The A-Team

UMBC Textbook Marketplace

Tuc Nguyen - Amra Mo - Jun Kwon - Hao Wu

Amir Beshir - Sophia Shin

IS 436 - Structured Systems Analysis and Design

Deliverable III - Process Modeling (D3)

April 18, 2019

**Table of Contents**

Requirements Definition…………………………………...……………………..….3

Introduction (Business Requirements)

User Requirements

Functional Requirements

Nonfunctional Requirements

Cultural and Political

System Description/Requirements

Data Flow Diagram………………………………………………………….……….5

Use Case Documentation…………………………………………………………….7

Use Case Scenarios…….…………………………………………...…………..........9

PWP & Kanban board……………………………………………………………….17

**Requirements Definition**

**Introduction (Business Requirements):**

**Purpose:** Our mission is to create an application for UMBC that would serve as a marketplace for students to sell or purchase their textbooks amongst each other as a listing. Each listing would have the student’s contact information, the condition of the book, subject and class the book belongs in, catalog #, book title, author, list the price seller want to sell it for. We want to provide a safe service where UMBC-only students and alumni can reach out to one another on campus to get rid of their textbooks easier, as well as make money without going through the trouble of selling / purchasing their books by having them shipped off online.

**User Requirements:**

* The system will keep track of all data required by the user to maintain a database by generating user listings.
* Should be understandable and minimal for non-technical people to use.
* Written for Customers

**Functional Requirements:**

Process-Oriented:

1. User must be able to sell and buy.
2. User must be able to create new accounts on our page.
3. User must be able to browse textbook listings.
4. User must be able to add, remove, and change any information.
5. Listing automatically becomes labeled as “pending” when buyer and seller agree to meet.
6. User may select new or used items

Information-Oriented:

1. Add new users
   * New users must register their username, password, and UMBC email to the database.
2. Update user info
   * Any changes made to any users username, password, email, address, and listings will be updated in our database.
3. Add new or user books
   * New entries must like book title, description, availability, price, seller, quantity, quality, edition, price, and author to the database.
4. Update new book/items
   * Any changes to the quantity, availability, price, condition, and seller must be updated in the database when changed.
5. Search
   * User can search for any books.
6. Request books/items
   * User can request for books.

**Non Functional Requirements:**

**Operational:** To start building the application to run on most web browsers, we need to choose a web framework that is tested to run successfully on Chrome, Firefox, Safari, Edge, and other important browsers. Use popular tools such as Jquery and other frameworks that are proven to work.

**Performance:** To maximize the performance of the application we need to choose a web server that is fast and can handle many user connections. Nginx and the Apache load balancers are good choices to use to distribute user traffic to different application servers. The system should be available for user 24 hours per day, 365 days per year. It should also be able to support simultaneous users at all other times.

**Security:** System administrators will be allowed permissions to maintain the system. Developers should be authorized to access the system source code to provide new features and fix bugs associated with the system. Student users will only have permissions to view and edit their own book listings and system account information. Interpersonal politics affects the performance of the entire project. Buyer and seller protection and information security should be a priority for any ecommerce site.

**Cultural and political:** Our primary tools will be from academic partners in the industry such as Dell, Cisco, IBM, etc. We could choose to use cloud services such as Amazon Web Services, Microsoft Azure, or Google Compute Cloud instead of buying our own hardware for the servers. Find partnerships in order to reduce costs in the future. In order to reduce costs now, make minimal mistakes in the programming, have effective project management practices, and use minimum viable product releases. Student’s personal information is protected in compliance with the Data Protection Act

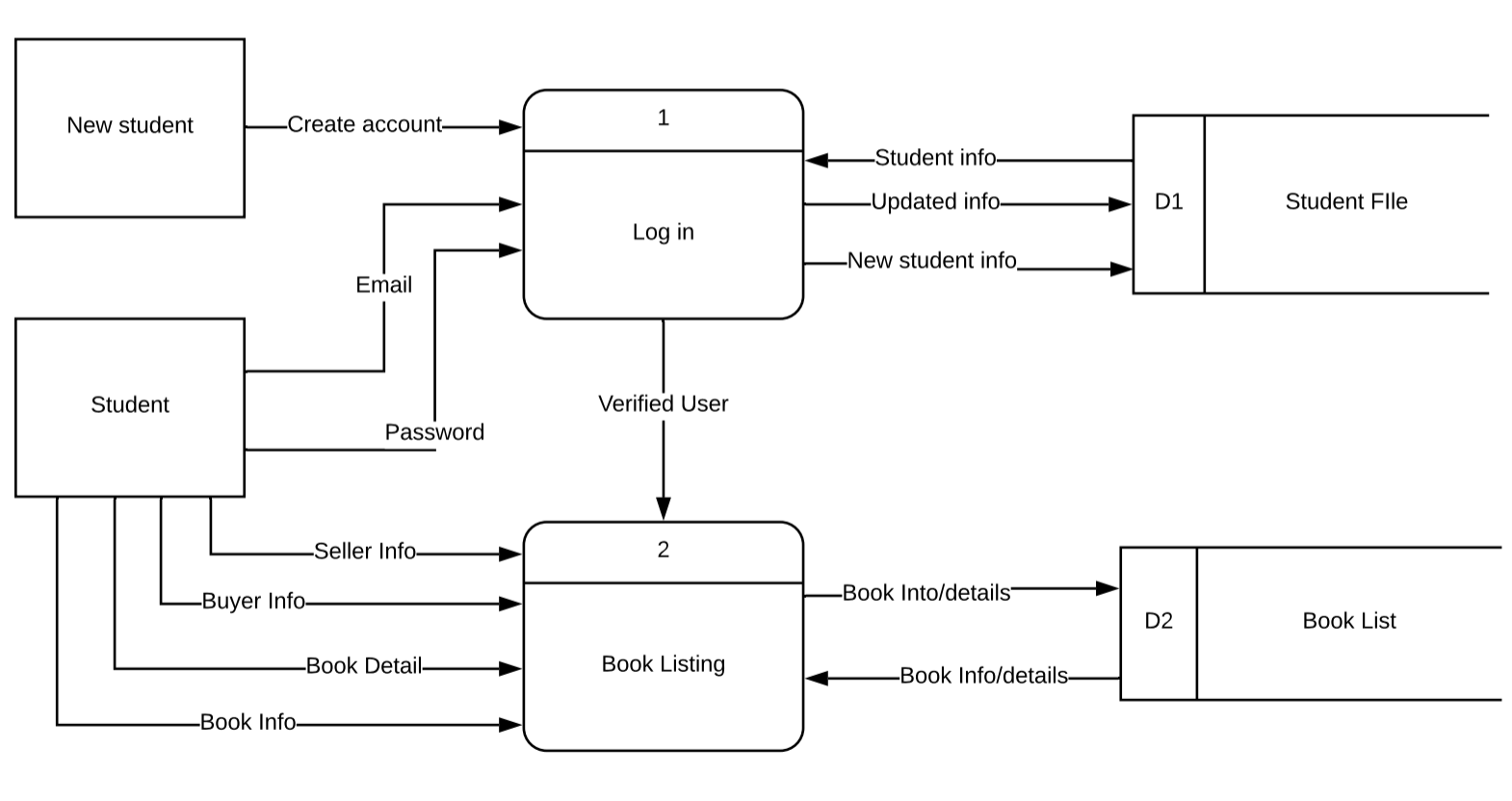
**System Description/Requirements:**

Subsystems could include a database system, a load balancing system, a network logging and metrics system, and a dedicated security system. Running the system on a Linux based platform such as Ubuntu LTS or CentOS would allow us to have a solid and free operating system that has security updates and supports our enterprise requirements.

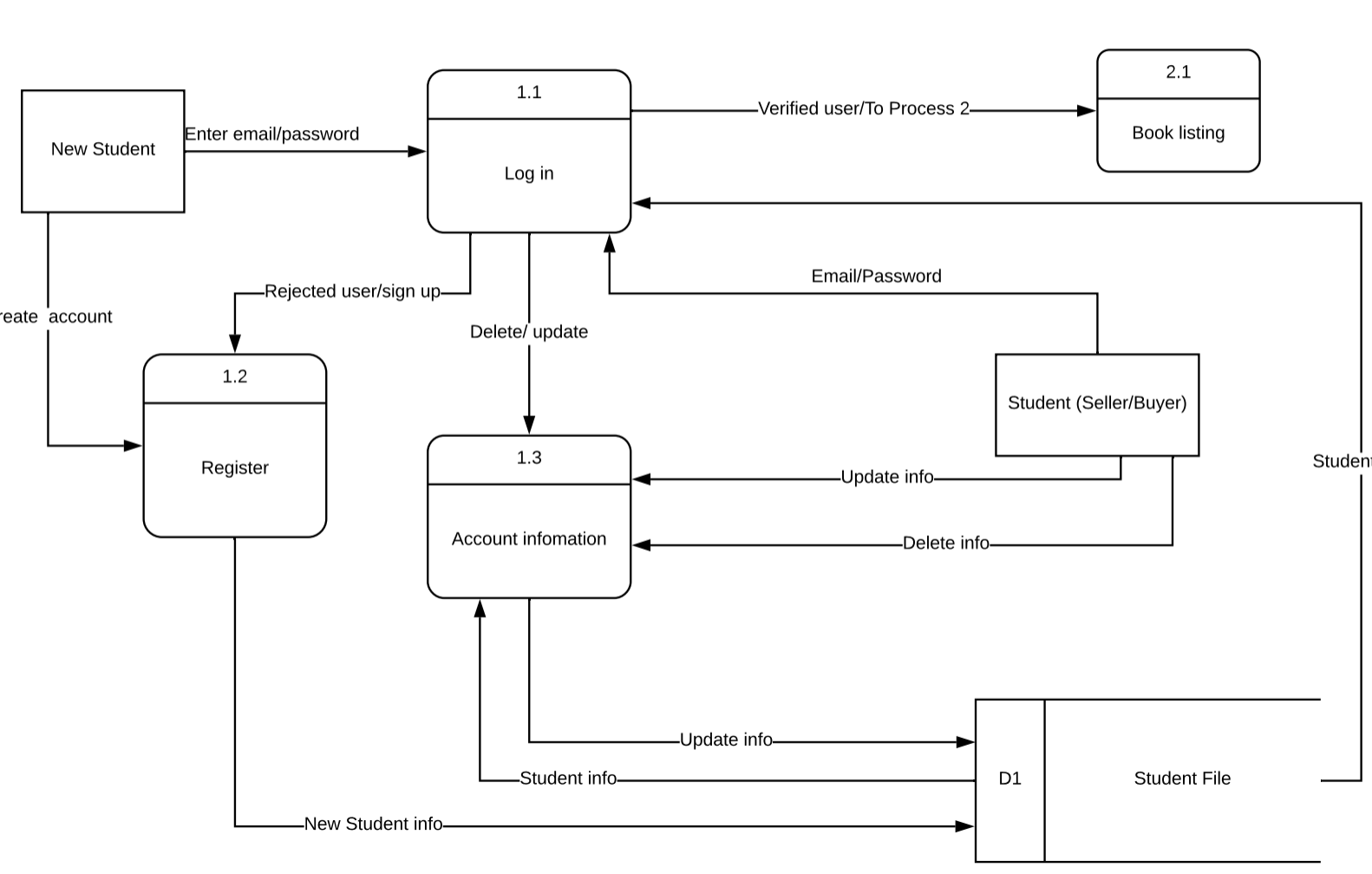
We need to build the system around the requirements. Use encryption, user rights restrictions, and connection blocking, and separation of duties for security. Use load balancers and commonly used frameworks in order to maximize usability across phone OSs and browsers. Cloud computing and open source software can reduce cost across the application.

**Data Flow Diagram:**

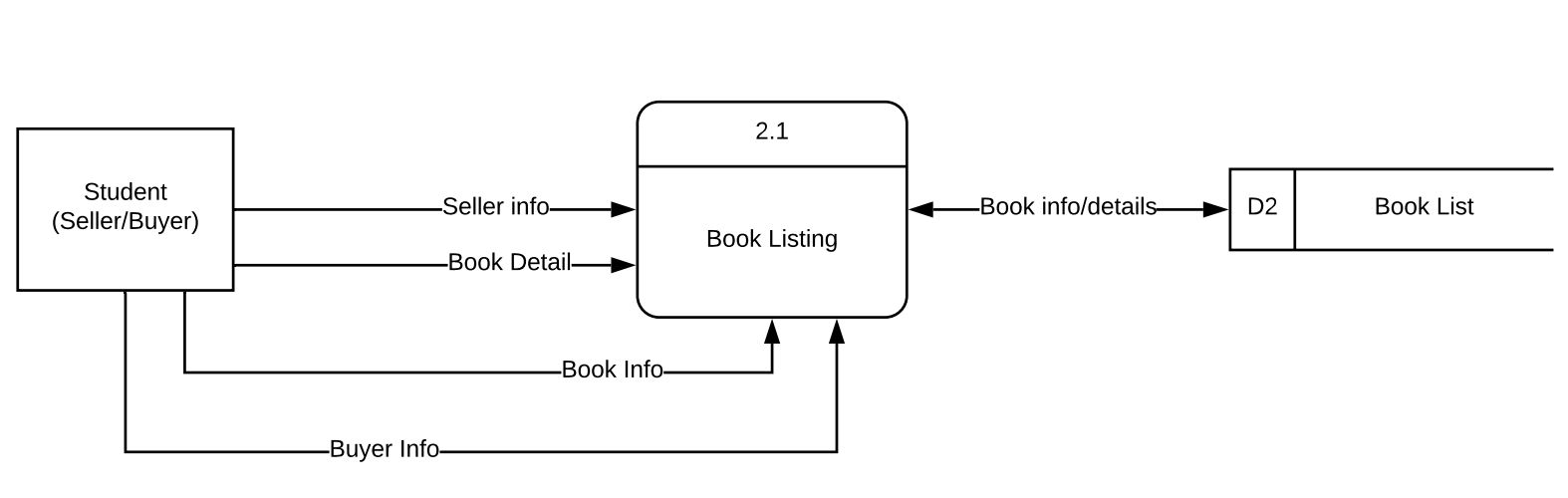
**Level 0 Diagram**

****

**Level 1 Process 1**

****

**Level 1 Process 2**

****

**Use Case Documentation**

**UMBC Textbook Marketplace System**

An application for UMBC that would serve as a marketplace for verified students to sell or purchase their textbooks amongst each other.

**Risk Factors**

· Before someone can register for the app, their email used to registrar must contain @umbc.edu domain name to ensure that they are a UMBC student or alumni.

· Must also include a picture of the student

· Student’s information could be compromised by a hacker.

· Feedback from beta testers to improve the product.

· Address security issues because student’s UMBC information is linked to their account with the app.

· The system must be operational prior in advance of the academic term starting.

· People designing the software may be inexperienced

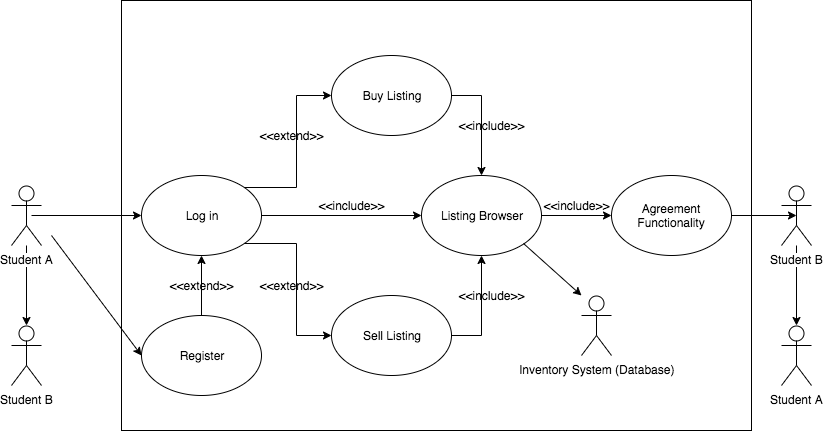
· Preventing system failure

· Can we be successful if we don’t support a web interface / mobile application?

· Easy for non-technical people to use

· How to handle simultaneous users

· How to handle database crashing.



**Architecture Diagram**

In our current stage, we will be required to have certain Subsystems. The following below would be potentially included.

**Subsystem Descriptions**

· System Access Subsystem

o This subsystem handles log in and log out

o Handles student profile

· Display Main Screen

o Handles a window that shows option for the user to pick amongst

o Buy Listing

o Sell Listing

o Listing Browser

· Book Info Subsystem

o Provides information about books on record

o Inventory System

· Manage Student Info Subsystem

o Selling Book

o Buying Book

· Database Subsystem

o Contains the data we need to store for application

o Retrieve, update, delete functions for data stored

o Information about books (Title, Catalog #, Author, Year, Edition, etc)

**Use Cases Scenarios**

**LOG IN**

This Use Case describes the process which students log in to the system.

**Actors**

· Student A

· Student B

**Priority**

This part of the Use Case is essential to the rest. Student must be able to successfully log in as a returning user. If not, the user will not be able to list, sell, or browse the book catalog.

**Pre-conditions**

**·**  If login information is valid and other credentials are correct, then user can enter the system.

· If login information is not valid and other credentials are incorrect, then user can not enter the system.

**“Used” Use Cases**

None

**Flow of Events**

**·** The Use Case begins with the user beginning the application

· The system displays a login screen

· Student enters username / UMBC email address, password

· System validates information

· Sets access permissions

· Displays main screen that will show a variety of options for the user

· User selects function

· If user:

- Selects Buy Listing

Use Buy Listing option

- If wanting to use Buy Listing option:

Use Agreement Functionality option

o Else,

o Selects Sell Listing

Use Sell Listing option

· If wanting to use Sell Listing option:

Use Agreement Functionality option

o Else,

o Selects Listing Browser

Use Listing Browser option

o End if

o Student will select a function

o End loop

o Use Case ends

**User interface**

**·** Prompts student to give username and/or UMBC e-mail, password

· Student gives user name, password

· Checks to see if user name, password is valid

**·**  Shows main screen that shows if student wants to buy, sell, or list

· Selects option

· Can use agreement functionality to if they want to buy or sell from Student to Student

**Scenarios**

**·**  Bad user name, password, or email

· User not in system

· No selection of a valid function

**Subordinate Use Cases**

· Display Main Screen

**REGISTER**

This Use Case allows students to registrar to create an account.

**Actors**

· Student A

· Student B

**Priority**

New students are registering into the application using their UMBC.EDU email account before login to use the application.

**Pre-conditions**

· Students must register with UMBC email.

**“Used” Use Cases**

None

**Flow of Events**

· Go to the Login Screen.

· Select register.

· Register using UMBC email.

· After registered, the application redirect back to the login screen.

**User Interface**

· Prompt up request for UMBC email and password

**Scenarios**

· Invalid email if use all other email except for UMBC.EDU.

**LISTING BROWSER**

This Use Case creates an entry for a book the student is selling, buying to view and use search feature

**Actors**

· Student A

· Student B

**Priority**

This part of the Use Case is not really essential unless student is wanting to use browse feature.

**Pre-conditions**

· If login information is valid and other credentials are correct, then user can enter the system.

· If login information is not valid and other credentials are incorrect, then user can not enter the system.

· Student wants to view Listing Browser

**“Used” Use Cases**

Log in

**Flow of Events**

· Displays main screen that will show a variety of options for the user

· User picks Listing Browser

· Can use search filter to view for options: buying, selling, books on system, etc.

**User Interface**

· Ability to filter by categories such as buying or selling, price, author, and title.

· Search function, with advanced search, able to search whether a listing is buying or selling, price ranges, catalog numbers, titles, authors, and editions.

**Scenarios**

· Book is not available in the system

**SELL LISTING**

This Use Case creates an entry for a book the student is wanting to sell

**Actors**

· Student A

· Student B

**Priority**

This part of the Use Case is not really essential unless student is wanting to create a sell listing.

**Pre-conditions**

· If login information valid and other credentials are correct, then user can enter the system.

· If login information is not valid and other credentials are incorrect, then user can not enter the system.

· Student wants to create sell listing

**“Used” Use Cases**

Log in

**Flow of Events**

· The Use Case begins with the user initiates if they want to create a sell listing

· Input information into fields

· Option to edit and delete

**User Interface**

· User will input valid information into fields such as contact information, book information, and pricing.

· Ability to edit and delete

**Scenarios**

· User wants to sell

· Invalid field information

**BUY LISTING**

This Use Case creates an entry for a book the student is wanting to buy.

**Actors**

· Student A

· Student B

**Priority**

This part of the Use Case is not really essential unless student is wanting to create a buy listing.

**Pre-conditions**

· If login information valid and other credentials are correct, then user can enter the system.

· If login information is not valid and other credentials are incorrect, then user can not enter the system.

· Student is wanting to create buy listing

**“Used” Use Cases**

Log in

**Flow of Events**

· The Use Case begins with the user initiates if they want to create a sell listing

· Input information into fields

· Option to edit and delete

**User Interface**

· User will input valid information into fields such as contact information, book information, and pricing.

· Ability to edit and delete

**Scenarios**

· User wants to sell

· Invalid field information

**AGREEMENT FUNCTIONALITY**

This Use Case creates an entry if Student A wants to contact Student B to Buy/Sell from listing.

**Actors**

· Student A

· Student B

**Priority**

This part of the Use Case is slight unless student wants to buy / sell, and will only allow the student to do so if all credentials and listing information is correct

**Pre-conditions**

· If login information valid and other credentials are correct, then user can enter the system.

· If login information is not valid and/or other credentials are incorrect, then user cannot enter the system.

· Student wants to Buy / Sell

**“Used” Use Cases**

Log in

Buy Listing

Sell Listing

**Flow of Events**

· User selects function

· If user:

o Selects Listing Browser

Use Listing Browser option

· When seller/buyer is interested:

Send contact information to listing’s creator. Creator can then decide to message the interested party or not

o End if

o Student will select a function

* Use Case ends

**User Interface**

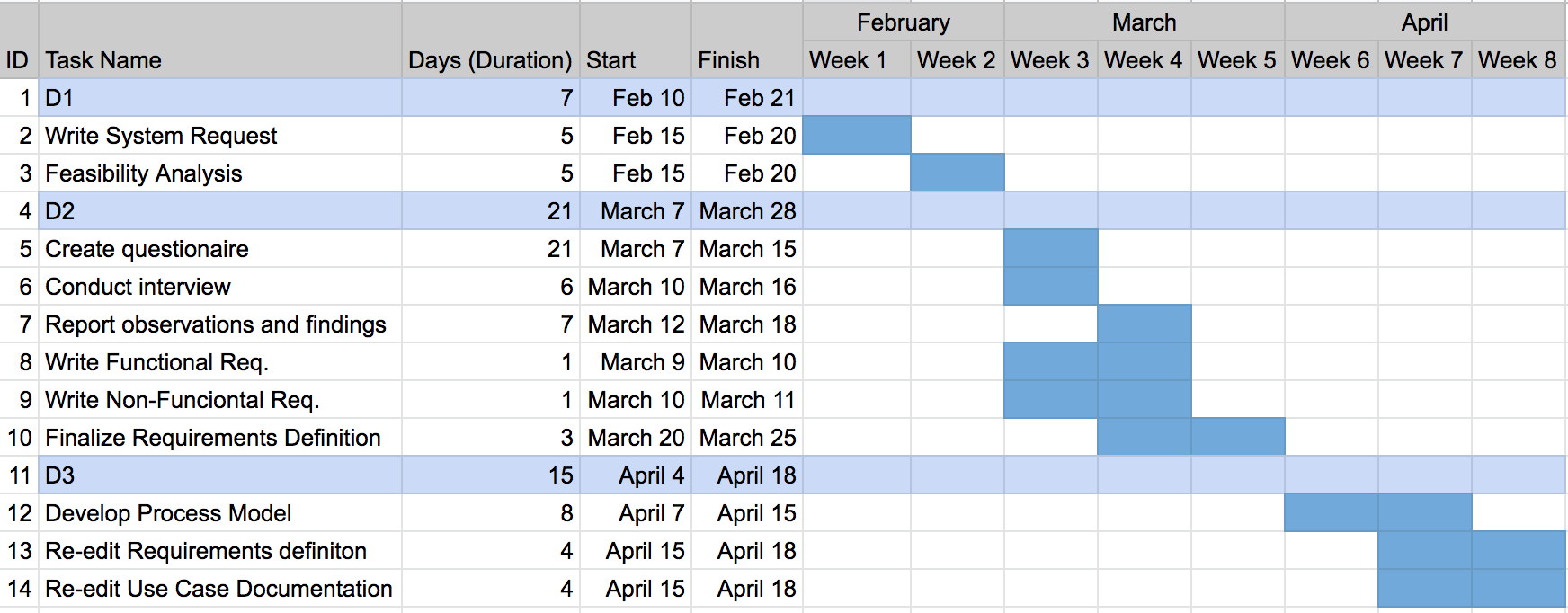
-User selects function and can input information into field for contact

**Scenarios**

· Information not valid

· Fields not valid

**PWP Plan**



**Kanban Board**

