Implementation Document

for

MindHive

Version 1.0

Prepared by

Group #: 2 **Group Name: MindHive**

Sahil Bansal	**200836**	**sbansal.sb2002@gmail.com**
Harsh Trivedi	**200422**	**harsh.trivedi.mikki@gmail.com**
Parth Maniar	**200671**	**pmaniar1906@gmail.com**
Ujwal Jyot Panda	**201060**	**ujwal12366@gmail.com**
Rishav Bikarwar	**200792**	**ris27hav@gmail.com**
Adi Pratap Singh	**200036**	**apsinghnsa@gmail.com**
Aakash Om Trivedi	**200003**	**aakashtri13@gmail.com**
Bhavya Garg	**200270**	**gargs.jpr@gmail.com**
L Gokulnath	**200542**	**lgokulnath02@gmail.com**
Gavish Garg	**200385**	**gavish94170garg@gmail.com**

Course: **CS253**

Mentor TA: ***Sri Madhan***

Date: **19/03/2022**

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
v1.0	All group members	First Version of Implementation Document	19/03/2022

1 Implementation Details

Programming Language, Framework and Libraries for Backend

MindHive has been mainly written in **Python**. We have used **Django**, a Python-based free and open-source framework for web development. It follows the *model-view-controller* architectural pattern. We chose Django because we believe it is **robust yet simple**, reliable, and scalable. There are numerous advantages of Django, which are briefly stated below:

1. The syntax is simple, similar to Python
2. It has its own built-in server
3. Secure: prevents SQL injection, cross-site scripting and has a robust user authentication functionality.
4. Object-oriented design
5. Scalable: web apps written in Django can handle high web traffic efficiently
6. It comes with an admin interface for developers
7. It supports rapid development
8. Excellent community support
9. It comes with a built-in DB (SQLite)
10. Encourages modularity of code

Considering the above-stated advantages of Django, it is no wonder that one of the most popular apps today, Instagram, was initially built in Django.

We have also used many Python and Django libraries, a detailed list of which is given in the GitHub repository.

Database

- We have used the **SQLite database engine**. It comes built-in with Django and is small, fast, and highly reliable.
- It is the most used DB engine in the world.
- For deploying our software on the server, we plan to **migrate to PostgreSQL**, which is considered the most potent Database Management System. We haven't used PostgreSQL straight away because it is overkill if there are only a few users.
- Moreover, we can migrate to PostgreSQL very easily in Django with just a few lines of code.

Front-end

- We have used **HTML**(HyperText Markup Language) as a standard markup language for documents designed to be displayed in a web browser.
- We have used **CSS**(Cascading Style Sheets) for Styling.
- We have also used the Bootstrap framework for making our website more responsive.
- We have used **Java-Script** as a scripting language to develop interactive web pages.
- We have also used the **jQuery** framework to use the **AJAX** tool to make and process requests without page reload.

2 Codebase

Github Repository- *https://github.com/ujwaljp/CS253_Project_MindHive*

Code Structure-

The repository contains an implementation/MindHive folder which contains all the code for our web app.

The project is divided into several small apps-

- Answers - represents the class of answer
- Questions - represents the class of questions
- Comments- represents the class of comments
- Users - represents the class of users
- Home - represents the home page, and contains the content class
- Notifications - implements functionalities of notifications
- Media - Contains all the images
- Tags - implements functionalities for adding tags to questions
- MindHive - app to integrate all the above apps.

Two more folders are present-

- Templates - it has the HTML files for the pages
- Static - it stores static files such as CSS and javascript files.

Some important files/folders in each of the apps mentioned above -

- models.py - describe each object's class.
- admin.py - registers and customises the admin view for the models
- apps.py - registers apps with Django
- forms.py - contains the form to either create a new object of the given model/class or edit the already existing object.
- urls.py - contains URL patterns that match a URL with a view for a page
- views.py - contains functions that take web requests and return web responses
- templates - has the HTML files for the pages

3 Completeness

Ideas Implemented From SRS

- Maintain user profile, add a profile picture. We have allowed the user to change the username and profile image.
- Reset password functionality via email authentication using SMTP server.
- Provided the tag functionality so that users can select their topic of interest and easily filter the questions based on a tag.
- We have allowed the user to add questions with the tags so that only people who have chosen the given tag can see it.
- Also, the user can edit their question to make the required changes afterwards as it is often needed to change the question later.
- There is the option to ask questions anonymously if the user wants it allowing the user to get their doubt cleared without any hesitations.
- Have implemented functionality for users to like/dislike questions, follow/bookmark questions where for the following question we receive notification also.
- Add images/markdown text while asking questions to make the question more descriptive.
- Implemented notification functionality (if a user asks a question, they will get a notification if someone answers that question)
- Users can search questions by providing keywords.

Future Improvements

- Using PostgreSQL instead of SQLite, as SQLite allows only one user to edit the database and is not scalable.
- Adding functionality for searching for tags while adding for questions.
- Also, add the following user functionality.
- Sorting functionality- sort questions by date/upvotes on the home page.
- Functionalities for editing an answer.
- Blogs- Add blog features to enable users to write blogs.
- Add features to request answers from specific users while adding a question or mentioning them by using @ so that we can ask a question from a particular user.
- Also, we are planning to have a poll feature on our app.
- We are planning to have the database of all the students so that as we get the email id, we can automatically fill all fields possible.
- Currently, our search function shows only those questions that a user has searched for. We plan to improve our search function, which will also display related questions and the searched questions.

4 Process to run the code

Please run the following commands on the terminal:

- $ git clone https://github.com/ujwaljp/CS253_Project_MindHive.git
- $ cd CS253_Project_MindHive
- $ cd implementation
- $ pip install -r Requirements.txt
- $ cd *MindHive*
- $ python manage.py runserver
- Go on the localhost web address which must have been printed on the terminal

Appendix A - Group Log

MEET DATE	TOPIC DISCUSSED	DURATION
17 February 2022	We discussed how to develop our website like which Programming language, framework to use.	90 minutes
20 February 2022	(Meet with TA) We finalized our Programming Language as Python.	120 minutes
22 February 2022	We started working on the features of the project and everyone gave their ideas.	120 minutes
25 February 2022	We had a meeting with the professor to ask some doubts.	60 minutes
27 February 2022	We made the UI of our product roughly so that it is easy to see what we have to make and what are the features.	120 minutes
26 February 2022	Decided on our language and the framework and made the team for the frontend and backend	90 minutes
1 March 2022	Have meet with TA to decide the language	60 minutes
3 March 2022	Setup the git repository and decided some rules before pushing in the main branch.	60 minutes
7 March 2022	Distributed the work of frontend and back end within ourselves and who will be responsible for which part of the project.	90 minutes

10 March 2022	Again have a meeting to know the progress and the speed with which our work is going.	30 minutes
14 March 2022	Cleared each other's doubts and also changed some models.	60 minutes
15 March 2022	Solved the issue of front end and backend integrations.	90 minutes
17 March 2022	Started making the implementation document and decided briefly on the template.	60 minutes
19 March 2022	Meet to do some final touchup on the web app.	90 minutes
20 March 2022	Final meet before implementation submission	60 minutes