IIIT Software Engineering Course

KARMA

TEAM 9

Akshay Kolge
Mathew John
Ramkrishna Maheta
Sanket Markan
Siddharth Bhatore

Introduction

This document is aimed to give a detailed description of the design strategy our team is going to follow to build **Karma.** Requirements and goals are also listed in the beginning. After that a system overview and a detailed overview of it is given. And a detailed architecture is provided with block diagram. Then we have launch phases where we give different phases in which we will complete the project and their completion time estimates. Then information about Scaling, security, metrics, databases, rollback strategy is given. to end it, references and appendix is also provided.

Terminology

1. **Mentor** : A person who will give tips to people following a certain goal.
2. **Personal space** : Secure space which only the user can see. Goals under personal space can be seen only by that user.
3. **Journal** : A diary where date wise progresses or activities is added.
4. **Project** : When a user has a certain goal, he can start a project in his personal space.
5. **Goal** : A user defined objective under which all the projects will be listed by different users.
6. **Progress** : For a certain day(date), things done by the user to get closer to his goal.

Motivation

The key motivations are:

1. To make our team more confident on what we are going to build and how we are going to do it.
2. For further reference in the future when the team will be implementing features and encounters any problems.

Requirements

Functional requirements:

- Start a new project for a specific goal in personal space (Must)
- Journal type setting for projects, progress addition.(Must)

User can easily write progresses in a journal type format and navigate easily through his progresses.

- Option for sharing the project under a common goal.(Should)
- Upvote a project (Should)
- Upvote a progress (Should)
- Follow a goal (Should)
- Follow a project (Should)
- Follow a User (Should)
- Comment on goal (Should)
- Comment on project (Should)

- Comment on progress (Should)
- Categorize goals (Should)
- Search user/goal (Should)
- Profile of the user (Should)

Non-Functional Requirements :

● Performance

Database design should be such that queries and changes to data should not take more than 1 second.

● Reliability

Data should not be lost.

● Availability

Site should not be down for more than 1 minute per day.

● Security

Personal space should be secure. Credentials would not be compromised.

● Maintainability

Application will be very scalable so that it can be enhanced and used beyond the scope of this course too.

● Portability

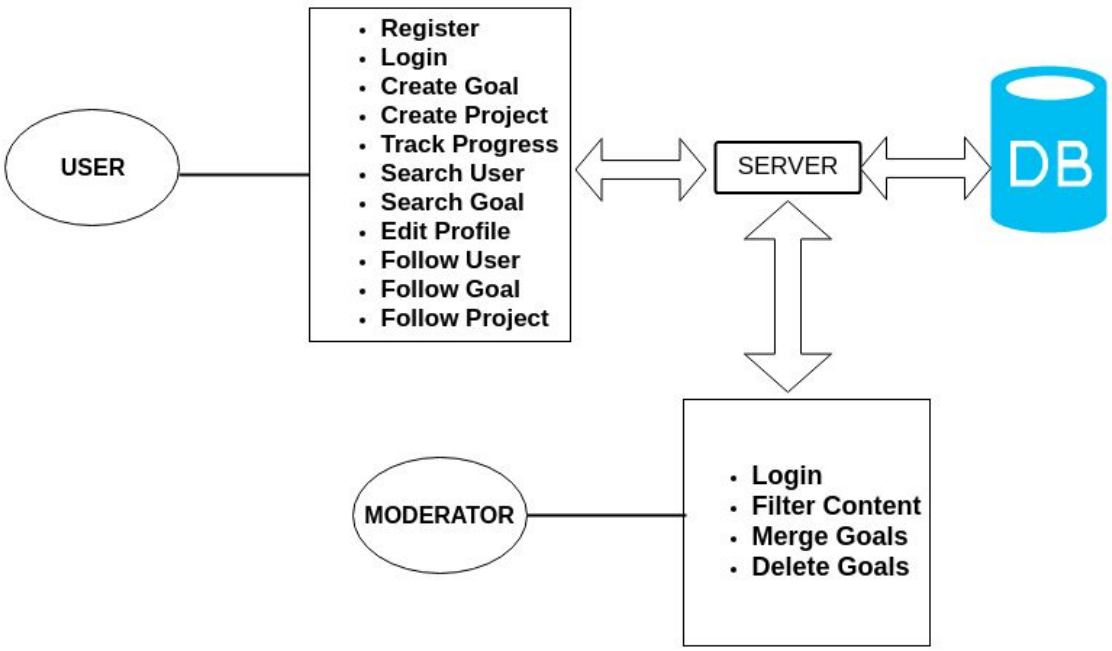
Application will be accessible through any gadget with internet connection and a browser.

Goals

1. The application should be easy to use so that minimal technical knowledge is required to make use of it.
2. We aim to make this a successful contribution the society.

Proposed Design

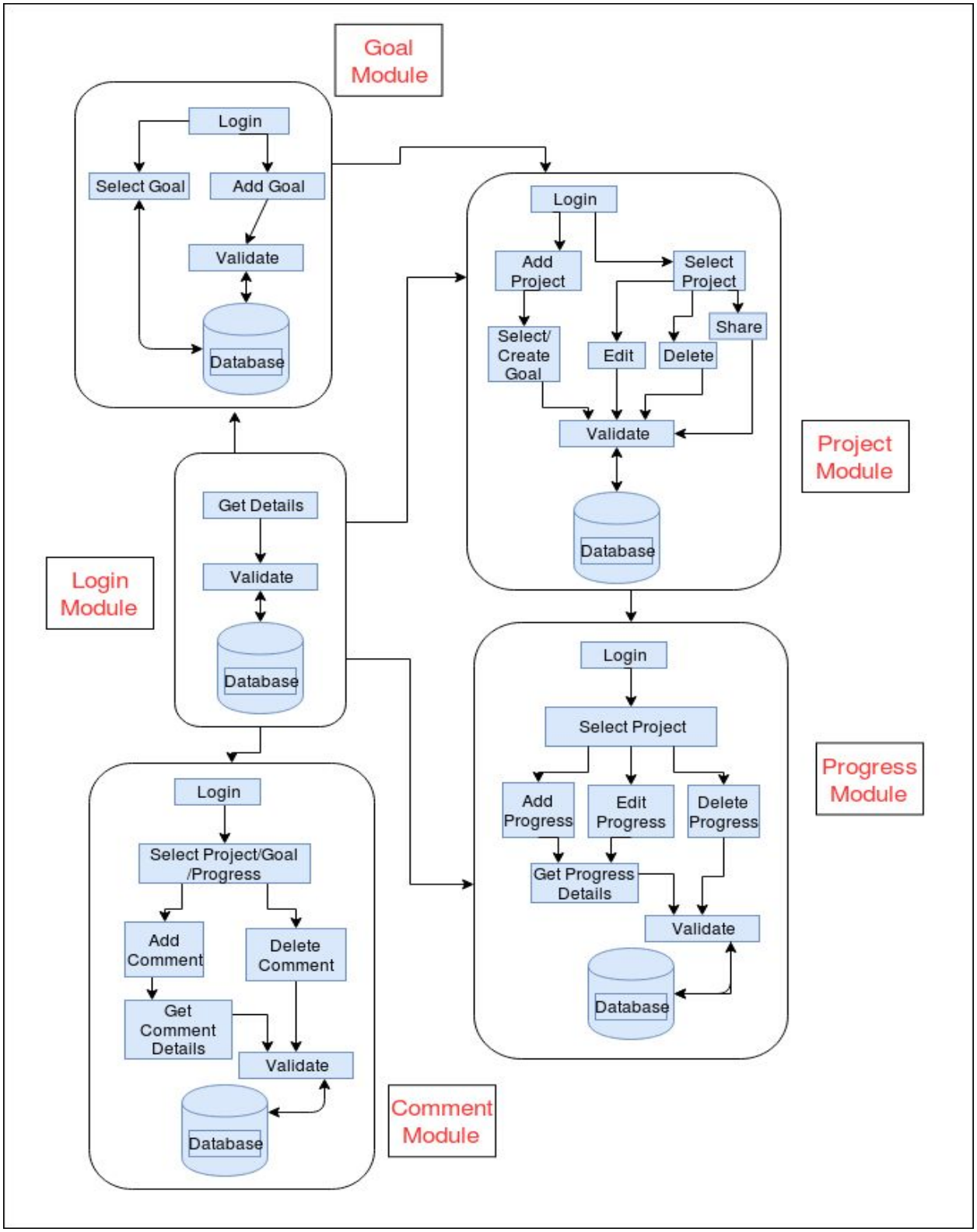
High Level Architecture



High Level Architecture Overview

1. Server has the access to the database and moderator and user access the server through their interfaces.
2. User can register and login and do all kinds of tasks which are explained in the detailed architecture.
3. Moderator can login and filter content, merge goals and delete goals. It is explained more clearly in next section.

Detailed Architecture



Detailed Architecture Overview

1. Login Module

Module required to authenticate the user. First user enter his/her details and it is validated, if it is correct then user is logged in.

2. Goal Module

Goal module has two features - view goals and add goals. In add goal after entering information it is validated and if it is correct new goal is created.

3. Project Module

Using this module, a user can add/edit/delete/share a project. In add project we have to first select goal from existing goals or create a new goal for this project, then information provided in add/edit is validated and corresponding action is complete.

4. Progress Module

Using this module, a user can add/edit/delete a progress under a project.

5. Comment Module

This module is used to add a comment under a project/progress/goal.

Major Components

Application type : Web Application

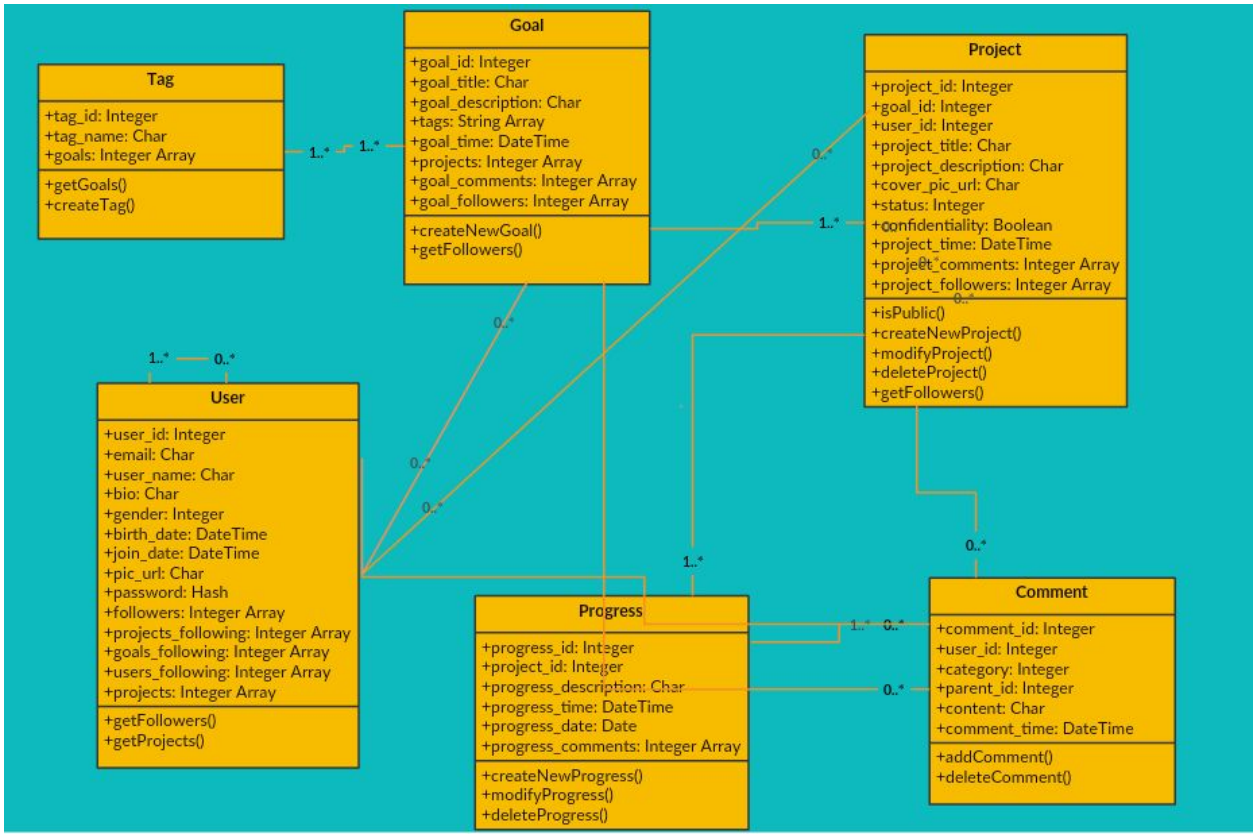
Database : postgreSQL/MySQL

Backend Server : Ruby on Rails

Front-End : AngularJS

Deployment Server : Heroku

We have decided to build a web application for this course as it aims everyone with a browser and internet access. Our team is quite familiar with Ruby on Rails and it is a very popular framework too, so we have decided to choose Ruby on Rails as a backend framework to support our application. The reason to choose AngularJS is that we tend to keep our application simple and elegant. It is simpler to deploy on heroku using postgreSQL for Ruby on Rails. We have chosen heroku to deploy because it is free.



Class Diagram for Karma

Classes:

1. **Goal :** This class would create instances of goals by the users. A goal can be created but it can't be deleted or edited. It can be deleted by the moderator. We should be able to get followers of the goal and we have attributes which represent basic information of the goal, and a projects array which contains ID of all the projects under it.

2. **Project :**When a user has a certain goal, he can start a project in his personal space.This class would create instance of project . A project can be private to user or publicly shared.
   A project has a unique project id , and it has a goal id associated with it (to denote the goal under which this project is classified). It has a project description and a cover picture(Optional). It also has a status(running/completed). All users can comment on the project and follow the project. A user can modify the project and delete it. He can also get the list of followers of a particular project.

3. **Progress :** A progress is written for a certain goal under a certain date. So it has attributes like project_id which is its parent and basic information is there. It will also have an array of comments which will be updated with every deletion and addition of a comment. It has methods to create, modify and delete a progress.

4. **User :** This class create instances of the users. A user will have basic information which can be viewed in the above diagram. It has attributes such as followers which is an array

containing IDs of all the followers of the user. Then it has other arrays such as projects_following, goals_following, users_following and projects with respective functionalities. It has methods like get_followers and get_projects with obvious functionalities.

5. **Comment :** This class has a unique comment id. It has a user id associated with it referencing the user who wrote the comment. There is a category field denoting whether the comment is under goal,project or progress. The field parent id points to the corresponding goal , project or progress. content stores the string denoting the comment and the field comment_time stores the date time of the comment.

6. **Tag :** Goals can have tags. So, a tag instance has three attributes. A unique ID, its name and an array of goals which have that tag. It has methods like getGoals() and createTag().

Data Model

Subject to change in future releases of this document.

Launch Plan

Phase 1

Details

27 Feb - 14 Mar:

- This will be the first iteration with aim to deliver the must have features of the application.
- User will be able to register and login to the site.
- User will be able to create a project under a goal and add progresses datewise. It will be confined to his personal space.
- Proper testcases for testing of the functionalities will be written.

Phase 2

Details

15 Mar - 31 Mar:

- This will be the second and final iteration aimed to deliver should have features of the project.

- Rest of the features as written above in this document will be implemented and tested.

Metrics

Following are the metrics :

1. Program response time : The time a system takes to perform a particular event and deliver the output . The response time is guaranteed to be 2 sec.
2. Request per second (RPS) : The number of requests which our application can handle on the heroku server.
3. Concurrency : The number of threads that our services could spans concurrently .

Security

Any person with registered account can access the services on the website after entering valid login credentials. The password of the user will be stored in an encrypted format using appropriate encryption scheme. Sign in of the user will also be done using appropriate signing schemes. The private data of a user will be accessible only to himself.

Scaling

Breaking Point/Known Scaling Limits

If the RPS exceeds 200 then the system might not remain reliable.

Heat Management

If a page attracts unusual number of users, then the system might become unresponsive.

Databases

Database Design - Part 1

Tag	
PK	**tag_id**
	tag_name

Goal	
PK	**goal_id**
	title
	description
	goal_time

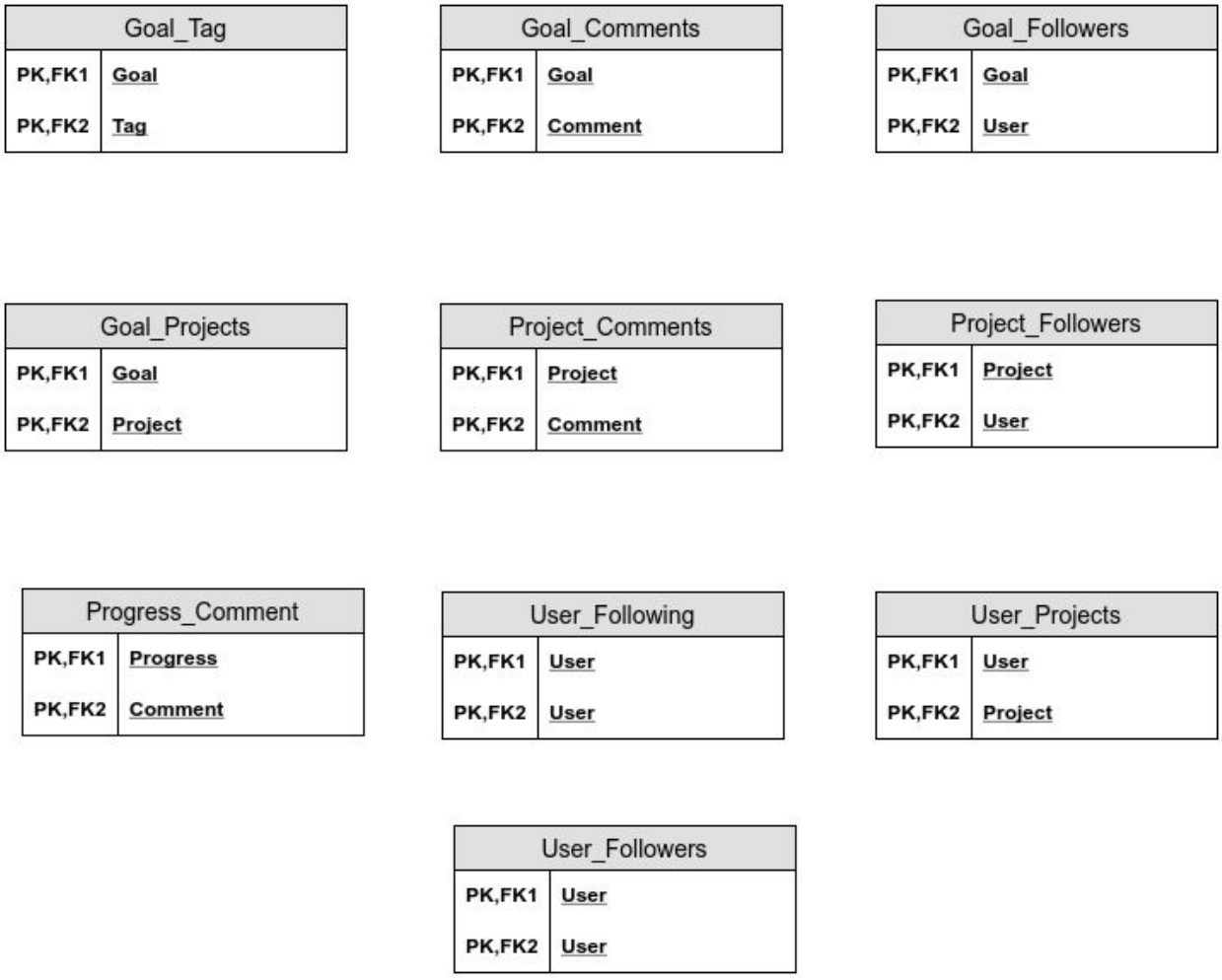
Comment	
PK	**comment_id**
FK	User
	parent_id
	content
	time
	category

Project	
PK	**project_id**
FK	Goal
FK	User
	project_title
	project_description
	pic_URL
	confidentiality
	project_time

User	
PK	**user_id**
	email
	username
	bio
	birth_date
	join_date
	profile_pic_url
	password

Progress	
PK	**progress_id**
FK	Project
	description
	date

Database Design- Part 2



Rollback Strategy

Currently out of scope of the project.