University of Dhaka

Department of Computer Science and Engineering

CSE-3112 : Software Engineering Lab

Project Title: **TestCraft**

Software Requirement Analysis Document

Developers:

Name: Md Emon Khan

Roll: 30

Name: Mahmudul Hasan

Roll: 60

Contents

1 Introduction

TestCraft is an online test designing and test-taking platform which offers a solution for tutoring individuals and institutions. With this web application, users can easily create mock exams, practice tests, or online assessments designed for their specific audience.

The web application offers some advanced features, including automated assessment capabilities. This ensures a seamless and efficient evaluation process, saving time for both test creators and participants. Additionally, **TestCraft** provides automated notification services, keeping users informed about important timelines, results, and any pertinent updates from their associated institutions. This feature enhances communication and ensures that all stakeholders stay well-informed throughout the testing process.

1.1 Purpose of the System

The primary goal of **TestCraft** is to deliver a user-friendly test creation tool designed for individuals or institutions involved in tutoring. This enables tutoring professionals to unwind and focus on personal aspects while efficiently managing their tutorship responsibilities.

A crucial goal of our project is to afford students an easily affordable practice tool, enhancing their abilities. Students, hopefully, will be able to reflect on past assessments, get insights into their strengths and weaknesses for further development. Additionally, **TestCraft** provides tutors with better vision into the progress of each individual student they tutor.

TestCraft serves as a dynamic medium for interaction between educators and students that promises the best possible outcomes for both.

1.2 Scope of the System

The software requirements specified in this Software Requirements Analysis Document (RAD) is about the **TestCraft** system, version 1.0. **TestCraft** is an online assessment and test creation platform designed for educators and educational institutions. The scope of this RAD encompasses the entire **TestCraft** system, including its various modules and functionalities.

1.3 Objective and Success Criteria of the System

The primary objective of the **TestCraft** system is to provide a dynamic and interactive platform for educators and students which will enrich the educational experience by improving learning outcomes.

Success criteria for the system include:

- Efficient test creation and administration
- Easy assessment and feedback mechanisms
- Better communication between tutors and students
- Improved student engagement and performance
- Enhanced performance analysis and suggestions for improvement for an individual
- Better facility for the teachers to keep track of each student and their condition
- Good performance and satisfactory user experience
- Increasing user involvement

1.4 Definitions and Acronyms and Abbreviations

- AES: Advanced Encryption Standard
- AI: Artificial Intelligence
- GDPR: General Data Protection Regulation
- MCQ: Multiple Choice Question
- RAD: Required Analysis Document
- RBAC: Role Based Access Control
- TBD: To Be Decided

1.5 References

[This section is skipped for now as we have not written any more documents on this project. Hopefully, it'll contain actual references in the future]

1.6 Overview

This RAD (Requirements Analysis Document) outlines the functional requirements and specifications for **TestCraft**, a platform designed for educational purpose, specifically focusing on test administration, assessment, and progress tracking. Let's break down the document:

- **Use Case Model:** The document begins with a Use Case Model, detailing various interactions between different actors (users) and the system. These use cases include actions such as giving assessments, seeing assessments, self-analysis, and advertising channels.
- **Dynamic Model:** This section provides visual representations of system behavior through sequence and activity diagrams. These diagrams illustrate the flow of interactions between users and the system components.
- **User Interface:** The User Interface section describes the screens and components of the **TestCraft** application. It describes the landing page, login, registration, console, and specific tabs within the console like Channels and Progress Analysis.
- **Software Interface:** This section includes details about how the software interacts with external systems or services
- **Hardware Interface:** This section typically describes the hardware requirements or dependencies of the software.
- **Supporting Information:** The Installation and Deployment sections offer guidance on setting up and deploying the **TestCraft** application. Instructions are provided for both frontend and backend components, including the use of Node.js and deployment
- **Overview:** Overall, this RAD document is a guide for understanding the requirements and implementation details of **TestCraft**. It outlines the functionalities available to different types of users and provides instructions for installation and deployment.

2 Overall Description

2.1 Product Perspective

TestCraft is the only project of our team. It is a new, self contained product with unique features and functionalities that only belong to this project.

2.2 Product Function

Users will have different roles in different channels they're involved in.

- Users have to authorize to get started
- Users can open their own channel in which they'll automatically be assigned a tutor. They can invite teachers or students to join that channel
- Users can join a channel as students by requesting enrollment or through an invitation
- Each channel will feature the teachers with group based online test scheduling and taking abilities
- Teachers will be able to create tests on their own or using their own or shared problems database
- New problems can be generated using Artificial Intelligence
- Students can take tests, see their results, compare with past assessments

2.3 User Profiles

- **Primary User**: The primary users interact with the software core functionalities. They are the main users of the application
    - **Frequency of Use**: Daily
    - **Technical Expertise**: Not applicable
    - **Security or Privilege**: Experience level
    - **Functionalities Used**: Core user level functionalities
- **Administrators**: Administrators manage the application, configures the settings, and oversees user access
    - **Frequency of Use**: Regular
    - **Technical Expertise**: Advanced knowledge on application development and cyber security
    - **Security or Privilege**: Elevated privilege, oversees the project
    - **Functionalities Used**: Development environment functionalities

- **Stakeholders**: The stakeholders are the investors, sponsors and other interested bodies
    - **Frequency of Use**: Occasional
    - **Technical Expertise**: Varied
    - **Security or Privilege**: User level security. Enhanced experience

2.4 Constraints

- **Corporate Policies:** Ensuring data security is pretty challenging with the use of raw MySQL. The developer team is required to do their best to provide the best security experience following the best principles of Cyber Security
- **Hardware Limitations:** Limited hardware may impose limited user experience enhancement. Focusing on the main features get priority over everything else
- **Specific Tools and Technologies:** The application is developed using a full JavaScript stack consisting of ReactJS and ExpressJS. Uses MySQL for authentication and database management
- **Language Requirements:** The application is mainly featured in International Language. However, Bangla is also made available

2.5 Assumptions and Dependecies

Assumptions:

- The project assumes the availability of various thrid-party APIs:
    - Google Generative AI
    - SSLCOMMERZ payment gateway
- Stable and reliable internet connectivity from users are assumed as well as latest versions of browser and web framework supports from the user devices
- Adequate resources and support from the hosting body for better performance and to maintain scalability

Dependencies:

- The project development environment is reliant on NodeJS v21, and MySQL v8.2
- The project uses MySQL v8.2 for user authentication, storing and managing data
- The project is built on Windows 11 and MacOS systems

3 Proposed System

3.1 Overview

TestCraft is an educational platform designed to improve the way educational assessments are conducted, managed, and analyzed. It has a user-friendly interface, extensive feature set, and thoughtful implementation. **TestCraft** tries to provide a very good experience for both students and teachers.

Scenarios:

Scenarios describe the various scenarios that may occur in the user's end. It's a narrative description of the sequence of events that occur during a use interaction.

Use Cases:

Use cases describe the various user tasks that the user might carry out. Here are some of the major use cases:

1. **Advertisement Interaction:** Users can explore and join channels through advertisements
2. **Announcement Management:** Within channels or group pages, users can efficiently create and publish announcements
3. **Test Creation and Participation:** Teachers can create tests with ease, set problems, and manage submissions, while students can participate in tests
4. **Progress Analysis:** Users have access to self-analysis tools, enabling them to track their academic progress, identify areas for improvement, and receive personalized suggestions.

User Interface:

The **TestCraft** user interface is designed following the conventional design standards. Each of the components, the console, channels, and progress analysis, is designed to optimize user experience and accessibility.

Software and Hardware Interface:

TestCraft is compatible with all modern web browsers and operating systems. It requires minimal hardware resources. So, it's accessible across a wide range of devices and configurations. A moderate to high-bandwidth internet connection is recommended for optimal performance.

Overall

TestCraft is a tool for educational institutions, instructors, and students, enabling them to do test administration, assessment, and progress tracking processes. The functional and non functional requirements, use cases, scenarios are well defined in this section.

3.2 Functional Requirements

3.2.1 Requirement 1

- **id**: REQ-01
- **name**: Authentication
- **description**: The system allows users to register using for an account and log in using username and password. Login credentials must be stored securely. The system should feature the following:
  - Validate user credentials
  - Provide options for password recovery
  - Delete account using valid credentials
- **priority**: top
- **reference:** TBD

3.2.2 Requirement 2

- **id**: REQ-02
- **name**: User profile
- **description**: The system should allow the users to edit and update personal details. Personal details may include:
  - Profile photo
  - Birth date
  - Work/Job details
  - Education details
  - Location/Birth place details
- **priority**: high
- **reference**: TBD

3.2.3 Requirement 3

- **id**: REQ-03
- **name**: User's problems database
- **description**: Users should be allowed to create their own well organized database of problems and questions along with solutions. Users should be able to share their database with other users.
- **priority**: high
- **reference**:

3.2.4 Requirement 4

- **id**: REQ-04
- **name**: Channel
- **description**: Users can create their own channel giving it a proper name. They should be able to:
  - Invite others into the channel

    - Advertise the channels publicly
    - Enroll interested bodies into the channel
    - Assign roles (Teacher/Student) to members of the channel
    - Create announcements into the channel
- **priority**: top
- **reference**:

3.2.5 Requirement 5

- **id**: REQ-05
- **name**: Group
- **description**: Teachers in a channel should be able to create groups within the channel with specific students enrolled into the channel. They should also be able to create announcement in each group separately
- **priority**: high
- **reference**:

3.2.6 Requirement 6

- **id**: REQ-06
- **name**: Test crafting
- **description**: A test crafting tool to build and schedule a test in a specific channel or a specific group. Users should be able to:
    - Create test using their own or shared problems database
    - Generate problems using AI
    - Generate problems using AI based on their own database or a specific document or a specific specification
- **priority**: top
- **reference**

3.2.7 Requirement 7

- **id**: REQ-07
- **name**: Test giving
- **description**: The teachers should be able to create a group specific or a channel specific test. They should be able to
    - Create test
    - Schedule and give the test to students creating an announcement (must) for the test
    - Assess the students' answers
- **priority**: high
- **reference**:

3.2.8 Requirement 8

- **id**: REQ-08
- **name**: Test taking
- **description**: The students should be able to take a test by
    - Entering to the test arena through the announcement
    - Giving answers for each problem by:
        - Typing in the respective answer of each problem in the field below the problem
        - Uploading an image containing the handwritten answer
        - If the problem is an MCQ, then selecting one of the options
- **priority**: high
- **reference**:

3.2.9 Requirement 9

- **id**: REQ-09
- **name**: Assessment

- **description**: The teachers should be able to provide each student a detailed assessment of their answer of a particular question. They could:
  - Type in the assessments themselves
  - Generate the assessments using AI

  Each student should be able to
  - View their respective test assessments and marks
  - Complaint on any issue in the assessments
- **priority**: high
- **reference**:

3.2.10 Requirement 10

- **id**: REQ-10
- **name**: Self Analysis
- **description**: The students should be able to view an analysis of his performance based on previous performances and the topic the test is related to. They should be able to view a comparative analysis with the collective results of the students in the channel.
- **priority**: high
- **reference**:

3.3 Non Functional Requirements

3.3.1 Usability

- Users should immediately be able to understand the use of the application
- Users with no prior experience should also be able to become friendly with the UI within minutes
- System should provide proper tool tips and help messages while hovering over interface elements

- Development should be done following application development best practices with optimum project structure, proper use of naming conventions, security best practices, modular and object oriented approach to development.

3.3.2 Reliability

- Proper authentication methods should be implemented, and data security should be ensured
- Personal details and information should be handled securely
- Environment variables should be kept hidden
- A separate database server is expected to be created for data security
- Ensuring scalability and performance is top priority
- Proper error handling mechanisms should be implemented

3.3.3 Performance

The overall performance of the application depends mostly on requirements REQ-06, REQ-09, and REQ-10. Assuming a minimum of robust 10Mbps connection in the user's end, the following should be the performance constraints:

- **Database utilization**: Basic database queries to save or retrieve user information, channel information, other test taking data, channels advertisement based on user preference etc should not take more than 2-3 seconds.
- **REQ-06**: Test crafting involves creating test using user's premade database or an AI problem generation tool. User should be able to retrieve data from the database by querying or filtering the database using various constraints. This should not take more than 5-10s. The AI problem generation tool must not take more than a minute to generate a 100 problem problemset and send to the user's end.
- **REQ-09**: Assessment involves typing in the assessments by the user manually or AI assessment. The assessment process by AI should not take more than 10 seconds for a 100 questions

- **REQ-10** Self analysis may involve retrieving small amount of data from a huge database and then processing it using various statistical functions. It also involves plotting the relevant graphs of an analysis. This process should not take more than 10 seconds for one student.

3.3.4 Supportability

- Develop and maintain comprehensive user documentation that includes installation guides, user manuals, and troubleshooting steps
- Establish a knowledge base or online resource center with frequently asked questions (FAQs) and tutorials
- Define the process for deploying software updates and patches, including rollback mechanisms if necessary
- Specify requirements for monitoring system health and performance to identify potential issues proactively

3.3.5 Implementation

- **Programming Language**: The application will be developed using the following programming languages:
  - JavaScript
    - React
    - Express
    - Node.js
  - HTML, CSS
    - TailwindCSS
  - MySQL

  Full stack JavaScript for application development and MySQL for database management is a popular choice among developers. They have a huge community for support
- **Database** The application will use MySQL for database management which is a very powerful database management system
- **Preferred Development System** Windows 11, and MacOS operating systems are chosen as the development system for their popularity, dynamic user experience, availability of powerful development tools.

3.3.6 Scalability

- **User Load**:
  - The system should be able to facilitate hundreds of thousand of concurrent users without significant performance degradation
  - The application should be designed to scale with increasing user engagements and demands
- **Data Volume**:
  - The software should efficiently handle a data volume of up to several 100 gigabytes without compromising performance
  - The database architecture should be scalable to accommodate future growth in data storage needs

3.3.7 Security

- **Data Security**
  - All sensitive user data (e.g., passwords, financial information) must be encrypted at rest and in transit using industry-standard algorithms (e.g., AES-256)
  - Implement access controls to restrict unauthorized access to data based on user roles and permissions
  - Data deletion requests from users should be fulfilled within a specified time frame (e.g., comply with data privacy regulations like GDPR)
- **User Authentication and Authorization**
  - The software shall employ a secure user authentication mechanism (e.g., strong password policies, multi-factor authentication)
  - User roles and permissions should be clearly defined and enforced to control access to functionalities and data (e.g. RBAC)
  - Implement session management best practices to prevent unauthorized access after user inactivity
- **Security Considerations**
  - **Vulnerability Management:** Establish a process for regularly identifying, evaluating, and patching security vulnerabilities in the application.

- **Penetration Testing:** Conduct periodic penetration testing to simulate real-world attacks and identify potential security weaknesses.
- **Secure Coding Practices:** Enforce secure coding practices within the development team to minimize vulnerabilities introduced during development.
- **Logging and Monitoring:** Implement logging and monitoring mechanisms to track user activity and identify suspicious behavior.

3.3.8 Maintainability

- The application should be well-documented using clear comments and industry-standard coding practices
- The code base should be modular and organized, with reusable components to facilitate future modifications
- Automated testing frameworks and unit tests should be implemented to ensure code quality and regression prevention

3.3.9 Testability

- The application should be designed with modular components that can be independently tested in isolation
- Utilize well-defined interfaces between modules to facilitate unit testing and integration testing
- Develop test cases that cover various user scenarios, including positive and negative test cases
- Strive for 100% code coverage through unit testing to ensure all critical functionalities are tested
- Ensure proper error handling and logging mechanisms that are implemented to aid in test analysis

3.4 System Models

3.4.1 Scenarios

- **Scenario 1: Sign In**

- A new user arrives at the platform and wants to sign up for an account. They fill out a registration form and verify their email address.
- An existing user logs in using their username and password.

- **Scenario 2: Create Database** A user creates a new collection for storing problems in the database. They can then add entries (rows) to this collection.
- **Scenario 3: Create Channel** A user creates a new channel on the platform by filling out a form with a channel name and other relevant information.
- **Scenario 4: Create Group** A user creates a new group within a channel or directly in the console by providing necessary details.
- **Scenario 5: Invite People to Channel** A user invites other users (by username) to a specific channel, potentially assigning them different roles within the channel.
- **Scenario 6: Join Channel** A user finds a channel they are interested in and requests to join by clicking a "Join Channel" button.
- **Scenario 7: Make Announcement** A user within a channel or group creates an announcement by typing a message and then publishing it.
- **Scenario 8: Create Test** (Teacher) A teacher creates a new test within a channel or group. They define the test structure, add problems, and submit the test.
- **Scenario 9: Take Test** (Student) A student finds a test announcement within a channel or group and chooses to take the test. They answer the test questions and submit their answers.
- **Scenario 10: Grade Test** (Teacher) A teacher reviews a completed test, sees individual student responses, assigns grades (assessments) for each question, and submits the grades.
- **Scenario 11: View Grades** (Student) A student sees their grades (assessments) for a completed test within a channel or group.

- **Scenario 12: Self-Analyze Progress** A user accesses the progress analysis tool, selects filters for the data (e.g., date range), and views their personal progress details.
- **Scenario 13: Advertise Channel** A user creates an advertisement to promote their channel to a wider audience on the platform. They fill out a form with details about the channel.

3.4.2 Use Cases

- **Use Case 01**
  - **name**: Authenticate
  - **actor**: User
  - **flow of events**:
    - The user finds themselves on the landing page from where they get an idea of the website
    - Then they proceed to get started with the application by opening an ID or by logging in to an existing ID.
    - User chooses either to register or to log in depending on whether they already have an ID
    - For login, the user needs only correctly put their username and password
    - For registration, the user may need to enter their email, choose an available username, create a strong password, fill out a personal information form and then submit
    - Upon submission, an email verification must be performed
    - After verifying the email, user may proceed to use the application to their interest
  - **entry condition**: User clicks on either Login or Register button
  - **exit condition**: User authenticates and is authorized into the application
- **Use Case 02**
  - **name**: Build Database
  - **actor**: User
  - **flow of events**:

        - After getting started, the user finds themselves in the console
        - User chooses Database tab
        - User proceeds to create a problems database by creating a new collection
        - User inserts some rows into the collection
        - User saves the collections
    - **entry condition**: User creates a new data collection
    - **exit condition**: User saves the collection

- **Use Case 03**
    - **name**: Create Channel
    - **actor**: User
    - **flow of events**:
        - User finds themselves in the console
        - User chooses the Channels tab
        - User chooses to create new channel by clicking Create New Channel button
        - User fills out the form using necessary information about the channel to be
        - User submits the form
    - **entry condition**: User clicks Create New Channel button
    - **exit condition**: User submits the form

- **Use Case 04**
    - **name**: Create Group
    - **actor**: User
    - **flow of events**:
        - User finds themselves inside channels tab in the console
        - User clicks on Create New Group
        - Or user finds themselves inside a channel and clicks on Create New Group
        - User fills out the form with necessary information
        - User submits the form
    - **entry condition**: User clicks Create New Group button

- **exit condition**: User submits the form

- **Use Case 05**
  - **name**: Invite People into Channel
  - **actor**: User
  - **flow of events**:
    - User finds themselves inside the channels tab of console or inside the page of a channel
    - User chooses Invite People
    - User searches people using their username
    - User makes a list of people and assign each a role
    - User submits the invitation list
  - **entry condition**: User clicks invite button
  - **exit condition**: User submits the invitation list

- **Use Case 06**
  - **name**: Join Request
  - **actor**: User
  - **flow of events**:
    - User goes to the advertisement page
    - User chooses an advertisement to see the channel information
    - User clicks Join Channel
  - **entry condition**: User clicks join button
  - **exit condition**: User clicks join button

- **Use Case 07**
  - **name**: Announcement
  - **actor**: User
  - **flow of events**:
    - User is inside a channel or a group page
    - User types in an announcement
    - User publishes the announcement by clicking the Post button
  - **entry condition**: User types in an announcement

- **exit condition**: User clicks Post button

- **Use Case 08**
  - **name**: Test Craft
  - **actor**: User
  - **flow of events**:
    - User is inside a channel or a group page where they are a teacher
    - User chooses Create New Test
    - User fills out necessary forms
    - User sets the problems
    - User submits the test
  - **entry condition**: User chooses Create New Test
  - **exit condition**: User submits the test

- **Use Case 09**
  - **name**: Take test
  - **actor**: User
  - **flow of events**:
    - User is inside a channel or a group page where they are a student
    - User chooses a test from a test announcement
    - User finds themselves inside a test arena where they'll wait until the exam starts or see the test problems if it has already started
    - User fills out the solutions
    - User submits the answers by clicking Submit button
  - **entry condition**: User chooses a test from a test announcement
  - **exit condition**: User clicks Submit button

- **Use Case 10**
  - **name**: Give Assessment
  - **actor**: User
  - **flow of events**:

        - User is inside a channel or a group page where they're a teacher
        - User chooses a test already finished from a test announcement
        - User chooses each question and finds different answers by different students
        - User assesses each student's answer and submits each assessment
        - User leaves the page
    - **entry condition**: User chooses a test
    - **exit condition**: User leaves the page

- **Use Case 11**
    - **name**: See Assessment
    - **actor**: User
    - **flow of events**:
        - User is inside a channel or a group page where they're a student
        - User chooses a test already given from a test announcement
        - User sees assessments for each question answered
        - User leaves the page
    - **entry condition**: User chooses a test from a test announcement
    - **exit condition**: User leaves the page

- **Use Case 12**
    - **name**: Self Analysis
    - **actor**: User
    - **flow of events**:
        - User is in the console
        - User chooses Progress Analysis
        - User fills out the necessary filter parameters in the form
        - User submits the filter
        - User sees their progress details
        - User leaves the page

- **entry condition**: User chooses Self Analysis
- **exit condition**: User leaves the page

- **Use Case 13**
  - **name**: Advertisement
  - **actor**: User
  - **flow of events**:
    - User is in the console or in a channel
    - User chooses advertise channel
    - User fills out the necessary information to be shown in the advertisement
    - User submits the form
  - **entry condition**: User chooses advertise channel
  - **exit condition**: User submits the form

3.4.3 Use Case Model

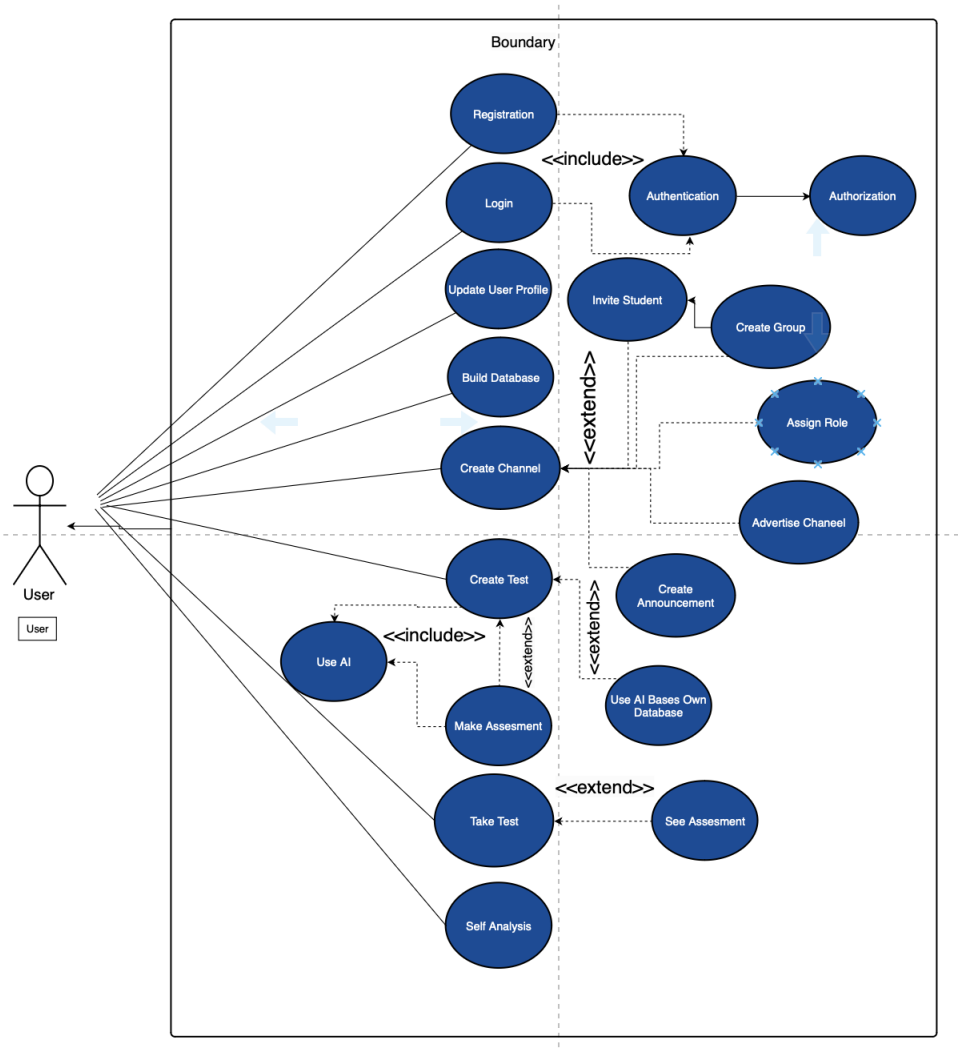

Figure 1: Use case Diagram for TestCraft

3.4.4 Dynamic Model

- **Sequence Diagram**

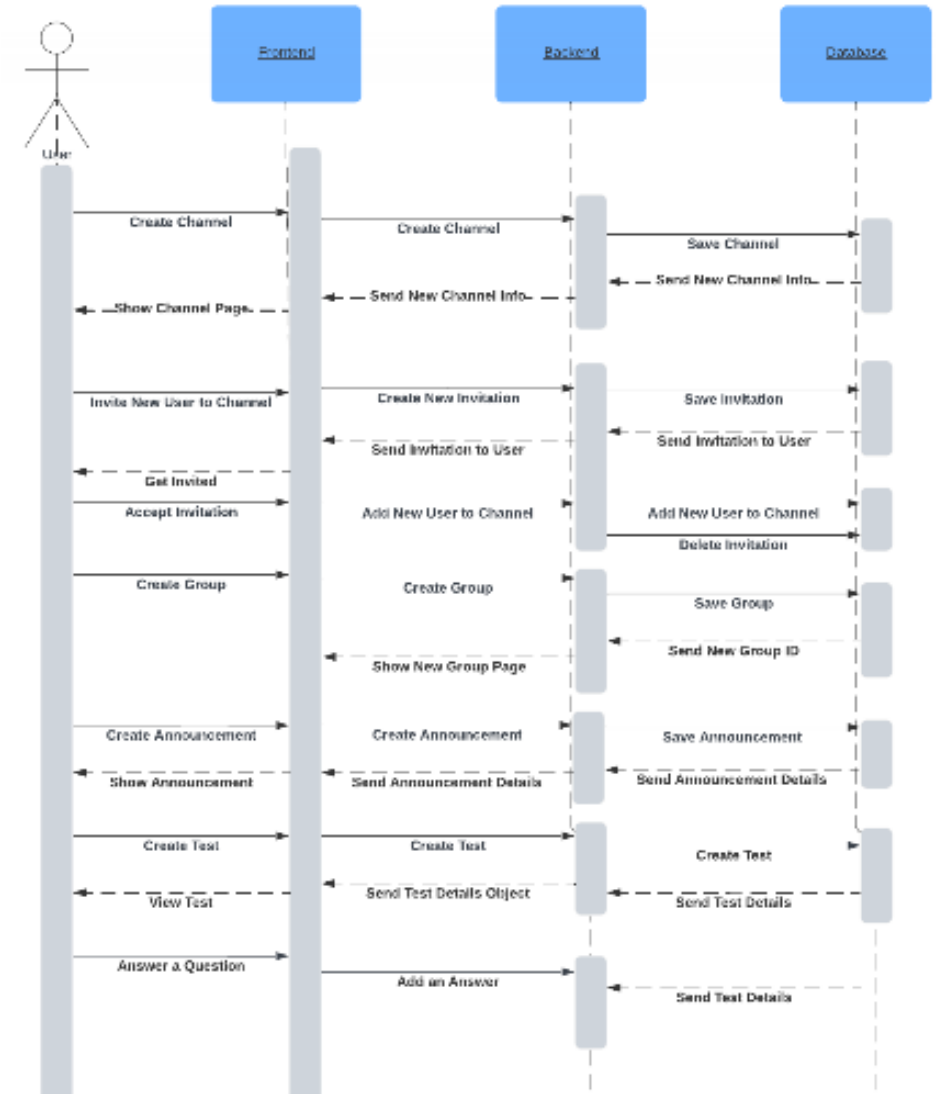

Figure 2: Sequence Diagram

- **Activity Diagram**

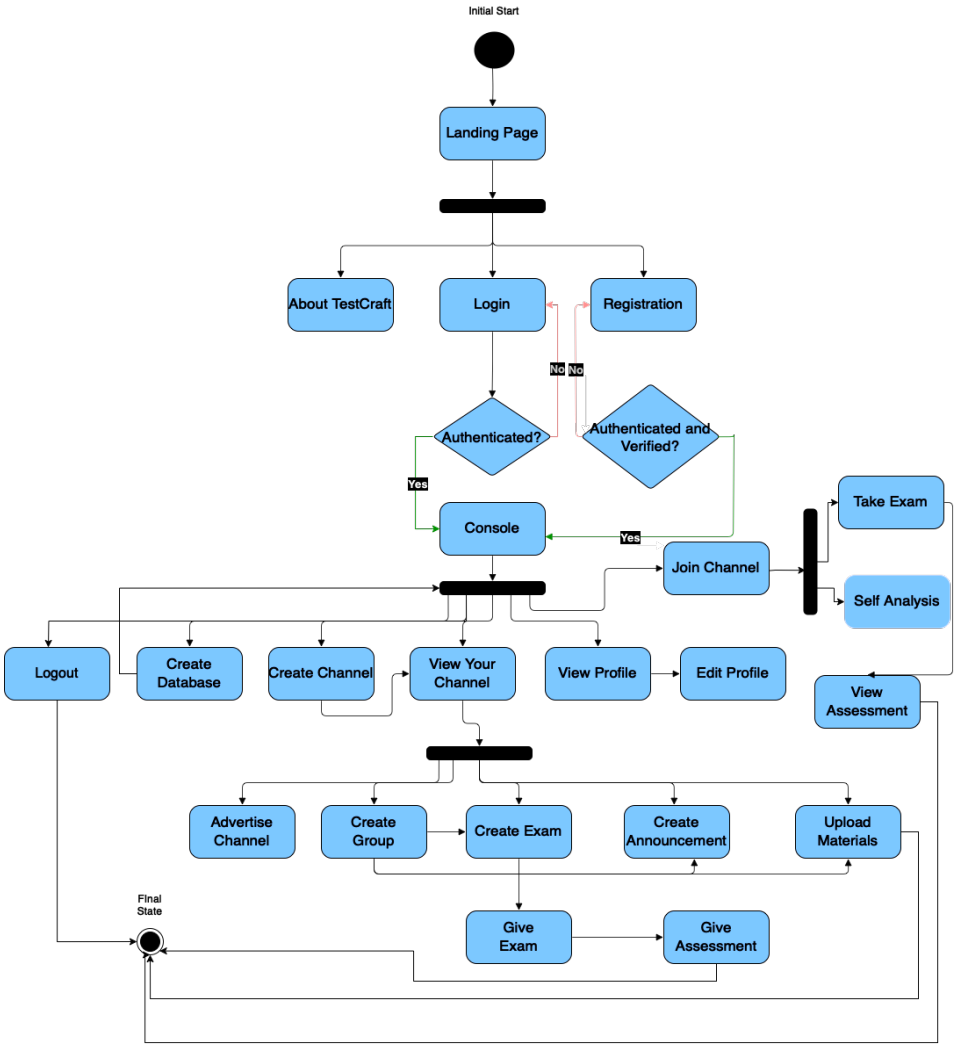

Figure 3: Activity Diagram

- **State Diagrams**

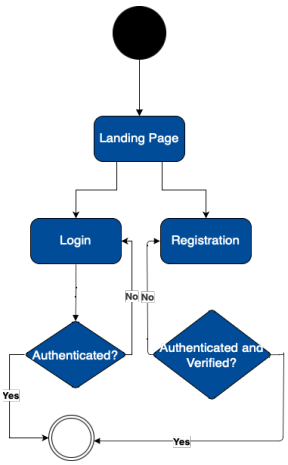

Figure 4: login state diagram

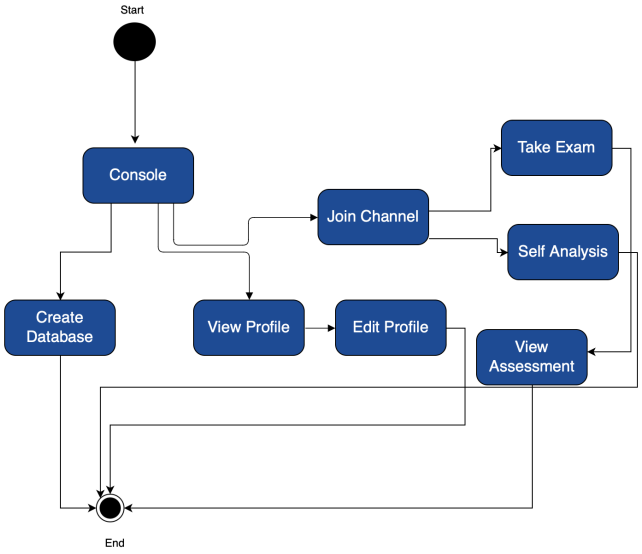

Figure 5: Student activity state diagram

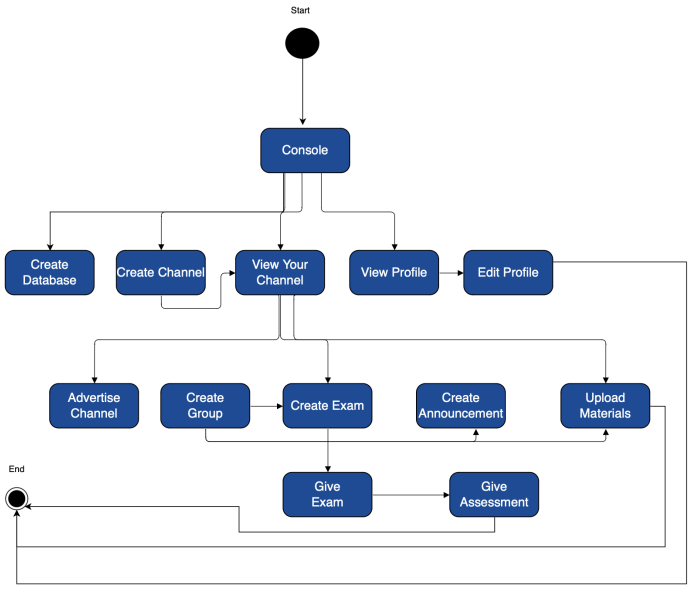

Figure 6: Teacher activity state diagram

3.4.5 User Interface

- User Interface User Interface The User Interface of **TestCraft** is designed according to conventional design standards with thoughtful implementations of different components. The User Interface of **TestCraft** can be estimated to have the following screens to the best of our knowledge, however this is subject to change based on changing requirements -

  1. Landing Page
     (a) about the project
     (b) login and registration buttons
  2. Login Page
     (a) email input
     (b) password input
     (c) login button
     (d) registration page route
  3. Registration Page
     (a) email field

(b) password field
(c) registration button
(d) login page route

4. Nav bar
   (a) Console route
   (b) Channels route
   (c) Explore route
5. Console Page
   (a) Console consists of some tabs
      i. Channels
      ii. Database
      iii. Progress Analysis
   (b) Channels tab contains more tabs
      i. Explore Channels: Contains advertisement of all channels available in a card view
      ii. Create New Channel: To create a new Channel
      iii. My Channels: Contains all of user's own channels and the channels they're involved with
      iv. Advertise Channel button
   (c) Progress Analysis tab contains:
      i. a filter to filter analysis
      ii. submit button
      iii. an analysis of all past tests taken (only when filter is submitted)
      iv. suggestions based on the progress analysis
6. Channels Page
   (a) List of all channels the user is involved in
7. Channel Page
   (a) view of all announcements
   (b) an announcement editor
   (c) a sidebar of
      i. groups: contains routes to all the groups
      ii. members: contains routes to all the members teachers and students separate
      iii. a settings option

8. Group Page
   (a) view of all announcements
   (b) an announcement editor
   (c) a sidebar of
      i. members: contains routes to all the members teachers and students separate
      ii. a settings option
9. Test Crafting tool
   (a) a form asking necessary information of the test to be created
   (b) an editor for the problems to be added
   (c) a database query tool
   (d) an AI generative tool
10. Test arena
   (a) a list of all the question with their marks
   (b) each question followed by options to write, select(form MCQs) or upload solutions
   (c) if the test's finished, an assessment view for each question
11. Explore Page
   (a) a tile view of all the advertisements of all the advertising and trending channels
   (b) a filter form for better search
   (c) each advertisement is expandable to view the channel information
   (d) each advertisement has a join button

- Software Interface
  - **Browser**
    All modern browsers support the interface. If you support older browsers and devices such as Internet Explorer which do not provide modern browser features natively or have non-compliant implementations, consider including a global polyfill in your bundled application. [reference: https://legacy.reactjs.org/docs]
  - **Operating System**
    - Any modern version of Windows
    - Any modern version of MacOS

        - Any modern unix operating system (e.g. Ubuntu, Debian)

- Hardware Interface
    - 2GB RAM or higher for better browser experience
    - Any modern processor is fine as all the heavy tasks will be handled by the servers, and only a static reactJS page will be sent to the client
    - A moderate to high bandwith internet connection as very high amount of data loading may be necessary

4 Supporting Information

- **Installation**
    - **Frontend:** If node is already installed, run the following commands in terminal:
      npm install
      npm run dev
    - **Backend:** If node is already installed, run the following commands in terminal:
      npm install
      npm start

- **Deployment**
  Deploy frontend server into Vercel from the git repository pretty easily. See https://vercel.com/ for more information