Show ‘N Tell Requirements Proposal

Sean C. Dodson

Stevenson University

IS 349

Show ‘N Tell Requirements Proposal

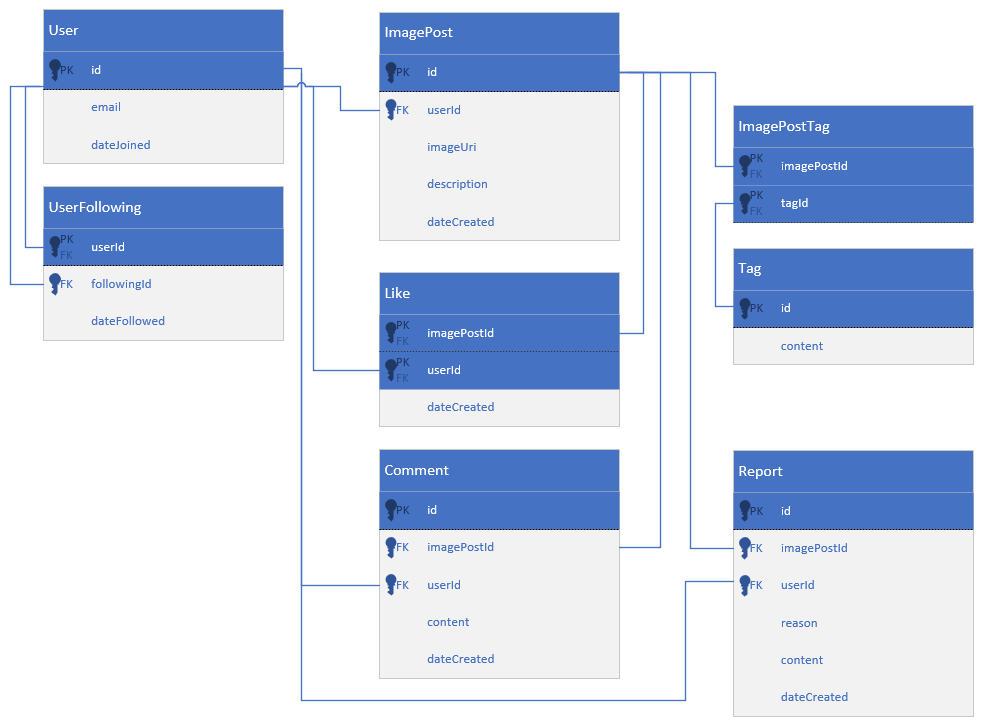
1. **Project Name:** Show ‘N Tell
2. **Description:**

Show ‘N Tell is a social media platform for users to upload and discover images of other user’s sentimental, valuable, or random items. Many people have interesting items with compelling background stories that they value and think other people would appreciate. Show ‘N Tell will allow people to upload images and write stories about their interesting items for the world to recognize.

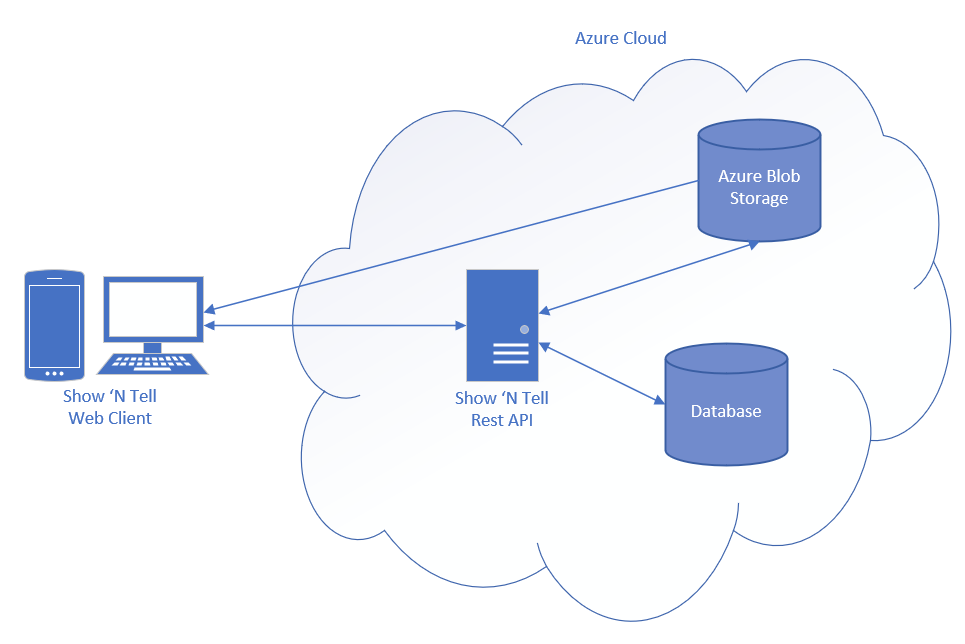
People are interested in and curious about other people’s items and stories. However, mainstream social media platforms do not specifically support this interest. Instagram, Twitter, and Facebook are general purpose social media platforms that include news, current events, status updates, advertising, and various images, such as selfies. Show ‘N Tell will specialize as a social media platform where users can endlessly explore, rate, and comment on other people’s interesting items. Endless exploration is a feature that has benefitted many social media platforms, such as TikTok where users can endlessly explore short video clips.

1. **Features (*Rated 1-5: 1 being the highest priority, 4+ being optional*)**
   1. Allow users to Create, Read, Update, and Delete (CRUD) images of their sentimental items with descriptions [1]
   2. Allow users to explore images posted by other users and leave comments or likes on images [2]
   3. Integration with Google Authentication to allow any user with a Google account to seamlessly use Show ‘N Tell [1]
   4. Allows users to follow each other and view a feed of images recently posted by users they follow [3]
   5. Add categories or hashtags for grouping user images and allowing users to filter specific content [4]
   6. Allow users to report inappropriate images. When an image reaches a certain amount of reports, the image will be temporarily blocked and reviewed [5]
   7. Implement real-time feed updates [5]

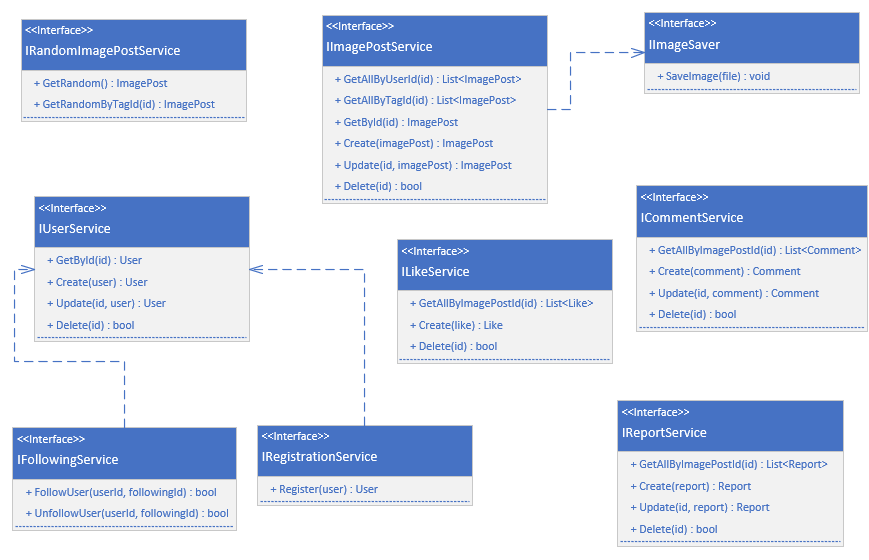
The most important feature is allowing users to create, read, update, and delete images of their items. This is the top priority because it accomplishes one of the main goals of the application which is to upload images of sentimental or valuable items. In order to upload images, users will need to authenticate. Therefore, adding Google authentication is the other most important feature. The second and third sets of priorities facilitate socialization and communication within the application and offer essential social value to the user. The fourth and fifth sets of priorities include useful features, such as real-time feed updates, but are not essential for the main functionality of the application.

1. **Database Diagram **

The database diagram supports creating, reading, updating, deleting, liking, commenting, and tagging images. Despite being optional features, the diagram also supports reporting images and users following other users. An important feature to note is that the User table will not contain a hashed password because Google will handle authentication.

1. **System Diagram**
   1. Architecture

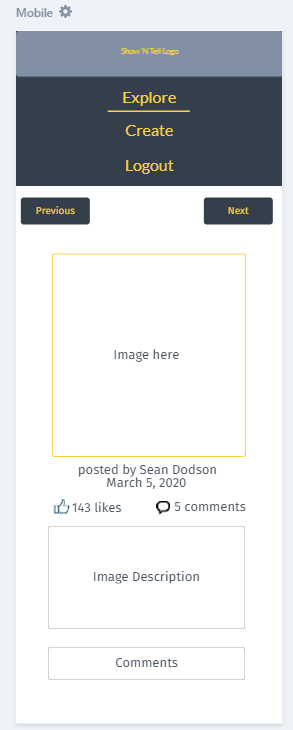
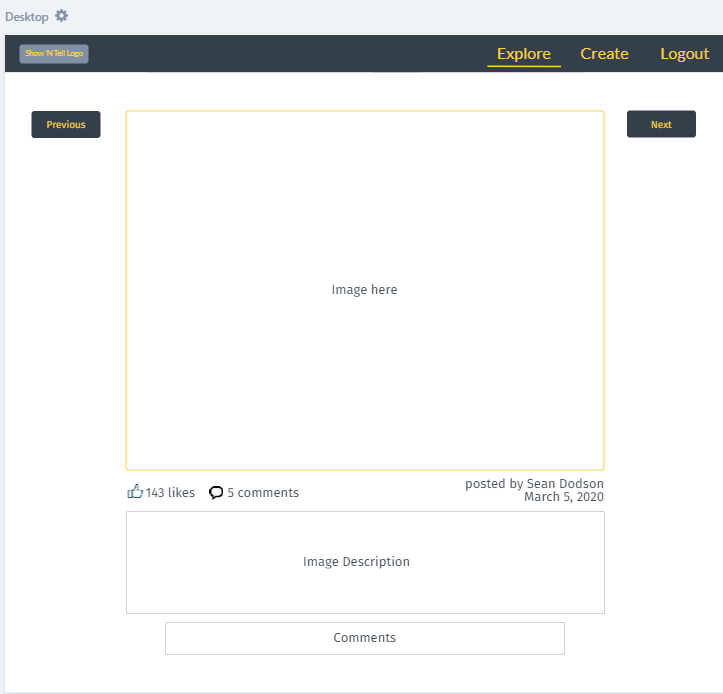
The web client will make Fetch API requests to the Show ‘N Tell Rest API. The Rest API will return JSON to the client consisting of data from the database and/or references to images in Azure Blob Storage. Azure Blob Storage will send images directly to the web client via URIs. Overall, the back end will be implemented in the cloud on Azure.

* 1. API Service Interfaces

The Rest API will implement the following domain services in order to accomplish the applications features. The application will use the ImagePostService, CommentService, UserService, ReportService, and LikeService interfaces to perform CRUD operations on the corresponding entities and perform specific data queries. Data queries may potentially be split up into separate interfaces in order to reduce unnecessary coupling and large classes, such as by implementing interfaces similar to the RandomImagePostService.

Although interfaces do not explicitly depend on other interfaces, dependency arrows are included in the diagram to demonstrate dependencies I expect to occur in the interface implementations. For example, I expect the ImagePostService to use an ImageSaver in the Create and Update methods. The ImageSaver interface will abstract the dependency on Azure Blob Storage and allow the option to switch to local, static file storage if necessary.

1. **UI Mockups**



The UI mockups display the “Explore” page of the application inside the default layout for desktop and mobile devices. The application must be accessible on desktop and mobile devices for maximum audience inclusion. This page is important for allowing users to continuously view images that users have posted and leave likes or comments. Clicking next will show the user a random image and clicking previous will show the user the image they last viewed, if available. The system will reuse the aesthetic design and layout of this page across other views, such as in the view for editing a post.

1. **Usage Requirements**
   1. Scalability – as a social platform, the back end must be scalable as user demand increases
   2. Response Time – the back end must rapidly respond to requests in order to fulfill users constant browsing
   3. Layout Responsiveness – the front-end must be sufficient for mobile and desktop usage
2. **Technologies Implemented**
   1. ASP.NET Core – to implement Rest API interfacing database
   2. Swagger UI – to document Rest API
   3. Entity Framework Core – to map entities to database tables
   4. Azure SQL Database – to store database data
   5. Azure Storage Accounts – one storage account to store user images as blobs and another storage account to host static website front-end
   6. Azure Functions – to listen to deletes in storage account and notify the database that an image was deleted (optional, but will ensure data integrity)
   7. Vue – to create a robust front-end
   8. Bootstrap – to style and layout front-end
3. **Project Task List**

The following is an estimate of the time required to complete the project tasks. Due to inferior experience with JavaScript development compared to C# development, I expect to contribute many hours to the front-end.

|  |  |  |  |
| --- | --- | --- | --- |
| Task | Start | Hours | End |
| Implement domain layer | 3/8 | 3 | 3/8 |
| Setup database and Entity Framework Core | 3/10 | 4 | 3/11 |
| Add database entity repositories | 3/14 | 8 | 3/17 |
| Implement Rest API controllers | 3/18 | 8 | 3/21 |
| Setup Azure Storage | 3/22 | 3 | 3/23 |
| Add Google server-side token authentication | 3/24 | 2 | 3/25 |
| Setup Swagger UI | 3/26 | 1 | 3/26 |
| Create Azure Function for image blob deletes | 3/27 | 5 | 3/29 |
| Add JavaScript API client service layer | 3/30 | 8 | 4/3 |
| Create front-end layout | 4/4 | 6 | 4/7 |
| Add component for viewing a user image post | 4/8 | 6 | 4/12 |
| Add Google client-side authentication logic | 4/13 | 3 | 4/15 |
| Add forms to create, update, delete, and report image posts | 4/16 | 10 | 4/20 |
| Add components for commenting and liking posts | 4/22 | 6 | 4/25 |
| Design a logo | 4/26 | 4 | 4/26 |
| Setup Azure deployment | 4/26 | 8 | 4/30 |
| Create demo video | 4/30 | 5 | 5/1 |