Software Engineering
Software Requirements Specification (SRS) Document

[Paper Path]

[9/26/2023]

[Version 1]

By: [Christopher Hochrein, Philip Sijerkovic, Amin Abdelrahim]

[Honor Code]

Table of Contents

au

1. Introduction

1.1. Purpose (Philip, Chris, Amin)

The goal of Paper Path is to make it easy for a newspaper team to complete projects by enabling effective communication between project members and allowing management of project materials.

1.2. Document Conventions (Philip, Chris, Amin)

The purpose of this Software Requirements Document (SRD) is to describe the requirements for the software, and how it will look for end-users. This includes a description of the different actors and their roles. This document will also list out the system requirements to run our application.

1.3. Definitions, Acronyms, and Abbreviations (Chris, Amin)

HTML	Stands for Hypertext Markup Language. This will be used to develop frontend services.
Spring Boot	A framework for developing web applications that run on the Java virtual machine. This will be used to help us develop our backend.
API	Stands for Application Programming Interface. This will be used to develop our system for sending and receiving documents between users. Quill.js

1.4. Intended Audience (Chris)

The intended audience of this document are any investors. It is also intended for any members of the production team, and any project managers.

1.5. Project Scope (Chris, Amin)

The goal of this software is to make it easy for any team behind a newspaper to complete drafts in a expeditious manner. This is in line with the goals of a newspaper business as speedy delivery of content is vital for an industry that moves as quickly as newspaper production.

Using this software will:

- Allow easy management of multiple versions of drafts and photos related to said drafts.
- Enable effective communication between editors and writers.
- Make it easier for writers and editors to communicate with photographers and get photos related to their current work.

1.6. Technology Challenge (Philip)

Using API's and Spring Boot are all new thing's for us in addition to using more than just basic HTML.

1.7. References (Philip)

https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/
https://quilljs.com/docs/api/

2. General Description

2.1. Product Perspective (Chris)

Paper Path was originally created out of the desire to make it easier to work on group projects together.

2.2. Product Features (Chris)

The apllication will allow employees of to fulfill their various roles. Writers are able to upload drafts for editors to view and edit. Once editors have finished editing a draft, they can then send it back to writers for revision, or they can mark the draft as completed. Writers and editors can also make requests for photos that will be uploaded to a bulletin that all photographers in the system can see and access. Photographers can select and respond to requests by uploading photos, which can then be accessed by the parties that made the requests.

2.3. User Class and Characteristics (Chris)

Our application is designed for use by a newspaper writing team. Users should know the basics of using a computer, but the system should be simple and intuitive. The system should allow them to fulfill tasks they were already doing in a more streamlined and accessible way.

2.4. Operating Environment (Chris)

The application will be designed to operate via the web on desktop operating systems.

2.5. Constraints (Chris)

Software must be able to run in browser.

2.6. Assumptions and Dependencies (Chris, Philip)

Software will be dependent on Spring Boot. It will be implemented with the Quill.js API to allow draft editing in-browser.

3. Functional Requirements

3.1. Primary (Chris)

- FR0: The system will allow writers to send drafts to editors.
- FR1: The system will allow editors to edit drafts and add notes for changes to be made. These can then be sent back to writers to produce the next draft.
- FR2: The system will allow writers and editors to post requests for photos to a bulletin viewable by photographers. These requests can contain a written description of the photo they are looking for.
- FR3: The system will allow writers and editors to access photos that are submitted in response to these requests and allow them to close requests.
- FR4: The system will allow photographers to view photo requests on the bulletin, choose requests to accept, and then upload a photo in response to a specific request.

3.2. Secondary (Chris)

User scheme so that user classes can see their relevant data

4. Technical Requirements

1.1. Operating System and Compatibility (Chris)

The application will be compatible with browsers run on desktop operating systems

1.2. Interface Requirements (Chris)

1.2.1. User Interfaces

Each user will have a separate interface, different users will be able to send different files to each other. There will be bulletin pages which display lists of requests made to the various user classes.

1.2.2. Hardware Interfaces

The web app will run on any desktop or laptop computer that can access the internet and access/interact with web pages.

1.2.3. Communications Interfaces

The operating device must be able to connect to the internet. HTTP will be used as the network communication protocol.

1.2.4. Software Interfaces

The frontend will be built using HTML and CSS, connected to the backend with Spring Boot and Java. The backend will be built with Java.

5. Non-Functional Requirements

5.1. Performance Requirements (Philip)

User Authentication, Image uploading, database integration

5.2. Safety Requirements (Philip)

Secure database, User Permissions, Secure API

5.3. Security Requirements (Philip)

Login System, role based authorization via Spring Security

5.4. Software Quality Attributes (Chris)

5.4.1. Availability

Should be available at all times

5.4.2. Correctness

Should always be correct in terms of user authorization and document consistency

5.4.3. Maintainability

Code base should be easy to maintain and modify

5.4.4. Reusability

Reusability is not vital to this project

5.4.5. Portability

Should be usable on any desktop operating system that has access to traditional web browsers

5.5. Process Requirements (Chris)

5.5.1. Development Process Used

Software Process Model

5.5.2. Time Constraint

Must be completed by December 6th.

5.5.3. Cost and Delivery Date

No cost. Delivery date is December 6th.

5.6. Other Requirements (Chris)

N/A

5.7. Use-Case Model Diagram (Chris)

5.8. Use-Case Model Descriptions

5.8.1. Actor: Editor (Philip)

- **Use-Case Name**: Access drafts
- **Use-Case Name**: Create Article requests
- **Use-Case Name**: Send drafts to writers
- Use-Case Name: Mark ready for publication

5.8.2. Actor: Writer (Amin A)

- **Use-Case Name**: Access drafts

- **Use-Case Name**: Send drafts to editors
- **Use-Case Name**: Create photo requests
- **Use-Case Name**: View photo requests responses
- **Use-Case Name**: View article requests

5.8.3. Actor: Photographer (Chris)

- **Use-Case Name**: Delete Photo
- **Use-Case Name**: Upload Photos
- **Use-Case Name:** Edit Photo requests

5.9. Use-Case Model Scenarios

5.9.1. Actor: Editor (Philip)

- **Use-Case Name**: Access drafts
    - ∉ **Initial Assumption**: User reaches a page with a grid list of all drafts/completed work saved in system
    - ∉ **Normal**: User can preview drafts in alphabetical/chronological order, then select a draft to continue working on it,
    - ∉ **What Can Go Wrong:** User could accidentally delete a draft. User could access drafts they don't have authorization for.
    - ∉ **Other Activities**: Delete a draft, make a copy of draft
    - ∉ **System State on Completion**: User can view and edit all authorized drafts

- **Use-Case Name**: Send drafts to writer
    - ∉ **Initial Assumption**: User has a draft to be sent to the writer.
    - ∉ **Normal**: User selects a desired draft and recipient.
    - ∉ **What Can Go Wrong**: Draft gets sent prematurely. Writer fails to receive draft.
    - ∉ **Other Activities**: View list of previously sent drafts
    - ∉ **System State on Completion**: Draft has been sent to writer

- **Use-Case Name:** Create article requests
    - ∉ **Initial Assumption:** User has access to the bulletin
    - ∉ **Normal:** User will write a description of the article they want and send the request to a designated writer
    - ∉ **What Can Go Wrong:** User could have errors in their writing of the request
    - ∉ **Other Activities:** Users can edit and delete requests they have uploaded
    - ∉ **System State on Completion:** The photo sent and received by the designated in writer

5.9.2. Actor: Writer (Amin)

- **Use-Case Name**: Access drafts
    - ∉ **Initial Assumption**: User reaches a page with a grid list of all drafts/completed work saved in system

    - ∉ **Normal**: User can preview drafts in alphabetical/chronological order, then select a draft to continue working on it. User can see edits/suggestions made by editor (if applicable)
    - ∉ **What Can Go Wrong:** User could accidently delete a draft. User could access other writers' drafts.
    - ∉ **Other Activities**: delete a draft, make a copy of draft
    - ∉ **System State on Completion**: User can view and edit all authorized drafts

- **Use-Case Name**: Send drafts to editor
    - ∉ **Initial Assumption**: User has a draft to be sent to the editor.
    - ∉ **Normal**: User selects a desired draft and recipient.
    - ∉ **What Can Go Wrong**: Draft gets sent prematurely. Editor fails to receive draft.
    - ∉ **Other Activities**: View list of previously sent drafts
    - ∉ **System State on Completion**: Draft has been sent to editor

- **Use-Case Name:** Create photo requests
    - ∉ **Initial Assumption:** User has an account with the permission to post to the bulletin board
    - ∉ **Normal:** User will write a description of the photo they want and upload the request to a bulletin board.
    - ∉ **What Can Go Wrong:** User could have errors in their writing of the request
    - ∉ **Other Activities:** Users can edit and delete photo requests they have uploaded
    - ∉ **System State on Completion:** The photo request is posted on the bulletin board and viewable by photographers in the system

- **Use-Case Name**: View photo request responses
    - ∉ **Initial Assumption**: User has uploaded a photo request and/or there has been a photo uploaded in response to a photo request.
    - ∉ **Normal**: User selects a photo request to view from the UI
    - ∉ **What Can Go Wrong:** User could close a request prematurely
    - ∉ **Other Activities**: Close a photo request
    - ∉ **System State on Completion**: User is able to view a photo request and content associated with it

- **Use-Case Name**: View article requests
    - ∉ **Initial Assumption**: Editor has uploaded a article request
    - ∉ **Normal**: User selects a article request to view from the UI
    - ∉ **What Can Go Wrong:** User could close a request prematurely
    - ∉ **Other Activities**: Close a article request
    - ∉ **System State on Completion**: User is able to view a article request and content associated with it

5.9.3. Actor: Photographer (Chris)

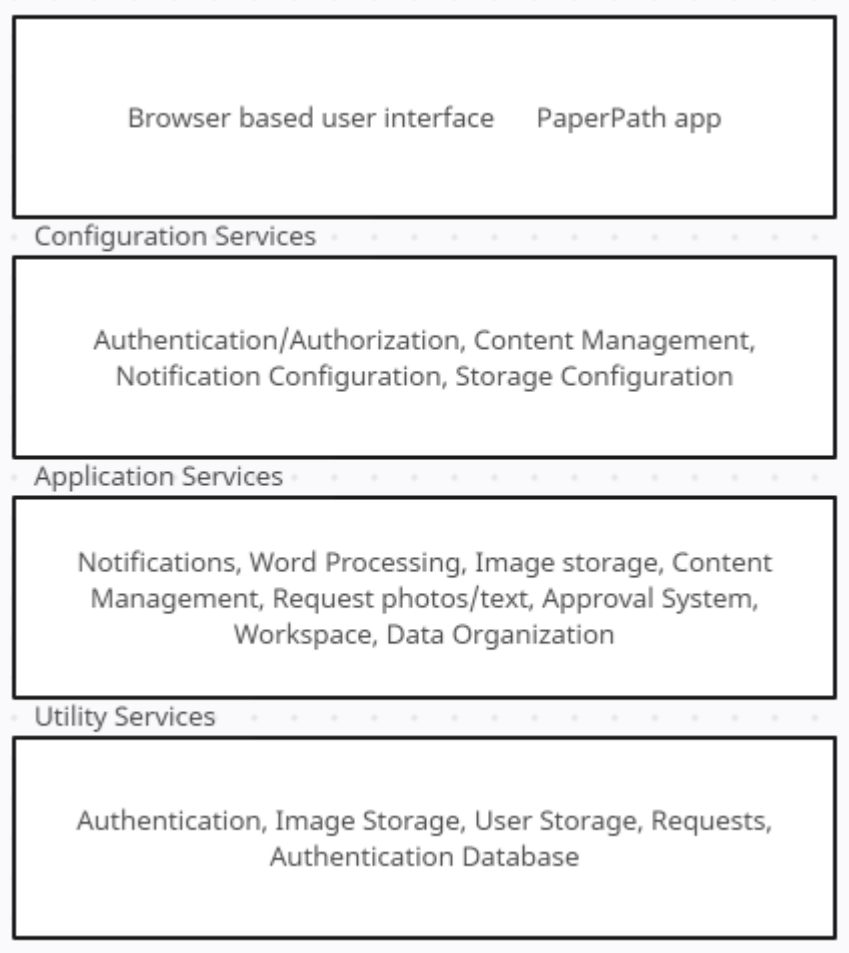
- **Use-Case Name**: Delete photo
    - ∉ **Initial Assumption**: User has uploaded a photo request and/or there has been a photo uploaded in response to a photo request.
    - ∉ **Normal**: User selects a photo request to delete from the UI
    - ∉ **What Can Go Wrong:** User could close the wrong request
    - ∉ **Other Activities**: N/A
    - ∉ **System State on Completion**: Request has been returned to public bulletin

- **Use-Case Name:** Upload photos
    - ∉ **Initial Assumption:** User has a photo to be uploaded and a request to upload it to
    - ∉ **Normal:** User accesses request, attaches file, and then uploads
    - ∉ **What Can Go Wrong:** user could upload wrong photo
    - ∉ **Other Activities:** N/A
    - ∉ **System State on Completion:** Photo has been uploaded to request, requester can view photo

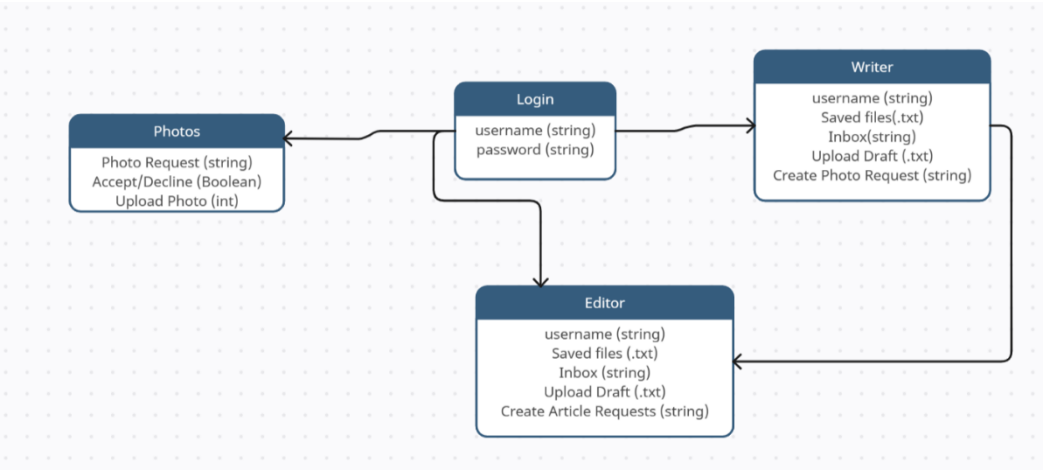
- **Use-Case Name:** Edit Photo requests
    - ∉ **Initial Assumption:** User has at least one photo request previously uploaded
    - ∉ **Normal:** User accesses request, edits request, and reuploads
    - ∉ **What Can Go Wrong:** User could upload in error
    - ∉ **Other Activities:** N/A
    - ∉ **System State on Completion:** Photo request has been reuploaded with altered content

6. Design Documents

6.1. Software Architecture (Layered) (Philip)

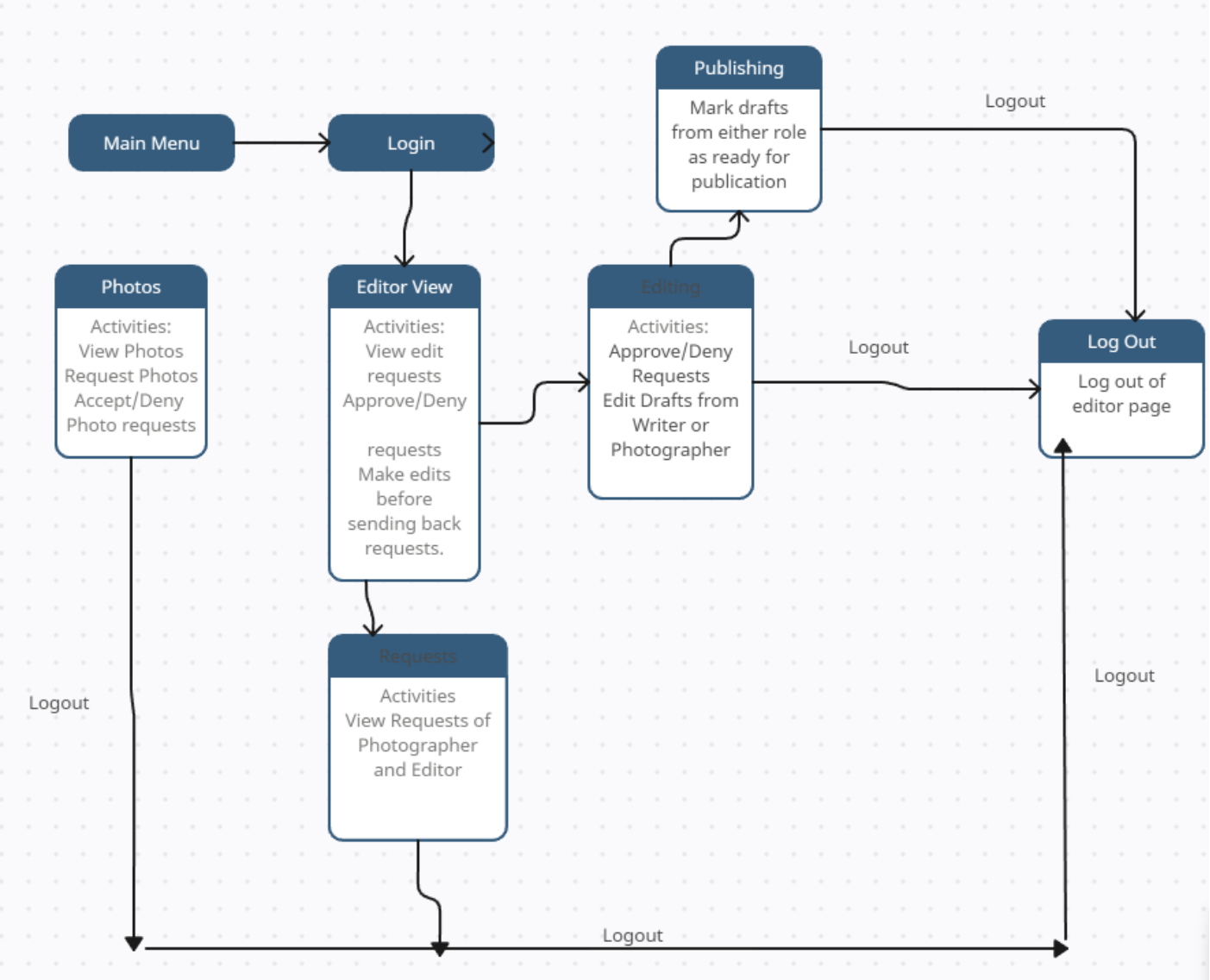


6.2. High-Level Database Schema (Amin)

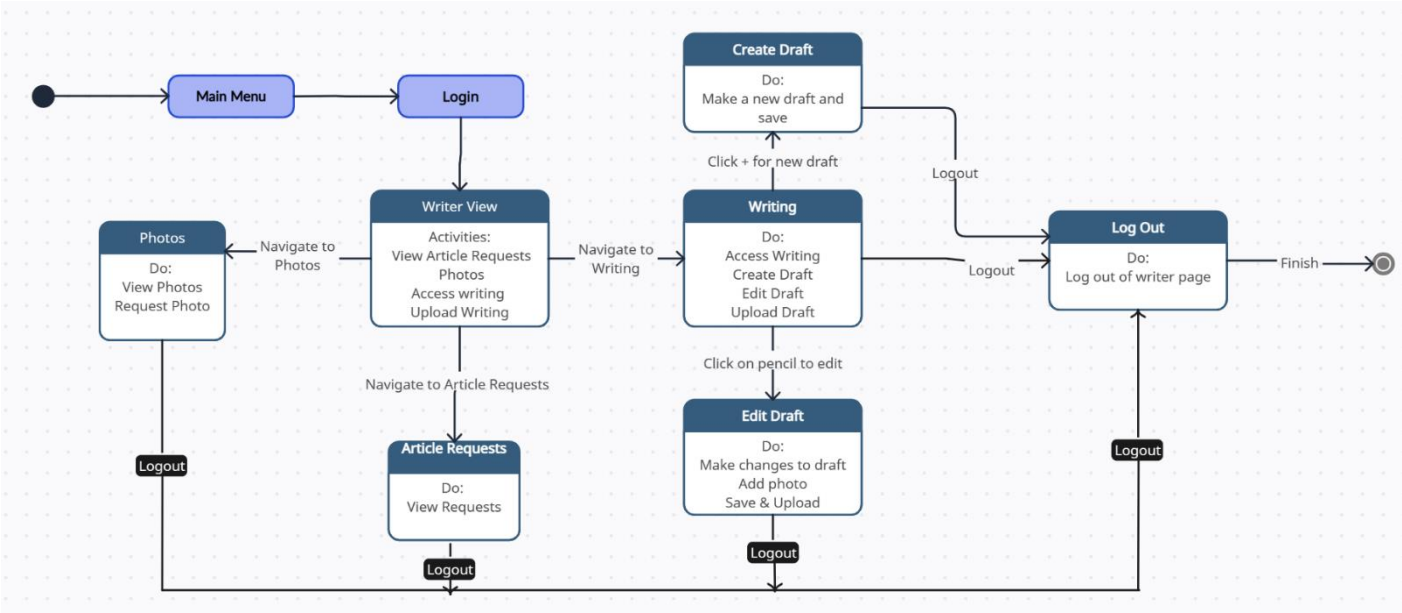


6.3. Software Design

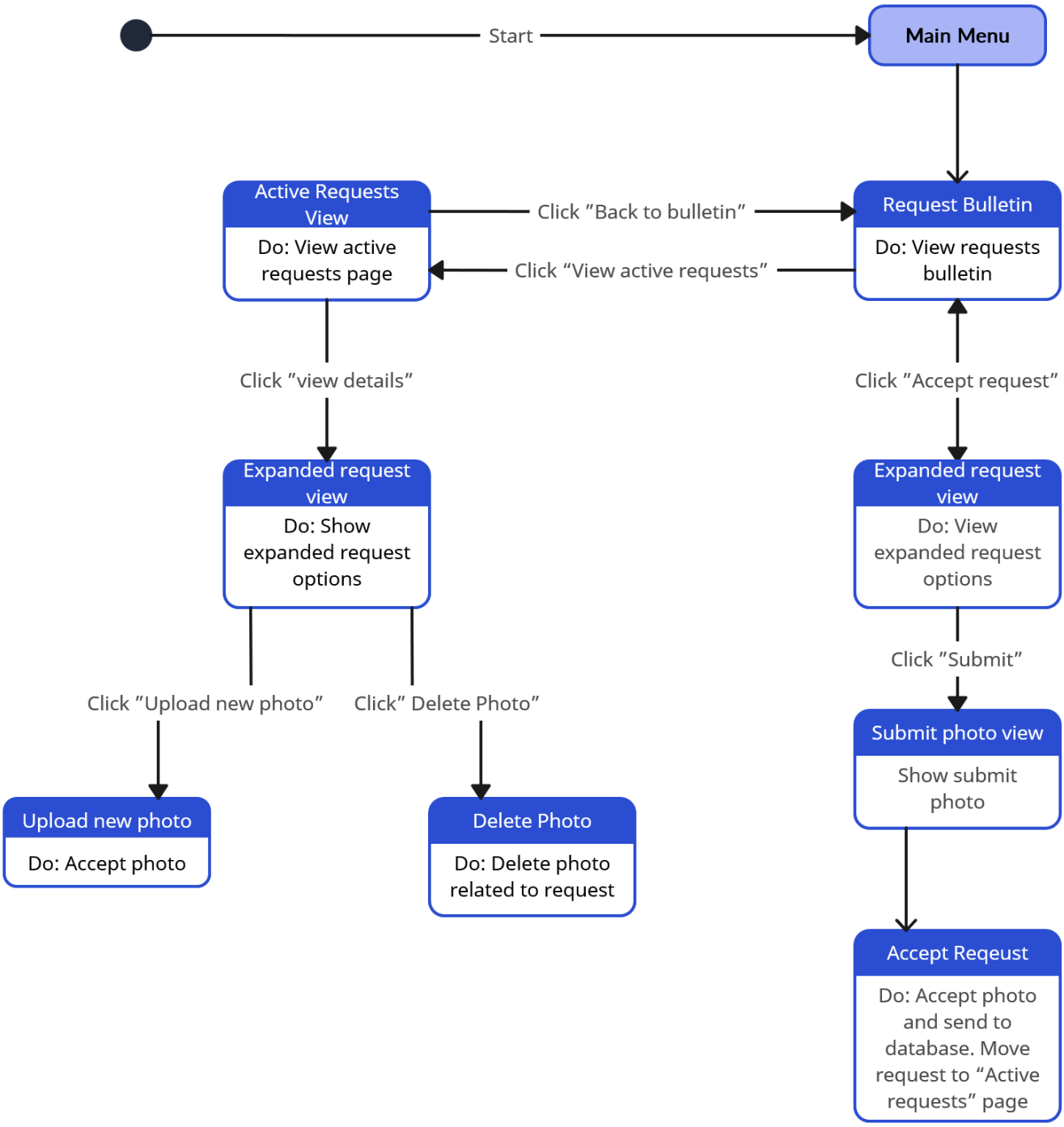
6.3.1. State Machine Diagram: Editor (Philip)



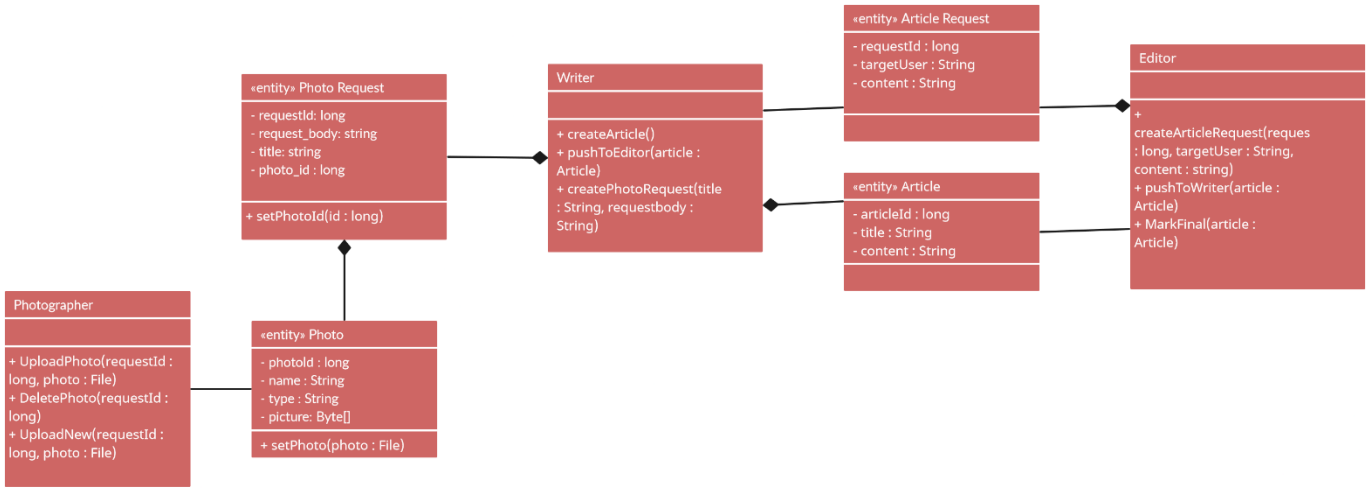
6.3.2. State Machine Diagram: Writer (Amin)



6.3.3. State Machine Diagram: Photographer (Chris)



6.4. UML Class Diagram (Chris)

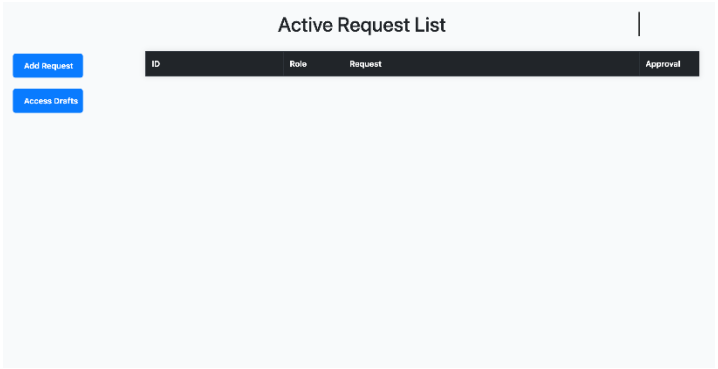


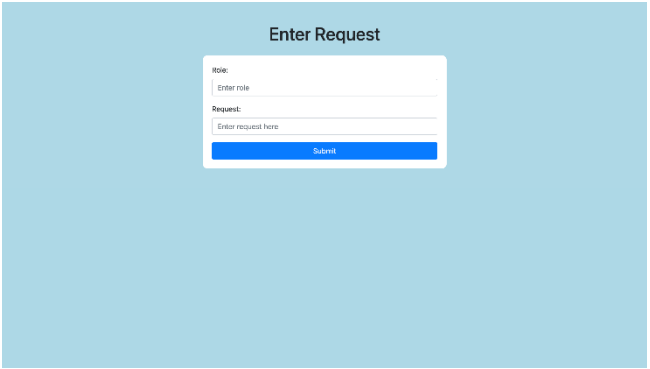
7. Scenario

7.1. Brief Written Scenario with Screenshots (Philip, Chris, Amin)

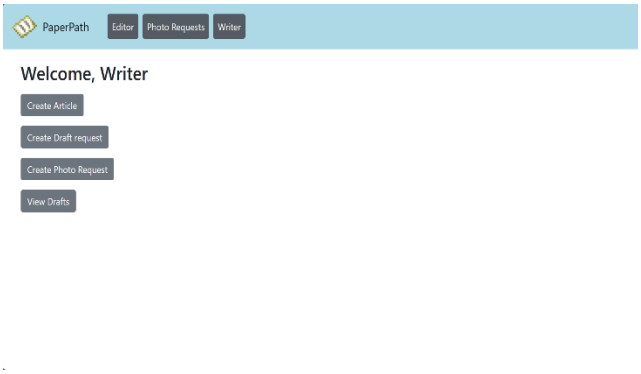
* The use cases being demonstrated are denoted with blue text

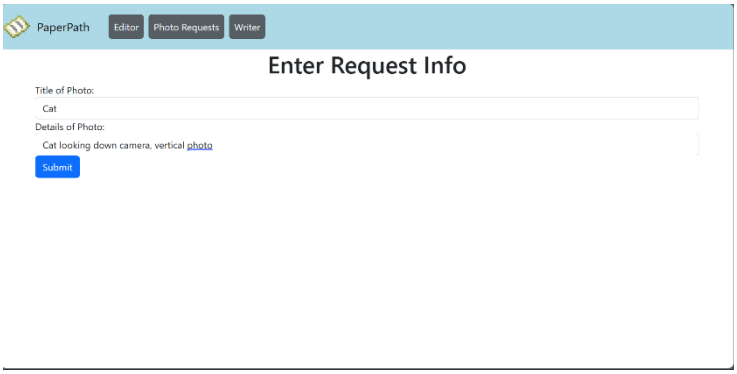
The editor wants to create an article request. The editor accesses the editor page, and then clicks the add request button to pull up the form (create article requests).



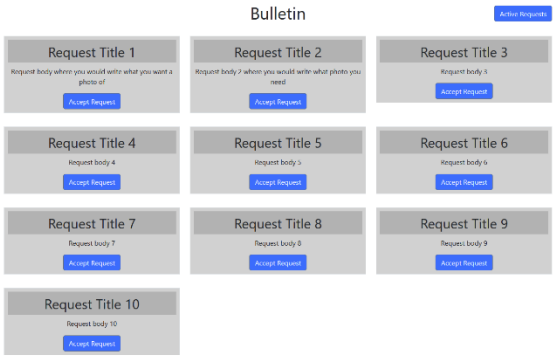


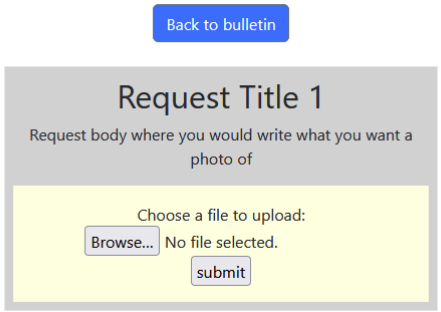
Switching to the writer, the writer moves to the “view article requests” page (only accessible through database currently) (view article requests).
The writer then decides they need a photo for the article, so they access their menu page, and click “Create Photo request” to pull up the form to create a photo request (create photo request).



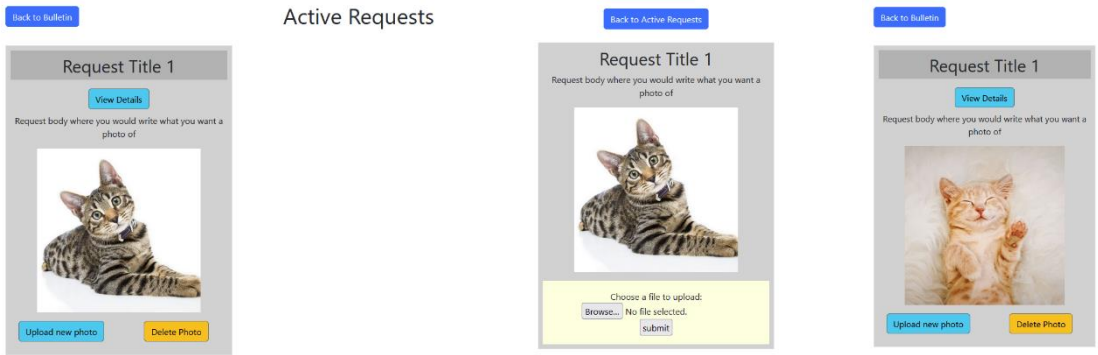


We now switch to the photographer. The photographer accesses the bulletin, finds the photo request submitted by the writer, and accepts it by uploading a photo (accept photo request/upload photo).

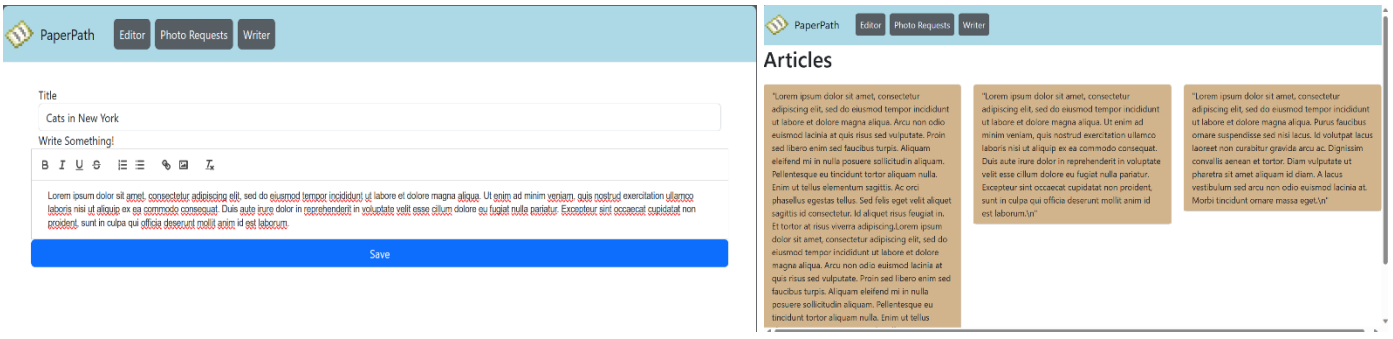




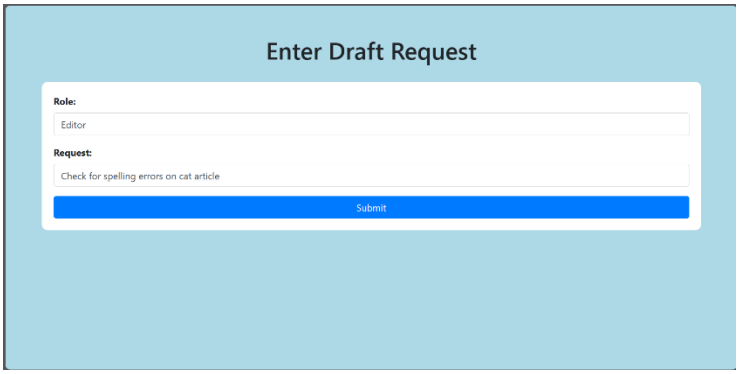
The photographer is then automatically moved to the active requests page. There, the photographer decides they uploaded the wrong photo so they choose to upload a new photo (edit photo request responses).



Back to the writer, they see their request has been complete and download the photo (right click image and save file) (view photo request responses).
Now the writer decides to create an article – From the writer main menu, they click “Create Article” to bring up the editing window. The draft view available in view drafts (Access drafts).



The writer now has a draft they want to send to the editor, so they navigate to the main menu and click “create draft request” (Request editor to revise draft).



Back to the editor, from their main menu they access the draft by clicking the “Access drafts” button (Access drafts). Additionally, the draft request is sent to a table on the main menu where the editor can choose to mark it as final. Click the green button to permanently save the request to the database, click the red to delete the draft entirely (Mark draft as final).