System Requirements Specification

Dekū

Client
Shawn Squire

Team 6
Boris Boiko
Raymond Chan
Gilbert Kuo
Andrew Naviasky
Jeremy Neal

4/30/2014

Table of Contents

1. Introduction

Welcome to the Dekū system requirements specification. Dekū is a social networking application, which can be thought of as Twitter but specifically targeted towards college students.

1.1 Purpose of this Document

This document is designed to explain the features of the Dekū web application, it's functions, and the conditions required for operation. The intended audience is the Dekū development team as well as the faculty customer, Shawn Squire.

1.2 Purpose of the Product

Dekū was designed to provide college students and faculty with a social networking web application. Users who use this site can post content (cards), favorite content (adding cards to your deck), as well as other features, like other social networking sites. The difference with Dekū and other social networking sites is that this web application is specifically geared towards college students and faculty to provide easy communication between certain topics regarding a specific college (administrative needs, classes, organizations, etc).

1.3 Product Scope

The Dekū web application consists of 24 use cases including things like: creating accounts, logging in, posting content, interacting with friends, etc. Please refer to the figures below for a further understanding of the actions which are available to users.

Figure 1. Use case 1: User logs in

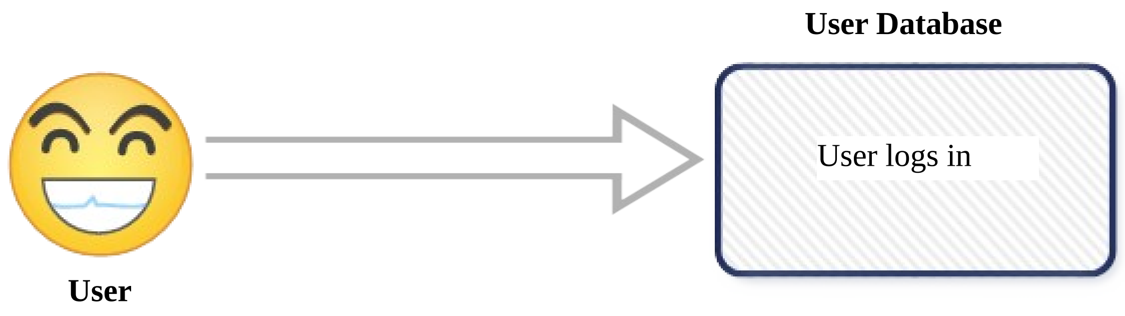


Figure 2. Use case 2: User creates account

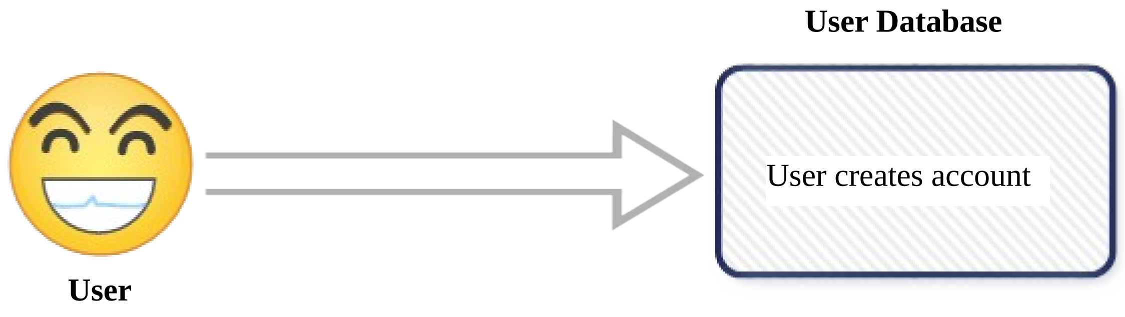


Figure 3. Use case 3: User loses password

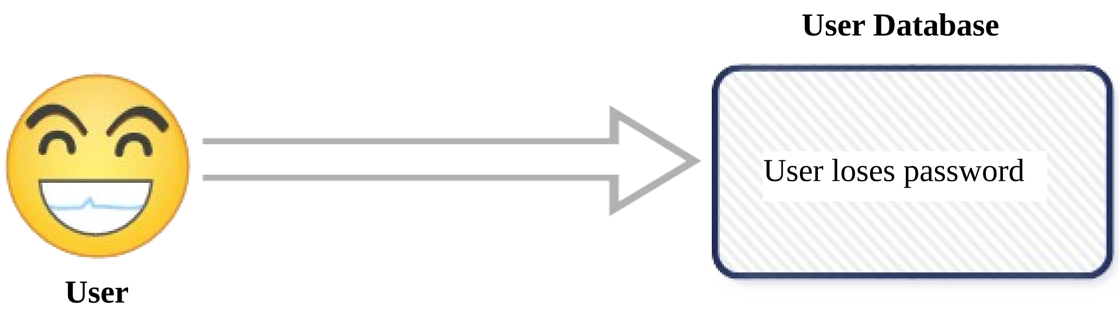


Figure 4. Use case 4: User deactivates account

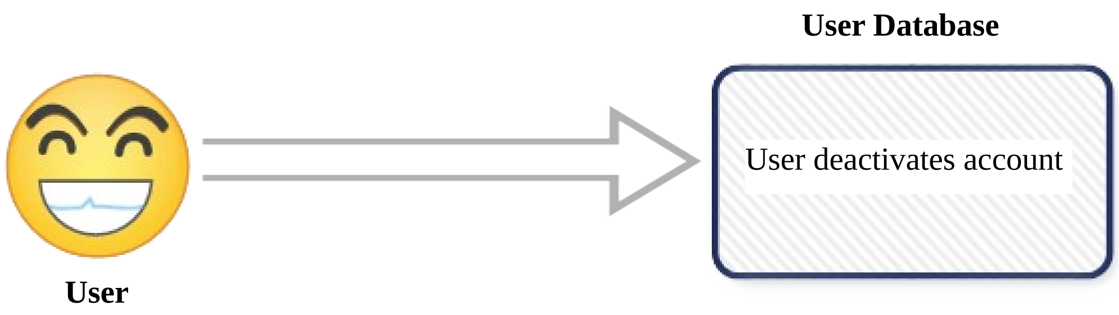


Figure 5. Use case 5: User deletes account

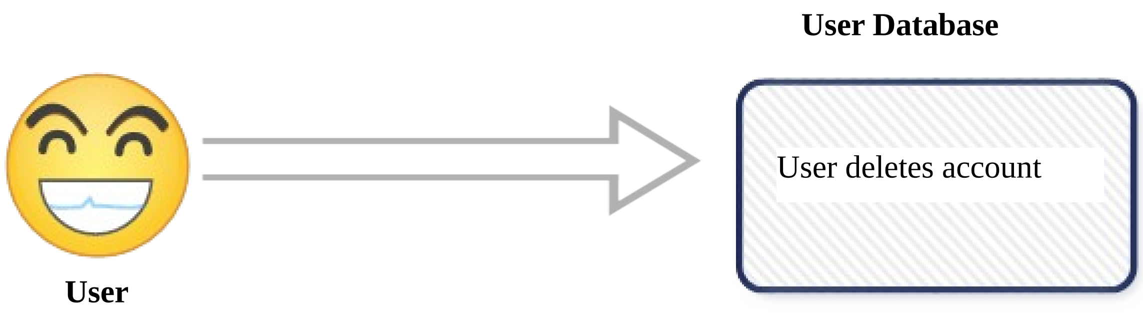


Figure 6. Use case 6: User creates a new card

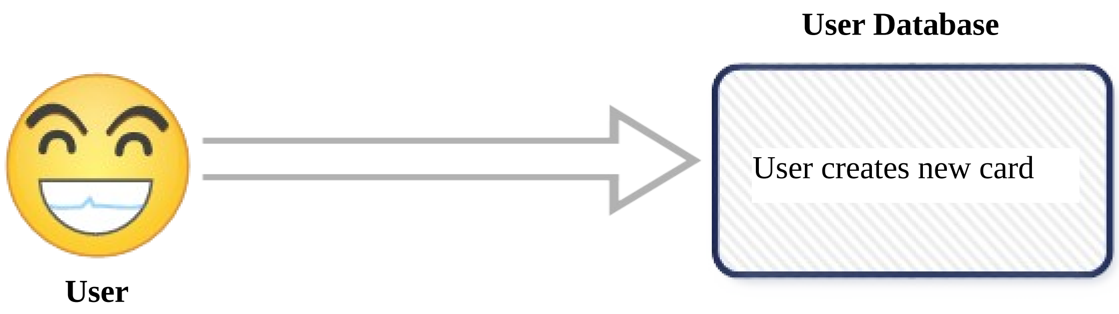


Figure 7. Use case 7: User searches by tag

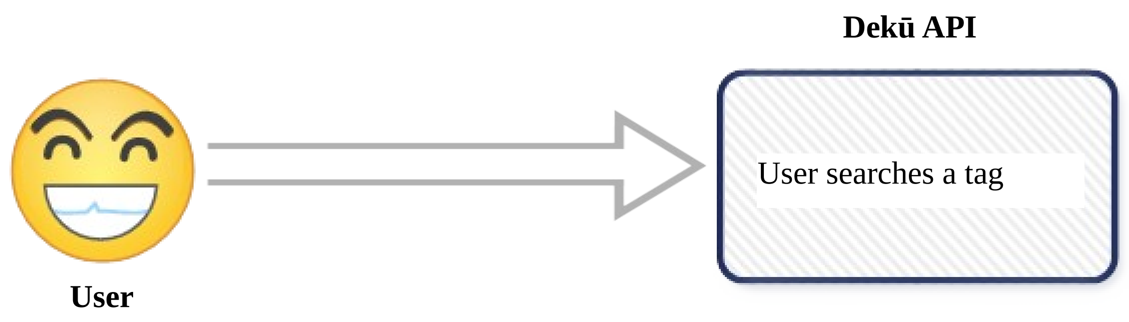


Figure 8. Use case 8: User inspects a card

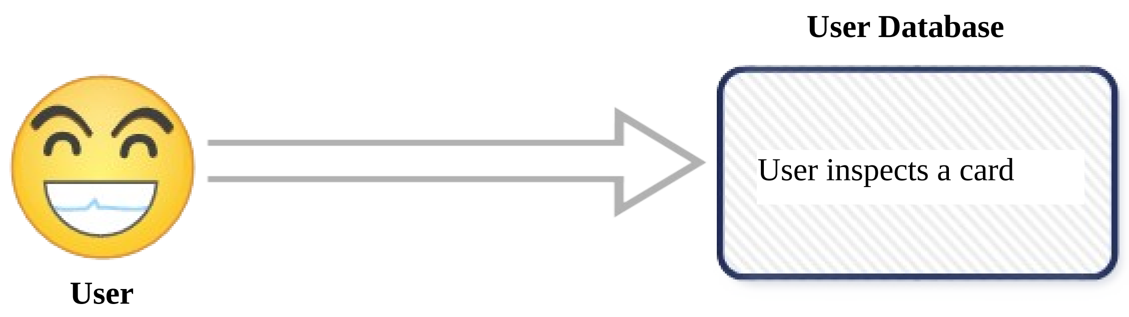


Figure 9. Use case 9: User watches other users

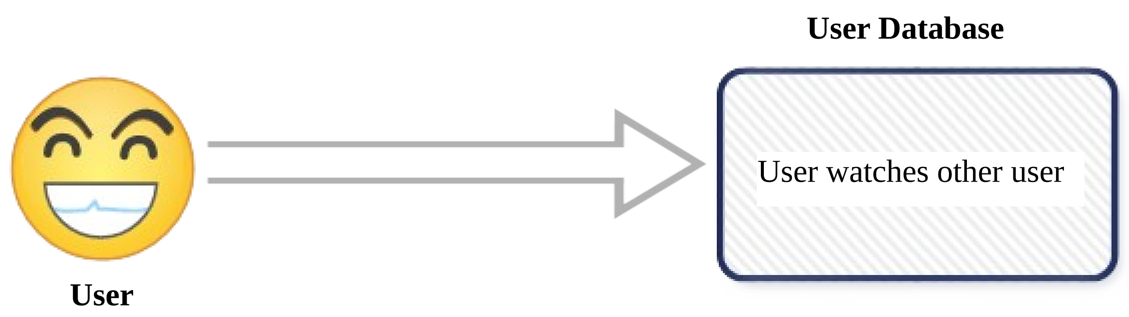


Figure 10. Use case 10: User stops watching another user

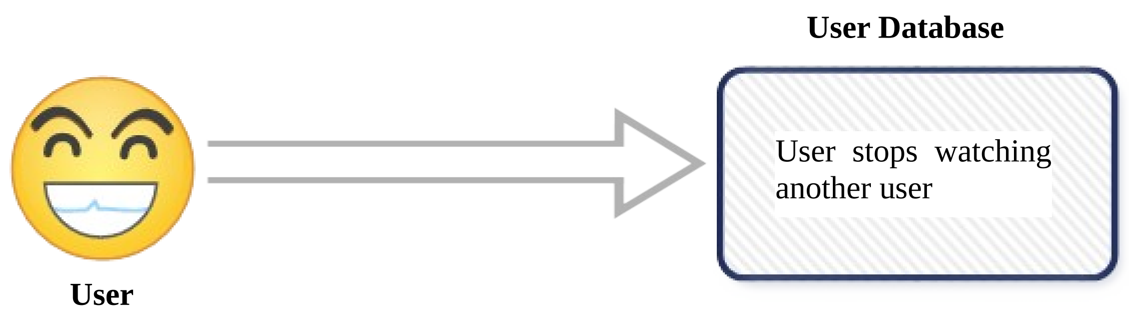


Figure 11. Use case 11: User can hide tag/creator

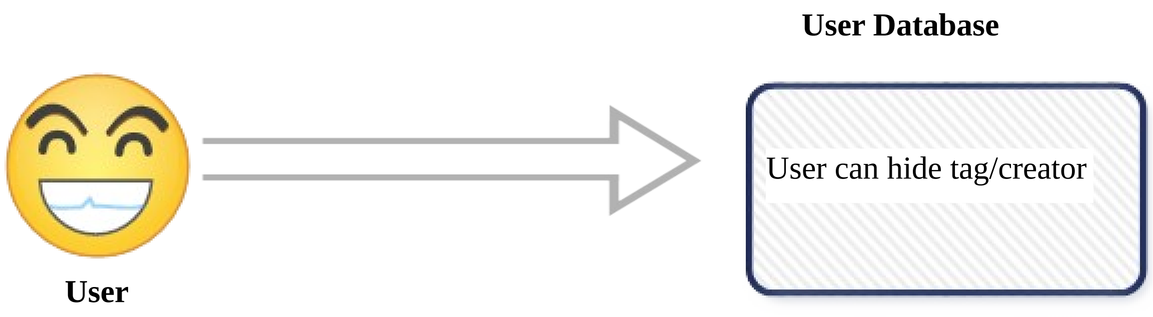


Figure 12. Use case 12: User deletes card

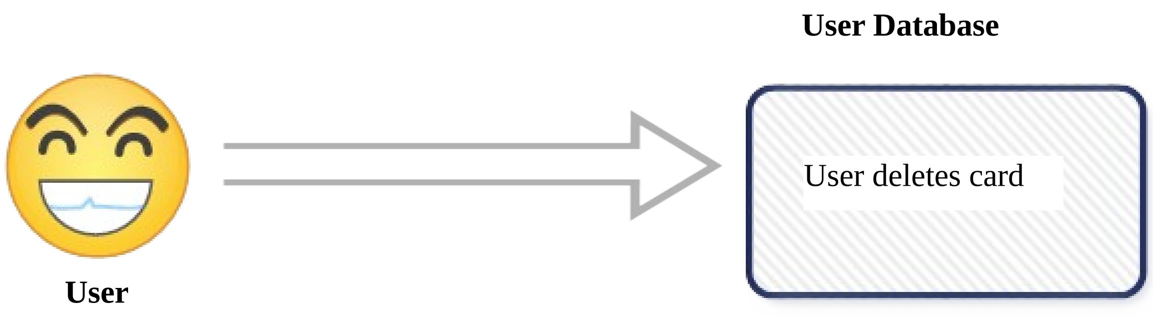


Figure 13. Use case 13: User views user profile

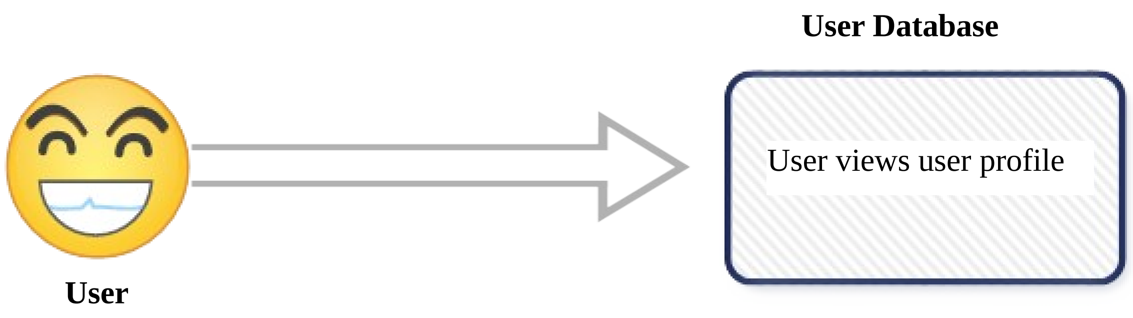


Figure 14. Use case 14: User has reputation

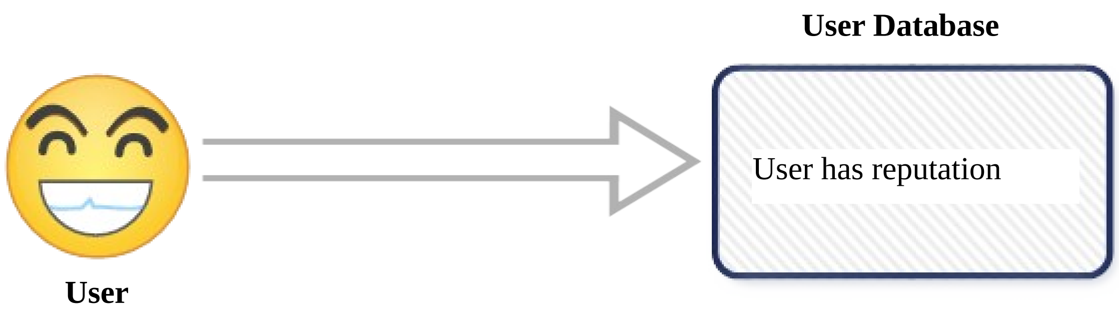


Figure 15. Use case 15: User receives notification

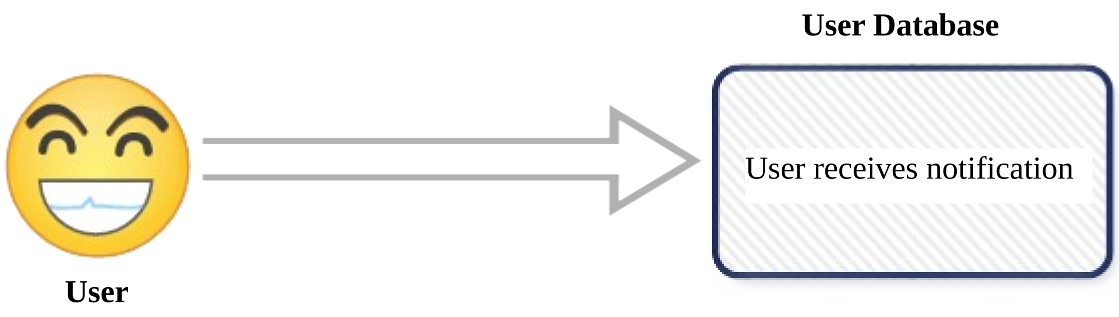


Figure 16. Use case 16: User logs out

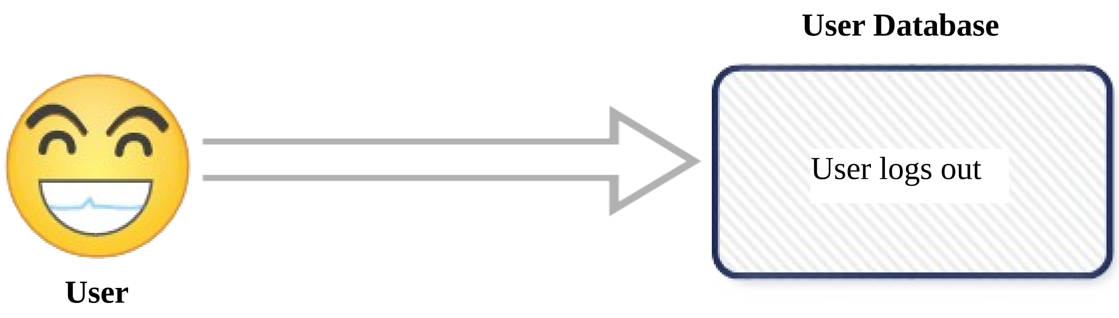


Figure 17. Use case 17: Cards are shared to entire campus

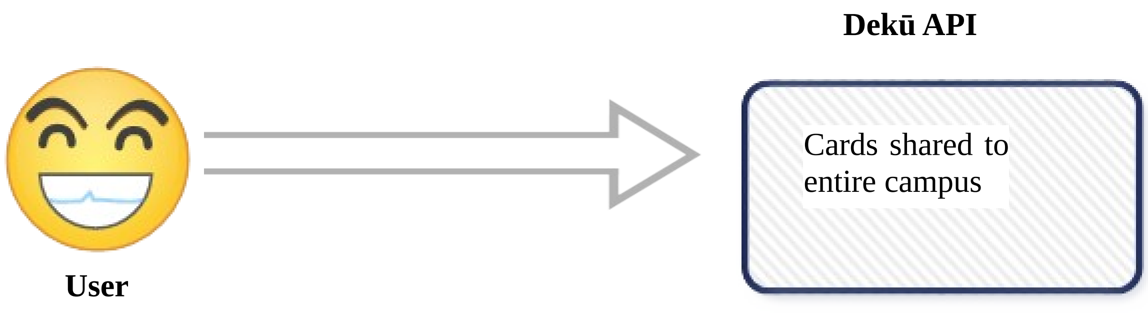


Figure 18. Use case 18: User has avatar

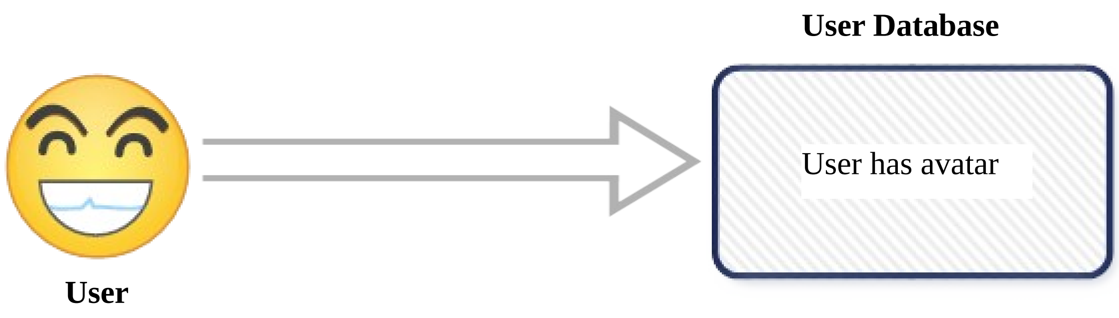


Figure 19. Use case 19: Card size dictated by popularity

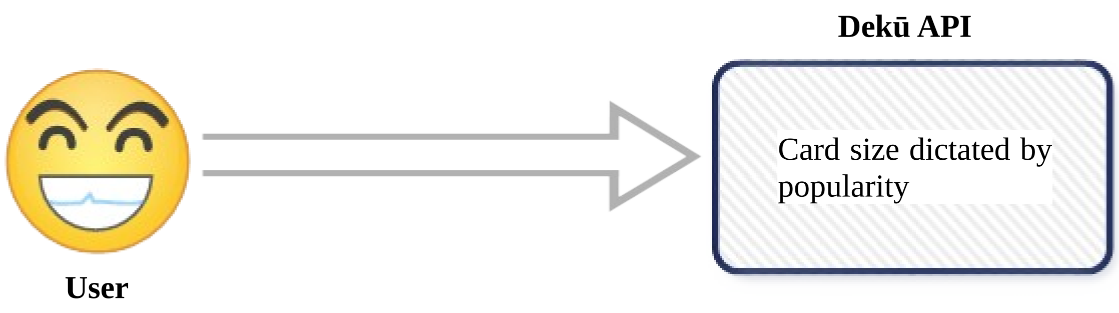


Figure 20. Use case 20: One suit and at least one tag per card

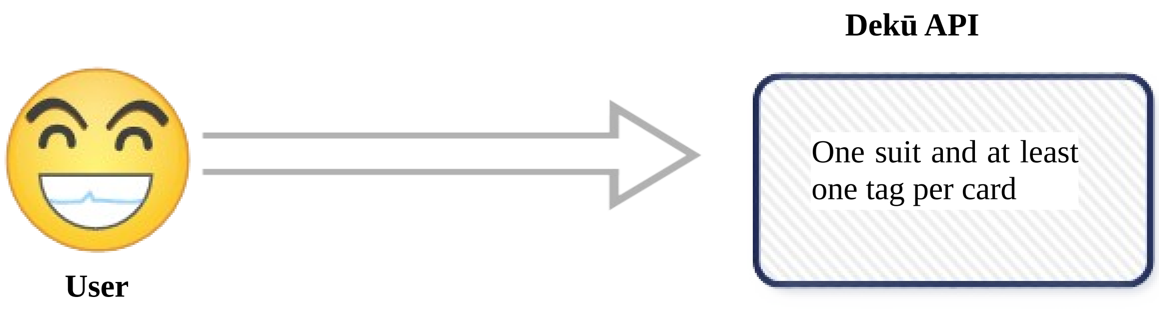


Figure 21. Use case 21: Cards have header which are color coded and divided by tags

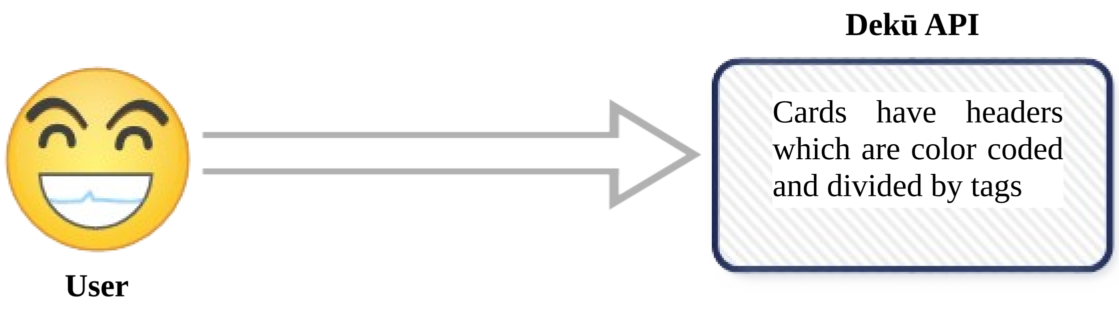

Dekū API
User
Cards have headers which are color coded and divided by tags

Figure 22. Use case 22: User has an administrator account

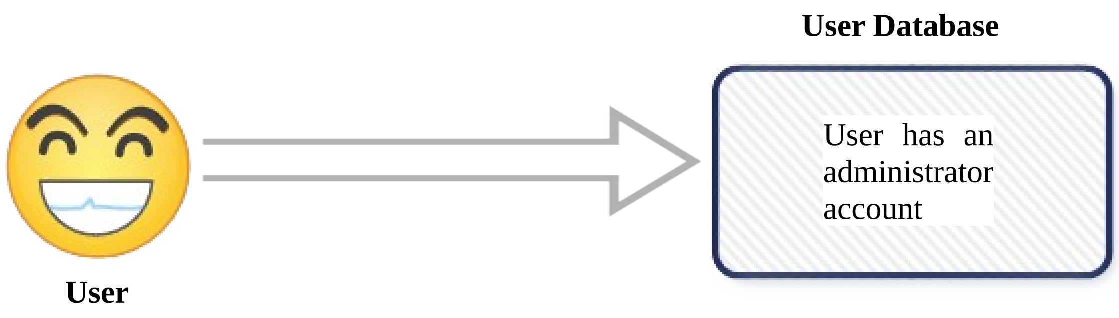


Figure 23. Use case 23: User has moderator rights

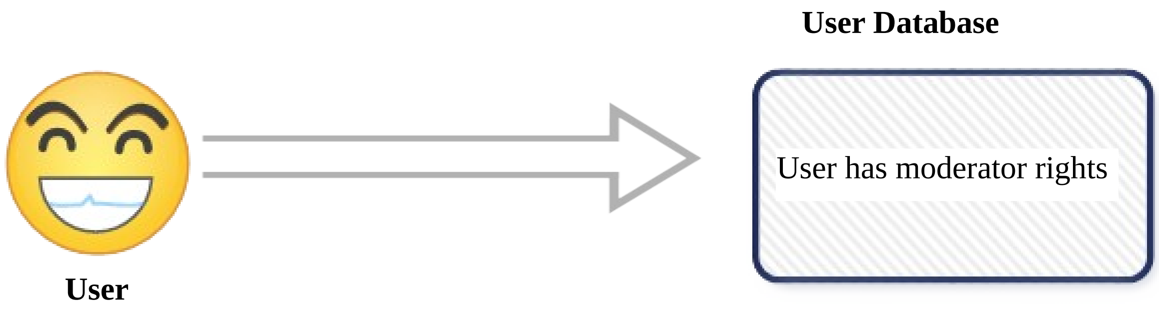
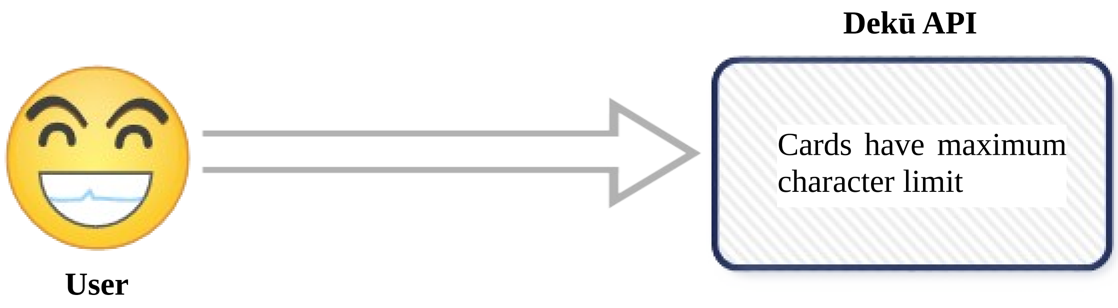


Figure 24. Use case 24: Cards have maximum character limit



2. Functional Requirements

In the cases below, the priority of each is rated based off 1 (highest) to 5 (lowest).

2.1 Use Case 1

Number:	1	
Name:	User logs in	
Summary:	When the user goes to the webpage for the site, they will first be greeted by the login page. They will have the options to create a new account or login with an existing account. The create a new account will be prominent, and there will be a separate area for logging in.	
Priority:	1	
Preconditions:	The user has an internet connection and is on the login page.	
Postconditions:	The user has either generated a new account or has logged into the site.	
Primary Actor:	The user	
Secondary Actor:	UI, database, server	
Trigger:	User enters information and submits it.	
Main Scenario:	**Step**	**Action**
	1:	The email/password combo are sent to the database.
	2:	The user is authenticated.
	3:	The user is sent to to the home page.
Extensions:		
	1a:	This is a new user: They generate a new account.
	2a:	The email/password combo is invalid: The user is requested to resubmit their information
Open Issues:	None: This is fully implemented.	

2.2 Use Case 2

Number:	2
Name:	User creates account
Summary:	At the login screen, if the user does not have an account, they can create a new one. They will be prompted to enter some initial information. This content includes class information that forms your bio; classes, major, year (this is updated every semester as well)
Priority:	1
Preconditions:	Use case 1
Postconditions:	The user now has an account, and is logged in.
Primary Actor:	The user
Secondary Actor:	The interface, the server, the database.
Trigger:	User clicks the “Create an account” button.

Main Scenario:	Step	Action
	1:	The user enters email, password, real name and a bio.
	2:	A new user account with the given info is added to the database.
	3:	The user is logged in.
	4:	The user is prompted to enter in their bio information, but is given the option to keep it private.
	5:	A default avatar is made, but they can replace it.
Extensions:		
	1a:	If the email address is not unique: Tell the user they already have an account and offer to retrieve their password.
Open Issues:	None: This is fully implemented.	

2.3 Use Case 3

Number:	3	
Name:	User loses password	
Summary:	In the event that a user is attempting to login and cannot remember their password, there will be an option to generate a new one by entering your email information. You then can enter a new password.	
Priority:	3	
Preconditions:	Use Case 2, the user must have an account with a valid email address	
Postconditions:	The user now has their password reset to something random which they are prompted to change.	
Primary Actor:	The user	
Secondary Actor:	UI, server, database	
Trigger:	The user clicks the button ‘forgot my password’	
Main Scenario:	**Step**	**Action**
	1:	The user is prompted to enter the email address that is associated with their account.
	2:	That email is submitted to the database for confirmation.
	3:	The email is authenticated
	4:	The system sends an email to that account with a randomly generated password.
	5:	The user submits that password at the login prompt.
	6:	Once they are logged in, they are immediately required to change their password to a permanent one.

	7:	The user submits a new password
	8:	The new password is stored in the user’s account.
Extensions:		
	3a:	The email is not authenticated: The user is prompted to enter their email address again.
	4a:	The user gets an email when they did not request a new password: The user has a button to report it and prevent the the password from being made
Open Issues:	Should there be a cap on how many attempts the user has for entering email? How would the random password be generated (what would it be composed of)? This is not implemented. Would be a useful, post spiral 3 implementation.	

2.4 Use Case 4

Number:	4	
Name:	User deactivates account	
Summary:	A user deactivates their account, removing the account and any authored content from other users.	
Priority:	2	
Preconditions:	The user has an account.	
Postconditions:	The account and any authored content is not viewable by other users.	
Primary Actor:	The user	
Secondary Actor:	Other users, the UI, the server.	
Trigger:	The user selects the “deactivate account” option from their account page.	
Main Scenario:	**Step**	**Action**
	1:	The user selects the “deactivate account” option from their account page.
	2:	A confirmation page is shown to the user, explaining what is happening to their account, and how to reactivate the account.
	3:	The user either cancels or confirms account deactivation.
Extensions:		
	1a:	If the user confirms account deactivation: They are blacklisted on the API, but they remain in the database.
	2a.	If the user cancels account deactivation: Nothing happens and they are returned to their account page.
Open Issues:	Should this require some sort of account verification (re-enter password?)	

	This is not implemented. Would be a post-spiral 3 feature.

2.5 Use Case 5

Number:	5	
Name:	Account deletion	
Summary:	A user deletes their account, permanently removing them from the server and database.	
Priority:	2	
Preconditions:	The user has an account.	
Postconditions:	The user’s account is removed from the system.	
Primary Actor:	The user	
Secondary Actor:	Other users, server, database	
Trigger:	User selects “delete account” option from their account page.	
Main Scenario:	**Step**	**Action**
	1:	The user selects the “delete account” option from the account page.
	2:	A confirmation dialog, explaining the permanency of this action.
	3:	The user either confirms or cancels account deletion.
Extensions:		
	1a:	If the user confirms account deletion: All data on that user is removed from the database.
	2a.	If the user cancels account deletion: Nothing happens and they are returned to their account page.
Open Issues:	None: This is fully implemented.	

2.6 Use Case 6

Number:	6	
Name:	User creates a new card	
Summary:	Every user with access to this system can post cards to the university. These cards are public and contain categories (suits), tags related to their information, and text only content.	
Priority:	1	
Preconditions:	The user has a valid account with the service.	
Postconditions:	A new card is posted to the site.	
Primary Actor:	The user- the author of the card.	
Secondary Actor:	UI, server, database	
Trigger:	User clicks the button ‘Deal New Card’	
Main Scenario:	**Step**	**Action**

	1:	The user is shown a template region to create a new card.
	2:	The user selects one category (a suit) for the card.
	3:	The user selects at least one tag to be associated with the card. These tags are location specific (Sherman Hall, Commons, Library, etc)
	4:	The user enters in text content for the card. It can contain links to other sites but it does not support multimedia
	5:	The card is created and added to the public deck.
Extensions:		
	2a:	The user does not select a suit: The card is not made until a suit is selected
	3a:	The user does not select at least one tag: The card is not made until at least one tag is selected
Open Issues:	None: This is fully implemented.	

2.7 Use Case 7

Number:	7	
Name:	User searches by tags	
Summary:	Since there is a lot of content that is being posted, there must be some way to filter for specific material. Therefore the user is able to search for content based on tags. A card has the following search options: by author, suit, or tag. This will remove any content that does not match the required filter. This content will be restored once the user leaves the search mode (clears the filters).	
Priority:	2	
Preconditions:	The user has a valid account, is logged in, and is on the home page.	
Postconditions:	Only the cards that match with the search parameters will remain.	
Primary Actor:	The user	
Secondary Actor:	UI, database, server	
Trigger:	The user clicks in the ‘Search by:’ text box.	
Main Scenario:	**Step**	**Action**
	1:	The user begins to type in their filter by category, tag, or person
	2:	The system provides options through suggestions matching the input
	3:	The user selects their parameter
	4:	The feed is updated to reflect only this content.
Extensions:		

	3a:	The user types in some tag that does not exist: Well, no content would be shown
Open Issues:	None: This is fully implemented. Also search by author and category	

2.8 Use Case 8

Number:	8	
Name:	User inspects a card.	
Summary:	The user inspects a card, showing the full content, any comments, and more information on the creator. Ability to report as Joker goes here.	
Priority:	1	
Preconditions:	The card exists	
Postconditions:	The card is flipped and the user can see more information.	
Primary Actor:	The user	
Secondary Actor:	UI, server	
Trigger:	The user double clicks on a card on a computer/long presses on a touch screen.	
Main Scenario:	**Step**	**Action**
	1:	The user double clicks/long presses a card.
	2:	The server then generates the back of the card.
	3:	The card flips and extended content is shown to the user.
Extensions:		
	1a:	The user can then comment on the card, mark it, add it to their deck, or report it as a joker.
Open Issues:	How will back navigation work? Will the user click outside of the card view, will there be a discrete button for it? A user can inspect, but the model does not update. Commenting, marking, adding do not function.	

2.9 Use Case 9

Number:	9
Name:	User watches other users
Summary:	This is in essence a following system. By watching someone's deck, their content is shown differently in your feed. This will be denoted by having their avatar and username highlighted so it is clear to the user this is someone that they follow.
Priority:	2
Preconditions:	Both the watcher and watchee have valid, activated accounts for the service.
Postconditions:	This user is now watching the deck of the other user.

Primary Actor:	The user	
Secondary Actor:	The database, UI, server	
Trigger:	Clicking the button to ‘Watch’ someone’s deck.	
Main Scenario:	**Step**	**Action**
	1:	The user visits the profile page of some other person
	2:	The user can click the button to ‘Watch Deck’ of that person
	3:	This user’s list of ‘Decks Watching’ is updated. Every time that person authors a card, it is marked with a highlight
Extensions:		
	None	
Open Issues:	Would there be some master list of the people you watch on your profile page? Is that something anyone can see or just you? Not implemented. This would be a post-spiral 3 feature.	

2.10 Use Case 10

Number:	10	
Name:	User stops watching another user.	
Summary:	A user stops following another user, changing the way they see that user’s cards.	
Priority:	3	
Preconditions:	The user was watching someone else.	
Postconditions:	The user is no longer watching the other user, and their UI reflects the change.	
Primary Actor:	The active user	
Secondary Actor:	UI, database	
Trigger:	The user selects “Stop Watching” option in that user’s profile.	
Main Scenario:	**Step**	**Action**
	1:	The user selects the “Stop Watching” option.
	2:	The user’s UI no longer highlights the other user’s content.
Extensions:		
	None	
Open Issues:	How do we handle someone repeatedly pressing watch/unwatch over and over? Not implemented. This would be a post-spiral 3 feature	

2.11 Use Case 11

Number:	11
Name:	User can hide tag/creator

Summary:	For one reason or another, the user decided they do not wish to see the cards posted under some tag or from some fellow user. Under their preferences section, there will be an option to 'Hide cards'. This will use a similar suggestion system to the search system and will allow the user to decide which tags or authors they want to ignore. The ones that they select will be added to a list that they can see below that option. At any time, they can remove that tag or author from the hide list.	
Priority:	4	
Preconditions:	The user has a valid, activate account and is logged in.	
Postconditions:	The cards matching the tags will not be seen by this user.	
Primary Actor:	The user	
Secondary Actor:	Database, server	
Trigger:	Select option to hide a tag/author	
Main Scenario:	**Step**	**Action**
	1:	The user selects the tag to hide from the text box (suggestions available)
	2:	That tag is taken and added to a list of content to hide
	3:	Anything matching that tag is not shown in the user's feed
Extensions:		
	2a:	The tag selected is not a valid tag or username: The tag is ignored and the user can submit another option
Open Issues:	Users shouldn't be allowed to hide a category, because that's really not a very specific topic. Also, should some tags (like locations) not able to be hidden? What is the extent to hiding content with multiple tags? What if the user want tag X hidden, but a card has tags X,Y,Z? is it hidden simply because it matches, or must it be a unique element? Not implemented. This would be a post-spiral 3 feature.	

2.12 Use Case 12

Number:	12	
Name:	User deletes card	
Summary:	Users have the ability to remove a card from their deck. It will disappear into the dark void of nothingness. Discarding the card.	
Priority:	3	
Preconditions:	The card was originally created; can't delete what didn't exist.	
Postconditions:	That card now no longer exists.	
Primary Actor:	User	
Secondary Actor:	Server, Database	
Trigger:	The user selects the option to discard card (unique button seen only by author of the card)	
Main Scenario:	**Step**	**Action**

	1:	The user opts to discard some card.
	2:	The user is asked to confirm this action.
	3:	The user confirms they really want to discard the card; it's ruining their hand and they don't need it.
	4:	That card is removed from the deck and database
Extensions:		
	3a:	The user decides they don't want to discard it: Stop the action
Open Issues:	What does this do for people that have this card in their deck? Does it vanish from their records as well? This is pretty much implemented. There is a page refresh where the card should just disappear. Still, fully functional.	

2.13 Use Case 13

Number:	13	
Name:	View user profile	
Summary:	Every user has their own bio and profile page. While the user is required to have a bio, they can choose to make the content private. The profile page shows all of the cards that are in the user's deck; what the authored and cards that they added from other users. If it is the current user's page then they will have the option to change account settings and preferences. If someone else is viewing it, then they will have options to 'Watch/Unwatch Deck' and hide the person, or report them as a Joker. They also have the ability to message the person, or 'Pass a card under the table' for a private chat.	
Priority:	2	
Preconditions:	Both users have valid accounts to the service	
Postconditions:	A user account exists and is publicly visible containing that author's cards	
Primary Actor:	User	
Secondary Actor:	Other users, server, UI, database	
Trigger:	User clicks on someone's avatar or name	
Main Scenario:	**Step**	**Action**
	1:	This user is directed to the page for the other person.
	2:	This page will have a tab for user bio and user deck.
	3:	Content will be generated from the user tags.
	4:	If it is that user's page, they will have an added tab for preferences and settings where they can update account information and tag preferences
	5:	If it is someone else's page, they can

		‘Watch/Unwatch’, hide, report, view the content, or message them. It’s a lot of options!
Extensions:		
	None for the moment	
Open Issues:	None: This is fully implemented. The only feature that fails is step 5, none of these except message work.	

2.14 Use Case 14

Number:	14	
Name:	User reputation	
Summary:	This will consist of an aggregate and separated score across several different categories like number of people watching, how many of your cards are in the decks of others, how popular are your average cards. This can be a breakdown that is visible from the bio page maybe.	
Priority:	3	
Preconditions:	The user has a valid account. No content or watchers just means score of 0. If they do have content it is processed through some algorithm.	
Postconditions:	The user’s score will be generated and posted to their page.	
Primary Actor:	The API since this is not a feature that the user is in control of.	
Secondary Actor:	The server, database	
Trigger:	The user has an account and has some activity like a popular post.	
Main Scenario:	**Step**	**Action**
	1:	Some public event occurs in relation to the popularity of a user’s post or profile (they are watched, a card was marked or added to a deck).
	2:	That event adds points to the user’s reputation in that category which is weighted for the aggregate score.
	3:	The new scores are updated
Extensions:		
	None at the moment	
Open Issues:	If the user has been marked as a joker repeatedly, is hidden from people, does that negatively affect his score? This is not implemented. It would be a post spiral 3 feature.	

2.15 Use Case 15

Number:	15
Name:	User receives notification.
Summary:	Users will receive notifications upon certain events. These have two parts: an initial toast when it occurs, and a marker in the header.

Priority:	4	
Preconditions:	A trigger event occurs (there are several)	
Postconditions:	The user is alerted to an event.	
Primary Actor:	The UI	
Secondary Actor:	The user, server, database	
Trigger:	User is being watched, user receives message, etc.	
Main Scenario:	**Step**	**Action**
	1:	An event occurs triggering an alert to the user.
	2:	A toast appears on screen, alerting the user immediately as to what happened.
	3:	A "tick" is added to the notification counter in the header.
Extensions:		
	1a:	If the user isn't logged in: The notification toast is not required, but the tick will still be added to their counter.
Open Issues:	A full list of events need to be defined. Not implemented. Would be a post-spiral 3 feature.	

Use Case 16

Number:	16	
Name:	User logs out.	
Summary:	The user logs out of the system.	
Priority:	1	
Preconditions:	The user is logged in.	
Postconditions:	The user is logged out.	
Primary Actor:	The user.	
Secondary Actor:	The server.	
Trigger:	User selects the "Log out" UI element.	
Main Scenario:	**Step**	**Action**
	1:	The user selects the log out UI element.
	2:	The user is logged out of their account.
Extensions:		
	None	
Open Issues:	None: This is fully implemented.	

2.17 Use Case 17

Number:	17
Name:	Cards are shared to the entire campus.

Summary:	When a card is generated, it is shown in the main feed to everyone on the campus.	
Priority:	1	
Preconditions:	A particular card exists.	
Postconditions:	The card is added to the feed of all users.	
Primary Actor:	The UI.	
Secondary Actor:	Users, server, database	
Trigger:	A card is grabbed from the server.	
Main Scenario:	**Step**	**Action**
	1:	A card is loaded from the server.
	2:	The card is shown in the feed of all users.
Extensions:		
	1a:	Users can search by tag: Cards may be hidden temporarily.
	2a:	Users can choose to ignore certain tags: Some cards are hidden at all times.
Open Issues:	None: This is fully implemented.	

2.18 Use Case 18

Number:	18	
Name:	User has avatar	
Summary:	A default avatar is generated for the user by the same process that makes the Github avatar. The user is able to upload their own photo for their avatar.	
Priority:	3	
Preconditions:	The user has an active, valid account.	
Postconditions:	The user's avatar is set	
Primary Actor:	User	
Secondary Actor:	UI, server, database	
Trigger:	Either creating new account or updating photo	
Main Scenario:	**Step**	**Action**
	1:	User selects 'Set avatar'
	2:	By default an avatar is generated Github style.
	3:	The user is given the choice to upload their own photo.
	4:	Whatever they select is their new avatar.
Extensions:		
	3a:	The file format of the uploaded photo is not compatible: Refuse image, reset to default and ask for another
Open Issues:	None: This is fully implemented. The only issue is user can't upload their own avatar. This is post spiral 3 feature.	

2.19 Use Case 19

Number:	19	
Name:	Card size is dictated by popularity.	
Summary:	Where viewport size permits, more popular cards are shown to be larger in order to draw the attention of the user.	
Priority:	2	
Preconditions:	The card is loaded from the server.	
Postconditions:	The card's rendering dimensions are calculated based on the card's relative popularity.	
Primary Actor:	UI	
Secondary Actor:	Server	
Trigger:	A card is pulled from the database	
Main Scenario:	**Step**	**Action**
	1:	A card is pulled from the database.
	2:	The server evaluates the card's popularity, and passes that to the UI.
	3:	The card is then rendered at the appropriate size based on its popularity.
Extensions:		
	None	
Open Issues:	How will this be handled on small screens (phones and tablets), where there isn't a lot of room for more than one or two cards at a time? This is not implemented. It would be a post-spiral 3 feature.	

2.20 Use Case 20

Number:	20	
Name:	One suit and at least one tag per card	
Summary:	This service is based by filtering tags. Every card must have some feature to define it. A suit is a category (administrative, club, big event, other) and tags are locations and other specific contexts. One suit for each card and at least one tag (location)	
Priority:	2	
Preconditions:	A new card is being made.	
Postconditions:	The card will now have these features.	
Primary Actor:	User	
Secondary Actor:	UI, database	
Trigger:	The user makes a new card	
Main Scenario:	**Step**	**Action**
	1:	The user is shown the four suits to select from.
	2:	The database associates that suit with the card.

	3:	The user types tags into a text box which gives suggestions as there is input.
	4:	The user selects all desired tags
	5:	The user enters content for the card and submits it
Extensions:		
	None	
Open Issues:	None: This is fully implemented. The only issue is there are no graphics, only text.	

2.21 Use Case 21

Number:	21	
Name:	Cards have a header which is color coded and divided by tags.	
Summary:	When cards are generated in the UI, there will be a header at the top which will display colors for each tag assigned to the card.	
Priority:	2	
Preconditions:	The card exists and has at least one tag (requirement for creating cards).	
Postconditions:	A header is shown with color coding for each tag.	
Primary Actor:	UI	
Secondary Actor:	Server	
Trigger:	The UI render call is made.	
Main Scenario:	**Step**	**Action**
	1:	The server sends card data to the UI.
	2:	The UI generates the card and subdivides the header based on the number of assigned tags.
	3:	The section for each tag is then colored appropriately.
Extensions:		
	None	
Open Issues:	None: This is fully implemented.	

2.22 Use Case 22

Number:	22
Name:	Administrator accounts
Summary:	Administrators must be defined in our system and given particular permissions.
Priority:	1
Preconditions:	An administrator account exists on the server.
Postconditions:	The administrative user is given certain superuser permissions.
Primary Actor:	The administrator
Secondary Actor:	The UI, server, database

Trigger:	An administrator account is created.	
Main Scenario:	**Step**	**Action**
	1:	An administrator account is created (a requirement at product launch)
	2:	The user account on the server is then given certain permissions beyond that of a normal user (address jokers, throttle users for bad behavior, ban users, assign moderators).
	3:	The UI will then reflect the new options that administrators have.
Extensions:		
	1a:	Admins can immediately remove jokers.
	2a:	Admins can throttle users as they see fit. What this entails still needs to be defined.
	3a:	Admins can ban users immediately. This will be functionally similar to delete a user account.
	4a:	Administrators can give regular users moderator rights
Open Issues:	Admininstrators exist, however all they can do is make users a moderator.	

2.23 Use Case 23

Number:	23	
Name:	Moderator rights	
Summary:	Moderators basically have the same power as the administrator; they just need a vote (three of a kind) to ban a user or delete Joker content. The can individually throttle a user in a first stage offense, but cannot assign new moderators.	
Priority:	2	
Preconditions:	The administrator has made this user a moderator	
Postconditions:	This user now has the aforementioned powers.	
Primary Actor:	Moderator	
Secondary Actor:	Server, UI, database	
Trigger:	Moderators are created	
Main Scenario:	**Step**	**Action**
	1:	This individual has some extra features when on a user’s profile. They can ban/throttle that user.
	2:	Such an action sends a message to other moderators (except for throttle) to take a vote on the action.
	3:	Three moderators must agree within 48 hours on the course of action.
	4:	Moderators may throttle users without a

		vote.
Extensions:		
	None	
Open Issues:	What system will the moderators have for communicating amongst themselves and voting on actions? Created, but no powers.	

2.24 Use Case 24

Number:	24	
Name:	Cards have a maximum character limit for their content.	
Summary:	When cards are created, their content can include text and links. The maximum number of characters must not exceed a certain number.	
Priority:	3	
Preconditions:	A card is being created.	
Postconditions:	The number of characters in the card’s content is restricted.	
Primary Actor:	UI	
Secondary Actor:	Server	
Trigger:	A card is being created.	
Main Scenario:	**Step**	**Action**
	1:	The user begins creating a card.
	2:	When entering in the content area, the UI must “listen” to the character count, and text field must stop taking characters after a preset amount.
	3:	Once input is accepted, the card is saved to the server.
Extensions:		
	1a:	If a link is entered and it goes beyond the prescribed length: Recommend a link shortener, like bit.ly
Open Issues:	Text limit set at 140 characters. Bit.ly link shortener not implemented.	

2A. Functional Requirements (Post Spiral 3)

These are requirements that would be implemented given that we had more time to work on the final product.

- **2.3 –** Use Case 3 – Reseting password. This would be done with Flask's emailing module. It would bring the user to a password reset page.
- **2.4 –** Use Case 4 – Deactivate account. Currently users can delete account, but deactivate is not

an option. This expansion is trivial and non essential, but in a marketable product is nice for user flexibility.

- **2.8 –** Use Case 8 – Inspect view works in that it will show the user any available information. It does not update the cards properties however.
- **2.9 –** Use Case 9 – Watching, or following, other users is a feature that may not be implemented for spiral 3 or the demo. The customer has stated that this feature would be nice to have, but for the scope of the project and given the time constraints, this feature is not mandatory.
- **2.11 –** Use Case 11 – Hiding users. This was not a required feature, but would be nice for user flexibility.
- **2.13** – Use Case 13 – The messages feature may not be implemented for spiral 3 or the demo. The customer has stated that this feature would be nice to have, but for the scope of the project and given the time constraints, this feature is not mandatory.
- **2.14 –** Use Case 14 – Reputation doesn't exist. This was not a hard requirement. It might have been nice to use for statistics purposes. In post-spiral 3 if other user components are done, it could be easily implemented.
- **2.15 –** Use Case 15 – Notifications. This does not exist, but it was not an important feature. It didn't have a very good place in the application, but it could be implemented down the road.
- **2.19 –** Use Case 19 – Popularity is not in place. This was not a major requirement. It would have been a nice feature for the purpose of telling important cards apart. Post-spiral 3 feature.

3. Non-Functional Requirements

This section covers the other requirements that exist more in the architecture of the application (ex. How databases will be handled, documentation of code, system frameworks, etc).

#	**Item**	**Priority 1 (highest) to 5 (lowest)**
1	Written in Python	2
2	Code well documented	2
3	Use Flask	3
4	Use SQL for database	3
5	Easy to maintain	3
6	Use JavaScript	3
7	Use Twitter Bootstrap	3
8	Use Vagrant	3
9	Use Apache	3
10	Use Backbone	3
11	Easy to use	2
12	Easy to navigate	2

4. User Interface

See the “User Interface Design Document”

5. Deliverables

See the “User Interface Design Document”

Deliverables for Spiral 2 include:

- Systems Requirement Specification
- System Design Document
- User Interface Design Document
- Coding Inspection Report
- Testing Report

6. Open Issues

Issues	**Scheduled for**
Data security	Spiral 3
Opening to other colleges	Spiral 3?
Verifying non-functional requirements	Ongoing

7. Appendix A – Agreement Between Customer and Contractor

The customer for the dekū social content and sharing site has agreed to this site and all its functionality, including creating accounts within a university context, sharing public content based on location, and being able to mark costs and pass messages through them. This document includes use cases for all of the functional requirements, as well as a collection of the non-functional requirements.

If changes to the requirements develop in the future, this document will be amended and all parties will sign off on the document to make sure that the changes are in accordance with the wishes of the customer.

Client

Name ____ Date ___________________
Print
Name ____ Date ___________________
Signature

Team

Name ____ Date ___________________
Print
Name ____ Date ___________________
Signature

Name ____ Date ___________________
Print
Name ____ Date ___________________
Signature

Name ____ Date ___________________
Print
Name ____ Date ___________________
Signature

Name ____ Date ___________________
Print
Name ____ Date ___________________
Signature

Name ____ Date ___________________
Print
Name ____ Date ___________________
Signature

8. Appendix B – Team Review Sign-off

This document affirms that all of the members of this team have contributed to and reviewed the material within this document. Any minor disagreements between members are listed below.

Team

Name ____ Date ________________
Print
Name ____ Date ________________
Signature
Comments
____
____

Name ____ Date ________________
Print
Name ____ Date ________________
Signature
Comments
____
____

Name ____ Date ________________
Print
Name ____ Date ________________
Signature
Comments
____
____

Name ____ Date ________________
Print
Name ____ Date ________________
Signature
Comments
____
____

Name ____ Date ________________
Print
Name ____ Date ________________
Signature
Comments
____
____

9. Appendix C – Document Contributions

Identify how each member contributed to the creation of this document. Include what sections each member worked on and an estimate of the percentage of work they contributed. Remember that each team member <u>must</u> contribute to the writing (includes diagrams) for each document produced.