Proposal:

Caelan Herzberg:

Programming Languages: C, C++, Java, SQL, HTML, JavaScript, Node.js.
Courses: COMS 327, COMS 363, COMS 319
Designed: Nintendogs browser game in HTML/Javascript for COMS 319

Jian Shi:

Programming language: C, C++, Java, SQL, HTML, SpringBoot framework.
Courses: COMS 363, 362, 327, 321
Internship: Not apply
Designed: Air ticket reservation system, snake game by C++, and homeworks applied:)

Wenqin Wu:

Programming languages:Java, C language, SQL, HTML, Node.js, HTML, JavaScript. Python(a little bit).
Course COMS: COMS 363, COMS 311, COMS 319.
Intern: NULL.
Designed: photo display website, Homework.

Josh Podlich:
Programming languages: Java, C++, C#, a little bit of SQL and HTML
Courses: COMS 227, 228
Internship: none
Designed: Card Game in Java for class assignment

Project Idea A:

Name: (RateMyLife) (Recommender)(Recommend Me)
Problem being addressed: provide recommendations for ISU students
Description of Project:

This app will provide users with tailored recommendations based on their interests in movies, video games, etc in order to introduce users to new media they would presumably enjoy. This will be done by letting the user “like” certain media and then checking for similar media based on genre. Users will also be able to view critic reviews and read/write personal reviews of media to not only find media they would like but to also let users voice their opinions.

The app will have different categories for users to check. For example, we have a new video game release or rank levels for users to see and make comments about it. On the other hand, users will rate their interests in video games. Users need to log in to leave comments or rate and guests can directly look at the recommendations through the app. Managers will be offered special accounts to delete comments that violate app rules such as dirty words etc. Manager will also have authority to ban accounts.

Language/Platform/libaries: Android, Springboot, Java, SQL, HTML, XML, Rotten Tomatoes API, Metacritic API.

Large/Complex:

Creating users, managers, administrators to control different functions may be complex.

Ranking system. For example, Spiderman has the most views in the movie section, then the spiderman is on the top level of the screen. Ranked by most views, most commented, published time etc by.

Retrieving media information from other reputable sources: The app will be able to pull critic reviews and general information from the internet. For instance, we could pull a critic review and movie info from rotten tomatoes

Recommendation algorithm: user will be able to “like” media and write reviews. Based on content the user gave a good rating/”liked” we will check for similar media to recommend. Our recommendation algorithm will be based on the genre of media the user enjoyed.

Project Idea B:

Name: On campus shopping

Assignment P02_ScreenSketches (unfinished):

Step 1 (Extension may just a idea for now and will not write any more use cases for extension currently):

Use-cases1: Register account
Actors: Guest User
Triggering Event: Users click on the register button if they do not have an account.
Success Guarantee: Provided a registration DB
Preconditions: User information(Basic info, Name, password, username, DOB…)
Main Success Scenario:

1. User can use the comment functions after registration
2. Users can switch accounts based on the registration information.(Log in, log out)

Extensions: Add on email or phone number register confirmation.

Use-cases2: user profile menu
Actors: User
Triggering Event: User selects profile button and clicks on it
Success Guarantee: Menu for User profile is opened
Preconditions: User must have a registered account

Main Success Scenario:

1. User profile menu is opened
2. User can choose to change profile picture, change password, change username, delete account, or return to home screen by selecting corresponding buttons
   Extensions:

Use-cases3: change profile picture
Actors: User
Triggering Event: User presses change profile picture button in change user profile menu
Success Guarantee: User profile picture is changed
Preconditions: User has a registered account and is in the user profile menu
Main Success Scenario:

1. User selects change profile picture button
2. User is prompted to select photo from phone gallery
3. User profile picture is changed to the selected profile picture

Extensions:

Use-cases4: change username
Actors: User
Triggering Event: User presses change username button
Success Guarantee: User username is changed
Preconditions: User has a registered account and is in the user profile menu
Main Success Scenario:

1. User selects change username button
2. User is prompted with textbox
3. User types in new desired username
4. User username is changed

Extensions:

Use-cases5: change password
Actors: User
Triggering Event: User presses change password button
Success Guarantee: User password is changed
Preconditions: User has a registered account and is in the user profile menu
Main Success Scenario:

1. User selects change password button
2. User is prompted with textbox
3. User types in new desired password
4. User username is changed

Extensions:

Use-cases6: delete account
Actors: User
Triggering Event: User clicks delete user account button

Success Guarantee: User account is deleted from database
Preconditions: User has registered account and is in profile menu button
Main Success Scenario:

1. User selects delete user account button
2. User is given a confirmation of if they really want to delete user account or not
3. User is prompted in a textbox to enter their password
4. User inputs correct password and account is set to be terminated in the database

Extensions:

Use-cases7: leave profile menu
Actors: User
Triggering Event: User clicks back button on profile menu
Success Guarantee: User is returned to the home page
Main Success Scenario:

1. User selects back button on profile menu
2. User is sent back to home page

Extensions:

Use-cases8: Leave Review
Actors: User
Triggering Event: User click comments box and write info in the box to commit
Success Guarantee: Comments are in the database and other user can see it
Preconditions: There is a database for the comments
Main Success Scenario:

1. The comments box has been selected
2. User type in some words
3. User selected done
4. Comments upload to the server database

Extensions:

User can edit the comments after pushing to the server

Use-cases9: View recommendations
Actors: User, Guest User
Triggering Event: User clicks on the item they want to view recommendations on
Success Guarantee: recommendations are opened
Preconditions: none
Main Success Scenario:

1. Recommendation page is opened
2. User can look through their recommendations
3. User can click on an item to leave a review

Extensions:

Use-Cases10: Ban user's account
Actors: Admin
Triggering Event: Admin selected the user and click ban selection
Success Guarantee: A banned user can't log in
Preconditions: A user violate the rules and user account is in active status
Main Success Scenario:

1. User has been selected
2. Admin ban account
3. The account has been diabled and can't login anymore

Extensions:

Admin can re-enable the account if that is a mistake.

Use-Case11: Delete comments
Actors: Admin
Triggering Event: Admin spot some toxic content in the comment
Success Guarantee: delete the comment in the DB
Preconditions: User posted comments
Main Success Scenario:

1. Society is more friendly and less dirty.
2. Users can delete comments themselves if they want to.

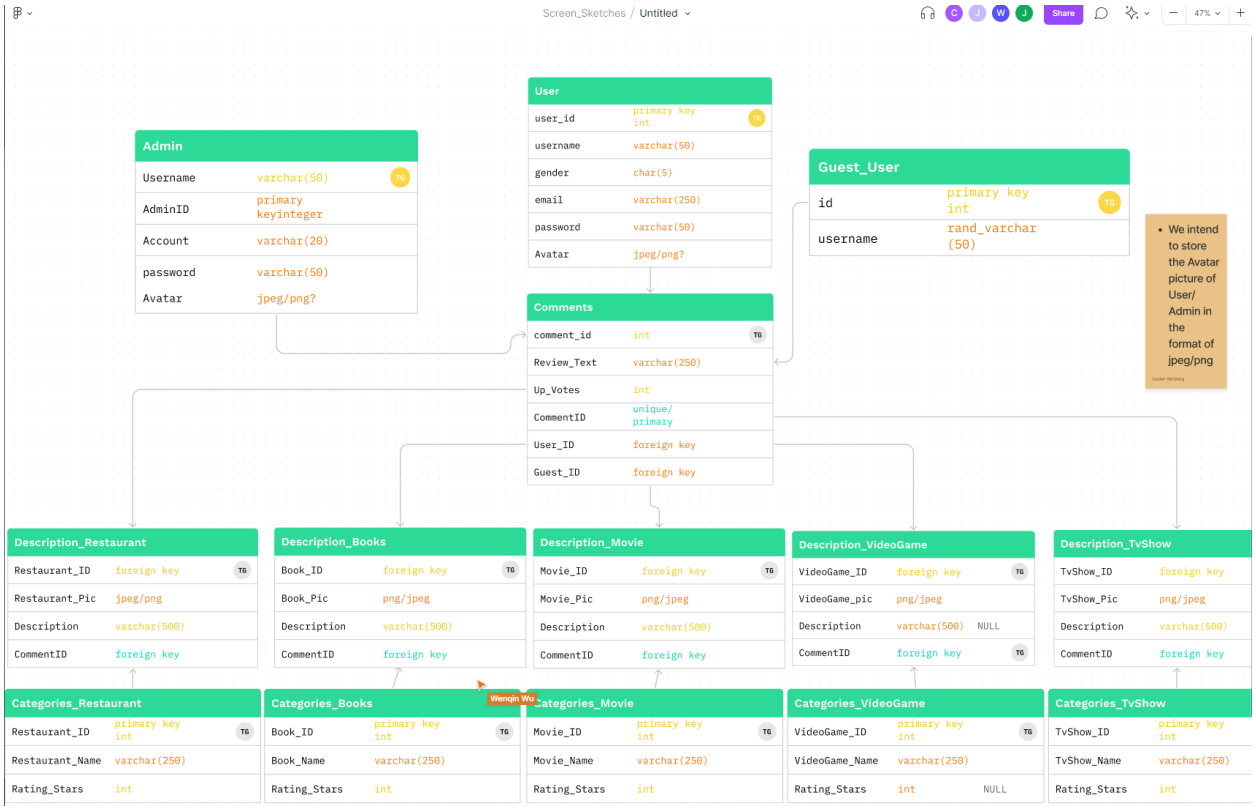
Extensions:

Add a report function so that the user can as admin to delete some potential toxic comment.

Step 2:

Step 3:

Tables_Relationship



Roles tables:

User table:

Primary Key Int: UserID

Attributes: Username(Varchar), Password(Varchar), Avatar(jpeg), gender(varchar), email(varchar)

Foreign Key:

Guest_User:

Primary Key Int: rand_int

Attributes: Username(Varchar),

Admin table:

Attributes: Username(Varchar), Password(Varchar), AdminID(Integer), account(varchar)

Primary Key Int: AdminID

Functional tables:

Comments table:
Attributes: CommentID(Integer), ReviewText(Varchar), upVotes(Integer), userID, guestID
Primary Key Int: CommentID
Foreign Key: UserID, guestID
—----------

Categories_Movie:
Attributes: Movie_name(Varchar), Rating_stars(integer)
Primary Key: Movie_ID (unique)

Description_Movie:
Attributes: Movie_pic(Unsure type), description(Varchar)
Foreign Key: Movie_ID, CommentID, Movie_name
—----------

Categories_VideoGame:
Primary Key: VideoGame_ID
Attributes: VideoGame_ID(Varchar), Rating_stars(integer)
Varchar: VideoGame_name

Description_VideoGame:
Attributes: VideoGame_pic(Unsure type), description(Varchar)
Foreign Key: VideoGame_ID, CommentID, VideoGame_name
—----------

Categories_Restaurant:
Attributes: Resturaunt_name(Varchar), Rating_stars(integer)
Primary Key: Resturaunt_ID (unique)

Description_Restaurant:
Attributes: Resturaunt_pic(Unsure type), description(Varchar
Foreign Key: Resturaunt_ID, CommentID, Resturaunt_name
—----------

Categories_TvShow:
Attributes: TvShow_name(Varchar), Rating_stars(integer)
Primary Key: TvShow_ID (unique)

Description_TvShow:
Attributes: TvShow_pic(Unsure type), description(Varchar)
Foreign Key: TvShow_ID, CommentID, TvShow_name
—----------

Categories_Books:
Attributes: Book_name(Varchar), Rating_stars(integer)
Primary Key: Book_ID (unique)

Description_Books:
Attributes: Book_pic(Unsure type), description(Varchar)
Foreign Key: Book_ID, CommentID, Book_name
—----------