The Stack 5

AR-MAP

TEST PLAN

Date: 12/3/2025

Italicized text is to be replaced with your content; Do not leave text italicized.

Table of Contents

Introduction

The Test Plan has been created to communicate the test approach to team members. It includes the objectives, scope, schedule, risks and approach. This document will clearly identify what the test deliverables will be and what is deemed in and out of scope.

1.1 Objectives

Briefly summarize what the system will do and why.

The AR-MAP system will let students and visitors explore the campus using a digital map / augmented reality. Users can see their live location, view building details, and navigate more easily. The goal is to make it easier for new and returning students to find buildings and classrooms quickly, using GPS and AR features that make navigation simple and interactive.

Sprint 1 of the project will deliver *list features and why these features are the focus of this sprint.*

1.2 Team Members

Resource Name	Role *(examples are given below)*
Vicky	Developer
Priscilla	Project Manager / Tester
Tedros	Tester
Edidiong	Developer/ Tester
Aimua	Developer/ Tester

2 Scope

The initial sprint will include 'must have' requirements. These and any other requirements that get included must all be tested.

The following sections indicate what is tested during each sprint. The scope of testing is determined at the beginning of the current sprint.

At the end of Sprint 1, a user must be able to:

1. List the things a user can do with the system. These should be taken from the requirements doc (with the correct User Story ID).

   1. Open the AR-MAP and view a base map with the AR marker labels.

   2.Use the device GPS to display users current location.

   3. Interact with the AR interface through touch or gesture.

   4. Zoom and pinch map as needed for precise navigation

2. Tested the database for the firebase.

Mention other kinds of testing that will be conducted (e.g. security).

Smoke tests

1. *App boots on physical Android test device & simulator*
2. *AR session starts; camera permission flow succeeds*

Unit/Integration (initial)

1. *Unit tests for location service wrapper & POI repository*
2. *Integration: map loads → markers render from Firestore mock*

At the end of Sprint 2, a user must be able to:

1. *List the things a user can do with the system. These should be taken from the requirements doc (with the correct User Story ID).*

Mention other kinds of testing that will be conducted (e.g. security).

Smoke tests

3. *App boots on physical Android test device & simulator*
4. *AR session starts; camera permission flow succeeds*

Unit/Integration (initial)

3. *Unit tests for location service wrapper & POI repository*
4. *Integration: map loads → markers render from Firestore mock*

At the end of Sprint 3, a user must be able to:

1. *List the things a user can do with the system. These should be taken from the requirements doc (with the correct User Story ID).*

Mention other kinds of testing that will be conducted (e.g. security).

Interactive Elements Test:

Checking the functionality of zooming, panning, rotating, and other user interactions.

Map Display and Rendering Test: Verifying correct loading, display, and rendering of campus map and rooms (e.g. points of interest).

Assumptions / Risks

2.1 Assumptions

This section lists assumptions that are made specific to this project.

1. Test data and credentials will be available before testing begins.

2. All APIs and backend services will be functional and accessible during test cycles.

3. All testers have required device access (Smartphones).

4. Users will be vaguely familiar with campus layout

2.2 Risks

The following risks have been identified and the appropriate action identified to mitigate their impact on the project. The impact (or severity) of the risk is based on how the project would be affected if the risk was triggered. The trigger is what milestone or event would cause the risk to become an issue to be dealt with.

#	Risk	Impact	Trigger	Mitigation Plan
1	Scope Creep – as testers become more familiar with the tool, they will want more functionality	High	Delays in implementation date	Each iteration, functionality will be closely monitored. Priorities will be set and discussed by stakeholders. Since the driver is functionality and not time, it may be necessary to push the date out.
2	Weekly delivery is not possible because the developer works off site	Medium	Product did not get delivered on schedule	implement daily stand-ups and mid-sprint code reviews.
3	AR libraries or SDK updates break existing functionality	High	SDK version updates	Keep tested values and retest when upgrading libraries.
4	Limited Test hardware	medium	Wait on devices to arrive	Use shared test results

5	Limited information on room layouts	medium	Not finding the data	Quickly ask campus resources like IT department for room data

3   Test Approach

The project is using an agile approach, with 3-week sprints. *Mention how you will conduct testing during the sprint in terms of the techniques you plan to do and when. Add a new subsection for each sprint.*

Sprint 1

1. Focus on unit testing and manual UI validation of base AR setup.

2. Verify GPS tracking accuracy, map load times, and UI stability.

3. Keep track of all issues for immediate review

Sprint 2

1. Focus more on unit testing and manual UI validation of base AR setup.

2. Still Verify GPS tracking accuracy, map load times, and UI stability in unity.

3. Log & track track of all issues for immediate review

Sprint 3

1.   Integration test between Search → Database → AR Marker rendering

2.   Authentication tests

3.   Usability test (user completes navigation in < 2 minutes)

3.1   Test Automation

Discuss the role of automated testing and how you plan to conduct any (including tools).

Automated testing will be introduced beginning in Sprint 3 to improve regression testing speed and reduce manual testing effort as the application grows in complexity. Automation is not replacing manual testing but will be used to repeatedly validate core functionality that must remain stable across sprints.

Role of Automated Testing

Ensures that critical features (such as login, database access, and AR navigation services) remain functional after new updates.

Reduces time spent on repetitive regression testing between sprints.

Helps catch breaking changes early, especially when Unity/AR SDK versions are upgraded.

3.2   Test Cases (Black Box)

3.2.1   Feature 1 (Name this based on the name in your Req doc)

Have a table for the test cases needed to test the User Story

Test Case ID	Description	Requirements Trace	Directions	Expected Output
TC-001	UI Accessibility through the WWW.	List the user story/stories this is linked to	1. Open a web browser and go to the address of the RRS.	The home page for the RRS should be displayed.

3.2.2   Feature n (Name this based on the name in your Req doc)

Have a table for the test cases needed to test the User Story. A sample is below

Test Case ID	Description	Requirements Trace	Directions	Expected Output
TC-101	Prepare to request a reservation at a restaurant.	RES-002 RES-003 RES-005 RES-006	2. Execute TCB-004.	All fields necessary to fill in the data specified in the directions are present.
TC-102	Request a reservation at a restaurant.	RES-001	1. Execute TC-101. 1. Click "Continue" button. 2. Click "Make Reservation!" button. 3.	A table reservation confirmation page should be displayed. Otherwise, a page should be displayed saying that the reservation could not be made.
TC-103	Cancel a previous reservation.	RES-007	1. Successfully make a reservation using TC-102. 2. Click "Return to Restaurant" button. 3. Click "Modify / Cancel Existing Reservation" button. 4. Enter confirmation number received from TC-003 in the "Reservation Confirmation Number" text box. 5. Enter the email address used in TC-003 in the "Email Address" text box. 6. Click "Modify/Cancel Reservation" button. 4. Click "Cancel Reservation" button.	A reservation cancellation confirmation page should be displayed.

3.3   Test Cases (White Box)

3.3.1..n Name by Feature (like above)

Organize the test cases using the table below. Be sure to have a directory in your repository for your test suite that follows a naming scheme that matches the items in the table below.

Like section 3.2, you will add to the table as you progress through each sprint.

Test Case ID	Description	Directions/Goals	Expected Output
TC-1001	Selecting a restaurant to visit.	Follow TCB-001 when there are no restaurants available. Does the system handle not having a restaurant to select?	Should load the home page without errors. However, there are no database checks for null pointers so if the DB fails, there will be an error.
TC-1002	Attempt to go to a restaurant page without selecting a restaurant.	Follow TCB-001 only do not select a restaurant. Does the system detect that no restaurant is selected.	The system should force the user to select a restaurant. Should fail however since there is no null pointer check in the code.

TC-1003	Check site links.	Browse all available pages clicking on each link available to make sure they point to pages. Also, check graphics as well.	Each page should link properly to intended pages.
TC-1004	Ensure data validation is working properly.	Follow TCB-006 only with improper input by leaving fields blank. Also attempt to use improperly formatted email and phone numbers, i.e. The last name for both fields.	System should check to make sure that fields are filled in and warned about otherwise. If there is an improper format for the email and phone number, that should be marked as well.
TC-1005	Attempt to use the system when the database is down.	Access any page without the database running. Check to make sure the error seems reasonable.	Although not a requirement, if the database connection cannot be made, a reasonable error message would be nice.
TC-1006	Attempt to use the system when JavaScript is disabled in the user's browser.	Can you still make a reservation with blanks in the field if you disable JavaScript? See how verbose the error handling is.	Since JavaScript does the error handling, not the code, disabling JavaScript should allow bad input.
TC-1007	Attempt to use apostrophes in text fields whose values get used in SQL statements.	Follow TCB-006 but use O'Conner as the last name. Checking to make sure data checking includes escaping characters.	SQL uses the apostrophe character as a special character, so does the system escape the apostrophe character if it appears in a name to prevent a bad SQL call.
TC-1008	Bounds checking on the Erlang implementation.	Use the Erlang page and try using a blank field or negative numbers.	Should fail since there is no error handling to round or check the input to the Erlang implementation.
TC-1009	Add an incentive directly to the database and ensure it is displayed.	1. Open a web browser and go to http://rrs.se.rit.edu/rrs/sql.jsp. 2. Enter the SQL statement "`INSERT INTO incentives (restaurant_id, start_time, end_time, description) VALUES (0, sysdate, TO_DATE('MM/DD/YYYY', '12/31/2002'), '<B>TC-1009 Incentive</B> <I>This is a new incentive!</I>')`" 3. Execute TCB-001.	The incentive just entered should be displayed on the page.

4 Test Environment

For example, A new server is required for the web server, the application and the database.

5 Test Schedule

Task Name *(sample is below, focus on spring 1 to start)*	Start	Finish	Effort	Comments
Test Planning	9/28	9/28	100	
Review Requirements documents	9/28	9/28	100	
Create initial test estimates		Not done		
Learn new test resources	10/8	10/8	100	
First deploy to QA test environment		Not done		
Functional testing – Sprint 1	10/11	10/12	100	
Iteration 2 deploy to QA test environment		Not done		
Functional testing – Sprint 2	10/28	11/7	100	
System testing	11/6	11/7	100	
Regression testing	10/31	11/7	100	
Usability Testing	11/07	11/23	100	
Resolution of final defects and final build testing	11/19	Not done	50%	
Deploy to Staging environment		Not done	0	
Performance testing	11/12			
Release to Production		Not done		