FlowDeck - Complete Project Documentation (Part 1)

📋 Table of Contents - Part 1

🎯 Project Overview

What is FlowDeck?

FlowDeck is a **production-ready, enterprise-grade organization workflow management system** built with Flask 3.0 and SQLite. It's designed to streamline team collaboration, task management, and organizational communication in a single unified platform.

Key Highlights

- ✅ **31 Database Tables** - Comprehensive normalized schema (3NF)
- ✅ **Real-time Communication** - Socket.IO powered chat and notifications
- ✅ **Role-Based Access Control** - Admin, Manager, Team Lead, Employee, Intern roles
- ✅ **Kanban Board** - Drag-and-drop task management
- ✅ **Meeting Management** - Schedule, track, and manage meetings with Google Meet integration
- ✅ **Time Tracking** - Built-in time logs and productivity analytics
- ✅ **Leave Management** - Request, approve, and track employee leaves
- ✅ **Multi-tenant Ready** - Organization-based separation

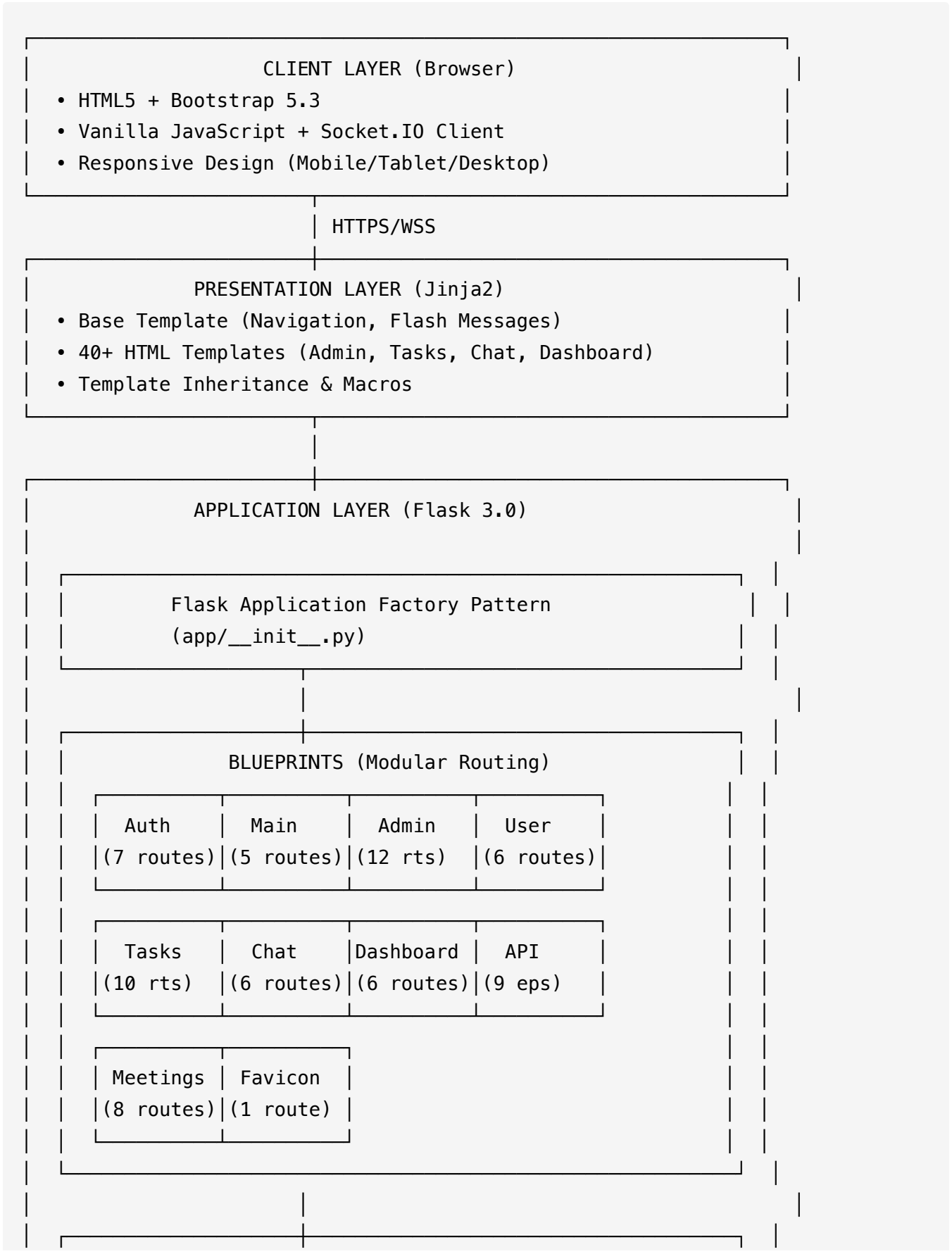
- ✅ **Indian Context** - Pre-seeded with 26 Indian employees, Indian holidays, Hindi/English messaging

Target Users

Role	Capabilities
Admin	Full system access, user management, organization settings
Manager	Team management, task assignment, leave approvals
Team Lead	Task management, team coordination
Employee	Task execution, time tracking, leave requests
Intern	Limited access, view-only permissions

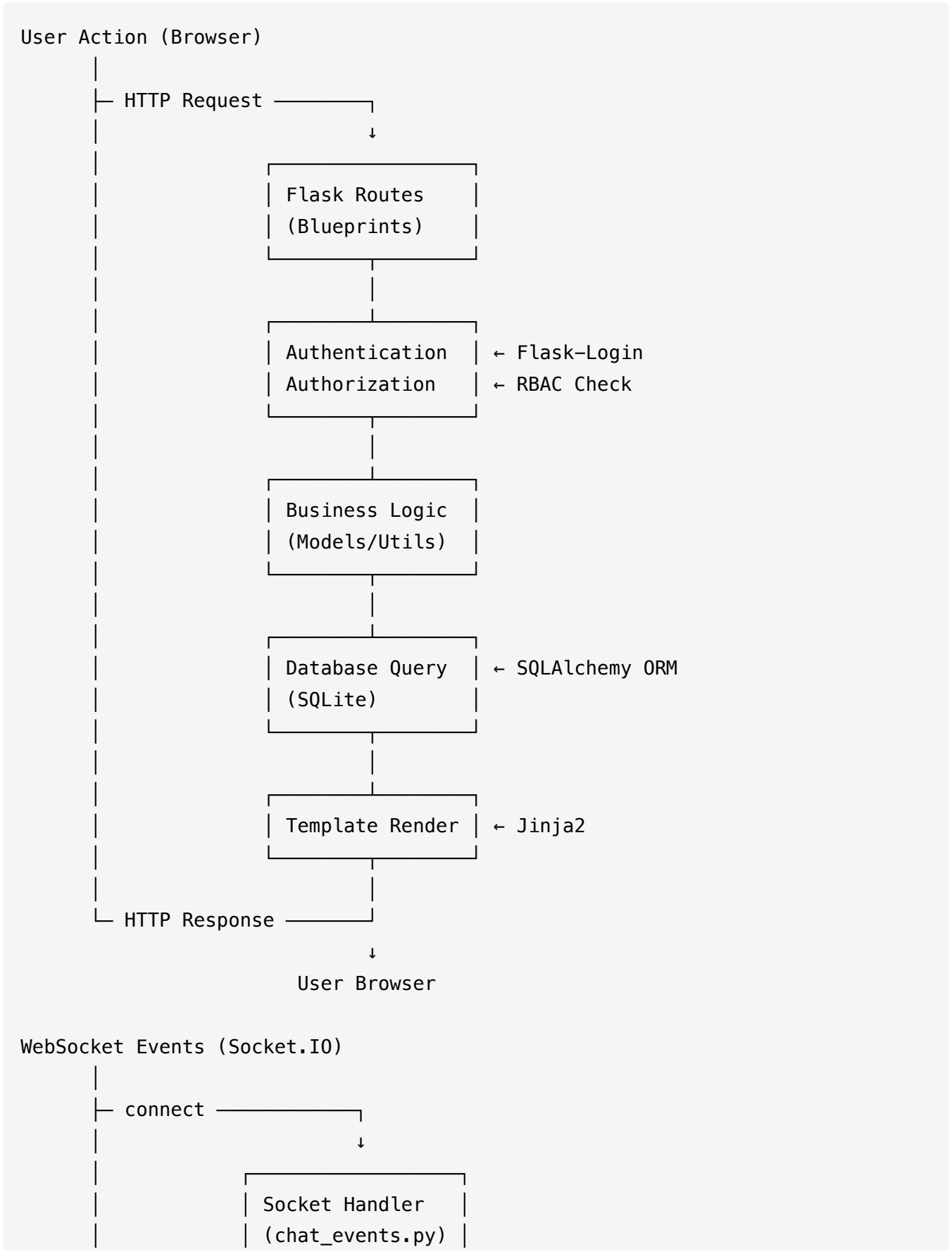
🏗️ System Architecture

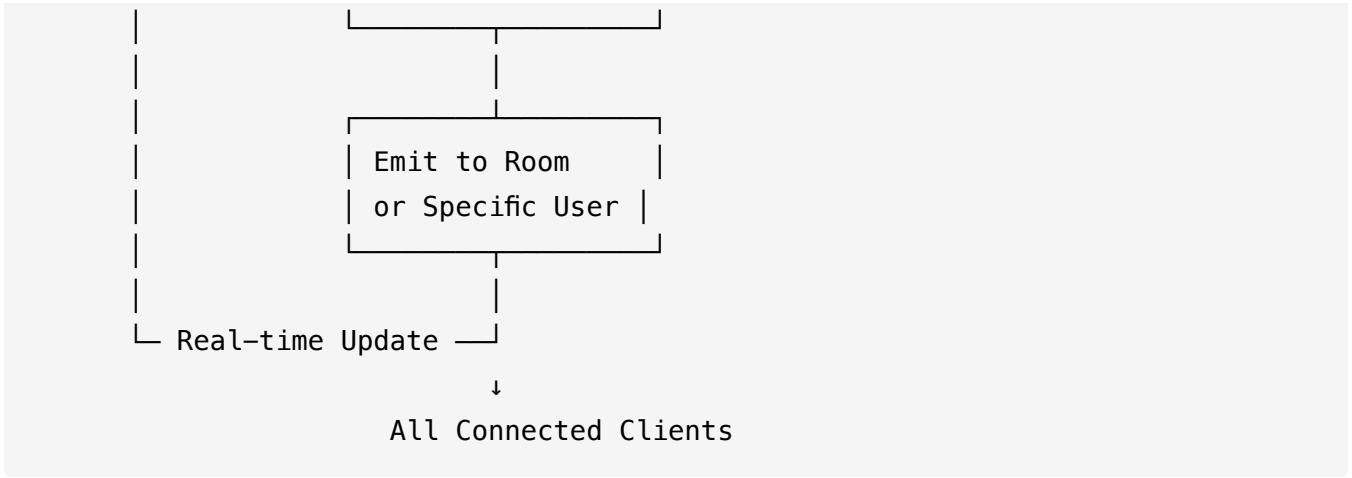
Architecture Layers


CLIENT LAYER (Browser)
• HTML5 + Bootstrap 5.3
• Vanilla JavaScript + Socket.IO Client
• Responsive Design (Mobile/Tablet/Desktop)
HTTPS/WSS
PRESENTATION LAYER (Jinja2)
• Base Template (Navigation, Flash Messages)
• 40+ HTML Templates (Admin, Tasks, Chat, Dashboard)
• Template Inheritance & Macros
APPLICATION LAYER (Flask 3.0)
Flask Application Factory Pattern
(app/__init__.py)
BLUEPRINTS (Modular Routing)
Auth
(7 routes)
Main
(5 routes)
Admin
(12 rts)
User
(6 routes)
Tasks
(10 rts)
Chat
(6 routes)
Dashboard
(6 routes)
API
(9 eps)
Meetings
(8 routes)
Favicon
(1 route)


SOCKET.IO EVENT HANDLERS
Chat Events (9)
• join_room
• leave_room
• send_message
• typing
• read_receipt
Notifications(3)
• notify_user
• mark_read
• broadcast
BUSINESS LOGIC LAYER
• Models (User, Task, Message, Meeting, etc.)
• Utilities (Email, AI, Validators, Seed)
• Authentication (Flask-Login, Password Hashing)
• Authorization (RBAC, Permissions)
DATA ACCESS LAYER (SQLAlchemy)
• ORM Mapping
• Query Builder
• Database Migrations (Flask-Migrate/Alembic)
• Triggers, Views, Indexes
DATABASE LAYER (SQLite/PostgreSQL)
• 31 Tables (3NF Normalized)
• Foreign Key Constraints
• Cascade Delete/Set Null
• Indexes on frequently queried columns

Request Flow Diagram


User Action (Browser)
HTTP Request
Flask Routes
(Blueprints)
Authentication
Authorization
← Flask-Login
← RBAC Check
Business Logic
(Models/Utils)
Database Query
(SQLite)
← SQLAlchemy ORM
Template Render
← Jinja2
HTTP Response
User Browser
WebSocket Events (Socket.IO)
connect
Socket Handler
(chat_events.py)



💻 Technology Stack

Backend Framework

Technology	Version	Purpose
Flask	3.0.3	Core web framework
Flask-SQLAlchemy	3.1.1	ORM for database operations
Flask-Login	0.6.3	User session management
Flask-SocketIO	5.3.6	Real-time WebSocket communication
Flask-Mail	0.10.0	Email notifications
Flask-WTF	1.2.1	Form handling & CSRF protection
Flask-Migrate	4.0.7	Database migrations (Alembic)

Database

Technology	Purpose
SQLite	Development database (easy setup)

Technology	Purpose
PostgreSQL	Production-ready (migration supported)
MySQL	Production alternative (migration supported)

Frontend

Technology	Version	Purpose
Bootstrap	5.3	Responsive UI framework
jQuery	3.x	DOM manipulation
Socket.IO Client	4.x	WebSocket client
Font Awesome	6.x	Icons
Chart.js	(Optional)	Analytics charts

Python Libraries

```
# Core Dependencies
Flask==3.0.3
Flask-SQLAlchemy==3.1.1
Flask-Login==0.6.3
Flask-Mail==0.10.0
Flask-SocketIO==5.3.6
Flask-WTF==1.2.1
Flask-Migrate==4.0.7

# Utilities
python-dotenv==1.0.1      # Environment variables
Werkzeug==3.0.3           # Security utilities
email-validator==2.2.0    # Email validation
Pillow==10.4.0            # Image processing

# Real-time Communication
python-socketio==5.11.2
python-engineio==4.9.1
eventlet==0.36.1

# Forms & Validation
WTForms==3.1.2

# HTTP & APIs
requests==2.32.3

# AI Integration (Optional)
openai==1.40.0

# Google APIs (Optional)
google-auth==2.32.0
google-api-python-client==2.140.0

# Email Services (Optional)
sendgrid==6.11.0

# Security
PyJWT==2.9.0
bleach==6.1.0
```

```
# Server
gunicorn==22.0.0          # Production WSGI server
```

Development Tools

- **Python**: 3.9 - 3.13 (3.11-3.12 recommended)
- **pip**: Latest version
- **venv**: Virtual environment
- **Git**: Version control

🗄️ Database Design

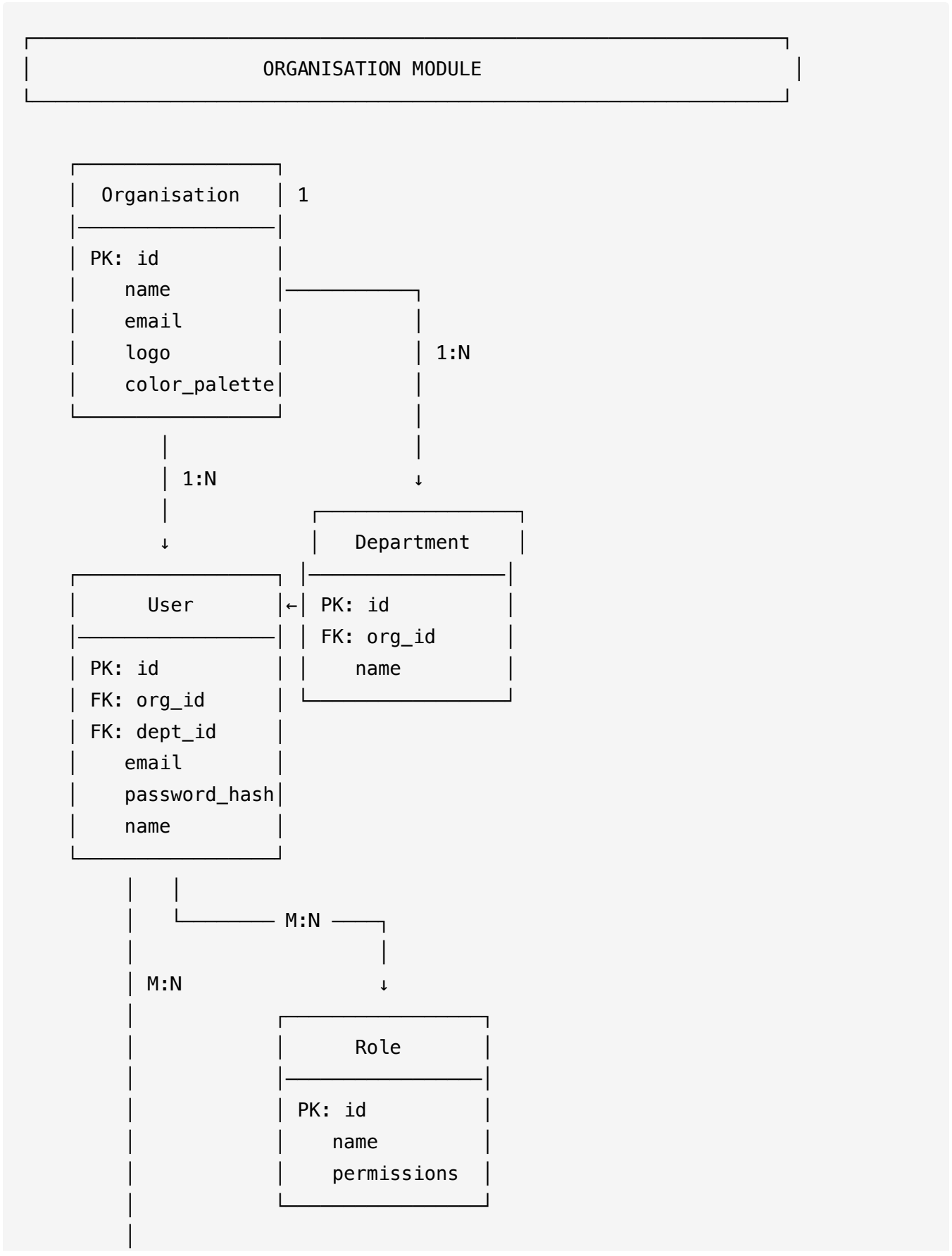
Database Schema Overview

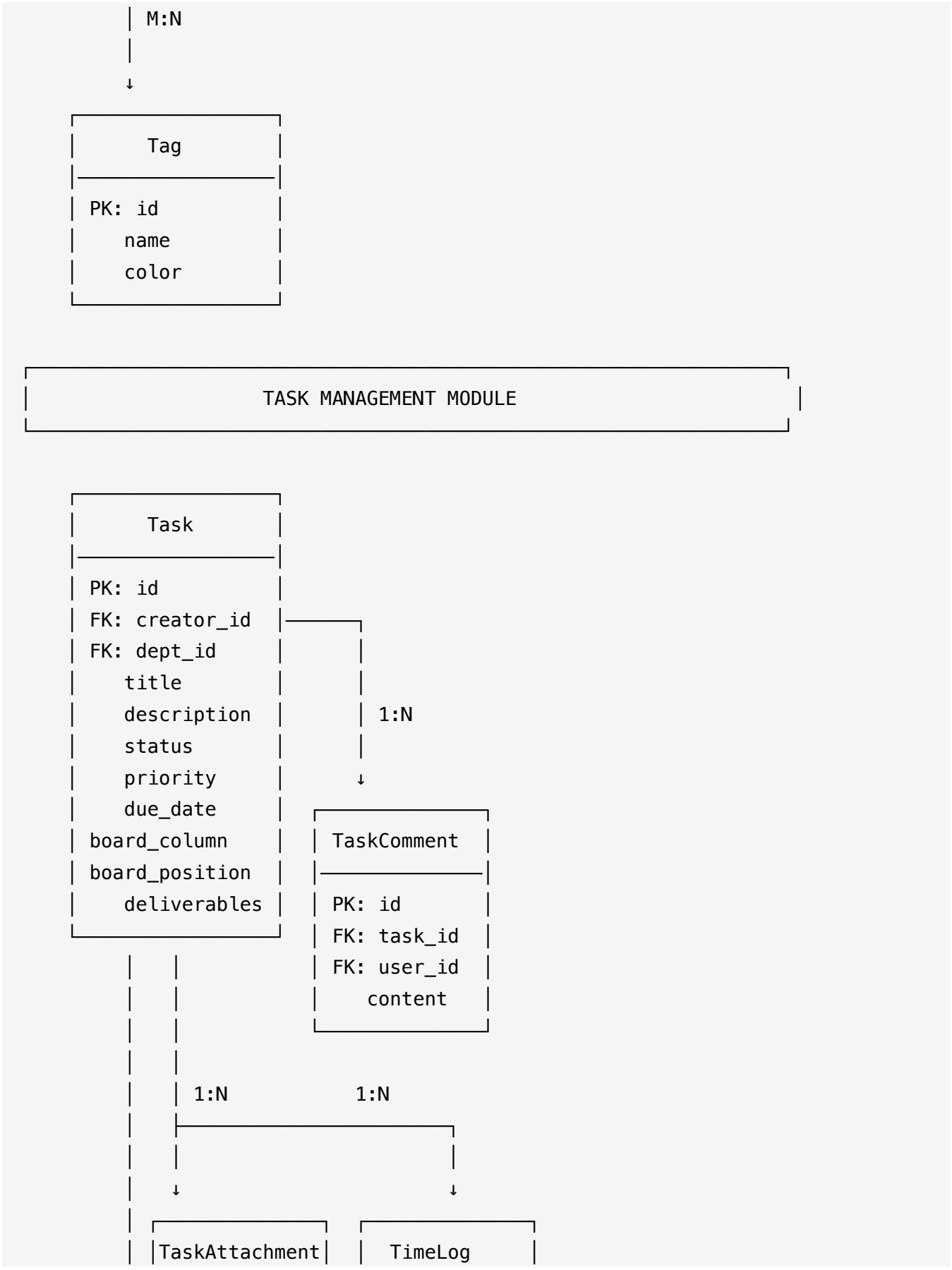
FlowDeck uses a **normalized relational database design (3NF)** with **31 tables** organized into logical modules.

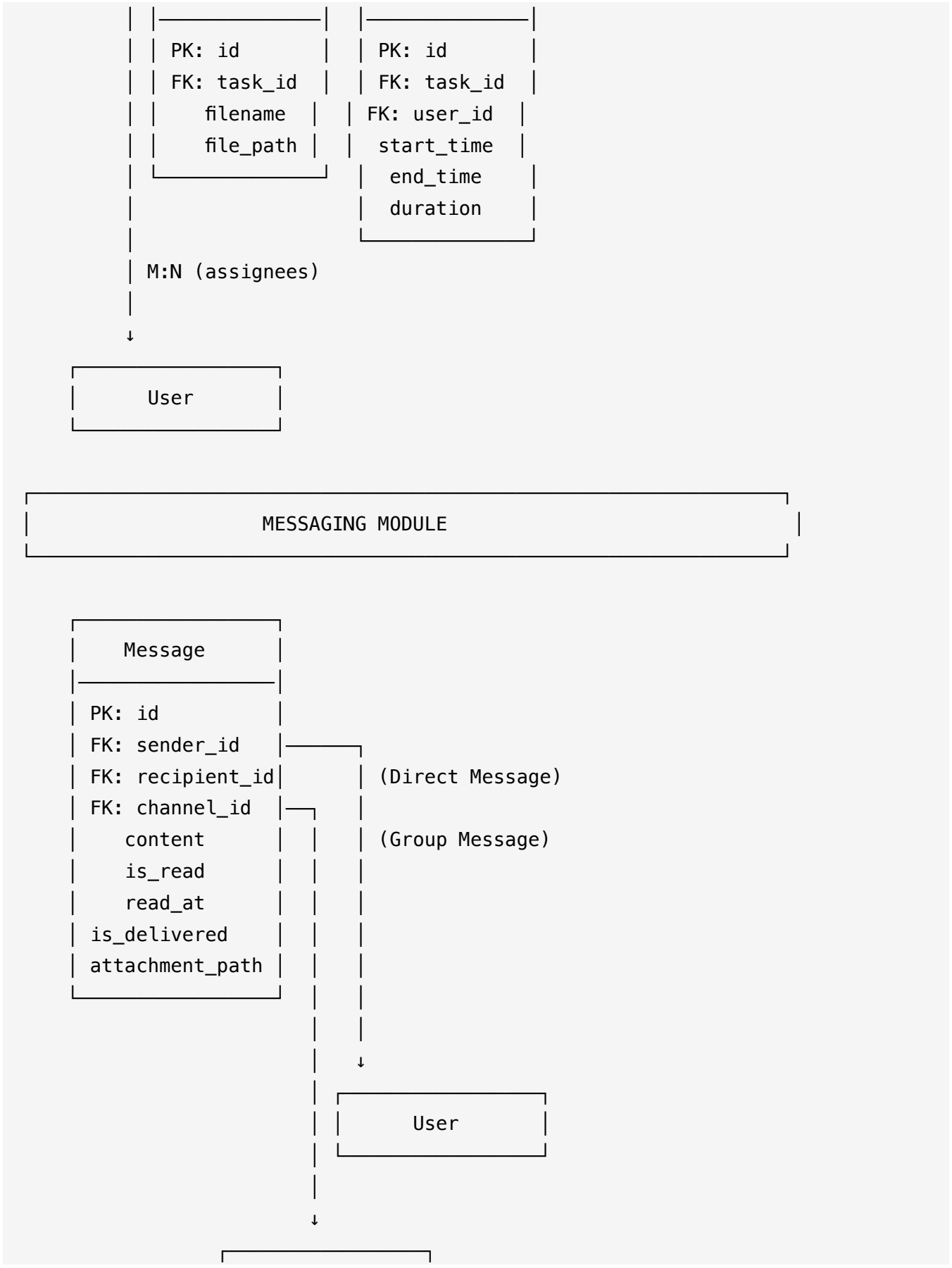
Database Statistics (Current Seed Data)

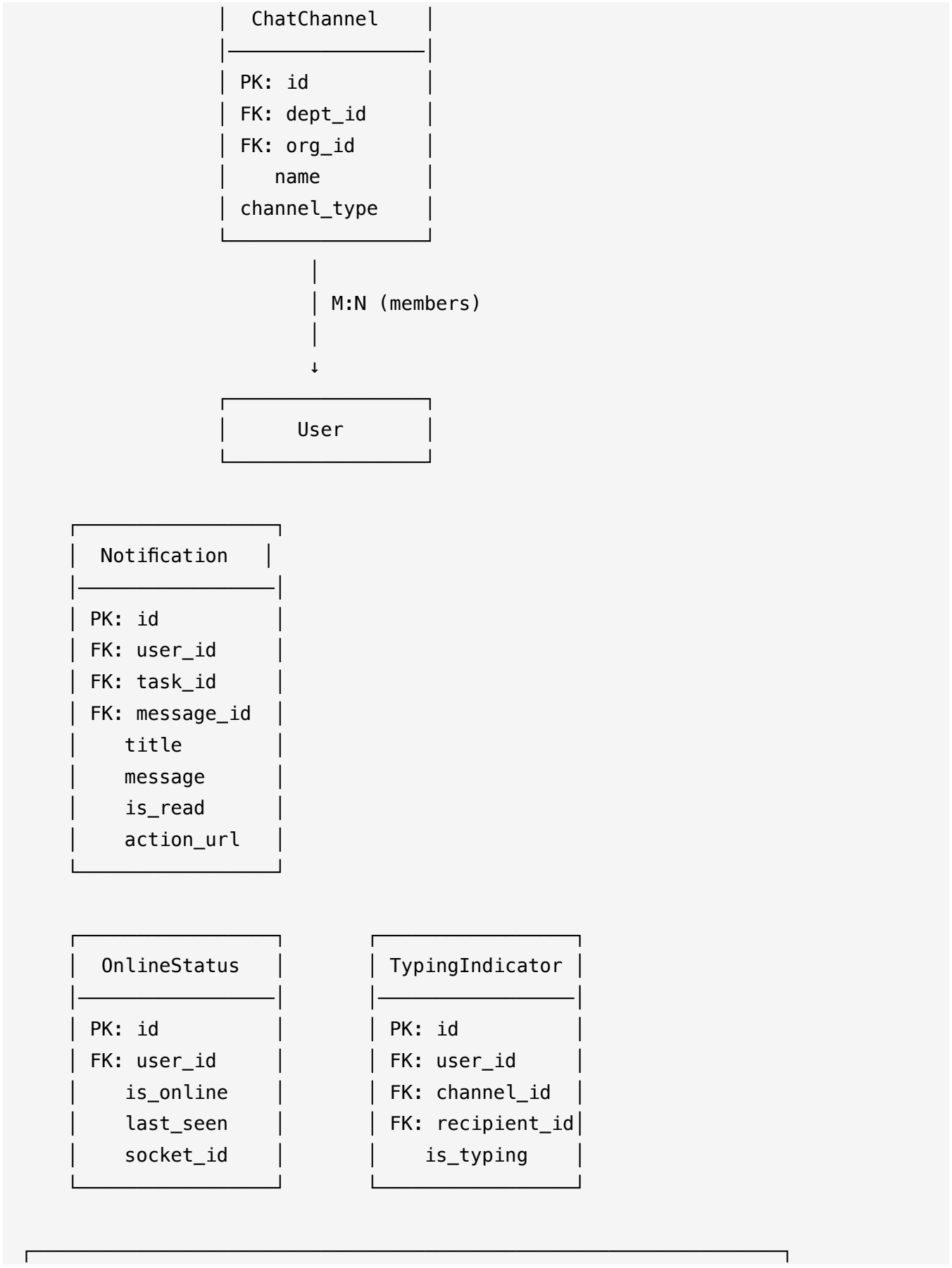
```
📊 DATABASE SUMMARY
═══════════════════════════════════════════
Organizations:        1
Departments:          8
Users:               26
Roles:                5
Tags:                12
Tasks:              134  (4-6 per person)
Task Comments:       89
Task Attachments:    23
Task History:       356
Time Logs:          273
Meetings:            10
Meeting Attendees:   47
Agenda Items:        18
Meeting Notes:       12
Meeting Attachments:  7
Messages:           195  (64 direct, 131 channel)
Chat Channels:        9
Notifications:      156
Online Status:       26
Typing Indicators:    0
Analytics Reports:    4
Holidays:            11  (Indian holidays)
Leave Requests:      37
Audit Logs:         412
System Settings:      6
Email Templates:      5
═══════════════════════════════════════════
```

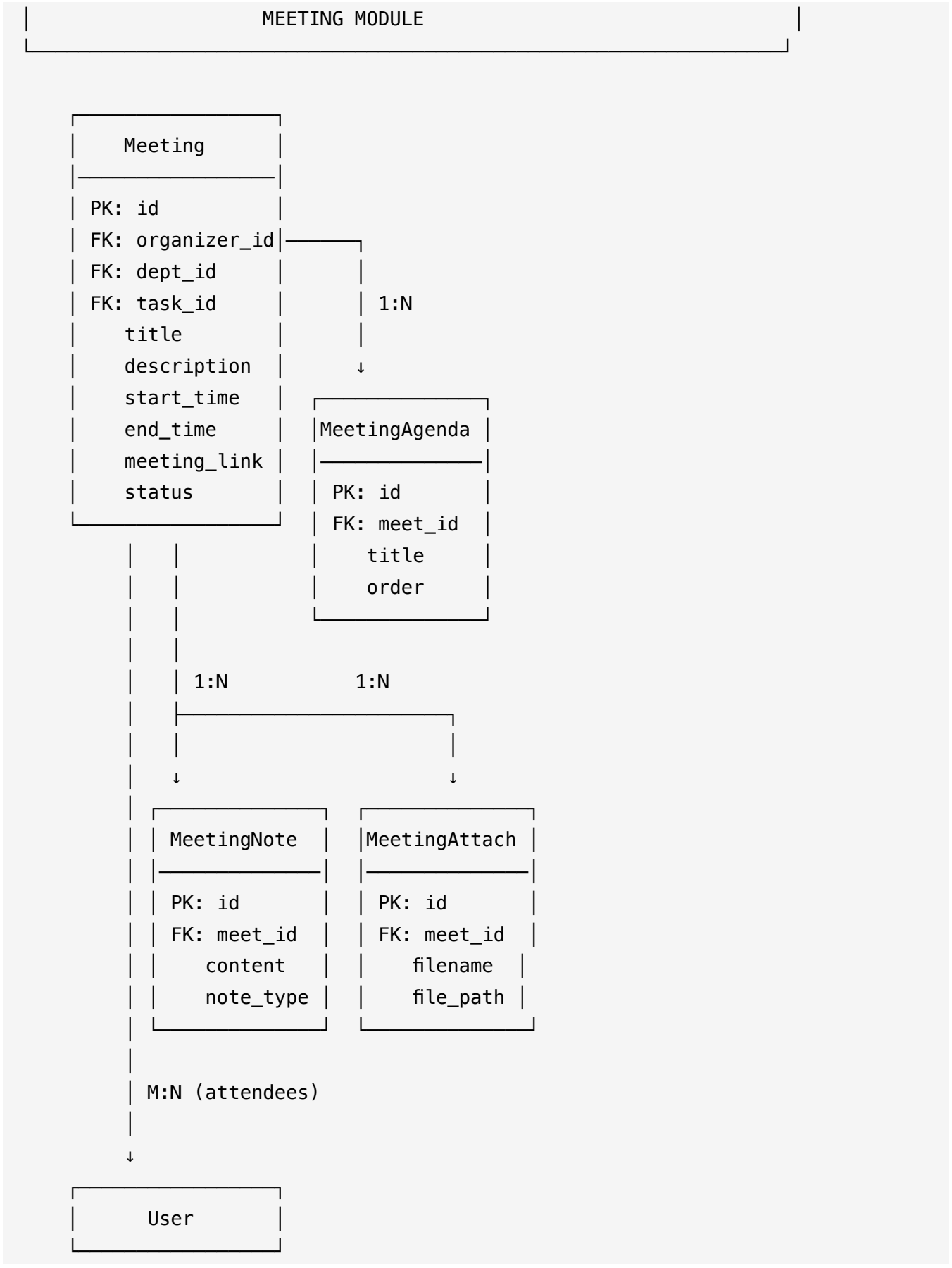
Entity-Relationship Diagram

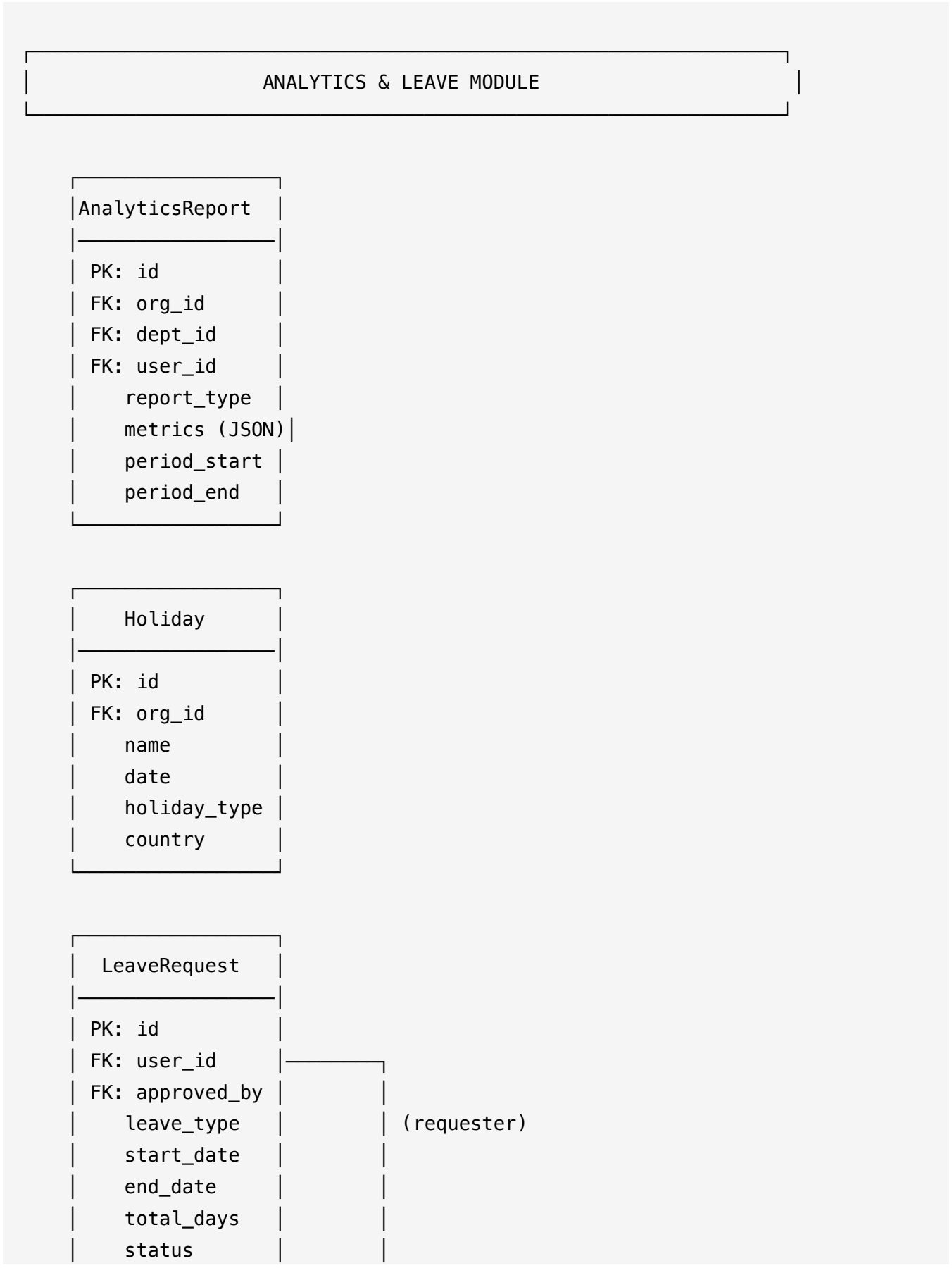

ORGANISATION MODULE
Organisation
PK: id
name
email
logo
color_palette
1
1:N
1:N
Department
PK: id
FK: org_id
name
User
PK: id
FK: org_id
FK: dept_id
email
password_hash
name
M:N
M:N
Role
PK: id
name
permissions

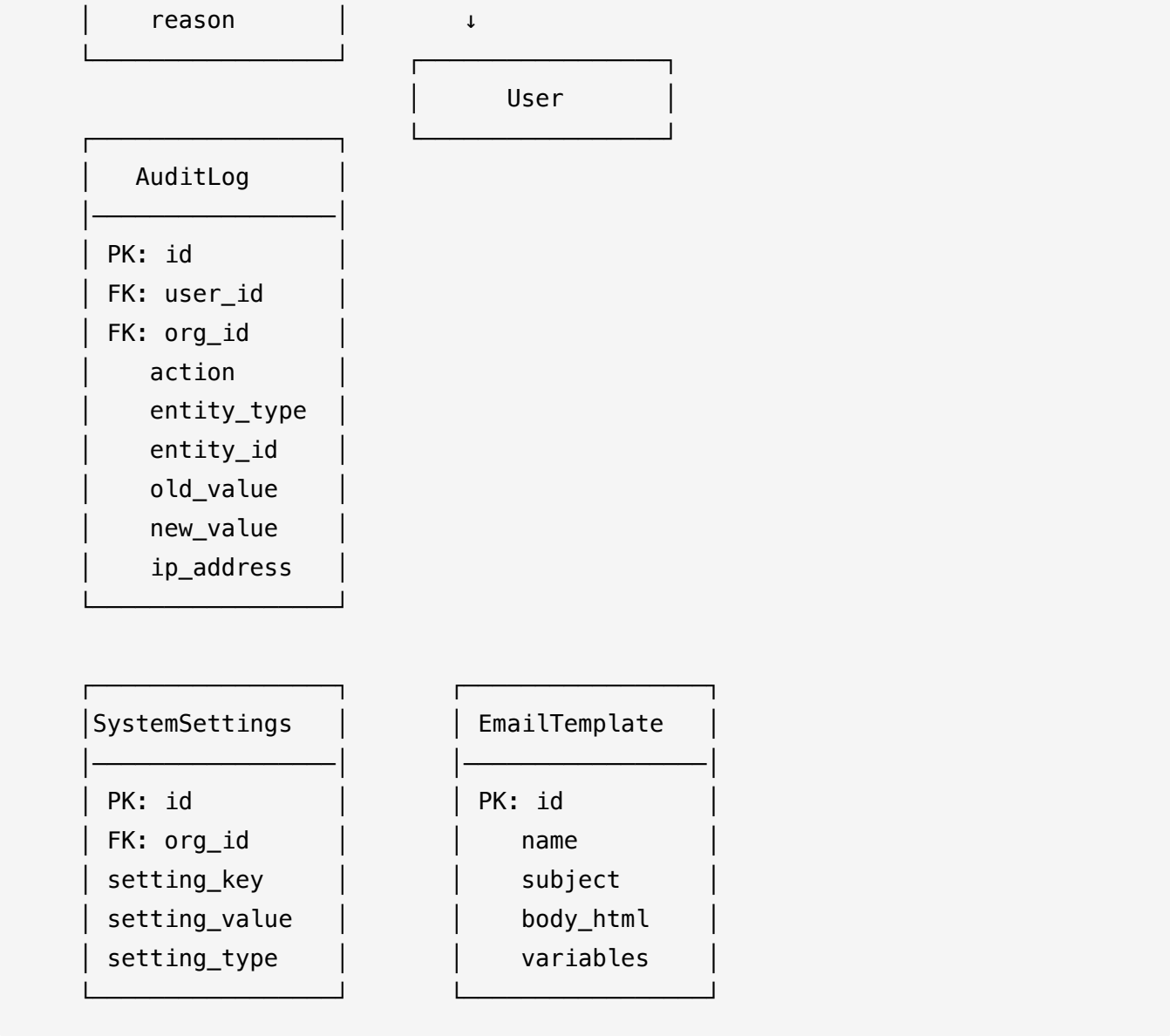

M:N
Tag
PK: id
name
color
TASK MANAGEMENT MODULE
Task
PK: id
FK: creator_id
FK: dept_id
title
description
status
priority
due_date
board_column
board_position
deliverables
1:N
TaskComment
PK: id
FK: task_id
FK: user_id
content
1:N
1:N
TaskAttachment
TimeLog


PK: id
FK: task_id
filename
file_path
PK: id
FK: task_id
FK: user_id
start_time
end_time
duration
M:N (assignees)
User
MESSAGING MODULE
Message
PK: id
FK: sender_id
FK: recipient_id
FK: channel_id
content
is_read
read_at
is_delivered
attachment_path
(Direct Message)
(Group Message)
User


ChatChannel
PK: id
FK: dept_id
FK: org_id
name
channel_type
M:N (members)
User
Notification
PK: id
FK: user_id
FK: task_id
FK: message_id
title
message
is_read
action_url
OnlineStatus
PK: id
FK: user_id
is_online
last_seen
socket_id
TypingIndicator
PK: id
FK: user_id
FK: channel_id
FK: recipient_id
is_typing


MEETING MODULE
Meeting
PK: id
FK: organizer_id
FK: dept_id
FK: task_id
title
description
start_time
end_time
meeting_link
status
1:N
MeetingAgenda
PK: id
FK: meet_id
title
order
1:N
1:N
MeetingNote
PK: id
FK: meet_id
content
note_type
MeetingAttach
PK: id
FK: meet_id
filename
file_path
M:N (attendees)
User


ANALYTICS & LEAVE MODULE
AnalyticsReport
PK: id
FK: org_id
FK: dept_id
FK: user_id
report_type
metrics (JSON)
period_start
period_end
Holiday
PK: id
FK: org_id
name
date
holiday_type
country
LeaveRequest
PK: id
FK: user_id
FK: approved_by
leave_type
start_date
end_date
total_days
status
(requester)



📊 Core Models & Relationships

1. User Management Models

Organisation

- **Purpose**: Multi-tenant container for all organizational data
- **Key Fields**: `name`, `email`, `logo`, `color_palette`, `theme`
- **Relationships**:

    - Has many Users (1:N)
    - Has many Departments (1:N)
    - Has many Settings, Holidays, Analytics

User

- **Purpose**: Core user entity with authentication
- **Key Fields**: `email`, `password_hash`, `name`, `designation`, `profile_picture`
- **Special Features**:
    - Password hashing with Werkzeug
    - Email verification token generation
    - Birthday tracking and age calculation
    - Leave quota management
    - Role-based permissions
- **Relationships**:
    - Belongs to Organisation (N:1)
    - Belongs to Department (N:1)
    - Has many Roles (M:N via `user_roles`)
    - Has many Tags (M:N via `user_tags`)
    - Creates many Tasks (1:N as creator)
    - Assigned to many Tasks (M:N via `task_assignees`)

Department

- **Purpose**: Organizational unit for team grouping
- **Key Fields**: `name`, `description`, `organisation_id`
- **Relationships**:
    - Belongs to Organisation (N:1)
    - Has many Users (1:N)
    - Has many Tasks (1:N)
    - Has one ChatChannel (1:1)

Role

- **Purpose**: RBAC role definition
- **Key Fields**: `name`, `description`, `permissions` (JSON)
- **Built-in Roles**:

```
- Admin      # Full system access
- Manager    # Team & task management
- Team Lead  # Task management only
- Employee   # Regular access
- Intern     # Limited access
```

- **Relationships**: Has many Users (M:N via `user_roles`)

Tag

- **Purpose**: Flexible categorization system
- **Key Fields**: `name`, `color`
- **Use Cases**: Role tags, skill tags, project tags
- **Relationships**:
  - Applies to Users (M:N via `user_tags`)
  - Applies to Tasks (M:N via `task_tags`)

---

2. Task Management Models

Task

- **Purpose**: Core task/work item entity
- **Key Fields**:

```
title           # Task name
description     # Detailed description
status          # todo, in_progress, done, archived
priority        # low, medium, high, urgent
due_date        # Deadline
board_column    # Kanban column (todo, in_progress, in_review, done)
board_position  # Position within column
deliverables    # JSON array of deliverable items
```

- **Kanban Features**:
  - Drag-and-drop positioning
  - Column-based organization

    - Real-time updates via Socket.IO
- **Relationships**:
    - Created by User (N:1)
    - Assigned to many Users (M:N via `task_assignees`)
    - Belongs to Department (N:1)
    - Has many Comments (1:N)
    - Has many Attachments (1:N)
    - Has many TimeLogs (1:N)
    - Has many Tags (M:N via `task_tags`)

TaskComment

- **Purpose**: Threaded comments on tasks
- **Key Fields**: `content`, `task_id`, `user_id`, `parent_id`
- **Features**:
    - Nested replies (self-referencing)
    - Edit tracking (`is_edited`)
    - Timestamp tracking

TaskAttachment

- **Purpose**: File uploads for tasks
- **Key Fields**: `filename`, `file_path`, `file_size`, `mime_type`
- **Supported Types**: Images, PDFs, documents, spreadsheets

TimeLog

- **Purpose**: Time tracking for tasks
- **Key Fields**: `start_time`, `end_time`, `duration`, `notes`
- **Features**:
    - Automatic duration calculation
    - User-specific tracking
    - Task productivity metrics

TaskHistory

- **Purpose**: Audit trail for task changes
- **Key Fields**: `action`, `field_changed`, `old_value`, `new_value`

- **Tracked Actions**: created, updated, status_changed, assigned, reassigned

---

3. Messaging Models

Message

- **Purpose**: Direct and group messaging
- **Key Fields**:

```
content          # Message text
sender_id        # Who sent it
recipient_id     # Direct message recipient (nullable)
channel_id       # Group channel (nullable)
message_type     # text, image, file, task_card
attachment_path  # File attachment
is_delivered     # Single tick ✓
is_read          # Double tick ✓✓
```

- **Message Types**:
  - **Text**: Regular chat message
  - **Image**: Image attachment
  - **File**: Document attachment
  - **Task Card**: Embedded task reference
- **Read Receipts**:
  - `is_delivered` + `delivered_at` = Single tick
  - `is_read` + `read_at` = Double tick

ChatChannel

- **Purpose**: Group chat rooms
- **Types**:
  - **Department**: Auto-created for each department
  - **Project**: Project-specific channels
  - **Custom**: User-created channels
- **Key Fields**: `name`, `description`, `channel_type`, `is_private`
- **Members**: M:N relationship via `channel_members` table

Notification

- **Purpose**: In-app notification system
- **Key Fields**: `title`, `message`, `notification_type`, `action_url`
- **Notification Types**:
  - `task_assigned` - New task assigned
  - `task_updated` - Task changed
  - `deadline_approaching` - Due date reminder
  - `message_received` - New message
  - `meeting_scheduled` - New meeting
  - `leave_approved` - Leave request approved

OnlineStatus

- **Purpose**: Real-time user presence
- **Key Fields**: `user_id`, `is_online`, `last_seen`, `socket_id`
- **Features**: Socket.IO session tracking

TypingIndicator

- **Purpose**: Live typing indicators
- **Key Fields**: `user_id`, `channel_id`, `recipient_id`, `is_typing`
- **Features**: Real-time "User is typing..." display

---

4. Meeting Models

Meeting

- **Purpose**: Schedule and manage meetings
- **Key Fields**:

```
title           # Meeting name
description     # Meeting purpose
start_time      # Start datetime
end_time        # End datetime
meeting_link    # Google Meet/Zoom link
status          # scheduled, in_progress, completed, cancelled
meeting_type    # general, standup, review, planning
```

- **Features**:
  - Multiple attendees (M:N)
  - Agenda items
  - Meeting notes
  - File attachments
  - Recurrence support
- **Attendee Statuses**: pending, accepted, declined, tentative

MeetingAgendaItem

- **Purpose**: Structured meeting agenda
- **Key Fields**: `title`, `description`, `duration_minutes`, `order`
- **Features**: Ordered list, completion tracking

MeetingNote

- **Purpose**: Meeting minutes and notes
- **Key Fields**: `content`, `note_type`
- **Note Types**: general, action_item, decision, follow_up

MeetingAttachment

- **Purpose**: Meeting documents (presentations, reports)
- **Key Fields**: `filename`, `file_path`, `file_size`

---

5. Analytics & Leave Models

AnalyticsReport

- **Purpose**: Pre-computed analytics and metrics
- **Key Fields**:

```
report_type       # weekly, monthly, department, user
metrics           # JSON with all metrics
tasks_created     # Count
tasks_completed   # Count
completion_rate   # Percentage
average_task_time # Hours
```

- **Report Types**:
  - Weekly team summary
  - Monthly organization report
  - Department performance
  - Individual user productivity

Holiday

- **Purpose**: Holiday calendar
- **Key Fields**: `name`, `date`, `holiday_type`, `country`
- **Holiday Types**: public, optional, organisation-specific
- **Pre-seeded**: 11 Indian holidays (Diwali, Holi, Republic Day, etc.)

LeaveRequest

- **Purpose**: Employee leave management
- **Key Fields**:

```
leave_type       # sick, casual, vacation, emergency
start_date       # Leave start
end_date         # Leave end
total_days       # Calculated days
status           # pending, approved, rejected
reason           # Leave reason
```

- **Workflow**:
    i. Employee submits request
    ii. Manager reviews
    iii. Manager approves/rejects with notes
    iv. System checks quota limits

AuditLog

- **Purpose**: Complete audit trail
- **Key Fields**: `action`, `entity_type`, `entity_id`, `old_value`, `new_value`
- **Tracked Entities**: All models (User, Task, Organisation, etc.)
- **Use Cases**: Security, compliance, debugging

SystemSettings

- **Purpose**: Configurable system parameters
- **Key Fields**: `setting_key`, `setting_value`, `setting_type`
- **Setting Types**: string, int, bool, json
- **Examples**: `default_task_status`, `max_file_upload_size`

EmailTemplate

- **Purpose**: Reusable email templates
- **Key Fields**: `name`, `subject`, `body_html`, `variables` (JSON)
- **Built-in Templates**:
    - Welcome email
    - Task assignment
    - Meeting invitation
    - Leave approval
    - Password reset

🚀 Setup & Installation

Prerequisites

```
# Required
✓ Python 3.9 - 3.13 (3.11 or 3.12 recommended)
✓ pip (latest version)
✓ Git

# Optional
✓ PostgreSQL (for production)
✓ Redis (for Socket.IO scaling)
```

Quick Start (5 Minutes)

Step 1: Clone Repository

```
git clone https://github.com/realshubhamraut/FlowDeck.git
cd FlowDeck
```

Step 2: Run Setup Script

macOS/Linux:

```
chmod +x setup.sh
./setup.sh
```

Windows:

```
setup.bat
```

The setup script will:

1. ✅ Check Python version
2. ✅ Create virtual environment
3. ✅ Install all dependencies

4. ✅ Create `.env` file
5. ✅ Initialize database
6. ✅ Seed demo data

Step 3: Configure Environment

Edit `.env` file with your settings:

```
# Flask Configuration
FLASK_ENV=development
FLASK_PORT=5010
SECRET_KEY=your-secret-key-here-change-in-production

# Database
DATABASE_URL=sqlite:///flowdeck.db

# Email (Gmail)
MAIL_SERVER=smtp.gmail.com
MAIL_PORT=587
MAIL_USE_TLS=True
MAIL_USERNAME=your-email@gmail.com
MAIL_PASSWORD=your-app-password
MAIL_DEFAULT_SENDER=noreply@flowdeck.org

# Optional: OpenAI for AI features
OPENAI_API_KEY=your-openai-api-key

# Optional: Google Calendar
GOOGLE_CALENDAR_API_KEY=your-google-calendar-api-key
```

Step 4: Start Application

```
# Activate virtual environment
source venv/bin/activate  # macOS/Linux
venv\Scripts\activate     # Windows

# Run application
python run.py
```

Server starts at: **http://127.0.0.1:5010**

Step 5: Login

Default Admin Credentials:

```
Email: rajesh.kumar@techvista.in
Password: admin123
```

Test Users (All password: password123):

```
CEO:       rajesh.kumar@techvista.in
CTO:       priya.sharma@techvista.in
Manager:   amit.patel@techvista.in
Developer: kavya.reddy@techvista.in
Designer:  arjun.singh@techvista.in
```

Manual Setup

If automated setup fails:

```
# 1. Create virtual environment
python3 -m venv venv
source venv/bin/activate

# 2. Install dependencies
pip install --upgrade pip
pip install -r requirements.txt

# 3. Initialize database
python run.py init-db

# 4. Seed with demo data
python seed_indian_data.py

# 5. Run application
python run.py
```

Database Seeding

Option 1: Indian Data (Recommended)

```
# Seeds 26 Indian employees, 134 tasks, 195 messages
python seed_indian_data.py
```

Option 2: Minimal Seed

```
# Basic seed with 1 admin user
python run.py seed
```

Option 3: Fresh Start

```
# Complete database reset and re-seed
python init_complete_database.py
```

Verify Installation

```
# Check routes
curl http://127.0.0.1:5010/__routes

# Check health
curl http://127.0.0.1:5010/__ping

# View database
sqlite3 instance/flowdeck.db
```

📁 Project Structure

```
FlowDeck/
│
├── app/                          # Main application package
│   ├── __init__.py               # App factory, extensions init
│   │
│   ├── models/                   # Database models (31 tables)
│   │   ├── __init__.py
│   │   ├── user.py               # User, Organisation, Department, Role, Tag
│   │   ├── task.py               # Task, Comments, Attachments, TimeLog, History
│   │   ├── messaging.py          # Message, ChatChannel, Notification
│   │   ├── meeting.py            # Meeting, Agenda, Notes, Attachments
│   │   └── analytics.py          # Analytics, Holidays, Leave, Audit
│   │
│   ├── routes/                   # Blueprint routes (62 routes)
│   │   ├── __init__.py
│   │   ├── auth.py               # Login, register, logout (7 routes)
│   │   ├── main.py               # Landing pages (5 routes)
│   │   ├── admin.py              # Admin panel (12 routes)
│   │   ├── user.py               # User profile (6 routes)
│   │   ├── tasks.py              # Task management (10 routes)
│   │   ├── chat.py               # Messaging (6 routes)
│   │   ├── dashboard.py          # Dashboard (6 routes)
│   │   ├── meetings.py           # Meetings (8 routes)
│   │   ├── api.py                # REST API (9 endpoints)
│   │   └── favicon.py            # Favicon handler (1 route)
│   │
│   ├── sockets/                  # Socket.IO event handlers
│   │   ├── __init__.py
│   │   ├── chat_events.py        # Chat events (9 events)
│   │   └── notification_events.py # Notification events (3 events)
│   │
│   ├── database/                 # Database utilities
│   │   └── __init__.py           # Triggers, views, indexes
│   │
│   ├── utils/                    # Utility modules
│   │   ├── ai.py                 # AI integration (OpenAI)
│   │   ├── email.py              # Email templates & sending
│   │   ├── validators.py         # Custom validators
│   │   ├── quotes.py             # Motivational quotes
```

```
│   │   └── seed.py               # Database seeding
│   │
│   ├── templates/                # Jinja2 templates (40+ files)
│   │   ├── base.html             # Base template
│   │   ├── main/                 # Landing pages
│   │   │   ├── landing.html
│   │   │   ├── about.html
│   │   │   └── contact.html
│   │   ├── auth/                 # Authentication
│   │   │   ├── login.html
│   │   │   ├── register.html
│   │   │   └── forgot_password.html
│   │   ├── admin/                # Admin panel (12 templates)
│   │   │   ├── dashboard.html
│   │   │   ├── users.html
│   │   │   ├── departments.html
│   │   │   ├── analytics.html
│   │   │   └── ...
│   │   ├── dashboard/            # User dashboard
│   │   │   ├── index.html
│   │   │   ├── analytics.html
│   │   │   ├── calendar.html
│   │   │   └── notifications.html
│   │   ├── tasks/                # Task management
│   │   │   ├── kanban.html       # Kanban board
│   │   │   ├── list.html         # List view
│   │   │   ├── view.html         # Task details
│   │   │   ├── create.html
│   │   │   └── edit.html
│   │   ├── chat/                 # Messaging
│   │   │   ├── index.html        # Channel list
│   │   │   └── direct.html       # Direct messages
│   │   ├── meetings/             # Meeting management
│   │   │   ├── list.html
│   │   │   ├── view.html
│   │   │   ├── create.html
│   │   │   └── edit.html
│   │   ├── user/                 # User profile
│   │   │   ├── profile.html
│   │   │   ├── edit_profile.html
```

```
│   │   │   └── settings.html
│   │   └── errors/               # Error pages
│   │       ├── 403.html
│   │       ├── 404.html
│   │       └── 500.html
│   │
│   └── static/                   # Static assets
│       ├── css/
│       │   └── main.css          # Custom styles
│       ├── images/               # Images
│       └── uploads/              # User uploads
│           ├── profiles/         # Profile pictures
│           ├── attachments/      # Task attachments
│           ├── chat/             # Chat files
│           ├── meetings/         # Meeting files
│           └── logos/            # Organization logos
│
├── instance/                     # Instance-specific files
│   └── flowdeck.db               # SQLite database (auto-generated)
│
├── venv/                         # Virtual environment (created by setup)
│
├── run.py                        # Application entry point
├── seed_indian_data.py           # Comprehensive seed script
├── init_complete_database.py     # Database initialization
│
├── requirements.txt              # Python dependencies
├── .env                          # Environment variables (created by setup)
├── .gitignore                    # Git ignore rules
│
├── README.md                     # Project README
├── ARCHITECTURE.md               # Architecture documentation
├── DOCUMENTATION_PART_1.md       # This file
│
├── setup.sh                      # Setup script (macOS/Linux)
└── setup.bat                     # Setup script (Windows)
```

🔐 Security Features

Authentication

- ✅ Password hashing with Werkzeug (PBKDF2)
- ✅ Session management with Flask-Login
- ✅ Email verification tokens
- ✅ Password reset functionality
- ✅ "Remember me" functionality
- ✅ Session timeout

Authorization

- ✅ Role-Based Access Control (RBAC)
- ✅ Permission checking decorators
- ✅ Route-level protection
- ✅ Resource-level access control
- ✅ Organisation data isolation

Data Protection

- ✅ CSRF protection (Flask-WTF)
- ✅ SQL injection prevention (SQLAlchemy ORM)
- ✅ XSS protection (Jinja2 auto-escaping)
- ✅ File upload validation
- ✅ Input sanitization
- ✅ Secure password generation

Audit Trail

- ✅ Complete audit log (AuditLog model)
- ✅ Task history tracking
- ✅ IP address logging
- ✅ User agent tracking
- ✅ Timestamp on all actions

🎨 UI/UX Features

Responsive Design

- ✅ Mobile-first approach
- ✅ Bootstrap 5.3 grid system
- ✅ Tablet optimization
- ✅ Desktop layout

Modern UI Components

- ✅ Kanban drag-and-drop board
- ✅ Real-time chat interface
- ✅ Calendar view
- ✅ Interactive charts
- ✅ Modal dialogs
- ✅ Toast notifications
- ✅ Loading spinners
- ✅ Progress bars

User Experience

- ✅ Instant search
- ✅ Keyboard shortcuts
- ✅ Contextual menus
- ✅ Inline editing
- ✅ Auto-save drafts
- ✅ Undo/redo support
- ✅ Dark mode ready

📈 Performance Optimizations

Database

- ✅ Indexes on frequently queried columns
- ✅ Query optimization with eager loading
- ✅ Database connection pooling
- ✅ Lazy loading where appropriate

Caching

- ✅ Static file caching
- ✅ Template caching
- ✅ Query result caching (ready for Redis)

Frontend

- ✅ Minified CSS/JS
- ✅ Image optimization
- ✅ Lazy loading images
- ✅ CDN for libraries (Bootstrap, jQuery)

Real-time

- ✅ WebSocket connection pooling
- ✅ Room-based broadcasting
- ✅ Event throttling
- ✅ Connection state management

🧪 Testing

Test Structure

```
tests/
├── test_models.py        # Model unit tests
├── test_routes.py        # Route integration tests
├── test_auth.py          # Authentication tests
├── test_api.py           # API endpoint tests
└── test_sockets.py       # Socket.IO tests
```

Run Tests

```bash
# Install test dependencies
pip install pytest pytest-cov pytest-flask

# Run all tests
pytest

# Run with coverage
pytest --cov=app tests/

# Run specific test
pytest tests/test_models.py::test_user_creation
```

📊 Metrics & Analytics

Available Metrics

- **Task Metrics**: Creation rate, completion rate, average time
- **User Metrics**: Productivity score, task count, time logged
- **Department Metrics**: Team efficiency, workload distribution
- **Organization Metrics**: Overall performance, trends

Analytics Dashboard

- Weekly summary reports
- Monthly performance reviews
- Department comparisons
- User productivity rankings
- Task distribution charts
- Time tracking summaries

🌐 Deployment

Production Checklist

```
# 1. Update environment variables
SECRET_KEY=<strong-random-key>
FLASK_ENV=production
DATABASE_URL=postgresql://user:pass@host:port/dbname

# 2. Use production database
# PostgreSQL or MySQL instead of SQLite

# 3. Use production WSGI server
gunicorn -w 4 -b 0.0.0.0:5000 run:app

# 4. Set up reverse proxy
# Nginx or Apache

# 5. Enable HTTPS
# Let's Encrypt SSL certificates

# 6. Configure firewall
# Allow only necessary ports

# 7. Set up monitoring
# Sentry for error tracking
# New Relic for performance
```

Docker Deployment

```
# Dockerfile
FROM python:3.11-slim
WORKDIR /app
COPY requirements.txt .
RUN pip install -r requirements.txt
COPY . .
CMD ["gunicorn", "-w", "4", "-b", "0.0.0.0:5000", "run:app"]
```

```yaml
# docker-compose.yml
version: '3.8'
services:
  web:
    build: .
    ports:
      - "5000:5000"
    environment:
      - DATABASE_URL=postgresql://postgres:password@db:5432/flowdeck
    depends_on:
      - db
  db:
    image: postgres:14
    volumes:
      - postgres_data:/var/lib/postgresql/data
    environment:
      - POSTGRES_PASSWORD=password
      - POSTGRES_DB=flowdeck
volumes:
  postgres_data:
```

📝 Summary - Part 1

What We've Covered

✅ **Project Overview** - Enterprise workflow management system
✅ **Architecture** - Multi-layered Flask application
✅ **Technology Stack** - Flask 3.0, SQLite, Socket.IO
✅ **Database Design** - 31 tables in 5 modules
✅ **Entity Relationships** - Comprehensive ER diagrams
✅ **Core Models** - User, Task, Message, Meeting, Analytics
✅ **Setup Instructions** - Automated and manual setup
✅ **Security Features** - Authentication, authorization, audit
✅ **Performance** - Optimizations and best practices

Generated: October 21, 2025
Version: 1.0
Author: FlowDeck Development Team
Repository: https://github.com/realshubhamraut/FlowDeck

📋 Table of Contents - Part 2

🛣️ Routes & Endpoints

Route Summary (62 Total Routes)

Blueprint	Routes	Purpose
Auth	7	Login, register, logout, password reset
Main	5	Landing page, about, contact
Admin	12	User/dept/org management, analytics
Dashboard	6	User dashboard, calendar, notifications
Tasks	10	Task CRUD, Kanban, comments, attachments
Chat	6	Direct messages, channels, file upload
Meetings	8	Meeting CRUD, attendees, agenda

Blueprint	Routes	Purpose
User	6	Profile, settings, leave requests
API	9	REST endpoints for AJAX
Favicon	1	Favicon handler

Authentication Routes (`/auth/*`)

```
GET  /auth/login              # Login page
POST /auth/login              # Process login
GET  /auth/register           # Registration page
POST /auth/register           # Process registration
GET  /auth/logout             # Logout user
GET  /auth/forgot-password    # Password reset page
POST /auth/reset-password     # Process password reset
```

Login Types:

- **User Login**: Regular employee access
- **Admin Login**: Admin with organization ID

Demo Credentials:

```
CEO:       rajesh.kumar@techvista.in / admin123
Developer: kavya.reddy@techvista.in / password123
All users: password123
```

Task Routes (`/tasks/*`)

```
GET  /tasks/                  # List view (paginated)
GET  /tasks/kanban            # Kanban board view
GET  /tasks/<id>              # Task details
GET  /tasks/create            # Create task form
POST /tasks/create            # Process task creation
GET  /tasks/<id>/edit         # Edit task form
POST /tasks/<id>/edit         # Update task
POST /tasks/<id>/delete       # Delete task
POST /tasks/<id>/comment      # Add comment
POST /tasks/<id>/update-status # Update status (Kanban drag-drop)
```

Filters:

- Status: `todo`, `in_progress`, `in_review`, `done`
- Priority: `low`, `medium`, `high`, `urgent`
- Search: Title/description
- Sort: Due date, priority, status, title

Admin Routes (`/admin/*`)

```
GET  /admin/dashboard         # Admin overview
GET  /admin/users             # User list
GET  /admin/users/create      # Create user
POST /admin/users/create      # Process user creation
GET  /admin/users/<id>/edit   # Edit user
POST /admin/users/<id>/delete # Delete user
GET  /admin/departments       # Department list
GET  /admin/analytics         # Analytics dashboard
GET  /admin/leave-requests    # Leave approvals
GET  /admin/organisation      # Org settings
```

Chat Routes (`/chat/*`)

```
GET  /chat/                    # Channel list
GET  /chat/direct/<user_id>    # Direct message view
POST /chat/send                # Send message (AJAX)
POST /chat/upload              # Upload file
GET  /chat/channels            # List all channels
POST /chat/mark-read           # Mark message as read
```

Meeting Routes (`/meetings/*`)

```
GET  /meetings/                # Meeting list
GET  /meetings/<id>            # Meeting details
GET  /meetings/create          # Create meeting form
POST /meetings/create          # Process meeting
GET  /meetings/<id>/edit       # Edit meeting
POST /meetings/<id>/edit       # Update meeting
POST /meetings/<id>/cancel     # Cancel meeting
POST /meetings/<id>/respond    # RSVP (accept/decline)
```

🔌 Socket.IO Events

Chat Events

Event	Direction	Purpose
`connect`	Client → Server	User connects, join rooms
`disconnect`	Client → Server	User disconnects, update status
`send_message`	Client → Server	Send text/file message
`new_message`	Server → Client	Broadcast new message
`new_direct_message`	Server → Client	Send DM to recipient

Event	Direction	Purpose
`message_delivered`	Client → Server	Mark as delivered (✓)
`message_delivered_ack`	Server → Client	Confirm delivery
`mark_message_read`	Client → Server	Mark as read (✓✓)
`message_read_ack`	Server → Client	Confirm read status
`typing`	Client → Server	User is typing
`user_typing`	Server → Client	Show typing indicator
`stop_typing`	Client → Server	User stopped typing
`user_online`	Server → Client	User came online
`user_offline`	Server → Client	User went offline

Notification Events

Event	Direction	Purpose
`notify_user`	Server → Client	Send in-app notification
`notification_read`	Client → Server	Mark notification as read
`leave_request_notification`	Server → Client	New leave request alert

Room Structure

```
// User joins these rooms on connect:
- "user_{user_id}"           // Personal room for DMs
- "channel_{channel_id}"      // Each chat channel
- "department_{dept_id}"      // Department broadcasts
- "organisation_{org_id}"     // Org-wide broadcasts
```

⚡ Key Features Implementation

1. Kanban Board (Drag & Drop)

Technology: HTML5 Drag & Drop API + Socket.IO

How it Works:

1. User drags task card to new column
2. JavaScript captures `drop` event
3. AJAX POST to `/tasks/<id>/update-status`
4. Server updates `board_column` and `board_position`
5. Socket.IO broadcasts update to all users
6. All connected users see real-time change

Columns:

- `todo` → To Do
- `in_progress` → In Progress
- `in_review` → In Review
- `done` → Done

2. Real-time Chat (Direct & Group)

Direct Messages:

- One-on-one conversations
- Read receipts (✓ delivered, ✓✓ read)
- File attachments (images, documents)
- Task card sharing
- Typing indicators

Group Channels:

- Department channels (auto-created)
- Project channels (manager-created)
- Member management
- Channel history

Message Types:

- `text` - Plain text message
- `image` - Image attachment
- `file` - Document attachment
- `task_card` - Embedded task reference

3. Meeting Management

Features:

- Multiple attendees (M:N relationship)
- Google Meet link generation
- Agenda items with ordering
- Meeting notes (action items, decisions)
- File attachments (presentations)
- Email invitations
- RSVP tracking (pending/accepted/declined)

Meeting Types:

- General, Standup, Review, Planning, Client, One-on-one

4. Leave Management

Workflow:

1. Employee submits leave request (start/end date, reason)
2. System checks quota availability
3. Manager receives notification (Socket.IO + Email)
4. Manager approves/rejects with notes
5. Employee notified of decision
6. Quota automatically updated

Leave Types:

- Annual Leave (quota-based)
- Sick Leave (quota-based)

- Personal Leave (quota-based)
- Emergency Leave (special approval)

Quota Tracking:

- Set by admin/manager per user
- Auto-deducted on approval
- Remaining days calculated
- Used days history

5. Time Tracking

Features:

- Start/stop timer per task
- Manual time log entry
- Duration auto-calculation
- Time log history
- Productivity reports
- Billable hours tracking

Usage:

1. User opens task
2. Clicks "Start Timer"
3. Works on task
4. Clicks "Stop Timer"
5. System calculates duration
6. Adds to task's total hours

6. Analytics & Reports

Available Reports:

- Weekly team summary
- Monthly organization report
- Department performance
- Individual productivity

- Task completion trends
- Time utilization

Metrics Tracked:

- Tasks created/completed
- Completion rate (%)
- Average task time (hours)
- Overdue tasks count
- Team velocity
- User productivity score

📊 Current Status & Progress

✅ Completed Features

Module	Feature	Status
Authentication	Login/Logout	✅ Complete
	Registration	✅ Complete
	Password Reset	✅ Complete
	Email Verification	✅ Complete
	RBAC (5 roles)	✅ Complete
User Management	Profile Management	✅ Complete
	User CRUD (Admin)	✅ Complete
	Department Assignment	✅ Complete
	Role Assignment	✅ Complete
Task Management	Task CRUD	✅ Complete
	Kanban Board	✅ Complete

Module	Feature	Status
	Drag & Drop	✅ Complete
	Comments	✅ Complete
	Attachments	✅ Complete
	Time Tracking	✅ Complete
	Task History	✅ Complete
	Multiple Assignees	✅ Complete
Messaging	Direct Messages	✅ Complete
	Group Channels	✅ Complete
	Read Receipts	✅ Complete
	File Upload	✅ Complete
	Typing Indicators	✅ Complete
	Online Status	✅ Complete
Meetings	Meeting CRUD	✅ Complete
	Attendee Management	✅ Complete
	Agenda Items	✅ Complete
	Meeting Notes	✅ Complete
	Attachments	✅ Complete
	RSVP Tracking	✅ Complete
Leave	Leave Requests	✅ Complete
	Quota Management	✅ Complete
	Approval Workflow	✅ Complete
Analytics	Reports Generation	✅ Complete

Module	Feature	Status
	Metrics Tracking	✅ Complete
Database	31 Tables	✅ Complete
	Seed Script	✅ Complete
	Indian Data	✅ Complete

🔧 Recent Fixes

October 21, 2025:

- ✅ Fixed Jinja2 `'min' is undefined` error
  - Added `min` and `max` to Jinja globals
  - Fixed task list pagination display

🚧 Pending Features (From flowdeck.txt)

Priority	Feature	Status
HIGH	Google Meet Integration	🔴 Pending
	Auto-send meeting emails	🔴 Pending
	Task assignment notifications	🔴 Pending
	AI Task Assignment	🔴 Pending
	Project Creation (with AI)	🔴 Pending
MEDIUM	Assignee Search	🔴 Pending
	Email Verification UI	🔴 Pending
	Footer Fix (Chat pages)	🔴 Pending
	Avatar Display (Chat list)	🔴 Pending
	Holiday API Integration	🟡 Partial (Manual holidays)

Priority	Feature	Status
	Leave Request Chat Icon	🔴 Pending
LOW	Developer Access `/developer`	🔴 Pending
	Forgot Password Link	🔴 Pending
	Machine Learning Models	🔴 Pending

🐛 Known Issues & Fixes

Issue #1: Task List Page Error (FIXED ✅)

Error:

```
jinja2.exceptions.UndefinedError: 'min' is undefined
```

Location: `templates/tasks/list.html` line 679

Cause: Jinja2 doesn't include Python's `min` / `max` by default in newer versions

Fix Applied:

```
# app/__init__.py
app.jinja_env.globals.update(min=min, max=max)
```

Issue #2: Chat Footer Position

Status: 🔴 Open

Description: Footer doesn't stay at bottom on chat pages

Pages Affected:

- `/chat/`

- `/chat/direct/<id>`

Proposed Fix:

```
/* Add to main.css */
.chat-container {
    min-height: calc(100vh - 200px);
    display: flex;
    flex-direction: column;
}
.chat-footer {
    margin-top: auto;
}
```

Issue #3: Avatar Not Showing in Chat List

Status: 🔴 Open

Description: Profile pictures not displaying in chat user list

Page: `/chat/`

Proposed Fix:

```
<!-- templates/chat/index.html -->
<img src="{{ url_for('static', filename='uploads/profiles/' + user.profile_picture) }}"
     onerror="this.src='{{ url_for('static', filename='images/default-avatar.png') }}'"
     class="rounded-circle" width="40" height="40">
```

Issue #4: Email Verification Button

Status: 🔴 Open

Description: No visible prompt for email verification

Page: `/user/profile`

Proposed Fix:

```
{% if not current_user.is_email_verified %}
<div class="alert alert-warning">
    <i class="fas fa-exclamation-triangle"></i>
    Email not verified.
    <a href="{{ url_for('auth.resend_verification') }}" class="btn btn-sm btn-warning">
        Verify Email
    </a>
</div>
{% endif %}
```

🚀 Future Enhancements

Phase 1: Immediate (Next Sprint)

1. **Google Meet Auto-Create**
   - API integration with Google Calendar
   - Auto-generate meeting links
   - Sync with user's Google Calendar
2. **Email Notifications**
   - Meeting invitations
   - Task assignments
   - Leave request updates
   - Weekly summary digest
3. **In-Chat Leave Request**
   - Add icon next to image upload
   - Modal form to submit leave
   - Auto-notify manager in chat

Phase 2: Short-term (1-2 months)

4. **AI Task Assignment**
   - OpenAI API integration
   - Parse requirements document
   - Suggest assignees based on skills

    - Auto-generate subtasks
5. **Project Module**
    - Project CRUD
    - Link tasks to projects
    - Project timeline view
    - Milestone tracking
6. **Advanced Search**
    - Elasticsearch integration
    - Full-text search across all entities
    - Filters and facets
    - Search suggestions

Phase 3: Long-term (3-6 months)

7. **Developer Portal** (`/developer`)
    - Organization subscription management
    - Billing and invoicing
    - Usage analytics
    - API key management
8. **Machine Learning**
    - Task priority prediction
    - Time estimation
    - Workload balancing
    - Anomaly detection (delayed tasks)
9. **Mobile App**
    - React Native or Flutter
    - Push notifications
    - Offline support
    - Camera integration
10. **Integrations**
    - Slack integration
    - Jira import
    - GitHub commits
    - Zapier webhooks

📦 Database Seed Summary

Current Seed Data (`seed_indian_data.py`)

Organizations: 1

- TechVista Solutions Pvt Ltd

Departments: 8

- Engineering, Design, QA, Product, Marketing, Sales, HR, Operations

Users: 26 (Indian names)

- 1 CEO, 1 CTO, 5 Managers, 16 Employees, 3 Interns

Tasks: 134 (4-6 per person)

- Properly distributed across Kanban columns
- Role-specific task templates
- Engineering: "Implement X API", "Fix Y bug"
- Design: "Create X mockups", "Design Y screens"
- QA: "Test X feature", "Write Y test cases"

Messages: 195

- 64 Direct messages (16 conversations)
- 131 Channel messages (9 channels)
- Hindi/English mix (Hinglish)

Meetings: 10

- Sprint planning, standups, reviews
- Multiple attendees, agenda items

Holidays: 11 (Indian)

- Diwali, Holi, Republic Day, Independence Day, etc.

Leave Requests: 37

- Various statuses: pending, approved, rejected

Full Database Summary:

```
Organizations:        1
Departments:          8
Users:               26
Roles:                5
Tags:                12
Tasks:              134
Task Comments:       89
Task Attachments:    23
Task History:       356
Time Logs:          273
Meetings:            10
Meeting Attendees:   47
Agenda Items:        18
Meeting Notes:       12
Messages:           195
Chat Channels:        9
Notifications:      156
Holidays:            11
Leave Requests:      37
Audit Logs:         412
System Settings:      6
Email Templates:      5
```

🔒 Security Checklist

- ✅ Password hashing (PBKDF2)
- ✅ CSRF protection
- ✅ SQL injection prevention (ORM)
- ✅ XSS protection (auto-escaping)
- ✅ Session management
- ✅ RBAC authorization
- ✅ File upload validation

- ✅ Audit logging
- ⚠️ Rate limiting (Recommended for production)
- ⚠️ Two-factor authentication (Future enhancement)
- ⚠️ API key authentication (Future enhancement)

🎯 Quick Reference

Start Development Server

```
source venv/bin/activate
python run.py
# Server: http://127.0.0.1:5010
```

Re-seed Database

```
python seed_indian_data.py
```

Create Admin User

```
python run.py create-admin
```

Database Console

```
sqlite3 instance/flowdeck.db
```

View All Routes

```
curl http://127.0.0.1:5010/__routes
```

Debug Endpoints

```
GET /__ping       # Health check
GET /__routes     # List all routes
GET /__whoami     # App info
```

📚 Additional Resources

Documentation Files

- `README.md` - Project overview and setup
- `ARCHITECTURE.md` - Detailed architecture
- `DOCUMENTATION_PART_1.md` - Database, models, architecture
- `DOCUMENTATION_PART_2.md` - This file (routes, features, status)
- `flowdeck.txt` - Feature requests and bugs

Code Organization

- `app/models/` - Database models (5 files)
- `app/routes/` - Route blueprints (11 files)
- `app/sockets/` - Socket.IO events (2 files)
- `app/templates/` - Jinja2 templates (40+ files)
- `app/utils/` - Utilities (5 files)

Important Files

- `run.py` - Application entry point
- `requirements.txt` - Python dependencies
- `.env` - Environment configuration
- `seed_indian_data.py` - Data seeding script

🎓 Learning Resources

Flask Best Practices

- Use blueprints for modular code ✅
- Application factory pattern ✅
- Environment variables for config ✅
- Database migrations (Flask-Migrate) ✅

Real-time Communication

- Socket.IO room-based messaging ✅
- Connection state management ✅
- Event throttling for performance ✅
- Graceful fallbacks ✅

Database Design

- Normalize to 3NF ✅
- Use foreign key constraints ✅
- Index frequently queried columns ✅
- Cascade deletes where appropriate ✅

📞 Support & Contributions

Repository: https://github.com/realshubhamraut/FlowDeck

Contact: contactshubhamraut@gmail.com

License: MIT (if applicable)

Generated: October 21, 2025
Version: 1.0

Status: Production-Ready (with pending enhancements)

← Back to Part 1