

Project Proposal – JMU-Share

Group 7

Miles Greatwood, Chris Recinos, Kendrick Smith, Jacob Wilson

I. Overview

Many college students have had difficulties during their undergraduate career that have led to them seeking online resources to help them with their classes. However, many of these same students can spend hours trying to find online content to help them prepare for a test or supplemental notes to clarify their own notes but never find notes for what they need help with. JMU-Share plans to address this problem by creating a new web application specific to the JMU community that allows users to upload their class notes or view other notes. We plan to limit most features to strictly the JMU community by requiring a @dukes.jmu.edu or @jmu.edu email address to register and login in. With a student's dukes.jmu.edu account, they will be able to upload their own notes, view other notes, and comment on other's notes. If a faculty member registers with a jmu.edu account, they will be able to post their own notes for a class as well as endorse student's notes or comment on a note with corrections or clarifications.

As for the financial market value, JMU-Share will not generate any revenue, our main goal is to help students feel more confident about their classes throughout the semester via peer-assisted learning.

The main transactions that will be supported by JMU-Share are uploading notes to a specific department and course number, searching for notes within first a department and then the course number, commenting on notes for further clarification or explanation, rating notes based on their helpfulness and detail, endorsing notes, and a user may remove a note that they have uploaded. Admin will also be able to remove any note if a note appears to lack as a supplemental resource.

II. Dynamic Content

1. Search Results

To find notes that the user wants, they will be able to search by most of the attributes in the posts table. They can search by class, author of notes, rating, date, and subject. There will be selection options at the top of a page and below notes that match the search criteria will appear. The results will depend on the search and will include a post title, author name, class title, rating info, a possible professor endorsement, and a hyperlink connecting to the actual notes themselves.

2. User Profiles

Each user will have a profile with their email, username, role, and previously posted notes. All the previous posts will have ratings and hyperlinks to them. The user information will come from the user table. The notes titles, ratings and hyperlinks will come from the posts table.

3. Comments on notes

To make the notes more interactive, users will be able to make comments like sticky notes on specific locations in a post. These comments will be seen by all users and will facilitate a discussion and improving of notes. Along with in line sticky note style comments there will also be a comments section at the bottom of the notes page for more lengthy discussion. The content and location of the comments will come from the comments table. Every time a new comment is made it will be stored in the comments table and upon refresh will appear to other users looking at the same notes page.

4. Customized notes

Over the course of a semester there is an enormous amount of material covered. Individual users will have the option to create a customized set of notes that pulls from a variety of other users posts and their own posts. This page will be highly customizable for the user and will avoid the need for the user to click through 30 documents to study for a test. They will select sections of posts that they want to use for their custom notes and it will create a new continually updated post in the posts table. Each time they select a section of another persons notes for insertion into their personal customized notes, their post will be updated in the posts table. It is very much like creating a Frankenstein collaboration of notes.

III. Necessary Database Tables

- User Table
 - contains a list of user information:
 - - id (unique identifier for each user; long)
 - - password (used for login; varchar(255))
 - - active (is the user active or not; boolean)
 - - email (used for login; varchar(255))
 - - name (name of the user; varchar(255))
 - - username (username for the user; varchar(255))
 - - role id (foreign key to roles table; long)
- Posts Table
 - contains each posts information including:
 - - author id (foreign key to the user table storing the author of the post; long)
 - - class (foreign key to the class table; long)
 - - text (contains the body of the text for the note; varchar(255))
 - - image (information on the image(if included); jpg)
 - - rating (average post rating from 1 to 5; float)
 - - id (the post's unique identifier in the database; long)
 - - endorsement information (if a professor believes the notes are adequate or should be used as an example; boolean)
- Comments Tables
 - contains a set of comments for the associated posts this information includes:
 - - post id (used to link tables; long)
 - - comment text information (the actual comment; varchar(255))
 - - author (foreign key to the user table; long)
- Roles Table
 - contains possible roles for a user to have
 - -id (unique identifier for each role; long)
 - -role name (the String name for each role; varchar(255))
- Class Table
 - contains all created classes and professors associated with them
 - - id (unique identifier for the database; long)
 - - class number (the number for the class; integer)
 - - department (the department for the class; varchar(255))
 - - professor id (foreign key to the user table, only professors; long)

IV. Use Case Descriptions

1. Create an Account

Abstract: A student or professor uses this use case to create their account on JMU-Share.

Precondition: Student or professor has a valid JMU email.

Postcondition: The student or professor has an account on JMU-Share.

Steps:

1. Student or professor chooses to create an account on JMU-Share
2. System requests a valid JMU e-mail to use as the account username
3. Student or professor provides an email.
4. System sends confirmation email.
5. Student or professor clicks link in confirmation email.
6. Account is created.

Alternatives:

3a1. Student or professor provides an invalid email

3a2. System returns to step 2

2. Recover Password

Abstract: A user uses this use case to recover their password if they forget it.

Precondition: User has access to a valid email associated with the application.

Postcondition: User has recovered the password to their account.

Steps:

1. User hits the recover password button.
2. System prompts for a valid email that's associated with a created account.
3. User enters a valid email.
4. System sends password to the user.

Alternatives:

3a1. User enters an invalid email.

3a2. System returns to step 2.

3. Search Through Posts By Class

Abstract: A student uses this use case to search through note postings by a specific class

Precondition: Student is logged into JMU-Share

Postcondition: Student views all results for their search

Steps:

1. Student selects a class from the options and requests search results.
2. System returns any postings matching the searched class.

4. Search Through Posts By Professor

Abstract: A student uses this use case to search through note postings by a specific professor.

Precondition: Student is logged into JMU-Share.

Postcondition: Student views all results for their search.

Steps:

1. Student enters a professor's name and requests search results.
2. System returns any postings matching the searched professor.

5. Search Through Posts By Post Author

Abstract: A student uses this use case to search through note postings posted by a specific author

Precondition: Student is logged into JMU-Share.

Postcondition: Student views all results for their search.

Steps:

1. Student enters an author's name and requests search results.
2. System returns any postings matching the search author.

6. Add a Post

Abstract: A user uses this use case post a note to the system

Precondition: User is logged into JMU-Share.

Postcondition: System creates the post or adds the new post as a comment to a preexisting post

Steps:

1. User enters their notes into the box provided.
2. System prompts the student for confirmation.
3. System creates the posting with the students information.

Alternatives:

- 1a1. User chooses an image file to upload.
- 1a2. System moves to step 2.
- 2a1. User hits no in the confirmation window.
- 2a2. System moves to step 1.

7. Delete a Post(student or professor)

Abstract: A user uses this use case to delete a note they posted from the system

Precondition: User is logged into JMU-Share.

Postcondition: User's note is removed from the system.

Steps:

1. User chooses a note that they uploaded to the system.
2. System asks for confirmation.
3. System removes the note.

Alternatives:

- 2a1. User hits no in the confirmation window.
- 2a2. System moves to step 1.

8. Comment on pages

Abstract: A user uses this use case to add a comment to a note or notes.

Precondition: User is logged into JMU-Share.

Postcondition: User's comment is logged to the note.

Steps:

1. User chooses a note that has been uploaded to the system.
2. User provides the comment.
3. System prompts for confirmation.
4. System checks the comment to make sure it's valid and appropriate.
5. System adds the comment to the selected note

Alternatives:

- 3a1. User hits no in the confirmation window.
- 3a2. System moves to step 2.
- 4a1. System finds the comment to be invalid or inappropriate.
- 4a2. System moves to step 2.

9. Rate Posts

Abstract: A user uses this use case to rate a note based on how useful they feel it is.

Precondition: User is logged into JMU-Share.

Postcondition: User's rating is stored in the system.

Steps:

1. Student chooses a note posting from the system.
2. Student selects between 1 and 5 stars for the posting.
3. System stores the choice.

Alternatives:

- 2a1. Student hits a different star selection on the posting.
- 2a2. System removes the previous star selection and adds the new selection to the database.

10. Subscribe to a Post

Abstract: A student uses this use case to favorite a note posting and mark it to be followed by their account

Precondition: Student is logged into JMU-Share.

Postcondition: Posting is marked to be followed by the student's account

Steps:

1. Student chooses note posting from the system.
2. Student hits the favorite button
3. System marks the post to be followed by the student's account
4. System notifies student whenever a new post is made upon the following note posting.

Alternatives:

- 2a1. Student hits the favorite button again.
- 2a2. System removes favorite marking and moves to step 1.

11. Mark a note posting as correct/ideal or endorse an answer

Abstract: A professor uses this use case to mark a set of notes as perfect answer or best answer, similar to what Piazza does.

Precondition: Professor is logged into JMU-Share.

Postcondition: The system displays the note posting as professor-endorsed.

Steps:

1. Professor chooses note posting from the system.
2. Professor hits the endorsement button.
3. System marks the post as endorsed as a good answer by the professor.

Alternatives:

- 2a1. Professor hits the endorsement button again.
- 2a2. System removes the endorsement marking on the post.
- 2a3. System moves to step 1.

12. Delete A Post(administrator)

Abstract: An administrator uses this use case to delete any note from the system

Precondition: Administrator is logged into JMU-Share.

Postcondition: The note is removed from the system.

Steps:

1. The Administrator chooses a note that has been uploaded to the system.
2. System asks for confirmation.
3. System removes the note.

Alternatives:

- 2a1. Administrator hits no in the confirmation window.
- 2a2. System moves to step 1.

13. Sort Posts

Abstract: A student uses this use case to sort displayed posts by the date they were posted or by their current 5-star rating.

Precondition: Student is logged into JMU-Share.

Postcondition: System displays all postings in order based upon the chosen sorting method.

Steps:

1. Student chooses the sort drop-down list.
2. Student selects one of the sort options in the list.
3. System displays the sorted results to the student.

14. Update Profile

Abstract: A user uses this use case to update their email and password for their account.

Precondition: User is logged in.

Postcondition: User's profile information is changed.

Steps:

1. User clicks the *Update Profile* button.
2. System prompts the user for their login information.
3. User provides their login information.
4. System validates the login information.
5. System displays the profile information.
6. User changes their profile information.
7. User hits the save changes button.
8. System saves changes.

Alternatives:

4a1. Login credentials are not successfully validated.

4a2. System returns to step 2.

7a1. User fails to save changes.

7a2. System discards profile changes.

15. List Users

Abstract: An administrator uses this use case to list all users that are registered to the application.

Precondition: The administrator is logged in.

Postcondition: The administrator has the list of all users registered to the application.

Steps:

1. The administrator requests the application to list all users.

2. The system displays all users in alphabetical order.

