A. INTRODUCTION

1. Team Name: Filmaholics
2. Team Members: Kiran Datwani, Kha Bui, Nhan Bui, Maegan Chow
3. Application Title: Cinopsis

   Description: The movie theater is expensive so individuals with limited funds such as college students should not waste their money seeing bad films. This website will help college students get their limited money's worth. Users can log on and rate movies that they have seen. They can also check out reviews left by others about a movie that they want to see and decide if they want to spend money on that movie ticket or not. Admin are mainly there to ensure users comply with the terms of service and can view all reviews posted by all users.

   Functional Requirements Specification:

   - User login
   - Users can check reviews, add movies, add ratings and write reviews
   - Pick a movie from the movie page
   - Admins are able to view all reviews
   - Users can view what they have posted

4. Type of program e.g. Browser Extension, Web Application, ..etc

   Web Application

5. Development Tools

   Development language: JavaScript + HTML + CSS

   IDE: IntelliJ

B. REQUIREMENTS

1. **Security and Privacy Requirements:**
   - Establish and define your security and privacy requirements for your program.

        - The web application must keep the users' login information secure
        - The web application must keep the users' data private and only accessible by the user
    - Also, come up with a system of keeping track of security flaws that may arise throughout the development process. Describe this plan/system in your report.
        - The Github projects tab will be used to keep track of security flaws that may arise during the development phase. With Github we can have a list of issues that all the members on the team can see; the issues will be made into tasks that will be sorted by priority. Each team member will then be assigned to the tasks and as each task is solved and completed, it will be moved into the "Done" section and will let all team members know which tasks were completed. As more issues arise, they can be added on to the project board for future problem solving.

2. **<u>Quality Gates (or Bug Bars)</u>:**
    - Security:
        - Critical (A security vulnerability that would bring the highest amount of risk/damage)
            - Elevation of privilege (being able to execute arbitrary code or receiving more privilege than authorized)
        - Important (A security vulnerability that would bring a high amount of risk/damage, but less than Critical)
            - Information disclosure (an attacker can be able to locate and read information anywhere on the system, including information not meant to be seen)
            - Spoofing (an entity is disguised as another entity of their choosing to access certain parts of the system)
            - Tampering (Permanent modification of any user or system data in a default scenario which continues even after restarting application)

        - Moderate (A security vulnerability that would bring a moderate amount of risk/damage, but less than Important and more than Low)
            - Denial of service (easily exploited by a small amount of data or induced by a client)
            - Security assurances (security feature is implemented to provide security protection)
        - Low (A security vulnerability that would bring the lowest amount of risk/damage)
            - Information disclosure (random memory and/or sensitive data being exposed)
            - Tampering (Temporary modification of any user or system data in a specific scenario which discontinues after restarting application)
    - Privacy:
        - Critical (high risk for legal liability of application)
            - Lack of notice and consent (sensitive, personal data/information is transferred without the consent of the user)
            - Lack of data protection (no security to protect users' private information such as login credentials)
            - Insufficient legal controls (sensitive data is shared with third parties without consent from the user)
        - Important (high risk of damaging organizations/application's reputation)
            - Lack of data protection (no security to protect users' non-personal data/information such as movie reviews)
        - Moderate (minimum concerns will be raised, but the repercussions will not be as severe as Critical and Important)
            - Lack of user controls (non-essential anonymous data is collected and transferred with the user being unable to stop the collection and transfer)
            - Data minimization (anonymous data from users is transmitted to third parties)

        - Low (users may be displeased, but little to no repercussions will take place)
            - Unable to determine filter specific demographics such as age and gender
            - Unable to encrypt sensitive data

3. **Risk Assessment Plan for Security and Privacy:**
    - Access security and privacy:
        - Discussion of new features in the application, and its significance to security and privacy of the users
            - The type of features and how they will be stored
            - Determining if information is sensitive or can be kept anonymous
    - Threat Modeling:
        - Login (users and admins)
        - Users access to the movies (and ratings) database
    - Security Reviews:
        - Private Information (of users and admins)
        - Movie (and ratings) database

C. DESIGN

1. **Design Requirements:**

    1. User data must be protected

    To protect user data, all identifying information should be encrypted on the application at all times unless explicitly chosen to be revealed by the user. This could include users' sign-up emails, passwords, birthdays, gender, and location. Sensitive information such as passwords should never be transferred in the application without being encrypted. All user data that needs to be stored in the database should be encrypted as well. No user should be able to see another user's

personal information unless they have admin capabilities and the appropriate permissions. A change of user data should require the user to authenticate their identity using unmodified data that the user previously entered.

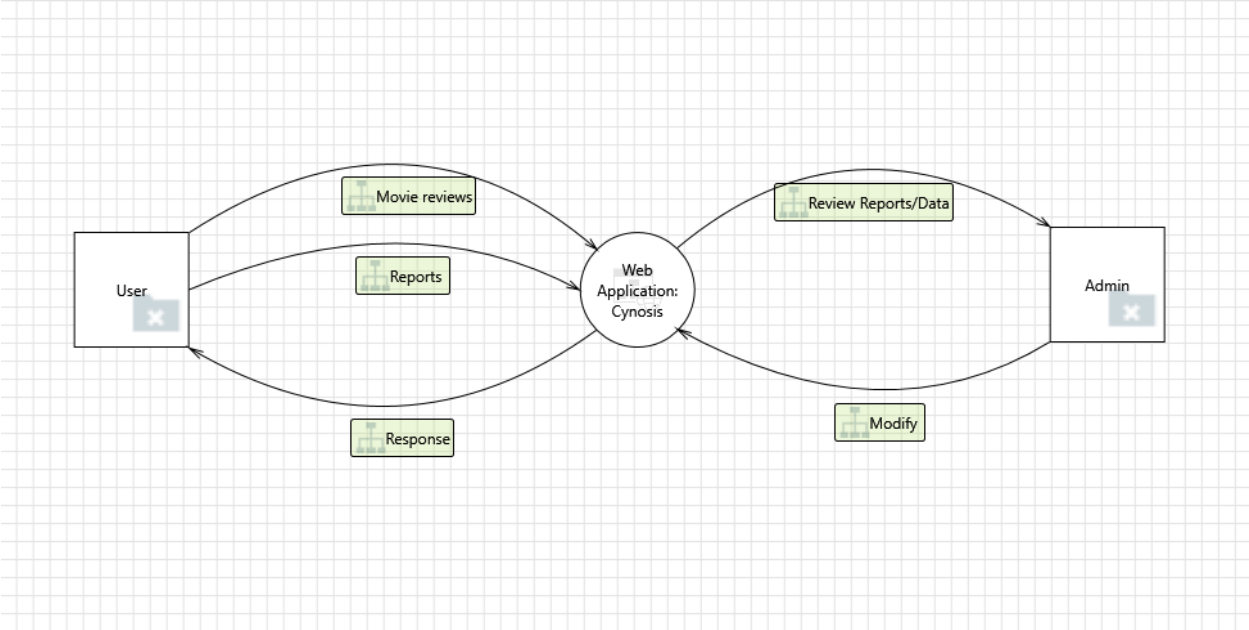
2. Users must not be able to be identified through program functionality

To preserve user confidentiality, users' contributions to the website are anonymous. Although users are able to view what they have posted, there are no usernames or other identifying information attached to the publicly accessible reviews so that a user's identity will never be threatened or revealed by a review that they submit

2. Attack Surface Analysis and Reduction:
    - Privilege Levels
        - User
            1. Can read reviews
            2. Can write reviews
            3. View their own data (reviews, login information)
        - Admin
            1. Can read all reviews
            2. Can change data shown on website as necessary
            3. Can add movies
    - List of vulnerabilities
        - Login access (as a user or admin)
        - Guest accounts (and their login access)
        - Personal information (private data is protected and unable to be shared with third parties)
        - Users and Guests (cannot directly access/edit reviews written by others and personal data by other users)

3. Threat Modeling:

- Create a threat model for your program.
  - You may use the SDL threat modeling tool, but you are not required to.
  - Your threat model should contain a graphic representation of all the information in your program and how it is all connected (a data flow diagram (DFD)).
  - Add privilege boundaries to your diagram and categorize your possible threats.
  - If the above guidelines for this section should feel confusing, i.e. regarding what is deliverable, just complete steps 1 and 3 here.



D. IMPLEMENTATION

1. Approved tools:
   -

Tool/Framework	Version
Digital Ocean	2021.1.2
Semantic UI React	7.7.2

Meteor	2.2
React	17.0.2
IntelliJ	2020.3.2
TestCafe	v1.14.0
JavaScript	ES6.ECMAScript 2018
HTML	5
CSS	2.1

2. Deprecated/Unsafe Functions:
    - As of version 2.0.0 the page grid functions of Semantic UI have been deprecated as it uses percentage gutters which made it unnecessarily difficult to style page contents. As an alternative, a ui container is a fixed width responsive container for holding page contents.
    - As of version ES6 of JavaScript, the Date.prototype.getYear() function has been deprecated due to the function failing to report full years when used for years greater than or equal to the year 2000. This function can be used to add a date and time tag to each review made by a user, so it is important to be aware of the updated syntax of getFullYear() to fully express the necessary years.
    - In HTML, it can be repetitive and confusing to add many stylistic components directly in HTML code, such as for the background color or a background image. An alternative would be to use CSS functions, and input direct elements such as h1, p, or title that would specify stylistic elements.

3. Static Analysis:

Tool	**Version**

eslint	v7.0.0

- ESLint (tool)
  - This is a static analysis tool for JavaScript that will be used to ensure all contributions meet a certain coding standard. ESLint is used to find potential issues that conflict with the code, in different levels of severity from warnings to errors.
- Experience
  - This tool helps find errors in the code, and is especially useful when the code is yet to run, which saves time so that the errors are found and fixed immediately and efficiently. Our team faced no difficulty combining and executing our code as we had no conflicting issues and maintained the same style using ESLint.

E. VERIFICATION

1. Dynamic Analysis:

   For our project we will be using Iroh.js. Iroh is a tool that patches your code to be able to record everything happening inside it, but without changing the original program.

2. Attack Review Surface:

   There have been no development changes, or updates in any of these tools. There have not been major vulnerabilities relating to these tools reported so we expected there to not be any changes given the time frame of this project.

3. Fuzz Testing:

We attempted to check and hack into our application through different methods to test our security. Firstly, we checked our login system multiple times, and made sure that a user wasn't able to login with the same email more than once. There was no reason for more than one person to have the same email, and it also helped the user remember that they had already created an account with our application. Secondly, we made sure that they cannot get into their account with an incorrect password. An incorrect password helps keep the application secure from malicious hackers trying to hack into private, personal accounts. We were completely successful for both login trials, and guaranteed that it was secure for our team and potentially future users. Thirdly, we attempted to access the app's pages while not logged in, such as the Movies page and the Reviews page, but instead of showing the pages (and private information of users and admins), we were sent back to the login page, and were asked for our email and password credentials. This ensures that people who don't have an account won't be able to access personal data. Lastly, we tested whether users and admins were able to get into each others' pages, but it was not possible for users to access admin data, and vice versa, it wasn't possible for admins to access user data. This helps our application remain safe and confidential for all users and admins that use the application.

4. Static Analysis Review:

While developing our app, we made sure ESLint was running in all our files, so that we were able to execute our code with as minimal to little errors as possible. We ran into common ESLint errors in the development process, such as having extra imported files at the beginning of files, or the opposite where we didn't have the correct files that are supposed to be imported into the files. We also kept in mind other styling errors that might occur, such as making sure to remove any extraneous functions imported from semantic ui react and having an extra line at the end of our code for each source file. We focused on following the coding standards and rules from ESLint along with maintaining the structure and organization of our code.

5. Dynamic Review:

   At the current stage in our app's development, we downloaded and utilized Iroh.js to troubleshoot our code. We were able to understand how our code was behaving during execution and if there were errors in our code during runtime. It was efficient and time-saving to be on the same page with all the group members when we focused on using Iroh.js to maintain our code and troubleshoot any current and potential problems.

F. INCIDENT RESPONSE PLAN

Incident Response Plan: A Privacy Escalation Team

- Escalation Manager: Kha will be the escalation manager. This position oversees the other three roles and makes sure communication flows smoothly.
- Legal Representative: Maegan will be the legal representative. This position needs to keep up to date on laws and regulations regarding data protection and security so that projects are always in compliance with the law and user concerns are able to be addressed.
- Security Engineer: Nhan will be the security engineer. This position will comprise testing the website and being in charge of recommendations for how to improve security.
- Public Relations Representative: Kiran will be the public relations representative. This position is in charge of monitoring the contact email and responding to those who do contact us if the email does not concern a true security problem. If a genuine issue is brought forth, she will loop in the escalation manager. She is also in charge of drafting statements to release if users take their issues public

Incident Response Plan: Incident resolution procedures

Contact Email: filmaholics@gmail.com

Privacy Escalation Response Process:

1. After we are notified via our contact email that there is an issue, Kiran brings in Kha to determine the gravity and nature of the issue presented

2. Kha gives the problem to Maegan, if related to a legal concern (such as user data usage); Nhan, if related to a security concern that needs to be patched out immediately; and/or Kiran, if related to an issue that could negatively impact the website's user base as a whole
3. If multiple people are needed to address the issue, Kha facilitates communication between them and encourages the path to a solution to follow a strict timeline
4. After a solution has been implemented, Kiran reaches out to the user who brought up the issue with the development and apologizes on behalf of Filmaholics

G. FINAL SECURITY REVIEW

- **Final Security Review Decision:**

  Passed FSR

- **Final Security Review Decision Explanation:**

  During the whole process of coding our website, we chose ESLint as our form of static analysis. We had ESLint running in the background the entire time we were coding to ensure that there would be no errors or to reduce as many errors as possible within our code.

  As for dynamic analysis, we ran Iroh.js to check our code and no significant errors were found.

  Users are unable to create multiple accounts with the same email, and admin pages are not accessible by non admin accounts. Therefore, we have decided to grade our application with a Passed FSR.

H. CERTIFIED RELEASE AND ARCHIVE REPORT

1. **Release Version:**

   Cinopsis V.1.00

2. **Summary of Features:**
    - Secure login functionality
    - Secured User/Admin accounts and personal data
    - Finding new movies and their reviews
    - Writing your own reviews

3. **Version Number**

V.1.00

4. **Future Development Plans**
    - Search bar functionality
    - Automatically updated ratings on each movie card
    - Delete review functionality
    - “Suggestion” box for suggestions of movie additions by users

5. **Technical Notes:**

    To Use:

    To use our program, our website has been deployed to an IP address that can be visited here.

    To Install and Run:

    1. Please visit our repository
    2. Download the master branch as a ZIP file
    3. In your terminal, change to the app directory
    4. Install meteor in the app directory by using the command meteor npm install

5. Start the program by using the command meteor npm run start
6. Visit the running program at http://localhost:3000