Contacts Module – Super Detailed Spec (v1)

Primary URLs:
- Contacts / People You May Know / Invitations: `https://connecwrk.com/contacts/`
- My Contacts (full list of confirmed connections): `https://connecwrk.com/my-contacts/`

Related surfaces in global header:
- `Contacts` icon in top nav (next to Home / Messages / Notifications).
- Profile dropdown → links into Contacts & My Contacts.

Left rail (context shared across the whole platform):
- User mini-card: avatar placeholder, name, "My Pages" link, profile completion %, connections count, profile views count.
- Quick CTA (varies by user type): e.g. “If you are a Talent/Artist Create your profile and connect with clients →”.
- Navigation links:
- MSME Connect
- Freelancer Connect
- Assignments
- Talent Connect
- MSME Jobs
- Articles and Trivia

---

1. Page-Level Overview

1.1 Contacts Page (`/contacts/`)

This page is the social graph hub. It does **three** jobs in one screen: 1. **Search for contacts** (top-left search bar):
- Input: "Enter name or email address".
- Submit button: `Search` (teal).
- Behavior: queries existing platform users (by name / email) and returns them with an "Add as Contact" CTA.

1. **Invite external contacts by email** (top-right invite panel):
2. Input placeholder: `Example: abcd@gmail.com, xyz@yahoo.com`.
3. Help text: `Separate email addresses with commas`.
4. Button: `Invite`.

5. Sends invites to people who are **not yet on ConnecWrk**.

6. **Graph growth / discovery** below:

7. `Your connections (X)` + avatar row + "See all" → links to `/my-contacts/`.
8. `People you may know` list (infinite list of suggested members with `Add as Contact`).
9. Each suggestion tile shows:
    - Avatar (placeholder if no photo)

    - Member full name (UPPERCASE in UI screenshot, e.g. `BALWANT SINGH`)
    - CTA button `Add as Contact`.
10. Layout: 2-column responsive grid (left/right card columns per row).

1.2 My Contacts Page (`/my-contacts/`)

This surfacing is the user's **accepted / confirmed** network.
From here, the user can:
- Browse all confirmed contacts.
- (future/expected) open DM via Messages.
- (future/expected) remove / block contacts.

> Note: screenshot not provided for `/my-contacts/`, but "Your connections (1) – See all" is clearly linked, so `/my-contacts/` is the canonical list view of status = confirmed.

---

2. Data Model & State Machine

2.1 Contact Status Enum

```
const CONTACT_STATUS = {
  PENDING: 0,         // Request sent, waiting for the other user to act
  CONFIRMED: 1,       // Both parties are now connected
  DECLINED: 2,        // Request explicitly rejected
  BLOCKED: 3,         // Either side has blocked the other
  UNCONNECTED: 4      // No relationship
};
```

2.2 Tables / Entities (conceptual)

Users - `userId` (PK) - `firstName` - `lastName` - `profilePhoto` - `companyName` - `jobTitle` - `userProfTitle` (headline / professional title)

Contacts - `contactId` (PK) - `userId` (FK → the owner of this row) - `friendUserId` (FK → the other party) - `status` (0/1/2/3/4 per enum above) - `initiatedBy` (who sent the request, userId) - `createdAt` - `updatedAt`

Why two-way?

When a request is accepted, backend is expected to either:
- Flip `status` to `CONFIRMED` for the existing record and also ensure a mirrored record so both users see each other in `/my-contacts/`, OR
- Use a symmetric row that represents both users.

The spec assumes **mirrored rows** for analytics and quick queries per current patterns in similar networking platforms.

---

3. API Layer (Full Detail)

All endpoints below require **Bearer token auth** unless stated. On 401, client should:
- force logout OR
- show re-auth modal and block further actions.

3.1 Suggested / Discovery List – “People you may know”

GET `/api/contact/peopleyouknow?page=1&search=john`

Purpose

Returns recommended people that the current user is *not yet connected to*.

Inputs

- `page` (number, required): pagination cursor/page index.
- `search` (string, optional): free-text (name, email, company) to narrow.

Matching / Ranking logic (server-side)

- Exclude: people already in `PENDING`, `CONFIRMED`, `BLOCKED` with current user.
- Boost rank for:
- Shared company / organization.
- Mutual connections count.
- Industry/skill similarity.
- Same/similar city or region.
- People active recently.

Response

```
{
  "success": true,
  "data": [
    {
      "userId": "123",
      "firstName": "John",
      "lastName": "Doe",
      "profilePhoto": "john.jpg",
      "companyName": "Tech Corp",
      "jobTitle": "Software Engineer",
      "userProfTitle": "Senior Developer",
      "reqSent":
0,              // 0 = can send request, 1 = request already pending
      "mutualConnections": 5     // optional: used for UI badges later
    }
  ],
  "totalPage": 3,
  "currentPage": 1
}
```

Error cases

- 401 Unauthorized → prompt login.
- 500 Internal Server Error → show toast "We couldn't load suggestions right now. Please try again."

UI usage

For each returned user:
- Show avatar (lazy-loaded); default placeholder if `profilePhoto` empty or fails.
- Show FULL NAME (UPPERCASE style).
- Show CTA button: **Add as Contact**.
- If `reqSent === 1`, CTA should be disabled / replaced with `Request Sent`.

3.2 Send Contact Request ("Add as Contact")

GET `/api/contact/addcontact?frienduserId=456`

Purpose

Initiate a connection request from current user → `frienduserId`.

Backend flow

1. Validate caller != target.
2. Check that no active or pending record exists.
3. Insert `Contacts` row with:
4. `status = CONTACT_STATUS.PENDING (0)`
5. `initiatedBy = currentUser`
6. Fire notification to the recipient.
7. Return success.

Response

```
{
  "success": true,
  "message": "Request sent"
}
```

Errors / guards

- Attempt to connect with self → 400 `Cannot connect with yourself`.
- Already connected / pending / blocked → 400 `Connection already exists`.
- Rate limit exceeded (spam prevention, e.g. >50/hour) → 429.
- 401 if auth missing.

Frontend state update

Immediately set `reqSent` for that userId to `1` so the UI reflects "pending" without waiting for another refresh.

```
requestStatus[userId] = true; // disables button in People you may know
```

3.3 Approve / Decline Incoming Request

These are actions **the recipient** can take on a pending request.

Approve
`GET /api/contact/approvefriendreq?frienduserId=456`

Decline
`GET /api/contact/declinefriendreq?frienduserId=456`

Approve flow

- Find pending row where `frienduserId` (the request sender) matches the param and `status = 0` and `friendUserId = currentUser`.
- Update status → `1` (CONFIRMED).
- Create mirrored row for symmetry if needed.
- Increment `connections` counter for both accounts.
- Trigger notification back to requester: "<name> accepted your connection request".

Decline flow

- Update status → `2` (DECLINED).
- Optionally record timestamp to prevent spam re-request for X hours.
- MAY NOT increase counters.

Responses

```
{ "success": true, "message": "Connection approved" }
```

```
{ "success": true, "message": "Request declined" }
```

Errors

- 400 if no matching pending req.
- 401 unauthorized.
- 500 internal server error.

3.4 My Contacts List (Confirmed Connections)

GET `/api/contact/mycontactlist?page=1`

Purpose

Return all confirmed (status = 1) connections for current user. Used by `/my-contacts/` and the mini-strip under "Your connections (X)" at the top of `/contacts/`.

Query logic (conceptual SQL)

```
SELECT *
FROM Contacts
WHERE (
  userId = :currentUserId OR friendUserId = :currentUserId
)
AND status = 1
ORDER BY createdAt DESC
LIMIT 20 OFFSET (page-1)*20;
```

Response structure

```
{
  "success": true,
  "data": [
    {
      "userId": "789",
      "firstName": "Asha",
      "lastName": "Verma",
      "profilePhoto": "asha.jpg",
      "companyName": "FinEdge",
      "jobTitle": "Finance Associate",
      "userProfTitle": "Credit Analyst"
    }
  ],
  "totalPage": 5,
  "currentPage": 1
}
```

UI expectations

- `/my-contacts/` should render an infinite/paginated scroll of confirmed contacts.
- Each card: avatar + name + headline + optional actions:
- "Message" → deep-link to `/messaging/` with that user (DM thread).
- "Remove / Block" (future).

3.5 Pending Requests (Incoming)

GET `/api/contact/pendingcontactreqlist?page=1`

Purpose

List all contact requests **waiting on me** to approve or decline.

Query logic (conceptual SQL)

```
SELECT *
FROM Contacts
WHERE friendUserId = :currentUserId
AND status = 0
AND initiatedBy != :currentUserId
ORDER BY createdAt DESC
LIMIT 20 OFFSET (page-1)*20;
```

Response example

```
{
  "success": true,
  "data": [
    {
      "userId": "333",
      "firstName": "Tarun",
      "lastName": "Kumar",
      "profilePhoto": "tarun.png",
      "companyName": "BuildRight",
      "jobTitle": "Project Manager",
      "userProfTitle": "Operations Lead",
      "requestReceivedAgo": "2h ago"
    }
  ]
}
```

UI expectations

- Each row should show Approve / Decline buttons.
- After Approve → move that entry into `/my-contacts/`.
- After Decline → hide it from this list.

3.6 Invite External Contacts (Email-based growth)

Top-right panel on `/contacts/` shows:

"Invite people to ConnecWrk (Separate email addresses with commas)"
Input placeholder: `Example: abcd@gmail.com, xyz@yahoo.com`
Button: `Invite`

POST `/api/contact/sendinvitation`

Purpose

Let a user invite non-ConnecWrk people by email, optionally in bulk.

Auth

Bearer token required.

Request Body

```
{
  "email": "email1@domain.com, email2@another.com"
}
```

- Multiple emails allowed, comma-separated.
- Client must pre-validate emails before calling.

Validation Rules (client-side before POST)

```
const validateEmail = (email) => {
  const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
  return emailRegex.test(email.trim());
};
```

- Split by comma, trim, run regex on each.
- If ANY invalid, block submission and show inline error.

Response

```
{
  "success": true,
  "message": "Invitations sent"
}
```

Error cases

- 400 invalid email(s) → show toast "One or more emails are invalid".
- 401 unauthorized → force login.
- 429 rate limit (too many invites) → show "Please try again later".
- 500 → "We couldn't send invites right now".

UI behavior after success

- Clear the invite input.
- Show success toast.
- Optionally track analytics event: `invite_sent` with count of emails.

---

4. Frontend State & UX Behavior

4.1 React-ish State Shape

```
const [contactState, setContactState] = useState({
  // Lists
  myContacts: [],              // CONFIRMED connections (for /my-contacts/)
  pendingContacts: [],         // Requests I need to approve/decline
  suggestedContacts: [],       // "People you may know"

  // Paging / search
  loading: false,
  searchQuery: '',
  currentPage: 1,
  totalPages: 1,

  // Request tracking (optimistic UI after Add as Contact)
  requestStatus: {
    // [friendUserId]: true if request already sent
  }
});
```

4.2 Optimistic Update Flow for "Add as Contact"

1. User clicks **Add as Contact** for `userId=123`.
2. Immediately set:

```
setContactState((prev) => ({
  ...prev,
  requestStatus: {
    ...prev.requestStatus,
    [123]: true
  }
}));
```

3. Fire `/api/contact/addcontact?frienduserId=123`.
4. If API returns error, revert and surface toast.

4.3 Search for Existing Members

Top search bar (`Enter name or email address`) should: - Debounce user input.
- Call `GET /api/contact/peopleyouknow?...&search=<query>` OR a dedicated search endpoint

if provided later.
- Replace the suggestions list with search hits.
- Maintain "Add as Contact" CTA.

4.4 Connections Strip ("Your connections (X)")

- Shows avatar(s) of confirmed contacts.
- `See all` → navigates to `/my-contacts/`.
- Use `/api/contact/mycontactlist?page=1` for both the count and for preview avatars.
- If 0, hide avatars and link to `/contacts/` suggestions as onboarding.

---

5. Notifications & Messaging Integration

5.1 Notification Events

When I send someone a request, they get a platform notification:

```
notifyUser(recipientId, {
  type: 'CONNECTION_REQUEST',
  fromUserId: senderId,
  fromUserName: senderName,
  message: `${senderName} wants to connect with you`,
  actions: ['confirm', 'decline']
});
```

When they accept:

```
notifyUser(requesterId, {
  type: 'CONNECTION_CONFIRMED',
  fromUserId: recipientId,
  fromUserName: recipientName,
  message: `${recipientName} accepted your connection request`
});
```

5.2 Messages Integration

- After a connection is CONFIRMED, the UI should surface a "Message" CTA that deep-links to `/messaging/` (the chat module).
- This ensures that networking → conversation is one smooth funnel.

---

6. Security, Privacy, Compliance

6.1 Authorization / Abuse Controls

- Verify `currentUserId !== frienduserId` for `/addcontact`.
- Prevent duplicate requests if already pending or confirmed.
- Respect BLOCKED state: if either user has blocked the other, `/addcontact` and `/approvefriendreq` MUST 400 with `Cannot connect with this user`.
- Rate limit: max ~50 new outgoing requests/hour to stop spam.
- Cool-down: after `DECLINED`, optionally impose a 24h (configurable) lockout so sender can't insta-spam again.

6.2 Privacy Settings Alignment

Privacy Settings (see Privacy & Data Controls spec) define who can see what: - **contactsTabVisibility / contactListVisibility** might be `all_members`, `connections`, or `none`.
- `/my-contacts/` must respect that when another user views your profile.
- **allowMentions** and **openToOpportunities** do not block contact requests, but they affect discovery / recruiter outreach messaging in future.
- **searchEngineVisibility**: does not apply inside the app, but matters for public/SEO pages.

The Contacts module must **call and respect** those settings if/when rendering someone's connections publicly.

6.3 Personal Data Safety / RAG guardrails

- Contact suggestions may expose `firstName`, `lastName`, `companyName`, `jobTitle`, `userProfTitle`.
- DO NOT expose email, phone, or personal identifiers in *People you may know* unless both users have granted visibility.
- AI assistant (Connie) must follow RAG limits:
- Connie can say "You and BALWANT SINGH are not connected yet. You can send a contact request from https://connecwrk.com/contacts/."
- Connie **must not** leak private contact details (email, phone, resume data, etc.) unless that data is already visible to the requesting user per privacy settings.
- Connie must not claim "I sent a contact request" because Connie cannot click buttons or mutate state. Connie must instead instruct: "Click ‘Add as Contact’ next to BALWANT SINGH on the Contacts page."
- Connie must not invent a relationship status. It must ground responses in actual stored status from `/api/contact/mycontactlist` or `/api/contact/pendingcontactreqlist` or else clearly say "I don't have that info."

---

7. Performance & Reliability

7.1 Pagination & Caching

- Page size default: 20.
- Keep `currentPage`, `totalPage` in component state.
- For scrolling, load next page when ~80% scroll reached.

- Cache results in-memory for 5 minutes to avoid hammering the API when navigating back/forth between Contacts and My Contacts:

```
const CONTACT_CACHE_TTL = 5 * 60 * 1000; // 5 minutes

const cache = {
  'peopleyouknow:page1:search=': {
    data: [...],
    timestamp: Date.now(),
    expires: Date.now() + CONTACT_CACHE_TTL
  }
};
```

7.2 Image Optimization

```
<img
  src={IMAGEPATH.upload + "userphoto/" + profilePhoto}
  onError={addDefaultSrc}
  loading="lazy"
  alt={`${firstName} ${lastName}`}
/>

function addDefaultSrc(ev) {
  ev.target.src = "/user-placeholder.jpg"; // fallback avatar
}
```

- Lazy-load avatars for suggestions and My Contacts list.
- Always provide alt text (accessibility + SEO).

7.3 Offline / Error UX

- If `/peopleyouknow` fails → show friendly empty state:
  "We can't load suggestions right now. Please try again."
- If `/mycontactlist` fails on `/my-contacts/` → show:
  "We couldn't load your contacts. Refresh the page."
- All toasts should avoid leaking backend error traces.

---

8. UI Copy / On-screen Strings (from screenshot + expected)

- **Search for contacts**
  Placeholder: `Enter name or email address`
  Button: `Search`

- **Invite people to ConnecWrk (Separate email addresses with commas)**
  Placeholder: `Example: abcd@gmail.com, xyz@yahoo.com`
  Button: `Invite`

- **Your connections (1) – See all**

- Clicking `See all` routes to `/my-contacts/`.

- **People you may know**
  Within each tile:

- BIG NAME (e.g. `BALWANT SINGH`)
- Button: `Add as Contact`

- Left rail CTA variant observed:

- "If you are a Talent/Artist Create your profile and connect with clients →"
  (This CTA changes based on role, e.g. MSME / Freelancer / Talent.)

---

9. Analytics / Logging

Track the following client-side events: 1. `contact_search_submitted`
- props: `{ queryLength, resultCount }`

1. `contact_invite_submitted`
2. props: `{ emailsCount }`

3. `contact_request_sent`

4. props: `{ targetUserId }`

5. `contact_request_approved` / `contact_request_declined`

6. props: `{ fromUserId }`

7. `my_contacts_viewed`

8. props: `{ totalContactsShown }`

These analytics can help growth + anti-abuse (spam bursts, invite quality, etc.).

---

10. RAG / Connie Integration Requirements (VERY IMPORTANT)

This module is ingested into Connie (the AI assistant) for retrieval-augmented responses. Connie must:

1. **Ground every answer in platform reality.**

2. Connie should reference actual flows:
    - "Go to Contacts at https://connecwrk.com/contacts/."
    - "Click Add as Contact next to their name."
    - "If they accept, they'll appear under 'Your connections' and in https://connecwrk.com/my-contacts/."
3. **Respect privacy settings.**
4. If a user asks "What is Arpit Bansal's email?", Connie must refuse and explain that contact details are private and only shared by the user themselves.
5. Connie can describe *how* to reach them in-platform: "Send a contact request, then you can message them via the Messages tab once you're connected."
6. **Never claim to take actions it cannot perform.**
7. Connie cannot actually send requests or invites.
8. Approved phrasing:
    - "You can click 'Add as Contact' next to BALWANT SINGH on your Contacts page."
    - "Use the Invite box on https://connecwrk.com/contacts/ to paste comma-separated emails, then click Invite."
9. NOT approved:
    - "I've added BALWANT SINGH to your contacts." ❌
    - "I've invited your team." ❌
10. **Be honest about visibility and status.**
11. Connie can check (via future backend connector) if a user is in `/api/contact/mycontactlist` vs `/api/contact/pendingcontactreqlist`.
12. If Connie does *not* have that data in-context, answer must include uncertainty:
    - "I can't see whether your request is approved yet. You can confirm under 'Your connections' on https://connecwrk.com/my-contacts/."
13. **Preserve safety and anti-harassment rules.**
14. If someone was declined or blocked, Connie must nudge toward respectful behavior instead of suggesting harassment / repeat-spam.
15. Example answer:
    - "It looks like your previous request wasn't accepted. You'll have to wait before sending another connection request."
16. **Use correct terminology.**
17. Use "Add as Contact" (not "Follow" or "Friend request").
18. Use "Your connections" for confirmed contacts.

19. Use "People you may know" for recommendations.

20. **Avoid PII leakage.**

21. Connie must not surface private phone numbers, emails, resumes, etc., even if they appear in training.
22. Connie must stick to public profile fields already surfaced in UI cards (name / title / company), plus general guidance.

---

11. End-to-End User Journeys

Journey A: Add someone from "People you may know"

1. User lands on `https://connecwrk.com/contacts/`.
2. Scrolls to "People you may know".
3. Clicks `Add as Contact` next to `BALWANT SINGH`.
4. Frontend immediately flips that row to `Request Sent`.
5. Backend creates pending row (status=0) via `/api/contact/addcontact`.
6. Target user gets notification.
7. When target user accepts via `/api/contact/approvefriendreq`, both become CONFIRMED.
8. The sender now sees that person under `Your connections` and `/my-contacts/`.

Journey B: Invite someone not on ConnecWrk

1. User types multiple emails into Invite box (e.g. `a@x.com, b@y.com`).
2. Client validates each email with regex.
3. Client POSTs `/api/contact/sendinvitation` with the combined email string.
4. Backend sends invitation emails + logs invites.
5. UI clears the field + shows success toast.

Journey C: Approve a pending request

1. Target user opens Contacts (or a dedicated Pending view surface).
2. Sees an incoming request from `TARUN KUMAR` with `Approve` / `Decline`.
3. Clicks Approve.
4. Frontend calls `/api/contact/approvefriendreq?frienduserId=<TARUN_ID>`.
5. Backend updates status to CONFIRMED and notifies TARUN.
6. TARUN now sees this user in `Your connections` and can DM them via Messages.

---

12. Future Enhancements / To-Do

- **Block / Unblock UI:** surface BLOCKED state (status=3) and allow unblocking.
- **Mutual connections badge:** display `mutualConnections` count in People you may know cards to improve trust.
- **Sorting & filtering in `/my-contacts/`:**
- By name A→Z / Z→A.
- By company.
- By "Recently active".

- **Bulk actions / multi-invite improvements:**
- CSV paste import.
- Show which emails already exist on ConnecWrk vs which got invited.
- **Connection strength scoring:**
- Weighted by mutual connections (40%), same company (30%), same industry (20%), geo proximity (10%).
- Use to sort suggestions.
- **Anti-harassment cooldown:**
- Enforce wait (e.g. 24h) after DECLINED before allowing another request to same user.
- **RAG hook for Connie:**
- Allow Connie to surface “I see you’ve sent a request and it’s pending” ONLY if we explicitly pass her the real-time status from `/api/contact/pendingcontactreqlist`.
- Never hallucinate statuses.

TL;DR

The Contacts system is a lightweight professional networking graph with:
- Discovery (`/api/contact/peopleyouknow`)
- Requests (`/api/contact/addcontact`)
- Approval / Decline (`/api/contact/approvefriendreq`, `/api/contact/declinefriendreq`)
- Confirmed network (`/api/contact/mycontactlist`)
- Pending queue (`/api/contact/pendingcontactreqlist`)
- External invitations (`/api/contact/sendinvitation`)
- Privacy integration, abuse prevention, analytics, and AI (RAG) safety rules built in.

This spec should be treated as the source of truth for Contacts + Network growth flows on ConnecWrk.