Software Design Specifications

CodeClassy

Version: 0.0.1

Project Code	*F210701*
Supervisor	*Dr. Muhammad Rafi.*
Co Supervisor	NA
Project Team	*Saif Ul Islam - 18K 0307* *Muhammad Hassan Zahid - 18K 0208* *Tashik Moin Sheikh - 18K 0142*
Submission Date	*7th December, 2021*

Document History

Version	Name of Person	Date	Description of change
0.0.1	Saif Ul Islam	7th Dec, ‘21	Document Initialized

Distribution List

Name	Role
Dr. Muhammad Rafi	Supervisor
NA	Co Supervisor

Document Sign-Off

Version	Sign-off Authority	Project Role	Signature	Sign-off Date
	Muhammad Rafi	Supervisor	RΛFi	Dec 07, 2021

Document Information

Category	Information
Customer	FAST-NU
Project	CodeClassy
Document	Software Design Specification
Document Version	0.0.1
Status	Draft

Author(s)	Saif Ul Islam, Muhammad Hassan Zahid, Tashik Moin Sheikh
Approver(s)	Dr. Muhammad Rafi
Issue Date	7th December, 2021
Document Location	Online, Google Docs
Distribution	Advisor Project Coordinator's Office (through Advisor)

Definition of Terms, Acronyms and Abbreviations

Term	Description
ASP	Active Server Pages
DD	Design Specification

1 Introduction

1.1 Purpose of Document

The purpose of this document is to provide a comprehensive overview of the design decisions made within the software project, and showcase the depth in details in terms of data, architecture, interface, procedure related design. It is developed to provide a relationship between the functional requirements and the structure, implementation, data association and modeling, and the overall architecture of this software application. It aims to do this by providing different viewpoints of the system as per the representation in need in discussion to an audience.

This document is written keeping in mind the target audience of primarily the FYP team, the jury, faculty at FAST. By following technical and industrial standards, we wish to make this document available to any and all software developers that would be interested in learning more about the software project. Thus, it is our aim to write this document in such a presentational format that gives way to,

1. *Clarity over the depths of the technical details of the project (intrigue)*
2. *Gives rise to questions that would allow us to refine this document (retain feedback)*

It is complex to answer the exact questions regarding our "design methodology" - while our approach so far with the project requirements, design decisions and requirement gathering has been based around Agile, our work has been split into concrete modules because of the 30%-70% requirement of FYP I, FYP II respectively, which seem to align with a Spiral Design Model instead.

How are we doing Agile? A good contrast is by aligning with the Agile Manifesto,

1. ***Individuals and interactions** over processes and tools - our team communication is intensive, almost everyday, with discussions about approaches/problems/flaws/fixes, next steps and so on. We have not rendered ourselves rigid to follow a very "strict" process or on the basis of a tool, instead letting tools help us achieve our goals*
2. ***Working software** over comprehensive documentation - Our focus has been on development and incremental delivery with iterations on modules. We wish to have a working, in-order, demonstrable software over documentation that just says what we're going to do (except this documentation, of-course).*
3. ***Customer collaboration** over contract negotiation - we take in and value additions, idea improvements, and discussions with our supervisor, instead of noting down strict "contracts" that do not detail what is left to be desired.*
4. ***Responding To Change** over following a plan - we make adjustments in the requirements as per request where it is needed*

1.2 Intended Audience

As mentioned above, the target audience of interest are,

1. *The FYP team itself*
2. *The Jury*
3. *Our Supervisor*
4. *Any and all potential faculty at FAST (once submitted, it will be visible to all)*
5. *Software Developers and Solution Architects (review and analysis)*

1.3 Document Convention

*The font is "**Arial**" with font size "**10**".*

1.4 Project Overview

This software project aims to integrate many virtual educational needs into one place. As compared to current platforms which provide a niche solution to a niche problem in evaluating software needs and its development, CodeClassy attempts to bring together multiple features together to reduce complexity levels, high intense communication needs, and remove redundancy in information sharing, alerts, events, updates, and so on.

On the high level, the project attempts to (for now) bring together 3 main components together as integration pieces, which are,

1. *Classroom Management System*
2. *Quiz Management System*
3. *Coding Assignments (Remote Code Execution + Real Time Collaboration)*

These are the main high level components of the system that define the functional requirements for the requirement process. All together, these components encapsulate entities logic for Teachers, Students, Sections, Classrooms (and more).

As stated before in "Purpose Of Document", we would like to use the Agile approach for building the software

1.5 Scope

Documentation is not in scope of the project. We do not intend to write code or functionalities that serve as "User Documentation" - that is, helping users navigate the application. Since there is the idea of an MVP, we wish to develop more and more features and functionalities to fulfill our committed requirements, rather than spending too much time on writing "comprehensive" documentation which is not the aim of an FYP anyways.

The system will not integrate multiple login options, only working with a main email/password schema.

2 Design Considerations

The system should be in all cases reliable, robust, adaptive, and flexible for development and use purposes. For this intent, we spent a considerable amount of time trying to work towards a better design flow and structure for the project, in order to solve major problems that existed with the design of the system on a finer-grained level.

The issues related to design were,

1. *What is the architecture we would like to develop our front-end and back-end on? What frameworks/libraries we believe align with the workflow we have and how would we like to break the project down into smaller components*
2. *How would we go about solving the remote code execution problem and what tools would we use to enable real-time collaboration? It is a niche and difficult problem to solve - what approaches could we use to counter these problems?*
3. *What would we use to build rapid UI, and how would we manage user states and sessions on the frontend?*
4. *How would we deal with authentication?*
5. *How would we talk to the database?*

A few other affecting considerations can be about,

1. ***Compatibility** - since it is a web application, we care about compatibility on different browsers enough so that it does not break for our users*
2. ***Extensibility** - the project is designed as such for ease* in *adding of new modules with capabilities or further functionalities without having to touch or modify contents of other modules*
3. ***Modularity** - modules are designed to be well-defined, independent components which would lead to better maintainability. Dependencies are clearly defined where needed and are well-intentioned and well-contained.*
4. ***Fault-tolerance** - the application is not yet planned to be beyond local development environments, so this consideration does not really apply*
5. ***Maintainability** - the codebase is broken down into modules, so bugs if any are module-contained, easy to trace and detect.*
6. ***Reliability** - because we are working on the local environment, it is easy to check for reliability for the software to do as it is expected within stated conditions for a specified period of time.*
7. ***Reusability** - we are using already developed software and packages where we believe we can use the functionality that they provide us, reducing our time to develop from scratch.*
8. ***Robustness** - that the software operates under stress of users and inputs or tolerate unpredictable or invalid input*
9. ***Security** - the software is able to withstand and resist hostile acts and influences*
10. ***Usability** - the interface is designed to be simple and easy to use without the need to involve user documentation manuals*
11. ***Performance** - the software will be usable across a number of platforms of mobile, desktop, and laptops*
12. ***Portability** - since it is a web application, the software is perfectly portable*
13. ***Scalability** - the software adapts well to increasing data or added features or number of users*

In such a case, our starting points were to,

1. *Finalize and decide on the main requirements of the project that we were going to work on through requirements analysis and engineering*

2. *Determine the scope of both the functionalities we were going to develop and the implementation details of how we were going to accomplish them - define scope limitations and scope creep*
3. *Define what tasks need to be done till when and how, and perform task allocation*
4. *Revaluate work as needed*

This approach will allow us to address design solutions according to our workflow and Agile Design, that would help in developing a code base that is maintainable, adaptable, modular, and extensible for further development.

2.1 Assumptions and Dependencies

As per concern with the team's idea of system and software design, we imagine design to be concerned with data modeling, entities, abstractions, architectural patterns, and strategies (design models being a strategy to achieve those goals). Our sense of assumptions encapsulates notions about the project's timeline, deliverables, and the workflow that we assume will be in place, in order to help the project run but can't be guaranteed.

It is a want to have those assumptions to never be proven wrong, otherwise there will be a direct impact on the project.

*And so, what were our **assumptions** starting FYP I? They were as the following,*

1. *We would be able to finish all the of the requirement engineering and analysis before Mid I (FYP defense) in order to not stumble into unforeseen territory and have a team consensus on what must be done and what should be left for now, as per a team timeline*
2. *We would be able to finish our **Classroom Management Module** completely, all of the backend/front-end as well as the integration with the user flow before Mid 2*
3. *We would be able to finish all of the Quiz Management System prior to the finals completely, all of the back-end/front-end, as well as the integration with the user flow before the finals.*
4. *We will not have any changes and will be fixated on a set of requirements at the start of the semester*
5. *We would be able to manage personal time to handle this large project as per our expectations - we realize realistically and responsibly that this project is of large scope and it's importance is gigantic, and that missing deadlines would keep spiraling into other deadlines*
6. *We assumed course workload as well as personal workload in our personal lives would be very light due to this being our final year - we were very wrong about that*

For each of the above assumptions, many were altered,

1. *A change occurred after the FYP defense where we were asked to add classrooms separately from sections, with the concept of teacher coordinators and students as assistants*
2. *While most of our work related to the **Classroom Management Module** was completed before Mid 2 in terms of back-end/front-end, we couldn't work on integration and have been working on it since*
3. *We are currently in the process of discussing our strategy for gathering all requirements for the Quiz Management System so we can start work on it in terms of design, implementation, interface, data modeling, and such. As such, our timeline has "extended".*
4. *A bit of uncertainty with data modeling led us to exploring Remote Code Execution, Real Time Collaboration solutions, and we were generally unwilling to start implementation until our defense was approved. We were scared of "what if this does not get approved?", or "what if we have to make changes at the core of requirements?". While these fears were not of fruition, it led the*

team to realize document related deadlines/timelines and how much they can affect a project's progress

5. *We were unable to sync ourselves after Mid I properly, leading to each member working individually and unable to contribute as effectively as possible. We might have miscalculated the time and effort the physical presence in University along with responsibilities at home along with personal time for mental health, sleep, time allocation for other work and assignments will take. Had we been working full time on this project, there surely would have been more progress*
6. *As said before, we were wrong about that, and life being unexpected, has turned our initial expectations of ourselves over the project into frustration at some points of the project's development*

For dependencies, we were blocked by a few factors we were waiting on/for to pass,

1. *FYP Defense - there was no reason why we couldn't start work prior to the defense, and if we had the opportunity of getting our project approved prior to the fixed date, we might have taken that option*
2. *Data modeling had to be altered at different times to cater to requirements*

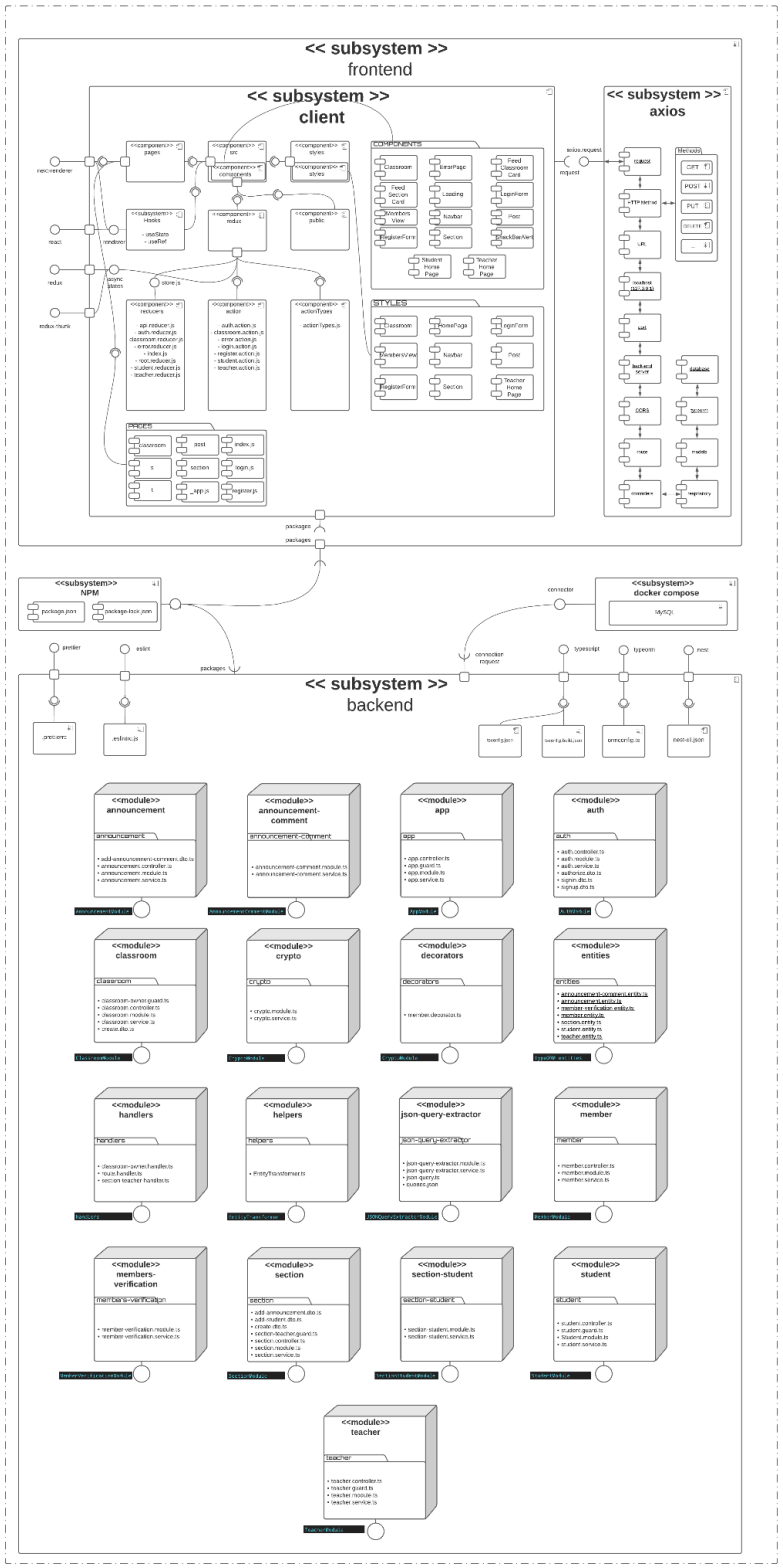
2.2 Risks and Volatile Areas

There is little to no time to consider, implement, and practice a contingency path. As such, we should not be facing any further changes in the requirements at the moment and our choice of technology is likely to remain consistent throughout the project - we have not faced any conditions where we wouldn't use our current set of tools and implement functionalities further with those.

It's also important to note that any further developments and requirements would further alter our timeline as altered it already is. If any developments do arise, our initial approach would be as always to understand if it's a priority, and if so, when does it become a priority to get done first.

System Architecture

2.3 System Level Architect



2.4 Software Architecture

[The software architecture should include how the User level Layer will interact with the Database layer. Use a diagram for showing the interaction between the layers.

- *User Interface Layer*
- *Middle Tier*
- *Data Access Layer.*
- *Or other*

You can give any other architecture also]

3 Design Strategy

The design strategies involve revolving around the single responsibility principle and that of modularity, where each module is designed to serve a unique purpose and has a unique identity within the system. Each piece of technology exists for a specific purpose, and that purpose only. It is to encapsulate the logic of a certain problem to be solved in a certain way.

Our architecture is mostly generic in terms of a client/server architecture, with the client providing server-side rendering for the interface, and the backend, being inspired by Angular, follows a similar approach of using modules collected into functional sets, applications being defined by modules, root modules enabling bootstrapping, feature modules, and so on.

What is our back-end? We're working with NestJS, a very strong framework on top of NodeJS meant to provide a robust architectural solution for backend development. NestJS provides a lot of benefits right out of the box. NestJS works by separating business logic from modules from service from tests, and the main entry file of the project

We wanted to enforce an architecture for great design, and enforcement just happened with NestJS.

Apart from this, the main overall organization of the system consists of a front-end, a back-end, a MySQL container being spun up thanks to Docker Compose. On a higher-level note, it is a client-server architecture implementation.

As for the key abstractions, we wanted to solve our problems with solutions such as,

1. ***Server Side Rendering** - the user gets the initial page immediately in response to their request*
2. ***Redux** - state management is often hectic and can become unmaintainable. For this reason, we decided to with Redux*
3. ***Modules** - we wanted to develop services as a per module setup, which is what NestJS provides*
4. ***Material UI** - we wanted to rapidly develop UI, so we chose the option of Material UI*

Mechanisms to achieve this come built-in with the libraries and frameworks we are using, with minimal need to develop a setup from scratch.

We wanted to take this approach/strategy for a couple of reasons,

1. *We believe that good design is essential for good software. We did not want to write an application using just any tool (like NodeJS or express) because we wanted to expose ourselves to more paradigms and take an approach that speaks good software engineering*
2. *We wanted the code base to be maintainable and easily workable on by group members thanks to the modular system*

For example, some areas for consideration for the system can be,

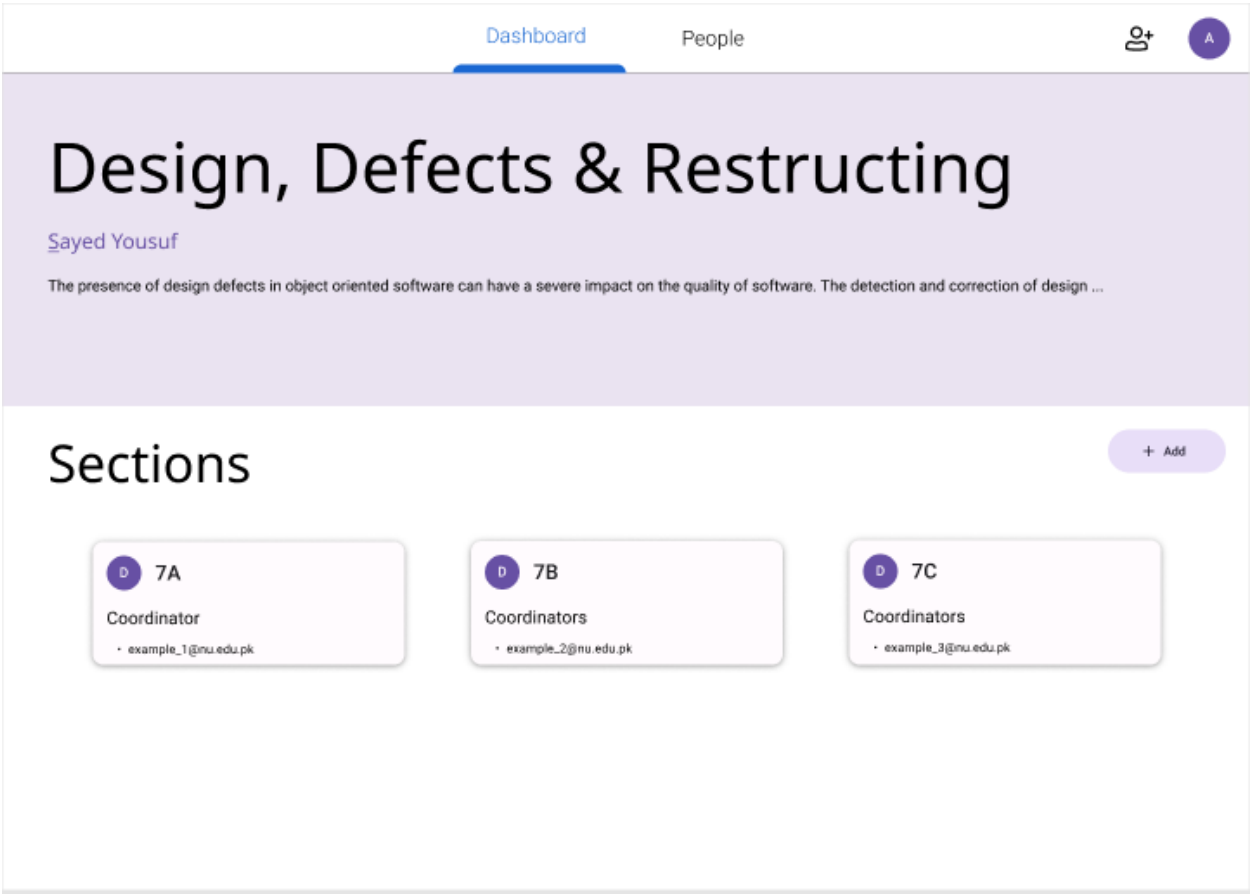
1. ***Future System Extension Or Enhancement** - the project is structured as such to be catered towards future development quite easily. The project is split, again, into modules, with pages and their respective services and calls that they need to operate*
2. ***System Reuse** - the paradigms on which the project is built promotes reusability via components on the front-end side, and through modular reuse on the back-end side*
3. ***User Interface Paradigms** - this project is simply a web application. Users will use the application via GUI (Graphical User Interface)*

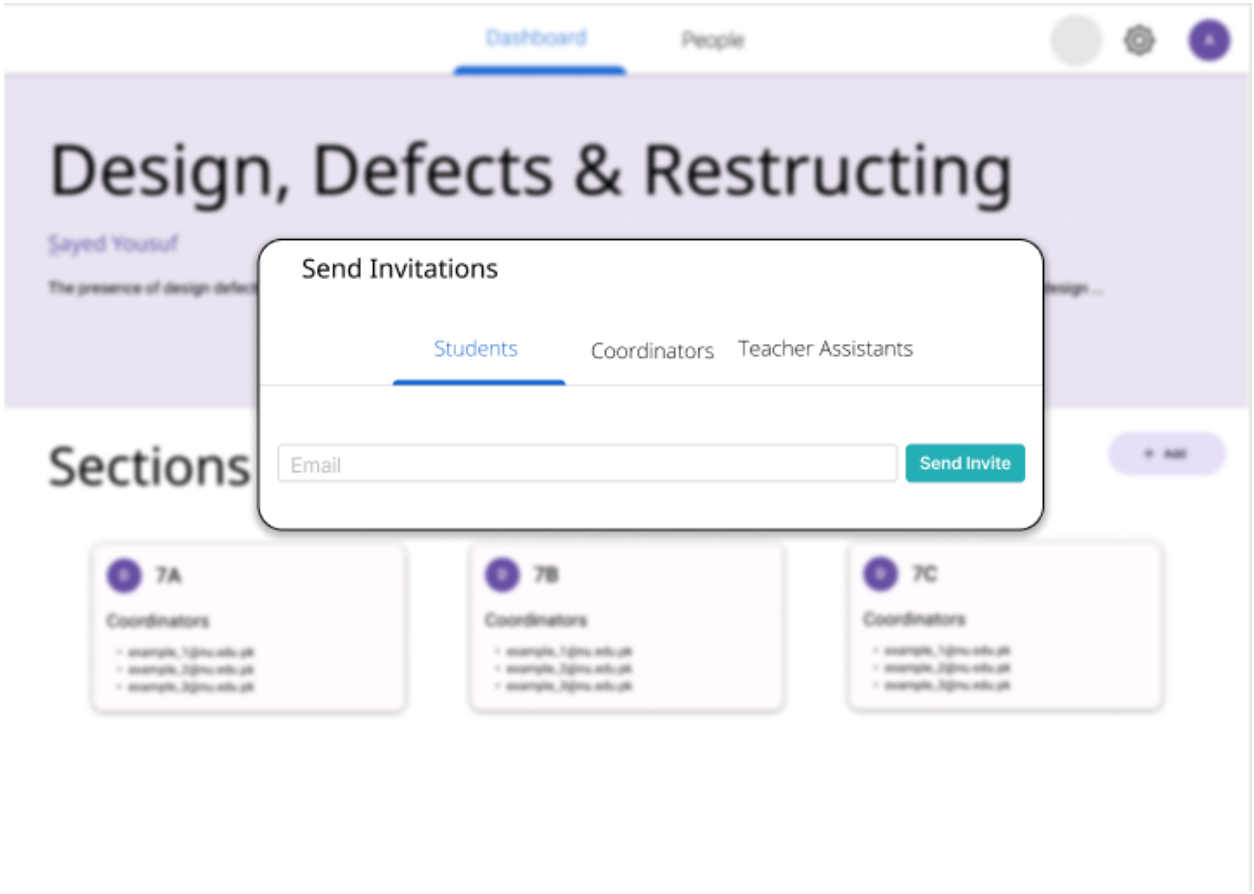
4. ***Data Management (Storage, Distribution, Persistence)*** *- the environment in use is a development environment, and hence the database server being used exists locally on the machine. Besides this as a storage, we do not have a distributed network for distribution, redundancy, and persistence*
5. ***Concurrency & Synchronization*** *- there is almost always a single user, which does not derive much focus towards proper concurrency and synchronization programming*

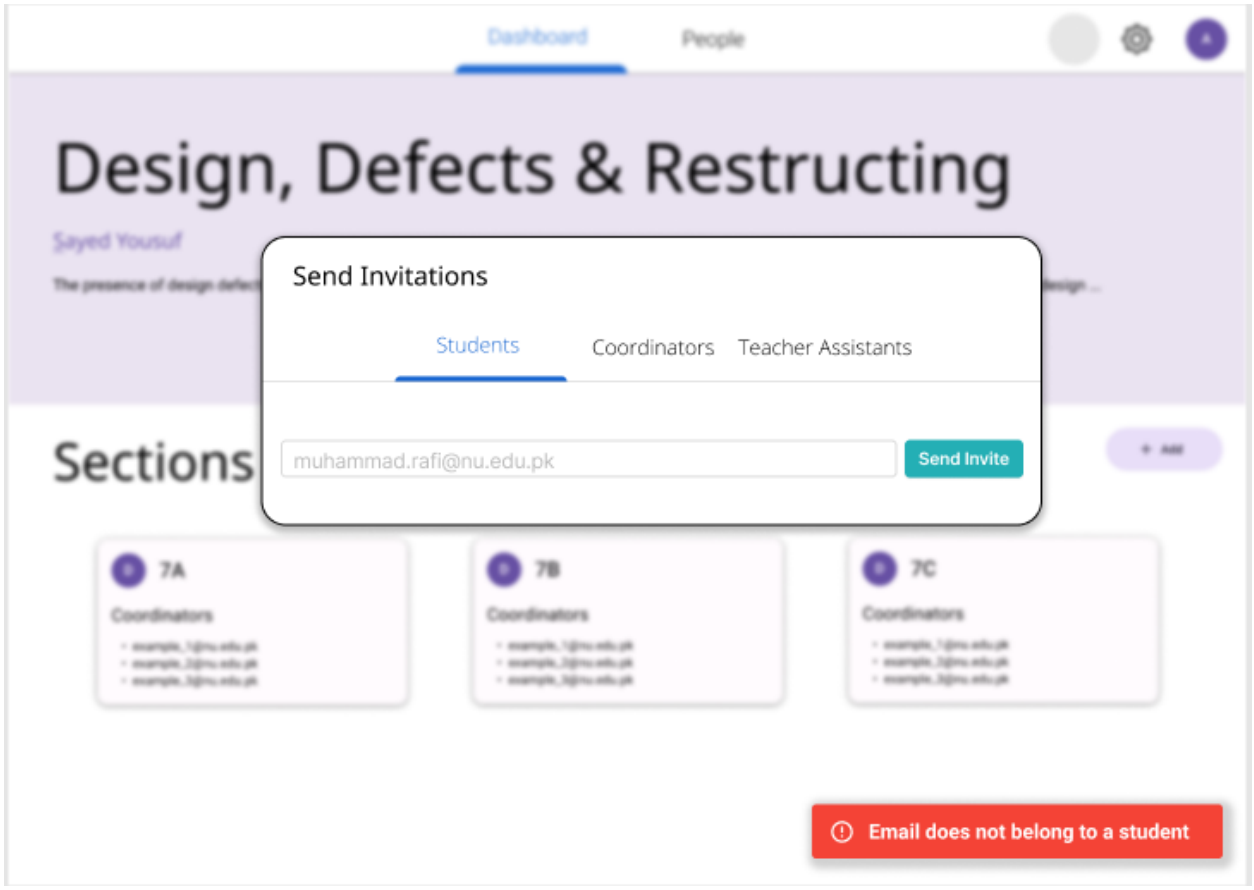
4 Detailed System Design

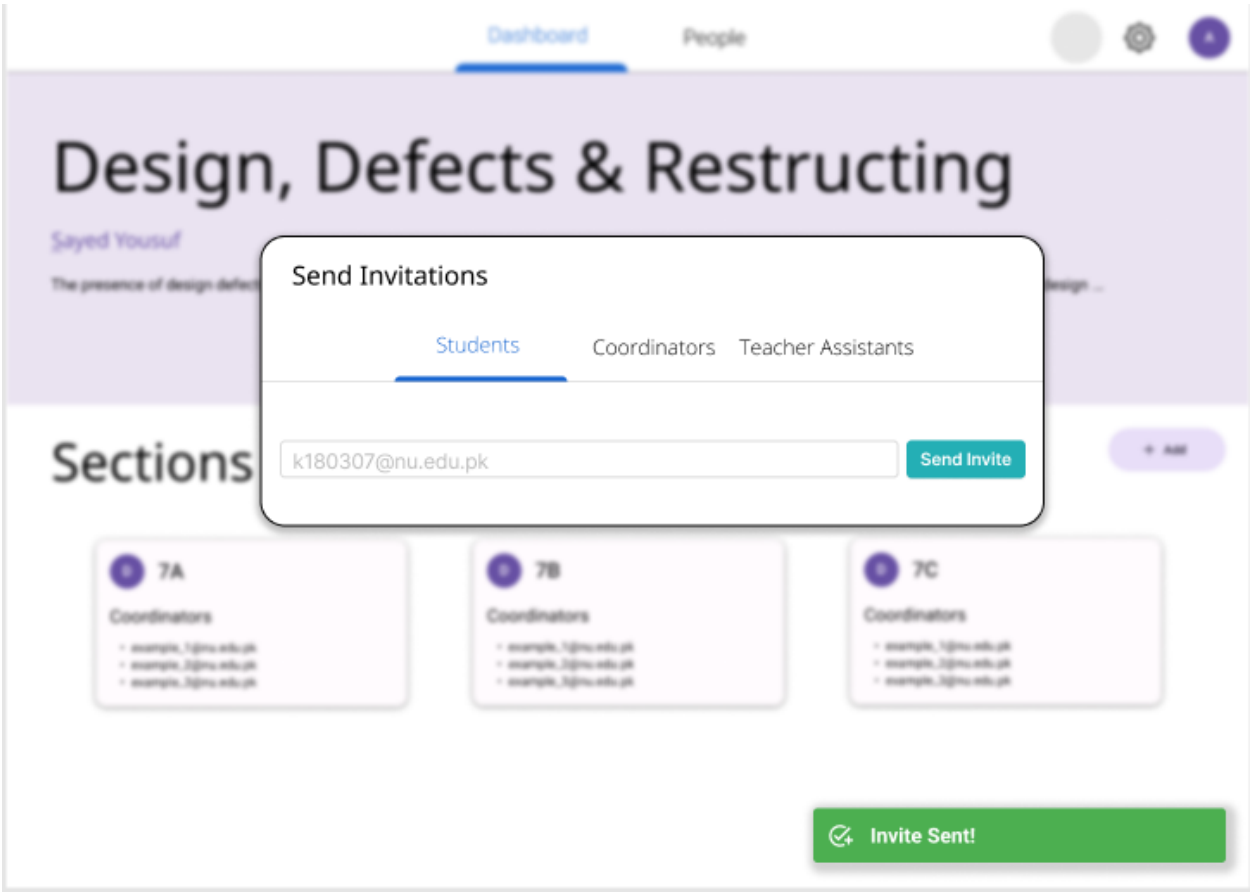
Detailed GUIs

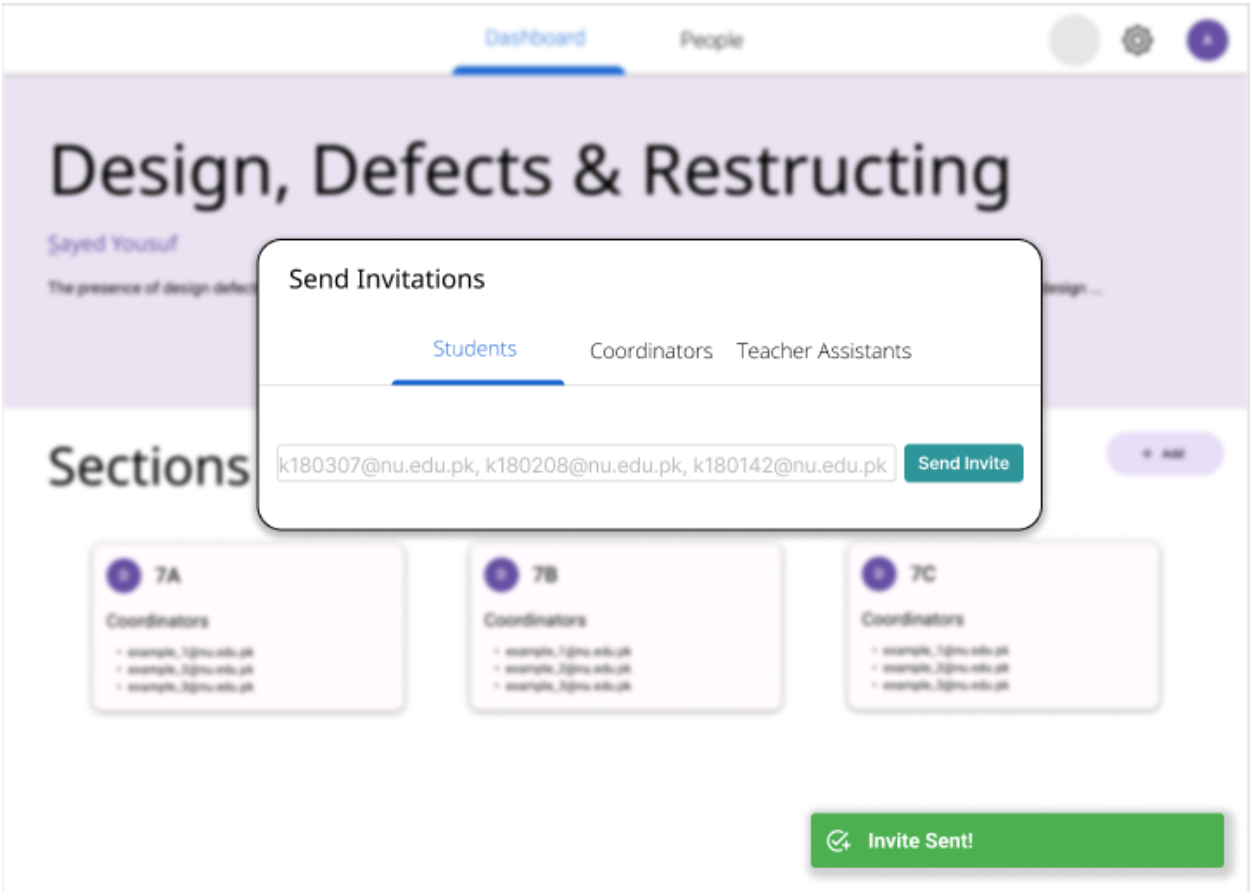
The GUI is VOLATILE. The below wireframes are only "visual perspectives", that is, how something may look like, instead of being exactly the same as the wireframes.

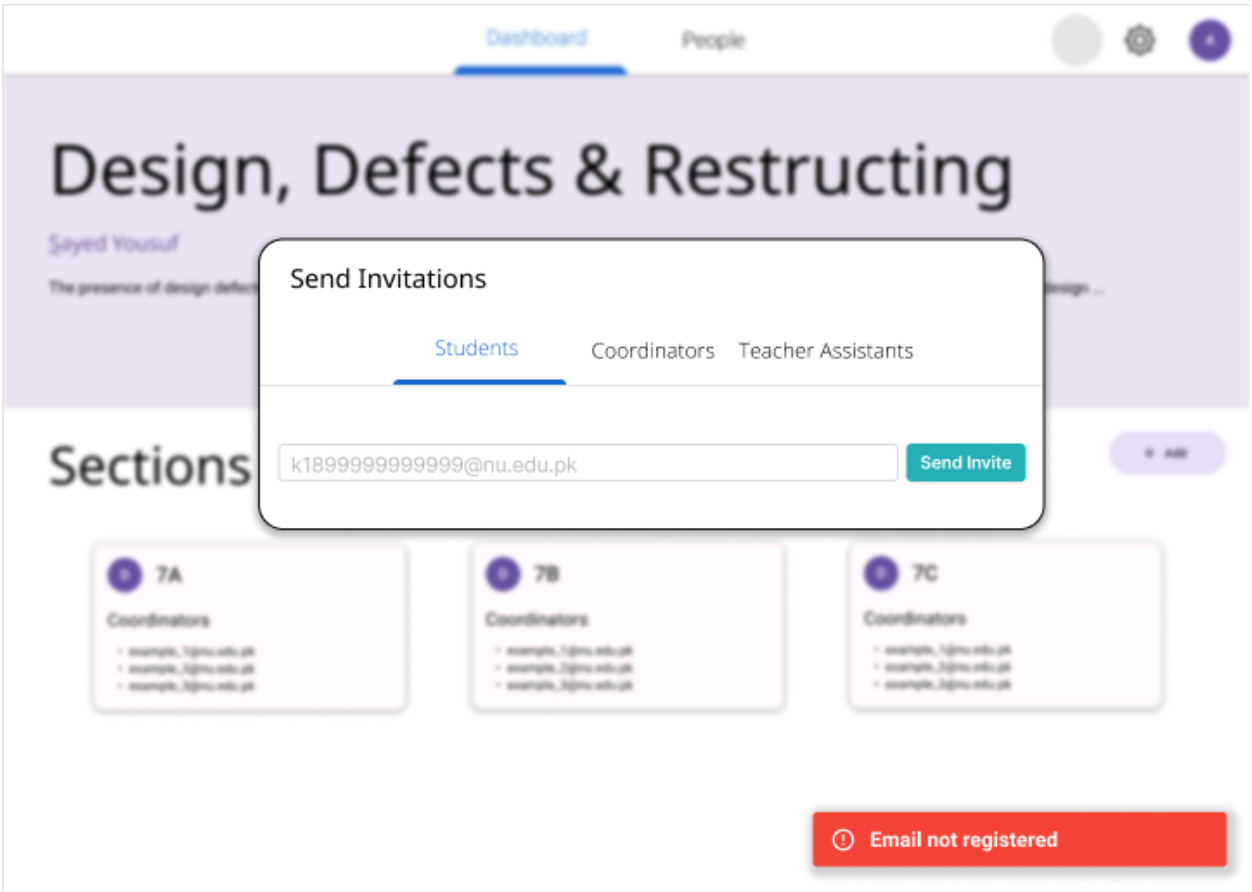


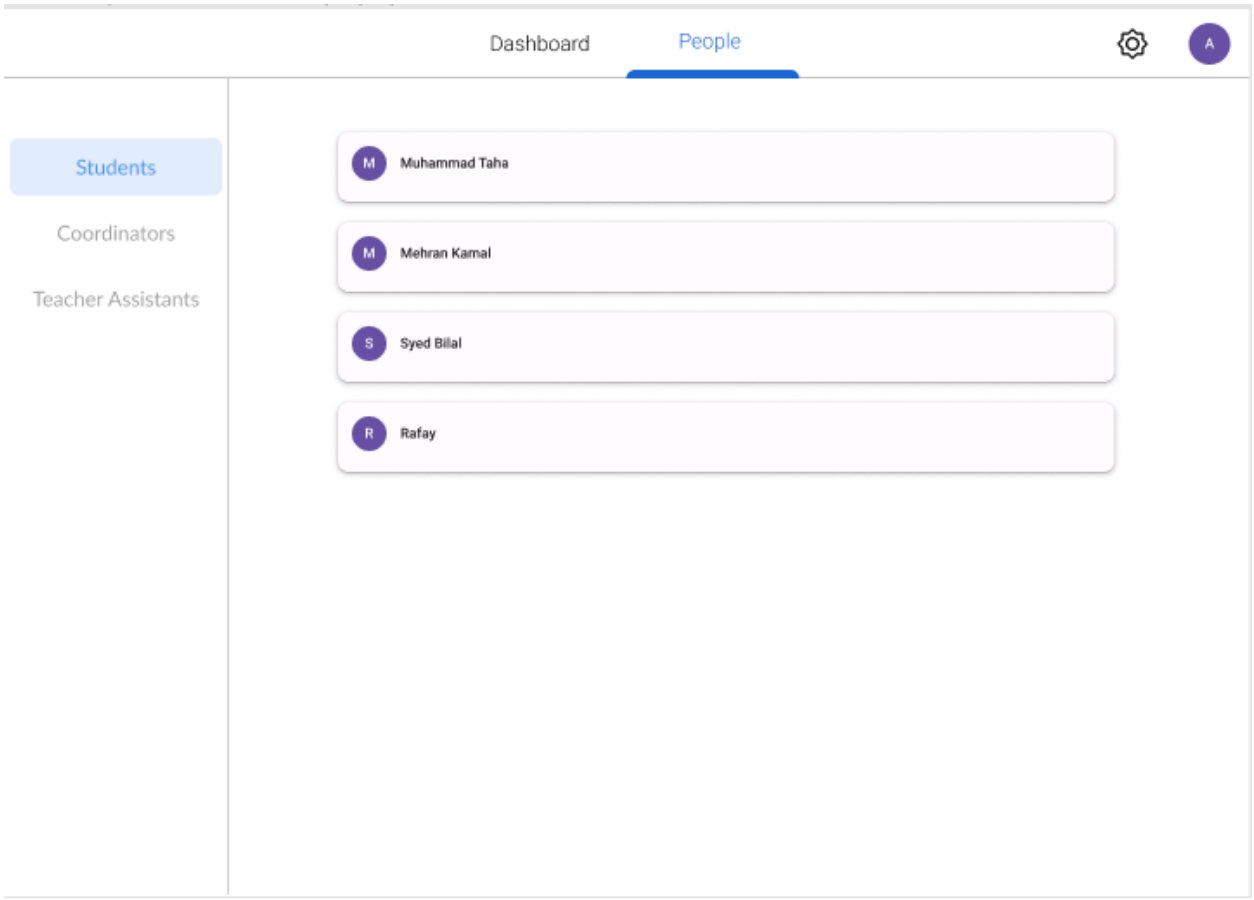

Dashboard
People
Design, Defects & Restructing
Send Invitations
Students
Coordinators
Teacher Assistants
Email
Send Invite
Sections
7A
7B
7C

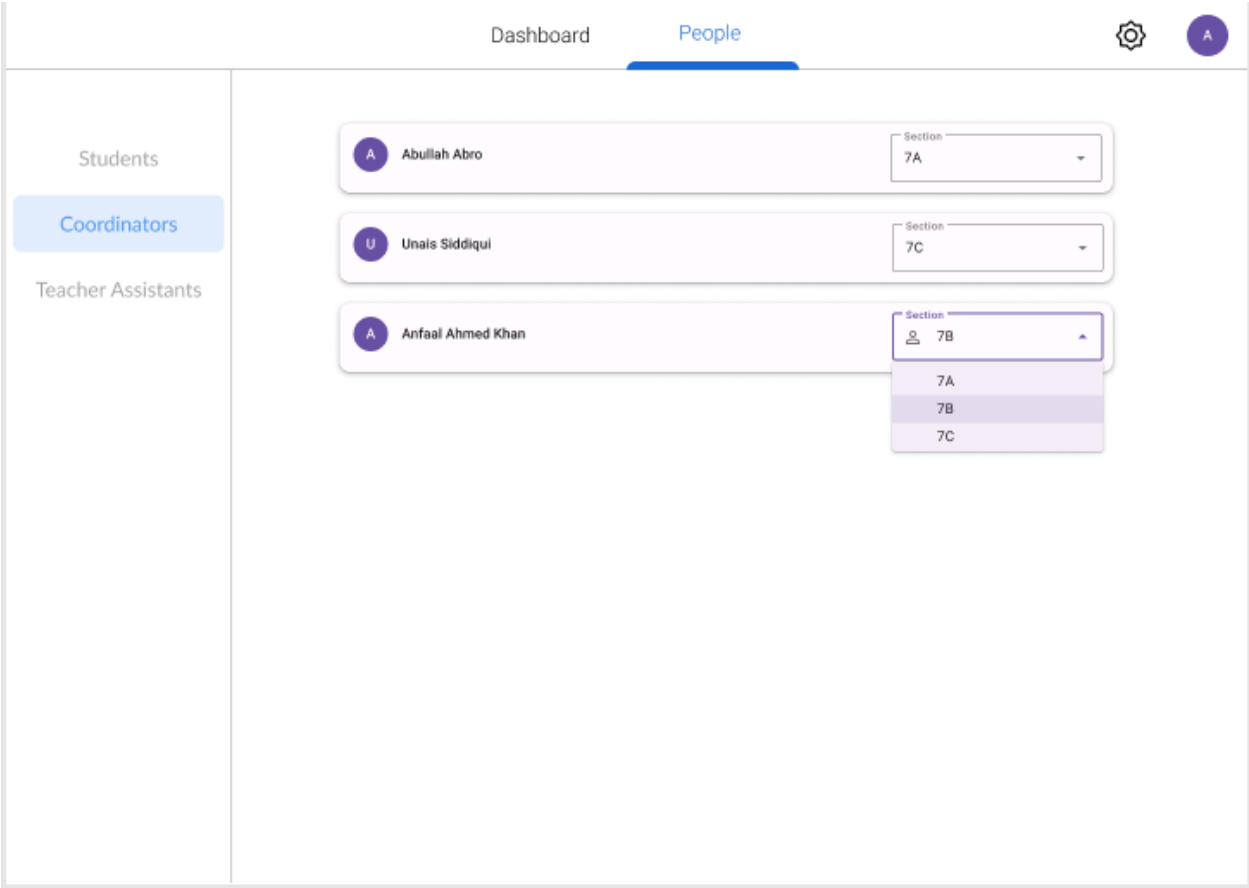

Dashboard
People
Design, Defects & Restructing
Sayed Yousuf
Send Invitations
Students
Coordinators
Teacher Assistants
muhammad.rafi@nu.edu.pk
Send Invite
Sections
7A
Coordinators
7B
Coordinators
7C
Coordinators
Email does not belong to a student

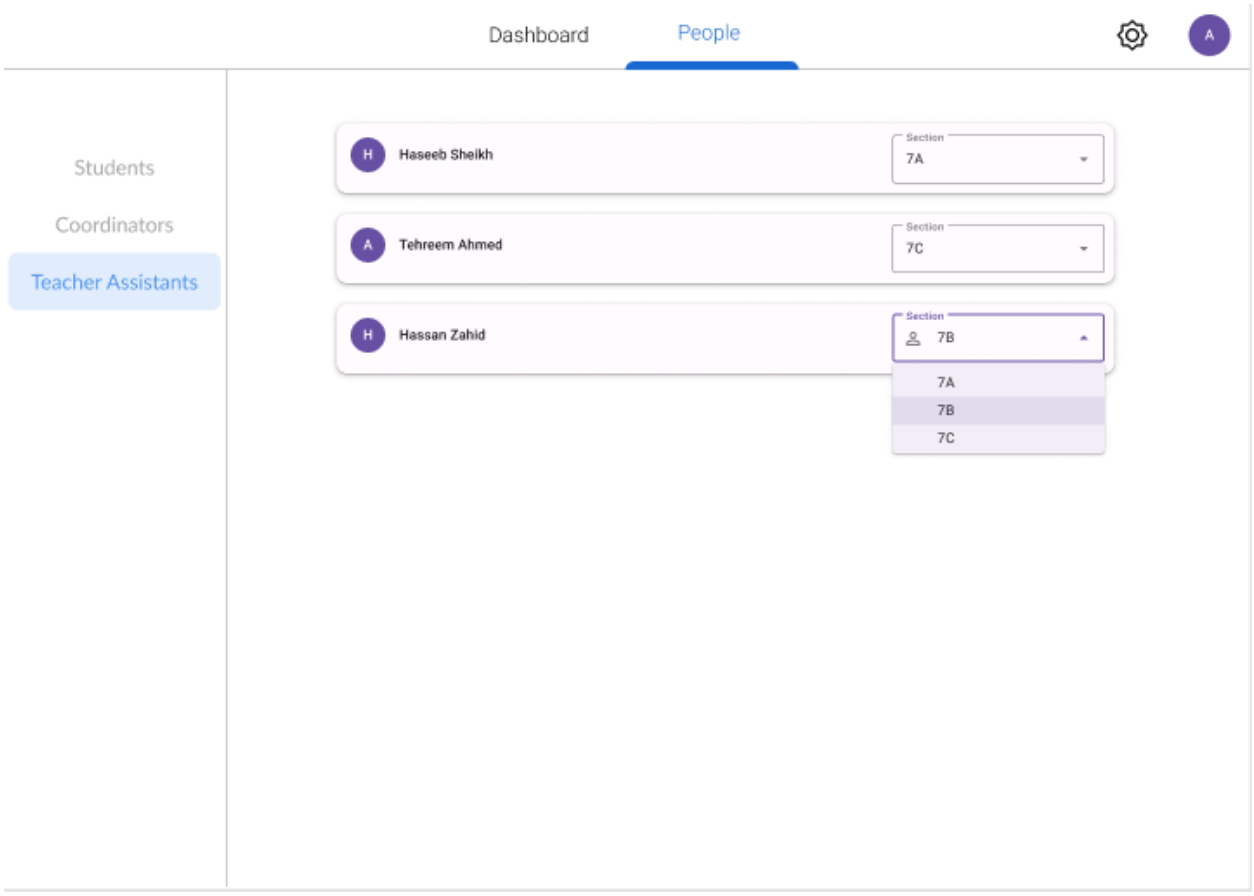

Dashboard
People
Design, Defects & Restructing
Send Invitations
Students
Coordinators
Teacher Assistants
k180307@nu.edu.pk
Send Invite
Sections
7A
7B
7C
Coordinators
Invite Sent!

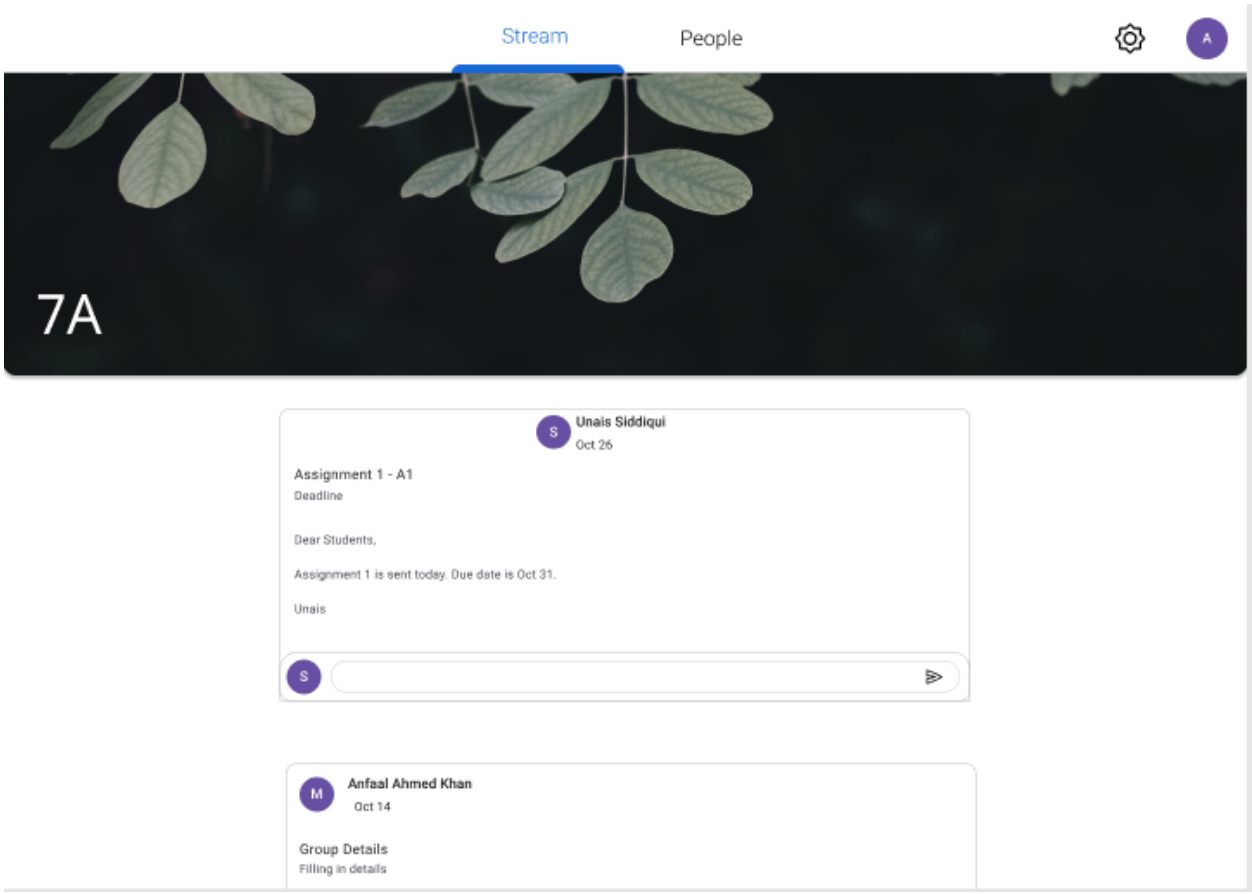

Dashboard
People
Design, Defects & Restructing
Sayed Yousuf
Send Invitations
Students
Coordinators
Teacher Assistants
k180307@nu.edu.pk, k180208@nu.edu.pk, k180142@nu.edu.pk
Send Invite
Sections
7A
Coordinators
7B
Coordinators
7C
Coordinators
Invite Sent!


Dashboard
People
Design, Defects & Restructing
Send Invitations
Students
Coordinators
Teacher Assistants
k1899999999999@nu.edu.pk
Send Invite
Sections
7A
7B
7C
Email not registered


Dashboard
People
A
Students
Coordinators
Teacher Assistants
M
Muhammad Taha
M
Mehran Kamal
S
Syed Bilal
R
Rafay

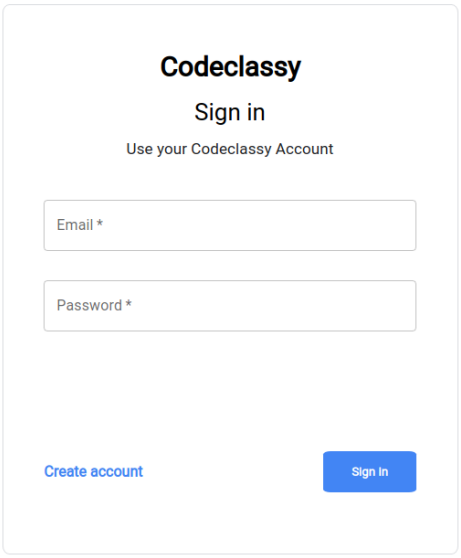

Dashboard
People
A
Students
Coordinators
Teacher Assistants
A
Abullah Abro
Section
7A
U
Unais Siddiqui
Section
7C
A
Anfaal Ahmed Khan
Section
7B
7A
7B
7C


Dashboard
People
A
Students
Coordinators
Teacher Assistants
H
Haseeb Sheikh
Section
7A
A
Tehreem Ahmed
Section
7C
H
Hassan Zahid
Section
7B
7A
7B
7C

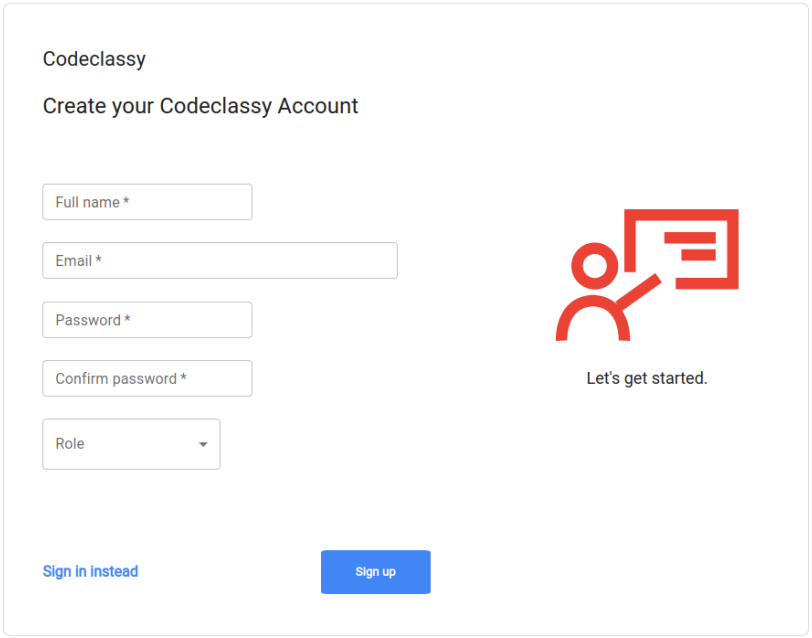


Current Visuals

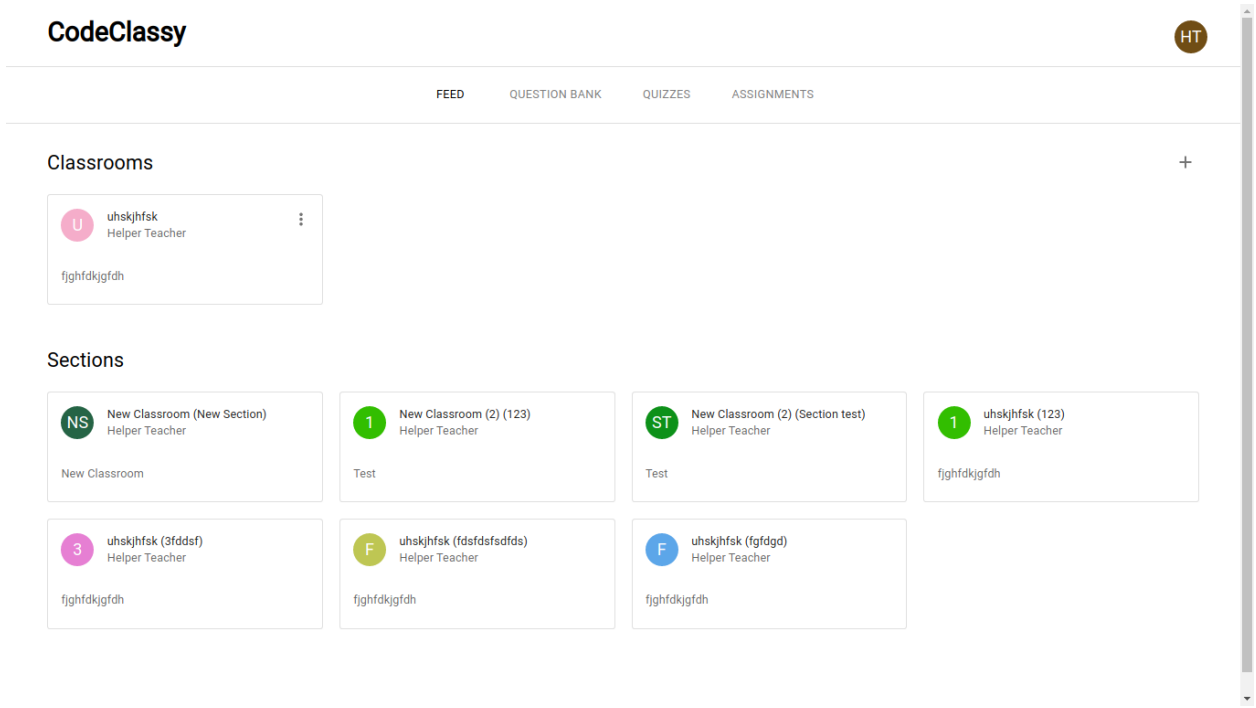
Login



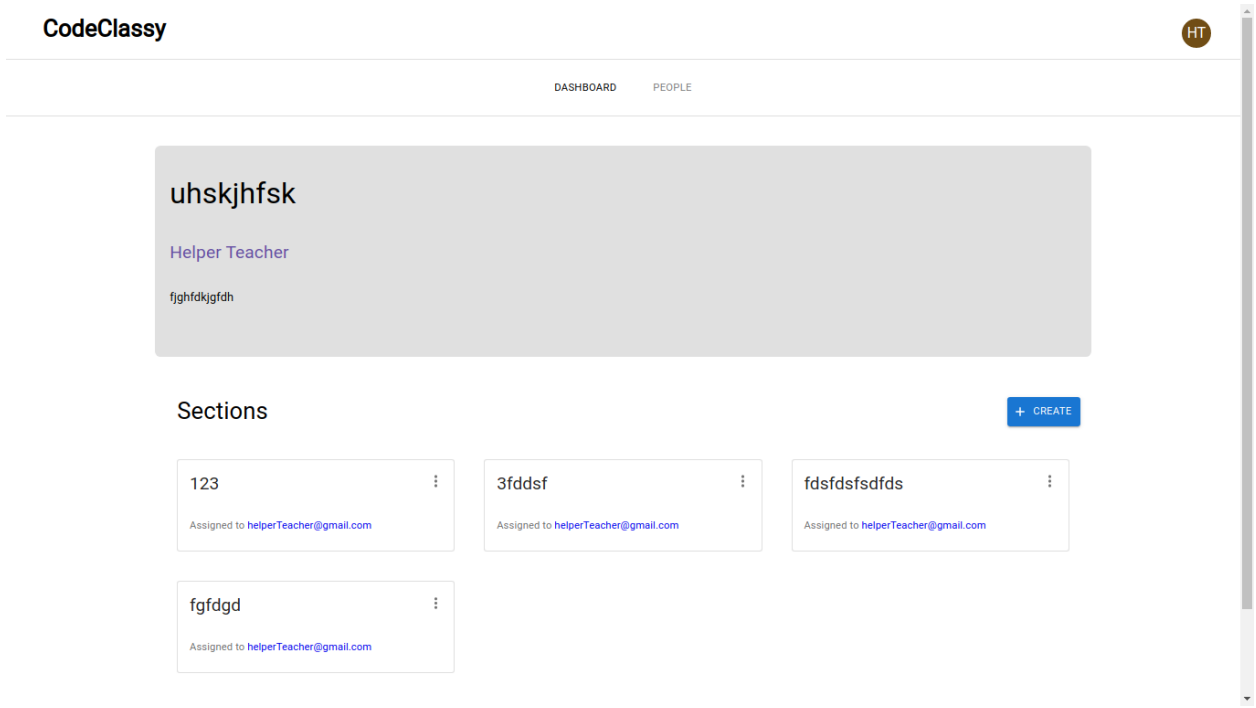
Register



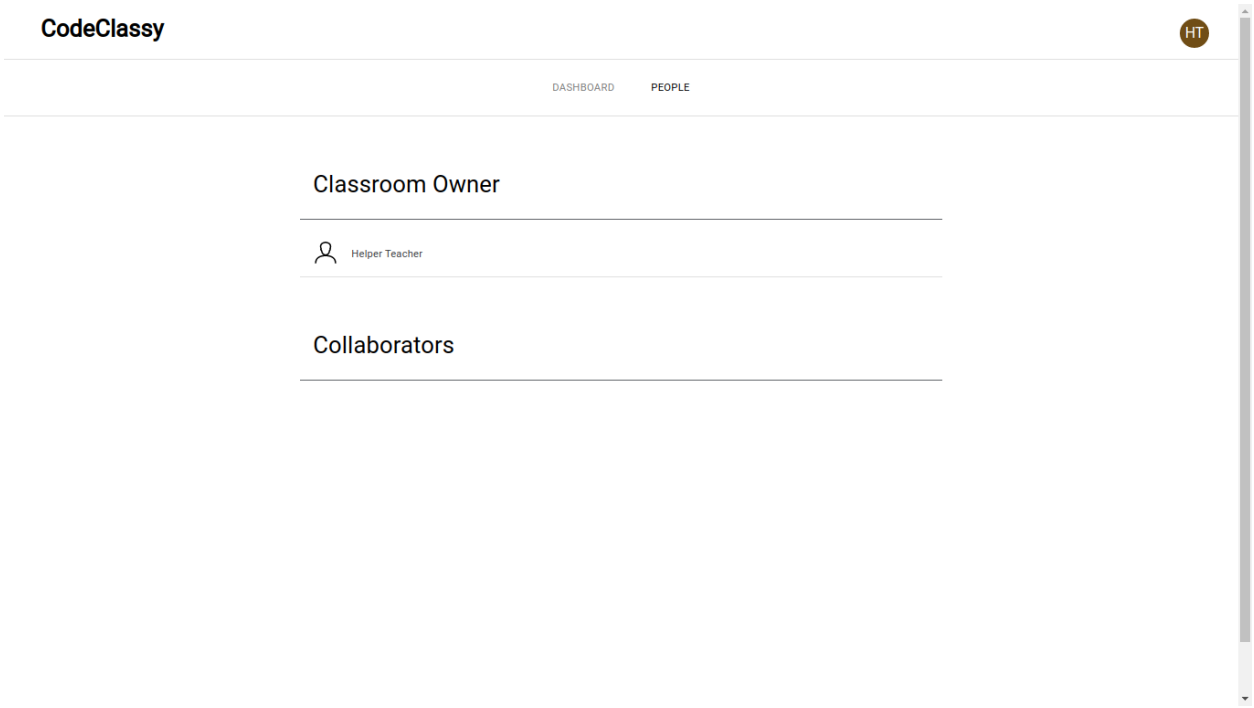
Teacher Home Page



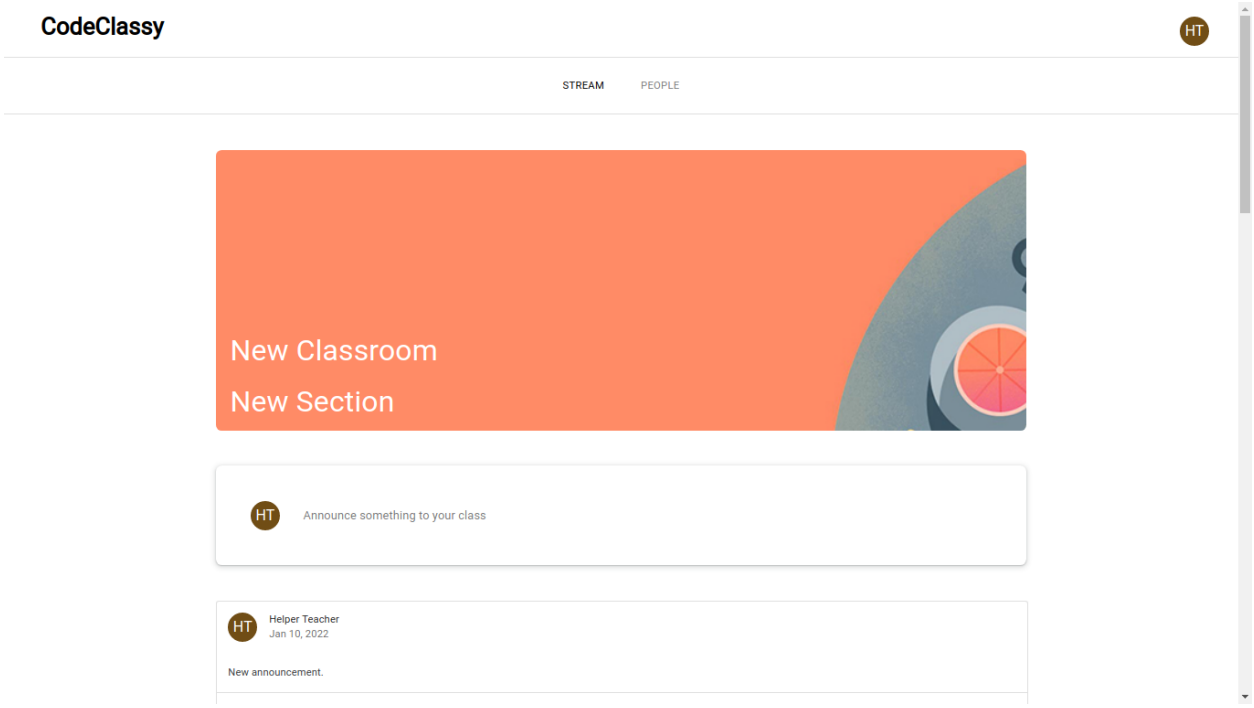
Classroom



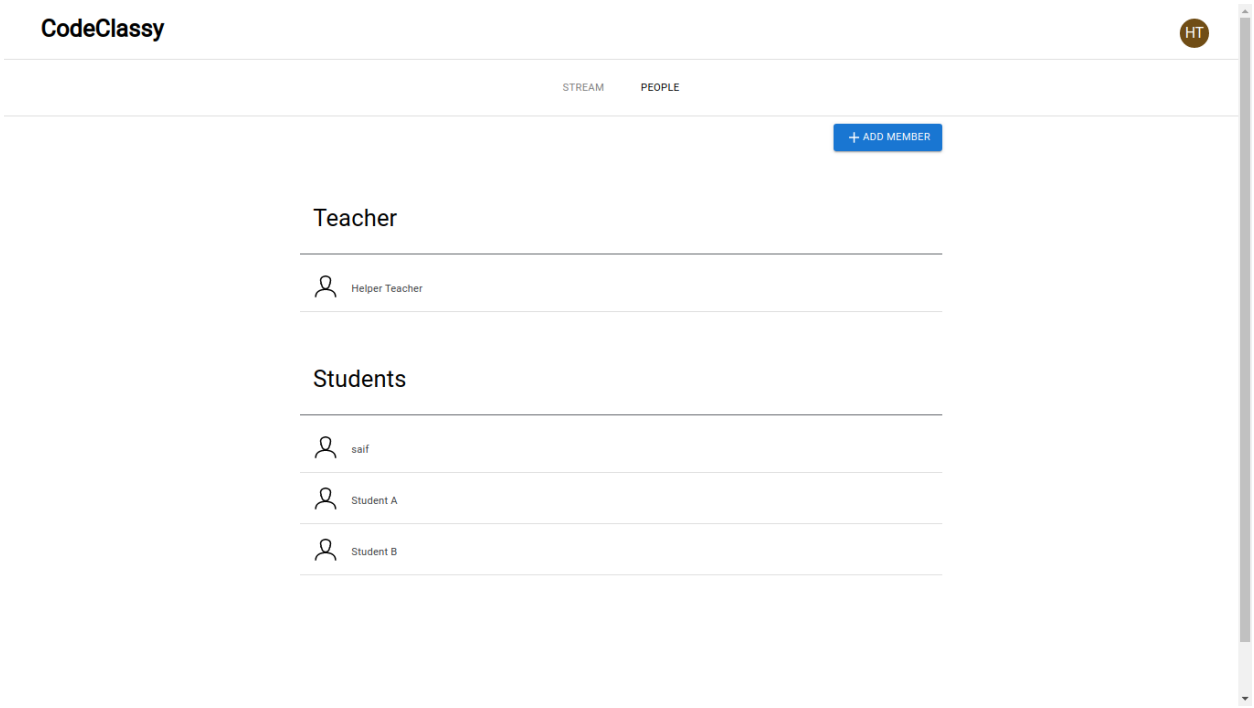
Classroom - People



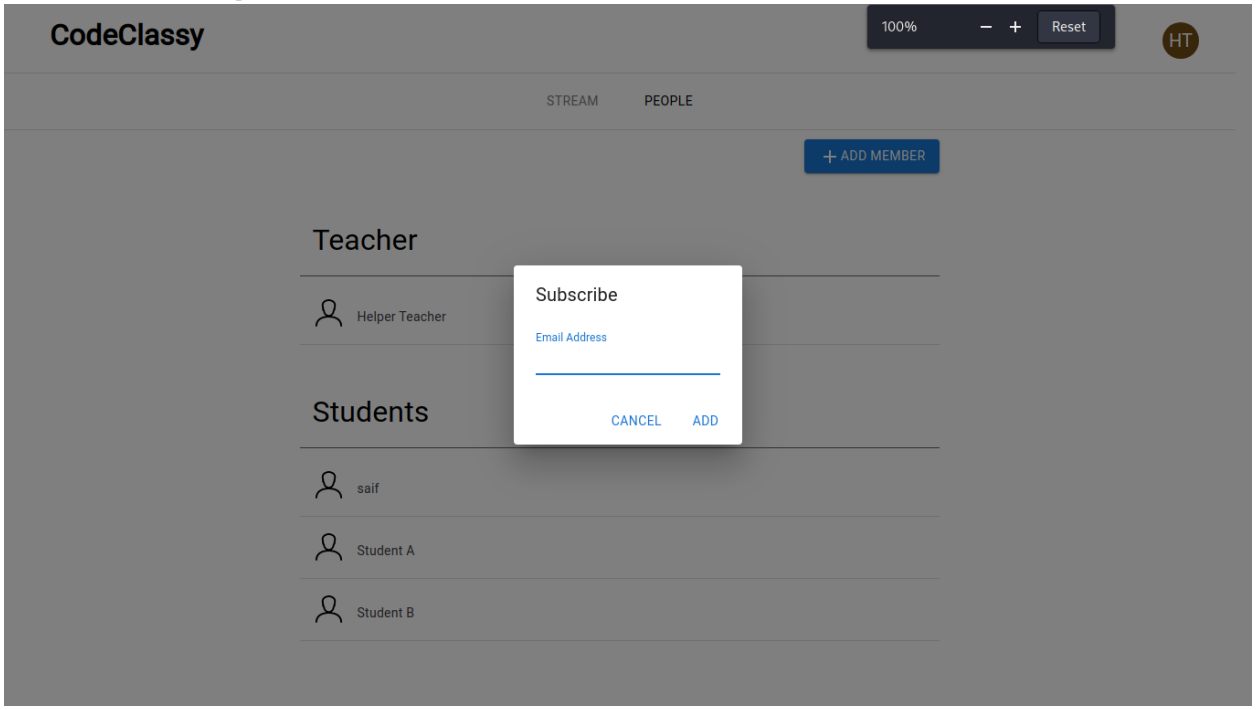
Section



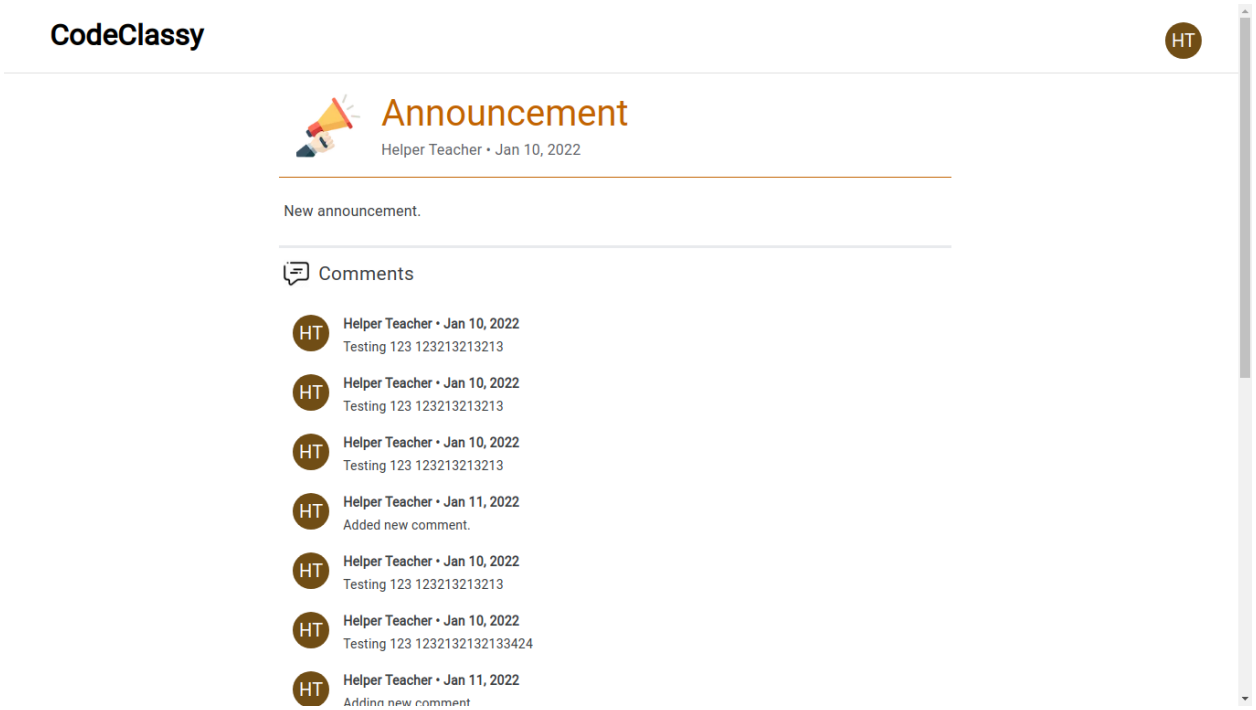
Section - People



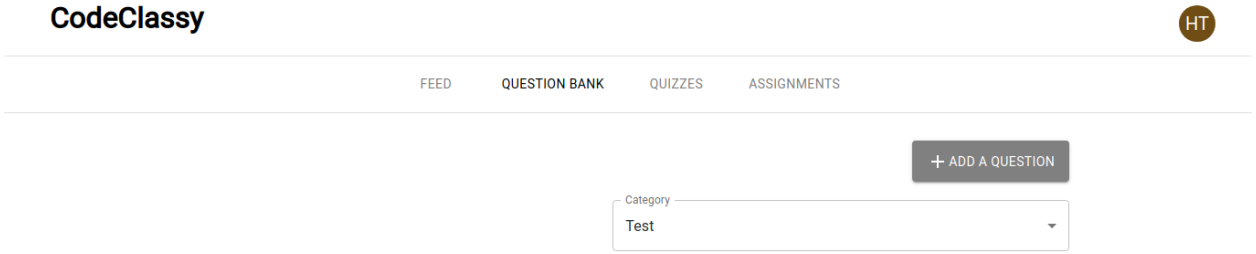
Section - People - Add Member



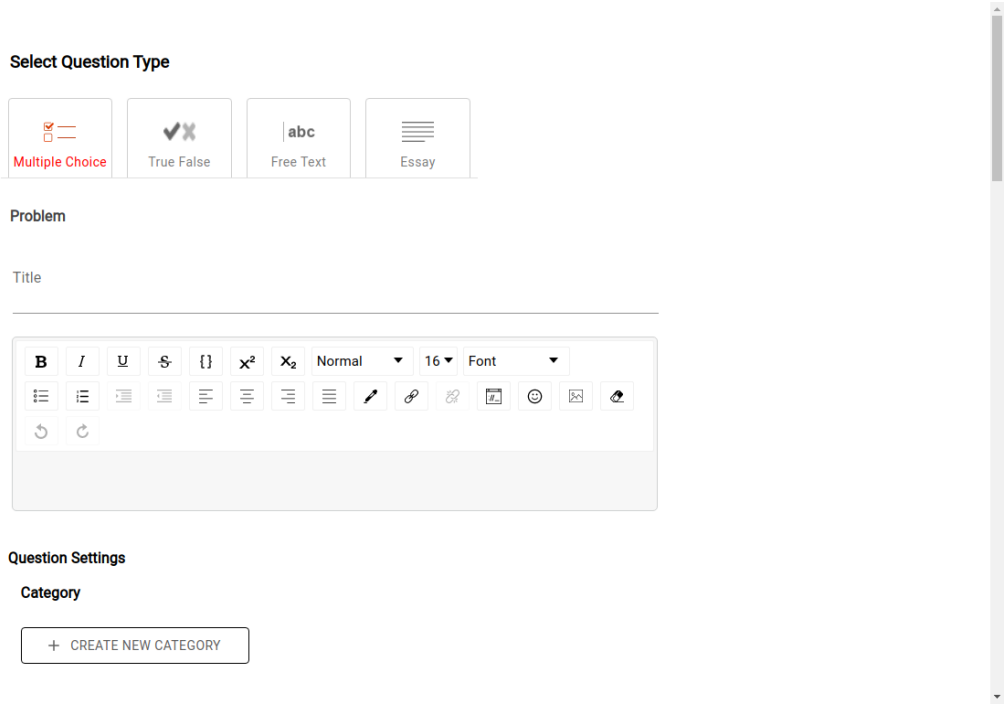
Announcement



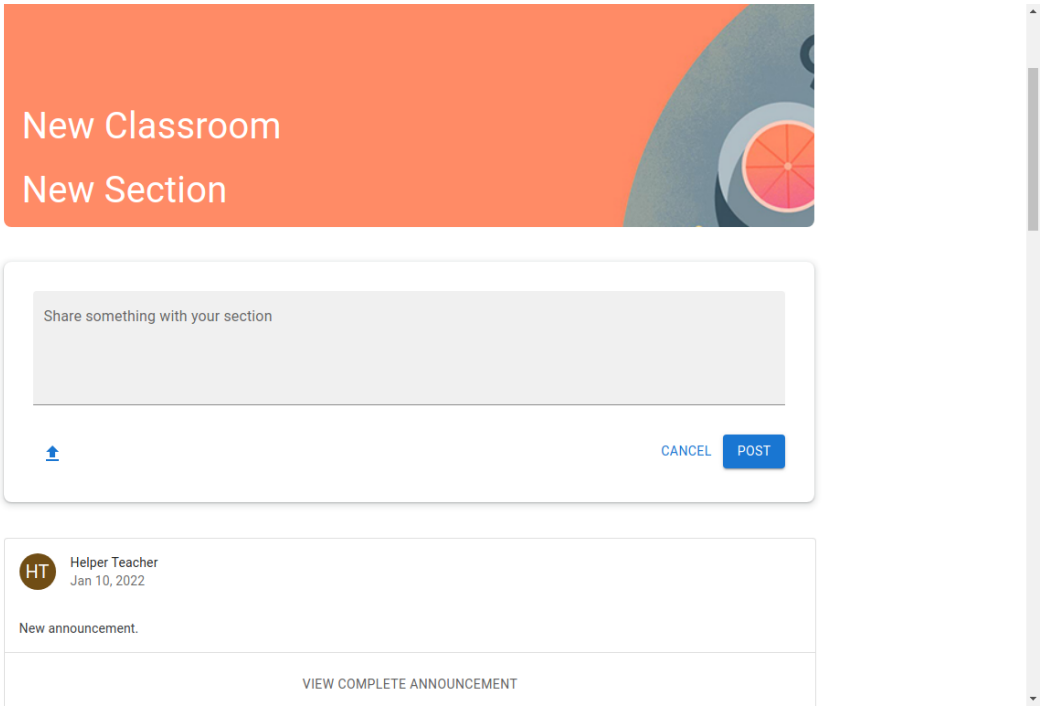
Question Bank



New Question Creation

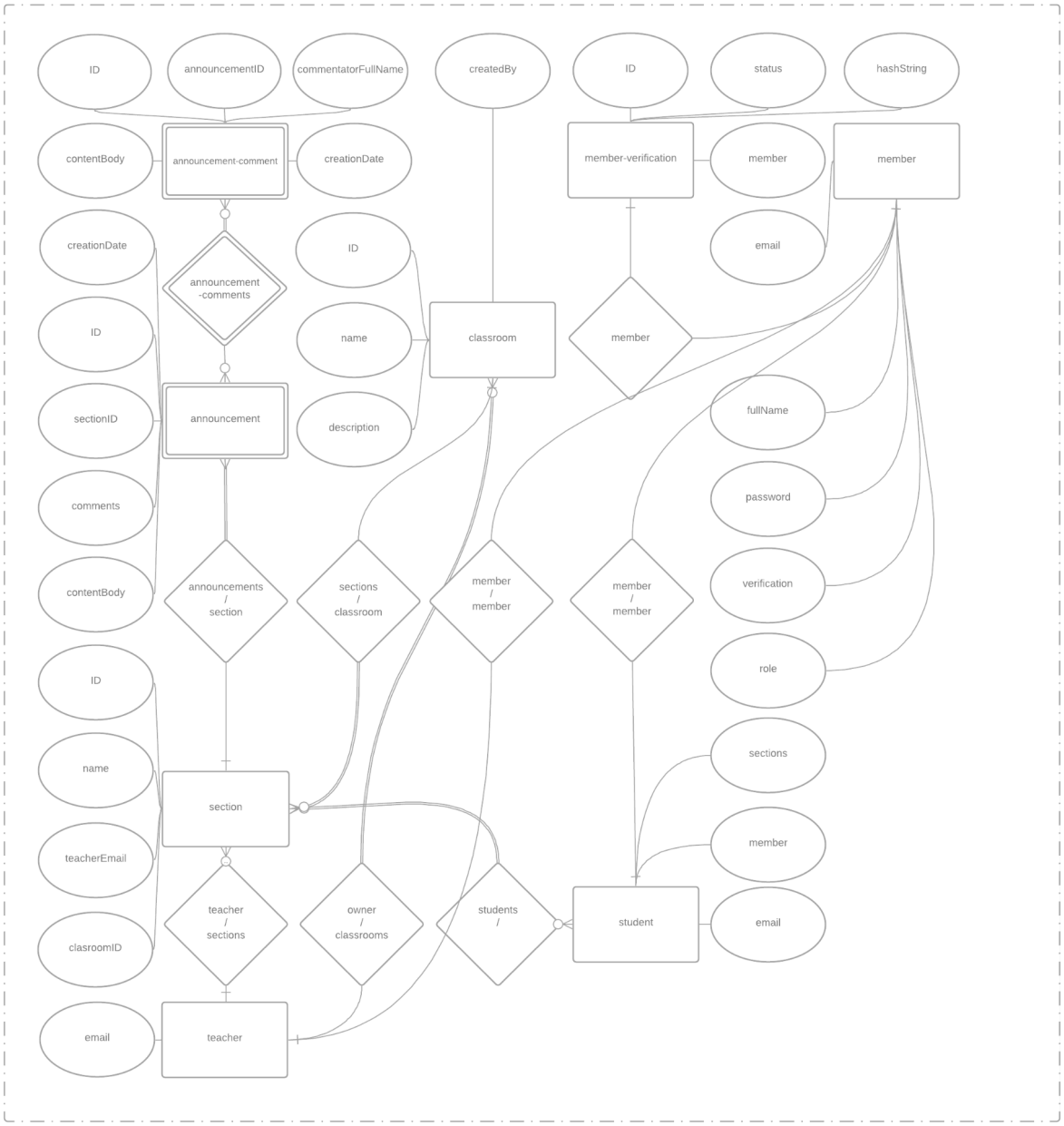


Classroom - Announcement



4.1 Database Design

ER Model



Data Dictionary

announcement

Column Name	#	Data Type	Not Null	Auto Increment	Key	Default	Extra	Expression	Comment
ID	1	varchar(36)	[v]	[ ]	PRI				
sectionID	2	varchar(255)	[v]	[ ]	MUL				
contentBody	3	varchar(255)	[v]	[ ]					
creationDate	4	datetime(6)	[v]	[ ]		CURRENT_ DEFAUI			

annoncement_comment

Column Name	#	Data Type	Not Null	Auto Increment	Key	Default	Extra	Expression	Comment
ID	1	varchar(36)	[v]	[ ]	PRI				
announcemen	2	varchar(255)	[v]	[ ]	MUL				
commentatorI	3	varchar(255)	[v]	[ ]					
contentBody	4	varchar(255)	[v]	[ ]					
creationDate	5	datetime(6)	[v]	[ ]		CURRENT_ DEFAUI			

classroom

Column Name	#	Data Type	Not Null	Auto Increment	Key	Default	Extra	Expression	Comment
name	1	varchar(255)	[v]	[ ]					
description	2	varchar(255)	[v]	[ ]					
createdBy	3	varchar(255)	[v]	[ ]	MUL				
ID	4	varchar(36)	[v]	[ ]	PRI				

member

Column Name	#	Data Type	Not Null	Auto Increment	Key	Default	Extra	Expression	Comment
email	1	varchar(255)	[v]	[ ]	PRI				
fullName	2	varchar(255)	[v]	[ ]					
password	3	varchar(255)	[v]	[ ]					
role	4	varchar(255)	[v]	[ ]					

member_verification

Column Name	#	Data Type	Not Null	Auto Increment	Key	Default	Extra	Expression	Comment
status	1	tinyint	[v]	[ ]		0			
hashString	2	varchar(255)	[v]	[ ]					
memberEmail	3	varchar(255)	[ ]	[ ]	UNI				
ID	4	varchar(36)	[v]	[ ]	PRI				

section

Column Name	#	Data Type	Not Null	Auto Increment	Key	Default	Extra	Expression	Comment
ID	1	varchar(36)	[v]	[ ]	PRI				
name	2	varchar(255)	[v]	[ ]					
classroomID	3	varchar(255)	[v]	[ ]	MUL				
teacherEmail	4	varchar(255)	[v]	[ ]	MUL				

section_students

Column Name	#	Data Type	Not Null	Auto Increment	Key	Default	Extra	Expression	Comment
sectionID	1	varchar(36)	[v]	[ ]	PRI				
studentEmail	2	varchar(255)	[v]	[ ]	PRI				

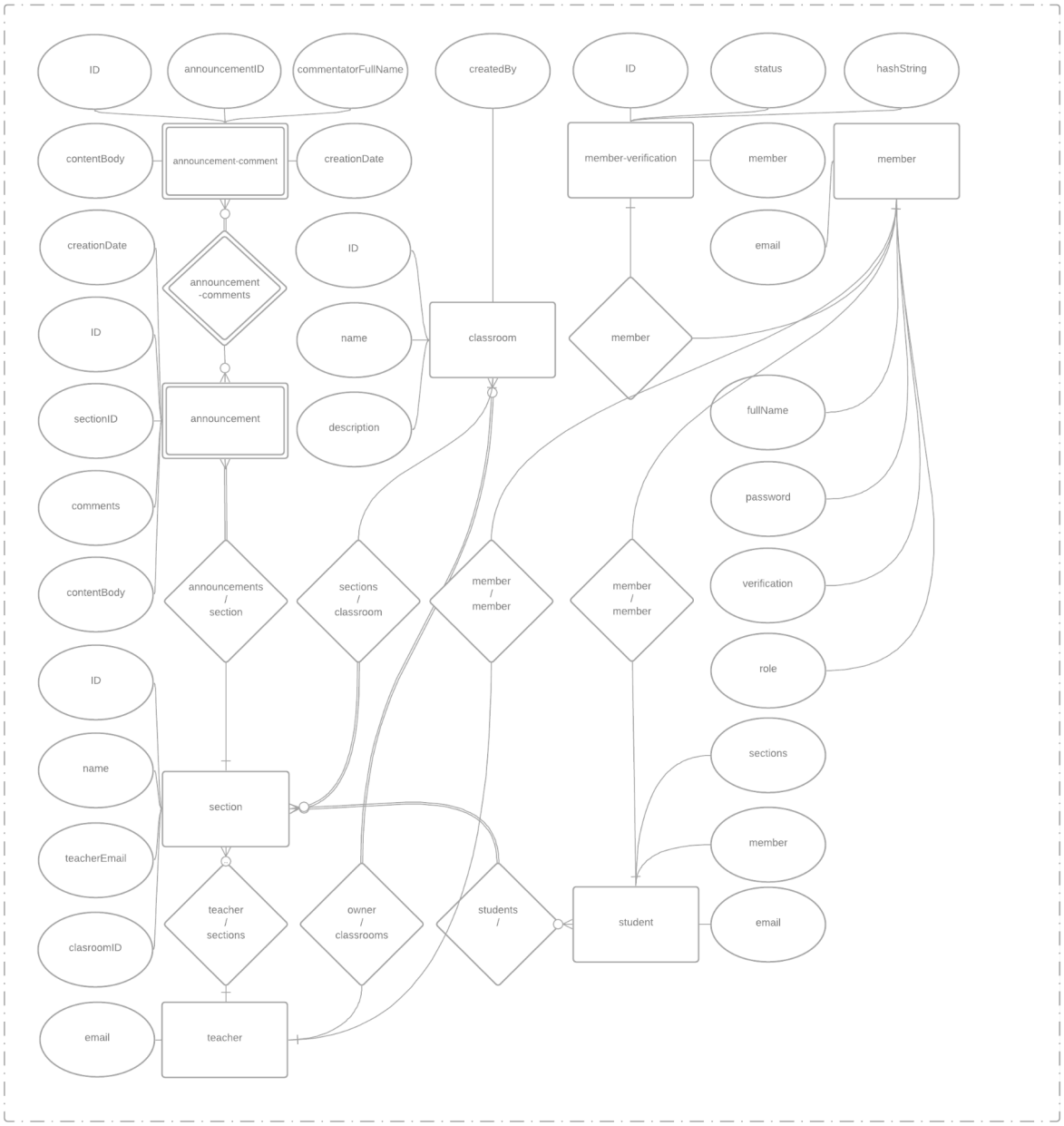
student

Column Name	#	Data Type	Not Null	Auto Increment	Key	Default	Extra	Expression	Comment
email	1	varchar(255)	[v]	[ ]	PRI				

teacher

Column Name	#	Data Type	Not Null	Auto Increment	Key	Default	Extra	Expression	Comment
email	1	varchar(255)	[v]	[ ]	PRI				

4.1.1 ER Diagram



The Entity Relationship diagram consists of currently the following entities,

1. *announcement-comment (WEAK) - for each of the comment that can be posted on each announcement*
2. *announcement (WEAK) - for a post that a teacher can make in a section*
3. *section (WEAK) - for a section that exists under a classroom*
4. *teacher (STRONG) - for teacher representation within the data model*
5. *student (STRONG) - for the student representation within the data model*
6. *member (STRONG) - for the member representation within the database*
7. *member-verification (STRONG) - for verifying the member entity*

8. *classroom (STRONG) - for the classrooms which will be lead by the teacher and where sections can be created*

4.1.2 Data Dictionary

4.1.2.1 announcement-comment

announcement-comment						
Name	announcement-comment					
Alias	NA					
Where-used/how-used	*For each of the comment that can be posted on each announcement*					
Content description	*[\]*					
Column Name	**Description**	**Type**	**Length**	**Null able**	**Default Value**	**Key Type**
email	*The email of the commenter*	*VARCHAR(255)*	*255*	*[v]*	*NA*	*PRI*

4.1.2.2 announcement

announcement						
Name	announcement					
Alias	NA					
Where-used/how-used	*For a post that a teacher can make in a section*					
Content description	*[\]*					
Column Name	**Description**	**Type**	**Length**	**Null able**	**Default Value**	**Key Type**
ID	*The announcement ID*	*VARCHAR*	*36*	*[v]*	*NA*	*PRI*
sectionbID	*The section ID to which the announcement belongs to*	*VARCHAR*	*255*	*[v]*	*NA*	*MUL*

contentBody	*The actual announcement content*	*VARCHAR*	*255*	*[v]*	*NA*	*NA*
creationDate	*The date at which the announcement was created*	*DATETIME*	*6*	*[v]*	*NA*	*NA*

4.1.2.3 section

section						
Name	*Section*					
Alias	NA					
Where-used/how-used	*For a section that exists under a classroom*					
Content description	*[\]*					
Column Name	**Description**	**Type**	**Length**	**Null able**	**Default Value**	**Key Type**
ID	*The section ID*	*VARCHAR*	*36*	*[v]*	*NA*	*PRI*
name	*The name of the section*	*VARCHAR*	*255*	*[v]*	*NA*	*NA*
classroomID	*The ID of the classroom that the section belongs to*	*VARCHAR*	*255*	*[v]*	*NA*	*MUL*
teacherEmail	*The email of the teacher to whom the section is assigned*	*VARCHAR*	*255*	*[v]*	*NA*	*MUL*

4.1.2.4 teacher

teacher	
Name	teacher
Alias	NA
Where-used/how-used	*For teacher representation within the data model*

Content description	[/]					
Column Name	**Description**	**Type**	**Length**	**Null able**	**Default Value**	**Key Type**
email	*The email of the teacher*	*VARCHAR*	*255*	*[v]*	*NA*	*PRI*

4.1.2.5 student

student						
Name	student					
Alias	NA					
Where-used/how-used	For student representation within the data model					
Content description	[/]					
Column Name	**Description**	**Type**	**Length**	**Null able**	**Default Value**	**Key Type**
email	*The email of the student*	*VARCHAR*	*255*	*[v]*	*NA*	*PRI*

4.1.2.6 member

member						
Name	*member*					
Alias	NA					
Where-used/how-used	*For the member representation within the database*					
Content description	[/]					
Column Name	**Description**	**Type**	**Length**	**Null able**	**Default Value**	**Key Type**
email	*The email of the member*	*VARCHAR*	*255*	*[v]*	*NA*	*PRI*
fullName	*The full name of the member*	*VARCHAR*	*255*	*[v]*	*NA*	
password	*The password of the member*	*VARCHAR*	*255*	*[v]*	*NA*	
role	*The role of the member*	*VARCHAR*	*255*	*[v]*	*NA*	

4.1.2.7 member-verification

member-verification						
Name	member-verification					
Alias	NA					
Where-used/how-used	*For verifying the member entity*					
Content description	[/]					
Column Name	**Description**	**Type**	**Length**	**Null able**	**Default Value**	**Key Type**
status	*The status of the verification*	*tinyint*	*NA*	*[v]*	*0*	*NA*
hashString	*The hash generated from the string*	*VARCHAR*	*255*	*[v]*	*NA*	
memberEmail	*The email of the member*	*VARCHAR*	*255*	*[ ]*	*NA*	*UNI*
ID	*The ID of the member to be verified*	*VARCHAR*	*36*	*[v]*	*NA*	*PRI*

4.1.2.8 classroom

classroom						
Name	classroom					
Alias	NA					
Where-used/how-used	*For the classrooms which will be lead by the teacher and where sections can be created*					
Content description	[/]					
Column Name	**Description**	**Type**	**Length**	**Null able**	**Default Value**	**Key Type**
name	*The name of the classroom*	*VARCHAR*	*255*	*[v]*	*NA*	*NA*
description	*The description of the classroom*	*VARCHAR*	*255*	*[v]*	*NA*	*NA*

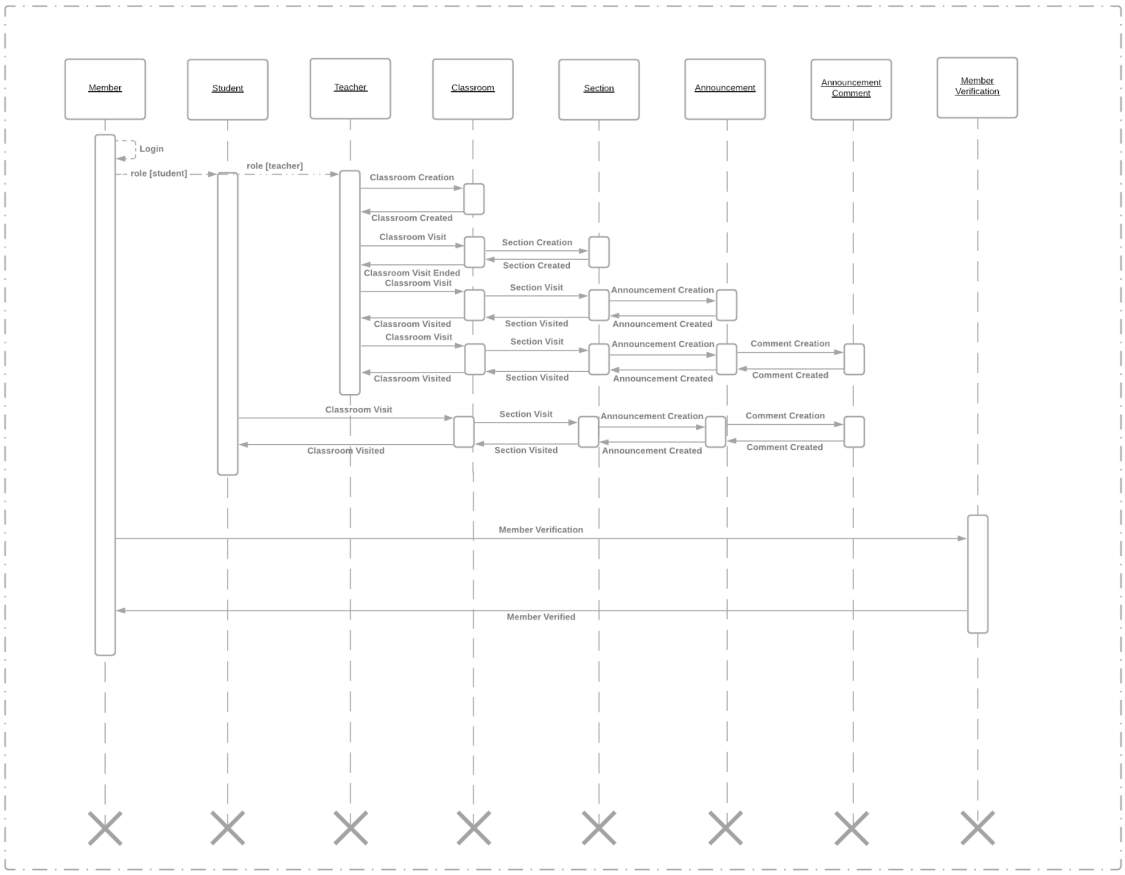
createdBy	*The person who created the classroom*	*VARCHAR*	*255*	*[v]*	*NA*	*MUL*
ID	*The ID of the classroom*	*VARCHAR*	*36*	*[v]*	*NA*	*PRI*

The notation to for the content description is given below:

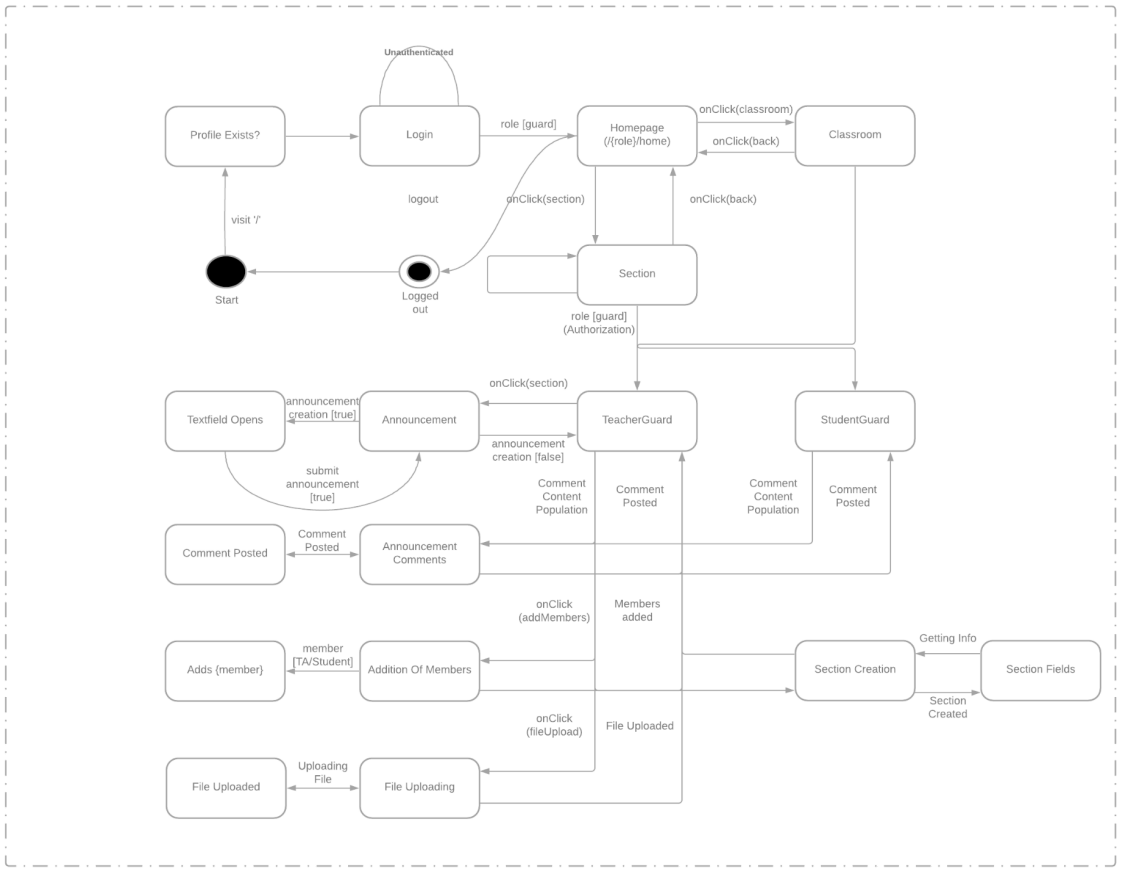
Data construct	***Notation***	***Meaning***
	=	*is composed of*
Sequence	+	*And*
Selection	*[]*	*either-or*
Repetition	*{}n*	*n repetitions of*
	( )	*optional data*
	* … *	*delimits comments*
]		

4.2 Application Design

4.2.1 Sequence Diagram



4.2.2 State Diagram



4.2.2.1 References

*[1]: **James Ivers**, JI, November 2021, "Untangling the Knot: Enabling Rapid Software Architecture Evolution", Software Engineering Institute, https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=742903*

5 Appendices

NA