uBucket

A University Based Bucket List

University of Illinois at Urbana-Champaign
CS 428/429: Software Engineering II
Professor Darko Marinov
Spring 2014

Authors:
Metin Askaroglu
Akshay Bajaj
Estayvaine Bragg
Katie Chrzanowski
Ricky Lung
Khulan Myagmardorj

Table Of Contents

1. Description and Process …… 3
1.1 Description of Project ……3
1.2 Motivation ……...3
1.3 Process ……3

2. Requirements and Specifications……...4
2.1 User Stories……..4
2.2 Use Cases……4

3. Architecture and Design ……... 6
3.1 Architecture of System …….. 6
3.2 UML Diagrams ……6
3.3 Design of System ……...… 11

4. Future Plans ……..……. 15
4.1 Future Plans …… 15
4.2 Personal Reflections ……... 15

Section 1: Description and Process

1.1 Description of project

Every year, thousands of university students graduate without truly exploring their college campus, and fail to visit campus landmarks and complete various college traditions. uBucket is a social todo list mobile application meant to manage university students' bucket lists. It will act as a social means to submit, comment, and upvote bucket list stories within their university's network as well as provide a space where each individual can privately keep track of the items they would like to complete. This application will be available to everyone on all smartphone platforms for free. uBucket will finally allow students to compose a list of things they want to do at their university before leaving, and serve as a means to share, discuss, and accomplish these tasks before they graduate and leave campus.

1.2 Motivation

Our motivation for this project roots from both of us being seniors and feeling like we have limited time here. We want to appreciate all aspects of this university, and are currently in the process of writing our personal bucket lists. We needed a way to keep track of the popular bucket list suggestions from students in this university and a method to share our ideas with others.

1.3 Process

Our team this year has decided to follow the Extreme Programming software development process. This process was most beneficial to us as its key methodologies promoted more productivity and a quicker turnaround by having weekly iteration checkpoints with our moderator and frequent releases. Instead of having traditional code review sessions, XP's pair programming aspect advocates greater efficiency because it allows us to have continuous code reviews while programming. The test driven aspect of XP also contributed to more effective programming as it assisted in making sure edge cases are covered within our source code from the beginning as well as allowed quick code changes at a low cost. This process allowed us to be more effective in producing a deliverable as well as keeping us constantly moving forward towards our next goal.

Section 2: Requirements and Specifications

2.1 User Stories

- As a user, I would like to be able to use the app simply by making an account.
- As a user, I would like to have my bucket list progress saved as a unique user.
- As a user, I would like to be able to submit bucket stories to accomplish while at the University.
- As a user, I would like to be able to add private bucket list stories to my bucket list without anyone else on the community seeing it.
- As a user, I would like to be able to mark off a story off my personal bucket list once I have finished doing it.
- As a user, I would like to be able to see a profile page for myself.
- As a user, I would like a way to see the newest bucket list stories for my university.
- As a user, I would like to allow a new private bucket list item to be also added to the public bucket list.
- As a user, I would like to view all information on the app asynchronously.
- As a user, I would like to see the most popular bucket stories and have the ability to add it to my own bucket list.
- As a user, I would like to see the most popular bucket stories and have the ability to comment on it.
- As a user, I would like to see the most popular bucket stories and have the ability to upvote it.
- As a user, I would like to visibly see how much progress I have made with completing items on my bucket list.

2.2 Use Cases

Public Bucket List

Immediately upon logging into uBucket, a public bucket list of user-made items from the user's university will be displayed. Bucket list items that were created by the users and acknowledged to be submitted to the public bucket list will appear there. The items are pulled from the Parse database and displayed with all associated information such as the number of comments and upvotes. Through this view, users are able to add new stories to their own personal bucket list. Once a new story is added to the public list, the app asynchronously updates it and the user will be able to view it immediately.

Private Bucket List

Once logged in, the user can access their own private bucket list. This view may contain stories that are shared between the public list, or will be stories that are unique to the user that no other users are able to access. They are able to compose their own item and choose whether or not to release it publicly.

At the top of their screen will be a progress bar that updates asynchronously when they add, remove, or complete a bucket list item. All data of what items are in / completed is stored in Parse.

Commenting on Stories

Each public bucket list item has its own page for comments that correlate with that item. The comments are all stored within a table in Parse. Each comment also has a unique user associated with it. All users will be able to view the comments from any given bucket list item on the public bucket list, and the comments will be updated asynchronously across systems.

Upvoting Stories

Much like comments, public bucket list items have an upvote count associated with them. When a public list item is created, it is stored in a table in Parse and its upvote count is initialized to 0. Users who view the item are given the option to "upvote" the bucket list item, which will increase its count by one. A higher count will signify to users that this is a popular task, which may incline more users to add it to their own private bucket list.

Personal Profile

When using the app for the first time, a user is able to register an account. Email addresses and usernames are unique for each user. Each user then has their own personal profile which displays the number of bucket list items they have saved in their private bucket list, as well as how many they have completed. Their profile also has a link to a feedback page, where they are able to submit a bug or question to the developers which will be stored in Parse. Future updates will include the feedback being sent to a dedicated email address rather than stored in the database.

Section 3: Architecture and Design

3.1 Architecture of System

uBucket was written entirely with web development languages, such as AngularJS, HTML, CSS, and XML. By writing all of this on top of *PhoneGap*, a mobile development framework, we're able to easily transform web development code into native iOS, Android, Windows Phone, and Blackberry applications. The biggest advantage of using *PhoneGap* is that we can guarantee our users a uniform and seamless experience across all platforms without having to allocate resources towards each individual platform. On top of *PhoneGap*, we're using *Ionic Framework*, which is front-end web framework for developing mobile apps. Our team managed to cut out our entire back-end because we could hide implementation details through the app's compilation. This being the case, we used *Parse* as our primary database, and it connected to our app by acting as a service. In a nutshell, our team used *PhoneGap* to create cross-platform apps and *Ionic* to power the front-end experience for the user while storing all data in *Parse*.

This application was written using a Model-View-Controller (MVC) architectural model, as per the fundamental design dictated by AngularJS. Our models were services, which allowed us to mass-produce objects when necessary, controllers, which manipulated the services to provide the views with information, and the views, which were responsible for rendering front-end HTML pages to the user. Using these three different components, we were able to provide a very stable, efficient, and controlled experience to the user.

3.2 UML Diagrams

The following few pages contains the UML diagrams for various parts of our system as listed:

- Adding a bucket list item to either or both the private and public bucket list pages
- Viewing the public and private bucket list pages and its respective functionalities
- Existing user logging in and new user registration functionalities
- User profile and reporting feedback pages

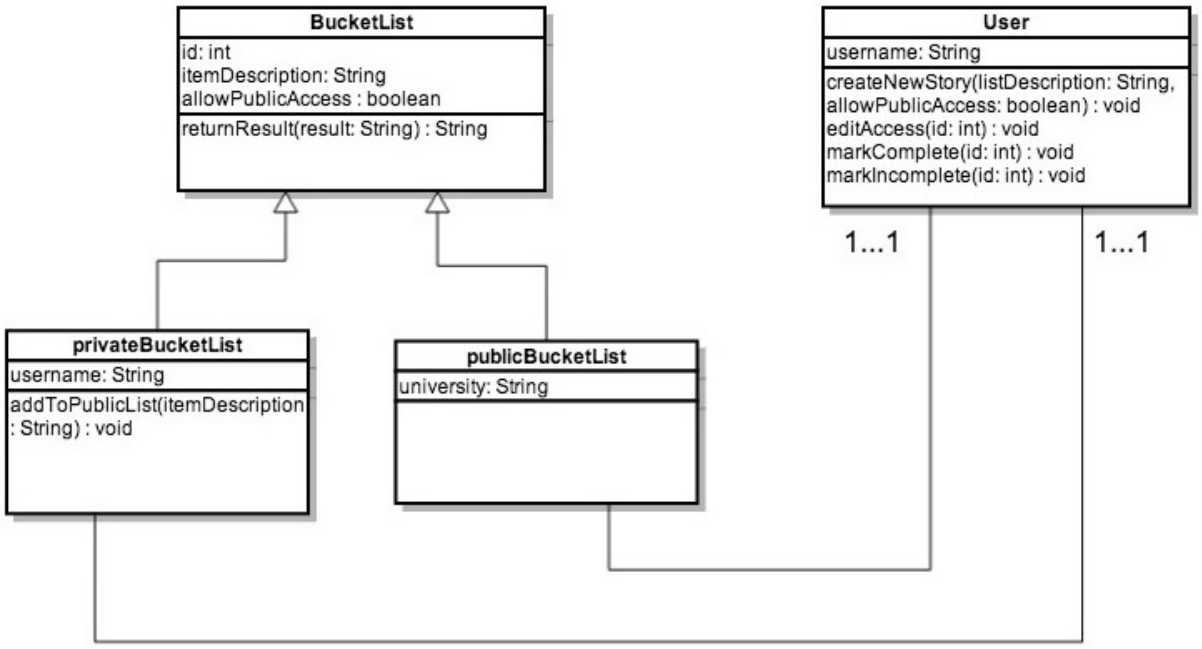

Figure 4.1: Class diagram for adding an item to private and public bucket list pages

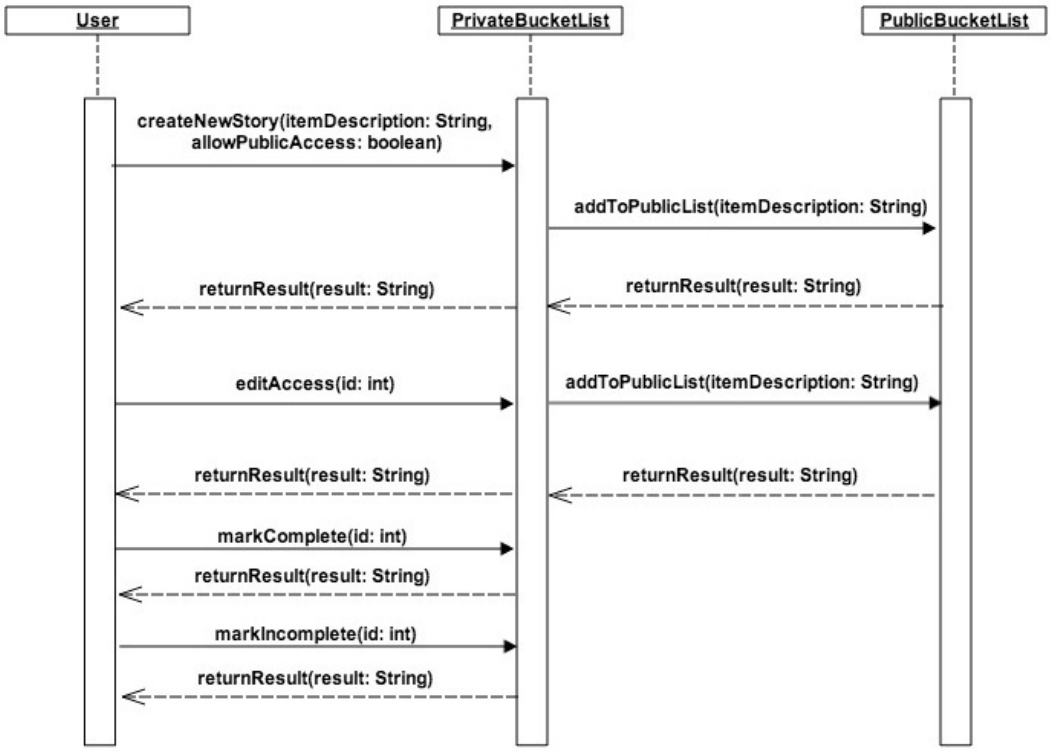

Figure 4.2: Sequence Diagram for adding an item to private and public bucket list pages

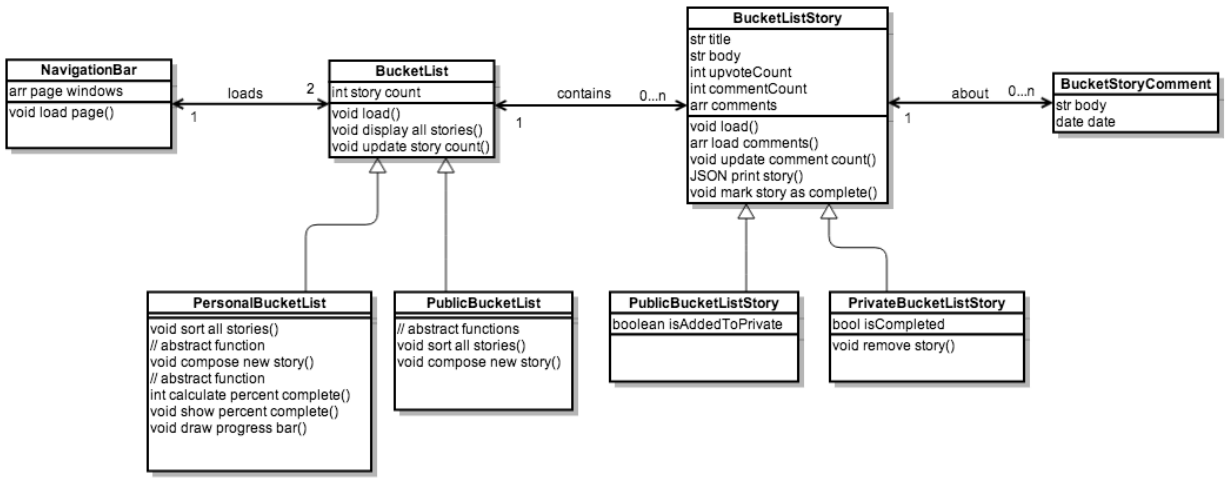

Figure 4.3: Class diagram for public and private bucket list pages

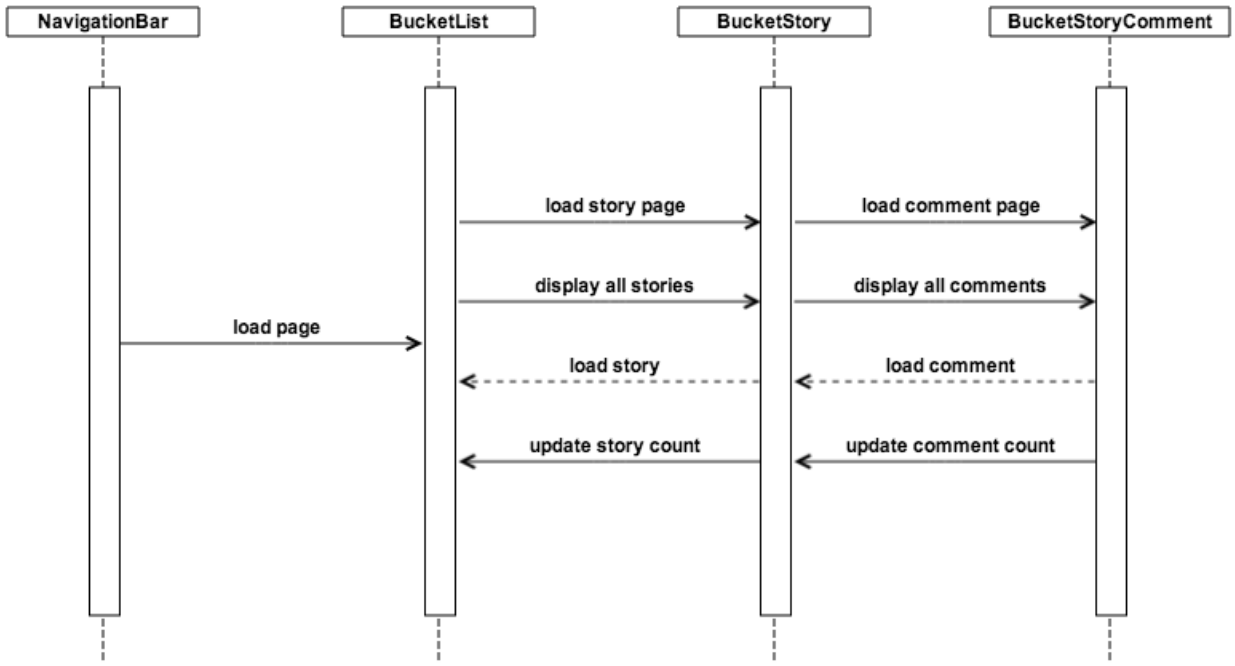

Figure 4.4: Sequence diagram for public and private bucket list pages

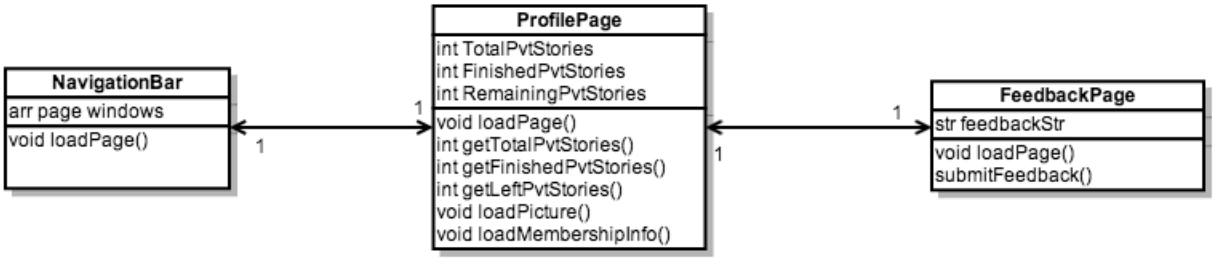

Figure 4.5: Class diagram for user profile and reporting feedback pages

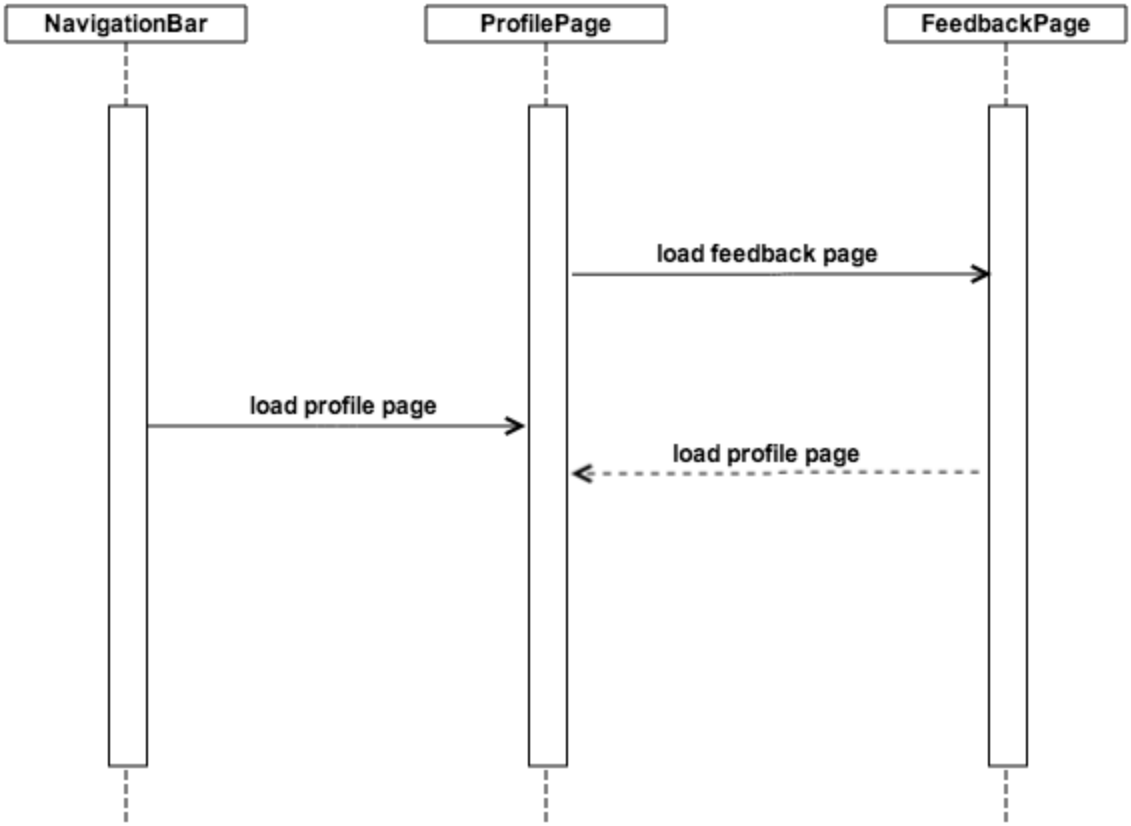

Figure 4.6: Sequence diagram for user profile and reporting feedback pages

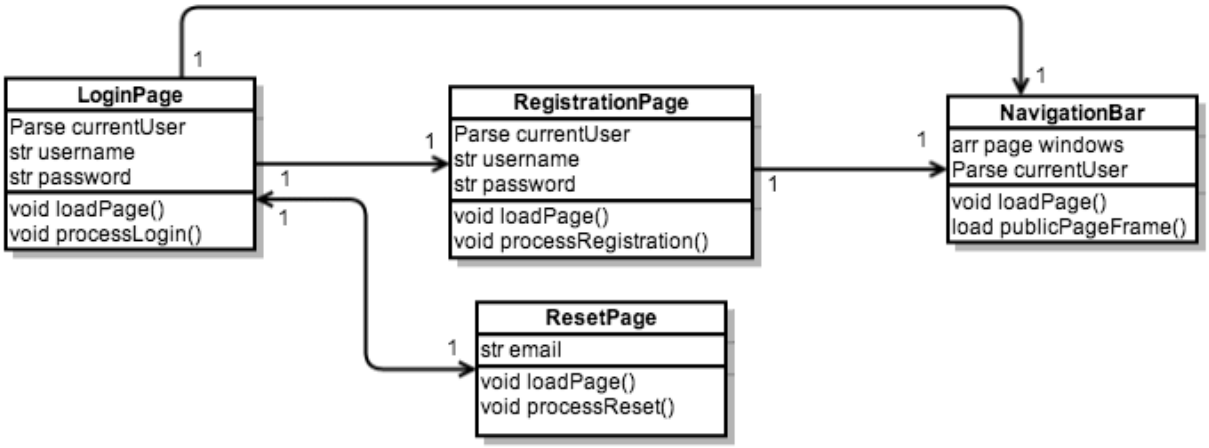

Figure 4.7: Class diagram for existing user logging in and new user registration

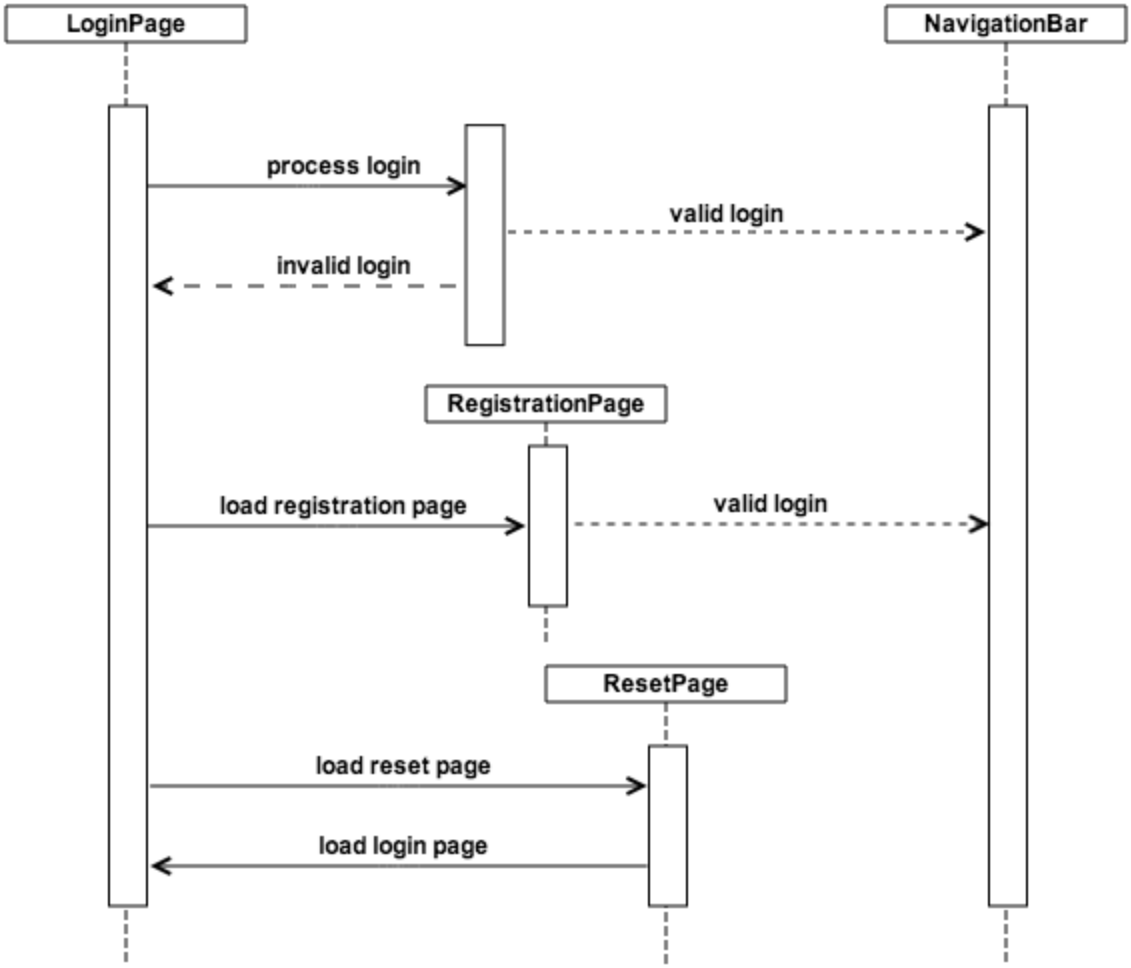

Figure 4.8: Sequence diagram for existing user logging in and new user registration

3.3 Design of System

In detail, this app is mostly composed of controllers and views. This is because the services and factories needed were used mostly for connecting to the *Parse* database. The following is an in-depth account of all application functionalities and how they flow and connect with one another.

Logging in

Upon launching the app from start, we're presented with the login page. As per Figure 4.8, the sequence of pages that the user can visit vary greatly here, and all actions go through `LoginCtrl`, which is the controller for this part of the sequence. The user has the option to log in, register an account with our app, or reset a password associated with an email address.

If the user decides to attempt logging in, we process the credentials with *Parse*. If the login is valid, we store the token and credential information in our app's active memory and change the state from the login page to the index page, where the public bucket list is the primary view. However, if the login is invalid, the user remains on the same page and is notified that the credentials are incorrect.

If the user would like to register an account with our app, they're presented to the registration state, where we register a username and password into our database, store the token and credential information, and proceed them to the index page.

Lastly, if the user would like to reset their password, we direct them to the reset password state, where they input their email address, and we request *Parse* to send them an automated email with a link to reset their password. The `LoginCtrl` controller is responsible for generating these views and processes the user's request against *Parse* with the data provided.

Public Bucket List

After the user has successfully initiated a registered session with our app, and they're directed to our index page, they're presented with the public bucket list page. This public bucket list page allows users to scroll through publicly shared bucket stories, upvote them, add them to their personal bucket list, submit a bucket story, and view comments for each story.

When the public bucket list is loaded for every user, the `PublicCtrl` controller is in charge of processing and controlling all data that's displayed on this page. Upon loading this page, we retrieve the user's private bucket list so that we know which public bucket stories are eligible to be added to their private bucket list. This removes the possibility that the user adds a bucket story to their bucket list more than once. We also retrieve the upvote counts for each public story and whether the user has upvoted the story already. After the page has been completely loaded, the user has many actions they can perform on the page.

The first action that the user can perform is upvoting a bucket story. Black arrows represent that the user hasn't upvoted the story yet, and blue arrows represent that the user has already upvoted the

story. Upon clicking the arrow, we update the boolean value (true for upvoted and false for not upvoted) of that story for the user and use CSS to change the color of the arrow.

The second action that the user can perform is adding a story to their private bucket list. When the public bucket list was initially loaded, the "+" button at the right of each story generated to represent that the user has not already added them to their private bucket list. Upon clicking the "+" button, the controller appends this story to an array of stories added to the private bucket list, request *Parse* to update its private bucket list table, and remove the "+" button from that story's view. Thus, when a public bucket story has been added to a user's private list, we instantly update our database and asynchronously remove the "+" button.

The third action for the user is submitting a new bucket story. This submission opens up a new modal, where the user inputs a bucket story title and can toggle a switch to make this a private or public story. All submitted stories are added to the user's private bucket list, and public stories are added to the front page for all users to upvote, add, and comment on. By default on the public bucket list, the switch is toggled to public by default. Upon submitting the story, the new bucket story is asynchronously added to the public page.

The fourth action that the user can perform is viewing comments for each public bucket story. When each bucket story is rendered on the public list, we also display an updated comment count for each story. Each comment count is hyperlinked to direct the user to another controller and view, which will load all the comments for that bucket story, which is controlled by `PublicCommentsCtrl`.

Public Bucket Story Comments

When the user has clicked on the comment count for a story in the public page, they're directed to a different view powered by `PublicCommentsCtrl`. This comment page for the bucket story loads the story title, an upvote button, add a new comment button, all of the comments pertaining to the bucket story and the date each one was posted.

When a user clicks on the new comment compose button, a new modal window slides up from the bottom of the screen prompting the user to enter a new comment for the respective story. The user can then enter his/her comment and click on *Post Comment* to submit their comment. Upon submission their comment is stored in the `comments` database in *Parse* and the modal window slides back down out of the view of the screen. The user also has the option to cancel composing their comment which simply slides the modal down out of the view of the screen without submitting their comment.

Private Bucket List

Upon clicking the tab labeled "*My List*" contained in the navigation bar, the user will be directed to his/her own private bucket list. Here is where the user may keep track of all of the items he/she would like

to complete separate from the public, university-wide list. All functionality contained in this page is controlled by the `PrivateCtrl` controller. The user may carry out a variety of options on this page.

The first action the user may perform is adding a new bucket list item to his/her own list. Upon clicking the new story compose button, a new modal window slides up from the bottom of the screen prompting the user to enter a new bucket list item. The user can then enter his/her new story item as well as determine if they would like to also share their new story to the public, university list. If the user toggles the switch "on" and then clicks on "*Create Story"*, their new story will be stored in both the `BucketToDoPrivate` as well as `BucketToDo` databases in *Parse* and the modal window slides back down out of view of the screen. If the user toggles the switch "off" and then clicks on "*Create Story"*, their new story will be stored in just the `BucketToDoPrivate` database in *Parse* and the modal window slides back down out of view of the screen. The user also has the option to cancel composing their comment which simply slides the modal down out of the view of the screen without submitting their comment.

The second action the user may perform is deleting comments. The user may do this by sliding from right to left on top of a specific story they would like to delete. After sliding, a red delete button will appear to the right of the story. Upon clicking on the delete button, the application will prompt the user if he/she is sure they would like to delete the story. If the user confirms the deletion, the story will disappear from the list and will also be deleted from the `BucketToDoPrivate` database in *Parse*. If the user does not confirm the deletion, the prompt will go away and the story will not be deleted.

The third action the user may perform is checking and unchecking bucket list items. The user may check off an item he/she has already completed by clicking on the empty circle to the left of the bucket list item. Upon clicking, the circle will fill with a check and the progress bar at the top of the list will update to reflect the amount of items completed. The appropriate data in *Parse* also will be updated to reflect the change. The user may uncheck an item by clicking on the checked circle to the left of the bucket list item. Upon clicking, the circle will become empty and the progress bar at the top of the list will update to reflect the change. The appropriate data in *Parse* also will be updated to reflect the change.

User Profile

Upon clicking the tab labeled "*Profile*" contained in the navigation bar, the user will be directed to his/her user profile. On this page the user can view his/her username, picture, account creation date and a summary of his/her private bucket list items such as the total number of posts they have made, the number of items they have completed and the number of items they have remaining that all update according to the changes the user makes to his/her list. All functionality on this page is powered by the `AboutCtrl`

controller. Upon clicking on the button directly below the summary labeled "*Report a bug*", the user will be directed to a new page where they may provide feedback of the application.

Feedback

After clicking the "*Report a bug*" button on the previous user profile page, the user will be directed to a new Feedback page where they are prompted to enter any questions, comments or concerns they have with our application. After the user enters his/her feedback, they may click on the "Submit" button. Upon submission their feedback will be stored in the `FeedbackMessages` database in *Parse* and a success message, "*Feedback sent. Thank you!*", will appear below the submit button. If the user attempts to submit the same feedback again, an error message will appear below the submit button, "*This has already been submitted!*", and the feedback will not be entered into the database. The user may click on the back button to return to their user profile page. All functionality on this page is controlled by the `FeedbackCtrl` controller.

Section 4: Future Plans

4.1 Future Plans

After we fully secure our application we plan to first submit it to the Android mobile application marketplace so that anyone with an Android device can download and use our application. After monitoring our success in the Android marketplace we will then submit our application to the iOS application store to allow users with iOS devices to download and use our application.

4.2 Personal Reflections

Metin Askaroglu

uBucket has been an amazing learning experience for me. To develop something from just an idea has been a highly stressful and highly rewarding opportunity. It allowed me to apply all that I've learned in CS427 and CS428 concurrently to find a happy median for myself in the way that software gets developed. It has also taught me how to use new technologies and to more effectively communicate roles in a larger project. Time management, communication skills, technical skills, application of the course material, and interest in the development of the product all came together to give me an experience that I would definitely recommend to those interested in taking the course.

Akshay Bajaj

Being able to select a team based on project interests and design an entire project from scratch gave our group an incredible opportunity to push our imagination and experience on so many levels. Instead of having to focus on one project topic (Jenkins plugins) like we did in CS 427, we were given the ability to create something from scratch, and expand on it as much as we wanted. Being able to modify Extreme Programming and improve it for the needs of our group, we were able to become a much more independent and efficient group in terms of commits, tasks finished, and understanding components of this framework. Also, this project taught me how to explore new programming concepts that are very foreign to me, and still be headstrong enough to persist until I understand it inside and out. Most importantly, this project taught me what it takes to plan out an entire project and split it into small release periods such that the entire team can make progress and learn at the same time. I learned how to develop my first mobile app using web frameworks I was afraid of a couple months ago, and that's been incredible to gain a hands-on experience without worrying about getting fired.

Estayvaine Bragg

This project was a great demonstration of what working on real-life projects is like. Our group was comprised of people with different strong suits which we had to utilize throughout the development cycle. Trusting in one another's technical abilities allowed us to make consistent progress on our project. We used a handful of new technologies which were new to the entire group. However, as a team we were able to learn and teach one another along the way to deliver during each iteration. This was a great team in that everyone contributed to the group and meshed well. This project experience taught me more of the non-technical skills that are required in the software development process.

Katie Chrzanowski

I had a really positive experience while working on this project. Being one of the largest group projects I have worked on in my undergraduate career, I learned a lot about time management, teamwork, and adapting to new technologies. Although we had several problems along the way such as with our original database, working cross-platform, and figuring out how to use technologies and frameworks that no one had prior experience with, we were able to work through everything and create an awesome product. This class as a whole has helped me to understand and implement important software development practices that I will carry with me beyond the university.

Ricky Lung

The project provided a great platform for exploring various new technologies throughout the semester. AngularJS was my first exposure to a more comprehensive Javascript framework involved in organizing everything around the MVC model while maintaining asynchronicity in view rendering and model manipulation. Parse was also the first time I had ever come in contact with such convenient methods of maintaining and accessing data sets without having to dirty my hands too much with database infrastructure. I also appreciated working and collaborating with the team. In general the CS curriculum does not often support project teams of these sizes and just the exposure to working in teams is a learning experience. In particular I would say I now have a better appreciation of the necessity of organization and coordination when development teams grow in size.

Khulan Myagmardorj

This project overall was a really great experience for me. I learned so much regarding both my technical skills as well as my soft skills. We used so many new technologies that none of us really ever had experience in such as Parse, AngularJS, the PhoneGap and Ionic frameworks, and the Protractor framework used for testing. Working in a relatively large team also simulated a typical work environment which I believe will help me in coordinating and cooperating with others on new projects. The most difficult part of working on this project was the general lack of resources online for these new technologies. However by working together and making our own discoveries we learned as a team and were able to complete a successful, working project that we can be proud of.