|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| A picture containing text, outdoor, building, white  Description automatically generated**Student name:** | **Antony Vladimir Montemayor Terrazas**  **Bruna Gonçalves Heleno**  **Elton Nakaoji** | | | | | |
| **Student number:** | **3105325**  **3009733**  **3093840** | | | | | |
| **Faculty:** | **Computing Science** | | | | | |
| **Course:** | **WT** | | | **Stage/year:** | **3** | |
| **Subject:** | **Web Technology** | | | | | |
| **Study Mode:** | Full time |  |  | Full-time |  |  |
| **Lecturer Name:** | **Haseeb Younis** | | | | | |
| **Assignment Title:** | **Assignment 4:** **Final Project** | | | | | |
| Date due: | **01/05/2025** | | |  | | |
| Date submitted: | **01/05/2025** | | |  | | |
| **Plagiarism disclaimer:**  *I understand that plagiarism is a serious offence and have read and understood the college policy on plagiarism. I also understand that I may receive a mark of zero if I have not identified and properly attributed sources which have been used, referred to, or have in any way influenced the preparation of this assignment, or if I have knowingly allowed others to plagiarise my work in this way.*  *I hereby certify that this assignment is my own work, based on my personal study and/or research, and that I have acknowledged all material and sources used in its preparation. I also certify that the assignment has not previously been submitted for assessment and that I have not copied in part or whole or otherwise plagiarised the work of anyone else, including other students.*  **Signed: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | | | | | | |
| Please note: Students MUST retain a hard / soft copy of ALL assignments as well as a receipt issued and signed by a member of Faculty as proof of submission. | | | | | | |

**Q&A WEBSITE**

**Contents**

[1 Introduction 5](#_Toc196906863)

[2 Front-end functionality 5](#_Toc196906864)

[2.1 Common Features 6](#_Toc196906865)

[2.2 Controllers – API Routes 6](#_Toc196906866)

[3 Back-end functionality 7](#_Toc196906867)

[3.1 How to log in and make changes 9](#_Toc196906868)

[3.2 User Groups / Permissions 9](#_Toc196906869)

[4 Pages 9](#_Toc196906870)

[4.1 Index 9](#_Toc196906871)

[4.2 Sign Up 9](#_Toc196906872)

[4.3 Sign In Page 10](#_Toc196906873)

[4.4 About 10](#_Toc196906874)

[4.5 Privacy Policy 11](#_Toc196906875)

[4.6 User 11](#_Toc196906876)

[4.7 Forgot Password 11](#_Toc196906877)

[5 CRUD Operations 11](#_Toc196906878)

[6 List of Features 12](#_Toc196906879)

[7 Technology Stack 13](#_Toc196906880)

[8 Design Specifications 13](#_Toc196906881)

[8.1 Inspired by 13](#_Toc196906882)

[8.2 Sitemap 14](#_Toc196906883)

[8.3 Wireframe Templates 15](#_Toc196906884)

[8.4 Colour 17](#_Toc196906885)

[8.5 Fonts 18](#_Toc196906886)

[8.6 Logo | Favicon 18](#_Toc196906887)

[9 Application Structure 18](#_Toc196906888)

[9.1 Database Structure 19](#_Toc196906889)

[9.2 Diagram of Final Database Design 20](#_Toc196906890)

[10 Project Timeline 21](#_Toc196906891)

[11 Instructions 22](#_Toc196906892)

[11.1 Username and password for team online account 22](#_Toc196906893)

[11.2 Set up instructions 22](#_Toc196906894)

[11.3 How to install on a new machine 22](#_Toc196906895)

[12 Summary of division of Work 22](#_Toc196906896)

[13 Reflection on how the project went 22](#_Toc196906897)

# Introduction

This project is part of the requirement of the discipline Web Technologies of the Computer in Science Course at Griffith College Cork. In this document we are going to talk through the development of the Question and Answer – Q&A – website, since the research/design phase till final result.

Our Q&A website is aimed to developers, providing a safe space to express their ideas. To ask and answer technical questions. The platform includes authentication, CRUD operation, templating, and validation

# Front-end functionality

Front-end was made with React, a library for web and native user interfaces. It is designed to provide an engaging and responsive user experience.

* Responsive Design: the website adapts to various screen sizes, ensuring optimal viewing on desktops, tablets, and mobile devices. This responsiveness is achieved through the use of SCSS and flexible layouts.
* Interactive Elements: interactive components such as navigation menus, buttons, and text to enhance user engagement.
* Smooth Navigation: intuitive navigation, pages have same layout, allowing users to access different sections such as About, Privacy Policy, Sign In, Home without problem.
* Performance Optimization: using a common layout component across pages to keep things fast and consistent. It helps avoid re-rendering shared UI elements like navbar and footer on every page change.
* Accessibility Compliance: the website was designed with accessibility in mind, incorporating recognizable icons for intuitive navigation, colour choices tested for sufficient contrast ratios, legible typography for improved readability, and keyboard navigability.
* Modern Aesthetics: utilizing design principles, the website’s designed aimed for a clean and professional appearance. It uses consistent colour schemes, typography, and spacing, enhancing readability and visual appeal.
* Integration with External Services: through APIs website connects to MongoDB database to fetch user data, handle form submissions, and create new user accounts.

## Common Features

**Navbar**:

* It has a logo icon that redirects to home page.
* A home icon that also redirects to home page
* User icon that redirects to Sign In form



Figure 1: Navbar

**Footer**:

* About link: redirects to About page
* Privacy Policy: redirects to Privacy Policy page



Figure 2: Footer

## Controllers – API Routes

**Authentication**

|  |  |  |
| --- | --- | --- |
| **Method** | **Route** | **Description** |
| GET | /login | Displaying login page |
| POST | /login | Authentication user & return JWT |
| GET | /logout | Logs out the user (client must discard token) |

**User Management**

|  |  |  |
| --- | --- | --- |
| **Method** | **Route** | **Description** |
| GET | /user | Fetch logged-in user profile |
| GET | /user/:name | View a public user profile |
| POST | /user/status | Update user status |
| POST | /user/privacy | Set user privacy settings |
| POST | /user/php | Upload profile picture |

**Messaging**

|  |  |  |
| --- | --- | --- |
| **Method** | **Route** | **Description** |
| POST | /user/getM | Fetch received messages |
| POST | /user/refuse | Delete a receive message |
| POST | /user/replay | Reply to a message |
| POST | /post/:userID | Send a message |

**Messages API**

|  |  |  |
| --- | --- | --- |
| **Method** | **Route** | **Description** |
| GET | /api/messages | Get all messages |
| POST | /api/messages | Create a new message |
| GET | /api/messages/:id | Get a specific message |
| PUT | /api/messages/:id | Update a message |
| DELETE | /api/messages/:id | Delete a message |
| GET | /api/messages/search/:query | Search messages |

# Back-end functionality

* User Authentication & Session Management: manages user accounts, enabling sign-in, sign-up, and logout functionalities. It handles password hashing, token-based authentication (JWT), ensuring safe access and persistent user sessions across visits.
* Data Storage and Retrieval: user profiles and messages are stored in MongoDB database. It uses API endpoint to create, read, update, and delete (CRUD) this data, ensuring seamless communication between the client and server.
* Form handling and Input Processing: data submitted is processed on the server, validated for correctness, and then stored. Protecting against malformed or malicious inputs.
* Access Control and Privacy Settings: It enforces access rules based on user preferences. Profiles may be marked as public, private, or only friends, with backend logic ensuring correct visibility and access restrictions are applied.
* Scalability and Maintainability: The backend was designed with modularity in mind, separating authentication, routing, database access in its architecture, making the system easier to maintain, extend, and scale as more features or users are added.
* Security Measures: It implements HTTPS protocols, input sanitization, to encrypt data in transit to protect both data and users.

## How to log in and make changes

To log in and make changes, user is required to be registered on the database. To register user has to go to SIGN UP page, fill the form, providing their Username, Password and Privacy, Profile Picture and Status are optional. Having their information accepted, user can log in on the system.

SIGN IN page asks for username and password, if the information provided match with the information stored on the database, user is allowed to access the system and makes changes to their profile and messages.

## User Groups / Permissions

# Pages

## Index

A screenshot of a computer

AI-generated content may be incorrect.

Figure 3: Home Page

## Sign Up

A screenshot of a computer

AI-generated content may be incorrect.

Figure 4: Sign Up Page

## Sign In Page

A screenshot of a computer

AI-generated content may be incorrect.

Figure 5: Sign In Page

## About

A screenshot of a computer

AI-generated content may be incorrect.

Figure 6: About Page

## Privacy Policy

A computer screen shot of a policy

AI-generated content may be incorrect.

Figure 7: Privacy Policy Page

This page shows a privacy policy image, just to show case how would be; however, it is not a real privacy policy.

## User

A blue screen with white text

AI-generated content may be incorrect.

Figure 8: User Page

## Forgot Password

# CRUD Operations

This project implements full CRUD (Create, Read, Update, Delete) operattions for managing users and messages within a social messaging system. Users can create accounts, update their profile information, set privacy preferences, and manage their statuses. Authentication is handled through a login system that uses JWT tokens for secure access. Messages can be created by users, either as direct messages or wall posts, and recipients can choose to accept, refuse, or reply them. The system also allows users to retrieve (read) their messages, including searching for specific content. Messages can be updated with new content and deleted.

# List of Features

* User Authentication:
  + Users can login and logout
    - Email token generation
    - Display error on incorrect email/password
  + Users can add profile pictures
  + Users can set their status
  + Users can set their privacy to allow who can send messages to them
* User profile:
  + Users can upload a profile picture
  + Set a status
  + Side Navigation menu
* Q&A Posting
  + Users can post, edit, and delete questions
  + Users can reply to messages
  + Users can refuse messages
* Search & Filtering
  + Users can search over content
* Responsive UI
  + Works across devices with react frontend
* Database Management
  + MongoDB for storing user data and messages (questions and answers)
* Session Handling
  + Uses cookies and session for state maintenance
* Deployment
  + Hosted online: https://www.tripleequal.dev/7

# Technology Stack

* React: Frontend UI
* Node.js: Backend Server
* Express.js: API routing & middleware
* MongoDB: Database
* Jest and Supertest: Unit Test
* Joi: validation
* JSON Web Token (JWT): Authentication
* Languages
* JavaScript
* HTML
* TypeScript
* SCSS

# Design Specifications

## Inspired by

Three main websites were used as inspiration:

* Stack Overflow: Leading Q&A platform, strong community, stringent moderation
* Quora: General Q&A site, broader topics, but less technical than Stack Overflow
* Reddit (Programming Subreddits): Community based discussion but lacks specialized Q&A structure

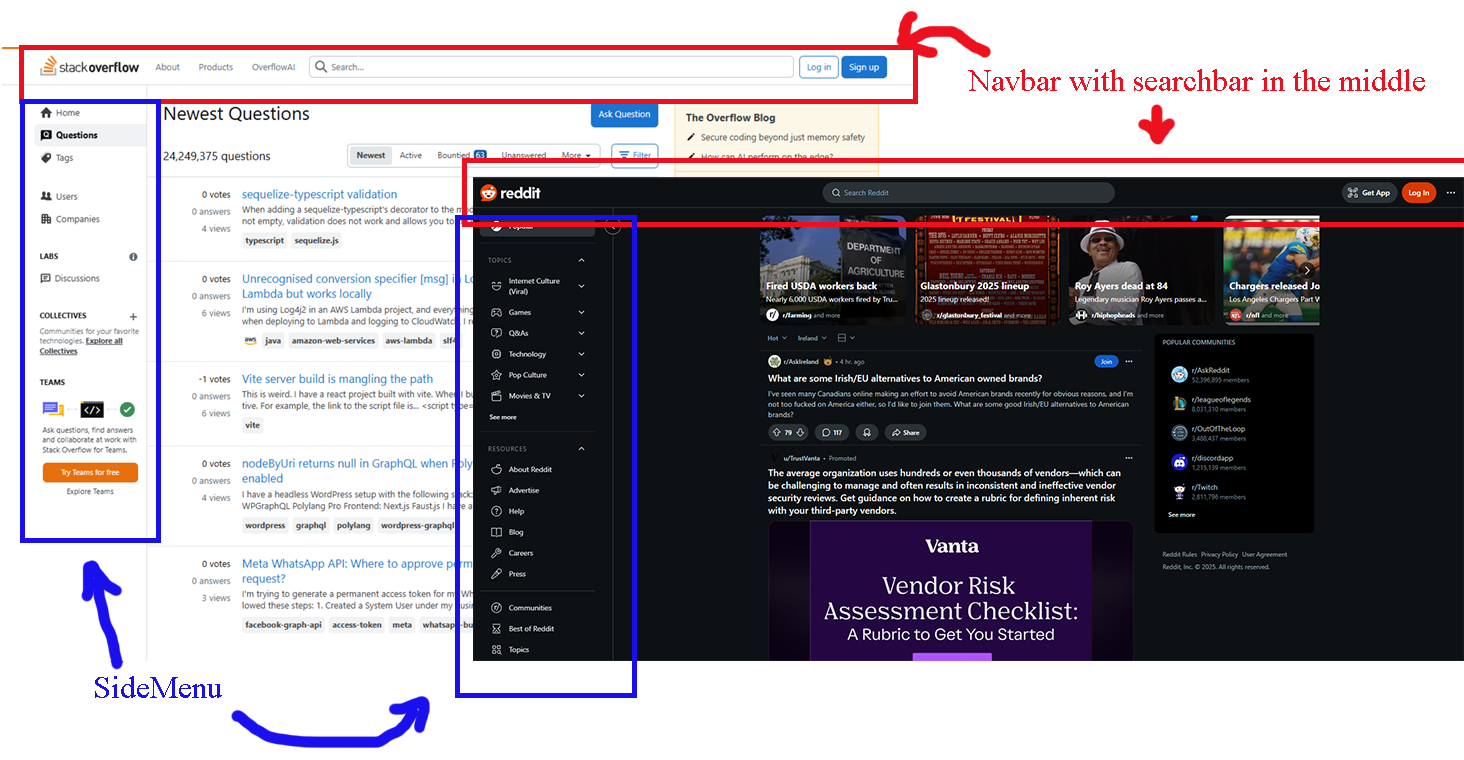


Figure 9: Design Inspiration

## Sitemap

All pages are connected through navbar and footer.

A diagram of a company

AI-generated content may be incorrect.

Figure 10: Sitemap Diagram

|  |  |  |
| --- | --- | --- |
| **Path** | **Method** | **Description** |
| /home | GET | Main page |
| /user/ | GET | User displayed of the user profile |
| /user/:name | GET | User interface |
| /login | GET | Login Interface |
| /login | POST | Method to get cookies and redirect to ‘/user’ |
| /logout | GET | Delete cookies and redirect to ‘/login’ |
| /user/getM | POST | Get messages to display in ‘/user’ |
| /user/refuse | POST | Delete a message posted for a user |
| /user/replay | POST | Replay a message |
| /user/php | POST | Adds a profile picture |
| /user/status | POST | Adds a status |
| /user/privacy | POST | Allows the user to decide who can send them messages |
| /post/:userID | POST | Add a message to the user |
| /about | GET | About the project |
| privacy | GET | Privacy |

## Wireframe Templates

A screenshot of a computer

AI-generated content may be incorrect.

Figure 11: Main Layout

A screenshot of a computer

AI-generated content may be incorrect.

Figure 12: User Page Layout

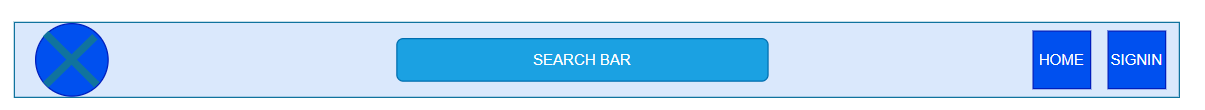


Figure 13: Navbar Layout

A screenshot of a computer

AI-generated content may be incorrect.

Figure 14: User Sidebar Layout

## Colour

|  |  |  |
| --- | --- | --- |
| **Code** | **Name** | **Usage** |
| #f5f5f5 | White Smoke | Text and input background |
| #3333cc | Palatinate Blue | Texts - hidden |
| #ffcc00 | Tangerine Yellow | Highligth and buttons |
| #111178 | Soft Midnight | Navbar background |
| #1b1b6e | Midnight Blue | Background page |
| #3333CC | Darkblue | Images highlitgh / Sign In |

## Fonts

* Roboto for clarity and modern appearence | Mainly used in titles
* OpenSans for fall back and clarity | Mainly used in paragraphs
* Helvetica Neue for inputs and textarea

## Logo | Favicon

Logo was generated by AI. It represents a developer in front of a computer surrounded by stylized tech elements, in a flat vector style with a colour palette of blues, dark yellow and white, set against a dark background. The developer is half man, half woman to represent all genres of programming developers



Figure 15: Logo

# Application Structure

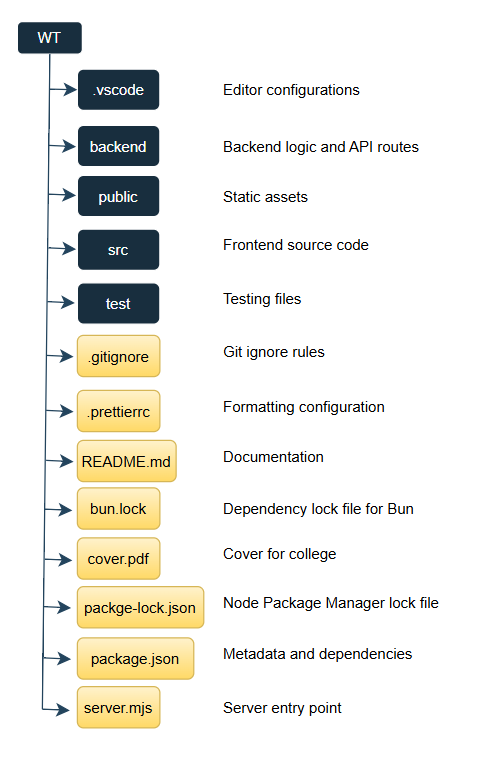


Figure 16: Folder Structure

## Database Structure

**Users Collection**

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| \_id | ObjectID | Unique identifier for the user |
| name | String | Full name of the user |
| username | String | Unique username |
| password | String | Hashed password |
| profilePicture | String | URL or path to profile picture |
| status | String | User’s status message |
| privacy | String | Privacy setting for messages (public, friends, private) |
| createdAt | Date | Timestamp of account creation |
| updatedAt | Date | Timestamp of last update |

**Messages Collection**

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| \_id | ObjectId | Unique identifier for the message |
| content | String | Message text content |
| sender | ObjectId | Reference to the sender (User) |
| recipient | ObjectId | Reference to the recipient (User) |
| replies | Array | List of reply objects (content, date) |
| createdAt | Date | Timestamp of when message was created |
| updatedAt | Date | Timestamp of last message update |

**User Example**

A computer screen with white text and green text

AI-generated content may be incorrect.

Figure 17: User Example

## Diagram of Final Database Design

A diagram of a server

AI-generated content may be incorrect.

Figure 18: Database Diagram

# Project Timeline

|  |  |
| --- | --- |
| **Task** | **Duration** |
| Start project: gathering information, research, user cases, design specifications | 2 weeks |
| Milestone 1: Project Proposal and Design Documentation | Due date 07/03 |
| Server-side components:   * Backend Setup (Node.js): set up server, routes, API endpoints * Configure MongoDB, design schemas for users, Q&A data | 2 weeks |
| Milestone 2: Server-side documentation and demo | Due date 23/03 |
| Client-side:   * Frontend development (React): build responsive pages * Integrate with RESTful backend | 2 weeks |
| Milestone 3: Client-side documentation and demo | Due date 16/04 |
| Testing and Q&A:   * Conduct unit tests, integration tests, user acceptance | Throughout project |
| Deployment and Final Review:   * Deploy to hosting environment * Final checks and fixes | 1 week |
| Milestone 4: Final Project documentation and demo | Due date 01/05 |

# Instructions

## Username and password for team online account

## Set up instructions

## How to install on a new machine

# Summary of division of Work

Work was evenly divided.

# Reflection on how the project went

This project helped us to deepen our understanding of full-stack development. We had the opportunity to work with React, a framework that we had not use extensively before. This made us more comfortable with hooks, component-based architecture, and managing state across different parts of the application. React’s flexibility and reusability made it easier to build dynamic, interactive features while maintaining clean code organization.

Another, and probably the most important part of the project, was learning how to integrate the front-end with a backend API, allowing real-time communication with MongoDB database. This helped us to understand the structure of RESTful APIs and how to manage data flow between client and server. Working with fetch requests, authentication, and CRUD operations in a real application environment to build scalable and functional web apps.

Additionally, we used GitHub for version control, what allows us to practice commit messages, branching for new features, and resolving merge conflicts. Making collaboration smoother and maintaining a clear development history. Altogether, this project gave us practical experience that will be used in future projects.