



Database Group

COMP 730/830, Spring 2017

Professor Jonas

LEADERSHIP ROLES

Database Managing Architect:

[REDACTED]

Database Quality Assurance Director:

[REDACTED]

Database Communications Director:

Patrick McElhiney

DEVELOPMENT ROLES

Graduate Database Engineer, Level 1:

[REDACTED]

Graduate Database Engineer, Level 2:

Patrick McElhiney

Graduate Database Engineer, Level 3:

[REDACTED]

OOP Database Specialist:

[REDACTED]

Data Fabrication Specialist:

[REDACTED]

CONTACT INFO

[REDACTED]

[REDACTED]

Patrick R. McElhiney

(603) 742-5112

prr22@wildcats.unh.edu

[REDACTED]

[REDACTED]

To Do List

Key:	Needs to be Assigned	Not Started	Unfinished	On-Going	Completed
-------------	----------------------	-------------	------------	----------	-----------

Task Description:	Assigned To:	Due Date:
Revise UML Diagrams	Patrick	03/29/2017
Revise Crow's Feet Diagram	Patrick	03/29/2017
Go Through Last Year's Class's Files from Server	Everyone	03/29/2017
<ul style="list-style-type: none"> • Examine SQL files, determine if we're going to just use the same structure, or go with Software Development's new structure 		
Produce SQL Tables for MySQL Database	Patrick McElhiney	04/05/2017
<ul style="list-style-type: none"> • Per planning by Database/Software • These need to be produced like the last class did, including Field Name, Data Type, and Length and/or Pattern (for instance for dates). 		
Get lamp.unh.edu Server Running Correctly	Patrick	04/05/2017
Activity Diagram uploaded to wiki		04/06/2017
Update "Simple Use Case Diagram"	Patrick McElhiney	04/07/2017
Generate Table Data for Database Based on Data Model	Patrick McElhiney	04/07/2017
Generate SQL Code for Database Incl. Table Data	Patrick McElhiney	04/07/2017
Update "Registration Diagram"		04/07/2017
<ul style="list-style-type: none"> • Upload to Wiki 		04/12/2017
Configure Access Control for MySQL on Server	Patrick McElhiney	04/08/2017
<ul style="list-style-type: none"> • phpMyAdmin Account Access, "root" • DB: MEANDYOU Access, "meandyou" • DB: MEANDYOU2 Access, "meandyou2" 		
Complete URC Slide and E-mail to Professor		04/10/2017
Complete GRC Slide and E-mail to Professor	Grad Students	04/10/2017
Create Lists of QA Questions		04/12/2017
Master Attribute List on Wiki		04/07/2017
Create 4 Additional Sequence Diagrams		04/12/2017
<ul style="list-style-type: none"> • Upload them to Wiki 		04/12/2017
Resolve Any Database Issues with Software and Front-end, such as "0" index issue	Patrick McElhiney	04/12/2017
Coordinate with Software Development RE: Data Model	Patrick	04/12/2017

See Next Page for more Tasks...

To Do List

Key:	Needs to be Assigned	Not Started	Unfinished	On-Going	Completed
-------------	----------------------	-------------	------------	----------	-----------

Task Description:	Assigned To:	Due Date:
Clean up the Documentation (on-going)	[Redacted]	05/02/2017
<ul style="list-style-type: none"> • Master Documents • Wiki Pages 	Patrick McElhiney Everyone	(on-going) (on-going)
Update Job Descriptions / Tasks List	[Redacted]	04/11/2017
Generate PHP Calls for MySQL For Front-End (GUI)	[Redacted]	04/11/2017
Keep All Work Progress Documented in Gantt Chart	Patrick McElhiney	05/02/2017
<ul style="list-style-type: none"> • Microsoft Project 		(on-going)
Produce SQL SELECT, INSERT, UPDATE commands for Front-End based on new Data Model	Patrick McElhiney	04/12/2017
Get Configuration Management System Populated	[Redacted] Patrick	04/12/2017
<ul style="list-style-type: none"> • Import all versions of SQL Files, UML Diagrams, Master Documents, and other Deliverables into SVN Server on lamp.unh.edu 		
Produce Sample Data for 20,000 users	[Redacted]	04/12/2017
<ul style="list-style-type: none"> • 20,000 Random Named User Accounts <ul style="list-style-type: none"> ○ Attributes are in attributes table • 10,000 Fully Functional Searches <ul style="list-style-type: none"> ○ Searches are coded to actual users 		
Project Implementation Document	[Redacted] Patrick	04/19/2017
Functional Requirements	[Redacted] Patrick	04/19/2017
<ul style="list-style-type: none"> • User Account Requirements • Requirements for the Database 		
Test Database/Software for QA Issues	[Redacted]	04/19/2017
<ul style="list-style-type: none"> • Import data from Excel and create SQL files • Dump data into database • Coordinate Testing with Software Development <ul style="list-style-type: none"> ○ Coordinate with Mac for testing Engine 	Patrick McElhiney [Redacted]	04/17/2017 04/19/2017 04/19/2017
Fix Issues with Database and Prepare for Prototype	[Redacted] Patrick	04/19/2017
<ul style="list-style-type: none"> • Ensure that all the bugs are worked out • Fix issues with incompatibilities between Software & Database, Front-end & Database 	Patrick McElhiney [Redacted] Patrick	04/19/2017 04/26/2017
Wiki step by step screenshots for Demo	[Redacted]	04/26/2017
Create Requirements Analysis Document (RAD)	[Redacted]	04/26/2017
Create System Design Document (SDD)	[Redacted]	04/26/2017
Package up Files for Next Year's Class	Patrick [Redacted]	05/02/2017

How Weekly Assignments Work:

If your name is not listed to finish a specific task during any week, your job is to assist everyone who is leading the project operations during that week per the schedule.

- Engineers:** Always stay in contact with the Architect. If you don't perform well in any given week, it will be reflected in your assessment of the Team. We will not be the ones that grade you down because of failures – Professor Jonas will be able to tell from the Graduate Student logs who is working, and who isn't. And if we need to refer to the Gantt Chart or this document to determine what you have been doing, you're probably not in good shape.
- Each group member should take responsibility for the entire project.
 - If any part of our project responsibilities is failing – it is ultimately the responsibility of the other Engineers to pick up the pieces and take over. This is a team effort.
 - Make sure you send all completed and incomplete materials by the due date to the Architect ([REDACTED]), Quality Assurance ([REDACTED]) and Communications (Patrick McElhiney). If you have been assigned a multi-week project, be sure to provide updates and evidence of your work to us.
 - If you need help, please ask a Graduate Student.
 - All students are expected to spend at least 10 hours a week working with the team on the project.
- Architect:** Always stay in contact with the person who is responsible for delivering the finished item(s) by the specified due date(s), and enforce the deadlines. Your job is to be the manager for the group, which includes assigning tasks, and all of the other responsibilities listed in this document.
- QA:** Quality Assurance should develop questions and test parameters on a weekly basis, including testing all deliverables from other members, and answer the QA questions to determine if the quality of the workmanship is up to par.
- Communications:** Keep all members on the same page, not just with what our group is doing, but what all the groups are doing.
- Grad Student:** If you are a Graduate Student, you have additional responsibilities as defined in the Job Descriptions, and per any other agreed to schedule or timeline per the group. You also must assist any Undergraduate Student that needs help.
- Weekly logs in the Wiki

EXPECTATIONS OF TEAM MEMBERS

Per the Class [Syllabus](#),

All Team Members are Expected To:

- Assume Developer Roles.
- Write the User Requirements of a Real-World Team Project, **Intensively**, through:
 - **C**apture,
 - **A**nalyze,
 - **R**efine,
 - **D**ocument.
- Participate in Weekly Development Activities, by Working in Teams to Build Models of a Real-World System, Including:
 1. Requirements Elicitation and Analysis,
 2. System and Object Design,
 3. Implementation and Testing,
 4. Project and Configuration Management,
 5. Infrastructure Maintenance,
 6. System Deployment to the End User.
- Deliver a Proof-of-Concept, or Prototype of Me&You.
- Carry on the Architectural Task of System Analysis.
- Plan for, Manage, and Mitigate Risk Factors the Team Might Encounter.
- Improve Personal and Interpersonal Communication through Interaction with Team Members and a Real Client.

JOB DESCRIPTIONS

DATABASE MANAGING ARCHITECT

- Responsible for all database designing / database engineering tasks.
 - Develops the overall database design / database structure, and disseminates information about it to task other group members to develop the IP assets and configurations that will be needed for the Database.
 - Develops Use Cases and Use Case Diagrams to lead the Database Group in the development of the database for Me&You.
 - Develops block diagrams of the Database.
- Manages the Database Group on a weekly basis, assigning tasks to Graduate and Undergraduate students.
 - Manages / Updates the Job Descriptions for the entire Database Group, assigning additional work / sub-blocks as needed based on the overall needs of the Database Group.
 - Facilitates the development of the core infrastructure with the Software Group and Front-end (GUI) Group, based on the architecture of the Database design.
- In charge of packaging up everything for Database Group for the next Semester.
- Maintain communication with the client (*Professor Jonas*).
- Keeps Graduate Logs Starting on March 1st, 2017.
- Coordinates the Quality Assurance process with the Database Quality Assurance Director.
- Communicates with other Architects regarding the Project Scope and Configurations of the overall Project Database Core.
- Coordinates communications with the Database Communications Director, and with other groups per and through the Database Communications Director.
 - Communicates the Database Design and Engineering Scheme to Front-end (GUI) and Software Development Groups
- Acts as Graduate Database Engineer Level 3, in support of the Database Architect's goals and tasks on a weekly basis.
 - Sets up Server for Database, and imports Database to Server.

JOB DESCRIPTIONS (continued...)

DATABASE QUALITY ASSURANCE (QA) DIRECTOR

(QA, REQUIREMENTS ELICITATION (RE), CONFIGURATION MANAGEMENT (CM))

- Responsible for all Database Group QA processes
 - Ensures **Software Reliability** through **Debugging Faults Detected**
 - Plans Testing of System and Subsystems
 - **Usability Testing, Unit Testing, Integration Testing, Structural Testing, and System Testing**
 - Coordinates with other QA Directors from Front-end (GUI) and Software Development, to streamline the process of developing QA procedures for all three groups including Database group with input from the other QA Directors.
- Develops **Quality Assurance Questions** for the Database Group to determine if **System is Correct, Complete, Consistent, and Unambiguous**
 - In charge of all Requirements Elicitation processes
 - Refines lists of Attributes based on input from the Front-end (GUI) and Software Development Groups, as well as the Database Group.
- Keeps Graduate Logs Starting on March 1st, 2017.
- Acts as Graduate Database Engineer Level 1, in support of the Database Architect's goals and tasks on a weekly basis.

CONFIGURATION MANAGEMENT

- Maintains systems associated with Implementation and Source Code Control using Redmine or similar CM program in coordination with the Architects of all three groups, including:
 - **versioning**
 - **feature request fulfillment**
 - **change requests, and**
 - **bug tracking**

Release Management

- Ensures proper documentation
 - **In Source Code**
 - *Commenting what specific code does*
 - *Commenting for the API (**Application Program Interface**)*
 - **In User Manual** (which you need to create)
 - **In Developer's Manual**
 - *You need to review it with the specific developers who worked on the specific parts of the project for consistency and detail orientation, and if something doesn't meet the requirements of the system, it must be changed*
- Ensures QA with the Source Code, i.e. establishing baselines, versioning
- Double checks the Source Code Control processes and procedures
- In charge of the Maintenance of all Documentation from the Database Group, working with the Communications Director to disseminate the information to other groups on a timely basis.
 - Ensure Master Document for Database is consistent with Front-end (GUI) and Software Development Master Documents

JOB DESCRIPTIONS (continued...)

DATABASE COMMUNICATIONS DIRECTOR

- Coordinate all communications between Graduate Students with weekly Zoom meetings (*Tuesdays 4:00PM – 6:00PM*), before class meetings (*Wednesdays 3:30PM – 5:30PM*), and after class meetings (*Wednesdays 8:30PM – 9:30PM*)
 - Between Front-end (GUI) Group and Database Group
 - Between Software Development Group and Database Group
 - Between Front-end (GUI) Group and Software Development Group
- Maintain communication with the client (*Professor Jonas*) and communicate and coordinate all contributions to the project, overall.
 - *“Improve personal and interpersonal communication through interaction with team members and a real client.”*
- Ensures that all groups are on the same page, per the communication requirements between different groups – including between Architects, between QA Directors, and between Engineers.
- Coordinate all communications between Graduate Students and Undergraduate Students in the Database Group (*Starting in March 2017 – May 2017*)
- Develops and Maintains Timeline Plan, based on the Deliverables Identified by the Database Architect, and the Database Quality Assurance Processes as Identified by the Database Quality Assurance Director
 - *“Plan for, manage, and mitigate risk factors the team might encounter.”* in terms of the planning of deliverables, and the timing of the project, by managing the group on a ramping-up basis from week to week using communications and Microsoft Project management tool with Gantt Chart
 - Keep track of how many hours each student has spent towards the project in Microsoft Project, or otherwise in some type of table or chart such as Excel
- Updates the Wiki page on a weekly basis.
- Keeps Graduate Logs Starting on March 1st, 2017.
 - Develops outline form for Graduate Student Logs, and disseminates this to every Graduate Student by March 1st, 2017.
- Acts as Graduate Database Engineer Level 2, in support of the Database Architect’s goals and tasks on a weekly basis.
 - Develops an Outline Form for the Database Group Documentation, and disseminates this document to the Database Group, as well as the other groups as an example for them to copy and use with their own information.

JOB DESCRIPTIONS (continued...)

OBJECT ORIENTED PROGRAMMING (OOP) DATABASE SPECIALIST

(MYSQL)

- Programs the various constraints and properties of the various types of databases, whichever are decided to be used in the implementation of the OOP system.
- Develops and refines Use Cases and Use Case Diagrams with the Database Architect.
- In charge of all database programming aspects of the project.
- Develops and Refines Use Case Diagrams, Class Diagrams & their Object (Instance) Diagrams, Scenario Diagrams, Sequence Diagrams, Activity Diagrams
- Develops and Refines Documentation for Communications Director

DATA FABRICATION SPECIALIST

(MYSQL, EXCEL, .CSV EXPORTS)

- Creates sample data to import into the database through the creation of Excel spreadsheets, exported to .csv files, that are used to populate the database structures, using scenarios for each of the possible combinations of different types of relationships, data profiles, user profiles, sample policies, sample attributes, sample properties, etc.
- Tests the functionality of the database by populating queries based on different characteristics identified by scenarios, to verify that the database is correctly attributed and populated, and it works correctly.
- Populates the database with entries based on the scenarios that are based on different possibilities of the attributes defined by the Software Development group and the Database group, based on the Database Structure developed by the Database Architect.
- Updates the database based on new data, and tests Front-end (GUI)'s entry forms for accuracy.
- In charge of testing the database for speed and making improvements to the structure to speed up operations.

TEAM RESOURCES

- Experience
 - [REDACTED] (Architect) works as a support technician. He took COMP 820, the Graduate Databases class last semester.
 - [REDACTED] (Quality Assurance) works as a systems administrator. She has experience with MySQL.
 - Patrick McElhiney runs his own Marketing Firm, and has experience with MySQL – mostly based on Linux deployments. He has a Linux shell that is available if the Database Group needs a Linux box to test out anything.
- Schedules (*When are people available?*)
 - [REDACTED]
 - **Monday to Friday** – After 4PM.
 - **Saturday** – All Day
 - [REDACTED]
 - **Tuesday** – 2PM to 4PM
 - **Wednesday to Friday** – 8AM to 4PM.
 - **Saturday** – 9AM to 12PM.
 - Patrick R. McElhiney
 - **All Days, All Times** – Until Further Notice
 - [REDACTED]
 - **Need availability schedule.**
 - [REDACTED]
 - **Virtually All Days, All Times**

TEAM RESOURCES (continued...)

- Computer Programs
 - [REDACTED] has **MySQL 5.7.17** loaded on **Ubuntu**, running on **Virtual Box** on his Mac Book.
 - Has **MySQL 5.7.17** loaded on **Windows** running on a **VM** on his Mac Book.
 - Has **Microsoft Visio Professional**
 - [REDACTED] has a Desktop computer at home, and may need to borrow a laptop from [REDACTED] for this project.
 - Has **MySQL** running locally, and has the **Demo** running on that system.
 - Patrick McElhiney has:
 - **Microsoft Project** (for Gantt Charting)
 - **Microsoft Visio Professional** (for UML / Crow's Feet Diagrams)
 - cPanel-based VPS Linux Box, Dedicated IP, MySQL 5.6.35 with Apache 5.6.35, PHP 5.10.1, and phpMyAdmin to access the MySQL Databases through a web-based interface. If it is needed for testing purposes, let him know.
 - [REDACTED] Lucid Chart
 - [REDACTED] Has Adobe Creative Cloud
 - Uses Data Generator (www.datagenerator.com)
 - [Notepad++](#) may be needed.
 - [Eclipse Neon](#) is a good tool for programming for Java, but also for C, C++, COBOL, D, Fortran, Haskell, JavaScript, Julia, Lasso, Lua, NATURAL, Perl, PHP, Prolog, Python, R, Ruby, Rust, Scala, Cloujure, Groovy, Scheme, and Erlang.
- Team Collaboration Resources
 - Slack – OOSE Slack, setup by [REDACTED] – he is Administrator.
 - [Download Slack](#) for Desktop
 - oose-team.slack.com
 - Zoom
 - zoom.unh.edu
 - Patrick McElhiney's Meeting ID: 308-196-7750
 - Join from PC, Mac, Linux, iOS or Android: <https://unh.zoom.us/j/3081967750>
 - Office 365 - For all Office 365 Apps, Including SharePoint, Skype, Etc.
 - <http://wildcats.unh.edu>
 - Me&You Project Wiki
 - <https://foss.unh.edu/projects/index.php/comp730:MeAndYou>

TEAM RESOURCES (continued...)

- Other Useful Websites

- **Redmine** – William Jones (Software Architect) is setting this up for Project Management, Configuration Management, Feature Requests, and Bug Tracking.
 - <http://www.redmine.org/projects/redmine/wiki/Download>
- **Object Management Group** - see this site for UML Specifications, and other types of Diagram specifications.
 - <http://www.omg.org/spec/>
- **COMP 730/830 Course Website** – Professor Michael Jonas
 - <https://pubpages.unh.edu/~mcy59/comp/730/>

- Books

Patrick M. has several books loaned out to students in COMP 830/730. There are currently no books loaned out to anyone in the Database Group. If you decide to borrow any of the below books, please ensure that they are returned when they are no longer needed for the project. If additional books are needed, you can check them out at the Library, or ask Patrick M.

Book Name:

- PHP / MySQL
- Java / MySQL
- Database Systems
- Domain-Driven Design
- Relational Database Design and Implementation
- MySQL / PHP (New Book 4/6/2017)
- MySQL (New Book 4/6/2017)
- PHP Cookbook / Recipes (New Book 4/6/2017)

Availability:

William Rivera has it.
Available from Patrick M.
Available from Patrick M.
Available from Patrick M.
Available from Patrick M.
Available from Patrick M.
Available from Patrick M.
Available from Patrick M.

MEETINGS SCHEDULE

All Groups, All Members

→ Unless Specifically Excused Due to Schedule Conflict – Must Notify Patrick McElhiney in Advance

Weekly Zoom Meetings:

Tuesdays 4:00PM – 6:00PM

Graduate Students – Starting 2/21/2017 through 5/14/2017

⇒ All Graduate Students Also Meet on 3/14/2017 During Spring Break

Undergraduate Students – Starting 3/21/2017 through 5/14/2017

- [zoom.unh.edu](https://zoom.us)
- Patrick McElhiney's Meeting ID: 308-196-7750
- Join from PC, Mac, Linux, iOS or Android: <https://unh.zoom.us/j/3081967750>

Software Development

Before Class Meetings:

Wednesdays In-Person 3:30PM – 5:30PM

Graduate Students – Starting 2/22/2017 through 5/10/2017

⇒ We Will Not Meet in Person on 3/15/2017 During Spring Break

Undergraduate Students – Starting 3/22/2017 through 5/10/2017

Front-end (GUI) & Other Groups as Needed

After Class Meetings:

Wednesdays In-Person 8:30PM – 9:00PM

Graduate Students – Starting 2/22/2017 through 5/10/2017

⇒ We Will Not Meet in Person on 3/15/2017 During Spring Break

Undergraduate Students – Starting 3/22/2017 through 5/10/2017

Final Stretch

Last Week TBD Meetings:

May 11th through May 15th (Please Request Vacation Time in Advance if Possible)

- ⇒ All Groups, All Members – Meetings TBD
- ⇒ Separate Groups, All Members – Meetings TBD
- ⇒ Separate Groups, Graduate Students Only – Meetings TBD
- ⇒ All Graduate Students, All Groups – Meetings TBD
- ⇒ Communications Director, Select Students (*for Makeup Work*) – Meetings TBD
- ⇒ Architects, Professor Jonas – Meetings TBD

Database Architecture (Week 1)

Relational Model

- Rows and Columns
- Keys (Primary and Foreign)
- SQL Statements

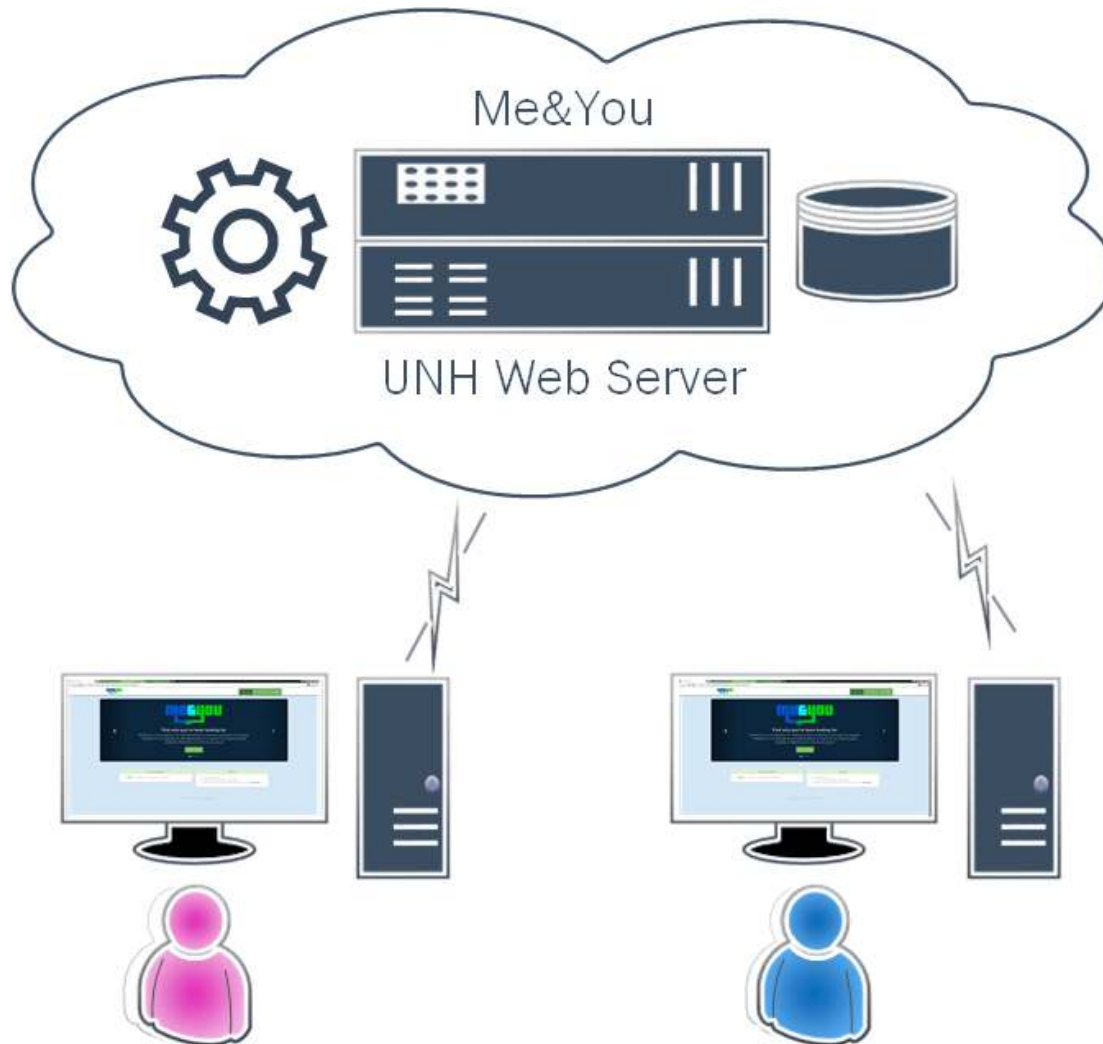
MySQL (Open Source)

- Reliable Relational DBMS
- Redundancy (mysqldump)

Disk Layer (Storage Model)

- SATA 3 6.0Gb/s SSD
- RAID 5 Array Redundancy

Database Architecture (Week 1)



Crush Search:



1. FirstName
2. LastName
3. Email
4. PhoneNumber
5. Hometown

Database Architecture (Week 1)

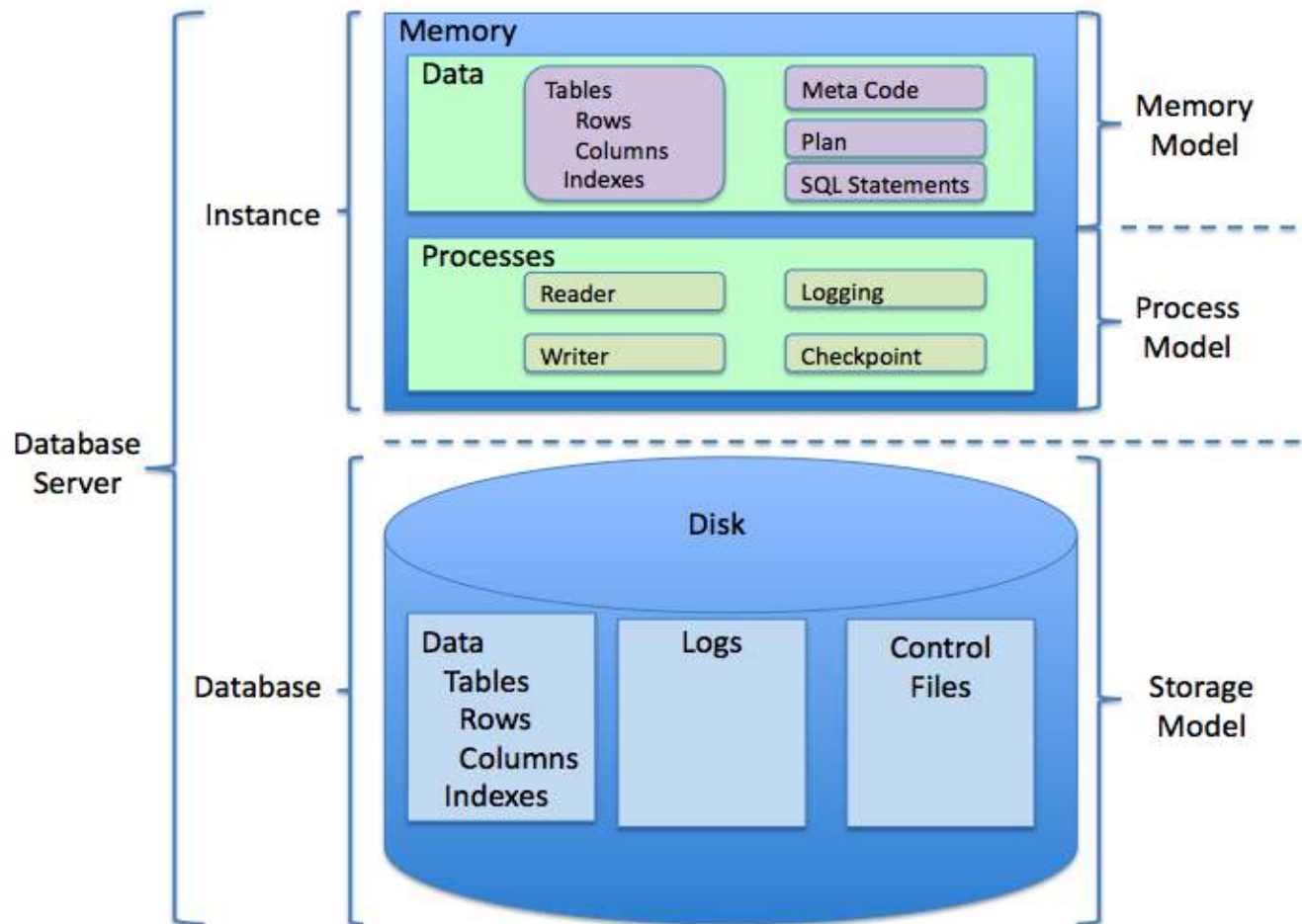
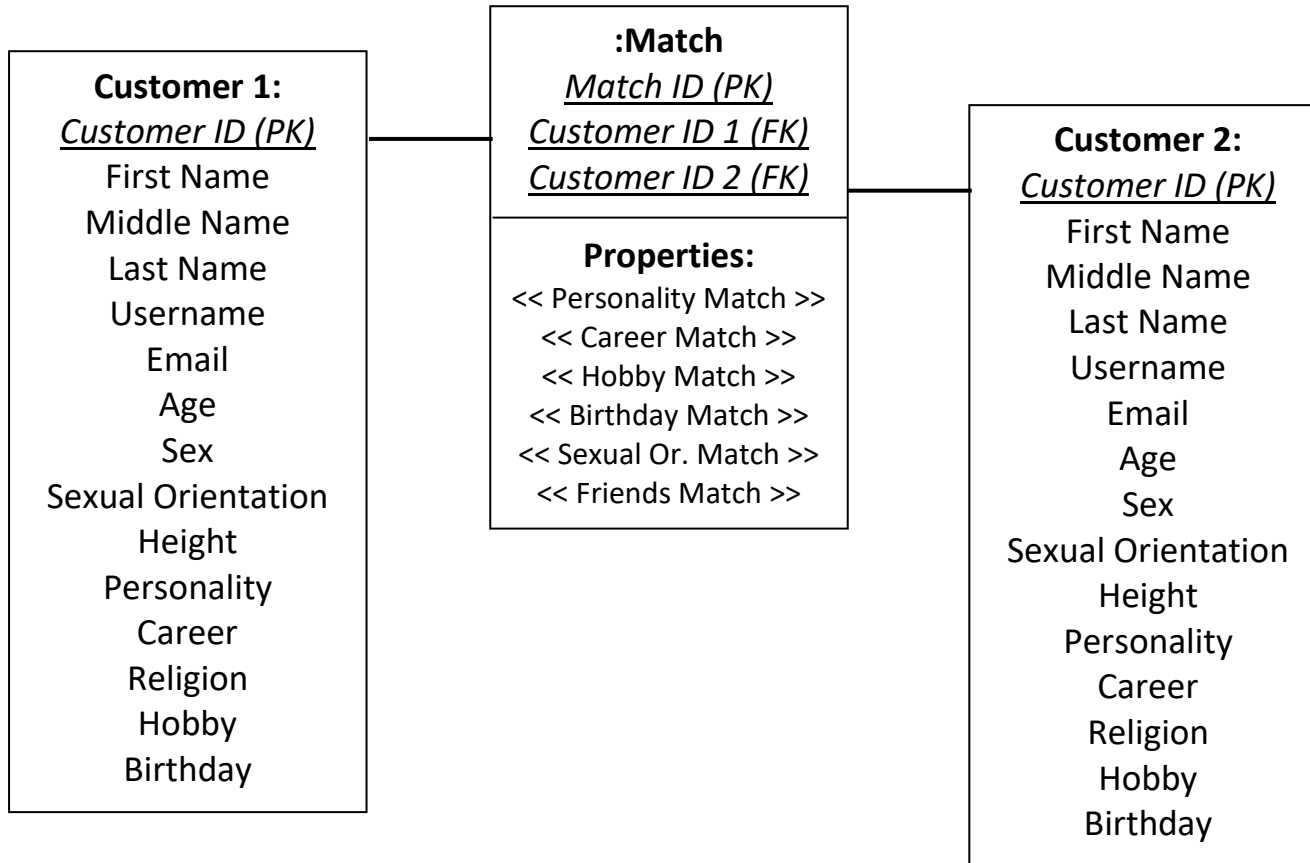


Image Source: Wikipedia, Retrieved on 03/18/2017 from (https://en.wikipedia.org/wiki/Relational_database_management_system#/media/File:RDBMS_structure.png)

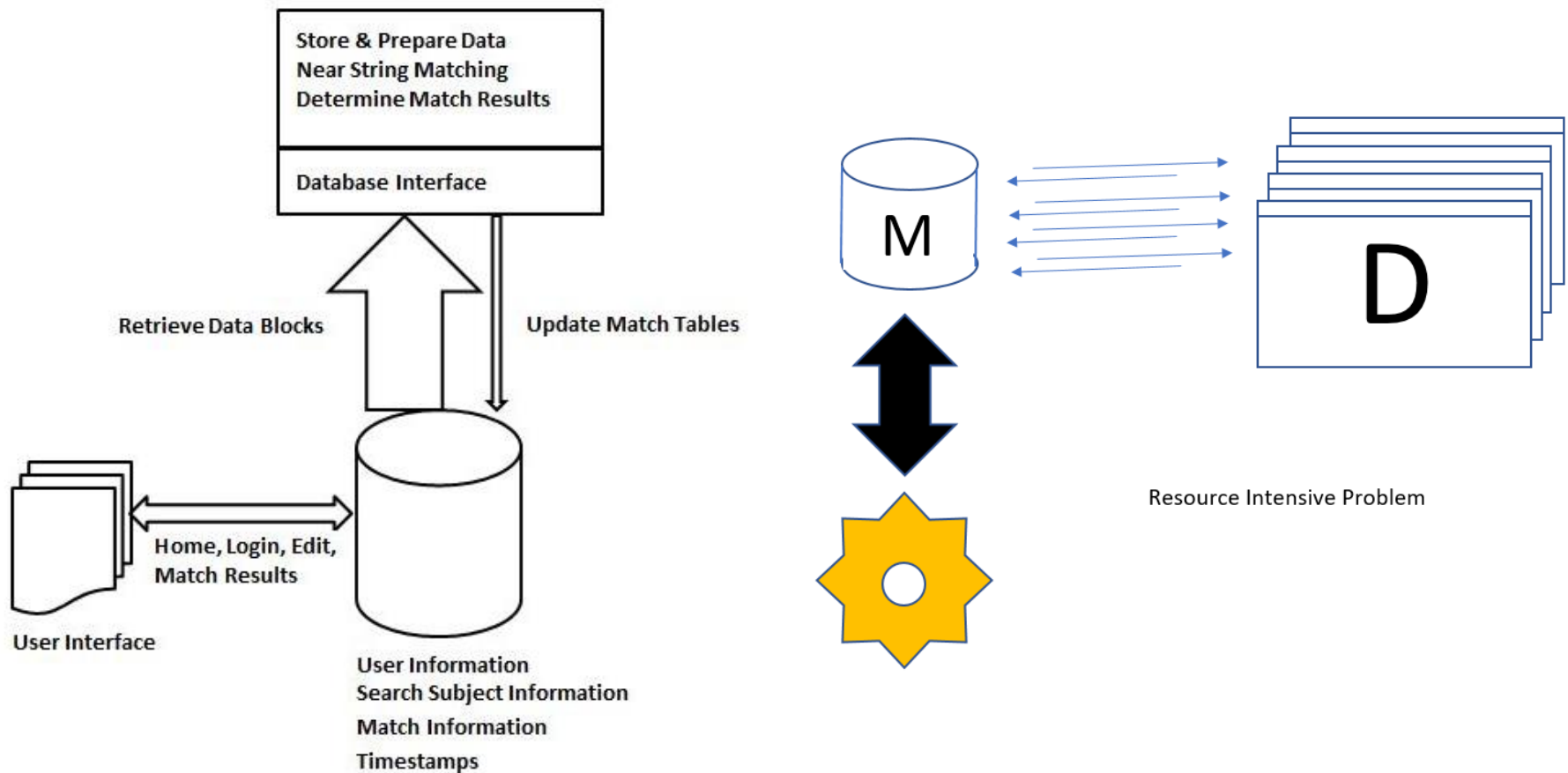
Reference: Wikipedia, Relational database management system. https://en.wikipedia.org/wiki/Relational_database_management_system

Database Architecture (Week 1)



Database Architecture (Week 1)

MeAndYou Top Level Architecture:

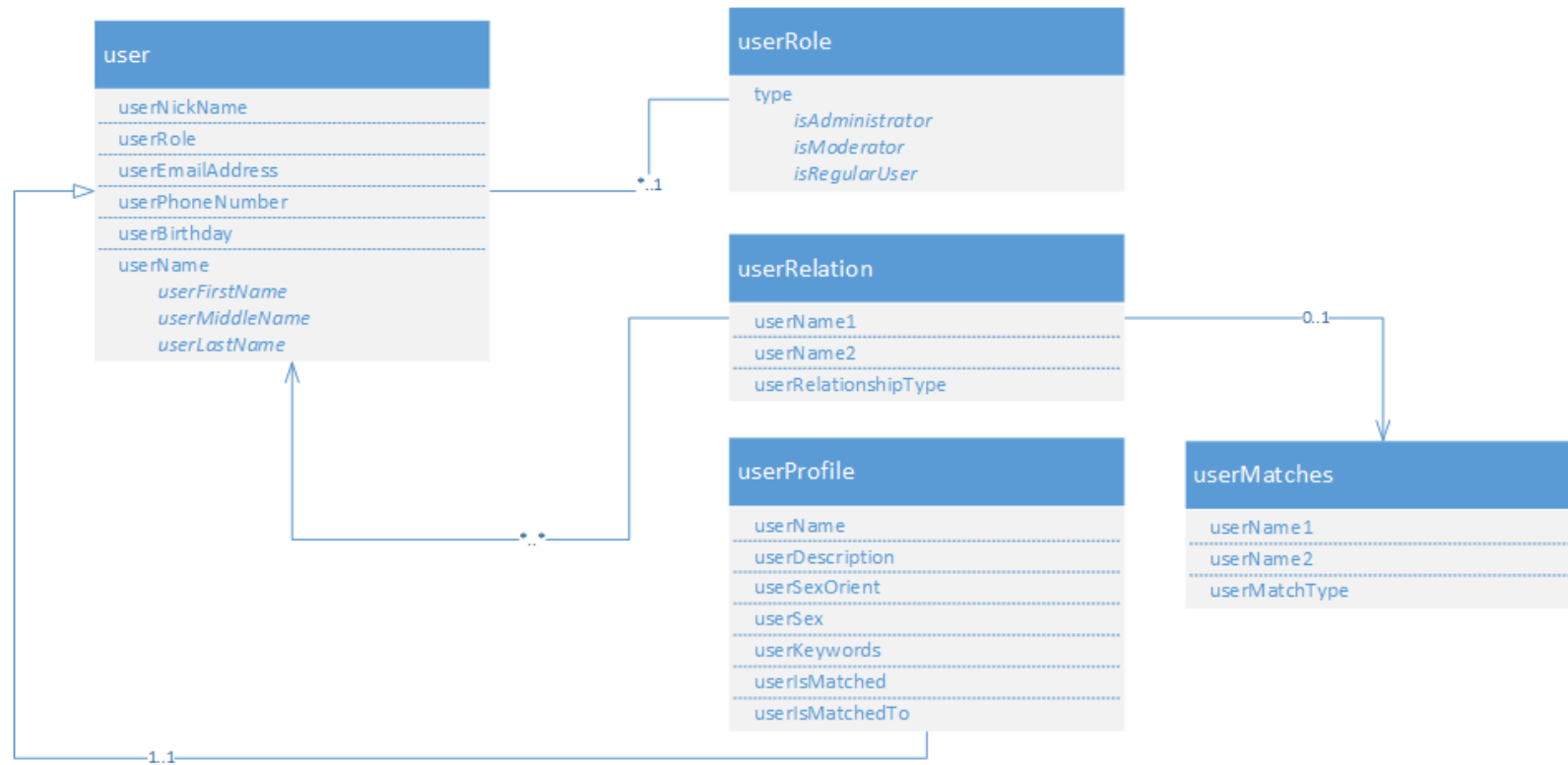


Possible Matching Attributes – Master List

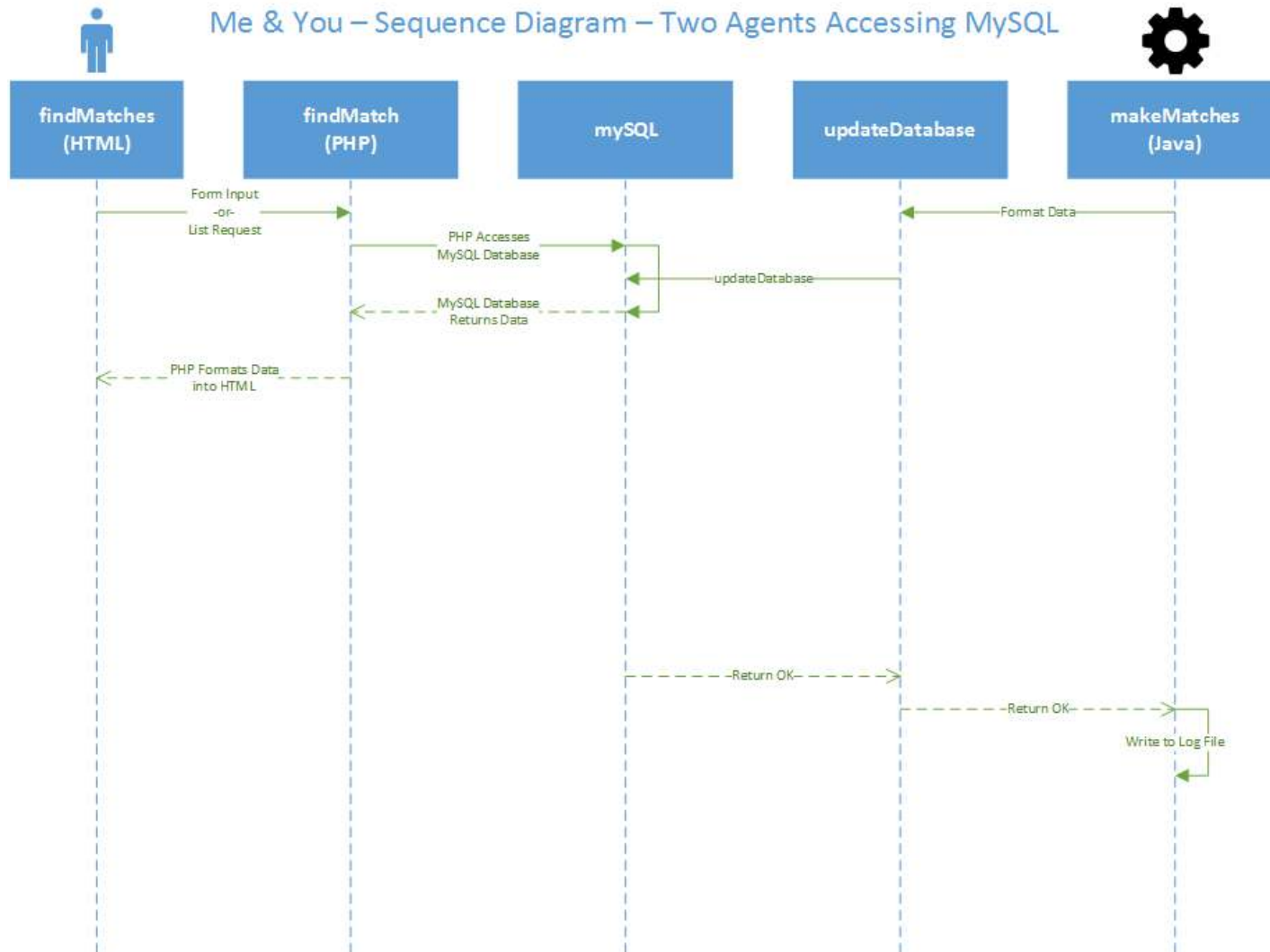
- Accomplishments
- Activities
- Age
- Alias
- Athletics
- Birthday
- Body Type (i.e. Athletic, More to Love, etc.)
- Books Read
- Businesses Owned
- Career Field
- Casual Encounters (gym, school, market, etc.)
- Charity Work / Contributions
- Citizen of the Year
- Communication Habits
- Consumer Behavior
- Culture
- Denomination (Religious)
- Distance
- Drinks?
- Educational Background (i.e. GPA, Classes taken, Schools attended)
- E-mail Addresses (possibility for multiple addresses)
- Employer
- Ethnicity
- Faith
- Family Related
- Family Values
- Favorites
- Financial Background
- First Name
- Food They Eat
- Friends in Common
- Government Involvement
- Hair Color
- Has Kids?
- Height
- Heritage
- Hobbies
- Hometown(s)
- How Long They've Been Single
- IQ
- Income
- Interests
- Integrity
- Languages Spoken
- Last Name
- Legal
- Leisure Activities
- Loyalty
- Maiden Name
- Married Name
- Middle Name
- Movies Watched
- Music They Listen To
- Nationality
- Nickname(s)
- Occupation
- Organization(s) (i.e. NGOs, non-profits)
- Passion
- Personality
- Pets
- Place of Origin
- Places of Worship Membership
- Political Views & Involvements
- Power
- Press History
- Psychology
- Race
- Real Estate
- Relationship Marketing
- Relationship Status (Never Married, Married, Divorced, Separated, Widowed, etc.)
- Religion (Religious Beliefs, Spirituality...)
- Reputation
- Security
- Sex
- Sexual Orientation
- Shopping Habits
- Smokes?
- Success Rating
- Surroundings
- Talents (i.e. play a musical instrument)
- Technology Use
- Title(s)
- Type of Car They Drive
- Values
- Want Kids?
- Weight
- Work

UML Diagrams (Week 2)

Me&You userProfile and Relationships Class Diagram

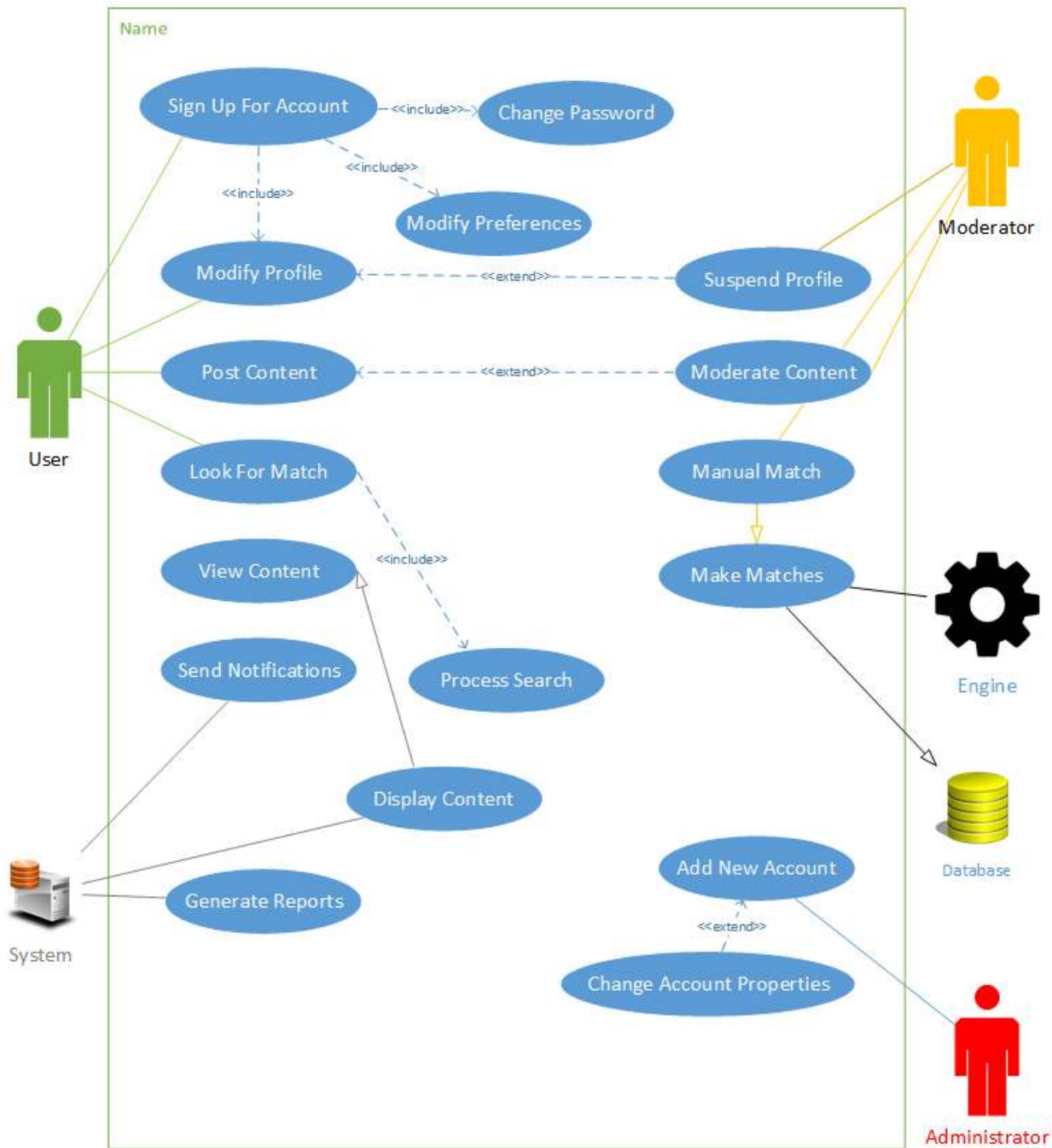


UML Diagrams (Week 2)



UML Diagrams (Week 2)

Me&You – General Use Case Diagram



REQUIREMENTS ELICITATION AND ANALYSIS

- Functional Requirements:
 - Information will be organized and stored in the database
 - Attributes are entered via GUI and stored in database
 - Database uses relational model
 - Database will store modification flags (matches made)
 - Database will store functional data needed for the system
 - Persistent Data
 - Searches
 - User Profile
 - Passwords should be encrypted.
 - Search Results (Matches)
 - Cached Data (commonly used data)
 - Template
 - Indexes
 - Access Control
 - Event-driven control, which will dispatch the request to the appropriate object whenever an event becomes available that meets the minimum-security requirements, by verifying:
 - System admin will have access to administer the database (users will not)
 - Policy-Based Rules, through lists of user privileges that will be stored in a database table
 - Policies will define what data can be accessed by what users, based on the data that they provide for their searches.
 - Filtering will take place through the GUI, as well as protection of execution from the database in the engine.
 - Access control lists will be used to block users, or limit how much site traffic and database accesses they can perform before a timeout initiates.
 - The access control lists need to block SQL Injection and Buffer Overflow possibilities.
- Non-functional requirements:
 - Database attribute entries are the same as the user account requirements
 - Response time to return request to user must be minimal
 - Once the prototype is completed the server will be shutdown.
 - Keep the project within budget.

SYSTEM DESIGN

System Design

- Hardware must allow us to serve the goal of 20k-25k users.
- Hardware Mapping
 - Power Blade 510
 - 2x Xeon Processors
 - 64GB of RAM
 - Host Operating System: Windows Server 2008 R2
 - VMWare
 - Windows Server VPS #1 – Me&You – Production Server
 - Uses Apache, MySQL, PHP/CSS
 - Windows Server VPS #2 – Me&You – Development Server
 - Uses Apache, MySQL, PHP/CSS
- Ideally to deploy at scale we would implement utilizing a Platform as a Service (PaaS) for the Website, which comprises the front-end of the system including the Graphical User Interface (GUI). This will allow for meeting growing demand and increased market share.
- We need to get Professor Jonas to sign off on the minimum requirements before we add any other features beyond Crushes, and the bare minimum. Then if everything is working, we will venture into adding more features.

OBJECT DESIGN

Object Design

- Properties / Attributes will be stored in columns
- Data Members will be stored in rows
- Software Group and Database Group will define tables, together
 - Need to define the maximum length in characters and data types for each field
(*from Software Development Group*)
- Front-end (GUI) Group needs statistics from the database, such as how many registered users there are, how many matches there are, etc.
- Just to do Crush searches for 25,000 users, we will need 125,000 rows (5 Attributes x 25k users) if there are 5 required attributes per match
 - It could wind up being more than this, if more Attributes were added
 - It could also be more if we do other search types, such as Love, Family, and Friend
 - Is there a more efficient way to conduct the searches without creating so many different rows in the database?
 - We don't want to have to create that many rows for every single search on the website, each time a search occurs, or is processed by the Engine.
 - Is there a way to sort the results to make it more efficient?
- Matches table linking to the User Profile table needs to be added to William Jones' Demo, based on what was discussed during the meeting on 3/14/2017.
 - We need to get William Jones' demo working on the UNH server to have Front-end (GUI) and Database Groups can work on it. We should get it by 3/22/2017.

IMPLEMENTATION AND TESTING

Implementation

- **Software Group:** Java Engine will sort database
- **Front-end (GUI) Group:** PHP will access database
- **Database Group:** Database will be loaded with SQL files created on User PCs
 - MySQL 5.5? 5.7.17?
 - We're going to use MySQL, because it's what we know, and what we have access to with our budget of \$0. We will have to use the existing internal UNH server for this project.
- Software Group said that maximum field lengths will be used for all fields.

Testing

- Create custom data pertaining to different types of profiles
- Testing for up to 25k users, up to 10k searches - or 5,000 potential matches being processed.
- [REDACTED] is the Quality Assurance Director
 - [REDACTED] needs to write a script to inject the 20k-25k user records, and then inject the 10k searches into the database after that, to see how the Engine works to process the massive load of requests, and monitor for errors in the Administrative logs on the server.
 - Coordinate with Macuei Mathiang, Quality Assurance for Software Development in regards to this.
 - Use SQL INSERT statements through the back-end to dump the data into the database. Also, familiarize yourself with how to clear out the database and reset it with your script.
 - [REDACTED] also needs to test that the database is being correctly accessed in numerous test cases that are duplicated every time the database needs to be tested with Front-end (GUI).
 - Coordinate with Josh Quigley, Quality Assurance for Front-end (GUI) in regards to this.
- Creation of Fake Records & Searches
 - [REDACTED] will create the 20k-25k fake user records
 - [REDACTED] will also create the 10k of fake searches, that use the Engine – so they must link to actual user records.
 - The easiest way is to just create user records is like numbering them as First10001, Last10001, and then you basically fill out the attributes for searching by looking for a match between the first 5,000 users – so up to First15000, Last15000 with the attributes between the two matching records coinciding in each other's searches. If you need help, ask Patrick McElhiney.

PROJECT MANAGEMENT

Project Management

- The Project Manager is Professor Jonas
- Task Plan, Schedule, Budget, Organization, and Project Environment will be defined.
 - **Stage I: Project Initiation Process**
 - Schedule is in syllabus.
 - Who develops the schedule?
 - Professor Jonas?
 - Budget is \$0.
 - Organization – org chart that we made. Master Documents.
 - Develop Project Standards
 - Assign Communication Infrastructure
 - E-mail – to send files.
 - Meet in Class – Every Week on Wednesday from 5:31PM to 8:30PM
 - Slack – to share text conversations without voice / video conversations
 - Zoom Meetings
 - Skype for Business
 - Blogs – Wiki Journal Entries for Graduate Students
 - Setup Meeting & Reporting Procedures
 - Template for Meeting Minutes
 - Template for Graduate Student Logs
 - Template for Master Documents and other forms of documentation.
 - Development Methodology and Development Tools
 - **Stage II: Project Monitoring and Control Process**
 - Ensures that project is executed per the task plan and budget.
 - If the Project Manager deviates from the Schedule, resources will be re-appropriated or the schedule will be re-planned.
 - The software project management plan (PMP) is updated to reflect any of the changes above.
 - **Stage III: Software Quality Management Process**
 - Ensures that the system under construction follows the required quality standards.
 - A separate Quality Management team executes this process to avoid conflicts of interest.

CONFIGURATION MANAGEMENT

Configuration Management

- Configuration Manager is [REDACTED], Architect.
 - Make sure team meets all deliverable deadlines, i.e. all work is completed by due date(s).
 - Make sure there are no last-minute changes before a live Demo using the Database on the Production Server.
 - Be sure to split up the work load among all group members.
 - Make sure all group members have the necessary software and assigned tasks to stay busy each week.
- [REDACTED] is Auditor
 - Responsible for selecting the version of the prototype website with MySQL Database that we will release for grading to Professor Jonas.

Configuration Management Process – focuses on tracking and control of work products

- Items under configuration management include source control, development models, software development plan, and all documents visible to the project participants.
- 1st Function of configuration management is the identification of configuration items.
 - Which subsystems are likely to change?
 - Which subsystem interfaces should not change?
 - Each subsystem likely to change is modeled as a configuration item, and its state labeled with a version number.
- 2nd Function: Manage Change through a Formal Process
 - Change Request is first logged, and then analyzed, then accepted if consistent with the goals of the project
 - The change is then approved or rejected depending upon the forcing impact of the change on the overall system
- 3rd Function: Record sufficient status information on each version of each configuration item and its dependencies.

INFRASTRUCTURE MAINTENANCE

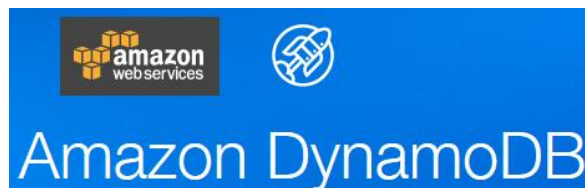
- Documentation
 - Update with Quality Assurance so that next year's class or a graduate student for a Master's Project can pick up the project and launch it with our recommendations to scale up to millions or billions of users (in the cloud).
 - Usability to the End User / Other Actors – Quality Assurance
 - User Experience Testing (*next year*)
 - Incorporating Marketing / Business Plan with Existing Website (*next year*)
 - Each team needs to document their work in granular detail, per Professor Jonas' requests.
- Little to no additional maintenance or capital expenditures. We are just building a prototype.

SYSTEM DEPLOYMENT TO THE END USER

- We are building a prototype; therefore, we are not actually deploying the system to the end user.

The Future – Cloud Deployment

- Amazon RDS
 - MySQL as a Service
 - Scalable
- Next year's class, or a graduate student for a Master's Project, will be launching the project through integration of the existing platform with cloud infrastructure, such as Amazon AWS or Azure.
 - Database should be scalable, while we are limited to using MySQL for this class because of our budget.
 - Ideally, this type of website should be launched using a NoSQL key-value database in the cloud, such as:



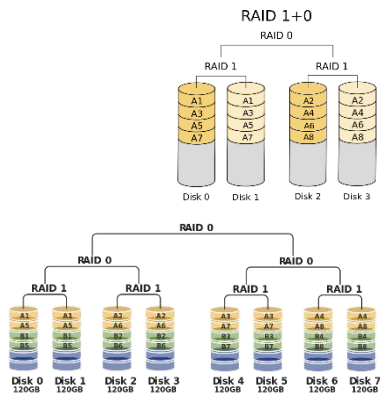
- **Dynamo DB through Amazon AWS**



- **Apache Cassandra** which was developed by Facebook (<https://cassandra.apache.org/>).
- ✓ It's not a single thread, because it's disbursed.
- ✗ Software Development found information for searching from the web in this way, with server clusters, but **not for any searching at the field level with MySQL** such as from a C# application running in the background (Engine).
- Also, the website should be built on the Linux platform, as it is more secure.

SYSTEM DEPLOYMENT TO THE END USER

- In the future class(es), they will need to split up the attributes table across multiple volumes.
 - This can be done in SQL and Oracle, where the table can be split into different disks on the host machine with partitioning.

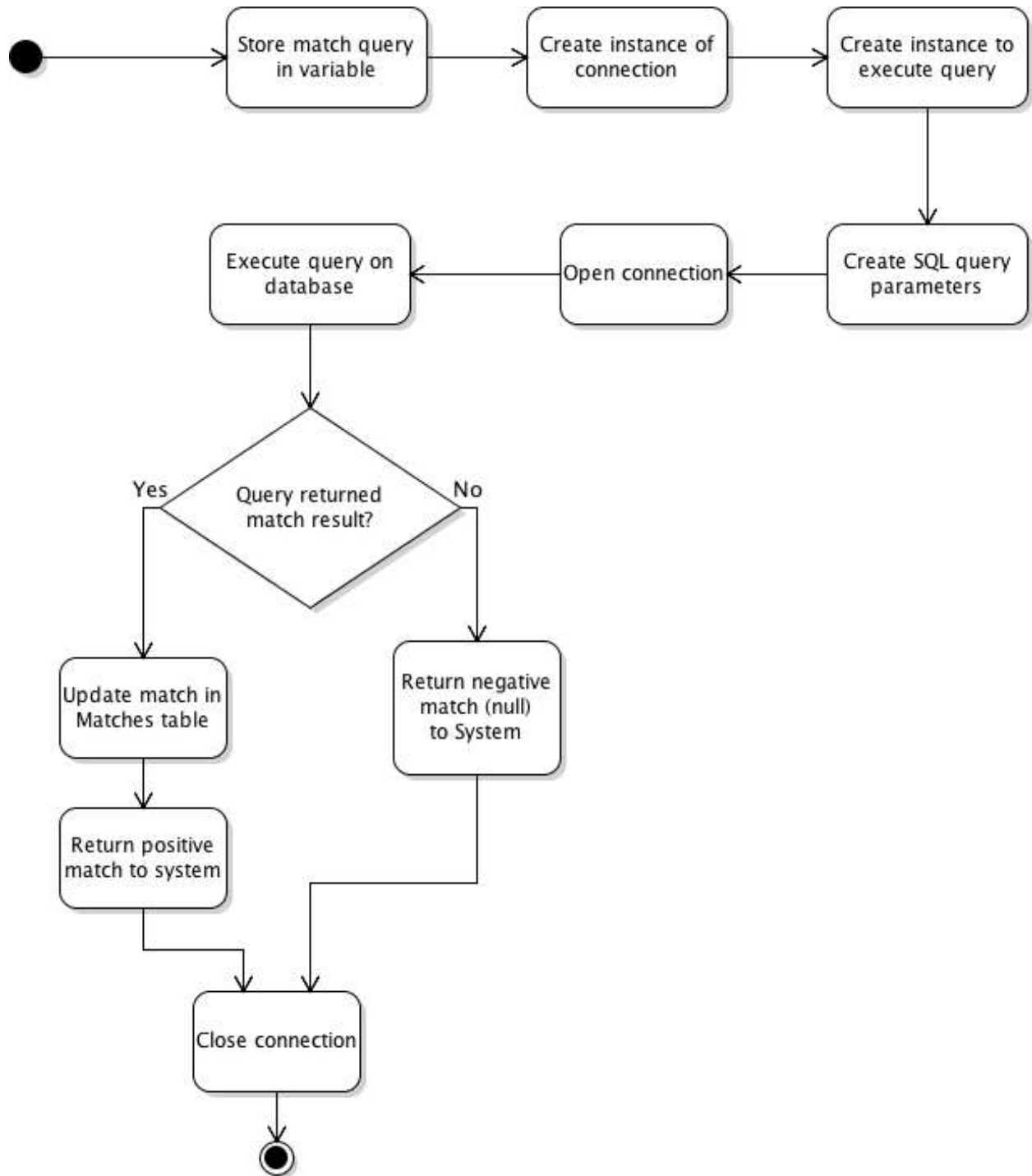


- Instead of having the table on one drive, it could go on a RAID 10, aka RAID 0+1 (striping with mirror) which would speed up the read speed, and read intensive SSDs could be used for this purpose.

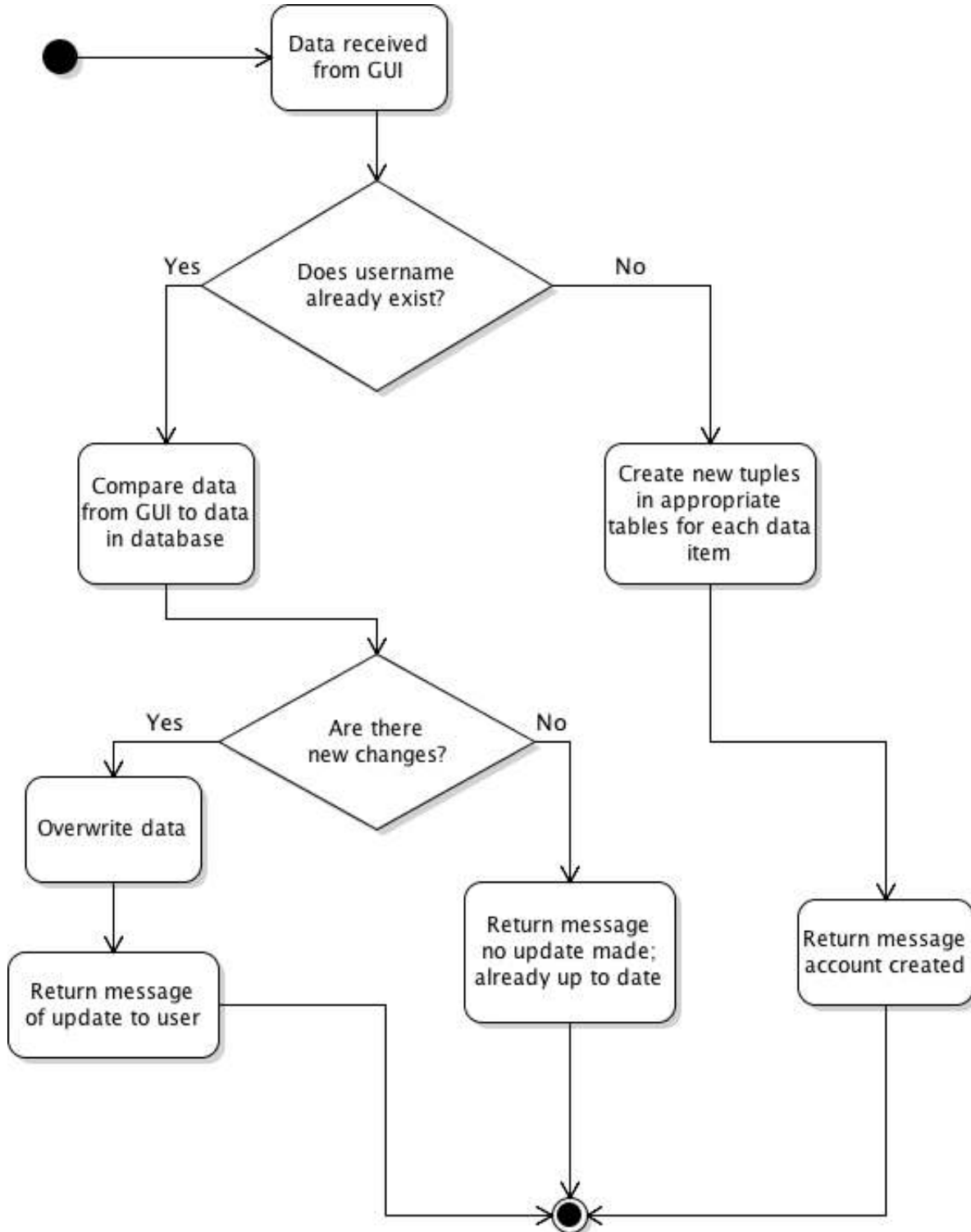
- For extremely read intensive access to disk drives, RAID 100, aka RAID 10+0 (striped mirrored striped, or plaid RAID) would work well, because there would essentially be 2 copies of everything, as well as the striped volumes which are also mirrored.

- **Wikipedia: Nested RAID Levels** (https://en.wikipedia.org/wiki/Nested_RAID_levels)
- In the future class(es), they may also want to look at splitting up the Attributes table across multiple servers.
 - This could be done with Microsoft Cluster Server (MSCS), or a computer program that allows servers to operate together as a single computer cluster, to provide failover and increased availability of applications, or parallel calculating power in case of high-performance computing (HPC) clusters.
 - This could be done by clustering servers based on the first letter of the value of the String that is being searched for, such as when searching for someone named “Bob”, it would search in the “B” section of the server clusters in the cloud.
 - This would essentially be virtualized threading at the application level.
 - In the future, the data would have to be stored across thousands of servers, perhaps, or if it became the size of Facebook, it would need its own dedicated data centers to manage all the data.
 - Apache Cassandra, AWS Dynamo DB, and other NoSQL databases are built for this type of application with a vast improvement in efficiency, especially considering how many interactions the website would have at a single moment. There could be hundreds of thousands of users on the site simultaneously if it was created properly.

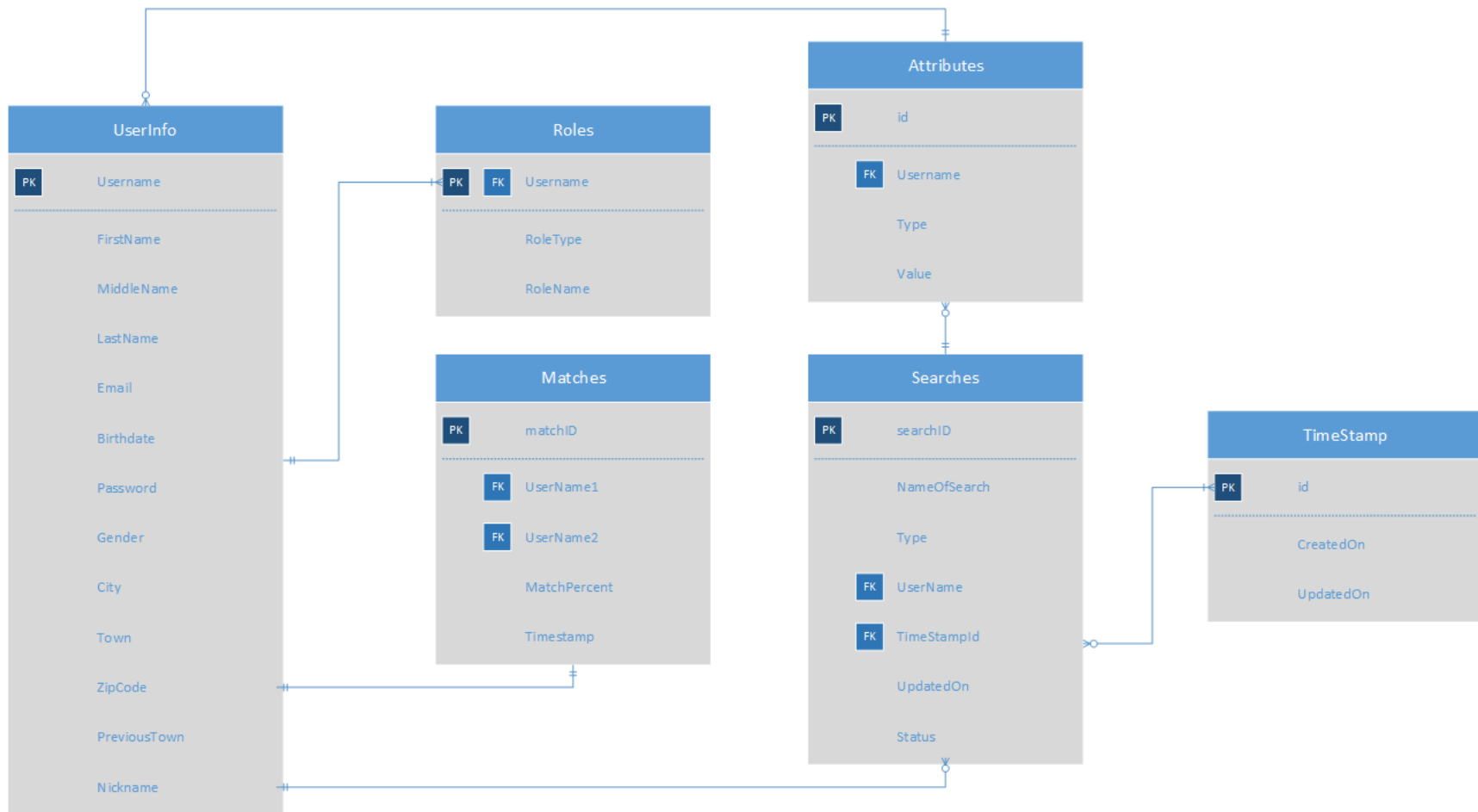
UML Diagrams (Week 4)



UML Diagrams (Week 4)

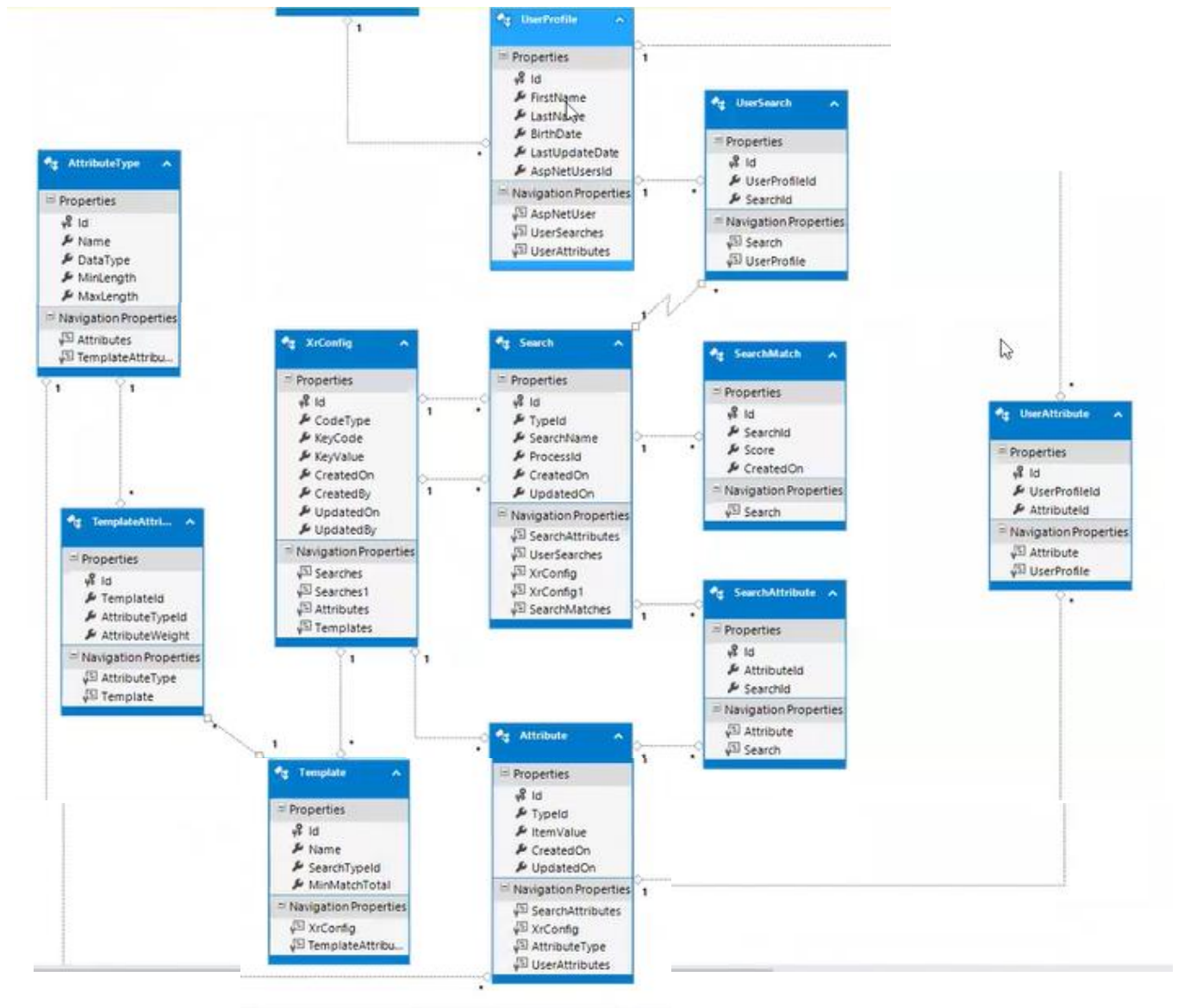


Crow's Feet Diagram (Week 4)



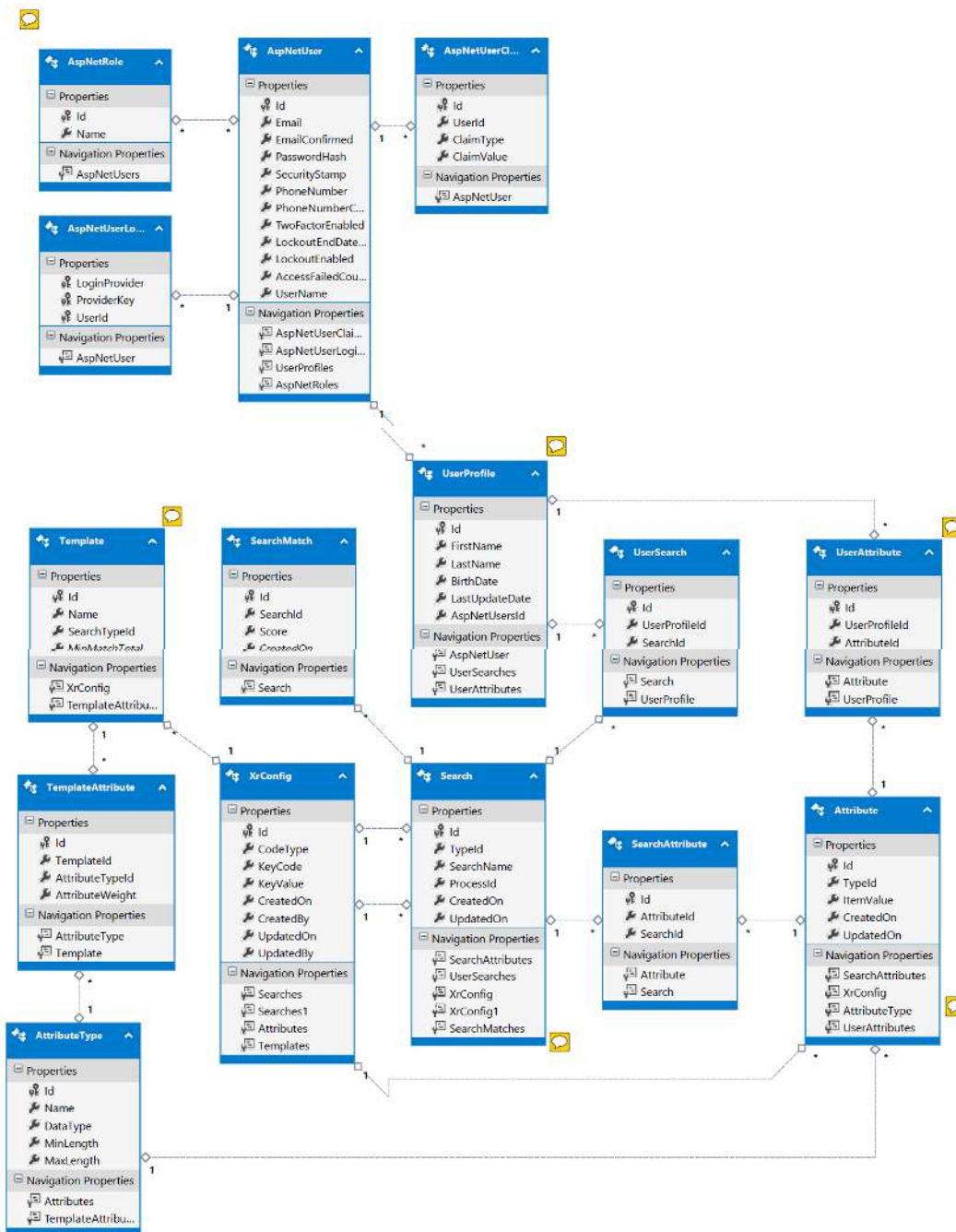
This diagram was developed in class on 3/8/2017, based on the information that we have received from Software Development as of that time. The current issue is that the database model keeps changing, as Software Development continues to add more and more fields that they need for their engine to process the matches.

Partially Working Demo – Database (Week 5)



Multiple screenshots patched together from Graduate Student meeting on 3/14/2017, showing most of William J.'s computer screen with his Azure account, with the Database Structure that he is using with the Demo he has created on his own.

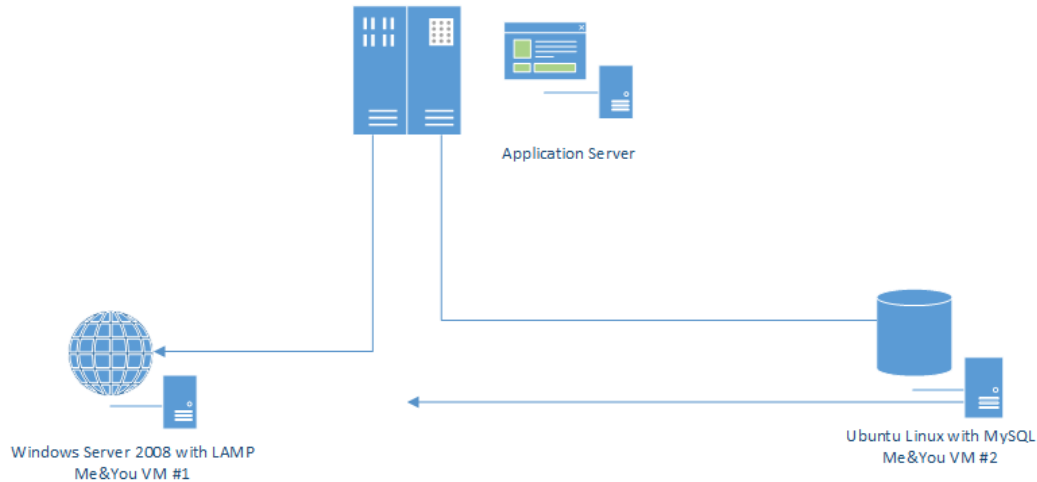
Data Model from Software Develop. (Week 6)



The database structure shown above was received from William Jones on 3/21/2017. Per discussion about it in class, ██████████ said that the field names were too generic, and that we would need to change them.

Old System Configuration Diagram (Week 7)

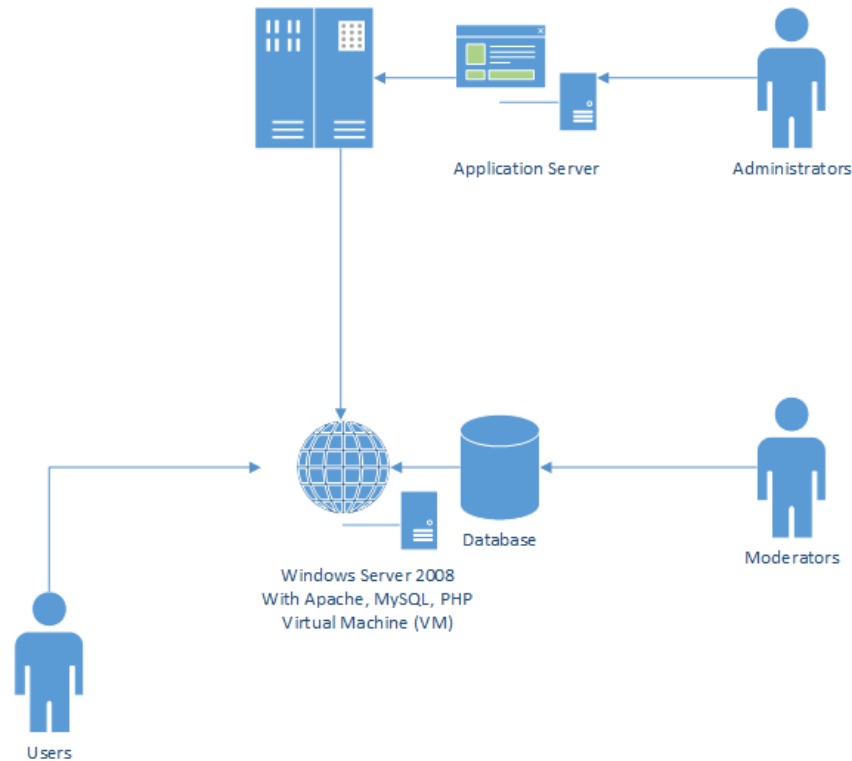
Old Configuration for Me&You Servers (VMs)



This diagram shows the old structure of how we found out how the old systems were working, with one VM that was hosting the website, and another VM configured to host the database. This is a kind of archaic design model, because it used separate servers, or Virtual Machines, for different purposes – when everything could have just been configured on the same server. This shows that they probably didn't know what they were doing, and judging by how the website doesn't even work as of the way we found it – they probably didn't know what they were doing at all.

Configuration Diagrams (Week 7)

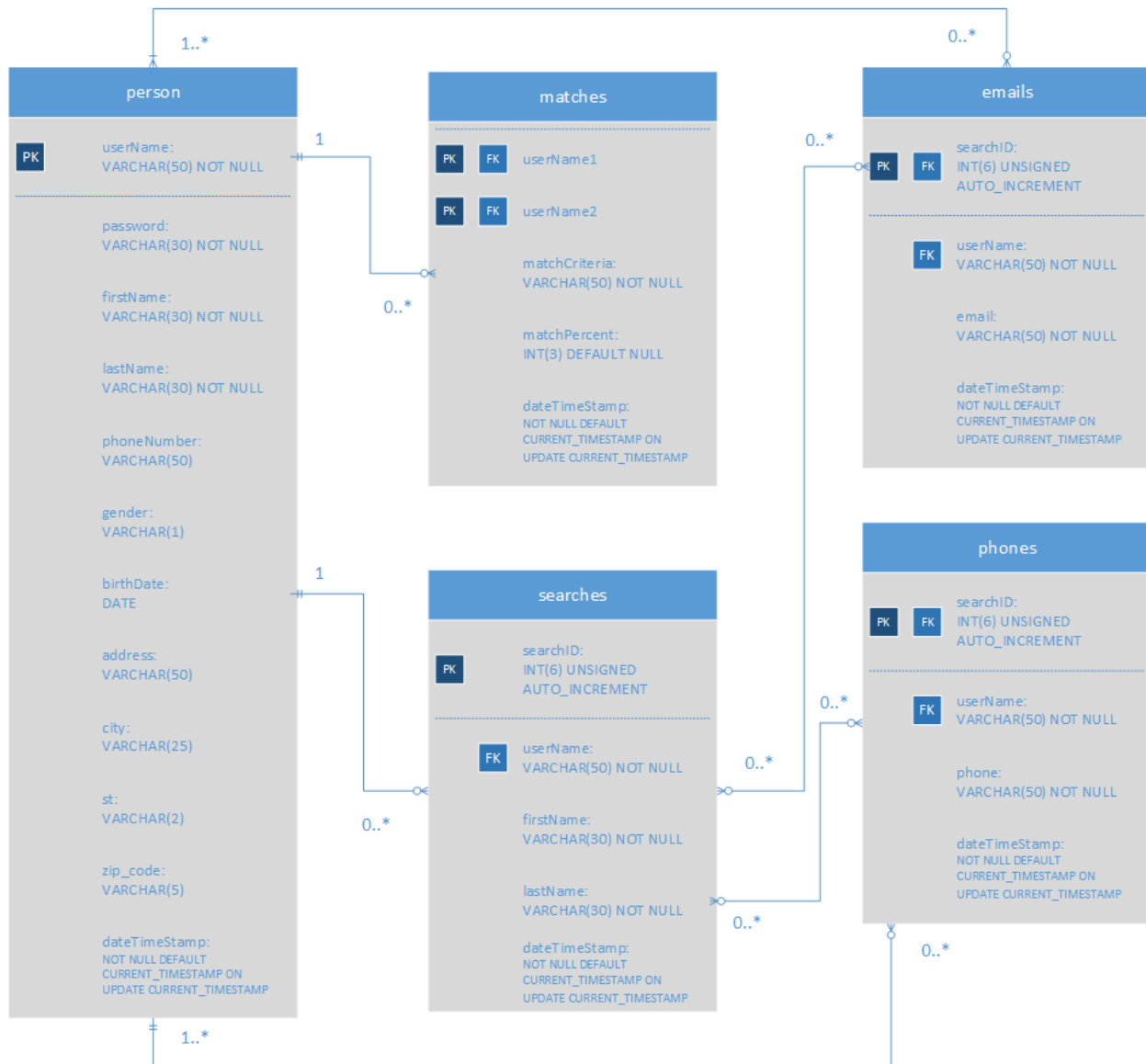
Spring 2017 - Me&You System Layout with Actors



This is the new System Design layout, which shows how the various actors from our Use Cases connect to different parts of the systems. Users connect to the website through the web server (HTTP), while Moderators can connect to the backend and edit the database records manually. Administrators can connect to the host system, which in this case is the same physical server that the Virtual Machine is running on.

Crow's Feet Diagram (Week 7)

DATABASE MEANDYOU SQL Crow's Feet Diagram – 3/23/2017 v2



This diagram shows a Crow's Feet Diagram for the corrected version of the SQL file "create6.sql" which was determined to be the newest one in the file repository, directory "meandyou", also merged with "SQL\create.sql". This model is quite different than what William Jones has come up with in his Software Development group, but this is much simpler, and it was already done for us. I suppose what they were trying to do was to make a table for each of the attributes. I still like William J.'s way of creating a Template and Attributes Table.

Crow's Feet Diagram (Week 7) (continued...)

Notes on Changes to SQL:

- **Database Name**
 - Set the database name correctly to “YOUANDME”, it was set to “MEANDYOU”.
- **NOT NULL Settings**
 - “NOT NULL” statements were removed from gender, birthDate, address, city, state, and zip_code, because the fields are not required to populate a search, i.e. you can just search by firstName lastName.
- **dateTimeStamp Settings**
 - Set all of the dateTimeStamp columns to properties: “NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP”
 - “NOT NULL” – Value of field cannot be NULL
 - “DEFAULT CURRENT_TIMESTAMP” – Sets the default field value to the current server timestamp.
 - Timestamp Format Set To: YYYY-MM-DD HH:MM:SS
 - “ON UPDATE CURRENT_TIMESTAMP” – Sets the value of the field to the current server timestamp when the table row is updated.
- **Person Table**
 - **User Name**
 - The userName is the User’s e-mail address.
 - **Gender**
 - gender was changed from VARCHAR(6) to VARCHAR(1), because it can be represented by letter codes, such as:
 - F – Female
 - M – Male
 - T – Transgender
 - **City**
 - city was changed from VARCHAR(15) to VARCHAR(25) because there are city names that are longer than 15 characters.
 - **State**
 - The field “state” was spelled as “state_” incorrectly, so that was fixed.
 - state will be a drop-down menu of 2-letter state codes, so it was set to VARCHAR(2)
 - **Zip Code**
 - There was no need to include the additional four numbers on the end of the zip code, as we have no use for them, however this should be noted as such.
- **Matches Table**
 - In table “matches”, userName and matchUserName were both increased to VARCHAR(50) to correspond with the size of the maximum length of e-mail address in the “person” table – the same as person:userName.
 - Added in the fields from meandyou\SQL\create.sql, such as “userName1”, “userName2”, “matchCriteria”, “matchPercent”, and made PRIMARY KEY out of (userName1,userName2).
- **E-Mails Table**
 - In the table “emails”, field “email” was changed to VARCHAR(50) from VARCHAR(30) to correspond with the size of the max. length of e-mail in person:userName field.

SQL Code – Revision #1, v3 (Week 7)

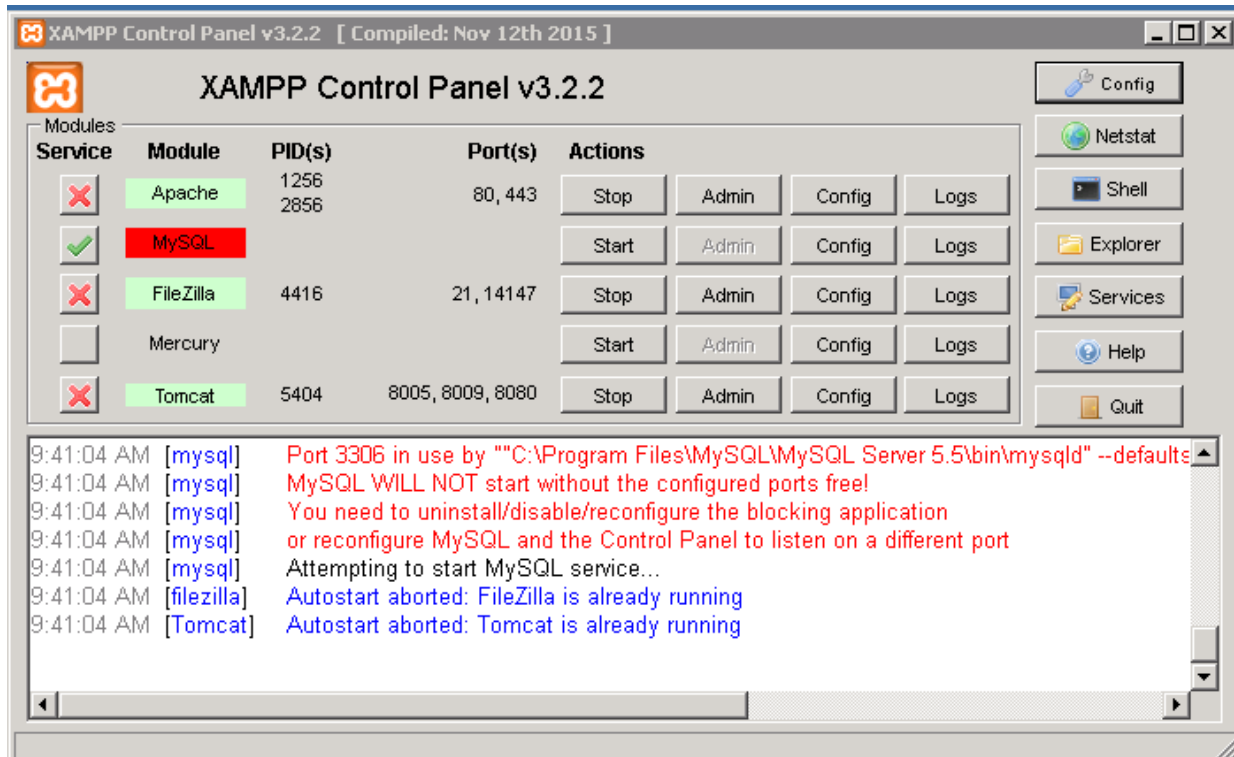
```
DROP DATABASE MEANDYOU;
CREATE DATABASE MEANDYOU;
USE MEANDYOU;
SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET time_zone = "-05:00";
CREATE TABLE person (
  userName VARCHAR(50) NOT NULL PRIMARY KEY,
  password VARCHAR(30) NOT NULL,
  firstName VARCHAR(30) NOT NULL,
  lastName VARCHAR(30) NOT NULL,
  phoneNumber VARCHAR(50),
  gender VARCHAR(1),
  birthDate DATE,
  address VARCHAR(50),
  city VARCHAR(25),
  st VARCHAR(2),
  zip_code VARCHAR(5),
  dateTimeStamp TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
CREATE TABLE matches (
  userName1 VARCHAR(50) NOT NULL,
  userName2 VARCHAR(50) NOT NULL,
  matchCriteria VARCHAR(50) NOT NULL,
  matchPercent INT(3) DEFAULT NULL,
  dateTimeStamp TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
ALTER TABLE matches
  ADD PRIMARY KEY (userName1,userName2),
  ADD CONSTRAINT matches_userName_fk_1 FOREIGN KEY (userName1) REFERENCES person (userName),
  ADD CONSTRAINT matches_userName_fk_2 FOREIGN KEY (userName2) REFERENCES person (userName);
CREATE TABLE searches (
  searchID INT(6) UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
  userName VARCHAR(50) NOT NULL,
  firstName VARCHAR(30) NOT NULL,
  lastName VARCHAR(30) NOT NULL,
  dateTimeStamp TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  FOREIGN KEY (userName) REFERENCES person(userName)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
CREATE TABLE emails (
  searchID INT(6) UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
  userName VARCHAR(50) NOT NULL,
  email VARCHAR(50) NOT NULL,
  dateTimeStamp TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  FOREIGN KEY (searchID) REFERENCES searches(searchID),
  FOREIGN KEY (userName) REFERENCES person(userName)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
CREATE TABLE phones (
  searchID INT(6) UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
  userName VARCHAR(50) NOT NULL,
  phone VARCHAR(50) NOT NULL,
  dateTimeStamp TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  FOREIGN KEY (searchID) REFERENCES searches(searchID),
  FOREIGN KEY (userName) REFERENCES person(userName)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Issues To Resolve with SQL (Week 7) (continued...)

Notes on SQL Issues:

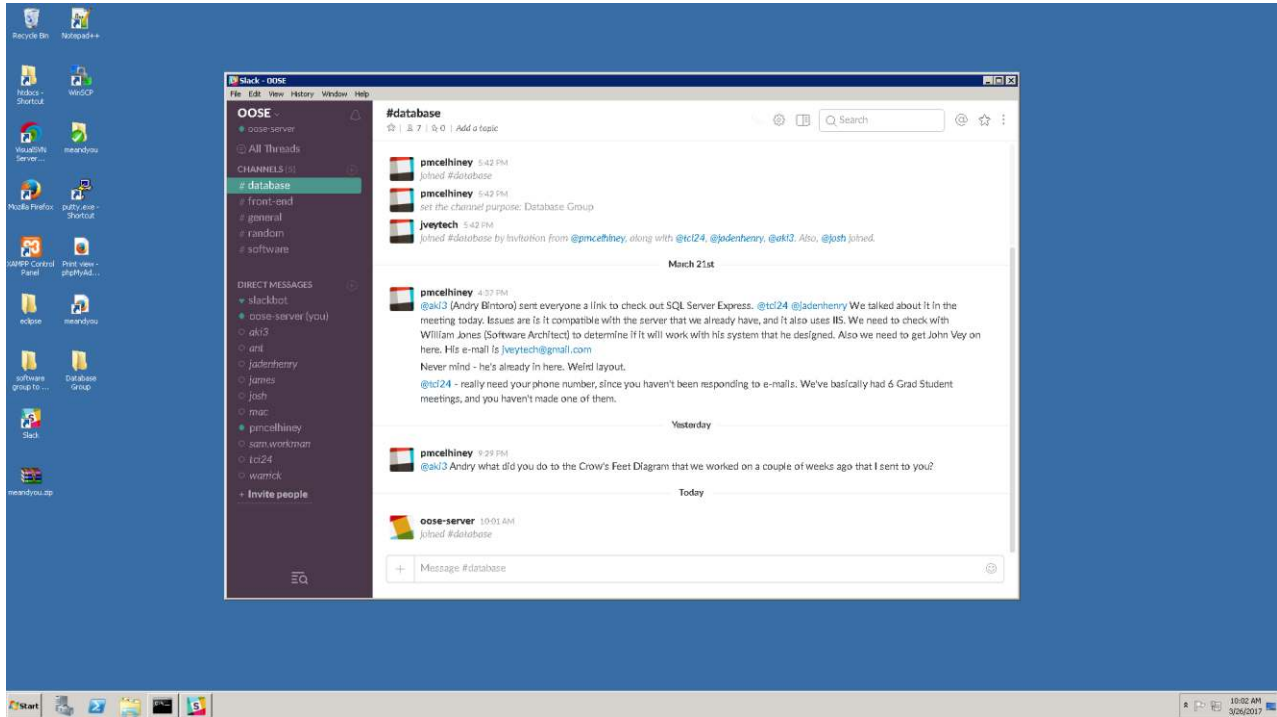
- **Separate Sets of Files**
 - These settings above have been conformed mostly to the structure in the seven SQL files in the root directory of the “meandyou” folder:
 - **meandyou\create.sql**
 - **meandyou\create2.sql**
 - **meandyou\create3.sql**
 - **meandyou\create4.sql**
 - **meandyou\create5.sql**
 - **meandyou\create6.sql**
 - There is another set of SQL files that I tried to merge for the most part without changing the structure of the database. What was not changed, that is different in the “SQL” folder’s SQL files:
 - **meandyou\SQL\create.sql**
 - For table “emails”, it has userName VARCHAR(50) NOT NULL
 - For table “emails”, it has no Primary Key
 - For table “matches”, it has userName1 VARCHAR(30) NOT NULL
 - It should be **userName1 VARCHAR(50) NOT NULL** if used.
 - For table “matches”, it has userName2 VARCHAR(30) NOT NULL
 - It should be **userName2 VARCHAR(50) NOT NULL** if used.
 - For table “matches”, it has matchCriteria VARCHAR(50) NOT NULL
 - It is unknown what a matchCriteria is for – I will have to research this more.
 - For table “matches”, it has matchPercent INT(3) DEFAULT NULL
 - This seems to correlate to a number between 0 and 100 to determine the percentage of the match that was found. It is unclear if this is needed.
 - For table “person”, it has st VARCHAR(30) NOT NULL
 - Just a different spelling of state – but it should be VARCHAR(2) with a drop-down menu on front-end.
 - For table “phones”, it has phone INT(9) NOT NULL
 - It should be phone VARCHAR(10) at a minimum – but it may actually need about 15 characters when you consider the International numbers.
 - According to an article (<http://stackoverflow.com/questions/723587/whats-the-longest-possible-worldwide-phone-number-i-should-consider-in-sql-varc>) the longest International phone number combination is 22 characters. However, it recommends to set it to 50 characters, because it doesn’t actually take up that much room because it’s a VARCHAR.
 - For table “searches”, it has status VARCHAR(10) DEFAULT NULL
 - It is unknown what “status” is for – something to do with the Engine. I will have to research this more.
 - For table “matches”, it has an ALTER TABLE statement that reads “ADD UNIQUE KEY ‘username_match’ (‘userName1’, ‘userName2’), and then “ADD KEY ‘userName2’ (‘username2’).
 - It is unclear what this is for.
 - There are CONSTRAINT definitions in this file that I have not added.

Progress Made (Week 7)



The server is running, besides that MySQL is not running correctly with the installation of XAMPP – which led Patrick M. to believe that there were two installations, or two versions of MySQL running. The error states that C:\Program Files\MySQL\MySQL Server 5.5\bin\mysqld is running on port 3306, where XAMPP is trying to start MySQL as well.

Progress Made (Week 7) (continued...)



Slack was installed on the server, using the credentials:

Microsoft Outlook:

OOSE Server
oose-server@outlook.com
2017OOSE
1/1/1983

Slack Password:

2017pw=OOSE

SQL File Progress (Week 7) (old design...)

```
--
-- Database: `meandyou`
--
-- Version: 2017-03-26 v4
-- Developer: Patrick R. McElhiney
-- Organization: MCE123 c/o University of New Hampshire, Manchester

DROP DATABASE MEANDYOU;

CREATE DATABASE MEANDYOU;

USE MEANDYOU;

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET time_zone = "-05:00";

--
-- Table: 'person'
--
CREATE TABLE person (
  userName VARCHAR(50) NOT NULL PRIMARY KEY,
  password VARCHAR(30) NOT NULL,
  firstName VARCHAR(30) NOT NULL,
  lastName VARCHAR(30) NOT NULL,
  phoneNumber VARCHAR(50),
  gender VARCHAR(1),
  birthDate DATE,
  address VARCHAR(50),
  city VARCHAR(25),
  st VARCHAR(2),
  zip_code VARCHAR(5),
  dateTimeStamp TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Table: 'matches'
--
CREATE TABLE matches (
  userName1 VARCHAR(50) NOT NULL,
  userName2 VARCHAR(50) NOT NULL,
  matchCriteria VARCHAR(50) NOT NULL,
  matchPercent INT(3) DEFAULT NULL,
  dateTimeStamp TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Primary key and foreign keys for table 'matches'
--
```

```
ALTER TABLE matches
ADD PRIMARY KEY (userName1,userName2),
ADD CONSTRAINT matches_userName_fk_1 FOREIGN KEY (userName1) REFERENCES person (userName),
ADD CONSTRAINT matches_userName_fk_2 FOREIGN KEY (userName2) REFERENCES person (userName);

--
-- Table: 'searches'
--
CREATE TABLE searches (
  searchID INT(6) UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
  userName VARCHAR(50) NOT NULL,
  firstName VARCHAR(30) NOT NULL,
  lastName VARCHAR(30) NOT NULL,
  dateTimeStamp TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
  FOREIGN KEY (userName) REFERENCES person(userName)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Table: 'emails'
--
CREATE TABLE emails (
  searchID INT(6) UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
  userName VARCHAR(50) NOT NULL,
  email VARCHAR(50) NOT NULL,
  dateTimeStamp TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
  FOREIGN KEY (searchID) REFERENCES searches(searchID),
  FOREIGN KEY (userName) REFERENCES person(userName)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Table: 'phones'
--
CREATE TABLE phones (
  searchID INT(6) UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
  userName VARCHAR(50) NOT NULL,
  phone VARCHAR(50) NOT NULL,
  dateTimeStamp TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
  FOREIGN KEY (searchID) REFERENCES searches(searchID),
  FOREIGN KEY (userName) REFERENCES person(userName)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Test data: 'person'
--
INSERT INTO person (userName, password, firstName, lastName, phoneNumber, gender, birthDate, address, city,
st, zip_code, dateTimeStamp) VALUES
('aprice5@wufoo.com', '7rWjXR', 'Adam', 'Price', '5555555560', 'M', '1986-06-18', '34 Boon Dr', 'Allentown', 'NH',
'03107', '2016-05-01 20:03:47'),
('bmills4@abcnews.com', 'oNeamSXqBA', 'Betty', 'Mills', '5555555559', 'F', '1983-07-15', '34 Fort Way',
'Trexlertown', 'GA', '03101', '2016-05-01 20:03:45'),
('jfowler8@51.la', 'bv4E969C', 'Joan', 'Fowler', '5555555563', 'F', '1984-11-19', '32 Grass St', 'Hudson', 'NH', '03172',
'2016-05-01 20:03:48'),
```

```
('khunt6@newyorker.com', 'fjiPPrZ1SUzB', 'Keith', 'Hunt', '5555555561', 'M', '1989-01-22', '34 Frank St',  
'Londonderry', 'VA', '03109', '2016-05-01 20:03:48'),  
('lburke1@google.fr', 'iqNv9ciiOBg', 'Lawrence', 'Burke', '5555555556', 'M', '1986-05-19', '34 City Rd', 'Bedford',  
'MA', '03134', '2016-05-01 20:03:36'),  
('mstone2@netscape.com', 'UJQ2gR2', 'Marie', 'Stone', '5555555557', 'F', '1986-06-19', '34 Howard Dr', 'Hooksett',  
'PA', '03178', '2016-05-01 20:03:39'),  
('ncarroll7@drupal.org', 'M3Q8f9j', 'Nicole', 'Carroll', '5555555562', 'F', '1982-03-09', '34 Teepee Dr', 'Merrimack',  
'CO', '03104', '2016-05-01 20:03:48'),  
('sriley0@eventbrite.com', 'Hbo9HfMcz9I1', 'Susan', 'Riley', '5555555555', 'F', '1986-04-19', '34 Bank Dr',  
'Manchester', 'NH', '03102', '2016-05-01 20:03:35'),  
('fuller3@microsoft.com', 'cnDVU1RJx', 'Tammy', 'Fuller', '5555555558', 'F', '1985-10-19', '34 Tree Rd', 'Candia',  
'OH', '03136', '2016-05-01 20:03:43'),  
('twilliamson9@elegantthemes.com', 'BzmQiu', 'Todd', 'Williamson', '5555555564', 'M', '1986-08-19', '50 Poppins  
Dr', 'Nashua', 'NH', '03123', '2016-05-01 20:03:49');
```

--

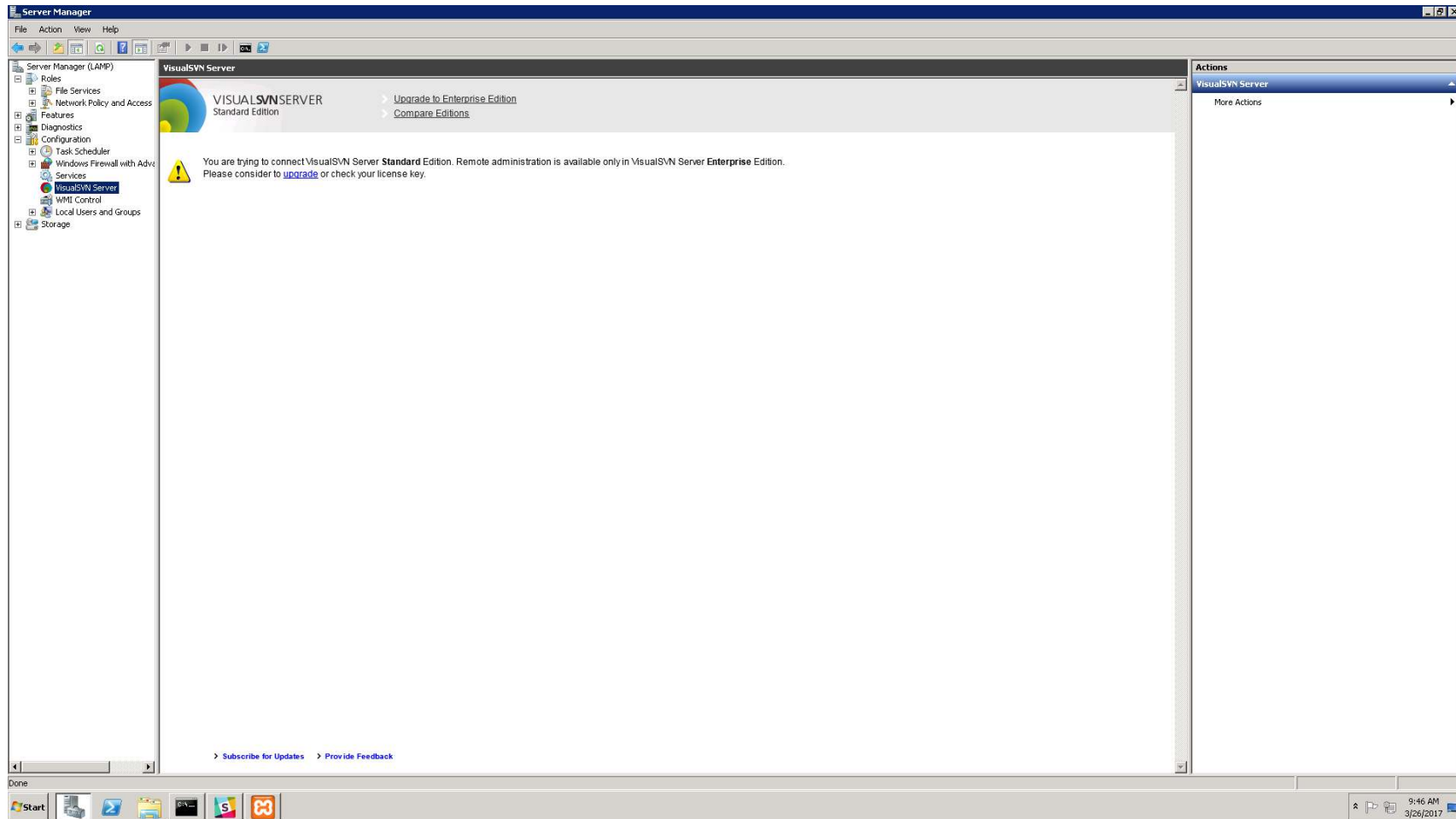
-- Test data: 'searches'

--

INSERT INTO searches (userName, firstName, lastName) VALUES

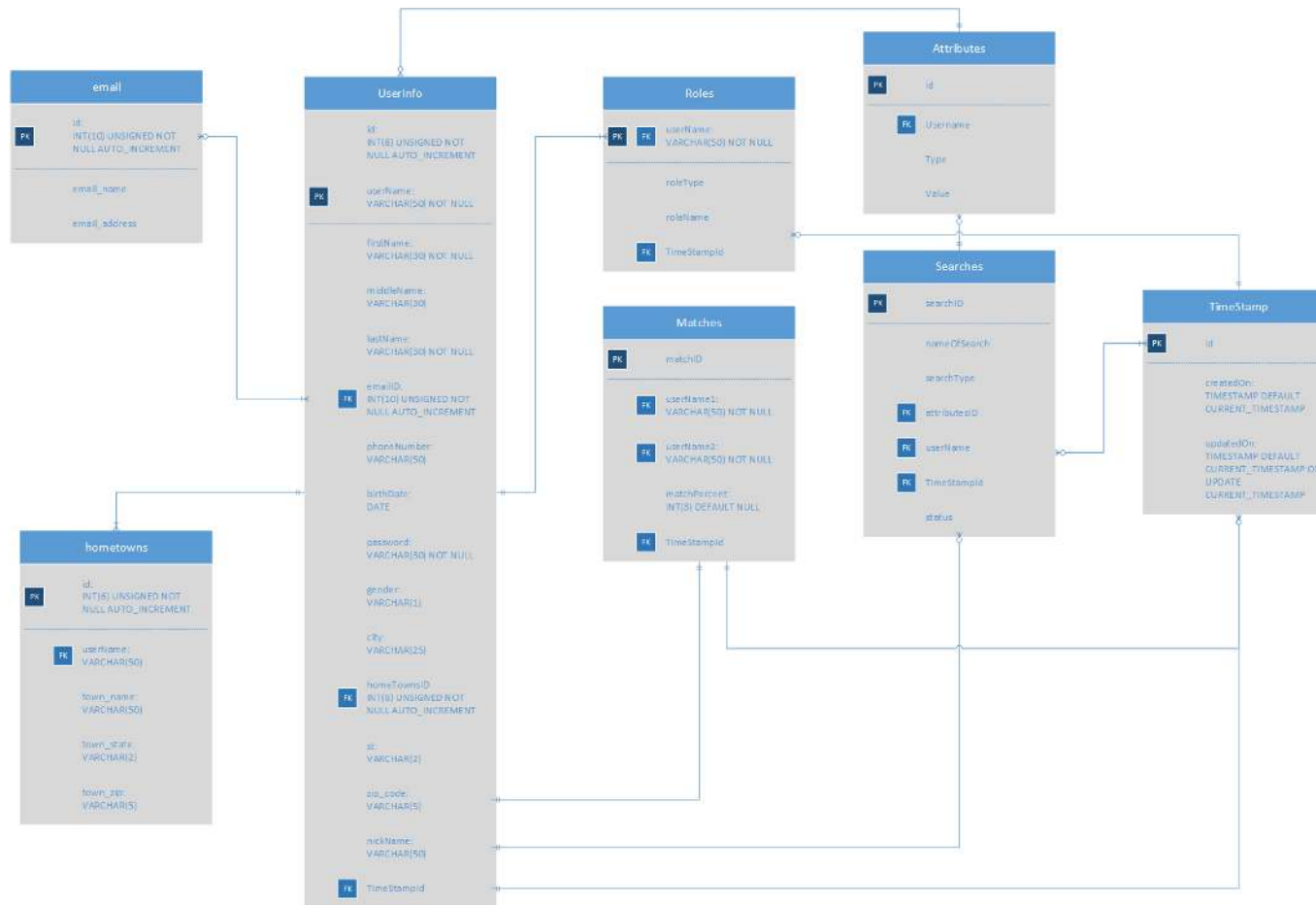
```
('sriley0@eventbrite.com', 'Lawrence', 'Burke'),  
('sriley0@eventbrite.com', 'Marie', 'Stone'),  
('lburke1@google.fr', 'Susan', 'Riley'),  
('lburke1@google.fr', 'Marie', 'Smith'),  
('lburke1@google.fr', 'John', 'Smith'),  
('mstone2@netscape.com', 'Lawrence', 'Burke'),  
('mstone2@netscape.com', 'Mike', 'Jonas'),  
('fuller3@microsoft.com', 'Susan', 'Riley'),  
('fuller3@microsoft.com', 'Susan', 'Burke'),  
('fuller3@microsoft.com', 'Adam', 'Price'),  
('bmills4@abcnews.com', 'Lawrence', 'Burke'),  
('bmills4@abcnews.com', 'Susan', 'Riley');
```

Issues (Week 7)



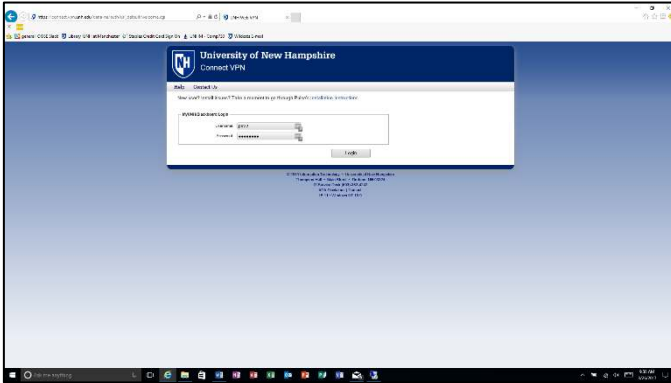
SVN Server says that it isn't licensed to work with Remote Desktop.

Additional Progress (Week 7)

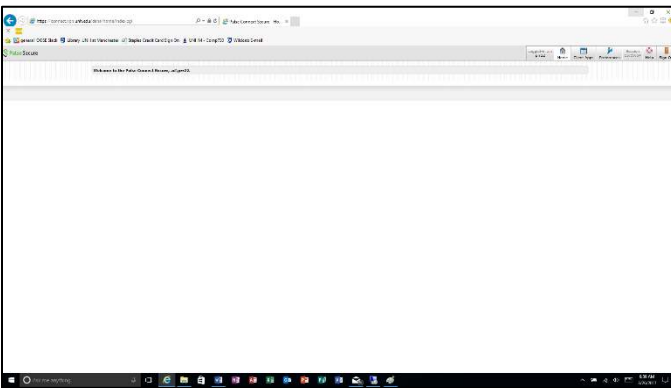


This shows progress towards making the new data model, combining what the past class did with what William J. needs for his Software configuration. There may be one or two additional tables that are needed to be added to this diagram, when we actually figure out how his works. i.e. Xrconfig

Connecting to Server (Week 7)



← connect.vpn.unh.edu



← VPN Connected Screen



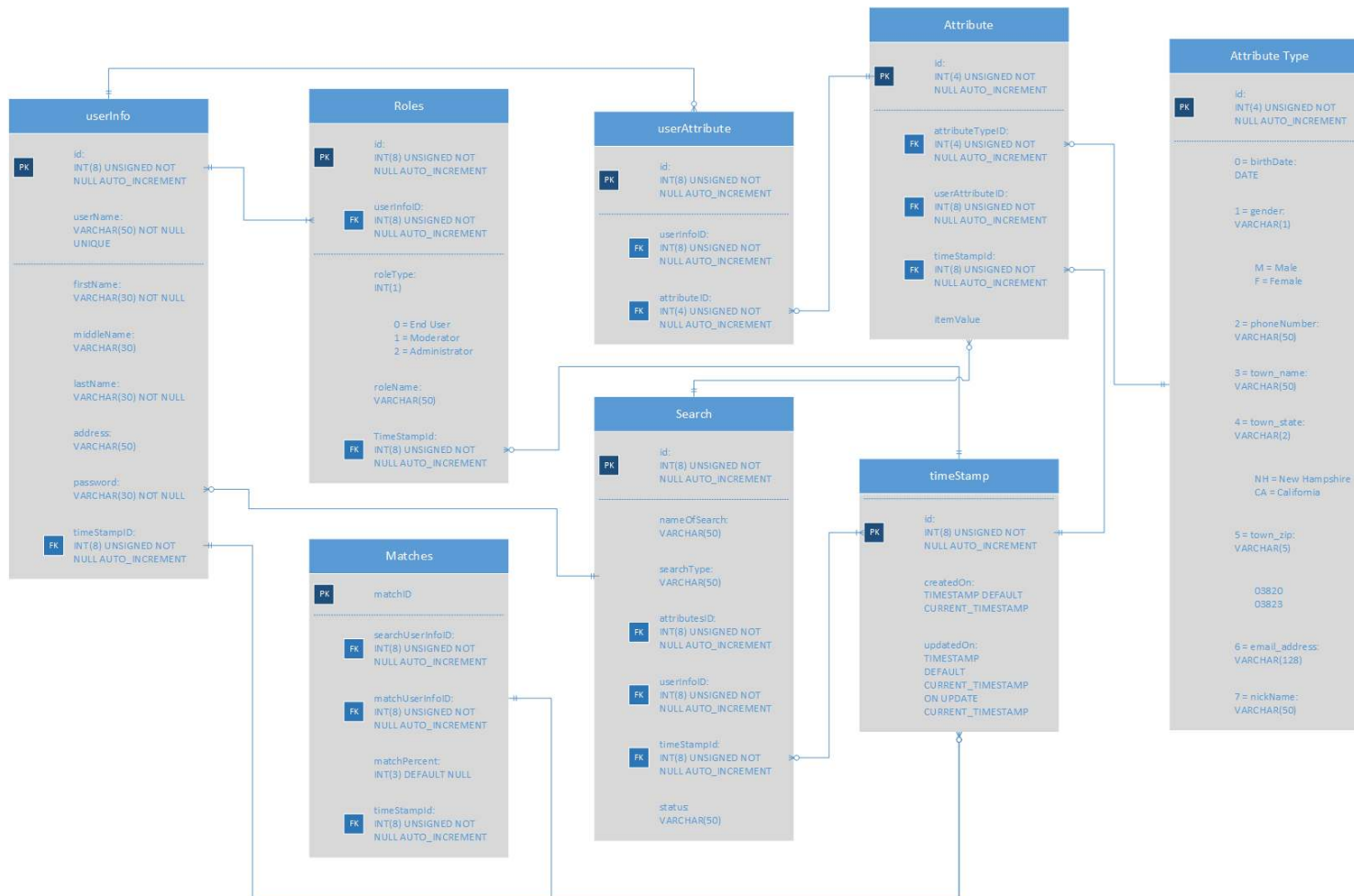
RDP → lamp.unh.edu

Computer: 132.177.188.84

User name: Administrator

Password: mj.US730

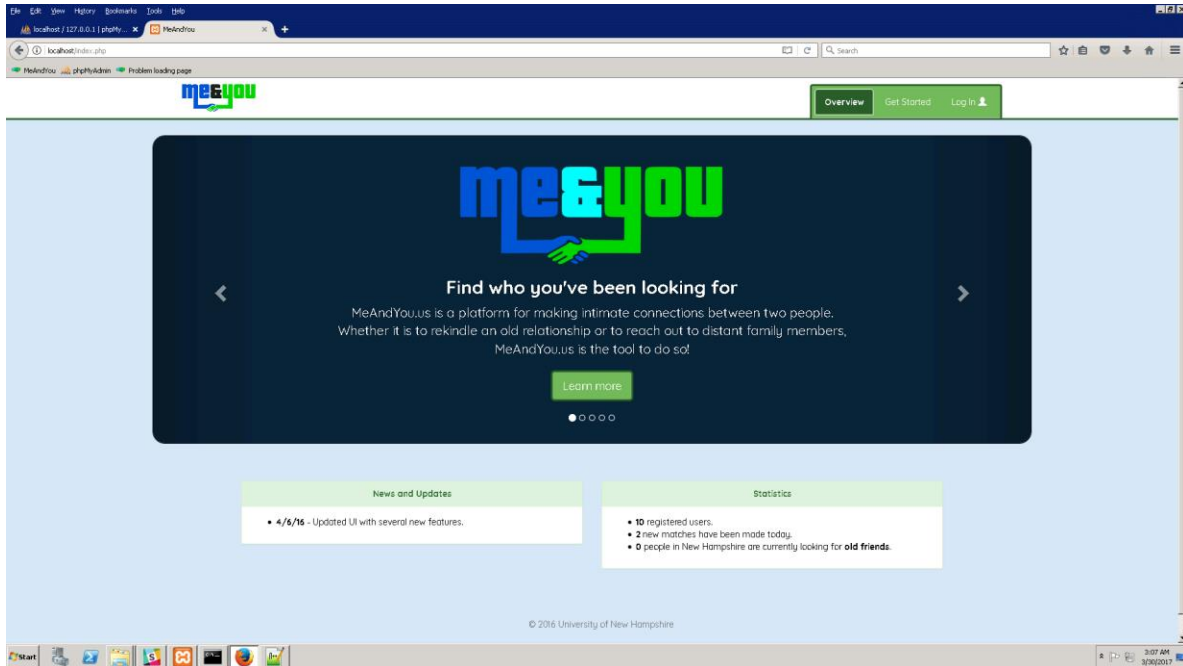
Data Model (Week 8)



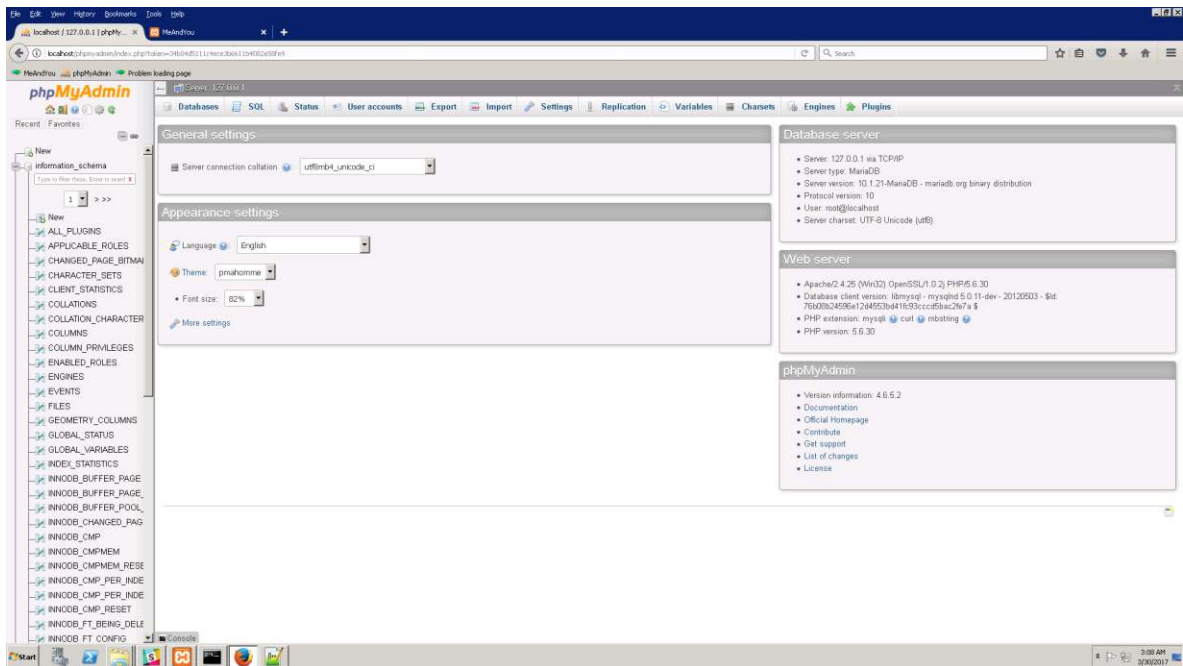
We finalized the data model in class on 3/29/2017, with input from William Jones (Software) and William Rivera (Front-end).

Need to generate SQL code.

XAMPP is Working (Week 8)



Website is working now, because Patrick fixed XAMPP.



phpMyAdmin is working now also.

Data Generator (Week 8)

The screenshot shows the generatedata.com web application interface. At the top, there is a logo with two green dice and the text "generatedata.com". To the right, there are links for "Login" and a language dropdown menu set to "English". Below the header, there are navigation tabs: "Generate" (active), "About", "News", and "Donate".

The main content area includes a large text input field for a custom data string, with a "SAVE" button and icons for calendar, refresh, and undo. Below this is a "COUNTRY-SPECIFIC DATA" section with a dropdown menu currently set to "All countries".

The "DATA SET" section features a table with columns: Order, Column Title, Data Type, Examples, Options, Help, and Del. There are four rows, each with a "Select Data Type" dropdown in the Data Type column. Below the table is an "Add" button with a text input set to "1" and a "Row(s)" button.

The "EXPORT TYPES" section has tabs for CSV, Excel, HTML (selected), JSON, LDIF, Programming Language, SQL, and XML. Below the tabs, there are radio buttons for "Data format" set to "<table>", and a checkbox for "Use custom HTML format" with an "edit:" button.

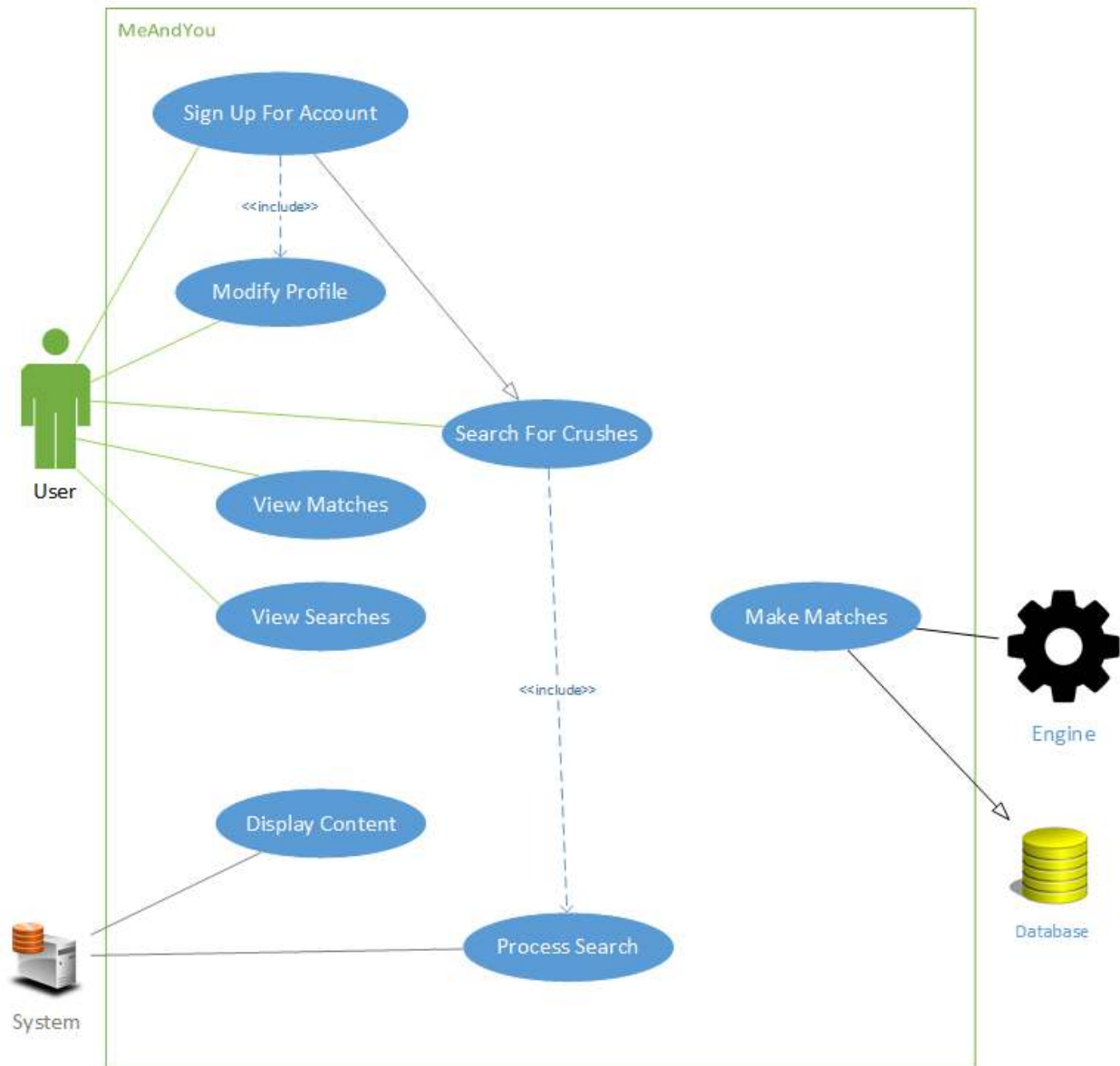
At the bottom, there is a green bar with a "Generate" button, a text input set to "100" rows, and radio buttons for "Generate in-page" (selected), "New window/tab", and "Prompt to download", along with a "Zip?" checkbox.

Footer text includes "3.2.4 | Documentation | Report a bug" with a bug icon.

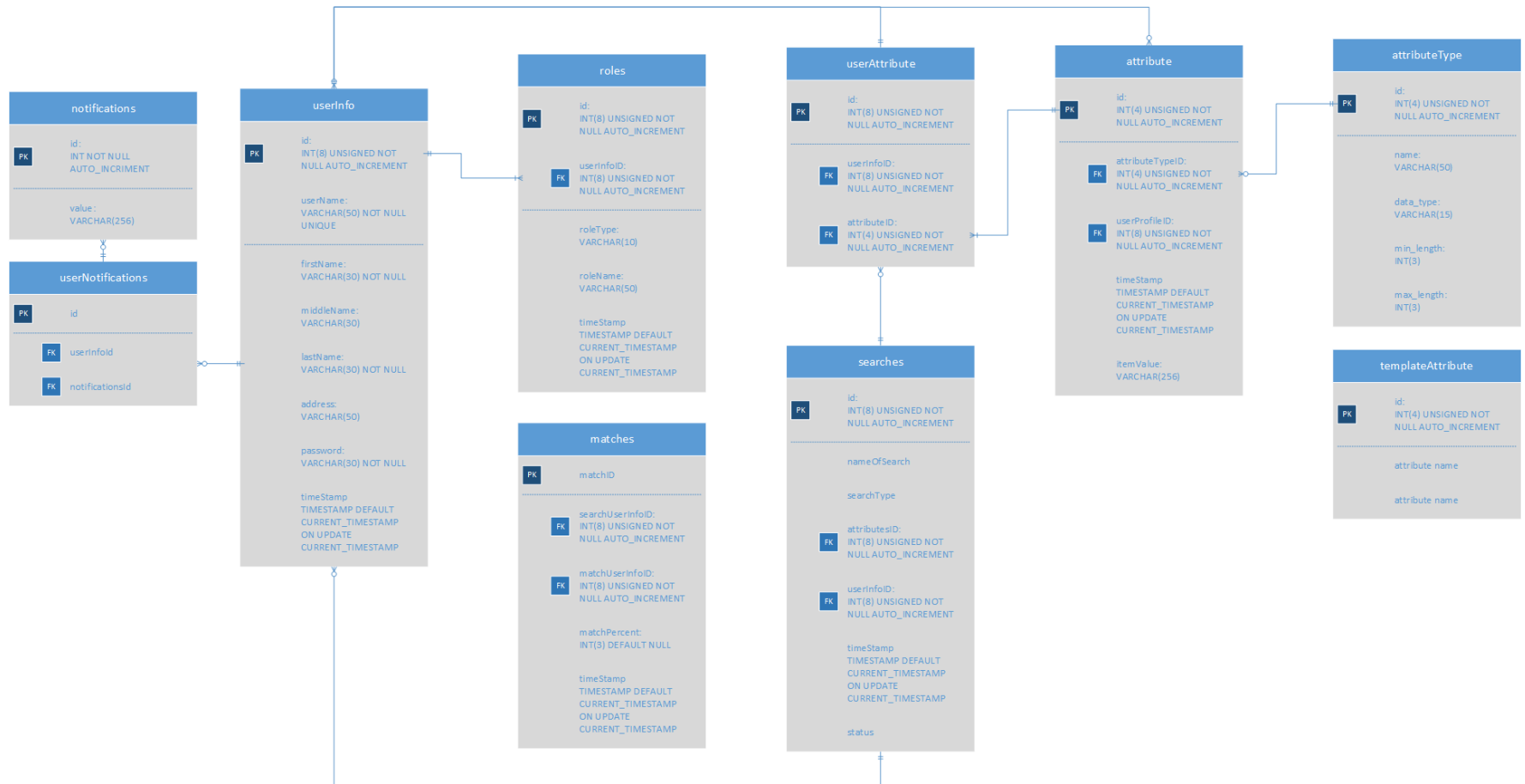
[REDACTED] has subscribed to datagenerator.com to generate our custom data that we need to test the system.

“Phase 0” - Use Case Diagram (Week 9)

Me&You – Prototype Use Case Diagram



Revised Crow's Feet Diagram (Week 9)



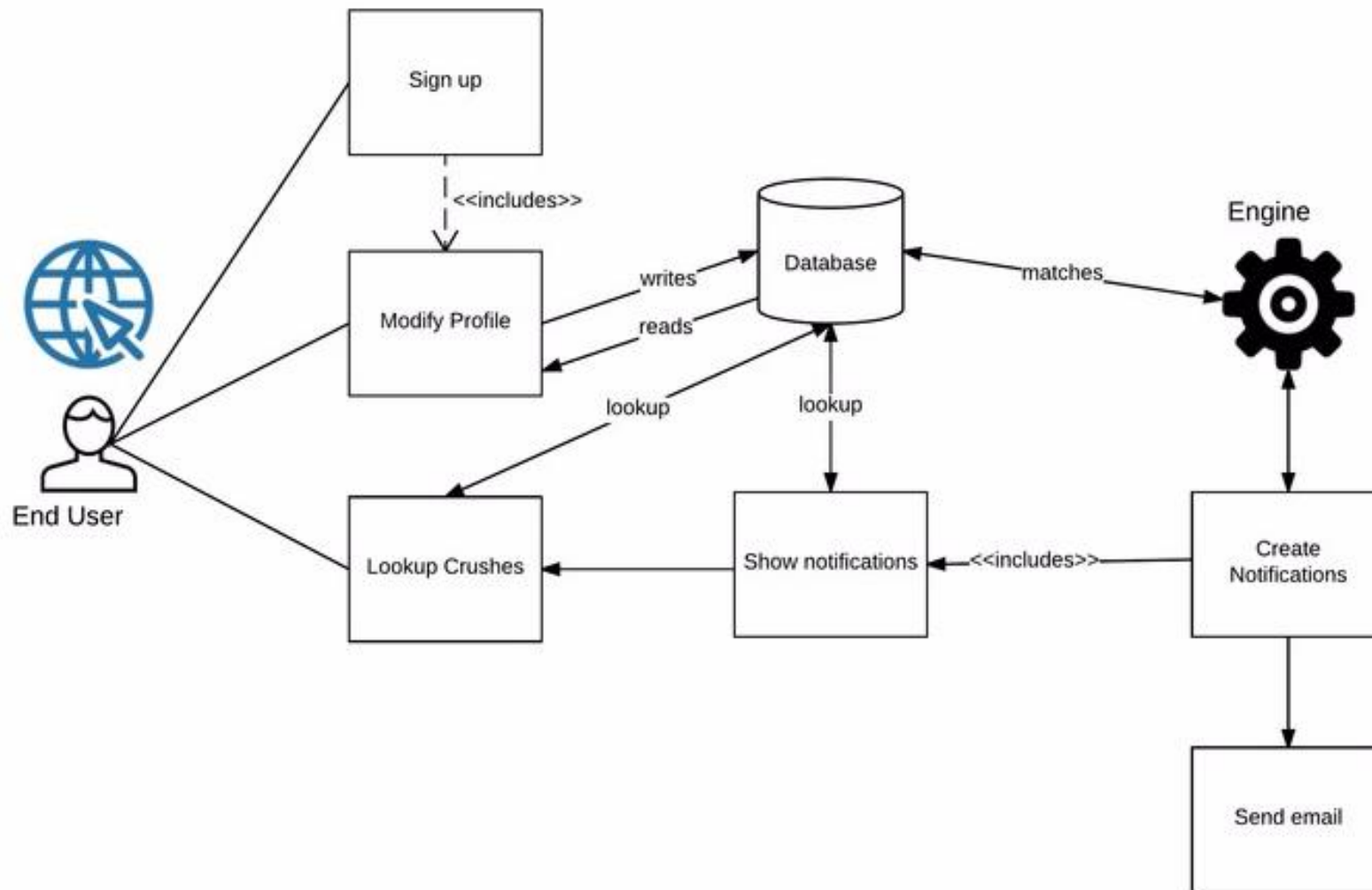
Revised SQL Code (Week 9)

```
--
-- Database: `meandyou`
--
-- Version: 2017-04-04 v7
-- Developer: Patrick R. McElhiney
-- Organization: MCE123 c/o University of New Hampshire, Manchester
DROP DATABASE MEANDYOU;
CREATE DATABASE MEANDYOU;
USE MEANDYOU;
SET time_zone = "-05:00";
CREATE TABLE userInfo (
  id MEDIUMINT NOT NULL AUTO_INCREMENT,
  userName VARCHAR(50) NOT NULL UNIQUE,
  firstName VARCHAR(30) NOT NULL,
  middleName VARCHAR(30),
  lastName VARCHAR(30) NOT NULL,
  address VARCHAR(50),
  password VARCHAR(30) NOT NULL,
  timeStamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (id)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
CREATE TABLE roles (
  id MEDIUMINT NOT NULL AUTO_INCREMENT,
  userInfoId MEDIUMINT NOT NULL,
  roleType INT(1) NOT NULL,
  roleName VARCHAR(50),
  timeStamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (id)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
CREATE TABLE matches (
  id MEDIUMINT NOT NULL AUTO_INCREMENT,
  searchUserInfoId MEDIUMINT NOT NULL,
  matchUserInfoId MEDIUMINT NOT NULL,
  matchPercent MEDIUMINT(3) DEFAULT NULL,
  timeStamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (id)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
CREATE TABLE attribute (
  id MEDIUMINT NOT NULL AUTO_INCREMENT,
  attributeTypeId MEDIUMINT NOT NULL,
  userAttributeId MEDIUMINT NOT NULL,
  timeStamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  itemValue VARCHAR(50),
  PRIMARY KEY (id)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE userAttribute (
  id MEDIUMINT NOT NULL AUTO_INCREMENT,
  userInfoId MEDIUMINT NOT NULL,
  attributeId MEDIUMINT NOT NULL,
  PRIMARY KEY (id)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
--
-- Foreign keys for table 'userAttribute'
--
ALTER TABLE userAttribute
```

```
ADD CONSTRAINT userAttributeuserInfoIdfk FOREIGN KEY (userInfoId) REFERENCES userInfo (id);
ALTER TABLE userAttribute
ADD CONSTRAINT userAttributeattributeIdfk FOREIGN KEY (attributeId) REFERENCES attribute (id);
CREATE TABLE searches (
  id MEDIUMINT NOT NULL AUTO_INCREMENT,
  nameOfSearch VARCHAR(50),
  searchType VARCHAR(50),
  attributeId MEDIUMINT NOT NULL,
  userInfoId MEDIUMINT NOT NULL,
  thisTimeStamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  thisstatus VARCHAR(50),
  PRIMARY KEY (id)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
CREATE TABLE attributeType (
  id MEDIUMINT NOT NULL AUTO_INCREMENT,
  itemName VARCHAR(50),
  itemType VARCHAR(50),
  PRIMARY KEY (id)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
CREATE TABLE time_Stamp (
  id MEDIUMINT NOT NULL AUTO_INCREMENT,
  updatedOn TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (id)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
--
-- Foreign keys for table 'roles'
--
ALTER TABLE roles
ADD CONSTRAINT rolesuserInfoIdfk FOREIGN KEY (userInfoId) REFERENCES userInfo (id);
--
-- Foreign keys for table 'matches'
--
ALTER TABLE matches
ADD CONSTRAINT matchessearchUserInfoIdfk FOREIGN KEY (searchUserInfoId) REFERENCES userInfo (id),
ADD CONSTRAINT matchesmatchUserInfoIdfk FOREIGN KEY (matchUserInfoId) REFERENCES userInfo (id);
--
-- Foreign keys for table 'searches'
--
ALTER TABLE searches
ADD CONSTRAINT searchesuserInfoIdfk FOREIGN KEY (userInfoId) REFERENCES userInfo (id),
ADD CONSTRAINT searchesattributeIdfk FOREIGN KEY (attributeId) REFERENCES attribute (id);
--
-- Foreign keys for table 'attribute'
--
ALTER TABLE attribute
ADD CONSTRAINT attributeattributeTypeIdfk FOREIGN KEY (attributeTypeId) REFERENCES attributeType (id),
ADD CONSTRAINT attributeuserInfoIdfk FOREIGN KEY (userInfoId) REFERENCES userInfo (id);
```

Case Flow Diagram (Week 9)



██████'s Requirements Analysis Doc. (Week 9)

Functional Requirements:

- Users must be able to register via a web browser.
- User dashboards are secure and a user must be logged in to perform a search.
- Users must be able to edit their profiles
- Single user with the same username or email address cannot create more than one profile.
- Users must be able to perform searches for crushes (users must know that person to some degree); searching for random people is not allowed.
- Users must be notified if a match is made.
- System must be able to support 20,000 users.

Authentication: Secured or encrypted connection and authentication is required for users. This includes unique username and password.

System Administration: The system admin will have access to database (users will not). The system admin can edit site content, menu items, and general configuration parameters; the system admin will be able to create, update, manage, and edit site (not user) content as desired. Administration and management of the website is done through a password-protected browser interface. This will be implemented in a future phase; it will not be implemented in Phase I.

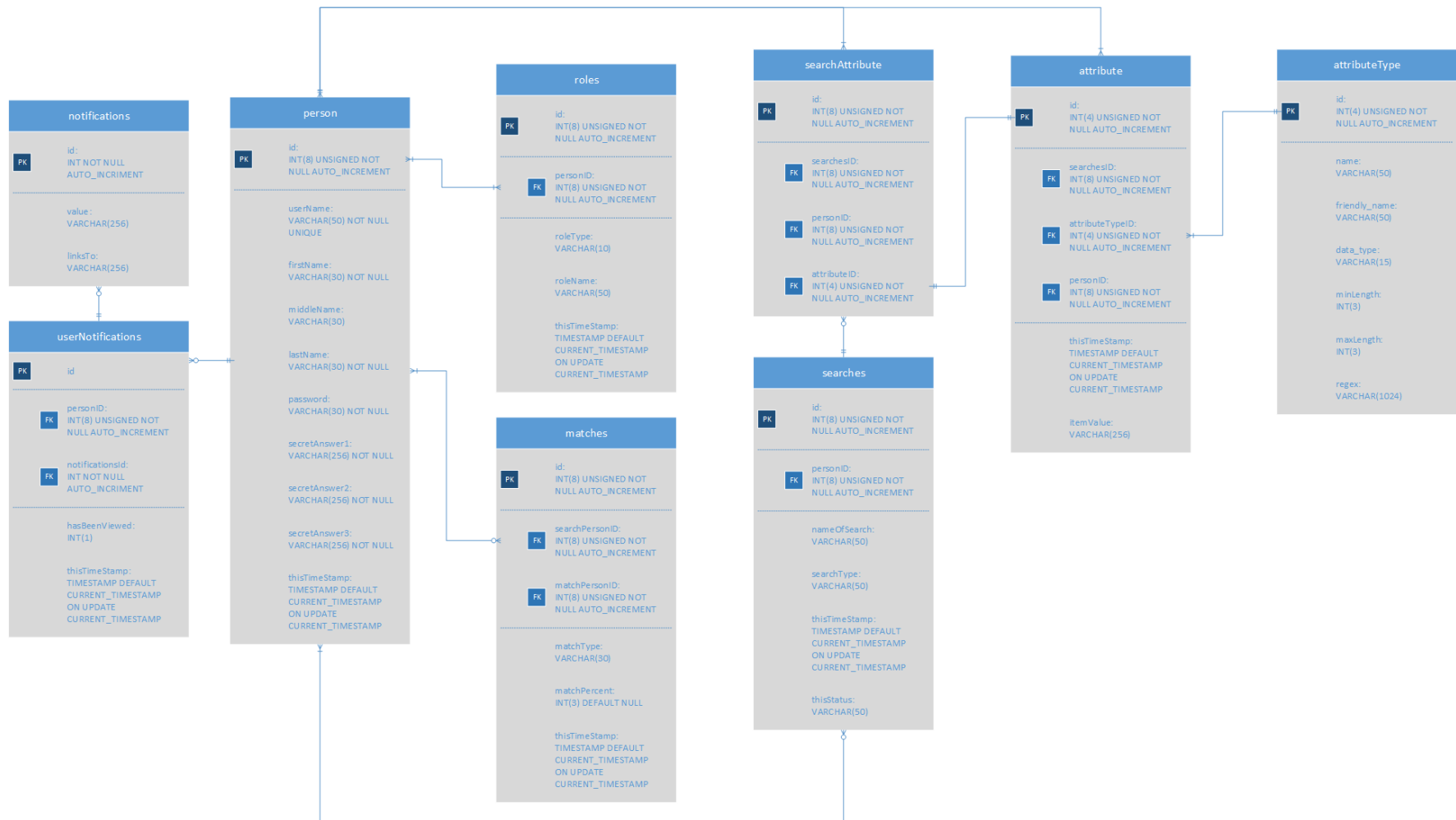
Database Functional Requirements:

- Information will be organized and stored in the database.
- The database will store modification flags (matches made).
- The database will store functional data needed for the system.
- The database must be scalable.
- Only complete data (registration and/or searches) will be stored.
- The search engine will search the database and attempt to make matches if they exist. If a match is made, it will reload/update the database.
- The search engine will search for close matches.

Non-functional requirements:

- The response time to return request to user should be minimal. (Note: “minimal” is not defined and must be revised.)
- The system will run locally on Windows Server 2008.
- The database will use a relational model.
- Search engine programming will be done in C#.

Data Model (Final Version?) (Week 10)



SQL Code (Week 10)

```
--
-- Database: 'meandyou2'
--
-- Version: 2017-04-06 v12
-- Developer: Patrick R. McElhiney
-- Organization: MCE123 c/o University of New Hampshire, Manchester
DROP DATABASE MEANDYOU2;
CREATE DATABASE MEANDYOU2;
USE MEANDYOU2;
SET time_zone = "-05:00";
CREATE TABLE notifications (
  id MEDIUMINT NOT NULL AUTO_INCREMENT,
  value VARCHAR(256),
  linksTo VARCHAR(256),
  PRIMARY KEY (id)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
CREATE TABLE userNotifications (
  id MEDIUMINT NOT NULL AUTO_INCREMENT,
  personId MEDIUMINT,
  notificationsId MEDIUMINT,
  hasBeenViewed INT(1),
  thisTimeStamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (id)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
CREATE TABLE person (
  id MEDIUMINT AUTO_INCREMENT,
  userName VARCHAR(50) NOT NULL UNIQUE,
  firstName VARCHAR(30) NOT NULL,
  middleName VARCHAR(30),
  lastName VARCHAR(30) NOT NULL,
  password VARCHAR(30) NOT NULL,
  secretAnswer1 VARCHAR(256),
  secretAnswer2 VARCHAR(256),
  secretAnswer3 VARCHAR(256),
  thisTimeStamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (id)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
CREATE TABLE roles (
  id MEDIUMINT NOT NULL AUTO_INCREMENT,
  personId MEDIUMINT NOT NULL,
  roleType INT(1) NOT NULL,
  roleName VARCHAR(50),
  thisTimeStamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (id)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
CREATE TABLE matches (
  id MEDIUMINT NOT NULL AUTO_INCREMENT,
  searchPersonId MEDIUMINT NOT NULL,
  matchPersonId MEDIUMINT NOT NULL,
  matchType VARCHAR(30),
  matchPercent MEDIUMINT(3) DEFAULT NULL,
  thisTimeStamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (id)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
CREATE TABLE searchAttribute (
  id MEDIUMINT NOT NULL AUTO_INCREMENT,
  searchesId MEDIUMINT DEFAULT NULL,
  personId MEDIUMINT NOT NULL,
  attributeId MEDIUMINT NOT NULL,
  PRIMARY KEY (id)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
CREATE TABLE searches (
  id MEDIUMINT NOT NULL AUTO_INCREMENT,
  personId MEDIUMINT NOT NULL,
  nameOfSearch VARCHAR(50),
  searchType VARCHAR(50),
  thisTimeStamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  thisstatus VARCHAR(50),
  PRIMARY KEY (id)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
CREATE TABLE attribute (
```

```

id MEDIUMINT NOT NULL AUTO_INCREMENT,
  searchesId MEDIUMINT,
attributeTypeId MEDIUMINT NOT NULL,
personId MEDIUMINT,
thisTimeStamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
itemValue VARCHAR(256),
PRIMARY KEY (id)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
CREATE TABLE attributeType (
  id MEDIUMINT NOT NULL AUTO_INCREMENT,
  name VARCHAR(50),
  friendly_name VARCHAR(50),
  data_type VARCHAR(50),
  minLength INT(3),
  maxLength INT(3),
  regex VARCHAR(1024),
PRIMARY KEY (id)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

INSERT INTO notifications (`id`, `value`, `linksTo`) VALUES
(1, 'Thank you for joining Me&You!', ''),
(2, 'Your Password Has Been Reset!', ''),
(3, 'You have a new match!', '/ViewMatchesPage.php'),
(4, 'You have no matches for your search. But we will keep your search active for 30 days.', '/ManageYourSearches.php'),
(5, 'Your search has expired! Please try another search.', '/StartANewSearch.php');
INSERT INTO userNotifications (`id`, `personId`, `notificationsId`, `hasBeenViewed`, `thisTimeStamp`) VALUES
(1, 1, 1, 0, '2016-05-01 20:03:47'),
(2, 2, 2, 0, '2016-05-01 20:03:47'),
(3, 3, 3, 0, '2016-05-01 20:03:47'),
(4, 4, 4, 0, '2016-05-01 20:03:47'),
(5, 5, 5, 1, '2016-05-01 20:03:47'),
(6, 6, 1, 1, '2016-05-01 20:03:47'),
(7, 7, 2, 1, '2016-05-01 20:03:47'),
(8, 8, 3, 1, '2016-05-01 20:03:47'),
(9, 9, 4, 0, '2016-05-01 20:03:47'),
(10, 10, 5, 0, '2016-05-01 20:03:47');
INSERT INTO person (`id`, `userName`, `firstName`, `middleName`, `lastName`, `password`, `secretAnswer1`, `secretAnswer2`, `secretAnswer3`, `thisTimeStamp`)
VALUES
(0, 'default@defaultprofile.com', 'Default', 'Default', 'Default', 'uqqFUG87ryrwe', 'Default123', 'Default456', 'Default789', '2016-05-01 20:03:47'),
(1, 'aprice5@wufoo.com', 'Adam', 'Birch', 'Price', '7rWjXR', 'Allentown', 'Pasta', 'Jennifer Lopez', '2016-05-01 20:03:47'),
(2, 'bmills4@nbcnews.com', 'Betty', 'Bee', 'Mills', 'oNeamSXqBA', 'Trexkertown', 'Pizza', 'Bill Clinton', '2016-05-01 20:03:47'),
(3, 'jfowler8@51.la', 'Joan', 'Bay', 'Fowler', 'bv4E969C', 'Hudson', 'Candy', 'Albert Einstein', '2016-05-01 20:03:47'),
(4, 'khunt6@newyorker.com', 'Keith', 'Beck', 'Hunt', 'fjiPPrZ1SUzB', 'Londonderry', 'Meatballs', 'Jennifer Lawrence', '2016-05-01 20:03:47'),
(5, 'lburke1@google.fr', 'Lawrence', 'Chan', 'Burke', 'iqNv9ciiOBg', 'Bedford', 'Cereal', 'Hillary Clinton', '2016-05-01 20:03:47'),
(6, 'mstone2@netscape.com', 'Marie', 'Blue', 'Stone', 'UJQ2gR2', 'Hooksett', 'ice cream', 'Paul Ryan', '2016-05-01 20:03:47'),
(7, 'ncarroll7@drupal.org', 'Nicole', 'Bree', 'Carroll', 'M3Q8f9j', 'Merrimack', 'CoOkIeS', 'Kittens', '2016-05-01 20:03:47'),
(8, 'sriley0@eventbrite.com', 'Susan', 'Bryn', 'Riley', 'Hbo9HfMcz9I1', 'Manchester', 'Indian Food', 'Elmo', '2016-05-01 20:03:47'),
(9, 'tfuller3@microsoft.com', 'Tammy', 'Doe', 'Fuller', 'cndVU1RjX', 'Candia', 'Chinese Food', 'Joeseeph Gemini Daddy 3.0', '2016-05-01 20:03:47'),
(10, 'twilliamson9@elegantthemes.com', 'Todd', 'Claude', 'Williamson', 'BzmQiu', 'Nashua', 'Hamburgers', 'Nancy Pelosi', '2016-05-01 20:03:47');
INSERT INTO roles (`id`, `personId`, `roleType`, `roleName`, `thisTimeStamp`) VALUES
(1, 1, 1, 'End User', '2016-05-01 20:03:47'),
(2, 2, 1, 'End User', '2016-05-01 20:03:47'),
(3, 3, 1, 'End User', '2016-05-01 20:03:47'),
(4, 4, 1, 'End User', '2016-05-01 20:03:47'),
(5, 5, 1, 'End User', '2016-05-01 20:03:47'),
(6, 6, 1, 'End User', '2016-05-01 20:03:47'),
(7, 7, 1, 'End User', '2016-05-01 20:03:47'),
(8, 8, 1, 'End User', '2016-05-01 20:03:47'),
(9, 9, 2, 'Moderator', '2016-05-01 20:03:47'),
(10, 10, 3, 'System Administrator', '2016-05-01 20:03:47');
INSERT INTO matches (`id`, `searchPersonId`, `matchPersonId`, `matchType`, `matchPercent`, `thisTimeStamp`) VALUES
(1, 10, 9, 'Crush', 85, '2016-05-01 20:03:47'),
(2, 9, 10, 'Crush', 85, '2016-05-01 20:03:47'),
(3, 8, 5, 'Crush', 90, '2016-05-01 20:03:47'),
(4, 5, 8, 'Crush', 90, '2016-05-01 20:03:47'),
(5, 1, 2, 'Crush', 100, '2016-05-01 20:03:47'),
(6, 2, 1, 'Crush', 100, '2016-05-01 20:03:47');
INSERT INTO searchAttribute (`id`, `searchesId`, `personId`, `attributeId`) VALUES
(1, 1, 1, 71),
(2, 2, 1, 72),
(3, 3, 1, 73),
(4, 4, 1, 74),
(5, 5, 1, 75),
(6, 6, 1, 76),
(7, 7, 1, 77),
(8, 8, 1, 78),
(9, 9, 1, 79),
(10, 10, 1, 80),
(11, 1, 2, 81),
(12, 2, 2, 82),

```

```

(13', '3', '2', '83),
(14', '4', '2', '84),
(15', '5', '2', '85),
(16', '6', '2', '86),
(17', '7', '2', '87),
(18', '8', '2', '88),
(19', '9', '2', '89),
(20', '10', '2', '90'),
(21', '1', '3', '91),
(22', '2', '3', '92),
(23', '3', '3', '93),
(24', '4', '3', '94),
(25', '5', '3', '95),
(26', '6', '3', '96),
(27', '7', '3', '97),
(28', '8', '3', '98),
(29', '9', '3', '99),
(30', '10', '3', '100'),
(31', '1', '4', '101),
(32', '2', '4', '102),
(33', '3', '4', '103'),
(34', '4', '4', '104'),
(35', '5', '4', '105'),
(36', '6', '4', '106'),
(37', '7', '4', '107'),
(38', '8', '4', '108'),
(39', '9', '4', '109'),
(40', '10', '4', '110');
INSERT INTO searches (`id`, `nameOfSearch`, `searchType`, `personId`, `thisTimeStamp`, `thisStatus`) VALUES
(0, 'Default Search', 'Default', '0', '2016-05-01 20:03:47', 'Complete'),
(1, 'Adam Price is hot.', 'Crush', '1', '2016-05-01 20:03:47', 'Complete'),
(2, 'Betty Mills is hot.', 'Crush', '2', '2016-05-01 20:03:47', 'Complete'),
(3, 'Joan Fowler is hot.', 'Crush', '3', '2016-05-01 20:03:47', 'Complete'),
(4, 'Keith Hunt is hot.', 'Crush', '4', '2016-05-01 20:03:47', 'Complete'),
(5, 'Lawrence Burke is hot.', 'Crush', '5', '2016-05-01 20:03:47', 'Complete'),
(6, 'Marie Stone is hot.', 'Crush', '6', '2016-05-01 20:03:47', 'Complete'),
(7, 'Nicole Carroll is hot.', 'Crush', '7', '2016-05-01 20:03:47', 'Complete'),
(8, 'Susan Riley is hot.', 'Crush', '8', '2016-05-01 20:03:47', 'Complete'),
(9, 'Tammy Fuller is hot.', 'Crush', '9', '2016-05-01 20:03:47', 'Complete'),
(10, 'Todd Williamson is hot.', 'Crush', '10', '2016-05-01 20:03:47', 'Complete');
INSERT INTO attribute (`id`, `searchesId`, `attributeTypeId`, `personId`, `thisTimeStamp`, `itemValue`) VALUES
(1, '0', '1', '1', '2016-05-01 20:03:47', '1986-06-18'),
(2, '0', '1', '2', '2016-05-01 20:03:47', '1983-07-15'),
(3, '0', '1', '3', '2016-05-01 20:03:47', '1984-11-19'),
(4, '0', '1', '4', '2016-05-01 20:03:47', '1989-01-22'),
(5, '0', '1', '5', '2016-05-01 20:03:47', '1986-05-19'),
(6, '0', '1', '6', '2016-05-01 20:03:47', '1986-06-19'),
(7, '0', '1', '7', '2016-05-01 20:03:47', '1982-03-09'),
(8, '0', '1', '8', '2016-05-01 20:03:47', '1986-04-19'),
(9, '0', '1', '9', '2016-05-01 20:03:47', '1985-10-19'),
(10, '0', '1', '10', '2016-05-01 20:03:47', '1986-08-19'),
(11, '0', '2', '1', '2016-05-01 20:03:47', 'M'),
(12, '0', '2', '2', '2016-05-01 20:03:47', 'F'),
(13, '0', '2', '3', '2016-05-01 20:03:47', 'F'),
(14, '0', '2', '4', '2016-05-01 20:03:47', 'M'),
(15, '0', '2', '5', '2016-05-01 20:03:47', 'M'),
(16, '0', '2', '6', '2016-05-01 20:03:47', 'F'),
(17, '0', '2', '7', '2016-05-01 20:03:47', 'F'),
(18, '0', '2', '8', '2016-05-01 20:03:47', 'F'),
(19, '0', '2', '9', '2016-05-01 20:03:47', 'F'),
(20, '0', '2', '10', '2016-05-01 20:03:47', 'M'),
(21, '0', '3', '1', '2016-05-01 20:03:47', '5555555560'),
(22, '0', '3', '2', '2016-05-01 20:03:47', '5555555559'),
(23, '0', '3', '3', '2016-05-01 20:03:47', '5555555563'),
(24, '0', '3', '4', '2016-05-01 20:03:47', '5555555561'),
(25, '0', '3', '5', '2016-05-01 20:03:47', '5555555556'),
(26, '0', '3', '6', '2016-05-01 20:03:47', '5555555557'),
(27, '0', '3', '7', '2016-05-01 20:03:47', '5555555562'),
(28, '0', '3', '8', '2016-05-01 20:03:47', '5555555555'),
(29, '0', '3', '9', '2016-05-01 20:03:47', '5555555558'),
(30, '0', '3', '10', '2016-05-01 20:03:47', '5555555564'),
(31, '0', '4', '1', '2016-05-01 20:03:47', '34 Boon Dr'),
(32, '0', '4', '2', '2016-05-01 20:03:47', '34 Fort Way'),
(33, '0', '4', '3', '2016-05-01 20:03:47', '32 Grass St'),
(34, '0', '4', '4', '2016-05-01 20:03:47', '34 Frank St'),
(35, '0', '4', '5', '2016-05-01 20:03:47', '34 City Rd'),
(36, '0', '4', '6', '2016-05-01 20:03:47', '34 Howard Dr'),
(37, '0', '4', '7', '2016-05-01 20:03:47', '34 Teepee Dr'),
(38, '0', '4', '8', '2016-05-01 20:03:47', '34 Bank Dr'),
(39, '0', '4', '9', '2016-05-01 20:03:47', '34 Tree Rd'),

```

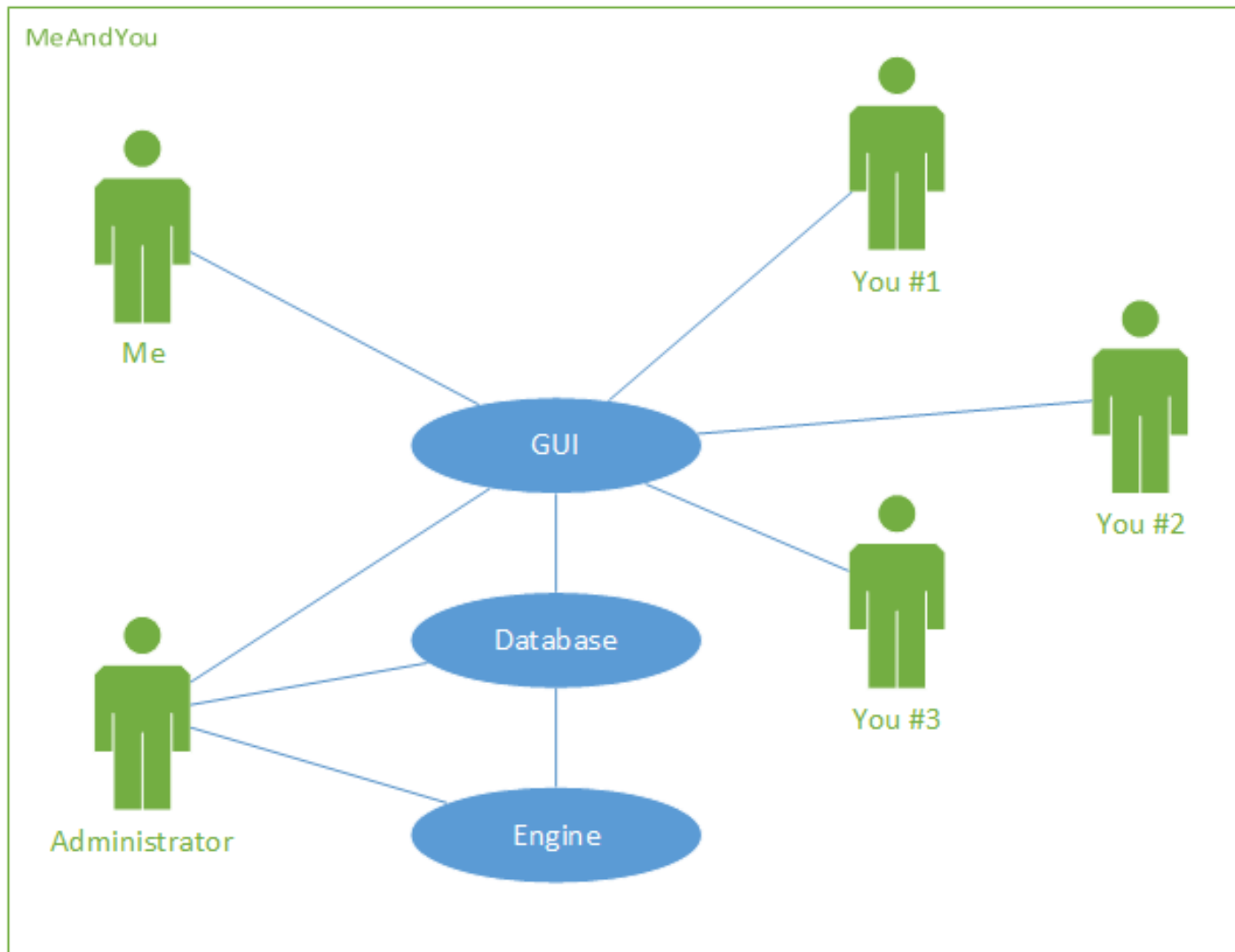

(40', '0', '4', '10', '2016-05-01 20:03:47', '50 Poppins Dr'),
(41', '0', '5', '1', '2016-05-01 20:03:47', 'Allentown'),
(42', '0', '5', '2', '2016-05-01 20:03:47', 'Trexlerstown'),
(43', '0', '5', '3', '2016-05-01 20:03:47', 'Hudson'),
(44', '0', '5', '4', '2016-05-01 20:03:47', 'Londonderry'),
(45', '0', '5', '5', '2016-05-01 20:03:47', 'Bedford'),
(46', '0', '5', '6', '2016-05-01 20:03:47', 'Hooksett'),
(47', '0', '5', '7', '2016-05-01 20:03:47', 'Merrimack'),
(48', '0', '5', '8', '2016-05-01 20:03:47', 'Manchester'),
(49', '0', '5', '9', '2016-05-01 20:03:47', 'Candia'),
(50', '0', '5', '10', '2016-05-01 20:03:47', 'Nashua'),
(51', '0', '6', '1', '2016-05-01 20:03:47', 'NH'),
(52', '0', '6', '2', '2016-05-01 20:03:47', 'GA'),
(53', '0', '6', '3', '2016-05-01 20:03:47', 'NH'),
(54', '0', '6', '4', '2016-05-01 20:03:47', 'VA'),
(55', '0', '6', '5', '2016-05-01 20:03:47', 'MA'),
(56', '0', '6', '6', '2016-05-01 20:03:47', 'PA'),
(57', '0', '6', '7', '2016-05-01 20:03:47', 'CO'),
(58', '0', '6', '8', '2016-05-01 20:03:47', 'NH'),
(59', '0', '6', '9', '2016-05-01 20:03:47', 'OH'),
(60', '0', '6', '10', '2016-05-01 20:03:47', 'NH'),
(61', '0', '7', '1', '2016-05-01 20:03:47', '03107'),
(62', '0', '7', '2', '2016-05-01 20:03:47', '03101'),
(63', '0', '7', '3', '2016-05-01 20:03:47', '03172'),
(64', '0', '7', '4', '2016-05-01 20:03:47', '03109'),
(65', '0', '7', '5', '2016-05-01 20:03:47', '03134'),
(66', '0', '7', '6', '2016-05-01 20:03:47', '03178'),
(67', '0', '7', '7', '2016-05-01 20:03:47', '03104'),
(68', '0', '7', '8', '2016-05-01 20:03:47', '03102'),
(69', '0', '7', '9', '2016-05-01 20:03:47', '03136'),
(70', '0', '7', '10', '2016-05-01 20:03:47', '03123'),
(71', '0', '1', '0', '2016-05-01 20:03:47', '1986-06-18'),
(72', '1', '1', '0', '2016-05-01 20:03:47', '1983-07-15'),
(73', '2', '1', '0', '2016-05-01 20:03:47', '1984-11-19'),
(74', '3', '1', '0', '2016-05-01 20:03:47', '1989-01-22'),
(75', '4', '1', '0', '2016-05-01 20:03:47', '1986-05-19'),
(76', '5', '1', '0', '2016-05-01 20:03:47', '1986-06-19'),
(77', '6', '1', '0', '2016-05-01 20:03:47', '1982-03-09'),
(78', '7', '1', '0', '2016-05-01 20:03:47', '1986-04-19'),
(79', '8', '1', '0', '2016-05-01 20:03:47', '1985-10-19'),
(80', '9', '1', '0', '2016-05-01 20:03:47', '1986-08-19'),
(81', '0', '2', '0', '2016-05-01 20:03:47', 'M'),
(82', '1', '2', '0', '2016-05-01 20:03:47', 'F'),
(83', '2', '2', '0', '2016-05-01 20:03:47', 'F'),
(84', '3', '2', '0', '2016-05-01 20:03:47', 'M'),
(85', '4', '2', '0', '2016-05-01 20:03:47', 'M'),
(86', '5', '2', '0', '2016-05-01 20:03:47', 'F'),
(87', '6', '2', '0', '2016-05-01 20:03:47', 'F'),
(88', '7', '2', '0', '2016-05-01 20:03:47', 'F'),
(89', '8', '2', '0', '2016-05-01 20:03:47', 'F'),
(90', '9', '2', '0', '2016-05-01 20:03:47', 'M'),
(91', '0', '3', '0', '2016-05-01 20:03:47', '555555560'),
(92', '1', '3', '0', '2016-05-01 20:03:47', '555555559'),
(93', '2', '3', '0', '2016-05-01 20:03:47', '555555563'),
(94', '3', '3', '0', '2016-05-01 20:03:47', '555555561'),
(95', '4', '3', '0', '2016-05-01 20:03:47', '555555556'),
(96', '5', '3', '0', '2016-05-01 20:03:47', '555555557'),
(97', '6', '3', '0', '2016-05-01 20:03:47', '555555562'),
(98', '7', '3', '0', '2016-05-01 20:03:47', '555555555'),
(99', '8', '3', '0', '2016-05-01 20:03:47', '555555558'),
(100', '9', '3', '0', '2016-05-01 20:03:47', '555555564'),
(101', '0', '4', '0', '2016-05-01 20:03:47', '34 Boon Dr'),
(102', '1', '4', '0', '2016-05-01 20:03:47', '34 Fort Way'),
(103', '2', '4', '0', '2016-05-01 20:03:47', '32 Grass St'),
(104', '3', '4', '0', '2016-05-01 20:03:47', '34 Frank St'),
(105', '4', '4', '0', '2016-05-01 20:03:47', '34 City Rd'),
(106', '5', '4', '0', '2016-05-01 20:03:47', '34 Howard Dr'),
(107', '6', '4', '0', '2016-05-01 20:03:47', '34 Teepee Dr'),
(108', '7', '4', '0', '2016-05-01 20:03:47', '34 Bank Dr'),
(109', '8', '4', '0', '2016-05-01 20:03:47', '34 Tree Rd'),
(110', '9', '4', '0', '2016-05-01 20:03:47', '50 Poppins Dr'),
(111', '0', '5', '0', '2016-05-01 20:03:47', 'Allentown'),
(112', '1', '5', '0', '2016-05-01 20:03:47', 'Trexlerstown'),
(113', '2', '5', '0', '2016-05-01 20:03:47', 'Hudson'),
(114', '3', '5', '0', '2016-05-01 20:03:47', 'Londonderry'),
(115', '4', '5', '0', '2016-05-01 20:03:47', 'Bedford'),
(116', '5', '5', '0', '2016-05-01 20:03:47', 'Hooksett'),
(117', '6', '5', '0', '2016-05-01 20:03:47', 'Merrimack'),
(118', '7', '5', '0', '2016-05-01 20:03:47', 'Manchester'),
(119', '8', '5', '0', '2016-05-01 20:03:47', 'Candia'),

```

(120', '9', '5', '0', '2016-05-01 20:03:47', 'Nashua'),
(121', '0', '6', '0', '2016-05-01 20:03:47', 'NH'),
(122', '1', '6', '0', '2016-05-01 20:03:47', 'GA'),
(123', '2', '6', '0', '2016-05-01 20:03:47', 'NH'),
(124', '3', '6', '0', '2016-05-01 20:03:47', 'VA'),
(125', '4', '6', '0', '2016-05-01 20:03:47', 'MA'),
(126', '5', '6', '0', '2016-05-01 20:03:47', 'PA'),
(127', '6', '6', '0', '2016-05-01 20:03:47', 'CO'),
(128', '7', '6', '0', '2016-05-01 20:03:47', 'NH'),
(129', '8', '6', '0', '2016-05-01 20:03:47', 'OH'),
(130', '9', '6', '0', '2016-05-01 20:03:47', 'NH'),
(131', '0', '7', '0', '2016-05-01 20:03:47', '03107'),
(132', '1', '7', '0', '2016-05-01 20:03:47', '03101'),
(133', '2', '7', '0', '2016-05-01 20:03:47', '03172'),
(134', '3', '7', '0', '2016-05-01 20:03:47', '03109'),
(135', '4', '7', '0', '2016-05-01 20:03:47', '03134'),
(136', '5', '7', '0', '2016-05-01 20:03:47', '03178'),
(137', '6', '7', '0', '2016-05-01 20:03:47', '03104'),
(138', '7', '7', '0', '2016-05-01 20:03:47', '03102'),
(139', '8', '7', '0', '2016-05-01 20:03:47', '03136'),
(140', '9', '7', '0', '2016-05-01 20:03:47', '03123');
INSERT INTO attributeType (id, `name`, `friendly_name`, `data_type`, `minLength`, `maxLength`) VALUES
(1, 'birthdate', 'Birthday', 'DATE', 6, 10),
(2, 'gender', 'Gender', 'VARCHAR(1)', 0, 1),
(3, 'phoneNumber', 'Phone Number', 'VARCHAR(50)', 10, 50),
(4, 'streetAddress', 'Street Address', 'VARCHAR(100)', 5, 100),
(5, 'town', 'City / Town', 'VARCHAR(50)', 3, 50),
(6, 'st', 'State', 'VARCHAR(2)', 0, 2),
(7, 'zipCode', 'Zip Code', 'VARCHAR(15)', 0, 15),
(8, 'email', 'E-Mail Address', 'VARCHAR(50)', 0, 50),
(9, 'nickName', 'Nickname', 'VARCHAR(50)', 0, 50),
(10, 'firstName', 'First Name', 'VARCHAR(30)', 0, 30),
(11, 'middleName', 'Middle Name', 'VARCHAR(30)', 0, 30),
(12, 'lastName', 'Last Name', 'VARCHAR(30)', 0, 30);
--
-- Foreign keys for table 'userNotifications'
--
ALTER TABLE userNotifications
  ADD CONSTRAINT userNotificationsPersonIdfk FOREIGN KEY (personId) REFERENCES person (id),
  ADD CONSTRAINT userNotificationsNotificationsIdfk FOREIGN KEY (notificationsId) REFERENCES notifications (id);
--
-- Foreign keys for table 'roles'
--
ALTER TABLE roles
  ADD CONSTRAINT rolesPersonIdfk FOREIGN KEY (personId) REFERENCES person (id);
--
-- Foreign keys for table 'matches'
--
ALTER TABLE matches
  ADD CONSTRAINT matchesSearchPersonIdfk FOREIGN KEY (searchPersonId) REFERENCES person (id),
  ADD CONSTRAINT matchesMatchPersonIdfk FOREIGN KEY (matchPersonId) REFERENCES person (id);
--
-- Foreign keys for table 'searchAttribute'
--
ALTER TABLE searchAttribute
  ADD CONSTRAINT searchAttributePersonIdfk FOREIGN KEY (personId) REFERENCES person (id),
  ADD CONSTRAINT searchAttributeAttributeIdfk FOREIGN KEY (attributeId) REFERENCES attribute (id),
  ADD CONSTRAINT searchAttributeSearchesIdfk FOREIGN KEY (searchesId) REFERENCES searches (id);
--
-- Foreign keys for table 'searches'
--
ALTER TABLE searches
  ADD CONSTRAINT searchesPersonIdfk FOREIGN KEY (personId) REFERENCES person (id);
--
-- Foreign keys for table 'attribute'
--
ALTER TABLE attribute
  ADD CONSTRAINT attributeSearchesIdfk FOREIGN KEY (searchesId) REFERENCES searches (id),
  ADD CONSTRAINT attributePersonIdfk FOREIGN KEY (personId) REFERENCES person (id),
  ADD CONSTRAINT attributeAttributeTypeIdfk FOREIGN KEY (attributeTypeId) REFERENCES attributeType (id);

```

Simple Use Case Diagram (Week 10)



4/6/2017 Notes (Week 10)

The data model has been updated based on all changes that were requested. We realized that we were not properly documenting all of the changes to each version of the SQL / Data Model, so we may have to go back and make documentation detailing the changes that we made in each version, and there are probably at least 20 different versions. The reason why we didn't document the changes was because we made them mostly in class and from meetings that were recorded, so we thought it would essentially be overkill to document all of the changes again, since everything was already being recorded.

Next we will be working on many different aspects of documentation, and getting the database running on the server. Note that we changed the name of the database to MEANDYOU2, so that we can keep the old database running, so we will still be able to use the demo that last year's class made for us.

Patrick R. McElhiney made most of the changes to the data model and the SQL code based on input from others, and he did all of the coding and diagramming. Next he will be working on creating all of the necessary queries and insert / update statements that Front-End needs to access the database. But at this point it's just good that we have a working data model that we can implement on the server, and we can get that running so other groups can begin to query the database.

██████████ has been doing a lot of work on the Wiki updates, and ██████████ is going to work on the Master Attributes list and post it to the Wiki. ██████████ is working on some diagrams that we need – Activity Diagrams, and ██████████ is helping her with that. ██████████ is working on updating the data generated into the format that is needed for the database. Hopefully he understands the data model, however we will touch basis at the meeting tomorrow at 4PM, and also next week to make sure.

We also created a Things to Do document, in which each person has been assigned tasks for the upcoming weeks. Patrick will fill in wherever we get behind, as he has more time to spend on the project than everyone else.

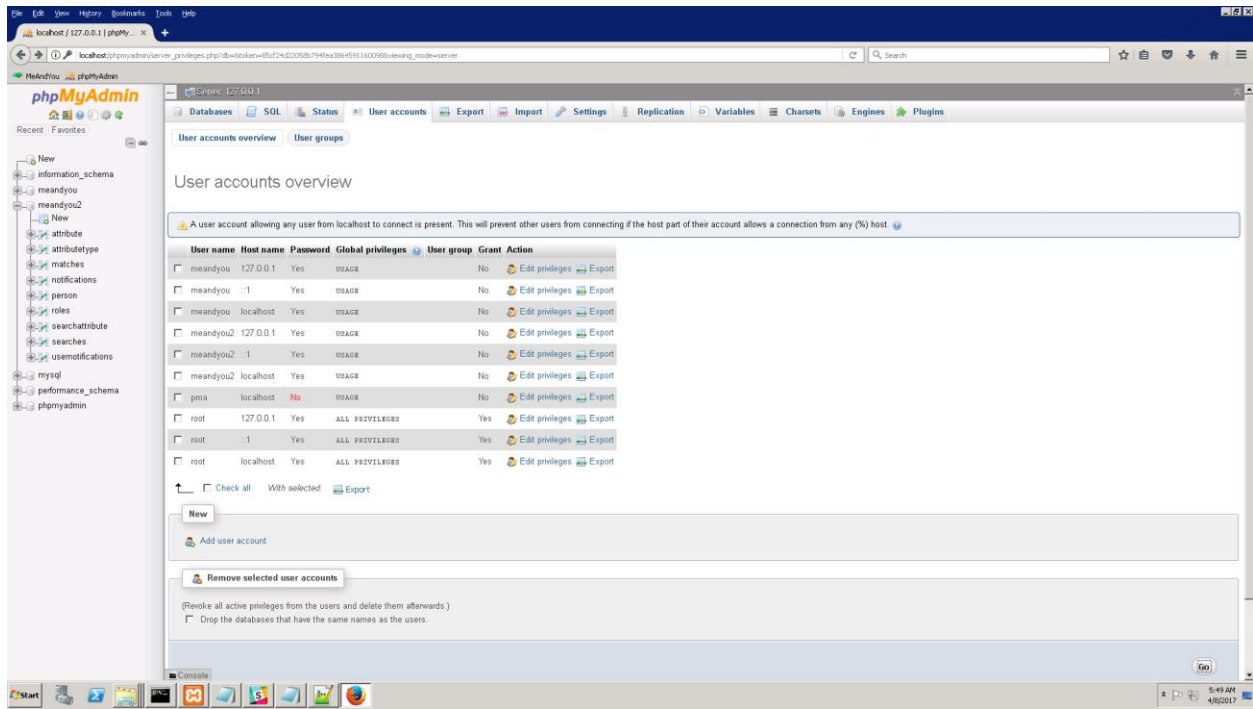
There's a Sample Data file that Patrick created, and updated as of recently for Database Group.

██████████ needs to download the latest version of the data to see the correct data points. Patrick also knows how to easily convert .csv files into SQL code, with a bit of manual coding of INSERT statements – which is how he made the INSERT statements for this week's SQL file. We need to make sure that ██████████ is on the same page as far as the new data model.

The new Simple Use Case Diagram and Activity Diagram have been created and uploaded to Wiki.

There are new tasks on Page 3 of this document, including the task of updating Job Descriptions by ██████████, and assigning who is going to create the RAD, SDD, and ODD documents – Patrick M. has the templates for them. They will go on the Wiki page when they are completed.

Database Access Control (Week 10)



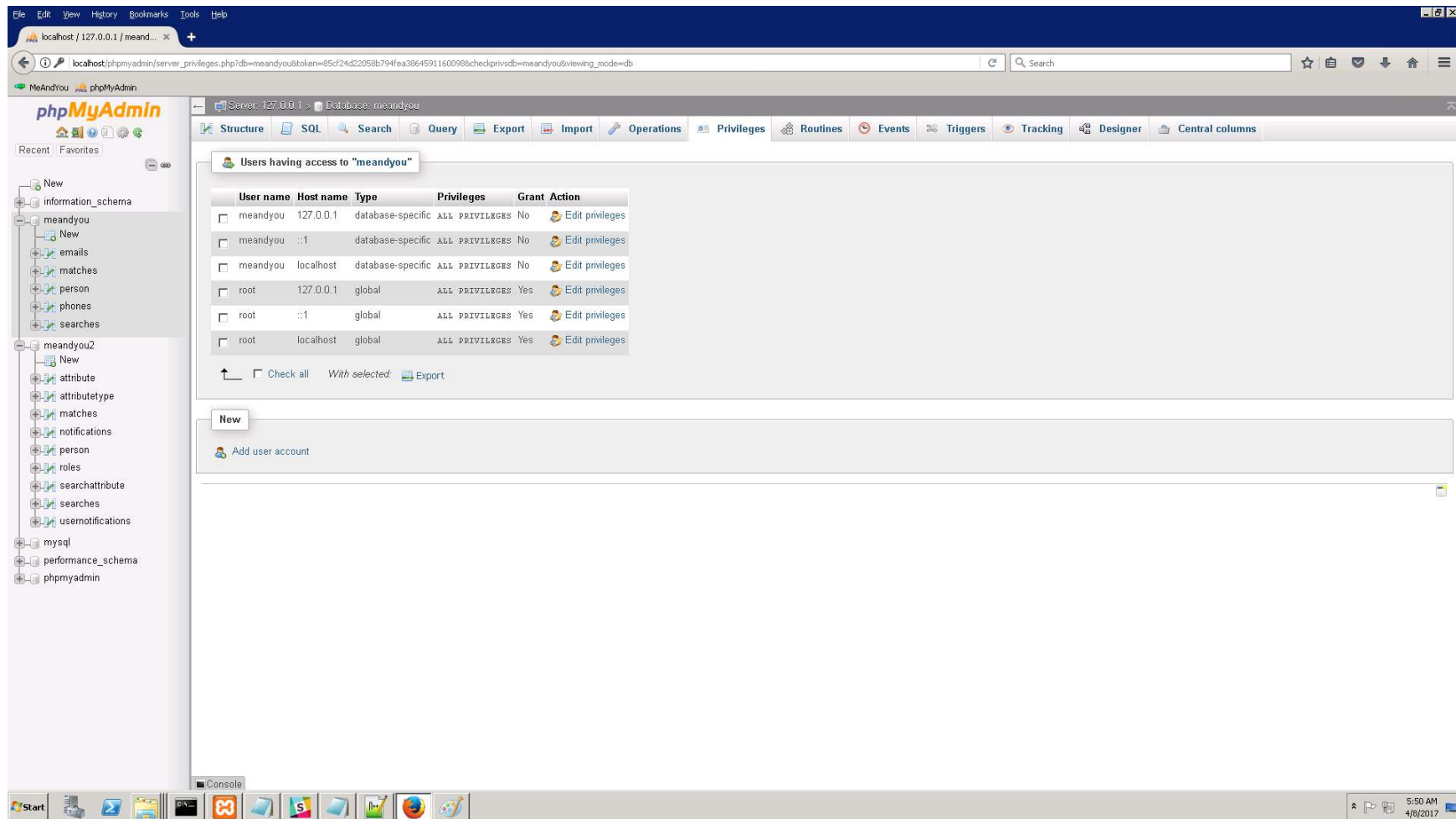
There are three different sets of user credentials for MySQL now.

- Username: **root**
 Password: **pw=root0408123**
 This account is used for phpMyAdmin only, because it has global access!
- Username: **meandyou**
 Password: **youandme2123**
 This account has database-specific access to "MEANDYOU" database, excluding GRANT access.
- Username: **meandyou2**
 Password: **youandme2123**
 This account has database-specific access to "MEANDYOU2" database, excluding GRANT access.

In a real world environment, accounts meandyou and meandyou2 would not share the same password. All of these accounts are set to only be accessed from **localhost**, **127.0.0.1**, and **:::1**.

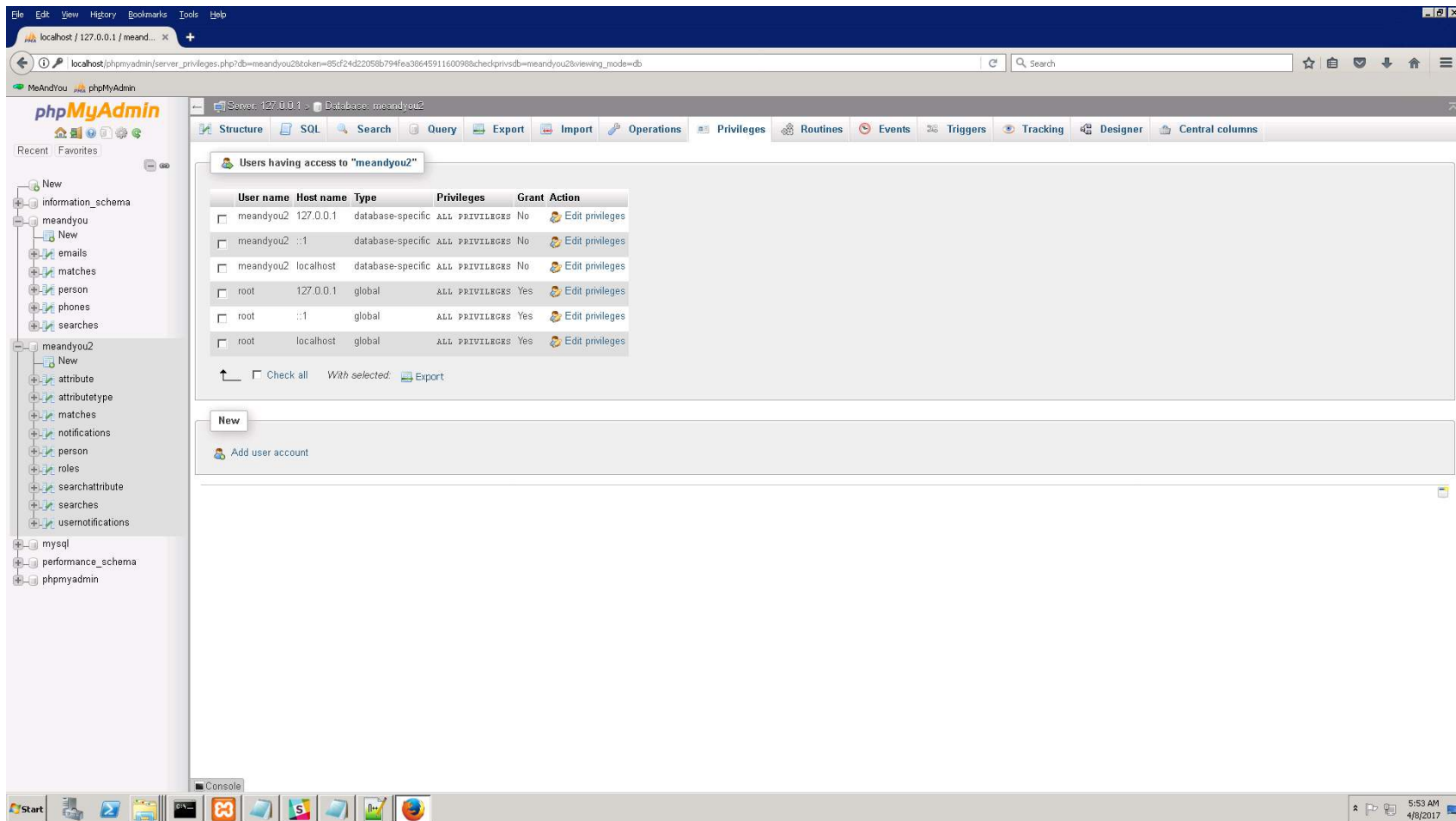
The previous class had them set as everything using the "root" account, without a password, and there was access that anyone outside of the server - including "Any" host could access the database without a password, with any username. They basically didn't know what they were doing with MySQL.

Database Access Control (Week 10)



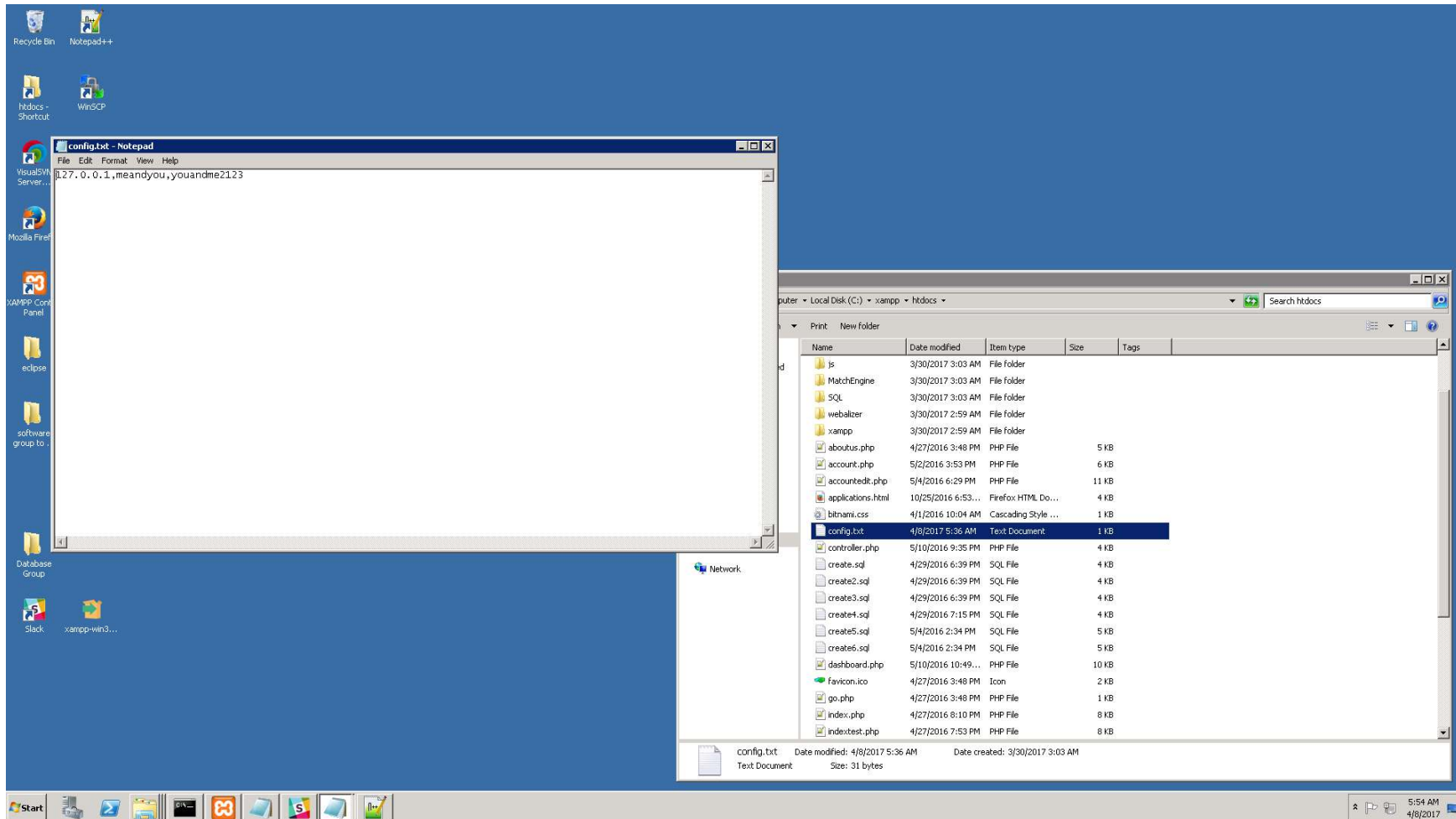
Configuration of User Account “meandyou” for Database “MEANDYOU”

Database Access Control (Week 10)



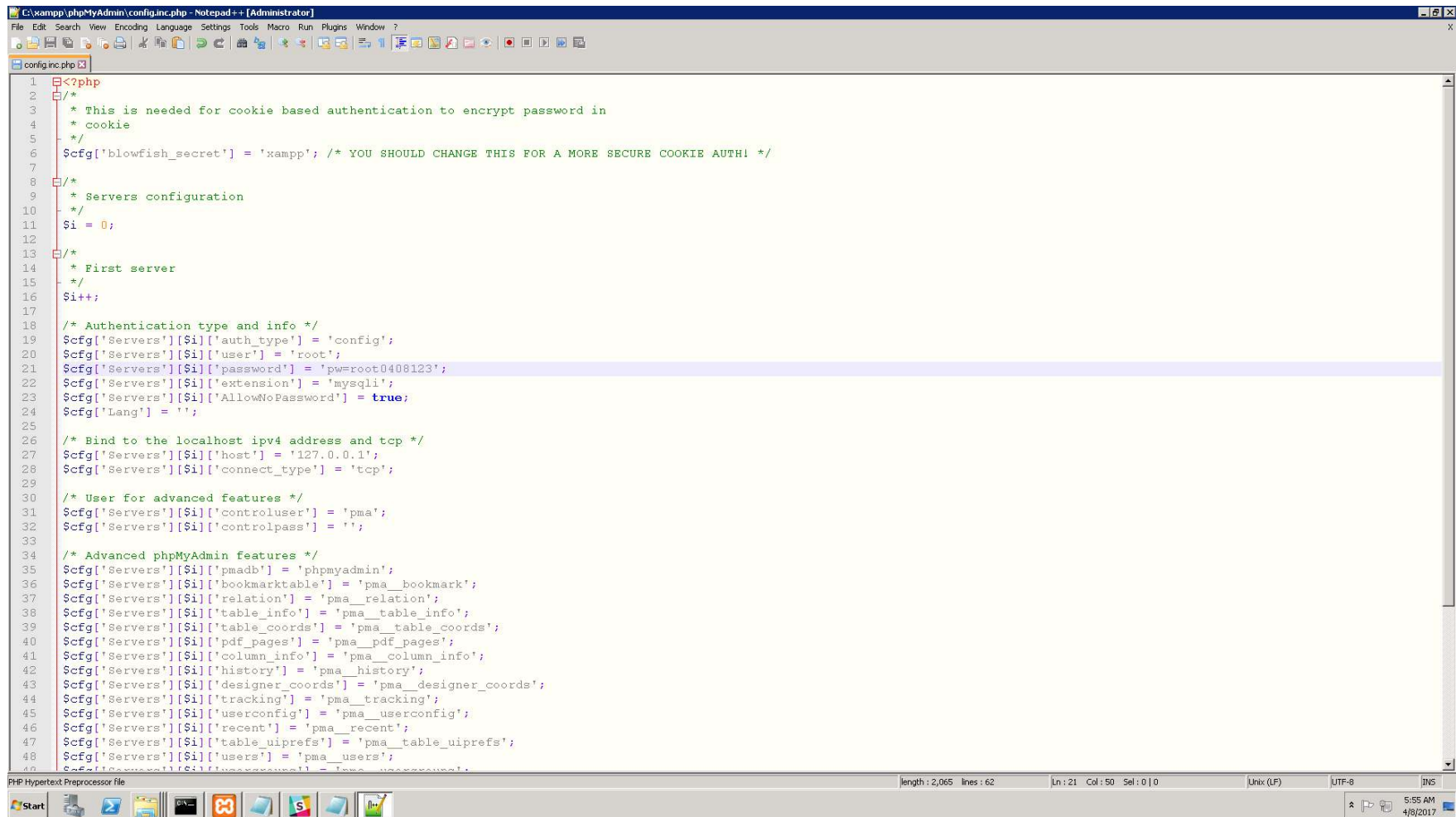
Configuration of User Account “meandyou2” for Database “MEANDYOU2”

Database Access Control (Week 10)



Configuration of **Demo** in C:\xampp\htdocs\config.txt

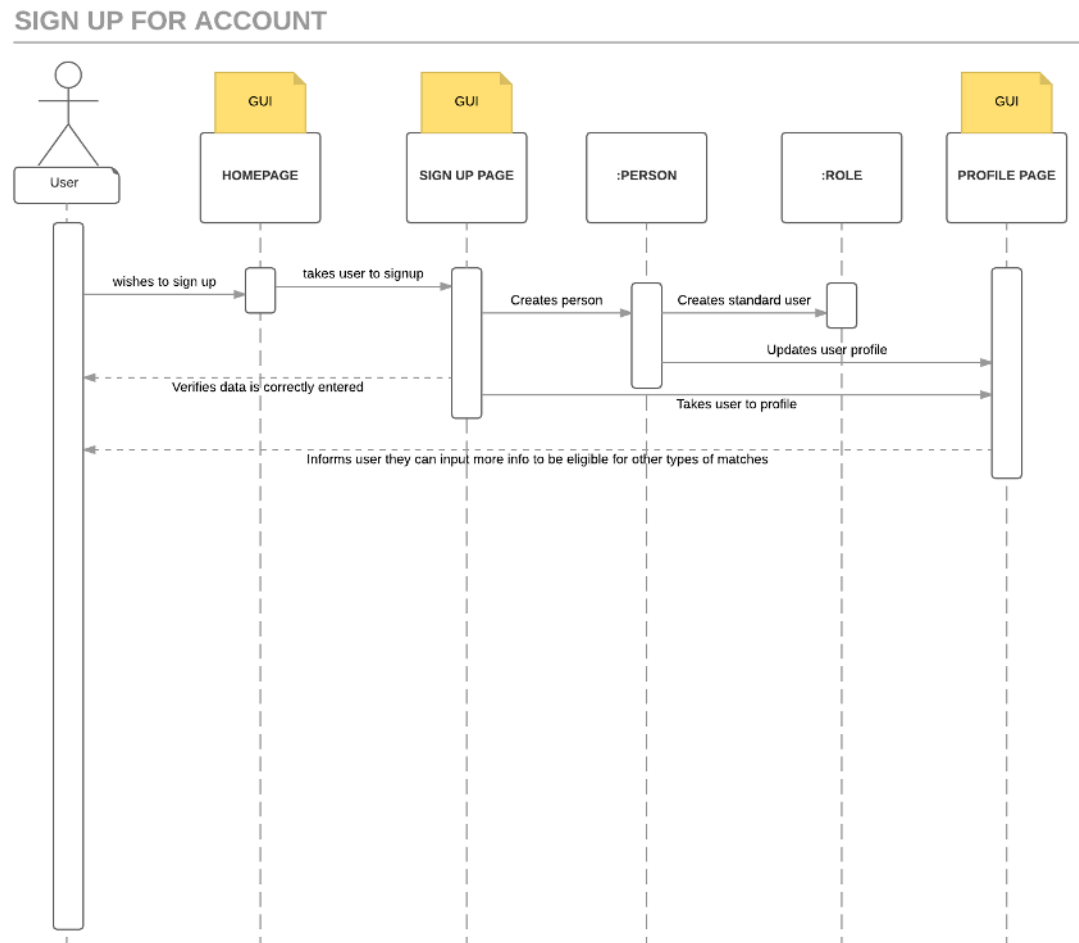
Database Access Control (Week 10)



```
1 <?php
2 /*
3  * This is needed for cookie based authentication to encrypt password in
4  * cookie
5  */
6 $cfg['blowfish_secret'] = 'xampp'; /* YOU SHOULD CHANGE THIS FOR A MORE SECURE COOKIE AUTH! */
7
8 /*
9  * Servers configuration
10 */
11 $i = 0;
12
13 /*
14  * First server
15 */
16 $i++;
17
18 /* Authentication type and info */
19 $cfg['Servers'][$i]['auth_type'] = 'config';
20 $cfg['Servers'][$i]['user'] = 'root';
21 $cfg['Servers'][$i]['password'] = 'pw=root0408123';
22 $cfg['Servers'][$i]['extension'] = 'mysqli';
23 $cfg['Servers'][$i]['AllowNoPassword'] = true;
24 $cfg['Lang'] = '';
25
26 /* Bind to the localhost ipv4 address and tcp */
27 $cfg['Servers'][$i]['host'] = '127.0.0.1';
28 $cfg['Servers'][$i]['connect_type'] = 'tcp';
29
30 /* User for advanced features */
31 $cfg['Servers'][$i]['controluser'] = 'pma';
32 $cfg['Servers'][$i]['controlpass'] = '';
33
34 /* Advanced phpMyAdmin features */
35 $cfg['Servers'][$i]['pmadb'] = 'phpmyadmin';
36 $cfg['Servers'][$i]['bookmarktable'] = 'pma_bookmark';
37 $cfg['Servers'][$i]['relation'] = 'pma_relation';
38 $cfg['Servers'][$i]['table_info'] = 'pma_table_info';
39 $cfg['Servers'][$i]['table_coords'] = 'pma_table_coords';
40 $cfg['Servers'][$i]['pdf_pages'] = 'pma_pdf_pages';
41 $cfg['Servers'][$i]['column_info'] = 'pma_column_info';
42 $cfg['Servers'][$i]['history'] = 'pma_history';
43 $cfg['Servers'][$i]['designer_coords'] = 'pma_designer_coords';
44 $cfg['Servers'][$i]['tracking'] = 'pma_tracking';
45 $cfg['Servers'][$i]['userconfig'] = 'pma_userconfig';
46 $cfg['Servers'][$i]['recent'] = 'pma_recent';
47 $cfg['Servers'][$i]['table_uiprefs'] = 'pma_table_uiprefs';
48 $cfg['Servers'][$i]['users'] = 'pma_users';
49 $cfg['Servers'][$i]['usergroups'] = 'pma_usergroups';
```

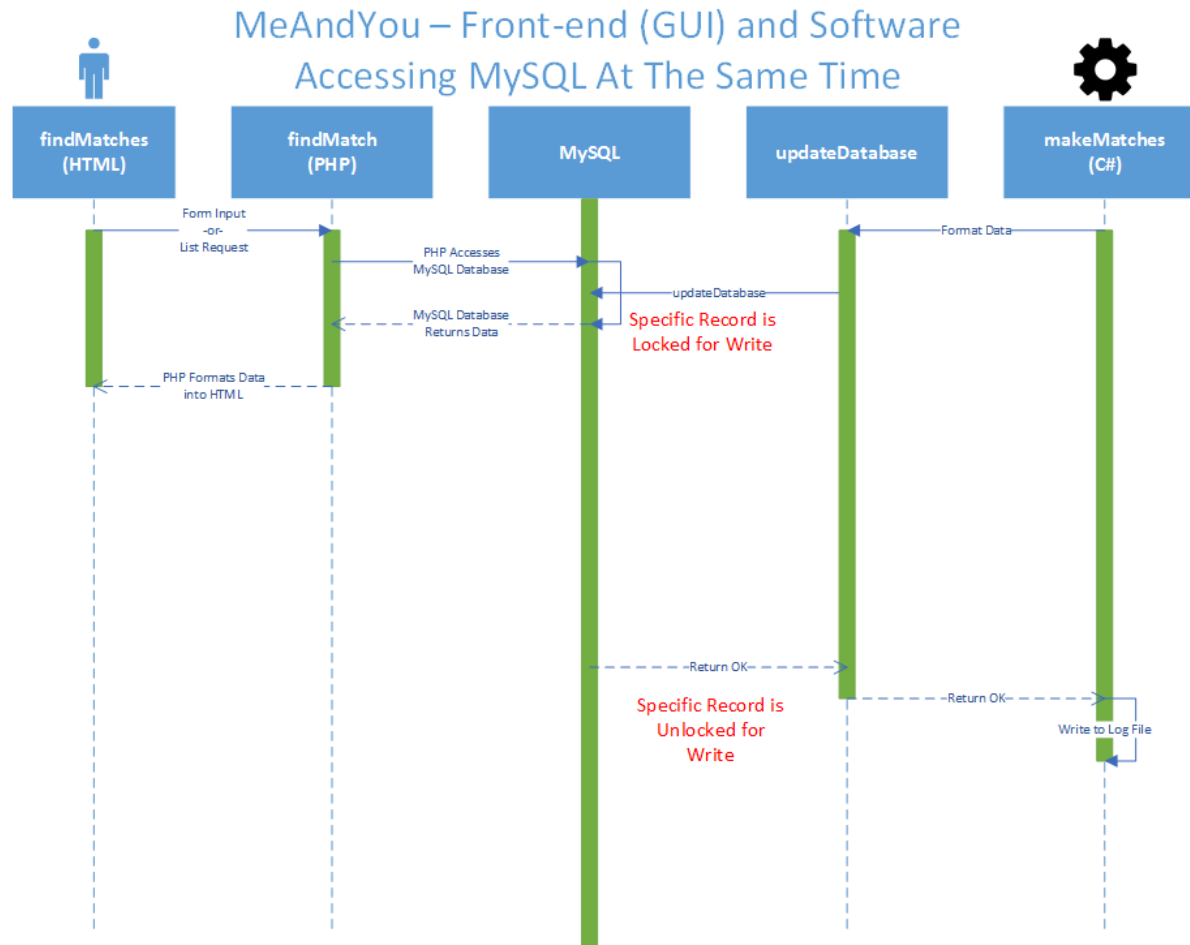
Configuration of **phpMyAdmin** in C:\xampp\phpMyAdmin\config.inc.php

UML Diagram – Sign Up For Account (Week 10)



This sequence diagram shows the process and messages involved in a user signing up for an account.

Database Write Control (Week 10)



This diagram shows that records will be locked when they are being written to by either Front-end (GUI) or Software. They will have to write in code to wait until the lock is released if they are trying to write to a specific entry that is locked – otherwise the user will get an error.