Hackathon- group 1

Farmers connect

A social networking web application for farmers

Abstract

Technical specification for a web application developed for farmers to connect with each other

# **Problem statement**

Develop a web application where farmers can socialize and share their thoughts and ideas or products. Farmers should be able to connect with each other, share posts with text, images and videos. They should be able to like, comment and share the posts. They should be able to chat with their connections/other online farmers

# **Application overview**

Farmers’ Connect is a simple web application designed and developed for farmers. It provides following features to the members of this application.

1. Login/Register
2. Share posts
3. Share pictures/Videos
4. Like other user’s post
5. Comment on other user’s post
6. Follow/Unfollow a user
7. Chat with online users

# **Technology Stack**

1. Mongo DB (5+)
2. Express (4+)
3. React (16+)
4. Node JS (14+)

# **Core development team**

1. Khurshid Alam – Senior technical analyst

An enthusiastic back end developer having expertise in nodeJS

1. Aditya Khadilkar – Technical analyst

Front end developer experienced in React

1. Manjeet Chauhan – Associate software engineer

Skilled developer with HTML/CSS/React/Angular and Node experience.

1. Suchita Wase – Senior consultant UI

Innovative and skillful UI/Web designer

1. Abhishek Prakash – Software engineer

Skilled developer with HTML/CSS/React/Typescript/javascript/Angular experience.

1. Archana Makude – Associate technical architect

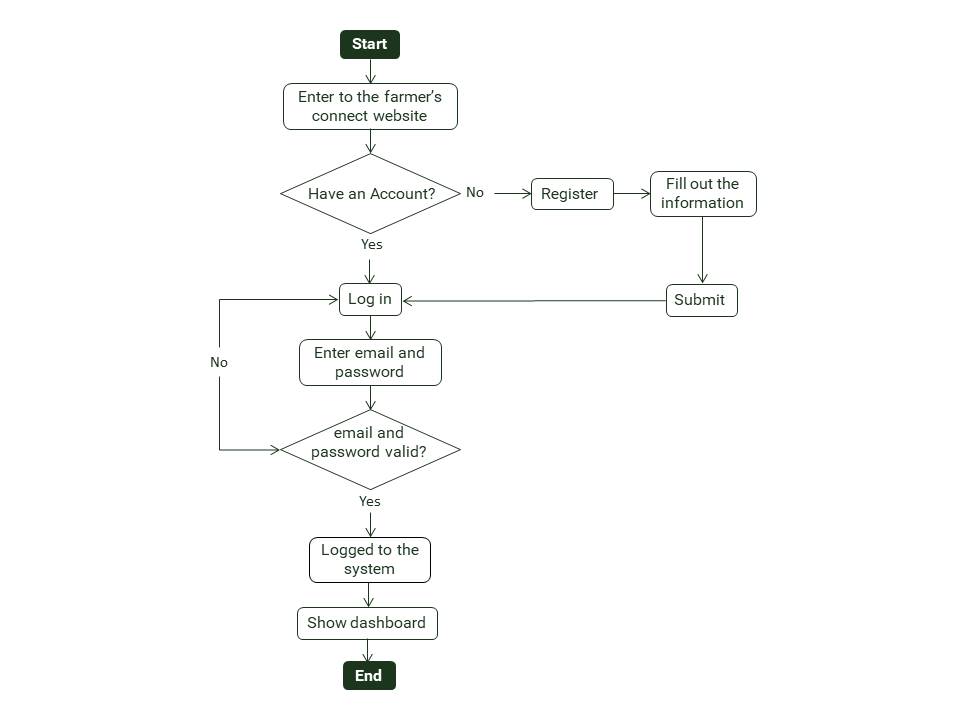
Experienced back end developer with expertise in NodeJS/Java/System Architecture and documentation

# **Scope**

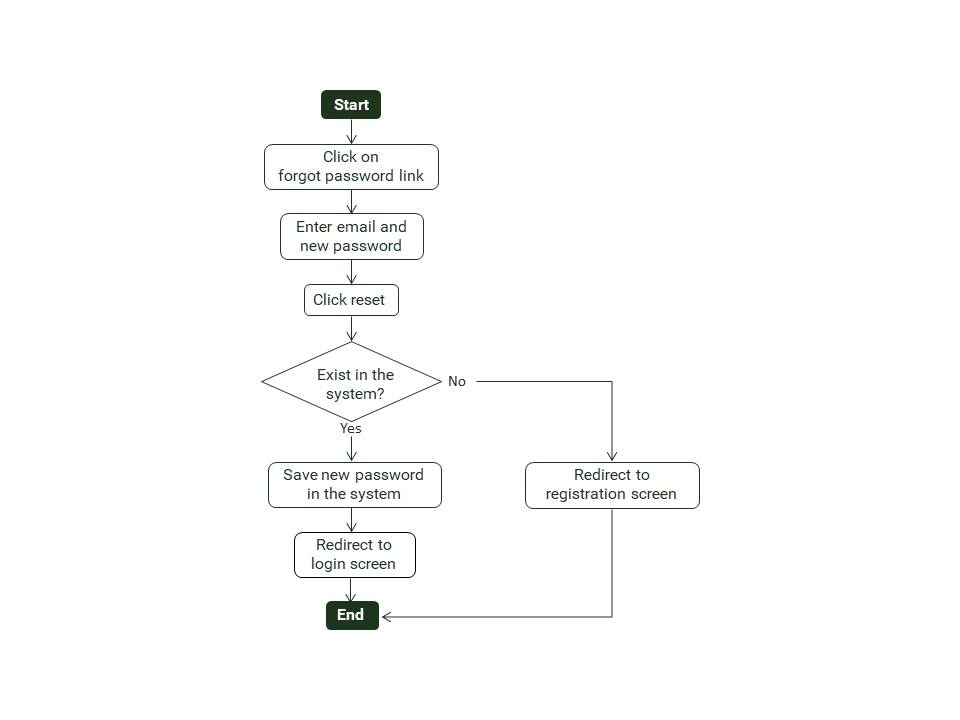
1. Login: Basic email id/password login is implemented as the advanced login needs more resources like SMTP server (for email verification) or notification service for OTP based login.
2. Forgot password: The only verification done at the system is existence of userid. Old password will be replaced by new password if the user email matches in the system
3. Dashboard: Once logged in, user will be able to see a dashboard where he can see the posts by other users in the middle pane, list of online members in the right pane and useful links to latest news, important articles in the left pane.
4. Search User: User can type the name in search box provided in the top menu bar to search other users by name.
5. View users’ profile: User can view other user’s profile.
6. Follow/Unfollow user: user can follow a particular user by clicking follow button on his profile. The followed users will be added into the friends list of a user.
7. View my profile and update profile: User can click on profile option at the top and view own profile, Update profile, upload profile picture.
8. Create post: User can create a new post from dashboard. The post can contain text, image and video. The post is public and can be seen by any other user.
9. View/Like/Comment/Share Post: User can view other users’ posts. Like the post by hitting like button. Share the post with other contacts.
10. In the left pane, user will be able to see links to useful resources like important articles, news, youtube videos etc. Currently this list is statically maintained.
11. Online users: Application will show list of users (online users at the top) in the right side pane of the dashboard. The order and preference is not set. It will show first 10 online users on the screen.
12. Chat: User can chat in a general chat room or in a personal one-to-one chat.

# **Flow Diagrams**

1. Login/Registration:



1. Forgot password



1. Create post
2. Update Profile
3. Share/Comment/Like post

# **Database Design**

The application uses a NoSQL database MongoDB to save system data. As NoSql is easy to maintain and doesn’t enforce referential integrity it is the best fit for social networking apps.

Currently system uses de-normalized form of data to keep it simple in its initial version. Later it can be normalized by adding references between the documents.

Farmers’ connect application mainly uses following data collections

1. users

|  |  |  |
| --- | --- | --- |
| ***Field*** | **Data type** | **Remarks** |
| \_id | ObjectId | primary\_key, system generated, do not change |
| username | string | unique key |
| email | string | unique key |
| password | encrypted string |  |
| profilePicture | string | path to local storage |
| coverPicture | string | path to local storage |
| followers | list of strings | list of \_id of followers |
| followings | list of strings | list of \_id of followings |
| isAdmin | boolean | default false |
| createdAt | datetime | date time when user created |
| updatedAt | datetime | date time when profile updated |

1. posts

|  |  |  |
| --- | --- | --- |
| ***Field*** | **Data type** | **Remarks** |
| \_id | ObjectId | primary\_key, system generated, do not change |
| userId | string | References \_id of users |
| desc | string | Text added in the post |
| likes | list of strings | list of users liked the post |
| comments | list of objects | list of {userId, comment} |
| media | string | media identifier |
| uploadType | string |  |
| createdAt | datetime | date time when user created |
| updatedAt | datetime | date time when profile updated |

# **Security**

1. Users’ password are saved in encrypted format in database.
2. Users are authenticated with JWT token at each request.
3. All the APIs are secured with JWT authentication.
4. Express rate limit is implemented to handle DoS attacks from a single IP address