**Requirements Definition**

**Functional Requirements**

1. User Authentication and Access Controls
   1. The system must require all users to initially create an account, or sign in with the username and password created by the admin.
      1. On first login, the user must be able to create an account by entering a username and password.
      2. For any subsequent login, the user must be able to enter their username and password.
         1. If entered correctly, the system allows the user access to features they are authorized to view.
         2. If entered incorrectly, the system does not allow the user access and allows the user to attempt to sign in again.
   2. Users with Admin rights will have access to certain features.
      1. Able to manage all users accounts.
      2. Able to manage all movies.
      3. Able to get review from users.
      4. Able to send emails to all users.
      5. Able to manage the database.
2. User Profile Management
   1. Each user is able to edit their own account.
      1. User will be able to login.
      2. User will be able to edit their own username.
      3. User will be able to edit their own password.
      4. User will be able to delete their own account.
      5. User will be able to logout.
   2. Users should be able to see their personal list of movies they have posted to lend and borrowed.
   3. Users without Admin rights should not be able to edit other users information.
3. User Actions- Lender
   1. Lenders should be able to post movies.
      1. Lenders should be able to manage the details about the movie.
         1. Lenders should able to set the search-ability level of the movie.
         2. Lenders should be able to add a description to each movie they post.
         3. Lenders should be able to edit the details of the posted movie.
         4. Lenders should be able to set the availability status of a movie.
   2. Lenders should be able to remove a posted movie.
   3. Lenders should be able to lend out a movie.
      1. Lenders should be able to receive a borrow request from borrowers.
      2. Lenders should be able to set conditions for movies to be borrowed.
      3. Lenders should be able to communicate/coordinate with the borrowers.
   4. Lenders should be able to view all movies posted for lending.
      1. Lenders should be able to search for movies.
      2. Lenders should be able to filter for available movies.
   5. Lenders should be able to see their personal list of movies they posted to lend.
      1. Lenders should be able to view the “contract” for each movie that they are lending.
4. User Actions- Borrower
   1. Borrowers should be able to view all movies available for lending.
      1. Borrowers should be able to search for movies.
      2. Borrowers should be able to filter for available movies.
   2. Borrowers should be able to borrow movies.
      1. Borrowers should be able to send a borrow request to lenders.
      2. Borrowers should be able to agree/disagree with the conditions set by the lender.
      3. Borrowers should be able to communicate/coordinate with the lenders.
   3. Borrowers should be able to view their personal list of movies they are borrowing.
      1. Borrowers should be able view the “contract” for each movie that they are borrowing.
5. System Actions
   1. The web server should correctly generate and display pages are they are requested to be navigated to.
   2. The web server should correctly interact with the database.
   3. The web server should be able to automatically send alerts to users.
      1. The web server should check for movies that need to be returned soon and send an email/notification to the borrower.
      2. The web server should check for movies that should be returned soon and send an email/notification to the lender.

**Non-functional Requirements**

1. The project can be run on a google cloud server.
2. Cloud server cost must not exceed the free credit given by google.
3. Team coding efforts will be coordinated using github projects features, including cards and issues.
4. No passwords will be directly stored in the database, they must be first hashed using an industry standard hash suitable for password storage.
5. Predesigned authentication tools supplied by the django framework will be used for authentication.
6. Import passwords for team accounts will not be shared over insecure means.
7. The database will be regularly backed up.