TEST PLAN FOR

VAMOOSE

Note that you can refine your testing plan as the project development goes. Keep the change log as follows:

ChangeLog

Version	Change Date	By	Description
v1	28-02-2025	All members	Initial Test Plan: add unit tests to core features and set up Continuous Integration (CI) for automated testing on pull requests.
v2	03-16-2025	All members	Expanded scope to cover all endpoints and new features (Interactive Map, Chat & Polls, Itinerary) with updated unit, integration, and manual acceptance tests.

1 Introduction

1.1 Purpose

- Define the purpose of testing for **Vamoose!**
- Ensure all features function correctly and meet user expectations
- Identify bugs early and maintain high software quality

1.2 Scope

This test plan applies to:

- Backend APIs *(Node.js, Express, Prisma, PostgreSQL)*
    - Covers API endpoints, database interactions, and business logic
- Trip Planning, Member Management, Expense Tracking, Interactive Map & Location Sharing, Real-Time Chat & Polls, and Itinerary Builder with Notifications
    - Tests all features related to trip creation, member roles, invitations, financial tracking, map functionalities, chat functionalities, poll functionalities, and itinerary event management with notifications

1.3 Testing Objectives

- Verify trip creation, updates, and deletions
- Ensure trip membership and invitations work correctly
- Test expense tracking and calculations
- Validate authentication and authorization mechanisms
- Validate interactive map functionalities and location sharing features
- Confirm real-time chat messaging, reactions, and poll operations function as expected
- Validate itinerary creation, updates, deletion, assignments, and notification triggers
- Detect performance bottlenecks

---

2 Functional Requirements

2.1 Core Features

Trip Planning Dashboard

- Users can create, update, and delete trips
- Users can view a dashboard with past and upcoming trips

Trip Member Management

- Admins/creators can invite members to a trip
- Users can accept or reject invitations
- Admins/creators can remove members
- Users (except the creator) can leave a trip

- Admins/creators can promote/demote member roles

Group Expense Tracking & Cost Splitting

- Users can add and delete expenses
- Users can set and update a trip budget
- Users can split expenses among trip members

Interactive Map & Location Sharing

- Users can pin/unpin a location for a trip
- Users can view a shared map with pinned locations
- Users can share or disable live location sharing
- Users can search for nearby places and request directions

Real-Time Chat & Polls

- Users can send and fetch chat messages
- Users can react to or remove reactions from messages
- Users can create polls, cast votes, complete polls, and delete polls/votes
- Users receive notifications (e.g., new messages, poll creations)

Itinerary Builder

- Users can create, update, and delete itinerary events
- Users can assign/unassign members to/from events
- Users can add, update, and delete event notes
- Users receive notifications (e.g., event reminders)

User Authentication & Account Management

- Users can register, log in, and log out securely

---

3 Non-Functional Requirements

3.1 Performance and Scalability

- The application should handle multiple concurrent users
- The system should scale to support growing data & traffic

3.2 Security & Privacy

• Encryption must be used for sensitive user data

• Strong password policies (e.g., 8+ characters, uppercase, number, special character)

• Unauthorized users cannot access private trips or expenses

3.3 Usability and Accessibility

- User-friendly interface, accessible on both desktop & mobile

3.4 Maintainability and Extensibility

- Code should support **future enhancements** without major restructuring

---

4 Roles and responsibilities

Name	Net ID	GitHub username	Role
Raman Bhandari	bhandar1	ramanbhandari	Full Stack Developer / Testing Manager
Chukwunaza Chukwuocha	chukwuo1	Chuks-x	Backend Developer / QA Analyst
Anmolpreet Khangura	khangura	anmolKhangura	Full Stack Developer / QA Analyst
Pritha Das	dasp4	prithaxx	Backend Developer
Anmolpreet Singh	sin121	Qprah	Full Stack Developer

---

5 Test Methodology

5.1 Test Levels

Test Level	Description
Unit Testing	Test individual functions (controllers) & components (Jest)
Integration Testing	Validate API-to-database interactions (Supertest and Jest); ensure endpoints correctly persist and retrieve data
Acceptance Testing	Manual end-user testing: Are in the documentations folder in the Repo: https://github.com/ramanbhandari/vamoose

5.2 Test Cases

Core Feature 1: Trip Planning Dashboard

Test Scenario	Expected Result	Type
Create a new trip with valid data	Trip is successfully created	Unit
Attempt to create a trip with missing fields	API returns 400 with an error message	Unit
Attempt to fetch a single trip	should fetch a trip successfully when authorized	Unit
Attempt to fetch trips with a valid filters	should fetch trips successfully when valid filters are provided	Unit
Attempt to fetch trips with filters that don't match	should return 200 with an empty list if no trips match the filters	Unit
Attempt to delete a trip with member/admin role	should return 403 if non-creator tries to delete	Unit

Attempt to delete a trip with creator role	should delete a trip successfully	Unit
Attempt to delete multiple trips with valid trip ids with creator role	should delete multiple trips successfully	Unit
Attempt to update trip details with valid data and creator role	should update a trip successfully	Unit
Attempt to update trip details with member role	should return 403 if member tries to update	Unit
Create a trip	Trip data is persisted and matches the input	Integration
Fetch trip details	Correct trip data is retrieved	Integration
Update trip details	Updated trip data is persisted and matches the new input	Integration
Delete a trip	The trip is removed from the database	Integration

Core Feature 2 : Trip Member Management

Test Scenario	**Expected Result**	**Type**
Attempt to create an invite for members not part of the trip	should create an invite successfully	Unit
Attempt to create an invite for members part of the trip	should return 400 if user is already a member	Unit
Attempt to validate an invite against the right user	should validate invite successfully	Unit
Attempt to update a members role as a creator	should update a member role successfully as a creator and return member details	Unit
Attempt to fetch a single member part of a valid trip	should return a single trip member successfully	Unit
Attempt to fetch a trip member not part of trip	should return 403 if requester is not a member of the trip	Unit
Attempt, as a non-creator, to	should allow a member to leave the trip	Unit

leave a trip		
Attempt, as a creator, to leave the trip	should prevent the creator from leaving	Unit
Attempt to remove multiple members as a creator	should successfully remove multiple members	Unit
Attempt to remove multiple members as a creator where the batch contains the creator	should return 403 if the creator is being removed	Unit
Invite a user to a trip	Invitation is saved in the database	Integration
Accept an invitation for a trip	The member is added to the trip in the database	Integration
Remove a member from a trip	The member is removed from the trip in the database	Integration

Core Feature 3 : Group Expense Tracking & Cost Splitting

Test Scenario	**Expected Result**	**Type**
Add expense to a trip without specifying users in expense split	Expense is created successfully and split equally among all members of the trip	Unit
Add expense to a trip with specified users in expense split	Expense is created successfully and split equally among specified members	Unit
Fetch single trip expense	Should return the trip expense	Unit
Delete single expense from trip	Should delete and return the deleted expense	Unit
Delete multiple expenses from trip	Should delete the trips and return their ids	Unit
Attempting to delete multiple expenses from trip where some of the expenses are invalid	Should partial delete the valid expenses and return the ids of the deleted and ignored expenses	Unit
Attempting to delete single or multiple expenses including an expense which the user is not a part of	Should return 403 for single expense deletion attempt and 404 for multiple, with appropriate message	Unit

Any Expense CRUD operation involving users who are not part of the trip	Should return 403 with appropriate error message	Unit
Any Expense CRUD operation without authorization	Should return 401 if userId is not present.	Unit
Any Expense CRUD operation with invalid or missing fields	Should return 400 with appropriate error message	Unit
Any unexpected error while operating on an expense	Should return 500 with appropriate error message	Unit
Add an expense to a trip	Expense is recorded correctly in the database	Integration
Settle expense share	Expense share is marked as settled in DB	Integration
Delete an expense from a trip	The expense is removed from the database	Integration

Core Feature 4 : Real-Time Chat & Polls

Test Scenario	**Expected Result**	**Type**
Send chat message with valid content	Returns 201 status with created message object	Unit
Access messages without authentication	Returns 401 Unauthorized error	Unit
Update existing message with new text	Returns 200 with updated message with modified text	Unit
Non-member attempts to send trip message	Returns 403 with appropriate error message	Unit
Request messages for valid trip	Returns 200 with messages sorted by creation date (ascending)	Unit
Create poll with valid data as trip member	Returns 201 status with created poll object	Unit
Unauthenticated user attempts poll creation	Returns 401 Unauthorized error	Unit

Non-member tries to create poll	Returns 403 with appropriate error message	Unit
Retrieve polls with TIE status	Returns formatted poll data with tie details	Unit
Batch delete polls with creator permissions	Returns success message with deleted poll count	Unit
Cast a vote to a poll as a trip member	Returns 201 with a success message	Unit
Send a chat message	The chat message is stored in the database and can be retrieved	Integration
Cast a vote on a poll	Vote is recorded in the database	Integration
Complete a Poll	Poll status is updated in database as completed with a winner id of a poll option attached	Integration

Core Feature 5 : Itinerary Builder

Test Scenario	**Expected Result**	**Type**
Create an itinerary event with valid data (within trip dates)	Returns 201 with event details	Unit
Attempt to create an event with assigned users not in the trip	Returns 400 with appropriate error message	Unit
Delete an itinerary event as the creator	Returns 200 with success message	Unit
Batch delete events where some IDs are invalid/no permission	Returns 200 with partial deletion count	Unit
Assign valid trip members to an event (admin/creator)	Returns 200 with success message	Unit
Attempt to assign non-trip members to an event	Returns 400 with appropriate error message	Unit
Unassign users from an event with valid permissions	Returns 200 with success message	Unit
Add a note to an event as a trip	Returns 201 with note details in response	Unit

member		
Attempt to update a note without being the note creator	Returns 403 with appropriate error message	Unit
Attempt to delete a note as the note creator	Returns 200 with success message	Unit
Create an itinerary event	The itinerary event is created and stored in the database and can be retrieved	Integration
Assign a member to an itinerary event	The assignment is persisted in the database and can be verified by subsequent data retrieval	Integration
Delete an itinerary event	The itinerary event is removed from the database and event notes and user assignments for this event are removed from the database	Integration

Core Feature 6 : Interactive Map & Location Sharing

Test Scenario	**Expected Result**	**Type**
Pin a new location with valid data as trip member	Returns 200 with location pin details	Unit
Pin a location with invalid data	Returns 400 with appropriate error message	Unit
Fetch pinned locations for a trip as a trip member	Returns 200 with a list of pinned locations	Unit
Remove a pinned location	Returns 200 with a success message	Unit
Pin a new location as non trip member	Returns 403 with appropriate error message	Unit
Remove a pinned location as non trip member	Returns 403 with appropriate error message	Unit
Remove a pinned location from a trip as a (trip admin)	Returns 200 with a success message	Unit
Update notes for a pinned location	Returns 200 with a success message	Unit

Update notes for a pinned location as non trip member	Returns 403 with appropriate error message	Unit
Attempt to update notes for an non existing pin	Returns 404 with appropriate error message	Unit
Pin a new location	The new location is persisted and retrievable in the database	Integration
Unpin a location	Location is removed from the database	Integration

5.3 Test Completeness

The criteria that will deem testing complete.

- Automated tests run on every pull request via CI/CD
- Minimum 80% backend test coverage
- All API endpoints tested
- All critical/high-priority bugs fixed before release
- User Acceptance Testing (UAT) conducted on staging

6 Resource & Environment Needs

6.1 Testing Tools

- Jest for JavaScript/Node.js unit testing.
- GitHub Actions for CI/CD.
- ESLint for code quality.
- Thunder Client/Postman for API testing.

6.2 Test Environment

The minimum **hardware** requirements that will be used to test the Application.

Component	Minimum Requirement	Recommended
CPU	AMD Ryzen 3 / Intel i3	AMD Ryzen 5 / Intel i5
RAM	4GB	8GB+
Storage	250GB HDD/SSD	SSD
Internet	5 Mbps	10 Mbps+

Operating Systems:

- Windows 10 / MacOS / Linux

Browsers:
- Chrome, Firefox, Edge, Safari

Major Dependencies:

Component	**Version(at least)**
Frontend	
React.js Next.js React	15.1.7 19.0.0
React-dom	19.0.0
MUI	6.4.2
Axios	1.7.9
Supabase	2.48.1
Backend	
Node.js	14+
Express	4.21.2
Prisma	6.4.1
Supabase	2.48.1
Development	
Jest	29.7.0
Supertest	7.0.0
Typescript	5.7.3

CI/CD & Deployment:
- CI/CD Pipeline runs tests on every pull request
- GitHub Actions automates testing and deployment
- Production deployment via Google Cloud Run

7 Terms/Acronyms

Make a mention of any terms or acronyms used in the project

TERM/ACRONYM	DEFINITION
API	Application Program Interface
CI	Continuous Integration
CD	Continuous Deployment
JWT	JSON Web Token
QA	Quality Assurance
UAT	User Acceptance Test
AUT	Application Under Test