**The Plan**  
 The CIS Department Student App will be a one-stop shop web app for CIS student needs relating to the department. It will keep track of an individual student’s pre-professional and professional program GPA and the status of the student’s professional program application. It will have information about the student’s advisor, how far along they are in their degree program, and allow the student to fill out the professional program application automatically.

Optionally, we may integrate information on students’ CIS accounts and Beocat account. Also in consideration is a calendar for 018/015 events that are coming up along with tracking of 018/015 credit.

Our stakeholders are CIS students in the Pre-Professional and Professional programs, CIS department faculty and staff, and advisors to these students. Initially, we will be looking to redo the professional application backend to prepare the student webapp for use, since there has already been some work completed in that area. Once we have completed that, our next goal is to attach the existing code to the new application backend. We will then move to develop the front-facing experience.

After project completion, it will be maintained by CIS faculty and staff.

For the front end, we will be using React JS, as the previously built parts of the project already use that library. For the back end, we will be working with NodeJS and SQL (PostgreSQL) to interact with KSIS and organize the students’ data.

**The Execution**

For this semester, we have solely worked on replacing the existing professional program application website. The existing backend is in PHP and our goal has been to update it to something more modern and to improve maintainability while doing so. We are close to having the original recreated with JavaScript and HTML and once this is completed, we will move towards deploying the new version and subsequently updating it to go beyond what the existing application does.

**Client**

The client consists of all of the React JS elements. All three UI pages; the Home Page, Application Page, and the Profile page are all contained within the client. The client also contains the .css file for all of these elements. The UI design was based on the prior professional program application and the feedback gathered from our customer meetings. These elements utilized two packages, react and react-router-dom. These packages are necessary for the react pages to load and switch pages.

**Database**

Using Knex and Postgres, we have created a series of database migrations and seeds to prep the information that we will eventually be pulling from DARS and the applications database. Each migration makes one of the required tables in the DB, and the seed files will be used to add debug info or admin users as needed.

As we move towards production, we are looking at pulling from K-State's Degree Audit Reporting System (DARS) to pull student data to automatically fill the application and avoid the existing lengthy manual approval system. We will verify that the student is the right person by going through the Central Authentication System (CAS) before pulling anyone’s data. Once we have the data pulled. We will only need to send it to be approved with any attached notes from the student.

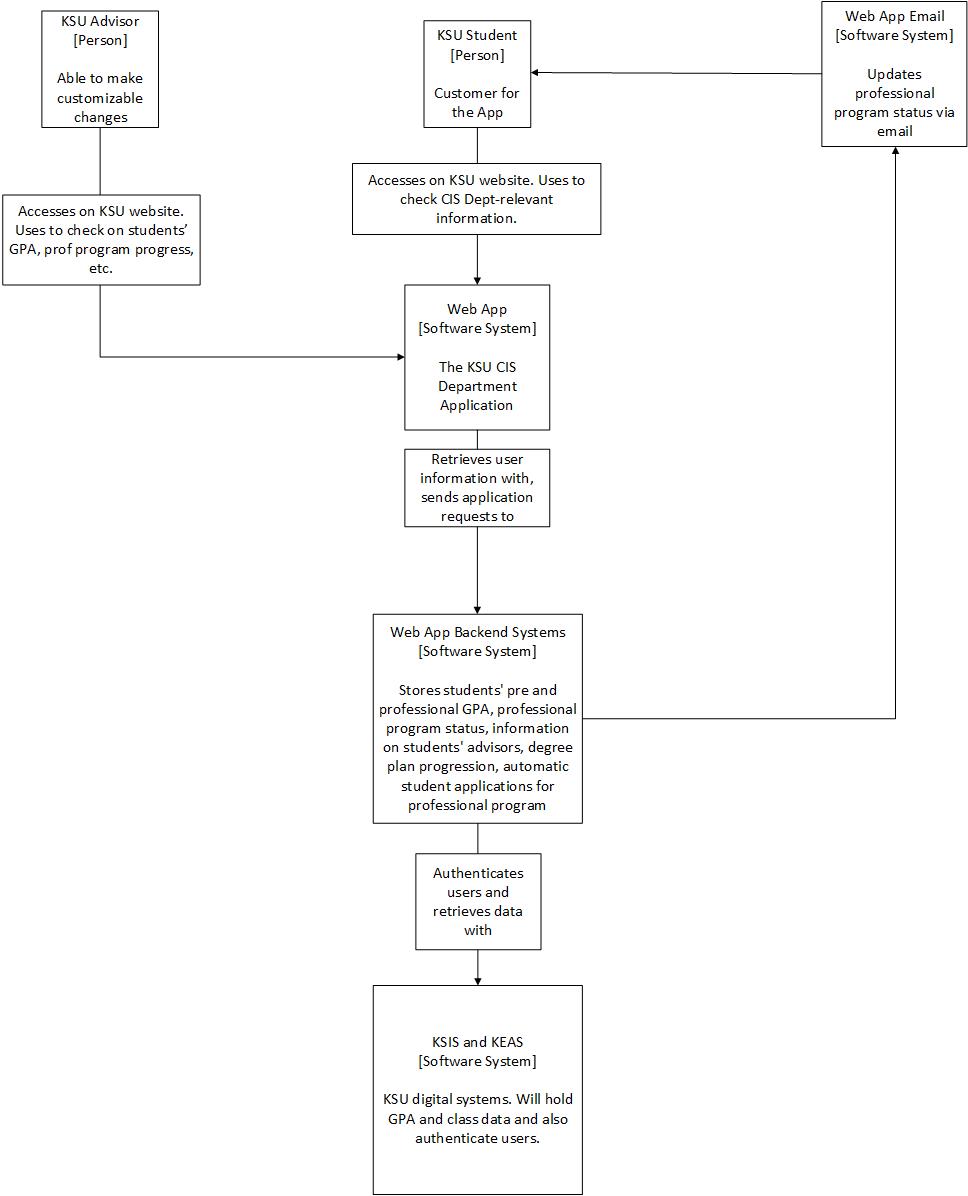
**Server**

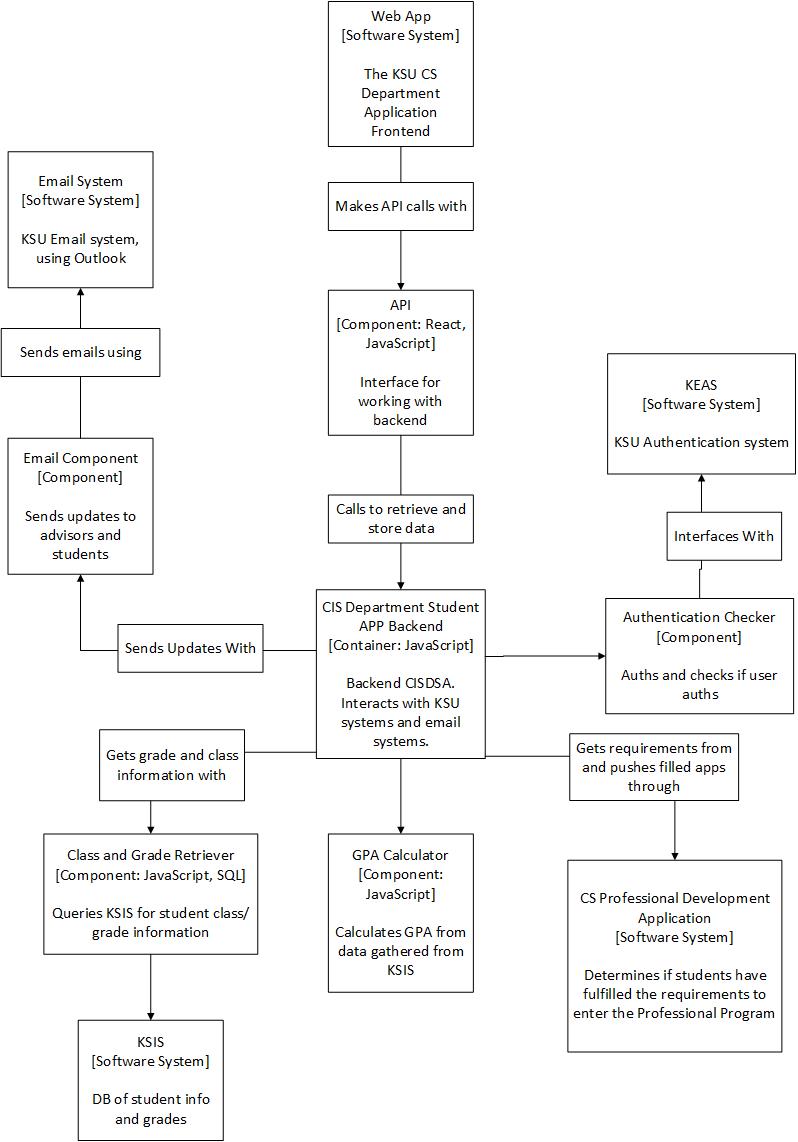
The backend code’s objective for this project is to retrieve what the client side wants to display, reach into the database for that information, and display it on the client side. Going in alphabetical order in the server folder, first is the configuration folder. This folder handles all configurations for the server side. This includes, but is not limited to, the CAS data, the logger for the terminal, and the session. Within the CAS file, a new CAS authentication variable is created to authenticate through the KSU resources that our program is eligible to retrieve data from. The configuration folder also has a default file, this may not be necessary as it enables the data base to be active and not disabled when necessary. The logger file just handles situations while developing to make messages within the terminal color-coded log levels along with a timestamp of each log line. This also saves the logs inside files. The session file reaches into the database to see how long the user has been using that specific session.

The second folder within the server are the endpoints, or routes. The routes are the path in which the client requests talk to the appropriate endpoints. Again, in alphabetical order, the first folder is for the apply page. Within this folder there are two files, “process-application” and “show-application”. The process application’s functionality is to process any additional information the student may provide in their application communicating with the database to add or remove any additional information. The show application’s functionality is to show the status of the student’s progression in the preprofessional program at Kansas State University. This will show what courses the student has already completed and the cumulative GPA of those courses. The next folder within the endpoints is the disabled folder. This folder has two files “disable” and “isDisabled”. The disable file’s functionality is to disable the database when the student wants to disable their application. The isDisabled file’s functionality is to tell the student that their current application is disabled and additional effort may be needed to continue with the process. Next is the email endpoint which has two files, “process-email” and “send-email”. These two endpoints functionality is to use node-mailer and configure the appropriate email about the student’s application. This can look like an email saying they have passed all the needed requirements for the application and they have been accepted, or an email saying there are some issues and they need to continue to work on the application as it does not meet the requirements. Next is the login folder. This folder contains the files “process-login” and “show-login”. The process login file reaches into the database and CAS to make sure the student is a KSU student using an eligible login so they can reach the rest of our program and continue with their application process. The show login file shows the login screen that has been created to the student so they are able to login. The last folder is the profile folder. Currently this folder is empty as at first we did not see a reason for the student to need a profile page on the application program, but it was concluded that additional information about their status within the preprofessional program, such as GPA, can be displayed to them to make their lives easier. Additionally, the endpoints folder has an api and index file to use the routes for the application that were described above.

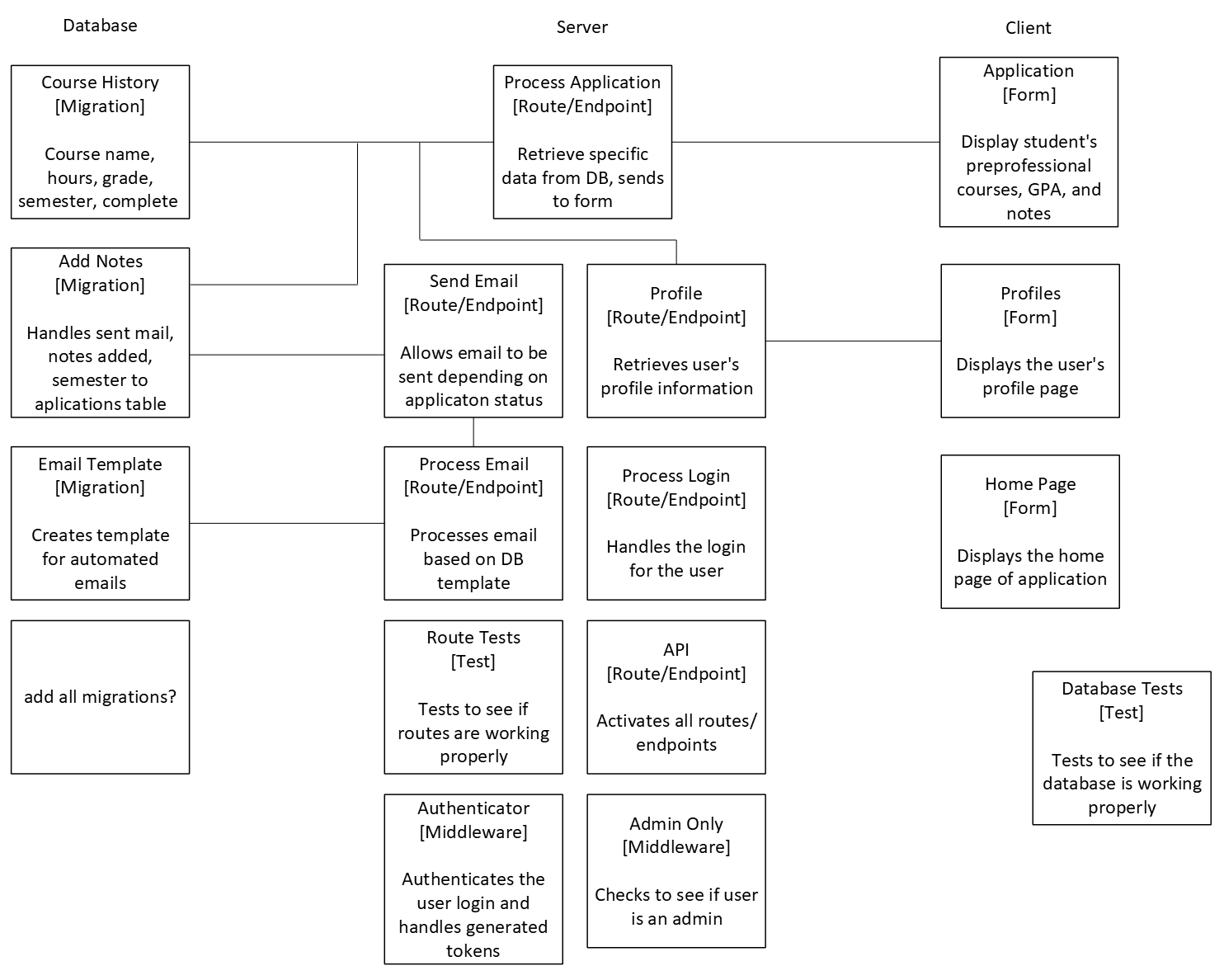
The server also contains a folder for the middleware. This includes an “admin-only” file that checks to see if the user that is logging in to the application is considered an admin. An “auth” file that authorizes the login, logout, and json web tokens. A “request-logger” file to help with the logger file described above, and to skip the logger file when not in development mode. Lastly, the “token” file to handle the json web tokens.

Lastly, the server has models. All of the models are made to specifically help the endpoints reach the database and execute correctly. However, I made a mistake and started to use an express extension called Winston and this extension needs a specific database type that we are not using for this application. Additionally, while we were doing our code review with Dr. Bean, he had informed me that I may not need to use the models in the first place and most of the work that I had done was too much and I just needed to make it more simple yet more effective and efficient. In conclusion, the models that are currently in our project will more than likely not be used in the final product.



In the figure above, you can see that we will have customers go through KSU webpages to get to the website, where they will be able to view their information and submit applications. Not pictured is the fact that they will need to go through KSU’s SSO to get onto the pages first.

Per the figure above, we are going to be initially sending customers through KEAS and then we will query their grade/class information from KSIS with SQL. We can then compare the data with requirements pulled from the application and then allow students to submit professional applications.



Above is our in progress diagram for the application aspect as it exists currently.