LA GRANDEE INTERNATIONAL COLLEGE

Simalchaur, Pokhara Nepal

Final Report Defence

On

“Pop Corn Box”

Submitted to:

Bachelor of Computer Application (BCA) Program

In partial fulfilment of the requirements for the degree of BCA under

Pokhara University

Submitted by:

Name:	Course	Semester	P.U. Registration Number
Krishna Gurung	BCA	6th	2021-1-53-0353
Prajal Gurung	BCA	6th	2021-1-53-0359

Date:22/02/2025

Acknowledgement

We would like to express our gratitude to our BCA coordinator Mr**. Kundan Chaudhari**, Project supervisor **Mr. Sunil Sapkota** and LA Grandee International College for their support and contributions to the development of Pop Corn Box.

This project is done for the in partial fulfilment of the requirements for BCA (Bachelor of Computer Application) program under Pokhara University. Our project was made possible by the effort and dedication of our team members. We thank our dedicated team for their hard work and contributions to the game. We are grateful for the guidance and mentorship provided by our respected sir **Mr. Sunil Sapkota.**

Sincerely,

Krishna Gurung

Prajal Gurung

Declaration for

"PopCornBox"

Student's Declaration

We, **Prajal Gurung** and **Krishna Gurung** being students of the sixth semester at **LA GRANDEE International College**, Faculty of Science and Technology 'kha', Pokhara University, do hereby declare that the project proposal submitted to the aforementioned institution is an original work completed by us in partial fulfilment of the requirements for the Bachelor of Computer Application (BCA) program, under the supervision of Sir **Mr. Sunil Sapkota**. We further state that no resources other than those specifically listed have been utilized in the completion of this project.

Name: Prajal Gurung
Class Roll No.: 15
PU-Registration No.: 2021-1-53-0359
Semester: 6th Semester

Date: 22/02/2025
Signature: .....

Name: Krishna Gurung
Class Roll No.: 9
PU-Registration No.: 2021-1-53-0353
Semester: 6th Semester

Date: 22/02/2025
Signature: .....

Supervisor's Declaration

I hereby recommend that this project entitled "PopCornBox" is done under my supervision by **Prajal Gurung & Krishna Bahadur Gurung** during their sixth Semester in partial fulfilment of the requirements for the degree of BCA under Pokhara University is completed to my satisfaction and be processed for final evaluation.

_____

Mr. Sunil Sapkota

Date:22/02/2025

Letter of Approval

We certify that we have examined this report entitled "**PopCornBox**" and are satisfied with the project defense. It is satisfactory in the scope and quality as project in partial fulfillment of the requirements for the degree of **Bachelor in Computer Application (BCA)** under **Pokhara University**.

_____	_____	_____
Supervisor	External Examiner	Program Coordinator
Mr. Sunil Sapkota	Er. Asmit Nepali	Mr. Kundan Chaudary
Ast. Professor		Ast. Professor
LA Grandee International College		LA Grandee International College

Date: 22/02/2025

Abstract

The “PopCornBox” project is focused on developing a web-based application that aims to simplify movie discovery, profile management, and payment handling, creating a seamless and user-friendly environment for movie lovers. By addressing usual challenges faced by current movie streaming platforms, such as complex user interfaces, intrusive ads, and inefficient watch-list or playlist management, “PopCornBox” enhances the overall movie-watching experience.

TABLE OF CONTENTS

List of Figures

List of Tables

Abbreviations

DFD	Data Flow Diagram
PCB	Pop Corn Box
DB	Database
ERD	Entity Relational Diagram
UI	User Interface
CSS	Cascading Style Sheet
JS	Java Script
OTP	One Time Password
TMDB	The Movie Database
API	Application Programming Interface

1. Introduction

A Movie Streaming App is a specialized software application designed to allow users to watch movies over the internet. This platform aims to provide an on-demand entertainment service, where users can access a vast library of movies without needing to download its files via internet. As viewers increasingly seek personalized and user-friendly experiences, the demand for innovating and streaming application has grown.

The primary goal of this project is to efficiently manage and deliver a vast array of video content to users while ensuring a high-quality, uninterrupted streaming experience. Additionally, it simplifies administrative tasks like user subscription management and payment handling, enabling smooth business operations for the platform provider.

The motivation behind creating a "PopCornBox" is to address the evolving needs of movie enthusiast by providing a seamless and enriching platform for discovering and enjoying required movie contents. This proposal outlines the vision of objectives and development plan for "PopCornBox", highlighting streaming experience for normal movie watcher.

Throughout the project, various software design, including ER diagrams, DFD, project Gantt chart will be employed to support its development. The coding will do in the web technologies like for frontend: HTML, CSS, and JS and for backend: Nodejs, Express.js, MongoDB. The project tasks will be evenly and equally distributed among team members according to their skills and knowledge.

2. Problem Statement

- Complicated and Unmanaged User Interfaces (UI)
- Commercial Interruptions
- Inefficient Watch-list Management

3. Objectives

- Billing and Payment Management
- Membership Management
- Simplify User Interface (UI)

4. Background Study

We initiated our investigation by recognizing the necessity for a Movie Streaming Web Application. Initially, our research was focused on identifying the underlying reasons that necessitate the implementation of “PopCornBox”. We gathered various project requirements through visiting website like Netflix, Amazon Prime Video, Hulu, Disney+, etc.

During our analysis, we investigate the common problems faced by existing movie application. The primary goal of the project was to create a system that could be easily managed and provide security while covering all the key aspects of “PopCornBox”, such as membership management, subscription, and simple user interface.

However, it became apparent that the project had limitations, primarily due to complexity of user interface, commercial interruption and inefficient watchlist management. While visiting “Amazon Prime” (https://www.primevideo.com), one of the popular movie streaming apps, we noticed/detected that its UI has been widely criticized for its cluttered and confusing layout. We also noticed numerous “Hulu” (https://www.hulu.com) users frequently about the high frequency and repetitive ads which ultimately disrupts the flow of movie-watching experience, making it less enjoyable. Similarly, Efficient watch list management is crucial for users who want to keep track of movies which enables users to organize and prioritizes watch list which we found lacking various trending movie streaming sites like Amazon Prime, Netflix (https://www.netflix.com), Hulu, etc.

After having gone through the analysis, we have tried to cover-up these problems or drawbacks of existing movie streaming sites. From this analysis, we will be developing the “PopCornBox”, movie app designed to tackle these issues and provide more streamlined and enjoyable movie-watching experience.

5. Requirement Documents

5.1 Functional Requirements

5.1.1 User Features

1. **User Registration and Login**
a. Users must be able to create an account, log in, and log out securely.
b. Support for password recovery via email.
2. **Membership and Subscription Management**
a. Users can choose between subscription plans (monthly, yearly, etc.).
b. Option for upgrading or downgrading membership plans.
3. **Movie Browsing and Search**
a. A search bar to allow users to find movies
4. **Watch-list Management**
a. Allow users to create, view and delete their personalized watch-lists.
5. **Video Streaming**
a. Stream movies directly in-app with adaptive streaming based on internet speed.
6. **Billing and Payment Management**
a. Secure payment options

5.1.2 Admin Features

1. **User Management**
a. View, edit user accounts.
2. **Admin Management**
a. Edit admin account.

5.2 Non-Functional Requirements

- Usability: The application must have a user-friendly interface with easy navigation.
- Performance**:** The application must meet certain performance requirements such as fast loading time and low memory usage.

- Security and privacy: The application must have security and privacy features which are authentication and authorization.

6. System Design

Dataflow and Er-Diagram are used for understanding the system's design and its functionalities, and both are important for creating proper documentation.

6.1 Er-Diagram

An Entity-Relationship (ER) diagram is a visual representation of a database's structure. It uses entities (objects or concepts) and their relationships to illustrate how data is organized and connected within a database system. ER diagrams are widely used in database design and modelling to help understand and plan data relationships.

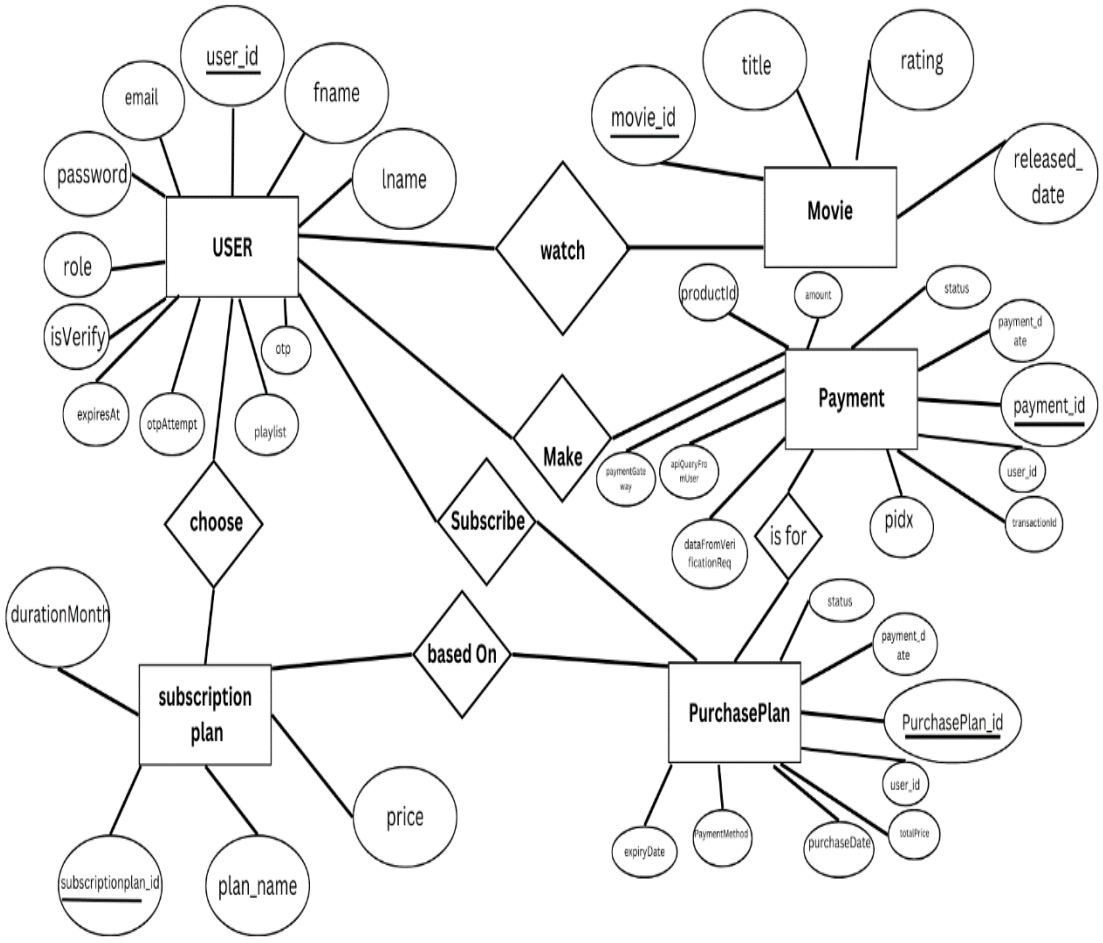

Figure 6.1.1: ER- Diagram

6.2 Dataflow Diagram

It is a diagrammatic representation that portrays the flow of data in a system or a process. Helps communicate the general data flow structure of a proposed system to the system designer, programmer, and end-users.

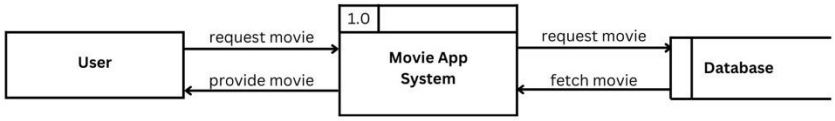

Figure 6.1.1: Level 0 DFD

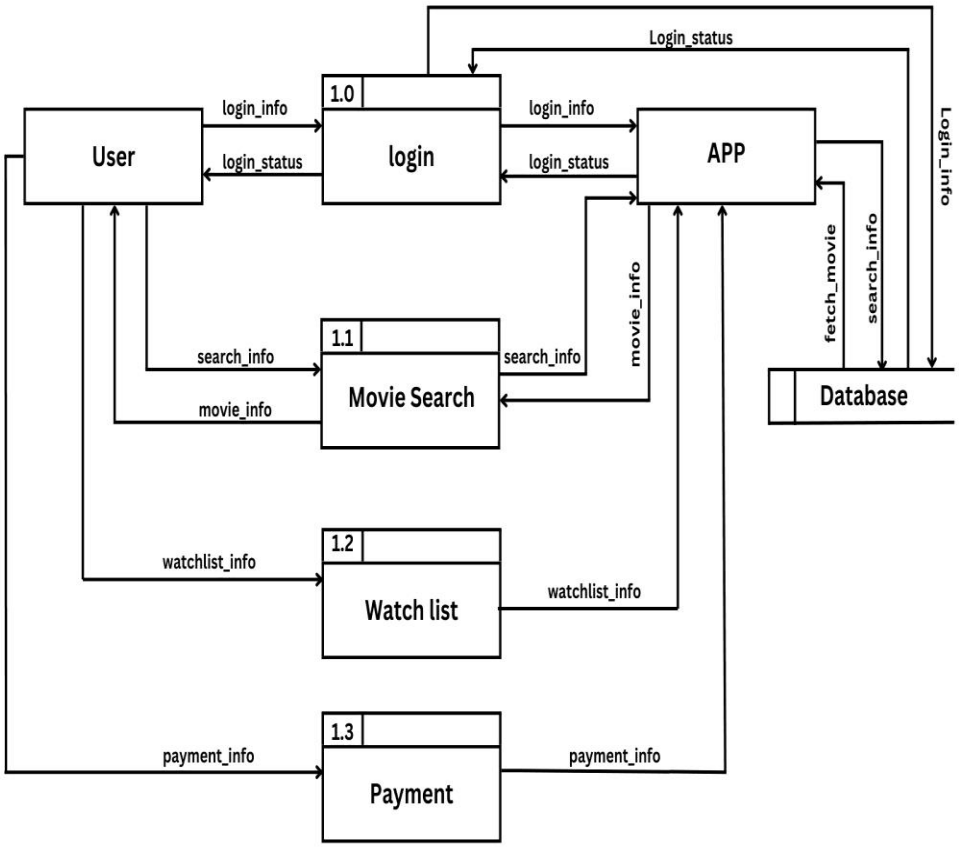

Figure 6.2.2: Level 1 DFD

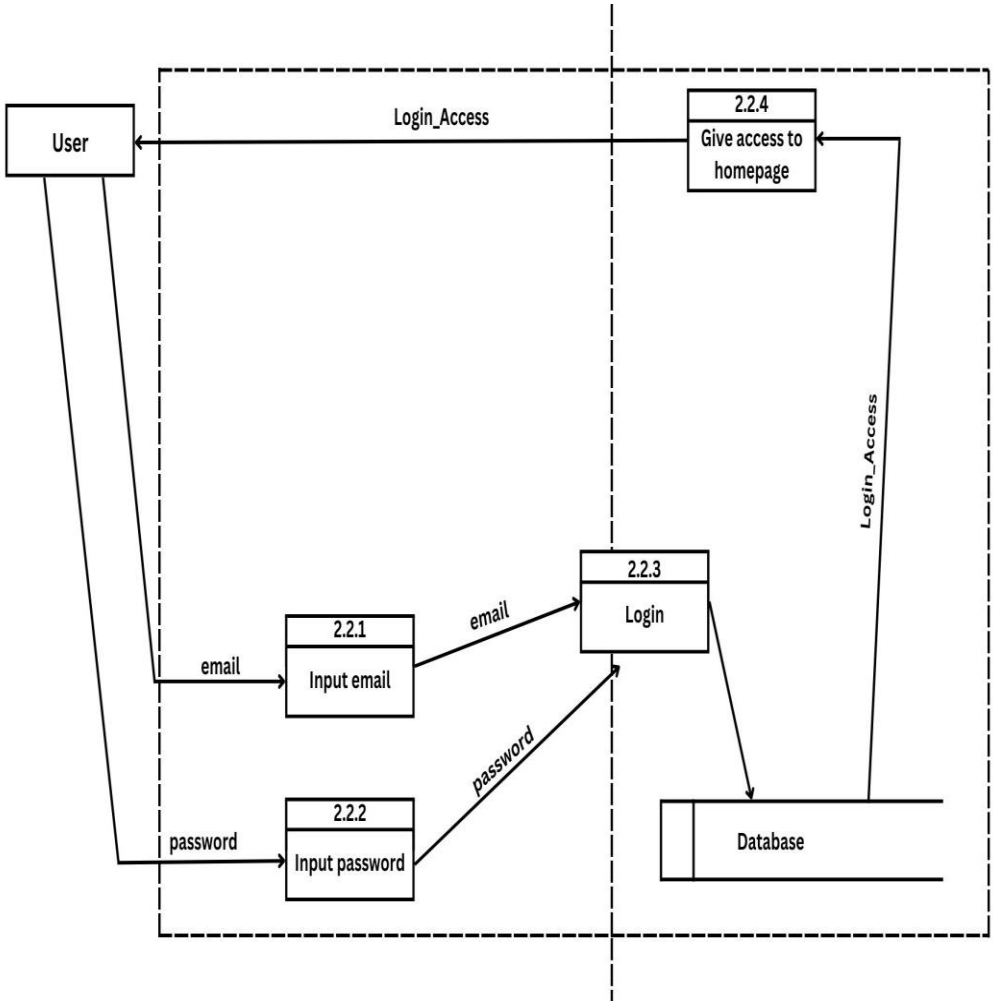

Figure 6.2.3: Level 2 DFD

7. Methodology

Figure 7.1: Agile Model

1. Sprint 0: Project Initiation and Planning

This is the foundational phase where the groundwork for the project is laid. Although not always considered a "sprint" in traditional Agile, it is essential for setting up the project.

Activities:

- Project Name: Define the name of the project.
- Objectives: Clearly outline the goals and objectives of the project.
- Problem Identification: Identify the problem the project aims to solve.
- Background Study: Conduct research to understand the context and existing solutions.
- Project Gantt chart: Create a high-level timeline (Gantt chart) to visualize the project phases and milestones.
- Requirement Gathering: Start gathering initial requirements from stakeholders.

Deliverables:

- Project chart or initiation document.
- Initial list of requirements.

2. Sprint 1: Frontend Development - Login and Registration

In this sprint, the focus is on building the frontend components for user authentication.

Activities:

- Coding: Develop the frontend for login and registration pages.
- UI/UX Design: Ensure the design aligns with user experience best practices.
- Review Requirements: Revisit the requirement documents to ensure alignment with the project goals.
- System Design: Begin drafting the system architecture, focusing on the frontend components.

Deliverables:

- Functional login and registration pages.
- Updated system design documentation.
- Revised project Gantt chart reflecting progress

3. Sprint 2: Frontend Development - Dashboard

The second sprint focuses on developing the dashboard, which serves as the central hub for users.

Activities:

- Coding: Build the frontend for the dashboard
- Requirement Documents: Study the updated requirement documents to ensure all features are covered.
- System Design: Refine the system design to include the dashboard's integration with backend services.
- Project Gantt chart: Update the timeline to reflect progress and adjust for any changes in scope.

Deliverables:

- Functional dashboard interface.
- Updated system design and requirement documents.
- Revised Gantt chart.

4. Sprint 3: Backend Development - Initial Setup

This sprint marks the transition to backend development, laying the foundation for server-side logic and database interactions.

Activities:

- Coding: Begin coding the backend, focusing on core functionalities like user authentication, API endpoints, and database schema design.
- Methodology: Document the development methodology being used (e.g., Agile, Scrum).
- References: Compile references for technologies, frameworks, and libraries used in the project.
- Project Gantt chart: Update the timeline to reflect backend development progress.

Deliverables:

- Initial backend setup with basic functionalities.
- Documentation of methodology and references.
- Updated Gantt chart.

5. Sprint 4: Backend Development - Full Implementation

This sprint focuses on completing the backend development and ensuring seamless integration with the frontend.

Activities:

- Coding: Finalize the backend implementation.
- Testing: Conduct testing to identify and fix bugs.
- Project Chart: Update the project chart to reflect the status of backend development.
- References: Add any new references or resources used during this phase.

Deliverables:

- Fully functional backend.
- Test cases and results.
- Updated project chart and references.

6. Sprint 5: Testing, Deployment, and Review

The final sprint involves comprehensive testing, deployment, and a review of the entire project.

Activities:

- Testing: Perform testing to ensure the application meets all requirements.
- Deployment: Deploy the application to the “Render” and “Railway”.
- Documentation: Finalize all documentation.

Deliverables:

- Fully tested and deployed application.
- Comprehensive documentation.

8. Project Gantt chart

A Gantt chart is a popular project management tool used to visualize the schedule of a project. It displays tasks or activities against time, allowing project managers to track progress, manage dependencies, and allocate resources efficiently.

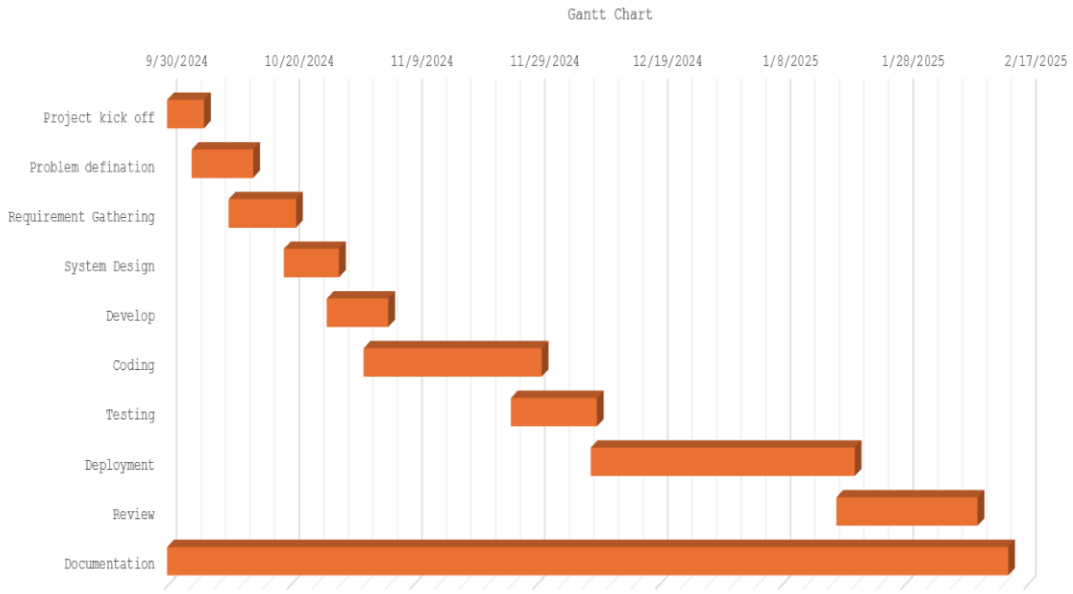

Figure 8.1: Project Gantt chart

9. Work Assigned

The different task identified for the compilation of the project were divided among the team members, with accordance to their talent and capabilities, and performed accordingly. Later they were integrated together to form a single unit. The division of task between four of us is tabulated below.

S.N.	Name of the member	Work assigned	Remarks
1.	Krishna Gurung	Coding Documentation Problem Identification Coding of Major Module	
2.	Prajal Gurung	Documentation UI / UX Design Support in Coding System Design Gantt Chart	

Table 9.1: Work Assigned

10. Testing

Software testing is the process of evaluating a software product to ensure it function as intended. It involves running test cases to identify bugs and verify that the software meets expectations.

We had performed following test cases:

Test Case 1

Title: Login Testing

Test Case Id	**Test Case**	**Test Steps**	**Test Data**	**Expected Results**
1	Invalid Login	1. Navigate to the login page. 2. Enter valid credentials. 3. Click "login".	Email = abc Password = v78	Incorrect email and password
2	Valid login	1. Navigate to the login page. 2. Enter valid credentials. 3. Click "login".	Email = gprajal88@gmail.com Password = 12345	User is redirected to the dashboard or home page.
3	Empty Field	1. Navigate to the login page. 2. Enter valid credentials. 3. Click "login".		Please fill out this field

Table 10.1: Login Testing

Test Case 2:

Title: Registration Testing

Test Case Id	**Test Case**	**Test Steps**	**Test Data**	**Expected Results**
1	Duplicate Registration	1. Navigate to the register page. 2. Enter valid credentials. 3. Click "register".	First name=Prajal Last name=Gurung Email=gprajal88@gmail.com Password = 12345	Email already exist
2	Valid register	1. Navigate to the register page. 2. Enter valid credentials. 3. Click "register".	First name = Prajal Last name = Gurung Email = gprajal88@gmail.com Password = 12345	User is redirected to the login page
3	Empty Field	1. Navigate to the register page. 2. Enter valid credentials. 3. Click "register".		Please fill out this field

Table 10.2: Registration Testing

Test Case 3:

Title: Home Page Testing

Test Case Id	Test Case	Test Steps	Test Data	Expected Results
1	Homepage load	1. Login with valid credentials. 2. redirect to the homepage		Payee user can only load to homepage
2	Logout functionality	1. Login with valid credentials. 2. Click “logout”.		User is redirected to the login page and session is deleted

Table 10.3: Home Page Testing

Test Case 4:

Title: Subscription Testing

Test Case Id	Test Case	Test Steps	Test Data	Expected Results
1	Verify subscription plans are displayed dynamically	1. Navigate to the payment page. 2. Check the order of plans.		Plans are displayed
2	Verify user can select a subscription tier	1. Click on a subscription tier (eg : “yearly”). 2.Proceed to payment	Selected:900	User is redirected to payment gateway

Table 10.4: Subscription Testing

Test Case 5:

Title: Forget Password Testing

Test Case Id	**Test Case**	**Test Steps**	**Test Data**	**Expected Results**
1	Forget Password	1.Navigate to the forget page. 2.Enter valid credentials. 3.Click “Send reset link”.	Email = gprajal88@gmail.com	Redirect to verify otp and reset password
2	Verify Forget Password	1.Enter valid credentials 2.Click “set new password.”	Otp = 193402 Password = 54321	Password reset successfully and redirect to login page
3	Verify forget password with invalid detail	1.Enter invalid credentials. 2.Click “Set new password”	Otp = 123456 Password = 54321	Invalid otp.

Table 10.5: Forget Password Testing

11.Project Results

The “PopCornBox”, built with modern technologies like Node.js, and MongoDB. Popcorn Box ensured a smooth and responsive experience for all users. Future enhancements include adding personalized movie recommendations, expanding payment options, and developing a mobile application to enhance accessibility further.

- Functional Movie Streaming App: Simple design that enhances user experience where movies are fetched from TMDB API.
- Security: It secures the user profiles by using authentication and authorization like username and password.
- Payment Gateway: Integration with Khalti for payment processing.
- Deployment: It is deployed on “Render” and “Railway”, ensuring accessible to users.

12. Future Enhancements

Below, we have listed enhancements which can range from improving user experience to adding new features that enhance functionality of our PopCornBox:

- Download Movie to watch offline.
- Review or rating system.
- Allow users to purchase premium content or rent movies not included in their subscription.
- Notify users about subscription expiry, new movies, or updates to their watch-lists.
- Filters for categorization (e.g., trending, genre, language, ratings).

13. Conclusion

In conclusion, “PopCornBox” is designed to provide a smooth and enjoyable movie streaming experience for users, while making it easier for administrators to manage their operations. By addressing common issues faced by other streaming platforms such as complicated interfaces, intrusive ads, and poor watch-list management “PopCornBox” offers a cleaner, more user-friendly environment.

We have developed the system using a combination of modern web technologies and tools, ensuring that it is secure, reliable, and easy to navigate.

Ultimately, PopCornBox will allow movie enthusiasts to enjoy uninterrupted viewing. With its simple interface and thoughtful features, PopCornBox is set to become an efficient and trusted solution for movie streaming needs.

14. Annexures

Code Snippet for User Authentication:

1. Register

```
const register = async (req, res) => {
  try {
    const { fname, lname, email, password } = req.body;

    if (!fname || !lname || !email || !password) {
      return res.status(400).send("All fields are required");
    }

    const existingUser=await UserModel.findOne({email:email});
    if(existingUser){
      return res.status(400).render("register", { message: "Email already exist" });
    }
    // const hashedPassword = await bcrypt.hash(password, saltRounds);
    const user = await UserModel.create({
      fname: fname,
      lname: lname,
      email: email,
      password: password
    });

    console.log(user);
    res.render("login", { message: "Succesfully Signup" });
  } catch (error) {
    console.error("Error finding user:", error);
    return res.status(500).send({
      error: error.message()
    });
  }
};
```

2. Login

```
const login = async (req, res) => {
  const { email, password } = req.body;

  if (!email || !password) {
    return res.status(400).send("All fields are required");
  }

  try {
    const user = await UserModel.findOne({ email, password });
    if (!user) {
      return res.render("login", { message: "Invalid email or password" });
    }

    const token = setUser(user);
    res.cookie("uid", token, { httpOnly: true, secure: true });
    res.redirect("/home"); // After successful login, redirect to homepage
  } catch (error) {
    console.error("Error finding user:", error);
    return res.status(500).render("login", { message: error.message });
  }
};
```

3. Forget-Password

```
const forgetPassword = async (req, res) => {
    try {
      const { email } = req.body;

      const existingUser = await UserModel.findOne({ email: email });

      if (!existingUser) {
        return res.status(404).render("forget-password", { message: "Email not found" });
      }

      const result = await sendmail(existingUser.email);
      if (!result) {
        console.error("Failed to generate OTP or send email");
        return res.status(500).render("forget-password", { message: "OTP could not be generated or email sending failed" });
      }

      existingUser.otp = result.otp;
      existingUser.expiresAt = new Date(Date.now() + 15 * 60 * 1000); // OTP expiration time (15 minutes)
      await existingUser.save();

      console.log(existingUser.email)
      // Pass 'email' to the view so it's available in the form
      return res.render("verify-otp", { email: existingUser.email, message: "OTP sent successfully. Please verify it." });

    } catch (e) {
      console.error('Error in forgetPassword:', e);
      return res.status(500).render("forget-password", { message: "An unexpected error occurred. Please try again later." });
    }
  };
```

4. Verify-otp

```
const verifyOtp = async (req, res) => {
  try {
    const { otp, newPassword, email } = req.body;  // Extract email, otp, and newPassword from req.body

    console.log("Email received in verifyOtp:", email); // Log email to check its value

    const existingUser = await UserModel.findOne({ email: email });
      console.log("ex"+existingUser.otp)
      console.log("otp"+otp)
    if (!existingUser) {
      return res.status(404).render("verify-otp", { message: "User not found" });
    }

    // Check if OTP is correct and has not expired
    if (String(otp) !== String(existingUser.otp)) {
      return res.status(400).render("verify-otp", { message: "Invalid OTP", email: email });
    }

    if (new Date() > existingUser.expiresAt) {
      return res.status(400).render("verify-otp", { message: "OTP has expired", email: email });
    }

    // Set new password (make sure to hash it)
    existingUser.password = newPassword;
    existingUser.otp = undefined;  // Clear OTP after use
    existingUser.expiresAt = undefined;  // Clear expiration date
    await existingUser.save();

    // After successful reset, redirect to login
    return res.redirect("/login");

  } catch (e) {
    console.error('Error in verifyOtp:', e);
    return res.status(500).render("verify-otp", { message: "An unexpected error occurred. Please try again later." });
  }
};
```

15. Reference:

Netflix. (1997, August 29). Retrieved from Netlfix: https://www.netflix.com

Hulu. (2007). Retrieved from Hulu: https://www.hulu.com

Asana. (2008). Retrieved from Asana: https://asana.com/resources/agile-methodology

Prime, Amazon. (2005, February 2). Retrieved from Prime Video: https://www.primevideo.com