Algorithm for the code of Complaint Management System.

1. **Import Required Libraries**
    - Import necessary libraries for data processing, machine learning, and NLP (e.g., pandas, scikit-learn, gensim, nltk, numpy).
2. **Preprocessing**
    - **Download NLTK resources**: Download the stopwords and punkt tokenizer from the NLTK library.
    - **Load Dataset**: Read the complaint data from a CSV file (complaints.csv).
    - **Text Preprocessing Function**:
        - Convert text to lowercase.
        - Tokenize the text.
        - Remove stopwords from the tokenized words.
3. **Train Word2Vec Model**
    - Prepare a list of tokenized sentences from the complaint descriptions.
    - Train a Word2Vec model on the preprocessed sentences to generate word embeddings.
4. **Get Average Word Vector**
    - Create a function to compute the average Word2Vec vector for each complaint.
    - If the complaint has no words in the Word2Vec model, return a zero vector.
5. **Vectorize Complaint Data**
    - Apply the average vector function to all complaints to convert each complaint into a numerical vector.
6. **Split Data for Training and Testing**
    - Split the complaint dataset into training and testing sets using an 80/20 split ratio.
7. **Random Forest Classifier with Hyperparameter Tuning**
    - Define a hyperparameter grid for the Random Forest model (e.g., n_estimators, max_depth, etc.).
    - Use GridSearchCV to perform hyperparameter tuning and find the best model based on accuracy.
8. **Train the Model**
    - Fit the Random Forest model on the training data.
9. **Evaluate the Model**
    - Predict complaint categories for the test data using the best model.
    - Calculate accuracy and display the classification report.
10. **Classify New Complaints**
    - Create a function to classify new complaints by converting the input complaint to its average Word2Vec vector and predicting its category using the trained model.
11. **Recommend Categories**
    - Create a function to recommend similar categories for a new complaint based on cosine similarity with existing complaints.
12. **Main Program Loop**
    - Continuously prompt the user for complaint input.
    - For each user input:
        - Recommend categories based on the input.
        - Classify the input into a predicted category using the Random Forest model.
        - Print the recommendations and the predicted category.
    - Exit the system when the user types 'exit'.