FULL-STACK INTERNSHIP — TECHNICAL ASSESSMENT
Project: **Real-Time Customer ↔ Business Chat Service (ServiHub)**
Submission deadline: **Sunday • 1 June 2025 • 23 : 59 (SGT)**

---

0 Executive summary

Design, build, and deploy a **production-ready real-time chat service** that lets customers converse with businesses on the ServiHub platform. Reviewers will:

1. Open **two browser tabs** (one pretending to be the customer, one the staff agent).
2. Send messages back and forth.
3. Expect **instant delivery**, presence & typing indicators, and unread badges.

Your solution must ship with:

- Typed **REST + WebSocket APIs**
- **PostgreSQL** persistence via **Prisma**
- Authentication using **ServiHub JWTs**
- CI, automated tests, Docker support
- Clear docs **and** a **public demo URL**

---

1 Scope & core use-cases

#	Actor flow	Expected behaviour
1	Customer opens chat widget on a Business page	Load **or** create a 1-to-1 conversation (`customerId ↔ businessId`). Show unread badge.
2	Multi-agent support room	Any staff member (`ParticipantRole = AGENT`) can reply. Label shows “Business • Alice”.
3	Presence + typing	Push **online / offline** + **“is typing…”** via Redis pub/sub.
4	Offline notifications	If customer is offline, store message **and** trigger the platform’s e-mail / push hook.
5	*(Stretch)* Community broadcast	Business can post announcements to an opt-in community chat.

---

2 Required deliverables

Path / artefact	What we expect
`docs/architecture.md`	High-level diagram, component list, major trade-offs (gateways, Redis fan-out, failure modes).
`prisma/schema.prisma`	**Exact** models in §3.2 with migrations.
`prisma/seed.ts`	Seeds **2 businesses, 1 customer, 1 agent, 1 demo conversation**.
`src/plugins/chat.ts`	Fastify plugin registering REST + `@fastify/websocket` routes.
`src/services/…`	Typed service layer, plus `@fastify/rate-limit` (20 msgs / 5 s).
`widgets/ChatWidget.tsx`	React widget ≤ 20 kB gzipped (Zustand / SWR, themeable CSS variables).
`tests/`	Jest unit **+** integration tests (≥ 80 % coverage, WS mocks).
`Dockerfile` & `docker-compose.yml`	App + Postgres + Redis listening on **port 3000**.
`.github/workflows/ci.yml`	Lint → type-check → test → upload coverage badge.
`README.md` & `DEPLOY.md`	Local dev, env vars, **copy-paste deploy steps**.
(Bonus)	End-to-end encryption demo • Searchable history (Typesense) • Voice-room POC • `Reaction` table.

---

3 Technical requirements

3.1 Backend

- **WebSocket-first** (`@fastify/websocket`) with **SSE fallback**.
- Verify ServiHub **JWT** in WS upgrade handler; reject unauthenticated sockets.
- **Rate-limit** (20 msgs / 5 s) using `@fastify/rate-limit`, backed by Redis for multi-node.
- **Stateless** nodes; Redis pub/sub (or NATS) for cross-node broadcast.
- `BIGSERIAL` IDs + **compound indexes** (see schema).
- JSON logs; `/health` returns **200 OK**; WS ping every **25 s**.

3.2 Data model (Prisma 5)

```
enum ConversationType   { DIRECT SUPPORT_ROOM COMMUNITY }
enum ParticipantRole    { CUSTOMER AGENT OWNER }
enum MessageContentType { TEXT FILE IMAGE VIDEO OTHER }

model User {
  id           BigInt       @id @default(autoincrement())
  participants Participant[]
  messages     Message[]    @relation("MessageSender")
}

model Business {
  id            BigInt       @id @default(autoincrement())
  name          String
  conversations Conversation[]
}

model Conversation {
  id            BigInt       @id @default(autoincrement())
  business      Business      @relation(fields: [businessId], references:
[id])
  businessId    BigInt
  type          ConversationType
  participants  Participant[]
  messages      Message[]
  createdAt     DateTime      @default(now())
  updatedAt     DateTime      @updatedAt
  @@index([id, createdAt])            // infinite scroll
}

model Participant {
  id             BigInt       @id @default(autoincrement())
  conversation   Conversation @relation(fields: [conversationId],
references: [id])
  conversationId BigInt
  user           User         @relation(fields: [userId], references: [id])
  userId         BigInt
  role           ParticipantRole
  joinedAt       DateTime     @default(now())
  @@unique([conversationId, userId])  // join once
}

model Message {
  id             BigInt            @id @default(autoincrement())
  conversation   Conversation      @relation(fields: [conversationId],
references: [id])
  conversationId BigInt
  sender         User?             @relation("MessageSender", fields:
[senderId], references: [id])
  senderId       BigInt?
  contentType    MessageContentType @default(TEXT)
  body           String?
  fileUrl        String?
  mimeType       String?
  createdAt      DateTime          @default(now())
  readAt         DateTime?
  editedAt       DateTime?
  deletedAt      DateTime?         // soft-delete
  attachments    Attachment[]
  @@index([conversationId, createdAt])
  @@index([senderId, createdAt])
}

model Attachment {
  id        BigInt  @id @default(autoincrement())
  message   Message @relation(fields: [messageId], references: [id])
  messageId BigInt
  url       String
  mimeType  String
  width     Int?
  height    Int?
  sizeBytes Int?
}

/* Optional stretch — reactions */
model Reaction {
  message   Message @relation(fields: [messageId], references: [id])
  messageId BigInt
  user      User    @relation(fields: [userId], references: [id])
  userId    BigInt
  emoji     String
  @@id([messageId, userId, emoji])
}
```

3.3 Front-end widget

- **React + TypeScript**; exported **UMD bundle** `chat-widget.umd.js`.
- Responsive bubbles, file preview, unread badge, staff labels.
- **WCAG 2.1 AA** — ARIA live regions + full keyboard navigation.

3.4 Non-functional

Concern	Requirement
Performance	≤ 150 ms **P99** RTT on localhost @ 100 sockets
Security	JWT ACL, HTML sanitisation, **soft-delete (`deletedAt`)**, 10 MiB upload cap
Accessibility	WCAG 2.1 AA compliant widget
Dev experience	`npm run dev`, `docker-compose up`, CI green on fresh clone

3.5 Environment & style

- Target runtime: **Node 20 LTS**, **PNPM 8**, **Prisma 5**.
- Code must pass **ESLint + Prettier** (`npm run lint`).
- Commit messages: **Conventional Commits** style preferred.

---

4 Hosting & demo

4.1 Platform matrix (choose what suits you)

Layer / need	✅ Works well	⚠️ Limitations
Static widget	Vercel, Netlify, Render, etc.	—
REST API	Any of the above (serverless or server-full)	—
WebSocket gateway	Render ★, Railway ★, Fly.io ★, Heroku ★	Vercel Edge Functions **do not support WS**
Managed Postgres	Add-ons on Render / Railway / Fly / Heroku	Free tiers often limit connections

4.2 Demo checklist

1. **Public frontend URL** — e.g. `https://chat-demo.servihub.app`.
2. **Public backend URL** — e.g. `https://chat-api.onrender.com`; `/health` returns 200 OK.
3. Two tabs exchange messages in real-time; SSE fallback proven by disabling WS in DevTools.
4. `DEPLOY.md` documents exact steps or CI pipeline used.

---

5 Evaluation rubric (100 pts)

Domain	Pts	Indicators
Code quality	25	Idiomatic TS, modular, lint-clean
Real-time design	20	Correct WS + SSE flow, Redis fan-out, back-pressure handling
Data modelling	12	Normalisation, **compound indexes**, migrations
Front-end UX	10	Embeddable, responsive, accessibility, theming
Testing & CI	15	≥ 80 % coverage, WS integration tests, CI badge
Security & perf	10	JWT verify, rate-limit, performance evidence
Docs & DevX	8	Clear README / DEPLOY, Docker compose
Soft-delete	**+2 bonus**	Uses `deletedAt` correctly

---

6 Submission checklist

1. **Public GitHub repo** with commit history.
2. Add `.env.example` (JWT secret, DB URL, CORS origin, etc.).
3. Fresh-clone test must work:
4. `git clone <repo> && cd chat-servihub`
5. `docker-compose up --build -d`
6. `npm ci && npm test`
7. Tag release **`v0.1.0`**.
8. Include:
    - `README.md` — local setup, scripts, widget embed snippet.
    - `DEPLOY.md` — deploy steps or CI pipeline config.
    - `RETROSPECTIVE.md` — trade-offs, known issues, next steps.
    - *(Optional)* live demo link.

---

7 Retrospective prompt

1. **If you had two extra weeks, what would you build next — and why?**
2. **What was the single biggest trade-off you made during development?**
