GNG1503

Manuel d’utilisation et de produit pour le projet de conception

VisionMakers Inventory

Soumis par:

VisionMakers - FA11

Maya Tudor, 300359997

Juliette Demers, 300357969

Chaden Lmakroud, 300345765

Kelly Dalek Kazungu, 300386290

Mohammed Chouta, 300228982

Décembre 10, 2023

Université d’Ottawa

Table des matières

Liste de figures

Liste de tableaux

Liste d’acronymes et glossaire

Table 1. Acronymes

Acronyme	Définition
ESP32	Puce qui fournit le Wi-Fi
LCD	Liquid Crystal Display (Affichage à cristaux liquides)
MDF	Medium Density Fibreboard
MUP	Manuel d'utilisation et de produit
PN&I	Plates-formes Numériques & Interopérabilité
RFID	Radio Frequency Identification (Identification radiofréquence)
USB	Universal Serial Bus
VMI	VisionMakers Inventory
VSCode	Visual Studio Code

Table 2. Glossaire

Terme	Acronyme	Définition
Initialisation du système	IS	Lorsque le système de vérification est mis à zéro : le nombre d’échanges (ajouts et enlèvements) = zéro

1 Introduction

Ce manuel d’utilisation et de produit (MUP) fournit les informations nécessaires à des administrateurs système pour utiliser efficacement le VisionMakers Inventory (VMI) et pour la documentation de notre prototype.

Pour tirer le meilleur parti des fonctionnalités de VMI, nous avons pensé que le client recherchaient un guide complet et facile à utiliser. Le document est présenté de manière à guider les utilisateurs à travers chaque étape, de la configuration initiale à l'utilisation régulière du système.

L'objectif principal de ce manuel est de permettre aux utilisateurs (membres du personnel, gestionnaires d'inventaire et gestionnaires de stocks) de comprendre et d'utiliser plus facilement toutes les fonctionnalités du VMI. Ce document traite de l'installation, de la configuration et de l'utilisation continue de VMI.

Des mesures de sécurité et de confidentialité sont incluses pour garantir que les données d'inventaire privées sont conservées en sécurité. Les utilisateurs peuvent explorer les avantages de ce prototype de système de gestion d'inventaire et devenir habile à l'utilisation de VMI en suivant prudemment ces instructions.

2 Aperçu

Le VMI répond à un besoin fondamental dans le secteur des PN&I des Services Partagés Canada. Le problème à résoudre est la nécessité d'un système de suivi d'inventaire automatique, efficace et simple, visant à minimiser les pertes d'argent et de temps. Dans ce contexte, les besoins fondamentaux des utilisateurs sont les suivants : un système paramétrable, fiable, combinant des solutions physiques et logicielles, et surtout, respectant un budget de 50$ et pouvant être conçu avant la Journée de la Conception (30 novembre 2023).

Les caractéristiques principales incluent un suivi automatique de l'inventaire (zéro perte), une interface utilisateur et une conception qui intègre à la fois des éléments physiques et logiciels. Cette combinaison fait du VMI un produit très efficace pour le suivi d'inventaire, qui respecte les contraintes de budget spécifiées.

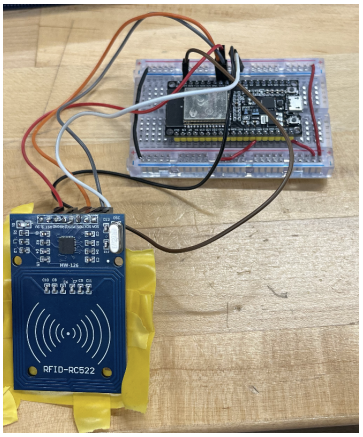


Figure 1.1

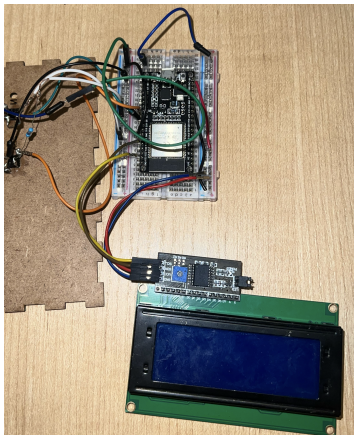


Figure 1.2

Figure 2.1

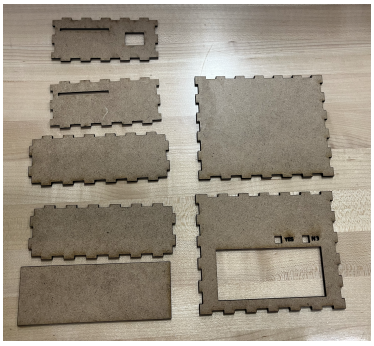


Figure 2.2

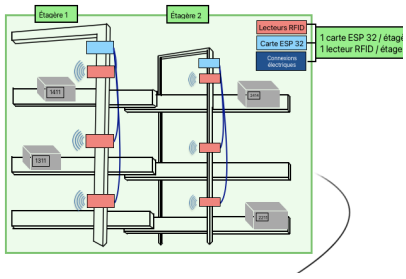


Figure 3

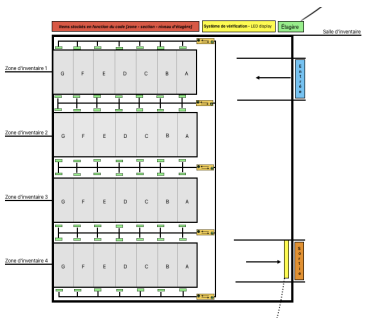


Figure 3

Un résumé de l’ensemble du produit peut être visualisé par les photos ci-haut, qui démontrent le prototype final. Les principales fonctions du VMI incluent un suivi automatisé des stocks, une interface et une conception puissante qui assure l’utilisation fiable et durable.

Le système d'inventaire possède une architecture composée de deux systèmes interdépendants dans des boîtiers (MDF), chacun ayant un rôle important dans le suivi automatisé des items dans la salle d'inventaire.

Le premier système, sur chaque étagère, est composé d'un microcontrôleur ESP32 et d'un lecteur RFID. Chaque lecteur RFID est connecté au microcontrôleur de l'étagère, ce qui permet la transmission d'informations vers le microcontrôleur. Les données sont ensuite transmises à l'ordinateur central par un câble micro USB. Ces lecteurs RFID automatisent l'identification des items sur chaque étagère, ce qui facilite la collecte de données. (110 mm x 75 mm x 50 mm)

Le deuxième système, retrouvé à la sortie de la salle, est composé d'un microcontrôleur ESP32, d'un écran LCD et de deux boutons de vérification (oui et non). Ce système fonctionne en comparant les lectures avec un calcul interne, ce qui permet de déterminer les échanges d'items (ajouts et enlèvements). À la sortie, l'utilisateur, généralement un travailleur, interagit en indiquant si les lectures correspondent au calcul interne en appuyant sur les boutons OUI ou NON. Après chaque réponse utilisateur, le système réinitialise automatiquement le calcul, se préparant ainsi pour le prochain utilisateur. (120 mm x 95 mm x 35 mm)

Ce système combine donc l'automatisation des lectures sur chaque étagère avec une vérification manuelle à la sortie par les utilisateurs. Ceci assure un suivi précis des items, avec une réinitialisation automatique après chaque interaction utilisateur. L'interdépendance entre les deux systèmes garantit une vérification soigneuse et une gestion fiable de l'inventaire dans la salle.

2.1 Conventions

La simplicité du système permet peut d’action de la part de l’utilisateur.Les travailleurs dans la salle d’inventaire n’ont que deux responsabilités, d’où une d’elles est d’effectuer les échanges d’items (les lectures RFID sont automatiques).

1) Placer l’item sur l’étagère. // Enlever l’item de l’étagère.

   La deuxième s’agit de vérifier le nombre d’ajouts et d’enlèvements à leur sortie par le bouton OUI ou NON. Consulter l’encadrement inférieur de la Figure 3 pour une représentation réelle de l’affichage de l’écran de vérification et la Figure 2

pour une représentation physique du boîtier, de l'écran et des boutons de vérification.

2) Cliquer OUI on NON sur le boîtier de l'écran LCD à la sortie de la salle. Notez l'erreur.

Les gérants de la salle n'ont aucune action à prendre lorsque la vérification est positive. Lors d'une réponse négative de la part de l'utilisateur, le gérant est averti par une notification automatique et doit manuellement corriger l'erreur.

1) Corriger manuellement les erreurs d'échanges lors d'une notification de vérification négative. Le travailleur de la salle d'inventaire a noté l'erreur.

2.2 Mises en garde et avertissements

Quelques points à savoir avant d'utiliser le produit physique :

1) Les boitiers s'ouvrent par le haut, veuillez faire attention aux fils et aux composantes fragiles.
2) Afin d'éviter toute interférence, ne pas rapprocher les cartes de paiement électronique ou autres cartes magnétiques au lecteur RFID.
3) Évitez de toucher les composants délicats, tels que les cartes ESP 32, avec des mains chargées d'électricité statique. Faire usage de bracelets antistatiques lorsque requis.
4) Employez uniquement les adaptateurs d'alimentation conseillés pour chaque composant afin de prévenir les dangers de surtension.
5) Planifiez des contrôles réguliers pour assurer que chaque composante opère efficacement. Remplacez ou réparez les éléments dysfonctionnels au plus tôt que possible.

3 Pour commencer

Le guide suivant propose des instructions complètes pour la configuration, l'utilisation, l'installation, la navigation et comment le quitter. Chaque section est jointe d'images pour montrer les étapes plus en détails. Nous vous prions de suivre le plus que possible l'ordre suggéré pour la meilleure utilisation possible.

3.1 Considérations pour la configuration

3.1.1 Prototype physiques

La configuration physique du système est présentée sous cette section. Utilisez les outils inclus dans le kit et suivez les étapes indiquées par les illustrations. Puis, vérifiez que toutes les connexions ont été établies correctement.

Partie RFID:

Étape 1: Ouverture et Vérification

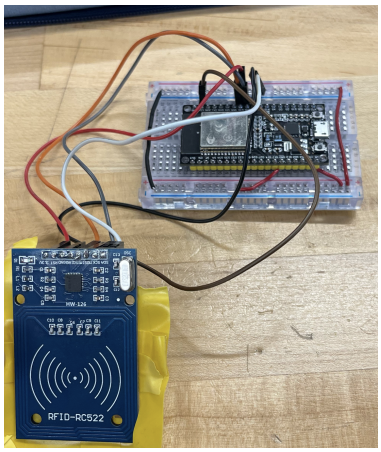


Figure 1.1

a) Lorsque vous ouvrez le paquet vous y trouver le système suivant

b) Assurez-vous que tout soit bien connecté comme dans la **Figure 1.1** (si il semble y avoir un problème contactez le support technique)

Étape 2: Assemblage boîte

Figure 1.2

a) Puisque le système vous est donné, vous n'avez juste qu'à assembler la boîte.

b) En vous fiant sur la **Figure 2.1** assurer de coller les morceaux à la bonne place **sans coller immédiatement le couvercle**
(Pour ceci utiliser la colle trouver dans le kit)

Étape 3: Assemblage complet

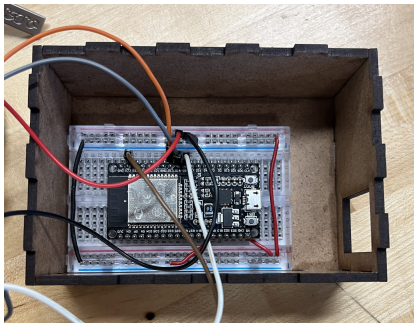


Figure 4

a) Après avoir créé la boîte, avec la colle (dans le kit) coller la plaque qui a l'ESP 32 au fond de la boite comme montré dans la **Figure 4**

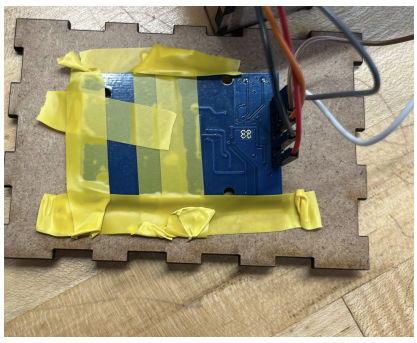


Figure 5

b) Maintenant, avec le ruban électrique, coller la face du lecteur RFID au couvercle.
(Il est important que la face soit coller pour assurer d'être capable de faire les lectures comme le montre la **Figure 5**)

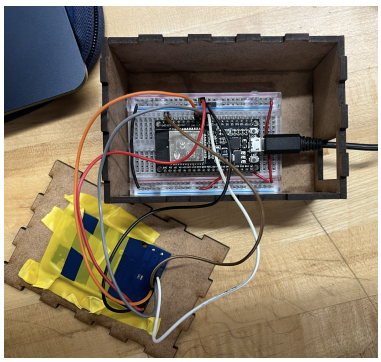


Figure 6

c) Après avoir collé les 2 systèmes, le tout devrait ressembler à la **Figure 6**

d) Avec le câble micro USB (pas inclus) faire la connexion par le troup dans la boîte.

Figure 7

e) Fermer le tout et vous avez fini l'assemblée du premier système. (**Figure 7)**
(si vous voulez coller le couvercle vous le pouvez, par contre, nous ne le recommandons pas en raison de au cas ou il y a des probleme)

Partie Écran:

Étape 1: Ouverture et Vérification

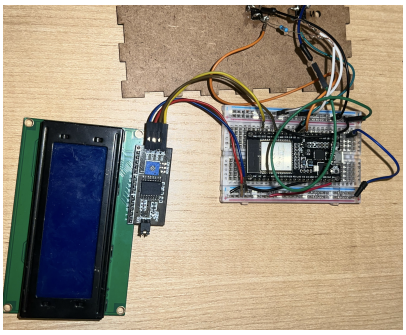


Figure 1.2

a) Lorsque vous ouvrez le paquet vous y trouver le système suivant

b) Assurez-vous que tout soit bien connecté comme dans la **Figure 1.2** et que les boutons soit bien souder au couvercle
(si il semble y avoir un problème contactez le support technique)

Étape 2: Assemblage boîte

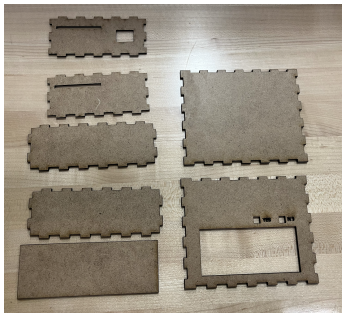


Figure 2.2

a) Puisque le système vous est donné, vous n'avez juste qu'à assembler la boîte.

b) En vous fiant sur la **Figure 2.2** assurer de coller les morceaux à la bonne place **sans coller immédiatement le couvercle**
(Pour ceci utiliser la colle trouver dans le kit)

**évidemment les boutons sont déjà pré souder, alors vous n'avez pas à le faire

Étape 3: Assemblage complet

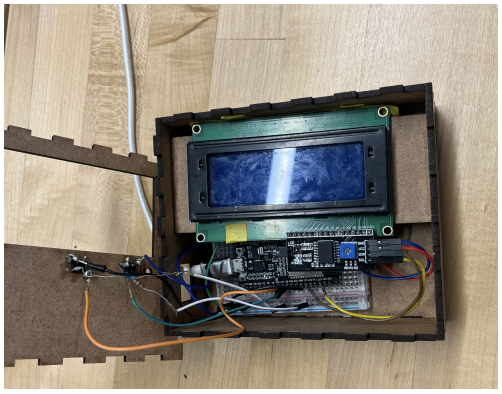
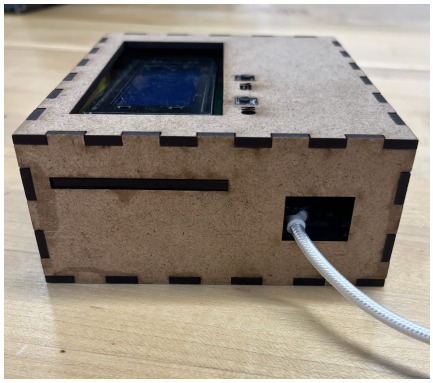


Figure 8

a) Après avoir créé la boîte, avec la colle (dans le kit) coller la plaque qui a l'ESP 32 et l'écran au fond de la boite comme montré dans la **Figure 8**



b) Assurez-vous que la vue du côté est la même que la **Figure 9.1** pour le côté au le câble micro USB fait connexion.

Figure 9.1

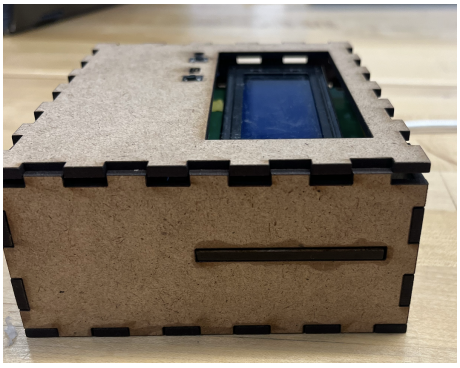


Figure 9.2

c) Assurez-vous que la vue du côté est la même que la **Figure 9.2** pour l'autre côté.

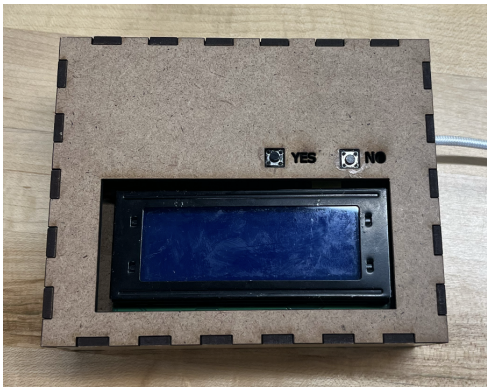


Figure 10

d) Fermer le tout et vous avez fini l'assemblée du deuxième système. (**Figure 10)**
(si vous voulez coller le couvercle vous le pouvez, par contre, nous ne le recommandons pas en raison de au cas ou il y a des probleme)

3.1.2 Prototype logiciels

La configuration logicielle du système est présentée sous cette section. Vérifiez que vous disposez des appareils et des bases de données d'entrée et de sortie nécessaires, et vérifiez que vous disposez des communications requises. Appliquer les instructions suivantes sous un environnement Linux (Debian 12 de préférence) et s'assurer d'avoir Python3 installé:

Étape 1:
Ouvrir un terminal et taper les commandes suivantes:

- Python3 -m venv GNG1503
- Mkdir GNG1503

- Source bin/activate
- Pip install flask

Étape 2:

- Brancher le scanner
- Lancer la commande: *sudo chmod a+rw /dev/ttyUSB0*
- Brancher l'écran
- Lancer la commande: *sudo chmod a+rw /dev/ttyUSB1*
- Sur l'éditeur Visual Studio Code, installer l'extension PyMakr et cliquer sur son icône après qu'elle apparaisse.

Étape 3:

- Télécharger le code a partir du lien suivant: https://github.com/MedChouta/GNG1503 et mettre tout le contenu du dossier dans le dossier GNG1503 créé avant.
- Sur VSCode, ajouter un nouveau projet en cliquant sur le signe +, en sélectionnant le dossier Scanner puis en choisissant *ttyUSB0* (Faire de même pour Screen en choisissant *ttyUSB1)*

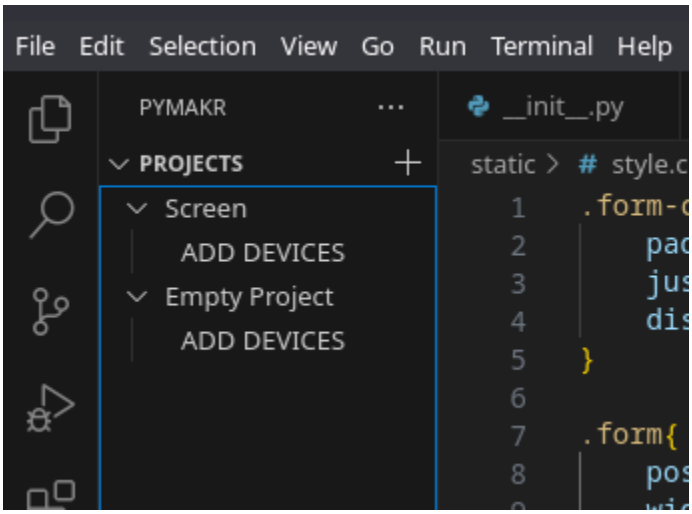


Figure 11

- Connecter les deux boitier en cliquant sur l'icône suivante:

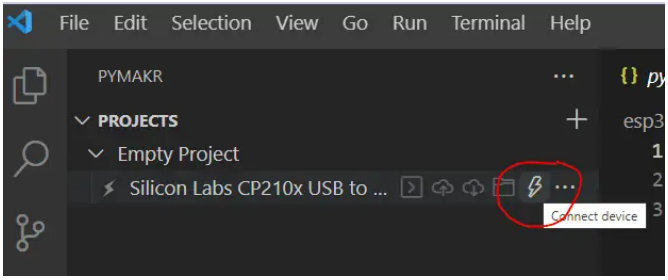


Figure 12

- Cliquer sur les trois point a cote et cliquer sur *stop script* (pour les deux boitiers)

Étape 4:

- Connecter le scanner et l'écran au réseau wifi local:
  - Dans le fichier Screen/boot.py, modifier la ligne 37 en mettant le nom du wifi comme premier paramètre et le mot de passe comme deuxieme parametre (Faire de même pour le fichier Scanner/boot.py a la ligne 32)

```
34
35      if not connected:
36          sta_if.active(True)
37          sta_if.connect("ESP", "esp32_123")
38          print("No connection :(")
39
40          while not sta_if.isconnected():
41              pass
42
43      print("Connected: " + str(sta_if.isconne
44
```

Figure 13

- Connecter l'ordinateur au même réseau local:
  - Trouver l'adresse IP locale (commencent généralement par 192.168) en lançant la commande: *ip addr*
  - Remplacer L'adresse IP dans le code par celle de l'ordinateur dans le fichier Scanner/boot.py a la ligne 11 (Faire de même pour le fichier Screen/boot.py a la ligne 9):

```
 8    import urequests, ujson
 9
10
11    IP = "192.168.187.183"
```

Figure 14

- Telecharger le code sur les boitier en cliquant sur le nuage de gauche

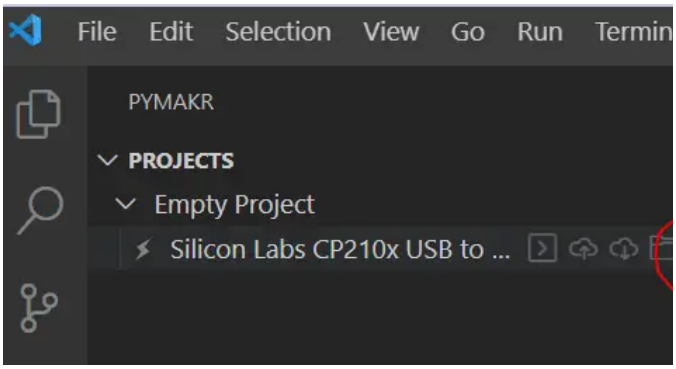


Figure 15

Étape 5:

- Lancer la commande: *flask --app __init__ run --host=0.0.0.0*
- Ouvrir un navigateur et taper *localhost:5000*

3.2 Considérations pour l'accès des utilisateurs

- Chacun des utilisateurs de VMI auront accès à des points, cependant ces points varient en fonction de deux situations : si l'utilisateur est un utilisateur prioritaire ou non.

  Les utilisateurs non prioritaires auront accès à :

- l'inventaire répertoriant les articles présents dans l'entrepôt au moment où celui-ci est vérifié.
- La quantité de chaque items de l'entrepôt.
- Une autorisation d'ajout ou de retrait d'items par le billet d'une carte de scanner RFID.

- Le client prend les décisions concernant la sélection et la hiérarchisation des utilisateurs. Une fois les identités de ces personnes fournies, les utilisateurs considérés comme prioritaires auront accès à des fonctionnalités plus avancées que les autres.

- Une fois cela terminé, nous proposons aux utilisateurs prioritaires un accès à plusieurs point uniques comme :
  - L'accès à un historique de toutes les actions effectuées au saint de lentrepot (ajout et le retrait d'items).
  - Une possibilité de bannir les utilisateurs non désirés.
  - Une liste de toutes les personnes ayant un compte sur VMI.
  - Tous les autres points auxquels les utilisateurs non-prioritaires ont accès.

3.3 Accéder/installation du système

3.3.1 Prototype physiques

Si vous ne possédez pas le boîtier, veuillez suivre les étapes précédentes sur la formation du produit physique. Dans le cas ou vous possédez les boîtiers avec les composantes déjà à l'intérieur, veuillez suivre ces étapes :

1) Analysez le boîtier et son intérieur pour assurer que toutes les composantes s'y trouvent et sont reliées adéquatement (voir Figure 1.1 pour le système RFID et la Figure 1.2 pour le système de vérification par écran LCD).
2) Vérifier que les composantes sont bien sécurisées dans la boîte et ne vont pas se déplacer.
3) Spécifique au boîtier des lecteurs RFID :
   a) Vérifier que les lecteurs RFID utilisés sont bien configurés et compatibles avec notre système RFID.
   b) Calibrer les lecteurs RFID pour assurer qu'ils fonctionnent efficacement. Se référer aux instructions du fabricant pour la méthodologie de calibration.
4) Brancher à l'aide du câble micro USB (le câble qui sort du boîtier, attaché à la carte ESP 32) le système à un ordinateur.

5) Ajouter les couvercles aux boîtiers.
6) Ouvrir l'interface sur l'ordinateur.
7) Effectuer un essai pour vérifier le fonctionnement du lecteur RFID, de l'écran LCD et des boutons.
    a) Si le lecteur RFID ne perçoit pas le tag, consulter le support technique.
    b) Chaque lecture RFID devrait apparaître sur l'écran. La réponse OUI devrait permettre au système de réinitialiser. La réponse NON devrait inciter une alerte de défaillance.
    c) Si le système de vérification ne fonctionne pas correctement, consultez le support technique.
8) Sécuriser les boîtiers dans leurs emplacements dans la salle d'inventaire.
    a) Un boîtier de lecteur RFID par étagère.
    b) Le boîtier du système de vérification (l'écran LCD) est installé à la sortie de manière évidente pour limiter les réponses manquantes.
9) Commencer l'utilisation du système d'inventaire.

3.3.2 Prototype logiciels

Pour allumer un système il suffit tout simplement d' avoir un compte utilisateur associé à votre courriel et ainsi se connecter sur ce compte. La création d' un compte utilisateur est simple et rapide car il suffit d'accéder à la page du système et de remplir le formulaire vous demandant votre identité , votre courriel et enfin vous demandant d' entrer un mot de passe qui vous permettra d'accéder au système.Pour la réinitialisation et/ou la modification du mot de passe, il est recommandé de le faire en ces étapes qui sont:

- Donner/fournir le mot de passe que vous voulez modifiez(si vous voulez modifiez votre mot de passe)
- Accéder à la page d'accueille du système vous demandant de vous identifiez par votre compte utilisateur , ensuite cliquer sur 'mot de passe oublié' ou 'forgot password' et vous allez être rediriger sur une page de gestionnaire des mots de passe
- Enfin donnez votre nouveau mot de passe et le confirmer en l'écrivant une deuxième fois

De cette façon votre mot de passe sera réinitialisé et/ou modifier selon vos besoins.

3.4 Organisation du système & navigation

3.4.1 Prototype physiques

<u>Les composantes du boitier RFID :</u>

- Lecteur RFID : permet la détection d'items pour permettre la confirmation sur l'écran LCD.
- ESP 32 : permet la liaison entre le système (lecteur RFID) physique et la partie concernant l'interface logiciel.
- Fils : permet de lier les composantes tel que le lecteur RFID et l'ESP 32 ensembles pour établir un lien direct avec l'inventaire de la partie logiciel, ce qui permet de former un système.

Les composantes du boitier LCD :

- Écran LCD : Permet une confirmation visuelle simple de la détection du lecteur RFID.
- ESP 32 : permet la liaison entre le système (lecteur RFID) physique et la partie concernant l'interface logiciel.
- Fils : permet de lier les composantes tel que le lecteur RFID et l'ESP 32 ensembles pour établir un lien direct avec l'inventaire de la partie logiciel, ce qui permet de former un système.

Les deux boitiers sont complémentaires et sont tout aussi liés à la partie logicielle qui se focalise essentiellement sur l'inventaire des items au sein de l'entrepôt.

3.4.2 Prototype logiciels

Les deux boitiers sont complémentaires et sont tout aussi liés à la partie logicielle qui se focalise essentiellement sur l'inventaire des items au sein de l'entrepôt.

Après que vous (l'utilisateur) vous rendez sur la page d' accueil vous pouvez voir tous les items dans l'entrepôt en d'autres mots vous pouvez voir comment se présente toute l'entrepôt et ainsi avoir une image de l'entrepôt.

3.5 Quitter le système

3.5.1 Prototype physiques

Pour procéder au rangement du système physique, veuillez débrancher les boîtiers du dispositif informatique en usage (ordinateur, tour, etc.) et enrouler le câble USB autour du boîtier auquel il est connecté. Veuillez noter que ces boîtiers assurent la protection des composants qu'ils renferment.

3.5.2 Prototype logiciels

Les actions nécessaires pour quitter/éteindre correctement le système il est vivement recommandé de bien fermer toutes les fenêtres avant de se déconnecter du système et ensuite de le quitter.

En vous déconnectant vous assumez que vous êtes conscient que vous fermez votre session ; et lorsque vous vous reconnectez vous trouverez vos actions sans aucune modification.

Et pour vous déconnecter, il suffit de cliquer en haut à droite de votre écran sur la page sur laquelle vous êtes entré pour travailler le mot ‘Log out ou bien Se déconnecter’ ,et ainsi vous pouvez quitter le système.

4 Utiliser le système

A chaque entrée le système demande à l' utilisateur de se connecter à son compte associer à son courriel et ensuite après avoir se connecter à son compte, il peut bien voir les items dans l'entrepôt, leur date d'entrée et l'étagère où ils se situent, ainsi qu' une option de supprimer un item.

Il y a aussi un onglet intitulé 'AJOUT ou ADD' dans le coin inférieur gauche de l' interface où se situe les items dans l'entrepôt , et de la l' utilisateur pourra ajouter un item dans l' entrepôt. Et le système va lui demander le nom de l' item ainsi que où il veut le mettre(la localisation de l' item). L'interface logiciel fonctionne correctement au niveau du login et démontre bien les items dans l'entrepôt, leur date d'entrée et l'étagère où ils se situent.

4.1 Accéder à l'Inventaire

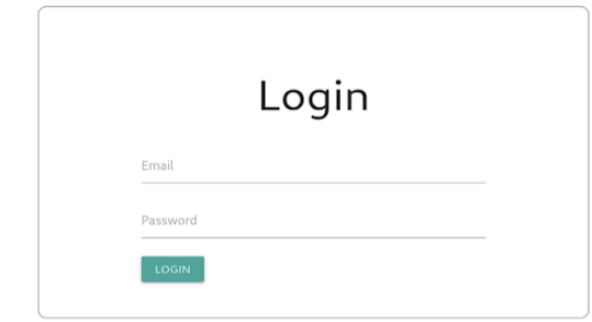


Figure 16

A chaque entrée le système demande à l' utilisateur de se connecter à son compte associer, et de fournir son mot de passe afin qu' il puisse accéder aux données de l' entrepôts .

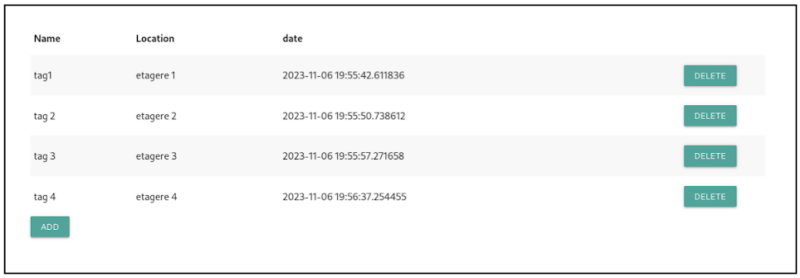


Figure 17

Après avoir pu se connecter au système il sera rediriger vers un nouvel onglet lui montrant l' entrepot toute entière , et ainsi lui permettant de voir les items dans l'entrepôt, leur date d'entrée et l'étagère où ils se situent, ainsi qu' une option de supprimer(DELETE) un item comme le montre le schéma ci haut.

4.1.1 Création d'un compte

Pour pouvoir accéder au système, l' utilisateur devrait avoir un compte le reliant avec le système(l'entrepôt) qui y est associé à son courriel afin d' avoir des notifications aux besoins s' il y a quelque chose à signaler de suspect.

4.2 Ajout d'un Nouvel Item

4.2.1 Comment

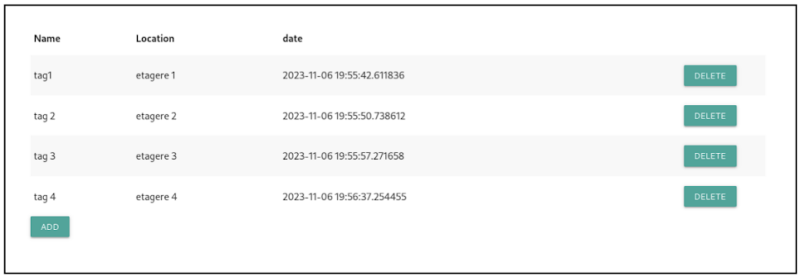


Figure 18

Comme illustré sur cette image dans le coin inférieur gauche se trouve un mot (ADD ou AJOUT en français) permettant ainsi à l' utilisateur d' accéder à la page d' ajout d item;

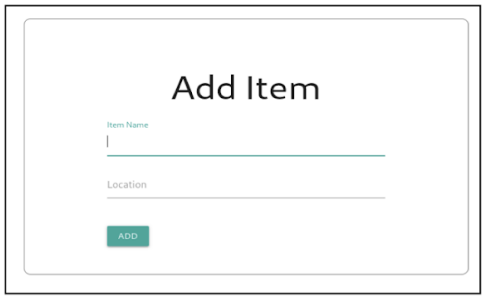


Figure 19

Après avoir été redirigé sur cette page le système demandera à l' utilisateur de donner le nom de l' item qu il veut entrer dans l'entrepôt (ITEM Name) et ainsi de donner sa localisation càd de lui donner un emplacement dans l' entrepôt(numéro d’ étagère) , et enfin terminer son ajout en cliquant sur ‘ADD ou AJOUT’ sur son écran.

5 Dépannage & assistance

De nombreuses entreprises dépendent de la gestion d'inventaire, et la mise en place d'un système productif peut vous aider à éviter des erreurs coûteuses. Les sous-sections ci-dessous fournissent un aperçu des processus de récupération et de correction des erreurs ainsi que des conditions d'erreur potentielles.

5.1 Messages ou comportements d'erreur

Divers messages d'erreur ou comportements anormaux peuvent être rencontrés par l'utilisateur. Déterminez les causes principales, leurs conséquences probables et les mesures correctives associées.

5.1.1 Problème de connection

Lors de problèmes de connexion, la cause la plus fréquente est un problème avec le réseau. La démarche à faire dans ce cas est de:

- Redémarrer votre Wi-Fi
- Vérifier votre connection au réseau

Par contre si le problème ne se résout pas, contactez le support technique.

5.1.2 Erreur dans le système

Lorsque des messages d'erreur s'affiche, la cause la plus fréquente est un problème avec la partie logiciel. La démarche à faire dans ce cas est de redémarrer le système au complet. Par contre si le problème ne se résout pas, contactez le support technique.

5.2 Considérations spéciales

Une approche distincte du dépannage est nécessaire dans certaines situations. Pour garantir une résolution réussie, examinons ces situations.

5.2.1 Mise en garde

Il y est important de ne pas modifier ou changer la configuration du système sans avoir été autorisé. Si jamais vous avez besoin pour quelque soit la raison de modifier ou changer la configuration, contactez le support et suivez strictement la procédure qui sera fournie.

5.3 Entretien

Pour éviter les pannes, un entretien régulier est indispensable. Pour assurer la robustesse du système d'inventaire, conformément aux instructions fournies dans cette section.

- Il est important de vérifier le système quotidiennement pour s'assurer que tout est en place.
- Il est aussi important de prendre en contre la mise à jour de certains aspects logiciels. Pour cela, assurez vous de regarder mensuellement pour des mises à jour.

5.4 Assistance

Pour bénéficier d'une assistance d'urgence ou d'un système d'assistance, veuillez suivre les procédures ci-dessous:

5.4.1 Assistance d'urgence

En cas d'urgence demandant une assistance immédiate, veuillez contacter les responsables d'assistance d'urgence 24/7. Les personnes responsables et leurs coordonnées sont les suivantes :

Personne responsable: Maya
Adresse e-mail: mtudo102@uottawa.ca
Instructions: Pour des cas d'urgence, contactez l'assistance d'urgence en spécifiant clairement le problème.

5.4.2 Assistance système

Pour tout problème non urgent mais demandant une assistance système, veuillez contacter le responsable de support technique ou de production. Les personnes responsables et leurs coordonnées sont les suivantes :

Support Technique

Personne responsable: Mohammed
Adresse e-mail: mchou091@uottawa.ca
Instructions: Pour des problème technique non urgent contactez le support technique lundi à vendredi entre 8h à 17h

Support de Production

Personnes responsable: Chaden et Juliette
Adresse e-mail: clmak070@uottawa.ca et jdeme075@uottawa.ca
Instructions: Pour des problème ou question par rapport à la production contactez le support de production de lundi à vendredi entre 8h à 17h

5.4.3 Sécurité

Pour tout suspicion de problème de sécurité, contactez notre responsable de sécurité et suivez bien tous procedure donner spécifique:

Personnes responsable: Kelly
Adresse e-mail: kkazu095@uottawa.ca
Instructions: Contactez les de lundi à vendredi entre 8h à 17h

6 Documentation du produit

6.1 Prototype final

Ce système de gestion d'inventaire utilise les RFID et une vérification par LCD qui a pour objectif d'automatiser le processus de suivi des stocks. Son importance est fixée par l'amélioration de précision, de l'efficacité, et de la rapidité de gestion des inventaires, contribuant ainsi à réduire les erreurs humaines (zéro perte). Ici est ou se trouve la documentation en détails complète avec tout code pour la création du prototype final. Les instructions seront séparées en 3 parties; conception électrique, mécanique et logiciel.

6.1.1 NDM (Nomenclature des Matériaux)

Voici la liste de matériaux que nous avons utilisés lors de ce projet.

Nomenclature des Matériaux				
Nº	**Description du composant**	**Lien**	**Quantité**	**Prix unitaire ($)**
1	Carte ESP 32	https://edu-makerlab.odoo.com/fr_CA/	2	13,99
2	Plaque de prototypage	https://edu-makerlab.odoo.com/fr_CA/	2	2,50
3	Lecteur RFID	https://edu-makerlab.odoo.com/fr_CA/	1	1,00
4	Tags RFID	https://bc-robotics.com/shop/mifare-classic-13-56mhz-rfidnfc-blue-tag/	1	0,75
5	Écran LCD	https://edu-makerlab.odoo.com/fr_CA/	1	10,56
6	Boutons	https://edu-makerlab.odoo.com/fr_CA/	2	0,25
8	Résistors - 330 Ω	https://edu-makerlab.odoo.com/fr_CA/	2	0,01
9	Fils électroniques (male-male)	http://makerstore.ca/	6	0,10
10	Fils électroniques (male-femelle)	http://makerstore.ca/	4	0,10

11	Câble connexion	http://makerstore.ca/	1	0.80
12	MDF (12'' X 24'')	https://makerstore.ca/shop/ols/products/mdf	1	2,50
13	Câble micro USB	-	2	-
14	Ordinateur	-	1	-
15	Python	-	Programme	-
			Total	**50.11**

****Notez bien**: Pour ce projet, notre gestionnaire de projet nous a fourni le tag RFID parce que la classe en raison de plus de la moitié de la classe en avait besoin, alors, nous ne savons pas en réalité combien ils ont coûté ou de quel site ils ont été acheter. **Donc notre total que nous avons dépensé est de 49.36 $**

6.1.2 Liste d'équipements

Voici la liste d'équipement que nous avons utilisés lors de ce projet.

- Découper laser (formation requise)
- Colle
- Soudage (formation requise)
- Ruban électirque

6.1.3 Instructions - Conception Électrique

Étape 1: Connection du système RFID

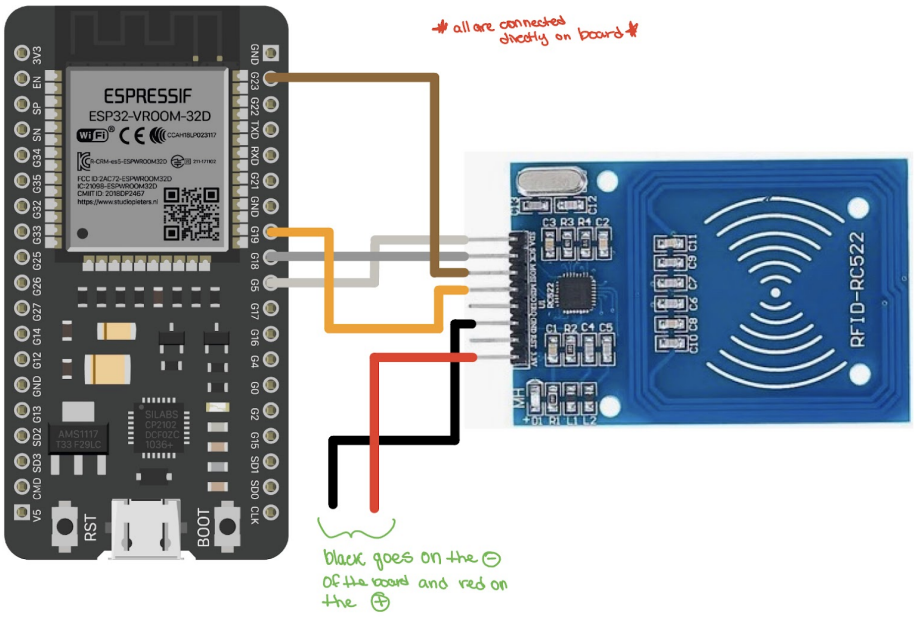


Figure 20

Pour ce qui est du processus de création du système RFID avec ESP32, il faut d'abord savoir quoi connecter a quoi, alors pour nous avons utiliser:

- ESP32
- Lecteur RFID (RC522)
- Des fils de connection
- Plaque de prototypage

Tout d'abord faire la connexion des composantes comme le démontre le schéma de la **Figure 20** ou l'image de la **Figure 1.1.** Après, on passe à la connexion du deuxième système.

Étape 2: Connection du système LCD

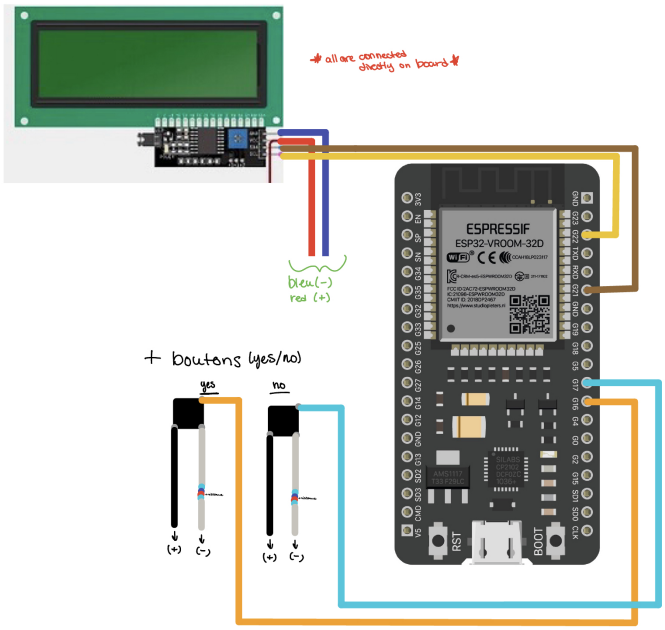


Figure 21

Pour ce qui est du processus de création du système RFID avec ESP32, il faut d'abord savoir quoi connecter a quoi, alors pour nous avons utiliser:

- ESP32
- Écran LCD
- Des fils de connection
- Plaque de prototypage
- 2 boutons

Tout d'abord faire la connexion des composantes comme le démontre le schéma de la **Figure 21** ou l'image de la **Figure 1.2.**

Étape 3: Soudage des boutons

Après avoir fait les connexions il nécessitent de faire des tests et s'assurer que le tout fonctionne bien. Lorsque les composantes fonctionnent ensemble, il ne nécessitent que de soudeurs les boutons (formation est requise), pour s'assurer que les fils ne se déconnectent pas.

6.1.4 Instructions - Conception Mécanique

Pour les boitiers nous avons pris comme décision d'utiliser du MDF, par contre il est possible de les faire par imprimante 3D. En raison de contraintes de temps, il était plus facile pour nous d'utiliser le MDF par découpe laser.

Étape 1: Création des boîtiers

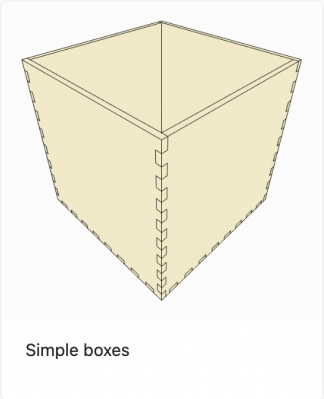


Figure 22

Grâce à un lab, l'équipe au complet on reçu la formation pour la découpe laser. Alors, nous avons trouvé un site qui aide à créer des boîtes. https://www.makercase.com/#/. Une fois sur le site on a décidé d'aller avec «Simple boxes » (**Figure 22**)

Pour la boîte pour le RFID nous avons décider les mesures suivantes (sur makercase, 110 mm x 75 mm x 50 mm),

Couvercle et base : 110 mm X 75 mm
Côtés : 75 mm X 50 mm
- *Trou câble Micro-USB (sur 1 des côtés) : 15 mm X 20 mm*

Avant et arrière : 110 mm X 50 mm

Pour la boîte pour le LCD nous avons décider les mesures suivantes (sur makercase, 120 mm x 95 mm x 35 mm),

Couvercle et base : 95 mm X 120 mm
- *Trou écran : 100 mm X 45 mm*
- *Trous boutons (2) : 6.5 mm X 6.5 mm*

Côtés : 110 mm X 50 mm
- *Trou câble Micro-USB (sur 1 des côtés) : 15 mm X 20 mm*
- *Trous plate-forme (2 cotés) : 50 mm X 3.5 mm*
- *Platforme : 135 mm X 50 mm*

Avant et arrière : 35 mm X 120 mm

Étape 2: Découpages des boîtiers

Dans l'annexe 9 il y a plus d'instructions pour utiliser la machine. Il est important que la machine soit mise à 0.0001 inch. Pour l'écran, vector et engraved puis le RFID seulement vector

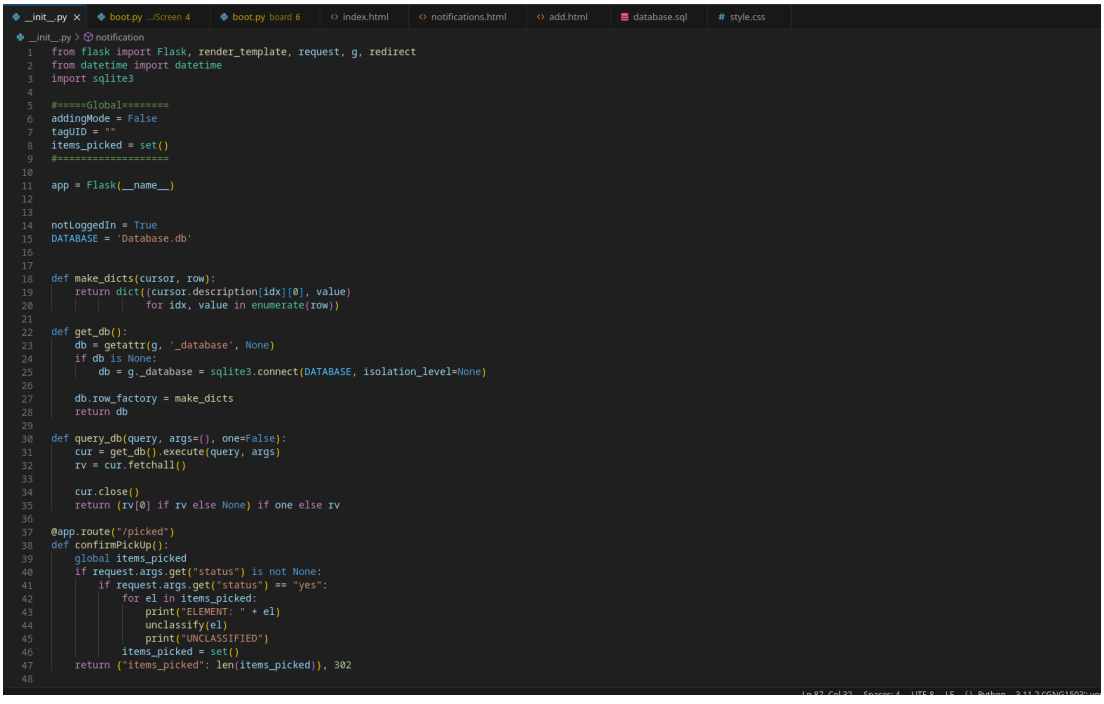
Étape 3: Assemblage

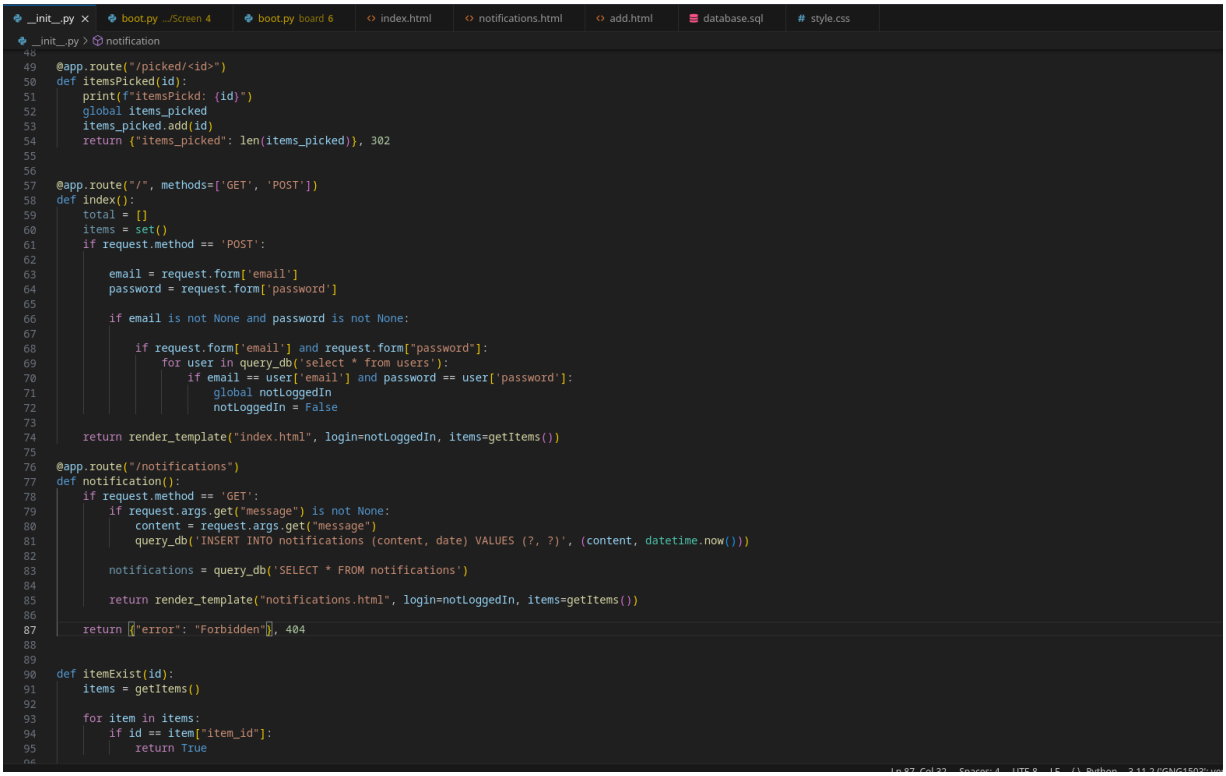
Pour l'assemblage, nous avons utilisé de la colle forte pour coller ensemble le tout, sauf pour les couvercle. Après on a mis le système dans le bon boîtier comme le montre les figures 8 et 6.

6.1.5 Instructions - Conception Logiciel

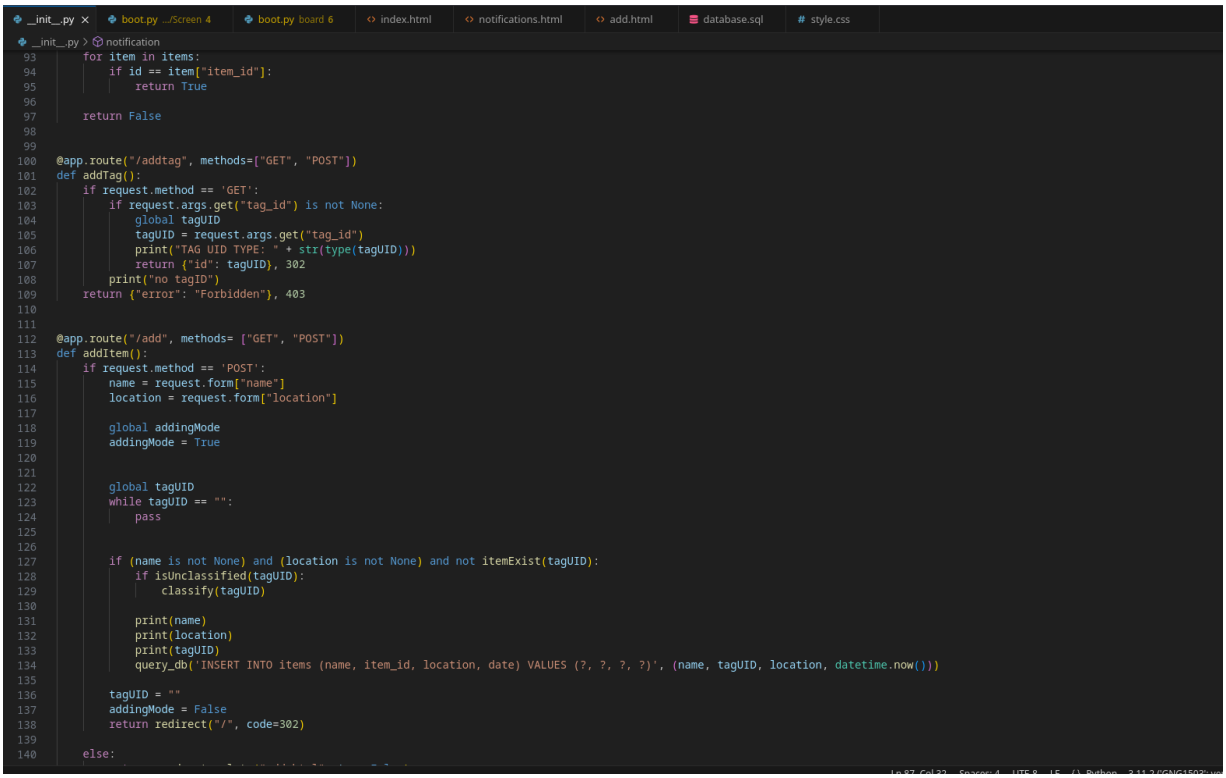
Dans la partie ci dessous, vous y trouver les code que nous avons créé pour ce système (python)

Server code

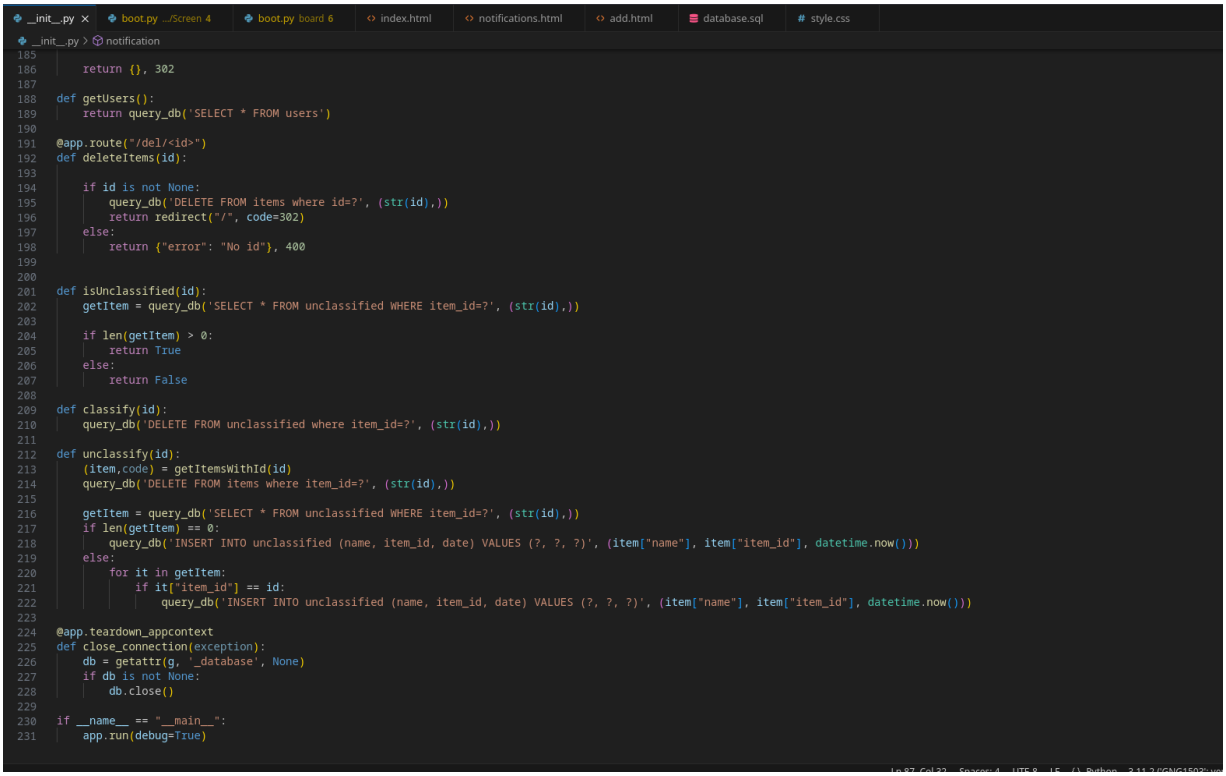




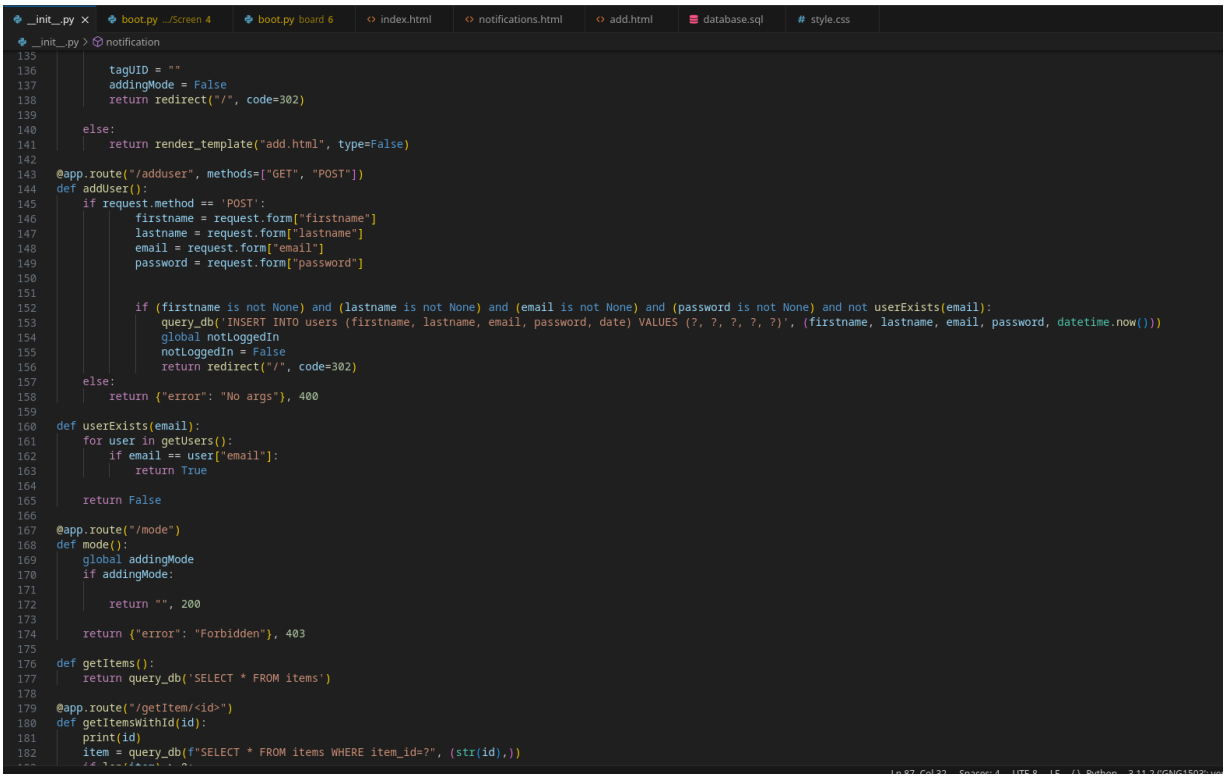
```python
@app.route("/picked/<id>")
def itemsPicked(id):
    print(f"itemsPickd: {id}")
    global items_picked
    items_picked.add(id)
    return {"items_picked": len(items_picked)}, 302


@app.route("/", methods=['GET', 'POST'])
def index():
    total = []
    items = set()
    if request.method == 'POST':

        email = request.form['email']
        password = request.form['password']

        if email is not None and password is not None:

            if request.form['email'] and request.form["password"]:
                for user in query_db('select * from users'):
                    if email == user['email'] and password == user['password']:
                        global notLoggedIn
                        notLoggedIn = False

    return render_template("index.html", login=notLoggedIn, items=getItems())

@app.route("/notifications")
def notification():
    if request.method == 'GET':
        if request.args.get("message") is not None:
            content = request.args.get("message")
            query_db('INSERT INTO notifications (content, date) VALUES (?, ?)', (content, datetime.now()))

        notifications = query_db('SELECT * FROM notifications')

        return render_template("notifications.html", login=notLoggedIn, items=getItems())

    return {"error": "Forbidden"}, 404


def itemExist(id):
    items = getItems()

    for item in items:
        if id == item["item_id"]:
            return True
```



```python
    for item in items:
        if id == item["item_id"]:
            return True

    return False


@app.route("/addtag", methods=["GET", "POST"])
def addTag():
    if request.method == 'GET':
        if request.args.get("tag_id") is not None:
            global tagUID
            tagUID = request.args.get("tag_id")
            print("TAG UID TYPE: " + str(type(tagUID)))
            return {"id": tagUID}, 302
        print("no tagID")
    return {"error": "Forbidden"}, 403


@app.route("/add", methods= ["GET", "POST"])
def addItem():
    if request.method == 'POST':
        name = request.form["name"]
        location = request.form["location"]

        global addingMode
        addingMode = True


        global tagUID
        while tagUID == "":
            pass


        if (name is not None) and (location is not None) and not itemExist(tagUID):
            if isUnclassified(tagUID):
                classify(tagUID)

            print(name)
            print(location)
            print(tagUID)
            query_db('INSERT INTO items (name, item_id, location, date) VALUES (?, ?, ?, ?)', (name, tagUID, location, datetime.now()))

        tagUID = ""
        addingMode = False
        return redirect("/", code=302)

    else:
```



```
__init__.py × | boot.py .../Screen 4 | boot.py board 6 | index.html | notifications.html | add.html | database.sql | style.css
__init__.py > notification
185
186     return {}, 302
187
188 def getUsers():
189     return query_db('SELECT * FROM users')
190
191 @app.route("/del/<id>")
192 def deleteItems(id):
193
194     if id is not None:
195         query_db('DELETE FROM items where id=?', (str(id),))
196         return redirect("/", code=302)
197     else:
198         return {"error": "No id"}, 400
199
200
201 def isUnclassified(id):
202     getItem = query_db('SELECT * FROM unclassified WHERE item_id=?', (str(id),))
203
204     if len(getItem) > 0:
205         return True
206     else:
207         return False
208
209 def classify(id):
210     query_db('DELETE FROM unclassified where item_id=?', (str(id),))
211
212 def unclassify(id):
213     (item,code) = getItemsWithId(id)
214     query_db('DELETE FROM items where item_id=?', (str(id),))
215
216     getItem = query_db('SELECT * FROM unclassified WHERE item_id=?', (str(id),))
217     if len(getItem) == 0:
218         query_db('INSERT INTO unclassified (name, item_id, date) VALUES (?, ?, ?)', (item["name"], item["item_id"], datetime.now()))
219     else:
220         for it in getItem:
221             if it["item_id"] == id:
222                 query_db('INSERT INTO unclassified (name, item_id, date) VALUES (?, ?, ?)', (item["name"], item["item_id"], datetime.now()))
223
224 @app.teardown_appcontext
225 def close_connection(exception):
226     db = getattr(g, '_database', None)
227     if db is not None:
228         db.close()
229
230 if __name__ == "__main__":
231     app.run(debug=True)
```

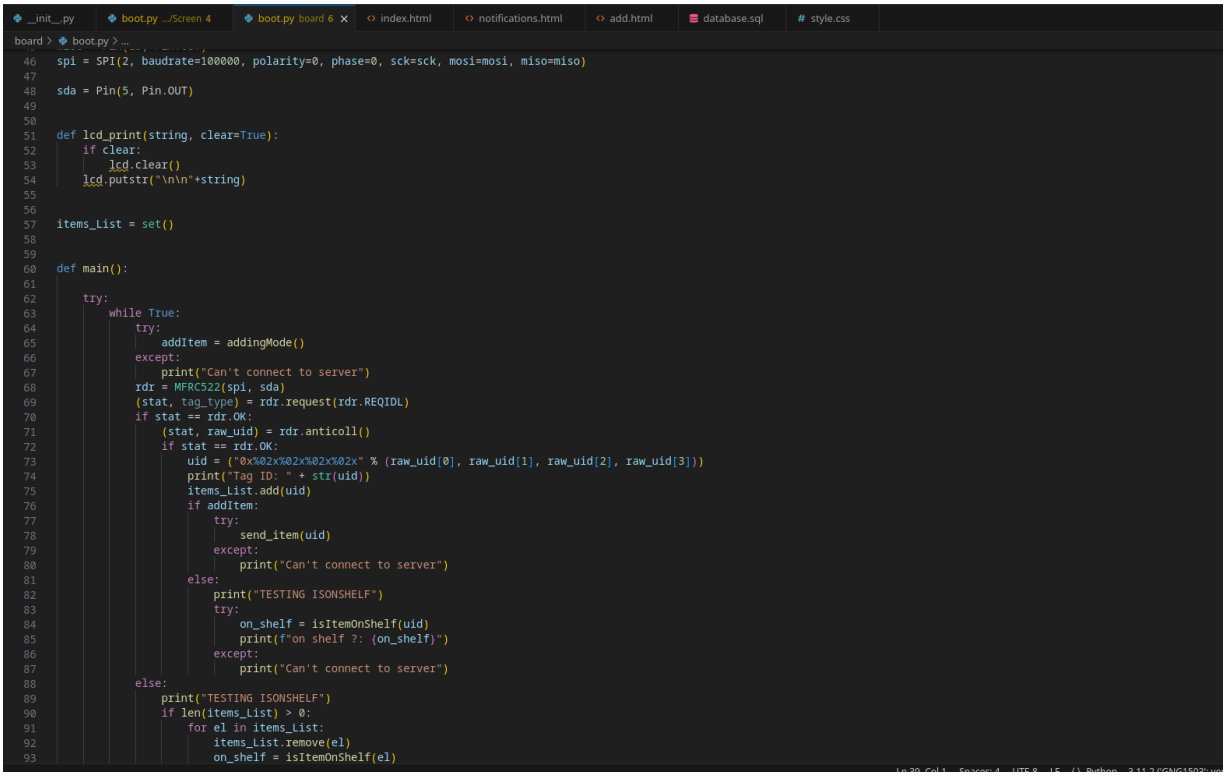


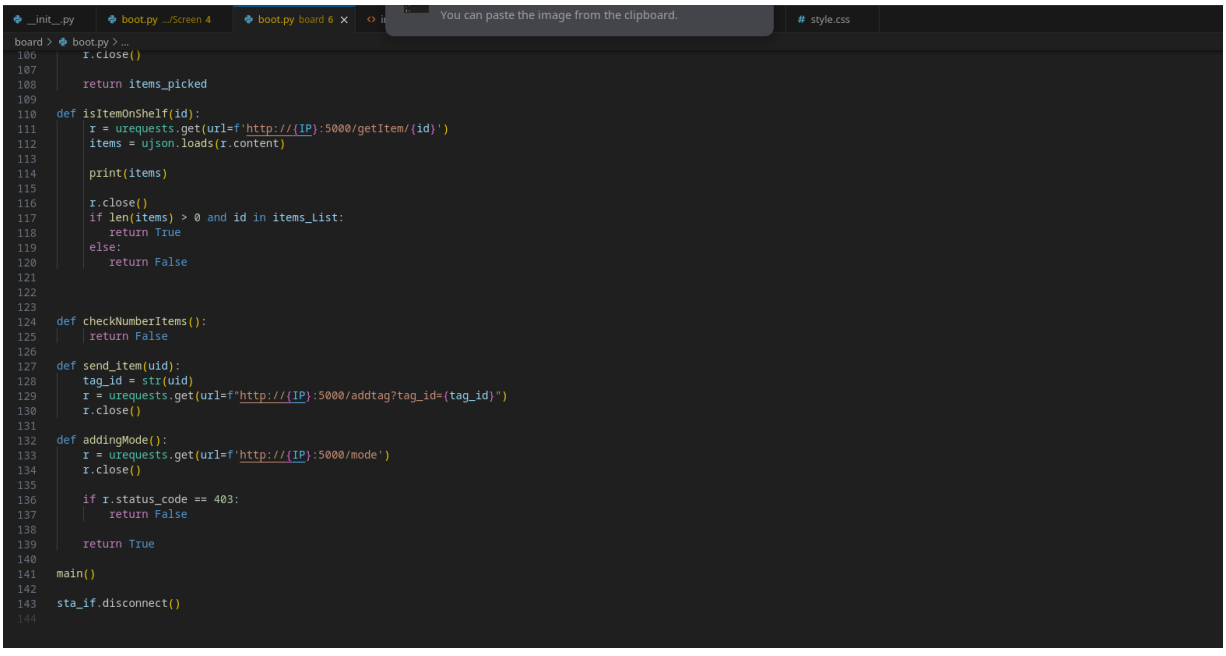
```
__init__.py × | boot.py .../Screen 4 | boot.py board 6 | index.html | notifications.html | add.html | database.sql | style.css
__init__.py > notification
135
136         tagUID = ""
137         addingMode = False
138         return redirect("/", code=302)
139
140     else:
141         return render_template("add.html", type=False)
142
143 @app.route("/adduser", methods=["GET", "POST"])
144 def addUser():
145     if request.method == 'POST':
146             firstname = request.form["firstname"]
147             lastname = request.form["lastname"]
148             email = request.form["email"]
149             password = request.form["password"]
150
151
152             if (firstname is not None) and (lastname is not None) and (email is not None) and (password is not None) and not userExists(email):
153                 query_db('INSERT INTO users (firstname, lastname, email, password, date) VALUES (?, ?, ?, ?, ?)', (firstname, lastname, email, password, datetime.now()))
154                 global notLoggedIn
155                 notLoggedIn = False
156                 return redirect("/", code=302)
157     else:
158         return {"error": "No args"}, 400
159
160 def userExists(email):
161     for user in getUsers():
162         if email == user["email"]:
163             return True
164
165     return False
166
167 @app.route("/mode")
168 def mode():
169     global addingMode
170     if addingMode:
171
172         return "", 200
173
174     return {"error": "Forbidden"}, 403
175
176 def getItems():
177     return query_db('SELECT * FROM items')
178
179 @app.route("/getItem/<id>")
180 def getItemsWithId(id):
181     print(id)
182     item = query_db(f"SELECT * FROM items WHERE item_id=?", (str(id),))
```

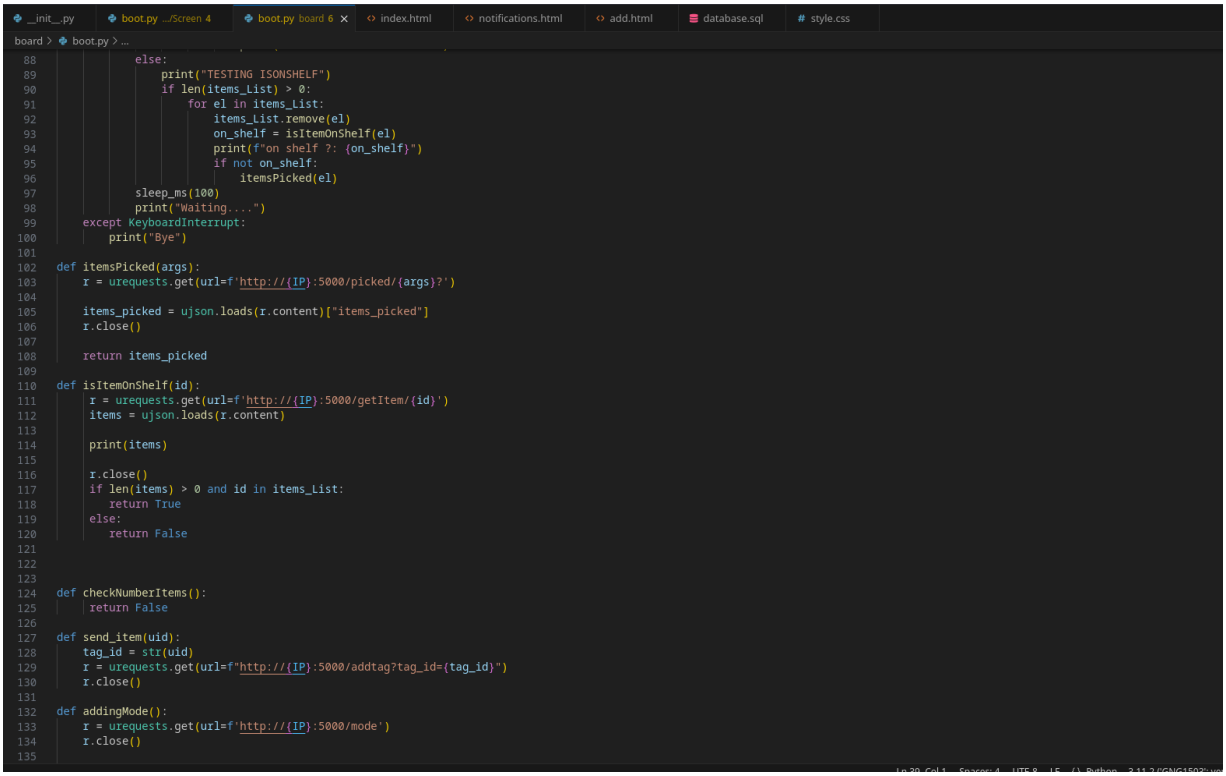
```python
from flask import Flask, render_template, request, g, redirect
from datetime import datetime
import sqlite3

#=====Global=======
addingMode = False
tagUID = ""
items_picked = set()
#==================

app = Flask(__name__)


notLoggedIn = True
DATABASE = 'Database.db'


def make_dicts(cursor, row):
    return dict((cursor.description[idx][0], value)
                for idx, value in enumerate(row))

def get_db():
    db = getattr(g, '_database', None)
    if db is None:
        db = g._database = sqlite3.connect(DATABASE, isolation_level=None)

    db.row_factory = make_dicts
    return db

def query_db(query, args=(), one=False):
    cur = get_db().execute(query, args)
    rv = cur.fetchall()

    cur.close()
    return (rv[0] if rv else None) if one else rv

@app.route("/picked")
def confirmPickUp():
    global items_picked
    if request.args.get("status") is not None:
        if request.args.get("status") == "yes":
            for el in items_picked:
                print("ELEMENT: " + el)
                unclassify(el)
                print("UNCLASSIFIED")
            items_picked = set()
    return {"items_picked": len(items_picked)}, 302

```

RFID code

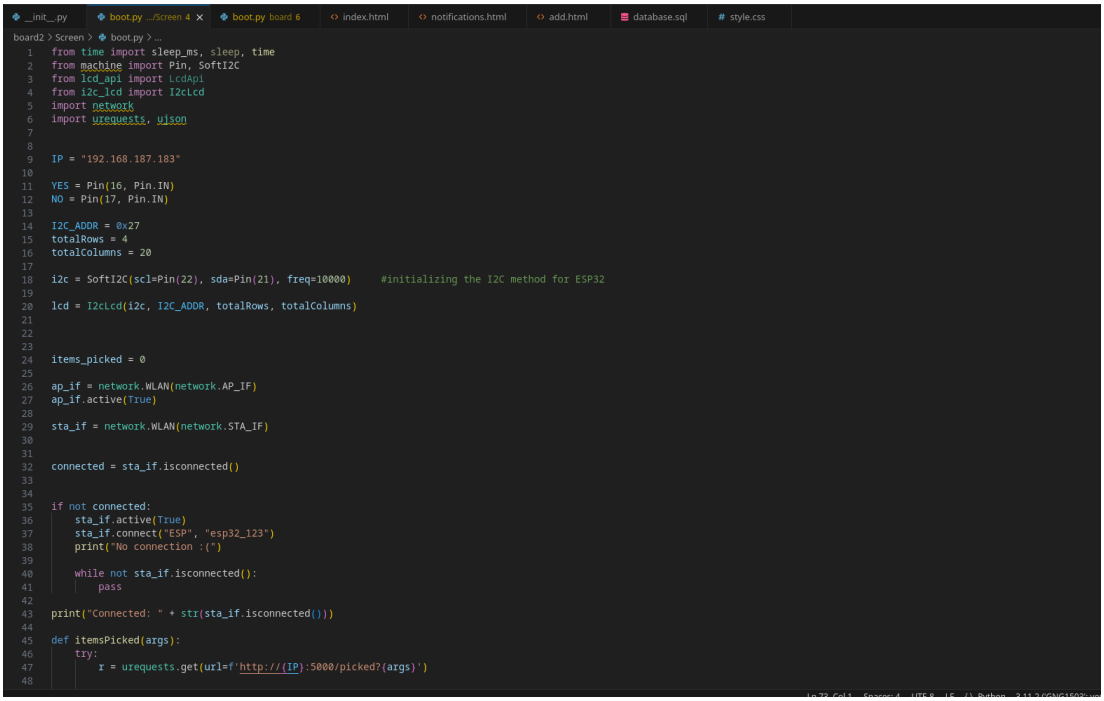
```python
from time import sleep_ms, sleep, time
from machine import Pin, SPI, SoftI2C
from lcd_api import LcdApi
from i2c_lcd import I2cLcd
from mfrc522 import MFRC522
import network
from os import uname
import urequests, ujson


IP = "192.168.187.183"

YES = Pin(16, Pin.IN)


items_picked = 0

ap_if = network.WLAN(network.AP_IF)

ap_if.active(True)



sta_if = network.WLAN(network.STA_IF)


connected = sta_if.isconnected()


if not connected:
    sta_if.active(True)
    sta_if.connect("ESP", "esp32_123")
    print("No connection :(")

    while not sta_if.isconnected():
        pass

print("Connected: " + str(sta_if.isconnected()))

red = Pin(17, Pin.OUT)
green = Pin(4, Pin.OUT)

sck = Pin(18, Pin.OUT)
mosi = Pin(23, Pin.OUT)
miso = Pin(19, Pin.OUT)
spi = SPI(2, baudrate=100000, polarity=0, phase=0, sck=sck, mosi=mosi, miso=miso)

sda = Pin(5, Pin.OUT)
```

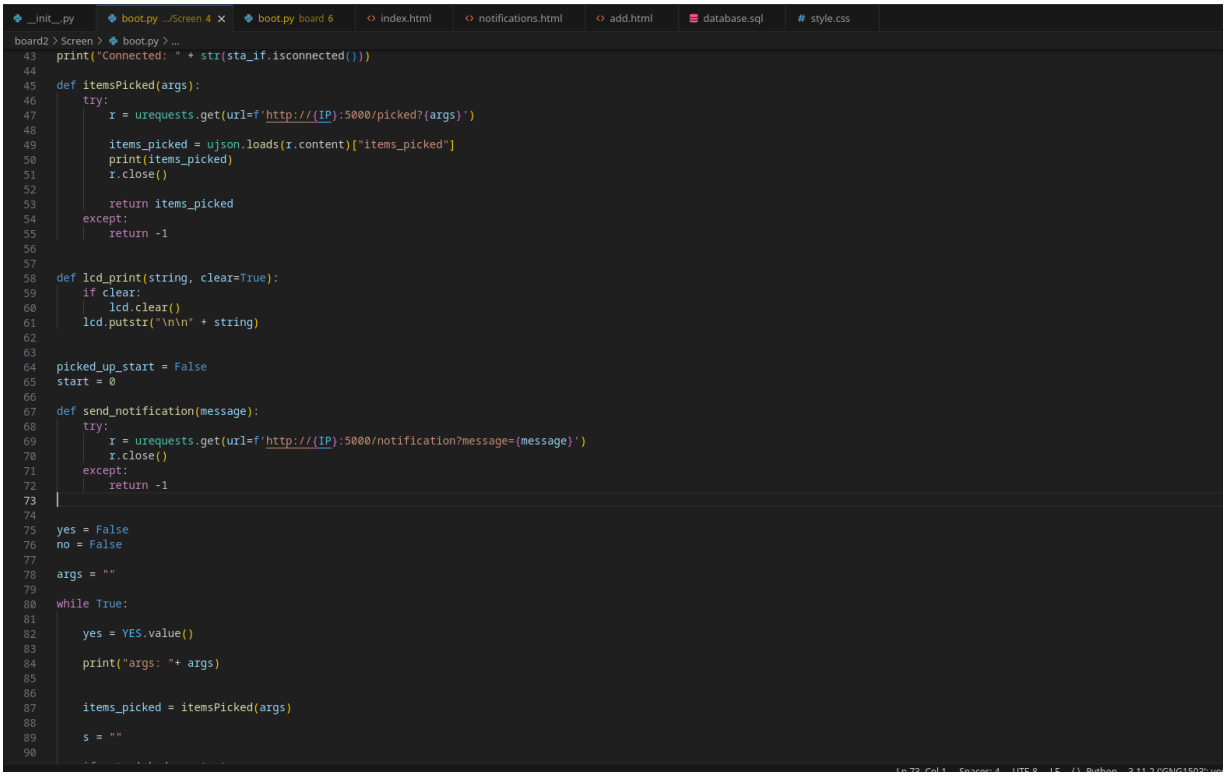

__init__.py
boot.py .../Screen 4
boot.py board 6
index.html
notifications.html
add.html
database.sql
style.css
board > boot.py > ...
spi = SPI(2, baudrate=100000, polarity=0, phase=0, sck=sck, mosi=mosi, miso=miso)
sda = Pin(5, Pin.OUT)
def lcd_print(string, clear=True):
if clear:
lcd.clear()
lcd.putstr("\n\n"+string)
items_List = set()
def main():
try:
while True:
try:
addItem = addingMode()
except:
print("Can't connect to server")
rdr = MFRC522(spi, sda)
(stat, tag_type) = rdr.request(rdr.REQIDL)
if stat == rdr.OK:
(stat, raw_uid) = rdr.anticoll()
if stat == rdr.OK:
uid = ("0x%02x%02x%02x%02x%02x" % (raw_uid[0], raw_uid[1], raw_uid[2], raw_uid[3]))
print("Tag ID: " + str(uid))
items_List.add(uid)
if addItem:
try:
send_item(uid)
except:
print("Can't connect to server")
else:
print("TESTING ISONSHELF")
try:
on_shelf = isItemOnShelf(uid)
print(f"on shelf ?: {on_shelf}")
except:
print("Can't connect to server")
else:
print("TESTING ISONSHELF")
if len(items_List) > 0:
for el in items_List:
items_List.remove(el)
on_shelf = isItemOnShelf(el)


__init__.py
boot.py .../Screen 4
boot.py board 6
You can paste the image from the clipboard.
style.css
board > boot.py > ...
r.close()
return items_picked
def isItemOnShelf(id):
r = urequests.get(url=f'http://{IP}:5000/getItem/{id}')
items = ujson.loads(r.content)
print(items)
r.close()
if len(items) > 0 and id in items_List:
return True
else:
return False
def checkNumberItems():
return False
def send_item(uid):
tag_id = str(uid)
r = urequests.get(url=f"http://{IP}:5000/addtag?tag_id={tag_id}")
r.close()
def addingMode():
r = urequests.get(url=f'http://{IP}:5000/mode')
r.close()
if r.status_code == 403:
return False
return True
main()
sta_if.disconnect()

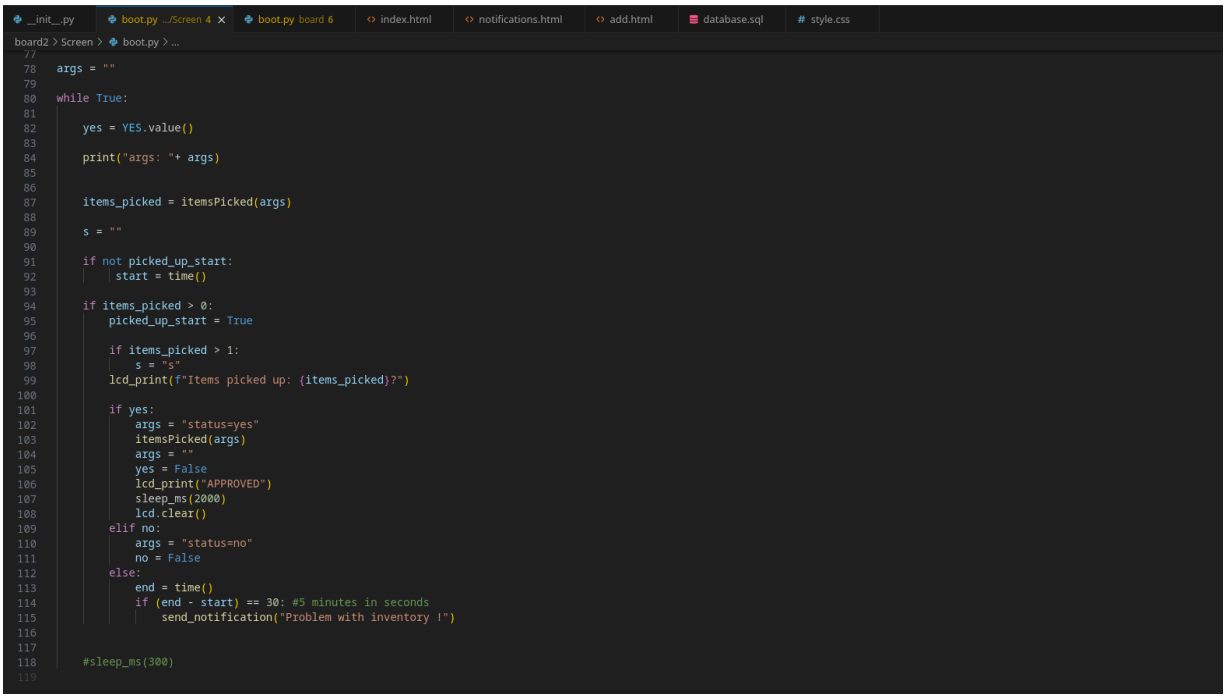


LCD display code



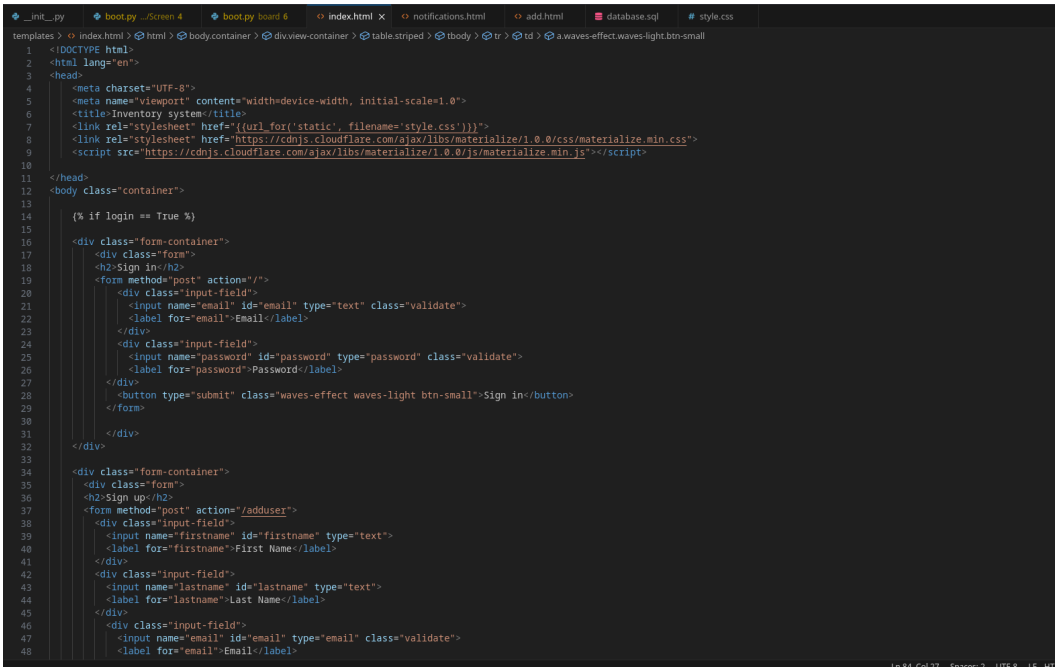


```
__init__.py  boot.py .../Screen 4  boot.py board 6  index.html  notifications.html  add.html  database.sql  style.css
board2 > Screen > boot.py > ...
print("Connected: " + str(sta_if.isconnected()))

def itemsPicked(args):
    try:
        r = urequests.get(url=f'http://{IP}:5000/picked?{args}')

        items_picked = ujson.loads(r.content)["items_picked"]
        print(items_picked)
        r.close()

        return items_picked
    except:
        return -1


def lcd_print(string, clear=True):
    if clear:
        lcd.clear()
    lcd.putstr("\n\n" + string)


picked_up_start = False
start = 0

def send_notification(message):
    try:
        r = urequests.get(url=f'http://{IP}:5000/notification?message={message}')
        r.close()
    except:
        return -1

yes = False
no = False

args = ""

while True:

    yes = YES.value()

    print("args: "+ args)


    items_picked = itemsPicked(args)

    s = ""
```

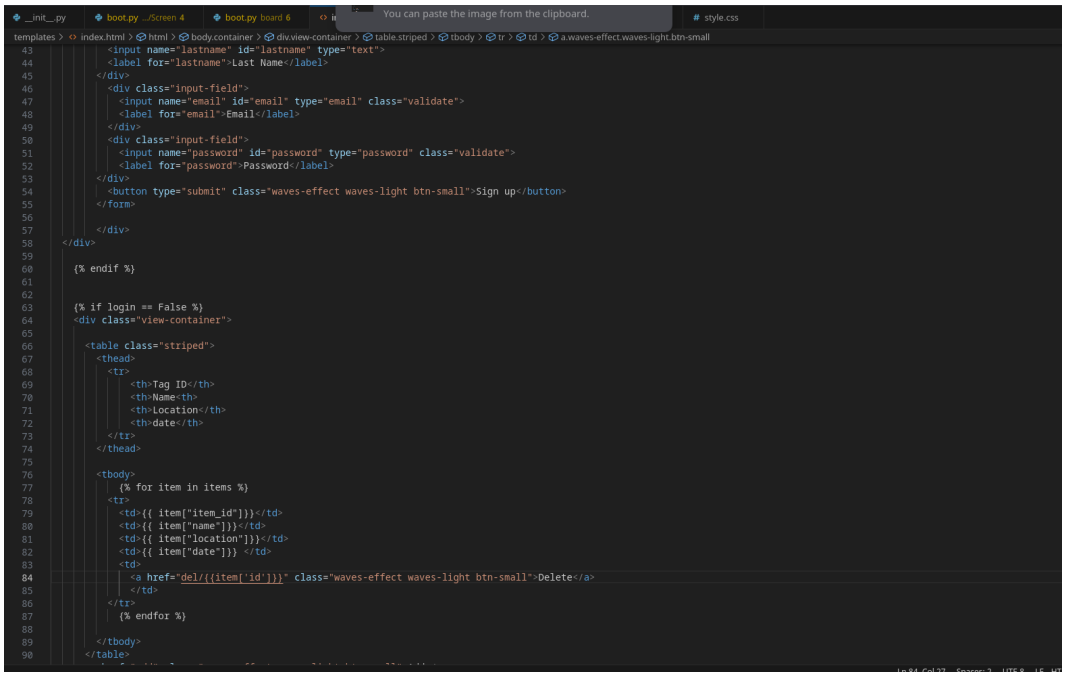


```
__init__.py  boot.py .../Screen 4  boot.py board 6  index.html  notifications.html  add.html  database.sql  style.css
board2 > Screen > boot.py > ...
args = ""

while True:

    yes = YES.value()

    print("args: "+ args)


    items_picked = itemsPicked(args)

    s = ""

    if not picked_up_start:
        start = time()

    if items_picked > 0:
        picked_up_start = True

        if items_picked > 1:
            s = "s"
        lcd_print(f"Items picked up: {items_picked}?")

        if yes:
            args = "status=yes"
            itemsPicked(args)
            args = ""
            yes = False
            lcd_print("APPROVED")
            sleep_ms(2000)
            lcd.clear()
        elif no:
            args = "status=no"
            no = False
        else:
            end = time()
            if (end - start) == 30: #5 minutes in seconds
                send_notification("Problem with inventory !")


    #sleep_ms(300)
```

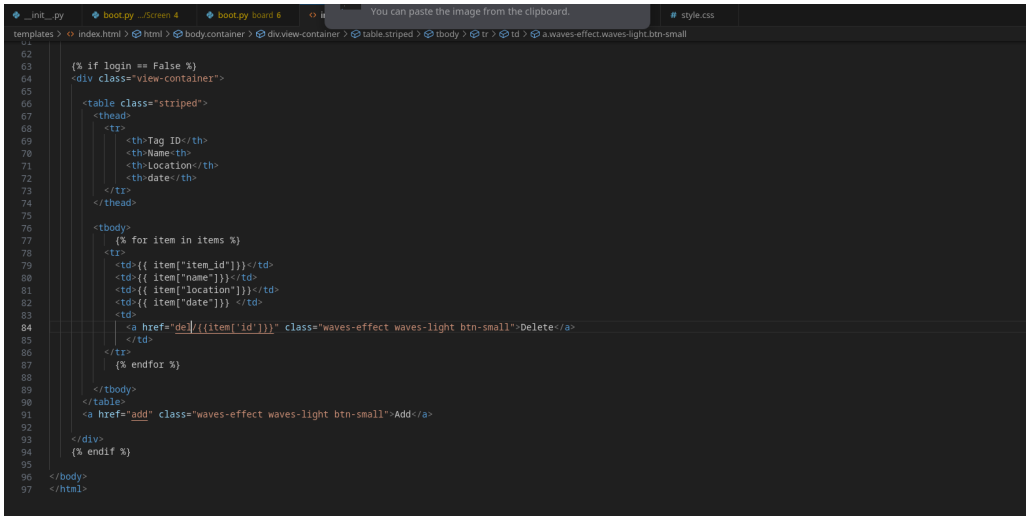
User interface (sign in/out + inventory) code



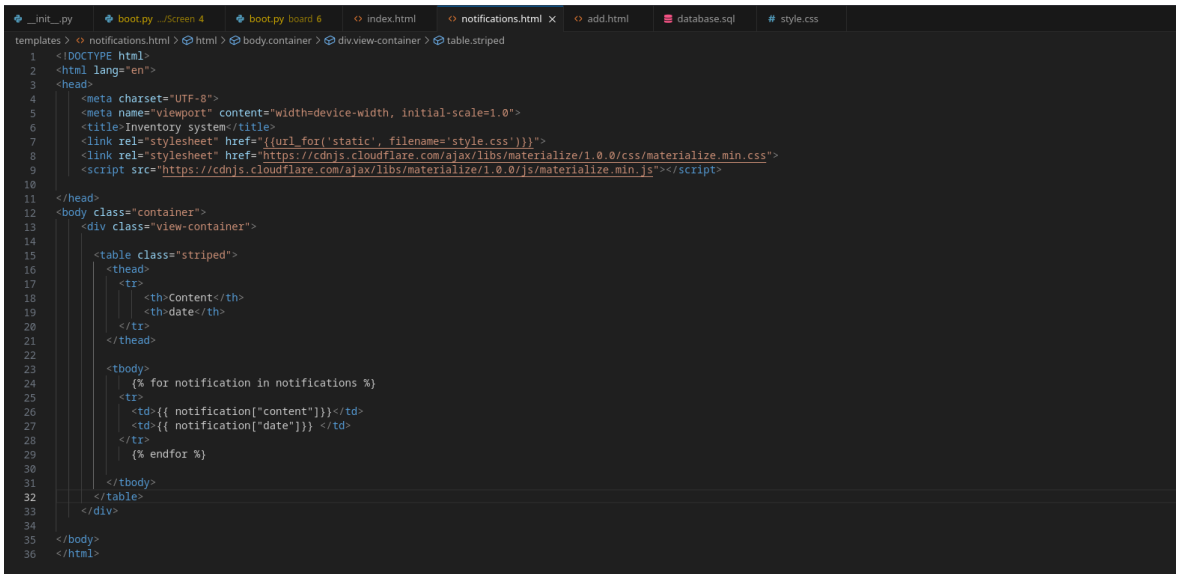
```
templates > index.html > html > body.container > div.view-container > table.striped > tbody > tr > td > a.waves-effect.waves-light.btn-small
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Inventory system</title>
    <link rel="stylesheet" href="{{url_for('static', filename='style.css')}}">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/css/materialize.min.css">
    <script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/js/materialize.min.js"></script>

</head>
<body class="container">

    {% if login == True %}

    <div class="form-container">
        <div class="form">
        <h2>Sign in</h2>
        <form method="post" action="/">
            <div class="input-field">
              <input name="email" id="email" type="text" class="validate">
              <label for="email">Email</label>
            </div>
            <div class="input-field">
              <input name="password" id="password" type="password" class="validate">
              <label for="password">Password</label>
          </div>
            <button type="submit" class="waves-effect waves-light btn-small">Sign in</button>
          </form>

        </div>
    </div>

    <div class="form-container">
      <div class="form">
      <h2>Sign up</h2>
      <form method="post" action="/adduser">
        <div class="input-field">
          <input name="firstname" id="firstname" type="text">
          <label for="firstname">First Name</label>
        </div>
        <div class="input-field">
          <input name="lastname" id="lastname" type="text">
          <label for="lastname">Last Name</label>
        </div>
          <div class="input-field">
            <input name="email" id="email" type="email" class="validate">
            <label for="email">Email</label>
```

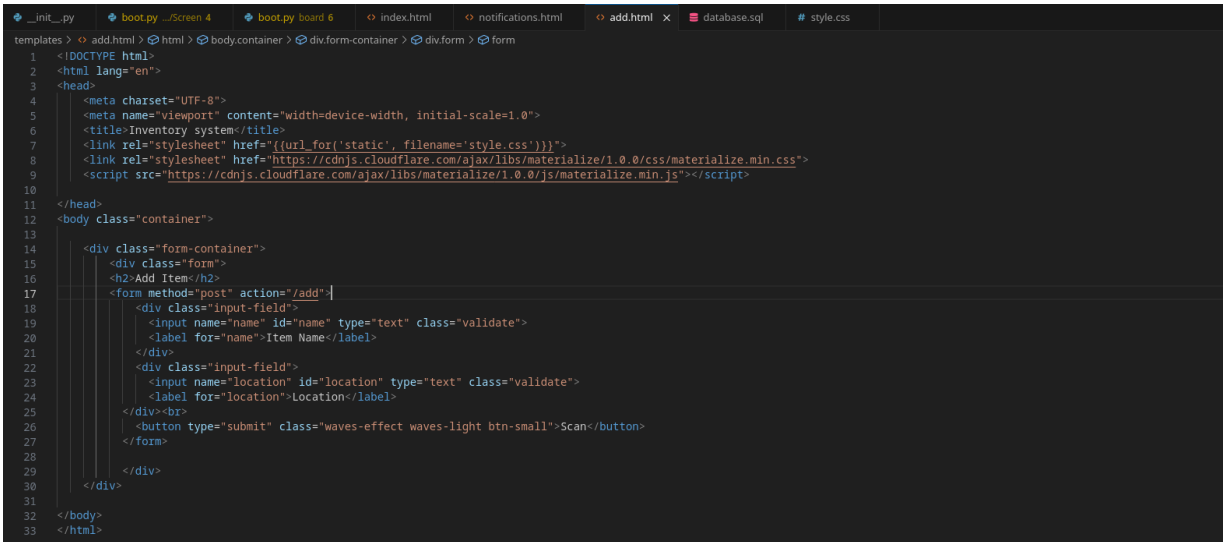


```
templates > index.html > html > body.container > div.view-container > table.striped > tbody > tr > td > a.waves-effect.waves-light.btn-small
            <input name="lastname" id="lastname" type="text">
            <label for="lastname">Last Name</label>
          </div>
            <div class="input-field">
              <input name="email" id="email" type="email" class="validate">
              <label for="email">Email</label>
            </div>
            <div class="input-field">
              <input name="password" id="password" type="password" class="validate">
              <label for="password">Password</label>
          </div>
            <button type="submit" class="waves-effect waves-light btn-small">Sign up</button>
          </form>

        </div>
    </div>

    {% endif %}


    {% if login == False %}
    <div class="view-container">

      <table class="striped">
        <thead>
          <tr>
              <th>Tag ID</th>
              <th>Name<th>
              <th>Location</th>
              <th>date</th>
          </tr>
        </thead>

        <tbody>
            {% for item in items %}
          <tr>
            <td>{{ item["item_id"]}}</td>
            <td>{{ item["name"]}}</td>
            <td>{{ item["location"]}}</td>
            <td>{{ item["date"]}} </td>
            <td>
              <a href="del/{{item['id']}}" class="waves-effect waves-light btn-small">Delete</a>
              </td>
          </tr>
            {% endfor %}

        </tbody>
      </table>
```



Notification + add item code

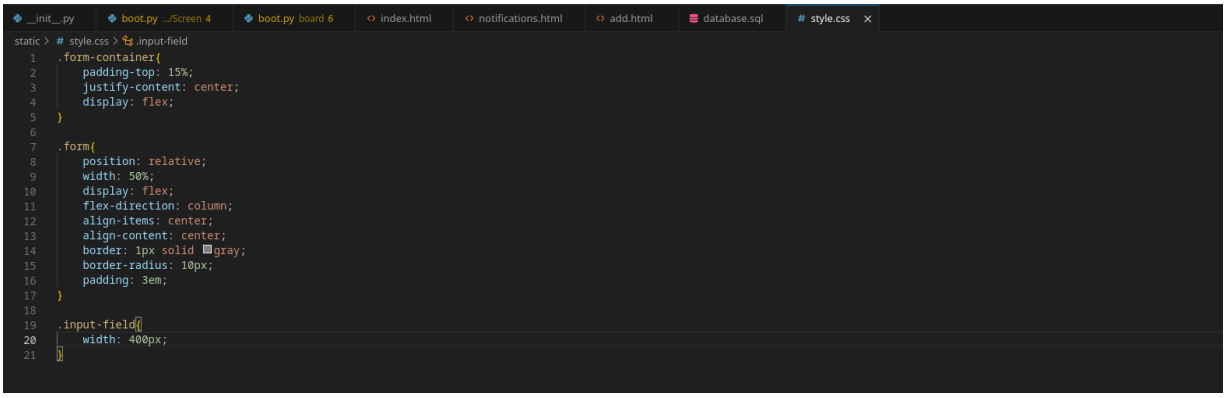




templates > add.html > html > body.container > div.form-container > div.form > form

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Inventory system</title>
    <link rel="stylesheet" href="{{url_for('static', filename='style.css')}}">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/css/materialize.min.css">
    <script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/js/materialize.min.js"></script>

</head>
<body class="container">

    <div class="form-container">
        <div class="form">
        <h2>Add Item</h2>
        <form method="post" action="/add">
            <div class="input-field">
              <input name="name" id="name" type="text" class="validate">
              <label for="name">Item Name</label>
            </div>
            <div class="input-field">
              <input name="location" id="location" type="text" class="validate">
              <label for="location">Location</label>
          </div><br>
            <button type="submit" class="waves-effect waves-light btn-small">Scan</button>
          </form>

          </div>
    </div>

</body>
</html>
```

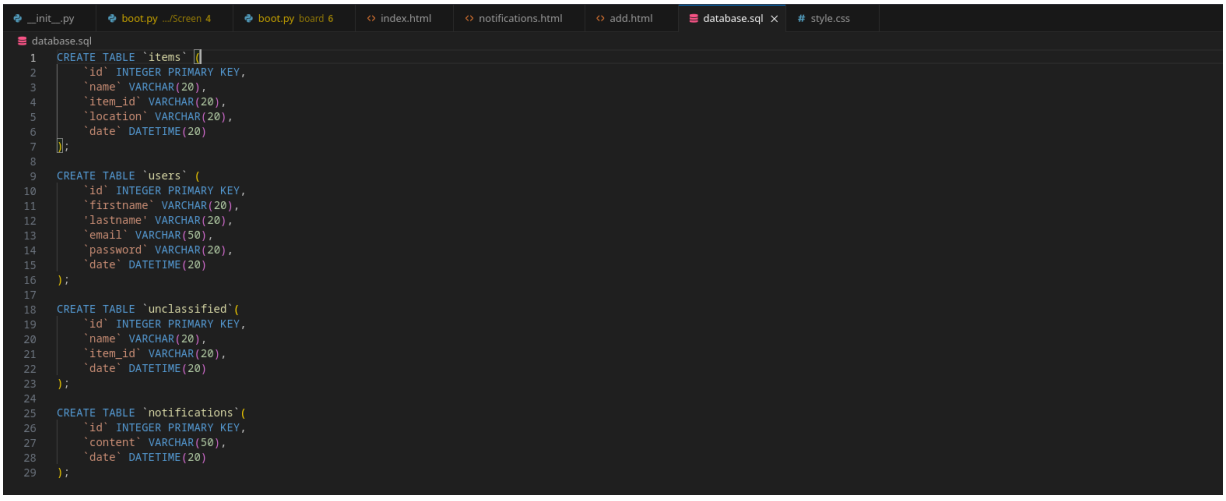
CSS code



static > style.css > .input-field

```
.form-container{
    padding-top: 15%;
    justify-content: center;
    display: flex;
}

.form{
    position: relative;
    width: 50%;
    display: flex;
    flex-direction: column;
    align-items: center;
    align-content: center;
    border: 1px solid gray;
    border-radius: 10px;
    padding: 3em;
}

.input-field{
    width: 400px;
}
```

Database (sql)



database.sql

```
CREATE TABLE `items` (
    `id` INTEGER PRIMARY KEY,
    `name` VARCHAR(20),
    `item_id` VARCHAR(20),
    `location` VARCHAR(20),
    `date` DATETIME(20)
);

CREATE TABLE `users` (
    `id` INTEGER PRIMARY KEY,
    `firstname` VARCHAR(20),
    `lastname` VARCHAR(20),
    `email` VARCHAR(50),
    `password` VARCHAR(20),
    `date` DATETIME(20)
);

CREATE TABLE `unclassified`(
    `id` INTEGER PRIMARY KEY,
    `name` VARCHAR(20),
    `item_id` VARCHAR(20),
    `date` DATETIME(20)
);

CREATE TABLE `notifications`(
    `id` INTEGER PRIMARY KEY,
    `content` VARCHAR(50),
    `date` DATETIME(20)
);
```

6.4 Essais & validation

Plan d'essais de prototypage développé (prototype 3)

Objectif du test (pourquoi)	**Description du Prototype Utilisé et de la Méthode de Test** (quoi)	**Description des Résultats à Documenter et Comment il seront Utilisés** (comment)	**Essais**
Force appliqué vs déformation de la boîte (degré de protection des composantes)	**Prototype utilisé:** Physique complet **Matériaux utilisés:** - MDF (boîtiers) - PLA (boîtier ESP 32) - Carte ESP 32 - Capteur PIR - Écran LCD - LED - Buzzer - Lecteur RFID - Tag RFID	**Information mesuré:** - Déformation verticale et horizontale des boîtiers en fonction de la force appliquée **Documentation:** - Déformation verticale et horizontale mesurées par une règle - Données enregistrés dans un document partagé **Données importantes:** - Déformation verticale (mm - Déformation horizontale (mm) - Charge appliquée (kg)	**a.** déformation verticale et horizontale des boîtiers en fonction de la force appliquée TERMINE : lorsque la déformation et moins de 1 mm pour 20 kg
Usabilité et efficacité des boîtiers		**Information mesuré:** - % d'efficacité lorsque utilisés avec les boîtiers **Documentation:** - % d'efficacité = [(essais réussis)/(essais totales)] x 100 - Données enregistrés dans un document partagé **Données importantes:** - % d'efficacité	**b.** taux d'efficacité du programm TERMINE : lorsque le % d'efficacité est de 100%

Avec notre premier prototype nous avons pu analyser le chemin de chaque fils afin que chaque composantes puisse avoir une tension optimale, et ensuite avec le test de notre deuxième prototype nous avons nous avons pu stimuler le fonctionnement du programme et du processus de la salle d'inventaire, et ainsi pour aboutir à la validation finale nous avons suivis ces démarches comme il est décrit ci dessous

2	**Résultats à mesurer - Taux d'efficacité du programme**
Essai	Essai réussis : écran LCD démontre correctement le nombre d'items ajoutés et enlevés par le calcule des lectures RFID

	Essais réussis	Essais faillites	% d'efficacité = $\frac{essais\ réussis}{essais\ totales} x\ 100$
1	2	3	40 %
2	3	2	60 %
3	1	4	20 %
4	5	5	100 %

3.1	**Résultats à mesurer - Déformation en fonction de la force appliquée**		
Essai	Force appliqué (kg)	Déforation vertical (mm)	Déformation horizontal (mm)
1	20	cassé	cassé
2	20	2	0
N	20	< 1	< 1

Les résultats du dernier essai démontrent que les boîtiers ont la capacité de résister aux forces externes et de protéger les composantes du système.

3.2.a	**Résultats à mesurer - Taux d'efficacité des composantes dans les boîtiers**		
Essai	Items passés	Lectures RFID	% d'efficacité
1	5	4	80 %
N	5	5	100 %

Les résultats du dernier essai démontrent que les boîtiers n'interfèrent pas dans le fonctionnement des composantes. Ainsi, le matériel des boîtiers n'affecte pas l'identification par radiofréquence des lecteurs RFID et les dimensions des boîtiers ne bloquent pas le champ de visibilité du capteur de mouvement. De plus, les boîtiers sont bien conçus car le LED et l'écran sont bien visibles et le son du buzzer n'est pas diminué.

3.2.b	**Résultats à mesurer - Vérification du taux d'efficacité du programme dans les boîtiers**		
Essai	<u>Essai réussis :</u> écran LCD démontre correctement le nombre d'items ajoutés et enlevés par le calcule des lectures RFID		
	Essais réussis	Essais faillites	% d'efficacité = $\frac{essais\ réussis}{essais\ totales} x\ 100$
1	5	5	100 %

7 Conclusions et recommandations pour les travaux futurs

Au cours de notre travail sur le prototype du système VMI, nous avons appris plusieurs leçons cruciales. Tout d'abord, ce projet nous a aidés avec la compréhension nette des besoins et des attentes des clients versus ceux des utilisateurs. De plus, il nous a permis d'apprendre à évaluer nos erreurs et nos incompréhensions au niveau des attentes des clients, pour ensuite aller de l'avant et améliorer notre solution pour qu'elle ne soit pas affectée. Finalement, le travail de ce projet nous à permis de développer notre capacité de gestion du temps et d'établir des partages de tâche égale au sein du notre équipe. Au sujet de travaux futurs, nous recommandons fortement de se prendre à l'avance sur la décision de matériel et de documentation. Ceci donnera plus de temps avec le développement des prototypes et de l'interface.

Avec quelques mois supplémentaires, nous aurions terminé et amélioré l'interface, effectué davantage de tests et ajouté les éléments que nous avons dû abandonner. Ceux cis, dont l'important pour l'amélioration du système, incluent le capteur PIR. Il favorise la précision du système de vérification LCD en déterminant l'instant d'entrée de l'utilisateur, d'où le moment d'initialisation. D'autres composantes auraient pu être utilisées, mais ne respectaient pas notre budget. Ce système peut ainsi être conçu avec des composants plus efficaces et précis.

8 Bibliographie

Librairies utilisées:

LCD screen:

https://peppe8o.com/download/micropython/LCD/lcd_api.py

https://peppe8o.com/download/micropython/LCD/i2c_lcd.py

RFID scanner:
https://github.com/Tasm-Devil/micropython-mfrc522-esp32/blob/master/mfrc522.py

APPENDICES

9 APPENDICE I: Fichiers de conception

Pour plus d'information sur les essais de prototype et pour savoir comment faire la découpe laser.

Table 3. Documents référencés

Nom du document	Emplacement du document et/ou URL	Date d'émission
FA11 - Livrable G	https://docs.google.com/document/d/1k5un09vNdCqZ_MFrPZXvUFgl2ILwpP6oNO_gOvi5gQs/edit?usp=sharing	Nov 5, 2023
FA11 - Livrable H	https://docs.google.com/document/d/1LuTpgIoM2K2kYTbB6DbyAq1Mmd3UUR7CI7HYmbAKNCM/edit?usp=sharing	Nov 13, 2023
Manuel de Lab de Découpe Laser	https://uottawa.brightspace.com/d2l/le/dropbox/382673/276224/DownloadAttachment?fid=15070047	Octobre 2023