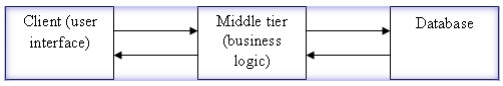
# Software Architecture and Components

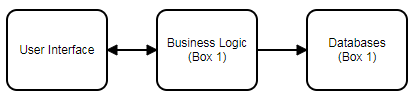
Artefact 1 includes the designs of architecture patterns and components that are used to cater for the client’s request I designed and proposed to my team. The designs revolve around the business scope and focus to model an efficient schematic that allows the data flow to be smooth and secure.

I utilized the three-tier software architecture pattern to allow the client to talk to the backend to access the website while the backend fetches the necessary data from the database as shown in Figure 1. This ensures service availability for the users to access the website at their leisure while the backend keeps the database safe at the same time as the users cannot access the database themselves.



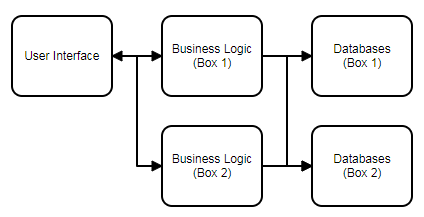
*Figure 1: Software architecture pattern*

Shown in Figure 2 is the physical design of the architecture pattern. For the first sprint, the client and the developers came to agreement to have a working product first, adhering to agile development procedure. It has been discussed to evolve the current physical design to a more high availability (HA) model shown in Figure 3 in the future.



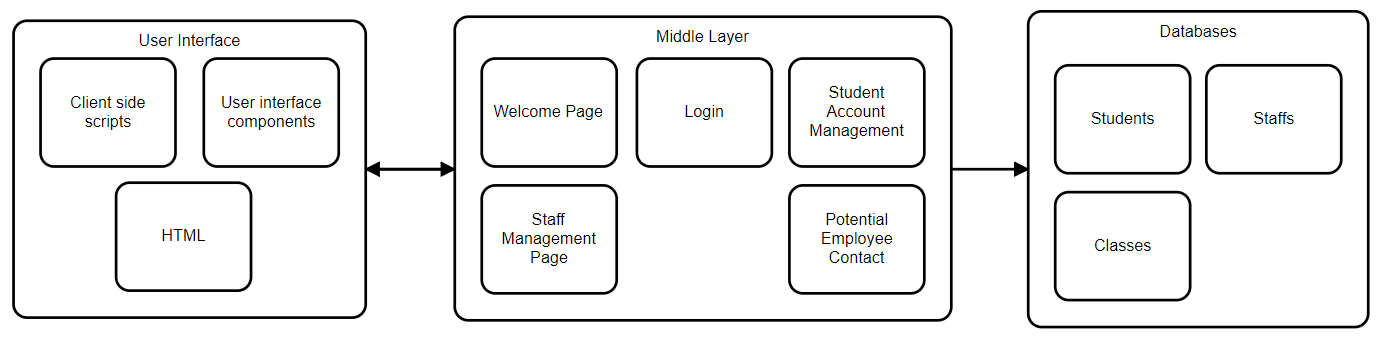
*Figure 2: Physical Design*

The HA design allows the service to be available at all times even when the server needs to be updated or is shutdown. As an example, shown in Figure 3, the business Logic box 1 would be active while accessing the database box 1 and having a backup in database box 2. If the business Logic box 1 is down due to unforeseen circumstance, instead of the service cutting off, the users will be redirected to Business Logic box 2 and continue to provide for their needs. If the server needs to be updated or Business Logic needs to be modified, rather than shutting down the entire network, the update would be done on the one of the box then switched for HA and update the other box.



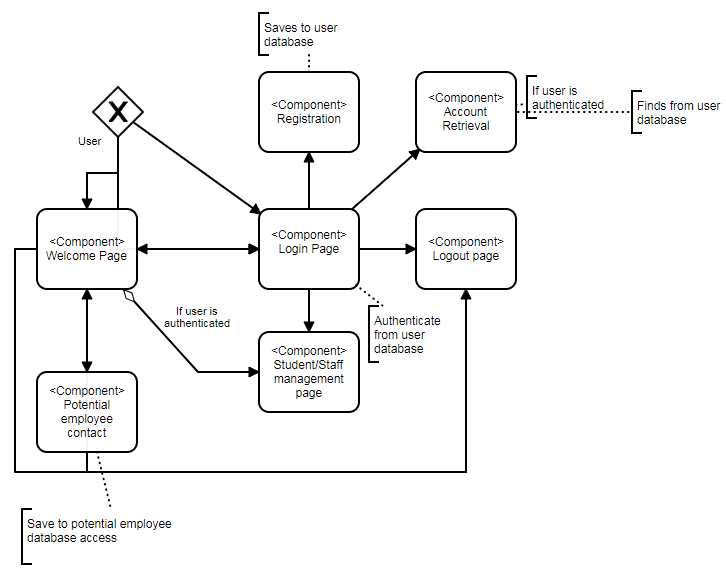
*Figure 3: Physical Design (Optional)*

To adhere to agile development procedure, I only included certain components within each layer of services following the MoSCoW criteria. During sprint 1, the group 5 has implemented welcome page, login services, student account management page, staff management page and potential employee contact page with database connection to allow immediate deployment.



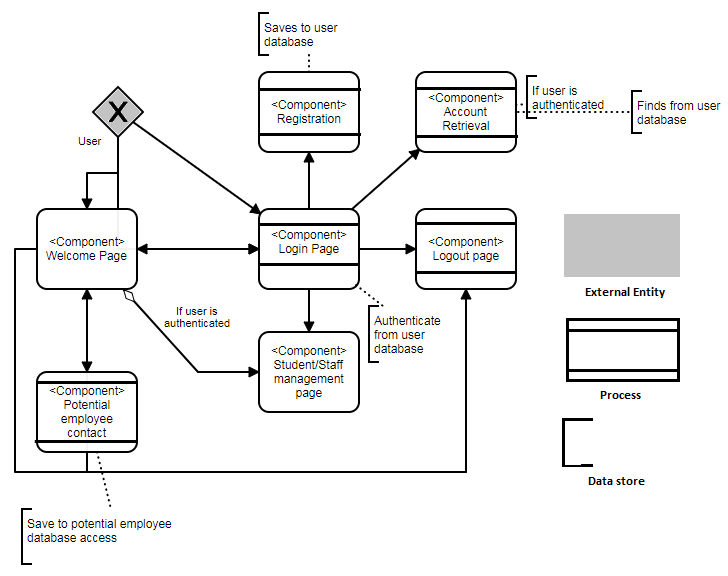
*Figure 4: Sprint 1 Component Diagram*

Another artefact I created was the software component diagram present the relation each components have to the group. It first starts from the user marked by x within a diamond shape shown in Figure 5. The user can access the welcome page and browse without log in, however cannot access student/staff management page without authenticating as of sprint 1. The user can access the potential employee contact component without any account. This component will save the recorded data to the potential employee database. The user can register freely or retrieve their account through the login page. The registration component access’s the user database to record the user’s account information and allows access to management page. The account retrieval page finds an existing account if the user is authenticated through a procedure.



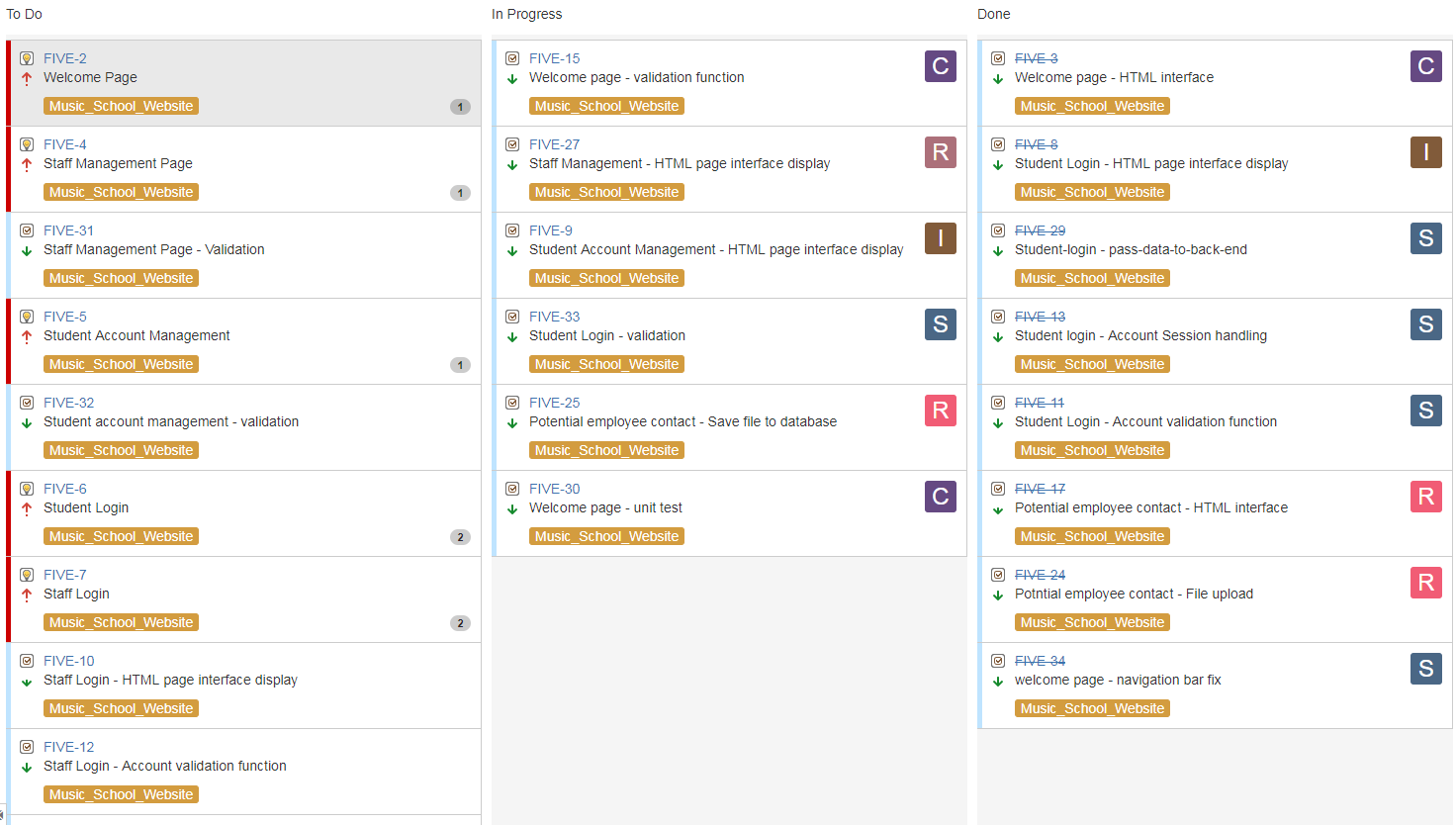
*Figure 5: Software Component Diagram*

The data flow shown in Figure 6 represents the flow of how the data is handled where the user is considered an external entity as they brings input to the server. The process components are there to conduct algorithms to process the data given by the user. Finally the data is stored and accessed once the process has been completed.



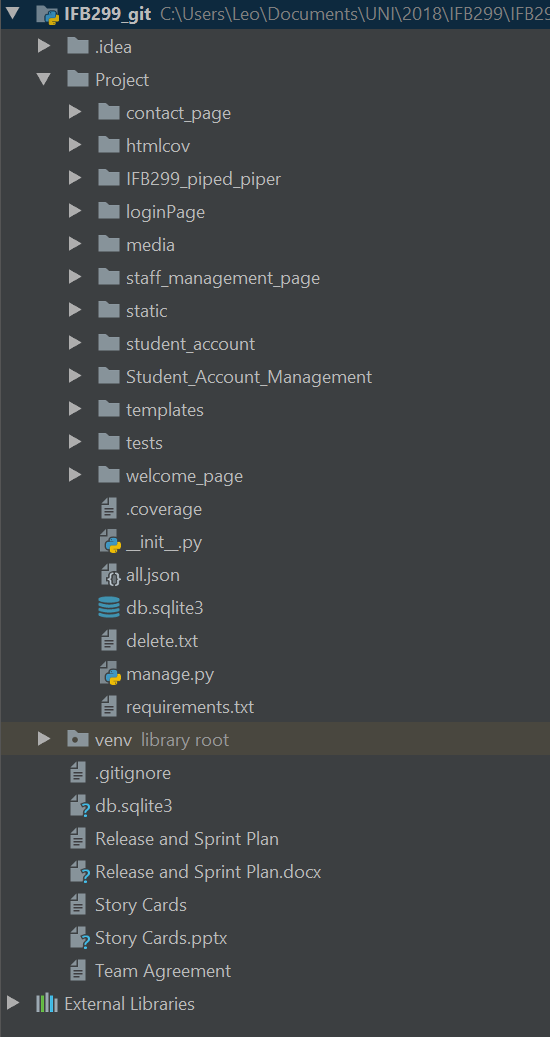
*Figure 6: Data Flow*

All architectural and component artefacts were used in a chain reaction of sort. The software architecture pattern was used to create the logical and physical design. Which was then extended to a HA model. The physical design inspired how and where each components of the project will be placed in. I then created each components mentioned in Figure 4 within JIRA as user stories and task cards. As shown in Figure 7, all tasks created are then assigned to individuals within the team while considering their skill levels in development as not everyone within the team knew how to code.



*Figure 7: Jira Tasks*

Once the tasks have been assigned, I assisted each member of the team to create a new app accordingly, adhering the software architectural design to create the app in a proper structure. As shown in **Error! Reference source not found.**. Each of the application folder represents a component within the component diagram in Figure 4.



*Figure 8: IntelliJ App Structure*

By utilizing the software component diagram and data flow in Figure 5 and Figure 6, I created a template in the IDE for the team to follow in order to complete the sprint 1.