Web Studio: Week 2 Turn-in

Name: Ivy Chen
UNI: ic2389
Due: Friday 9/14/2018 @2PM
COMS6998 Adv. Web Studio

Original Goals

High Level Goal

1. Allow users to look up description and main actors in a movie

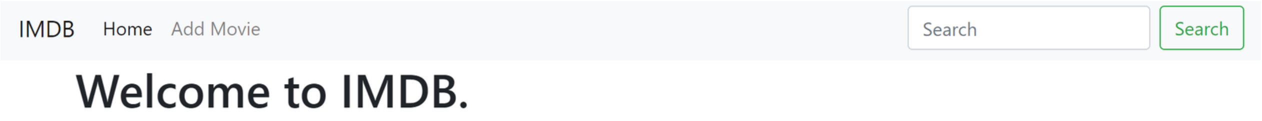
Low Level Goal

1. Install Flask and flask-bootstrap
2. Make two different web pages
3. Page 1 — main page lists movies
4. Page 2 — see details of one movie
5. Make a route such that when you click on a movie on Page 1, it takes you to individual movie page
6. Make a page to create a new movie
7. Make a page to update a movie's information
8. Make a button to delete a movie on admin page

Lessons Learned Through Iteration

1. My plan was to make my high level goal such that a user could search for a movie, but I ran into problems regarding design decisions and functionality of a search feature. For example, I would need to have a large volume of movies, decide how fuzzy/flexible the search would work, if users could search based on year/name/actor, etc.

   Since I'm not too familiar with searching using the Flask framework, I decided to solve it by changing my high-level goal to a goal that is more focused on the base content functionality of IMDB. That is, users will be able to view information about a movie including movie cast and a movie description. Later on, I will consider how to build a search feature.



Welcome to IMDB.

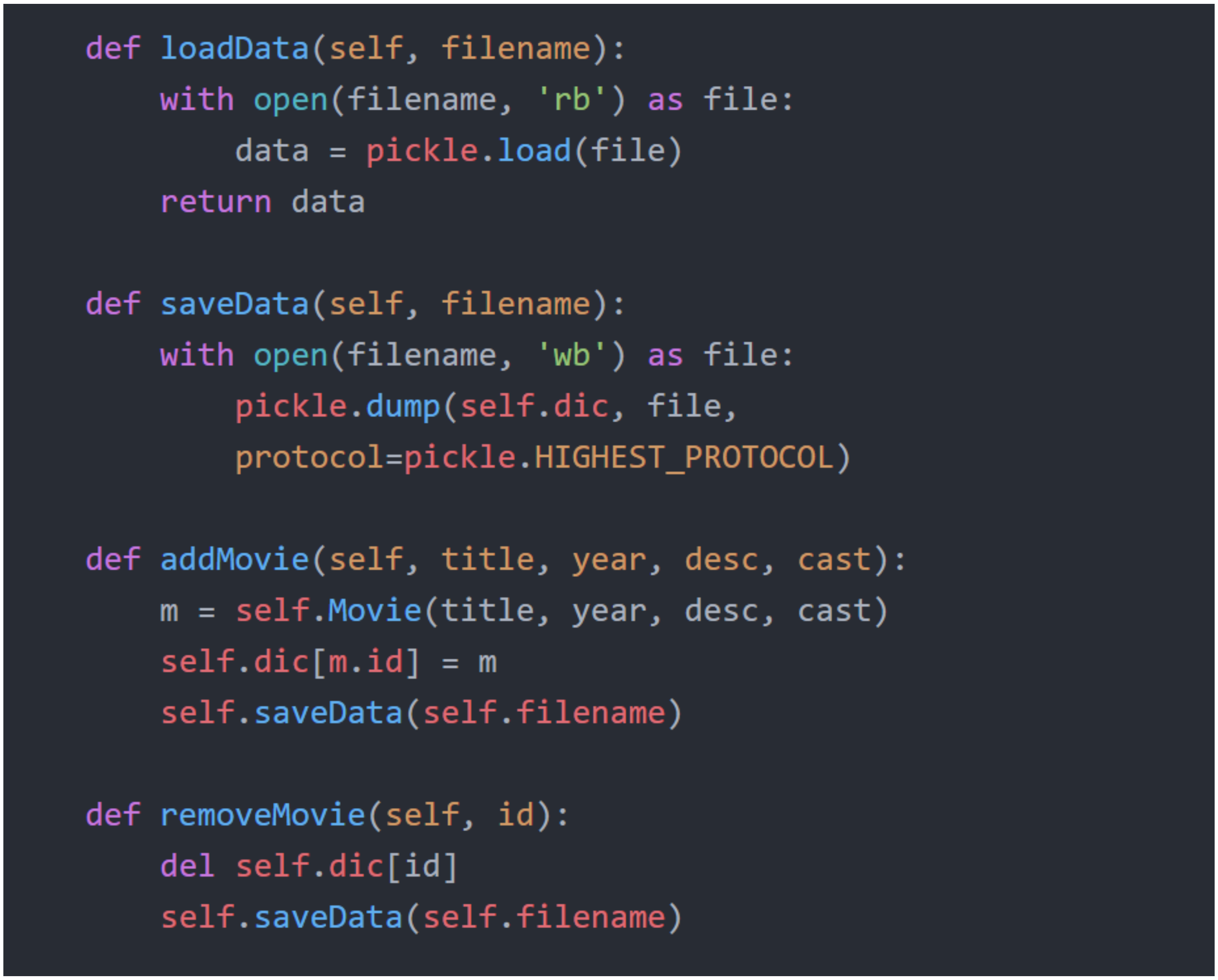
Removed the search bar on the right, because I need to decide on functionality it supports.

2. My plan was to make a page that would allow a user to update the information for a movie. But I ran into the problem of figuring out where to put that feature. At first I thought I'd create a button for updating a movie linked from an individual movie page, but then I realized that a supposed admin would have to search to find the movie, navigate to the movie page, and then navigate to an update page.

   I solved it by removing the subgoal for updating a movie's information at the moment. My plan is to think of better admin UX. For example, maybe an admin can see an edit feature on the movie's detailed page and a modal form pops up to allow an admin to edit a movie's information.

3. My plan was to store movie data hardcoded into a Python dictionary. But then I ran into the problem of persisting the data so that if a new movie was added, the dictionary would update and the new movie would persist. Or there is a admin mass-edit page where there's a list of movies that can be updated and deleted. Or maybe the edit feature should appear in multiple places.

   I solved it by using the `pickle` module in Python to persist a dictionary to a pkl file. This data file is loaded each time the application starts up, and each update to the database triggers a corresponding pickle dump to the data file. Still not the best way to handle data management, but it serves as a mock until I replace it with a database.



```
    def loadData(self, filename):
        with open(filename, 'rb') as file:
            data = pickle.load(file)
        return data

    def saveData(self, filename):
        with open(filename, 'wb') as file:
            pickle.dump(self.dic, file,
            protocol=pickle.HIGHEST_PROTOCOL)

    def addMovie(self, title, year, desc, cast):
        m = self.Movie(title, year, desc, cast)
        self.dic[m.id] = m
        self.saveData(self.filename)

    def removeMovie(self, id):
        del self.dic[id]
        self.saveData(self.filename)
```

Code for loading and saving a dictionary using the pickle module

Goal Progress

High Level Goal

1. Allow users to look up description and main actors in a movie
   Accomplished!
   - Created a base template, main and detailed movie page templates.
   - Populated the data store with some samples.
   - Created a page for adding movies through the GUI.
   - Added a `delete` feature to movie page.
   - See low level goals for details.
   - Project repo: https://github.com/ivychen/COMS6998-AdvWebStudio

Low Level Goal

☑ ~~**Install Flask and flask-bootstrap**~~

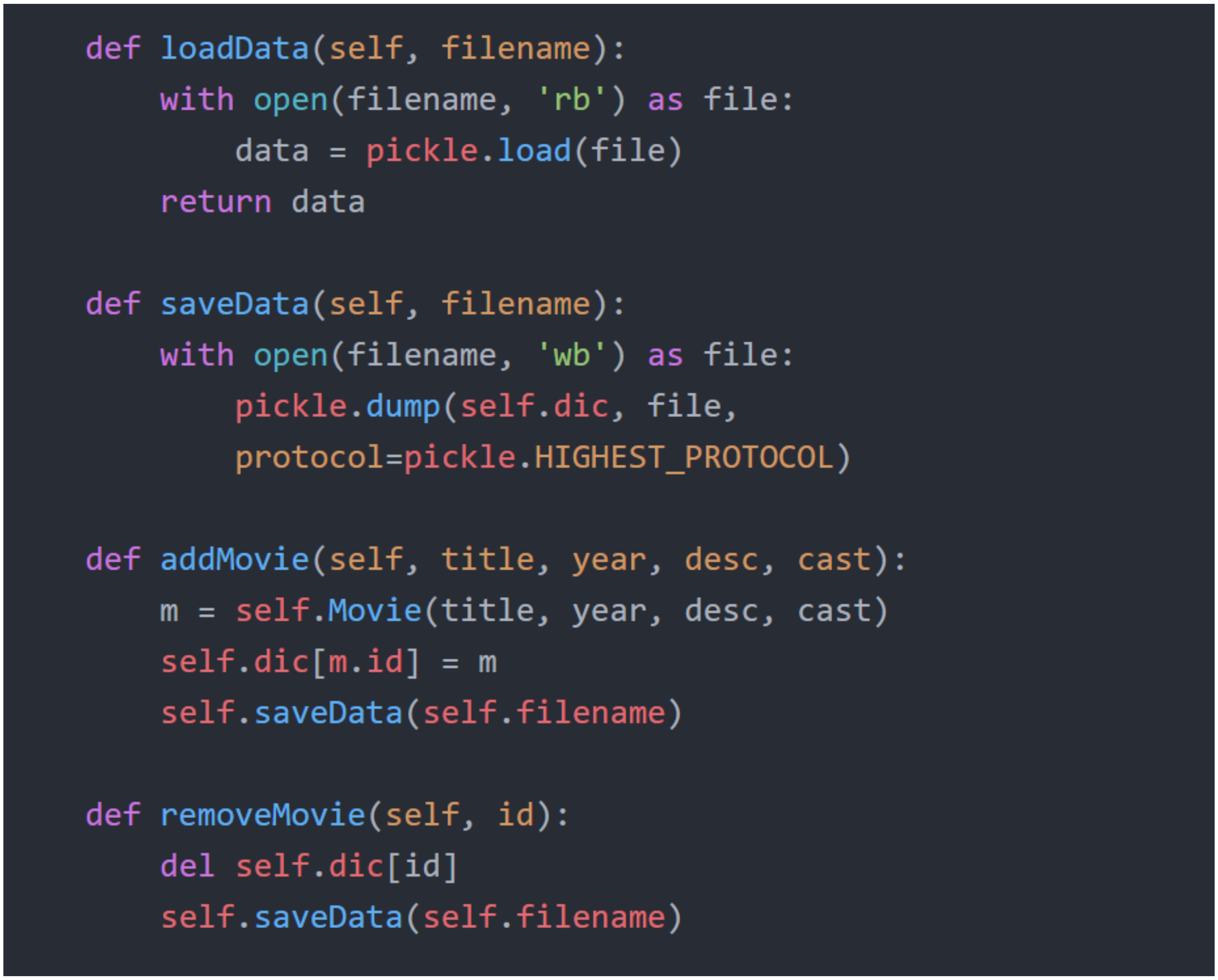
```
// requirements.txt
click==6.7
dominate==2.3.1
Flask==1.0.2
Flask-Bootstrap==3.3.7.1
flask-nav==0.6
itsdangerous==0.24
Jinja2==2.10
MarkupSafe==1.0
visitor==0.1.3
Werkzeug==0.14.1
```

- ☑ ~~Make two different web pages~~

  Successfully made the following pages:

```
addMovie.html
base.html
main.html
movie.html
```

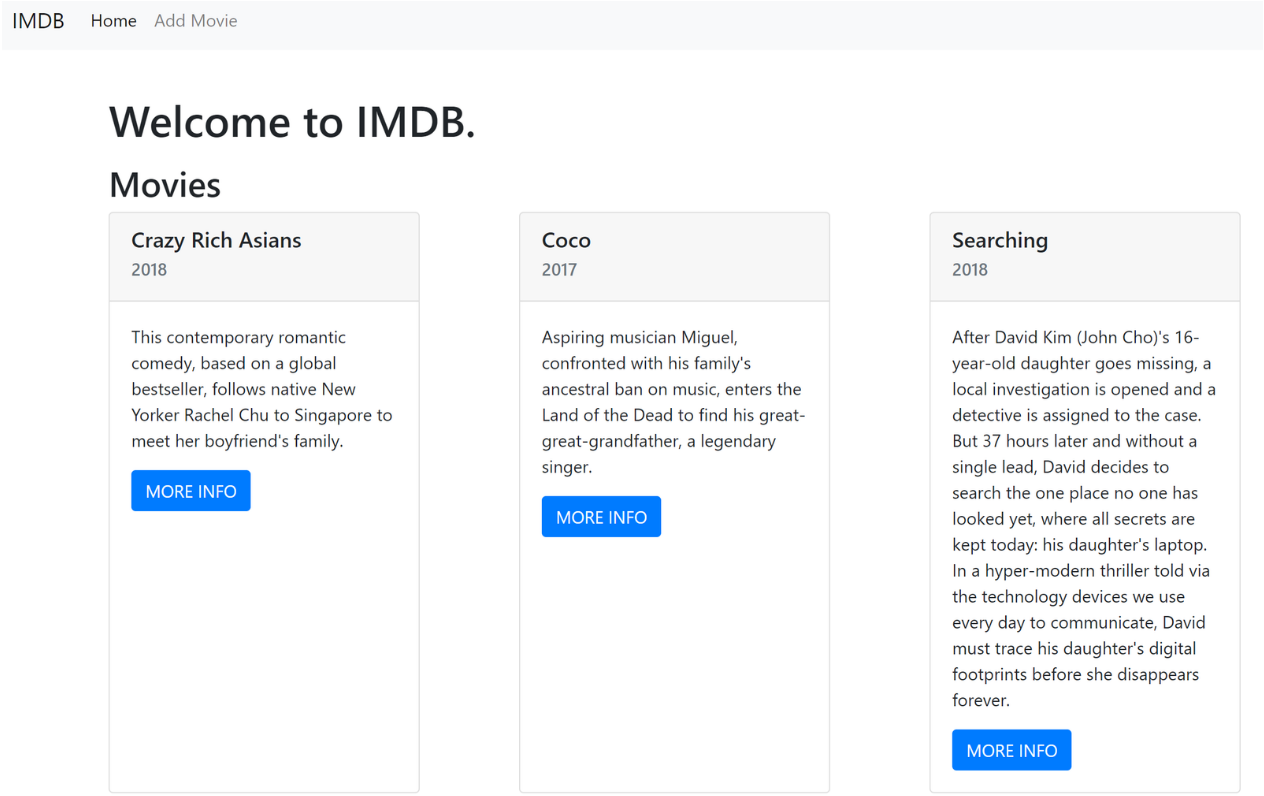
- ☑ ~~Figure out a simple way to persistently store movie data without using a database~~



```
    def loadData(self, filename):
        with open(filename, 'rb') as file:
            data = pickle.load(file)
        return data

    def saveData(self, filename):
        with open(filename, 'wb') as file:
            pickle.dump(self.dic, file,
            protocol=pickle.HIGHEST_PROTOCOL)

    def addMovie(self, title, year, desc, cast):
        m = self.Movie(title, year, desc, cast)
        self.dic[m.id] = m
        self.saveData(self.filename)

    def removeMovie(self, id):
        del self.dic[id]
        self.saveData(self.filename)
```

Code for loading and saving a dictionary using the pickle module

☑ ~~Page 1 — main page lists movies~~



Welcome to IMDB.

Movies

Crazy Rich Asians

2018

This contemporary romantic comedy, based on a global bestseller, follows native New Yorker Rachel Chu to Singapore to meet her boyfriend's family.

Coco

2017

Aspiring musician Miguel, confronted with his family's ancestral ban on music, enters the Land of the Dead to find his great-great-grandfather, a legendary singer.

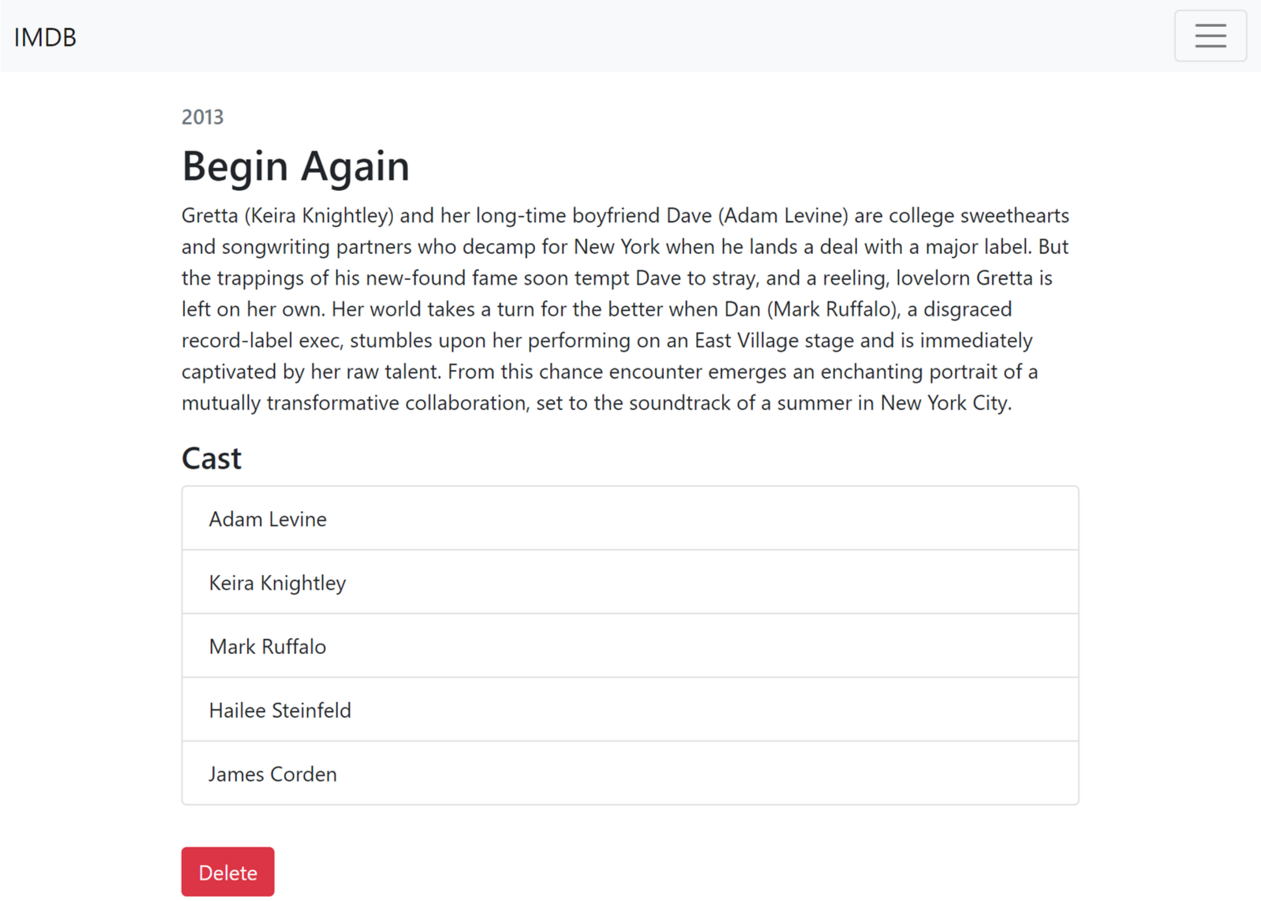
Searching

2018

After David Kim (John Cho)'s 16-year-old daughter goes missing, a local investigation is opened and a detective is assigned to the case. But 37 hours later and without a single lead, David decides to search the one place no one has looked yet, where all secrets are kept today: his daughter's laptop. In a hyper-modern thriller told via the technology devices we use every day to communicate, David must trace his daughter's digital footprints before she disappears forever.

Page 1: Main index page

☑ ~~Page 2 — see details of one movie~~



2013

Begin Again

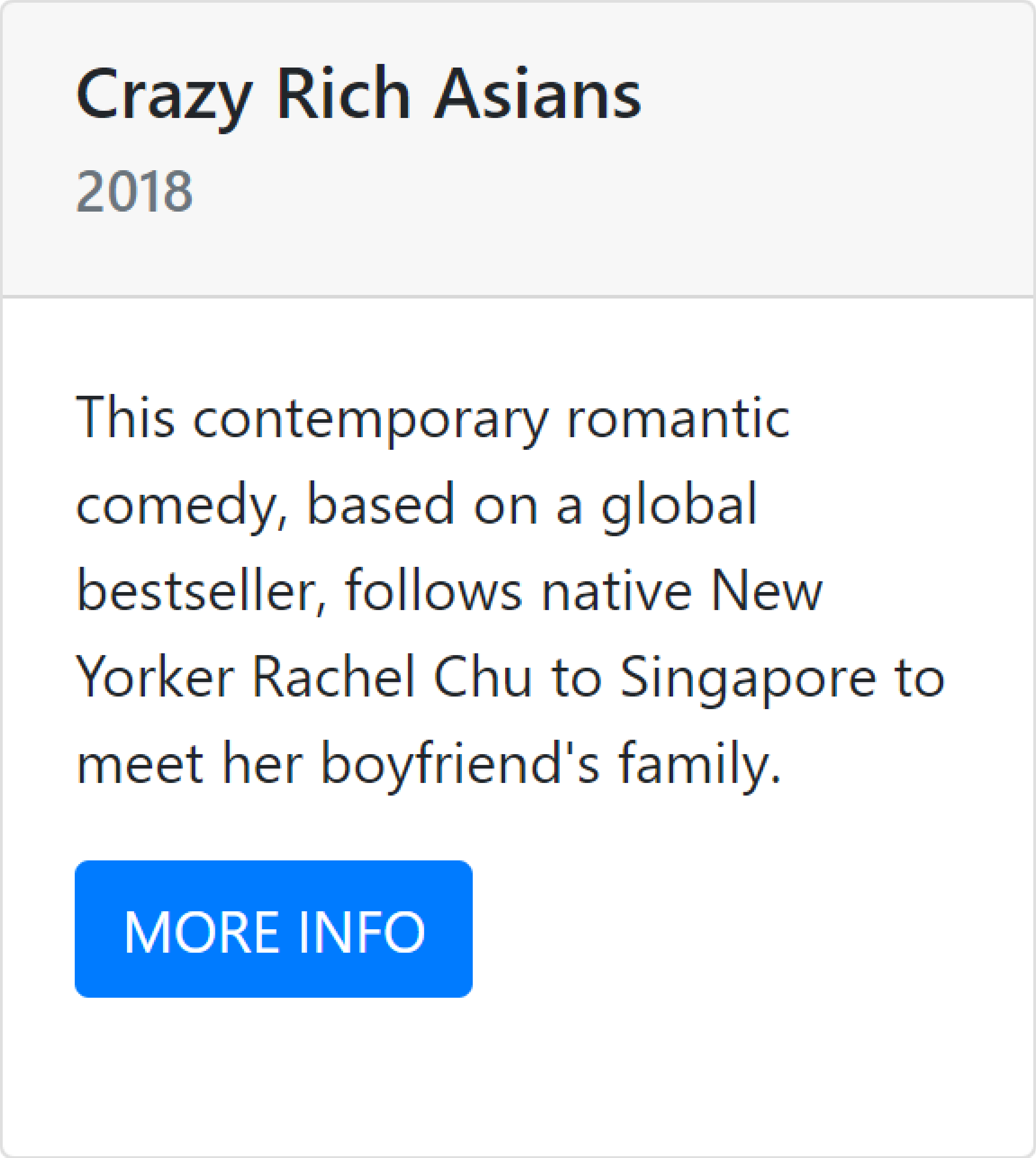
Gretta (Keira Knightley) and her long-time boyfriend Dave (Adam Levine) are college sweethearts and songwriting partners who decamp for New York when he lands a deal with a major label. But the trappings of his new-found fame soon tempt Dave to stray, and a reeling, lovelorn Gretta is left on her own. Her world takes a turn for the better when Dan (Mark Ruffalo), a disgraced record-label exec, stumbles upon her performing on an East Village stage and is immediately captivated by her raw talent. From this chance encounter emerges an enchanting portrait of a mutually transformative collaboration, set to the soundtrack of a summer in New York City.

Cast

- Adam Levine
- Keira Knightley
- Mark Ruffalo
- Hailee Steinfeld
- James Corden

Page 2: Movie details page

☑ ~~Make a route such that when you click on a movie on Page 1, it takes you to individual movie page~~



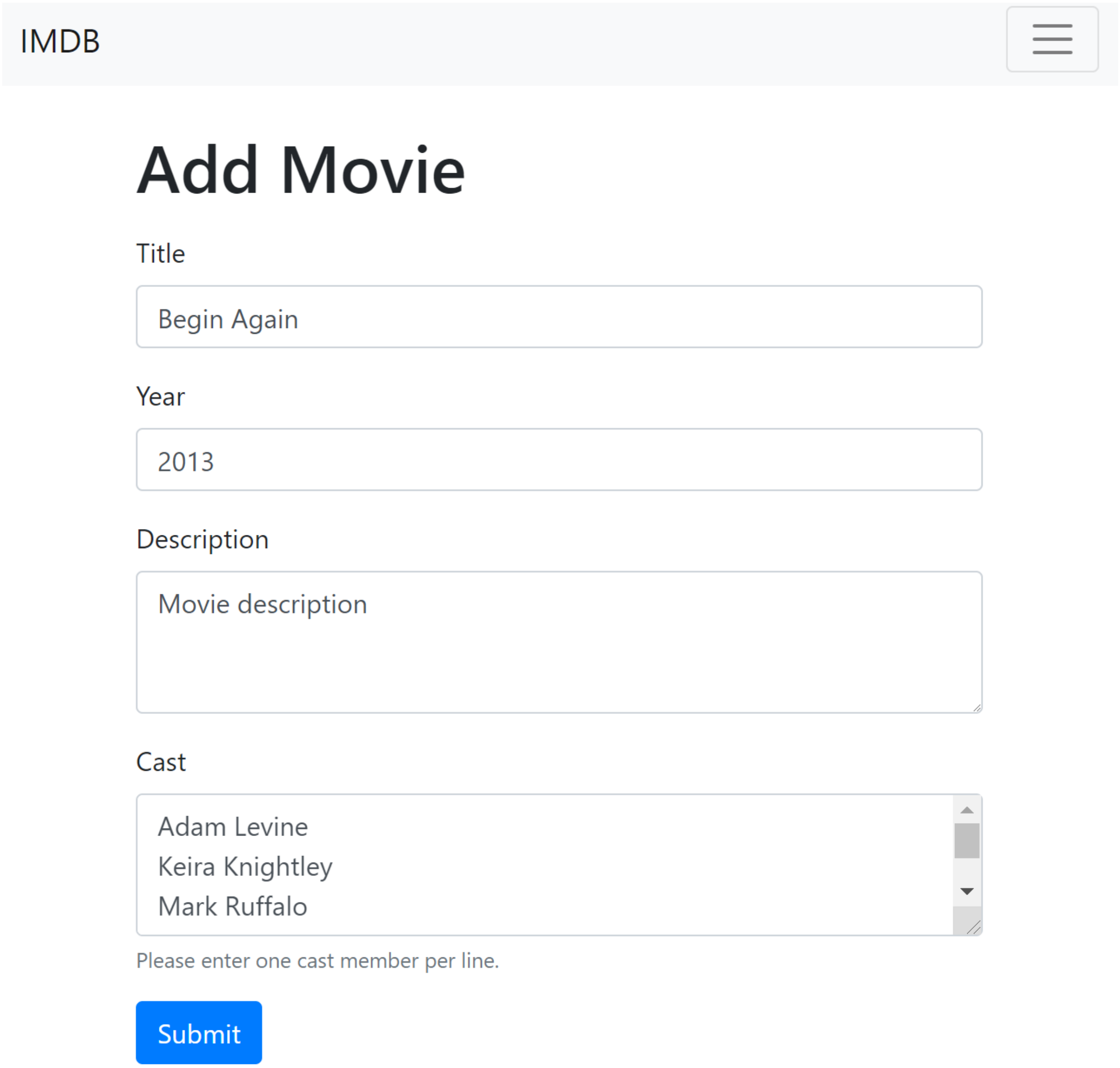
Crazy Rich Asians

2018

This contemporary romantic comedy, based on a global bestseller, follows native New Yorker Rachel Chu to Singapore to meet her boyfriend's family.

Each movie card on the main page has a "MORE INFO" link to the movie's detailed page

☑ ~~**Make a page to create a new movie**~~



IMDB

Add Movie

Title

Begin Again

Year

2013

Description

Movie description

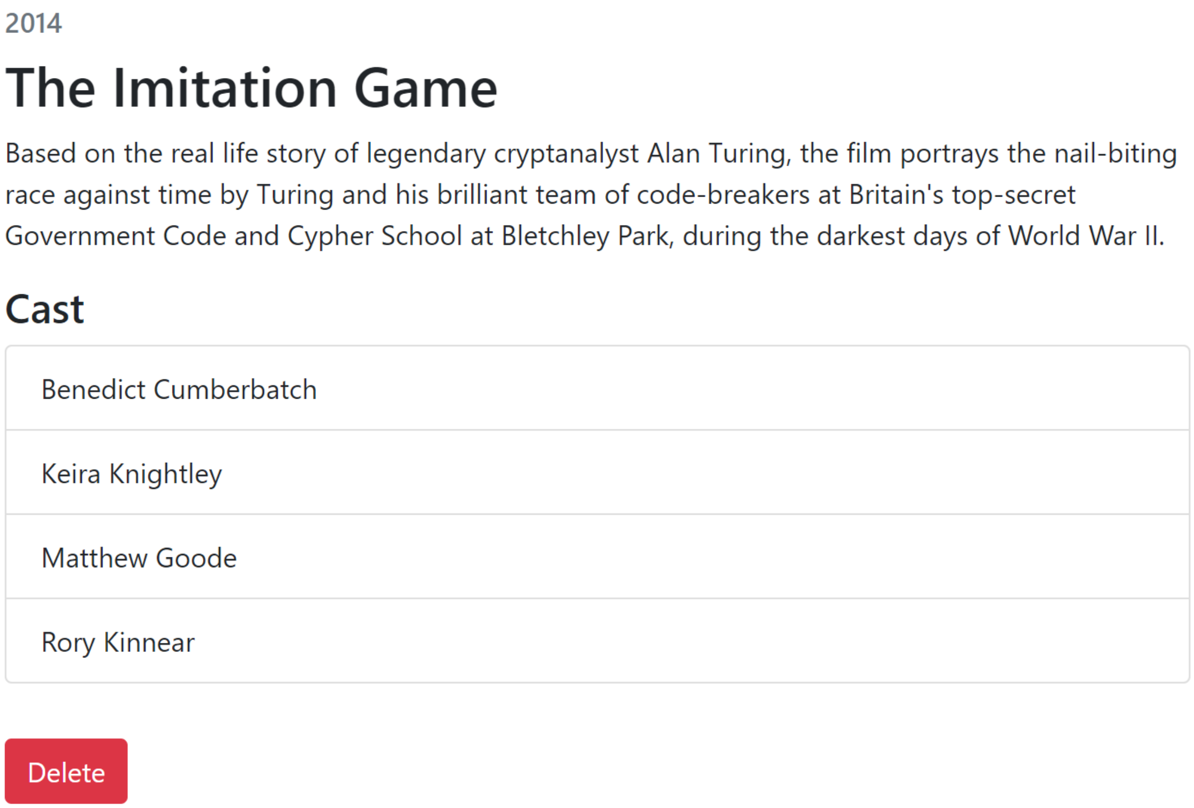
Cast

Adam Levine
Keira Knightley
Mark Ruffalo

Please enter one cast member per line.

□ **Make a page to update a movie's information [DEFERRED]**

☑ ~~**Make a button to delete a movie on movie detail page**~~



2014

The Imitation Game

Based on the real life story of legendary cryptanalyst Alan Turing, the film portrays the nail-biting race against time by Turing and his brilliant team of code-breakers at Britain's top-secret Government Code and Cypher School at Bletchley Park, during the darkest days of World War II.

Cast

Benedict Cumberbatch

Keira Knightley

Matthew Goode

Rory Kinnear

Each detailed movie page has a “DELETE” button on the bottom of the page

---

Be prepared to discuss your progress during studio. You will be graded on whether or not you did each part.