**Virtual Discord Office App Documentation**

StackFrame Studios • Project Blueprint

Prepared by: StackFrame Studios  
Date: April 2025

Part 1: Requirements & Planning

1. Functional Requirements:

- Allow users to register and sign in securely.

- Provide a form where users can describe their company/job setup.

- Let server creators browse and accept incoming requests.

- Include built-in messaging so clients and creators can chat.

- Admins should be able to monitor requests and manage creators.

- Notify users via email or in-app when request updates happen.

- Users should be able to track their request status (pending, in progress, done).

2. Non-Functional Requirements:

- Should scale easily if the user base grows.

- Needs to keep user info secure.

- Should be usable on phones and desktops.

- Optional: add support for multiple languages later.

3. Target Users:

- Remote teams

- Freelancers

- Startups

- Educational groups

---

Part 2: Tech Stack

Frontend:

- React.js for the core interface

- Material UI for consistent, modern styling

- React Hook Form for handling input

- Zustand only if global shared state becomes complex

Backend:

- FastAPI with Python for fast, clean backend work

- PostgreSQL hosted on Neon

- Firebase Auth for free and easy user authentication

- Firebase Realtime DB to power chat features

- Nodemailer (Gmail or SendGrid) to send out emails

Hosting:

- Vercel for frontend deployment

- Render for backend hosting

- GitHub Actions for auto-deploy

- Neon for PostgreSQL (serverless)

---

Part 3: Architecture & Workflow

User Flow:

1. User signs up and logs in.

2. They submit a request form.

3. Creators browse these requests and accept ones they like.

4. The matched creator and user chat via in-app messaging.

5. The creator finishes building the server and marks it complete.

6. The user gets notified and can rate the creator.

Request Logic:

- Forms suggest channels and roles based on industry.

- Creators finalize the setup.

---

Part 4: Implementation

Creator Verification:

- Creators apply with their experience or portfolio.

- Admins review and approve or deny.

- Verified badges are shown on trusted profiles.

Chat System:

- Built with Firebase Realtime DB

- Chat only starts once a creator accepts a request

- Messages show name, time, and live updates

Request Statuses:

- Pending

- Accepted

- In Progress

- Completed

- Cancelled

Backend API:

- Create, update, and fetch requests

- Messaging endpoints tied to request IDs

- Creator applications and admin approvals

Database Tables:

- Users, Requests, Messages, Creator Applications

Notifications:

- Email alerts for important events like approval or completion

---

Part 5: Deployment & Collaboration

Deployment:

- Vercel auto-deploys frontend from GitHub

- Render hosts FastAPI backend

- Neon hosts the database

Firebase Rules:

- Set up read/write access only for users in a chat

- Use Firebase's built-in rule simulator to test

Collaboration:

- Use GitHub branches per feature

- Pull requests for clean code reviews

- GitHub Actions for linting and testing

Extras:

- User ratings for creators

- Option to tip or subscribe for premium services

---

Part 6: Design & UX

UI Components:

- Reusable forms and buttons

- Request cards

- Creator profile tiles

- Chat interface with typing indicators

- Status labels and page feedback

Accessibility:

- Clear color contrast and keyboard nav

- Helpful error messages and labels

Performance:

- Lazy-load heavy components

- Cache data smartly

- Use chat listeners efficiently

Analytics (Mostly Extra work):

- Track user behavior with Plausible or PostHog

- See what industries or features are most used

Onboarding:

- Clean landing page

- Guided tooltips for first-time users

- FAQ and visual walk-through

---

GAMBARE! GAMBARE!!!