Executive Summary:
GlassMADA: A Memory Assistive Display for Persons with Alzheimer's

CSE 218: Mubin Wang, Weichen Liu, University of California San Diego
CSE 118: Chris Weller, Jason Tan, Xiaoran Peng, University of California San Diego

---

A. KEY PROBLEMS ADDRESSED

GlassMADA addresses and tackles the cognitive and memory issues concerning people in the earlier stages of Alzheimer's. Our solution aims to behave as a tool for the retrieval and review of forgotten information relevant to the user's personal life. Specifically, we focused on assisting the user with recognizing people and their relative affiliation as well as recording daily activities via image capture. With real-time people recognition, people with Alzheimer's can avoid the embarrassment of not knowing their families or friends or having to ask who they are. With daily activities recorded in form of photos, people with Alzheimer's and their caregivers will be able to review images in a reflective, therapeutical session about what has happened during the day.

B. DESIGN IDEAS

GlassMADA is based on Google Glass and supported by a web application. Google Glass is a perfect fit because it is portable and versatile. People with early-stage Alzheimer's are able to wear the glass for a long time without the risk of losing it. At the same time, Google Glass can perform various desired tasks: people/object recognition, photo-taking, accessing Internet, and GPS positioning. Besides Google Glass, we used a Flask RESTful server to host a web application, MADA Timeline, that allows people with Alzheimer's and their caregivers to review the photos taken throughout previous days. MADA Timeline provides a better user interface as well as reduces the computation requirement on Google Glass.

C. IMPLEMENTATION

The system implementation can be divided into two main components: the glassware and the web application.

The glassware component, developed using Java in Android Studio, is responsible for the control logic on Google Glass, people recognition, and photo-taking. People recognition is achieved by actively scanning a QR-code that has the encoded information of a certain person. We used Scandit library to fulfill this functionality. To do the photo-taking, we used the default Google Glass camera. All the photos taken will be uploaded to a RESTful server.

The web application is built with vanilla JS, jQuery and HTML/CSS for the front-end and is supported by a Flask server deployed on Heroku platform. It receives the images taken from Google Glass, decodes the data, and displays and filters the images and contextual data in a timeline format.

D. FEATURES

GlassMADA now has three major features:

1. Recognizing people via scanning QR-codes. The scanning runs continuously and text information will pop up if a QR-code is detected and read.

2. Taking photos. The user can activate the camera by voice command or clicking the touch-pad. The photo taken will be uploaded automatically.

3. Showing the photos of a day in a timeline on the web application. This can be seen as a graphic summary of the day of the user.

E. EVALUATION

GlassMADA now can read the information from QR-codes instantly and continuously, which will help people with Alzheimer's know their associations with others. The photo-taking on Google Glass and photo-displaying on the web application serve to recording the activities of the people with Alzheimer's. Voice trigger makes it easier for the user to control the glassware. As a proof of concept, GlassMADA shows that it is possible to build a memory assistive solution for people with early-stage Alzheimer's based on Google Glass and Web service.

Nevertheless, there are a few drawbacks of GlassMADA. First, Google Glass has a short battery life, takes low resolution images, and becomes hot quickly, which significantly reduces its usefulness. Secondly, scanning QR-code is a naive solution for people/object recognition.

F. FUTURE DIRECTIONS

Future work should focus on improving the usability and intelligence of the system. One idea is to make Google Glass able to retrieve more information. For example, the system can be extended to recognize objects and fetch related information. And facial recognition will be a better solution than QR-scanning to identify people. For daily activity recording, body camera can be integrated to provide steady and high quality photo-taking. Moreover, more contextual awareness can be added. The system can run tasks more intelligently based on the time, location, and so on.

GlassMADA: A Memory Assistive Display for Persons with Alzheimers

CSE 218: Mubin Wang, Weichen Liu, University of California San Diego
CSE 118: Chris Weller, Jason Tan, Xiaoran Peng, University of California San Diego

1. INTRODUCTION

GlassMADA aims to address the memory difficulties experienced by people in the early stages of Alzheimer's disease. Memory loss can be embarrassing, frustrating, and even dangerous in certain situations. Our goal is to leverage current technologies to build a system that help with the retrieval and review of lost information. GlassMADA provides following features: helping people with Alzheimer's recognizing others, taking photos as a means to record the daily activities, and displaying the photos in an interactive web application . These are intended to help early-stage Alzheimer's patients because they have memory difficulties yet still able to learn new technology.

GlassMADA is implemented with glassware on Google Glass and a supporting web application. Our design focuses on the usability and unobtrusiveness of the system. Considering the users have memory difficulties, we tried to minimize the operational demands on them. User experience is another concern of our design. The web application is used to receive, store, and arrange the photos taken during the day and then render them in a time-line, which will provide a better experience reviewing them for the people with Alzheimer's and their caregivers.

We implemented the people recognition part via QR-code scanning. A typical scenario is: people with Alzheimer's wear the Goolge Glass and whenever there is a QR-code in front of the camera, it will display the information encoded in that QR-code. Family members and care givers can print out QR-codes with their information and wear it on their person. Hence the persons with Alzheimer's will recognize them right away without needing to ask potentially embarrassing questions. The QR-code scanning can be turned on and off on the glass by the user. Photo-taking can be triggered by voice or clicks on touch-pad. All pictures taken will be uploaded to a RESTful server. Moreover, the glassware will be set as the home display on Google Glass, so whenever users activate the glass, they will be able to know how to work with it. Plus, we provide voice trigger to activate the glassware and to select menu items, considering that controlling the Google Glass with touch-pad can be confusing sometimes.

In section 2 we will talk about project motivation and background. In section 3 we describe the design of the system, including design ideas, decisions, and how the design evolves over time. We had a meaningful and inspiring interview with a person whose family member is afflicted with Alzheimers Disease, and she gave us valuable insights as to what we can do to help. In section 4, we will give the full and detailed information of the implementation. It consists of the architecture, technologies used, and main features of GlassMADA. In section 5, we will present the tests we conducted and the results. We will also talk about our evaluation of the performance and usability of the system. In section 6, we will outline the collaboration within the team and the post-mortem. Finally, in section 7, we give our conclusions and learnings. Additionally, we will list the future work that can be done to improve GlassMADA, or a memory assistive wearable device in general.

2. MOTIVATION AND BACKGROUND

Alzheimer's disease, the most common form of dementia and a progressive disease which affects memory and cognitive abilities negatively over time. As time passes, people with Alzheimers experience worse and worse dementia symptoms, starting with mild memory lapses such as forgetting familiar words and location of everyday objects to more severe, late-stage conditions such as the inability to react to surroundings, loss of movement control, and even personality changes. In most situations, people with Alzheimer's will require a caregiver to provide daily guidance. Medication, dietary supplements, and mental exercises may be used to treat and minimize the symptoms, but unfortunately, there currently does not exist a cure for this disease.[1] Ultimately, symptoms do persist and people with Alzheimers will forget important information about the people, places, and common objects in their daily lives.

People with Alzheimer's need not only alleviation solutions, but also require a method of regaining information they will inevitably lose and do so without the anxiety and embarrassment of having to constantly inquire others. A solution which could provide this would benefit the person's well-being, relationship with their caregiver, and cognitive abilities through a non-medical, simulating and therapeutic fashion at a reasonably low cost. One of multiple solutions is to retrieve what was once lost is through by reviewing photos of past daily events. In the commercial market, there are already tools that can provide partial solution to these needs. The Narrative Clip 1 is an example of wearable camera technology that can record the user's daily life passively through image capture or video recordings; the visual content may be uploaded by connecting to a desktop computer.[2] These images and video recordings are sources can be reviewed by a person with Alzheimer's and their care-taker to rediscover the past. In addition, a person with Alzheimer's require a method to relearn name and social associations to those close to them. Facial recognition software and QR apps such as BioID Face Recognition Web Service or QR Droid Code Scanner, are readily available tools and can be used to map people and objects to forgotten information relevant to a user with Alzheimer's.[3][4] Mappings can range from faces to names, faces to family affiliation, QR codes on devices to directions for operation, or QR codes on objects to procedural actions. Although there exists such helpful technology, they are used in commercial settings and thus are not being applied to the field of Alzheimer's.

Our project, Glass MADA (Memory Assistive Display for Alzheimers), combines these features and performs as an unobtrusive and bundled, external memory system with a target demographic towards people in the early stages of Alzheimers. Glass MADA is a Glass app which incorporates the functionalities of QR and body cameras which are used to recall information based on QR codes and to capture images to be reviewed later. Our solution requires a mixture of augmented reality using the prism to display real-time data and contextual awareness using location and time-based photos. The goal is to enable the retrieval and review of forgotten information to end users in a passive, comprehensive form so that an user is not overwhelmed by being required to perform any moderate amount of user input, but at the same time, is provided the crucial, detailed information they need.

3. DESIGN

Our design for the project can be broken down into two phases. The first is preliminary design, where we came up with design ideas of our own by understanding what symptoms people with Alzheimers have, what their needs are and what people can do to help them. The second phase began after we met with a family member of a person afflicted with Alzheimers disease, who gave us more insights into the daily life of a person with Alzheimer's. She provided us with very concrete and helpful problems arising in the life of a person with Alzheimers and we will describe these two phases and how our ideas evolved in the following sections.

3.1 Preliminary Design

As we describe in previous part, the most significant degradation for persons with Alzheimers is the loss of memory, especially with regards to difficulty with remembering recent events (short-term memory loss). Therefore, we broke down the subject of their memory into two parts: people and things. For people, we wanted to utilize the camera on Google Glass to perform QR code scanning and instantly display the information on the screen. For things, we wanted to design a task management system for caregivers to create task through back-end system and remind the people with Alzheimer's disease through pop ups on the display. Context-awareness is important for our target audience as it can help lower the operation burden on the wearer, so contexts like time and place can be integrated into the task reminder system. We will describe these design ideas in detail.

3.1.1 *QR Code Scanning.* A QR code is a type of matrix barcode in which information can be stored. With the built-in camera, Google Glass is capable of capturing this label and extract information from it. In order to remind user who they are interacting with, the first iteration of our product is to retrieve the name from the QR code attached to the person theyre interacting with. The user can simply scan the QR code using Google Glass and their corresponding names will be displayed on the screen. Embedding the name in the QR code is, however, only the first step. For future work, a more robust server can be built and more information about the person can be encoded and retrieved through requests from Google Glass.

Another benefit of this approach is that it can also be applied to help people with Alzheimer's disease remember how to do certain tasks. For example, caregivers can attach a QR code on a microwave oven to remind them how to use it properly. QR codes can also be attached to the bottom of pill bottles to remind them of their daily dosages of various medication. This will be further discussed in the future work section.

3.1.2 *Task Management and Reminder System.* The second part of our design explored how caregivers could help people with Alzheimer's disease. It is difficult for caregivers to be physically near the person with Alzheimer's disease at all times; fortunately a wearable device can. We want to leverage this feature of wearable devices to provide an interface for the caregiver to create tasks and automatically remind the person with Alzheimer's disease of various duties by various means utilizing context-awareness.

A control dashboard will be built for caregivers to create tasks with corresponding content, time and location. The server system will push tasks to Google Glass so it will create pop-up task display at a certain time or in a certain location. For example, caregivers could create a doctors appointment task for user ahead of time. The Google Glass will alert the users before the exact time and guide them to the place where they need to be.

3.2 Evolution

3.2.1 *Interview.* Our design evolved after we met with a family member of a person with Alzheimer's. She helped us focus on certain very real needs of Alzheimer's patients and their families and support systems and made our ideas more concrete and specific.

At first, she suggested we think about what stage of this dementia we want to focus on by informing us of the degeneration curve of Alzheimer's disease. As the impairment gets worse, people with Alzheimer's disease can find it increasingly difficult to learn new things and, due to frustration, may entirely refuse to wear assistive devices like the Google Glass and other wearables all together. When asked about the QR code idea, she gave us positive feedback and stressed that this function can be very

helpful. We were concerned providing the user with a name alone would not be sophisticated enough to make a positive, and she completely eliminated this doubt.

The most important part that helped us evolve our design was the suggestion that recording daily activity for them can be very beneficial. For one thing, they can use the recording to remind themselves what happened throughout each and every day, instead of feeling worried or frustrated due to their symptom of short-term memory loss. Additionally, caregivers will be able to track the users activity so as to be reassured and prevent potential catastrophe.

3.2.2 *Challenges.* First, any operation with the Glass would be a burden to people with Alzheimer's disease and we, as developers, should make efforts to ensure the application runs smoothly without any intervention. This could be a challenge to us as the Glass itself requires a lot of interaction to fully function.

The second challenge we need to face is the hardware limitations of Google Glass. The common phenomenon of overheating can cause the people with Alzheimer's disease feel unwilling to wear it. Moreover, current battery endurance is also impossible to support all-day functioning. As an alternative, we may use other body cameras to solve this problem. However, due to time limitation, we have decided to focus our project on Google Glass and leave this exploration in future work.

3.2.3 *Pivot.* After the Interview, we realized that our seemingly good idea turned out to be somewhat unfocused. The task management and reminder system tend to lay too much burden on the users because it's difficult to maintain Google Glasss workflow without manual intervention. Unobtrusiveness should be our priority when designing the application for people with Alzheimer's disease.

3.3 Final Design

In this part, we'll illustrate our final design of our project. At first, we'll explain our design rationale at a top-level. Then, we will introduce additional features we want to add to our system (i.e. Record and Review) and the whole workflow of our application.

3.3.1 *Rationale.* Unobtrusiveness is our first concern. It's very difficult for people with Alzheimer's disease to learn new skills due to memory problems, so we should eliminate the potential operations performed on the Google Glass and adopt passive interaction. We should also consider how to handle unforeseen exceptions that may occur, whose solutions are outside of the users potential capabilities of interaction.

3.3.2 *Record and Review.* As we previously discuss, recording what happened during the day and providing an interface for review can be very helpful both for people with Alzheimer's disease and caregivers. To better record what happened during the day, we will leverage contextual information and the built-in camera in Google Glass. As for contextual information, each picture will be combined with time and location so that exactly what happened can be understood in review.

3.3.3 *MADA Timeline.* MADA timeline is our interface for review. We set up a back-end server and display the information interactively on a timeline. With information visualization, users find it much easier to browse all the activities happened in each day. And to further meet users' needs, we added filters so that they can filter the activities according to person, time and location.

3.3.4 *Work-flow.* QR code scanning will remain as one of our important features. We will combine the two features to optimize our work-flow. In other words, we will trigger QR code scanning and context capturing at a certain interval and upload the information to our server. If there is any QR code in the view, Google Glass will also display the information on the screen to remind the wearer in

Table I. Comparison between two phases of design

	Preliminary Design	Final Design
Feature1	QR code for people recognition	
Feature2	Task Reminder	Record and Review with MADA Timeline

real-time. Context information includes time, location and picture taken. The comparison of our two phases of design can be seen in Table I.

4. SYSTEM DEVELOPMENT

4.1 Architecture

GlassMADA as a system consists of three major components: the GlassMADA glassware, a RESTful server, and a web application MADA Timeline.

GlassMADA glassware is what the person with Alzheimer's mostly interacts with. The glassware fulfills most information retrieval and recording functions. It does people recognition via QR-code scanning and display the result immediately. And the glassware allows photo-taking. Theses pictures will serve for future reviews of the user's daily activities. Moreover, the glassware is designed as a Live-Card, which replaces the default clock card of Google Glass. Thus the user will notice the instructions once they turn on the glass. All operations on GlassMADA can be triggered by voice or touch-pad.

RESTful Server can be seen as a middleware between GlassMADA and MADA Timeline. It handles the communication between other two components. For example, once photos are taken on Google Glass, they will be uploaded to the RESTful Server. Then MADA Timeline can fetch these photos for display issuing requests to the RESTful Server. Google Glass has Mirror APIs, which makes it work seamlessly with a RESTful server.

MADA Timeline is a web application for reviewing the photos by Google Glass during the day. It provides graphic and user-friendly display. Also it supports filtering photos based on person, time, location. Besides, it is integrated with Google Maps APIs, so users can view on map where a photo is taken.

This architecture enables parallel and incremental developing. GlassMADA, RESTful server, and MADA Timeline have well-defined responsibilities and interfaces. They communicate with each other via standard HTTP requests. Since the functionality are well-defined, our team can work simultaneously on three parts and make decisions or modifications without interfering other parts. The system integration is also easier, because the information passed by HTTP requests are understood and agreed by every team member, we can easily test the communication. Further, we separate the communication part of each component, thus we can achieve modularity developing each individual part.

The biggest advantage of this architecture is that it makes extension much easier in the future. For example, if we want to achieve face recognition, all we need to work on is the glassware. Besides, the glassware development is well structured, thus we can add features without breaking the integrity of GlassMADA.

4.2 Technology Used

4.2.1 *GlassMADA Client.* Google Glasss operating system is Glass OS which is based on Android. It uses Android API 19 KitKat. Most of Androids libraries such as Scandit and Zxing can be directly applied to our new application, but the Glass has its unique features which differentiate it from the traditional touch-based interfaces present in most modern Android devices (namely smartphones), so we can not directly apply some design methods. [5; 6]

We used Android Studio for glassware development. The code is mainly written in Java and XML.

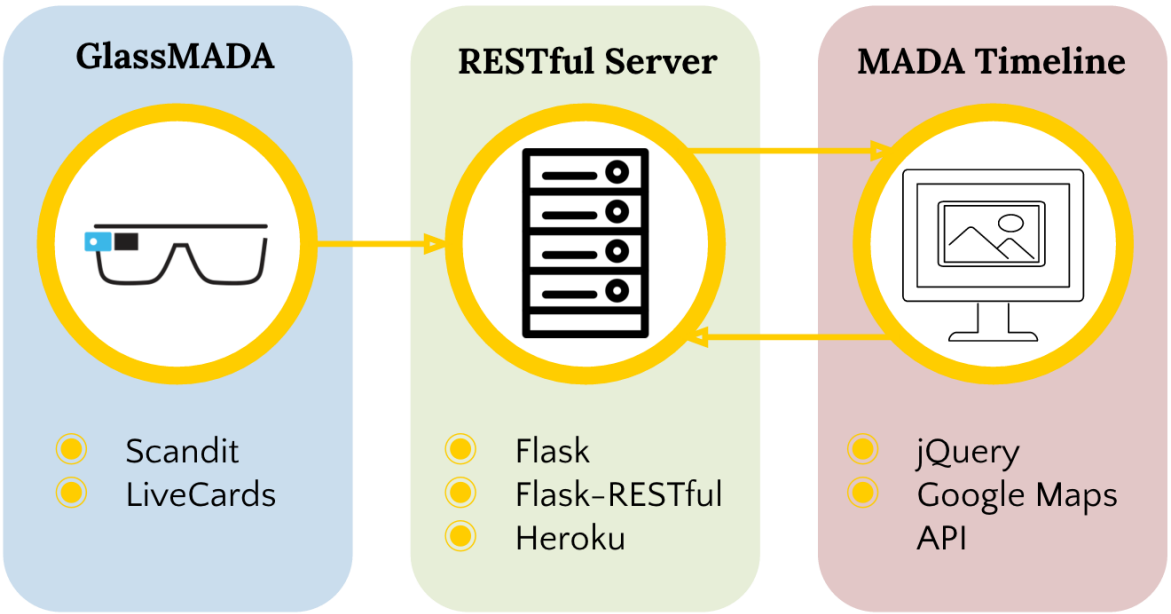

Fig. 1. The architecture of GlassMADA

For QR-code scanning, we used the Scandit[7] library. Scandit provides the highest quality in mobile barcode scanning solutions for smartphones, tablets and wearable devices. It has high performance speed and accuracy in a variety of challenging conditions, including low light and shadows,glare and metal reflections,skewed angles, blurry and damaged barcodes, small barcodes, barcodes in motion, and multiple barcodes. The latest SDK in particular added support for Google Glass, which makes it a perfect fit for our application. We chose the Community SDK for Pre-Commercial Use.

For photo-taking, we used the default Google Glass camera[8]. In the glassware, we use the camera application to capture photos and fetch the image via passing Intent. The feature is implemented in Java and uses Android libraries.

4.2.2 *RESTful Web Server.* On the server-side, a RESTful webserver was determined to be a necessity, functioning as an intermediary capable of relaying information from the Google Glass GlassMADA application to the MADA Timeline Web Client over the internet. After much deliberation, Python was selected for its ease of use, relative familiarity, support for rapid prototyping and iteration by means of developer friendly features such as a REPL and duck typing, and its multitude of web frameworks. For a longer term project, NodeJS may have been a better candidate for its inherent asynchronicity and coding paradigms, but in our case for a rapid prototype, the longer initial time to gain development velocity would prove far too difficult and outweigh the potential benefits of faster response times or other features of a web-first language. In the end, Python proved itself to be more than capable for the limited requirement list dictated by our intended feature set.

After surveying the bountiful landscape of web technology frameworks and comparing and contrasting the various features each had to offer, Flask was ultimately selected due to its lightweight nature and plentiful documentation. As a microframework, Flask provides many useful features such as unit test support and an integrated web server, lowering the barrier of entry required to get a full-fledged web server up and running with little-to-no boilerplate code and minimal file structuring. [9] Layered

on top of Flask, Flask-RESTful provides an even more streamlined interface to building RESTful webservices and APIs on top of the already accessible built-in RESTful utilities, making for a low-impact development solution. [10] This applications structure was heavily based on work by Miguel Grinberg and his ongoing series concerning Python webservices built with Flask. [11]

Late into development gunicorn was added as a WSGI Python webserver. [12] While this allowed for some interesting local testing scenarios, its inclusion was due primarily to Herokus requirement list; because Heroku was chosen for its ease of setup and free development plan modeling, we were required by Heroku to add a few boilerplate necessities to get things functional - gunicorn being the primary of which. [13]

4.2.3 *MADA Timeline Web Client.* The MADA Timeline is based off Blueprints' Vertical Timeline and is built upon HTML5/CSS for the structure and style of the web application which includes the listing components, smooth itions of the page, and the animations of buttons.[14] In addition, the Timeline uses vanilla JavaScript to perform XMLHTTP requests to the Flask RESTful server to query for newly uploaded images. JavaScript provides the functionality of decoding pulled JSON data (retrieved through a GET HTTP request), creating local object representations, populating the page with image information, and filtering this information upon user request dynamically. Since the Timeline contains a local list of the server image data, the Timeline is very independent from the server when executing filters and only requires a network and connection to the server upon page loads and database updates.

4.2.4 *Google Maps APIs.* We use Google Maps Geocoding API to convert geographic coordinates (a pair of latitude and longitude) received from the Glass to human-readable addresses. Google Maps Geocoding API supports two kinds of response format: JavaScript Object Notation (JSON) and XML. For our convenience, we use JSON format and make request using jQuery's getJSON function, which includes a built-in JSON parser.

4.2.5 *Glassware.* Google Glass is an optical head-mounted display which is worn like a just like a pair of eyeglasses. Considering the hardware, it has a CPU which enables it to deal with light programs, a camera to capture nearby scenes and a display right in front of the right eye. Besides, it has the ability to have communication to World Wide Web via Wi-Fi or Bluetooth. With all those features, glass could run our program on it, take photos, scan QR code, upload and fetch data from the server all on its own. More importantly, the glass is always within users sight with its unique wearable feature. As a member of Android family, the glass takes advantage of many of existing libraries to make development easier.

The glass still has some limitations on both hardware and software. In the following paragraph, we would discuss the relevant technology and limitations.

4.3 Features

4.3.1 *Unobtrusive UI.* As an entirely different hardware to other Android devices, Google glass has unique features that provide users with a unique experience.

Firstly, the display in front of the right eye functions as a digital overlay to our real world which creates an augmented reality experience for its users. Glass offers several kinds of user interface paradigms through which it is capable of providing users with the desired information. The glass uses immersion views and live cards to present users with information that are suitable for different situations.

Live cards appear in the present section of the timeline which lies in the left of Home card and display real-time information[6]. Live cards are suitable for situations which need data flow without interrupts. So we would launch our live card as our main working space, and this live card would stay

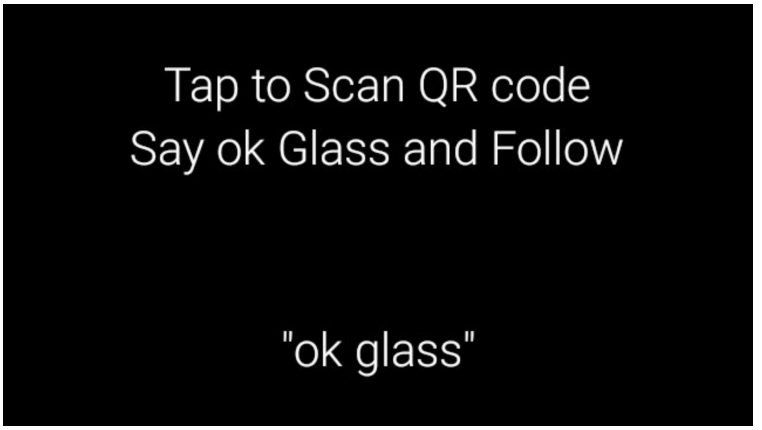

Fig. 2. Example of our Live card view.

at the left side of Home card as our running application. Whenever glass wakes up, users will see the last live card they have. It would make our glassware easier to use because users dont need to start the application every time their devices are woken up and prevents the need to frequently operate the glass to switch to our application. However, live cards are rendered with android service which cant do CPU intensive work like QR code scanning and photo taking, so for that we need another method to achieve it.

Immersion is an individual part outside timeline, giving us more freedom to control over the user experience[6]. We can use standard Android solutions like layouts to organize user input, present information to users, and place tasks requiring CPU resources. Those features make it suitable for us to scan QR code or take photos, and most of our work would be done here. To achieve our goal, we build immersions with standard Android activities, layouts to do QR scanning work and photo taking work. Once the works finished, users would go back to the live card to have a persistent experience.

4.3.2 *Voice Control.* Unlike normal Android devices with touch screens, the Glass can only be operated manually through its touch pad. Users need to operate the touchpad several times to carry out a usual action on device with a touchscreen which seem to be burden of users. However, voice input gives users and developers another way that are totally hands-free to interact with the device.

In our project, we utilize voice input to start GlassMADA and execute actions within activities and services. Tapping the touchpad to wake up Glass, users can use voice command in the home card to start our application GlassMADA if the program is not running. Users need to say ok glass to see list of commands which involve all the supported applications. We can see our command hello tofu chowmein in the list. Users would say our command to start the application.

We also use Contextual voice commands inside our applications to avoid touchpad operations. We build Contextual voice commands in both live cards and immersions to perform actions instead of hands. In our application, actions are mostly switches between different cards. So, voice commands in live card would switch current view to immersion view, voice commands in immersion view help users to go back to live card.

In case of some errors happening on device, the same action can be achieved through both touchpad and voice input, for instance, scanning QR code can be done with voice command ok glass than scan QR code or some taps over touchpad in live cards.

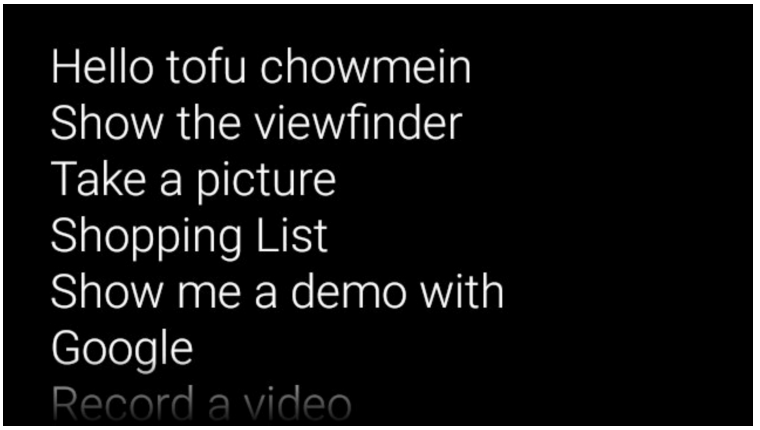

Fig. 3. Voice Command List in Home Card

4.3.3 *People Recognition.* One of our major goals is to achieve people recognition via Google Glass. Considering the time limit placed on this project, we decided to employ QR-scanning as the main approach to do the work. Family members and caregivers will wear QR-codes that encoded their personal information, such as their names and associations to the people with Alzheimer's. When a QR-code is captured by the Google Glass camera, a piece of decoded text information will be shown instantly on the display so the user is capable of recognizing faces without the need for further inquiry. We believe this saves the user from much trouble and embarrassment.

QR-code scanning is power consuming and tends to heat up the Glass. In order to combat this, we made QR-code scanning a feature that can be turned on and off by the user. This feature is folded into the application menu, which can be switched via voice or touch-pad on Goolge Glass.

When the QR-code scanning process is activated, a new immersion Activity will run on the Google Glass. The camera will be turned on, and the screen will display the real-time video stream. If there is a QR-code in front of the camera and detected, the glassware will decode the information and show it on the screen in form of a piece of text. The text will last for a short period of time and then disappear. And user can use the glassware to read next QR-code. The detecting and decoding happens continuously and passively, hence there is no user operation needed. The QR-code can be read in a relatively wide range of distances and angles.

The QR-code can be generated using open and free tools. All family members and caregivers need to do is to encode their information, print the QR-code, and attach the codes to them whenever it is necessary.

Moreover, the QR-code scanning is suitable for almost every possible scenario: family setting, social gathering, hospital, etc.

4.3.4 *Photo Taking.* One of the most desirable features from people with Alzheimer's is recording their activities throughout the day, so that they can store and review such information. In other words, this can complement their memories. A great way to record the day is by taking photos. We took advantage of Google Glass camera to achieve this feature.

We implemented the photo-taking by leveraging the default Google Glass camera. The photo-taking feature is folded in the menu of the application. It can be triggered by voice or touch-pad. Whenever the user find something interesting or worth of remembering, he/she can take the photo right away. After the photo is taken, he/she can decide to accept or discard the photo. To reduce the operational

need, we added the logic that will automatically upload the photo accepted to the server and store it for future review, thus the user does not have to do this.

Due to the privacy concerns, the default camera application enforces that the user has to ”tap to accept” once the picture is taken. This shows the respect towards the user’s and others’ privacy.

4.3.5 *MADA Timeline.* The MADA Timeline is the user interface for reviewing photos that have been taken by GlassMADA. MADA Timeline was built in modules where each module (one per functionality) was implemented, fully unit tested, then integrated into the Timeline incrementally. Modules include: retrieval of data from RESTful server, decoding of data (includes submodules: decoding base64 encoded image, decoding coordinates to address, formatting UTC time), populating graphical list, filtering local object list, and manual uploading of files. Using the first module, the web application can pull this data from our server sending an asynchronous XMLHTTP request with appropriate headers upon page load. The JSON format of this data has been established between the MADA Timeline and RESTful server through an API. After successfully fetching the JSON data, MADA Timeline will parse and decode the data, subsequently creating JavaScript objects and storing them locally. During this decoding process, the Timeline converts UTC time into a human-readable format, decode base64 encoded strings back to an image source, and produces a human-readable address from geographic coordinates using Google Maps Geocoding API.

Since GlassMADA sends useful, extra contextual information accompanying each photo to our server, the web application will then fill out the personal timeline with item blocks, creating a list populated with photos with the date, the time, and the location of where the photo was taken along with the possibility of a person the user met in the photo detected through QR scanning. A typical item block displayed on the page contains the following about the photo: the time (in 12hr format with AM/PM), the day of the week, the date (in Month Day, Year format), a Google Marker which opens another tab and links to Google Maps with a URL query selection to the latitude and longitude coordinates of the location, the name of the person met (if QR scanned), the address of the location, and last, but not least, the photo sent by GlassMADA.

Furthermore, the population of the Timeline can be filtered using a dropdown menu through three categories: person, location, and time. If the person or location category is selected, an input search text-box is displayed to filter the list by person name or address. If the time category is selected, three additional drop-down menus are displayed for the user to filter the month, day, and year. As mentioned, we opted to perform local storage as performance optimization in favor of time and networking costs as opposed to memory costs for this filtering feature. By using local objects, we can execute query and filtering operations very quickly on client-side without relying on potentially unstable communications with the server over a connection.

If the user desires to view images that were not strictly uploaded via GlassMADA, he/she may use the Gallery Preview upload feature to manually upload photos from any device (e.g. smart phone or computer) used to view MADA Timeline The gallery displays scaled-down photos in a grid-like format, allowing for a broad overview of a multitude of photos. Upon mouse hover, a photo is enlarged slightly to enable the user to see the photo more clearly; right-clicking and selecting ”View Image” will open a new tab to show the full-sized photo.

With these features, MADA Timeline’s overall interactive graphical display provides the user with Alzheimer’s and his/her caretaker with image data that they can use to backtrack key events during their day or previous days. In addition, the caretaker may use the Timeline as a centralized tool to monitor the person who they are looking after in real-time (e.g. where is the person if he/she is lost, what is the tasks are the person doing or digressing from, etc).

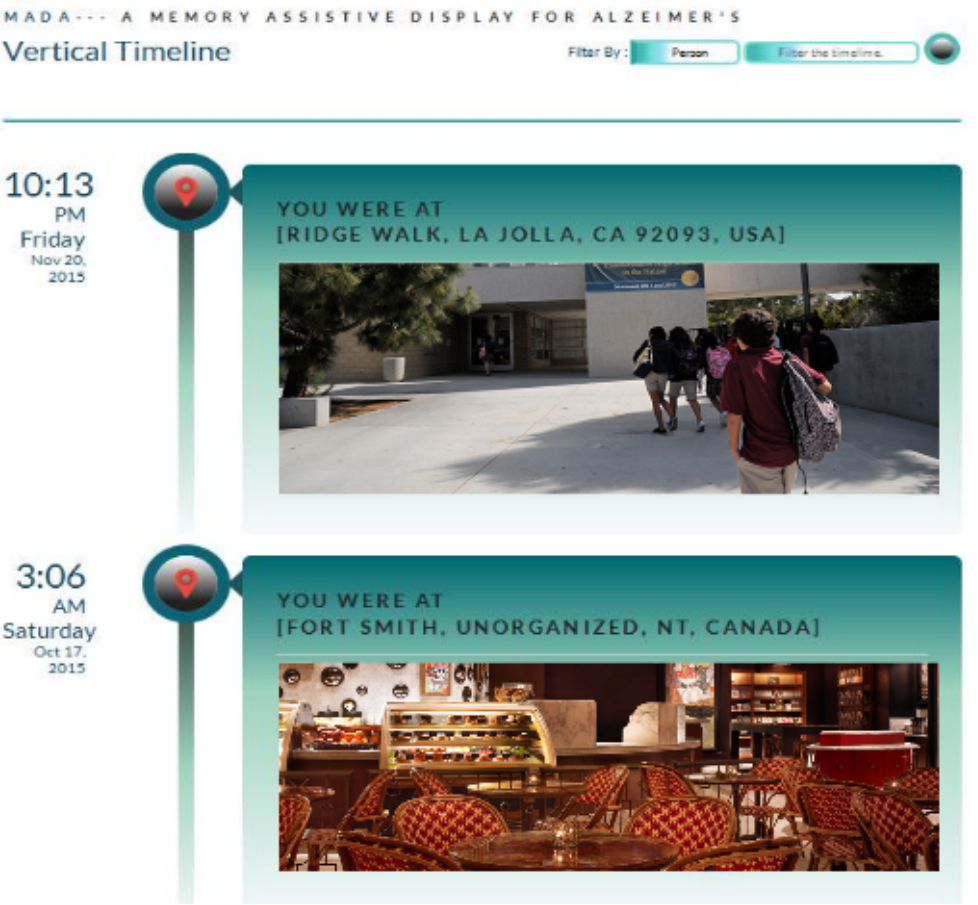

Fig. 4. Example of the MADA Timeline. Two item blocks are displayed. On the left-side of each block, the time context is shown for the photo. On the right of that is a reference Google Marker to the coordinates context. On the right-hand side of each block shows the name of the location and actual image taken. On the top left, the filter menu is available for use.

5. TESTING AND EVALUATION

5.1 QR-code scanning

We have conducted a series of tests on the robustness and usefulness of the QR-code scanning feature, including:

(1) Power consumption
(2) Heating problem
(3) Working distance
(4) Working situations

The tests were conducted indoors. We found that if the Google Glass is fully charged and the QR-code scanning process runs continuously, the battery life can last about 80 minutes. The glass tends to overheat after 10 minutes use of QR-code scanning. In terms of usability, GlassMADA has a satisfactory performance. When the QR-code is about the size of 10cm * 10cm, it can be detected within 1

Fig. 5. The left is the screen shot of QR-code scanning process running. When a QR-code is detected, a text will pop up showing the encoded information, as shown in the right.

meter range. If the QR-code gets blown up to 30cm * 30cm, it can be detected in a 3-4 meters range. So generally speaking, the bigger the QR-code, the better scanning result. We also have tested QR-code scanning from various angles, it turns out that GlassMADA can handle 80% of cases. Moreover, the QR-scanning feature works in dark situations, and it never crashed during out tests.

In summation, the QR-scanning result is satisfactory and can suit daily needs. However our primary concerns are with battery life and overheating. This is more of a hardware problem than a software implementation problem. The hope is that future Google Glass or other wearable device can tackle this.

5.2 GlassMADA UI Testing

To test user interaction with GlassMADA, the tests were carried out with persons without Alzheimers to gauge our applications feasibility. We asked two users without Alzheimers who have no experience with Glass technology to try our application in attempt to evaluate and verify our applications usability concerns. All the tests were conducted in an indoor environment for a a period of time no longer than one hour. We have gathered the users feedbacks about their experience with glassware and our app in a Q&A format.

Question 1: How much effort it take to use GlassMADA and Google Glass in general? Answer: The instructions are really helpful to me. I used the voice command to switch to QR scanning phase. The voice command is sensitive and accurate, but the touchpad is not so easy to use. I dont know how to use [Google Glass] and I got lost trying to figure out how to get back into the application.

Question 2: What did you think were the best parts of our work? Answer: I think the instructions helped me go through all the parts are the best. I cant do it without them.

Question 3: What do you think were the worst experiences? Answer: Sometimes I tried to wake up Glass with the touchpad and did a wrong swipe on it. Then I went somewhere else and couldnt go back. Thats really terrible.

In general, our application is favored by users, but the limitation of Glass itself make it hard to use. Furthermore, the response to Question 2 reveals to us that users need more help to operate a new device like Google Glass. Considering that Glass and wearable displays are a relatively new technology which may be hard to learn, it may be a matter of time before people to acclimate and learn to devices such as Glass.

5.3 Google Glass

Aside from the generally positive features we leveraged to build our application, there are numerous limitations in Google Glass, many of which we faced over the course of development.

Although we tried to sandbox our users and lock them within our application, they still managed to switch to other irrelevant cards, interfaces and menus by mistake or curiosity. Whats more, Google Glass doesnt provide users with enough instructions to get started.

Additionally, voice input also has its limitations. It works fine with a few candidate commands, but it appears difficult to pick up the right one when there are a lot of candidates. In our application, we minimized the number to avoid this problem.

5.4 MADA Timeline

Due to how MADA Timeline was developed in an incremental process, each feature was tested thoroughly and was only integrated into the web application after satisfactory output. However, there are still a few limitations to the Timeline. Firstly, the Timeline heavily relies on well-formatted data from GlassMADA. Any string data that does not match according to the API between MADA Timeline, server, and GlassMADA will not be decoded properly, resulting in either inconsistent information or no information displayed at all on the interface. Second, there is a resolution and image format restriction in order for an image to be encoded and decoded using base64 in our system; a typical image should be in .png format and should be 2MP or less in size. Third, the MADA Timeline's performance in terms of execution time is dependent on a adequate connection with the server. Although the Timeline can produce the initial list in under 0.5s in normal conditions, any slow connection will hinder the display speed of list items on the Timeline. We try to mask any delay under 0.5s with a fade-in animation for each image pulled. Additionally, we have not tested the Timeline's execution time for pulling a day's worth of images (over 100 at bare minimum) captured by GlassMADA as the app is intended to do. It should be expected that as the number of images increases, the display speed will also increase in linear time. Nonetheless, after the initial GET request from the server, the Timeline performs independently (filtering and gallery functions) using its locally stored objects. Reflecting upon the general design and core features we wanted behind the Timeline, we believe we have executed the implementation of this tool according to what was required. The Timeline is an graceful output mechanism for reviewing photos taken by a user with Alzheimer's.

5.5 Overall Evaluation

We have described our goals and risks in previous sections and now we'll review and describe how successful we were in accomplishing them. For our high-level goal of retrieval and review of lost information, we designed our GlassMADA application with two main features. The first is QR-code scanning which will be helpful to users by reminding them of who they are currently meeting with; QR code information will be retrieved through Google Glass built-in camera. The second feature is to record and review the activities during the day. We will leverage context-awareness technology to retrieve context information such as time and location together of pictures taken. As for review, we have built MADA Timeline with graphic interface and data visualization so that users can better review the lost information. We also enhanced the usability of the interface by adding features like filters so that interested information can be easily located.

As for the four risks, we brought up our own ways to solve and we'll briefly summarize as follows.

Satisfaction of Needs.. In our initial phase of design, we mainly gathered information about the symptoms and needs of people with Alzheimer's through the Internet. We were worried that those information may not be effective for us to figure out what their true needs are. So later, we set up

an interview with a person whose family member has Alzheimer's and received very helpful and concrete feedback to revise initial design. We redesigned our application and came up with the new features of recording and review as well as MADA Timeline. With those, we are now confident that our application will meet the true needs of them.

Android / Glass experience.. Lack of Android and Glass development experience was also a risk for us initially but we successfully develop our application with the help of Google Developers Guide and team collaboration.

Integration of QR framework.. Since we don't have enough time and we don't think it's necessary to develop the QR-scan framework from scratch, a huge part of development was to integrate a current working library into our application. Unfortunately, we spent a lot of time researching and attempting to integrate a library named *Zebra Crossing*, but it was not operational. Later, we switched to another library called *Scandit*. We found it to be fully functioning as we thought and integrated it into our application successfully.

Glass Hardware.. Google Glass suffered from the limitations of battery life and the phenomena of overheating. Although we didn't directly solve the problem due to shortage of time, we proposed the usage of alternative devices such as a body camera that would help solve this problem. We will illustrate this more in the future work.

6. COLLABORATION

Over the course of the development phase, our team held weekly meetings at a minimum for designing, development work, and progress updates. For each meeting, we wrote meeting memos (mostly in the form of minutes) about what we needed to accomplish during that meeting, what we discussed, what problems have surfaced (whether it be a technical or people issue), how we tackled these problems, and what we planned for the upcoming week(s) including what features should be completed by what deadline, when is the next meeting, and who has been allocated what task. These memos were posted online and shared using Google Docs at the end of each meeting as documentation so that we could review them individually as a method of diffusing unnecessary ambiguity or concerns. If any questions or explicit clarification was required, we used Facebook Messenger to communicate outside of meetings. Although communication through this channel was messy and unorganized at certain moments during busy development, we would be able to discuss and resolve any assumptions during physical meetings. As the project evolved into new stages, we collaborated in more specific ways in addition to all aforementioned actions.

During the design and architecture phase, at weekly meetings, we came together as one group to brainstorm possible project problems, assess pros and cons of project solutions, and reasoned about risks such as time constraints and feasibility concerning potential features. Initially, we found it extremely difficult to brainstorm a well thought-out idea (a specific problem, reasons for why it needs to be solved, a solution, etc) for our project that everyone would be satisfied and passion with, but ultimately, after two days of discussion we agreed on GlassMADA. After finalizing a concrete proposal, we started to outline the architecture of our project based on the features in a systematic manner to compartmentalize each section (GlassMADA, RESTful server, MADA Timeline). This defined specific endpoints of our overall system; each endpoint would be an interface to connect two moving parts. For example, GlassMADA and our RESTful server would be implemented independently and by abstracting away the details inside each module, they interconnected simply through a definitive API.

During the development phase of the project, our team structure was based on our projects architecture design. We split our team into three teams accordingly. On the GlassMADA team, Mubin and Xiaoran were responsible for building an Glass app integrated with Scandit QR framework for people

Fig. 6. Our team structure.

recognition and active/passive photo taking for image collection. On the MADA Timeline team, Jason and Weichen were in charge of designing and creating an interactive and graphical web application to display images and contextual data. On server-side, Chris implemented a Flask RESTful server, devised a concrete API and schema used for storing the data image, provided the CRUD functionalities for data transfer, and deployed the app on Heroku cloud. Through this division of labor, we were able to develop at a greater velocity by working in parallel on three separate components at once. Furthermore, since each component had to comply to a specific API, integrating each part was a trivial task.

During the last phase where we collaborated to finish our final presentation and report we first set up a meeting to brainstorm our structure of the powerpoint slides and included corresponding content for each page. As for the presentation, we presented details mainly according to the rubric and took different roles in the discussion. Mubin was responsible for leading the discussion and Jason took notes on the Google Docs while the rest of us brought up our ideas. Then, we worked collaboratively on the Google Slides, decided who was going to present which slides, and met again later to practice to ensure our presentation would meet time constraints and had a smooth flow. As for the final report, we broke down the report into several sections and each one of us chose one part to finish. We met together and combined each separate part. Here is the breakdown of who did what section of the report:

Mubin wrote the Executive Summary, Introduction, System Architecture, a portion of Technology Used (Scandit and custom Glass camera software), a portion of Features (QR scanning and photo-taking), and a portion of Testing and Evaluation (QR scanning). Jason wrote the Motivation and Background, a portion of the Technology Used (JS, jQuery, HTML/CSS), a portion of Features (MADA Timelime), a portion of Testing and Evaluation (MADA Timeline), and Conclusion and Future Work. Weichan wrote the Design, a portion of Technology Used (GoogleMaps Geocode API), and a section of Collaboration. Chris wrote a portion of Technology Used (Flask RESTful server and Heroku). Xiaoran wrote a portion of Technology Used (Glassware development), a portion of Features (LiveCard immersion display and voice-triggered command), a portion of Testing and Evaluation (user testing), and a portion of Collaboration.

Since several of us were not native English speakers and Chris did not have many sections in the final report, Chris (with the help of Jason) proofread the whole report after compilation and made sure every section is cohesive and understandable. For this phase, we worked very well together by splitting

up tasks as we normally did. The major issue that we came across was the deadline to finish the report. Because the report was due at the end of the quarter when the group was busiest, some members had problems finishing their part when it was necessary to meet the time constraint. By helping each other fill in missing sections of the report, we managed to overcome this problem in the end.

In general, we believe our team dynamic was very solid in terms of division of labor, structure, and communications. At most moments, we knew what we were doing, when was the due date for our part, and what others were doing in case a concern had surfaced. Although our structure seemed rigid, it was still flexible to a certain degree where we could assist another team when they needed it whether it be research work or development work. If we had the opportunity to go back and do something differently, we would have hoped to discuss and communicate with our liaison earlier to get in contact the interviewee whose family member is afflicted with Alzheimers as this meeting was the turning point in our whole project. We confirmed real needs of our targeted group and thus, we were able to concretely decide which features were required. If we had known earlier, we would have been able to complete all desired features and added more helpful functionalities as well.

7. CONCLUSION AND FUTURE WORK

7.1 Conclusion

Our team has created an memory assistive system with GlassMADA based upon a Google Glass application for imaging input, Flask RESTful server for storing images and contextual JSON data, and MADA Timeline for filtered, graphical display output; this system provides people with Alzheimer's with the option to retrieve and review of forgotten information in an ubiquitous-directed manner. Still, there is much to improve and polish for a well-rounded system.

7.2 Future Work

First of all, on the wearable component of our system, GlassMADA still requires automated and better image capturing functionalities deemed necessary for a truly unobtrusive, smart application. To improve the feature of people recognition, we believe in designing an implementation of QR scanning or facial recognition through an API which smartly determines when to actively scan based on context. By doing so, we can prolong the time before the occurrence of Glass' hardware issues such as overheating and short battery life. Furthermore, delegation of the automated image capturing task to a body camera would lower the computational stress on Google Glass. Also, we want the application to record and send contextual information (location, time, and names of people) along with the photos. Without this critical information, the experience of reviewing of these photos on MADA Timeline would be similar to a stranger looking at irrelevant images. On top of that, we would like to add more context such as voice recognition, video recording, etc to create a more extensive data profile per image or video which can be reviewed on the Timeline. Secondly, the Flask server, acting as the middle-man, is missing two key database requirements: user accounts and quality checking. Currently, the Flask server contains one list of aggregated image data which is not specific to any user. Due to time-constraints, we have not scaled the project to allow for multiple users. Akin to any web application, GlassMADA requires user accounts which stores only the personal list of images taken by the specific user. GlassMADA would send image data to the server for a specific account and MADA Timeline would receive image data from the server for the account with corresponding login credentials. Of course, the image data may be corrupted, ill-formatted, or unacceptable to be displayed on the Timeline. The server should include verification of incoming data which, if evaluated to be acceptable, stored in the database. Examples of data issues would be: too large of an image size, geographical coordinates do not exist, image is in an incompatible format. For images, if verification fails, the image may be manipulated through

cropping, format conversion, and/or re-encoding from base64 to another representation as a recovering mechanism to correct the data instead of disposing it. By implementing verification at this endpoint, photo taking on the user side may be done on any image capturing device (such as switching to a body camera if so desired) as long as it provides the correct data that our server expects. Lastly, with the above developments, the Timeline must successfully decode, display, and filter any extra contextual data in a meaningful structure on the web application. We would also like to support interaction between the timeline and gallery features by extracting contextual information from uploaded images and appending corresponding item blocks for these images to the timeline itself, creating a more cohesive web application by involving, what is currently, two distinct units into one primary timeline feature. As a final feature to balance the time and geospatial foci of the web application, the MADA Timeline should include a route map which pins Google Markers at locations of taken photos on a Google Map to indicate the movement trail of the user.

In the bigger picture, GlassMADA points in the direction of data systems which attempt to extend the biological boundaries of human memory capacity through external storage. Actualizing the concepts of resourceful storage behind the theoretical, information-collecting Remembrance Agent, a fully contextually-aware version of GlassMADA in the distant future would attempt to intelligently retrieve, correlate, and recall information according to impromptu user needs and/or requests.[15] GlassMADA will be expanded to aid not only people with dementia, but to those who seek more information whether it was lost or not.

APPENDIX

REFERENCES

Alzheimer's Association. What Is Alzheimer's? http://www.alz.org/alzheimers_disease_what_is_alzheimers.asp, 2015. [Online; accessed 2-Dec-2015].

Narrative. The Narrative Clip. http://getnarrative.com/narrative-clip-1, 2015. [Online; accessed 4-Dec-2015].

BioID. Face Recognition and Voice Recognition Playground. https://playground.bioid.com/, 2015. [Online; accessed 4-Dec-2015].

DroidLa. QR Droid Code Scanner. https://play.google.com/store/apps/details?id=la.droid.qr&hl=en, 2015. [Online; accessed 4-Dec-2015].

Wiki. Google glass. https://en.wikipedia.org/wiki/Google_Glass, 2015. [Online; accessed 5-December-2015].

Google. Google glass developers. https://developers.google.com/glass, 2013. [Online; accessed 5-December-2015].

Scandit. Scandit: Barcode Scanner SDK. http://www.scandit.com/, 2015. [Online; accessed 1-Dec-2015].

Google. How to take photo with Google Glass. http://developer.android.com/training/camera/photobasics.html#TaskManifest, 2015. [Online; accessed 1-Dec-2015].

Armin Ronacher. Flask. http://flask.pocoo.org/docs/0.10/license/, 2013. [Online; accessed 1-Dec-2015].

Twilio. Flask-RESTful. http://flask-restful-cn.readthedocs.org/en/0.3.4/, 2015. [Online; accessed 1-Dec-2015].

Miguel Grinberg. Designing a RESTful API using Flask-RESTful. http://blog.miguelgrinberg.com/post/designing-a-restful-api-using-flask-restful, 2013. [Online; published 28-July-2013; accessed 1-Dec-2015].

Benoit Chesneau. Gunicorn: Green Unicorn HTTP Server. http://gunicorn.org/, 2015. [Online; accessed 1-Dec-2015].

Orion Henry James Lindenbaum, Adam Wiggins. Heroku — Cloud Application Platform. https://www.heroku.com/, 2015. [Online; accessed 1-Dec-2015].

Tympanus. Blueprints Vertical Timeline. http://tympanus.net/Blueprints/VerticalTimeline/, 2015. [Online; accessed 20-Nov-2015].

Bradley Rhodes Jeffrey Levine Jennifer Healey Dana Kirsch Roz Picard Thad Starner, Steve Mann and Alex Pentland. Augmented reality through wearable computing. *Presence: Teleoperators and Virtual Environments*, 6.4(1):386–398, 1997.