Department of Computer Science & Engineering
The University of Texas at Arlington

Detailed Design Specification
CSE 4317: Senior Design II
Spring 2022

Team GI
Senior Design Lab Management System

Jonathan Banda
Tochy Egeonu
David Rademacher
Ian McKenzie
Jake

REVISION HISTORY

Revision	Date	Author(s)	Description
0.1	2.07.2022	JB, T, D, I, J	document creation
0.1	2.9.2022	JB	Layer Database
0.1	2.10.2022	Tochy Egeonu	System Overview
3	4.21.2021	Tochy Egeonu	Revisions

Contents

1 Introduction **5**

2 System Overview **5**

3 Database Layer Subsystems **6**
- 3.1 Layer Hardware 6
- 3.2 Layer Operating System 6
- 3.3 Layer Software Dependencies 6
- 3.4 Subsystem Permissions 6
- 3.5 Subsystem User 7
- 3.6 Subsystem Account 7
- 3.7 Subsystem Classes 8
- 3.8 Subsystem Item 9

4 Frontend Layer Subsystems **10**
- 4.1 Layer Hardware 10
- 4.2 Layer Operating System 10
- 4.3 Layer Software Dependencies 10
- 4.4 Admin 10
- 4.5 Auth 11
- 4.6 Dashboard 11
- 4.7 Loading 11
- 4.8 NavComp 11
- 4.9 Profile 11
- 4.10 ViewItems 11

5 API Layer Subsystems **12**
- 5.1 Layer Hardware 12
- 5.2 Layer Operating System 12
- 5.3 Layer Software Dependencies 12
- 5.4 Subsystem Authentication 12
- 5.5 Subsystem Admin 13
- 5.6 Subsystem Student 14

6 Appendix A **15**

List of Figures

1 System architecture 5
2 frontend subsystems 10
3 Example subsystem description diagram 12

List of Tables

1 Introduction

We are building a centralized web app to store and retrieve information regarding students and items in the Senior design lab. Once logged in, the web app will identify whether or not the user is a admin or a student. All users will be allowed to see a overview of available items, but only admins will be allowed to modify any information regarding the items. Admins also have the ability to run reports regarding what items are overdue. Student accounts will be allowed to checkout and return items.

2 System Overview

The web app system will have 3 layers including, a database layer, a front-end layer, a API layer. The database layer plays the role of storing the information in models and maintaining integrity of the data. The front-end layer will be all the different pages included in the web app. The API layer will be the different function calls that happen between the front-end and database layer.

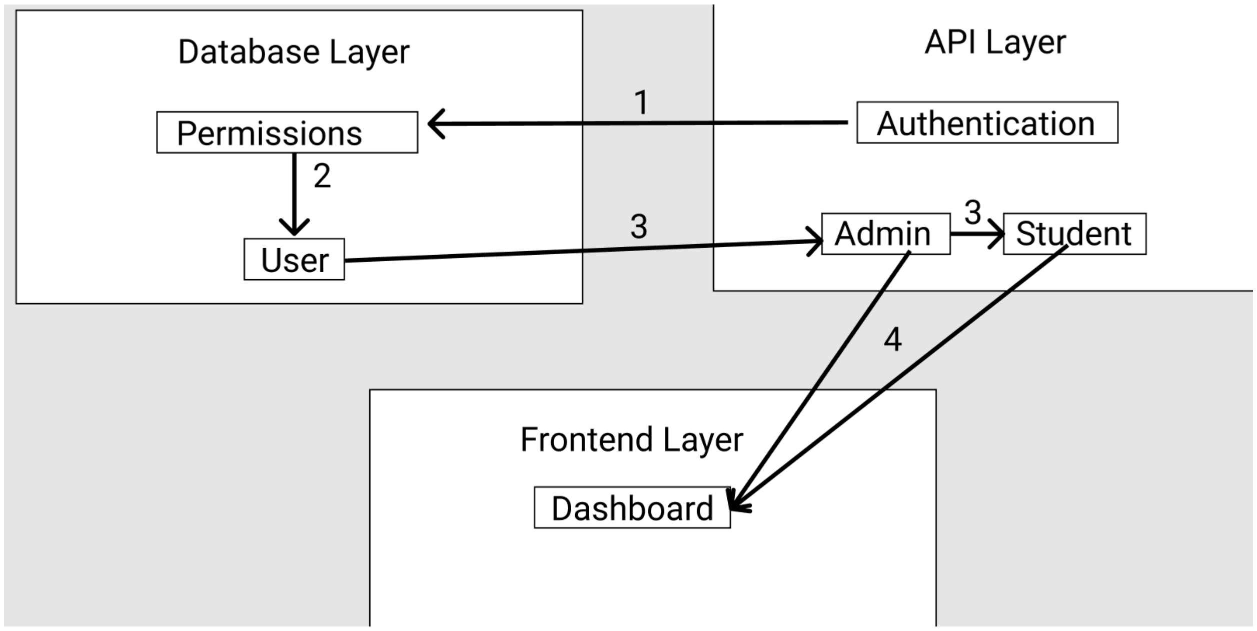

Figure 1: System architecture

3 Database Layer Subsystems

The database layer is the foundation of the senior design inventory system. We have choose to deploy the database online so that the data is not localized. Currently the web application's database connection is running on MYSQL 8.0.23 which is being hosted locally. A high level view of this layer consists of many individuals models that represent tables in the database. Each of these models serve their own purpose and have a relation to another.

3.1 Layer Hardware

No layer hardware.

3.2 Layer Operating System

A description of any operating systems required by the layer. The database layer does not require any specific operating system. That being said it should be implied that the web application be ran on an operating system capable of connecting to the internet view a web browser in order for the web application to make GET, POST, PUT, and DELETE requests to the online database.

3.3 Layer Software Dependencies

Software dependencies for the database layer include libraries such as axios that provide a way to make query requests to and from the database. Frameworks such as the REST API which allows the database layer to be useful to the users by putting the data onto the screens of the users. The Database itself will be running on MYSQL 8.0.23. In general Django versions 3.2 or higher should satisfy and dependencies regarding the integration of the database in the settings.py file.

3.4 Subsystem Permissions

The subsystem permissions is a built in model that is created when when the first migration is sent. This model is in regards to differentiating between users and admin of the Django web app. At a high level view it is used manage the backend server side of Django.

3.4.1 Subsystem Hardware

No supporting hardware.

3.4.2 Subsystem Operating System

No specific operating system

3.4.3 Subsystem Software Dependencies

No software Dependencies.

3.4.4 Subsystem Programming Languages

This subsystem is written in python 3 but only as an installed app call during initial set up of the Django environment. It is just required to be called upon with python in the settings.py file.

3.4.5 Subsystem Data Structures

No data structure, Django handles this part.

3.4.6 Subsystem Data Processing

No data processing, Django handles this part.

3.5 Subsystem User

This subsystem stores each and every user that is registered through the registration component. This is a built in Django model that handles the users cases. This model is referenced by permissions as well. At a high level the model is used to authenticate between the regular users and super users. This difference will render different views dependent on the jwt authentication once logged into the web application.

3.5.1 Subsystem Hardware

No supporting hardware.

3.5.2 Subsystem Operating System

No specific operating system

3.5.3 Subsystem Software Dependencies

In order to access this model and its data, there needs to be an import specified in the models.py file -> from django.contrib.auth.models import User

3.5.4 Subsystem Programming Languages

No programming languages

3.5.5 Subsystem Data Structures

No data structure, Django handles this part.

3.5.6 Subsystem Data Processing

No data processing, Django handles this part.

3.6 Subsystem Account

The Account subsystem is responsible with maintain the relation between a user and his/her dealing such as class, instructor, and items checked out.

3.6.1 Subsystem Hardware

No supporting hardware.

3.6.2 Subsystem Operating System

No specific operating system

3.6.3 Subsystem Software Dependencies

This class/model is dependant on two components in django's backend. The first component is that the declarations are written in python. The second component must use a special library to create the table fields in the class. The table fields must be written using a django->models format. This can be done by importing "models" from "django.db" allowing the creation of the class.

3.6.4 Subsystem Programming Languages

Given that the model is created in the models.py file the programming language "Python" must be installed to run migrations and makemigrations.

3.6.5 Subsystem Data Structures

The general data structure of the Account class is broken down bellow.

- user : One To One Field, Primary Key Value

- instructor : Foreign Key, referenc = instructor
- myClass : Integer Field
- section : Integer Field
- team : Char Field
- items : Many To Many Field, reference = users
- itemCount : JSONField, object

3.6.6 Subsystem Data Processing

No data processing, Django handles this part.

3.7 Subsystem Classes

The classes model/class represents the students current class placement. This is needed to differentiate between instructors and certain sections. This will facilitate any communication concerns between student and instructor.

3.7.1 Subsystem Hardware

No supporting hardware.

3.7.2 Subsystem Operating System

No specific operating system

3.7.3 Subsystem Software Dependencies

This class/model is dependant on two components in django's backend. The first component is that the declarations are written in python. The second component must use a special library to create the table fields in the class. The table fields must be written using a django->models format. This can be done by importing "models" from "django.db" allowing the creation of the class.

3.7.4 Subsystem Programming Languages

Given that the model is created in the models.py file the programming language "Python" must be installed to run migrations and makemigrations.

3.7.5 Subsystem Data Structures

The general data structure of the Classes class is broken down bellow.

- id : Auto Field, Primary Key
- instructor : IntegerField()
- sections : List Text Field
- number : models.IntegerField()

3.7.6 Subsystem Data Processing

No data processing, Django handles this part.

3.8 Subsystem Item

The Item class is the standard table view of storing each items' data into the database. The item needs a couple of features in order to usefull to our system. These features include name, serial number (Barcoding), Available, Total Quantity, Out, and description. Users will have the ability to check out an item or return an items each item is keyed by a unique serial number. This serial number will be used for performing tasks such as borrowing and returning items.

3.8.1 Subsystem Hardware

No supporting hardware.

3.8.2 Subsystem Operating System

No specific operating system

3.8.3 Subsystem Software Dependencies

This class/model is dependant on two components in django's backend. The first component is that the declarations are written in python. The second component must use a special library to create the table fields in the class. The table fields must be written using a django->models format. This can be done by importing "models" from "django.db" allowing the creation of the class.

3.8.4 Subsystem Programming Languages

Given that the model is created in the models.py file the programming language "Python" must be installed to run migrations and makemigrations.

3.8.5 Subsystem Data Structures

The general data structure of the Item class is broken down bellow.

- name : Char Field
- id : Auto Field, Primary Key
- type : Char Field
- location : Char Field
- description : Char Field
- total : Integer Field
- out : Integer Field
- available : Integer Field
- ser_no : Char Field

3.8.6 Subsystem Data Processing

No data processing, Django handles this part.

4 Frontend Layer Subsystems

The frontend layer encapsulates all software that creates a user interface. All outputs of the frontend are visible to the user; this layer is tasked with creating an intuitive layout by which the data within the application can be manipulated.

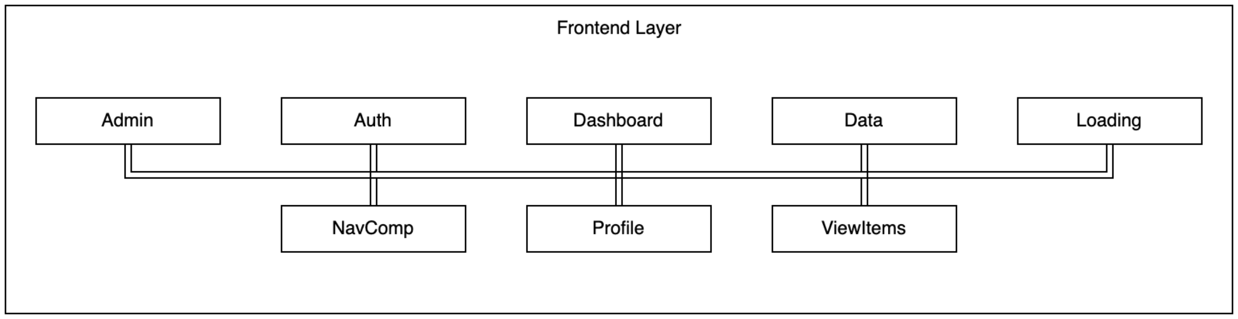

Figure 2: frontend subsystems

4.1 Layer Hardware

This layer has no hardware requirements, but input may be streamlined with the use of a card scanner device. This would allow users to paste their ID number into an input field as opposed to writing it with a keyboard.

4.2 Layer Operating System

This layer is run entirely within the web browser, so it is considered OS-independent.

4.3 Layer Software Dependencies

A React server must run on the remote web server to generate webpages and handle inputs. On the client machine, the web browser must support JavaScript.

4.4 Admin

The Admin component is available to users with elevated privileges, such as instructors and assistants. Login is required to access this section. From this interface, users are able to make global modifications to items and users in the database, such as adding, removing, or updating metadata.

4.4.1 Checkout

The checkout feature is a admin only feature. The admin will scan or type in the students ID. If the student account is found, it will start a session. In this session the student can scan items to add to cart, and remove items from the cart. The session is ended either when canceled or items in cart are checkedout.

4.4.2 Return

The return feature is a admin only feature. The admin will scan or type in the students ID. If the student account is found, it will start a session. In this session the student can scan items to add to cart, and remove items from the cart. The session is ended either when canceled or items in cart are returned.

4.4.3 Data

The Data component is tasked with one of the key features of the application: checking out and returning items. Here, users are able to associate their personal profile with items owned by UTA to create an automated inventory tracking system.

4.5 Auth

The Auth(entication) component handles the login form, where users are presented with input fields for personal identification. The outputs of this system persist through the user's session, communicating his or her privilege level to other subsystems.

4.6 Dashboard

The Dashboard component is the landing page for users who have been logged in. The actions and information available to the user on this page depend on his or her privilege level.

4.7 Loading

The Loading component is designed to give users feedback during operations which take time to perform on the backend or in the cloud.

4.8 NavComp

The Nav(igation) Comp(onent) provides the user with an organized list of actions whereby he or she may access the various functions provided by the application.

4.9 Profile

The Profile component handles the interface between the user and their personal data as stored in the database. Here, users can update contact information, view items that have been checked out in their name, and perform other related queries.

4.10 ViewItems

The ViewItems component displays the items from the database in a table, offering optional filters and categories to help users narrow their search.

5 API Layer Subsystems

The API layer hold all the functionality of the web app. This layer can be further broken down into further subsystems such as authentication, admin functions and the students functions. The hardware required for this layer include a card scanner, bar-code scanner, and a bar-code printer.

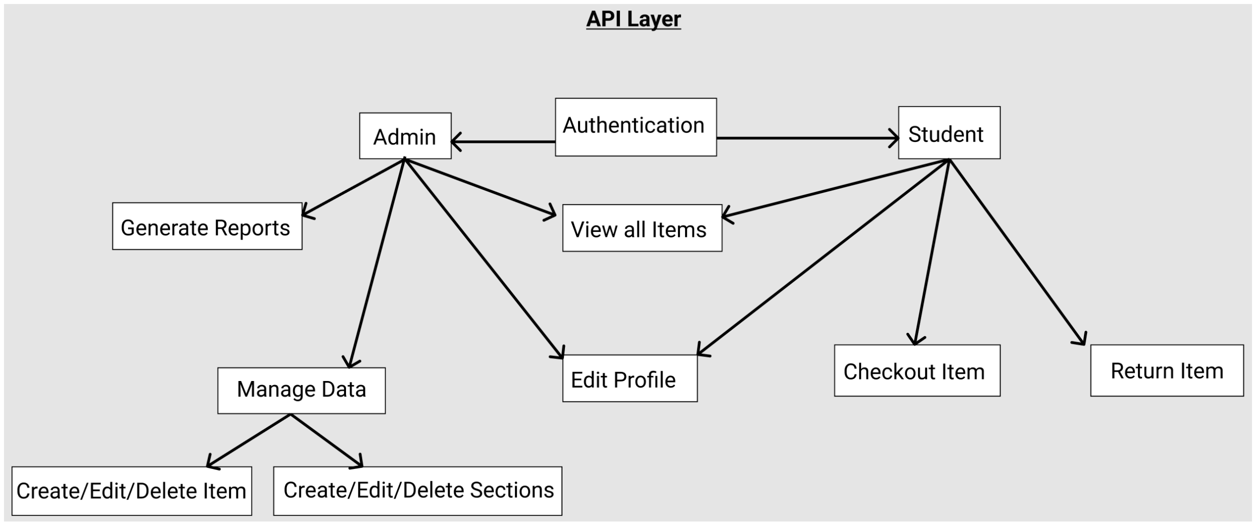

Figure 3: Example subsystem description diagram

5.1 Layer Hardware

The card scanner will only have a role in the authentication of the user extracting the users ID number. The bar-code scanner has a role in the student and admin functions. The students will use the bar-code scanner to identify items they want to check out or return. Admins will use the bar-code scanner for adding items to the database. The bar-code printer will only be used for admin functions, allowing the admins to create new bar-codes for items they want to add to the database.

5.2 Layer Operating System

No operating system restrictions

5.3 Layer Software Dependencies

The tech stack used for this web app include, react js, react bootstrap, redux js, and Django rest. The database is a mysql based database that utilizes JWT for authentication.

5.4 Subsystem Authentication

This subsystem will check whether this user has the right to access any information within the web app. If the user is logged in as a admin, they have rights to any and everything within the web app. If the user is a student, they first have to accept a disclaimer before they can access the web app internally. This layer will also allow for permission based rendering. Student accounts will have the options for checkout and returning items which admin accounts do not have. Admin accounts will have the ability to manage items, section information and run reports which students do not have.

5.4.1 Subsystem Hardware

The card scanner will be an optional hardware component for identifying the user. The user is still allowed to enter their ID number manually.

5.4.2 Subsystem Operating System

No operating system restrictions

5.4.3 Subsystem Software Dependencies

No software restrictions

5.4.4 Subsystem Programming Languages

The actual functionality of the authentication is handled by a pre-eisting library called JWT which allows for the used of session identifiers called tokens. Each token has a expiration that can be change and is currently set to 30 minutes.

5.4.5 Subsystem Data Structures

The model used for each user is a default model that comes in Django. This model had fields that allows different levels of permissions. The model also includes some of the basic information that all users require such as first name, last name, username(ID number) and email.

5.4.6 Subsystem Data Processing

The algorithm is JWT and is pre-existing. JWT can use a secret key, or a public and private key for authentication. For our web app the user is given a secret key to identify their current session. The token is generated and stored in the cache and checked every-time the user makes a request to the database. [1]

5.5 Subsystem Admin

The admin will be allowed to create new sections, add item to the database and generate reports on the items.

5.5.1 Subsystem Hardware

The bar-code printer will be used to generate identifiers for each item before they are added to the database. The bar-code scanner will be used to extract the ID from the bar-code and add the item to the database.

5.5.2 Subsystem Operating System

No operating system restrictions

5.5.3 Subsystem Software Dependencies

No software restrictions

5.5.4 Subsystem Programming Languages

No programming language restrictions

5.5.5 Subsystem Data Structures

The model used for item will hold basic information such as the item type, name, id, type, etc.

5.5.6 Subsystem Data Processing

This layer will run calculations regarding how many items are available, checked-out and overdue. The items that are either overdue or close to overdue will have special identifier. There will also be a function to send a reminder to the student that the item is overdue. The admin will also be allowed to modify all their own information such as their name and email.

5.6 Subsystem Student

The student account will only be allowed access to the site once they have accepted a disclaimer post registration. Once accepted the student will have the ability to see all the items that are available, checkout certain items, see the items they have checked-out and return the items.

5.6.1 Subsystem Hardware

The card scanner will be an optional hardware component for identifying the user. The user is still allowed to enter their ID number manually. The bar-code scanner will be used for checking-out and returning items.

5.6.2 Subsystem Operating System

No operating system restrictions

5.6.3 Subsystem Software Dependencies

No software restrictions

5.6.4 Subsystem Programming Languages

No programming language restrictions

5.6.5 Subsystem Data Structures

The student model will have an attached account model that will store their checked-out items, their section number, and their instructor.

5.6.6 Subsystem Data Processing

Once the student checks out an item they are given a certain due date for the items return. If the item is not returned by the due date the web app should notify the student. The student will also be able to modify all their own information such as their name, email, section and instructor.

6 Appendix A

Include any additional documents (CAD design, circuit schematics, etc) as an appendix as necessary.

References

[1] Auth0. Introduction to json web tokens.