Picture Perfect

Picture Perfect is an online movie ticket booking, review and rating service. The service helps users generate review and rating content for movies across the world. One may wish to publish only a rating or a full review of the movie in a language of choice. The service comprises of the following client components

- Desktop website
- Desktop Web-based administration and management console for backend operations
- Public REST API for integration with other movie and media portals

PicturePerfect Backend

The backend design of the system is composed of several microservices, please use the schema below to come up with a detailed design of the data model for the services. Implement the microservices in a language of choice

- **Catalogue** - This service is to retrieve and maintain the catalogue of movies, documentaries and television programs.
    - Typical user operations would include (both logged in or an unauthenticated user)
        - GET /movies/catalogue - Get a paginated list of movies, along with the associated media (links to the thumbnail pictures)
        - GET /movies/catalogue/{name} - Get a movie/documentary by name with detailed info and the media links images, videos
        - GET /movies/catalogue?{query} - Search a movie with filter and sort criteria
            - Filter - could be on any attribute name, language,

            - Sort - Sort the results in ascending or descending order
            - Paginate - To paginate the results to obtain the results in chunks
    - Typical backend admin operations would include
        - POST /movies/catalogue - Add a new item to the catalogue
        - PUT /movies/catalogue - Update an item in the catalogue
        - PATCH /movies/catalogue - Update a specific attribute to an item in the catalogue

- **IAM – Identity and Access management -** this is to authenticate a user, and identify if the user is general user or somebody who can manage the PicturePerfect operations based on a role and privilege
    - A generic user role - Should not have access to the backend console but only to the PicturePerfect website
        - POST /login - create a new token for a login session
        - POST /logout - Invalidate the session and logout
        - POST /reset - Reset the password to a new one
    - An admin user role - Should have access to both the backend console and the PicturePerfect website
        - POST /login - create a new token for a login session
        - POST /logout - Invalidate the session and logout
        - POST /reset - Reset the password to a new one

- **Ratings** - This service lets users add, update or delete an existing rating for a movie
    - Typical user operations
        - PUT /movies/rating/{movie} - Adds a new rating for a movie
        - DELETE /movies/rating/{movie} - Delete the rating for a movie added by the user
    - Typical admin operations
        - DELETE /movies/rating/{movie} - Deletes all the ratings applied to a movie
        - DELETE /movies/rating/{user} - Deletes all ratings set by a user
        - DELETE /movies/rating/{movie}/{user} - Delete the rating for a movie added by a user

- **Reviews** - This service lets users add or update an existing review for a movie
        - PUT /movies/review/{movie} - Add or update a new movie review
        - DELETE /movies/review/{movie} - Delete a movie review created by the user

- **Shows** - This service lists the cineplexes where the movie is being screened in a given city
    - User operations
        - GET /movies/shows/{city} - List all shows in all cineplexes in a city

            - This should have the ability to filter, paginate and sort the results
        - GET /movies/shows/{city}/{movie} - List the cineplexes screening a particular movie
    - Admin operations - In addition to the user operations above, admins can do the following
        - POST - /movies/shows - Add a new show or add a new cineplex
        - PUT - /movies/shows - Update the show timings
        - DELETE - /movies/shows - Delete a show
        - DELETE - /movies/shows/{movie} - Remove a movie from all screens

PicturePerfect UI

- Responsive UI that adapts to different screen sizes (mobile, tablet, desktop)
- Home page - Listing of top movies across different categories
- Login page
- Simple and easy navigation, via categories menu and bread crumbs
- Search and listing of results, along with filters and pagination of results
- Ratings component - Adding and updating movie ratings from a rating widget
- Reviews component - Adding and updating reviews from a review widget
- i18n enabled – The reviews can be in any language

Come up with a visual design of the UI and each of these components, apply the right design principles to ensure ease of use, richness, extension and adoption. Implement the UI with the following UI framework components

- ReactJS
- Redux
- TypeScript

PicturePerfect Backend Admin Console UI

- Adding/Removing/Disabling cineplexes
- Updating movie catalogue
- ~~Updating movie ratings~~
- ~~Updating movie reviews~~
- ~~Adding/Updating/Disabling promos and vouchers~~
- ~~Garbage collector ensures invalid, biased and reviews by automated bots are cleared up.~~

PicturePerfect Automation

- Implement a thorough unit-test suite

- Implement an API test suite for all the microservices
- Implement a UI test suite using Selenium WebDriver and to test multiple browsers

PicturePerfect CI/CD

- Implement a Jenkins job for running unit tests for each commit
- Implement a Jenkins pipeline for functional testing
- Implement a Jenkins pipeline for deployment
    - Deploy or upgrade microservices
    - Deploy or upgrade UI

PicturePerfect Non Functional Requirements

- Highly available
- Concurrent users simultaneously adding reviews and ratings
- Secure

Milestones

Tentative timeline	Topic
End of Sept ‘19	Comprehensive design document incorporating the following points. Get it reviewed with the assigned mentors (see Collaboration section for details) Technology Choices AWS Services to be used Deployment strategy - CI/CD System Design Diagram, Interaction between components UX - Layout mocks High-level test strategy
End of Oct ‘19	Minimum 1 user flow(ex: Catalogue functionality for user and admin) end to end implemented Flow completion includes Good looking UI with CSS REST API definitions and implementation DB Schema Frameworks in place, including build & deployment Frameworks include UI, Backend, Testing

End of Nov ‘19	2 more flows using the above framework Tweaking the framework to fit other flows
End of Dec ‘19	Remaining flows Testing Fit and finish

Collaboration

We expect a two-way dialogue as you go on to build this service. In order to enable collaboration, we have mentors assigned to you for each of the components.

	Backend	**UI**	**Automation**
Myna	BabuSrithar	Vivek	Rohan & Manish
Neha	Bhaskar	Sanchit	Manish & Rohan
Rakshit	Prakash	Baraneedharan	Rohan & Manish
Sharan	Pritesh	Shriharsha	Manish & Rohan
Vivek	Satyam	Shriharsha	Manish & Rohan