Visual Overview:

Video is added to the repo under the name ‘Report_Visual_Overview’. First logged in as a student, then an instructor, then a faculty advisor, then a graduate secretary, then showed the applicant feature.

Design Justification:

In our Phase 2 ARG Project, we decided to create an organized UI layout that is pleasing to look at and easy to navigate for the different users, such as applicants, students, graduate secretaries, instructors, faculty advisors and reviewers. We ensured that our UI looks the same throughout the different pages and roles, and we also made sure every role had easy access to their functionality, such as a graduate secretary assigning a student’s grade or allowing them to graduate. We changed SELECT statements in our methods to accommodate the new working database we created, ensuring all features from Phase 1 still work.

We created one login function that allowed all the roles to log into their different designated home pages. We integrated our components seamlessly, by creating new tables in our new database to fit 3NF, and integrated our different roles to have the same features. For example, a graduate secretary’s roles and responsibilities now work with current students and their ability to graduate and become an alumni. The REGS and ADS data of students are now integrated by fetching the data from the same tables such as the ‘student_course’ table. Students can send UAF and Form 1 forms to the faculty advisors for approval, and can communicate through an email feature.

Special Features:

We have integrated the email messaging system from Junha’s phase 1 to use it to store different messages such as forms, requests, and reguler emails into the system. Students can use it to submit form1 requests, graduation requests, and uaf form submission and faculty advisors and graduate secretary can approve or decline the form.

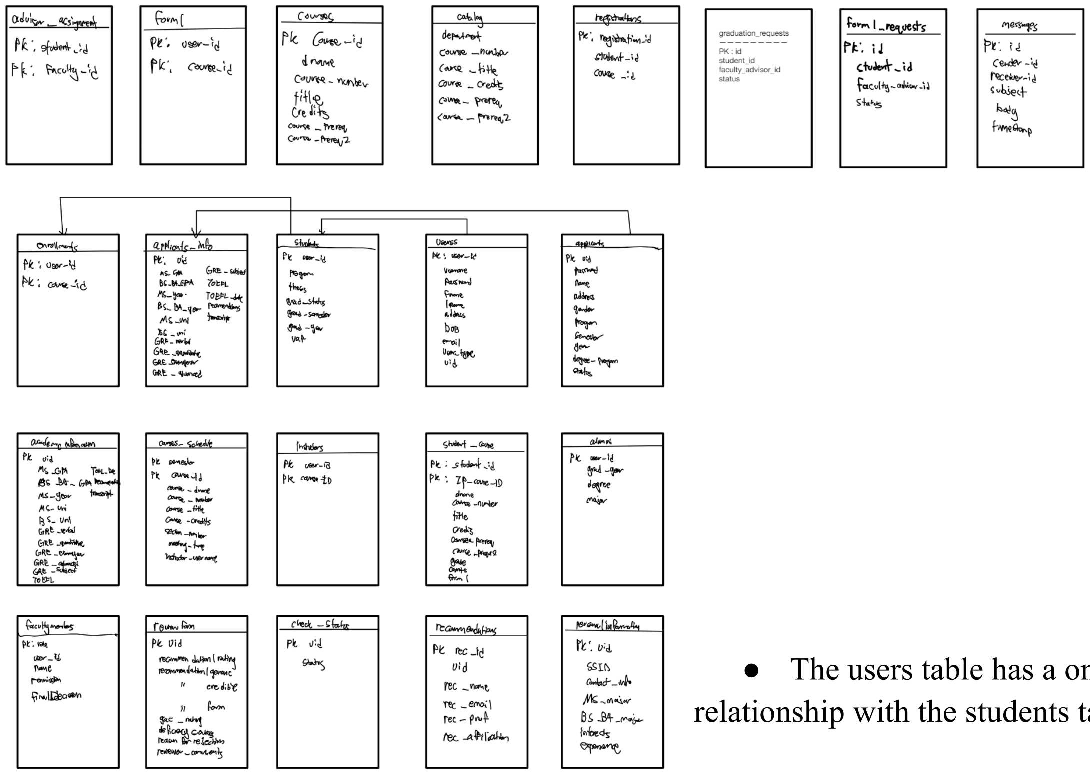
Work Breakdown:

Yan -
Create/integrate the login function to allow all types of roles to log into their designated home pages. Add the applicant's side admission decision for the applicant to accept or decline their admission functionality in main.py and html file. Add a function to allow applicants to pay an admissions fee. Add all of the APPS methods and functionality to the project (rewrite it to fit the new database).

Sara -
Contribute to create one combined database with tables to fit 2NF and 3NF. Create the matriculate function to matriculate an admitted applicant, into a new student. Rewrite registration function to fetch course information such as prerequisites and time conflicts from newly created database. Rewrite the assignGrade function for instructors and GS to assign student grades. Rewrite all REGS functions to accommodate the new database. Add a query for students' transcripts. Create one home.html page for all roles, edit it so different roles see different functionalities. Style *all* html files in the project to maintain a clean and organized UI layout using BootStrap.

Junha -
Contribute to create one combined database with tables to fit 2NF and 3NF. Create a query for list of graduate applicants, list of admitted students, list of graduating students. Integrate faculty functionality, and create all faculty advisor features. Allow System Administrator to access everything and create new users. Work on Alumni functionality. Create UAF functionality for new students to fill out, get approved by an advisor, then are able to register for classes. Integrate all ADS functionality to work with the new database. Add extra features (email system functionality).

DB Design



- The users table has a one-to-many relationship with the students table

- The applicants table has a one-to-one relationship with the applicants_info table. Each applicant has corresponding information stored in the applicants_info table.
- The applicants table also has a one-to-one relationship with the personalinformation table.
- The users table has a one-to-many relationship with the alumni table
- The users table has a many-to-many relationship with the courses table through the instructors table.
- The courses table has a one-to-many relationship with the courses_schedule table. Each course can have multiple schedules, but each schedule is associated with a single course.
- The courses_schedule table has a one-to-many relationship with the catalog table
- The students table has a many-to-many relationship with the courses table through the student_course table.
- The students table has a many-to-many relationship with the courses table through the enrollments table
- The students table has a many-to-many relationship with the courses table through the form1 table
- The students table has a one-to-many relationship with the advisor_assignments table.
- The users table has a many-to-many relationship with itself through the messages table.
- The students table has a one-to-many relationship with the form1_requests table
- The students table has a one-to-many relationship with the graduation_requests table.
- The applicants table has a one-to-many relationship with the recommendations table.
- The applicants table has a one-to-one relationship with the checkstatus table
- The applicants table has a one-to-one relationship with the reviewform table
- The users table has a one-to-many relationship with the facultymembers table
- The students table has a many-to-many relationship with the courses_schedule table through the registrations table