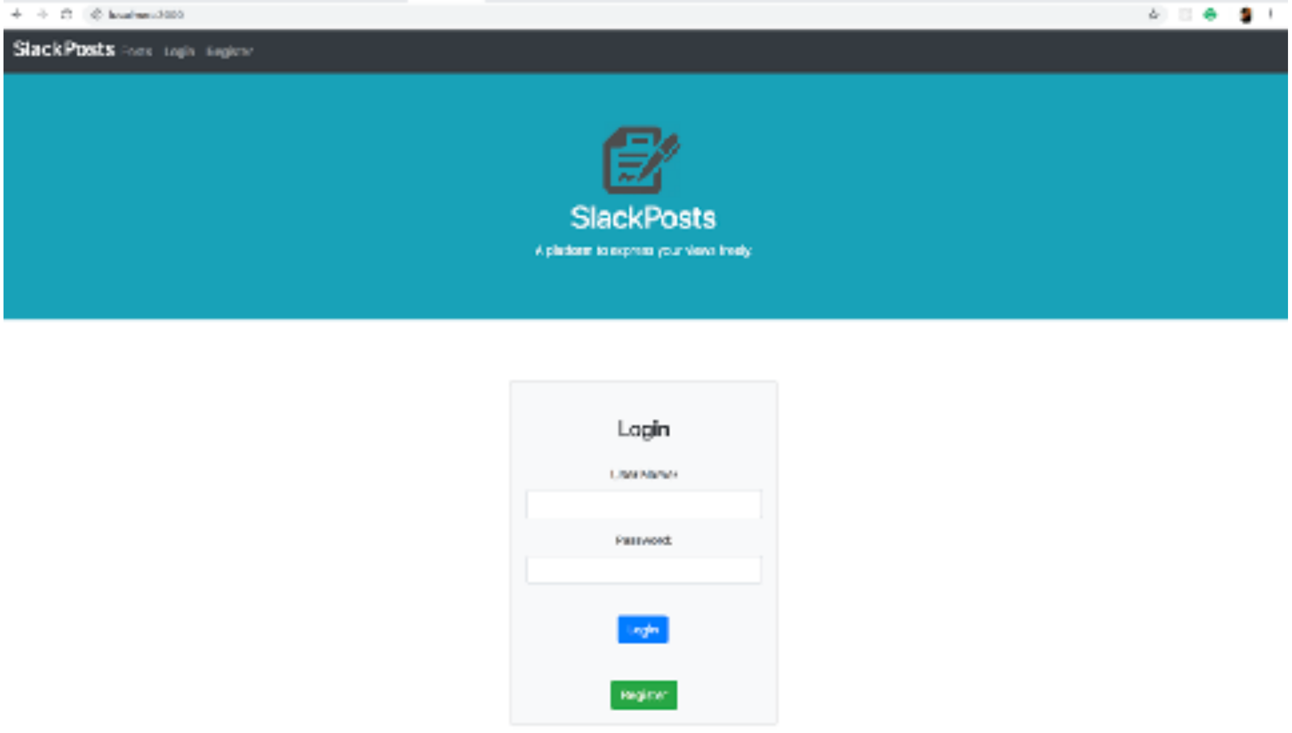
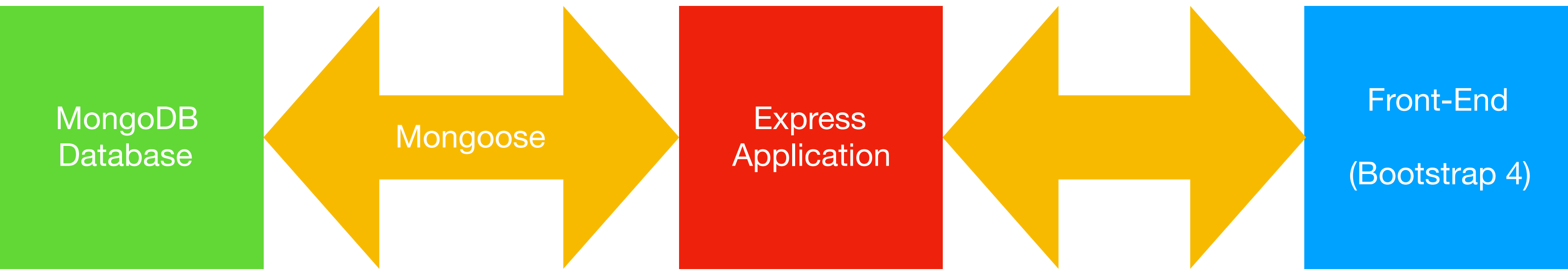


SlackPost is a Blog application developed using Express , NodeJs , MongoDB and Bootstrap 4.

NodeJs is used for backend of the application with express used for simplifying the development of the application. EJS is used as Express's view engine used to render pages. Further MongoDB is used as the database. Node package Mongoose is used to interact with the MongoDB database. The below diagram shows the overall flow of the project.



- **MongoDB Database:** Used to store data related to users and blog posts. The communication between mongoDB database and express application is using Mongoose Node Package.

- **Express Application:** Express application is used to read/write the data from the database and handle HTTP requests (GET and POST requests). The views are accordingly render based on the requests. The views are present in EJS format.

- **Front End:** The front end views are present as EJS views. Frontend UI was design using Bootstrap 4. The posts requests from form submissions are handled in express application using body-parser node package.

Database

MongoDB database named BLOG has two collections:

- **Users:** Stores information related to users. The schema of the users is the following

```
({
    UserName:String,
    password:String,
    title:String
});
```

- **Posts:** Stores information related to posted articles. The schema of posts is the following

```
({
    User:String,
    UserTitle:String,
    title:String,
    content:String,
    date:{type:Date,default:Date.now}
});
```

Express Application

Express application is used to read/write data to database and handle GET/POST requests. Accordingly it renders the EJS views.

Get Requests

- **GET('/')**

The homepage of the application

- **GET('/Register')**

Renders the registration view

- **GET('/CreatePost')**

Renders view that allows user to create posts

- **GET('/Logout')**

Logs out user from the account being used and returns to login page.

- **GET('/UserPost')**

Give author and title of a given blogpost it sends the blogpost information of respective blogpost and renders it using Blogpost.ejs view.

- **GET('/Posts')**

Renders view which has all the blogposts

Post Requests

- **POST('/Login')**

Handles post request from login form by checking if the give username/password login pair exists or not.

- **POST(''/AddUser')**

Handles post request from registration form by checking if the details entered by user is valid and Saving it to database.

- **POST('/CreatePost')**

Handles post request of the form used to create posts and saves it to the database.

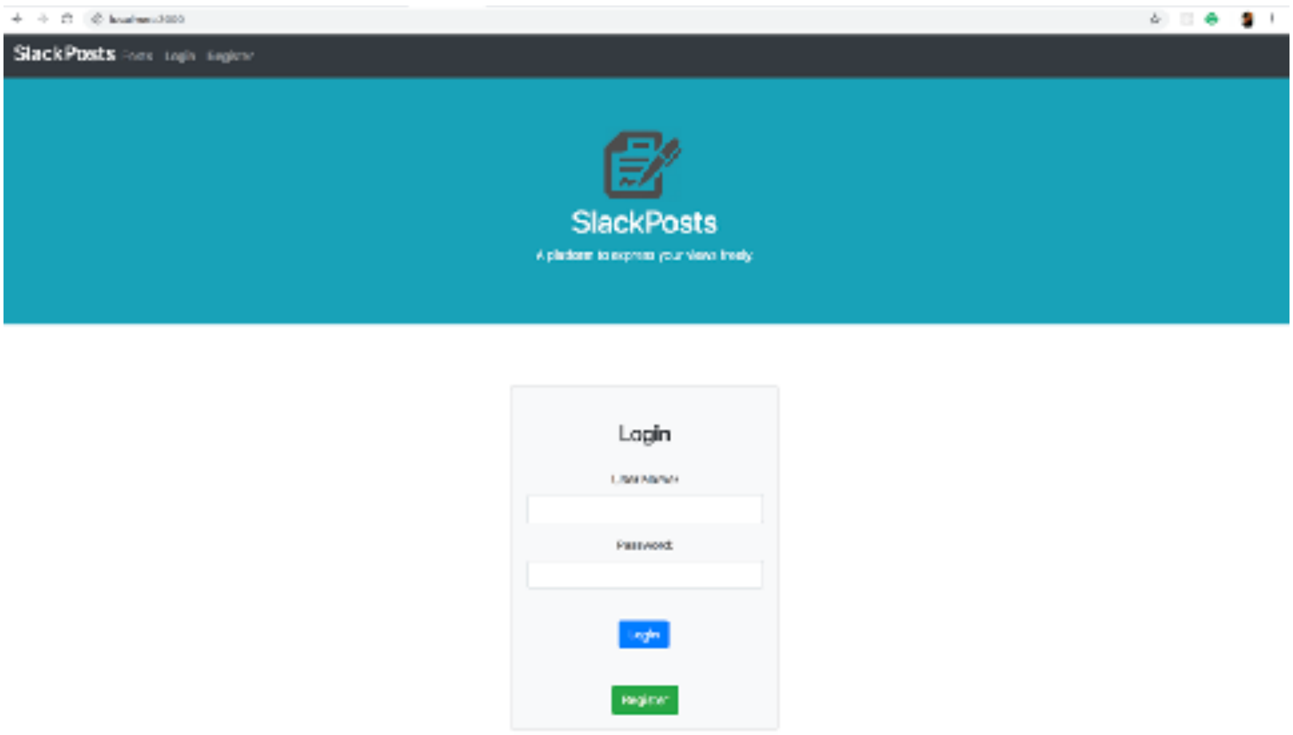
Frontend

The frontend uses Bootstrap 4 templates and EJS views. The EJS views are rendered according to the data passed by express framework. The following are the views used for application.

- Index
- Register
- Dash
- CreatePost
- BlogPost
- Nav (Partial)

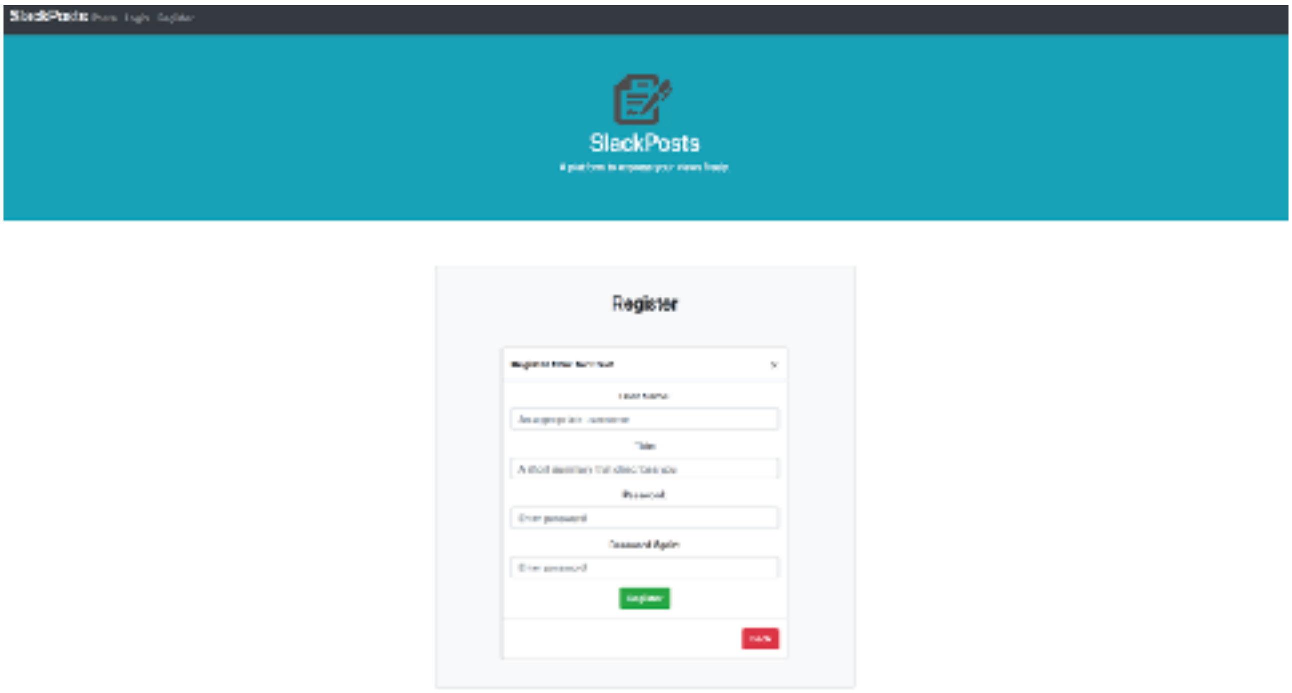
Index

Index view consists of the form which allows the user to login using desired credentials. The form sends a POST request POST('/Login').



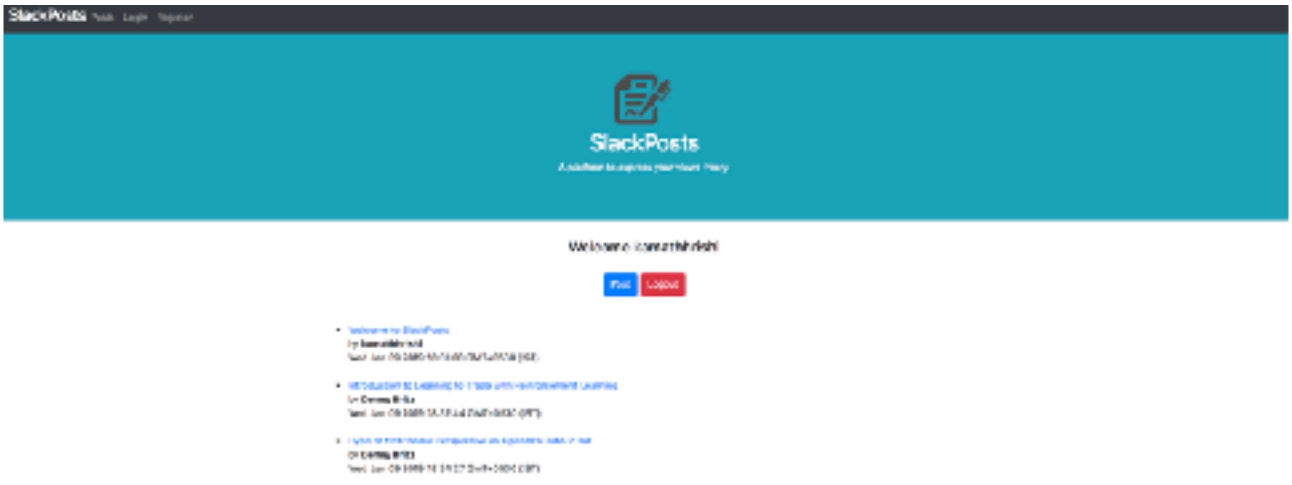
Register

Register view has a form allowing user to register. Form submission sends the request POST('/AddUser').



Dash

Dash is the view rendered when the user logs in with appropriate registered credentials. It consists of the posts posted on the blog.



SlackPosts Posts Login Register

SlackPosts

A platform to express your views freely

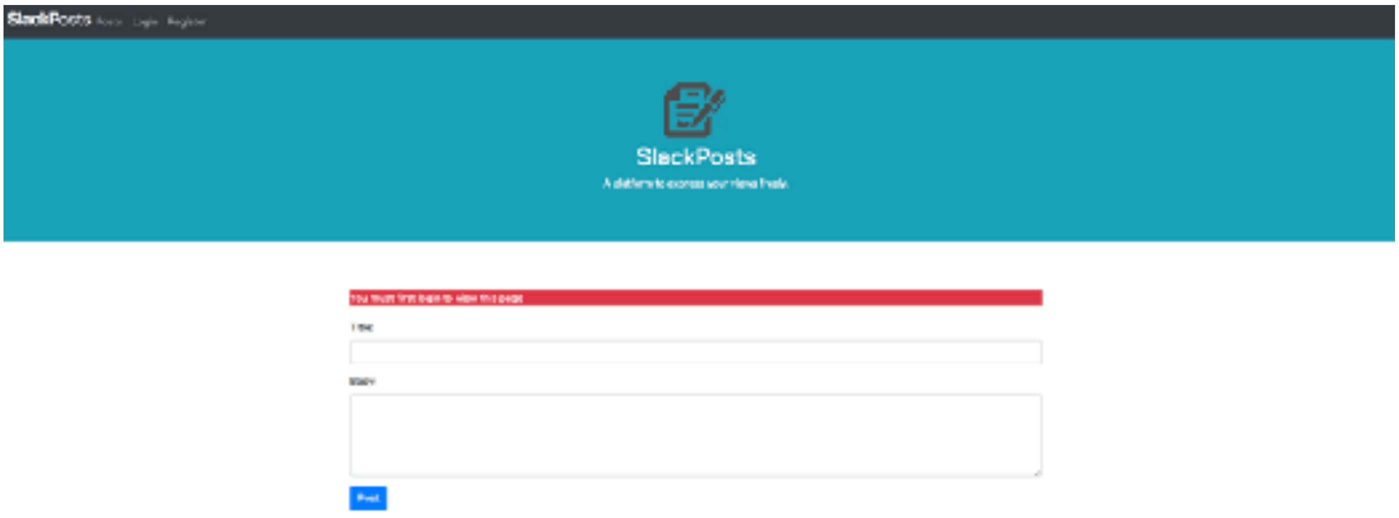
Welcome [illegible]

Post Logout

- [illegible]
  by [illegible]
  [illegible]
- [illegible]
  by [illegible]
  [illegible]
- [illegible]
  by [illegible]
  [illegible]

CreatePost

Create post is the view rendered using which the user can create blogposts



SlackPosts Posts Login Register

SlackPosts

A platform to express your views freely.

You must first login to view this page

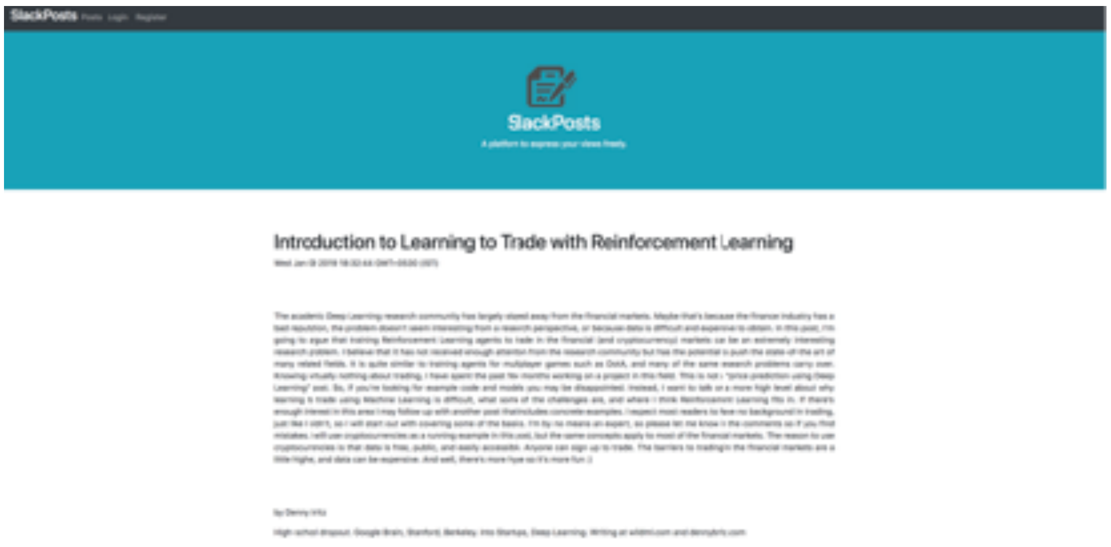
Title

Story

Post

BlogPost

The default view which renders blogposts in a given format. The blog post content is passed as JSON data to the view which is rendered as follows. The view is rendered when a specific blogpost is clicked on in Dash view.


SlackPosts Posts Login Register
SlackPosts
A platform to express your views freely

Introduction to Learning to Trade with Reinforcement Learning

Wed Jan 09 2019 18:32:44 GMT+0530 (IST)

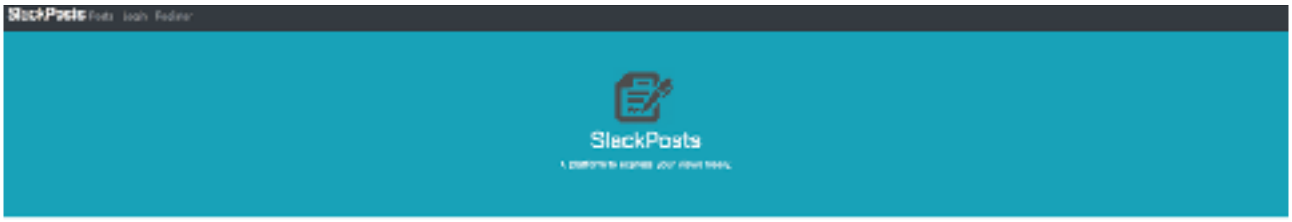
The academic Deep Learning research community has largely stayed away from the financial markets. Maybe that's because the finance industry has a bad reputation, the problem doesn't seem interesting from a research perspective, or because data is difficult and expensive to obtain. In this post, I'm going to argue that training Reinforcement Learning agents to trade in the financial (and cryptocurrency) markets can be an extremely interesting research problem. I believe that it has not received enough attention from the research community but has the potential to push the state-of-the art of many related fields. It is quite similar to training agents for multiplayer games such as DotA, and many of the same research problems carry over. Knowing virtually nothing about trading, I have spent the past few months working on a project in this field. This is not a "price prediction using Deep Learning" post. So, if you're looking for example code and models you may be disappointed. Instead, I want to talk on a more high level about why learning to trade using Machine Learning is difficult, what some of the challenges are, and where I think Reinforcement Learning fits in. If there's enough interest in this area I may follow up with another post thatincludes concrete examples. I expect most readers to have no background in trading, just like I didn't, so I will start out with covering some of the basics. I'm by no means an expert, so please let me know in the comments so if you find mistakes. I will use cryptocurrencies as a running example in this post, but the same concepts apply to most of the financial markets. The reason to use cryptocurrencies is that data is free, public, and easily accessible. Anyone can sign up to trade. The barriers to tradingin the financial markets are a little higher, and data can be expensive. And well, there's more hype so it's more fun :)

by Denny Britz

High school dropout. Google Brain, Stanford, Berkeley. Into Startups, Deep Learning. Writing at wildml.com and dennybritz.com

Nav (Partial View)

Nav partial view is included in every view. It includes the navbar and header with Slackposts Logo and title in every view. Since it remains constant for every view, it is a separate view included in every view.


SlackPosts Posts Login Register
SlackPosts
A platform to express your views freely