Assignment 6x : COL733

MyBook WebApp on Azure

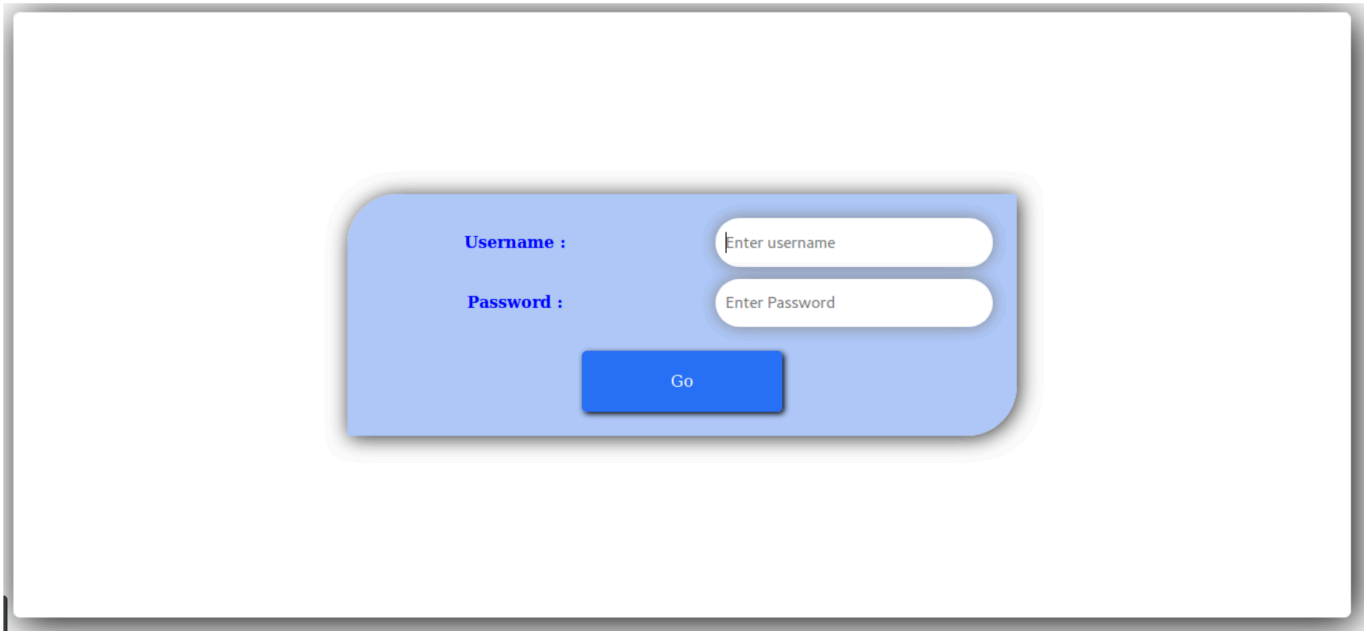
Group 20

Authors	Vaibhav Bhagee	Ankush Phulia	Rachit Arora	Kabir Chhabra
Entry No.	2014CS50297	2014CS50279	2014CS50292	2013CS50287

A. Design of the Web App

The app has a simple and effective design to deliver on its purpose while mantaining ability to scale.

1. It has a secured login portal with sessions that are mantained using tokens.

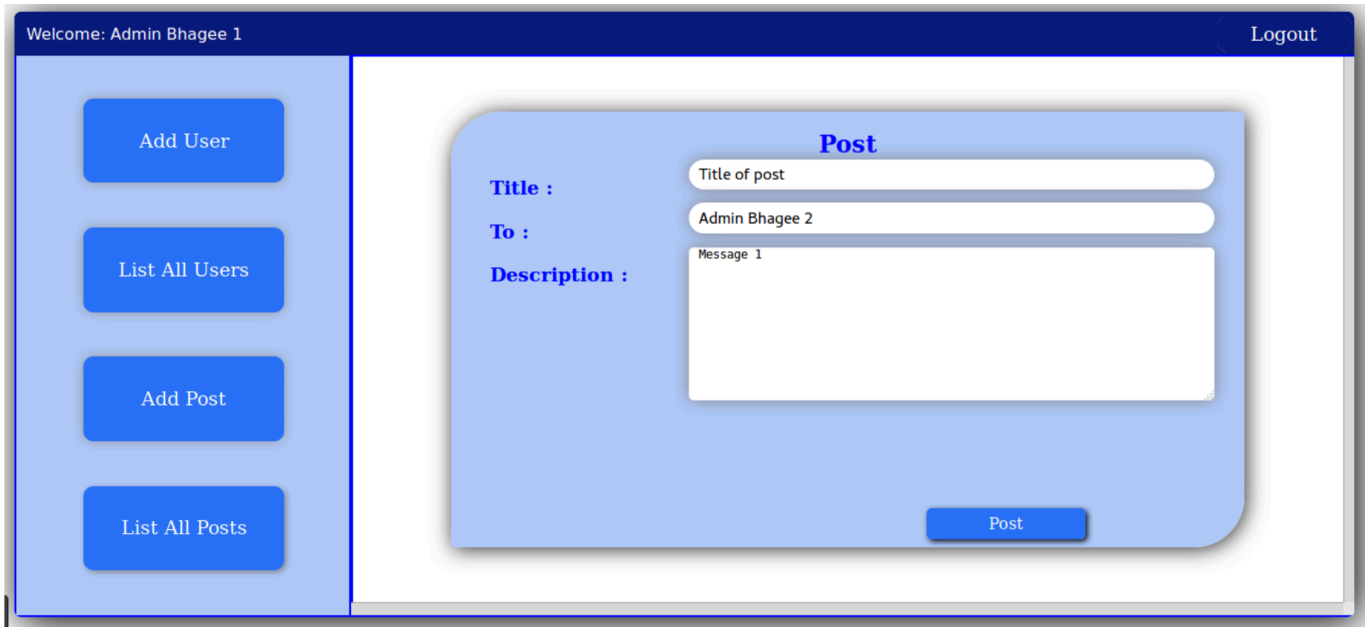


2. Users with admin capabilities are able to add new users.

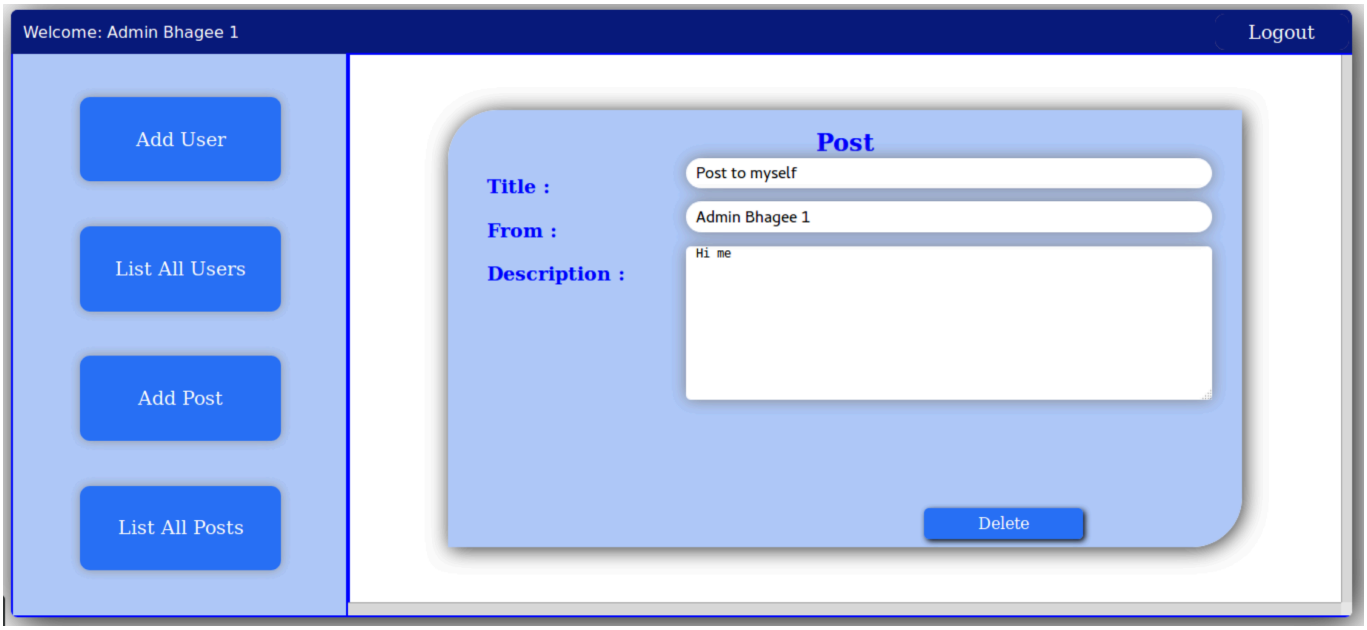
3. Users with admin capabilities are also able to modify/remove existing users.

4. Media files like images can be associated to a given user.

5. A given user can message other existing users on the MyBook network.



6. One can view all messages and delete messages addressed to user.



7. It is a single page application with focus on security, minimal api calls leading to lesser refreshes and a clean design.

APIs

- API 1 - The Login api - Login request with credentials : POST
- API 2 - Adding users to the database, editing and deleting details : POST
- API 3 - Fetching the list of users and displaying it : GET
- API 4 - Uploading media and text associated with a user : POST
- API 5 - Post a message to another user : POST
- API 6 - Get all messages addressed to a user/ delete them : GET
- API 7 - Hidden APT to clean up database documents for posts and users.

B. Implementation on Azure

Services Used

- **App Service** - For hosting the webapp.
- **App Service Plan** - For billing details associated with the web app.
- **Azure Cosmos DB account** - NoSQL Database with MongoDB API.
- **Storage Account** - Blob/Block/File storage.

Steps/Guide to Azure

- Create a resource group - ‘MyBook’, provides a superclass for all the services to be associated with

each other.

- Create instances of all the services mentioned below, give resource group as above and location as Southeast Asia.
  - **App Service Plan** - Decide the provisioning - storage, ram, access control, region and billing for an app service.
  - **App Service** - Create an app with a unique hosting name, decide the stack to be used - Node.js 6.11, access control, quotas, etc.
  - **Azure Cosmos DB Account** - Create a db and decide its id, API - MongoDB, access options, redundancy - master and slave replicas, consistency and obtain the connection string and primary master keys.
  - **Storage account** - For blob/file storage. Requires a 'container' which allows storage and access of blobs to/from it. Also can adjust access keys and encryption, CORS rules.

- Since the app is in Node.js, we require node and npm installed on the work machines.
- Set up a git repository for the project, link it to the azure account via the webapp options, and configure it to push changes to azure as well via git remote add azure.
- Specify the packages and various environment variables that the services need.
- From initial commit of the app to the repository we can deploy a basic web app on azure, accessible via .azurewebsites.net .
- All changes committed and pushed to git are also reflected in Azure, which automatically deploys the updated app.

C. References

- Microsoft Azure Documentation | Microsoft Docs
- The MongoDB Manual
- Node.js Docs