AUT Shuttle Ticketing System (Project Tessera)

Background:

The Auckland University of Technology stretches across three geographically distinct campuses in the Auckland region with a campus in Manukau (South Campus), Auckland CBD (City Campus) and Akoranga (North Campus). To link these campuses, AUT provides shuttle bus services in partnership with Pacific Tourways to allow staff and students to easily move between these campuses. Currently, tickets are business card sized pieces of paper with no means of verifying their authenticity. Drivers punch a hole or otherwise mark these tickets to show that they have been used and record passenger numbers when a passenger boards. These statistics are collated and presented to the transport team weekly. These stats are used to plan services and calculate total revenue.

Rationale:

The current ticketing system, has three main disadvantages. Tickets have no means of verifying their authenticity and can be easily forged. Tickets are fragile, easy to lose, and create waste. The manual reporting of passenger numbers provides unreliable business metrics to the transport team. By moving to a digital system, we can virtually eliminate ticket fraud, reduce paper waste, simplify drivers' jobs, and provide up-to-date, insightful metrics, and automated reporting features.

Goals:

With this project we aimed to:

- Create a passenger app available on iOS and Android platforms which could be used in lieu of a paper ticket
- Create an admin dashboard to provide ticketing statistics to the transport team
- Eliminate Ticket Fraud
- Speed up passenger boarding

Method:

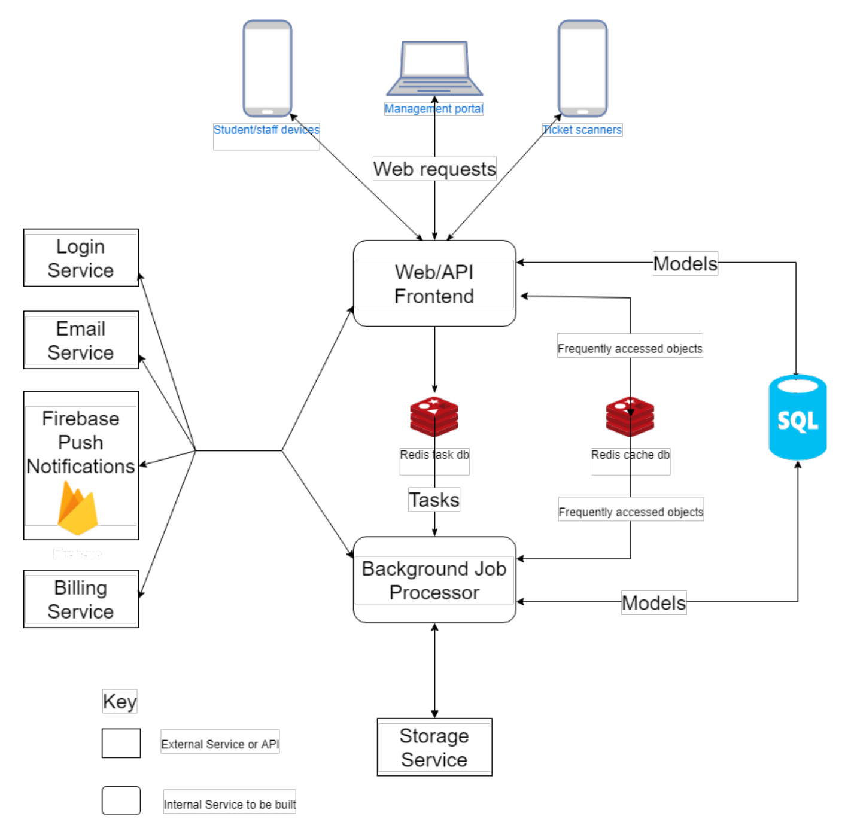
We chose to use a SCRUM based methodology. After some initial requirements gathering that involved meetings with the transport team, we started sprint zero. This included designing data models and API routes for our backend. Once completed, we moved on to iterative sprints. For each sprint, we set a goal, discussed what everyone's job was moving forward, and performed the actual development. Finally, we wrapped up each sprint with a sprint review, where we discussed what happened throughout that sprint. Throughout these sprints we tried to keep an accurate track of meeting minutes, as well as progress achieved. Despite this, we often found ourselves having gone a period of time without updating our minutes or related documentation. SCRUM/Agile, while marketed as a documentation light framework, still has a significant amount of documentation required and as such, on a small team, takes time away from development. After we started to fall behind with developing features, the importance of documentation reduced dramatically. Because of this, our project morphed and often did not follow SCRUM/Agile as closely as we could have. For example, we had the elements of a stand up meeting (what's blocking an individual, what went well and what they're currently working on) everytime we met. It simply wasn't recorded.

Quality Assurance:

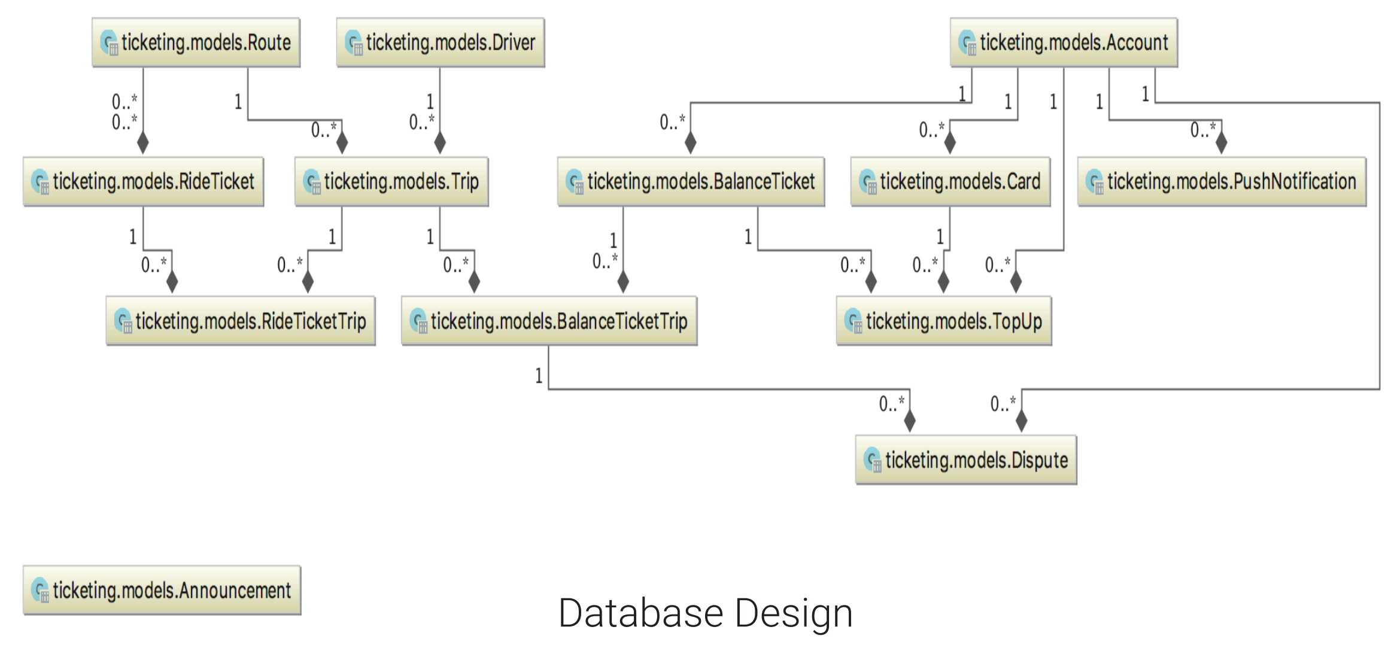
To ensure that all our code was up to standard all new features were peer reviewed, linted to ensure best practice and run through automated tests using Travis CI.

Results:

While we may not have produced all the features we set out to, what we did produce is a solid base for future work. We produced RSA signed QR codes that cannot be modified without notifying the system. This is vitally important to prevent users from cloning, copying or otherwise exploiting the ticketing system to allow them free rides. We modeled interactions and data in the system producing a set of thorough Django models. Both our iOS and Android apps are able to login, display user's QR code, as well as their current balance. The driver app can also log in and scan user tickets, causing their balance to be deducted by the cost of the ticket on the backend (yet another feature to prevent ticket abuse). Along with this, we also developed several key parts of the backend in Python: authentication, authorization, url routing, and business logic for creating and ending trips, charging different prices per route, and much more. As mentioned above, while we may not have completed all the features we set out to, this is a very solid base for a possible future team to work from.



Ticketing System Architecture Diagram


ticketing.models.Route
ticketing.models.Driver
ticketing.models.Account
ticketing.models.RideTicket
ticketing.models.Trip
ticketing.models.BalanceTicket
ticketing.models.Card
ticketing.models.PushNotification
ticketing.models.RideTicketTrip
ticketing.models.BalanceTicketTrip
ticketing.models.TopUp
ticketing.models.Dispute
ticketing.models.Announcement
1
0..*

Database Design

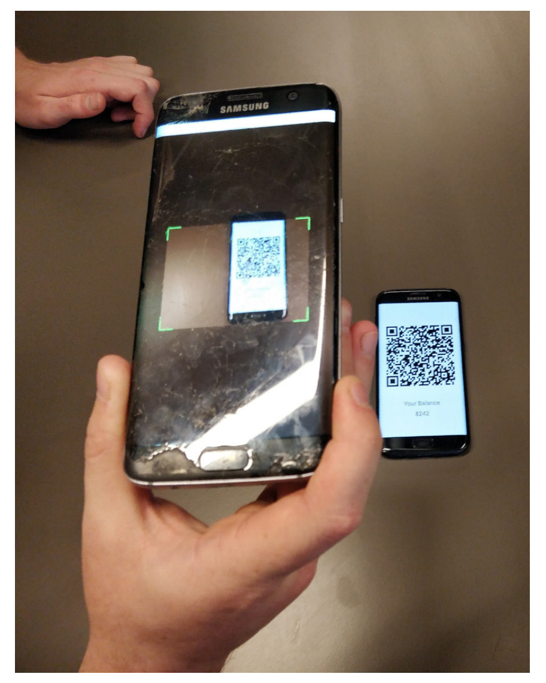
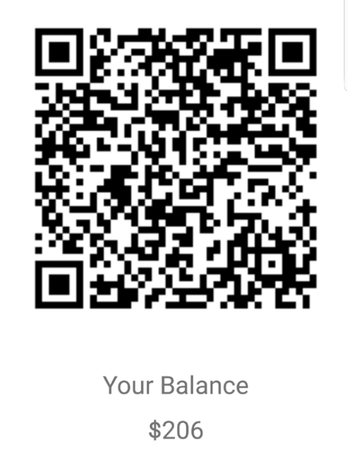


Photo shows driver scanning ticket to tag user onto service using the Zebra library (Zebra, 2018).



Tickets are QR Codes which contain:

- Passenger's unique ticket ID
- Whether the ticket is disposable or is virtual
- If the ticket is disposable, what its initial value is
- RSA signed hash of the contents

Technical Challenges:

Encryption and Security

- Tickets must be protected so that they cannot be forged
- Had to learn semantics of RSA public and private key signing in theory and in Python ("Public Key", n.d)
- Solved through reading RSA package documentation and trial and error

Rest Framework

- Having to learn a new novel framework
- Dealing with the frameworks idiosyncrasies
- Incomplete documentation in some cases
- Solved by reading documentation and reading and watching tutorials (Christie, n.d)

Android Studio/API

- Used to build Android apps
- Difficult Android API with a steep learning curve
- Solved this through previous experience and documentation

Unit Testing

- Unit tests breaking due to refactoring
- Solved by renaming test files to fit with Django's standards

Non-Technical Challenges:

Time Management
As with any group, project time management has been a problem, from meeting deadlines, to simply derailing and stopping work during a programming session. To try to solve this, we set ourselves and other goals for each of our meetings and programming sessions.

New Team Members
For this project, we had new team members joining half way into the project and we had to quickly integrate them into the team and also upskill them to keep the project progressing. We spent two weeks to upskill them and gave them documentation to read up on to understand what the team has been doing so far.

Distraction
Another issue we suffered from was team distraction. As a team, we got on very well with one another and had very similar interests. Often, one team member may start a conversation with another which would cause the rest of the team to stop work to engage in the conversation. This was a constant struggle, to find the right balance between breaks and work.

Future Work:

Continued development including integration with AUT's Single Sign on Service and billing system, implementing offline scanning of tickets in the driver app and an admin dashboard for realtime and historical metrics.

Acknowledgments:

We would like to thank our clients Sonia Simpson and Sally Vallely for their support of this project.

References:

- Christie, T. (n.d.). Django REST Framework. Retrieved from https://www.django-rest-framework.org/
- Zebra. (2018, October 04). Zxing/zxing. Retrieved from https://github.com/zxing/zxing
- Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1. (n.d.). Retrieved from https://tools.ietf.org/html/rfc3447

By Izaac Crooke, The Ton Le, Chris Lipscombe, Blake Mclean, Dominic Porter, Hayden Woodhead