4ME305_Final Assignment: Inherito (integration of web and mobile implementations)

Rema Salman
School of Computer Science, Physics and Mathematics, Linnaeus University, Sweden
rs223bc@student.lnu.se

Yuliya Vatsova
School of Computer Science, Physics and Mathematics, Linnaeus University, Sweden
yv222am@student.lnu.se

1. INTRODUCTION

Heritage discovery and preservation is part of knowing and understanding who we are and why the world we are born in is the way it is. Heritage can be national as found in museums and private collections but also personal by what we got from our ancestors. Harrington, in [1, p.3-32], defined heritage as a process rather than collection of objects, while she discussed the topic of "community" heritage referring to the process of which a space becomes a place. She described this process as when the community shapes a certain space with its values and its unique voice. Nature and even everyday life gives us heritage in the form of sound or pictures of unique moments, which we experience at a specific place or situation. Furthermore, the authors in [2] discussed how modern technologies open new routes for heritage creation and place-making, in the sense of a space that becomes "a place" after humans add a certain community value to the location [1]. Furthermore, [2] presents the idea of Social Production of heritage and exploring "boundaries between tangible, intangible, cultural and natural forms of heritage" [2]. The *Silence of nature Project,* for instance, allowed the creation of a soundscape and nature heritage at a specific location, in a very engaging and innovative way by employing audio recording and geolocation technologies[2]. For the purpose of this project, we decided to experiment by creating a similar system but for image sharing where images can be captured by anyone who wants to contribute to the process of heritage generating. Inherito consists of a web app that displays an imagescape for the heritage consumers while the mobile app provides a channel for heritage contribution. This can happen by taking pictures from spaces that the person wants to make into places. Later on, these contributions are displayed on a map where everyone else who wants to act as a consumer of this heritage. Consuming the heritage can be done via any device connected to the internet and it could happen at any moment in time - 2 seconds or 20 years later (if the project is still maintained). This construct of *making* and *consuming* places created by others allow all customers of the system to act as consumers and vice versa constantly. This gives the project features similar in nature to social media but in this case with laser-focus on our power and responsibility to create content which will become heritage for the generations after us. We believe that such features would differentiate our business idea/system from other existing social media (e.i. Instagram) and interactive map web-services (i.e. Google Places and Foursquare). Our product could be a useful solution for many customers, as the eurobarometer shows that more than half of the Europeans have used the internet to search for cultural heritage related information.[1]

2. RELEVANT PRODUCTS

Thinking of such a system in the context of nowadays social media landscape we have to admit that there are many image based applications which allow end-users to take pictures (mainly with their smart device) and to share it with their online community or everyone who is a customer of the application. Naturally, the most famous application which does that is *Instagram*[2]. It is an image and video sharing social media application which rises up from a personal image content to a place mostly for promoting business or creators, namely influencers[3]. Due to that, our system would guild the contributors to focus on heritage sharing by restricting the use for the back/main camera only to prevent selfie-taking images. What we can learn from this very successful example of consumer - contributor type of application (Instagram) is that image creation and sharing is a powerful tool for users' engagement. Allowing contributors to group images, like in Insta story, is exciting and motivating for both consumer and contributor roles. Additionally, Instagram's functions like photo grid and profile (showing all uploaded images from the user) have proven over the past decade to increase customers satisfaction. Clearly, our solution is not meant to promote business or places but to allow community members by their own collaboration to turn spaces into places and generate heritage content for future generations.

---

[1] Special Eurobarometer 466 Cultural Heritage Report September-October 2017 December 2017
[2] https://www.instagram.com/
[3] Instagram Business Account Vs. Personal. How Do You Identify A Business Account?

Another relevant example is *Cooltura*[4]- an application focused on "cloud-based technologies to enable cultural engagement" [3]. It involves mobile devices and native features, i.e. geolocation to use and reuse cultural heritage content. The Cooltura was developed as a platform under the TAG CLOUD project, co-founded by the European Commision. It supports "creating/detecting "hot spots" for visitors"[3] of places of cultural heritage and allows social sharing. This project sets the need of new ways of presenting and experiencing our common european heritage but also the opportunities for employing modern technologies to interpret the cultural value of this heritage in a manner typical for our modern society. This application is in the same domain as Cooltura shares domain with Inherito but its emphasis is on the cultural heritage mainly while Inherito focuses on social, nature and other kings of heritage. This application gave us inspiration and validation of our idea of a digital product that encourages heritage creation and visualization.

3. SYSTEM OVERVIEW

The system that we developed for the purposes of this project focuses on the domain of creating heritage and place-making by adding personal and community value to any location via image sharing. The system consists of three parts: a web client using map and visualizing data from a server, connected to a realtime database and weather API, and a mobile client using location visualization and updating the real-time database with image, text and location upload. The mobile application provides image data capturing and annotation and the web application allows data description and interpretation. The system expands on four main components: 1) ***Data capturing***: capturing views with the use of mobile device's camera, 2) ***Data annotation***: a) *automatically* by using device's geolocation and b) *intended* by allowing the person to add a caption of the personal value to this view, 3) ***Data description***: uploaded data is shared on the web as an interactive map/imagescape with points of image clusters and 4) ***Data interpretation***: the web application is offering places of interest based on their value. Any location becomes a place of interest either when it is considered as a point of interest from the perspective of Inherito's goals or spaces that have proven themselves as spaces of excess supply. *Places of interest* from Inherito's perspective are spaces with established value as a center of culture, economy, social importance or as a nature sightseeing. *Spaces of excess supply* are considered locations which have a lot of data uploaded by multiple contributors. This would requalify a location into a place [2] as we humans add social value to a particular point and create heritage of it [1]. Further descriptions of the system requirements and architecture is provided in the next subsections.

System Functional Requirements and System Actors

Figure 1 presents the use case diagram of the Inherito system, where the three defined system actors interact with the system. The actors are a contributor, consumer, and system manager. Each use case represents a function requirement of the system. Combining them achieves the system's basic flow/scenario for the specified role.

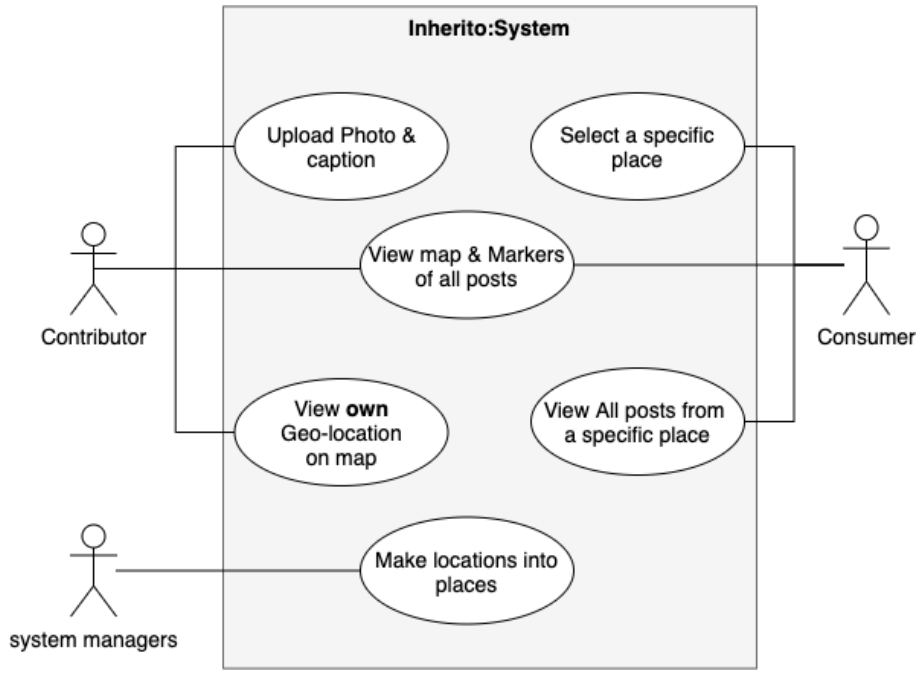

Figure 1. Use case diagram of Inherito system.

[4] https://apps.apple.com/us/app/cooltura/id1034307226#?platform=iphone

As visualized in the figure, the contributor is the only role that can upload an image, the consumer can select a place from the offered list, and the system manager has the authority to manipulate the list of offered places. Meanwhile, both contributors and consumers can view an interactive map with other uploads.

Scenario

Contributor - contribution of heritage is accomplished via mobile application. Its welcome screen displays on a map person's location using Google Maps API. It also encourages the customer to share heritage by a Call-To-Action button. By clicking on this button the phone's camera opens and the person can take a photo with the primary camera. As we do not want the application to become confused with other social media for image sharing, which happen to be mostly selfies, we restricted users' choice to the back camera only. The person can take a picture of a space and eventually of other people in that space but not a close shot of one's own face- a selfie. Clearly this does not prevent the upload of images showing " what I had for lunch today" but such images can be considered a social heritage at some point. After taking a picture the contributor is directed to another screen where the one can add an annotation as a caption to the image and upload it to the imagescape of this space. A success message confirms the action and the upload becomes visible on the map.

Consumer - Heritage can be viewed/consumed via the web application containing an interactive map. Once the map of the world is displayed, the end-user can choose a place from a drop-down menu with project's offered places. This zooms-in the map into a given coordinate on the map where markers indicate the locations and intensity of uploads. The image and textual data for the markers is fetched from a real-time database from Firebase. Furthermore, hovering on a marker displayed a pop up with the last image uploaded from these coordinates, date of upload and eventually some caption if the person has added one to this particular upload. If there are more images on the very same coordinates a "see more" option will direct the person to a gallery. Each upload is displayed as a card with an image caption. By clicking on a card a full screen view of the image will be displayed in a new tab which allows the consumer to experience in detail the provided visual piece of heritage. Additionally, the web-client displays the current weather information(humidity and pressure) for the given coordinates from the Node Weather API.

4. SOFTWARE ARCHITECTURE and TECHNOLOGIES of IMPLEMENTATION

The System's architecture is visualized in Appendix A and it includes Mobile and Web client, Backend/Server applications, as in most of the 2.0 Web applications. The client side allows users to interact with the system through a GUI while server side applications handle data and any other third party connections.

The mobile client uses web services from Google Maps API, while the web client uses the open-source leaflet.js library and maps from Mapbox for map visualizations. Both of these client applications communicate with the server application via HTTP protocols for fetching or posting data. The server application establishes a connection with the real-time database from Firebase for handling data, while it also works as a proxy between the client and the third party API (OpenWeather API). Bootstrap CSS framework is used for styles in the web-client.

The mobile client is developed with *Expo,* which is a framework built on top of *React Native* framework. The application uses the native *phone features*: 1) camera and 2) geo-location. The mobile client uses additional web services from *Google Maps API*. ***The web client*** is developed with *HTML, CSS, Javascript, Bootstrap framework* and a map from *Mapbox*[5] displayed with the use of *leaflet.js library*[6]. The web client uses a third party service from *OpenWeather API*[7] to show the weather in a selected place. We used *Node with express.js* to build our ***Backend/Server*****,** defending routes to transfer data with both client applications to write this data to the Firebase real-time database. The server also sends query requests to an external web Service API and serves the response back to the web-client.

The choice of technologies is based on our precious experience with the *React Native framework* and its efficiency and support for cross-platform development, Android and iOS. We used *Google Maps API* for the mobile client as it can be easily integrated in React Native. This made our data storing available for both client applications all the time. *Leaflet.js library* was used as it is quite simplistic but allows fast and easy development of interactive maps, using *Mapbox* which also allowed us to customize an online map for the web client. Besides that, we used *Bootstrap framework* because it allowed us to rapidly style the HTML elements. *The OpenWeather API* was a suitable choice for this project's current stage as it is open-source and free. The weather API provides reliable weather data by coordinates which are more useful as places outside of the cities might become spaces

[5] Mapbox: Maps, geocoding, and navigation APIs & SDKs
[6] Leaflet - a JavaScript library for interactive maps
[7] Current weather data

of excess supply. *Firebase Realtime Database* is a cloud-hosted NoSQL database that allows us to store and sync data across different devices and platforms (web and mobile) in real-time. It uses collections and documents to store data and provides offline storage for Android, iOS and Web to ensure application's functionality even for moments of lost connectivity. Firebase Realtime Database's features made this web service a reasonable choice of technology to fulfil the requirements of the system architecture - the back-end could serve the newly uploaded image and text data both to the web and mobile client applications.

5. TECHNICAL IMPLEMENTATION

The Server application is represented by *index.js* which imports all the used modules, including the exported module (*firebase.js*) that establishes the connection to the Firebase Realtime Database (RTDB). We used dotenv (*.env files*) to secure the credentials required for this connection, following the best practices. The server uses express.js to create an endpoint and the routes for the client applications. We created three endpoints: 1) *get request* for the uploaded data; 2) *get request* for the weather data, taking the city name as a request query or parameter; and 3) a *post request* that uses the request body to push them into the Firebase RTDB after validating the request's certain attributes, i.e. caption, location, image. All connections were validated and the responses include both error and success messages with their corresponding status codes. The same back-end application serves data to the web client. It uses cors.js (mechanism to allow the server to load data from the Firebase RTDB) and bodyParser.js (a middleware parsing the incoming to the server requests). By using a get method it fetches data from the posts route to the RTDB as a snapshot (a picture of the data in the DB reference at the current moment). The snapshot is pushed to a data array of objects which hold all data components of a single heritage upload. Then this data is sent to the web client which requests it after loading.

The Web-client uses HTML and Vanilla Javascript to create the UI and to request data from the proxy server. *Index.html* loads scripts and styles from the leaflet.js library, using a for the map and the *main.js*It also has a navigation bar with a drop-down menu with options for the offered places- London, Gothenburg, Växjö and Kaylaka park. We used Bootstrap Framework to handle the styles of the top navigation and other elements. The Mapbox's map is initialized with hard-coded coordinates [0,0] and zoom level of 3, which allows the customer to see the whole world map. The minimum zoom level of the map is 2 and the maximum is 18. This limitation was required to assure that the tiling will not display the world map multiple times on extreme zoom out as the map uses custom tiling to display accurately the image of a location. Furthermore, the extreme zoom in results in a view which does not serve well the requirements of the project. The map is initialized by passing the L leaflet element to the map in index.html and all zoom in and out mouse interactions are supported by the library. After the tile layer from Mapbox is added to the map. *main.js* handles the HTTP get request to the server and uses markers from leaflet.js to visualize the response from the server, using the function *getPosts()* and *handelDataDisplay(response.data)*. The latter loads all objects from the data array and sets markers with image and text data content to pop ups binded to each place marker. This function uses *unifyLocations(data)*, which groups and creates arrays with the same location. For the mouse hover to open each popup, we used the *on(event, callback)* function instead of the onclick default interaction. The Markers are loaded with the size of 150 meters from the map; therefore, they are invisible when the map is loaded due to the large scale and colored based on the number of uploads per location. The check is done in *handelDataDisplay(response.data),* where more than 5 uploads are colored with RED and less are colored with BLUE. Each pop up has a link to a gallery which displayed on the top of the map (modal component), styled with Bootstrap. Additionally, the *getSelectedOptionValue()* reads the input from the drop-down men. Then, sets new coordinates and a zoom level for the map based on the selected place, using a function from leaflet *flyTo(cityName, 13)* where cityName refers to the predefined coordinates and 13 is the zoom level. Only after a place is selected the markets become distinctly displayed as their size is 150 m which requires a zoom level over 8 to be displayed significantly big.

The mobile application includes *App.js* which handles the navigation of the different components, using React Navigation stack. Three main components are used can be found in the screens folder and initializes the *MapScreen* as a default home screen. We used *React Native Elements* to style the application's native components.

1) *MapScreen.js* uses the *bridges technology* (hardware API) to get the users' location after the permission is given. Then, we stored this location in the application's temporary state, using R*eact Hooks (useContext- LocationContext.js)* in order to access this data from other components. Meanwhile, *useEffect()* is used to request the data from the server when loading the component and the response is also saved in React state by a *useState()* function. After, this saved data is rendered as *Markers* on the map using a *MapView* with the parameters for *Google maps as a Provider*. Each marker has a popup that displays the number of uploads and the latest caption from the same location- *ShowPopupInfo()*.

2) The *CameraScreen.js* asks the user for an access permission t*o* the phone's native feature *(camera)*. After that, *takePicture()* is triggered by the user and we are passing response from the async function into the

handelNavigation(data.uri, `${data.base64}`). It uses *React Navigation* to navigate to the next screen while passing this data to the *SaveScreen.js.*

3) *SaveScreen.js* imports the user's location from the context and uses react props (the passed data from the *CameraScreen.js* and the navigation from *App.js*). After filling the caption field, the user triggers the *handelCaption()*. It implements the *post request* made to the *server,* while grouping the data into a *request body object.* Success and Error responses are handled by popups with a corresponding message that comes from the server. After both cases the user is redirected to either the *CameraScreen.js (error)* or *MapScreen.js (success)*.

Project's full implementation is available at github.

6. DISCUSSION and CONCLUSION

Inherito can be considered a social media type of product with web and mobile applications. What deferes it from currently most common social media is its goal. Inherito strives to show us that we have the power to turn any space into a place of social, nature or cultural heritage by adding social value to it. Furthermore, it aims to awaken the urge to create heritage or content, not only for our own delight but with thought for the future generations. Modern social media are preoccupied with creators generating content focused on here and now. Inherito wants to shift contributors' gaze in the future and give them a tool to leave a trace for those who come after us. In that sense, comparing Inherito with other social media or interactive map web or mobile services is not that reasonable as this project's goal goes beyond showing the latest or the trendiest today.

Working on Inherito was a great learning experience, which allowed us to explore further the use of web and mobile frameworks for rapid development of digital products. It also let us deepen our knowledge in working with the APIs mentioned in the previous section. Using a real-time database was a new learning experience which we recommended to others because we found it to be easier than relational DB (MySQL). The Leaflet.js library was a very easy and well documented framework for using Mapbox and allowed us to achieve great results in geolocation data visualization, in a very short time. In the process of implementing web and mobile applications, we experience a couple of technical difficulties. For example, rewriting the server code after accidental removal; certain code snippets were not working the same way for both of us, even after accurate push and pull via Github. Displaying new HTML elements on the map initialized with a library required more effort than doing the same on the top of other code developed by us. Passing data through functions in Vanilla JavaScript gave us a headache compared to React Native but working in a team requires adjustment to each other's skill related limitations.

If this project was meant to be further developed, we would think of adding authorization and authentication features for the contributors. As they are the most vital role in this product, it will be meaningful if we can provide a detailed view of all their contributions. This suggestion is inspired by Instagram's profile grid. Additionally, we suggest a grouping function to allow them tell a story with a set of images. However, what we consider to be the most important next step is the development of an algorithm to ensure that there will be no images with violent or inappropriate content.

Working on Inherito and learning about relevant projects across Europe and even the US gave us an unique opportunity to use our talents and skill to create a service that we were interested in. Despite that it does not solve any major problem of our every-day life it was meaningful to work on it. Inherito allowed us to think beyond the current and reminded us that we are part of an constantly changing and evolving world. Inherito, also, reminded us that we have the toolkit to lay a mark on this world which is both empowering and demanding.

References

[1] Harrington, J. T. (2004). *'Being here': heritage, belonging and place making: a study of community and identity formation at Avebury (England), Magnetic Island (Australia) and Ayutthaya (Thailand)* (Doctoral dissertation, James Cook University).

[2] Giaccardi, E., & Palen, L. (2008). The social production of heritage through cross‐media interaction: making place for place‐making. *International Journal of Heritage Studies, 14(3), 281-297.*

[3] Beltrán, M. E., Prakash, A., de los Ríos, S. *COOLTURA, an open data oriented platform to enable scalable services for cultural engagement through the cloud.*
https://www.w3.org/2013/share-psi/wiki/images/2/23/COOLTURA.pdf

APPENDICES

Appendix A: System Architecture Diagram