UniD

Product Requirements Document

Team Worthday

Jordan Ang
Alan Tran
Arthur Pan
James Yang
Nathan Vandervoort

Table of Contents

1. Introduction

Mobile app that incorporates and extends the functionality of the UCSB access card using the Workday Student cloud infrastructure in a convenient platform.

1.1 Problem

Students sometimes are forgetful or clumsy and drop their access cards. As a result, the students unable to make purchases using their student ID or swipe into the dining commons without having to manually confirm their identification. The process of manually confirming their identification is slow and creates traffic. Another issue with physical cards is that over time the image gets worn out or the magnetic strip is unable to be read by the card swiper. This also presents an issue of identification because the administrator can not confirm that the user is the cardholder.

1.2 Solution

UniD, a mobile student identification application, plans to digitize student identification cards using QR codes. Using a mobile solution prevents students from having to carry a card around. Everything can be accessed through the mobile device. Also, other people can not use the owner's QR code since a picture of the owner will appear when the code is scanned.

2. Technical Advancements and Innovation

2.1 Web Technologies

Previous web frameworks such as LAMP (Linux, Apache, mySQL, PHP/Python) are outdated in the modern era. They take too much times to set up and manage, especially when handling hundreds of thousands requests. Consequently our web framework will use the MERN stack (Mongo, Express.js, ReactJS, Node.js), substituting MongoDB and Express with DynamoDB and AWS Lambda, respectively. The best part of the MERN stack is that it's flexible, meaning we can still incorporate benefits of LAMP in our stack. Moreover, a major benefit of using the MERN stack is the uniformity in language: everything is in JavaScript/Java and passed on through JSON. LAMP requires the use of Linux or a Linux VM, and knowledge of three different languages.

Use of noSQL database and the AWS server allows us to let Amazon manage our server and ensure the quick scalability of our product. The best part is that it's still scalable with LAMP, meaning that if we do need a relational database, we can easily add it onto AWS ES2.

Student data is very valuable, and with our finished product we will be able to use this currently ignored information to its fullest extent. Not only will students benefit in knowing when areas are busy and where most of their peers are, administrators will have the benefit of added security at events through degradation-free authentication. For example, if someone's access card got stolen there is no way for the victim to recover such a key or get rid of it from the system themselves. He or she would have to notify the school which would take days to take effect. With QR codes and the web application we can simply delete the QR code from their account and instantly replace it, eliminating a prior vulnerability to the student body.

Android technology enables everyone with a smartphone to be able to have the added features that comes with the UniD account. With access cards, one would still need to use card readers to check ID's, but now any Android device can accomplish the same task instantly. This is especially convenient because Android smartphones are very common and affordable.

3. Goals and Requirements

3.1 Mobile Application (for administrators)

- Mobile (Android) application that administrators log into using Workday-authorized credentials
- An event or facility (known collectively as “authorizations”) can then be selected
- The following screen will scan QR codes representing students’ identifying codes
  - Upon successful scan, the activity will authenticate the student against Workday’s Student database
  - Upon successful authentication, the student’s and authorization’s information will be sent to our database for monitoring:
    - Who has already entered an event
    - Traffic at facilities
    - Number of credits left in the case of a student entering a dining commons
- Stretch goal: mobile app has option to create community service authorization entries
  - Allows event supervisor to quickly validate community hours for large numbers of volunteers

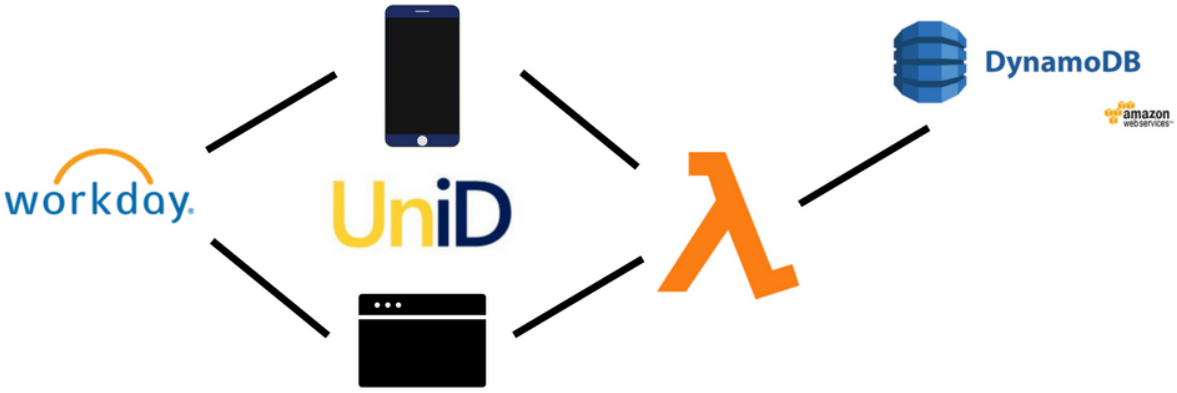
3.2 Web Application

- Student and administrator login/creation
- Students can view with mobile-friendly display:
  - Their unique QR code
    - With Apple Wallet integration
  - Dining commons credit balance
  - Traffic and trends at facilities
  - Upcoming events and popularity of current events
- Administrators can create authorizations depending on clearance level
- Administrators must valid preselected email (high-level admins can add these emails)

3.3 Backend Custom Database

- Custom database with information regarding authorizations related student activity

4. Product Structure



Our product will be broken up into three main parts: a mobile front end, a web app front end, and a backend database storing information about students and their check-in data.

The mobile front end will be targeted towards administrators checking in students to events or dining commons, and will be built with the Android SDK. The Android mobile application will also be using the Varvet QR Scanner library in order to handle check-ins. This app is never intended to be used on the student side, and will thus require a login from an administrator.

The web app front end, which is primarily targeted to be a mobile web application, will be built using React and hosted through Heroku. This web application will be a platform for students to generate their QR codes and view their check-in data at different events. This web application will also be a platform for administrators to view the events they have hosted. This app will also require a signin, as the web application contains private data for both students and administrators.

We will be using AWS Lambda to connect the mobile and web front ends together with our backend database. Both our mobile and web applications will hit custom API endpoints we design on Lambda to fetch information from our database, which will be utilizing DynamoDB. We chose to use Lambda and DynamoDB, as they are both hosted through Amazon Web Services. Our backend, specifically, will store any data relating to event data and student check-ins, but not authentication.

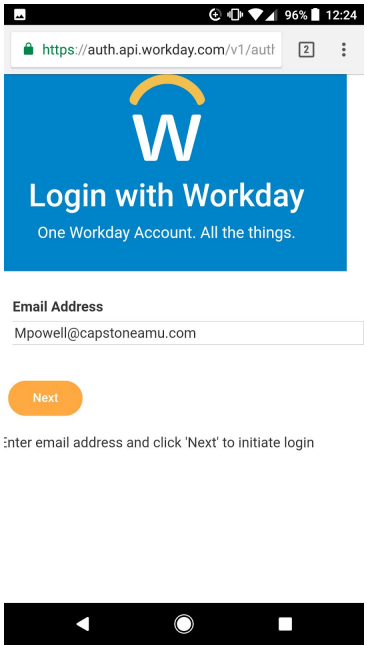
We will be using the Workday Student API to handle authentication of the students, using Workday's OAuth 2 technology. Workday's OAuth 2 API allows us to quickly check whether students are who they say they are when they attempt to check in. We will also be using Workday Student API to generate student objects. This is important, as we will be testing our entire model through a Workday database of students.

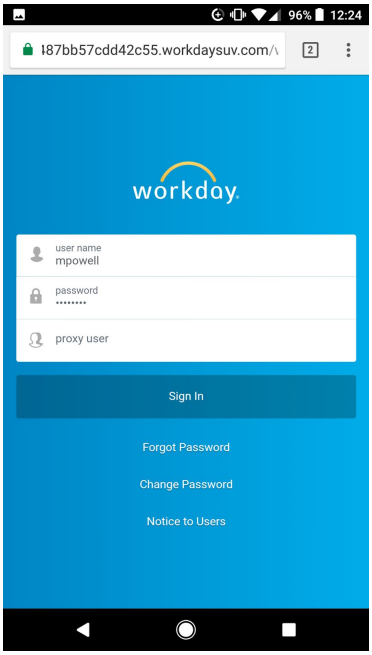
This structure is visualized at the beginning of this section.

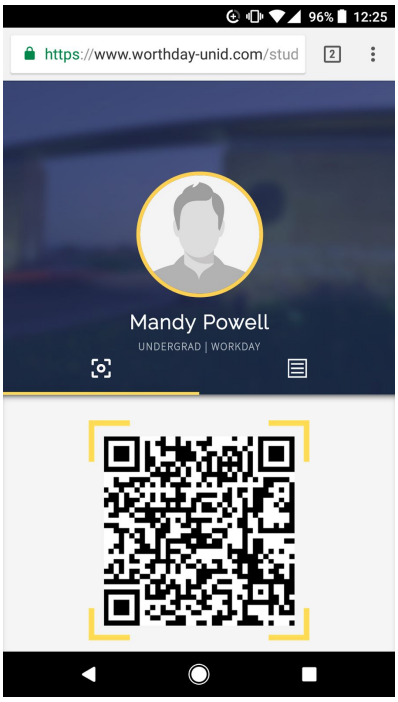
5. Assumptions

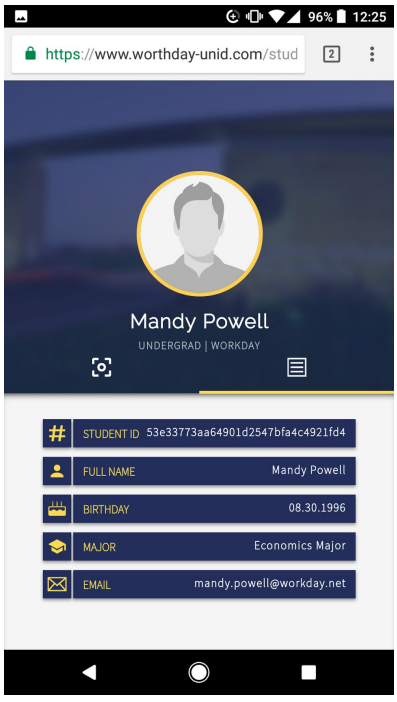
- Every student has access to the internet and can retrieve their QR code off the cloud database
- Administrators have access to the internet and therefore can access the cloud database so they can add, remove, or change student's profiles.
- Administrators have a mobile phone or QR code scanner to verify student QR codes

6. User Interaction and Design









7. User Stories and Use Cases

7.1 For Administrators

- As an administrator, I can view data of student activity so that I can have a better understanding of how the student body functions.
  - Acceptance: Workday admin can login using superuser credentials and view more information
- As a high level admin, I can create facility entries so that check-in staff can use them to quickly and securely validate people.
  - Acceptance: Aforementioned information includes option to create facility entries
- As a mid-level admin I can create event entries so that check-in staff can use them and ensure authenticity of students entering.
  - Acceptance: Aforementioned information includes option to create event entries
- As a low-level admin, I can check students into facilities and events so that UniD can accurately analyze data and the school can ensure campus wide security.
  - Acceptance: Upon scanning a student, their id is added to the student set for that event and the event id is added to the event set for that student
- As an admin, I can have the scanning app remember my login.
  - Acceptance: After logging in for the first time through Workday and obtaining a working token, the app will prompt the user to remember this token under a user name.
- As an admin, I can select my remembered login and the app will attempt to use the retained authentication token.
  - Acceptance: After the first login, the username will be displayed as a button which when pressed will try to use the remembered token and if it does not work will direct the user to login again.

- As the scanner, I can refresh the list of available authorizations in case more were added since my login session began.
  - Acceptance: Upon dragging from the top of the authorization selection screen or selecting “Refresh” from the drop-down menu, all available authorizations are displayed (currently this is a hard-coded list of three entries).
    Implemented in commits `5b64a35` and `6f96dfd`
- As the scanner, I can make the student’s information show longer to check their photo and information.
  - Acceptance: If the user is holding anywhere on the screen while a student’s information is showing, the information will not go away.
    Implemented in commit `429e548`
- As the scanner, I can see the information from the last student code I scanned.
  - Acceptance: When the review button is pressed, the student’s information will show again and a notification will be created (if not already) containing the information
    Implemented in commit `429e548`
- As the scanner, I want to be able to detect invalid QR codes
  - Acceptance: Upon scanning an invalid QR code, a Toast alert is shown
    Implemented in commit `d079c4b`
- As the scanner, I can logout at any time while scanning students
  - Acceptance: Upon selecting the logout option, the user is returned to the login screen
    Implemented in commit `bb3f963`
- As the scanner, I can view QR codes in the dark by way of the phone’s flash feature.
  - Acceptance: Upon clicking the flash icon, the flash is toggled (if available).
    Implemented in commit `e199464`
- As an admin, I can create community service entries in the scanning app.
  - Acceptance: There is an option in the drop down menu of the authorization-selection screen that creates a pop-up for creating a temporary community service entry.

- As an admin who created a temporary community service entry, I can authorize students for specific hours they contributed.
    - Acceptance: After creating a community service entry, the scanning activity then validates student's community service hours contribution by updating the student objects at the Workday Student database

7.2 For Students

- As a student, I can view how many swipes I have left at the dining commons.
  - Acceptance: When a student views his or her information, the amount of swipes the student has left is displayed.
- As a student, I can scan into the dining commons
  - Acceptance: When the student is scanned in for a meal, UniD deducts one swipe from the current user and notifies the scanner whether it was successful.
- As a student, I can view the menu at each of the dining commons.
  - Acceptance: When a student views the menu of the dining commons, UniD displays the current menu for the dining commons for that day.
- As a student, I can check into school events/facilities.
  - Acceptance: Student can use their QR code to check in and the scanner will show as valid and their data will be sent to the backend.
- As a student, I can view and receive notification of upcoming school events and activities.
  - Acceptance: Students can toggle notifications of upcoming events and the app will push notification of events to the students.
- As a student, I can view the amount of check-ins currently and hourly averages of check-ins
  - Acceptance: Students can see the current users that have checked in for that hour and see the average hourly check-in.
- As a student, I can receive updates and notification about my QR code usage
  - Acceptance: Students can report their stolen QR code and request a new one and see where their QR codes have been used.
- As a student, I can login to the application so that I can retrieve my QR code
  - Acceptance: Students are able to sign into Workday's OAuth2 with their account and retrieve their QR code.
    Implemented in commit: 3f7462d
- As a student, I can store my QR code into my passbook to be able to use offline
  - Acceptance: When viewing the QR code, an "add to Passbook" button will be available.

8. Code Samples

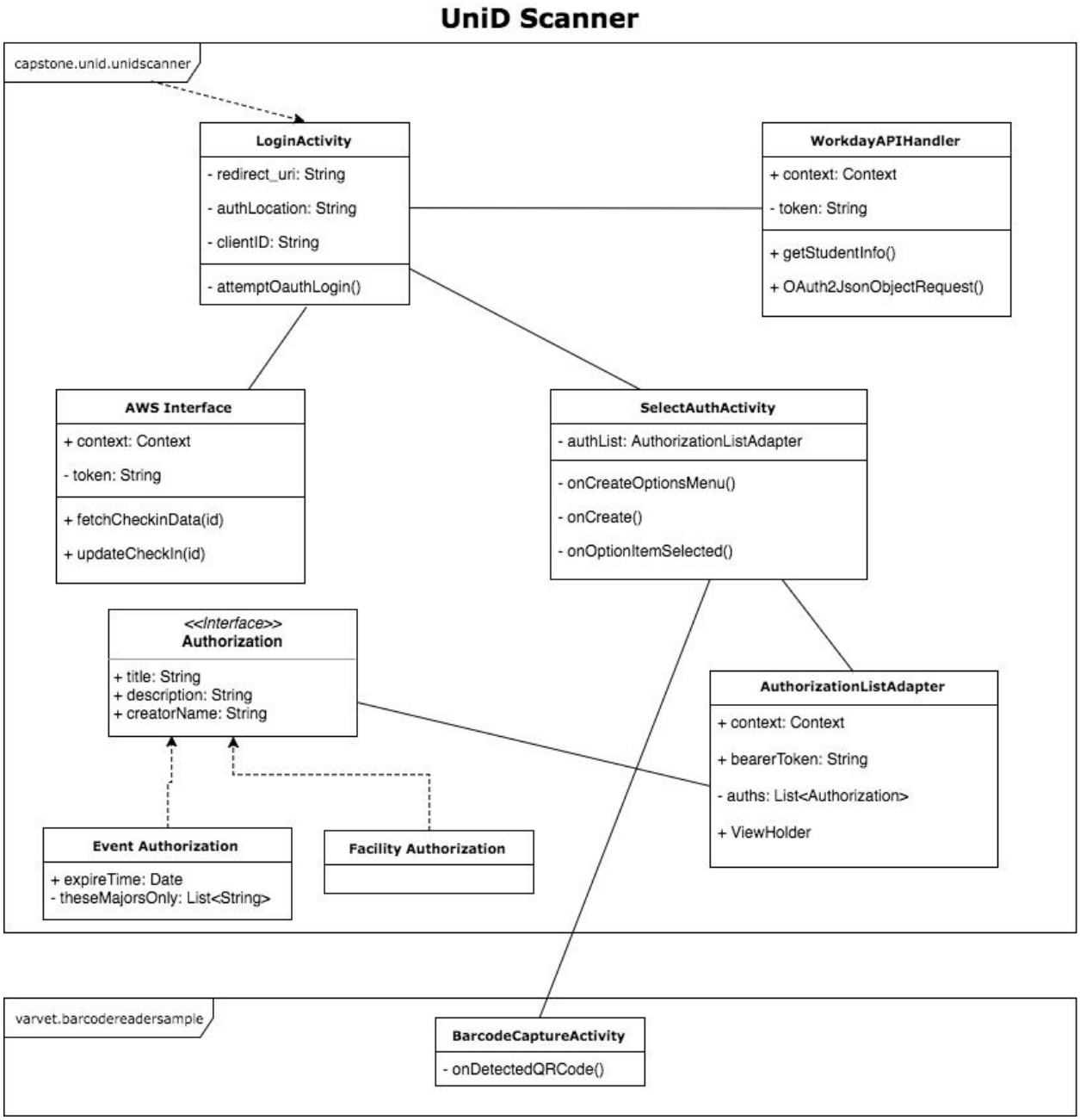
8.1 Android Scanner Application

```
// Handles QR code detection
override fun onDetectedQrCode(barcode: Barcode?) {
    if (barcode != null
            && !barcode.displayValue.all { it.isDigit() }
            && !displayingStudentInfo) {
        if (bearerTokenLost) return
        displayingStudentInfo = true
        if (Build.VERSION.SDK_INT >= 26)
            (getSystemService(VIBRATOR_SERVICE) as Vibrator)
                    .vibrate(VibrationEffect.createOneShot(150,10))
        else
            (getSystemService(VIBRATOR_SERVICE) as Vibrator).vibrate(150)

        handler.getStudentInfo(barcode.displayValue,
                object: VolleyCallback<Pair<String, String>>() {
            override fun onSuccess(response: Pair<String, String>) {
                lastStudentInfo = response
                showLastStudentIdFromUiThread()}

            override fun onFailure() {
                showToast("Invalid ID")}
        })
        displayingStudentInfo = false
    }
}

// Starts scanning activity for selected event
holder.view.setOnClickListener {
    if (bearerToken == null) {
        context.showToast(context.getString(R.string.bearer_token_lost))
        return@setOnClickListener
    }
    val intent = Intent(context, BarcodeCaptureActivity::class.java)
    intent.putExtra(context.getString(R.string.admin_bearer_token), bearerToken)
    intent.putExtra(context.getString(R.string.EVENT_INDEX), position)
    (context as Activity).startActivityForResult(intent, SCANNING_ACTIVITY)
 }
```

8.2 Web Application Login

```
function authenticate() {
	var url="http://www.worthday-unid.com/login";

	username_input = document.getElementById("username_input");
	password_input = document.getElementById("password_input");
	error_msg = document.getElementById("errorMessage");

	var params={
		"username":username_input.value.toLowerCase(),
		"password":password_input.value
	};

	$.post(url, params, function(resp)
		{
			var r = JSON.parse(resp);

			if(r["status"] == 200)
			{
				document.location.href =
"http://www.worthday-unid.com/student";
				errorMessage.style.display = "none";
			}
			else
			{
				password_input.value = "";
				errorMessage.innerHTML = "Incorrect username or password";
				errorMessage.style.display = "block";
				/*alert("Incorrect username or password");*/
			}
		});
}
```

9. System Model



10. Appendix

10.1 Technologies

- Android Studio and SDK
  - Mostly Kotlin with some borrowed Java code
  - Google Vision API
- DynamoDB for custom database
  - AWS Lambda for communication with database
- Workday Student API for student and administrator information
  - OAuth 2 for authentication
- React for web application framework
- AWS Beanstalk/heroku for web application host
- Github for version control