Batch: D-2	**Roll No.: 16010123324** **16010123325** **16010123331**
Experiment / assignment / tutorial No._1___	

TITLE: Requirement Specification Document

AIM: To learn and understand the way of analysing the gathered information in the previous phase for the development process and prepare requirement specification document. A concept of software engineering.

Expected Course outcome of Experiment:

Process of gathering requirements and converting them into specifications.
Document created will be used by both, the customer and the developer, to understand WHAT is going to be developed.

Books/ Journals/ Websites referred:

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

---

Pre Lab/ Prior Concepts:

Requirements analysis in systems engineering and software engineering, encompasses those tasks that go into determining the needs or conditions to meet for a new or altered product, taking account of the possibly conflicting requirements of the various stakeholders, such as beneficiaries or users. It is an early stage in the more general activity of requirements engineering which encompasses all activities concerned with eliciting, analyzing, documenting, validating and managing software or system requirements.

Requirements analysis is critical to the success of a systems or software project. The requirements should be documented, actionable, measurable, testable, traceable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design.

Conceptually, requirements analysis includes three types of activities:

- **Eliciting requirements**: the task of identifying the various types of requirements from various sources including project documentation, (e.g. the project charter or definition), business process documentation, and stakeholder interviews. This is sometimes also called requirements gathering.
- **Analysing requirements**: determining whether the stated requirements are clear, complete, consistent and unambiguous, and resolving any apparent conflicts.
- Recording requirements: Requirements may be documented in various forms, usually including a summary list and may include natural-language documents, use cases or process specifications.

New systems change the environment and relationships between people, so it is important to identify all the stakeholders, taken into account all their needs and ensure they understand the implications of the new systems. Analysts can employ several techniques to elicit the requirements from the customer. These may include the development of scenarios, the identification of use cases, the use of workplace observation or ethnography, holding interviews, or focus groups (more aptly named in this context as requirements workshops, or requirements review sessions) and creating requirements lists. Prototyping may be used to develop an example system that can be demonstrated to stakeholders. Where necessary, the analyst will employ a combination of these methods to establish the exact requirements of the stakeholders, so that a system that meets the business needs is produced

Different types of Requirements

- Functional requirements
- Usability requirements
- Reliability requirements
- Performance requirements
- Security requirements

A typical SRS document template is shared subsequently. This document acts as a reference and will be used by both, the customer (for whom the software system is to be developed), and the organization which develops the solution. Typically, prepared by the development organization at the early stage of development by the professionals after interacting with the customer.

Software Requirements Specification for:

Clink

Version 1.0

Prepared by Prepared by RTM & et al

KJSSE

24/07/2025

Table of Contents

Introduction

Purpose

The proposed system, Clink, offers a digital-first platform designed to simplify and amplify spontaneous, interest-based social planning moving beyond rigid calendar-based systems and fragmented messaging platforms. The system comprises two primary modules: the user-facing module as a mobile application and the host-facing module as an event creation and management interface.

Unlike traditional event platforms built for formal gatherings, Clink empowers everyday users not just organizers to create and share plans. A “host” on Clink might be someone going to a nearby gig, grabbing coffee, or attending a college event who’s simply looking for company. Users can discover such plans based on location, time, and shared interests, and express intent to join without formal RSVPs.

Clink includes its own built-in group chat system to support lightweight coordination among attendees, eliminating the need for third-party tools. The backend supports real-time updates, interest-based feeds, and efficient data storage to ensure a smooth, responsive experience. By decentralizing who can host and focusing on immediacy and casual coordination, Clink fosters real-world social spontaneity making it easier for people to meet without overplanning or flaking.

Product Scope

Clink is a lightweight social platform that helps users create, discover, and join spontaneous, interest based plans nearby. Anyone can host whether they’re planning a quick hangout or looking for company at an event. Instead of relying on formal RSVPs or third-party messaging apps, Clink offers flexible privacy options and its own in built group chat to enable seamless coordination. The app curates personalized plan feeds based on location, time, and user interests, making real world socializing easier, faster, and more spontaneous.

References

https://partiful.com/

https://lu.ma/

Overall Description

Product Perspective

Clink reimagines social planning by enabling users to create and join spontaneous, interest-based plans with ease. Instead of relying on rigid calendars or scattered group chats, users can instantly share clinks lightweight event links with friends and coordinate in real time through built-in chat.

Designed for everyday use, Clink encourages low-pressure meetups and organic interactions, making it easy to connect with friends over shared interests. Its smart discovery feed, and instant updates streamline coordination and foster more meaningful in person engagement.

By embracing spontaneity, Clink offers a refreshing and efficient way to socialize simplifying planning, boosting participation, and strengthening real world connections.

Product Functions

1. Clink Creation
   Users can create a spontaneous plan (“clink”) by entering basic details: activity name, time, location (manual or map-based), and interest tag(s). It takes under 30 seconds to create.

2. Clink Sharing
   Each clink generates a unique shareable link that can be sent via WhatsApp, Telegram, DMs, or in-app to friends. Clicking the link brings users to the clink page instantly.

3. In-App Chat for Every Clink
   Each clink has its own built-in group chat. Users who join the clink are automatically added to its chat room no need to switch apps to coordinate. Members can chat, share updates, or send media.

4. Real-Time Join & Participation
   Users can view who has joined a clink in real-time. As people join, the participant list updates dynamically and is visible to all clink members.

5. Interest-Based Clink Discovery
   The home screen shows clinks based on your selected interests (e.g., gigs, food, study, gaming). Users can filter clinks by tags, distance, time, or people they follow.

6. Clink Feed & Explore
   Users can explore public clinks nearby or within their social graph. Private clinks are only visible via direct invite/share link. Explore is location-aware and shows things happening "soon and nearby".

7. Host Controls & Live Updates
   Clink creators (hosts) can update details like time or location even after creation. All joined members receive instant in-app notifications and chat pings about changes.

8. Notification System
   In-app and push notifications keep users informed about:
   invites received, someone joining their clink, chat messages in joined clinks and plan reminders 30 mins before start.

9. Persistent Clink History
   Users have a personal dashboard that shows their past hosted, joined, or interested clinks. Useful for tracking events, rehosting similar clinks, or reconnecting with frequent friends.

10. Lightweight, Real-Time Architecture
    Clink is designed for fast, real-time usage, Firebase/WebSocket for instant updates and UI optimized for mobile, tap-first interactions.

Operating Environment

SOFTWARE REQUIREMENTS

Clink's system components are developed using modern web and mobile technologies to ensure real-time functionality, scalability, and cross-platform compatibility.

- Node.js
- Express.js
- React / React Native
- MongoDB Atlas or Firebase
- Socket.IO or Firebase Realtime Database
- Tailwind CSS

- JWT / Firebase Auth
- Geolocation & Push Notification APIs

HARDWARE REQUIREMENTS

- Any modern Android or iOS smartphone (with GPS and internet access)
- Cloud server or VPS (e.g., AWS EC2, Render, Vercel)
- Stable internet connection
- Administrator/host device: any laptop or desktop with modern web browser
- Cloud database and CDN services
- Optional edge devices or repeaters if hosting in areas with weak connectivity

Design and Implementation Constraints

Usability Requirement

The system shall provide access via a mobile-first application designed with an intuitive interface. Users will not require any prior training. Core interactions—such as swiping through plans, viewing details, and expressing interest—are modeled on familiar social interfaces to ensure quick onboarding and high usability.

Availability Requirement

Clink will be operational 24/7, 365 days a year. Its infrastructure will be hosted on scalable cloud services (e.g., AWS, Vercel, Firebase) to ensure consistent uptime and global availability.

Efficiency Requirement

In the event of a failure, Mean Time to Repair (MTTR) shall not exceed 5 business days. Automated database backups and a stateless architecture ensure fast rollback and recovery with minimal disruption to users.

Accuracy

The system shall ensure accurate interest matching and plan feed delivery based on real-time geolocation and user interest tags. Feed updates and interactions will reflect with at least 90% consistency across devices within 5 seconds.

Performance Requirement

The system shall load the personalized feed within 2 seconds under normal conditions. Interactions such as plan joining or event creation will receive acknowledgment in less than 1 second. Chat-related features (if enabled) and image-heavy views will render within 5 seconds.

Reliability Requirement

Clink shall maintain a 99.9% uptime SLA for core services. It shall guarantee data persistence through replication and integrity checks to safeguard user data (e.g., profile, plans joined, chat history). System shall be resilient to connection losses and recover gracefully on reconnect.

User Documentation

Step 1: Download & Install

- Go to the App Store (iOS) or Google Play (Android).
- Search for “Clink”
- Download and install the app.

Step 2: Sign Up & Onboarding

- Open the app and tap "Get Started"
- Sign up with email or Google/Apple ID
- Answer onboarding questions (interests, preferred hangout type, age range, location radius)

Step 3: Browsing the Feed

- On the home page, swipe left/right to explore nearby plans
- Tap a plan card to view full details and host information

Step 4: Joining a Plan

- Tap “Join” on a plan you’re interested in

- You'll be added to the attendee list
- Optional: add yourself to coordination chat if enabled by host

Step 5: Creating a Plan

- Tap "+" button to start a new plan
- Enter details like time, place, visibility (public/private), and description
- Publish and share

Step 6: Notifications

- Enable notifications to get plan updates, invites, and coordination messages
- Manage preferences from the settings tab

Step 7: Support

- Go to Profile → Settings → Help & Support
- Access FAQs, contact options, or report bugs

Assumptions and Dependencies

Assumptions:

1. Mobile Device Compatibility: Assumes users have Android 8.0+ or iOS 13+ with functioning GPS and internet.
2. Internet Access: Assumes a stable connection for retrieving feeds, joining plans, and receiving updates.
3. Host Availability: Assumes hosts provide correct plan information (time, place) and manage attendees responsibly.
4. User Familiarity: Assumes users are familiar with basic swipe/feed interfaces and location permissions.
5. Push Notifications Enabled: Assumes users allow notifications for coordination and reminders.

Dependencies:

1. Firebase or similar backend platform for database, authentication, and notifications.
2. Google Maps or Mapbox API for geolocation and event discovery radius.
3. Secure hosting platform (e.g., Vercel, Render, or AWS) for consistent backend performance.
4. Mobile OS updates (iOS, Android) to ensure long-term app compatibility.
5. Push Notification Services (APNs/FCM) for user alerts and plan activity.

External Interface Requirements

User Interfaces

1. **Hosts:**
   The host interface empowers users to create and manage spontaneous plans (called "clinks") with minimal effort. Through a streamlined creation screen, hosts can enter a title, location (text or map-based), date and time, visibility (public or private), and select relevant interest tags. They can also choose whether attendees can join freely or require approval.

   Once created, hosts access a dashboard showing all their upcoming, past, and draft clinks, each with quick actions to edit, reschedule, cancel, or duplicate events. Inside each clink, hosts can view a real-time list of participants, approve pending join requests (if enabled), and post updates directly into the integrated clink chat. System messages automatically notify participants of any changes (like time or location edits). The interface is designed for flexibility and control, supporting both one-off hangouts and recurring plans with ease.

2. **Attendees / Explorers:**
   Attendees can explore a personalized feed of active clinks filtered by interest, timing, and proximity. Each clink page shows full plan details, attendee list, and live chat access. Users can express interest, join instantly, or follow updates for public clinks. Private clinks are accessible via shared links only. The interface is built for low-friction action encouraging quick decision-making and organic meetups.

3. **Admins (System Moderators):**
   The admin interface is built to help moderators oversee community activity, enforce safety policies, and maintain a respectful environment across the platform. It offers tools for managing user behavior, responding to content reports, and analyzing engagement trends to support Clink's growth and health.

   Core admin functions include:
   - Viewing all public clinks in real-time, including details like title, host, location, and participant count
   - Reviewing reported clinks and investigating flagged content or behavior
   - Monitoring clink chats (only when flagged) to assess abuse, spam, or harassment complaints
   - Temporarily restricting users from creating or joining new clinks

- Suspending or deactivating user accounts for serious or repeated violations

- Accessing platform-wide analytics:
  - Number of clinks created
  - User participation rates
  - Most active times and locations

- Ensuring adherence to community guidelines and intervening when necessary to protect user safety

Hardware Interfaces

Logical Characteristics of Interfaces:

1. **User Interface (UI):**
The software product will provide an intuitive, mobile-first UI accessible via Android and iOS applications. A lightweight web interface may also be used by hosts or admins for plan management. The UI will support gesture-based navigation, interest selection, plan creation, and in-app chat.

2. **Database Interface:**
Clink will interface with a NoSQL or cloud-native database (e.g., Firebase Firestore or MongoDB Atlas) to manage user profiles, plans, real-time chat data, and interest metadata. The interface will ensure secure and efficient data storage, querying, and synchronization.

3. **Communication Interface:**
The system will use secure communication protocols (HTTPS) to manage data exchange between the client app, backend servers, and the database. This ensures real-time plan updates, location-based feed generation, and chat messaging.

Physical Characteristics of Interfaces:

1. **Mobile Device Interface:**
Clink will run on Android and iOS smartphones. It will access device features such as GPS (for location-based feeds), push notifications (for plan updates and chat), and camera (optional for future photo-sharing features).

2. **Web Browser Interface (optional):**
A responsive web dashboard may be accessible via standard web browsers, primarily for administrative use or extended event/host management. It will follow responsive design practices to ensure usability across desktop and tablet screens.

3. **GPS & Location Services Interface:**
The app will interact with the mobile device's location services to enable geo-filtered feed recommendations and distance-based plan visibility. Location access will be permission-based and handled per privacy best practices.

4. **Notification Interface:**
The mobile app will interface with native OS-level push notification services (Firebase Cloud Messaging for Android, Apple Push Notification service for iOS) to deliver updates related to joined plans, chat mentions, or host changes.

5. **Cloud Services Interface:**
The backend infrastructure will leverage cloud platforms (such as Firebase, AWS, or GCP) to handle authentication, real-time data (plans, chat), analytics, and file storage (if images/media are used in plans). These services will be accessed via RESTful APIs and WebSocket where needed.

Software Interfaces

The Clink platform interacts with multiple software components to ensure smooth functioning of interest-based social planning, user coordination, and real-time updates.

1. Database System

Clink uses a scalable cloud-native database such as Firebase Firestore or MongoDB Atlas to store user profiles, plans, interests, chat messages, and host metadata. Communication occurs via SDKs or secure REST/GraphQL APIs for efficient querying and data synchronization.

2. Operating System

The Clink mobile application is compatible with Android (version 8.0 and above) and iOS (version 13 and above). A responsive web interface is accessible through modern browsers such as Chrome, Safari, Firefox, and Edge for hosts and admin users.

3. APIs

Clink uses RESTful and WebSocket APIs for functionalities including:

- Authentication (OAuth2/Firebase Auth)
- Real-time feed updates
- Location services
- Push notifications
- Plan creation and discovery

- In-app group chat synchronization

All API interactions occur over HTTPS and conform to secure API design standards.

4. Client-Side Libraries

- Clink mobile app is built using React Native, leveraging libraries such as React Navigation, Redux Toolkit, and Firebase SDK for data binding, routing, and state management.
- The in-app chat feature may use socket.io or Firebase Realtime Database for real-time messaging.
- Image upload and media handling use third-party SDKs like Cloudinary or Firebase Storage.

5. Web Development Tools

The web interface is developed using React.js, TailwindCSS, and modern frontend tooling (e.g., Vite, Webpack). It supports asynchronous data fetching through Axios or GraphQL clients.

6. Network Communication

The mobile and web applications communicate with backend services over secure HTTPS protocols. WebSockets are used for real-time communication (e.g., plan updates, live chat). Location and event feed queries are optimized for low latency.

Data Flow

- Plan Creation: A user creates a plan through the mobile/web UI, which is stored in the backend database.
- Feed Generation: User opens the app → the system queries backend APIs → personalized plan feed is delivered based on location, time, and interests.
- Joining Plans: When a user marks interest, it updates the database in real time and adds them to the plan's participant list (and optionally, the chat).
- In-App Messaging: If chat is enabled, real-time messages are exchanged using WebSocket or Firebase channels.
- Notifications: Users receive push notifications for new plan invites, chat activity, or admin updates via FCM/APNs.

Shared Data

- User Profiles: Shared across mobile and web platforms for consistent personalization and discovery.

- Plan Metadata: Title, tags, location, privacy, time, etc., visible to participants and hosts.
- Chat Messages: Stored and synced across devices if chat is enabled.

Constraints

- Data Security: All communication is encrypted (HTTPS/TLS); authentication tokens are used for secure access control.
- Data Consistency: Real-time syncing ensures feed, chat, and RSVP data remain consistent across user devices.
- Privacy: Location and profile visibility are permission-controlled and adhere to app privacy settings.
- Offline Behavior: The app supports local caching for offline browsing with eventual synchronization.

Communications Interfaces

1. **Plan Feed Delivery:**
   Real-time communication between backend servers and the app to serve personalized feeds based on user interests, location, and time.

2. **Notifications:**
   Push notifications (via Firebase Cloud Messaging and Apple Push Notification Service) for plan activity (e.g., someone joins your plan, reminders, or chat mentions if enabled).

3. **Sync:**
   Seamless two-way data sync between mobile client and cloud database (e.g., Firestore or MongoDB Atlas) for profile updates, plan status, and chat messages.

4. **Security:**
   All data transfers use HTTPS/TLS protocols. Sensitive operations like plan edits or chat messages are authenticated with secure tokens (JWT/Firebase Auth).

5. **Network Compatibility:**
   The app must function on 4G/5G and Wi-Fi networks, with fallback caching to allow limited offline plan browsing or drafting.

6. **Authentication & Authorization:**
   OAuth2/Firebase-based secure login system. Only authorized users can create or join plans, edit their profile, or view chat channels they're a part of.

7. **Low Latency UX:**
   Quick plan loading, fast feed swiping, and near-instant updates on join activity to maintain a spontaneous, responsive user experience.

8. **Error Handling:**
   Graceful handling of connectivity drops or backend errors. Local caching and retry logic for actions like joining a plan or sending a chat.

9. **Message Format:**
   All backend communication follows JSON or REST/GraphQL schema. Frontend messages (e.g., error toasts, success banners) are concise and actionable.

10. **User-Friendly Interactions:**
    Communications should feel natural from "you've got 3 mutuals joining" to "Plan begins in 1 hour." Tone and delivery matter for real-world spontaneity.

System Features

Feature 1: Plan Creation (Host Module)

4.1.1 Description and Priority:
Enables users to create and publish spontaneous social plans. High priority for enabling core user-generated content.

4.1.2 Stimulus/Response Sequences:

- User selects "Create Plan" option.
- User fills in event title, time, location, visibility, interest tags.
- User optionally enables group chat and uploads a cover image.
- System validates the input and creates the plan.
- Plan becomes visible in feeds of relevant users.

4.1.3 Functional Requirements:

- REQ-1: The system shall allow any authenticated user to create a plan.
- REQ-2: The system shall provide fields for title, description, date, time, and location.
- REQ-3: The system shall support plan visibility options (public, friends, invite-only).
- REQ-4: The system shall tag plans with relevant interests for feed curation.
- REQ-5: The system shall allow image upload for plan thumbnails.
- REQ-6: The system shall auto-assign a unique ID to each plan.

- REQ-7: The system shall enable or disable in-built group chat based on host's preference.

Feature 2 : Feed & Plan Discovery

4.2.1 Description and Priority:
Displays personalized feed of plans based on time, location, and interests. High priority for engagement and discovery.

4.2.2 Stimulus/Response Sequences:

- User opens home/feed screen.
- System fetches and ranks nearby plans based on user's interests and preferences.
- User can swipe to skip or tap to view details.
- If interested, user taps "I'm in" to express interest or join the plan.

4.2.3 Functional Requirements:

- REQ-8: The system shall retrieve plans based on user's interests, current location, and time relevance.
- REQ-9: The system shall present plans in a scrollable/swipeable card format.
- REQ-10: The system shall allow users to view plan details and participant list.
- REQ-11: The system shall allow users to mark interest ("Join" or "I'm in").
- REQ-12: The system shall not require a formal RSVP to join.

Feature 3 : Built-In Group Chat

4.3.1 Description and Priority:
Enables real-time messaging for plan attendees. Medium priority for coordination.

4.3.2 Stimulus/Response Sequences:

- Host enables chat while creating or editing a plan.
- Upon user joining the plan, they are added to the chat group.
- System allows threaded replies and optional sub-channels (e.g., carpooling).
- Users may leave chat or report inappropriate behavior.

4.3.3 Functional Requirements:

- REQ-13: The system shall allow hosts to enable/disable chat per plan.

- REQ-14: The system shall create a unique group chat for each enabled plan.
- REQ-15: The system shall support real-time messaging with read receipts.
- REQ-16: The system shall allow reporting and blocking of users within chat.
- REQ-17: The system shall support optional sub-channels for features like carpool or meet-up points.

Feature 4: Profile & Interest Matching

4.4.1 Description and Priority:
Stores user interests and curates plan suggestions accordingly. High priority for personalization.

4.4.2 Stimulus/Response Sequences:

- Onboarding captures user interests and location.
- User can edit profile to update hobbies, interests, or bio.
- System uses this data to generate personalized feeds and suggest people/events.

4.4.3 Functional Requirements:

- REQ-18: The system shall allow users to select and update their interests.
- REQ-19: The system shall use this data to rank and recommend plans in the feed.
- REQ-20: The system shall allow users to view others' public profiles.

Feature 5: Admin Dashboard

4.5.1 Description and Priority:
Provides platform moderators with tools for safety, content moderation, and user management. High priority for community integrity.

4.5.2 Stimulus/Response Sequences:

- Admin logs into the dashboard.
- Admin views reported plans or users.
- Admin approves, warns, or blocks content/users.
- Admin views analytics for platform activity.

4.5.3 Functional Requirements:

- REQ-21: The system shall allow admin login with elevated access.

- REQ-22: The system shall display flagged content and allow moderation actions.
- REQ-23: The system shall allow blocking or muting abusive users.
- REQ-24: The system shall provide analytics on plan creation, user activity, and engagement trends.

Other Nonfunctional Requirements

Performance Requirements

- **Response Time:** ≤ 2 seconds for major actions (joining/creating plans).
- **Feed Load Time:** ≤ 5 seconds for personalized plan feed.
- **Concurrency:** ≥ 200 users active simultaneously.
- **Scalability:** Designed for 10,000+ users as user base grows.
- **Push Notifications:** Delivered within 3 seconds of event creation/update.
- **Uptime:** ≥ 99.9% server availability.
- **Disaster Recovery:** Full restoration within 24 hours.
- **Load Balancing:** Engaged at 70% server load to maintain performance.

Safety Requirements

SR-1: Data Security and Privacy

- Encrypted personal data.
- Users control visibility (public, limited, invite-only plans).

SR-2: Reliable Operation

- Backup servers ensure uptime.
- Regular maintenance during non-peak hours.

SR-3: Compliance

- GDPR-compliant consent and data deletion flows.

Security Requirements

- **Authentication:** Email/OTP login; optional 2FA.
- **Data Encryption:** End-to-end encrypted data flow (HTTPS); hashed user credentials.
- **Privacy Controls:** Host sets visibility; no open DMs without consent.
- **Audit Logs:** System logs admin access and suspicious activity.
- **Compliance:** GDPR, CCPA compliance; full privacy policy transparency.

Software Quality Attributes

- **Usability**:
Achieve at least 85% user satisfaction based on regular user surveys and feedback mechanisms.
- **Reliability**:
Maintain 99.9% system uptime to support uninterrupted real-time interactions.
- **Adaptability**:
Ensure the system is API-ready for seamless integration with other social platforms and tools.
- **Maintainability**:
Follow a modular code structure with more than 90% unit test coverage to simplify future updates and maintenance.
- **Interoperability**:
Deliver a mobile-first responsive design compatible with tablets and desktop browsers.
- **Testability**:
Attain 90% automated test coverage to ensure consistent functionality and easy debugging.

- **Portability**:
Enable deployment on any cloud infrastructure with support for both Android and iOS environments.

- **Flexibility**:
Offer configurable options for feed filters and notification settings to suit user preferences.

- **Correctness**:
Keep critical user operation error rates (e.g., plan creation and joining) below 1%.

Business Rules

1. **Host Role:** Any user can create a plan, set time, location, interest tags, and visibility level.
2. **Participant Role:** Users can browse feeds, request to join, or join public plans directly.
3. **Feed Curation:** Personalized by interests, location, time, and social signals.
4. **Privacy:** Hosts control participant visibility; no chat unless plan-creator allows.
5. **Real-Time Updates:** New plans and updates appear live in user feeds.
6. **Backup:** Regular data backups to prevent loss.
7. **System Availability:** Plan visibility based on current time and location context.
8. **Authentication:** Secure login required for all core actions.
9. **Integration Ready:** Open API endpoints for map, calendar, and third-party event sync.
10. **Error Handling:** All user actions return informative messages
11. **Cross-Device Compatibility:** Works across all modern smartphones.
12. **Audit Trail:** Admin audit access (only in internal moderation scenarios).
13. **Compliance:** All user data usage follows legal norms and app transparency standards.

Other Requirements

Database Requirements:

- Use MongoDB (scalable, document-based).
- Regular backups and sharding enabled.

Internationalization:

- Multilingual support (English first; extensible to others).

- Regional time formatting.

Legal & Regulatory:

- GDPR/CCPA compliance.
- In-app terms of service and privacy notice required on signup.

Error Logging:

- Backend logging for API errors, crashes, and frontend exceptions.

Scalability:

- Microservices-based backend for future scale-out.

Integration Support:

- Ready for calendar integration (e.g., Google Calendar).
- Supports event imports via CSV or ICS in the future.

Training & Support:

- App onboarding walkthrough.
- FAQ and contact form built-in.

Maintenance:

- Regular updates every quarter; hotfixes as needed.

Code Reuse:

- Modular front-end components built in React for use across screens.

Appendix A: Glossary

AES
Advanced Encryption Standard; symmetric-key encryption algorithm used to secure data at rest and in transit.

API
Application Programming Interface; a set of HTTP/HTTPS endpoints and protocols that enable communication between client applications and backend services.

Feed
Personalized, time- and location-aware list of Plans presented to the user in a swipeable or scrollable UI.

FR (Functional Requirement)
A specific software capability described in the SRS (e.g., FR-1, FR-2).

Geo-tagging
Attaching geographic coordinates (latitude/longitude) to Plans or user actions for location-based discovery.

Interest Matching
Algorithm that ranks and filters Plans based on overlap between user-selected interests and Plan tags.

JWT (JSON Web Token)
Compact, URL-safe token format used to represent authenticated user identity and permissions.

NFR (Nonfunctional Requirement)
A requirement that specifies criteria for system operation (e.g., performance, security), identified in the SRS (e.g., NFR-1).

OAuth
Open standard for token-based, secure authorization, used for third-party logins (e.g., Google Sign-In).

Plan
A spontaneous, interest-based social gathering created and managed by a Host within Clink.

REQ (Requirement)
A uniquely identified statement of system behavior in the SRS (e.g., REQ-1, REQ-2).

Socket.IO
Library enabling real-time, bidirectional communication between clients and Node.js servers.

WebSocket
Network protocol providing full-duplex communication channels over a single TCP connection for real-time updates.

Post Laboratory Activity:

1. You are required to prepare SRS document for any project. (It could be the mini project you have completed in semester IV
2. Prepare questionnaire for the allotted project considering your lab instructor is the client for requirement gathering.
3. Consider following scenario: An institute is interested in developing a Library Information System (LIS) for the benefit of students and employees of the institute. LIS will enable the members to borrow a book (or return it) with ease while sitting at his desk/chamber. The system also enables a member to extend the date of his borrowing if no other booking for that particular book has been made. For the library staff, this system aids them to easily handle day-to-day book transactions. The librarian, who has administrative privileges and complete control over the system, can enter a new record into the system when a new book has been purchased, or remove a record in case any book is taken off the shelf. Any non-member is free to use this system to browse/search books online. However, issuing or returning books is restricted to valid users (members) of LIS only.

   The final deliverable would a web application (using the recent HTML 5), which should run only within the institute LAN. Although this reduces security risk of the software to a large extent, care should be taken no confidential information (e.g. passwords) is stored in plain text.

   Prepare SRS document for the same in the format discussed in the write-up.

Post Lab Descriptive Questions answers must be handwritten and to be submitted BEFORE the next tern.

1. What are different techniques to gather information for software development?

Information gathering is a crucial step in understanding what the software must do. The common techniques include:

- **Interviews**: One-on-one discussions with stakeholders to understand needs, expectations, and pain points.
- **Questionnaires/Surveys**: Structured forms distributed to multiple users to collect standardized feedback.
- **Observation (Job Shadowing)**: Watching users perform tasks to understand workflow and real-world requirements.
- **Document Analysis**: Studying existing documentation (manuals, reports, legacy system specs) to extract useful information.
- **Workshops**: Interactive group sessions with stakeholders to collaboratively gather and refine requirements.
- **Brainstorming**: Generating creative ideas in group settings for possible features or solutions.
- **Prototyping**: Creating mockups or partial systems to elicit user feedback on functionality and usability.
- **Use Case/Scenario Analysis**: Describing interactions between users and the system to uncover functional requirements.
- **Focus Groups**: Engaging a selected group of users for detailed discussion and opinions about the system.

2. List verification and validation techniques for requirements.

- **Verification Techniques** (Are we building the product right?):
  - Requirements Reviews (peer or stakeholder reviews)
  - Walkthroughs (step-by-step discussions of requirements)
  - Inspections (formal checks for completeness, consistency, correctness)
  - Prototyping (early models to check understanding and feasibility)
  - Checklist-based evaluations
- **Validation Techniques** (Are we building the right product?):
  - User Acceptance Testing (UAT)
  - Requirement traceability (ensuring each requirement is linked to a business need and a test case)
  - Simulation or Modeling

- Feedback from stakeholders (via demos or mockups)
- Field Trials or Pilot Deployments