**APPLICATION DESIGN SPECIFICATION**

VERSION 1.0

September 19, 2024

**UP-TO-DATE (UTD)**

(News Aggregator Web Application)

**SUBMITTED TO**

Professor Gananand Kini

**SUBMITTED BY**

Sumiran Jaiswal

UMD Directory ID: sumi0309

UMD Email ID: [sumi0309@umd.edu](mailto:sumi0309@umd.edu)

Submitted in partial fulfillment of the requirements of ENPM680 (Introduction to Secure coding for Software Engineering)

**TABLE OF CONTENTS**

Table of Contents…………………………………………………...….…...i

List of Figures…………………………………………………...…….…...ii

Section 1 – Application Description……………………………........1

1.1. Project………………….…………………………………………….......1

1.2. Description………………………………………………………………1

1.3. Revision History…………….………………………………………...…1

Section 2 – Application Overview………………………………….…2

2.1 Purpose……………….…………………………………………………...2

2.2 Scope…………………….………………………………………………..2

2.3 Requirements…………………………………………………………….. 3

2.3.1 Estimates………………...……………………….……………….…..4

2.3.2 Traceability Matrix…………….……………………………………...5

Section 3 – Application System Architecture………………………5

3.1 MVC process flow………………………………………………...……...5

3.2 Components and their interaction………………………………...………7

Section 4 – Application Software Component…………………...10

4.1 Software components…………….……………………………………...10

4.2 Testing……………...…………….……………………………………...11

Section 5 – Application Software UI Components……………..12

Section 6 – Application Data Component…………………...........14

Section 7 – References.............................................................................16

Section 8 – Index…………………………………………………………17

**LIST OF FIGURES**

Figure 1 – MVC Architecture…………………..…......6 Figure 2 – System Interaction Diagram…………….…8

Figure 3 – Activity Diagram...……………...………..14

**SECTION 1 – APPLICATION DESCRIPTION**

**1.1 Project**

UP-TO-DATE

(UTD – a news aggregator)

**1.2 Description**

UTD (Up To Date) is a web application designed to keep users informed by aggregating the latest news in their region. Users can interact with the system in three roles:

1. Authors, who create and post news
2. Readers, who browse and search for news articles
3. Contributors, who suggest edits to improve content.

The application ensures users stay up to date by providing accurate and relevant news with features like search functionality and real-time updates.

|  |  |  |
| --- | --- | --- |
| Date | Comment | Author |
| 09-19-2024 | Version 1.0 completed | Sumiran Jaiswal |
|  |  |  |
|  |  |  |

**1.3 Revision History**

**SECTION 2 – APPLICATION OVERVIEW**

**2.1 Purpose**

The purpose of this document is to provide a detailed design overview of the UTD (Up To Date) web application. This document focuses on the design architecture, module breakdown, and interaction between the components of the system, including user interface components, class and entities, web pages, and database management. The intended audience includes developers, system architects, and stakeholders involved in the project, ensuring that the design is clearly understood for development and implementation purposes.

**2.2 Scope**

The scope of this document is to outline the design specifications for the web application. It details the key modules to be developed, including user interface components, class structures, application pages, and database management. The design will show how Authors, Readers, and Contributors interact with the system, as well as how data is managed. To illustrate these elements, UML diagrams will be used to represent system interactions, and wireframes will be included to show the layout and navigation of the user interface, where needed. This document ensures a clear understanding of the system architecture necessary for successful development and implementation.

**2.3 Requirements**

The UTD (Up To Date) web application will be developed using the following technology stack:

* Technology Stack: The application will be built using ASP.NET Core, a powerful framework for developing web applications. C# will be used as the primary programming language for developing the backend logic of the application.
* MVC Framework: The application will follow the Model-View-Controller (MVC) design pattern. This pattern separates the application into three main components:
  + Model: Manages the behavior and data.
  + View: Handles the presentation layer, displaying data to the user.
  + Controller: Handles page events and navigation between pages.

The MVC design pattern helps to enforce separation of concerns to help you avoid mixing presentation logic, business logic, and data access logic together.

* Database Management: SQL Server will be used for managing the application's database. It will store and handle data related to news articles, user information, and other critical data. The application’s database is set up using an Object-Relational Mapper (ORM). For this project, we will be using Entity Framework, Microsoft’s ORM solution. Entity Framework allows for seamless interaction between the application’s objects (models) and the underlying database. It simplifies data management by automatically mapping objects in the code to the relational database, allowing the application to perform CRUD (Create, Read, Update, Delete) operations with ease.
* Programming Languages: The development will involve several programming languages and technologies:
  + C#: For server-side logic and application development.
  + JavaScript: For client-side interactions and dynamic content.
  + HTML/CSS: For structuring and styling the web pages.
  + CSHTML: For creating dynamic web pages that combine HTML and C# code.

Functional requirements, including features such as the login page, news feed component, news adding interface, and search functionality, are specified in the updated SRS document. This ensures that the design aligns with the application’s intended functionality and user needs.

**2.3.1 Estimates**

|  |  |  |
| --- | --- | --- |
| # | Description | Hrs. Est. |
| 1 | Login page | 3 |
| 2 | News feed component | 6 |
| 3 | News adding interface | 4 |
| 4 | News Search component | 3 |
| 5 | Database designing | 1 |
| 6 | Miscellaneous | 5 |
| 7 | Total | 22 |

**2.3.2 Traceability Matrix**

|  |  |
| --- | --- |
| **SRS Requirement** | **SDD Module** |
| |  | | --- | | Author can create and post news with title, description, time, and place. |  |  | | --- | |  | | |  | | --- | | News Feed, News Add Interface (Model: NewsArticle, NewsController) |  |  | | --- | |  | |
| |  | | --- | | Reader can browse the news feed and view details of each article. |  |  | | --- | |  | | |  | | --- | | News Feed, News Details Page (Model: NewsArticle, NewsController) |  |  | | --- | |  | |
| |  | | --- | | Reader can search for specific news using keywords. |  |  | | --- | |  | | |  | | --- | | Search Page (Search Component, SearchController) |  |  | | --- | |  | |
| |  | | --- | | Contributor can edit and suggest changes to a news article. |  |  | | --- | |  | | |  | | --- | | Edit Page (Model: NewsArticle, ContributorController) |  |  | | --- | |  | |
| |  | | --- | | User login and registration functionality. |  |  | | --- | |  | | |  | | --- | | Login Page (Model: User, LoginController) |  |  | | --- | |  | |
| |  | | --- | | User logout functionality. |  |  | | --- | |  | | Logout (LoginController) |

**SECTION 3 – APPLICATION SYSTEM ARCHITECTURE**

**3.1 MVC process flow**

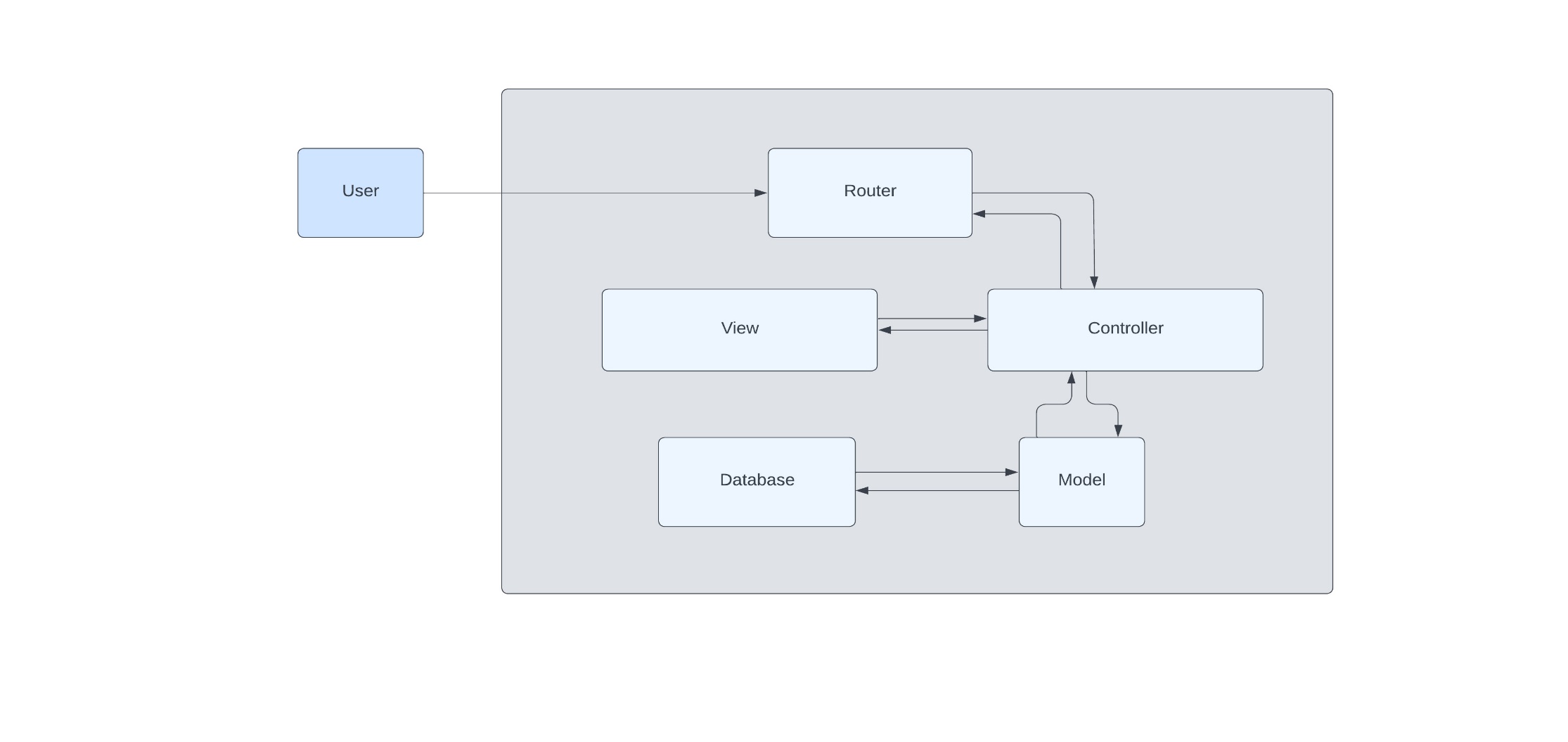
Below is the MVC system general architecture diagram, which represents how communication takes place and the relationships between all the components of the MVC architecture.

Fig. 1 MVC Architecture

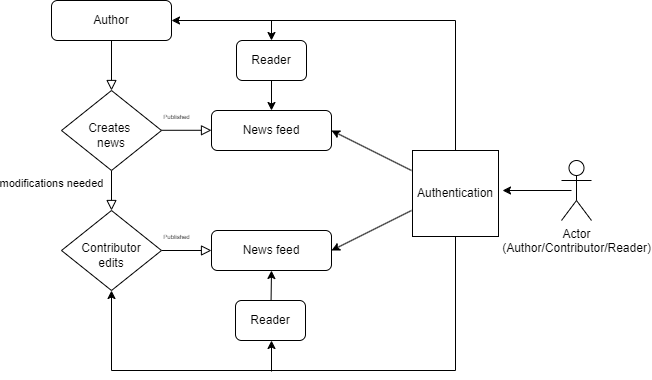
Communication Flow:

1. User Interaction: The user interacts with the application through the View. This interaction triggers an input or request.
2. Router: The request is sent to the Router, which determines how to process the request.
3. Controller: The Router directs the request to the appropriate Controller. The Controller processes the request and interacts with the Model to retrieve or manipulate data.
4. Model: The Model performs the necessary operations on the data and retrieves information from the database.
5. Response: The data retrieved by the Model is sent back to the Controller.
6. View Creation: The Controller then uses this data to create or update the View.
7. Presentation: The updated View is sent back to the user, displaying the relevant information.

This flow ensures that user requests are handled efficiently, with data processed and presented in a structured manner. The separation of concerns provided by the MVC architecture improves the application’s maintainability and scalability, allowing for easier updates and modifications to individual components.

**3.2 Components and their interaction**

Below is a diagram that showcases how the components of the application interact with each other.



Publishing

Publishing



Reader

Reader

Fig. 2 System interaction diagram

The following describes how the components interact within the system:

1. Login:
   * The user, acting as either an Author, Reader, or Contributor, first logs into the system to access the news feed.
2. News Feed:
   * Once logged in, the user can view the news feed. The feed displays the latest news articles published by Authors.
   * The Reader can browse, search, and read news articles from the feed.
3. Author:
   * The Author publishes news by adding a title, description, time, and location. The news article is then added to the feed, making it available to all users.
   * When a new news item is added, the application interacts with the data model using the Entity Framework (EF). The new article is mapped to an entity in the data model (context), which represents the news object.
   * The Entity Framework manages the insertion of this new news object into the database, converting it into a structured format that can be stored and retrieved efficiently.
4. Contributor:
   * If any modifications or updates are needed, the Contributor can make changes to an existing news article. After editing, the Contributor publishes the updated version, which is then reflected in the feed.
5. Interaction Flow:
   * The Author adds and publishes news.
   * The Reader views the news on the feed.
   * If necessary, the Contributor edits the news and publishes updates.

**SECTION 4 – APPLICATION SOFTWARE COMPONENTS**

**4.1 Software Components**

The web application consists of several software components, each playing a crucial role in ensuring smooth functionality and interaction between the different modules. Below is an overview of the key software components:

1. Model Components:

* The Model represents the data structure of the application. Each news article is treated as an object, with details like title, description, time, and place.
* The Entity Framework (EF) is used to map these objects to the database, ensuring that data is correctly stored and retrieved from the SQL Server.
* Models like NewsArticle, User, and Contributor define how the data is handled within the application.

1. Controller Components:

* Controllers manage the flow of information between the UI and the Model. They process user actions, perform necessary operations, and return data to be displayed.
* For example, the NewsController handles the creation, viewing, and editing of news articles by interacting with the Model and preparing the data for the View.

1. Database Management:

* The application uses SQL Server to store and manage all user and news-related data.
* The Entity Framework simplifies database operations, allowing data to be managed through code without writing SQL queries.
* Tables in the database store information about users, news articles, and their interactions.

1. Routing Components:

* The application uses ASP.NET Core MVC's routing to guide user requests to the correct Controllers.
* Whether it is viewing the news feed or adding a new article, routing ensures that each request is properly handled.

**4.2 Testing**

To ensure the UTD(UpToDate) web application functions as intended, various types of testing will be performed throughout the development process. These tests will check different aspects of the application, from individual components to the entire system.

Unittesting will be used to verify that individual components, such as adding or editing news articles, function correctly in isolation.Functionaltesting will check the application's key features like user login, news creation, and editing, ensuring they work as specified in the requirements. Additionally, UItesting will focus on the look, feel, and responsiveness of the user interface, making sure all elements like buttons and forms behave as expected.

**SECTION 5 – APPLICATION SOFTWARE UI COMPONENTS**

UTD’s User Interface (UI) is designed to be simple and user-friendly. Each component is developed using C# with the ASP.NET Core MVC framework. Below are the key UI components:

1. Login Page:
   * Users can either register or log in from this page.
   * It provides fields for username and password, along with a link to register for new users.
   * After logging in, users are directed to the landing page.
2. Landing Page:
   * This page appears after a user logs in and serves as the main navigation area.
   * It includes a navbar with the following options:
     + News Feed: Takes users to a page displaying all news articles.
     + Register/Login: Visible if the user is not logged in.
     + Logout: Allows users to log out.
     + Search News: Directs users to a page where they can search for news by keywords.
3. News Feed Page:
   * This is the main page where all news articles are displayed.
   * Each article shows the headline, author, and an option to view details such as the description, time, and place.
   * Users can also edit or delete news
   * There is an option for Authors to create a new news article.
4. Search Page:
   * Users can search for news articles by entering keywords in the search box.
   * The page displays a list of articles that match the search terms.
5. Edit Page:
   * Contributors can use this page to edit existing news articles.
   * They can change the title, description, time, and place before publishing the updated article.
6. Delete Page:
   * This page allows Authors or Contributors to delete a news article from the system.

Below is an activity diagram to demonstrate how the system operates:

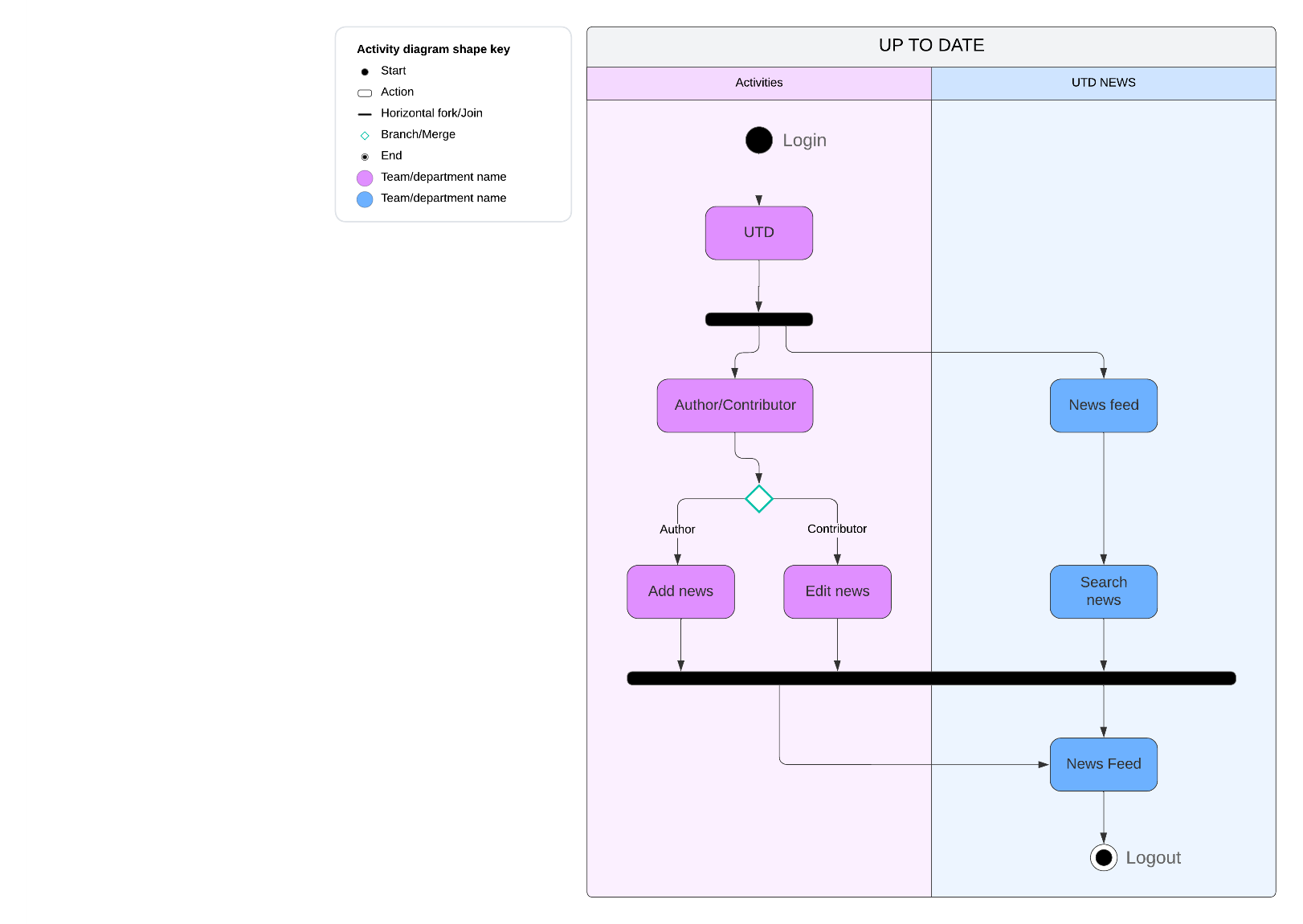


Fig. 3 Activity Diagram

**SECTION 6 – APPLICATION DATA COMPONENTS**

The application data component is designed to manage and store all relevant data for the UTD web application. The database model holds data related to users, news articles, and any interactions between these entities. This is done through a set of classes**/**objects that represent real-world entities like news articles, users, and contributions, each having specific properties (e.g., title, description, time, place, author).

The data flow is managed by:

* SQL statements that interact with the database, enabling CRUD (Create, Read, Update, Delete) operations on the news articles and user information.
* These operations supply the controller with lists of objects representing the news or user data, which are then passed to the view for display.

To manage the database efficiently, migrations are used to update and evolve the database schema. Migrations help in managing changes to the database over time, keeping track of each change and updating the database accordingly without losing data.

We are using EntityFramework, which is Microsoft'sObjectRelationalMapper **(**ORM**)**, to handle the database interactions. This framework allows us to work with the database using C#classes instead of writing complex SQL queries. The ORM maps these classes to database tables, making it easier to handle data through code.

The database is updated and maintained through migrations, ensuring that the structure evolves along with the application without any loss of data or functionality.

**SECTION 7 - REFERENCES**

1. Google Docs. "Sample SRS Document."

Accessed September 2024. <https://docs.google.com/document/d/1pgMutdDasJb6eN6yK6M95JM8gQ16IKacxxhPXgeL9WY/edit#heading=h.lomckg6w5y6>

1. Lucid Chart: "Online Diagram and Flowchart Software." Accessed September 2024.

<https://www.lucidchart.com>

1. Draw.io: "Diagramming Application for Creating Models."

Accessed September 2024.

<https://www.draw.io>

1. UTD Project: "Software Requirement Specification (SRS) Document."

Prepared in September 2024.

**SECTION 8 – INDEX**

Application Data Components………………………….................. 14

Application Software Components……………………................. 10

Application Software UI Components……………….…………. 12

Author……………………………………………………………….... 1, 2, 7

Components and their Interaction……………………..................... 7

Controller Components……………………………......................10, 11

Database Management……………………………………………10, 11

Design Overview……………………………………..…………………2

Data Flow………………………………………...…..…………………..14

Entity Framework………………………………………………….11, 14

Login Page…………………………………………….………………....12

Model Components……………………………………………………10

MVC Process Flow……………………………………..……………....6

News Feed Page………………………………………………………..12

Routing Components…………………………………….................... 11

Software Components………………………………………………...10

Software UI Components……………………………………………12

System Architecture……………………………………………………6

System Interaction……………………………………………………...7

Testing…………………………………………………………………….11

Traceability Matrix…………………………………………………..…5