<u>Progress Report 1 (PR1):</u>

SERVICES OFFERED

ID	**Service Title**	**Service Description (What & Why)**
S1	Movie Search	Create a movie search API that integrates OMDb to handle queries with filtering and pagination.
S2	Rating Summary	Build a movie ratings aggregation API that fetches, normalizes, and caches multi-source ratings with batch support.
S3	Genre	Build a genre-based movie discovery API with filtering, pagination, and caching.
S4	Top Box Offices	Build a movie analytics dashboard with search, box office rankings, aggregated ratings, and recommendations.
S5	Similar Movies	Build a movie recommendation API that suggests similar films using OMDb metadata matching with fallback logic.

SERVICES REQUIREMENTS

S1: Movie Search Service

Functional Requirements (FR)

Requirement Description	**Why**
Accept movie title or keyword as input and return basic movie detail	Core feature of movie lookup
Support partial or fuzzy word matching	Enhances usability when user only knows partial name
Provide advanced search filters such as (year, language, type)	Enables more refined search to narrow down query
If multiple matches, return paginated result with a	Improves readability and performance without

total count	having to load all at once
If no result, return JSON with empty array and message explaining	Prevents confusion of the software working or not
Integrate the OMDb API for backend data access	Dependency to access movie data
Expose GET /search endpoint with query parameters	Clear API structure

Quality Requirements (QR)

Requirement Description	**Why**
Latency: Average response latency < 500ms for standard searches.	Smooth user experience
Throughput: Handle ≥ 30 requests/second sustained load..	Supports concurrent users.
Availability: 99.0% during demo and testing	Reliability expectation.
Results ranked by best match score	Improves relevance and discoverability
Input sanitization and API key masking for security.	Prevent injection and exposure risks.
JSON logs include request ID and query string.	Supports debugging and monitoring.

S2: Rating Summary Service

Functional Requirements (FR)

Requirement Description	**Why**
Given a movie ID or title, fetch its IMDb, Rotten Tomatoes, and Metacritic ratings via OMDb API.	Central feature for user insight.
Compute average normalized rating (0–100).	Consistent comparison across rating sources.
Return JSON with breakdown of each source and computed average.	Provides transparency in rating logic.
Support batch requests for multiple titles.	Efficient for dashboards or lists.
Cache results for 10 minutes to reduce external API calls.	Improves performance and API quota usage.

Quality Requirements (QR)

Requirement Description	**Why**

Accuracy within ±1% of source ratings.	Ensures trustworthiness.
Response latency ≤ 400ms per single movie.	Quick insight delivery.
Throughput ≥ 20 requests/second sustained.	Scalability during load.
Display source unavailability gracefully with placeholders.	Prevents crashes on partial data.
Structured error handling for API timeouts.	Improves resilience.
Log rating discrepancies >5% across sources.	Detects anomalies and source drift.

S3: Genre Service

Functional Requirements (FR)

Requirement Description	**Why**
Accept movie title or IMDb ID, return list of genres associated with that film.	Core function for category exploration.
Support query /genre/<genre_name> to list all movies within a genre.	Enables genre-based browsing.
Provide optional filters by year, rating threshold, or language.	Improves discovery experience.
Return total movie count and pagination metadata.	Supports UI rendering.
Cross-reference with cached results to reduce repeated API hits.	Performance optimization.

Quality Requirements (QR)

Requirement Description	**Why**
Latency p95 < 600ms for single-genre queries.	User responsiveness.
Accuracy: genre tags must match OMDb data exactly.	Data consistency.
Handle ≥ 25 concurrent requests.	Scalability benchmark
Cache hit ratio >= 60% for repeated queries	Shows optimization
Log most frequently queried genres weekly	Supports analytics and UI tuning

S4: Top Box Office Service

Functional Requirements (FR)

Requirement Description	**Why**
Accept movie title/keyword and return matching list with ID, title, year, and poster	Keyword Search
Fetch and display key financial metrics (e.g., weekend gross, total revenue) for a given movie.	Box Office Data
Retrieve and display ratings from IMDb, Rotten Tomatoes, and Metacritic via OMDb API.	Aggregated Ratings
Generate and display real-time charts (e.g., Top 10 weekly or yearly)	Box Office Rankings
Provide a list of recommended movies based on movie type/director	Recommend for top box movie
Generate and return a ranking list based on the internal database, including cumulative box office.	Intuitively display hot trends in the film market

Quality Requirements (QR)

Requirement Description	**Why**
Average API response time < 500ms for core searches.	Low Latency
Sustain a load of ≥ 30 requests/second	High Throughput
Protect user data and API keys	Data Security
99.0% uptime for core services during demo/testing	High Availability
Box office data updates within 30 minutes of source update	Data Freshness
Degradation if external APIs (e.g., OMDb) fail	Error Handling
Support ≥ 500 simultaneous active users	Concurrent Users

S5: Similar Movies Service

Functional Requirements (FR)

Requirement Description	**Why**
Given a movie title or IMDb ID, fetch a list of similar or related movies using the OMDb API and/or genre and keyword matching.	Core functionality to enhance content discovery.

Use metadata (genre, director, actors, keywords) to determine similarity when API does not provide a direct "similar" endpoint.	Provides consistent fallback logic for broader coverage.
Return a list of up to 10 movies by similarity in genre.	Keeps results relevant.
Include basic details (title, year, poster, short plot) for each suggested movie.	Improves context and browsing experience for users.
If no similar movies are found, return an empty array with an explanatory message. (No similar movies were found).	Ensures user clarity and consistent API behavior.

Quality Requirements (QR)

Requirement Description	**Why**
Average response latency ≤ 500ms for single title queries.	Maintains user experience fluidity.
Similarity scoring must use consistent weighting (e.g., 40% genre, 30% cast, 30% keywords).	Ensures predictable, reproducible results.
Throughput ≥ 25 requests/second sustained.	Supports scalability during high traffic.
Cache hit ratio >= 60% for repeated queries	Shows optimization
Log most frequently queried genres weekly	Supports analytics and UI tuning
Display source unavailability gracefully with placeholders.	Prevents crashes on partial data.
Structured error handling for API timeouts.	Improves resilience.

PROJECT PLAN

- **Timeline: Please reference the excel file**
- **Tasks: Please reference the excel file**

CLOUD PLATFORM (IBM CLOUD OR OTHER)

- Our group has limited prior experience with IBM Cloud, though some members have used AWS and Google Cloud, giving us a strong foundation in cloud deployment concepts. We will apply that knowledge to learn IBM Cloud's tools for containerization, hosting, and deployment as we build the system.

MICROSERVICE ARCHITECTURE

- We have moderate familiarity with microservice architecture, having implemented small-scale services in previous coursework and projects. While this project will expand our hands-on experience, we understand the key principles of modularity, scalability, and inter-service communication.