Alumni Database Final Report

Members: Sam Barnes, Shannon Bull, Max Camm, Ryan Clocker, and Alex Titus

Part I: Project Description and Database Design

Project Description:

Our group created a database to store and track information on Salisbury University's alumni. Currently, Salisbury University uses a database which is shared by all University systems of Maryland. This limits some of the SU administrators' freedoms and doesn't allow for some of the capabilities they may want to see. Our database addresses this concern and helps our clients keep track of where SU students progress further in their life past college. Additionally, this database can make our user's lives easier by allowing them to access the information they need faster than before. Due to the fact that the current database is shared among several universities, the process of searching for the information of a specific alumna is more challenging. By eliminating the other universities, we make our database more personal and configurable to Salisbury's needs.

The clients of this database are be broken into two groups: administrators and users. Administrators are those working in the alumni house that can add data and view all data. The typical users of this database will be those seeking to track or get in contact with alumni for networking. Currently, examples of typical users at Salisbury are those working in the foundation house, alumni researchers, and gift development officers. Each of these three groups attempt to acquire donations for the university. Additionally, some individuals in the alumni house use the database to pull information for various departments that are trying to contact their specific alumni. An example of this would be if the Math and Computer Science department was hoping to send a flyer about the new computer lab to all previous graduates, they would contact the alumni house administrators to do this.

As for technical concerns, our database is supported on a few platforms including Internet explorer and Chrome. This database will be available for Windows and Mac OS but will not be specifically configured for mobile devices.

Requirements:

Our database must support operations that will allow users to search for various alumni based on various attributes. In addition to this, it must allow administrators to add alumni to modify the database. It is important to remember that the database administrators are the only individuals that can actually add alumni to the database. That being said, they will need to have an administrator-specific login to gain these

permissions. Once logged in, administrators will be able to select from a variety of options to both view, add, or search alumni information. Typical users will not be able to perform the same actions so we will make sure that their login limits their access. When a typical user is logged in, they will be able to use various search features to find alumni based on some of their attributes (graduation year, last name, etc.).

Additionally, there are some assumptions we need to address. In regards to storage, we are not keeping enough records of each alumna for storage to be a concern. In the long term, storage may be an issue if the school grows significantly. At this point, administrators would need to determine how far back they would like to track alumni and whether the information of alumni will continue to be saved once they are deceased.

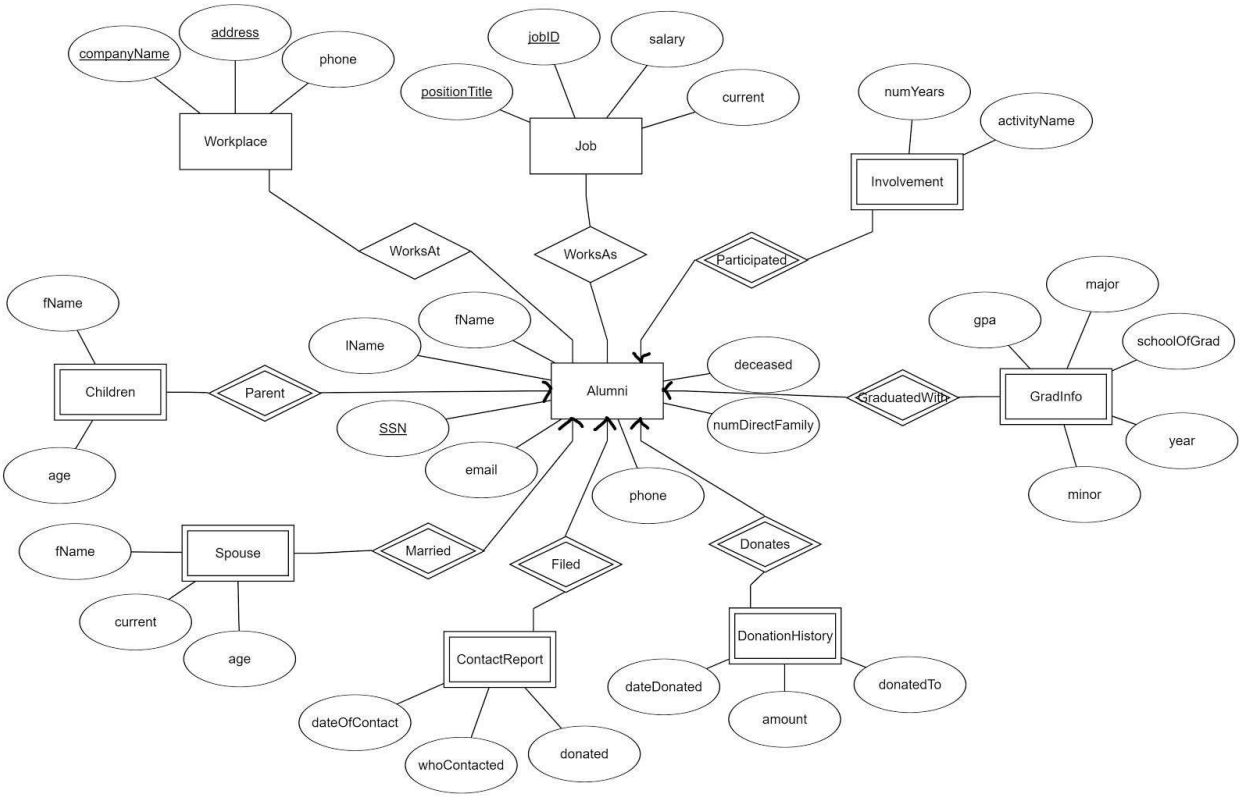
All of that being said, we have written up a brief overview of the attributes and relationships that our database will include.

- The database keeps track of **ALUMNI**, keeping their full name, phone number, email, a true false statement regarding whether they are deceased, the number of direct family members they have, and their unique *social security number*.
- Each of the saved **ALUMNI** will have **GRADUATION INFORMATION** which stores the year of graduation, major, minor, GPA, and school of graduation (Henson, Purdue, etc.)
- In addition, each of the **ALUMNI** will have a **WORKPLACE** that they work at. This will give information on their *company's name, address*, and phone number.
- Additionally, each of the **ALUMNI** will have a **JOB** which lists information specific to their position including their *jobID, positionTitle*, salary, and a true or false statement that provides if this is a current position.
- The database will also track **DONOR HISTORY** for each alumni which will record the alumni's name, the cumulative dollar amount donated, where the donation was dedicated (the football team, henson school, theatre, etc.), and the most recent date of donation
- Every **ALUMNI** may have many **SCHOOL INVOLVEMENTS** which tracks the name of the activity they were involved in and the number of years they were involved in that activity (for example SU track for 3 years).
- Each **ALUMNI** may have had multiple **SPOUSES.** For this entity, our database will track the spouse name and age. Each **SPOUSE** will also have a **WORKPLACE**.
- Additionally, each **ALUMNI** may have multiple **CHILDREN** which each have a name and age.
- Finally, each **ALUMNI** may have multiple **CONTACT REPORTS** where specific information regarding contact with the alumna is stored. This includes alumna name, the name of the person who was contacted, the date they were contacted,

a true false statement detailing if the contact resulted in a donation, and the unique *social security number* of the alumna that donated.

ER Diagram:

The requirements listed above were all taken into consideration to develop this ER diagram. This is a modified version of what was listed in our proposal and this is what was used to derive our relational schemas.



Initial Translated Database Schema:

Below, you will see the initial schemas which were derived from the ER diagram given above:

1. Alumni(SSN, fName, lName, email, phone, numDirectFamily, deceased)
2. Children(SSN, fName, age, foreign key (SSN) references Alumni (SSN))
3. Spouse(SSN, fName, age, current, foreign key (SSN) references Alumni (SSN))
4. ContactReport(SSN, dateOfContact, whoContacted, donated, foreign key (SSN) references Alumni (SSN))

5. DonationHistory(SSN, dateDonated, amount, donatedTo, foreign key (SSN) references Alumni (SSN))
6. GradInfo(SSN, gpa, major, minor, year, schoolOfGrad, foreign key (SSN) references Alumni (SSN))
7. Involvement(SSN, numYears, activityName, foreign key (SSN) references Alumni (SSN))
8. Job(positionTitle, jobID, salary, current)
9. WorksAs(positionTitle, jobID, SSN, foreign keys (positionTitle, jobID) references Job (positionTitle, jobID), foreign key (SSN) references Alumni (SSN))
10. Workplace(companyName, address, phone)
11. WorksAt(companyName, address, SSN, foreign keys (companyName, address) references Workplace (comanyName, address), foreign key (SSN) references Alumni (SSN))
12. Users(username, password)
    a. Note that this table is not related to any others in the diagram, but it is what we will use to track who is logging in and the privileges associated with their account (admin vs. standard user)

Normalized Database Schema:

After going through the process of translating our ER diagram to the relational schema for our database, we need to check the schema to ensure that they are normalized. First, we go through each schema and list all functional dependencies.

1. Alumni FDs: {SSN → all attributes; email → fName, lName}
2. Children FDs: {SSN → all attributes}
3. Spouse FDs: {SSN → all attributes}
4. ContactReport FDs: {SSN → all attributes}
5. DonationHistory FDs: {SSN → all attributes}
6. GradInfo FDs: {SSN → all attributes}
7. Involvement FDs: {SSN → all attributes}
8. Job FDs: {positionTitle, jobID → all attributes}
9. WorksAs FDs: {positionTitle, jobID, SSN → positionTitle, jobID, SSN; SSN → all}
10. Workplace FDs: {companyName, address → all, phone → companyName}
11. WorksAt FDs: {companyName, address, SSN → companyName, address, SSN; SSN → all}
12. Users FDs: {username → all attributes}

Now, we need to make sure that each schema is in 3rd normal form. To do this, we have to look at the list of functional dependencies and ensure that for each nontrivial

X → Y that X is either a superkey for the relation or each member of Y is an attribute of a key.

The Alumni schema violates 3NF: email is not a superkey and lName, fName are not attributes of a key. Below we have the decomposition:

a. Alumni(SSN, email, phone, numDirectFamily, deceased)
b. Alumni1(email, fName, lName)

It is clear to see that schemas 2 - 8 and 12 do not violate Third Normal Form because the only functional dependencies they have are that their key can functionally determine all other attributes.

We needed to be careful with the Workplace schema because you see that in addition to the key determining all attributes we see that the phone determines the companyName. However, this is still not a violation because even though phone is not a superkey, we know that companyName is an attribute of the key. This means this schema still satisfies Third Normal Form.

The only other schemas we needed to be weary of were the WorksAt and WorksAs schemas because there is more than one functional dependency. However, in both cases, we see that Third Normal Form is not violated. The extra functional dependency that "SSN → all" does not cause a violation because all attributes of this key make up the key and this satisfies the statement that "each member of Y is an attribute of a key".

With these modifications, we were able to create all of our tables and begin implementing and testing our database. The next section will discuss how this was performed.

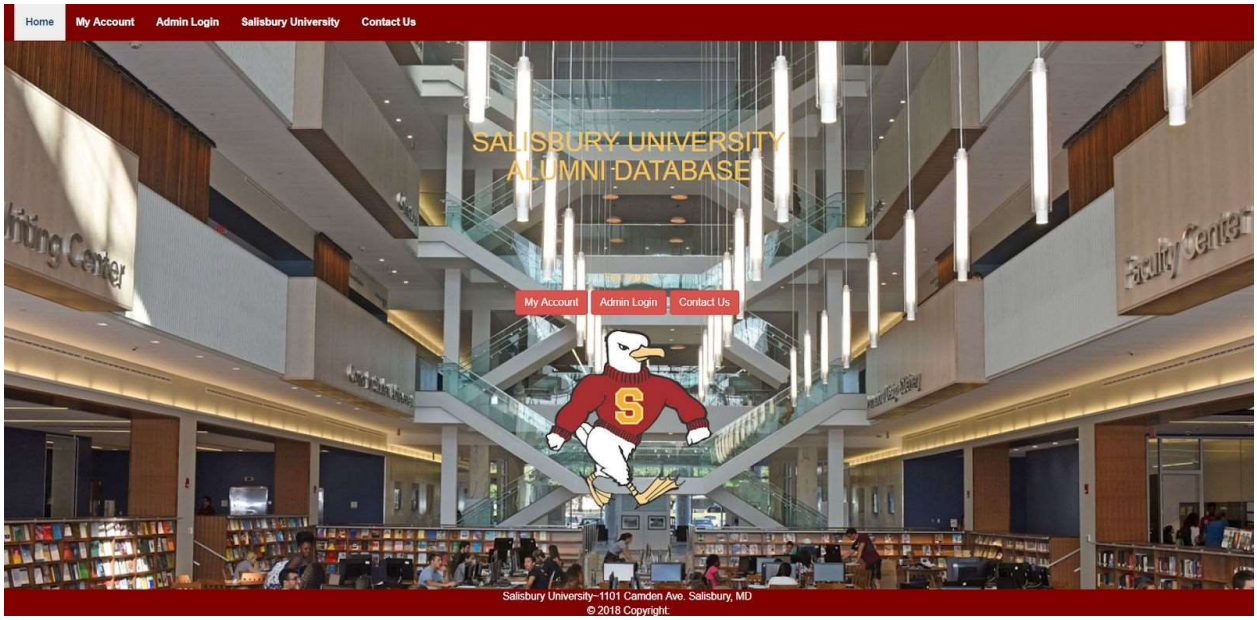
Part II: Implementation and Testing Details

Link to our web implementation: http://acadweb1.salisbury.edu/~sbull1/

Below, we have highlighted some of the functionalities we have implemented:

- Mandatory login screen to ensure that the user gains proper access
- Extensive search capabilities for standard user
- Capability to display sensitive information if the current user is an administrator
- Administrators are able to add or modify information in the database
- Link to Salisbury University's current homepage
- Search capabilities tailored to administrator's specific needs
- Compatible with multiple web browsers
- Mobile friendly
- Opportunities for database users to contact the alumni house

Much of our time was dedicated to implementing our database. Below, you will see the homepage which our user sees when they click the link above. We will break down the description of our components by discussing each of the tabs:



1. Home page: this page is used to access all other pieces of the database and is what a user will see when they first access our website.
2. My Account: this tab is where the standard user should go to login. They will need to enter a valid username and password. If they enter valid data, they will be redirected to a page that will allow them to choose if they want to search for a specific alumna or search for a group of alumni.
    a. The “search by individual” option will require them to enter a piece of information (first name, last name, email, or phone number) on the alumna they are searching for. It will return other information to help them get in contact with this alumna.
    b. The “search by group” option will allow them to search for groups of alumni at a time. The can look for alumni that graduated from a specific school (Henson, Perdue, etc.), had a specific major, or graduated a certain year.

    The standard user will only be allowed to view certain data and they will not be able to make any changes to the database.
3. Admin Login: this tab is where database administrators can go to view and modify the data. In order to gain these capabilities, they need to have proper admin privileges as designated by their login information. Once logged in, they will have the option to “search by individual”, “search by group”, or “add alumni data”. Their search pages differ from the typical user’s search pages because they will let them view sensitive alumni information such as their donation history.

4. Salisbury University: this tab will redirect the user to Salisbury University's homepage.
5. Contact Us: this tab sends the user to a page which has contact information for Salisbury University's alumni house. It includes a map and several pieces of contact information that the user may choose to take advantage of.

Throughout this process, we used a variety of software to support our development. An example of this is that we used bootstrap to format our data Additionally, our group worked in linux, mac, and windows operating systems. When it came to testing our application for correctness, we looked for fresh pairs of eyes who were independent of our development. With help from some volunteers, we were able to catch some errors in our formatting and grammar. To test the performance of our database we ran a series of tests that required our implementation to execute multiple queries to access information from our database.

Part III: Summary

Discuss what you have learned from the project, any team collaboration issues, including group meetings and member participation records and contribution details by each member.

Over the course of this project we learned a great deal of web development and how it can be useful in the implementation of databases. We were able to work together and learn from each other to achieve our final implementation. Our group met several times in person to work on this project. There are details about each of the meetings below, but please note that a significant part of the work was completed individually with communication through email or text. Not everyone needed to attend every meeting.

1. Discuss the database and begin writing the proposal. Split up responsibilities.
    a. Attendees: Sam, Shannon, Alex, Ryan
2. Finish the ER diagram and go over the last few details of the proposal
    a. Attendees: Shannon, Ryan, Alex
3. Work on a presentation for PHP, Javascript, HTML, and CSS. Divide up responsibilities.
    a. Attendees: Alex, Shannon, Ryan, Sam
4. Modify the presentation and divide up responsibilities for the PHP section.
    a. Attendees: Max, Ryan, Shannon
5. Meet to discuss functional dependencies
    a. Attendees: Max, Ryan, Shannon, Alex
6. Work on the database implementation (login and search)
    a. Attendees: Max, Ryan, Sam, Shannon
7. Work on the database implementation

    a. Attendees: Max, Ryan, Sam, Shannon, Alex
8. Work on the database implementation
    a. Attendees: Alex, Ryan, Shannon, Sam
9. Work on the database implementation
    a. Attendees: Ryan, Shannon, Sam
10. Database implementation
    a. Attendees: Ryan, Sam, Max, Shannon, Alex
11. Prepare for the presentation
    a. Attendees: Alex, Ryan, Sam, Max, Shannon

Below is a more detailed outline of each member's major contributions to this project.

- Sam was crucial in the implementation of our database. He has experience in web development and we were all able to learn from him. Over the course of this project, he used html, javascript, jQuery, and CSS to design our website. He was able to link all pages of our website and was instrumental in scripting the timeouts that allowed our login portals to successfully navigate to the appropriate search pages. He was our group's timeout engineer and aesthetic coordinator.
- Shannon initially met with Salisbury University's current alumni database administrator, Sandy Griswold. This helped the group understand the functionality of the current database as well as which new features need be implemented to improve user experience. She wrote the final database report and worked with the group on several sections. She successfully redesigned the ER diagram for better logic following feedback that was received on the proposal. After this, she worked with Ryan and Max to identify the functional dependencies in each schema which she then used to normalize all schemas in the database so that they all abide by third normal form. Following this, she made the tables in mySQL which would be used throughout the implementation phase of this project. She determined appropriate user interfaces for each privilege type and wrote the mySQL queries which were coded in php to access the database. She also wrote the php file used to "search by group" with Ryan. She worked with the group to test and analyze the website.
- Max contributed to several aspects of this project. In earlier stages, he worked with members to determine the functional dependencies for each schema. Following this, he worked with the group on the implementation of the website. During this time, he wrote the php file that allows administrators to search the database. This required an advanced understanding of php coding and mysql statement. He also assisted in testing and analyzing the final product.
- Ryan was a key team member. He was available from the start of the project and successfully created our group's initial ER diagram. After this point, he identified

key functional dependencies that were vital to helping normalize our database schemas. He has significantly researched how php can be used in our implementation and led our group in generating two login and search pages for the users. He also wrote the php code to connect the user's input in the search pages to our database that also queries the database and outputs the result. Following this, he helped teach other group members the proper way to link php and mysql and was crucial in error checking and formatting other members' codes. His work speaks for itself and he was our group's php master.

- Alex generated fake data for our database. He was able to use his knowledge of python to write a script which would come up with realistic data for all of our tables. This required research and time. His data was personalized to Salisbury University. Data generated for topics like involvement came from Salisbury's specific list of clubs and organizations. He also assisted in the final formatting of our webpage.