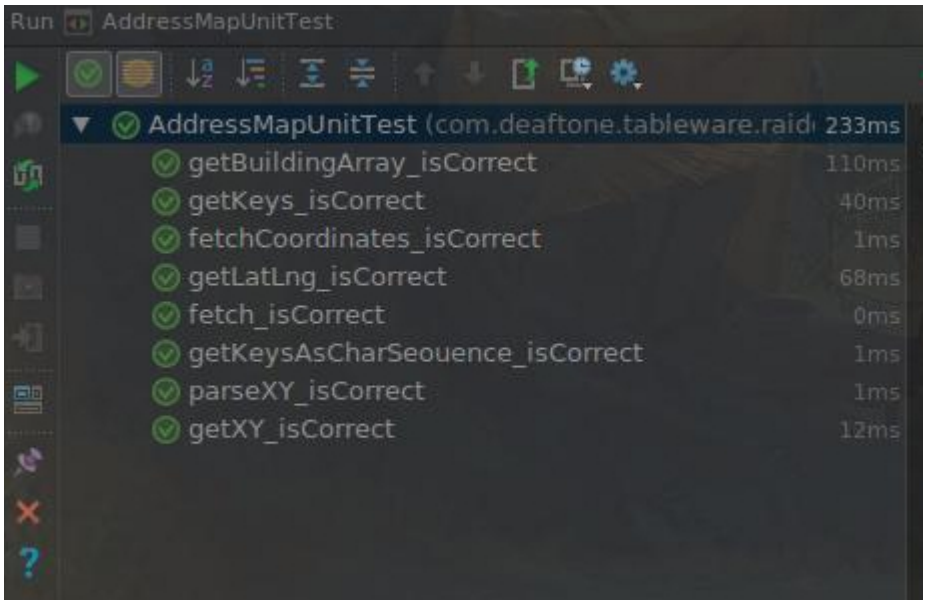
Test Plan
for RaiderNAV

Table of Contents

1.1 AddressMap Unit Tests

```
package com.deaftone.tableware.raidernav;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;
import static org.junit.Assert.*;
public class AddressMapUnitTest {
  @Before
  public void initialize() {
     AddressMap.initialize();
  }
  @Test
  public void getKeys_isCorrect() {
     assertEquals("ADMIN", AddressMap.getKeys().iterator().next());
  }
  @Test
  public void getKeysAsCharSeouence_isCorrect() {
     assertEquals("AGRI",AddressMap.getKeysAsCharSequence()[4]);
  }
  @Test
  public void getXY_isCorrect() {
     assertEquals(-101.874702,AddressMap.getXY("ADMIN")[1],.1);
  }
  @Test
  public void parseXY_isCorrect() {
     assertEquals(33.583427,AddressMap.parseXY("33.583427, -101.874702")[0],.1);
  }
  @Test
  public void getLatLng_isCorrect() {
     assertEquals(33.587275,AddressMap.getLatLng("ENGCTR").latitude,.1);
  }
  @Test
```

```
    public void getBuildingArray_isCorrect() {
        assertEquals("ADMIN",AddressMap.getBuildingArray().iterator().next());
    }
    @Test
    public void fetch_isCorrect() {
        assertEquals("33.587275, -101.875771", AddressMap.fetch("ENGCTR"));
    }
    @Test
    public void fetchCoordinates_isCorrect() {
        assertEquals("33.583427, -101.874702",AddressMap.fetchCoordinates("ADMIN"));
        assertEquals(null,AddressMap.fetchCoordinates("GARBAGE HALL"));
    }
}
```



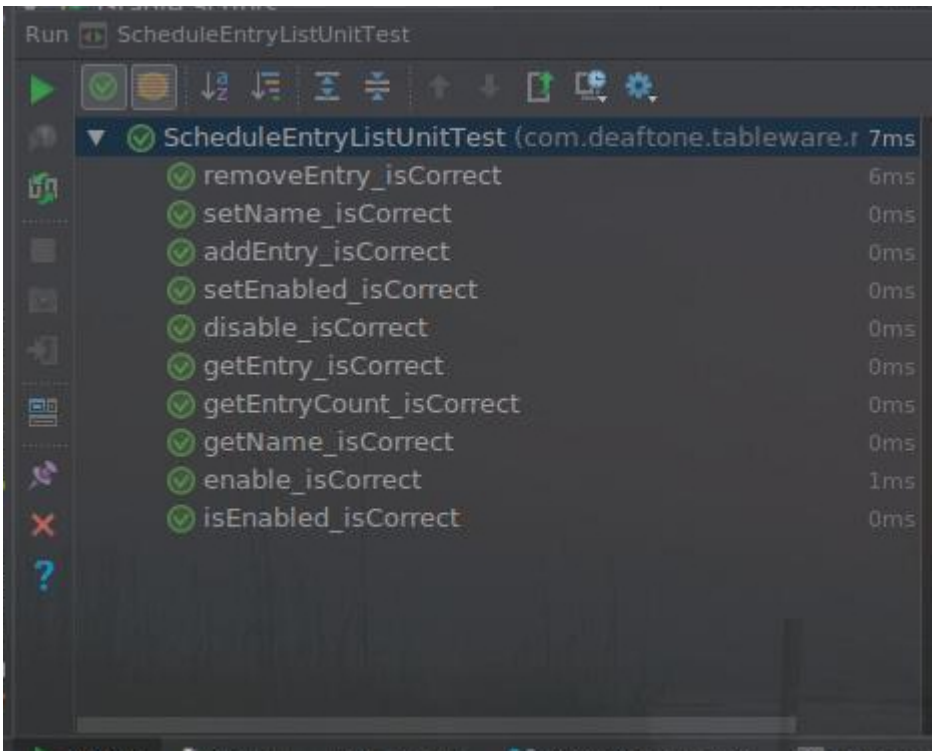
1.2 ScheduleEntryList Unit Tests

```
package com.deaftone.tableware.raidernav;
import org.junit.Before;
import org.junit.Test;
import static org.junit.Assert.assertEquals;
public class ScheduleEntryListUnitTest {
  ScheduleEntryList sel;
  private ScheduleSingleEntry sse;
  private final String coursenumber = "RUSN 1502";
  private final String building = "Foreign Language";
  private final String starttime = "0900";
  private final String endtime = "0950";
  private boolean[] days = {false, true, true, true, true, true, false};
  @Before
  public void initialize() {
     sel = new ScheduleEntryList();
     sse= new ScheduleSingleEntry(coursenumber, building, starttime, endtime, days);
  }
  @Test
  public void addEntry_isCorrect() {
     sel.addEntry(sse);
     assertEquals(building,sel.getEntry(0).getBuilding());
  }
  @Test
  public void removeEntry_isCorrect() {
     sel.removeEntry(sse);
     assertEquals(0, sel.getEntryCount());
  }

  @Test
  public void getEntry_isCorrect() {
     sel.addEntry(sse);
```

```
    assertEquals(building, sel.getEntry(0).getBuilding());
}
@Test
public void getEntryCount_isCorrect() {
    assertEquals(0,sel.getEntryCount());
}
@Test
public void getName_isCorrect() {
    assertEquals(null,sel.getName());
    sel.setName("nomad");
    assertEquals("nomad",sel.getName());
}
@Test
public void setName_isCorrect() {
    sel.setName("House");
    assertEquals("House",sel.getName());
}
@Test
public void enable_isCorrect() {
    sel.enable();
    assertEquals(true,sel.isEnabled());
}
@Test
public void disable_isCorrect() {
    sel.disable();
    assertEquals(false,sel.isEnabled());
}
@Test
public void setEnabled_isCorrect() {
    sel.setEnabled(false);
    assertEquals(false,sel.isEnabled());
}
```

```
    @Test
    public void isEnabled_isCorrect() {
        assertEquals(false,sel.isEnabled());
        sel.setEnabled(true);
        assertEquals(true,sel.isEnabled());
    }
}
```



1.3 ScheduleSingleEntry Unit Tests

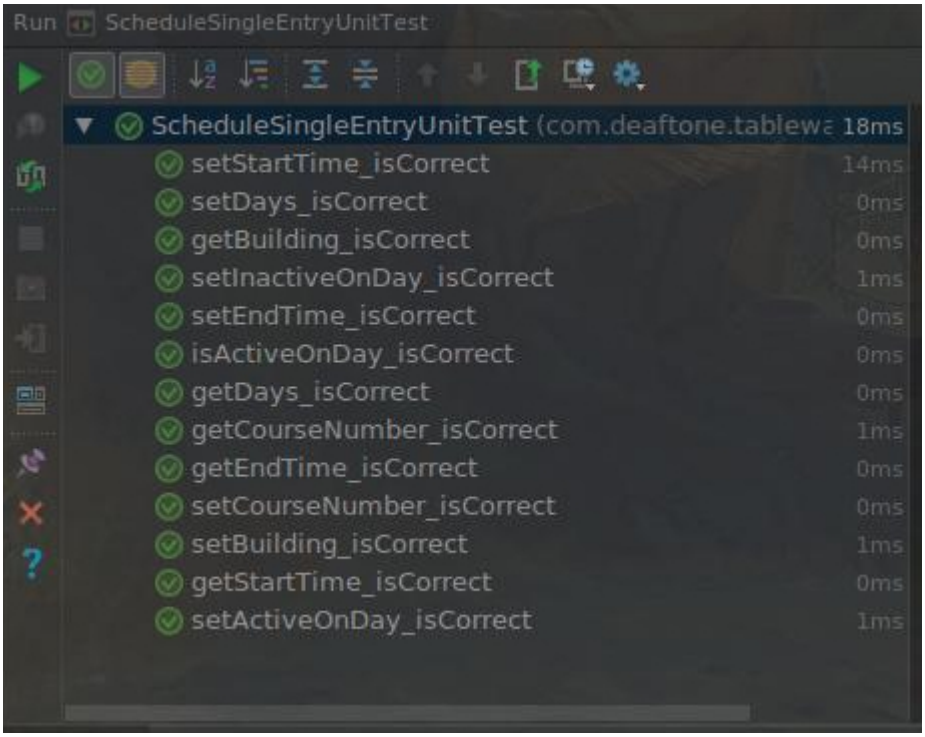
```
package com.deaftone.tableware.raidernav;
import org.junit.Before;
import org.junit.Test;
import static org.junit.Assert.assertEquals;
public class ScheduleSingleEntryUnitTest {
  private ScheduleSingleEntry sse;
  private final String coursenumber = "RUSN 1502";
  private final String building = "Foreign Language";
  private final String starttime = "0900";
  private final String endtime = "0950";
  private boolean[] days = {false, true, true, true, true, true, false};
  @Before
  public void initialize() {
    sse = new ScheduleSingleEntry(
        coursenumber, building, starttime, endtime, days);
  }
  @Test
  public void getCourseNumber_isCorrect() {
    assertEquals(coursenumber, sse.getCourseNumber());
  }
  @Test
  public void setCourseNumber_isCorrect() {
    sse.setCourseNumber("CS 4352");
    assertEquals("CS 4352", sse.getCourseNumber());
  }
  @Test
  public void getBuilding_isCorrect() {
    assertEquals(building, sse.getBuilding());
  }
  @Test
  public void setBuilding_isCorrect() {
```

```
    sse.setBuilding("Holden Hall");
    assertEquals("Holden Hall", sse.getBuilding());
}
@Test
public void getStartTime_isCorrect() {
    assertEquals(starttime, sse.getStartTime());
}
@Test
public void setStartTime_isCorrect() {
    sse.setStartTime("0300");
    assertEquals("0300", sse.getStartTime());
}
@Test
public void getEndTime_isCorrect() {
    assertEquals(endtime, sse.getEndTime());
}
@Test
public void setEndTime_isCorrect() {
    sse.setEndTime("0400");
    assertEquals("0400", sse.getEndTime());
}
@Test
public void getDays_isCorrect() {
    assertEquals(false, sse.getDays()[0]);
    assertEquals(true, sse.getDays()[1]);
}

@Test
public void setDays_isCorrect() {
    boolean[] newdays = {true, false, true, true, true, true, false};
    sse.setDays(newdays);
    assertEquals(true, sse.getDays()[0]);
```

```
    assertEquals(false, sse.getDays()[1]);
}
@Test
public void setActiveOnDay_isCorrect() {
    sse.setActiveOnDay(0);
    assertEquals(true, sse.getDays()[0]);
    sse.setActiveOnDay(1);
    assertEquals(true, sse.getDays()[1]);
}
@Test
public void setInactiveOnDay_isCorrect() {
    sse.setInactiveOnDay(0);
    assertEquals(false, sse.getDays()[0]);
    sse.setInactiveOnDay(1);
    assertEquals(false, sse.getDays()[1]);
}
@Test
public void isActiveOnDay_isCorrect() {
    sse.setActiveOnDay(0);
    sse.setInactiveOnDay(1);
    assertEquals(true, sse.isActiveOnDay(0));
    assertEquals(false, sse.isActiveOnDay(1));
}
```

}


Run ScheduleSingleEntryUnitTest
ScheduleSingleEntryUnitTest (com.deaftone.tablew 18ms
setStartTime_isCorrect 14ms
setDays_isCorrect 0ms
getBuilding_isCorrect 0ms
setInactiveOnDay_isCorrect 1ms
setEndTime_isCorrect 0ms
isActiveOnDay_isCorrect 0ms
getDays_isCorrect 0ms
getCourseNumber_isCorrect 1ms
getEndTime_isCorrect 0ms
setCourseNumber_isCorrect 0ms
setBuilding_isCorrect 1ms
getStartTime_isCorrect 0ms
setActiveOnDay_isCorrect 1ms

2. Use Case Testing

Formatting

Title:
Actors:
Requirement:
Main Scenario:
Alternatives:

Test Situations:
Test Coverage:
Base: number of main and alternative scenarios =
Test situations cover all cases
100% coverage of use case

Test Record:
Test Case:

2.1 Story 1

Use Case: Add Schedule
Summary: App User creates a new class schedule
Actors: User
Dependency:
Precondition: The application should be displaying the My Schedules screen
Description:
1. User presses Create New Schedule button on the My Schedules screen.
2. The application prompts the user to enter the schedule name.
3. User inputs schedule name and presses OK.
4. The system adds the schedule to a visible list on the My Schedules screen.

Alternatives:
3a. The user presses Cancel
3a.1 The system displays the My Schedules screen.
Postcondition: The user has created a schedule.

Title: Add Schedule
Actors: User
Requirement: R1
Main Scenario:
1. User presses Create New Schedule button on the My Schedules screen.
2. The application prompts the user to enter the schedule name.
3. User inputs schedule name and presses OK.
4. The system adds the schedule to a visible list on the My Schedules screen.

Alternatives:
3a. The user presses Cancel
3a.1 The system displays the My Schedules screen.
3b. The user inputs nothing and presses OK

3b.1 The system displays the My Schedules screen.

Test Situations:

1. New schedule is added to a visible list on the My Schedules screen.
2. User presses Cancel
3. The user inputs nothing and presses OK

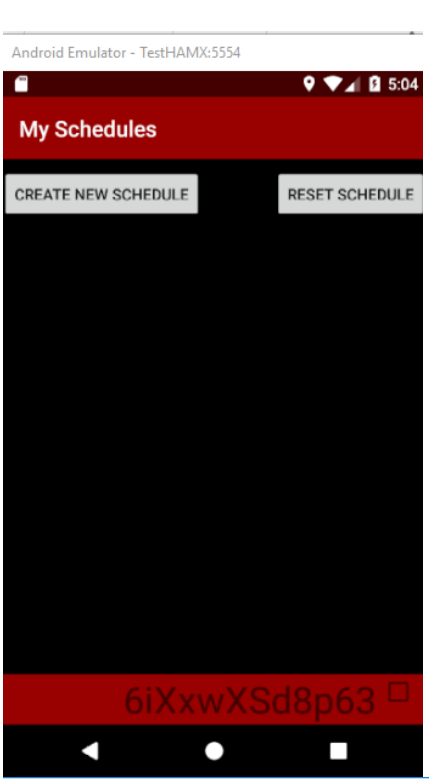
Test Coverage:

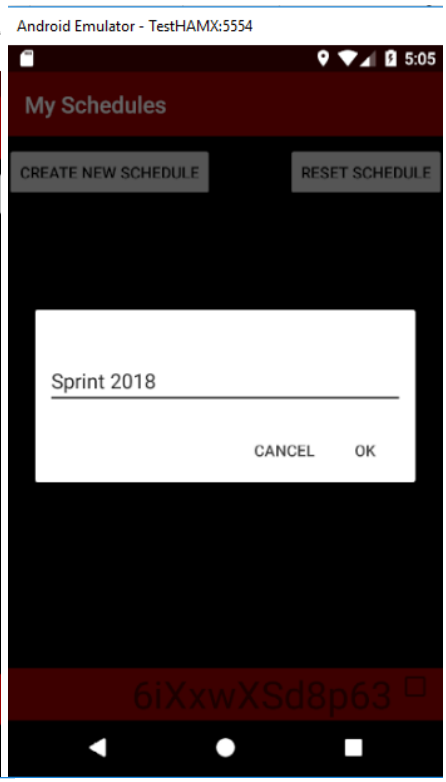
Base: number of main and alternative scenarios = 3
Test situations cover all 3 cases
100% coverage of use case

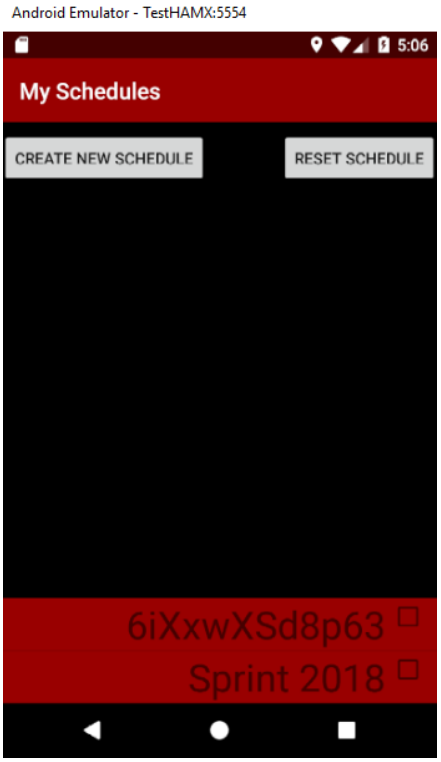
Test Record:

Test Case 1:

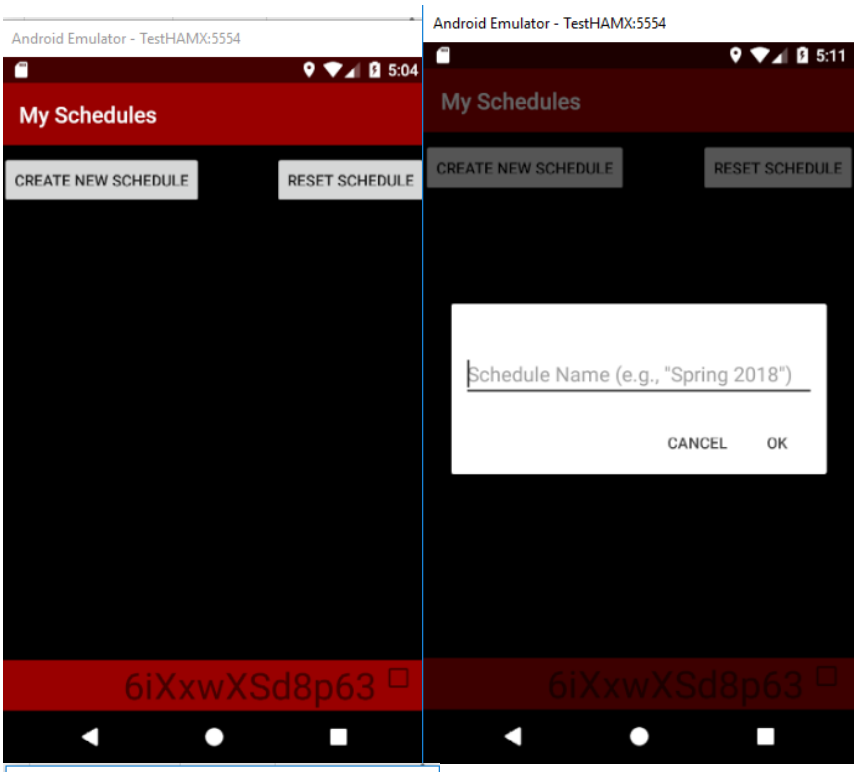
User presses Create New Schedule button on the My Schedules screen.
The application prompts the user to enter the schedule name.
User inputs schedule name and presses OK
The system adds the schedule to a visible list on the My Schedules screen.

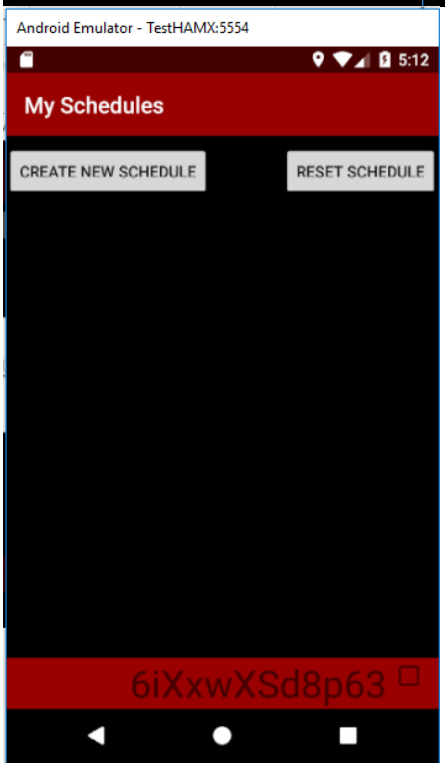




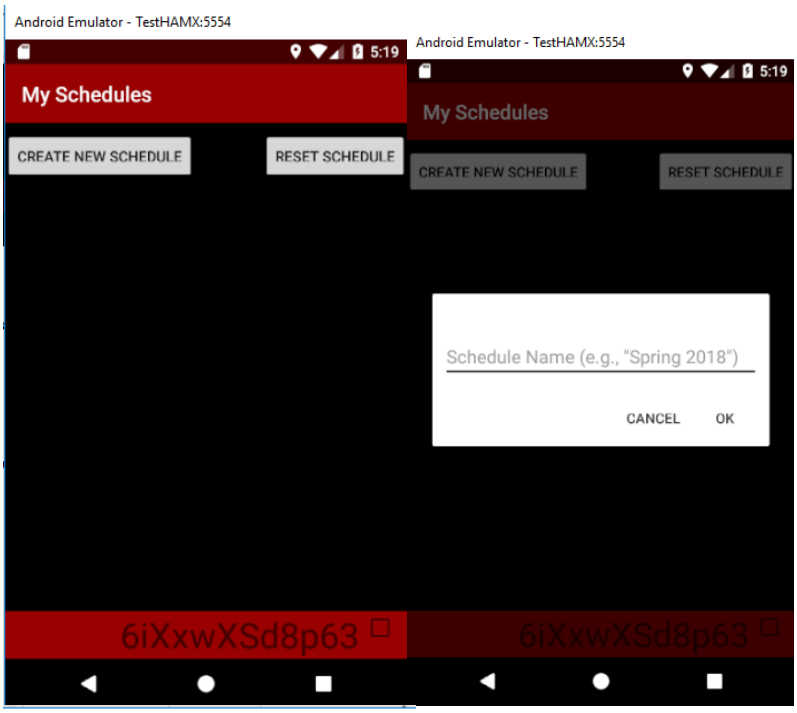


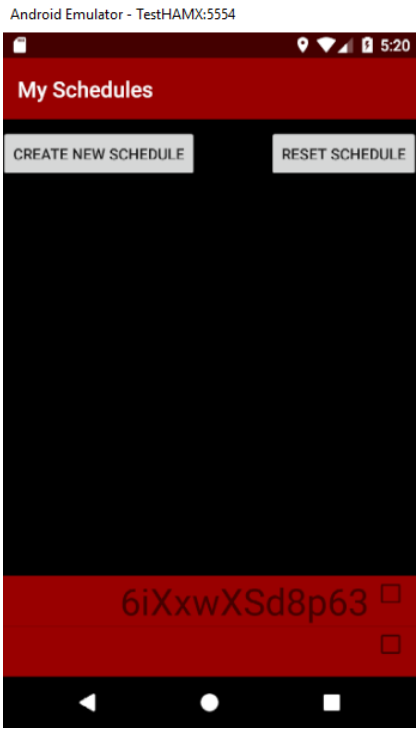
Test Case 2:
User presses Create New Schedule button on the My Schedules screen.
The application prompts the user to enter the schedule name.
The user presses Cancel
The system displays the My Schedules screen





Test Case 3:
User presses Create New Schedule button on the My Schedules screen.
The application prompts the user to enter the schedule name.
The user inputs nothing and presses OK
An empty schedule is added





Note on R1 Test Result:
The user inputs nothing and presses OK, the empty schedule should not be added.

2.2 Story 2

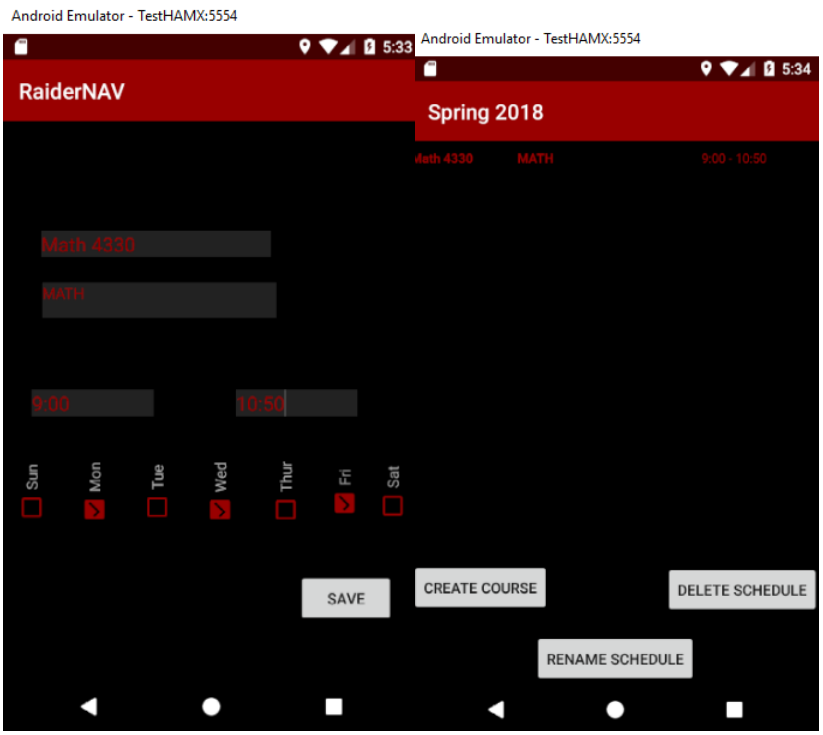
Use Case: Save Class schedule
Summary: App User saves a created class schedule
Actors: User
Dependency: View Saved Schedules
Precondition: The user has entered the course's details (name, building, days of week, and the starting and ending times) in the course edit screen.
Description:
1. User presses the Save button.
Alternatives:
1a. If the schedule saved for a specific day overlaps with regards to time with another schedule on the same day, the application displays an error message.
Postcondition: The class schedule is saved.

Title: Save Class schedule
Actors: User
Requirement: R2
Main Scenario:
User presses the Save button after entering course details
The class schedule is saved
Alternatives:
1a. The schedule saved for a specific day overlaps with regards to time with another schedule on the same day,
1a.1 The application displays an error message.

Test Situations:
1. User presses the Save button after entering course details, the class schedule is saved
2. The schedule saved for a specific day overlaps with regards to time with another schedule on the same day

Test Coverage:
Base: number of main and alternative scenarios = 2
Test situations cover all 2 cases
100% coverage of use case

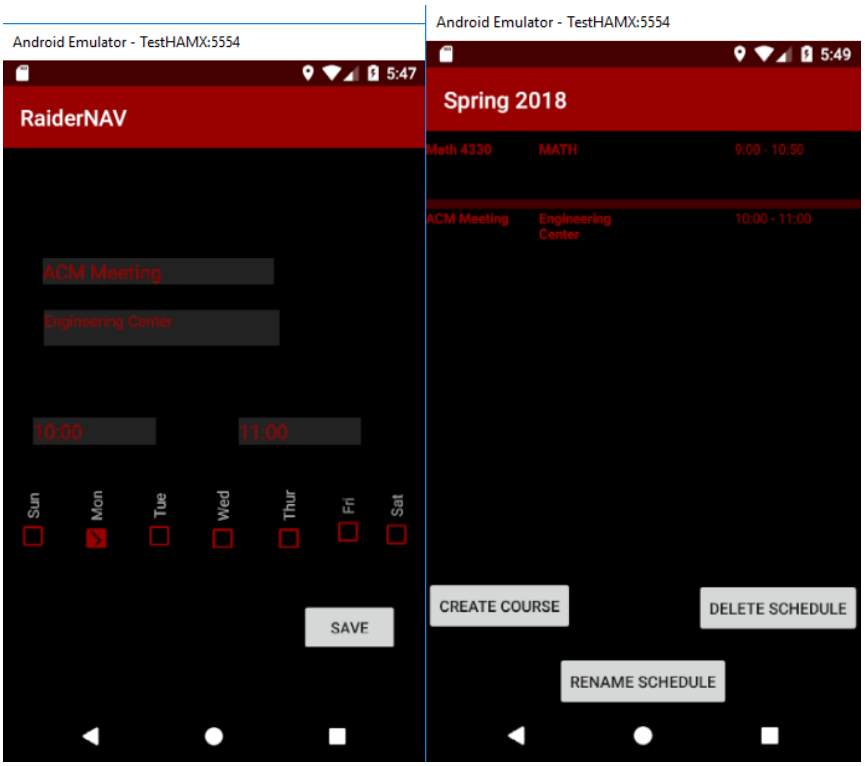
Test Record:
Test Case 1:
User presses the Save button after entering course details
The class schedule is saved



Test Case 2:

The schedule saved for a specific day overlaps with regards to time with another schedule on the same day

The class schedule is saved



Note on R2 Test:

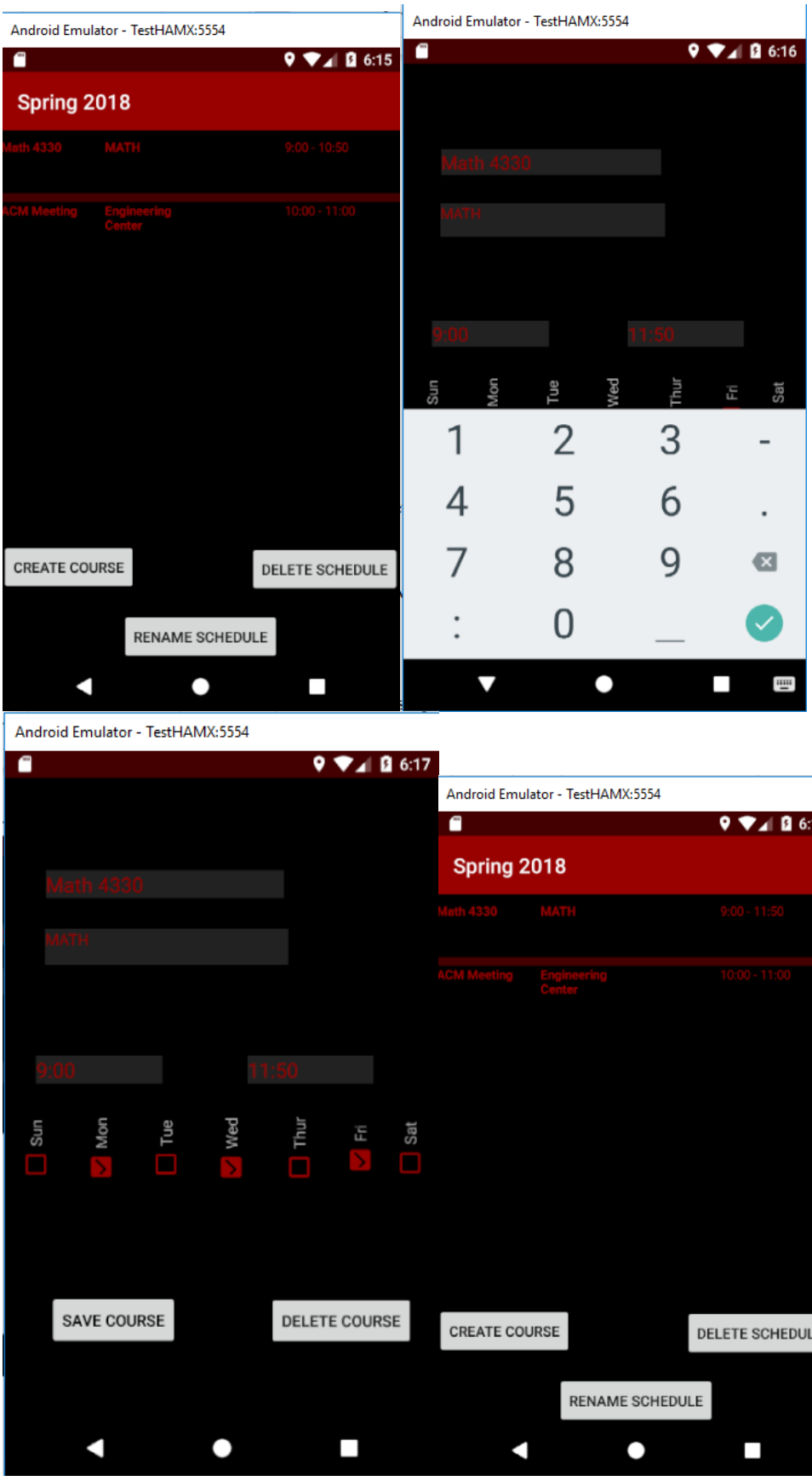
Time overlapped course should not be saved.

2.3 Story 3

Use Case: Edit Class schedule
Summary: App User edits a previously saved class schedule.
Actors: User
Dependency:
Precondition: The application should be in the My Schedules screen and the list of schedules must have at least one element.
Description:
1. The user presses desired schedule from list on My Schedules screen.
2. The application lists the courses saved to that schedule along with Create Course, Delete Schedule, and Rename Schedule buttons.
3. The user presses a desired course from the displayed list.
4. The application displays modifiable fields for the course name, building, start and end times, and days of the week, along with Save and Delete buttons.
5. The user fills in fields as desired and presses Save button.
Alternatives:
Postcondition: The user has edited a previously saved class schedule.
Title: Edit Class schedule
Actors: User
Requirement: R3
Main Scenario:
1. The user presses desired schedule from list on My Schedules screen.
2. The application lists the courses saved to that schedule along with Create Course, Delete Schedule, and Rename Schedule buttons.
3. The user presses a desired course from the displayed list.
4. The application displays modifiable fields for the course name, building, start and end times, and days of the week, along with Save and Delete buttons.
5. The user fills in fields as desired and presses Save button.
Alternatives: None

Test Situations:
Edited class schedule is saved
Test Coverage:
Base: number of main and alternative scenarios = 1
Test situations cover one case
100% coverage of use case

Test Record:
Test Case:
Edit Class schedule


Android Emulator - TestHAMX:5554
6:15
Spring 2018
Math 4330
MATH
9:00 - 10:50
ACM Meeting
Engineering Center
10:00 - 11:00
CREATE COURSE
DELETE SCHEDULE
RENAME SCHEDULE
Android Emulator - TestHAMX:5554
6:16
Math 4330
MATH
9:00
11:50
Sun
Mon
Tue
Wed
Thur
Fri
Sat
1 2 3 -
4 5 6 .
7 8 9
: 0 _
Android Emulator - TestHAMX:5554
6:17
Math 4330
MATH
9:00
11:50
Sun
Mon
Tue
Wed
Thur
Fri
Sat
SAVE COURSE
DELETE COURSE
Android Emulator - TestHAMX:5554
6:17
Spring 2018
Math 4330
MATH
9:00 - 11:50
ACM Meeting
Engineering Center
10:00 - 11:00
CREATE COURSE
DELETE SCHEDULE
RENAME SCHEDULE

2.4 Story 4

Use Case Name: Delete Class Schedule
Summary: This user case will allow user to delete a previously created class schedule.
Actor: User
Dependency: View Saved Schedule
Precondition: The application should be in the My Schedules screen and the list of schedules must have at least one element.
Description:
1. The user presses desired schedule from list on My Schedules screen.
2. The application lists the courses saved to that schedule along with Create, Delete, and Rename buttons.
3. The user presses the Delete Schedule button.
4. The application deletes the schedule from the schedule list and returns the user to the My Schedules screen.
Alternatives:
Postcondition: User has deleted the class schedule.

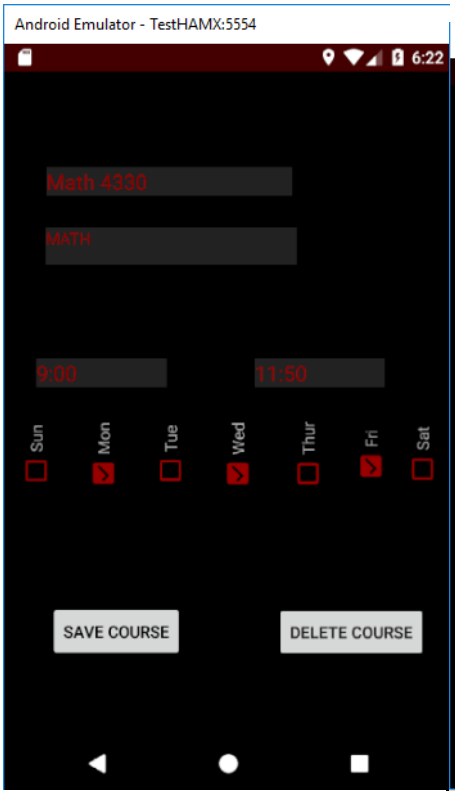
Title: Delete Class Schedule
Actors: User
Requirement: R4
Main Scenario:
1. The user presses desired schedule from list on My Schedules screen.
2. The application lists the courses saved to that schedule along with Create, Delete, and Rename buttons.
3. The user presses the Delete Schedule button.
4. The application deletes the schedule from the schedule list and returns the user to the My Schedules screen.

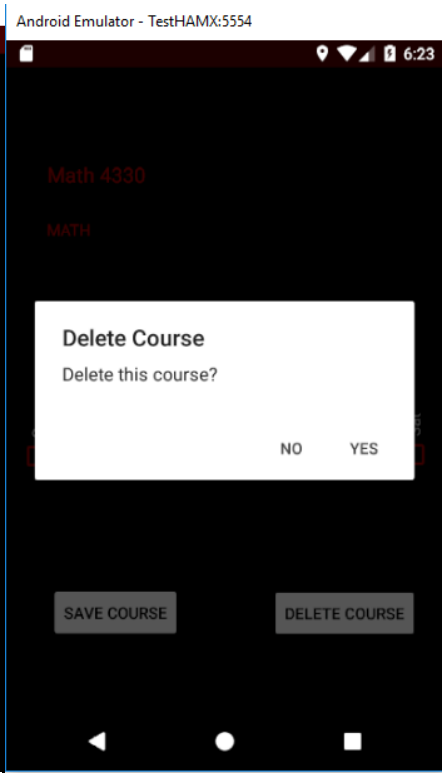
Alternatives:
None
Test Situations:
Class Schedule is deleted

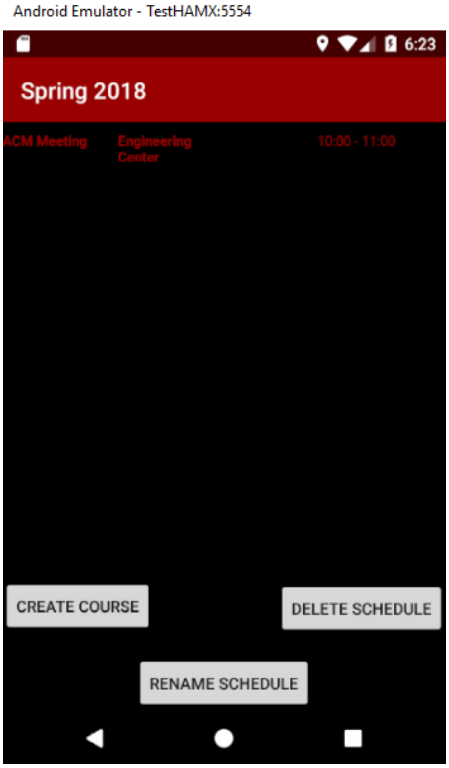
Test Coverage:
Base: number of main and alternative scenarios = 1
Test situations cover one case
100% coverage of use case

Test Record:

Test Case:
The user presses desired schedule from list on My Schedules screen.
The application lists the courses saved to that schedule along with Create, Delete, and Rename buttons.
The user presses the Delete Schedule button.
The application deletes the schedule from the schedule list and returns the user to the My Schedules screen.


Android Emulator - TestHAMX:5554
6:22
Math 4330
MATH
9:00
11:50
Sun
Mon
Tue
Wed
Thur
Fri
Sat
SAVE COURSE
DELETE COURSE


Android Emulator - TestHAMX:5554
6:23
Math 4330
MATH
Delete Course
Delete this course?
NO
YES
SAVE COURSE
DELETE COURSE


Android Emulator - TestHAMX:5554
6:23
Spring 2018
ACM Meeting
Engineering Center
10:00 - 11:00
CREATE COURSE
DELETE SCHEDULE
RENAME SCHEDULE

2.5 Story 5

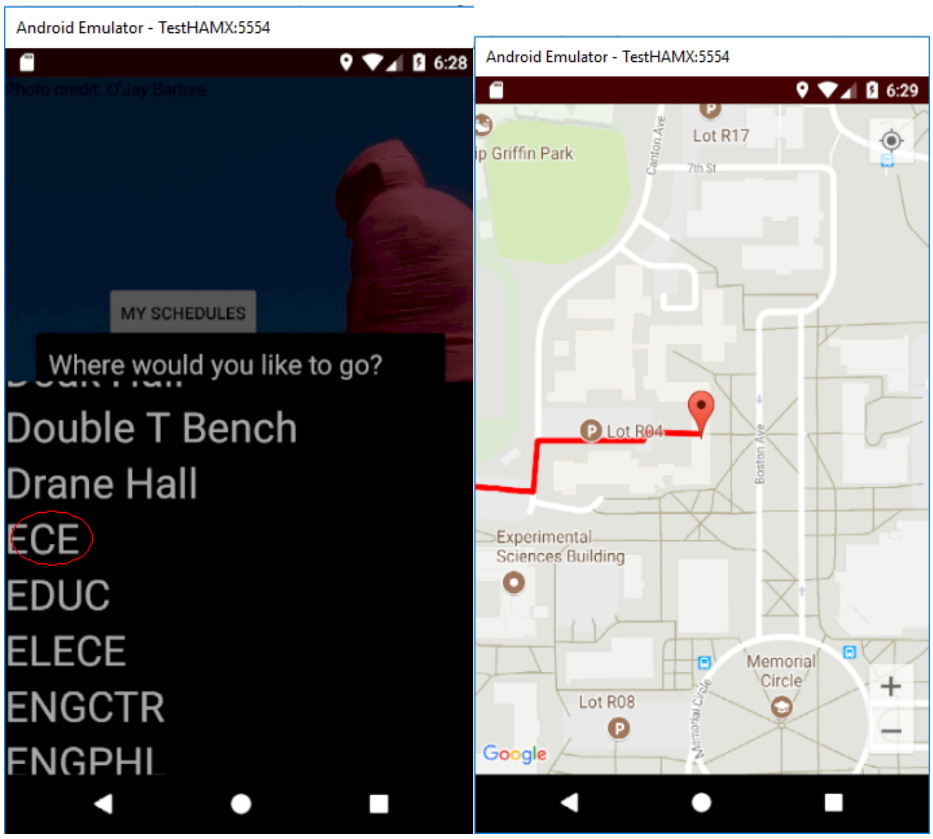
Use Case Name: Navigate to Campus Destinations
Summary: This use case allows users to select a building name/acronym from a drop-down menu which the system will then determine a geographic location for and add to the daily route.
Actor: User and Mapping Engine
Dependency: The name (or acronym) of the building must be known by the user.
Precondition: The user has selected the "Add course" or "Create Schedule" option.
Description:
1. The user selects the desired building name/acronym from the drop-down menu.
2. The user enters other pertinent details for the course/schedule and selects "Save".
3. The system passes the option selected to the CoordinateMap class.
4. The CoordinateMap class returns the location mapped to the desired selection.
Alternatives:
Postcondition: Coordinates representing the location of the user's selection have been received by the system.

Title: Navigate to Campus Destinations
Actors: User and Mapping Engine
Requirement: R5
Main Scenario:
1. The user selects the desired building name/acronym from the drop-down menu.
2. The user enters other pertinent details for the course/schedule and selects "Save".
3. The system passes the option selected to the CoordinateMap class.
4. The CoordinateMap class returns the location mapped to the desired selection.
Alternatives: None

Test Situations:
Test Coverage:
Base: number of main and alternative scenarios = 1
Test situations cover one case
100% coverage of use case

Test Record:

Test Case:
The correct location information was passed to system after user selected building.



2.6 Story 6

This section intentionally left blank.

2.7 Story 7

This section intentionally left blank.

2.8 Story 8

Use Case: Calculate Shortest Route
Actor: User
Dependency: None
Precondition: None.
Description:
1. The user determines starting point and destination they are going to
2. The app finds the shortest route from the user
Alternatives:
Postcondition: The shortest route between starting point and destination has been calculated.

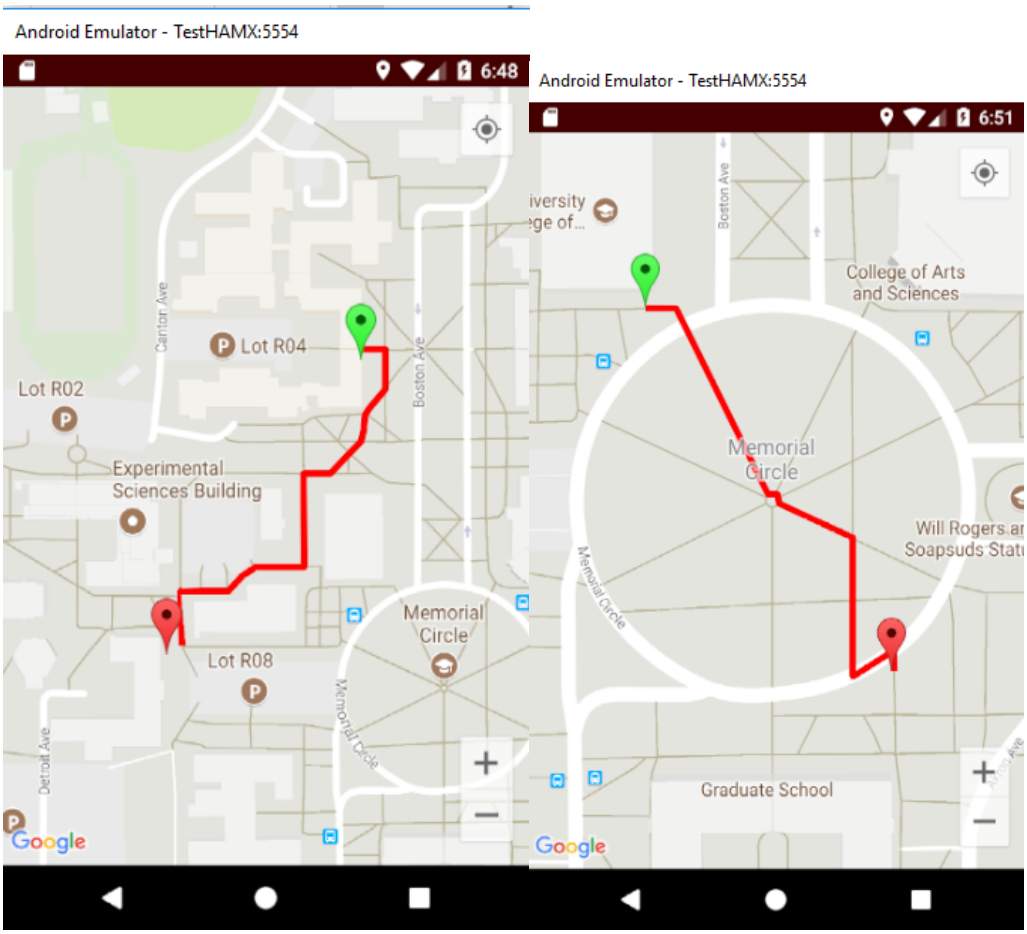
Title: Calculate Shortest Route
Actors: User and app

Requirement: R8
Main Scenario:
1. The user determines starting point and destination they are going to
2. The app finds the shortest route from the user

Test Situations:
Shortest route is generated on the map by the system.
Test Coverage:
Base: number of main and alternative scenarios = 1
Test situations cover one case
100% coverage of use case

Test Record:

Test Case:
1. The user determines starting point and destination they are going to
2. The app finds the shortest route from the user



2.9 Story 9

Use Case: Display current location
Actor: User
Dependency: None
Precondition: User has already calculated a route

Description:
1. User starts their route towards the destination
2. Current location is retrieved from the GPS
3. Every 3-5 seconds, the time sends an input to the GPS to recalculate location
4. Current location is constantly displayed to the user on the map

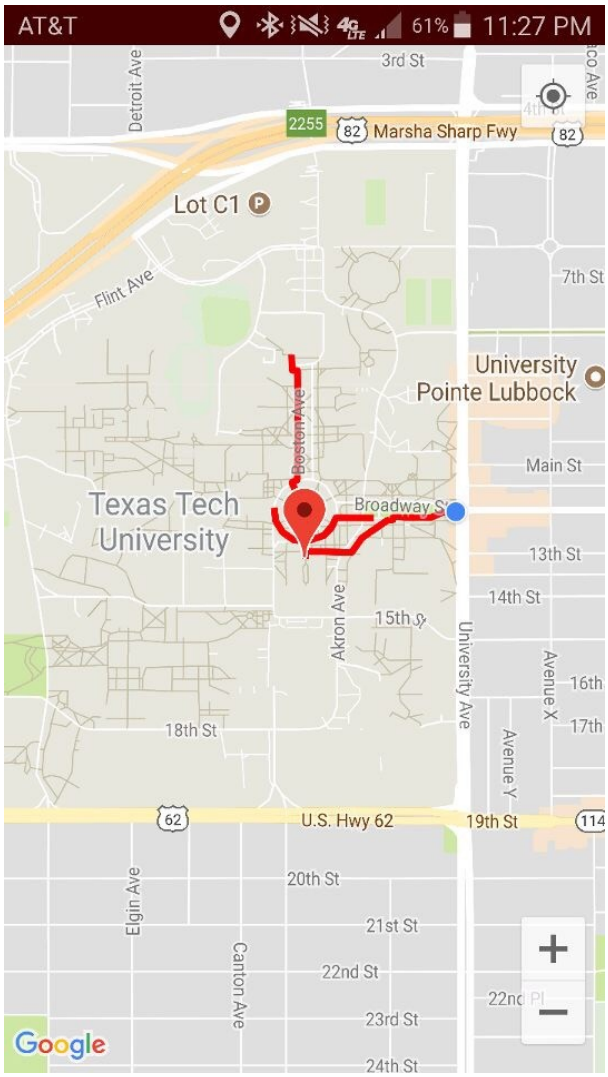
Postcondition:

Title: Display current location
Actors: App
Requirement: R9
Main Scenario:
App shows user's current location on the map.

Test Situations:
App shows user's current location on the map.
Test Coverage:
Base: number of main and alternative scenarios = 1
Test situations cover one case
100% coverage of use case

Test Record:

Test Case:
App shows user's current location (the blue dot) on the map.



2.10 Story 10

This section intentionally left blank.

2.11 Story 11

This section intentionally left blank.

2.12 Story 12

This section intentionally left blank.

2.13 Story 13

This section intentionally left blank.

2.14 Story 14

This section intentionally left blank.

2.15 Story 15

This section intentionally left blank.

2.16 Story 16

This section intentionally left blank.

2.17 Story 17

This section intentionally left blank.

2.18 Story 18

This section intentionally left blank.

2.19 Story 19

Use Case: Edit Class schedule
Summary: App User edits a previously saved class schedule.
Actors: User
Dependency:
Precondition: The application should be in the My Schedules screen and the list of schedules must have at least one element.
Description:
1. The user presses desired schedule from list on My Schedules screen.
2. The application lists the courses saved to that schedule along with Create Course, Delete Schedule, and Rename Schedule buttons.
3. The user presses a desired course from the displayed list.
4. The application displays modifiable fields for the course name, building, start and end times, and days of the week, along with Save and Delete buttons.
5. The user fills in fields as desired and presses Save button.
Postcondition: The user has edited a previously saved class schedule.

Title: Edit Class schedule
Actors: User
Requirement: R19
Main Scenario:
1. User click the existing schedule.
2. User edits the destination location, class time in the schedule.
3. User click the “save” button to finish editing
Alternatives: None

Test Situations:

Edited class schedule is saved

Test Coverage:

Base: number of main and alternative scenarios = 1

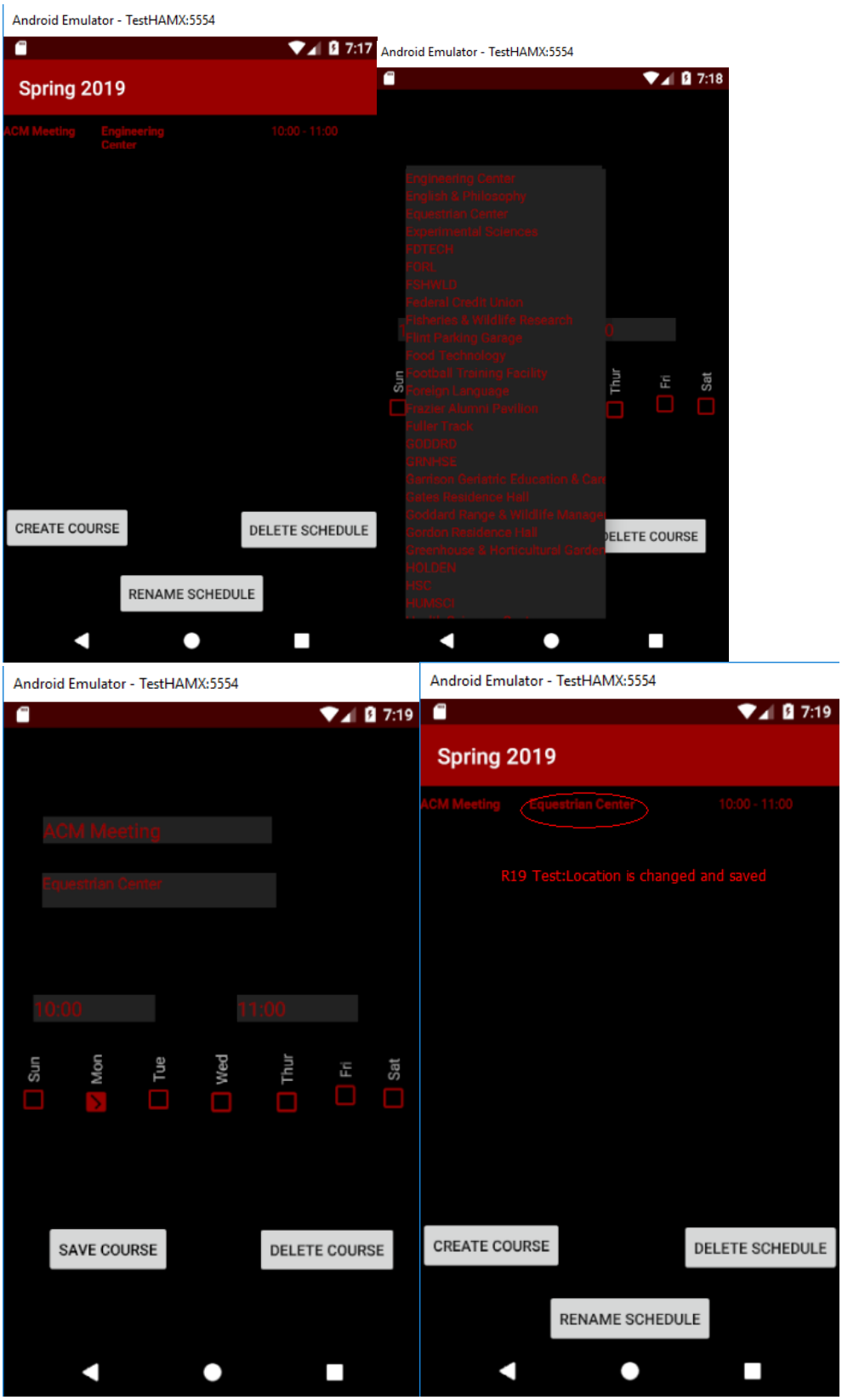
Test situations cover 1 case

100% coverage of use case

Test Record:

Test Case:

Edit schedule


Android Emulator - TestHAMX:5554
7:17
Spring 2019
ACM Meeting
Engineering Center
10:00 - 11:00
CREATE COURSE
DELETE SCHEDULE
RENAME SCHEDULE
Android Emulator - TestHAMX:5554
7:18
Engineering Center
English & Philosophy
Equestrian Center
Experimental Sciences
FDTECH
FORL
FSHWLD
Federal Credit Union
Fisheries & Wildlife Research
Flint Parking Garage
Food Technology
Football Training Facility
Foreign Language
Frazier Alumni Pavilion
Fuller Track
GODDRD
GRNHSE
Gates Residence Hall
Gordon Residence Hall
Greenhouse & Horticultural Garden
HOLDEN
HSC
HUMSCI
Sun
Thur
Fri
Sat
DELETE COURSE
Android Emulator - TestHAMX:5554
7:19
ACM Meeting
Equestrian Center
10:00
11:00
Sun
Mon
Tue
Wed
Thur
Fri
Sat
SAVE COURSE
DELETE COURSE
Android Emulator - TestHAMX:5554
7:19
Spring 2019
ACM Meeting
Equestrian Center
10:00 - 11:00
R19 Test:Location is changed and saved
CREATE COURSE
DELETE SCHEDULE
RENAME SCHEDULE

2.20 Story 20

This section intentionally left blank.

2.21 Story 21

Use Case: Display privacy policy
Summary: The app displays the privacy policy when the application is loaded the first time.
Actor: User
Precondition: User has opened the app for the first time.
Description:
1. The privacy implications of using the application is displayed to the user.
2. The user is prompted to select OK if he agrees to the terms.
Postcondition: The application home screen is loaded.

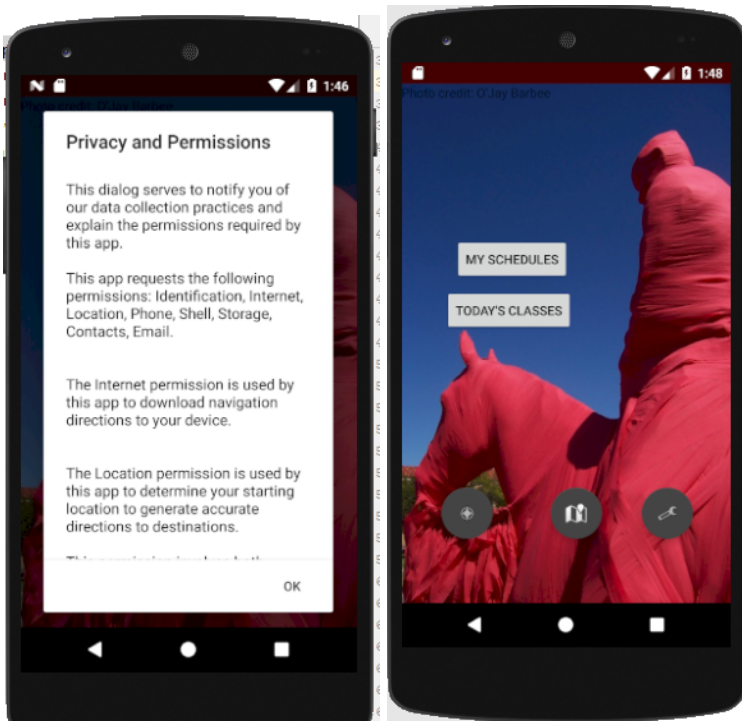
Title: Display privacy policy
Actors: User
Requirement: R21
Main Scenario:
1. The privacy implications of using the application is displayed to the user.
2. The user is prompted to select OK if he agrees to the terms.

Test Situations:
The privacy implication is displayed to the user.

Test Coverage:
Base: number of main and alternative scenarios = 1
Test situations cover all 1 cases
100% coverage of use case

Test Record:

Test Case:
User opens RaiderNav first time, the privacy implications is displayed to the user, after user click OK, APP shows the main screen.



2.22 Story 22

This section intentionally left blank.

2.23 Story 23

This section intentionally left blank.

2.24 Story 24

Use Case: Navigate to Unscheduled Location
Summary: The app shall allow the user to generate a map to a location which is not within any schedule and which will not be saved to any schedule.
Actor: User
Precondition: User is currently on the main screen.
Description:
1. The user presses the unscheduled location button.
2. The system displays the list of locations to choose from.
3. The user selects a location from the list and confirms the decision.
4. The system displays a map to the destination.

Alternatives:
3a. The user selects the “Cancel” button.
3a.1 The system returns the user to the main screen.

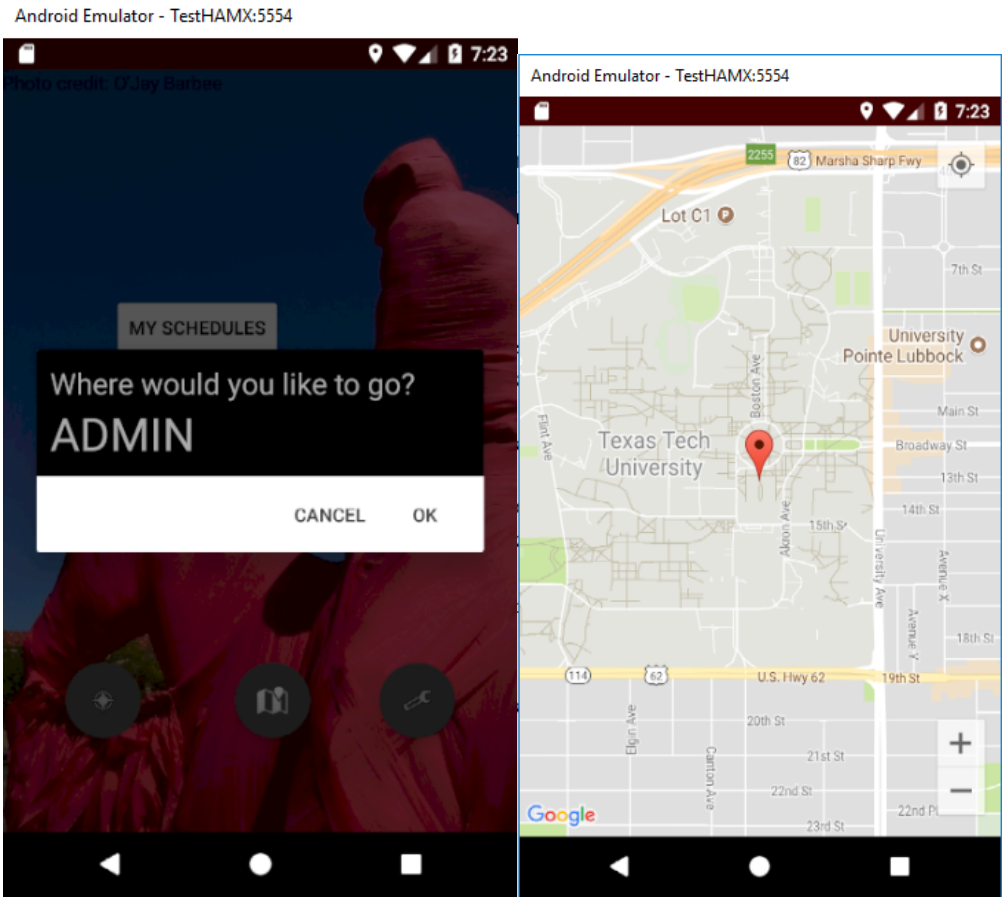
Postcondition: The user has generated a map to an unscheduled destination.

Title: Navigate to Unscheduled Location
Actors: User
Requirement: R24
Main Scenario:
1. The user presses the unscheduled location button.
2. The system displays the list of locations to choose from.
3. The user selects a location from the list and confirms the decision.
4. The system displays a map to the destination.

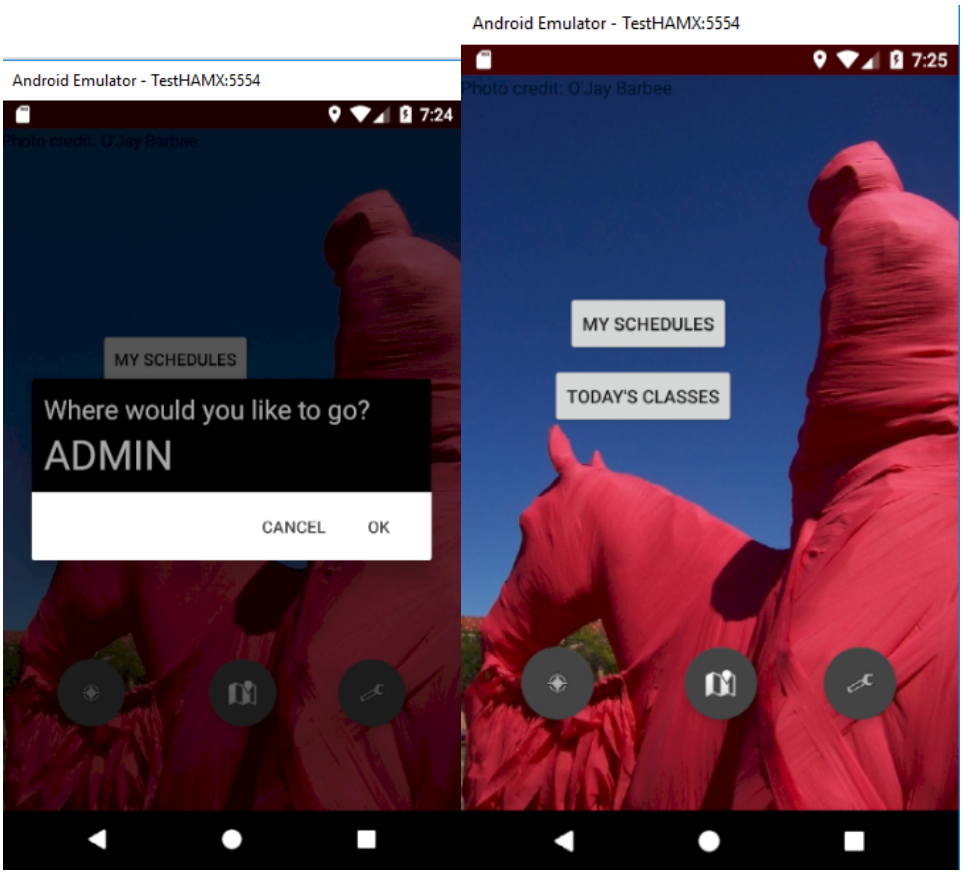
Alternatives:
a. The user selects the "Cancel" button.
3a.1 The system returns the user to the main screen.
Test Situations:
1. App shows the selected building's location on the map
2. App returns to the main screen after user selected "Cancel"
Test Coverage:
Base: number of main and alternative scenarios = 2
Test situations cover all 2 cases
100% coverage of use case

Test Record:

Test Case 1:
The user selects a location from the list and confirms the decision.



Test Case 2:
The user selects the “Cancel” button.



2.25 Story 25

This section intentionally left blank.

2.26 Story 26

This section intentionally left blank.

2.27 Story 27

This section intentionally left blank.

2.28 Story 28

This section intentionally left blank.

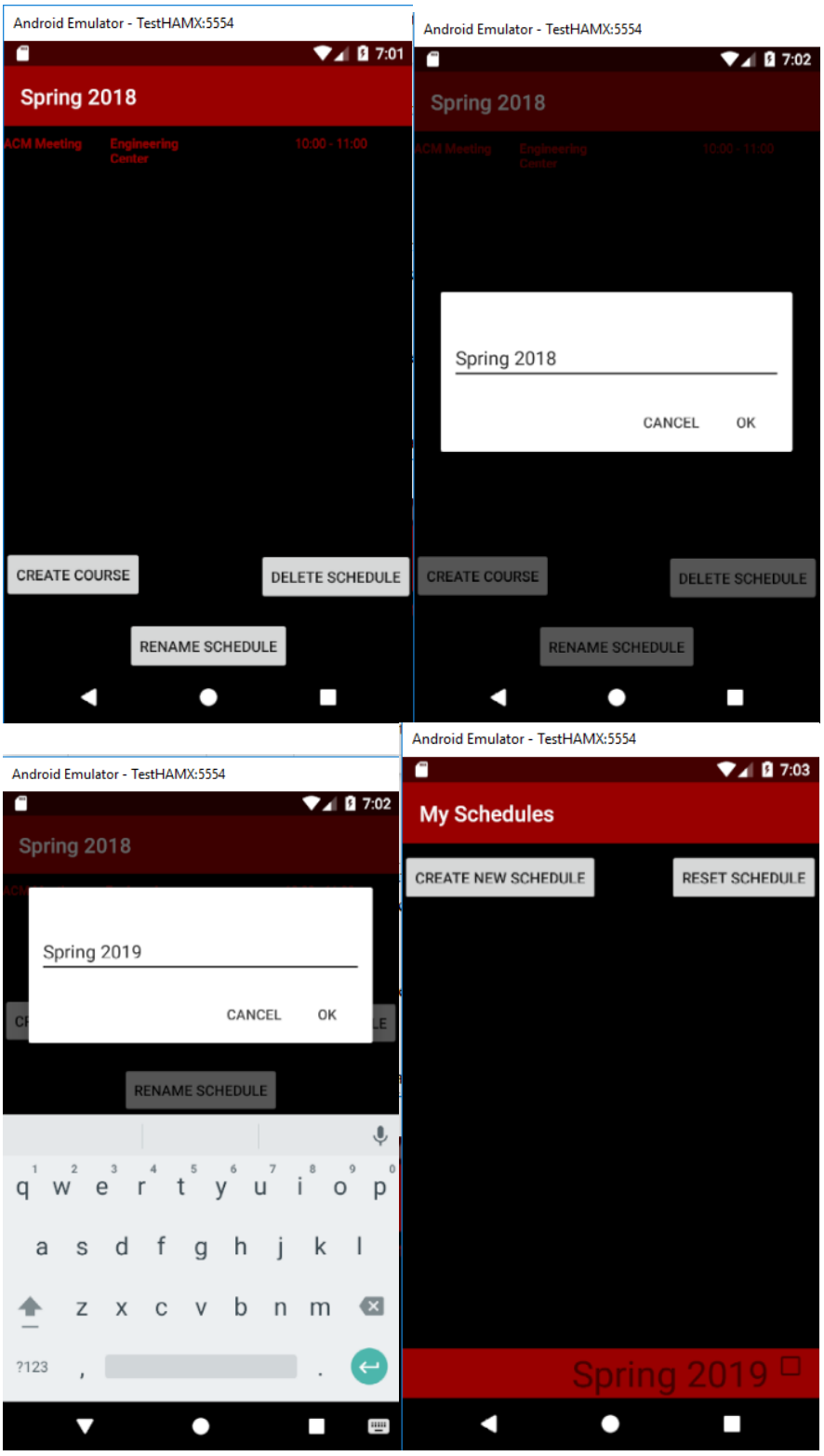
2.29 Story 29

Use Case: Rename Schedule
Summary: The app shall allow the user to rename a saved schedule.
Actor: User
Precondition: User previously saved a schedule in the app. User is currently on "My Schedules" screen.
Description:
1. The user selects a saved schedule.
2. The system displays the schedule details.
3. The user selects the "Rename Schedule" button.
4. The system displays a prompt showing the current schedule name.
5. The user enters a replacement name for the selected schedule using the Android keyboard and selects "Ok".
6. The system returns the user to the "My Schedules" screen.
Alternatives:
5a. The user selects the "Cancel" button.
Postcondition: The user has successfully renamed a saved schedule.

Title: Rename Schedule
Actors: User
Requirement: R29
Main Scenario:
1. The user selects a saved schedule.
2. The system displays the schedule details.
3. The user selects the "Rename Schedule" button.
4. The system displays a prompt showing the current schedule name.
5. The user enters a replacement name for the selected schedule using the Android keyboard and selects "Ok".
6. The system returns the user to the "My Schedules" screen.
Alternatives:
5a. The user selects the "Cancel" button.
Test Situations:
1. New schedule name is saved
2. The user selects the "Cancel" button.
Test Coverage:
Base: number of main and alternative scenarios = 2
Test situations cover all 2 cases
100% coverage of use case

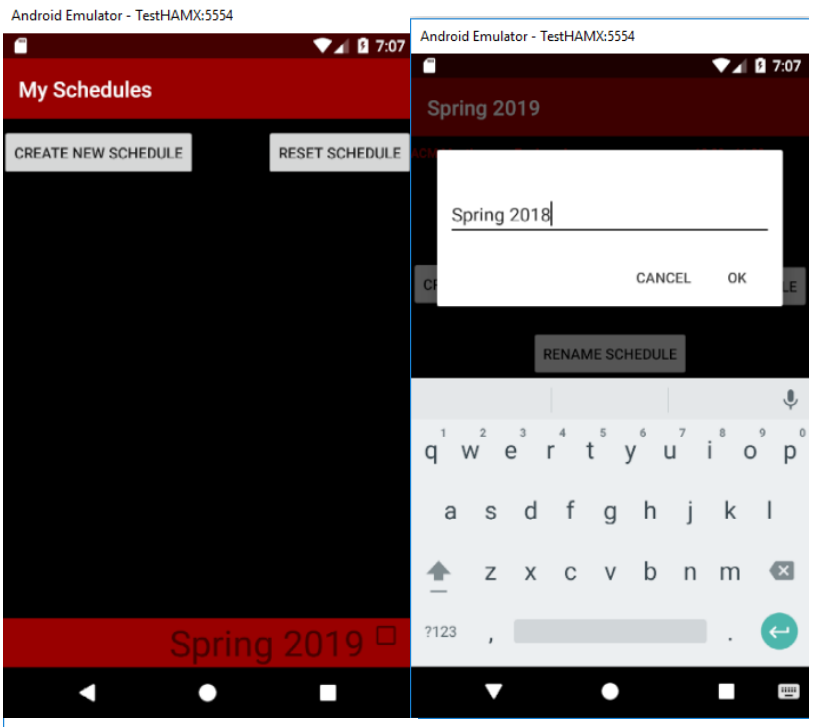
Test Record:

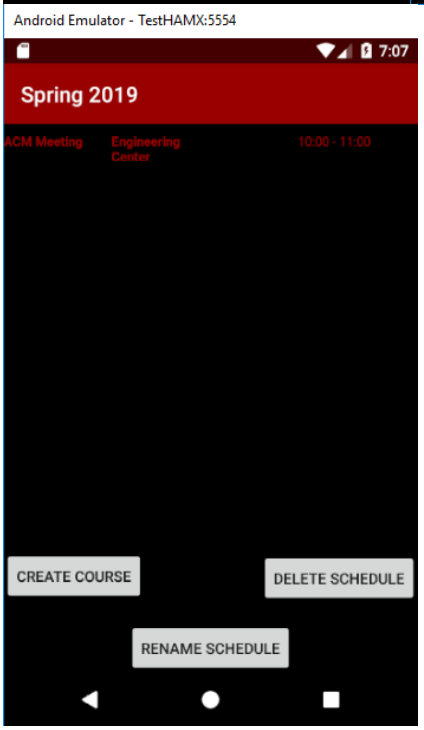
Test Case 1:
New schedule name is saved



Test Case 2:

The user selects the “Cancel” button.


Android Emulator - TestHAMX:5554
7:07
My Schedules
CREATE NEW SCHEDULE
RESET SCHEDULE
Spring 2019
Android Emulator - TestHAMX:5554
7:07
Spring 2019
Spring 2018
CANCEL
OK
RENAME SCHEDULE


Android Emulator - TestHAMX:5554
7:07
Spring 2019
ACM Meeting
Engineering Center
10:00 - 11:00
CREATE COURSE
DELETE SCHEDULE
RENAME SCHEDULE

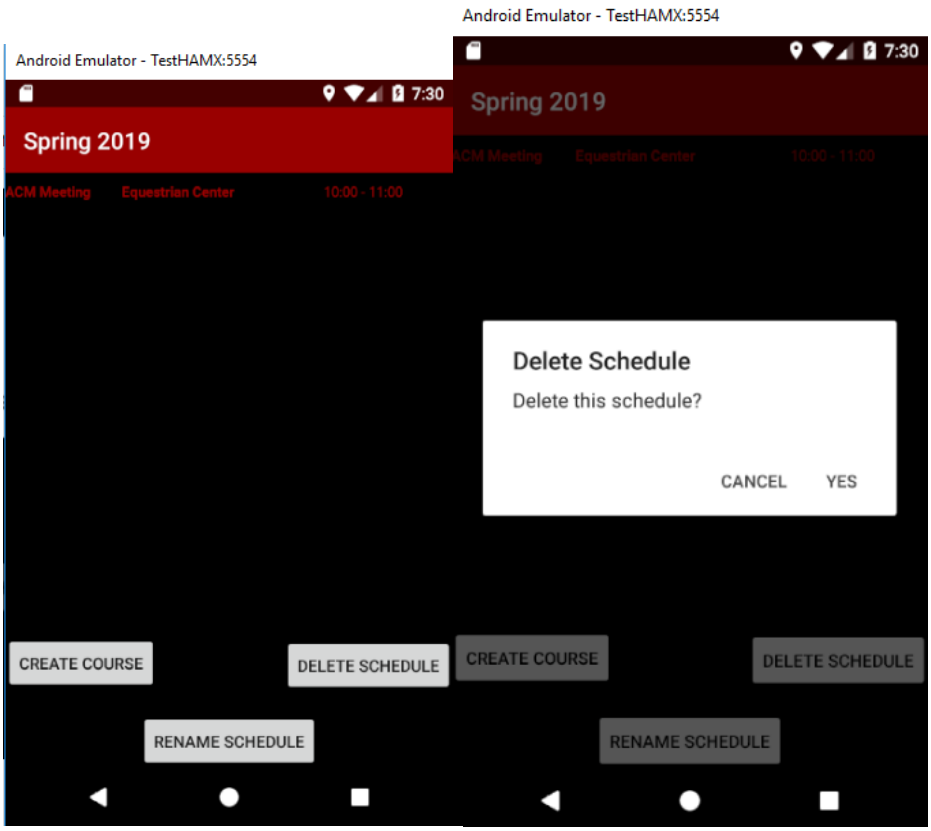
2.30 Story 30

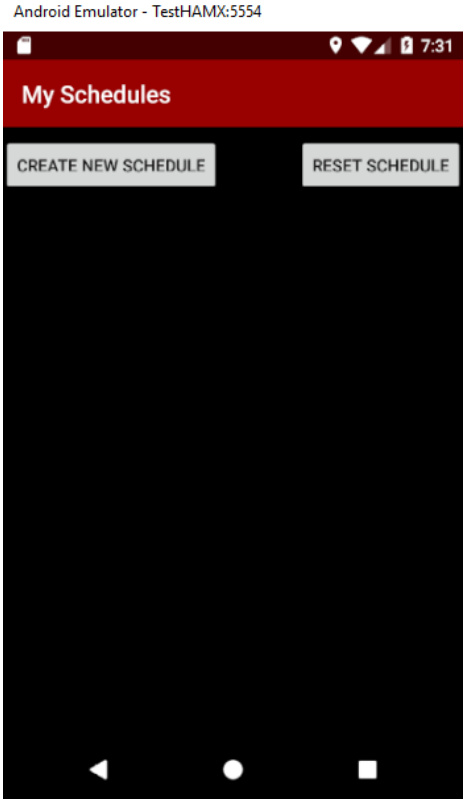
Use Case: Delete Notification
Summary: Before the user deletes a schedule, the user must be given a notification to ensure that the user wants to delete the schedule, because all data will be lost.
Actor: User
Precondition: The user is at the schedule screen.
Description:
1. The user presses the Delete button.
2. A notification is given to the user to confirm the deletion, along with an option to cancel and not delete the schedule and a Delete button to delete the schedule.
3. If the user presses Delete, then the schedule will permanently be deleted and the user is then taken back to the schedules page.
Alternatives:
3a. If the user presses cancel, then they will be taken back to the main schedule page.
Postcondition: The user is returned to the schedule screen if they pressed Cancel or to the My Schedules screen if the user confirmed schedule deletion.

Title: Delete Notification
Actors: User
Requirement: R30
Main Scenario:
1. The user presses the Delete button.
2. A notification is given to the user to confirm the deletion, along with an option to cancel and not delete the schedule and a Delete button to delete the schedule.
3. If the user presses Delete, then the schedule will permanently be deleted and the user is then taken back to the schedules page.

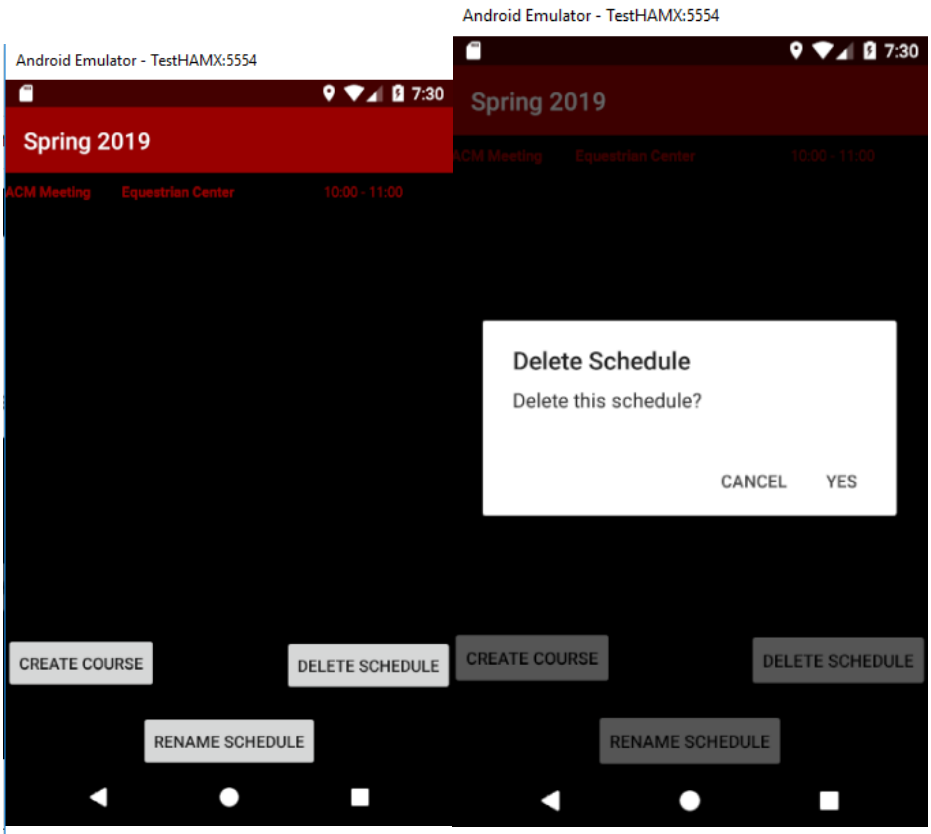
Alternatives:
3a. If the user presses cancel, then they will be taken back to the main schedule page.
Test Situations:
1.Schedule is deleted after notified
2.Schedule keep remained after user chooses “Cancel” the delete operation.
Test Coverage:
Base: number of main and alternative scenarios = 2
Test situations cover all 2 cases
100% coverage of use case

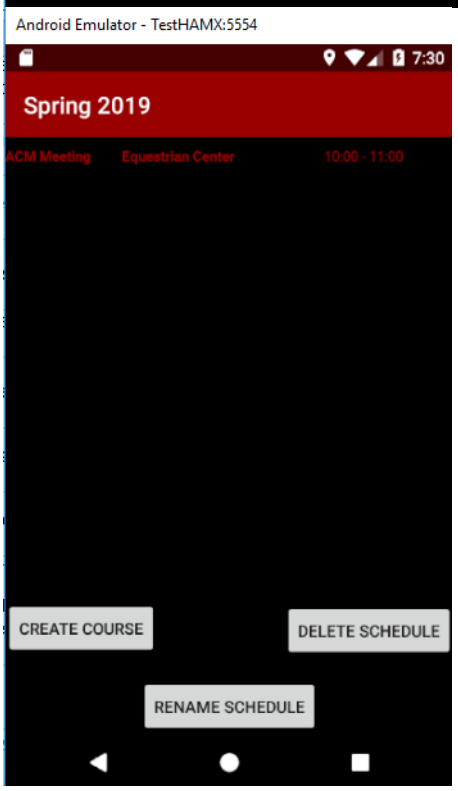
Test Record:
Test Case 1:
Schedule is deleted after notified





Test Case 2:
Schedule keep remained after user chooses “Cancel” the delete operation.


Android Emulator - TestHAMX:5554
7:30
Spring 2019
ACM Meeting
Equestrian Center
10:00 - 11:00
Delete Schedule
Delete this schedule?
CANCEL
YES
CREATE COURSE
DELETE SCHEDULE
RENAME SCHEDULE


Android Emulator - TestHAMX:5554
7:30
Spring 2019
ACM Meeting
Equestrian Center
10:00 - 11:00
CREATE COURSE
DELETE SCHEDULE
RENAME SCHEDULE

3. Acceptance Testing

Acceptance Criteria

User stories 1-6, 8, 10-16, 21, 24, 29, and 30 must be implemented.
Usability test scores must represent at least a 90% success rate.

Acceptance Evaluation

User stories 1-5, 8, 19, 21, 24, 29, and 30 implemented
User stories 6 and 10-16 NOT implemented.
User test score represents 92.8% success rate.

Acceptance Decision

Acceptance Rejected

Reason:

User stories 6 and 10-16 NOT implemented.