Aaron Piando
CS391 Homework 1

Story #1: Creating an Account

Description	As a BU student, I want to create an account on Spark!Bytes, So that I can see if there is any free food available.
Additional Detail	I am a current BU student with a BU ID and email. I have never used Spark!Bytes before but would like to see what the app has to offer.
Acceptance Criteria	Scenario: Creating an account successfully - Given I am a current BU student, - When I click on “Create an Account” - Then I can provide my BU ID and other personal information - To create a Spark!Bytes account - And access the features of the application Scenario: Unable to create an account - Given I am a current BU student, - When I click on “Create An Account” - And I enter my credentials incorrectly - Then the webpage should reload with error messages - So I can correct the mistakes and reattempt to create an account
Priority	High priority, since this is a basic feature which allows users to access the application
Checklist	- Independent as all eligible users should be able to create an account - Negotiable as these scenarios can be achieved using a variety of approaches - Valuable as it provides a key functionality to the application - Estimable as user creation and authentication is fairly straightforward - Sized appropriately as setting up user creation and authentication requires following a finite set of steps - Testable as it is possible to create automated tests which test if a user can create an account and if a user is denied access without valid credentials
Associated Tasks	1) Create a landing page interface with a

	"Login" button 2) Create a login page with a "Create An Account" button for new users 3) Allow for error messages to populate when appropriate 4) Redirect user to a confirmation page upon successful account creation
Task Assignments	This will be completed later when teams are assigned and tasks assigned

Story #2: Looking for Food

Description	As a BU student with a Spark!Bytes account, I want to look for food available near me, So that I can get a quick snack while studying.
Additional Detail	I am studying in the GSU and would prefer to stay in central campus to get the food.
Acceptance Criteria	Scenario: Logging into Spark!Byte successfully - Given I already have a Spark!Byte account, - When I click on "Login" - And I enter my username and password, - Then I should be redirected to a home page - From which I can access postings of available food - And filter by location Scenario: Failure to log in - Given I already have a Spark!Byte account - When I click on "Login" - And I incorrectly enter my username and password, - Then the webpage should reload with error messages
Priority	High priority, since this is a basic feature which allows the user to find available food
Checklist	- Independent as all eligible users should be able to access available food postings - Negotiable as there are various ways

	to implement a newsfeed of available food postings - Valuable as this feature achieves a key component of the application’s main objective - Estimable as allowing a user to access a newsfeed is fairly straightforward - Sized appropriately implementing a newsfeed of postings about available food requires a finite list of tasks - Testable as it is possible to write automated tests which confirm that the user is able to access the food postings after successfully logging in to the application
Associated Tasks	1) Create a login page which populates error messages when appropriate 2) Create home page and menu with a link to the newsfeed 3) Create a newsfeed webpage which displays all available food postings 4) Allow food postings to be tagged by region of campus (east, central, etc.) 5) Allow user to filter through postings based on various features, including location
Task Assignments	This will be completed later when teams are assigned and tasks assigned

Story #3: Denying Access

Description	As a Northeastern University student, I want to create a Spark!Byte account, So that I can steal food from BU students.
Additional Detail	I am not a BU student, so I do not have a valid BU ID or email.
Acceptance Criteria	Scenario: Failure to create an account - Given I do not have a valid BU ID or email to provide, - When I attempt to submit a login form, - Then I should be given a message explaining that access was denied
Priority	Medium priority, since it will be relatively difficult to implement a BU ID validation

	security feature given the limited resources for this project.
Checklist	- Independent as implementing this security feature would not hinder other eligible users from BU - Negotiable as there are various approaches to implement this feature - Valuable as it would add to the overall security and reliability of this app as a tool for the BU community - Estimable as this is a security feature that is already used in existing BU-only software - Sized appropriately as implementing this feature would be relatively straightforward with access to the necessary resources - Testable as it is possible to write automated tests to see whether or not the access denied message is displayed when appropriate
Associated Tasks	1) Obtain a frequently updated list of currently active BU IDs 2) Implement a security checkpoint during the creation of an account in which the entered BU ID is from the list of active IDs and is not already being used by an existing account 3) Create a separate webpage or add a popup box containing an access denial message which populates when appropriate
Task Assignments	This will be completed later when teams are assigned and tasks assigned

Story #4: Posting Food

Description	As a president of a student club, I want to give away leftover food from a club meeting, So that I can prevent food waste.
Additional Detail	I am a BU student. My club was expecting 20 members to show up to our monthly meeting, and we purchased snacks accordingly. However, only 10 students showed up, so we

	still have enough snacks to feed 10 students.
Acceptance Criteria	Scenario: Posting available food successfully - Given that I have food available for someone else to take, - And that I am already logged in to the Spark!Byte app, - Then I can press a button to access a form to post available food, - And I can provide information about the food, including a photo, a title, a description, a location tag, and an availability flag, - And I can post the food on the app - And I see a confirmation page to confirm that the food was posted Scenario: Unable to post available food - Given that I have not provided necessary details about the food I want to give away, - Or that I don't actually have food to give away and I just submitted a random, unrelated photo - Then I am given an error message explaining why my food was not posted
Priority	High priority, as this feature of the app is directly tied to the main functionality of the program
Checklist	- Independent as this feature would not hinder the experience of any other users and instead provides a key functionality for all users to enjoy - Negotiable as there are various features which can be added for improved UX but are not necessary for the core functionality of the app - Valuable as this is the main feature that allows students to actually exchange extra food - Estimable as a posting feature does not require an exhaustive list of requirements to successfully implement - Sized appropriately as this is feature can be easily implemented within the time constraints of this project - Testable as it is possible to write

	automated tests which check if data entered in the form is accepted or rejected whenever appropriate
Associated Tasks	1) Create a form for posting available food, implementing necessary input requirements and guardrails where necessary 2) Implement an AI-powered API capable of detecting whether or not a photo matches the title and description given by the user attempting to post available food 3) Create a webpage which provides confirmation that the user's post is now live and viewable by other users 4) Create a webpage or error message popup which alerts the user that their available food was not posted, along with the reason why
Task Assignments	This will be completed later when teams are assigned and tasks assigned

Story #5: Claiming Food

Description	As a Spark!Byte user, I want to claim food that I see on the Spark!Byte app, So that I can get a quick snack during my study break.
Additional Detail	I am studying on the second floor of CDS, and I see a posting for available food with the location tag set to Central Campus. I'd like to message the poster so I can claim the food.
Acceptance Criteria	Scenario: Successfully reaching out to the poster - Given that the food I want is still available, - I can send a message to the poster to claim the food, - And we can exchange information to designate a location for pickup Scenario: Failure to claim the food - Given that the food is no longer available, - I send a message to the poster to

	claim the food, - And the poster replies to let me know the food was taken and that the post I saw was not updated or old Scenario: Poster not responsive - Given that I reached out to the poster, - And the post still indicates that the food is currently available, - But the poster has not replied for over 2 hours, - Then the chat window should close
Priority	High priority, as this feature works in conjunction with previously mentioned features of the app to allow students to both give away and claim food
Checklist	- Independent as this feature does not hinder any other user stories - Negotiable as there are various ways to successfully implement this feature - Valuable as this feature works directly with other key components of the application to achieve the app's main functionality - Estimable as a chat feature between users has been done before and thus can be fairly easily implemented in this app - Sized appropriately as implementing this feature would not require an exhaustive list of requirements to be satisfied - Testable as it is possible to create automated tests to check if the chat behaves as expected based on the users' messaging actions and behaviors
Associated Tasks	1) Create the backend code to provide functionality for a chat feature that allows users to exchange texts with each other, implementing necessary restrictions on the length of messages, inappropriate language, time window for active chats, etc. 2) Create the frontend interface for the user to actually use the chat feature 3) Ensure that all existing and active chat windows are associated to an

	existing and active food posting
Task Assignments	This will be completed later when teams are assigned and tasks assigned

Story #6: Food Was Claimed

Description	As a Spark!Byte user to posted available food, I want to mark the food as claimed, So that people don't mistakenly message me to claim food that's already claimed.
Additional Detail	I already messaged another student and arranged to meet in front of Warren Towers so I can give him the extra food from my club meeting.
Acceptance Criteria	Scenario: Successfully flagging food as being claimed - Given that I have already arranged to give away the food to someone, - Then I should be able to flag the food posting so that it appears unavailable
Priority	High priority, as this feature is key for preventing posts for already claimed food from getting mixed in with posts which for currently available food
Checklist	- Independent as this feature does not hinder any other user stories and instead provides a necessary functionality to all eligible users' experience - Negotiable as there are many ways to indicate that the food in a post is no longer available, ranging from removing the post from public view entirely or simply replacing a green "available" flag on the post to a red "unavailable" flag on the post - Valuable as this is directly tied to a key feature of the Spark!Byte app and prevents miscommunication or misunderstandings between users - Estimable as the implementation of this feature would be relatively straightforward

	- Sized appropriately as there isn't an impossibly long list of requirements to satisfy to successfully add this feature - Testable as it is possible to create automated tests which check that the user can flag a food posting as unavailable after another user has claimed the food over chat
Associated Tasks	1) Create an additional form to allow the user to indicate that a currently active food post is now unavailable 2) Add another button to the chat interface to allow the poster to easily submit the form to mark a food posting as inactive
Task Assignments	This will be completed later when teams are assigned and tasks assigned

Story #7: Dealing With Disrespectful Users Over Chat

Description	As a Spark!Byte user, I want to report disrespectful users in the chat, So that I can feel safe while using the app and avoid harassment.
Additional Detail	I was trying to claim food through the chat feature, but the poster started sending rude and inappropriate messages.
Acceptance Criteria	Scenario: Successfully reporting a disrespectful user - GIven that I am currently chatting with a poster, - And the other user is sending me offensive messages, - Then I should be able to press a "Report User" button, - And submit a form with a brief reason for the report, - So that a text processing API acting as a moderator can review the flagged interaction, - And determine if either user is at fault and needs to be penalized Scenario: Blocking further messages - Given that I have reported another

	user - I should not longer receive any communications in the chat from them - And I should no longer be able to see their food postings to prevent any further interactions
Priority	Medium priority, since this feature isn’t directly connected to the core functionality of the app but still provides necessary security to enhance the UX
Checklist	- Independent since the ability to report another user wouldn’t impact other features of the app - Negotiable since reporting users could also be accomplished using other methods, like sending reports to a human chat moderator, using a text processor API to auto-flag conversations, etc. - Valuable as this feature helps promote building a safe and respectful community on the app - Estimable as reporting and blocking can be built with existing chat moderation practices - Sized appropriately since this requires only a few additional UI elements and backend programming - Testable since automated tests can be written to check whether or not a user is successfully blocked or flagged after initiating an inappropriate conversation
Associated Tasks	1) Add a “Report User” button to the chat interface 2) Implement backend logic to keep track of flagged users and the associated reasons for the flags 3) Allow reported users to be temporarily or permanently restricted by a moderator, whether an API or human 4) Revise the post feed code to prevent a user from seeing posts published by a blocked or restricted user
Task Assignments	This will be completed later when teams are assigned and tasks assigned

Story #8: Dealing With Users Posting Bad Food

Description	As a Spark!Byte user, I want to report food postings that are misleading, spoiled, or unsafe, So that other students don’t waste time or risk their health.
Additional Detail	Sometimes posters may upload photos of food that is clearly spoiled, unrelated to the description, or intentionally misleading. Other times, the poster may publish available food with no intention of actually giving the food away.
Acceptance Criteria	Scenario: Successfully reporting a bad food post - Given that I see a suspicious or unsafe food posting, - Then I can press a “Report Post” button, - And provide a brief reason for why the post should be flagged in a form, - So that moderators can review the post Scenario: Successfully taking down a bad food post - Given that multiple reports have been submitted for a food post, - Then the food post should be temporarily hidden from public view, - Until moderators determine whether it should be permanently removed
Priority	Medium priority, as this once again is not directly connected to the main functionality of the app and instead is intended to improve UX
Checklist	- Independent since this does not hinder the UX of other valid users - Negotiable as there are various ways to implement the process of flagging and taking down posts - Valuable as it protects community trust and food safety - Estimable since it is straightforward to add a reporting mechanism directly tied to posts

	- Sized appropriately as only minor UI and backend additions are required to successfully implement this feature - Testable as automated tests can be written to check if reports submitted by users are logged and reviewed by the moderator
Associated Tasks	1) Add a “Report Post” button to each food listing 2) Implement backend logic for logging and tracking reported posts 3) Automatically hide posts after a minimum number of reports 4) Provide moderators (if human) with an interface to review and either restore or permanently remove posts; otherwise, implement a text processing API to act as a moderator
Task Assignments	This will be completed later when teams are assigned and tasks assigned