GRST Programming Challenge Aggregator

Prepared by: Matthew Griffin, Ashish Rastogi, Kiritbhai Soheliya, Martin Tiernan

Software Engineering, Fort Hays State University

https://github.com/mgriffin3/GRST

Table of Contents

Customer Statement of Requirements (CSR) 2

Glossary of Terms 4

System Requirements 5

Functional Requirements Specification 9

User Interface Specification 18

Project Management 23

1. **Customer Statement of Requirements (CSR)**

**1.1 Problem Statement**

There are many websites that offer programming challenges. Typically, these sites are used by an interviewee preparing for a job interview, or a computer science student furthering their education. These sites vary in usability and focus. If you are looking to practice and hone your programming skills across a wide variety of topics for potential upcoming interviews or to help study for upcoming classes, you face a variety of problems. A typical sequence if you are looking for an appropriate challenge or challenges is as follows:

1. **Navigate the web to an appropriate site**
2. **Learn the site U.I. and navigation**
3. **Find a challenge that suits your interests**
4. **Repeat 1-3 until sufficiently prepared**

This current process leaves a lot to be desired and that forces the user to spend an unnecessary amount of time on web browsing and learning various site navigation. The potential solution should be one that eliminates the tedium that comes from having to repeat steps one and two of this process and save the users valuable time that they could use to prepare for an interview or an upcoming test.

**1.2 Solution**

The proposed solution should present these challenges from multiple sites and present them in a unified design that has a centralized location. Ideally the new sequence should be:

1. **Download/Access the app/site**
2. **Learn the site U.I. and navigation**
3. **Find a challenge that suits your interests**
4. **Repeat 3 until sufficiently prepared**

Users should no longer have to find challenge sites on their own, and they will be able to view challenges from other sites without having to learn a new interface. Thus, eliminating the repletion in steps one and two and ultimately saving the user time. Finally, by pulling from many sources the user will be offered with the broadest possible selection of challenges that they can find at a single location.

The proposed solution must be able to gather and store adequate information about each challenge. This information should be reliable, and a user should not have to worry about a challenge “disappearing” and be forced to scour the web. It should gather enough information that a user should be able to complete a challenge without the need of going to the source website, however they should be able to find the source website if desired.

To ensure that users can adequately locate and complete challenges, a serviceable interface must be provided so they can find challenges to their liking. Showing every problem from all the included sites could overwhelm potential users. Users must be able to use a menu to search and filter the database to limit the quantity of entries to a more manageable form. They may only be interested in problems from a single or several source sites, or only looking for problems of an easy difficulty. Finally, the results must be presented in a way that users can view and find the complete text and other challenge specific information.

A secondary goal of the project should be a more personalized user experience. This includes allowing the user to upload their own personal challenges. Also, a user may want to be to create a list of favorite challenges and save them for later. Users may want to be able to toggle the status of challenges between solved, viewed but not solved, not viewed, skipped, and newly downloaded. A user may wish to add text or attach files associated with the challenges, so they can revisit their implementations or, so they can review past solutions. These functions would allow a user a unique view of the site and provide additional advantages if they do continually use the product over time.

The only true way that this project solves the problem if a user finds it easier to locate challenges than manually going each site. Great care must be taken when creating the interface. The time to learn the created interface must be significantly lower than the time to learn the sum of the websites interfaces. Additionally, if searching the database is slow and unreliable, then the original process of visiting each site might be more convenient to the user. It is not enough to simply present the user with access to all challenges. A user **must** save time when comparing with the traditional method to be a true solution.

1. **Glossary of Terms**

**Challenge**- A programming challenge commonly found on the internet or in computer science coursework. Typically focuses on learning a specific concept over practical application.

**User-** A person that is using the application.

**Database –** the storage of all the collected challenges and text associated with the challenges and accompanying websites.

**Web Scraper-** A program that gathers publicly available data from the internet and typically stores it in a database.

**User Interface-** The interface that lets the user interact with and search within the database.

**Search/Query-** A user filters or limits the results using certain parameters such as site origin.

**Results-** The list of challenges that is presented to the user after a search query.

1. System Requirements Engineering
   1. Enumerated Front-end Functional Requirements

|  |  |  |
| --- | --- | --- |
| Identifier | Requirement | PW |
| REQ1 | The system shall allow user to search challenges using keywords | 1 |
| REQ2 | The system shall allow user to filter the searched result | 3 |
| REQ3 | The system shall allow user to mark challenges as complete | 5 |
| REQ4 | The system shall allow user to register | 4 |
| REQ5 | The system shall allow user to login and manage profile | 4 |
| REQ6 | The system shall allow user to save challenges into user profile | 5 |
| REQ7 | The system shall allow user to view challenges | 2 |

The key objective for this system is to allow users to search the challenges from the consolidated database. So, REQ1 has the highest priority, because emphasis is that users shall be able to search with no string attached.

Going with the natural flow of user expectations, users shall be able to view challenge/s that makes it a Second priority (REQ7) requirement.

User interaction can be made impactful if advanced search and filter capability is provided to user, that brings us to define the REQ2 and third highest priority requirement.

Generally, users use the challenge websites and resources at events where they need to appear for an interview or brush-up the skillset, with that in mind users don’t really want to create username and password if they had to use resources for limited period of time.

It’s an optional requirement for a user to register as a new user, login with registered user and manage profile to view history on search results, change password, change avatar etc., REQ4-5 will serve the purpose.

REQ3 & REQ6 are the lower priority requirements, serves as next step in using the resources from such a site

* 1. Enumerated back-end functional Requirements.

|  |  |
| --- | --- |
| Identifier | Requirement |
| REQ1 | The system shall run periodical web-scrapping queries from all listed website. |
| REQ2 | The system shall record result of web-scrapping queries in local database |
| REQ3 | The system shall allow administrator to initiate web-scrapping queries |
| REQ4 | The system shall allow administrator to add or remove challenges. |

The basic idea of this system is to gather various challenges from several websites and provide a one-stop tool to users. The primary requirement REQ1 is to allow the system to run automated web-scrapping queries periodically.

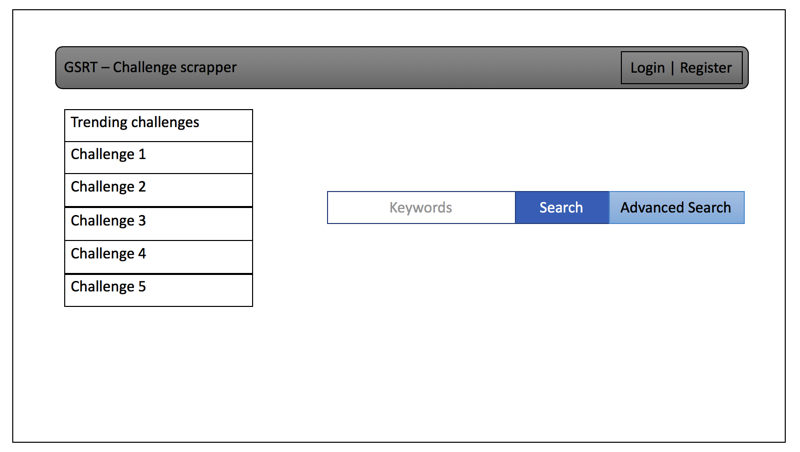
System will have local database to store all challenges, so that system doesn’t have to rely on external storage which can induces network related issues for the users, It is covered in REQ2.

The system will also allow administrator/s to run ad-hoc queries as needed.

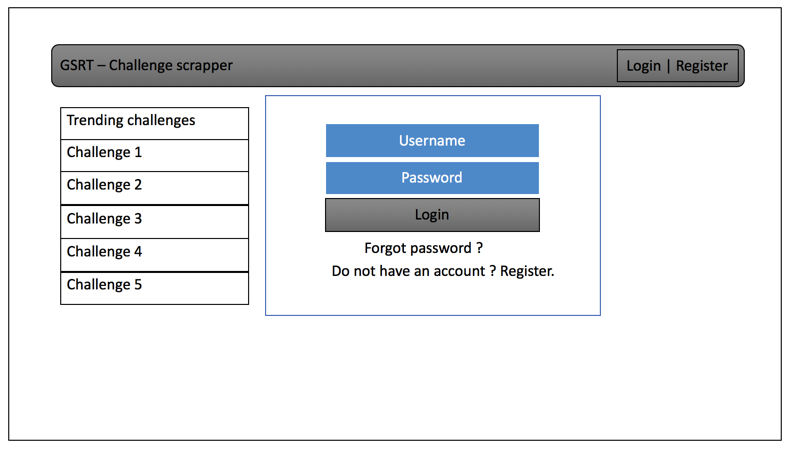
System will also provide administrative capability to add or remove challenges on need basis.

On-Screen Appearance Requirements

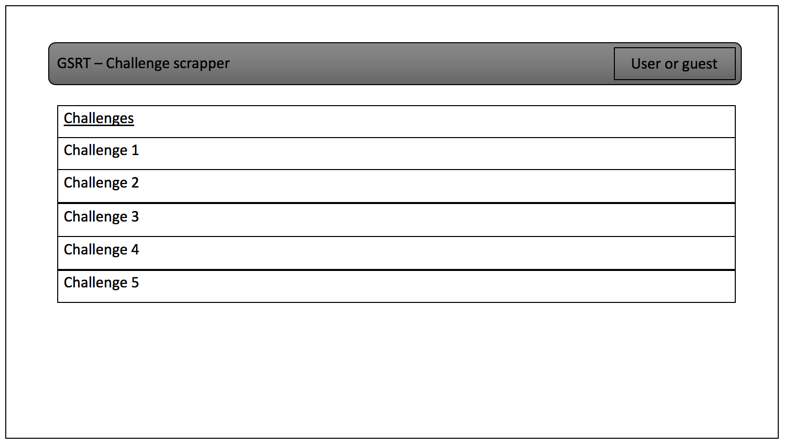
Page 1. This is the landing page of the website.



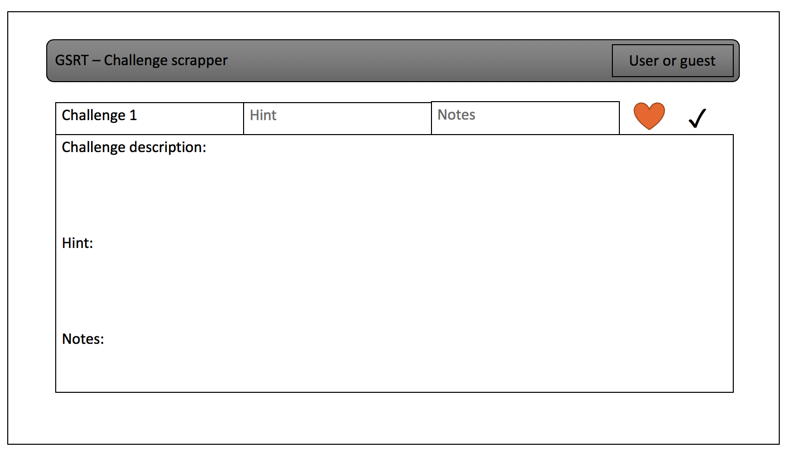
Page 2. User login page of the website, register page will have the same view. It’s a minimalistic design.



Page 3. It will display the challenges from the search criteria.



Page 4. Individual search results or challenges can be view separately, making notes capability is provided and will have the hint section too.



4. Functional Requirements Specification

Stakeholders:

This system is created for anyone seeking programming challenges online. This system will also be of interest to hiring managers and recruiters seeking to test incoming interviewees. Some examples of those interested are:

1. Computer Science undergraduate and graduate students
2. Programming teachers or professors
3. People seeking employment in software development
4. Hiring managers and recruiters at software firms
5. Hobbyist programmers

Additionally, the sites originating the challenges are also stakeholders. Our program relies on them for content, but it will expose users to challenge websites that they may never have heard of otherwise.

Actors:

We have identified four main actors in this system

1. User- a user that has registered in the system
2. Administrator- an administrator of the system responsible for updating/maintain database
3. Database- Stores programming challenges to present to user
4. External Sites- The websites where many of the challenges originate.

Use Case Casual Description:

We have separated the use case in to two categories one set for the front-end or a user searching the database, and one set for the backend for updating and maintaining.

Front-end Use Cases:

UC1: Search: allows a user to search the database and be presented with a list of appropriate challenges filtered too their desire.

UC2: View: allows user to view selected challenge and get the details

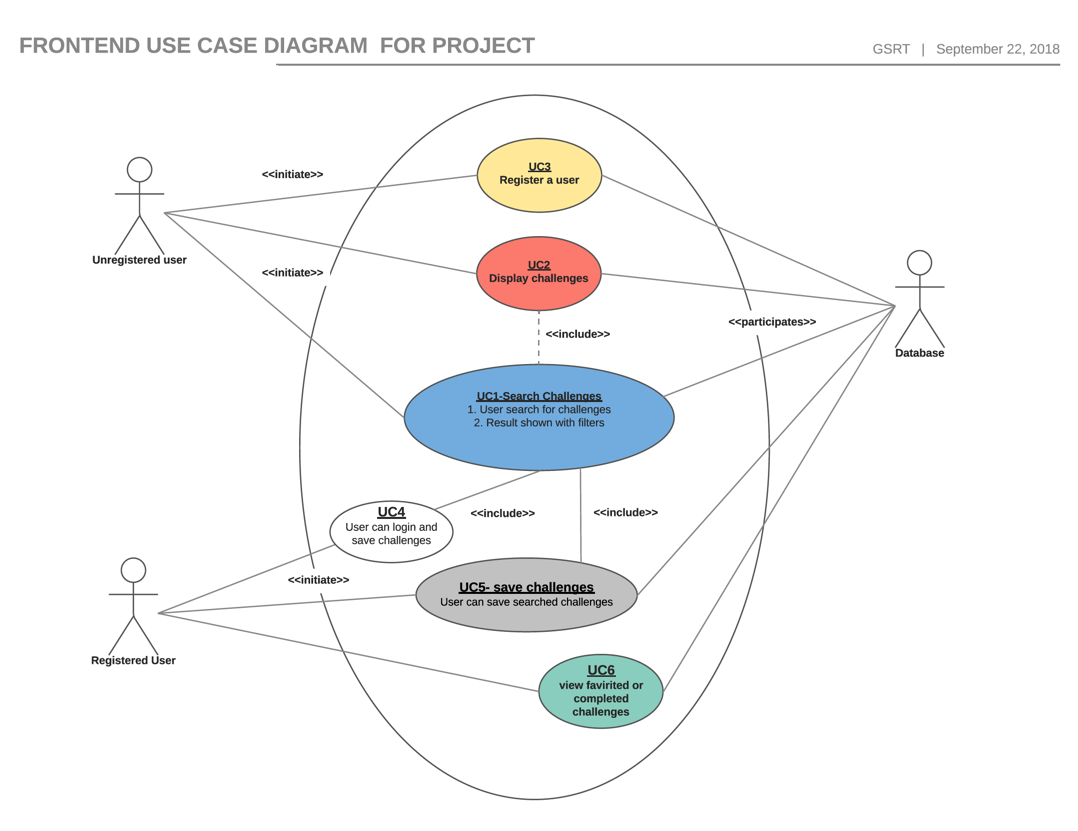
UC3: Register: a user registers an account in the system and store challenge specific information.

UC4: login: allows a user to login a view saved challenges.

UC5: Mark: allows a user to mark a viewed challenge as a favorite or completed.

UC6: View Saved: allows a user to view there favorited, or completed challenges

Front-end Use case diagram for GRST project



Front-end Traceability Matrix

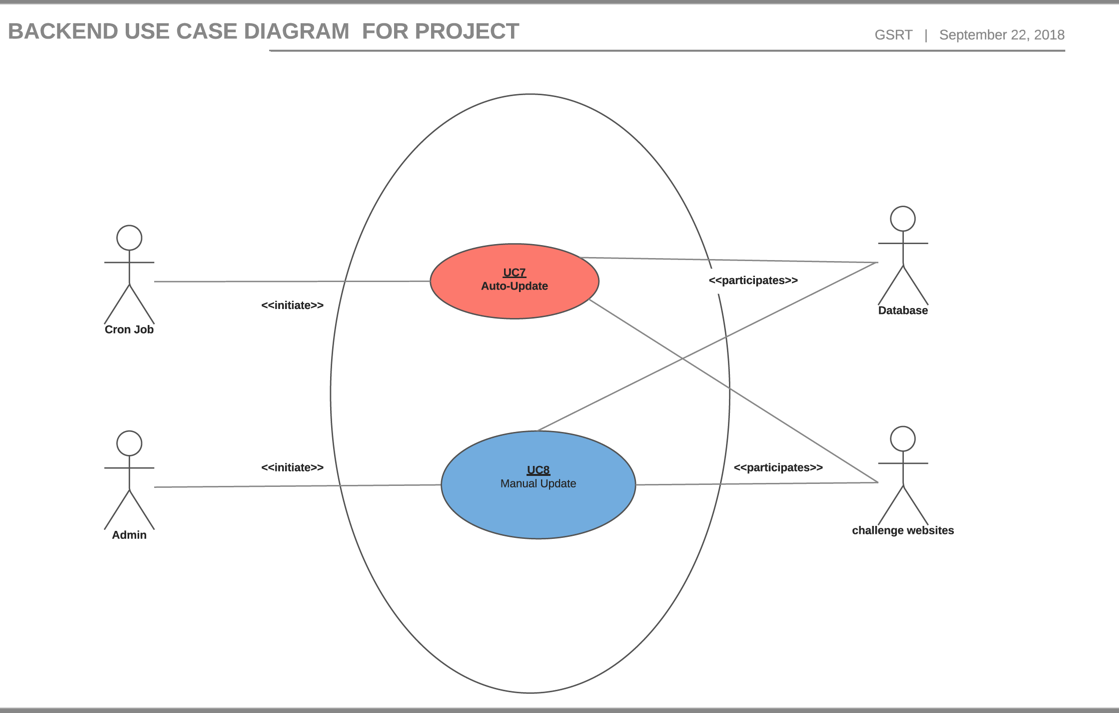
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **REQ** | **PW** | **UC1** | | **UC2** | | **UC3** | | **UC4** | | **UC5** | | **UC6** |
| **REQ1** | 1 | X | |  | |  | |  | |  | |  |
| **REQ2** | 3 | X | |  | |  | |  | |  | |  |
| **REQ3** | 5 |  | |  | |  | |  | | X | |  |
| **REQ4** | 4 |  | |  | | X | |  | |  | |  |
| **REQ5** | 4 |  | |  | |  | | X | |  | |  |
| **REQ6** | 5 |  | |  | |  | |  | | X | | X |
| **REQ7** | 2 |  | | X | |  | |  | |  | | X |
| **Max PW** | | 3 | 2 | | 4 | | 4 | | 5 | | 5 | |
| **Total PW** | | 4 | 2 | | 4 | | 4 | | 10 | | 7 | |

Back—end Use case:

UC7: Auto-Update: allows the system to update the database by querying the listed sties at set intervals

UC8: Manual-Update: allows an administrator to update the database by issuing queries or by adding or removing challenges selected by the administrator.

Back-end Use case diagram for GRST project



Back-end Traceability Matrix

|  |  |  |  |
| --- | --- | --- | --- |
| **REQ** | **PW** | **UC7** | **UC8** |
| **REQ1** | 1 | X |  |
| **REQ2** | 1 | X |  |
| **REQ3** | 1 |  | X |
| **REQ4** | 1 |  | X |
| **Max PW** | | 1 | 1 |
| **Total PW** | | 2 | 2 |

Front-end FDUC

|  |  |
| --- | --- |
| Use Case UC-1 & UC-2: | Search & View |
| Related Requirements: | REQ1 – REQ2 |
| Initiating Actor: | User |
| Actor’s Goal: | Search the programming challenges |
| Participating Actors: | Database |
| Preconditions: | User has stable Internet connection and updated browser |
| Success End Condition: | User is able to search and view desired programming challenge |
| Failed End Condition: | Search return empty list |
| Extension Points: |  |
| Flow of Events for Main Success Scenario: |  |
|  | **→** User opens web application through a web browser. |
|  | **←** Website loads a home page which has a trending challenges and an option to search challenges |
|  | → User clicks on the search box and enter a search query and press enter or click search button |
|  | Now UI will run a search query on the database. |
|  | ← UI will display the search result in a table format. Each challenge is a clickable link with metadata shown. |
|  | The data table will have options to filter and sort the entries. |
|  | → User will enter desired filter query into filter box. |
|  | → User will click on chosen challenge to view it. |
|  | ← Website will open the challenge page with detailed description, hint and optional notes tab for the particular challenge |
|  | Notes tab will be available for registered and logged in user. |

|  |  |
| --- | --- |
| Use Case UC-3 & UC-5: | Register & Mark |
| Related Requirements: | REQ-3 & REQ-4 |
| Initiating Actor: | User |
| Actor’s Goal: | A user registers an account in the system and store challenge specific information. |
| Participating Actors: | Database |
| Preconditions: | Unregistered user has stable Internet connection and updated browser |
| Success End Condition: | User is able to register a new account. User should be able to search, view & save desired programming challenge. |
| Failed End Condition: | User registration failed |
| Extension Points: |  |
| Flow of Events for Main Success Scenario: |  |
|  | **→** User opens web application through a web browser. |
|  | **←** Website loads a home page which has a trending challenges and an option to search challenges |
|  | → User click on register button |
|  | ← System loads user registration page. |
|  | → User inputs username, Full name and password. Then click register button. |
|  | Now system will query database in a backend to create username entry in users table. |
|  | ← Upon successful entry creation, website will display success message and keep the user logged in. |
|  | → User clicks on the search box and enter a search query and press enter or click search button |
|  | Now UI will run a search query on the database. |
|  | ← UI will display the search result in a table format. Each challenge is a clickable link with metadata shown. |
|  | → User will click on chosen challenge to view it. |
|  | ← Website will open the challenge page with detailed description, hint and optional notes tab for the particular challenge |
|  | Notes tab will be available for registered and logged in user. |
|  | → Now user will click on favorite icon (heart icon) to save the challenge in user profile. |
|  | UI will query the database to save this challenge in user profile table. |
|  | → UI will display the challenge display page with up to date user profile info. |

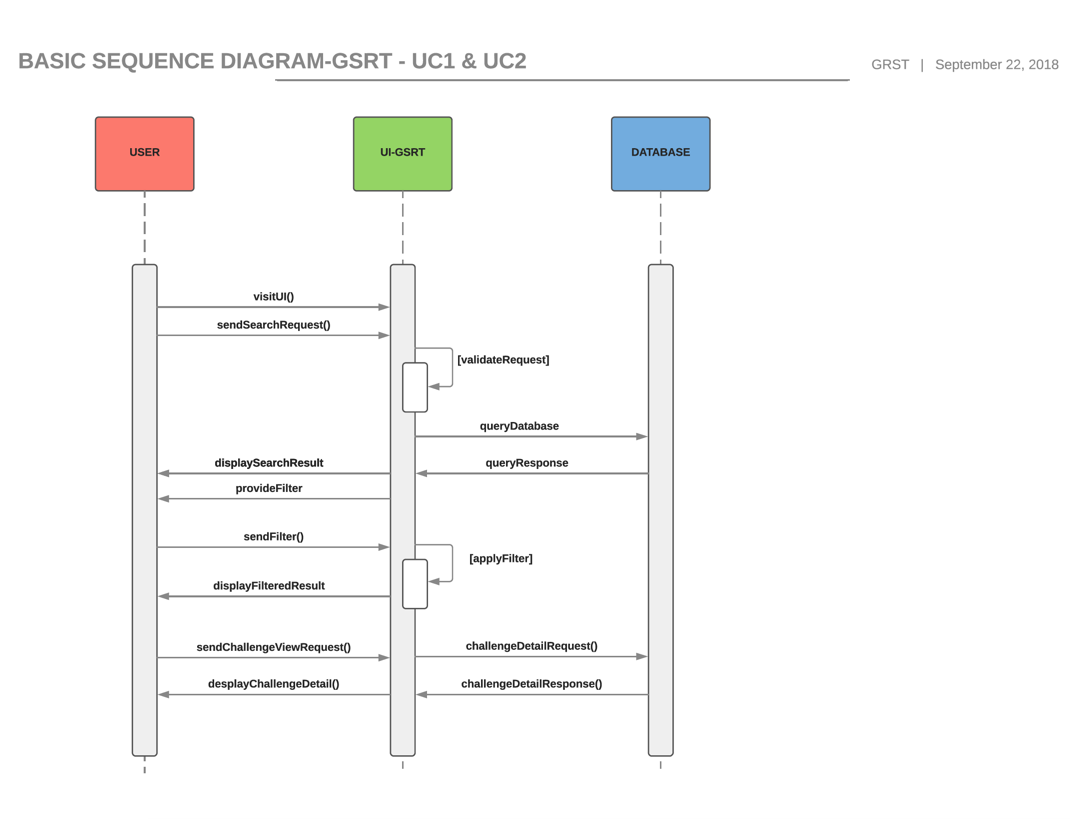
|  |  |
| --- | --- |
| Use Case UC-4: | Login |
| Related Requirements: | REQ5 |
| Initiating Actor: | User |
| Actor’s Goal: | Allow user to login and view saved challenges |
| Participating Actors: | Database |
| Preconditions: | Registered user has stable Internet connection and updated browser |
| Success End Condition: | User is able to login and view user profile |
| Failed End Condition: | User failed to login |
| Extension Points: |  |
| Flow of Events for Main Success Scenario: |  |
|  | **→** User opens web application through a web browser. |
|  | **←** Website loads a home page which has a trending challenges and an option to search challenges |
|  | → User click on login button |
|  | ← System loads user login page. |
|  | → User inputs username and password. Then click login button. |
|  | Now system will query database in a backend to validate user credentials in user table. |
|  | ← Upon successful user validation, website will load user profile page with user specific information like saved and solved challenges. |
|  | → User clicks on a favorite tab to view all saved challenges. |
|  | ← UI will display the favorite challenges list in the table format. |
|  | → User will click on chosen challenge to view it. |
|  | ← Website will open the challenge page with detailed description, hint and optional notes tab for the particular challenge |
|  | Notes tab will be available for registered and logged in user. |

back-end FDUCs

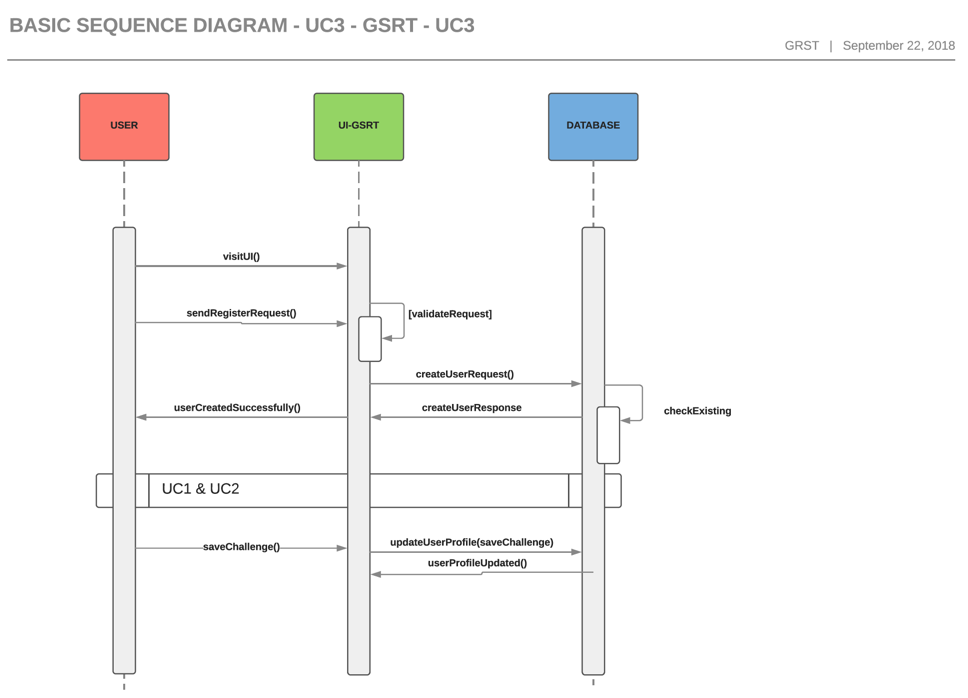
|  |  |
| --- | --- |
| Use Case UC-7: | Auto-Update |
| Related Requirements: | REQ 1 & 2 |
| Initiating Actor: | System Cron Job |
| Actor’s Goal: | System is able to fetch challenges from participating websites. |
| Participating Actors: | Participating challenge websites & Database |
| Preconditions: | Cron job is setup. |
| Success End Condition: | Database is updated with latest challenges from participating websites. |
| Failed End Condition: | Database entry failed to update. |
| Extension Points: |  |
| Flow of Events for Main Success Scenario: |  |
|  | **→** System starts the backend web scrapper scripts periodically as specified in cron job. |
|  | **←** Website will respond to web scrapper queries |
|  | → Web scrapper script will reformat the responses for local databases. |
|  | → Web scrapper script will save the latest result into the challenges table |

|  |  |
| --- | --- |
| Use Case UC-8: | Manual-Update |
| Related Requirements: | REQ 3 & 4 |
| Initiating Actor: | System Admin |
| Actor’s Goal: | System is able to fetch challenges from participating websites on demand. |
| Participating Actors: | Participating challenge websites & Database |
| Preconditions: | System user with admin privileges. |
| Success End Condition: | Database is updated with latest challenges from participating websites. |
| Failed End Condition: | Database entry failed to update. |
| Extension Points: |  |
| Flow of Events for Main Success Scenario: |  |
|  | → Admin user logs in to the system using a secure CLI session |
|  | → Admin user starts the backend web scrapper scripts using provided CLI. |
|  | **←** Website will respond to web scrapper queries |
|  | → Web scrapper script will reformat the responses for local databases. |
|  | → Web scrapper script will save the latest result into the challenges table |
|  | ← System will print a successful message on terminal. |

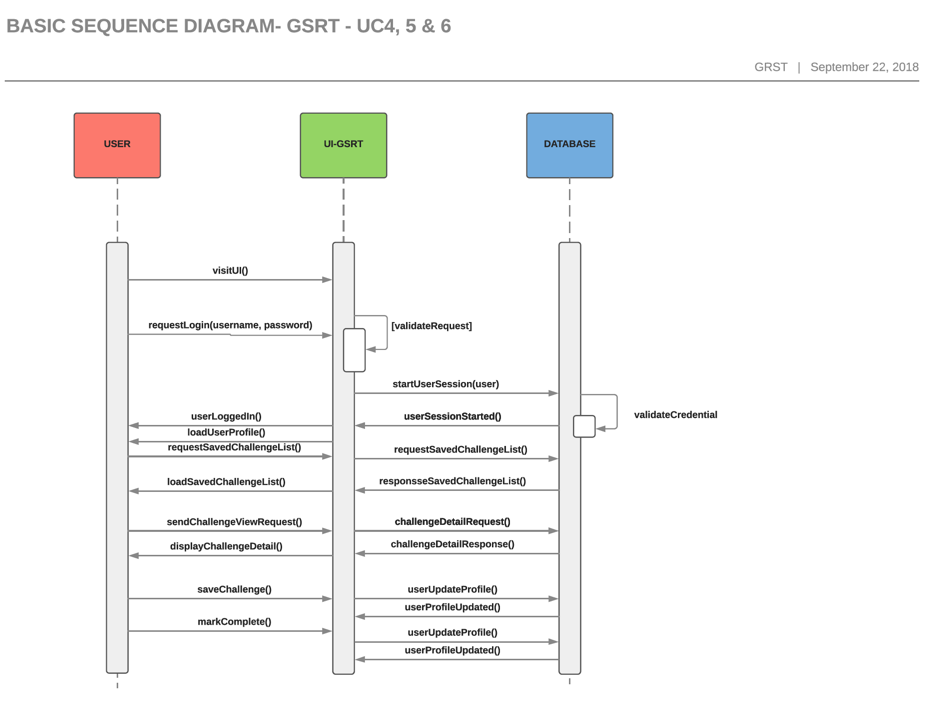
Sequence diagram for Use case 1 & UC2.



Sequence diagram for Use case 3.



Sequence diagram for Use case 4, 5 & 6.



4. User Interface Specification

UI Preliminary design:

We have created 5 html page for preliminary design.

Fig 1 Homepage

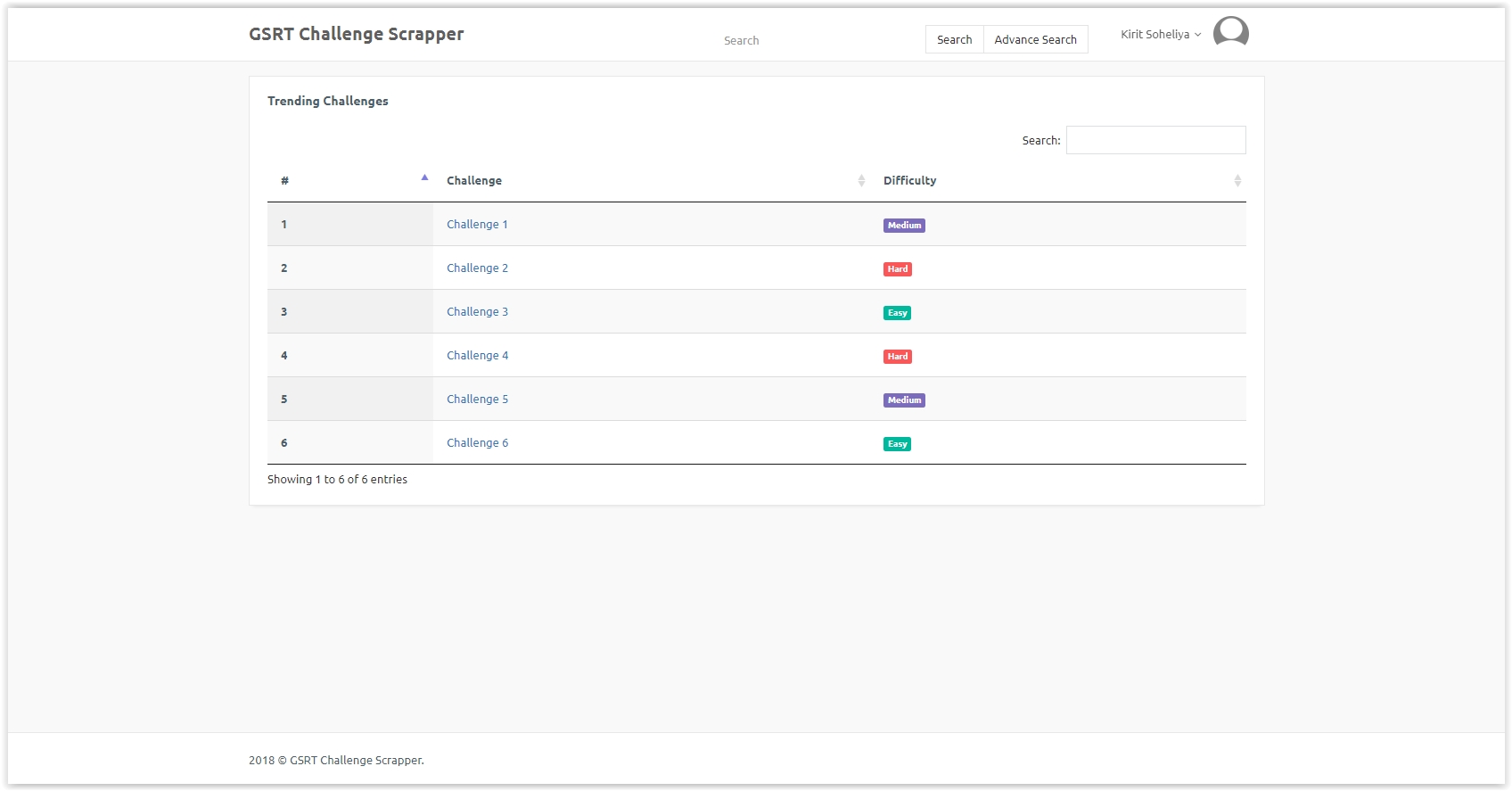


Fig 2 Register page

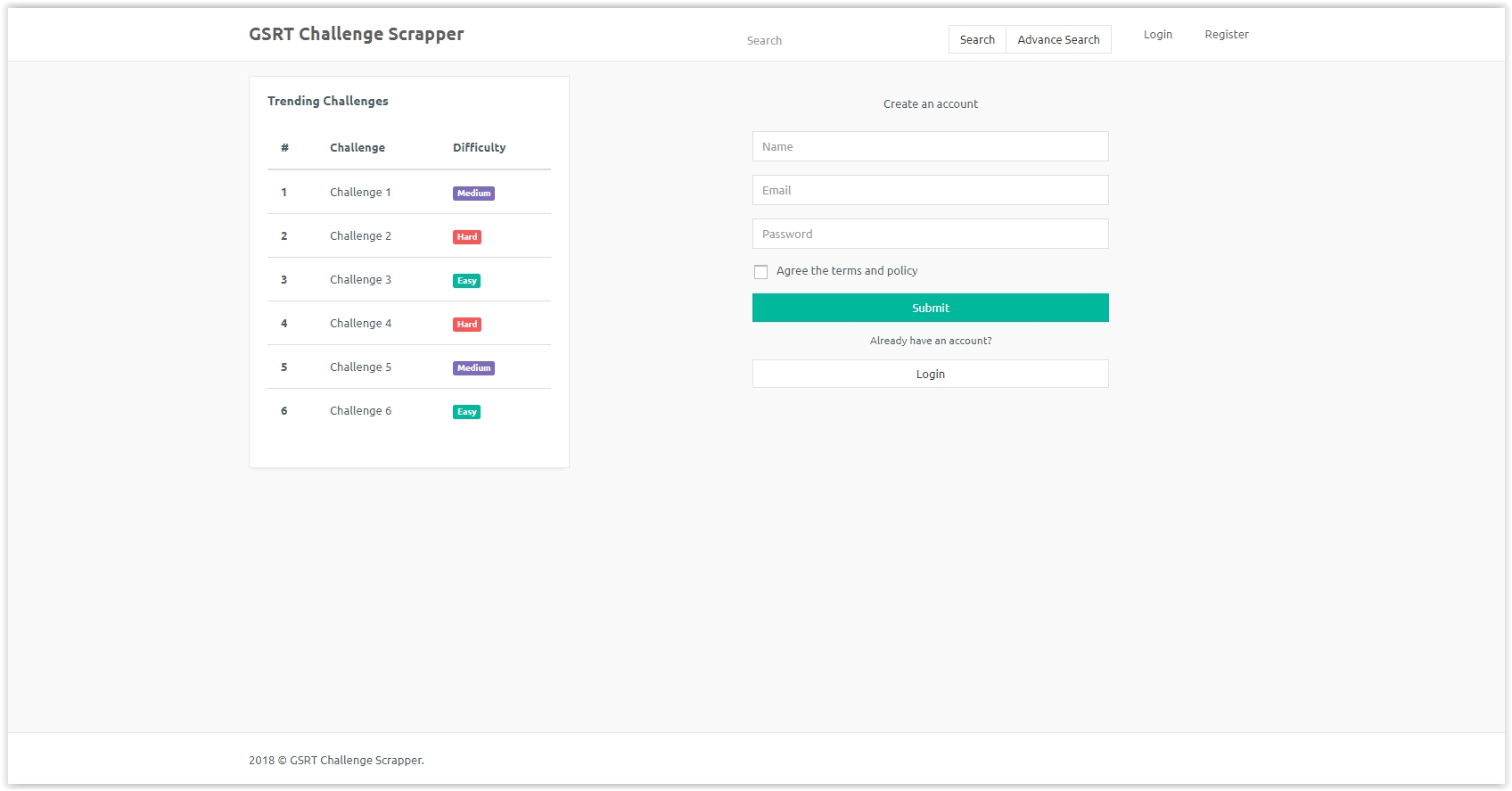


Fig 3 Login page

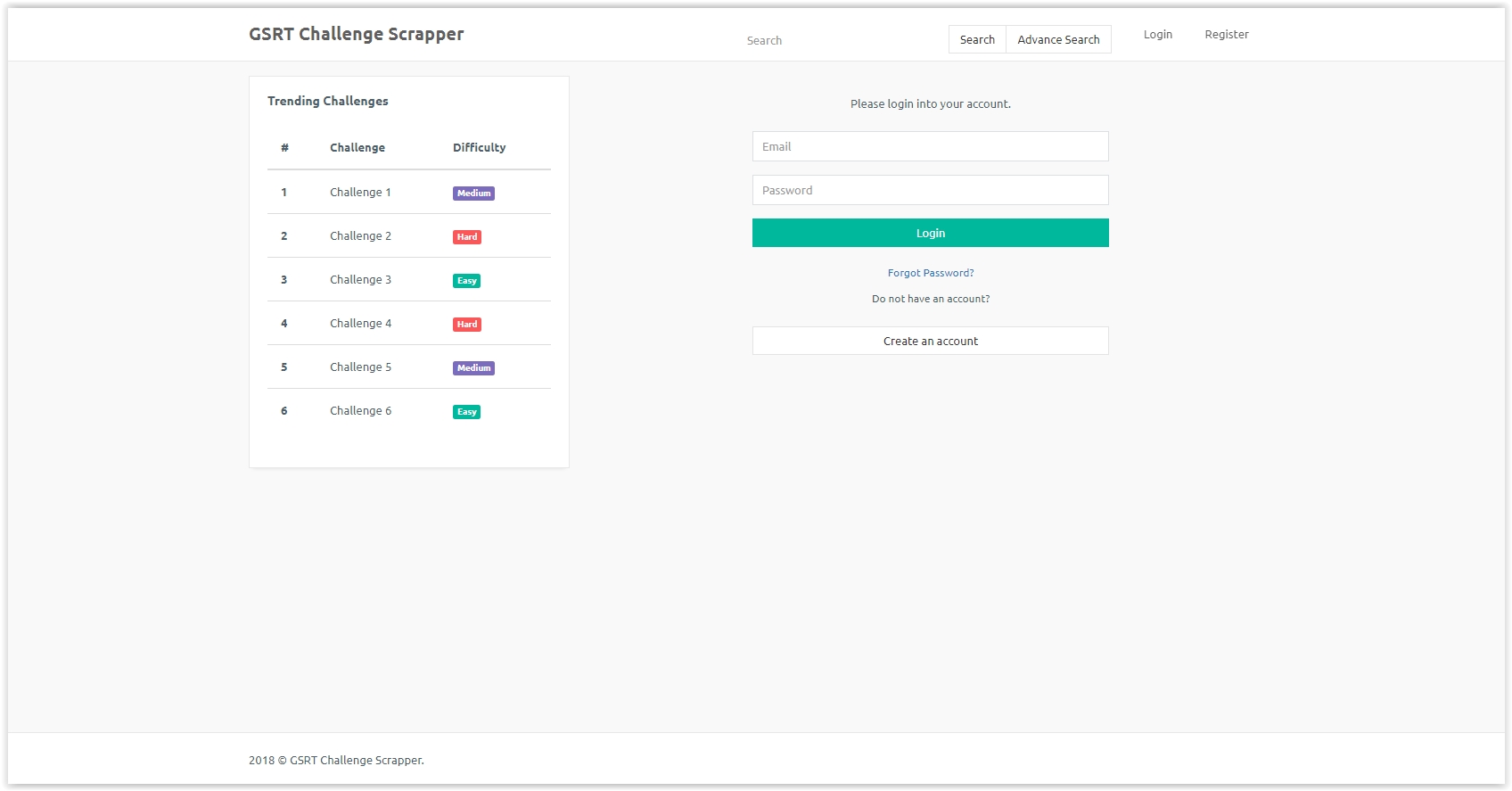
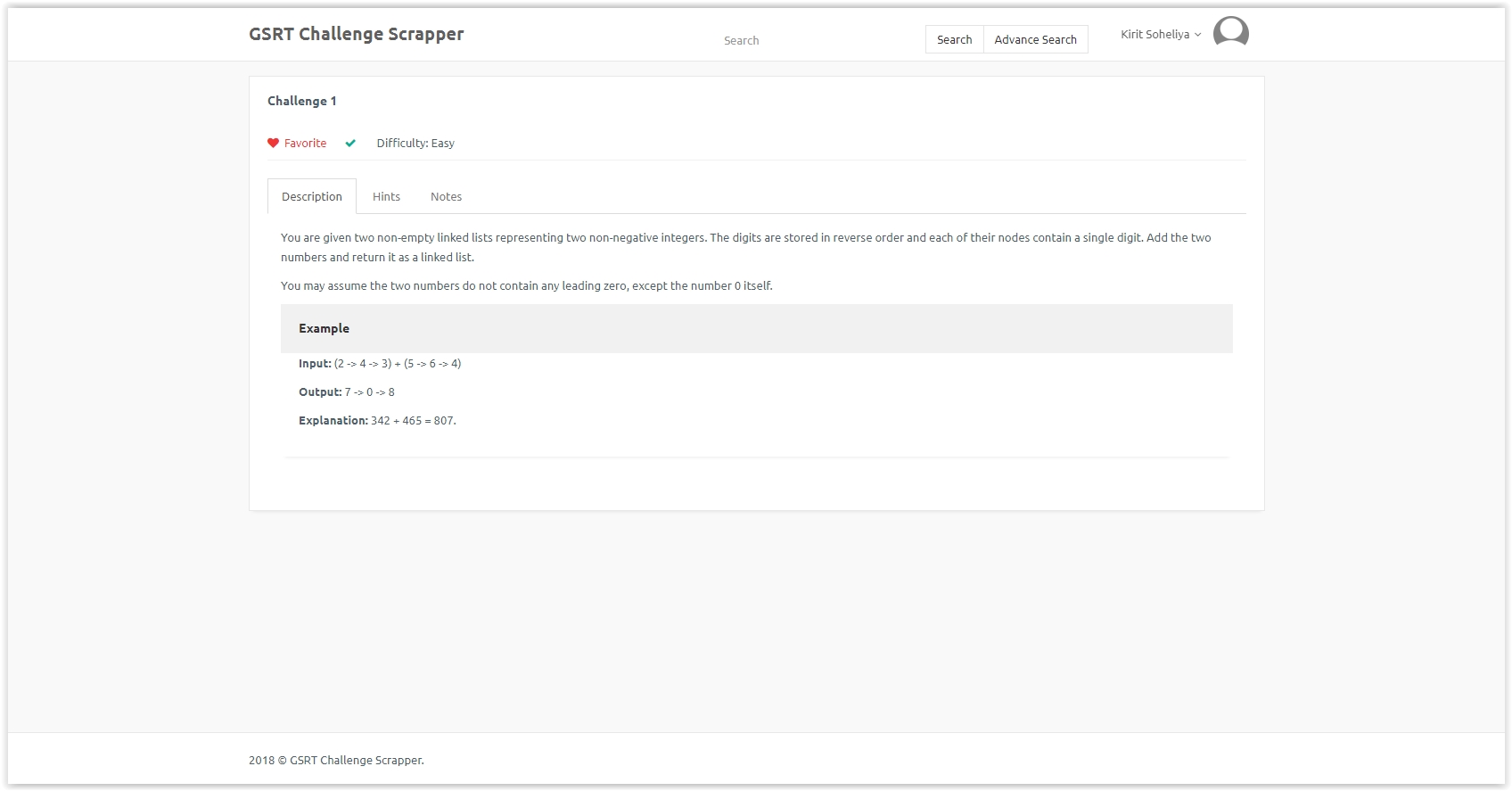


Fig 4 Challenge View page:

**UI User Effort Estimation**

Unregistered User medium effort

|  |  |
| --- | --- |
| Unregistered User medium effort | |
| Click on search box | 1 mouse click |
| Enter search string | Keyboard entry |
| click search icon | 1 mouse click |
| Click desired challenge from list | 1 mouse click |
| View challenge hint (optional) | 1 mouse click |
| Total - UI User Effort | 4 mouse click + 1 Keyboard Entry |

Unregistered User minimum effort

|  |  |
| --- | --- |
| Unregistered User minimum effort | |
| Click the challenge from trending challeng list | 1 mouse click |
| Total - UI User Effort | 1 mouse click |

Registered User medium effort - login

|  |  |
| --- | --- |
| Registered User medium effort - login | |
| Click login icon | 1 mouse click |
| Enter username | Keyboard entry |
| Click password box | 1 mouse click |
| Enter Password | Keyboard entry |
| Click login button | 1 mouse click |
| Total - UI User Effort | 3 mouse click + 2 Keyboard Entry |

Unregistered User maximum efforts - Signup

|  |  |
| --- | --- |
| Unregistered User maximum efforts - Signup | |
| Click Sign Up icon | 1 mouse click |
| Enter username | Keyboard entry |
| Click Full name box | 1 mouse click |
| Enter Full Name | Keyboard entry |
| Click password box | 1 mouse click |
| Enter Password | Keyboard entry |
| Click confirm password box | 1 mouse click |
| Enter Password | Keyboard entry |
| Click Sign Up button | 1 mouse click |
| Total - UI User Effort | 5 mouse click + 4 Keyboard Entry |

Registered User maximum effort

|  |  |
| --- | --- |
| Registered User maximum effort | |
| Click on search box | 1 mouse click |
| Enter search string | Keyboard entry |
| click search icon | 1 mouse click |
| Click desired challenge from list | 1 mouse click |
| View challenge hint (optional) | 1 mouse click |
| Click Mark Favorite Icon | 1 mouse click |
| Click Notes button | 1 mouse click |
| Update Notes (optional) | Keyboard entry |
| Click Completed Icon | 1 mouse click |
| Total - UI User Effort | 7 mouse click + 2 Keyboard Entry |

Unregistered & Registered User maximum effort - unsuccessful search result

|  |  |
| --- | --- |
| Unregistered & Registered User maximum effort - unsuccessful search result | |
| Click on search box | 1 mouse click |
| Enter search string | Keyboard entry |
| Click search icon | 1 mouse click |
| System display empty list with an Error | No click |
| Click on search box | 1 mouse click |
| Enter search string | Keyboard entry |
| Click search icon | 1 mouse click |
| Total - UI User Effort | 4 mouse click + 2 Keyboard Entry |

Domain Analysis

1. Domain Model  
   Show the process of deriving the domain model and then draw the diagram. Provide text description of:
   1. Concept definitions

|  |  |  |
| --- | --- | --- |
| **Responsibility** | **Type** | **Concept** |
| The boundary that allows the user to manipulate the app and see the results | d | interface |
| Converts the results of the search form into a database query | d | searchBuilder |
| The query produced by search builder | k | query |
| The result that the query returns from the database | k | result |
| Manages the connection to the database | d | dbConnection |
| Displays limited information about all results | d | challengeLister |
| Displays detailed information about a single result | d | challengeViewer |
| Organizes the results of the search into a format usable by challengeLister and challengeViewer | d | resultHandler |

The above are the concept definitions used by the front end. The interface allows the user to interact with the app. It allows the user to interact to modify the search, displays the list of challenges, allows the users to select a single challenge, and displays the details of a selected challenge.

The searchBuilder will read the results of the user’s search terms and convert it into a SELECT statement for the database. ResultHandler takes the results from the query and processes them into a usable form for challengeLister and challengeViewer.

ChallengeLister and challengeViewer create pages that are displayed to the user in the interface.

DbConnect will open a connection to the database, run a query, and close the connection when finished. It is the only concept used by both the front-end and back-end.

|  |  |  |
| --- | --- | --- |
| **Responsibility** | **Type** | **Concept** |
| Coordinates and delegates information between the various back end concepts | d | controller |
| Parses external website to find locations of new challenges | d | challengeFinder |
| Parses locations found by challengeFinder to get details of new challenges | d | detailFinder |
| List of locations created by challengeFinder | k | challengeList |
| Details of challenges created by detailFinder | k | challengeDetails |
| Manages the connection to the database | d | dbConnection |
| Converts challengeList into a form that can be inserted into database | d | listInserter |
| Converts challengeDetails into a form that can be inserted into database | d | detailInserter |

The above are concept definitions for the back-end.

Currently only the front-end and back-end concepts are shown. There are also numerous concepts derived from the register, login, mark, and save use cases. We are currently looking into options for third-party login solutions. Finding a pre-existing solution will likely be more secure and more efficient than what our group can come up with on a limited schedule and no budget. From an academic standpoint it is beneficial to be able to work with pre-existing code. If no suitable solutions can be found, some or all of these use cases will be revisited.

* 1. Association definitions

Front-End Associations

|  |  |  |
| --- | --- | --- |
| **Concept Pair** | **Description** | **Association Name** |
| Interface ↔ searchBuilder | The interface conveys the user’s search request to the searchBuilder. | Conveys requests |
| searchBuilder ↔ dbConnection | The searchBuilder conveys the query to the dbConnection | Conveys requests |
| dbConnection ↔ resultHandler | The dbConnection provides the results of the query to the result handler | Provides Data |
| resultHandler ↔ challengeLister | The resultHandler provides the data to the challengeLister that is necessary to prepare the list of challenges for the interface | Provides Data |
| resultHandler ↔ challengeViewer | The resultHandler provides the data to the challengeViewer that is necessary to prepare the list of challenes for the interface | Provides Data |
| ChallengeLister ↔ interface | challengeLister prepares the interface page with a list of challenges | Prepares |
| challengeViewer↔ interface | ChallengeViewer prepares the interface page with details of a single challenge | Prepares |

The front end associations are related to the user’s interaction with the search, database, and challenges.

|  |  |  |
| --- | --- | --- |
| **Concept Pair** | **Description** | **Association Name** |
| Controller ↔ challengeFinder | The controller instructs the challengeFinder to search an external website for challenges. | Conveys request |
| Controller ↔ detailFinder | The controller instructs the detailFinder to search an external website for details of one specific challenge. | Conveys request |
| ChallengeFinder ↔ listInserter | ChallengeFinder passes the challengeList to listInserter to be prepared as a query. | Prepares |
| ListInserter ↔ dbConnection | Provides an insert query generated by listInserter to dbConnection | Provides Insert Query |
| DetailFinder ↔ detailInserter | ChallengeFinder passes the challengeList to listInserter to be prepared as a query. | Prepares |
| DetailInserter ↔ dbConnection | Provides an insert query generated by detailInserter to dbConnection | Provides Insert Query |

The back-end associations are related to an administrator or chron interactions with other websites and the database.

* 1. Attribute definitions

|  |  |  |
| --- | --- | --- |
| Concept | Attributes | Description |
| searchBuilder | searchParameters | Site, keywords, difficulty, etc. |
| dbConnection | DbInfo, query | Dbinfo is needed to establish connection to the database. The query is the select query to be run. |
| challengeLister | ChallengeList | The list of results that match the search criteria |
| challengeViewer | ChallengeDetails | The details of the selected challenge. |
| challengeFinder | URL, linkKey | The URL of an external challenge website and what to look for to find links to a challenge |
| detailFinder | URL, challengeKey | The URL where a given challenge is located and how to find the relevant info |

* 1. Traceability matrix — show how your use cases map to your domain concepts.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | UC1 | UC2 | UC7 | UC8 |
| interface | X | X |  |  |
| searchBuilder | X |  |  |  |
| query | X |  |  |  |
| result | X | X |  |  |
| dbConnection | X |  | X | X |
| challengeLister | X |  |  |  |
| challengeViewer |  | X |  |  |
| resultHandler | X | X |  |  |
| controller |  |  | X | X |
| challengeFinder |  |  | X | X |
| detailFinder |  |  | X | X |
| challengeList |  |  | X | X |
| challengeDetails |  |  | X | X |
| listInserterdetailInserter |  |  | X | X |

The bulk of our functionality is made up of the 4 use cases listed above. Use Cases 3 through 6 require log-in functionality. As discussed above, our group is looking to see if any preexisting solutions for 3 and 4 exist. Once those are created, domain models for 5 and 6 can be created, but since those are low priority they have been skipped for now.

While building the domain model, our group realized that 7 and 8 are nearly identical except for the initializing actor. UC-7 is initialized by a chron daemon, and UC-8 is initialized by an administrator. These use cases should be combined into one in the future.

Additionally, in order to get the text of a challenge, our program must first get a link to that text usually from a menu page. Currently, this is all done in UC-7 (and UC-8), but me may want to consider getting the links as one use case and getting the details as another use case..

1. System Operation Contracts  
   Should be provided only for the operations of the fully-dressed use cases elaborated in Section 3.c), for their system operations identified in Section 3.d).
2. Mathematical Model

No mathematical models are used for this application.

6. Project Management

Above is a Gantt chart showing all required documentation that needs to be completed for Report 1. The in-depth description of events can be obtained from Dr. Marsic’s website and will bot be duplicated here. Longer term documentation goals are described in the syllabus and will be documented in depth until when are closer. The hierarchy of prerequisites for actually creating the project is still be examined. When that is complete, at the end of Report 1, an additional Gantt chart for the project itself will be created.

Next week begins Report 2. Part 1 only contains interaction diagrams and is based on the information created in Report 1. Additionally, our group will discover a solution for log-ins, begin creating a non-functional interface, and begin locating the search keys to find where links and challenges can be found on various challenge websites.

Breakdown of Responsibilities:

Report 1 Part 1:

Matthew Griffin: Project Management Section, Gantt Chart

Ashish Rastogi: Functional and Non-functional Requirements

Kiritbhai Soheliya: On-Screen Appearance Requirements

Martin Tiernan: CSR Problem Statement and Glossary

Report 1 Part 2/3:

* Matthew Griffin: Domain Model (part 3), Traceability Matrix (part 3), Project Management (parts 2 and 3).
* Ashish Rastogi: Use Case Diagram, Sequence Diagram .
* Kiritbhai Soheliya: Fully Dressed Use Cases, UI Preliminary Design, UI User Effort Estimation, System Operation Contracts (part 3).
* Martin Tiernan: Stakeholders, Actors & Goals, Use Case Casual Descriptions, and Traceability Matrix.