



e6 - waitr

Sophie Sacks, Sam Lavelle, Dina Razek

# Use Case and Use-Case Requirements

## Scenario and Solution

---

- Hard to know how busy eateries are
- Combined hardware and software solution: two radio frequency identification (RFID) scanners + web application
- Scan in at beginning and end of line
- Web application for users to view with machine learning in backend

## Quantitative Requirements

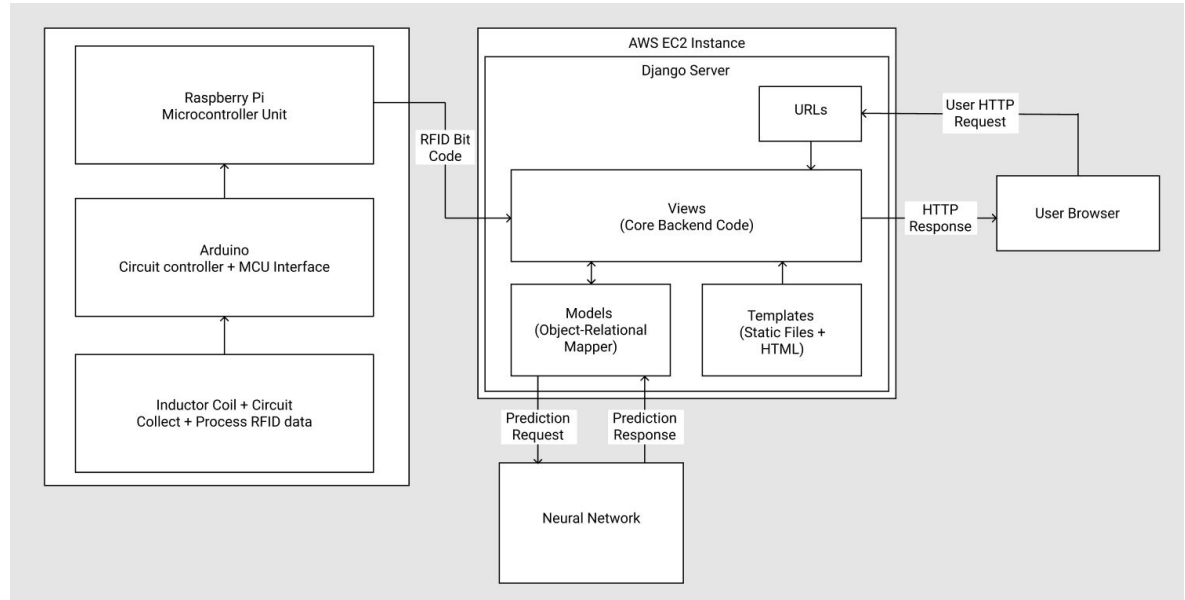
---

- AC-powered, OR battery powered scanner lasting **3 hours** or more
- Transmission from scanner to server within **2 seconds**
- Web application response time in under **2 seconds**
- Margin of error for wait time: **2 minutes**, or within **10%**
- Error validation on form and scan inputs
- Ability to keep track of up to **50 patrons**

# Solution Approach

## The Solution

- 2 RFID readers
- A board to send the data from the readers to the server
- A web application to publish the wait time
- Justified by accuracy, anonymity, and prior experience

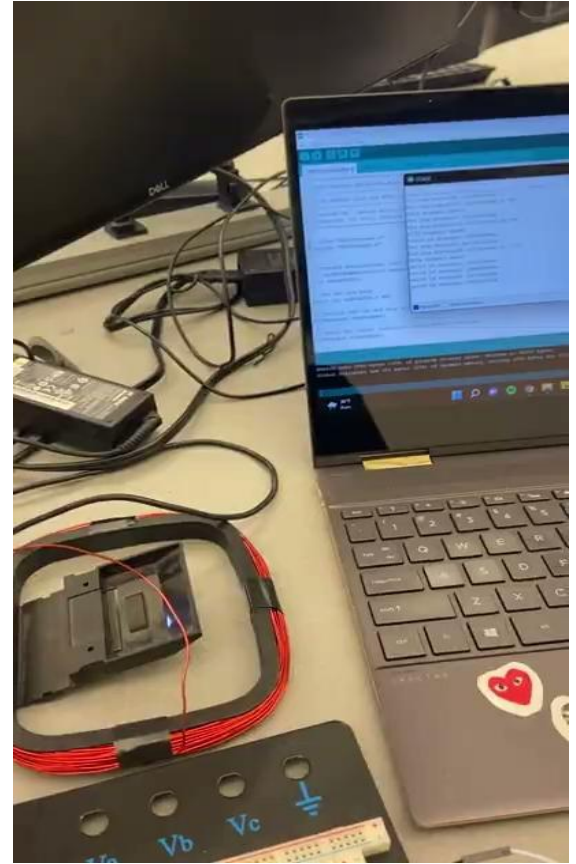


# Complete Solution

## Step 1: An ID card will be scanned on our entry RFID reader

---

- RFID reader detects presence of ID card and its bit code
- Arduino converts bit code signal to digital & sends to RPi
- RPi sends ID and entry time to web application through a post request
- Entry table in SQL database is updated with ID number and entry time










# Complete Solution

## Step 3: Preview the [web application](#)

- View the predicted wait times for today, whose values are populated from the ML model
- Submit a error wait time through the form to display the error catch
- Submit a successful wait time through the form to display the success status and updated live wait time



# Original Testing, Verification, and Validation

Test Inputs		Passing Test Outputs
ID card to be read on RFID scanner		RFID reader detects presence of ID card and its bit code
RFID scanner output (current date and time)		Received properly by the RPis within 2 sec
Date and time data from the RPis		Correct storage in the Django SQL database (entry vs. exit)
SQL database data		ML model predicting wait times
Web application frontend page reload		Accurate wait time prediction from the ML model (margin of error within 2 minutes or 10% of actual wait time)

# Final Testing, Verification, and Validation

<b>Use-Case Requirement</b>	<b>Design Requirement</b>	<b>Test</b>	<b>Passing Metric</b>	<b>Performance</b>
Battery lasting 3 hours or more	125kHz power source + power for op amp, Rpi, and Arduino	Test time from 100% to 0% during moderate use	Time to 0% > 3 hours	N/A - currently using AC sources, as req'd by ADALM
ID can be tracked	Circuit can accurately extract RFID bit code	Print from circuit to serial monitor; compare to commercial reader	Bit code is consistent across our circuit and commercial reader	The bit code is consistent
Web app receives live data within 2 seconds	Transmission from scanner to server within 2 seconds	Print time to make request to console	RFID scanner sends request in under 2 seconds	Average request time is <i>0.96 seconds</i>



# Final Testing, Verification, and Validation

<b>Use-Case Requirement</b>	<b>Design Requirement</b>	<b>Test</b>	<b>Passing Metric</b>	<b>Performance</b>
Web application is not noticeably slow for users	Web application response time in under 2 seconds	Print time from scan to page reload in console	From scan to page reload in under 2 seconds	Average request time is <i>1.84 seconds</i>
Input data within business hours	Error validation on form and scan inputs	Try adding wait time with exit time before entry time	Successfully show error to the user	Catches <i>all</i> logical/business errors
Ability to keep track of up to 50 patrons	Utilize a large enough database	Populate database with 50 separate points	No errors with handling 50 data points	Able to hold <i>all</i> necessary data at once

# Design Trade-Offs

---

- **Hardware: Power Source**
  - AC source (laptop) to allow for precise signal control
  - Cannot have independent system until independent +/-5V & 125kHz sources
- **Software: ML Learning Rate**
  - Upped learning rate to .01
  - Dropped loss function down by over 150 on average
- **Software: ML Mock Data**
  - Updating training data to be only on 15 minute intervals
- **Web App: Usability**
  - Amount of data given in graph is 15 min intervals (user testing)
- **Web App: Frontend and Backend Frameworks**
  - Frontend: Bootstrap
  - Backend: Django

