Batch: D-2	**Roll No.: 16010123324 16010123325 16010123331**
Experiment No. 7	
Grade: AA / AB / BB / BC / CC / CD /DD	
Signature of the Staff In-charge with date	

Title: Designing test plan document for Mini Project

Aim: To learn and understand the way of developing the software by classical methods of software engineering.

Planning and monitoring, testing, validating of the project using tools and prepares a document for the same by using the concept of software engineering

CO: 5 Test the given software for different test cases with proper test planning

Books/ Journals/ Websites referred:

1. Roger Pressman, Software Engineering: A practitioners Approach, McGraq Hill, 2010 ,6th edition
2. Ian Somerville , Software Engineering , Addison Wesley,2011,9th edition
3 http://en.wikipedia.org/wiki/Software_requirements_specification

Test Plan Template:

Clink

Prepared by:

Shreya Menon

Shreyans Tatiya

Siddhant Raut

TABLE OF CONTENTS

1.0 INTRODUCTION:

The Clink platform is a web and mobile application designed to enable users to discover, create, and participate in social events with real-time interaction and accessibility features. The system includes user, host, and admin roles, supporting features such as personalized event feeds, in-app chat, notifications, and moderation controls. This test plan documents the strategy, scope, and procedures to ensure that Clink meets requirements in functionality, performance, security, and accessibility.

2.0 OBJECTIVES AND TASKS:

2.1 Objectives

- Develop a robust testing framework aligned with Clink's functional and non-functional requirements.
- Verify the correctness and stability of core features, including event management, social collaboration, real-time updates, and accessibility.
- Maintain high quality standards and performance levels through systematic testing cycles. Facilitate clear communication and role clarity among the development, testing, and product teams.
- Define service-level expectations to guide acceptance criteria and release readiness.

2.2 Tasks

- Conduct functional testing covering multi-role workflows and UI controls.
- Perform usability testing for navigation, interface responsiveness, and theme adaptation. Undertake security testing focusing on data protection and authorization boundaries. Execute performance and load tests simulating concurrent users and events.
- Validate compatibility across supported platforms and browsers.
- Document results, manage test artifacts, and report defects with priority and severity.

3.0 SCOPE:

General: The scope of the testing effort for Clink encompasses the thorough evaluation of all functions and features of the Clink social planning platform, including but not limited to:

Comprehensive testing of core functionalities such as event creation, feeding discovery, participant management, in-app chat, and notifications.

Evaluation of the user interfaces and user experience (UI/UX) across mobile and web platforms to ensure intuitive, responsive, and accessible designs.

Testing all existing interfaces and integrations to verify seamless interoperability and data consistency across modules and services.

Assessing device and browser compatibility to validate performance and usability across all supported environments, including iOS and Android devices, as well as major web browsers.

Tactics: The following tactics will be implemented

Testing All Core Functions: Develop and execute comprehensive test scenarios that cover all functional aspects of Clink, specifically focusing on event management, real-time communications, and accessibility features.

UI/UX Evaluation: Conduct usability assessments, including heuristic reviews and obtaining feedback from real users, to ensure the platform meets high standards of user-friendliness and design consistency.

Testing Existing Interfaces: Collaborate with developers, UI/UX designers, and product managers to identify and validate all point-of-integration interfaces, confirming smooth and error-free data and control flows.

Compatibility Assessment: Perform rigorous cross-platform testing, using physical devices and emulators/simulators, across diverse operating systems and browsers to detect and address compatibility issues.

Stakeholder Collaboration: Engage key stakeholders from development, design, and QA teams in planning and executing tests, ensuring that sufficient resources and time are allocated for a comprehensive evaluation.

User Feedback Integration: Actively gather and incorporate feedback from actual users regarding platform usability and performance, further refining test cases and improving system quality.

Documentation: Maintain clear, concise, and well-structured documentation of the testing scope, methodologies, findings, and procedural updates. Disseminate this information to all stakeholders to ensure transparency, alignment, and informed decision-making.

4.0 TESTING STRATEGY:

The strategy for testing is to break down the Clink social planning application into small units and test each unit to ensure proper functioning of each unit. The team also plans to integrate the small units and perform testing on the integrated system to ensure that there is smooth functioning of the application as a whole, with special emphasis on real-time features, accessibility compliance, and multi-platform compatibility. Testing strategies planned to be used:

- Unit Testing
- System and Integration Testing
- Performance and Stress Testing
- User Acceptance Testing
- Batch Testing
- Automated Regression Testing
- Beta Testing

4.1 Unit Testing

Definition:
Unit testing validates individual code units within Clink, ensuring they perform their intended tasks correctly. It aims for comprehensive coverage, error minimization, and thorough requirement tracing for components such as event creation modules, real-time chat handlers, notification services, accessibility features, role-based authentication, and geolocation services.

Participants:

- Shreya Menon
- Shreyans Tatiya
- Siddhant Raut

Methodology:

- Develop test scripts with input data and expected outcomes for each functional unit including mobile app components (iOS/Android) and web modules.
- Execute unit tests systematically for each unit including frontend React/React Native components, backend Node.js APIs, and WebSocket/Socket.IO handlers.
- Test accessibility components including screen reader compatibility, high contrast modes, font scaling, and keyboard navigation functions.
- Identify and document defects in role-based access controls, real-time messaging, and location-based features, assess their severity and impact.
- Resolve defects, retest units for validation and confirmation across all supported platforms.
- Analyze code coverage for thorough testing across all modules including social features and admin moderation tools.
- Maintain requirement traceability linking test cases to SRS specifications for accessibility, real-time functionality, and mobile compatibility.

Completion Criteria: High code coverage (>85%), defect resolution, and complete requirement coverage including WCAG 2.1 accessibility standards compliance.

4.2 System and Integration Testing

Definition:
System and Integration Testing for Clink involves evaluating the combined functionality of multiple units, modules, or components, ensuring that they interact correctly and that the entire system functions as a unified whole. It tests end-to-end scenarios, real-time data synchronization, and identifies any issues related to integrated components including Google OAuth, Firebase/MongoDB services, push notification systems, and geolocation APIs.

Participants:

- Shreya Menon
- Shreyans Tatiya
- Siddhant Raut

Testing Approach:

Bottom-Up Integration Testing Rationale: Clink's architecture is built upon critical real-time infrastructure components (WebSocket services, database connections, authentication modules, accessibility handlers, geolocation services) that must function reliably before integrating higher-level social features.

This approach ensures solid foundational stability by testing and validating core modules first, then progressively integrating them into larger subsystems and finally into complete user workflows.

Use of Drivers: Test drivers will be created to simulate higher-level modules and user interface calls to control and test the lower-level components during the bottom-up integration process.

Methodology:
System and Integration Testing will follow this methodology:

Test Scenario Design: Test scenarios will be developed to evaluate various end-to-end user interactions including plan discovery through location-based feeds, real-time chat participation, live notification delivery, cross-platform synchronization between mobile and web, and role transitions from user to host to admin.

Real-Time Integration Testing: Validate WebSocket/Socket.IO connections for live chat, instant plan updates, and real-time notifications across multiple concurrent users and devices.

Cross-Platform Integration: Ensure seamless data synchronization between iOS, Android, and web platforms, testing handoff scenarios where users switch between devices.

Accessibility Integration Testing: Verify that accessibility features work correctly across all integrated modules, including screen reader announcements for real-time updates and keyboard navigation through complex workflows.

Third-Party Service Integration: Test Google OAuth authentication flows, geolocation API accuracy, push notification delivery reliability, and database synchronization with Firebase/MongoDB Atlas.

Social Features Integration: Validate plan sharing mechanisms, discovery feed algorithms, community moderation workflows, and user interaction tracking.

Completion Criteria: All critical user journeys function correctly with no blocking integration defects, real-time features operate within 2-second latency requirements, and accessibility compliance is maintained across all integrated flows.

4.3 Performance and Stress Testing

Definition:
Performance and Stress Testing for Clink evaluates the application's responsiveness, stability, and scalability under different loads and stressful conditions, with particular focus on real-time chat performance, concurrent event participation, location-based query efficiency, and mobile app battery optimization.

Participants:

- Shreya Menon
- Shreyans Tatiya
- Siddhant Raut

Methodology:

Real-Time Performance Testing: Assess WebSocket connection stability and message delivery latency under varying concurrent user loads, testing chat rooms with multiple participants and simultaneous plan updates.

Mobile Performance Testing: Evaluate battery consumption, memory usage, and network efficiency on iOS and Android devices during extended usage sessions with real-time features active.

Geolocation Performance Testing: Test location-based plan discovery response times and accuracy under various network conditions and geographic distributions.

Accessibility Performance Testing: Ensure screen reader interactions and high contrast mode transitions don't impact system responsiveness.

- **Load Simulation:** Create test scripts simulating realistic user behavior patterns including plan browsing, creation, joining, and active chat participation across mobile and web platforms.
- **Stress Testing:** Subject the system to extreme conditions such as high concurrent chat message volumes, simultaneous event updates, mass notification delivery, and geographic query clustering.

- **Cross-Platform Load Testing:** Distribute load across iOS, Android, and web clients to simulate realistic usage patterns.
- **Resource Monitoring:** Monitor CPU, memory, database connections, WebSocket connections, and mobile device resources to detect potential issues.

Completion Criteria: System maintains <2 second response times for core actions, real-time features operate smoothly with 200+ concurrent users, mobile apps demonstrate optimized battery usage, and accessibility features maintain performance standards.

4.4 User Acceptance Testing

Definition:
The purpose of the acceptance test is to confirm that the Clink system meets the needs of all user roles and is ready for operational use. During UAT, end-users representing different stakeholder groups (event participants, hosts, and administrators) compare the system against initial requirements with special attention to accessibility, social interaction quality, and cross-platform consistency.

Participants:

- Shreya Menon
- Shreyans Tatiya
- Siddhant Raut

Methodology:

Role-Based Testing: Separate UAT scenarios for participants (plan discovery, joining, chatting), hosts (event creation, management, moderation), and admins (platform oversight, user management, analytics).

Accessibility UAT: Include users with disabilities to test screen reader compatibility, keyboard navigation, high contrast modes, and font scaling across all user journeys.

Cross-Platform UAT: Test user workflows across iOS, Android, and web platforms to ensure consistent experience and seamless data synchronization.

Social Interaction UAT: Focus on real-time chat usability, plan sharing effectiveness, community guidelines enforcement, and notification relevance.

Real-World Scenario Testing: Simulate authentic social planning scenarios including spontaneous meetups, group coordination, and event modifications with real-time updates.

Geographic UAT: Test location-based features in various geographic contexts to ensure discovery algorithms work effectively.

Completion Criteria: All user roles confirm the system meets their needs, accessibility features receive positive feedback from users with disabilities, cross-platform experience is seamless, and real-time social features enhance rather than complicate event coordination.

4.7 Beta Testing

Participants:

- Shreya Menon
- Shreyans Tatiya
- Siddhant Raut

Methodology:
We plan on collecting feedback from diverse user groups including fellow classmates, community members with varying accessibility needs, and different age demographics through Google forms, in-app feedback tools, and structured interviews. The feedback collection will focus on Clink-specific aspects including:

- **Social Planning Effectiveness:** How well the platform facilitates spontaneous event coordination and community building
- **Real-Time Feature Usability:** User satisfaction with chat functionality, live updates, and notification relevance
- **Accessibility Experience:** Comprehensive feedback from users with disabilities on screen reader support, navigation ease, and visual accessibility features
- **Cross-Platform Consistency:** User experience across iOS, Android, and web platforms
- **Geographic and Cultural Adaptation:** Effectiveness of location-based discovery and cultural sensitivity of social features

We will also engage our lab teachers, accessibility consultants, and social media platform experts for professional insights on platform design, community management features, and technical architecture. Any constructive suggestions will be prioritized for implementation, with particular attention to accessibility improvements and real-time feature optimization.

5.0 HARDWARE REQUIREMENTS:

Hardware	Configuration	No. of units
Mobile Devices	Android 8+ or iOS 13+ smartphones with GPS, camera, and internet connectivity	Multiple for testing
Desktop/Laptop	Modern computer with 8GB+ RAM, multi-core processor, for web browser testing and development	2-3 units
Development Workstations	Intel Core i5/i7 or equivalent, 16GB RAM, 256GB+ SSD for development environment	2-3 units
Cloud Server Infrastructure	AWS/GCP/Azure cloud instances for backend APIs, database, and real-time services	As needed
Testing Devices	Various iOS and Android devices for cross-platform compatibility testing	5-10 devices

6.0 ENVIRONMENT REQUIREMENTS:

Modern web browsers (Chrome, Firefox, Safari, Edge) with active internet connection, mobile environments (iOS/Android), and cloud infrastructure for real-time services.

6.1 Main Frame

Test Environment Specifications:

- Cloud server instances (AWS/GCP/Azure) for backend APIs, real-time services, and databases
- Mobile testing environments supporting iOS 13+ and Android 8+
- SSL/TLS security for encrypted communication
- OAuth 2.0 authentication integration
- WebSocket/Socket.IO for real-time features

Special Test Tools:

- Cypress/Selenium for web automation
- Postman for API testing
- Lighthouse/Axe for accessibility auditing
- Screen reader software for accessibility testing
- Load testing tools (JMeter/Artillery)

Testing Needs:

- High-speed internet connection (100+ Mbps)
- Physical mobile devices for cross-platform testing
- Accessibility testing environment with assistive technology

Currently Unavailable Resources:

- Cloud infrastructure costs
- Premium testing tool licenses
- Comprehensive mobile device inventory

6.2 Workstation

Workstation Requirements:

- Modern laptops/desktops with multi-core processors
- 16GB+ RAM, 256GB+ SSD storage
- Multiple monitors for simultaneous testing

Software Environment:

- Node.js, React/React Native development frameworks
- Android Studio and Xcode for mobile testing
- Git version control, VS Code/WebStorm IDEs
- Docker for containerized testing environments

Network Configuration:

- Stable internet with low latency
- VPN capabilities for geographic testing
- Local network setup for multi-device scenarios

7.0 TEST SCHEDULE:

Duration	Milestone	Output
1st week	Unit Testing	Module validation documentation, individual component test reports, accessibility feature verification reports, and defect tracking logs.
2nd week	System Integration Testing	End-to-end workflow validation reports, real-time feature integration documentation, cross-platform synchronization test results, API integration reports, and developer feedback on system stability.
3rd week	Performance Testing and User Acceptance Testing	Load testing performance metrics, real-time communication latency reports, mobile app performance analysis, accessibility compliance audit results, and comprehensive user feedback from role-based testing scenarios.
4th week	Beta Testing and Automated Regression Testing	Real-world user feedback from diverse demographics, accessibility feedback from users with disabilities, cross-platform user experience reports, automated regression test suite execution results, and final deployment readiness assessment.

8.0 CONTROL PROCEDURES:

Problem Reporting

Document the procedures to follow when an incident is encountered during the testing process. If a standard form is going to be used, attach a blank copy as an "Appendix" to the Test Plan. In the event you are using an automated incident logging system, write those procedures in this section.

- If any problem or error has occurred during the testing of Clink, the tester can directly report the error to the developer through the project management system (Jira/Trello), also the developer makes note of errors to be corrected with priority classification (Critical, High, Medium, Low)
- All incidents must include: error description, steps to reproduce, platform/device information, expected vs actual behavior, and screenshots/screen recordings where applicable
- Real-time feature bugs and accessibility issues will be tagged with highest priority for immediate attention
- Cross-platform compatibility issues must specify the affected platforms (iOS, Android, Web)

Change Requests

Document the process of modifications to the software. Identify who will sign off on the changes and what would be the criteria for including the changes to the current product. If the changes will affect existing programs, these modules need to be identified.

If any error is found by the tester, and if it can be resolved and 70% of developer teams agree to make the changes, then the changes are to be adapted in necessary modules. For Clink specifically:

- Changes affecting real-time communication, accessibility features, or core user flows require unanimous team approval
- Cross-platform changes must be tested on all supported platforms before implementation
- Any modifications to authentication, data privacy, or security features require additional security review

9.0 FEATURES TO BE TESTED:

Feature	**Test Description**
User Authentication	Verify users can log into the system using Google OAuth integration
User Registration	Test if new users can register and create profiles with accessibility preferences
Plan Creation & Management	Validate event creation, editing, scheduling, and cancellation workflows
Plan Discovery Feed	Check if personalized event feeds are generated based on user interests and location
Real-time Chat	Test in-app messaging functionality with live message delivery and synchronization
Join/Leave Plans	Verify users can join and leave events with proper notifications to hosts
Accessibility Features	Test screen reader compatibility, high contrast mode, font scaling, and keyboard navigation
Host Dashboard	Test host controls for participant management, plan modification, and moderation
Admin Moderation	Verify admin capabilities for content moderation, user management, and platform oversight
Theme Support	Check if light and dark mode themes work properly with accessibility features

10.0 FEATURES NOT TO BE TESTED:

Identify all features and significant combinations of features which will not be tested and the reasons.

- **Calendar Integration Features:** External calendar sync (Google Calendar, Apple Calendar) - not implemented in current version
- **Advanced Analytics Dashboard:** Detailed user behavior analytics for hosts - planned for future releases
- **Video Chat Integration:** In-app video calling features - outside current scope
- **Payment Processing:** Paid event features and ticketing system - not included in MVP
- **Multi-language Localization:** Non-English language support - planned for later versions

11.0 RESOURCES/ROLES & RESPONSIBILITIES:

Shreya Menon: Test planning coordination, accessibility testing lead, mobile app testing execution
Shreyans Tatiya: Backend API testing, real-time features validation, cross-platform compatibility testing
Siddhant Raut: User acceptance testing coordination, performance testing management, final documentation

12.0 SCHEDULES:

Major Deliverables

Identify the deliverable documents. You can list the following documents:

- Test Plan
- Test Cases
- Test Incident Reports
- Test Summary Reports
- Accessibility Compliance Reports
- Cross-Platform Testing Reports
- Real-time Feature Performance Reports

13.0 DEPENDENCIES:

Identify significant constraints on testing, such as test-item availability, testing-resource availability, and deadlines.

Since Clink involves complex real-time social features, cross-platform compatibility (iOS, Android, Web), and accessibility compliance, key dependencies include:

- Availability of diverse mobile devices for comprehensive cross-platform testing
- Stable internet connectivity for real-time chat and notification testing
- Access to accessibility testing tools and assistive technologies (screen readers)
- Cloud infrastructure availability for load testing real-time features
- Coordination with third-party services (Google OAuth, Firebase/MongoDB, push notifications)
- Time constraints for thorough testing across multiple platforms and user roles

14.0 RISKS/ASSUMPTIONS:

Risks:

- Real-time features might experience latency issues during peak concurrent user testing
- Cross-platform synchronization could face delays due to network instability affecting chat and plan updates
- Accessibility testing may be limited by availability of specialized assistive technology equipment
- Third-party service dependencies (OAuth, push notifications) could cause integration testing delays

Assumptions:

- All team members have access to required mobile devices and development environments
- Test users representing different accessibility needs will be available for UAT
- Internet connectivity remains stable throughout real-time feature testing
- Google OAuth and Firebase services remain operational during testing periods

15.0 TOOLS:

Testing and Development Tools:

- Cypress/Selenium for automated web application testing
- Jest for React/React Native unit testing
- Postman/Newman for REST API testing and automation
- Lighthouse and Axe for accessibility compliance auditing
- Android Studio and Xcode for mobile app testing and debugging
- WebSocket testing tools for real-time communication validation
- Screen reader software (NVDA, JAWS, VoiceOver) for accessibility testing
- Git for version control and collaborative development
- Jira/Trello for issue tracking and project management

16.0 APPROVALS:

Name (In Capital Letters)	Title	Signature	Date
	Project Team Lead		
	Faculty Supervisor		
	QA/Accessibility Specialist		
	Technical Mentor		

Conclusion:

The Clink Test Plan has been developed to ensure comprehensive validation of all functional and non-functional requirements of the social planning platform. This document establishes a structured approach to testing that encompasses unit testing, integration testing, performance validation, and user acceptance testing, with special emphasis on accessibility compliance and cross-platform compatibility.

The testing strategy addresses Clink's unique requirements as a real-time social application, including WebSocket communications, mobile-web synchronization, role-based access controls, and WCAG accessibility standards. Through systematic execution of the outlined testing phases, the development team can ensure that Clink delivers a reliable, secure, and inclusive user experience across all supported platforms. The plan serves as a comprehensive guide for quality assurance activities, enabling effective coordination between testing, development, and stakeholder teams while maintaining transparency and accountability throughout the testing lifecycle.

Post Lab Descriptive Questions:

1. **Distinguish between Black Box and White Box Testing**

Aspect	Black Box Testing	White Box Testing
Definition	Testing method that examines software functionality without knowledge of internal code structure	Testing method that examines internal code structure, logic, and implementation details
Knowledge Required	No programming or internal system knowledge required	Requires extensive programming knowledge and understanding of code structure
Testing Focus	Focuses on validating external behavior and functionality against requirements	Focuses on internal logic, code paths, conditions, and structural elements
Performed By	Usually performed by QA testers or end users	Typically performed by developers or technical testers
Test Design Basis	Based on functional specifications and user requirements	Based on code structure, algorithms, and implementation details
Types of Bugs Found	Detects missing functionalities, interface issues, and incorrect outputs	Identifies logical errors, unreachable code, and structural defects
Testing Levels	Commonly used for system testing, acceptance testing, and integration testing	Primarily used for unit testing and component-level testing
Time Consumption	Generally faster to design but may take longer to execute	More time-intensive to design but provides faster debugging
Algorithm Testing	Not suitable for algorithm validation	Excellent for algorithm testing and optimization
Example for Clink	Testing if users can successfully create and join events without knowing the backend implementation	Testing the chat message delivery algorithm and database query optimization

2. **Explain traceability matrix.**

A Requirements Traceability Matrix (RTM) is a document that establishes relationships between requirements and test cases to ensure comprehensive test coverage and project transparency.

Purpose: Ensures all requirements are tested and validated Tracks the progress of testing activities Identifies missing test cases or requirements Facilitates impact analysis when requirements change Provides visibility to stakeholders on project coverage

Types of Traceability Matrix: Forward Traceability: Maps requirements to corresponding test cases, ensuring each requirement has adequate test coverage Backward Traceability: Maps test cases back to their originating requirements, ensuring no unnecessary tests exist Bi-directional Traceability: Combines both forward and backward traceability for complete coverage validation

Key Parameters in RTM:

- Requirement ID and Description
- Requirement Type and Priority
- Test Case IDs and Status
- Test Execution Results
- Defects and Resolution Status
- User Acceptance Test Status

Requirement ID	Requirement Description	Test Case ID	Test Status	Defects
R001	User Authentication via Google OAuth	TC001, TC002	Passed	None
R002	Real-time Chat Functionality	TC003, TC004, TC005	In Progress	DEF001
R003	Accessibility Features	TC006, TC007	Passed	None