

TightReads

University of California, Riverside

Github Repo Link: <https://github.com/crystalarrow/CS180-TightReads>

Authors

Name	Github Profile
Daniel Fonseca	github.com/dfons007
Son Phan	github.com/SunnyPhan
Jonathan Ho	github.com/crystalarrow
Jonathan Oaks	github.com/nailcliper
Rocio Roman	github.com/roochio

Last Updated 10/28/2019

Table of Contents

Team Members and Emails:	1
User Stories:	1
Authors	4
Description	6
Detailed Design	6
High-level Design Description	6
Front-End Design	6
Back-End Design (including Database)	6
Features/Capabilities	7
GUI Mock-Up	8
Functions + What they do	9
Src:	9
App.js	9
BookCard.js	9
BookList.js	9
Books.js	9
Firebase.js	10
Google.js	10
Header.js	10
Profile.js	10
SearchArea.js	10
Pages folder:	10
404.jsx	10
Book_UI.jsx	10
Homepage.jsx	10
Index.jsx	11
Makeaccounts.jsx	11
Profile.jsx	11
Search.jsx	11
Biographies	11

1. Description

This project is a mock-up of goodreads in our taste that is named TightReads. TightReads allows users to connect with others and share what they are currently reading. In addition to sharing what you read, TightReads should be able to recommend books to the user. TightReads will be built as a web application and will be available as a website only.

2. Detailed Design

a. High-level Design Description

A general overview of what we will be using to design TightReads is that it will be built in Python with flask as it's framework. We plan on using react for the front-end. TightReads will also implement Firebase as a database.

b. Front-End Design

We plan on using the JavaScript library, React, for the front-end to develop our user interface. Example UI Mockup (Rough Sketch, Will change by the final product):

c. Back-End Design (including Database)

For now, we will be hosting the web application locally. The back-end will be built using Python with the micro web framework, Flask. We will be using Google Books API to help with book searching/book information that we need. We will be integrating Firebase as a database for user credential storage and

book information storage.

3. Features/Capabilities

Create an Account	A user may create an account that is usable for TightReads.
Login/Logout	A user will be able to login and logout of a valid account.
Book Recommendations	TightReads should recommend books based on a users interests.
Whitelist/blacklist authors or genres	A user should be able to whitelist/blacklists authors or genres.
Book Sharing	A user should be able to generate a shareable link that they can give to a different user.
Book Searching	A user should be able to search for a book based on title, author, genre, or rating.
Book Profile	TightReads should be able to provide a book profile for searched up books.
Book Rating	A user should be able to rate a book on a scale of 1 to Tight.
Book List	A user should be able to see what books they have read, what they are currently reading, and what they want to read.
Profile Edit	A user should be able to edit their profile after creating one.

4. GUI Mock-Up

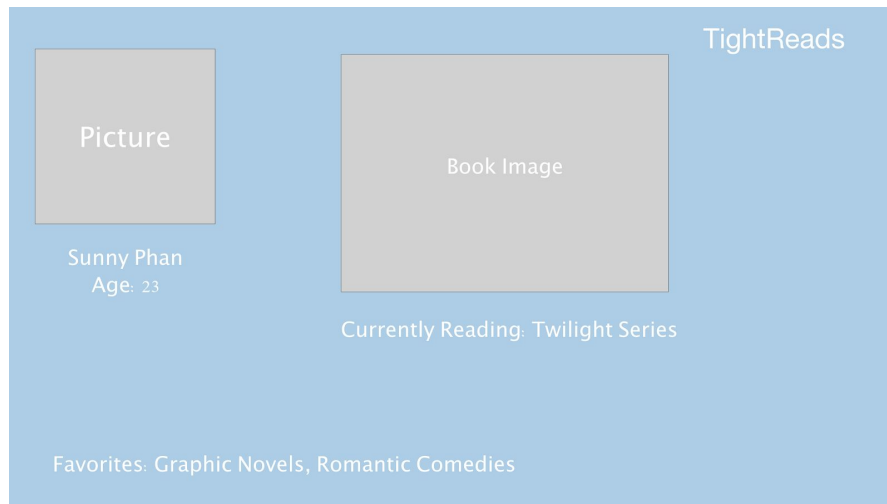


Image 1: Profile

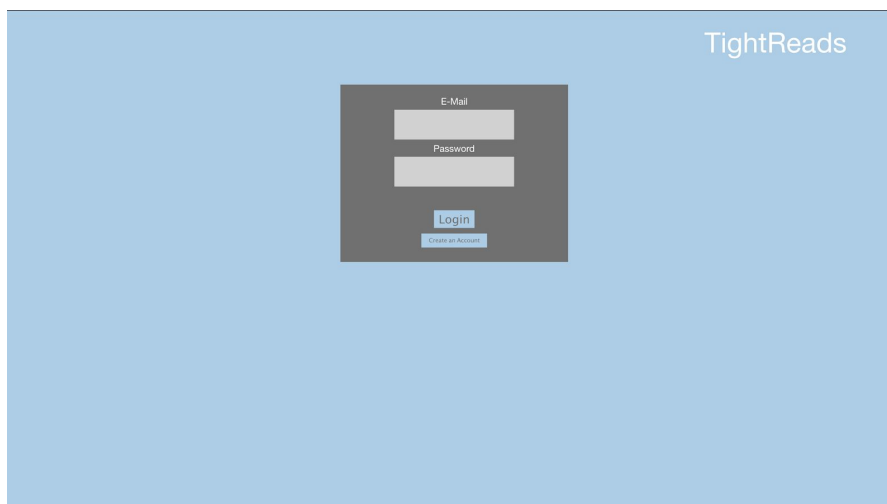
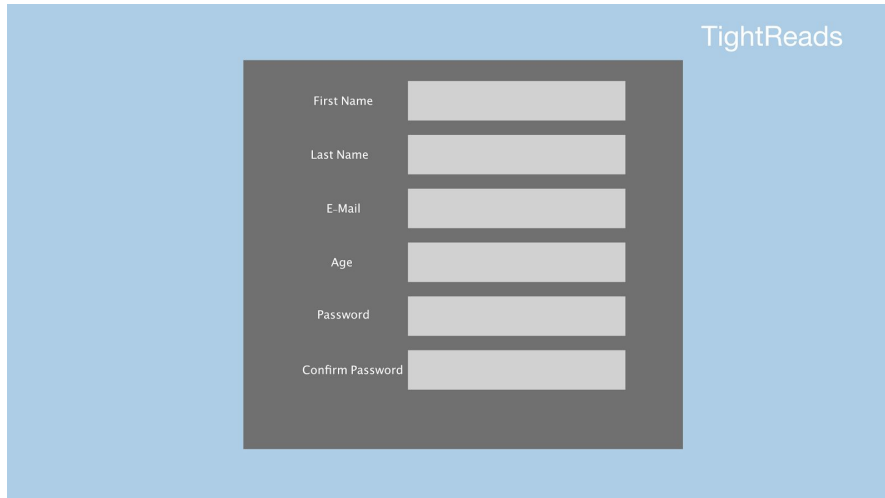


Image 2: Login Screen



TightReads

First Name

Last Name

E-Mail

Age

Password

Confirm Password

Image 3: Create an Account

5. Functions + What they do

Src:

App.js

- Handles the routing with the following paths:
 - “/”
 - “/makeaccounts”
 - “/bookprofile”
 - “/homepage”
 - “/books”
 - “/profile”
 - “/404”

BookCard.js

- Rendering code for each card in the search results that shows the book title, and author

BookList.js

- Rendering code for a list of books in the search results

Books.js

- Backend for book searching logic

Firestore.js

- Stores Firestore server credentials

Google.js

- Contains external functions for accessing the Google Books API

Header.js

- Rendering code for the page header

Profile.js

- Stores user profile information including
 - Name
 - Picture
 - Info
 - Status
 - reading lists
 - White/blacklist tags

SearchArea.js

- Code for the search bar

Rating.js

- Code for rating books which displays a star system of 5 stars

Pages folder:

404.jsx

- Fallback page when attempting to access a page that doesn't exist

Book_UI.jsx

- Page displaying book details
 - Image
 - Title
 - Author
 - Whitelist/Blacklist
 - Genre
 - Summary
 - External Link

- Book Rating
- Add/Remove Favorite books

Homepage.jsx

- Page displaying welcome message
- Pan through the different options that the website can be used for:
 - favoriting books
 - sharing books
 - finding new books
- On the bottom, shows recommended books based on a genre the user has favorited

Index.jsx

- Handles user login
- Landing page when the program is launched
- Contains fields for logging in to account and a link to account creation

Makeaccounts.jsx

- Handles account creation, parameters that store the user's first name, last name, email, and password
- This will check that:
 - The user has filled in all parameters
 - The email has proper email format
 - The passwords match
 - The account has not been already created
- When creating a new account, it will store the user credentials into the firebase server

Profile.jsx

- Displays user profile
- Displays the user's
 - Name
 - Picture
 - Information about the user
 - Interested genres
 - Recent reads lists
 - White/blacklist tags
 - Favorite books

Search.jsx

- Page displaying the search results for a given query

- The results are shown in book cards that the user can click into for more information

Biographies

Links to technologies we plan on using for this web application:

- <https://www.fullstackpython.com/flask.html>
- <https://developers.google.com/books/docs/overview>
- <https://getbootstrap.com/>
- <https://reactjs.org/>
- <https://www.npmjs.com/>