# Unit Testing

Tests on individual parts will be described here. This includes the Watech mobile app testing and Raspberry Pi parking lot prototype testing.

## Watech Parking Android App

### Test #1: Register

* Tested if data can be sent from register screen to Firebase
* Tested profile picture for registration

### Test #2: Forgot your password

* Tested email and phone verification through Firebase provided functions
* Tested if email gets sent for new password request
* Tested notification status once forgotten password process is submitted

### Test #3: Verify your passsword

* Tested verification code process, if received notification with code about forgotten password then you can get new password
* User can then enter new password and confirm password and submit for the new password if new and confirmed password match

### Test #4: Login

* Test if email exists in Firebase data structure ‘TestUsers’
* Test if login credentials are valid
* If user exists go to Home screen
* If user does not exist error pops up on screen

### Test #5: Home Screen

* Tested if parking lots were available in CardView structure. Each parking lot has a ‘View Details’ and ‘Reserve’ button
* Test when View Details button is pressed if it will display information about the parking lot
* Test when Reserve is pressed then it will allow you to get a parking pass
* Check Firebase if values have been added for the ParkingLocations for user id
* Check if user received notification on phone when Parking Pass is available.

### Test #6: Parking Passes

* Check if data is displayed and can access the reserved lot
* Select button will allow for payment

### Test #7: Order History

* After parking pass is bought check if it appears on the Order History Screen

### Test #8: Payment

* Check Orders table on Firebase to make sure order went through once payment is processed from Parking Passes screen

### Test #9: Add a Car

* This screen allows users to enter add a car by entering make, model, color, and license plate number.
* Check if that data is sent to the Firebase data structure “Cars”

### Test #10: Manage Cars

* Check if cars added in the “Add a Car” screen appear hear.
* This screen will show all of the cars associated with the user account
* Test if you can edit or delete cars by pressing the coordinating buttons

### Test #11: View Details Button (Parking Lot Data Screen)

* Test if GateStatus value on Firebase is changed for entry and exit and colors change on app accordingly
* Test proximity value reading from Firebase
* Test admin control sending values for gate control
* Test timestamp is read from GateStatus structure

## SMART Parking Lot Prototype

### Test #1: IR Break Beam

* Test if the sensor can detect if an object is blocking IR beam
* Test if receiver can detect the beam through the Python code
* Test if a HIGH value is received when the beam is detected and LOW value when beam is no detected.
* Check when HIGH value, noEntry is set to close gate
* Check when LOW value, entry is set to open gate

### Test #2: VCNL Proximity

* Test if send values based on if an object approaches the sensor. It will report the distance to Firebase under ProximityStatus
* Check if parking spot 1A is available if the sensor reads a value less than or equal to 2500. If the value is greater than 5000 the space is taken. If value is between 3000 and 4500 car is approaching the space.
* Database should be updated according to sensor value

### Test #3: Servo-Motor

* Once the IR beam receiver detects interference with IR beam it will raise the servo motor arm and allow entry/exit
* Check if GateStatus on Firebase has changed value for when gate is allowing “entry” or “noEntry”. Gate will lower arm after 2 seconds and status will change to noEntry.
* Entry status on Firebase show have a true value for entry and a timestamp.
* noEntry status will have a timestamp as well but will have a status of false.