

Project Specification
poetica: a personalized poetry experience

Aditi Narasimhan (Andrew ID: aditin)

Jaspreet Singh (Andrew ID: jasprees)

Product Backlog

- ☒ ~~Create mockups (A)~~
- ☒ ~~Integrate database (A)~~
- ☐ Models.py
- ☐ Homepage
 - ☒ ~~Template with navbar for all pages (except login/register) (A)~~
 - ☒ ~~HTML/CSS with dummy data (A)~~
- ☐ Login page
 - ☒ ~~HTML/CSS with dummy data (A)~~
 - ☐ OAuth
 - ☒ ~~Random poem quote from database~~
- ☐ Register page
 - ☒ ~~HTML/CSS with dummy data (A)~~
 - ☒ ~~Random poem quote from database~~
- ☐ Profile page
 - ☒ ~~HTML/CSS with dummy data (A)~~
 - ☐ Upload photo/bio
 - ☐ Link to all starred poems
- ☐ Upload poems page
 - ☐ HTML/CSS with dummy data
 - ☐ Upload poem + associated emotions/colors
 - ☐ Poem should be added to database
 - ☐ Automatically Star your uploaded poetry?
 - ☐ Popup(?) — would you like to upload another poem
- ☐ Poem-view page
 - ☐ HTML/CSS with dummy data
 - ☐ Display correct poem from database
 - ☐ Add aura
 - ☐ Star feature
- ☐ Keep going pop-up
 - ☐ Update criteria — for Discover
 - ☐ Keep going — for Random
- ☐ Random poem pathway
 - ☐ Randomly choose a poem

- ☐ Display poem-view page
- ☐ Discover poem pathway
 - ☐ Discover form page
 - ☒ ~~HTML/CSS with dummy data (A)~~
 - ☐ Filter database through form responses
 - ☐ Display poem-view page
- ☐ Connect all pages/links

Sprint #1 Backlog

Product Owner: Aditi Narasimhan (aditin)

- ☒ ~~Create mockups (Aditi)~~
- ☒ ~~Add datasets~~
 - ☒ ~~Emotions database + cleaning script (Aditi)~~
 - ☒ ~~Poem database + cleaning script (Aditi)~~
- ☒ ~~Models.py~~
- ☒ ~~Templates for all HTML pages (Jaspreet)~~
- ☒ ~~Homepage~~
 - ☒ ~~Template with navbar for all pages (except login/register) (Aditi)~~
 - ☒ ~~HTML/CSS with dummy data (Aditi)~~
- ☒ ~~Login page~~
 - ☒ ~~HTML/CSS with dummy data (Aditi)~~
 - ☒ ~~Random poem quote (Aditi)~~
- ☒ ~~Register page~~
 - ☒ ~~HTML/CSS with dummy data (Aditi)~~
 - ☒ ~~Random poem quote (Aditi)~~
- ☒ ~~Profile page~~
 - ☒ ~~HTML/CSS with dummy data (Aditi)~~
- ☒ ~~Top-liked poems page~~
 - ☒ ~~HTML/CSS with dummy data (Jaspreet)~~
- ☒ ~~Upload poems page~~
 - ☒ ~~HTML/CSS with dummy data (Jaspreet)~~
- ☒ ~~Poem-view page~~
 - ☒ ~~HTML/CSS with dummy data (Jaspreet)~~
- ☒ ~~Random poem pathway~~
 - ☒ ~~Display poem-view page (Jaspreet)~~
- ☒ ~~Discover poem pathway~~
 - ☒ ~~Discover form page~~
 - ☒ ~~HTML/CSS with dummy data (Aditi)~~
- ☒ ~~Make all links work (Jaspreet)~~

Models

We used Django models to keep track of our user profiles and also which poems they have starred. Each profile will store its user, bio, profile picture, content type for the picture, and a list of poems the user has starred.

```
from django.db import models
from django.contrib.auth.models import User

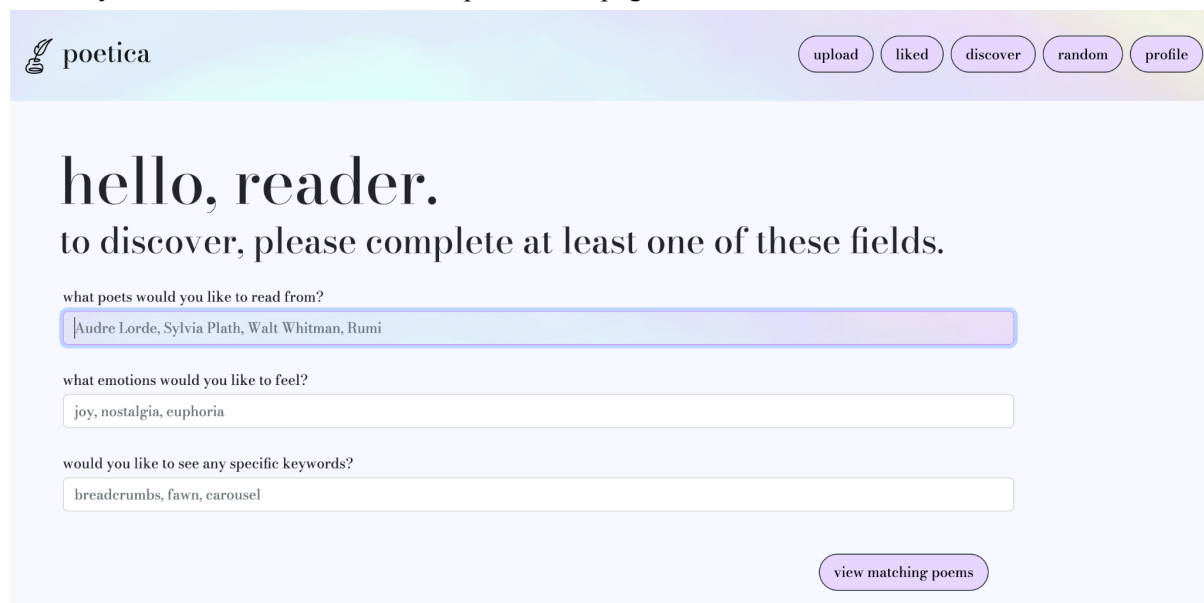
# Create your models here.
class PoemInfo(models.Model):
    title = models.CharField(max_length=100)
    author = models.CharField(max_length=50)

class Profile(models.Model):
    user = models.OneToOneField(User, on_delete=models.PROTECT)
    bio = models.CharField(max_length=200)
    profile_picture = models.FileField(blank=True)
    content_type = models.CharField(max_length=50)
    starred = models.ManyToManyField(PoemInfo)
```

Mockups

Discover Poem Page

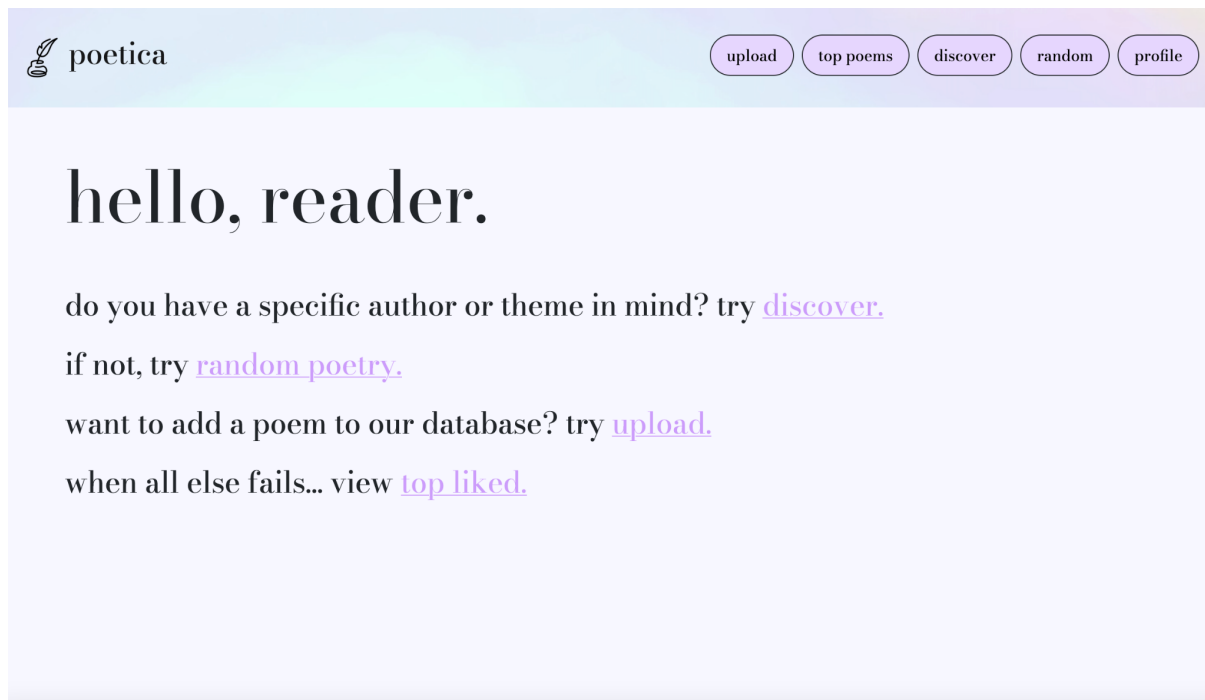
The discover poem page is where users can input poets, emotions, and keywords they would like to search for. They will then be redirected to the poem view page.



The mockup shows a web interface for 'poetica'. At the top, there's a navigation bar with a logo and five buttons: 'upload', 'liked', 'discover', 'random', and 'profile'. The main content area has a large heading 'hello, reader.' followed by the instruction 'to discover, please complete at least one of these fields.' Below this, there are three input fields with placeholder text: 'what poets would you like to read from?' (containing 'Audre Lorde, Sylvia Plath, Walt Whitman, Rumi'), 'what emotions would you like to feel?' (containing 'joy, nostalgia, euphoria'), and 'would you like to see any specific keywords?' (containing 'breadcrumbs, fawn, carousel'). A 'view matching poems' button is located at the bottom right of the form area.

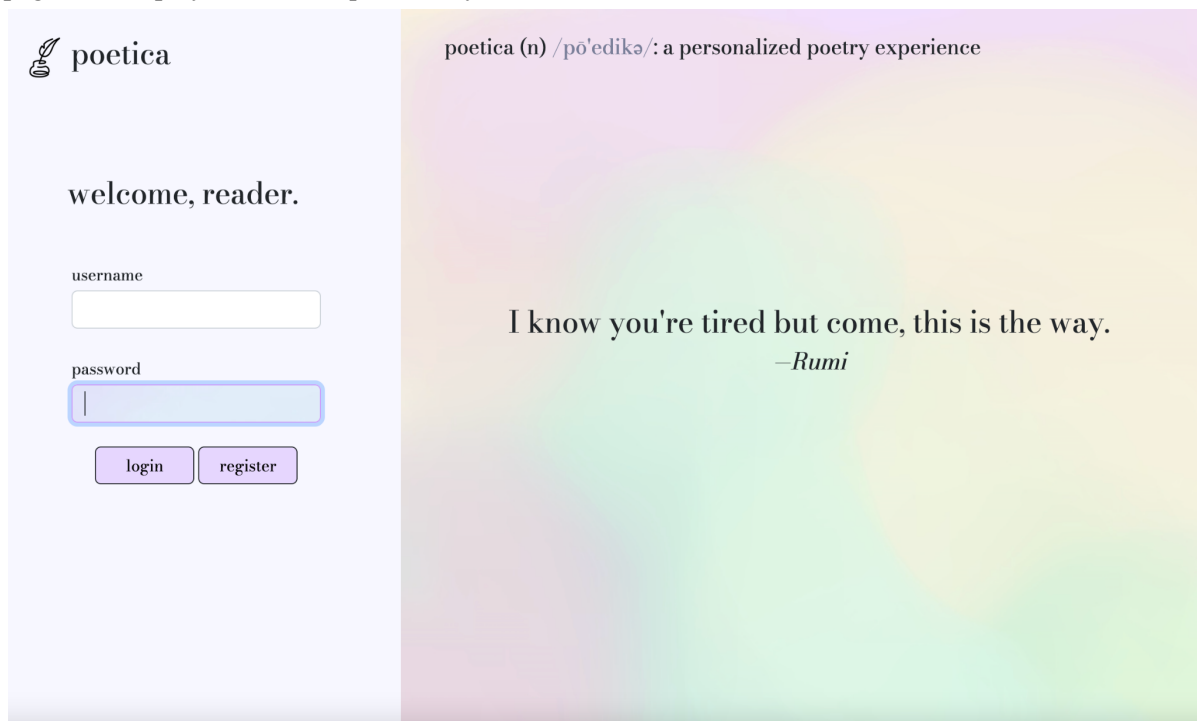
Homepage

The homepage is where users are directed to when they first access poetica. On the homepage, users are provided with links to other available pages.




Login Page

The login page is where users can log in to their account and get access to the rest of the site. The login page also displays a random quote every time it is accessed.



Register Page

The register page is where users can register a new account in order to gain access to the site. The register page will also display a random quote every time it is accessed.

 poetica

poetica (n) /pō'edikə/: a personalized poetry experience

welcome, reader.

email

username

password

confirm password


sign up

already have an account? [login.](#)

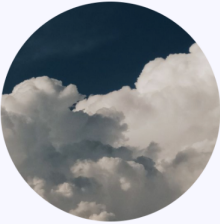
I know you're tired but come, this is the way.
—Rumi

Profile

The profile page is where users can view their own profile or other profiles. The information displayed includes username, biography, profile picture, and starred poetry.

 poetica

upload saved discover random profile



username

your bio here

update profile

username's saved poetry

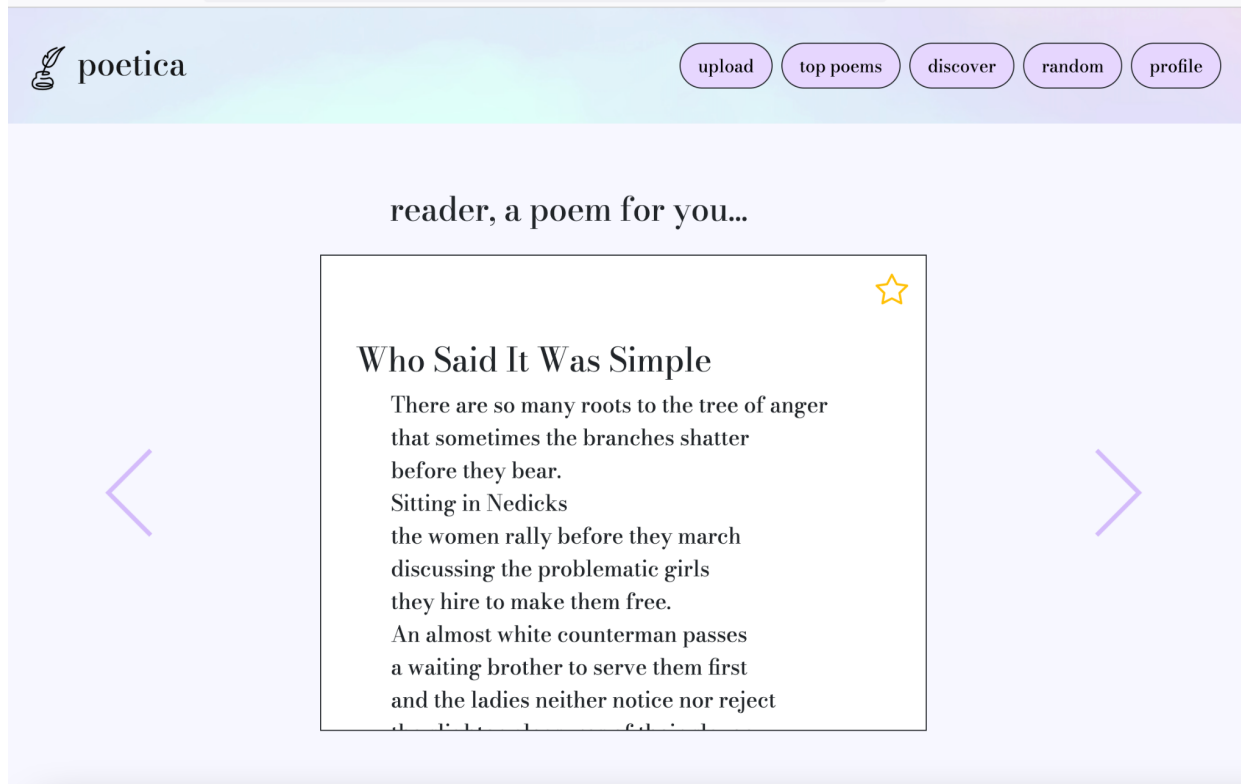
"One Art" — Elizabeth Bishop

"Song of Myself" — Walt Whitman

"Bell Jar" — Sylvia Plath

Poem-View Page

The poem view page is where users are shown poems. Users can press the right and left arrows to view new poems and can also input an emotion they associate with the poem.



Sprint #2 Backlog

Product Owner: Jaspreet Singh (jasprees)

- ☐ Upload poems page
 - ☐ Upload poem + associated emotions/colors
 - ☐ Poem should be added to database (A)
- ☐ Popups – Update criteria — for Discover, Keep going — for Random, Upload – Upload another?
- ☐ Poem-view page
 - ☐ Display correct poems from database
 - ☐ Add space for top 3 emotions
- ☐ Random poem pathway
 - ☐ Randomly choose a poem (A)
 - ☐ Display poem-view page (A)
- ☐ Discover poem pathway
 - ☐ Discover form page
 - ☐ Filter database through form responses