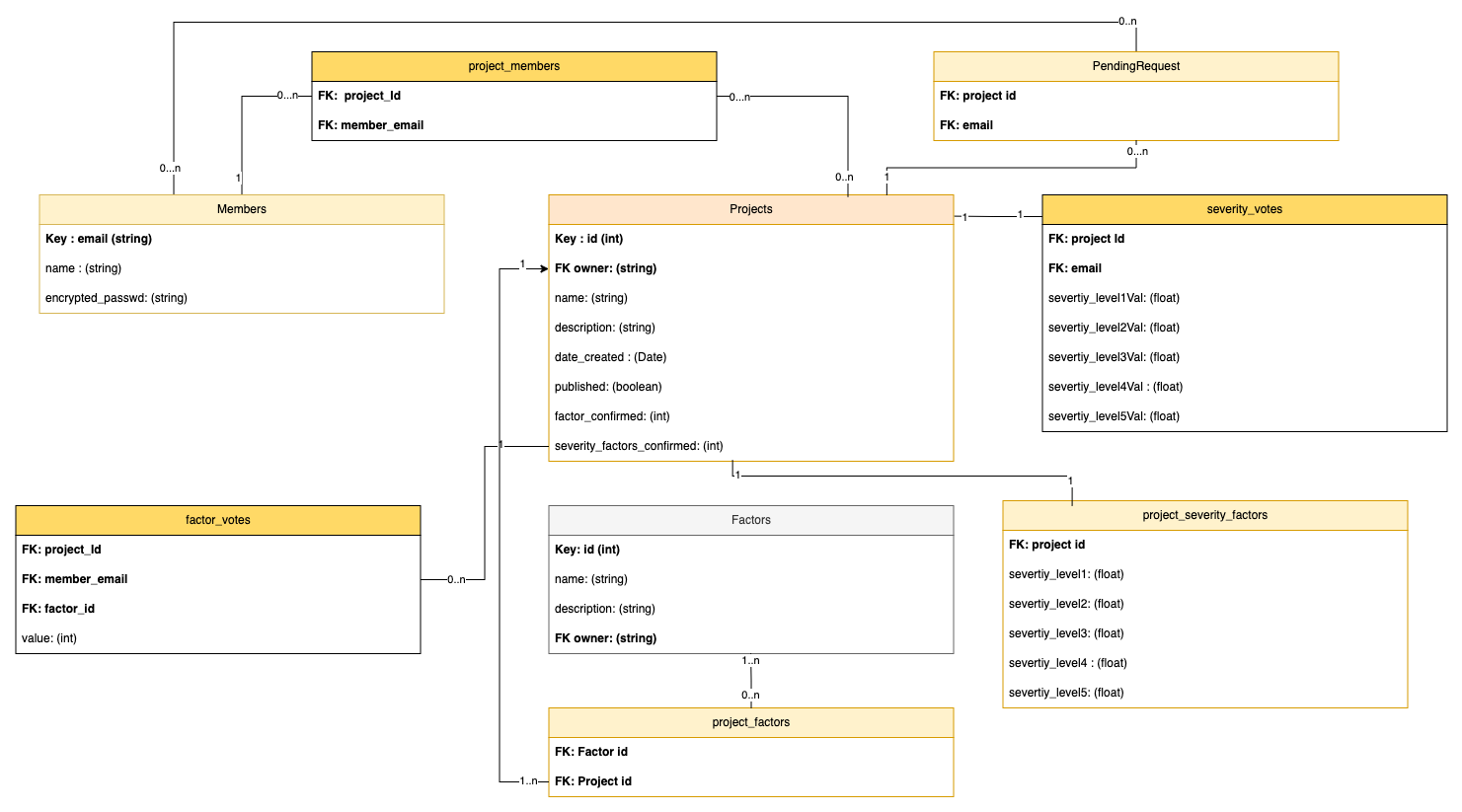
# Chapter 2

# Chapter 3

## Data Model



 **Members**: Stores details of all registered members, such as their email (which serves as the primary key), name and encrypted password.

 **Projects**: Contains information about projects, including their name, description, owner (a reference to a member), creation date, publication status, and factor confirmation counts. Acts as a central hub connecting various project-related features.

 **Project\_members**: A junction table that manages the many-to-many relationship between projects and members, tracking which users are associated with which projects.

 **Pending\_request:** Tracks pending membership requests for projects & members to join specific projects, awaiting approval.

 **Factors**: predefined factors that can be assigned to projects, including basic information like name, description, and owner.

 **Project\_factors**: A junction table connecting projects with their assigned factors, representing many-to-many associations between them.

 **Factor\_votes**: Records individual member votes on specific factors within projects.

 **Project\_severity\_factors**: Stores severity-related factors specific to each project in a one-to-one relationship.

 **Severity\_votes**: Records votes from members about the severity level of specific projects.

\*(Core Entities)

Data object main realtionships:

1. Members to Projects (Many-to-Many)

* Implemented through ProjectMembers junction table
* Members can be part of multiple projects
* Projects can have multiple members

1. Members to Projects through pending\_request (Many-to-Many)

* Tracks membership requests for projects
* Members can request to join multiple projects
* Projects can have multiple pending requests

1. Projects to Factors (Many-to-Many)

* Implemented through ProjectFactors junction table
* Projects can have multiple evaluation factors
* Factors can be used in multiple projects

1. Members to Factors through factor\_votes (Many-to-Many)

* Members can vote on factors for different project
* Tracks voting values for project factors

5. Projects to severity\_votes (One-to-Many)

* Members can provide severity ratings for projects
* Projects can receive multiple severity votes

6. Projects to Project\_severity\_factors (One-to-One)

* Each project has one set of final severity factor values
* Represents the aggregated severity levels

Database Architecture Overview

This database is implemented using PostgreSQL, leveraging its advanced features and relational capabilities to provide robust data management for our project assessment platform.

## Core Implementation

* Implemented using PostgreSQL 14, taking advantage of its JSON capabilities, full-text search, and advanced indexing.
* Schema normalized to Third Normal Form (3NF) to minimize data redundancy and maintain consistency.

## Data Integrity & Security

* Enforced referential integrity through foreign key constraints.
* Data validation through CHECK constraints and triggers.
* Encrypted sensitive data.