Software Engineering CSC648/848

PicturePerfect

Team 01 / Section 01
Alekya Bairaboina - Team Lead
Nandhi Kanti Vinay Kumar - Frontend Lead
Ishika Shah – Backend Lead
Jacob Lawrence – Github & Scrum Master
Nic Burns – Product Owner

“Milestone 2”

March 27, 2023

Table of Contents

Revision History Table

Revision ID	Revision Date	Revised By
1	March 20,2023	Team1
2	March 26, 2023	Team1

Data Definitions V2
Revision from M1

Collection Name	Definitions and Attributes	Usage

General User	*A general user has basic facilities like account creation, login, editing and deleting their own posts, making their posts, edit their images before creating a post, searching for a specific post etc. They also can delete any comment made on their own post, for example, in cases, where the original poster deems a comment as inappropriate.*	
	Post	A given user can make, delete and edit their own posts.
	Comments	A given user can comment on any post made public.
	Likes	A given user can like any post made public.

Admin User	*An admin has the privilege of deleting any given post, comment, banning and deleting any account and viewing monetary compensation details for any given post*.	
	Post	A given admin can delete/view any post.
	Comments	A given admin can delete/view any comment.
	General User accounts	A given admin can delete/view any General User account.
Database Admin	*A database admin has the privilege of viewing any database details as and when needed, ranging from MySQL tables to the photos posted*.	
	MySQL tables	

		A given Database Admin can view/edit the MySQL tables as and when deemed required.
User Profile	*A user profile is basically a short view of a given user's details, like, name, age, employment type, headline etc*.	
	Basic details	The basic details of a given user are specified in this component.
	Posts	The posts made by the given user are also displayed on their profile.
Comments	*A comment is a short piece written by a user to describe their opinions related to the post.*	
	Delete	Only the original poster and an admin user can delete a given comment.

	Write	Any given user can write a comment.
Posts	*A Post is a short description of a given image which is posted.*	
	Comments	Any given user can write a comment, but, only the original poster and an admin user can delete a given comment.
	Like	A Post maybe liked by any given user.
	Dislike	A Post maybe disliked by any given user.
	Tags	A post may have more than one tag(s), which are used to segregate posts. They could be defined based on the type of photo, location of a photo etc.

Functional Requirements V2

	Must Have		Desired		Opportunistic

S.NO		Functional Requirement Description	Details
	1	Create an account (login, signup and logout)	1.1) Users are required to create an account (need to first register, then login and logout) 1.2) We will just use basic information of the users such as username/email and password
	2	Search based on keywords/tags	2.1)Users can search for images based on specific keywords or tags based on categories, topics, or hashtags.

	3	Users can upload, share, repost by tagging original users and host images/photographs online	3.1) Users can upload images from their devices 3.2)Users can share their images and other users can repost the original pictures by tagging the original user who uploaded them.
	4	Commenting	4.1)Users can add comments on the photos and the number of comments will be shown 4.2)Users will have control over the comment section, and they can delete any comments they find offensive.
	5	liking	5.1) Users can like the photos posted and the number likes will be shown

			5.2) Users can dislike the photos posted as well and number of dislikes will be shown
	6	Photo organization and tagging	6.1)Users can organize their photos into albums to make it simpler to search for and share images related to any specific themes or occasions. 6.2)Users can add meaningful hashtags to their images to make them easier to find.
	7	Image Editing	7.1) Users can add text to their images by using text tools for captions, watermarks, and other purposes. 7.2)Users can crop, resize, rotate, and can also add filters to the images

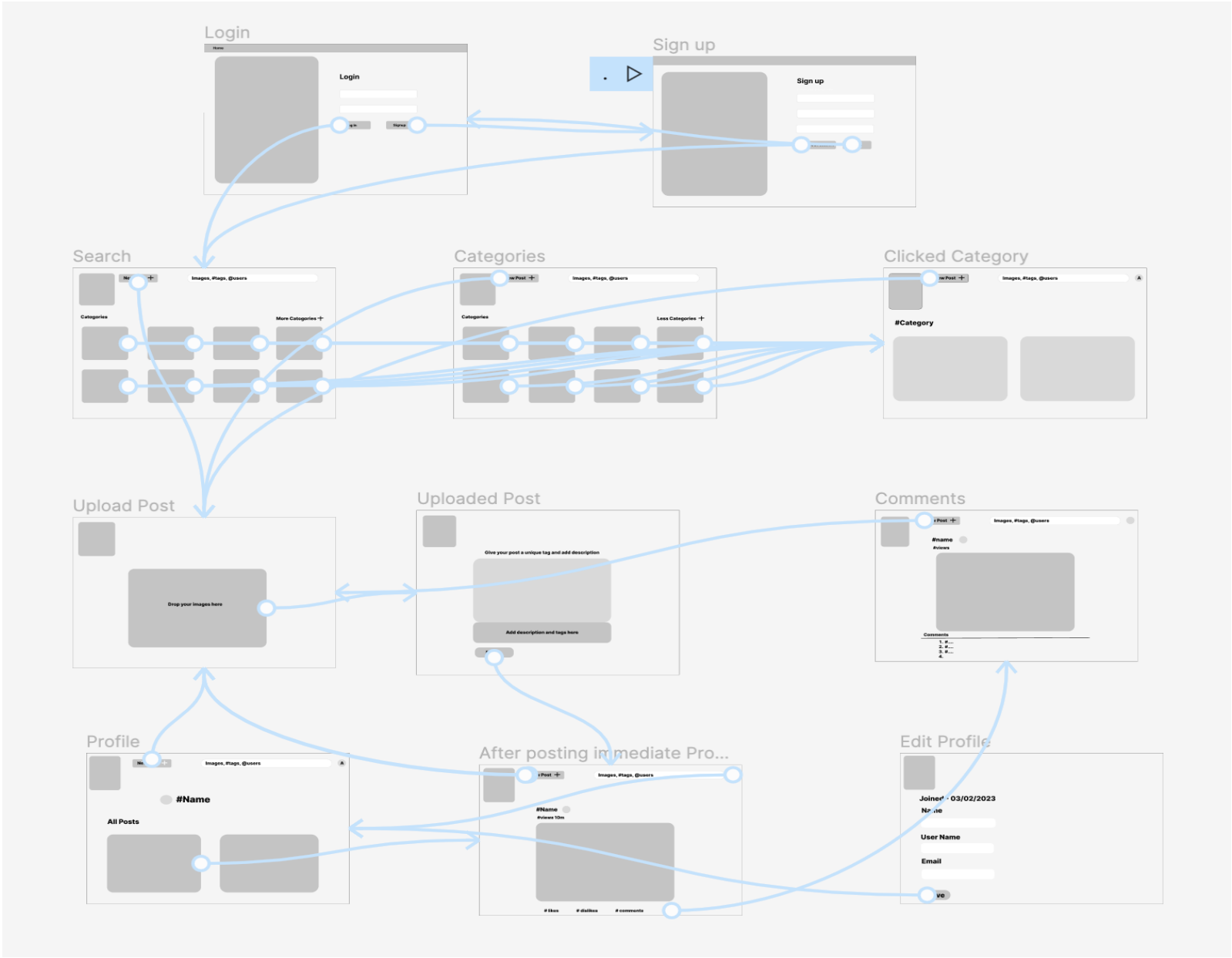
	8	Ads and sponsored content filtration	8.1) Users can be provided with no ads and sponsored content so that they can focus on the content without distractions. Our application will automatically remove adds, we will not have any ads or sponsored content in our application 8.2)Users would get revenue based on the number of clicks and downloads of the images.
	9	Privacy policies	9.1)Data about the users such as browsing activity, location information, and device information are not collected and users are given transparency into how their

			personal information is collected, stored, and used by providing a log of user activities on the platform. This log will indicate when the user logged in, logged out, and when they posted. Users can access this log through their personal profile, allowing them to monitor their activities on the platform and ensure that their personal information is being used appropriately.Only their username, name and email address will be collected. 9.2)Users can have the option of anonymous uploads without providing personal identifying information.

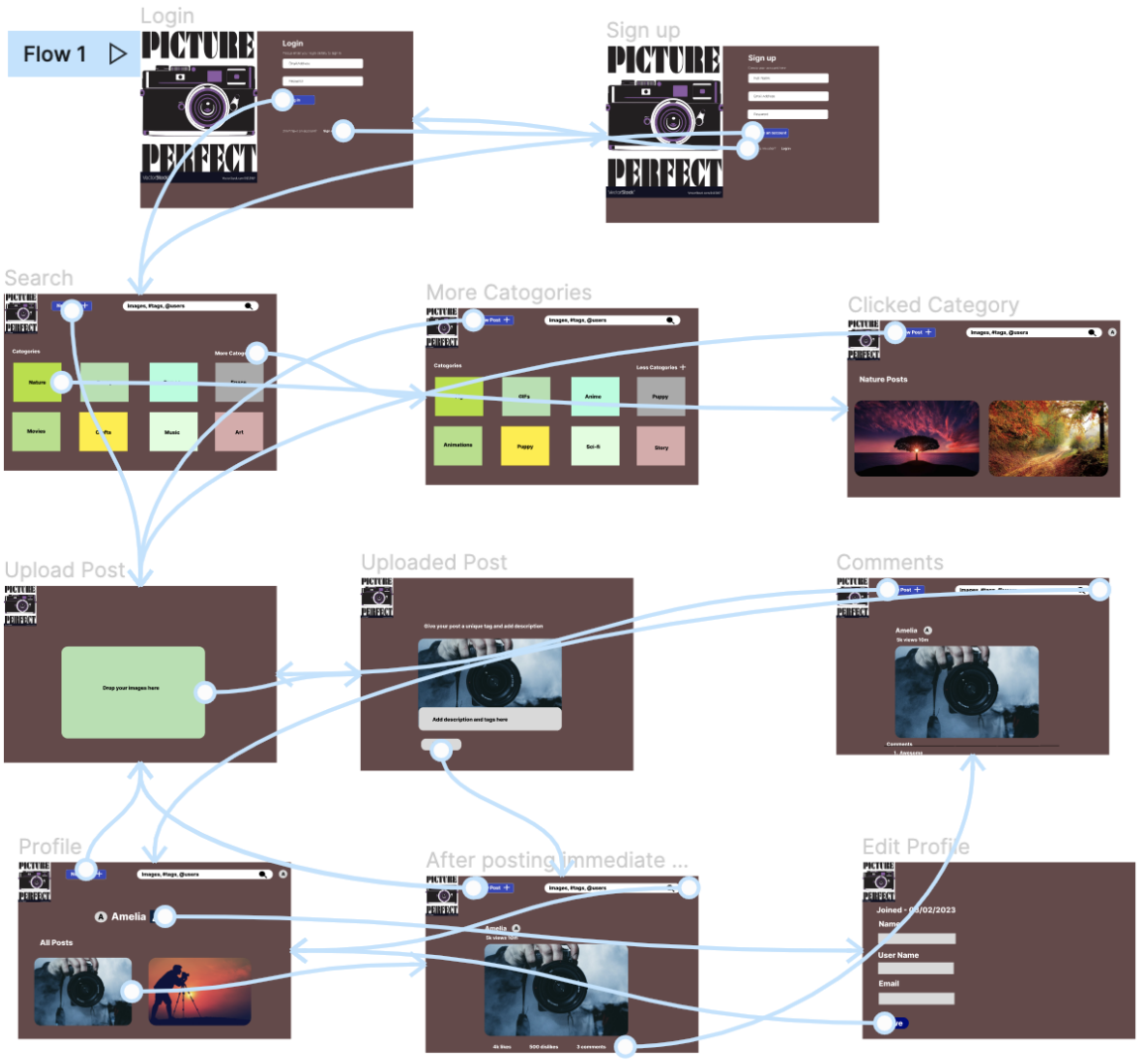
	10	Image sorting	10.1)Daily or weekly showcases of new or less popular images that have been uploaded, so that both new photographers and less popular images have the same chance of being seen by users. 10.2)Images are shown based on filters, we are going to have the latest posted and most popular filters that can be selected.

UI Mockups and storyboards

1. Gray and white wire Diagram


Login
Login
Sign up
Sign up
Search
Images, #tags, @users
Categories
More Categories +
Categories
Images, #tags, @users
Categories
Less Categories +
Clicked Category
Images, #tags, @users
#Category
Upload Post
Drop your images here
Uploaded Post
Give your post a unique tag and add description
Add description and tags here
Comments
Images, #tags, @users
#name
#views
Comments
Profile
Images, #tags, @users
#Name
All Posts
After posting immediate Pro...
Images, #tags, @users
#Name
#views 10m
likes
dislikes
comments
Edit Profile
Joined : 03/02/2023
User Name
Email

2. GUI Prototype



Link for the above GUI Prototype and gray and white wire frames of our application -
https://www.figma.com/file/Xme6qsYAs4Db9xThflGr5G/Picture-Perfect-app?node-id=0%3A1&t=WT6XpvEJbXHJPo5q-1

Meeting minutes

Meeting minutes (09-02-23):

- We discussed the methodology and learning plan we will follow.
- We discussed the plan for next week.

Meeting minutes (22-02-23):

- We discussed the Milestone 1 document and determined Monday as the draft 1 deadline
- Determined next development step to be UI design
- All group members in attendance

Meeting minutes (22-03-01):

- We caught up on progress regarding our M1 document

- Determined the deadline for the final draft of the M1 document
- Determined actionable improvements to the existing documentation
- All group members in attendance

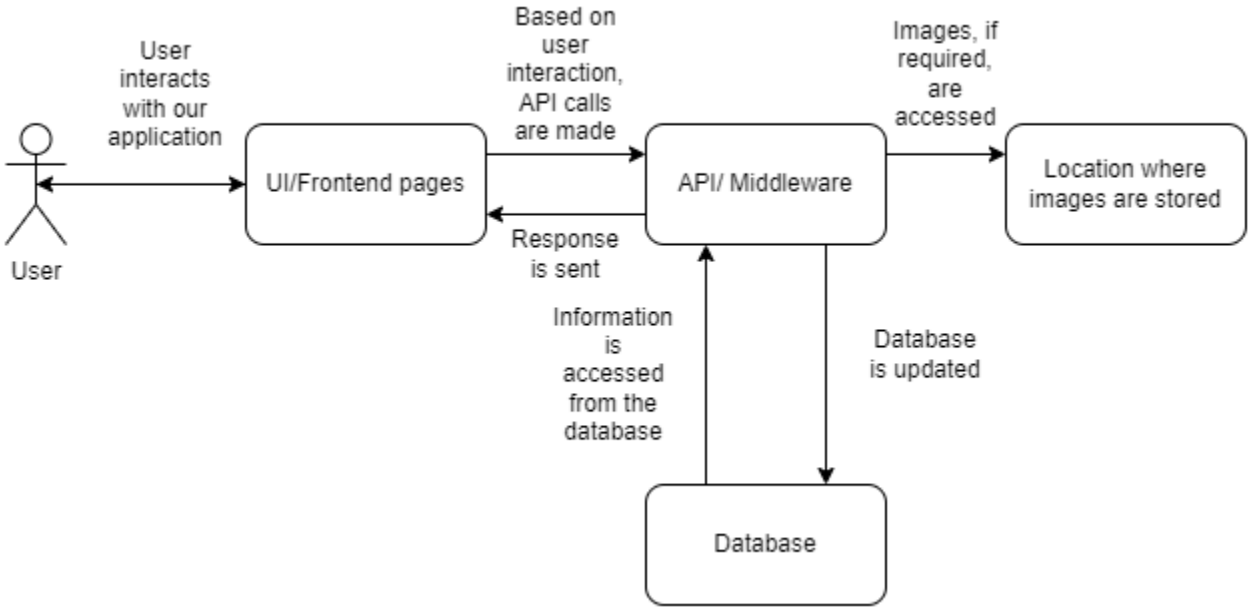
Meeting minutes (22-03-08):

- Decided M2 task allocation
- Decided deadlines for the tasks
- All group members in attendance

Meeting minutes (22-03-13)

- Updated progress
- Intend to finish M2 document by 03-15

High Level Architecture



Database Organization

Table name	Columns	Datatype and constraints
User	Name	VARCHAR()
	Email	VARCHAR();

		UNIQUE
	Userid	VARCHAR(); PRIMARY KEY
	Password	ENCRYPTED VARCHAR()
	DOB	DATE
	Date_joined	DATETIME
	Username	VARCHAR(); UNIQUE
	Phone_number	VARCHAR(10)
	User_pic	VARCHAR() as this is a path to the photo
	About	VARCHAR()

	User_type	VARCHAR()
Posts	Made_by	VARCHAR() FOREIGN KEY TO USER[USERID]
	Creation_date	DATETIME
	No_of_likes	INT
	No_of_dislikes	INT
	Points	FLOAT
	Is_reshared	BOOLEAN

	Post_id	VARCHAR(); PRIMARY KEY
	No_of_views	INT
	No_of_comments	INT
	Image_path	VARCHAR() as this is a path to the photo
	Description	VARCHAR()
Tags	Tag_id	VARCHAR(); PRIMARY KEY
	Name	VARCHAR()
	Descripti	VARCHAR()

	on	
Post_tags	Post_id	VARCHAR() FOREIGN KEY TO POSTS[POST_ID]
	Tag_id	VARCHAR() FOREIGN KEY TO TAGS[TAG_ID]
Comments	Post_id	VARCHAR() FOREIGN KEY TO POSTS[POST_ID]
	Comment_id	VARCHAR(); PRIMARY KEY
	Comment_made_	VARCHAR() FOREIGN KEY TO

	by	USER[USERID]
	comment_datetime	DATETIME
	Comment_desc	VARCHAR()
	Comment_likes	INT
	Comment_dislikes	INT

Operations allowed on tables

Table name	Insert	Update	Delete
User	✓	✓	✓
Posts	✓	✓	✓
Tags	✓	✗	✗
Post_Tags	✓	✓	✓
Comments	✓	✗	✓

Technical feasibility for DB Operations

- The proposed database schema for a social media platform appears technically feasible and can support the expected database operations. Here are some technical considerations for each operation:
-
- The User table has a unique identifier UserId as the primary key, which can ensure the uniqueness of each user record. The email and username fields are marked as unique, so no two users can have the same email or username. The password field is encrypted to ensure security. The phone_number field has a maximum length of 10 characters, which is suitable for most phone number formats.
- The Posts table has a unique identifier Post_id as the primary key, which can ensure the uniqueness of each post record. The foreign key constraint with the User table can ensure that each post record belongs to a valid user. The No_of_likes, No_of_dislikes, and No_of_comments fields are integer types, which can efficiently store and manipulate the corresponding numerical data. The Points field has a float data type, which can store decimal values. The

Is_reshared field is a boolean data type, which can efficiently represent the binary state of whether a post is a reshare or not.

- The Tags table has a unique identifier as the primary key, which can ensure the uniqueness of each tag record. The Name and Description fields are varchar types, which can efficiently store and manipulate text data. The Post_tags table has foreign key constraints with both Posts and Tags tables, which can ensure that each post-tag record is valid and belongs to a specific post and tag.
- The Comments table has a unique identifier Comment_id as the primary key, which can ensure the uniqueness of each comment record. The foreign key constraint Post_id with the Posts table can ensure that each comment record belongs to a valid post. The Comment_made_by field has a foreign key constraint with the User table, which can ensure that each comment is made by a valid user. The Comment_likes and Comment_dislikes fields are integer types, which can efficiently store and manipulate the corresponding numerical data.
- Overall, the proposed database schema is technically feasible. It supports the add, delete, and search operations efficiently. The required fields, primary keys, foreign keys, unique constraints, and indexes are defined correctly, ensuring data integrity and efficient query execution.

Additional Information (API signature)

General user APIs

Screen	Usage	API endpoint	Request Method	Input payload	Response structure	Comments

Homepage	API to login to the platform	/user_login	POST	{
				"username": string,
				"password": encrypted string,
				"user_type": string from options ("general", "admin")
				}
					{
					"status": "SUCCESS/FAILED",
					"isLoggedin": boolean,
					“userId”: string,
					"message": string
					}
						This API will validate user input data and confirm if the user is valid based on the info

						entered.

	API to register a new user	/register_user	POST	{
				"name": string,
				"email": string,
				"password": encrypted string,
				"dob": date,
				"username": string,
				"phonenum": string,
				"userpic": image,
				"about": string,
				"usertype": string
					{
					"status": "SUCCESS/FAILED",
					"isRegistered": boolean,
					"isUnique": boolean,
					"message": string
					}
						This API will register a user i.e., add the user to DB.

				}		

	API to view all public posts	/view_public_posts	POST	{
				"limit": int,
				"offset": int,
				"searchText": string,
				"sortby": string,
				“sortType”: “ASC/DESC”
				}
					{
					"status": "SUCCESS/FAILED",
					“noOfPosts”: int,
					"posts": [
					{ "post_id": string,
					"creation_date": datetime,
					"made_by": string,
					"no_likes": int,
						This API supports paginati on, sorting as well as search

					"no_dislikes": int, "isReshared": boolean, "no_views": int, "no_comments": int, "image": image, "desc": string },...], "message": string }	
	API to create new post	/create_post	POST	{ "username": string, "is_reshared": boolean, "image": image,	{ "status": "SUCCESS/FAILED", "isPostCreated": boolean,	This API is used to create post for a given user.

				"description": string }	"postid": string, "message": string }	

	API to like/dislike post	/like_dislike_post	POST	{
				"postid": string,
				"liked": boolean
				}
					{
					"status": "SUCCESS/FAILED",
					"isUpdated": boolean,
					"message": string
					}
						This API is used to update likes or dislikes for a post.

	API to add comment(s)	/add_comment	POST	{
				"postid": string,
				"comment": string,
				"username": string
				}
					{
					"status": "SUCCESS/FAILED",
					"isCommentAdded": boolean,
					"message": string
						This API is used to insert a comment to the DB.

					}	
	API to logout if user is logged in	/logout	POST	{ "username": string }	{ "status": "SUCCESS/FAILED", "isLoggedout": boolean, "message": string }	This API helps the user to logout from the system.
Display Post	API to logout if user is logged	/logout	POST	{ "username": string }	{ "status": "SUCCESS/FAILED",	This API helps the user to logout

	in				"isLoggedout": boolean, "message": string }	from the system.
	API to get post details like comment, likes etc.	/get_post_details	POST	{ "postid": string }	{ "status": "SUCCESS/FAILED", "post": { "post_id": string, "creation_date": datetime, "made_by": string, "no_likes": int, "no_dislikes": int, "isReshared": boolean,	This API is used to get details of a single post.

					"no_views": int, "no_comments": int, "image": image, "desc": string }, "message": string }	

	API to add 1 count to total views for given post	/add_view	POST	{
				"postid": string
				}
					{
					"status": "SUCCESS/FAILED",
					"isViewUpdated": boolean,
					"message": string
					}
						This API is used to update number of views for a given post.

	API to delete comment	/delete_comment	POST	{
				"postid": string,
				"commentid": string,
				"username": string
				}
					{
					"status": "SUCCESS/FAILED",
					"isCommentDeleted": boolean,
					"message": string
					}
						This API is used to delete a particular comment on the current logged in user's post.

	API to like/dislike post	/like_dislike_post	POST	{
				"postid": string,
				"liked": boolean
				}
					{
					"status": "SUCCESS/FAILED",
					"isUpdated": boolean,
						This API is used to update likes or dislikes

					"message": string }	on a given post.

	API to add comment(s)	/add_comment	POST	{
				"postid": string,
				"comment": string,
				"username": string
				}
					{
					"status": "SUCCESS/FAILED",
					"isCommentAdded": boolean,
					"message": string
					}
						This API is used to add comments ona given post.

User profile – Basic info tab	API to logout if user is logged	/logout	POST	{
				"username": string
				}
					{
					"status": "SUCCESS/FAILED",
						This API helps the user to logout

	in				"isLoggedout": boolean, "message": string }	from the system.

	API to view profile details	/view_user_profile	POST	{
				"username": string
				}
					{
					"status": "SUCCESS/FAILED",
					"name": string,
					"email": string,
					"dob": date,
					"username": string,
					"phonenum": string,
					"userpic": image,
					"about": string,
					"usertype":
						This API is used to display the details of a user profile.

					string, "message": string }	

	API to update profile details	/update_user_profile	POST	{
				"username": string,
				"updates": [{
				"updatedColumn": string,
				"updatedValue": string
				},...]
				}
					{
					"status": "SUCCESS/FAILED",
					"isUpdated": boolean,
					"message": string
					}
						This API is used to update user profile.

	API to update password	/update_password	POST	{
				"username": string,
				"updatedPassword": encrypted string
				}
					{
					"status": "SUCCESS/FAILED",
					"isUpdated": boolean,
					"message": string
					}
						This API is used to update the password for a given user.

	API to delete account	/delete_account	POST	{
				"username_to_delete": string
				}
					{
					"status": "SUCCESS/FAILED",
					"isDeleted": boolean,
					"message": string
					}
						This API is used to delete the account.

User profile – Posts tab	API to logout if user is logged in	/logout	POST	{
				"username": string
				}
					{
					"status": "SUCCESS/FAILED",
					"isLoggedout": boolean,
					"message": string
					}
						This API helps the user to logout from the system.

	API to list user posts	/list_user_posts	POST	{
				"limit": int,
				"username": string,
				"offset": int,
				"searchtext": string,
				"sortby": string
				}
					{
					"status": "SUCCESS/FAILED",
					"posts": [
					{ "post_id": string,
					"creation_date": datetime,
					"made_by": string,
					"no_likes": int,
					"no_dislikes": int,
					"isReshared": boolean,
					"no_views": int,
					"no_comments": int,
					"image": image,
					"desc": string
						This API lists the current logged in user's data.

					},...], "message": string }	

	API to delete post	/delete_post	POST	{
				"postid": string,
				"username": string
				}
					{
					"status": "SUCCESS/FAILED",
					"isPostDeleted": boolean,
					"message": string
					}
						This API is used to delete a user's post.

Admin user APIs remain more or less the same, except these few additional APIs

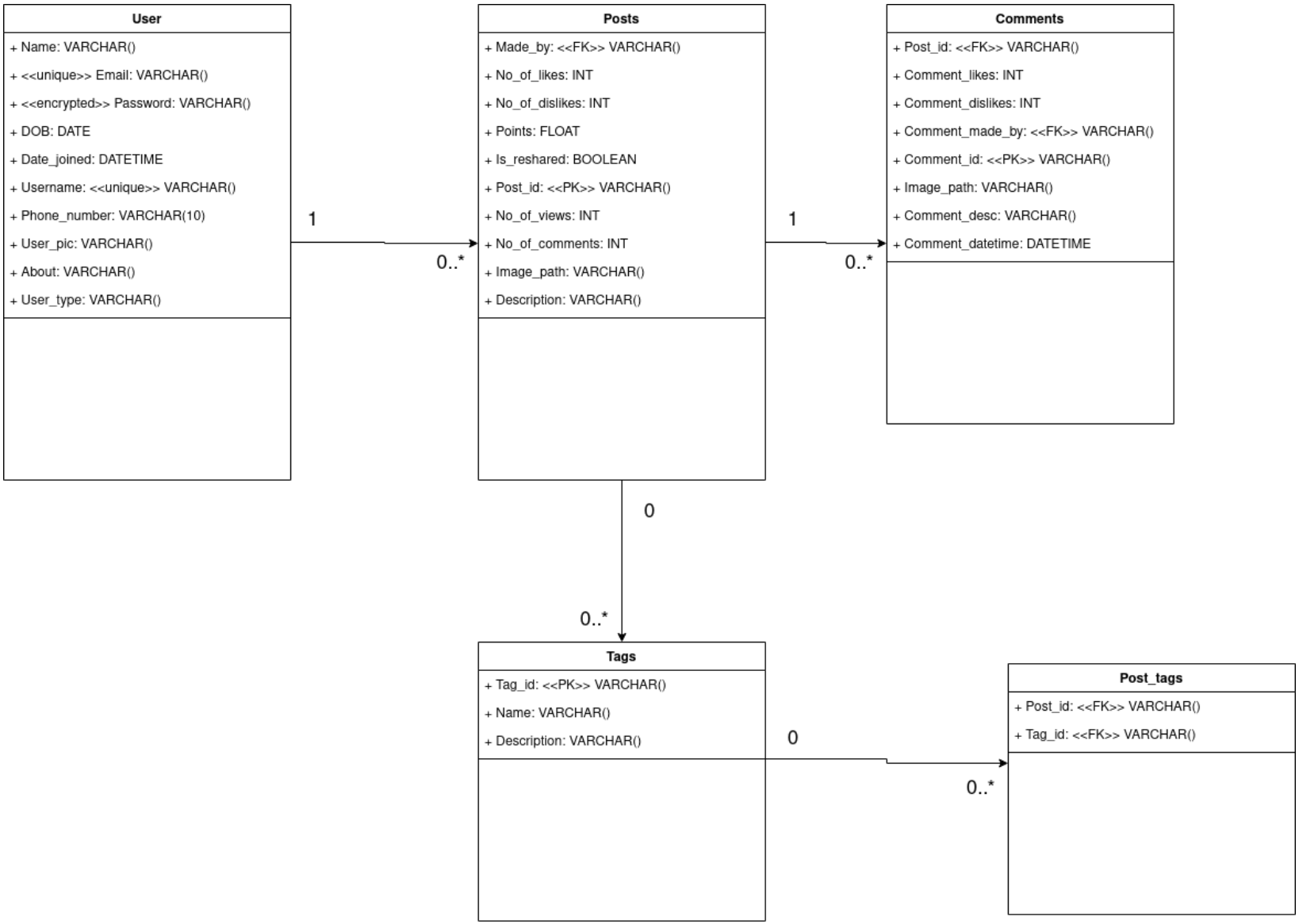
Screen	Usage	API endpoint	Request Method	Input payload	Response structure	Comments

Display Post	API to delete post	/delete_post	POST	{
				"postid": string,
				"username": string
				}
					{
					"status": "SUCCESS/FAILED",
					"isPostDeleted": boolean,
					"message": string
					}
						This API is used to delete any given post.

	API to delete any given	/delete_comment	POST	{ "postid": string, "commentid": string,	{ "status": "SUCCESS/FAILED",	This API is used to delete any

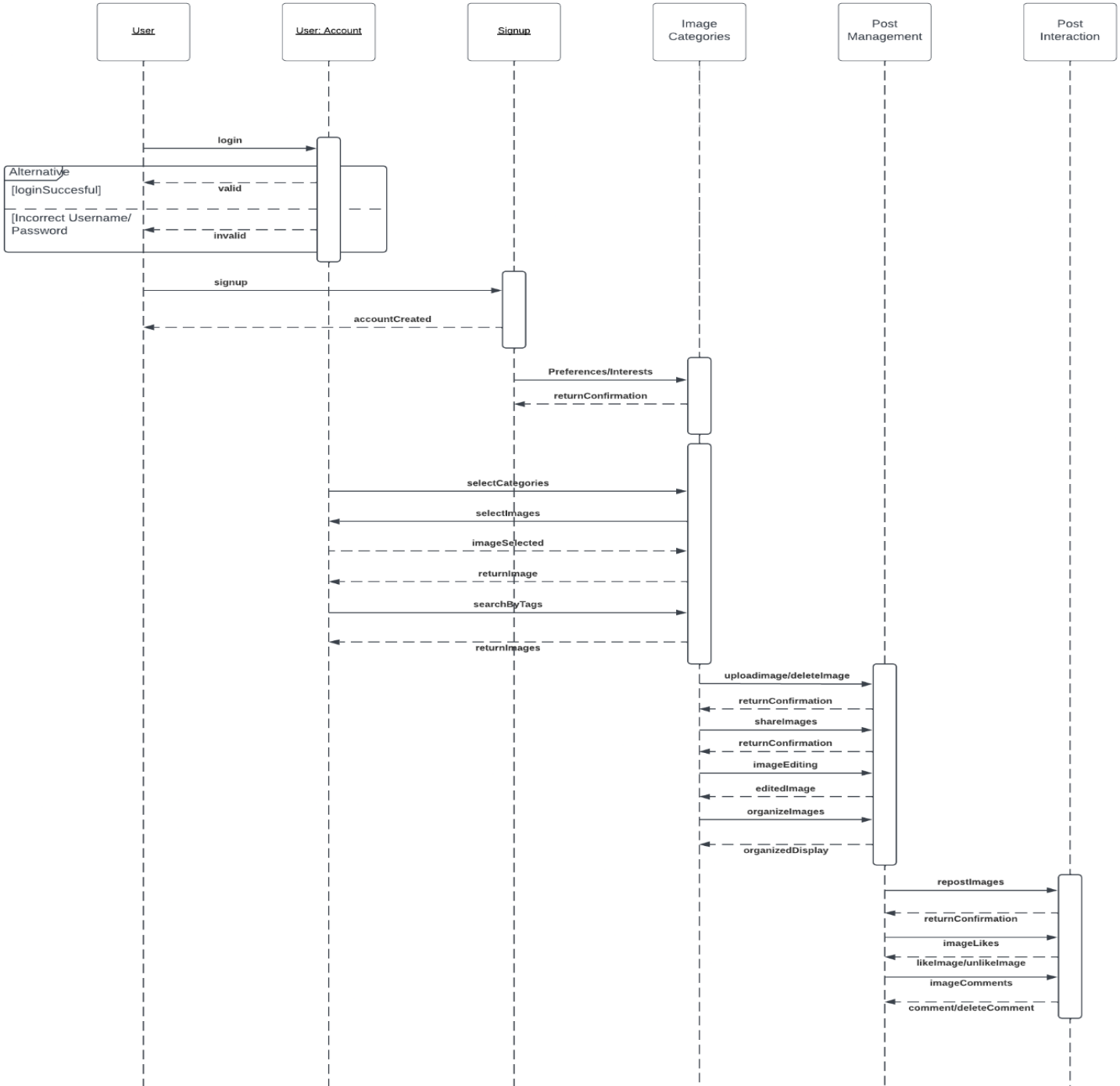
	comment made by any user			"username": string }	"isCommentDeleted": boolean, "message": string }	given comment.

High Level UML Diagrams



https://drive.google.com/file/d/1Hf_15WzH66Qnd_9-hYDaX0GGOIkTFd4m/view?usp=sharing

High-level sequence Diagram


User
User: Account
Signup
Image Categories
Post Management
Post Interaction
login
Alternative
[loginSuccesful]
valid
[Incorrect Username/ Password
invalid
signup
accountCreated
Preferences/Interests
returnConfirmation
selectCategories
selectImages
imageSelected
returnImage
searchByTags
returnImages
uploadimage/deleteImage
returnConfirmation
shareImages
returnConfirmation
imageEditing
editedImage
organizeImages
organizedDisplay
repostImages
returnConfirmation
imageLikes
likeImage/unlikeImage
imageComments
comment/deleteComment

Identify Actual key risks for the project at this time

Type of Risk	Risk Description	Mitigation Plan
Skills Risks	In case team member(s) are unable to catch up to the study plan owing to any factors such as personal life, assignments of other subjects etc, the team member(s) will not be skilled enough to work with others at the	Firstly, the team member is expected to inform the team lead and scrum master about the delay that they will be facing. Irrespective of that, the backend/frontend team will provide the necessary help according to the team member's

	required pace.	expertise on the required technology/tool. The team will take and give regular updates on their skillset.
	In case team member(s) are facing difficulties with a particular tool/technology with respect to learning it or using it.	The team member can ask for help in the team chat on slack channel, after which, any other team member with adequate resources and knowledge will help out.
	In case a team member is not able to move on to their	In such cases, the team member is expected to bring

	next task due to too many bugs in their task.	this up in the next scrum wherein the team lead and other members with similar skills will decide a plan to overcome such problems.
Scheduling Risks	In case the deadlines for the study plan are seeming steep and team member(s) are not able to follow it.	The team will discuss a more appropriate structure of the study plan and alter it accordingly.

	In case the deadlines for finishing given tasks within the given time frames is seeming hard.	The team member is expected to inform their concerns to the team lead after which the team lead will take measures to make sure there is no burnout or the team member has too less work. The deadlines will then be altered according to everyone's thoughts.
	In case the task has been altered or requires alteration.	The product owner will bring this up in the next scrum meeting, wherein

		everyone will be informed about the changes, as well as, the tasks and their respective details will be updated on the project tracker, which in our case is YouTrack (by Jetbrains).
Teamwork risks	In case someone is not able to make it to the scrum meeting at all or is not able to join on time.	The team member is expected to inform their availaibility to the scrum master. After the meeting, the scrum master will share the meeting minutes to

		all team members to make sure everyone is on the same page with respect to the project.
	In case a team member is not joining the scrum meeting at all.	The scrum master will keep a track on every team member's attendance in the scrum, and after a point inform the team lead of the same, who will then discuss the same with the absent member and appropriate actions

		will be taken.
	In case a team member is not able to keep up with the tasks regularly.	The team lead will then discuss the same with them and figure out the best steps that can be taken to make sure their is no burnout happening, nor is there too less work for anyone to work on. The work will be divided equally based on the skillset, level of learning etc.

	In case, miscommunication happens and a few tasks on hand are missed.	In this case, the tasks will be taken up on priority by everyone and will be aimed to finish at the earliest possible time.
	In case, a middleware engineer, the database manager and UI developer are not able to work hand in hand with each other due to any reasons such as less to no communication, time issues etc.	For this, the team members are expected to inform the scrum master, who will then work on finding the correct timings at which a combined discussion can happen between multiple subteams.

Legal Risks	In case a user uploads someone else's work/image/personal information without prior permission of the original poster, or without giving them credits for the same.	In these scenarios, the team will work on deleting said post and will take strict actions against the user, and the user will be banned from using our application.

Project Management:

From milestone 2, our team broke down all of the tasks, evenly and to the fairest degree amongst all the members. We all discussed what tasks we should work on then our team lead ultimately assigned us each task. This was based on our role and expertise on the applied topic. Our lead backend developer Ishika was in charge of architecture and database or organization, as well as data, definitions. Front end lead Vinay oversaw the creation of the wire diagram. Alekya helped out in front end development as well. The GUI mock-up was created by Alekya and It is a great representation of what we are trying to accomplish with our software. Also on the front end side. Jacob worked on creating high level UML diagrams. While simultaneously handling the session's minutes of meetings.

The biggest task of M2 is the prototype. Currently we are building the prototype. To track our progress we are using a software called YouTrack developed by Jetbrains. This was strongly recommended to use this program and it has been useful through and through. I(Nic) am the product owner. I am in charge of keeping track of everyone's tasks on YouTrack. The project management was also written by me.

Study Plan

1. Alekya – Team lead
2. Jacob – GitHub & Scrum master
3. Vinay – Front-end Lead
4. Ishika – Backend lead
5. Nic – Product Owner

Discrete Timeline:

March	Alekya & Vinay	Jacob	Ishika & Nic
03-08	~~Creating a React component~~	~~Basic Git command usage~~	~~Connecting to the DB via the cloud server~~

03-15	~~Routing amongst components via React~~	~~Proper branch management and etiquette~~	~~Using MySQL Workbench and basic SQL operations~~
03-22	~~Handling requests with Django~~	~~Resolving and preventing merge conflicts~~	~~Introductory SQL queries and commands~~
03-29	Displaying site data via Django	Cover rebasing	Top to bottom DB management
April	**Alekya & Vinay**	**Jacob & Ishika**	**Nic**

04-05	Iterate on frontend implementation	Iterate on backend implementation	Orchestrate product vision and improve user value
04-12	Alekya - Ensure on track development Vinay - Provide feedback on all UI	Ishika - Maintain software architecture Jacob - Ensure effective teamwork and development	Define exactly the product we are delivering
04-19	Prepare for M3 meetings	Prepare for M3 meetings	Prepare for M3 meetings