

DESIGN AND IMPLEMENTATION OF HERITAGE LINK: AN AI-INTEGRATED MOBILE APPLICATION FOR HERITAGE EDUCATION, COMMUNITY ENGAGEMENT, AND CULTURAL PRESERVATION

Mai Tung ¹, Jason Moya ²

¹ Leysin American School, Chemin de La Source 3, 1854 Leysin,
Switzerland

² California State Polytechnic University, Pomona, CA 91768

ABSTRACT

Heritage Link is an app that supports users' desire to have a platform for discussing, uploading, and learning more about ancient heritages [1]. Designed and built entirely using Flutter for UI development and Firebase for database management, Heritage Link offers a socially impactful solution to those interested in heritages [2][3]. As ancient heritages are not mainstream and are considered niche communities, this app will allow those who were hesitant on learning more about their local heritages and other heritages throughout the entire world by utilizing the AI scan feature on Heritage Link. Using the OpenAI API allows the Heritage Link to use artificial intelligence to allow all users to upload images of any ancient heritage near them or found online so they can meaningful responses filled with meaningful information [4]. As a method to give back to the ancient heritage community, there is a dedicated page for those interested to donate heritages preservation organizations.

KEYWORDS

Cultural Heritage, Mobile Application, Artificial Intelligence, Community Engagement

1. INTRODUCTION

People were not paying a lot of attention to ancient heritage, and I wanted them to be preserved and appreciated more. There are not enough people understanding and getting to know ancient heritages other than just looking at them. Tourists struggle to resonate with the history and significance of its importance. It is also costly to preserve some heritages and prevent them from degrading over time, and my app tries to bring these topics into the spotlight. With urbanization comes the breaking of ancient heritages to expand further, so we lose these structures due to expansion and loose part of our cultural identity. With the disconnect of the significance of ancient heritage, more people are likely to desire the destruction of heritages for the sake of expanding capitalism and opening more opportunities. I do not want to see society expand and abandon their cultural identity and the way our ancestor's created art or important historical land points. Preserving these artifacts can help people remember their history and witness the development of their people to protect historical significance from being erased.

Apps are extremely accessible for a large audience that allow people to prefer reading electronic devices over countless physical books to learn more about historical heritages [5]. Heritage Link advertises different heritages that exist beyond what the user might already know or have seen, but users also have the option to learn more about their own nearby heritages by using AI to scan their image and offer valuable information. The app is unique because of its niche in grouping up people interested in heritage conservation and learning new heritage sites near them or other people have posted in the community.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. Community Content Moderation

People could have ruined the app by posting inappropriate stuff, and since my app doesn't have any restriction or confine for these behaviors. This might be an issue affecting the communities environment. Since it will most likely be a niche community, the number of users to administrate will be little enough where a singular person can manage it, and if it upscales I could implement AI to check and allow post depending on a score.

2.2. Personalized Heritage Recommendations

The support page is not customized in that there are a fixed number of supported heritages the user can check out and learn about to donate. A recommendation page used to suite user's interested depending on their location or interest from the community would offer a better user experience.

2.3. Mitigating AI Misinformation Risks

Using an AI as a important aspect of the Heritage Link could lead to nonfactual or biased information about some heritages that misinform users. The solution is to be open with the users and tell them that the AI is not always correct and presents false information.

3. SOLUTION

Upon opening the Heritage Link app, the user will see when opening the app is the splash screen. The splash screen contains a logo, app name and the current version [6]. The splash screen provides a visual feedback in notifying the user the app is launching. After the splash screen, it takes users to the login page if the user hasn't signed in. Users will have to create an account if they have not previously created an account. After creating the account, there will be information of the users' email, user name and the date created stored into the Firebase [7]. The Firebase Authentication will verify the users email to see if it is valid. Aside from email validity, the Firebase Authentication stores users password so the app managers don't get to see the users' password [8]. There is also an authenticator in splash screen allowing users to return to homepage if they had logged in. Otherwise they goes back to the login page.

Once in the homepage, there will be a user icon on the top right of the screen where the users can change their user name, logout or delete account. There are four icon buttons at the footer the first is the homepage where users are currently in, second page contains AI scanning which can identify heritages and provide detailed information of the heritage, third page is the support page with multiple cards that represents different museums where users can donate money when they

click on the donate button taking them to the official museum donation page. The last button leads to the community page allowing users to post heritages, or users can either like or post a comment under each post.

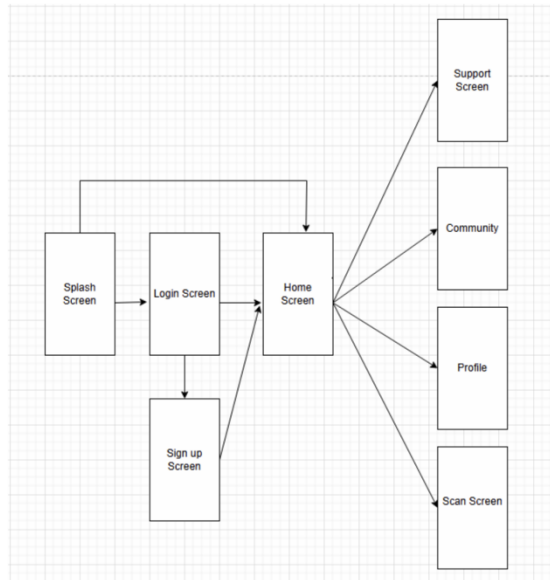


Figure 1. Overview of the solution

This is the splash screen and the routing. The splash screen is the first page that the users will see while opening the app, asynchronously displays the splash screen for 3 seconds, and after the 3 seconds waiting, it goes through routing. The splash screen uses authentication code to check if the user has logged in.

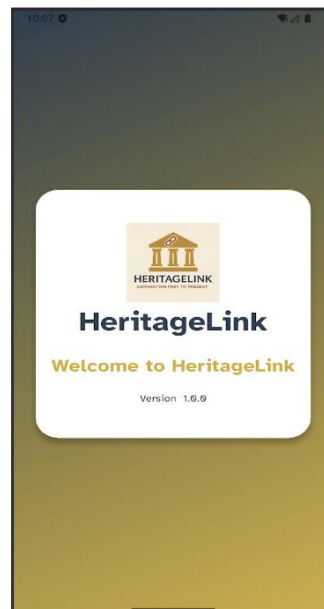


Figure 2. Screenshot of the HeritageLink

```

class _SplashScreenState extends State<SplashScreen> {
  @override
  void initState(){
    super.initState();
    init();
  }

  Future<void> init() async{
    await Future.delayed(const Duration(seconds: 3));

    Navigator.pushReplacement(
      context,
      MaterialPageRoute(
        builder: (context) => const Authgate()
      ) // MaterialPageRoute
    );
  }
}

```

Figure 3. Screenshot of code 1

InitState starts the initialization the moment they open the app, calling the function init. The await makes the counter wait for 3 seconds by using the "Future.delay" method. After waiting those 3 seconds, it uses the navigator to replace the splash screen with AuthGate. AuthGate gets information from Firebase "stream: FirebaseAuth.instance.authStateChanges()". If the connection state is waiting, it returns a loading indicator. If the app gets data from Firebase Authentication and if it is empty it goes to the mainscreen, otherwise it goes back to the login page. Once we are in our home page, a void call back is always waiting for the navigator. When we logout or delete our account, onLogout class with activate the signout function from Firebase authentication and place us back to the login page [9]. The snapshots will be loaded again when the void call back occurs and go through all the conditional statements again.

The purpose of the support pages is to let people to concern and care about other museums and heritages. This page allows users to donate money to the museum after clicking the donate button, it takes users to the official museum website donation page [10]. The museums will utilize the money for future preservations of heritages and other fundings.

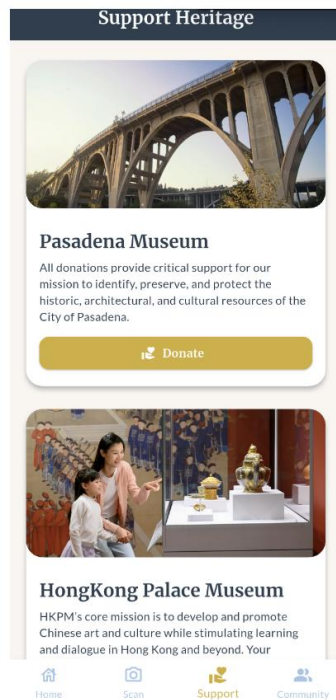


Figure 4. Screenshot of support page

```
body: ListView.separated(
  padding: EdgeInsets.all(20),
  itemCount: heritagePlaces.length,
  separatorBuilder: (_, __) => const SizedBox(height: 20),
  itemBuilder: (context, index){
    final Place = heritagePlaces[index];
    return Center(
      child: Card(
        shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(20)),
        elevation: 4,
        child: Column(
          children: [
            ClipRRect(
              borderRadius: BorderRadius.circular(20),
              child: Image.network(
                fit: BoxFit.cover,
                Place["imageUrl"],
                height: 180,
                width: double.infinity,
              ), // Image.network
            ), // ClipRRect
          ],
        ),
      ),
    );
  },
)
```

Figure 5. Screenshot of code 2

"List view.separated" creates a list with separators and the padding adds space for every Card that is created within the support page. Stream builder gets the info from the list of dictionaries, list view uses the format and the information created in the return as the blueprint. The "Place["imageUrl"]" gives a string to the Image widget displayed in the card, the place dictionary variable holds different key that have different values that are used on the card. The widget ClipRRect forces each corner of the picture to become round instead of squared for better visuals. By creating a singular Widget that the listview can use as the template, we only have to specify one card in how its layout and style is set. Beyond what is shown in the code other key values such as the card title, description, and donation link is used to build the card depending on the number of dictionaries inside of a list.

The profile page contains the users' information including their names created which is changeable and their email that is not changeable, as well as functions like logging out or deleting the account. To allow the ability to change the user's information we are using Firestore database to update. However, the users can only change their names but not their email because it refers to confidential information. On the other hand user names are not confidential because we utilize the data for user experience [11].

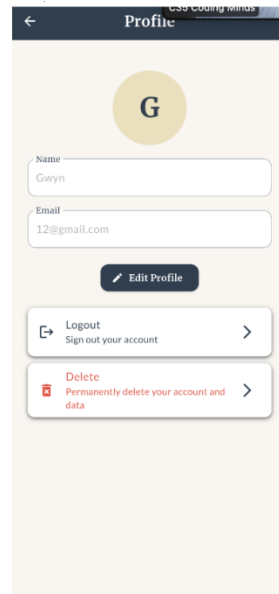


Figure 6. Screenshot of profile page

```
User?get user => FirebaseAuth.instance.currentUser;

Future<Map<String, dynamic>> _getUserData() async{
  if(user != null) {
    final doc = await FirebaseFirestore.instance
      .collection('Users')
      .doc(user!.uid)
      .get();

    return doc.data();
  }
}

Future<void> _saveProfile() async{
  if(!_formKey.currentState!.validate() || user == null) return;
  setState(() {
    _isLoading = true;
  });
  try{
    final updateData = {
      "Name":
        namecontroller.text
    };

    FirebaseFirestore.instance.collection('Users').doc(user?.uid).update(updateData);
    setState(() {
      _isLoading = false;
      _isEditing = false;
    });
  }
}
```

Figure 7. Screenshot of code 3

We are using Firebase authentication to get users unique ID and check if they are logged in. The `getUserData` function checks if the user is null because if it is null the user has not logged in and prevents further action in fetching user data [12]. When the user is valid, the `doc` variable will contain a dictionary that will await for the Firebase to retrieve from the user collection that obtains the user's specific document with all their information. Wherever the `getUserData` is used the function will return data retrieved as a dictionary. We called another functions called

`_saveProfile()` that returns nothing, but changes some booleans and update the user's data if the users are logged in or the text form fields are valid. When the variable `_isloading` equals to true, it demonstrates the circular process indicator in order to tell the users that it is loading and prevents users from spamming the same action multiple times.

4. EXPERIMENT

4.1. Experiment 1



Figure 8. Figure of experiment

The app appeals people who are a part of different heritage communities around the world with interest over structures historical and significant. This appeals to people who want to learn more about heritages they can support, get AI to help identify unknown heritage buildings, and share loved heritages with others.

This user feedback survey aims to understand the overall experience of using the app in terms of functionality, engagement, and user satisfaction. It collects insights on whether the app runs smoothly or feels clunky, and whether users find it interactive despite having only four main tabs.

Feedback on the helpfulness of each feature, particularly the AI scan function, is sought to guide future improvements. Users are also encouraged to report any bugs to support ongoing maintenance. The survey evaluates the enjoyability and user-friendliness of the community posting feature, as well as whether the support page inspires users to contribute donations and feel a stronger sense of heritage connection. Additionally, it asks whether the AI scan feature effectively provides useful information and whether the app lacks any desired functions that could make it more complete. Lastly, it explores users' perceptions of the app's audience beyond the heritage community and their sense of data security while using the platform.

5. RELATED WORK

Casual Learn: A linked data-based mobile application for learning about local Cultural Heritage made by Adolfo Ruiz-Calleja, Pablo García-Zarza, Guillermo Vega-Gorgojo, Miguel L. Bote-Lorenzo, Eduardo Gómez-Sánchez, Juan I. Asensio-Pérez, Sergio Serrano-Iglesias, and Alejandra Martínez-Monés [13]. Their app Casual Learn gives them a map to select cultural heritages nearby them with tasks for them to do and learn more about their nearby heritages. Heritage Link does not have a map; however it is for sharing between other users and Casual Learn is for more academic study rather than casual information about heritages. Heritage Link uses AI to get information about the user's heritage compared to Casual Learn getting their information from a database with documents about each heritage location.

6. CONCLUSIONS

One of the main limitations of my app project was time. I had a limited amount of time to work on it, which made it challenging to fully develop and polish every feature. Additionally, this was my first time using Flutter, a new programming language for me, so it took some time to adjust and understand its structure and workflow [14]. Tasks such as storing data, creating collections, and updating fields in Firebase Firestore were especially difficult to master within the short development period. These challenges sometimes slowed down progress and affected how efficiently I could implement new features.

If given more time, I would focus on adding new features and improving the existing ones to make the app more complete and engaging. For example, I would develop an AI-powered anti-spam system to monitor the community page and ensure content quality [15]. I would also enhance the profile page by allowing users to upload custom profile pictures. In the long term, I hope to expand the app by adding interactive elements such as a map displaying heritage locations and a "Discovery" page that recommends heritage information based on user interests. If I were to restart this project, I would aim to make the app more efficient by better understanding and organizing the Firestore database from the beginning.

REFERENCES

- [1] Cuno, James. "Who owns antiquity?: Museums and the battle over our ancient heritage." Who Owns Antiquity?. Princeton University Press, 2010.
- [2] Tashildar, Aakanksha, et al. "Application development using flutter." International Research Journal of Modernization in Engineering Technology and Science 2.8 (2020): 1262-1266.
- [3] Li, Wu-Jeng, et al. "JustIoT Internet of Things based on the Firebase real-time database." 2018 IEEE International Conference on Smart Manufacturing, Industrial & Logistics Engineering (SMILE). IEEE, 2018.
- [4] Kublik, Sandra, and Shubham Saboo. GPT-3: The ultimate guide to building NLP products with OpenAI API. Packt Publishing Ltd, 2023.

- [5] Hue, Jennifer E., Mark Rosenfield, and Gianinna Saá. "Reading from electronic devices versus hardcopy text." *Work* 47.3 (2014): 303-307.
- [6] Yilmaz, Yagiz, et al. "Investigating the impact of ransomware splash screens." *Journal of Information Security and Applications* 61 (2021): 102934.
- [7] Chougale, Pankaj, et al. "Firebase-overview and usage." *International Research Journal of Modernization in Engineering Technology and Science* 3.12 (2021): 1178-1183.
- [8] Moroney, Laurence. "Using authentication in firebase." *The Definitive Guide to Firebase: Build Android Apps on Google's Mobile Platform*. Berkeley, CA: Apress, 2017. 25-50.
- [9] Pramono, Luthfan Hadi, and Yohanes Krisna Yana Javista. "Firebase authentication cloud service for RESTful API security on employee presence system." *2021 4th International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*. IEEE, 2021.
- [10] Yates, Donna. "Museums, collectors, and value manipulation: tax fraud through donation of antiquities." *Journal of Financial Crime* 23.1 (2016): 173-186.
- [11] Sauro, Jeff, and James R. Lewis. *Quantifying the user experience: Practical statistics for user research*. Morgan Kaufmann, 2016.
- [12] Garg, Tashi. "React Query: Revolutionizing Data Fetching and Modernizing User Interfaces." *Authorea Preprints* (2025).
- [13] Ruiz-Calleja, Adolfo, et al. "Casual Learn: A linked data-based mobile application for learning about local Cultural Heritage." *Semantic Web* 14.2 (2022): 181-195.
- [14] Iverson, Kenneth E. "A programming language." *Proceedings of the May 1-3, 1962, spring joint computer conference*. 1962.
- [15] Saito, Takamichi. "Anti-spam system: another way of preventing spam." *16th International Workshop on Database and Expert Systems Applications (DEXA'05)*. IEEE Computer Society, 2005.