments, and other various information for productivity will allow both parties to be well organized and decrease stress-levels all together.

## Technology Challenges

**N/A**

## References

**N/A**

# General Description

## **Product perspective**

During our time at high school, we did not have access to the apps like Canvas that allowed us to view the list of assignments, our grades and other school-related assignments. We think that if we had access to a similar kind of app, we would not have missed the due dates or could have planned accordingly for upcoming tests. Therefore, we thought that designing an app that will help students keep track of their coursework can really help them succeed.

## **Product features**

Schooly is designed for students and teachers alike. This is intended to keep students and teachers on track and organized throughout the semesters. Allowing students to view **to-do** **lists**, **assigned works, view their grades**, and other **school-related assignments.** Similarly, teachers will be able to view **students**, and **courses**, **grades for all students**, and **assign work.** Additionally, there will be a third party, an administrator, that will be allowed to **overwrite grades, manage teachers' assignments, and view all students' grades.**

## **User class and characteristics**

Our application expects users to understand the basics of web apps but will be intuitive enough for those who have little experience with computers altogether. Our web app will be easy to follow and clearly labeled for each action a user may need to make. The important information will be displayed clearly at a glance for the user to access quickly.

## **Operating environment**

This web application is being designed for any educational environment ranging from Elementary Schools to College Campuses. The application could also be used within the home to help students finish their school assignments on time.

## Constraints

This application is meant for students and teachers alike, which could pose a problem as the computer literacy skills of some of the users could lead to some issues if the interfaces of the application are too confusing. In turn, the application should be easy to both navigate and simple to use.

## Assumptions and dependencies

Tentatively, we are planning to implement a calendar API in our application. It will help instructors to assign a task or a quick on a certain date. Also, it helps students to view what task is due on which date.

As of now, we are planning to use the Spring framework which is an excellent open-source application framework that provides infrastructure support for developing a Java application.

Vaadin is one of the dependencies that we might use but we are still at an early stage, so we need to explore more. Since we do not have much knowledge in JS, HTML, and CSS, this is an excellent framework that can assist us in writing the UI.

Thymeleaf is another dependency that we might use that will help us design our HTML files.

# Functional Requirements

## Primary

* FR0: The system will allow the admin to create or delete instructors and students.
* FR1: The system will allow the admin to lookup for the lists of instructors and students along their associated courses.
* FR2: The system will allow a course instructor to create or delete assignments and tests.
* FR3: The system will allow students to enroll into a course.
* FR4: The system will allow students to view the assignments and upcoming test dates for a course.
* FR5: The system will allow students to mark an assignment as complete or important or incomplete.

## Secondary

* FR0: The system will store created users’ information in a database
* FR1: The system will store created assignments and grades in a database

# Technical Requirements

## Operating System & Compatibility

* Compatible with most modern browsers (Mozilla, Chrome, Microsoft Edge, Safari).
* Compatible with most modern operating systems that can run the above browsers.

## Interface requirements

### User Interfaces

The UI will allow the user to sign in and out. It will allow users to click through various tabs to view different information and screens. For students, there will be grades, to-do, and course tabs; for Instructors, there will be similar tabs except for to-do being replaced with a tab to create assignments. The ‘grades’ tab for instructors will be to view all grades of all students. Administrators will have access to all grades of all students, edit and manage instructors and courses, and be allowed to remove or add students to a course.

The logic behind the interactions between the users and the software. This includes the sample screen layout, buttons and functions that would appear on every screen, messages to be displayed on each screen and the style guides to be used.

### Hardware Interfaces

* Any Microsoft Windows OS that can run a current browser software
* Any Apple OS that can run a current browser software
* A stable connection to the internet
* HTTP will be used to access the web-application

All the hardware-software interactions with the list of supported devices on which the software is intended to run on, the network requirements along with the list of communication protocols to be used.

### Communications Interfaces

TODO: Determination of all the communication standards to be utilized by the software as a part of the project

### Software Interfaces

* **Spring Framework** will be used for the backend framework in developing the Java web application.
* **Vaadin** is a dependency that will be used as a framework to assist in designing the UI.
* **MySQL** will likely be used as the DBMS.

The interaction of the software to be developed with other software components such as frontend and the backend framework to be used, the database management system and libraries describing the need and the purpose behind each of them.

# Non-Functional Requirements

## Performance requirements

* NFR1(R): The system will consume no more than 100MB of memory and storage.
* NFR2(R): The novice user (instructor, administrator, or student) should be able to traverse the web application easily and with few problems. They will be able to become comfortable in less than a week of use.
* NFR3(R): The expert user will be able to quickly check grades/grade assignments and feel very comfortable with the layout.

## Safety requirements

* The implementation of regular system Backups to prevent loss/manipulation of data.
* Implementation of features that allow users to undo certain types of actions (Within Reason).
* Implementation of Administrative overwriting privileges.

## Security requirements

* NFR5(R): Administrators and instructors will be required to have a strong password
* NFR6(R): Multi-factor authentication will be required for all users to ensure the security of grades, assignments, and other valuable data.

## Software quality attributes

### Availability

This web-application will be available for all schooling system that may need it. Students, teachers, and administrators will have access to their respective views and information.

### Correctness

User testing and review will be undergone to ensure correctness of this web application.

### Maintainability

As part of user testing, bugs that could be fatal will be corrected efficiently. In cases that the web-app may fail, we will have a maintainer monitoring the system and servers.

### Reusability

Schooly will be able to be reused throughout various schools and university settings.

### Portability

To continue to ensure correctness, this web-application will be ported to only laptops / personal computers. There will be room to expand into the mobile / tablet realm at a later date.

Detailing on the additional qualities that need to be incorporated within the software like maintainability, adaptability, flexibility, usability, reliability, portability etc.

## Process Requirements

### Development Process Used

Incremental Development Model (Agile)

### Time Constraints

Time is a limited factor that is determined by the final product required to be done by November 15 - November 22.

### Cost and Delivery Date

Cost estimate is $0 and a delivery date of November 15 - 22, 2022.

## Other requirements

* NFR4(R): The system will conform to FERPA guidelines to maintain student privacy.
* NFR7(R): Update ability to be ported to mobile devices.

All SRS/SRD should be:

● **Correct:** A method of analysis that ensures that the software meets the requirements identified.

● **Unambiguous:** There is only one interpretation of what the software will be used for, and it is communicated in a common language.

● **Complete:** There is a representation for all requirements for functionality, performance, design constraints, attributes, or external interfaces.

● **Consistent:** Must be in agreement with other documentation, including a systems requirements specification and other documents.

● **Ranked for Importance and/or Stability:** Since all requirements are not of equal weight, you should employ a method to appropriately rank requirements.

● **Verifiable:** Use measurable elements and defined terminology to avoid ambiguity.

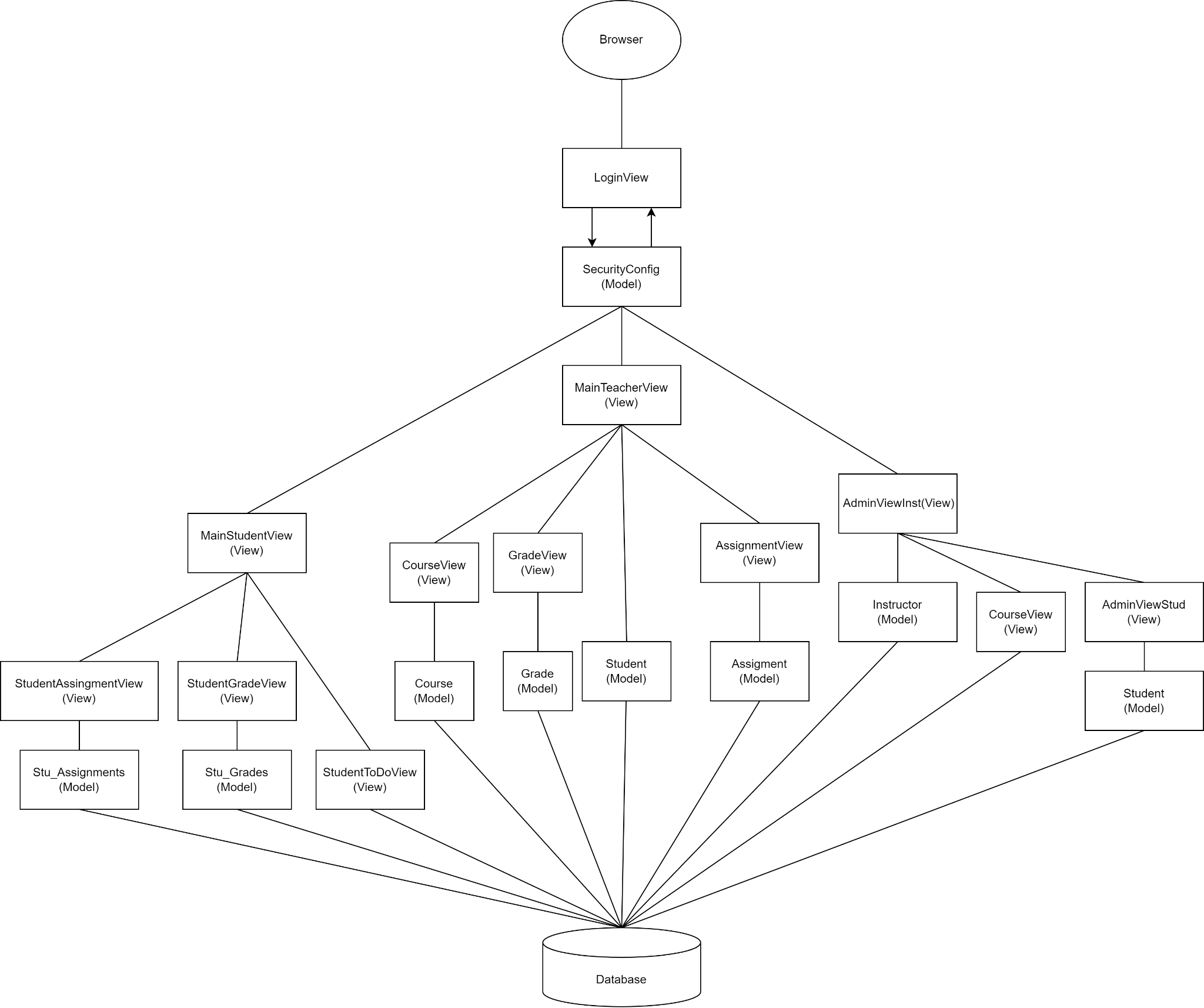
● **Modifiable:** A well-defined organizational structure of the SRS document that avoids redundancies can allow easy adaptation.

● **Traceable:** Ability to trace back to the origin of development and move forward to the documents produced from the SRS.

● **Legible and Professionally Presented**: Must use a consistent font and style. Must have proper formatting of tables and charts. Must be grammatically correct. Use active tense and concise sentences.

# Software Architecture

## MVC Diagram:



This project uses the Vaadin framework and does not really use the Spring MVC which is the original Spring web framework. Instead, Vaadin provides a Router class that helps to navigate through the application and structure the views into logical parts. Furthermore, there are event classes that help with the functionality of the application. For instance, there are functions such as click listeners. Whenever the user clicks on a button, the click listener function will register the click and do what the designer wants it to execute.

## Database Schema

course(courseID,name)

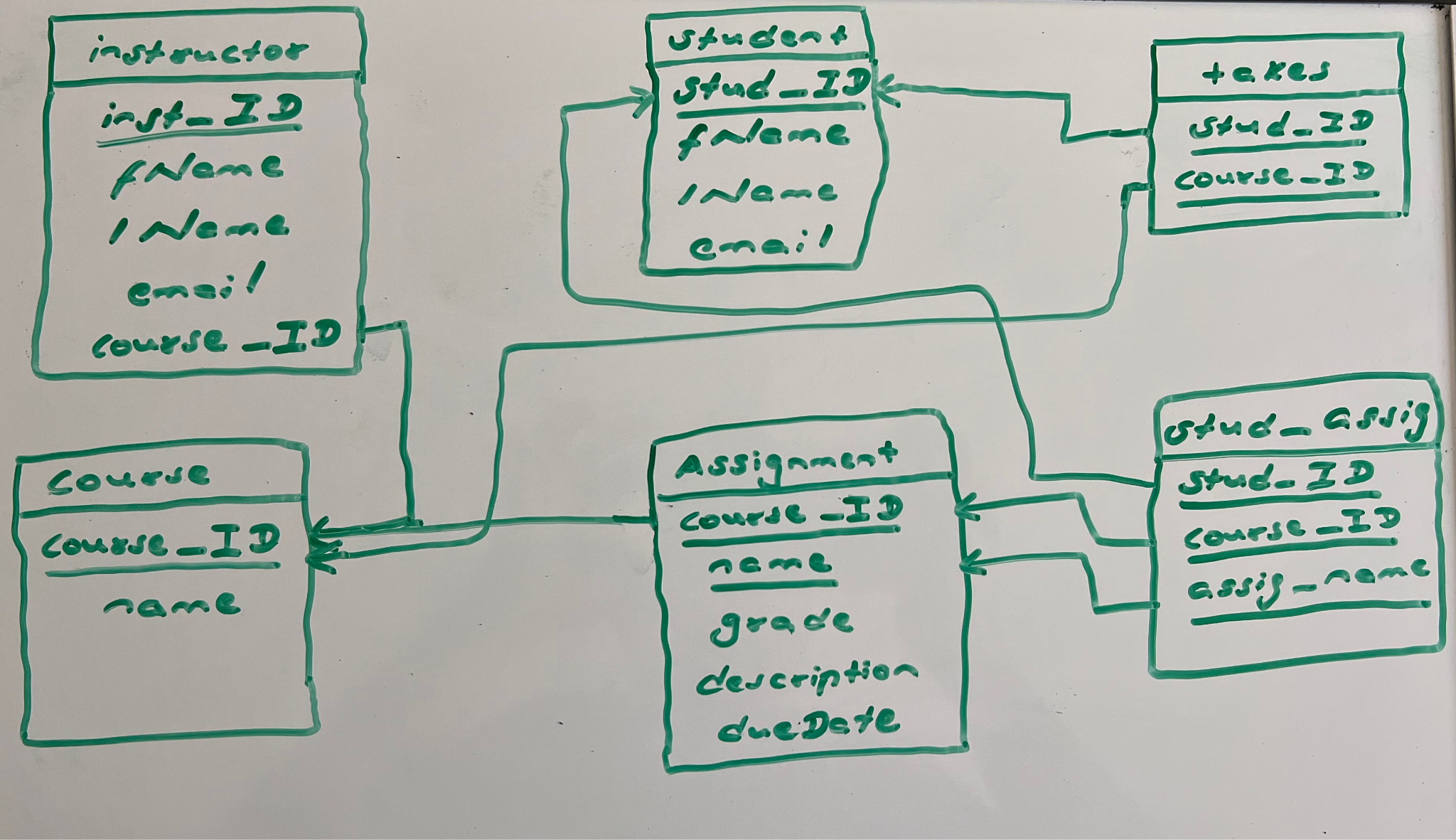
instructor(instID, fName, lName, email, courseID)

student( studID, fName, lName, email)

assignment( courseID, name, dueDate, description)

takes( studID, courseID)

stud\_assign( studID, courseID, name, grade)

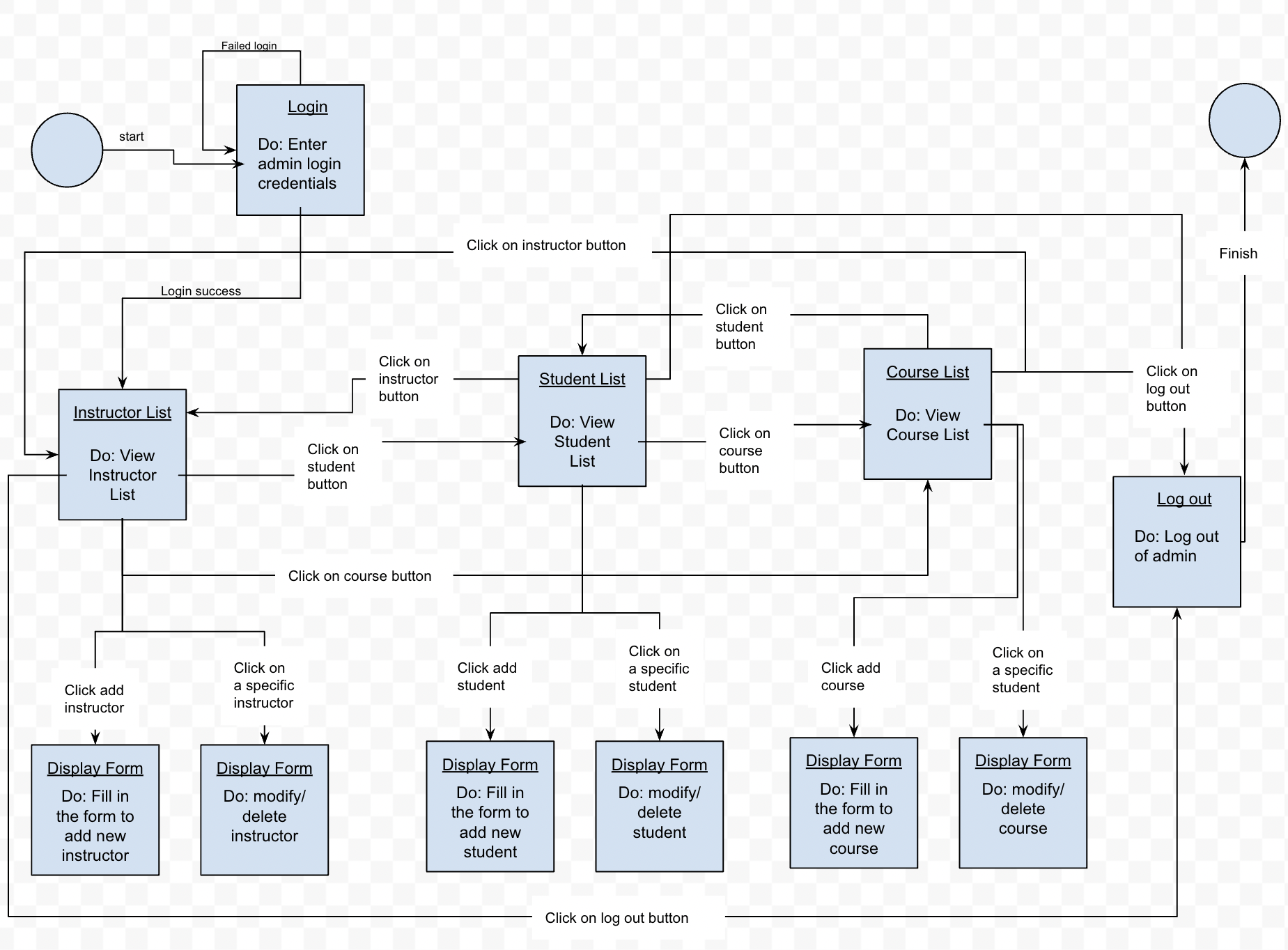


Our database stores all information about instructors, including courses, ID, and personal information. It stores student’s information including the course with associated instructor and email. Assignments are stored with its respective information, in addition to grades per student per class.

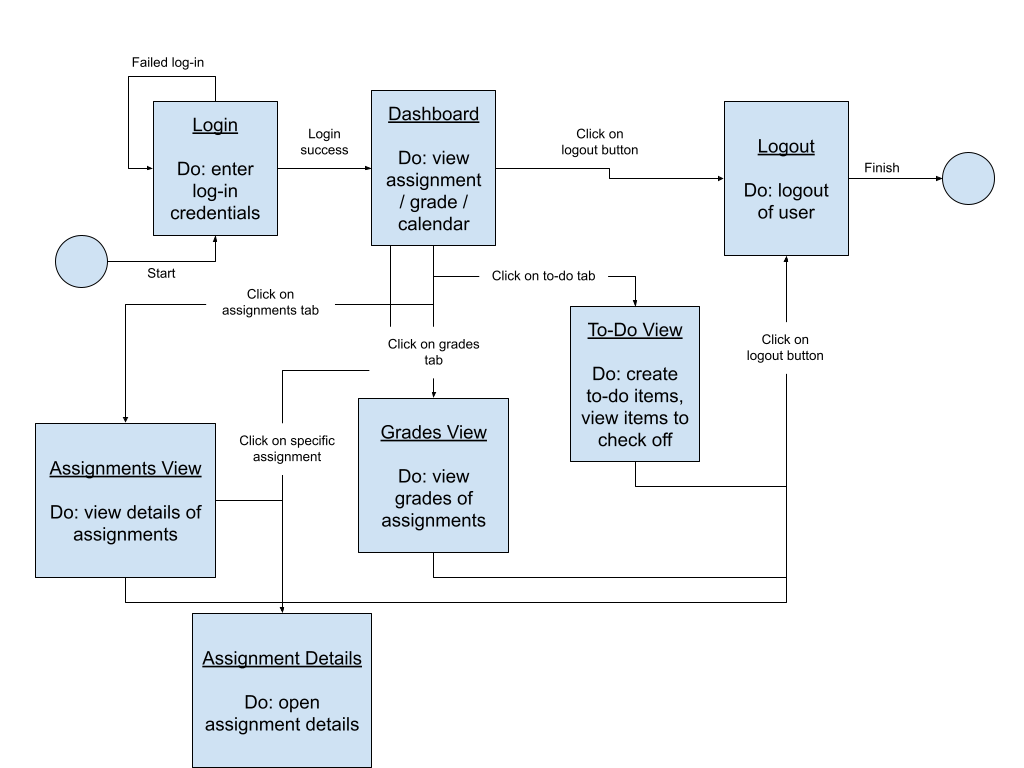
# Software Design

## State Diagram

### Admin State Diagram



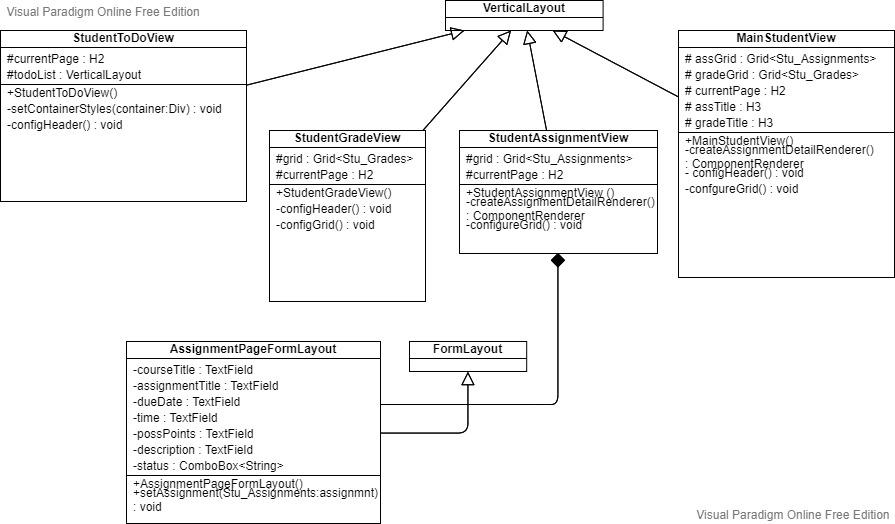
### Student State Diagram

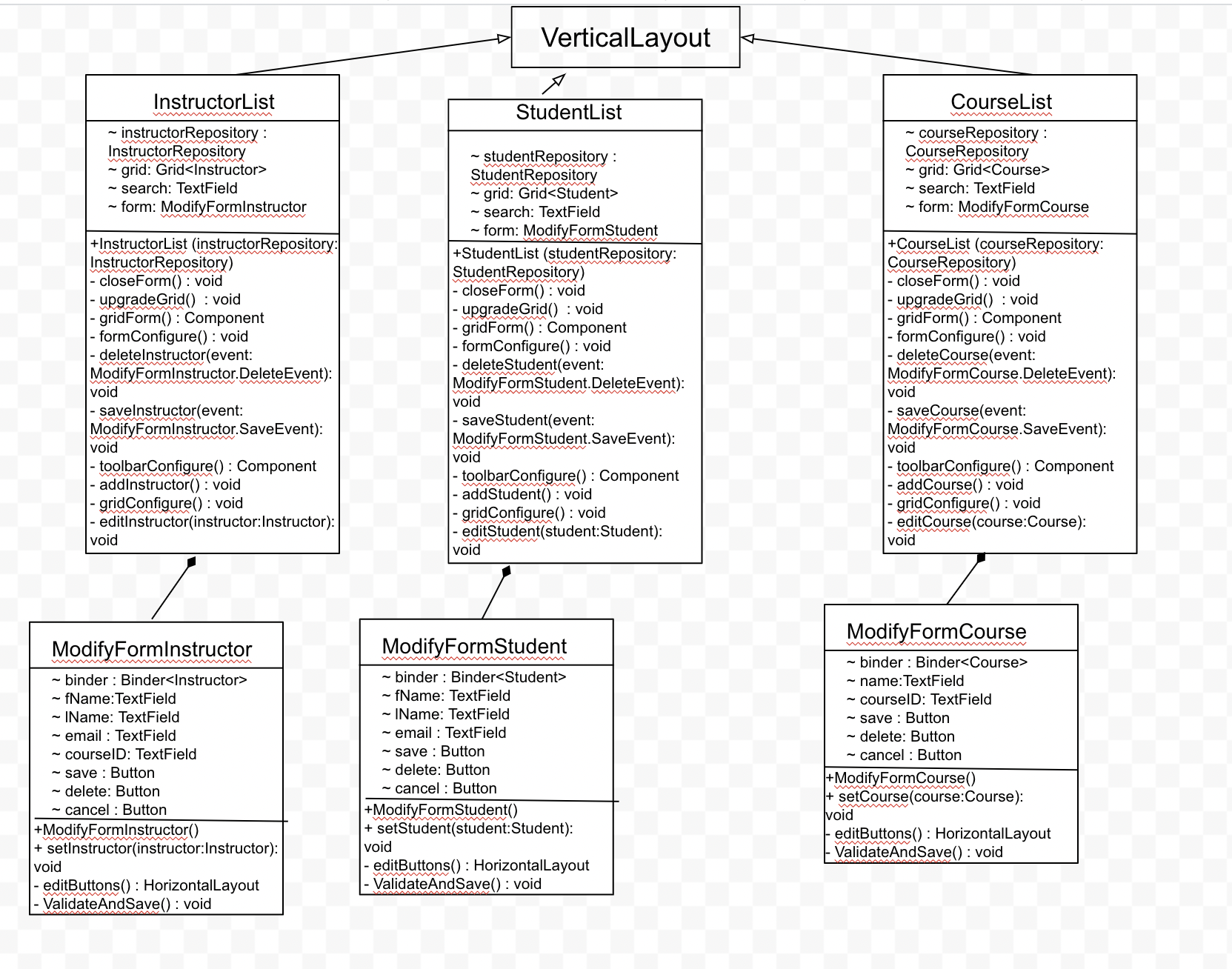


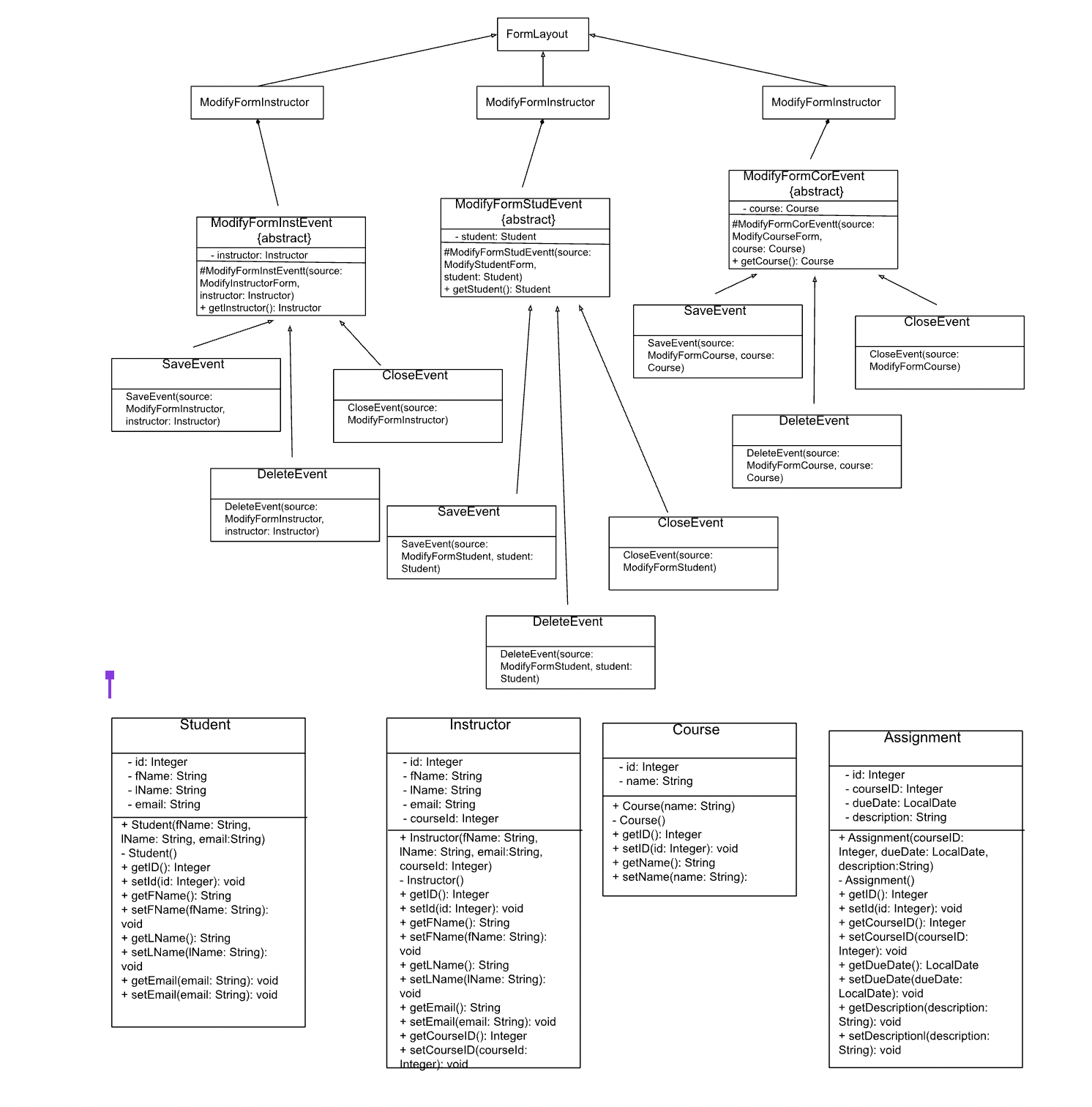
### Teacher State Diagram

## UML diagram

## 

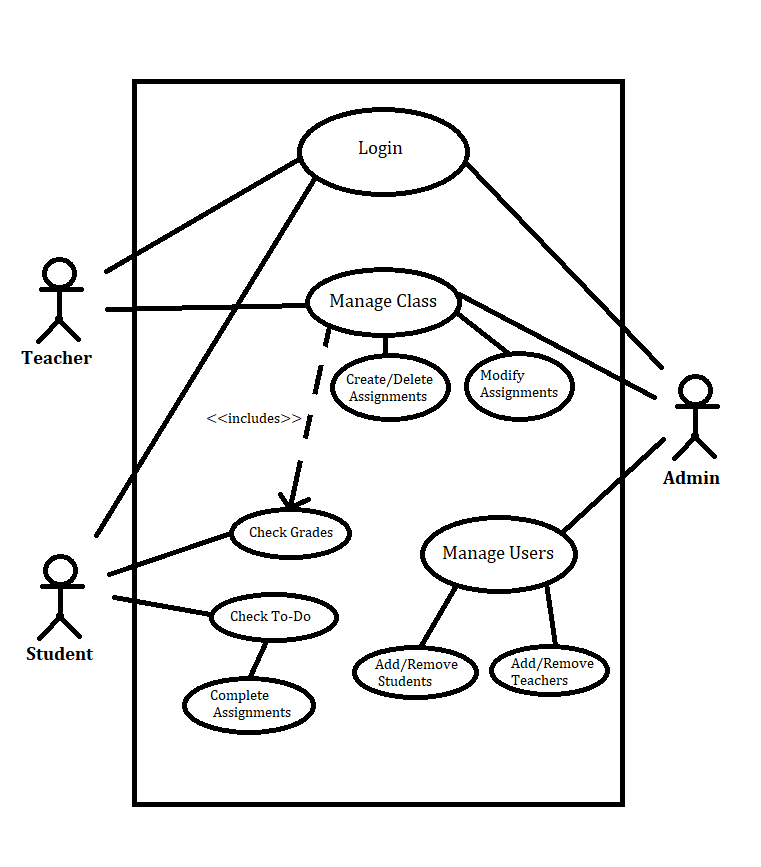






# Appendix

## Use-Case Model



**Section I: Brief descriptions**

* **Login (Andrew)**

The students, teachers, and admins should be able to login to the system with credentials provided by the admins.

* **Manage Class (Andrew)**

The Teacher shall be able to view students within their class and add, delete, or modify any assignments to the student’s to-do list. They can also view and modify student’s grades. The admin should be able to view the classes of any teacher to help

* **Create/Delete Assignments (Andrew)**

The teacher can assign any task to the students to do list for them to complete by the assigned date.

* **Modify Assignments (Andrew)**

The teacher can revisit any assigned assignments and change anything as they need to.

* **Check Grades (Evan)**

The student should be able to view any grades they have received for an assignment and the teacher should be able to view the grades their students have within their class.

* **Check To Do (Evan)**

The student shall be able to view their to do list once logged in and be able to click and view their assigned assignments.

* **Complete Assignments (Evan)**

The student should be able to complete any assignments that may be assigned to them through the class before the due date.

* **Manage Users (Nishant)**

The admin should be able to login and view all users that they have added to the system. This includes the ability to add and remove any users from the system.

* **Add/Remove Students (Nishant)**

The admin should be able to create logins for the students which includes which classes the students are assigned to.

* **Add/Remove Teachers (Nishant)**

The admin should be able to create logins for teachers to use the site which includes being assigned as teachers to their classes.

**Section II: Scenarios**

* **Login**
  + Initial Assumption: The teachers and students can login with a provided account. Their accounts are saved into a database
  + Normal: the teachers and students will enter their username and password to log into their assigned accounts.
  + What can go wrong: the teachers and students wrongly enter their credentials, which means they should be able to re-enter a limited number of attempts until they are locked out and will have to ask the administrator to reset their account.
  + Other Activities: Request support from an Administrator.
  + System state on completion: the students and teachers are logged in. students can view their To-Do list. Teachers can manage their classes through their dashboard.
* **Manage Class**
  + Initial Assumption: The teachers should be able to login with their provided account and can view the Manage class dashboard.
  + Normal: the teacher will push out any assignments, check grades, and make modifications.
  + What can go wrong: the teachers can assign the wrong assignment with the wrong date or give out inaccurate grades.
  + Other Activities: Assignments can be modified if errors are made.
  + System state on completion: The teacher is able to view their students and give them assignments for their to-do lists. The teacher can also modify the students' grades.
* **Create/Delete Assignments**
  + Initial Assumption: The teachers should be able to login with their provided account and can view the Manage class dashboard.
  + Normal: the teacher will be able to create and remove any assignments to any of their own classes
  + What can go wrong: the teachers can wrongly assign an assignment with the wrong date or the assignment could have inaccurate questions.
  + Other Activities: Assignments can be modified if errors were made.
  + System state on completion: The teacher sends out an assignment to all the students in her class. The students should be able to see this assignment on their To-Do list.
* **Modify Assignments**
  + Initial Assumption: The teachers should be able to login with their provided account and can view the Manage class dashboard.
  + Normal: the teacher will be able to revisit any assigned assignments and modify any mistakes that were made when initially published or to add anything relevant to a certain assignment.
  + What can go wrong: the teachers can forget to commit any changes after the modification was made to the assignment.
  + Other Activities: Assignments can be modified again if any more errors were made to the assignment or an admin could be contacted for assistance.
  + System state on completion: The modified assignment should be automatically updated for all students of the class, which the students should then be able to interact with.
* **Check Grades**
  + Initial Assumption: The teachers should be able to login with their provided credentials and can view their teacher dashboard. Students should be able to login with their provided credentials and be able to navigate their dashboard.
  + Normal: the teacher will be able to view the grades of any student within their class. Students should be able to view only their own grades within their class.
  + What can go wrong: the teachers can make a mistake when grading an assignment. The student could notice any mistakes with their grades.
  + Other Activities: The teacher can make modifications to the grades if necessary. Students can contact their teachers if any problems occur.
  + System state on completion: The teacher should be able to see the grades of everyone in her class, and students should be able to see their own grades.
* **Check To Do**
  + Initial Assumption: The student should be able to login using their own credentials and be able to understand and navigate the dashboard.
  + Normal: The student should be able to view any task that they have been assigned through the dashboard and to work on those assigned tasks.
  + What can go wrong: The student could accidentally complete the wrong assignment or fail to turn it in on time.
  + Other Activities: Students can add their own tasks to the To-Do list
  + System state on completion: The student is logged in. They can see their To-Do list which contains all tasks/assignments they need to complete.
* **Complete Assignments**
  + Initial Assumption: The student should be able to login using their own credentials and be able to understand and navigate the dashboard.
  + Normal: The student should be able to view any assignments and be able to submit anything as necessary.
  + What can go wrong: The student could accidentally complete the wrong assignment or unintentionally submit the wrong document.
  + Other Activities: Students can save their progress within a certain assignment if needed.
  + System state on completion: The student is logged in. They can complete any assignments that they need to complete.
* **Check Users**
  + Initial Assumption: The admin should be able to login using their credentials and to manage and perform any moderation tasks.
  + Normal: The admin should be able to view and modify all the users that are registered within the database.
  + What can go wrong: The admin could accidentally modify the wrong user or give someone the wrong role.
  + Other Activities: The admin can modify, add, or remove any users from the database.
  + System state on completion: The admin is logged in. They can complete any administrative task that they need to complete from the dashboard.
* **Add/Remove Students**
  + Initial Assumption: The admin should be able to login using their credentials and to manage and perform any moderation tasks.
  + Normal: The admin should be able to add a user as a student to the database or delete a student from the database.
  + What can go wrong: The admin could accidentally remove the wrong user from the database.
  + Other Activities: The admin could recover any removed users for a limited time.
  + System state on completion: The admin is logged in. The admin can manage any student users as needed.
* **Add/Remove Teachers**
  + Initial Assumption: The admin should be able to login using their credentials and manage or perform any moderation tasks.
  + Normal: The admin should be able to add a user as a teacher to the database or delete a teacher from the database.
  + What can go wrong: The admin could accidentally remove the wrong user from the database.
  + Other Activities: The admin could recover any removed users for a limited time.
  + System state on completion: The admin is logged in. The admin can manage any teacher users as the need arises.