

# JMU Share

JMUShare will be a web application designed for note sharing between students and professors. The overall system will have several subsystems that work together to create the final product that allows students to easily post notes, find their classmates notes, and discuss the content of the notes themselves. The Model-View-Controller architectural pattern will logically separate concerns and allow a simplistic design of the individual components necessary to build the entire system.

The first way we will be separating data from the rest of the system is that we will be implementing a database which will have tables for the User to store all of the information for a particular user, a Posts table to store information about each post, a Comments table to store comments for associated posts, a Roles table which will distinguish what kind of user a user is, a Class table to store information about all classes and professors associated with them, a School table to hold the list of Universities that our service will be compatible with, and an Image table that will have a link to a file for a specified Post.


We will have users interacting with the different tables through the use of Java classes in the Model component of the Model-View-Controller architecture. The user interface will utilize Java Server Pages with client side javascript and ajax to streamline user interaction and produce more useful content. The only way this application will prove more useful than other available options is if it is incredibly easy to find and post notes.

Our controller makes use of the struts 2 framework which greatly streamlines the process of accessing data in the model from the view. It also streamlines the process of creating the application overall which gives us more time to develop the user experience. By using the Model-View-Controller architectural pattern, it is our hope that our application will prove incredibly user friendly while also using the separation of concerns to create security redundancy. The redundancy of security checks in both the view and model will better restrict permissions so unauthorized users can't gain access to private user, posting, and class information.

## User Interface Standards

With the exception of the login page, every view component will have an identical menu bar fixed to the top of the web page and another at the bottom of the page. Since the site's name is JMU Share, we will keep the colors restricted to the JMU colors and white. A quick google search gives us the school colors. Along with the two colors below, we'll use white (#FFFFFF) as a consistent background for the notes and a light grey (#DDDDDD) to provide a similar feel to common text editors like MS word, libre office, and google docs.

### Colors

Purple	Gold
	
#450084	#CBB677

### Top menu bar

Home	Search	Post New Notes	Profile
------	--------	----------------	---------

As shown above, the selected page on the top menu bar will be gold colored. When a link on the bar is active it will glow darker. Listed are the four main pages of our website. These pages will have a similar look to the example page provided below with the same colors and fonts.

### Bottom menu bar

About
-------

The bottom menu bar has the same style as the top bar but only has the about link. Once the application has more publishing information that we want to share we'll add it to this bar in the center.

### Headings

Headings will be kept left justified like all text. The h1 heading will be reserved for the titles of notes which describe the note contents. The h2 heading will be used for the start of the comments sections, and the meta data about notes like class, term, and teacher. The h3 heading will be used for the other data related to notes including author, date submitted, and all of the search results data. Since search results need to have smaller text to fit more results in one page, all information about notes aside from the title will be set to the h3 heading size. The paragraph tags will be reserved for the content of notes making it easy to skim past the plain paragraph text when looking for descriptor information in bold headings.

### Logos

Once we decide on a logo, it will be included in the "Home" section of the menu bar. It will still link to the homepage but it will be an image in the place of the text. We'll also put the logo in the center of the bottom bar.

## Examples

This example of a notes page provides basic examples of most of the content on the site. The sections on all the pages will be wrapped with this gray border. The search results will be separated from the search form contents above, and profile information will be separated from user posting links listed below them. The top bar will always be visible but if the content is large enough, the bottom bar will be pushed off the end.

## Notes page

<a href="#">Home</a>	<a href="#">Search</a>	<a href="#">Post New Notes</a>	<a href="#">Profile</a>
----------------------	------------------------	--------------------------------	-------------------------

## Cryptography Ciphers and Their Weaknesses

**CS 457 Brett Tjaden - Fall 2016**

Notes Begin Here --- Cryptography - designing systems so only certain people can see through the disguise Symmetric key systems - both sender and receiver have the same key Cesar Cipher (small keyspace) Shifts letters A->D B->E Keyspace of 26 is terrible Find the letters on the key and swap them with the letters that make up the other two corners. Swap in same row If in the same row shift right or shift down Notes Begin Here --- Cryptography - designing systems so only certain people can see through the disguise Sorigue - known plaintext attack Ke B->E Keyspace of 26 is terrible Monoalphabetic replacement (letter frequencies) Every letter corresponds to another letter Keyspace is still 26, just harder to see the pattern Attack with language redundancy Lots of E's The word the Digraphs of th qu es Trigraphs like the and ing One Time pad Unbreakable Every letter is shifted or affected differently. HELLO -> 19726 -> INSNU How do you fix language redundancy? Use multiple keys Replace multiple letters at once Playfair (broken in 1914 by J Mauborgne) - known plaintext attack Key use keyword to make a key 5x5 square Remove duplicate letters from keyword Replace J w/ I since we have 26 letters and 25 spaces Fill rest of key after keyword with remaining letters in order Encrypting Split letters w/ x if double letters like ss or ee Add to end of text if odd Split letters into pairs of 2 Find the letters on the key and swap them with the letters that make up the other two corners. Swap in same row If in the same row shift right or shift down Notes Begin Here --- Cryptography - designing systems so only certain people can see through the disguise Symmetric key systems - both sender and receiver have the same key Cesar Cipher (small keyspace) Shifts letters Split letters w/ x if double letters like ss or ee Add to end of text if odd Split letters into pairs of 2

---

### Comments

Here is a comment Here is a comment Here is a comment Here is a comment Here is a comment Here is Yup it's a comment is a comment Here is a comment Here is a comment Here is a comment Here is a comment

Write comment here...

[About](#)

For the other examples, I have created them in photoshop instead of an html editor so they don't have valid css formatting. I just manually created them real quickly as creating the pages in html would be completing most of the project. The notes example above required enough design decisions to be made that I determined that making the rest of the examples real pages was unnecessary. They certainly show what the user interface should look like.


Profile page

Home

Search

Post New Notes

Profile



Miles Greatwood

Profile Information

Classes I'm currently in

Contact info

Other Information

Cryptography Notes 1

Cryptography Notes 1

Cryptography Notes 1

Cryptography Notes 3

About

Search page

Home

Search

Post New Notes

Profile

Title

School

Class

Teacher

Author

Sort By title

Sort by date Published

Sort by Rating

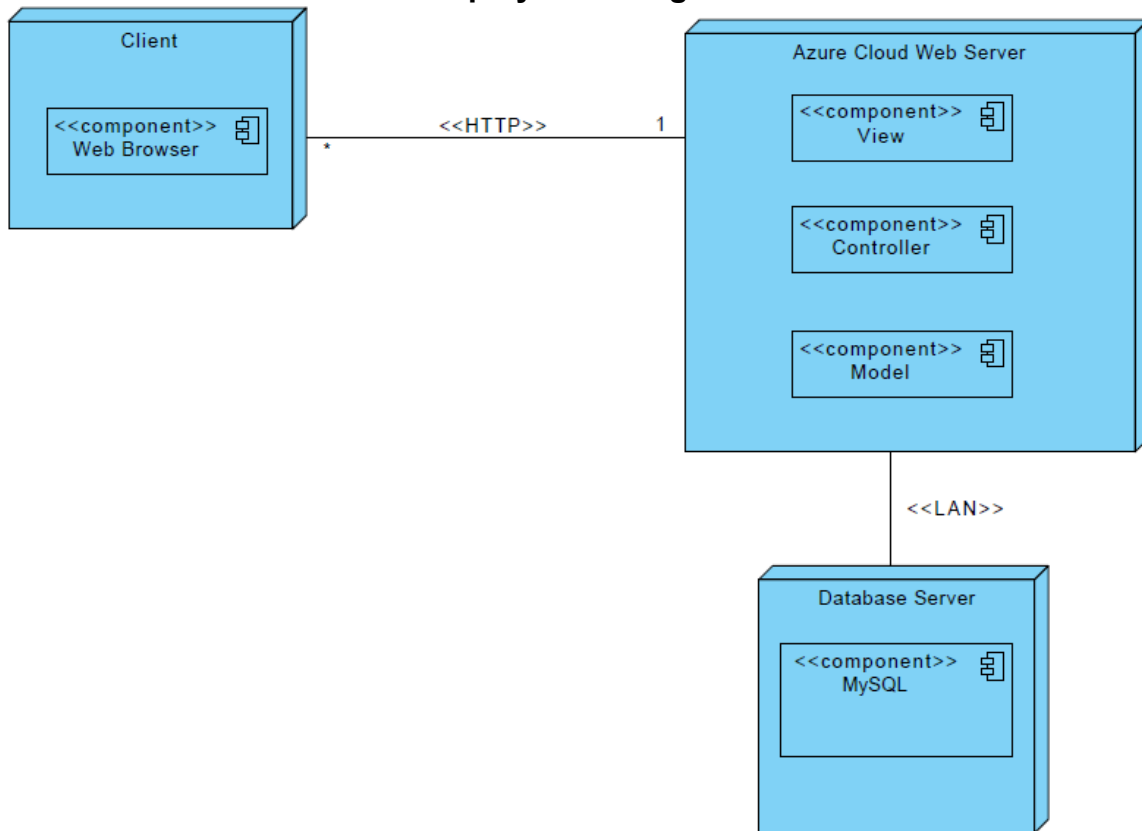
Sort by class

TITLE OF RESULTS 1

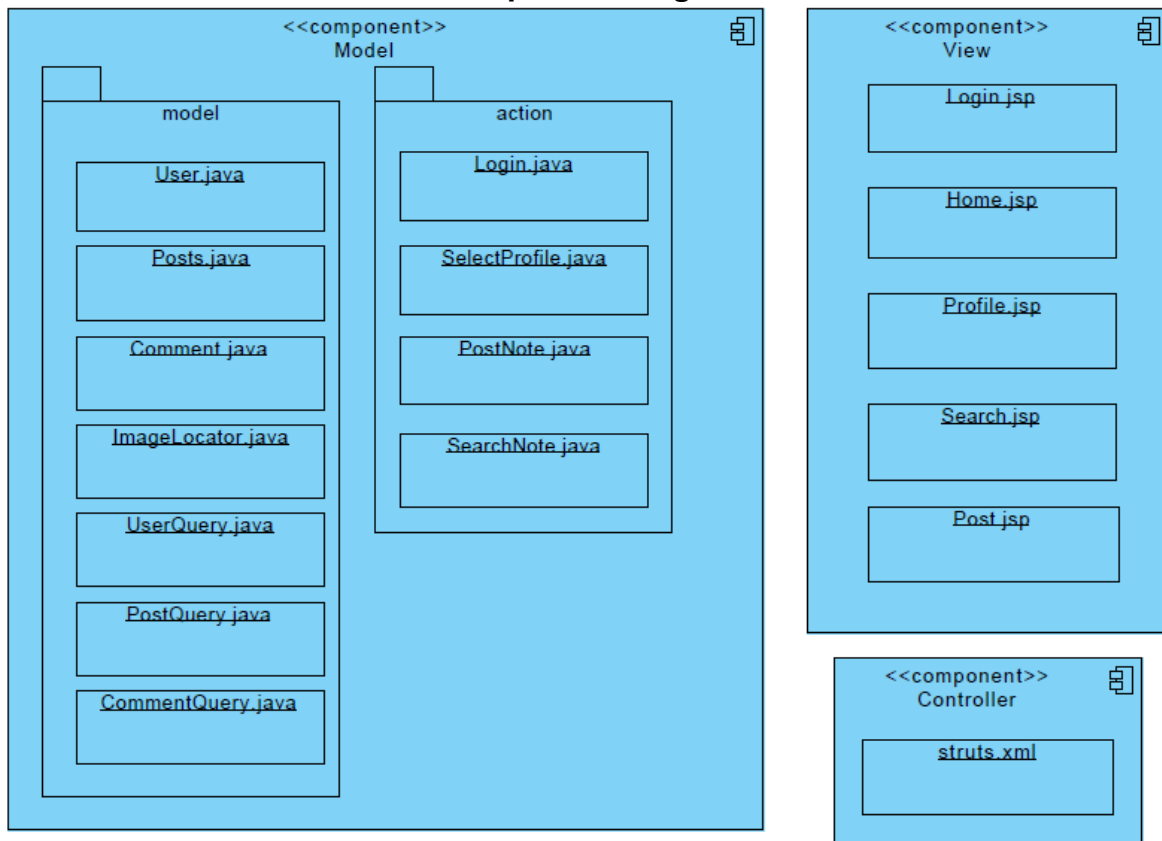
TITLE OF RESULTS 2

About

## Deployment Diagram



## Component Diagram



# Components

## Model

- User.java
- Posts.java
- Comment.java
- ImageLocator.java
- Database (MySQL)
- UserQuery.java
- PostQuery.jav
- CommentQuery.java

## Controller

- Struts.xml

## View

- Login.jsp
- Home.jsp (dynamic)
- Profile.jsp (dynamic)
- Search.jsp (dynamic)
- Post.jsp (dynamic) also called post new notes
- 

# Database Tables

**User Table** - contains a list of user information:

- id (primary key, unique identifier for each user; INTEGER)
  - The id is used as an indicator of what position the user is as part of the table, as it is the primary key.
- password (password SHA2hash for login; VARCHAR(128))
- The password is the corresponding field to the username and will be used to determine if the login information is correct
- email (used for signup; VARCHAR(255))
- The email provided when the user signs up to be a part of the system.
- name (name of the user; VARCHAR(255))
- The name provided by the user, will be the name of the author when a new post is made
- username (username for the user; VARCHAR(255))
- Part of the necessary login information, used to check the table for a password.

- schoolID (foreign key to schools table: INTEGER)
- Information that can be used to relate an individual to the school, and help provide more accurate search results. It is also used to help tag their posts as part of that school system
- roleID (foreign key to roles table; INTEGER)
- The roleID helps determine permissions of the user, like deleting posts and submitting certain types of posts.
- isProfessor (BOOLEAN that states whether the user can make a class)
- Similar to the roleID in that it grants special permissions for editing the documents and giving feedback such as endorsing the notes

**Posts Table** - contains each posts information including:

- Id (primary key for table : INTEGER)
  - The id is used as an indicator of what position the post is in the table, as it is the primary key.
- authorID(foreign key to the user table storing the author of the post; INTEGER)
- The authorID is used to connect this table to the user table as authorID matches id in the User Table
- classID (foreign key to the class table; INTEGER)
- The classID is used to connect this table to the class table as classID matches id in the Class Table
- text (contains the body of the text for the note; VARCHAR(255))
- This is the note itself, and includes any text the user needs to be part of any additional information to go along with the note, like suggestions or tips for how to study the material
- rating (average post rating from 1 to 5; FLOAT)
- The rating given by the users for the quality of the post, this value changes with each vote given by new users
- endorse (if a professor believes the notes are adequate or should be used as an example; BOOLEAN)
- Set to true if the professor has endorsed the piece as accurate study material, false otherwise

**Comments Tables** - contains a set of comments for the associated posts this information includes:

- id (primary key for table:INTEGER)
  - The id is used as an indicator of what position the comment is in the table, as it is the primary key.
- postID (foreign key used to link posts tables; INTEGER)
- The postID is used to connect this table to the Posts Table as postID matches id in the Posts Table
- comment (the actual comment; VARCHAR(255))
- Text provided to support or refute certain characteristics of the note, or to ask or answer questions about the note.
- author (foreign key to the user table; INTEGER)
- This is the username and will relate the comment back to the User Table

**Roles Table** - contains possible roles for a user to have

- id (unique identifier for each role; INTEGER)
  - The id is used to connect this table to the user table as id matches roleID in the User Table
- role name (the String name for each role; VARCHAR(255))
- The name of the roll, such as administrator, student, professor, etc

**Class Table** - contains all created classes and professors associated with them

- id (unique identifier for the database; INTEGER)
  - The id is used as an indicator of what position the class is as part of the table, as it is the primary key.
- class number (the number for the class; INTEGER)
- This is specific class number, ex. "CS345" in this 345 would be the class number
- class name (the department for the class; VARCHAR(255))
- This is the long version of the name of the class, ex. "CS345 - Software Engineering"
- class description (the department for the class; VARCHAR(255))
- This field is used to give more information about what topics the course specifically covers.



- department (the department for the class; VARCHAR(255))
- This is the abbreviated department title, ex “**CS**345” in this CS would be the department
- professor id (foreign key to the user table, only professors; INTEGER)
- The professor id is used to connect this table to the User Table as professor id matches id in the User Table (if a user for that professor exists)

**School Table** - table that stores the id number and string of

- school (university name: VARCHAR(255))
  - This is the title of the university, ex. “James Madison University”
  - Id (primary key :INTEGER)
  - The Id is used to connect this table to the User Table as Id matches schoolID in the User Table

**Image Table**

- fileLocation (path to the file on the server : VARCHAR(255))
  - This is a path name to the image file, so that the image itself is not stored as part of the table.
  - Id (primary key : INTEGER)
  - The id is used as an indicator of what position the image is as part of the table, as it is the primary key.
  - postID (foreign key : INTEGER)
  - The postID is used to connect this table to the Posts Table as postID matches id in the Posts Table