***TEST CASES:***

*#DATABASE ACCESS AND CONTROL  
  
##Setup Functions* **def** testDatabaseRetrieve(self): *#Tests database access retrieving current permit* self.assertTrue(retrieveField(2, **"Citation\_Number"**, 1))  
  
 **def** testParkingInsert(self): *#Tests database - parking table, for valid access and insertion* self.assertFalse(insertField(**"Parking"**))  
  
 **def** testOtherInsert(self): *#Tests database - other table, for valid access and insertion* self.assertFalse(insertField(**"Other"**))  
  
 **def** testHSInsert(self): *#Tests database - health and safety table, for valid access and insertion* self.assertFalse(insertField(**"HS"**))  
  
 **def** testVioInsert(self): *#Tests database - violations table, for valid access and insertion* self.assertFalse(insertField(**"Vio"**))  
  
*##Checking and Recieving* **def** testPasswordCheck(self): *#Tests database - password access for consistent passwords with email addresses* self.assertTrue(passwordField(**"Aaron.Luthor@atmiye.edu.au"**,**"Wally2"**))  
  
 **def** testInvalidPassword(self): *#Tests password input for validity against email in database* self.assertTrue(passwordField(**"Aaron.Luthor@atmiye.edu.au"**,**"Test"**))  
  
 **def** testUpdateViolation(self): *#Tests database - violations table for access and redability of updates* self.assertFalse(insertField(**"Vio"**))  
  
 **def** testPaidPayment(self): *#Tests database - payments table for accepted payments* self.assertFalse(paymentField(**"Paid"**))  
  
 **def** testPendingPayment(self): *#Tests database - payments table for pending payments* self.assertFalse(paymentField(**"Pending"**))  
  
 **def** testInvalidPaidPayment(self): *#Tests database - payments table for invalid payments* self.assertFalse(paymentField(**"Invalid"**))  
  
*##Teardown Functions* **def** testDatabaseDelete(self): *#Tests database teardown - tests for deletion of all previously inserted elements.* self.assertTrue(deleteField())  
  
*##Notifications* **def** testNotificationAddition(self): *#Tests notifications table for access and ability to add new notifications* self.assertTrue(noteAd(**"Test"**))  
  
 **def** testNotificationRemoval(self): *#Tests notifications table for valid removal. N.B. This makes the tests list empty.* self.assertTrue(noteRemove())  
  
 **def** testEmptyList(self): *#Tests notifications table for invalid removal if list is empty* self.assertTrue(noteRemove())  
  
*##Account Creation and Payment Inputs* **def** testUserInsert(self): *#Tests database - user table for valid insertion of test user* self.assertTrue(userInsert())  
  
 **def** testPaymentInsert(self): *#Tests database - payments table for payment insertion relating to particular permit numbers* self.assertTrue(paymentCheck(**"Insert"**, 255))  
  
 **def** testDetailsInsert(self): *#Tests database - payments table for details insertion relating to particular permit numbers* self.assertTrue(paymentCheck(**"Details"**, 255))  
  
 **def** testUpdatePayment(self): *#Tests database - payments table for ability to update payment details relating to particular permit numbers* self.assertFalse(paymentCheck(**"Update"**, 255))  
  
 **def** testCheckPaymentExist(self): *#Tests database - payments table for current outstanding payments relating to particular permit numbers* self.assertFalse(paymentCheck(**"Check"**, 255))  
  
 **def** testCheckPaymentAmount(self): *#Tests database - payments table for assigned numerical value of outstanding payments relating to particular permit numbers* self.assertFalse(paymentCheck(**"Amount"**, 255))  
  
 **def** testPaymentInvalidClaim(self): *#Tests database - payments table for invalid access claims* self.assertFalse(paymentCheck(**"InvalidFunction"**, **"Test"**))  
  
 **def** testRemovePayment(self): *#Tests database - payments table for valid removal of payments relating to particular permit numbers* self.assertTrue(paymentCheck(**"Remove"**, 255))  
  
 **def** testNonExistRemovePayment(self): *#Tests database - payments table for invalid removal of non-requried payments relating to particular permit numbers* self.assertFalse(paymentCheck(**"Remove"**, 255))  
  
 **def** testRemoveDetails(self): *#Tests database - payments table for valid removal of payments details relating to particular permit numbers* self.assertFalse(paymentCheck(**"RemoveD"**, 255))  
  
 **def** testNonExistRemoveDetails(self): *#Tests database - payments table for invalid removal of non-existent details relating to particular permit numbers* self.assertTrue(paymentCheck(**"RemoveD"**, 255))  
  
 **def** testCardProtection(self): *#Tests card protection function for valid card masking* self.assertTrue(cardNumCheck(1111111111111111))  
  
 **def** testFalseCardProtection(self): *#Tests card protection function for invalid cards* self.assertTrue(cardNumCheck(11111111111))  
  
 **def** testLoggedIn(self): *#Tests login function for websites* self.assertFalse(checkLogin(1))  
  
 **def** testLoggedOut(self): *#Tests logout function for websites* self.assertFalse(checkLogin(2))  
  
 **def** testUserRemove(self): *#Tests database - user table for valid removal of account details. N.B This removes thest user from the database.* self.assertTrue(userDelete(**"Test"**))  
  
 **def** testNonExistUserRemove(self):*#Tests database - user table for invalid removal of non existent account details* self.assertFalse(userDelete(**"NonExist"**))