Based on the provided document, here is a detailed, step-by-step report on the construction of the Movie Similarity and Recommendation System. The process moves from data ingestion and cleaning to advanced Natural Language Processing (NLP), feature engineering, and finally, visualization.

Phase 1: Environment Setup and Data Ingestion

The first stage involved setting up the Python environment and loading the raw datasets.

- **Library Installation:** The system began by installing necessary libraries for data manipulation (pandas, numpy), NLP (gensim, nltk), machine learning (scikit-learn), and visualization (matplotlib, seaborn)[1].
- **Data Loading:** Two specific datasets were loaded:
    1. movies_raw: Contains metadata like titles, genres, and release dates[2].
    2. reviews_raw: Contains critic reviews and scores[3].
- **Initial Inspection:** The shape of the data was verified. The movies dataset contained 17,712 rows, and the reviews dataset contained 387,508 rows[4].

Phase 2: Movie Metadata Processing

The system cleaned the movie-level data to extract useful features for later comparison.

- **Column Selection:** Only relevant columns were retained, specifically: rotten_tomatoes_link, movie_title, content_rating, movie_info, genres, and original_release_date[5].
- **Date Engineering:** The original_release_date was converted to a datetime object to extract the numeric year as a distinct feature[6].

Phase 3: Review Data Cleaning and Standardization

Because the raw reviews contained unstructured text and inconsistent scoring formats, significant preprocessing was required.

- **Score Conversion:** The review_score column was normalized. The code handled mixed formats, converting fractions (e.g., "3/5") into floats and handling standard numeric strings, while setting invalid entries to NaN [7].
- **Text Cleaning:** A clean_text function was implemented to process the review_content.
    - It converted text to lowercase[8].
    - It removed HTML tags and non-alphanumeric characters[9].

    - It removed "stopwords" (common words like "the", "is", "in") using the NLTK library to focus on meaningful keywords[10101010].
- **Filtering:** Rows with empty reviews after cleaning were dropped[11].

Phase 4: Topic Modeling (LDA)

To understand the *thematic* content of the movies based on what critics wrote, the system utilized Latent Dirichlet Allocation (LDA).

- **Tokenization & Corpus Building:** The cleaned reviews were tokenized (split into words)[12]. A dictionary was built, filtering out extreme outliers (words appearing in fewer than 10 documents or more than 50% of documents)[13]. A Bag-of-Words (BoW) corpus was created[14].
- **Model Training:** An LDA model was trained to identify **10 distinct topics** within the reviews[15].
- **Topic Analysis:** The system identified keywords associated with specific themes. for example:
    - *Topic 4* related to horror (words: "ghost", "horror", "monsters")[16].
    - *Topic 6* related to comedy (words: "funny", "comedy", "hilarious")[17].
- **Vectorization:** Each review was assigned a probability vector indicating how strongly it belonged to each of the 10 topics[18].

Phase 5: Feature Engineering and Aggregation

The system then merged the disparate data sources into a single "Master" feature matrix (X) where every row represented a unique movie.

- **Aggregating Topics to Movies:** Since there were multiple reviews per movie, the system grouped the review data by rotten_tomatoes_link and calculated the **mean** topic vector and mean review score for each movie[19].
- **Genre Encoding:** The genres column (which contained strings like "Action, Drama") was split and converted into numerical data using **Multi-Hot Encoding** (via MultiLabelBinarizer). This created a binary column for every genre (e.g., genre_Action, genre_Comedy)[20].
- **Merging:** The processed movie data (titles, years) was merged with the aggregated topic data and the encoded genre data[21].

- **Imputation and Scaling:**
  - Missing numeric values (for review_score and year) were filled with the column mean[22].
  - All numeric features were scaled using StandardScaler to ensure that features with large numbers (like Year) didn't dominate features with small numbers (like Topic Probabilities)[23].

Phase 6: Building the Recommendation System

With the final feature matrix constructed (X with shape 7216 movies $\times$ 33 features), the system implemented the similarity logic.

- **Cosine Similarity:** A function get_similar_movies was defined. It takes a movie title, finds its feature vector, and calculates the **Cosine Similarity** between that vector and every other movie in the database[24].
- **Ranking:** The results are sorted by similarity score (descending), and the top k matches are returned[25].
- **Testing:** The system tested this on the movie "Bright Eyes", successfully recommending similar films like "Captain January" (0.9647 similarity) and "Going My Way"[26].

Phase 7: Clustering and Profiling

The code included analytical steps to group movies and inspect individual profiles.

- **Topic Profiling:** A helper function get_movie_topic_profile was created to retrieve the specific topic distribution for a single movie (e.g., "Alien")[27].
- **K-Means Clustering:** The movies were grouped into 10 distinct clusters using the K-Means algorithm based on their feature vectors[28]. The system inspected "Cluster 0" to see which movies were grouped together[29].

Phase 8: Visualization

Finally, the system generated various plots to visualize the data distribution and relationships.

1. **Genre Distribution:** A bar chart showing the frequency of different genres (Drama and Comedy being the most common)[30].
2. **Review Score Distribution:** A histogram showing that average critic scores follow a roughly normal distribution[31].

3. **Score vs. Genre:** A boxplot analyzing how critic scores vary across different genres[32].
4. **Topic Composition:** A pie chart visualization showing the topic mix for the movie "Alien"[33].
5. **Similarity Bar Chart:** A visual ranking of the top 5 movies similar to "Alien" (e.g., "The Godfather Part II", "A Clockwork Orange")[34].
6. **Similarity Heatmap:** A heatmap displaying the pairwise similarity between a random sample of 30 movies to visualize how distinct or related the movies are in the vector space[35].