Data model (ER / SQL)

Entities

- `users` (1 row per user)
- `tasks` (each task assigned to exactly one user)

PostgreSQL schema (concise)

```
-- USERS
CREATE TABLE users (
  user_id        BIGSERIAL PRIMARY KEY,
  name           TEXT NOT NULL,
  email          CITEXT NOT NULL UNIQUE,
  registration_date TIMESTAMP WITH TIME ZONE NOT NULL DEFAULT now()
);

-- TASKS
CREATE TABLE tasks (
  task_id        BIGSERIAL PRIMARY KEY,
  title          TEXT NOT NULL,
  description    TEXT,
  assigned_user_id BIGINT NOT NULL REFERENCES users(user_id) ON DELETE
CASCADE,
  status         SMALLINT NOT NULL, -- 0: pending, 1: in_progress, 2:
completed
  priority       SMALLINT NOT NULL, -- 0: low, 1: medium, 2: high
  due_date       DATE,
  created_at     TIMESTAMP WITH TIME ZONE NOT NULL DEFAULT now(),
  updated_at     TIMESTAMP WITH TIME ZONE NOT NULL DEFAULT now(),
  version        BIGINT NOT NULL DEFAULT 1 -- optimistic locking
);

-- Helpful enum-like mapping via CHECK or separate lookup table can be
added,
-- but using SMALLINT is compact and query-friendly.
```

Important indexes

```
-- filter indexes
CREATE INDEX idx_tasks_status ON tasks(status);
CREATE INDEX idx_tasks_priority ON tasks(priority);
CREATE INDEX idx_tasks_due_date ON tasks(due_date);

-- common composite index for typical queries: filter by user + status +
priority
CREATE INDEX idx_tasks_user_status_priority ON tasks(assigned_user_id,
status, priority, due_date);

-- created_at index for leaderboard/time-range analytics
CREATE INDEX idx_tasks_created_at ON tasks(created_at);

-- optional: partial index for not-completed (speed up active tasks)
CREATE INDEX idx_tasks_not_completed ON tasks(assigned_user_id) WHERE
status <> 2;
```

API design (REST-style; concise)

- `POST /api/register` — register (name, email) → creates user.
- `POST /api/login` — returns token (or use OAuth).
- `POST /api/tasks` — create task (title, description, assigned_user_id, priority, due_date). Returns created task.
- `PUT /api/tasks/{task_id}` — update task (status, title, assigned_user_id, ...). Use optimistic locking (client sends `version`).
- `GET /api/tasks` — list/filter tasks. Query params: `status`, `priority`, `due_date_from`, `due_date_to`, `assigned_user_id`, `limit`, `offset`, `sort_by`.
- `GET /api/users/{id}/tasks` — tasks for a user (paged).
- `GET /api/leaderboard` — top users by completed tasks (params: `since`, `limit`).
- `GET /api/analytics/summary` — precomputed stats (counts per status/priority, tasks created per day).

Security: authenticate via JWT or session; enforce RBAC if needed.