Imperial College London

CO362: Software Engineering Group Project

Imperial College London

Department of Computing

# Kaizen Augmented Reality Visualisation

*Group 5:*
Athithan Dharmaratnam
Emyr Davies
Yejin Seo
Andy Li
Ali Chaudhry
Joan Tso

*Imperial College Supervisor:*
Dr. Anandha Gopalan

*Theodo Supervisors:*
Abbie Howell
Ambroise Laurent

January 7, 2019

# Contents

# Chapter 1

# Executive summary

## 1.1 Managing Software Efficiency and Development

Software Management is critical to application development. Studies by the Project Management Institute have shown that in the US, "$122 million is wasted for every $1 billion invested due to poor project performance".[7] Therefore, there is a necessity for software management methods and tools to mitigate poor project efficiency. Furthermore, there is also a need for appropriate methods and metrics to measure the output of an application development project. Relying on basic metrics such as the percentage of delivery dates achieved only gives a basic overview of success.[17] More in depth metrics combined with an appropriate cross project management tool is needed to fully evaluate the productivity of the application development process.

## 1.2 What is a Kaizen Board?

Kaizen translates to change for better. In project management terms this means continuous improvement.[9] A Kaizen board is a visual tool that helps agile teams to manage and track ideas for improving the efficiency of their software development process with a focus on the way the team works. Kaizen is a mentality which exists on all levels of the organisation with the goal being increased work productivity. The Kaizen board simply visualises the continuous improvement thought process by tracking in depth standardised metrics for software efficiency to help answer questions such as "How much software functionality did a team deliver in a given time period?".[17] Hence a Kaizen board can not only address problems with measuring software productivity, it also provides a way to improve upon deficiencies in productivity.

## 1.3 Augmented Reality Kaizen Board

A Kaizen board is essential for organisations with multiple teams and an improvement focused work culture. However, despite its obvious benefit, the overhead of utilising a completely physical Kaizen board often outweighs its performance benefits. The use of metrics to evaluate the performance of an improvement is often cumbersome to work out manually such as hand drawing graphs on whiteboards. Furthermore, a physical Kaizen board for each project limits communication between teams about the improvements as well as being "only a snapshot view of the current state of each project".[12]

However, having a completely digital Kaizen board can also minimise communication and discussion regarding improvements as colleagues can simply view their Kaizen board online without gathering and discussing their ideas with their colleagues which is a key part of the Kaizen mentality.[13]

Our approach is to create an Augmented Reality interface for the physical Kaizen board in order to bring the benefits of both the digital and physical worlds for software efficiency management. Our product includes features such as automated graph generation, cross project improvement tracking and a ticket specific messaging service packaged in a cross-platform mobile application.

## 1.4 Target Audience

The product has been designed specifically for an external client who utilises physical Kaizen boards in combination with a project management tool called Trello. However, the application can be used by any company who wishes to implement a Kaizen board for their agile projects with the only external software dependency being Trello for the digital agile board.

# Chapter 2

# Introduction

## 2.1 Motivation

When observing a Kaizen board, it is clear that all the data on the board is digitally generatable but was produced manually by hand. Although the data itself is useful, there are excessive overheads in time used for printing and updating graphs by hand. The hand-drawn graphs lower the clarity and interactions as well.

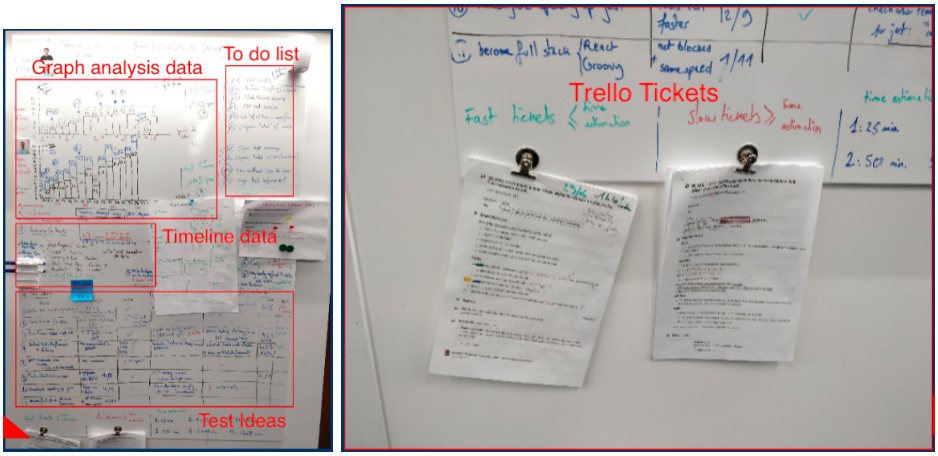



Figure 2.1: Theodo Kaizen Physical Board

Each team needs to use the handwritten Kaizen board every day as part of their scrum meeting. Thus automating the graph updates would reduce the daily scrum meeting time and would add up to increase efficiency by a noticeable amount. As you can see from Figure 2.1 the board is quite cluttered with loose bits of paper and physically printed Trello tickets. It lacks enough space for all the information and makes it difficult to read. With meetings often requiring printing jobs in order to share new information we not only cut down on paper wastage but also printing time.

## 2.2 Objectives

Our mobile application would:

- Support both iOS and android.
- Be able to display relevant information in AR display.
- Have a way of linking different Kaizen boards with its Trello boards.
- Enable users to comment and interact with each other through the app.

## 2.3 Achievements

Our mobile application can:

- Supports both android and iOS.
- Visualise relevant Trello information for a Kaizen board in AR.
- Support features such as filtering, labeling, timelines, summaries and expansion of data.
- Enable users to add comments to their Trello tickets
- Automatically analysed project progress in different ways.

# Chapter 3

# Project management

## 3.1 Project Plan

Since this project was proposed by a client for their specific use, we conducted an initial project meeting with them in addition to our preliminary group meeting. From this we created a four checkpoint iteration plan.

| Iteration | Initial Goals | Final Goal |
|---|---|---|
| 1 | As A User (AAUser), I can install the app on both android and iOS phones<br><br>AAUser, I can see the relevant Trello ticket next to the Kaizen board through my phone when using the app. | AAUser, I can install the app on android and iOS phones<br><br>AAUser, I can see the relevant Trello ticket next to the Kaizen board through my phone when using the app. |
| 2 | AAUser, I can easily access the related digital information of my Kaizen board through its QR code.<br><br>AAUser, I can compose a graph based on Google sheets and display it on my board in AR | AAUser I can visualise graphical data for a Trello ticket (Timeline of columns)<br><br>AAUser I can see the details of a Trello ticket such as users and labels |
| 3 | AAUser, I can interact with my coworker through the app.<br><br>AAUser, I can colour code and comment the parts in the board to signal something to my coworkers. | AAUser I can add a comment on the Trello tickets<br><br>AADev I have designed a cleaner AR Trello Menu to access Trello tickets |
| 4 | AAUser, I can take picture of a physical memo to be made into a digital memo that can be attached to the board. | AAUser, I can see an metrics about the project efficiency<br><br>AAUser, I can track all the Kaizen Improvements measures across a project |

Table 3.1: Project Plan Summary

The project plan was designed so that we would be able to learn about the libraries and frameworks utilised at the beginning of the project while incrementally increasing the functionality of the product. We allowed scope within the initial plan through risk assessments to account for mitigating variables such as time limitations and technical issues especially when working with a new technology such as Augmented Reality and Mobile App Development.

We have tried to capture the project requirements requested by the client via incorporating them into our initial project iteration plan. As we learnt more about the libraries and the scope of the project we adjusted or removed the iteration goals after Sprint Retro meetings with the client while still using the

initial goals as a rough guideline of the feature set the client required, resulting in Final Goal columns in Table 3.1.

## 3.2 Agile Methodologies

The Agile methodology that we chose to use was Scrum. This decision was made primarily because of how the Scrum methodology synchronised with the structure of the project itself, and was also influenced by our relationship with our client. The sprint system of Scrum fit well with the 4 checkpoint iterations of the project such that each iteration aligned with a 2 week sprint. This allowed for extreme clarity during the overall project and sprint planning throughout. The external client is a development company that also employs the Scrum methodology in their teams, so we were able to benefit from speaking to their in-house agile coaches and organised our meetings inline with the sprints.

### 3.2.1 Sprint Planning / Retro

A core part of the Scrum methodology is the concept of working in sprints. A sprint is a short period of time in which a usable product iteration is completed. The end of one sprint leads to the beginning of another during the project. Each Project is composed of a number of sprints with each sprint having a requisite sprint plan and sprint retro. The sprint plan is the objective of the sprint iteration along with the relevant tasks, assigned and organised appropriately.

Due to the 4 checkpoint structure of the project, we decided to align our sprints with the checkpoints such that we had 2 week sprints ending on the day of the checkpoint. The sprint planning and retros were completed in a meeting with our external client. Each meeting with our external client would first consist of a demonstration of our product at the end of the sprint followed by a "retro" for the sprint. Here, "retro" refers to the sprint retrospective, a group feedback session regarding the sprint from a general and development perspective. Some of the areas and questions discussed in the retro tackle the positive and negative things that happened during the sprint. For example, the group discussing what they thought went well throughout the sprint and what they thought did not work well, as well as touching on what could be improved. As a result of the retro discussion, improvement points are picked to focus on for the upcoming sprint whether it be better communication or a specific strategy change.

### 3.2.2 Stand-up Meetings

Another integral part of Scrum methodology is stand up meetings, a very quick meeting at the start of every day where each group member enlightens the group on what they are currently working, what they have already completed and anything they feel necessary to share as a result of their work to the rest of their team. As a group of 6 students doing different modules, it didn't seem feasible for us to all meet in one physical location at the start of every day. Instead, we decided to do digital stand ups using a plugin in Slack called StandupIy. The plugin would ask every team member 3 questions that we chose as a group and then send the entire group response to each member. This allowed for clear distillation of information across team regarding previous days progress and any unexpected issues that had arose.

## 3.3 Group Organisation

Having decided to utilise the Scrum development framework, we assigned the agile specific roles within the team as well as general software roles involved in the development of the application. The Table 3.2 shows the Task Allocation we assigned for each team member.

| Task Allocation | |
|---|---|
| Name / Role | Tasks |
| **Athithan Dharmaratnam**<br>Product Owner<br>Full Stack<br>Dev Ops | Front End: AR Menu, Comments, Redux Integration, Login Trello Oauth, Login Help Screen / Tutorial and Kaizen Improvements Screen<br><br>Back End: Trello API Calls, Database Calls and Schema<br><br>Dev Ops: Setting up Jenkins CI and CD, Back End EC2 instance, RDS Database Instance |
| **Ali Chaudhry**<br>Scrum Master<br>Full Stack | Front End: Login Screen, Board Pin Access and Initial Marker Setup<br><br>Back End: Trello API Calls, API Calls Unit Testing Suite |
| **Andy Li**<br>Full Stack<br>Dev Ops | Front End: Dynamic Interactive Timeline Graph<br><br>Back End: Setting up the Node.js server, Trello API Calls<br><br>Dev Ops: Back End EC2 instance |
| **Emyr Davies**<br>Front End | Front End: Image Markers, UX Test Script |
| **Yejin Seo**<br>Front End | Front End: Card Detail View, Displaying Graphs, UI Design Implementation, Animation<br><br>Back End: Trello API calls |
| **Joan Tso**<br>Full Stack | Front End: Project Set Up, Card Detail View<br><br>Back End: Trello API Calls, Graph Generation (Data Visualisation) Calls and Relevant Tests |

Table 3.2: Contributions of each team member

## 3.4 Software Development Practises / Strategies

### 3.4.1 Pair Programming

Pair programming was an integral part of our software development strategy, not only to achieve higher productivity as a team but also to enable as many people as possible to work on different aspects of the application. Another benefit of our pair programming approach was that no part of the codebase or application solely relied upon a single team member so that even in the case that one team member suddenly becomes unavailable due to sickness or other reasons development in all areas could continue. As a group of 6, we split ourselves into 3 groups of pairs from the beginning of the project mostly residing in the same teams but also switching around occasionally as and when needed to accommodate peoples desire to contribute to other parts of the project.

### 3.4.2 Version Control

#### Github

We used Github for version control in two separate repositories containing the two components of the application. The first repository is the Front End mobile application while the second repository is the Back End API Server. Github was chosen as it worked well with our Continuous Integration framework and Web hosting service.

#### Branch Management

For the Front End repository, we utilised the master branch as a release branch which consisted of all the features that are production ready. We chose not to have a deploy branch for the Front End and instead used a more liberal model of utilising branches for small features or hotfixes for bugs before deploying to the release master branch. We chose this model as the features we added were incremental changes to the main master branch and we were not deploying our application to users continuously on the Front End. Instead we generated standalone APK (Android Package) or IPA (iOS App Store Package) files for each end of iteration sprint for the client.

For the Back End repository, we utilised a similar model with the addition of unit tests occurring whenever we pushed to the master branch. The changes to the Back End usually involved adding similarly formatted API Calls to the Trello API or Database Calls as well as minor bug fixes. Therefore, we utilised short-lived branches for the additional API Call functions before merging to the master branch invalidating the necessity for a separate deployment branch.

#### Commit messages

In order to better organise the workload and help to debug, we made sure to include the College IDs of the members working on a particular commit in the message. Appendix C.1 shows an example commit message.

### 3.4.3 Continuous Integration / Deployment

For the Back End API Server, we set up Continuous Integration and Continuous Deployment using Github Webhooks and Jenkins. When a commit is made to master, a Github Webhook is fired which pushes the repository to the Jenkins server which is hosted on an AWS EC2 instance. The Jenkins server then runs the test bash scripts locally which upon successful execution then runs the deploy bash script which connects through an SSH connection to the Back End API server hosted on another AWS EC2 Instance. We utilised a production process manager called PM2 to restart the Back End API Server at the end of the Bash Deploy Script. See Figure 3.1 for a visual representation.

For the Front End since we were deploying to two different mobile platforms and were only generating demos at the end of a two week iteration, we chose not to use Continuous Integration and Deployment.

### 3.4.4 Consultations

During the project, we had two software engineering consultations that were of great benefit to us. The feedback received on the current implementation of our product architecture and our group organisation, in general, were very helpful. Also we able to discuss ideas during the consultations regarding the automated testing methods of our product and getting user testing feedback.

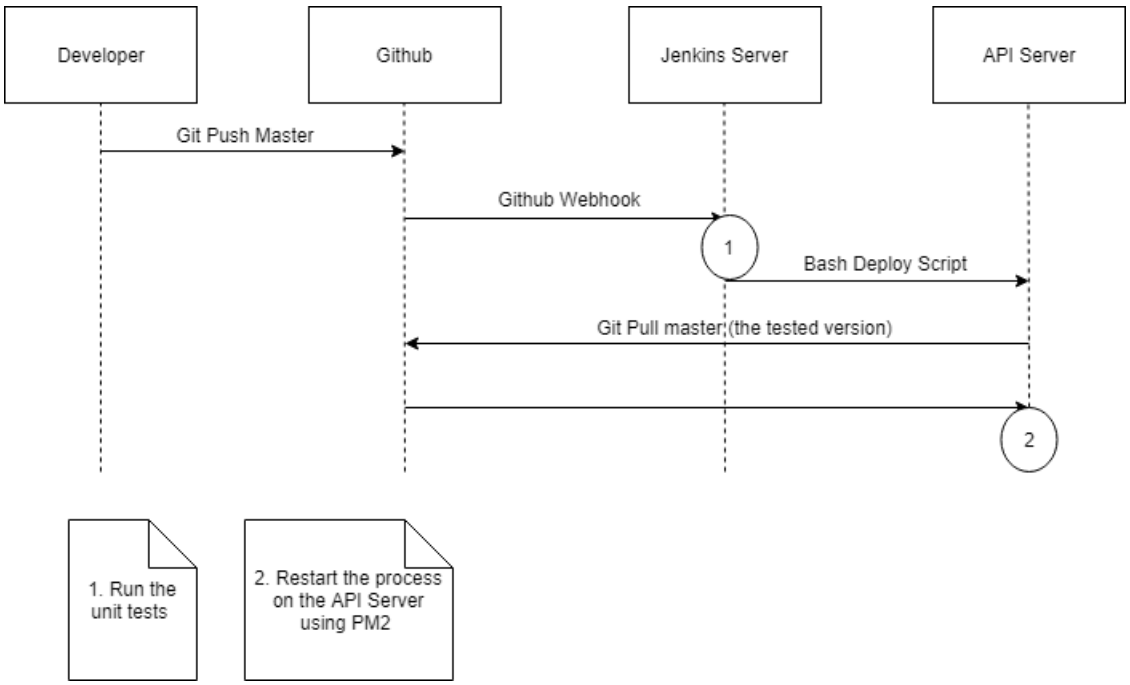



Figure 3.1: CI / CD Flow

## 3.5 Tools

### 3.5.1 Communication

Slack was our primary internal tool of communication throughout the project. Some of the reasons as to why we chose Slack were because of the channels feature which allows conversations to be specific to certain areas and to take place among relevant group members. Slack also offers a massive library of plugins. One of the plugins we used was Simple Poll which we used to organise meeting times.

Throughout the project, we communicated with our external client via email to organise meetings, various admin and any questions we would have during a sprint.

### 3.5.2 Web Hosting

Our API server required cloud hosting and we considered several options such as AWS, Microsoft Azure and the Google Cloud Platform. However, since we were operating as a small team with little hardware requirements, the choice of using AWS was simply due to more documentation being available regarding integration with other services such as Jenkins.

We used two AWS EC2 Instances running on Ubuntu 18.04 LTS to host the Back End API Server and the Jenkins server as it was easy to manage both from the same place using the AWS Developer Console. We then accessed the AWS EC2 Instance using an SSH client called PuTTY on Windows.

Since we had chosen to use AWS for the API server, we used an AWS RDS Instance to host the database running PostgreSQL. We chose PostgreSQL as we planned to use a relational database, while we choose PostgreSQL specifically as it is open source and provides better data integrity. In order to interface with the PostgreSQL database, we used a database management tool called pgAdmin 4 on Windows which provided a graphical interface to manage the database.

# Chapter 4

# Design and implementation

## 4.1 Background

In order to understand the design and implementation of the AR Kaizen mobile application, there is a need to understand some design constraints when working with Augmented Reality on a mobile device. In addition, since the mobile application is dependent on Trello, we have included a basic primer on the client's usage of Trello as a digital Kanban board.

### 4.1.1 Augmented Reality

Augmented reality (AR) is a type of interactive, reality-based display environment that takes the capabilities of computer generated display, sound, text and effects to enhance the user's real-world experience.[2] The purpose of our project is to use AR to integrate the computer-generated components with the real world so that it would simplify and improve the working process of the engineers working with Kaizen boards.

#### Augmented Reality Design Constraints

There are several types of AR and as shown below:

- *Marker Based Augmented Reality*
  Marker Based Augmented Reality produces an AR environment by layering the reality space with virtual space. To link these two spaces coherently, Marker Based AR uses a marker: a distinct 2D image. Marker based AR uses this marker as a centre point for the two spaces and integrates the computer generated component with the reality based component.

  For example, when a person uses their phone to use a Marker Based AR, the application displays camera inputs(reality space) along with the computer generated component(virtual space). The computer generated component's position and orientation would be calculated based on the marker read by the camera and be layered over the camera reads. Figure 4.1 shows a Marker Based AR business card.

- *Markerless Augmented Reality*
  Markerless Augmented Reality is sometimes known as position/location based AR. Pokemon-Go is an example of a Markerless AR. Markerless AR uses GPS, digital compass, velocity meter, or an accelerometer to recognise the reality components and layers the virtual component in position based on those.

- *Projection Based Augmented Reality*
  Projection Based Augmented Reality works by projecting artificial light onto reality and detecting user interaction to the projected virtual component. User interaction detection are usually done by differentiation between expected projection and altered projection.

- *Superimposition Based Augmented Reality*
  Superimposition Based AR replaces the original view of an object with an augmented version of it either partially or fully. Object recognition plays a vital role because this implementation has to detect the object that it would augment.


Figure 4.1: Marker Based AR used to augment a business card

Although Markerless AR is the most widely implemented on mobile applications, it is not suitable for our application which has to locate different Kaizen boards within a close proximity.

To use Projection Based AR, we would need extra hardware support such as a projector with a reflection sensor. We thought it was economically inefficient for the client to use this because they would need to install one interactive projector per project Kaizen board.

Finally, we came down with Marker Based AR and Superimposition Based AR. The big difference between the two is that Superimposition Based AR uses object detection while Marker Based AR uses image recognition. Considering the fact that we are extending a 2D object(Kaizen board), we thought using Superimposition Based AR would increase the complexity of the program without corresponding benefits. Therefore, we decided to implement our app in Marker Based AR.

### Viro React vs Expo

Our client requested the mobile application to support both iOS and Android devices as their employees mainly use these devices. We initially considered creating native applications that used the native AR libraries (ARKit for Apple and ARCore for Android) for each platform which may have made areas of the implementation less technically difficult such as application deployment but essentially would have doubled the development time required as we would have had to develop two mobile applications. Instead, we decided to use React Native which allowed us to develop a cross-platform mobile application. After researching, we found that the two main ways to implement marker based AR in React Native were through using Expo with Three.js or Viro React.

### Expo, Three.js and ARKit

As one of the most common tools used to build a React Native application, Expo was tempting to use as it had extensive documentation and useful developer features. However, to build an AR application with Expo was a different story. Currently, Expo's AR tools only support ARKit but not ARCore[3], which means it would only support iOS devices on AR. This invalidates our decision to use React Native to support both Android and iOS application, so we couldn't use this library.

**VIRO React**

VIRO React is a library for making a VR/AR application with React Native. Unlike Expo's AR kit, it supports both ARCore and ARKit and therefore allows our application to support both Android and iOS devices. The library not only features plane/surface detection but it support image recognition along with its useful testbed application.[4] It also has an efficient support system through Github and Slack.

Therefore, we decided to use Viro React to develop the Augmented Reality component of the application.

### 4.1.2 Trello

**What is Trello?**

Trello is a tool for project management that allows users to keep track of the status of tasks of their projects.[18] The main components that we use in the mobile application are listed below.

- Board
  A board contains information about a project. The user can create a new broad when they start a new project and invite the team members to join the board.
- List
  A board contains several lists which represent several stages of the tasks. (e.g. Doing, Done, Code Review, etc.) The user can create lists for the important stages of the tasks. The list our client uses is Sprint backlog, Doing, Code Review and Sprint X Done (where X is the number of that sprint).
- Card
  A list contains several cards which represent tasks or tickets of the project. The users can move cards to another list when a task enters another stage. The users can also add responsible members, checklists and comments to a card.
- Label
  Labels can be added to a card for classification purpose. The user can search for cards with specific labels and also customise the labels.

Our client uses Trello as a digital Kanban board. Figure 4.2 shows a screenshot of a Trello board we created for our project under the guidance of our client.

**Trello API**

The Trello API allows us to retrieve and update information from Trello. To use the Trello API, we used an API key and an access token that is generated after the user authorised our app to access their data on Trello. Most of the data used in the mobile application is polled from the client's Trello Board.[14]

### 4.1.3 Design principles

In order to decide which features to implement in the application, we listed out key use cases of the Kaizen board that we would design our features to try achieve.

- *Collaboration*
  Discussion and collaboration about the Kaizen improvements is a key part of the continuous improvement philosophy. Propagating and tracking the Kaizen improvements across multiple teams is important in actually making use of the Kaizen board. In addition, communicating about specific

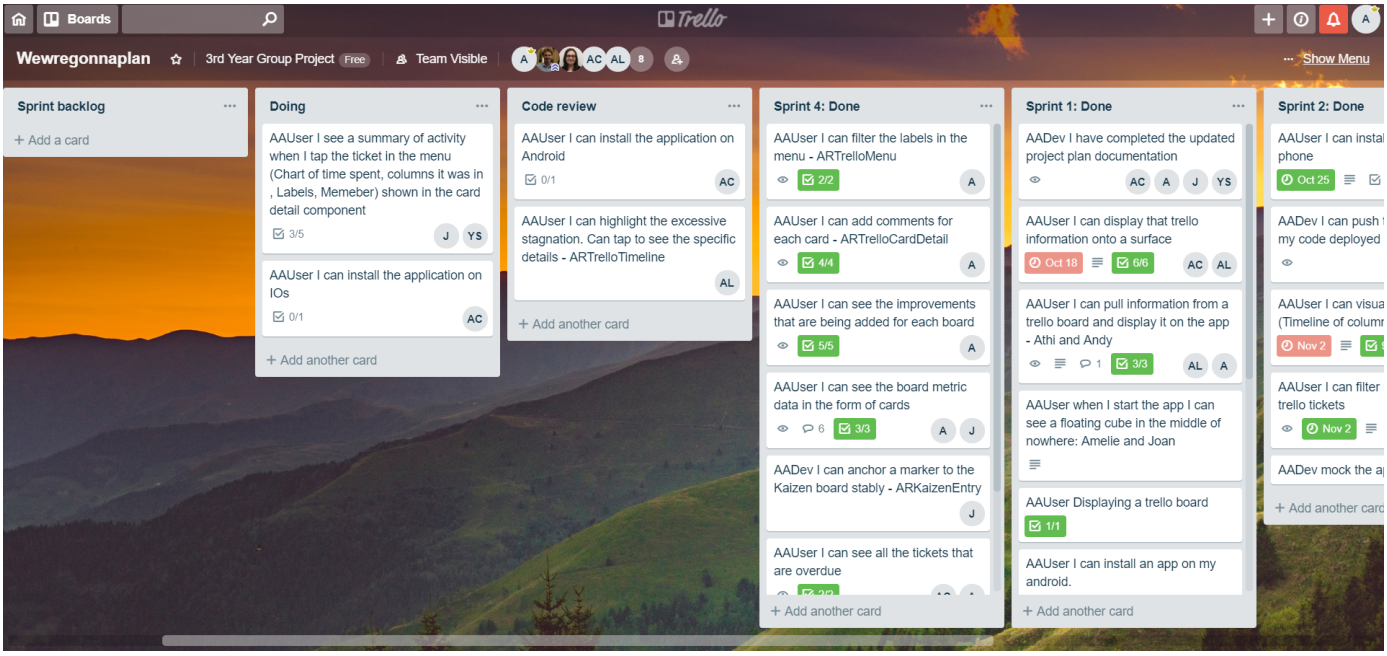



Figure 4.2: Screen shot of Trello Kanban workflow of our project

improvements and tickets outside of the designated Kaizen Board Meeting is part of the Kaizen mentality as it requires the users to always be thinking about improving the project efficiency to accelerate the rate of improvement.

- *Overall Project Management*
  The Kaizen board has various metrics which evaluate the project efficiency. From a project management perspective, these metrics should be as relevant as possible to clearly display the efficiency of the overall project. A manager could also be in charge of many teams, in which case the manager should be able to access these metrics across multiple projects easily. This allows the manager to quickly identify and respond to potential problems flagged by the metrics. Furthermore, a manager should be able to evaluate and capture an improvement to share it across the entire organisation. A Kaizen Board Meeting should be as focused as possible about the actual Kaizen improvements and project efficiency metrics to be time efficient.

- *Easily Customisable*
  The Kaizen board can be used to track many metrics and a key focus is to be able to quickly interchange and modify these metrics without significant time and effort as well as being able to store the format of these metrics when the metric is not being evaluated on a particular project.

## 4.2 Overview System Architecture

The external client's general requirement was to have an augmented reality mobile application that overlays on top of a physical Kaizen board. Therefore, we divided the system into two main components (Front End and Back End) which are supported by several supporting components used in the development of the application. The Front End component is the augmented reality (AR) mobile application while the Back End component consists of an API server, a database and a Jenkins server.

We decided to follow a modular design choice by separating the Trello API calls into an API server rather than bundle them into the mobile application. This was done as some of the features required

data processing of the data returned from the Trello API calls. In addition, we would also need to make queries to a database and hence to ensure extensibility and software robustness we separated the functionality of the mobile application to mainly be a data visualisation application while also taking in user input. See Figure 4.3 for the structure of the system.

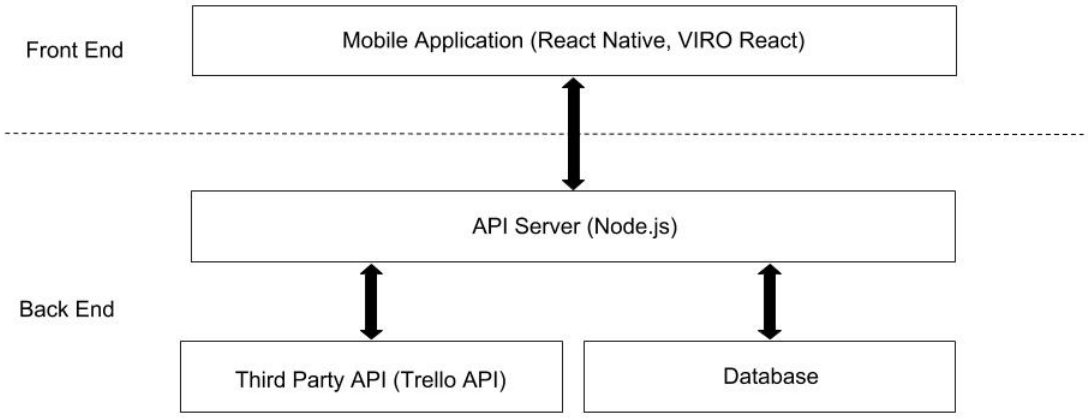



Figure 4.3: System Architecture

- Mobile Application
  The Mobile Application presents the user with the augmented reality interface overlay of the Kaizen board. In addition, it also displays data metrics and Kaizen Improvements.
- API Server
  The API server retrieves data from users' Trello board. It then analyses, processes or generates the polled data before passing it to the mobile application. It also queries the database to access Kaizen improvements data as well as handling the login authentication.
- Database
  The database stores the Kaizen improvements data.
- Third Party API
  Trello API was used to interact with users' Trello boards

## 4.3 Front End (Mobile application)

### 4.3.1 Technical Overview

As previously mentioned we have chosen to use React Native to develop our client side mobile application with React Viro used for the augmented reality component. Another reason we have chosen to utilise React Native is due to the plethora of community created parts and plugins. Given the limited development time frame we had as well as our general inexperience with mobile development, we found it helpful to have ready made React components rather than creating the standard components from scratch. We have also used the VIRO TestBed app for testing during the development process. (See Section 5.1.1)

A major library used in the mobile application was Redux. After iteration two we found that communication between React Components was getting cumbersome as normally, it is done through using React props which are passed from a parent React Component to a child React Component. The issue

was that we had a large depth size in the React parent child hierarchy. Furthermore, communication across children in different branches was not efficient as props had to be passed through their shared parents. Therefore, we utilised Redux as it provided a single source of truth via having a singular global store which contained all the shared variables. The variables are then accessed from Redux containers which pass down the shared variables as React props as well as allowing access to Redux reducers functions which can change the global Redux store. This methodology ensures that the global store maintains a read only state as the changes to the Redux store are made using the previous state and an action which returns a new state hence reducing the problems of race conditions. [10] We mainly used Redux to manage communication across components with a parent child hierarchy depth greater than three or across the three main views of the augmented reality interface which are the AR Menu View, AR Card Detail View, AR Card Timeline View as shown in Figure 4.4.

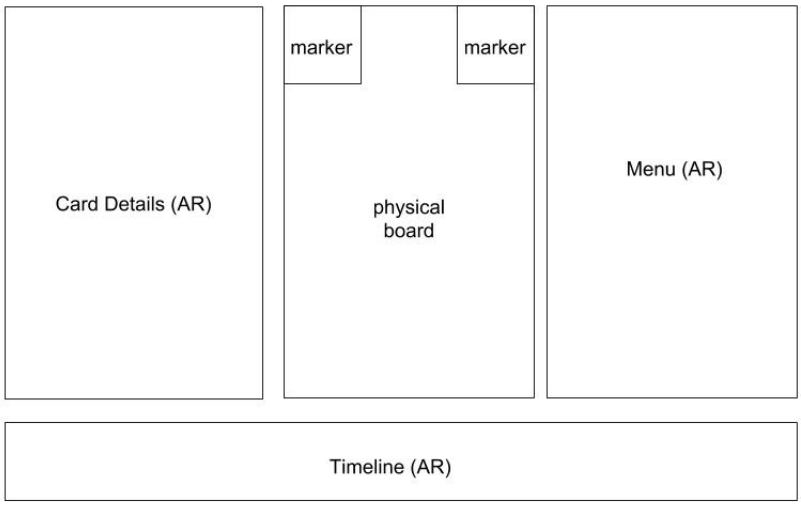



Figure 4.4: Design of the Augmented Reality interface

### 4.3.2 Card Menu View

The Card Menu View is the entry point of the augmented reality interface. The challenges we faced when designing this interface were mainly UX problems as well as the positioning of the AR Menu options and results in 3d space.
The design challenge here was to ensure that the physical board and the augmented reality overlay did not conflict. We also had to consider that the user was using a mobile device with limited screen size to view the overlay. Hence after some UX testing (see Appendix A: UX Test Script) and discussions with the client at the end of iteration meetings, we decided on a fixed menu view component where each menu option choice replaced the previous view in the same visual space. This replaced the previous design of having a persistent main menu and the subsequent options being displaced to the right of the main menu which made viewing with a phone a cumbersome task if there were many generated options, especially in the Trello list section. This solved the problem of an excessive horizontal view but generated a smaller problem with a potentially long list of generated boxes in the vertical direction which can be solved through implementing a scrolling function in future iterations.

The menu consists of several options for the data visualisation of the metrics and Trello Board information shown in Figure 4.5.

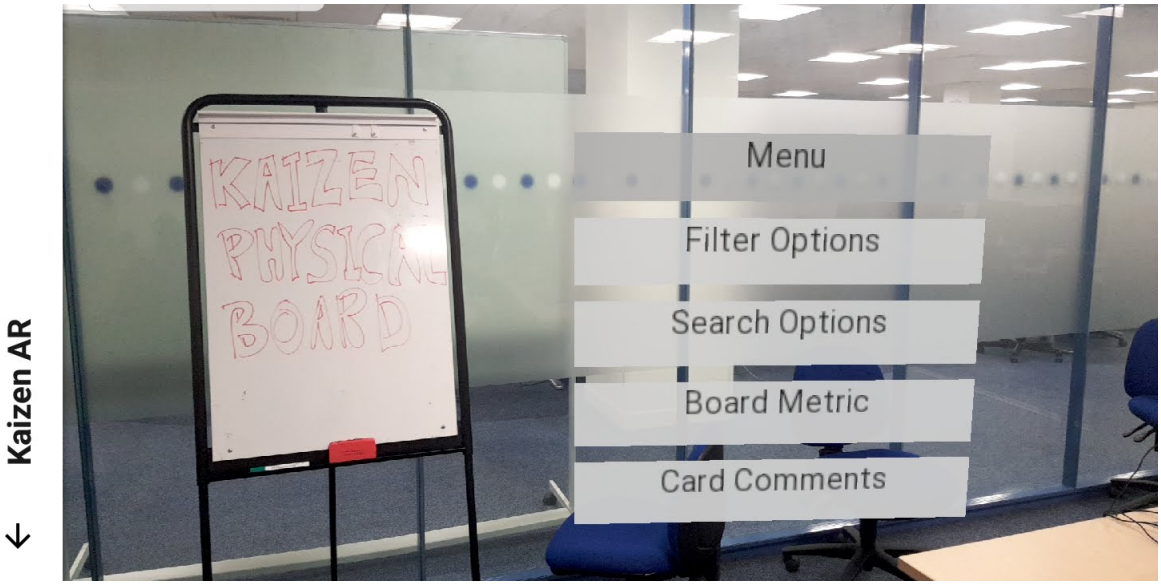



Figure 4.5: Menu Options

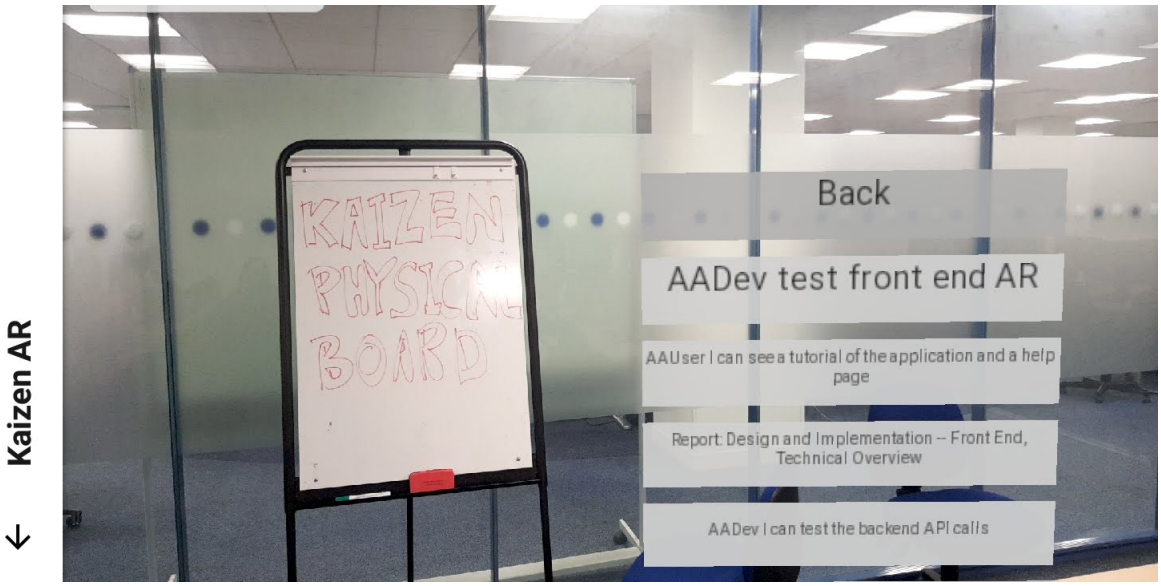



Figure 4.6: Search Menu Generated Trello Cards

1. **Search Options**
   The Search options is used to access a particular card in a Trello Board. Each choice made from the initial search options generates a smaller search space: Trello Board, Trello List, Trello Card as shown in Figure 4.6 and Figure C.3.

2. **Filter Options**
   The external client used a Trello feature called Labels which helped the client tag tickets in Trello. We extracted this feature using the Trello API Label calls for each Trello board. The user can choose a label which will then only display the cards with the associated label in the final Trello Card search space from the Search Options Sub Menu. The filtering Trello label ID was chosen on the Front End and stored using the Redux Store. Any subsequent Back End API Call would then only return Trello Cards with the matched Trello Label ID. An example filter menu label is shown

in Figure C.2.

3. **Board Metrics**
   In addition to the visualisation of Trello data, the application can also dynamically generate analytical data based on the input Trello data on the Back End API. The various options for the Board Metrics are shown in Figure 4.9.

4. **Card Comments**
   The comment screen is accessed from this option and overlays a modal on top of the AR Component using a React Native part called React Native Router Flux [16] which was also used to handle the general navigation of the application. We utilised this community created library as we could define all the routes in one central place and navigate and communicate between different screens of the application using the library function call "Actions.screenReference()".

### 4.3.3 Card Detail View

We introduced this feature as the client was interested in the progress of each card which they could assess by accessing more detailed information polled from the Trello API. One specific use case would be when a user wants to see how much time is left to complete the card. The card detail view will pop up on the left of the physical board when the user selects a Trello Card from the Trello Card search options sub menu. Redux is used to handle this communication across React Components using Redux Actions. See Figure 4.7 for the layout of the Card Detail View.

1. **Movement Graph**
   The movement graph shows number of the days the selected card has been in each Trello List. This helps the users to locate which particular stage of the project caused issues which can help improve project efficiency in future iterations.

2. **Due Date**
   Due date shows how many days are left until the Trello Card is due which helps with task allocation and project management. If the card is overdue, then this section will instead show the number of days the card is overdue.

3. **Members**
   This section shows the members assigned to the selected card so the users can easily knows who to contact for a particular problem related to a card.

4. **Checklist**
   Checklist shows the items in the selected card and their status (complete/incomplete) to give an insight into the progress of a ticket.

### 4.3.4 Card Timeline View

In the Board Metric View and the Card Detail View, we have displayed static graphs as an image generated from the Back Ends shown in Figure 4.8. However, our client also requested a feature that allowed user interaction with generated graphs after an end of iteration meeting. Hence, we built a graph composed fully out of Front End components, mainly using React Viro's FlexView. A FlexView is simply an interactive container. This allows for user interaction with each individual component of the graph, set by custom on-click events for each relevant part of the graph. This was mainly implemented as a proof of concept feature that could be extended. Therefore, in the current iteration, the on-click simply pops up a numerical value for the total time spent in each column as it is not easy to read visually due to the graph being scaled inconsistently between Trello Cards to better display smaller time intervals.

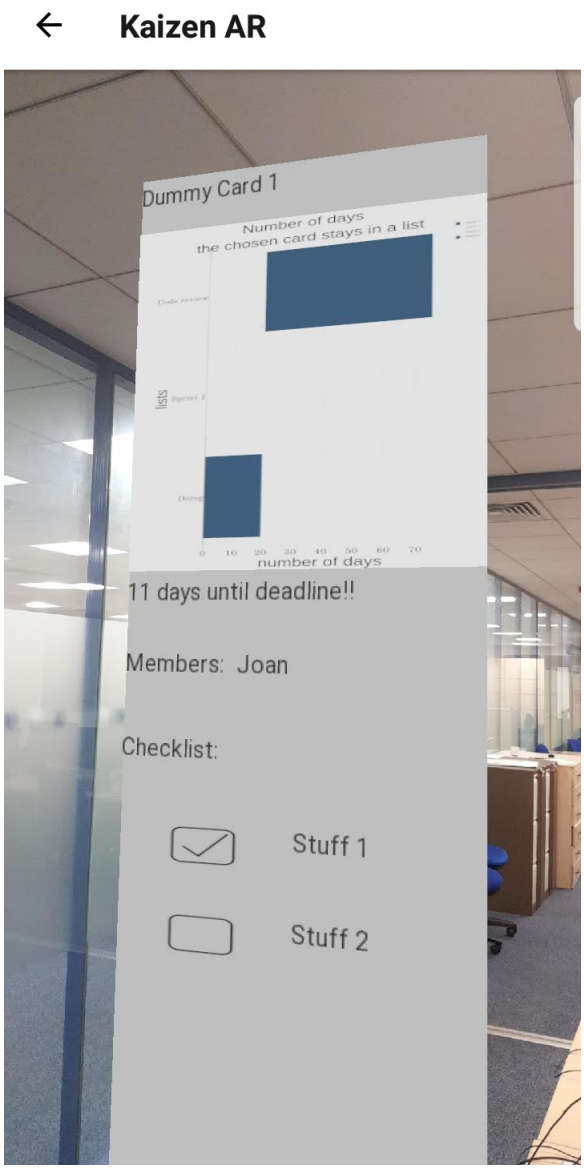



Figure 4.7: Screen shot of the card detail view

1. **Graph of Time per Column**
   Shows a timeline of which columns the selected ticket has been in from creation to most recent change. This helps project management as it visualises ticket stagnation which the user can then take measures to mitigate.

2. **Information Panel**
   If a bar from the timeline is selected, show how much actual time the bar represents.

### 4.3.5 Board Metrics View

The client's physical Kaizen board has various metrics that are hand drawn. Our goal was to augment the physical Kaizen board which also includes automated metric generation. Upon analysis of the metrics used by the client, we abstracted generic metrics which can be used across multiple projects. This mainly manifested in the form of graphs which are generated from the Trello board data. In addition, we also noticed that from a project management point of view, having the ability to recognise the failing tickets

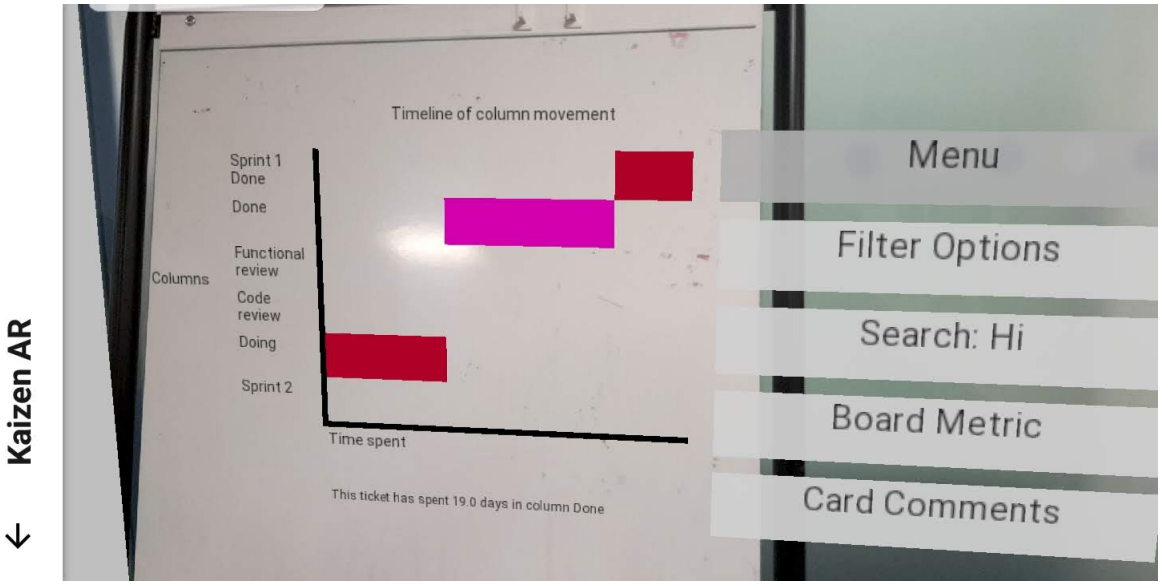



Figure 4.8: Screen shot of the card timeline view with selected bar

easily is beneficial when analysing the efficiency of the project. Hence, we added a flagging system using the overdue tickets options.

1. **Cards Per Column**
   (See Section 4.4.3)

2. **Performance**
   (See Section 4.4.3)

3. **Overdue Tickets**
   Sets a flag, which when displaying a ticket in the search option's Trello Card sub menu, highlights it if it is overdue as well as providing additional information such as the date it was due to better inform a project manager about problem cards in one view. This is shown in Figure C.4.

### 4.3.6 Login Screen

The landing page of the application starts with a tutorial which shows the user how to navigate it which is shown in Figure C.8a. We used a React Native library called React Native Copilot [5] which handled the visual aspect of the tutorial in a fluid way. We added this feature as UX test feedback suggested that the application was confusing to use. Hence to make the application more user friendly we also added a help page shown in Figure C.8b which contains a user guide on how to use the Augmented Reality interface which was also suggested to us by our project supervisor. For the help page, we utilised a swiping screen format using another React Native community library called React Native Swiper [6].

### 4.3.7 Kaizen Improvements

One of the disadvantages of having a physical Kaizen board is that the tracked changes are limited to the specific projects. In addition to providing an augmented reality interface, we have also added a cross-project screen which is accessed from the landing page shown in Figure 4.10. The Kaizen Improvements screen lists the approved and pending improvements that have taken place, as well as its evaluation. Kaizen is a mentality which focuses on continual improvements between and across projects. For organisations with multiple projects such as the current client, having an overview page is essential to manage


Figure 4.9: Board Metrics Menu

and keep track of all the improvements to avoid overlap and also to encourage discussion between projects.

From this page, you can also post new improvements to the database for the currently selected board stored in the Redux store which posts the status of the improvement to be set as default "Testing" stage.

### 4.3.8 Comment Modal

The client requested the ability to communicate and flag cards. After considering implementing an internal messaging system, we chose to instead use the existing comments feature on Trello Cards. This design choice was made as adding a messaging system was superfluous to the actual requirement. Using the existing Trello comment system ensured that the communication was specific to each card rather than being a general discussion which can be done using other tools like Slack.

The comments are polled using a unique Trello Card ID which is internally stored in the Redux Store after it is selected by the user using the search options submenu. The comment modal page was designed to be simplistic as it displays the user id and the comment on each card in list format shown in Figure C.5. The user can also add comments using the form box taken from the React Native library. The refresh of comments is done using a pull down refresh option and initially when the comment modal is first loaded. Since we were using Trello API comments it was difficult to implement automated Trello comment polling as a comment update event was not available in the Trello API. Therefore to have this feature we would have had to continuously poll from the Front End which was inefficient. Therefore we designed the comment update system to be user controlled.

One use case of the comment system is to flag tickets. If a project manager selects a particular card they can add a comment onto the digital Trello board which will then notify the people working on the card when they login into Trello. We did not want to be too intrusive when designing this flag for help feature through pushing email notifications from the application and instead choose to utilise the Trello Comments System.

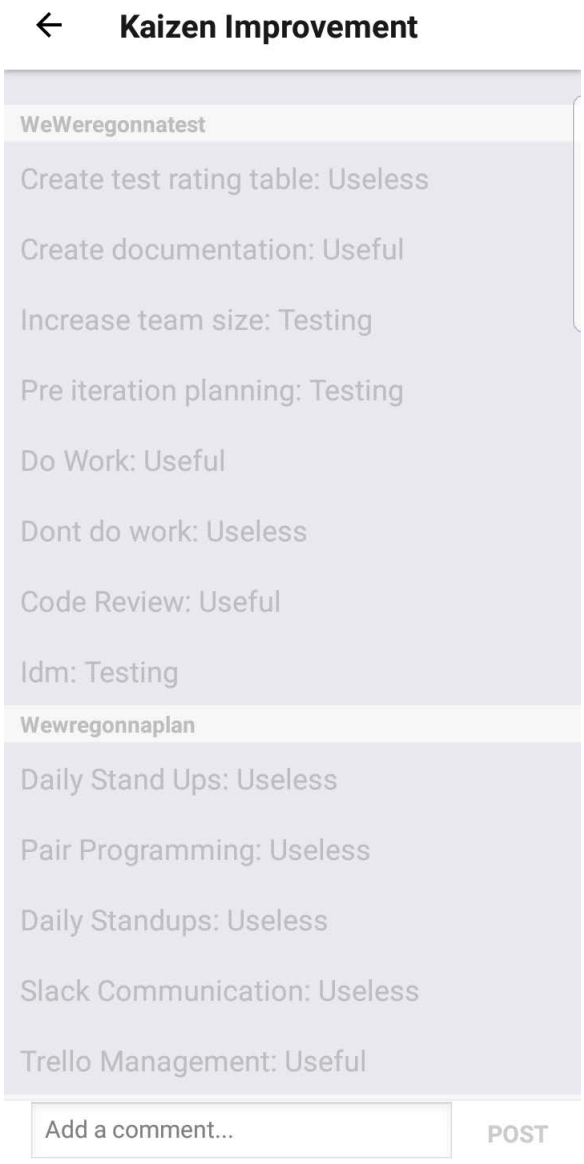



Figure 4.10: Kaizen Improvements Screen

### 4.3.9 Anchors

An anchor is a general term for markers used in Viro React. In the AR scene, the image produced by the camera, which is the real world, contributes to the "background". This background is then merged with virtual components through the use of anchors and camera tracking. Anchors can be things such as images or planes detected by the camera.

On initialisation, the camera will get bearings and centre itself at [0, 0, 0] representing [x, y, z] allowing it to track the movement of the real world with respect to the camera and react appropriately.

For our project, our client initially wanted to use QR codes and suggested using it to also anchor relevant Trello information. Due to the incompatibility of the camera and the QR library, we had to switch quickly and we thought of using an image marker instead of QR code. Image recognition allowed us to receive a callback on finding our specific image which meant we could anchor our component to the bearings of

the image marker.

Therefore, we could use the image marker to display our AR interface whenever it was detected by the camera keeping it "anchored" to the image and giving us a reliable position for our virtual component.

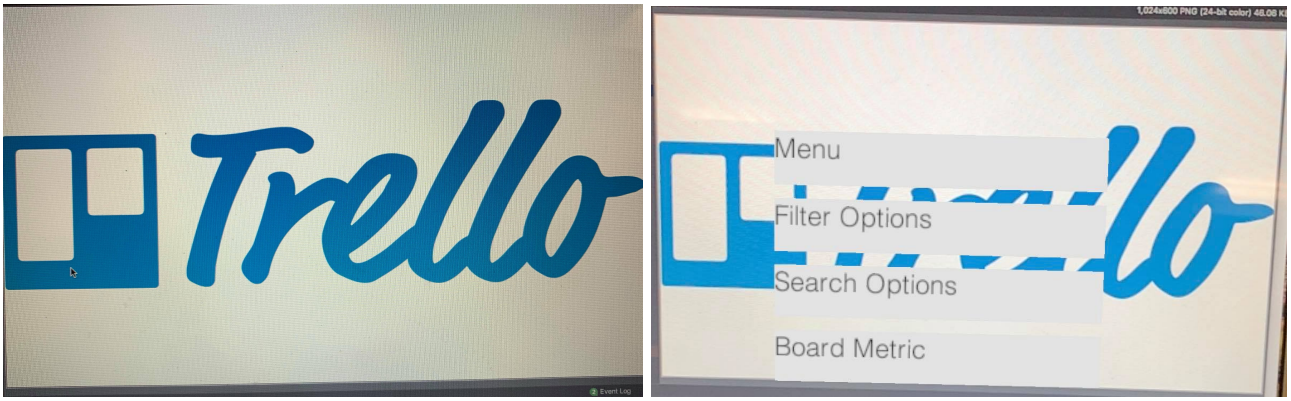



Figure 4.11: Example Image Marker in action using Trello Logo as the image marker

Without an image marker, the virtual component would be very dainty. It would rely on the camera detecting planes correctly in order to position and automatically anchor itself respective to the planes. This would often cause the image to be out of position by a very precise rotation. The rotation could change on every initialisation of AR scene and make it very unreliable and nearly impossible to interact with the AR components consistently.

The image recognition was quite buggy to work with, there were problems such as ARCore not being able to auto-focus the camera within the AR scene. Block colours could cause a crash as they are too prevalent within the real world and the anchor auto-updating its location with respect to the camera would cause our AR component to be very unstable/jittery which is a huge problem for an interactive component.

We were able to solve some of these problems with the help of the Viro React library. We added a dragging feature to the components which would stop the automatic updating of the marker detection. This made the components more stable, and along with locking them within a plane, they were kept disposable at all times. This implementation came with the bonus of being able to move components interactively to the desired location after having been initially positioned on the marker. Using the Viro React library we could also specify a real-world size for the image, this provided more reliability for recognition. This implementation, in unison with larger and more distinct images, enabled us to achieve a very high degree of component stability and detection accuracy.

## 4.4 Back End API Server

### 4.4.1 Technical

The Back End part of our project was a cloud based RESTful API application designed to communicate and provide the data for the client side application. The application was created using Express.js and Node.js, the server runs on an Amazon EC2 instance.

We decided to use Node.js because members of the team already had some experience with the platform. Node.js being based on JavaScript also enabled more people to contribute to the Back End since React Native is also based on JavaScript.

Express.js is one of the most commonly used Web Application frameworks for Node.js. Also, many of our team already had experience using this framework which encouraged us to select this. Express.js routing was used to configure and easily implement the different RESTful calls to the Trello API as well as the Database Queries.

We have four main groups of API calls:

- Data Processing: involves calls that filter the response of a GET call from the Trello API into a readable format, usually for small details like titles of a board, or labels of cards.
- Data Analysis: involves calls that analyse the response of calls from the Trello API and return an infographic generated with Plotly or data in AR render able format.
- Data Interaction: involves calls that interact with our database and comment system, using GETs to update the app view, and POSTs to update either the database or the Trello board itself.
- User Identification: Interfaces with the Trello Oauth API to retrieve a particular users Trello information.

### 4.4.2 Data Processing

To perform the data processing inside of the application we used Trello API's useful existing nested resource structure. As an example, `/boards/{id}` is a valid route, and so are the routes for its nested attributes, `/boards/{id}/actions`, `/boards/{id}/cards` etc. In addition, the API also allows for calls in both directions of the resource hierarchy, so `/cards/{id}/board` is a valid call.

An example of a nested resource call is when we fetch and filter the `/trello/getCardHistory/{id}`route. Here our Back End server calls the Trello API first with `cards/{id}/actions?filter=all`, then filters the response for `actions` that contain the `listAfter` key. Then we extract the important information for each action (column name and date), and then send the response with `res.send()`.

### 4.4.3 Data Analysis

To aid users with their Kaizen project management, we introduced graphs to give clear insight into project efficiency. For each graph, we perform the relevant API calls to get the required information then we do the appropriate analysis to generate certain types of graphs that our users find useful. The graphs include:

- Performance graph
  Cumulative frequency graph that shows the number of cards (tasks) that are moved to the Done list on or before the dates. This helps users to visualise project progress.
- Time-line graph
  A time-line graph that shows the number of days a card stays in a list and the order of the list it stays. This helps the users to locate which stage delays occur.

### 4.4.4 User Identification

The data our application displays is split into two parts. The first part is polled from Trello boards that are specific to a user's project while the second part is generated by the user. This second part is either in the form of the graph generation or the data for the Kaizen Improvements Page. Since most of the data utilised in the application is polled from Trello, we decided to forgo an internal login page. Instead, we made a design choice of utilising the Trello API login system shown in Figure C.6 to gain access to the

user ID of the Trello user which we then stored internally in a temporary session of the mobile application using the Redux Global Store. We used OAuth1 to login to the Trello API as shown in Figure 4.12. [11]

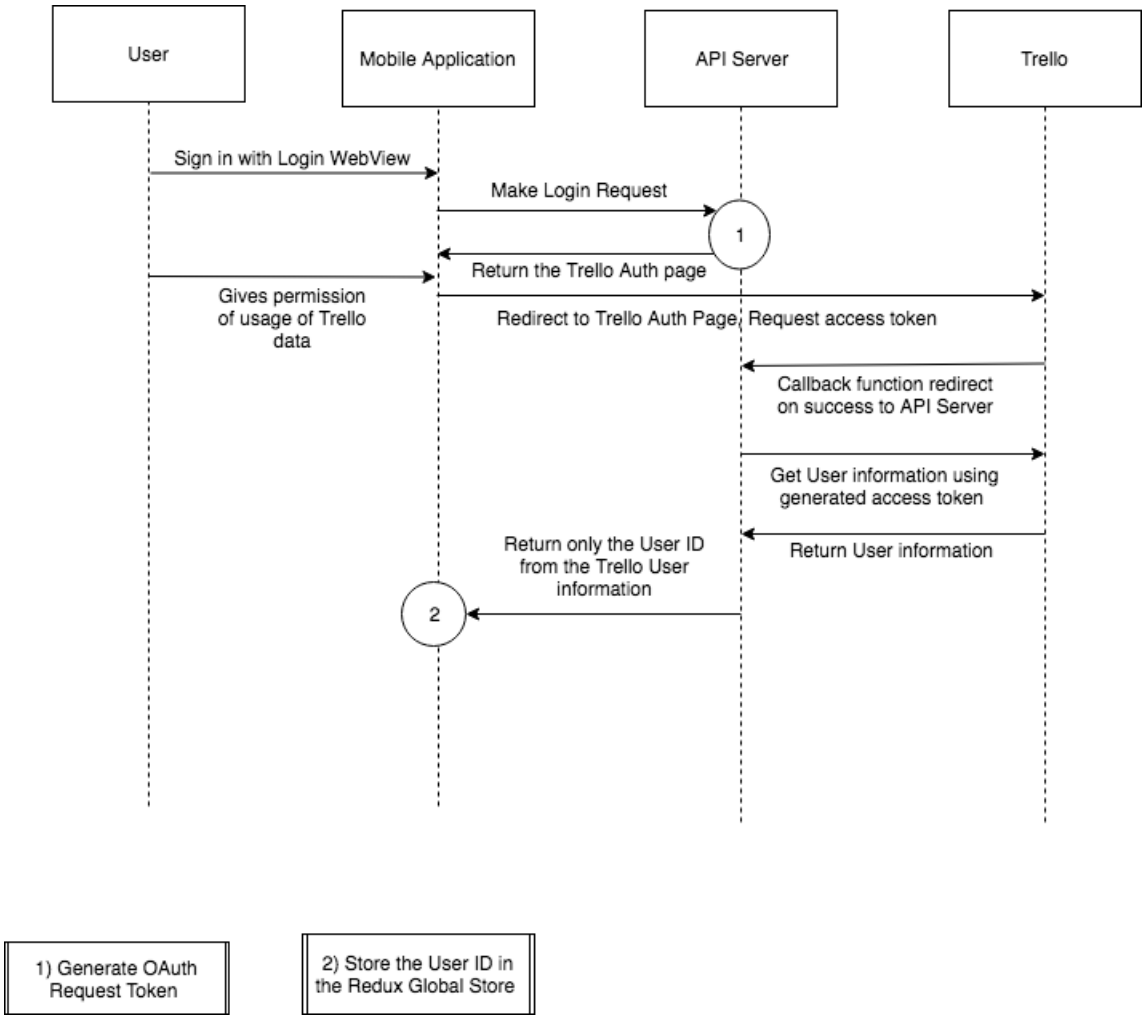



Figure 4.12: OAuth Flow

Previous iterations of the user identification system included utilising a QR code (requested by the client) to identify a particular board. However, this was not feasible as in order to scan a QR code, the camera is required. The camera is already in use for the augmented reality aspect of the application and the Viro React library used for augmented reality conflicted with multiple QR code scanner libraries we tried to integrate into the application.

The other notable identification system we tried was to use a direct mapping between a 6 digit code and an ID of the Trello board which the user was working on for a project. The 6 digit code was chosen as it allowed a significant number of mappings while being easier to remember than a board ID e.g. "5beafbd0d57a070d6a0b2141". However, it was cumbersome to load the mappings onto the database from the user point of view and storing this information overlapped with accessible Trello data. Therefore we removed this feature and choose the aforementioned method.

### 4.4.5 Database

The mobile application mostly polls data from Trello to visualise in augmented reality. However, we wanted to help our clients improve their organisational Kaizen usage. To achieve this, we made a sep-

arate database to store and fetch all the Kaizen improvements they tried on every project with the information on how effective that was, enabling our clients to freely store user-generated data.

The Schema of the Kaizen Improvements Table was designed to be minimal but it is also open to further extensions.

The Trello board ID, as well as the board name, is accessed from the Trello API board data which is then mapped to the improvement(Kaizen changes) and the status of the improvement. Future iterations of the improvements table could map each improvement to the team that worked on it by adding a team column.

| | id [PK] integer | board_id character varying (100) | improvement text | status character varying (100) | board_name character varying (100) |
|---|---|---|---|---|---|
| 1 | 1 | 5bbb7e4006d2af393fc53e4d | Create test rating table | Useless | WeWeregonnatest |
| 2 | 2 | 5bbb7e4006d2af393fc53e4d | Create documentation | Useful | WeWeregonnatest |
| 3 | 3 | 5bbb7e4006d2af393fc53e4d | Increase team size | Testing | WeWeregonnatest |
| 4 | 4 | 5bbb7c774791850822e10ac5 | Daily Stand Ups | Useless | Wewregonnaplan |
| 5 | 5 | 5bbb7c774791850822e10ac5 | Pair Programming | Useless | Wewregonnaplan |
| 6 | 7 | 5bbb7c774791850822e10ac5 | Daily Standups | Useless | Wewregonnaplan |
| 7 | 8 | 5bbb7c774791850822e10ac5 | Slack Communication | Useless | Wewregonnaplan |
| 8 | 9 | 5bbb7c774791850822e10ac5 | Trello Management | Useful | Wewregonnaplan |
| 9 | 6 | 5bbb7e4006d2af393fc53e4d | Pre iteration planning | Testing | WeWeregonnatest |
| 10 | 10 | 5bbb7e4006d2af393fc53e4d | Do Work | Useful | WeWeregonnatest |
| 11 | 11 | 5bbb7e4006d2af393fc53e4d | Dont do work | Useless | WeWeregonnatest |
| 12 | 12 | 5bbb7e4006d2af393fc53e4d | Code Review | Useful | WeWeregonnatest |

Figure 4.13: Kaizen Improvements Table Format

Previous iterations of the database stored the user id mapped to the board ids that the user has. This table was made redundant upon implementation of the Trello login screen which directly extracted the user id and stored it locally within the Redux global store during an instance of the mobile application. From this user id, we then are able to access the appropriate data from Trello API. We made a design choice to minimise the personal data of the user that we were storing to avoid any potential conflicts with the GDPR.

# Chapter 5

# Evaluation

## 5.1 Implementation Testing

For the testing of our features on our iterations, we divided our app into Front End and Back End and tested on each horizontal slice. Our Back End mainly fetches the requested data from our Trello board or generates a project management analysis. Our Front End collects the fetched data from the Back End and displays it in the right format in the AR platform. Due to the Augmented Reality component of our app, we couldn't automate the tests on the Front End but we could make unit tests on the Back End and do the iterations based on Test Driven Development(TDD). We also generated a dummy Trello project management environment for testing to simulate our client's situation where we have to integrate physical and digital components together to provide better project management and analysis.

### 5.1.1 Testing on Front End

Since our application is an augmented reality(AR) application, we always needed a full mobile device with a camera to test our Front End, and it is meaningless to test our app on an emulator. However, it is inefficient to generate an APK every time we make a change on our Front End, so we decided to use the AR TestBed app that was provided by our library: React VIRO Media TestBed App. We tested our product using this application by first implementing the display of user interface with dummy data and testing on the VIRO TestBed app. Then we integrated the Back End calls and redid the testing with real data.

#### Supported Devices

To test our app, we needed phone devices that support ARKit or ARCore. For iPhone, this means the testing device has to be iPhone SE, iPhone 6S or above. Unfortunately, not all our devices supported ARKit or ARCore, and we needed some support from the department to get enough devices to engage all our teammates to make our development process more efficient.

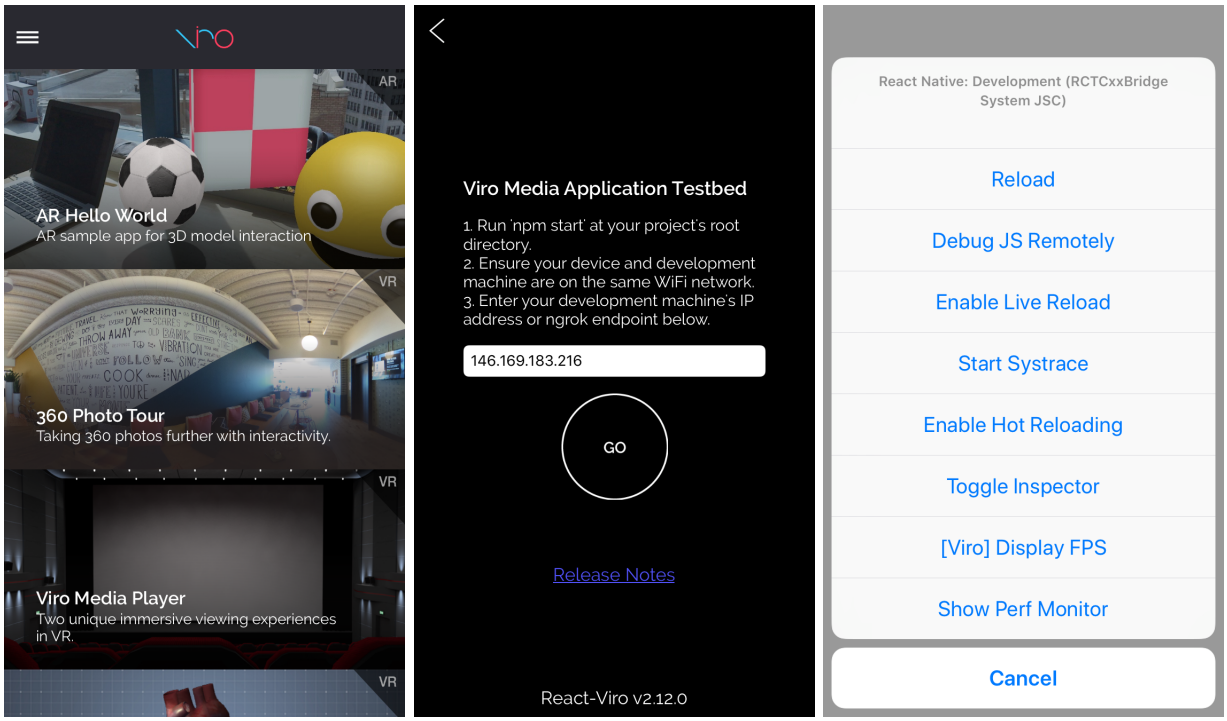
#### React VIRO TestBed App

One of the key features of React VIRO is that it supports testing on the actual device without recompiling or packaging the code by providing its TestBed App. The TestBed app can be accessed through a sub-page of React VIRO Media application which we can download through App Store or Google Play Store. The changes we made on our app is shown on the testing device when entering the local IP address of the developing environment while connected to the same network. It can also be accessed by the ngrok link that is shown on our terminal when we run npm start. However, we prefer using our local IP address because the ngrok link changes every time we rerun npm start and our team found it inefficient to retype the ngrok link.

One of the biggest advantages of using React VIRO Media application is that it supports React Native's reloading features even on the AR environment, enabling faster iterations without recompiling. After entering the test environment, we could bring up developer menu to not only let us reload the app, but provide different development features that we found useful such as 'Enable Live Reload' and 'Debug JS Remotely'. 'Enable Live Reload' would detect any saved changes on the Javascript file tracked by the

package server and reload automatically, speeding up the iteration. 'Debug JS Remotely' connected the application to a remote console on the browser to show any console reactions on user interactions inside the app.

Figure 5.1: React VIRO Media application




### 5.1.2 Testing on Back End

Test Driven Development was used to make sure that our Back End works as per our specification. On each iteration, after the meeting with our client, we would have our internal meeting with our teammates to discuss how we would divide the work. During this period, we analyse each feature that our client requested and decide which Back End features we need. Based on those needs, we generated simple tests that reflect the requirement and allocate people to implement the feature and have the test cases passed.

We used the SuperTest and Mocha frameworks to create a testing suite for the Back End. We decided to use Mocha since it is a JavaScript test framework that makes asynchronous testing easy.

The testing suite focuses on testing the API calls made within the Back End to the Trello API. An example unit test would be testing our API call to grab the Trello board via the board id.
The test cases can all be automatically run on the server using one of the scripts we wrote. So for any modifications/additions made on the Back End all the tests can be run automatically to ensure the desired functionality is achieved.

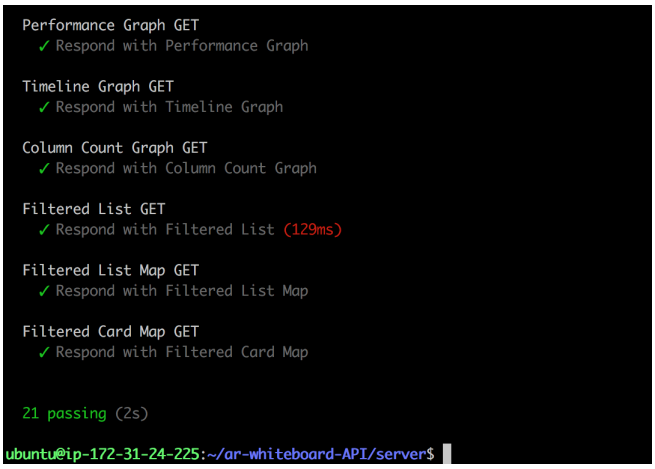
We have also made a dummy project management environment so that we can emulate our client's situation. A dummy Trello board was created for testing purposes to effectively test the RESTful calls interaction with the Trello API and also the client app.

Some features, however, were difficult and inefficient to implement automated unit tests. One of our features was to automatically analyse our client's project progress through graphical representations. In

this case, it was difficult to check the newly generated graph to match the expected one. For these features, the automatic unit test would check the RESTful API was a success returning the expected format and we manually used a web browser to check the graph is as expected.

Figure 5.2 shows the screenshot of the automated testing script being run:

Figure 5.2: Test script generated before feature implementation




## 5.2 Deployment

### 5.2.1 iOS Deployment

We faced a lot of technical challenges for the deployment. We were planning to use the Microsoft App Centre but it doesn't work with iOS without a proper distribution certificate. Since the Apple university program only allows us to deploy to our own device, it doesn't give us a distribution certificate. It was too expensive to pay for a proper account so we decided to just deploy to our own iPhones.

### 5.2.2 Android Deployment

We have used the Microsoft App Centre to deploy our app on Android. It generates an apk after testing on some devices. We will then distribute the apk within our team and try it out on our own Android devices.

## 5.3 Project Progress Evaluation on each sprint

The evaluation of project progress was generally done on sprint backlog meetings with our client. At these meetings, we evaluated ourselves on how much we delivered the last sprint's backlog in relation to how difficult each task was.

On each sprint, we generate a new Trello list on our project board and list the new features we have to implement. On each card, we subdivide each task into smaller checklists. Our project progress evaluation starts with checking how many checkboxes/Trello cards are undone and proceed by analysing our time management compared to our client's needs. This evaluation process on each sprint ensures that we are dedicating our time to enhancing our product in a way that would help our target users.

### 5.3.1 The three analytical approach

We decided to analyse our Project Progress from the following three different perspectives to maximise our project efficiency on each iteration. We integrated these three perspectives to conclude with the final Project Progress Assessment.

#### Objective observation of Project Progress in metrics

For evaluating the Project Progress in using objective metrics, we calculated how many tick boxes or backlog cards were undone at the end of each sprint. However, we understand that this isn't a very precise way of evaluating the project progress so we analysed the project progress in two different aspects for the conclusion.

#### Open Project Progress discussion in Developer's Perspectives

We had an open discussion on how the project progressed on each sprint. These discussions usually reveal how much time we spent on each sprint backlogs. We needed this process to ensure that we were not spending too much time on something that is not worth the time.

For instance, during our second iteration, our client asked us to implement a feature that would enable them to access the virtual board through QR code, later after we assigned a pair to work on this task, we found out that our AR library: React VIRO and the library we were planning to use: RC Camera required different versions of React. It was too time and human resource consuming to fix this problem line by line when we can simply suggest to our client a substitutable feature.

#### Open Project Progress discussion in Client's Perspective

Our client had an open discussion with us on each sprint's project progress. In this part, we evaluated how much each backlog was worth compared to the time and effort that was spent on it. This process is to make sure that our time is dedicated to making the product better to help our users.

Using the aforementioned QR code incident, we found out that our client did not place too much significance on being able to access the virtual board through the QR code as long as there is a mechanism to access the board. Through this process, we confirmed that replacing the QR code feature with a board pin mapped to a board ID is much more efficient in terms of developer time.

## 5.4 Project Method Evaluation on each sprint

Our project method evaluation was done on each sprint meeting by conducting an open brainstorming with memos on the whiteboard. Each team member had to write post-it notes on what they liked about this sprint, and what they disapproved about this sprint. Based on this discussion, we reevaluated our methods and updated it. This process was essential on detecting methodological problems that were slowing down the work.

For instance, our first sprint had a problem integrating the features into one product because we divided the work based on features and each pair worked on different features on different branches. The branches were not merged into master often enough and by the end of the sprint, it was difficult to merge everything into one. This problem was flagged by several memos on the whiteboard to indicate that it was a shared problem with our project management method. At the end of iteration one, we discovered the cause of the problem quickly and updated our method planning on version control to avoid making the same mistake for future iterations.

## 5.5 User Experience Evaluation

As our client was also our user, we were able to have the chance to collect direct user feedback on each iteration and update our project backlog based on such feedback. Thus, it enabled us to achieve higher final user approval. In addition, we also demoed our application to other 3rd year software engineering groups who also used Trello as their agile board in order to gain a more diverse range of feedback without solely relying on the client. This enabled us to get another perspective to the use case of the application which helped develop features such as the Trello commenting system.

### 5.5.1 User Experience Evaluation on each Iteration

On each sprint meeting, we met our users / client – Theodo developers – and collected their feedback in an open manner. We formed out next sprint backlog based on this feedback, forming a continuous user feedback structure of our project. The main user feedback on each iteration is as follows and we updated our sprint backlog based on this feedback.

| Iteration | Open User Feedback |
|---|---|
| 1 | • "It would be convenient if I can access the digital board in our AR app by detecting a QR code"<br>• "Right now, we track our project progress by manually updating the Kaizan board with markers. It would be great if this app can automatically track project progress in detail through graphs"<br>• "It is difficult to see the board without moving when the board columns are spread out widely. Can you collapse it and make it scrollable?" |
| 2 | • "We want to see how each tasks(Trello tickets) are done in detail. This data is helpful on understanding our projects. It would be wonderful if we can track the Trello card status based on time." |
| 3 | • "Right now, whenever there is a problem in a task(Trello card), we write a comment on the Trello card, reprint the tickets on paper and clip it onto the Kaizen white board. It would be great if this app can let us post and show the comments of Trello tickets."<br>• "I don't think someone can use this app directly without any explanation right now. It would be more user friendly if there is a tutorial when I start using this app" |

We requested feedback between each iteration to improve our app. An example of our process is the development of the board display. At our first iteration, we displayed all of the Trello board list columns horizontally in AR. Then we received feedback that the user would have to walk a long way to read all of the lists from left to right. We then implemented the viewport fixed AR Menu in the next iteration, which allowed users to choose a Trello Card within the same menu area.

### 5.5.2 Final User Experience Evaluation

For our final user experience evaluation, we wanted to make an objective and metric-based assessment to evaluate how we improved our user journey, therefore we made a survey. We used this survey to collect

the evaluation data from our end users. We formed the question that has metric answers and averaged our users reply to evaluate our project in final.

**How often do you encounter the difficulties that can be fixed by our app?**

Average reply: Daily

**In the scale of 1-10, how useful do you find our app?**

Average reply: 8

**How much time does our app save you on a daily basis?**

Average reply: 20 minutes a day

**On the scale of 1-10, how likely do you think you will use our app in daily basis?**

Average reply: 7.32

We also conducted user interviews on our application and these are the comments we got:

**"I had to print out Trello tickets every day for any small updates. With this app, I don't have to print out the tickets anymore. This can save much of our time and also save paper.**

**"This app can save at least 20 minutes everyday on each board. There are 10 teams with different boards in this company and that means it can save 200 minutes everyday for our company."**

**"We have been manually tracking project progress by updating the bar graph on whiteboard with markers. It was difficult to keep the graph in scale and visually tidy. Now I don't have to do any of that anymore."**

Based on these survey results and user interviews, we formed the improved user journey which is attached in Appendix A.5. Before the development of this application, the users had to print out the Trello tickets on paper, find the changes and highlight those changes to make it easy to find and then clip it onto the magnet and attach it onto the Kaizen whiteboard.

Now, with this application, the users can simply use their phone to enter into the AR platform that shows all the Trello tickets. Any changes can be digitally marked and easily accessed through tag and filter feature. See Appendix A.4 for the user journey utilising our product.

Appendix A.3 shows the user persona used in for the user journey.

## 5.6 Deliverables

Through our application, the users were be able to:

**Easily locate relevant information:** Users can use the app's AR interface with the physical Kaizen board allowing all the data to be viewable at once. Some of the information (e.g. Idea brainstorming notes, tables, etc) would be more convenient to display on a physical board while other data (e.g. task management, technical discussion, ticket list) would be better displayed as part of the AR interface of the app. Our application allows for a best mix of data visualisation between the physical and virtual worlds.

**Increase Productivity:** Minimise time in data processing for project management (e.g. generating metrics, counting tasks completed each week, making a graph to visualise project process. etc) allowing team members to spend more time on other project tasks. For example, reduce trivial work time of reprinting out the Trello tickets to stick to the whiteboard whenever there is a change.

**Higher Resource Efficiency:** Trello tickets no longer have to be printed out, significantly reducing paper usage and costs. This allows for higher resource efficiency especially since time, financial and resource usage are reduced especially since for each board around 20 pages of tickets had to be printed out.

# Chapter 6

# Conclusion

## 6.1 Lessons Learnt

For a lot of us, it was not only our first foray into AR development but mobile development in general. As most of us had previously encountered web development in the form of React, moving from React to React Native was not as challenging as we thought it would be. However, developing on mobile comes with significant challenges. For example, understanding the hardware limitations of a native phone application and its capabilities compared to other platforms. We elected to do a lot of computation heavy features on the Back End server, sacrificing speed for performance.

The interaction between different libraries also gave us another problem. In the earlier iterations of the app, we were able to generate an APK file for Android deployment. However, as development continued and we added more libraries to our app, the APK generation would start to fail more frequently. It was very hard for us to pinpoint the cause of this error. Another issue we had with the interaction between libraries stemmed from how different libraries that used the phone camera interacted with each other. This lead to us wasting time trying to implement the QR code scanner as our main form of Trello board identification.

From a project management perspective, we found trying to follow the Scrum ideology difficult at times. This was because of timetable conflicts, and the stacking of quite a few coursework deadlines from other courses. It was hard for us to split up work between all six of us at the beginning of the project, as there was a lack of communication between group members at times. This was mainly because some of the group members hadn't worked with the others before. We tried to fix this by doing daily online stand-ups on Slack using the standup bot Standuply. It was difficult to ensure everyone did this everyday, but along with biweekly in-person meetings, we as a group found a stride slowly as the second iteration started.

## 6.2 Future Extensions

Our main goal for this app was to transform data retrieved from an existing Agile framework into useful graphics and visualisations to aid project management. Currently, our app is only able to retrieve data from Trello, therefore for future extensions, the application can be extended to interact with other frameworks such as JIRA and Asana which both have APIs that can be made use of. Due to our modular design philosophy, this extension would be easy to implement as you would only have to switch out the Back End module.

In our second iteration, we mentioned that we wanted to add Google Sheets integration so that users could add graphs in Google Sheets and display them in AR next to the board. This feature was not implemented due to product feature priorities and time limitations. Since our Back End was designed to be easily extensible, adding new API modules would not require too much modification of existing code.

Another point of interest is the Trello inbuilt feature called the Power-Up. These are free JavaScript plugins that can be developed just for Trello. Future extensions could take advantage of the Power-Up to allow for more interaction between Trello and the app. To add more interaction between Trello and our app, we would write a Power-Up for Trello that also interacted with our Back End API server. We could

allow users to save the graphs that were generated during each use of the app, and attach them to the relevant tickets after each Kaizen session. We could also give users the function of creating the graphs from within Trello without using our app, by directly calling our Back End API with a Trello Power-Up.

We could also introduce caching data in the app and in the Back End server. As our data only changes if the board changes, we could set up a webhook and only make new calls to the Trello API when our board data changed. Otherwise, we would access our cached copy instead, allowing for higher performance at the cost of native and/or serverside memory.

# Chapter 7

# Bibliography


[1] *The Ultimate Guide to Understanding Augmented Reality (AR) Technology*
https://www.realitytechnologies.com/augmented-reality/

[2] *Augmented Reality (AR)*
https://www.techopedia.com/definition/4776/augmented-reality-ar

[3] *AR*
https://docs.expo.io/versions/latest/sdk/AR

[4] *Viro AR*
https://viromedia.com/viroar/

[5] *React Native Copilot*
https://github.com/okgrow/react-native-copilot

[6] *React Native Swiper*
https://github.com/leecade/react-native-swiper

[7] Project Management Institute: *Project Management Institute Report 2016*
http://www.pmi.org/-/media/pmi/documents/public/pdf/learning/thought-leadership/pulse/pulse-of-the-profession-2016.pdf

[8] *Insights and Trends: Current Portfolio, Programme, and Project Management Practices*
https://www.pwc.com.tr/en/publications/arastirmalar/pages/pwc-global-project-management-report-small.pdf

[9] *What Is A Kaizen Board?*
https://leankit.com/learn/lean/what-is-a-kaizen-board/

[10] *Redux Documentation*
https://redux.js.org/introduction/three-principles

[11] *Oauth1 Documentation*
https://oauth1.wp-api.org/docs/introduction/OAuth.html

[12] Jeff Roussel: *3 Reasons Kaizen Boards Often Disappoint*
https://blog.kainexus.com/improvement-disciplines/kaizen/kaizen-boards/3-reasons-kaizen-boards-often-disappoint

[13] Shmula Contributor: *Digital vs Physical Kanban Boards*
https://www.shmula.com/digital-vs-physical-kanban-boards/22755/

[14] *Introduction*
https://developers.trello.com/v1.0/reference#introduction

[15] *Testbed Apps*
https://docs.viromedia.com/docs/develop-with-viro

[16] *React Native Router Flux*
https://github.com/aksonov/react-native-router-flux

[17] Michael Huskins, James Kaplan, and Krish Krishnakanthan:
*Enhancing the efficiency and effectiveness of application development*
https://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/enhancing-the-efficiency-and-effectiveness-of-application-development

[18] *Getting Started with Trello*
https://trello.com/en-GB/guide/trello-101

# Appendix A

# HCD Techniques

## A.1 UX testing script

Hello there. Are you free to answer some short questions on our application. My name is "Tester" and joining me is "Moderator". I have asked him along to take some notes as we talk. I hope that is okay. We would like you to use a working prototype. The idea of this meeting is to see if we can improve an app that is currently under development. You are going to help us test the app. It's important to understand that we are testing the app and not you. So you can relax! Here is the app. First off what do you think? What are your initial impressions of the login screen? What do you think the app does?

—-Carry on questions with specific feature to UX test——

UX test on AR menu view:

Can you tell me whether you prefer the menu to expand horizontally when a button is pressed?

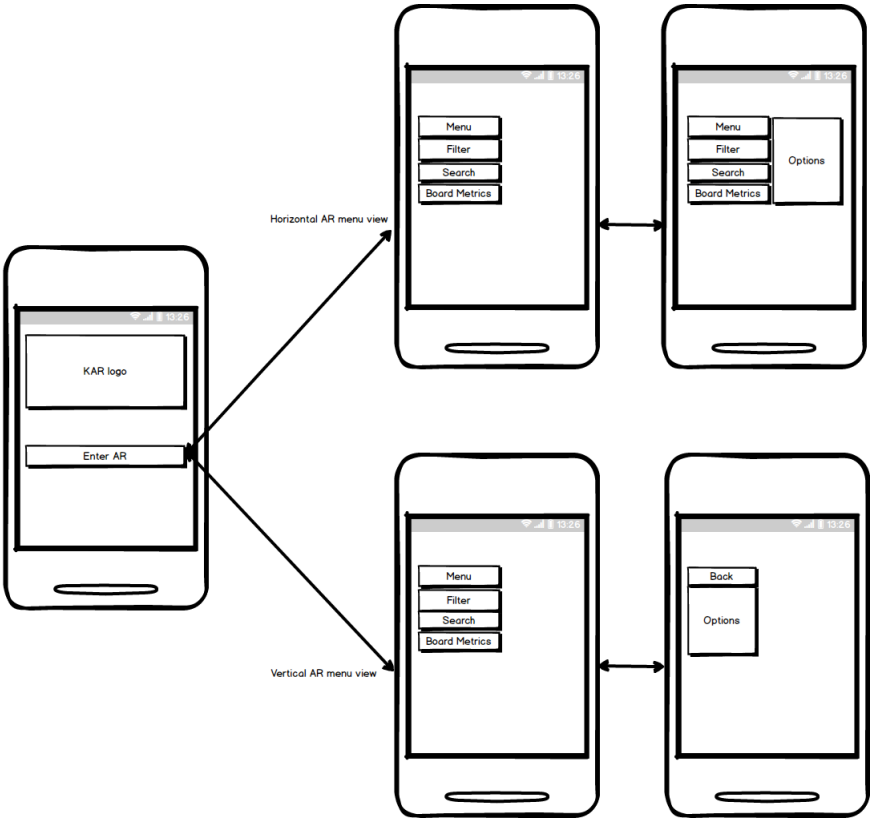



Figure A.1: Menu View Mockup

UX test on authentication login screen:

Would you prefer to have a unique pin to enter to retrieve your Trello information or to authenticate by logging into your Trello account?

The most important thing to remember is that we need you to explain what you are thinking. Try to think out loud and talk about the various options you are considering. Before you click on any link

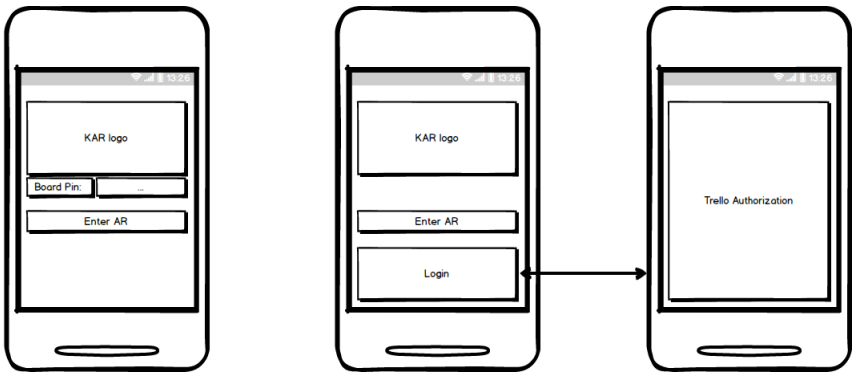



Figure A.2: Authentication Mockup

explain what other options you considered and why you picked the one you did.

—-End of UX test—-

Thank you for your help. Finally, if you have any questions or any additional comments to add please feel free to ask.

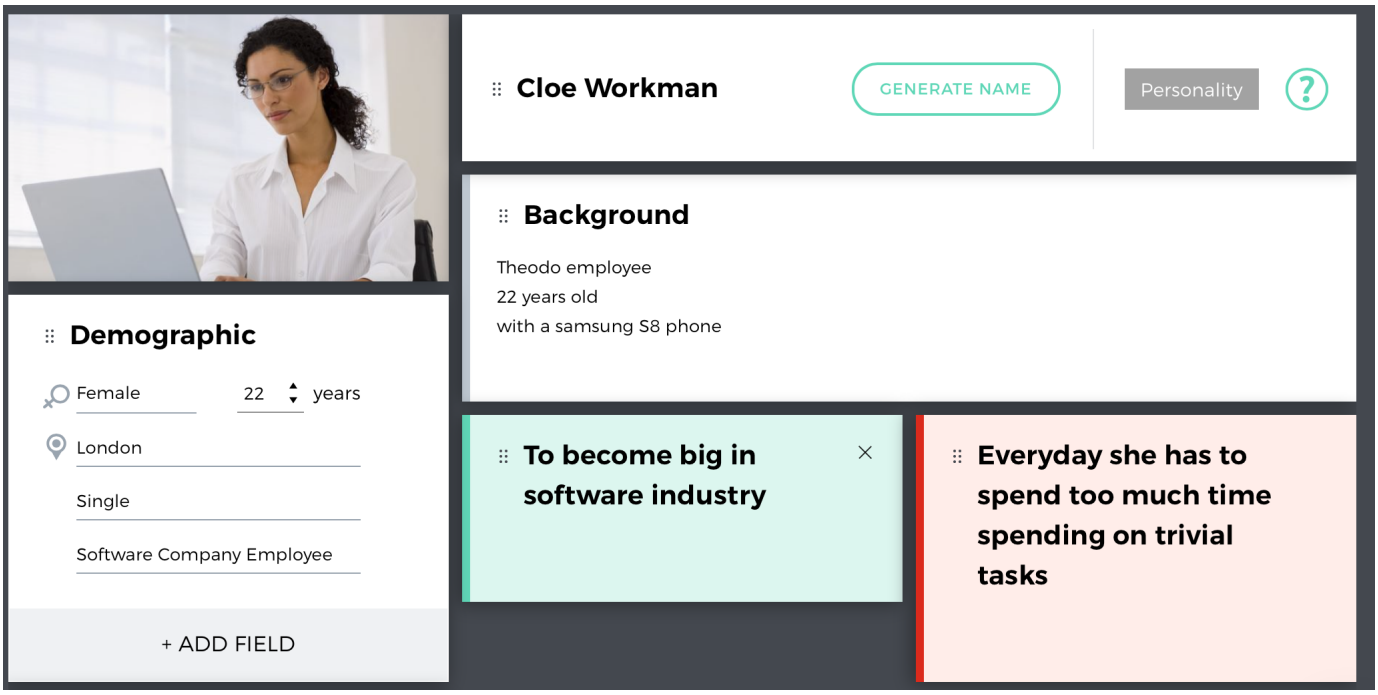



Figure A.3: Persona we generated to represent our users

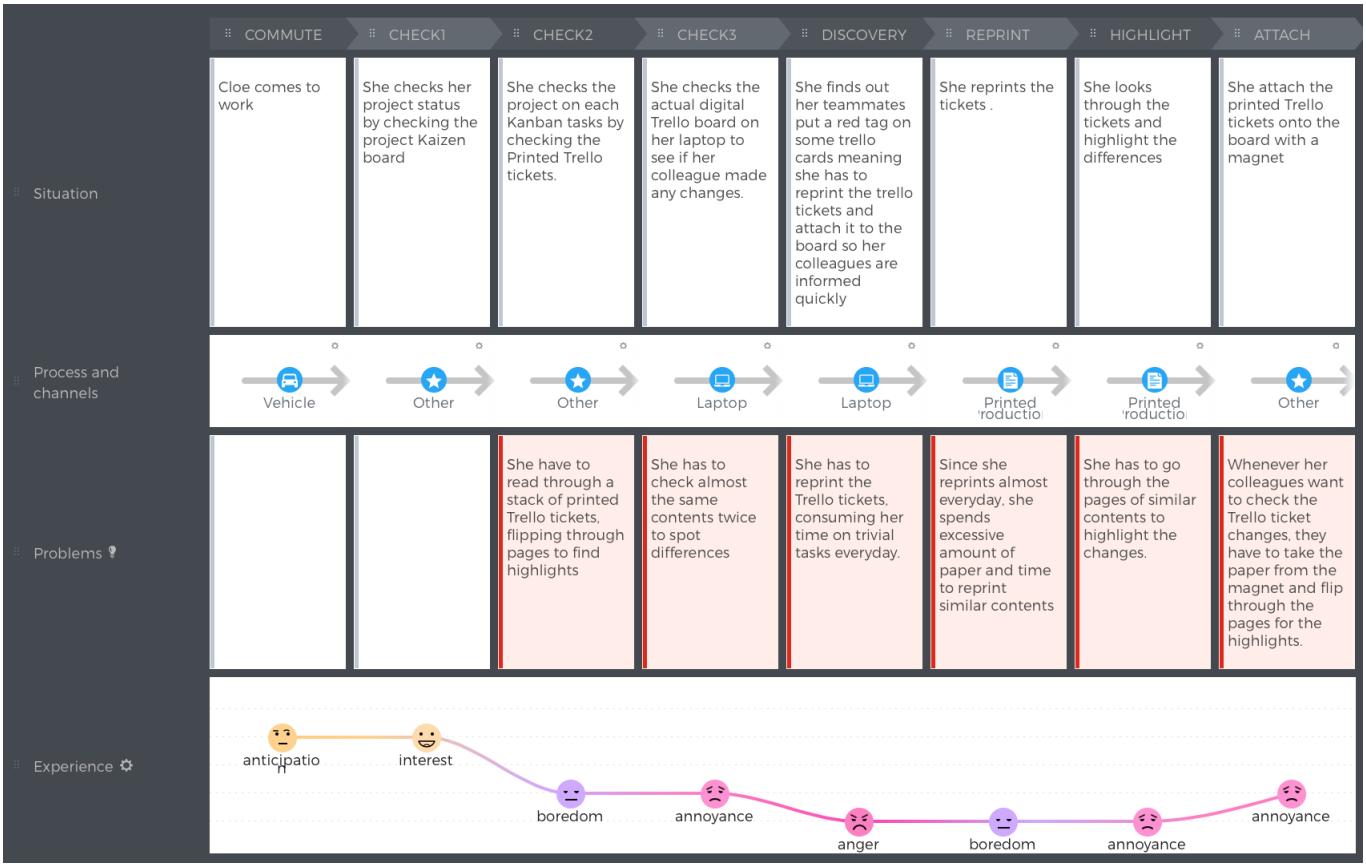



Figure A.4: User Journey Without KAR

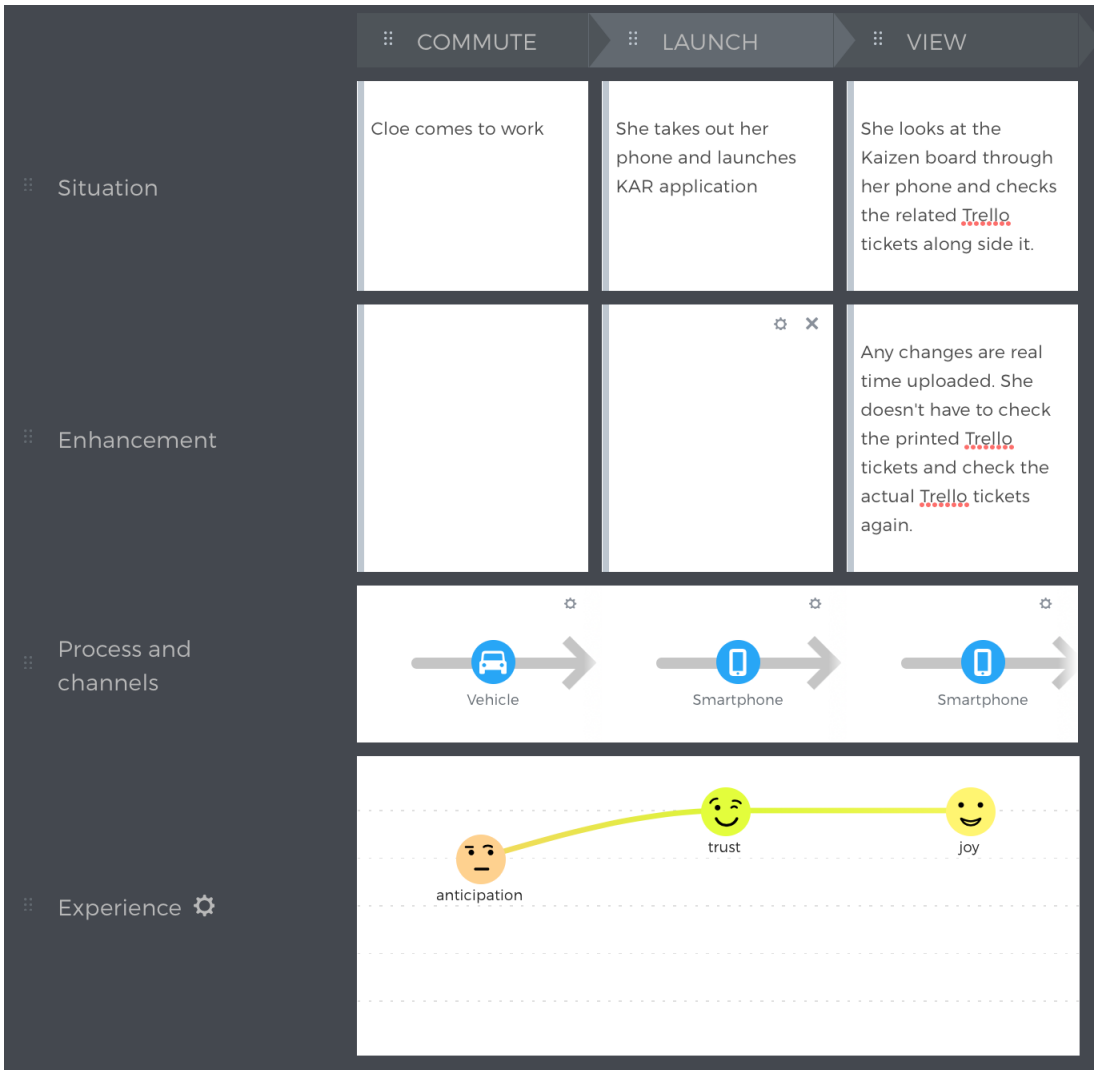



Figure A.5: User Journey With KAR

# Appendix B

# User Guides

## B.1 General User Guide

Load the application on your android phone or iOS phone which will launch the application to the landing page. See Appendix C.7.

Follow the initial walk through of the contents of the landing page which can be skipped. See Appendix C.8a.

From the landing page you have to first Login in with your Trello account via the Login Button. This will navigate you to the Trello login page within the application where you will enter your Trello login details upon which you will be returned back to the Landing Page. See Appendix C.6.

Once you have logged in you can now access the Augmented Reality Interface using the Enter AR button which should now also display your name.

To use the Augmented Reality Interface, you must point your phone camera at the marker which you will have placed on the top right corner of the Physical Kaizen board. On a successful scan, the AR Menu will pop up to the right of the marker.

## B.2 Team Member Specific User Guide

From the AR Menu you can choose the following options. See Figure 4.5.

1. *Filter Options*
   Allows you to filter the Trello Card results based on a Trello Label assigned on your Trello board by displaying the Trello Labels. See Appendix C.2.

2. *Search Options*
   Find a Trello Card to display additional information. You search first through the Trello boards and from the chosen Trello Board, you select a Trello List. Finally upon choosing the Trello List you can select a Trello Card from the list which will bring you back to the AR Menu with the chosen Trello Card title shown in the search options button. The Card Detail View and Card Timeline View will also pop up for the chosen card.See Appendix C.3.

3. *Card Comments*
   Allows you to communicate with team members and also flag certain cards. When a card has been selected via the search options sub menus, you can the see the comments on the card as well as post comments. If you want to refresh the comments pull down the screen from the top. See Appendix C.5.

## B.3 Project Manager Specific User Guide

From the AR Menu, a project manager can also access the Board Metric Sub Menu which allows the manager to see various metrics about the project. The metrics available are the cards per column, the

performance of the group and the tickets that are overdue. See Figure 4.9.

From the landing page, a project manager can access the Kaizen Improvement page to see the state of the improvements measures across all projects in the organisation as well as add an improvement. When adding an improvement, to add to a project, the manager can select a board from the AR Menu Search Option Sub Menu. To refresh the improvements, the manager can pull down from the top of the Kaizen Improvements Page. See Figure 4.10.

The card detail view pops up on the right of the physical board when the manager clicks on a card from the menu. The manager can drag the view to the position that at which it is comfortable to read. Then the manager will be able to see information of the specific task like members responsible, number of days until due date, sub-tasks completed, etc. The view will get updated once the manager clicks on a new card. See Figure 4.7.

The card timeline view pops up on the left of the physical board when a ticket is selected. This view can also be dragged to a comfortable position. It displays a graph with bar components that can be selected to see how long the selected ticket has been in each column.See Figure 4.8.

# Appendix C

# Images


Figure C.1: Commit Message Example


Figure C.2: Filtering Feature

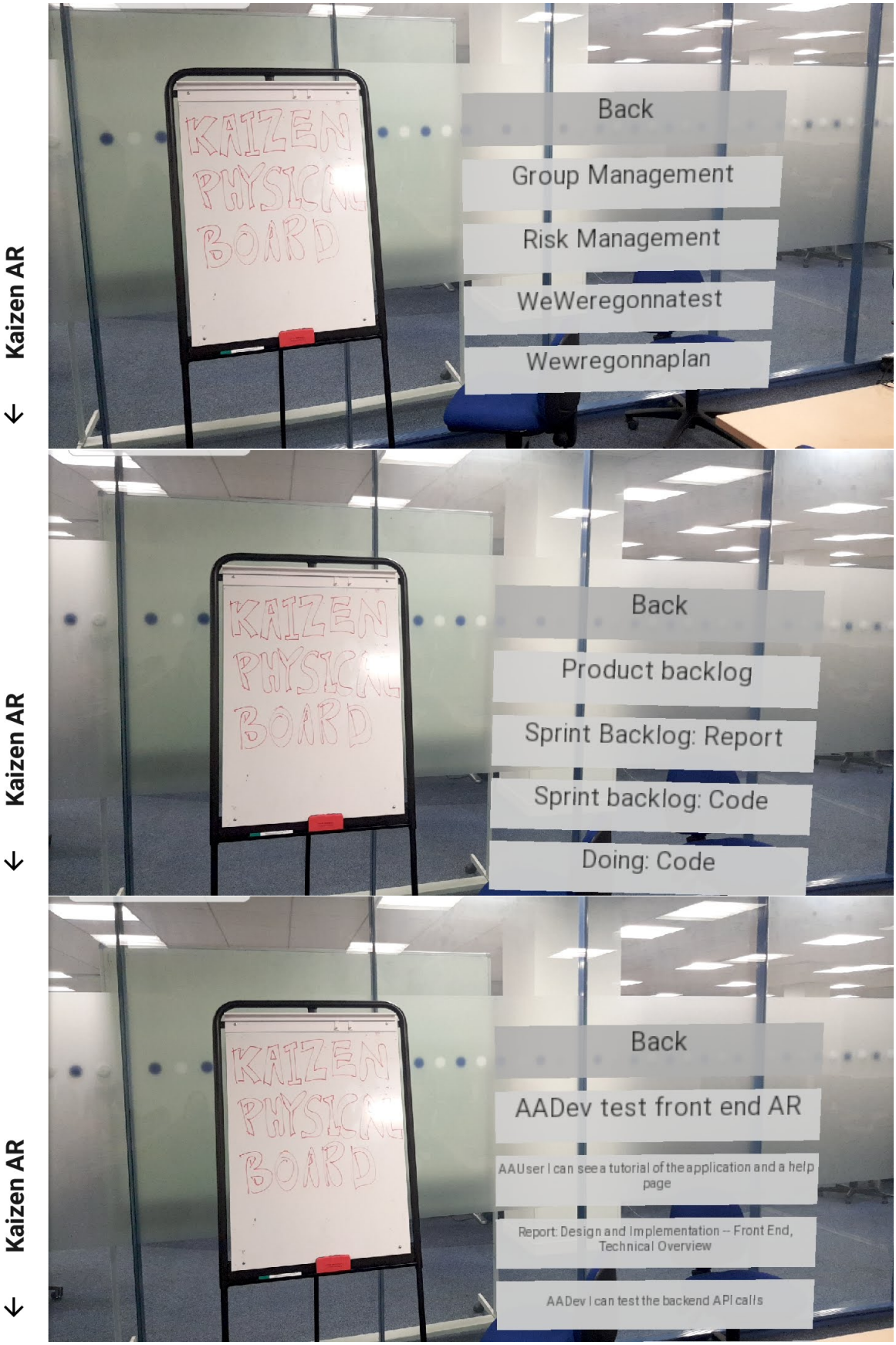



Figure C.3: Search Menu Generated Options: Board View, List View and Card View shown from top to bottom


Figure C.4: Overdue Card Example

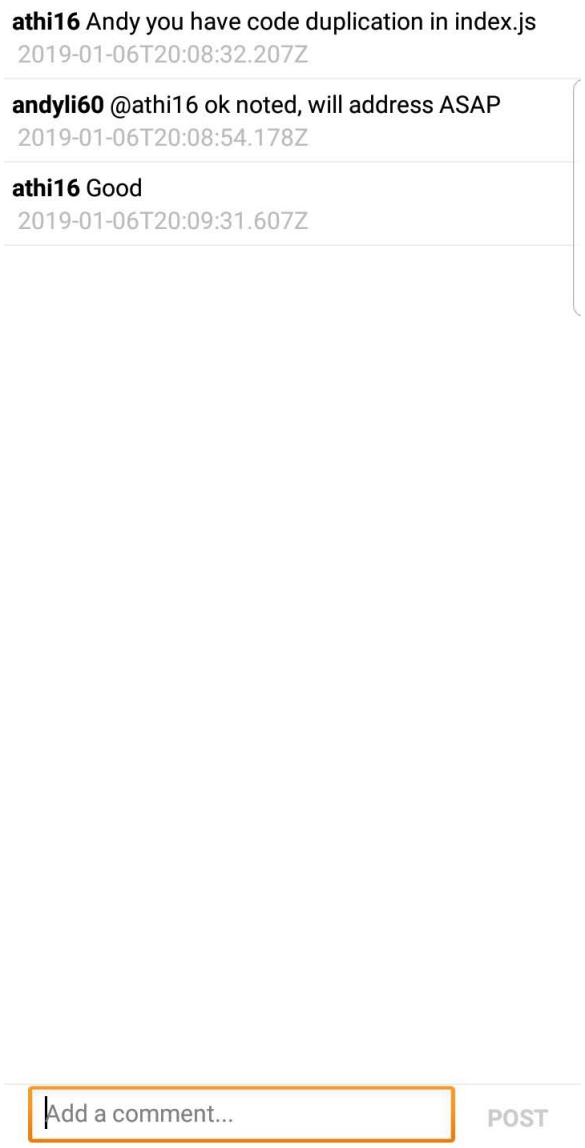



Figure C.5: Comment Modal

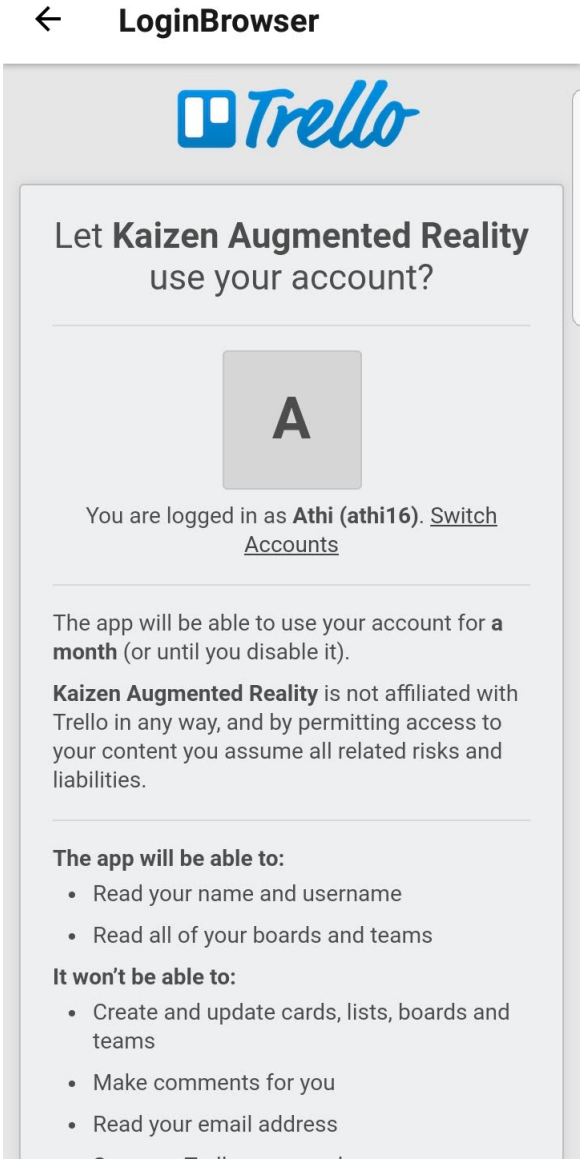



Figure C.6: Login Oauth Screen using React WebView

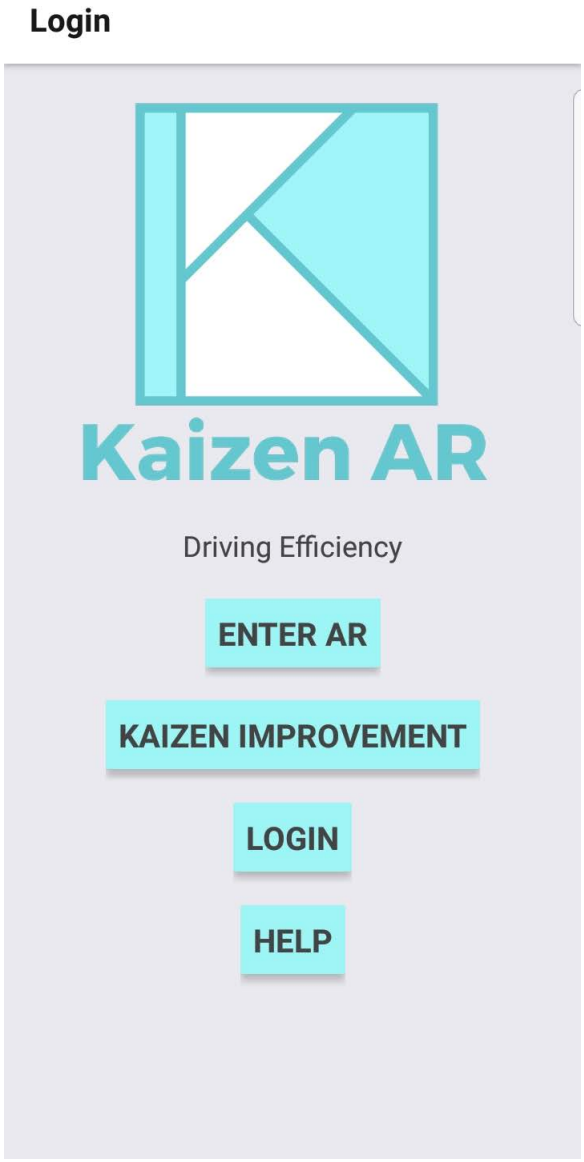
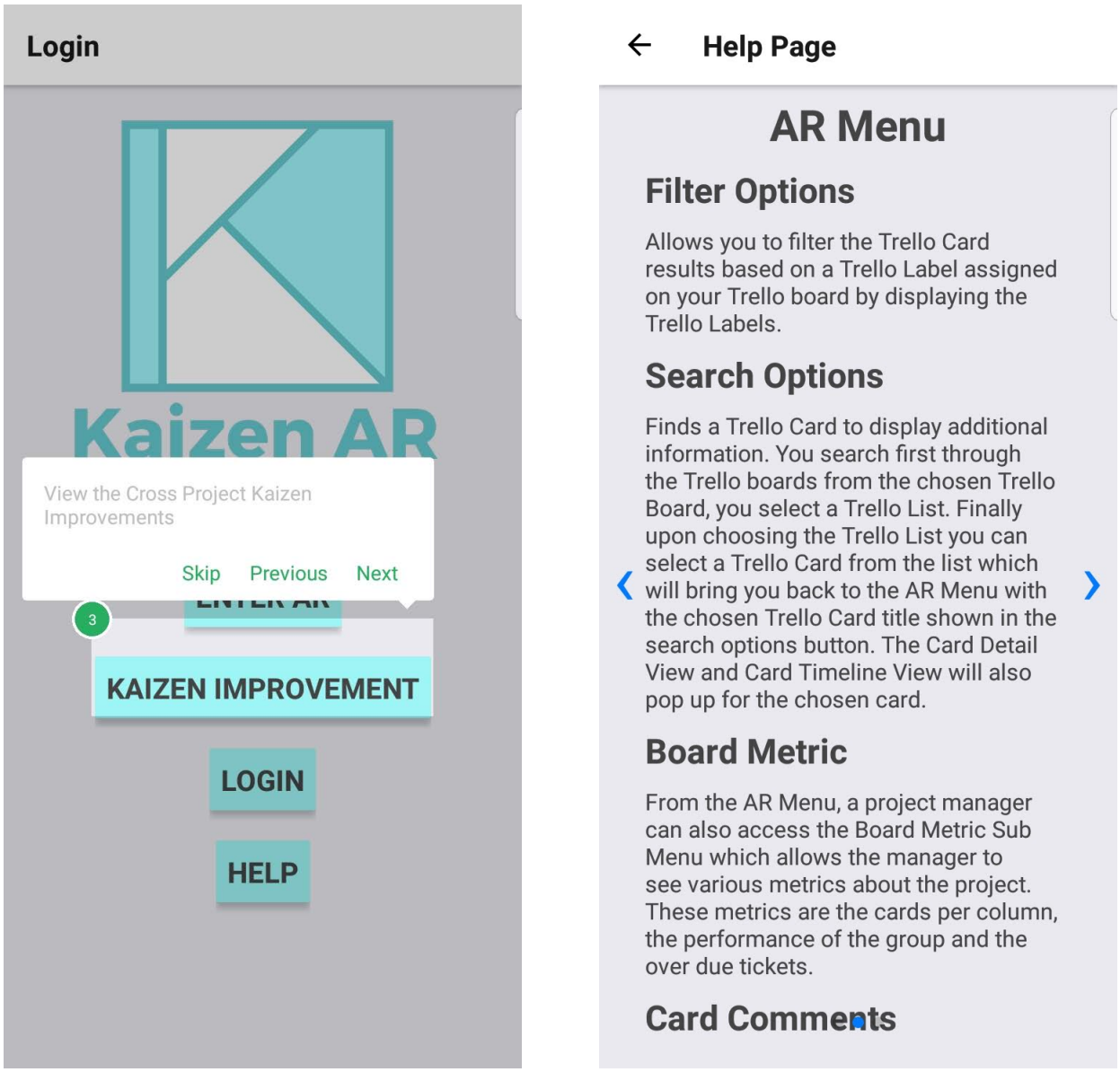



Figure C.7: Login Screen




(a) Login Screen with Tutorial

(b) Help Page

Figure C.8: Login page and Help page