1 Introduction

The PWA allows user to search, create, and add comments to both events and user stories. Social features such as like, follow, interested, and going are integrated. Users are able to tag an event with their stories, which will appear in the event page. Users can create a story by taking a picture with their camera or upload a picture locally. Each user story can receive likes and comments, which the latter is implemented using Socket.IO [4, 1]. Service worker is implemented to cache requests for offline usage. MongoDB is used to store and synchronise data between the client and server. IndexedDB is used to store data loaded from MongoDB for offline usage.

The Progressive Web App (PWA) allows users to search, create, tag events with stories. WebRTC is implemented for users to take pictures with front and environment camera. Uploads are made available for photo selection from a folder. Each story can receive likes and comments which is implemented with SocketIO and users are able to follow one another. Attended and interested events are stored and displayed for each user. Service worker is implemented to allow offline usage with IndexedDB used to store data locally. MongoDB is used to store data but can only be retrieved when user is online. The search function (via location) was implemented with Google API, allowing selected location to be displayed on the map. For security purposes, users can only login with respective Google Account.

2 Diagrams

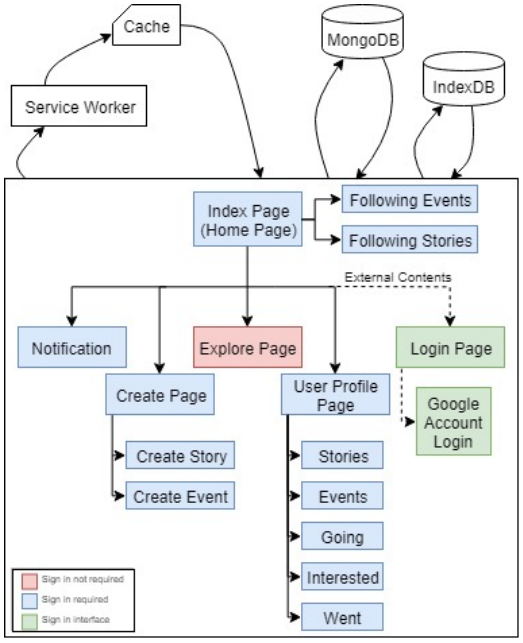

Figure 1: Demonstrates the flow of each web page in this PWA system along with the respective partial pages and external content pages.

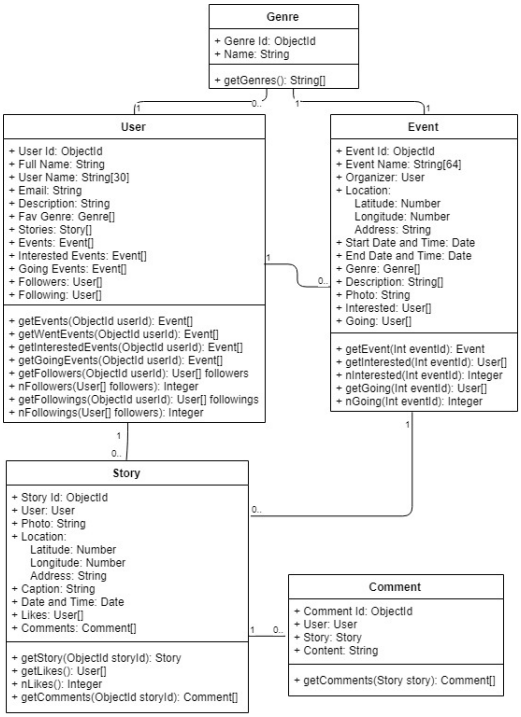

Figure 2: Displays the structure of the database along with the types of content stored.

Figure 1 and figure 2 are the detailed description of the PWA system structure with figure 1 describing the front-end, data storage and data retrieval, while figure 2 describe the types of data stored in the database along with the relationship between the documents.

3 Interface to Insert and Search Data via Forms

- **Challenges**:
  - Usage of search libraries which would allow suggestions and auto complete.
  - ElasticSearch only allowed online searching and would cause errors when users are offline.
- **Solution**:
  - Online library that would support offline searching so that users can make use of this functionality both during online and offline
  - Not yet implemented
- **Requirements**:
  - Users should be able to search for events both during online and offline.
  - Users should be able to search events based on events based on partial event names.
- **Limitations**: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus bibendum turpis in sollicitudin molestie.

4 Interface to Search Data via Map

- **Challenges**:
  - Implementation of Google API could not be cached.
- **Solution**: Not yet implemented
- **Requirements**:
  - Allow users to search for events' location using map.
  - Users should be able to use this search both offline and online.
- **Limitations**: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus bibendum turpis in sollicitudin molestie.

5 PWA – Caching of the App Template Using a Web Worker

- **Challenges**:
  - Image of events are not cached on initial load.
  - Unable to cache user page through the use of service worker.
- **Solution**: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus bibendum turpis in sollicitudin molestie.
- **Requirements**:
  - Users should be able to view a basic template with required data for when offline.
  - Users should be able to previously loaded events and stories without the ability to make any changes to the events, stories, or their profiles.
- **Limitations**:
  - Users are only able to view a basic offline template if no data were loaded in the past when the user was online.

6 PWA: Caching Data Using IndexedDB

- **Challenges**:
  - The usage of IndexedDB increases with every *put*() operation. [2, 3].
- **Solution**:
  - There is no known solution at the moment, please refer to the links for more informat
- **Requirements**:
  - Optimise IndexedDB to cache data to allow offline usage with limited data available.
  - Users should be able to access essential data when offline for the application to serve its purpose.
- **Limitations**:
  - IndexedDB is stored locally and could not be updated until user is online.

7 NodeJS Server Including Non-Blocking Organisation of Multiple Dedicated Servers

- **Challenges**: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus bibendum turpis in sollicitudin molestie.
- **Solution**: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus bibendum turpis in sollicitudin molestie.
- **Requirements**: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus bibendum turpis in sollicitudin molestie.
- **Limitations**: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus bibendum turpis in sollicitudin molestie.

8 MongoDB

- **Challenges**:
  - Loading of initial data to populate the database.
  - The retrieval and storage of images in the form of bits.
- **Solution**:
  - Initial population of database was resolved with the use of promises.
  - Image retrieval and storage was implemented using *base*64 encoding.
- **Requirements**:
  - Data stored in MongoDB is stored online. Data is not stored locally and cannot be retrieved offline. When user is online, data is retrieved from the database and displayed on the PWA.
- **Limitations**:
  - Initialisation of database needs to be run separately (refer to 'Extra Information').
  - Data stored on MongoDB can only be accessed when user is online.

9 Quality of the Web Solution

- **Challenges**: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus bibendum turpis in sollicitudin molestie.
- **Solution**: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus bibendum turpis in sollicitudin molestie.
- **Requirements**: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus bibendum turpis in sollicitudin molestie.
- **Limitations**: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus bibendum turpis in sollicitudin molestie.

10 Conclusions

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus bibendum turpis in sollicitudin molestie.

11 Division of Work

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus bibendum turpis in sollicitudin molestie.

12 Extra Information

- **Initial population of MongoDB**: run 'npm run initdb' to drop database and repopulate database with default data.

References

[1] F. Ciravegna, *Week 6 - mongodb and socket.io*, 2019.

[2] *Google/leveldb github issues #593*. [Online]. Available: `https://github.com/google/leveldb/issues/593` (visited on 24/03/2019).

[3] *Google/leveldb github issues #603*. [Online]. Available: `https://github.com/google/leveldb/issues/603` (visited on 24/03/2019).

[4] *Socket.io*. [Online]. Available: `https://socket.io/` (visited on 31/03/2019).