README.md - Grip

Technology Lending Library

Team Members: Tareq Alomar, Cody Barrett, Jacob Levy, Nick Volenec

Executive Summary

There is a technology lending library that teachers/instructors in the local area have access to. This is a great asset for the instructors who use it but in its current form is very difficult for both the instructors borrowing and the instructors lending. On the lending side, all the products are manually tracked and requests from borrowers have to be manually responded to. On the borrowers side, a request for a product is sent blindly, with no knowledge of whether or not the requested product will be available in the near future if at all, or whether the request was received by the right individual.

While our initial scope was to implement a system that could complete the item lifecycle, our results are limited to login, cart functionality, and order placement.

Project Goals

Our goals/objectives are as follows:

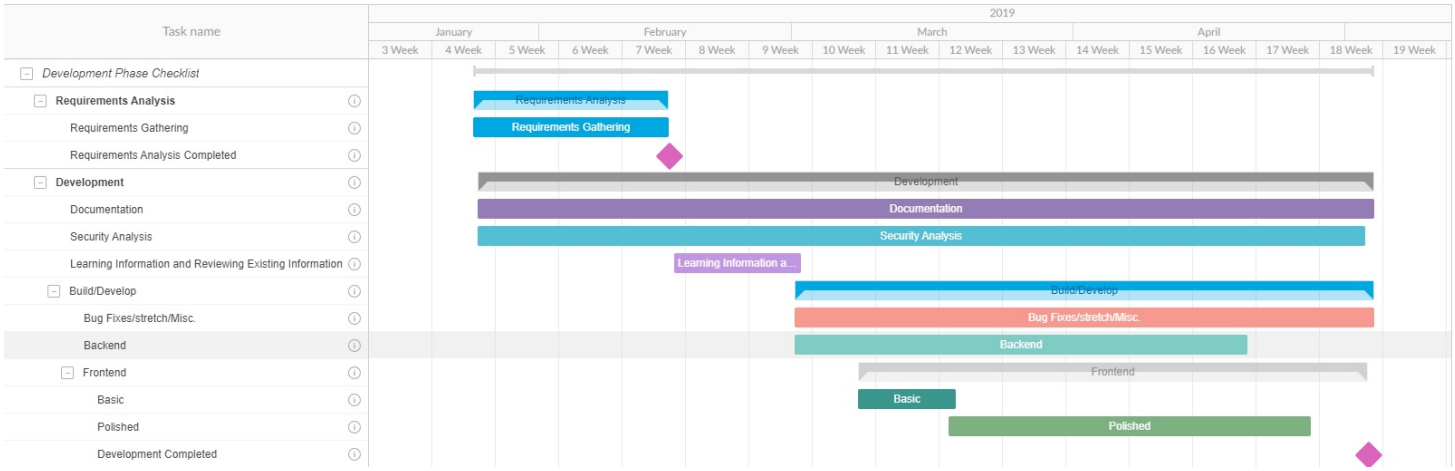
- To automate the request process. *Partially successful, orders may be placed and reviewed through the Django admin panel.*
- To create a web application that will allow instructors to see what products are currently available. *Partially successful, we did not fully meet this scope. However, the admin can see all items in the database via the Django admin panel.*
- To allow lenders to track inventory. *Partially successful, if an order is successfully placed, items allocated to that order will update to "Checked Out" status and will be assigned to the user who placed the order.*

Our stretch goals/objectives are:

- To allow a new user to register via the web app. *Unsuccessful, we did not meet this goal.*
- To allow borrowers to place all items needed for a module into their "cart" in one click. *Successful, we implemented package functionality, and multiple packages can be added to the cart at once.*

Successful completion of this project will allow instructors to more easily lend out and borrow products. This will result in more use of the library by its intended users and thus more effective education practices for the instructors utilizing the system.

Proposed Project Timeline



Risk List

Risk	Impact	Likelihood
Illness of team member(s)	4	4
Lack of technical proficiency	3	4
Hardware Failure	5	2
Inadequate Documentation	4	1
Act of God	8	1

Project Methodology

Literature Review

Keywords: Django Security, Ember.js Security, postgreSQL security, tech lending library, Web App Security, docker security.

Researching for information about the technology lending library, we found that Iowa State University has a Laptop and Equipment checkout which we could use for some ideas. When we looked into it we found a lot of it had to do with checkout times and when the items needed to be returned. Although according to our requirements we do not need to worry as much about lending dates, we were able to keep these ideas in mind moving forward. Otherwise, a lot of the items we found involved security surrounding Django, Ember.js, PostgreSQL, and Docker security including patches, application security features, and how to secure web applications from attacks like SQL injection, Cross Site Scripting, and other common web application attacks.

Technical Plan

Our plan is to develop a system with a PostgreSQL and Django back end and an Ember.js front end, built in a Docker container. PostgreSQL will be used to keep track of the inventory and it's state as well as users including their roles: Borrower, Lender, and Administrator. Django will be used to develop all server-side logic. Our front end will be in Ember.js, and there will be a JSON API which will allow the Django back end to communicate with the Ember.js front end. The interfaces presented will be based on the roles assigned to the user. A borrower will be presented with an interface that allows them to request items. A lender will be presented with an interface that will allow them to fulfill requests from borrowers and also to add new items to the inventory. An administrator will be able to add new borrower and lender users as well as manage inventory. A user may have multiple roles, i.e. a lender could also be an administrator. We will leverage Django’s built-in security features to create an application that is secure. We will test our design and the security of the system on an ongoing basis as the system is developed.

Our development structure was as follows:

- Worked in weekly “Sprints” in which the team would work on items discussed at weekly meetings.
- At weekly meetings we would discuss progress made, issues encountered, and questions that were raised along with allowing for collaborative work to solve problems encountered.

For the second milestone, we focused on building our backend models and serializers.

For the third milestone, we mostly focused on essential frontend functionality. We adjusted the backend as necessary and developed login, cart, and order placement functionality. Time was also allocated towards input validation and user input response.

Results

Milestone 1

- Created a timeline
- Created a risk list
- Created a technical plan
- Created user stories
- Created C4 diagrams
- Deciding what resources will be used, such as Django, Ember.js, etc.

Milestone 2

- Learned the basics of the Django and Ember.js frameworks
- Familiarized ourselves with the existing codebase provided by Dr. Hale
- Created our own development environments
- Implemented our database schema within models.py
- Started connecting Django models to usable views for the frontend
- Started modifying the frontend codebase for use in our application

Milestone 3

Implemented new functionality such that the user may now:

- Login to the website
- Place items into their cart by itemtype and quantity
- Place items into their cart by adding a package containing a group of itemtypes and respective quantities
- Remove individual itemtype-quantities from their cart
- Empty their cart
- Place an order if sufficient items exist

Findings

The following features would be useful for future work:

- Allow new users to register
- Allow admin or lenders to add multiple new items to the database with an automated sequential barcode assignment
- Allow admin or lenders a more usable view of all items than is provided by the Django site administration panel
- Allow users to see the availability of items before placing the order

- View and update orders
- Migrate orders to history model once completed

Issues and lessons learned:

- Ember observers must be used to override lazy-loading. Without them, console.log() provides no useful information.
- Backend includes are useful for gathering related information in a single request.
- Parallelizing work efforts is necessary. We lost some time waiting on login functionality and later discovered that we could implement placeholder data without requiring backend data.
- We failed to implement object-level permissions, which are necessary for a secure application.

Requirements

Resource	Dr. Hale Needed?	Investigating Team Member	Description
Server Space	Yes	Tareq	A domain name provided by Dr. Hale
Docker	No	Nick	open-source application level container framework
Django	No	Tareq/Nick	open-source web framework based on Python
PostgreSQL	No	Tareq	Object-rational database management system for back-end
Ember.JS	No	Tareq	JavaScript framework for front-end

User Stories

Borrower: Title: Be able to request an order from UNO Tech Lending Library. Description: As a teacher in Omaha area, I need be able to request a package (via app), So that I can use UNO's resources. Acceptance Criteria:

- A list of items pop up when I click request.
- When I select an item, I would be able to select the quantity.
- When I select the quantity, I should be able to find the item(s) in my cart to check out.

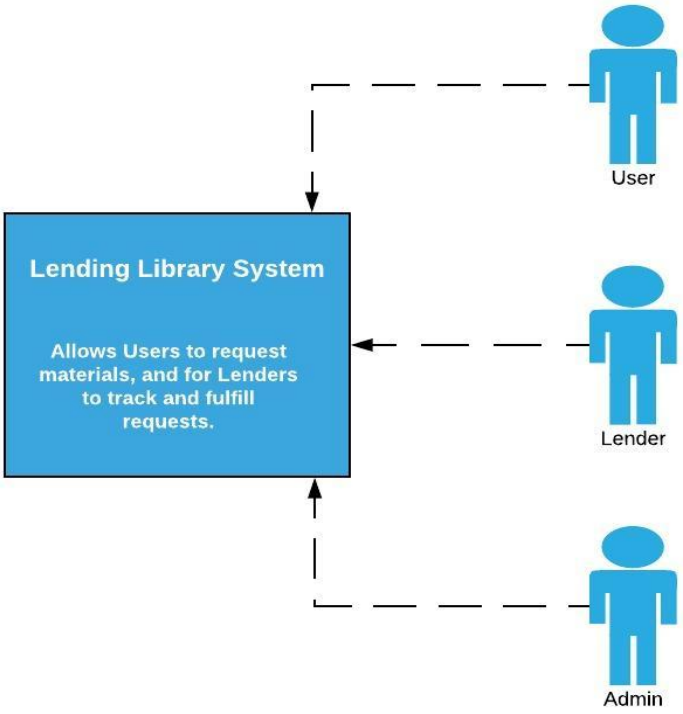
Use Case: request an order

- The use case begins when the user clicks "request"
- The system responds by displaying item.
- The user chooses the item & its quantity.
- The system responds if the item is avaliable to lend them, put it into user's cart. -[Alt1] The user is ready to checkout. -[Alt2] The user wants to continue to order more items.
- The user case ends when the user confirms to checkout all the requests.

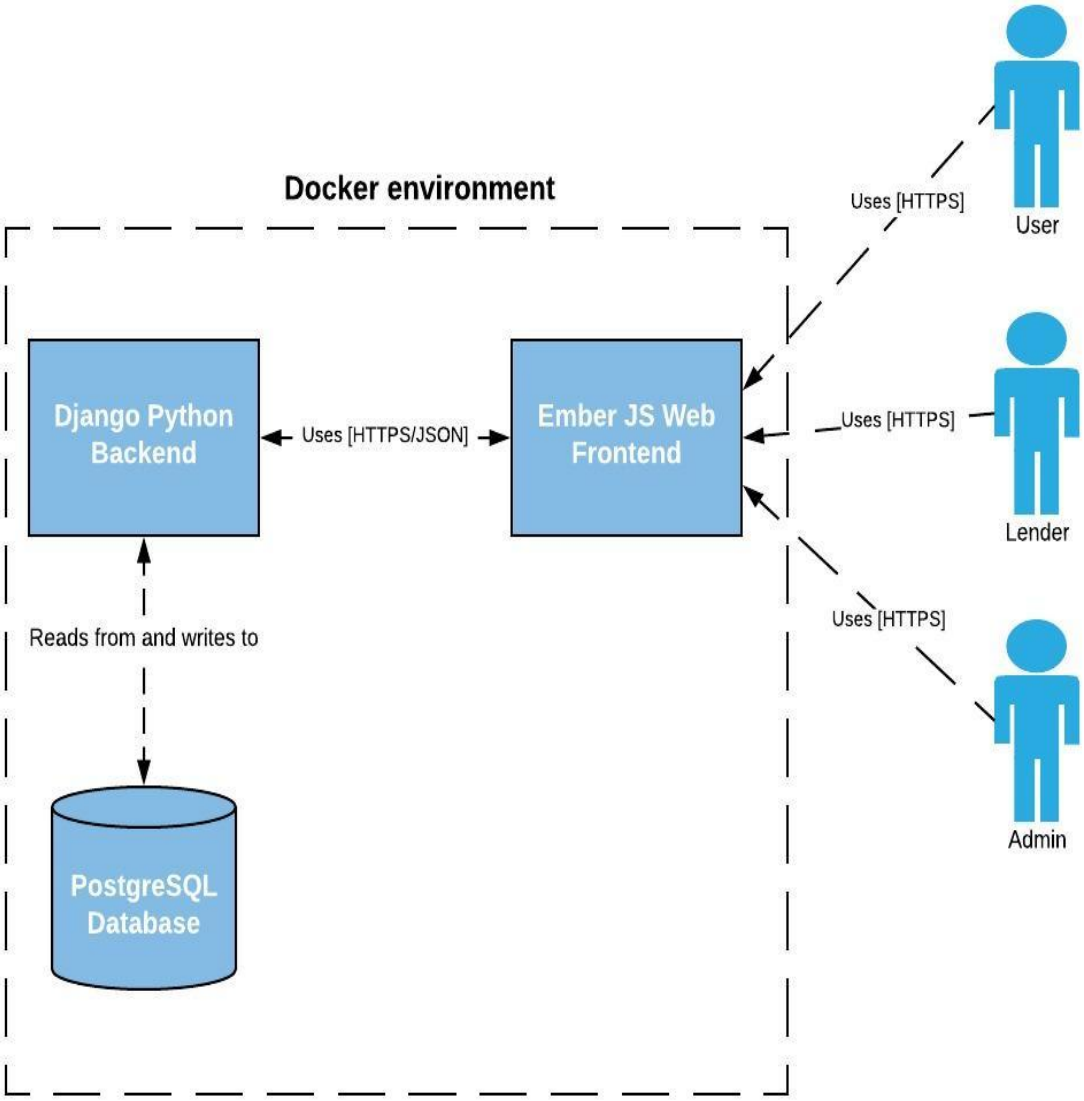
Lender: Title: Be able to track an order that has been checked out by a teacher. Description: As a UNO faculity and staff, I need be able to track a package (via app), so I can keep monitor UNO's resources. Acceptance criteria: -Pending? -arrived? -search by an individual order to track? or see all the orders at once to track? -limitation time for lending?

C4 diagrams

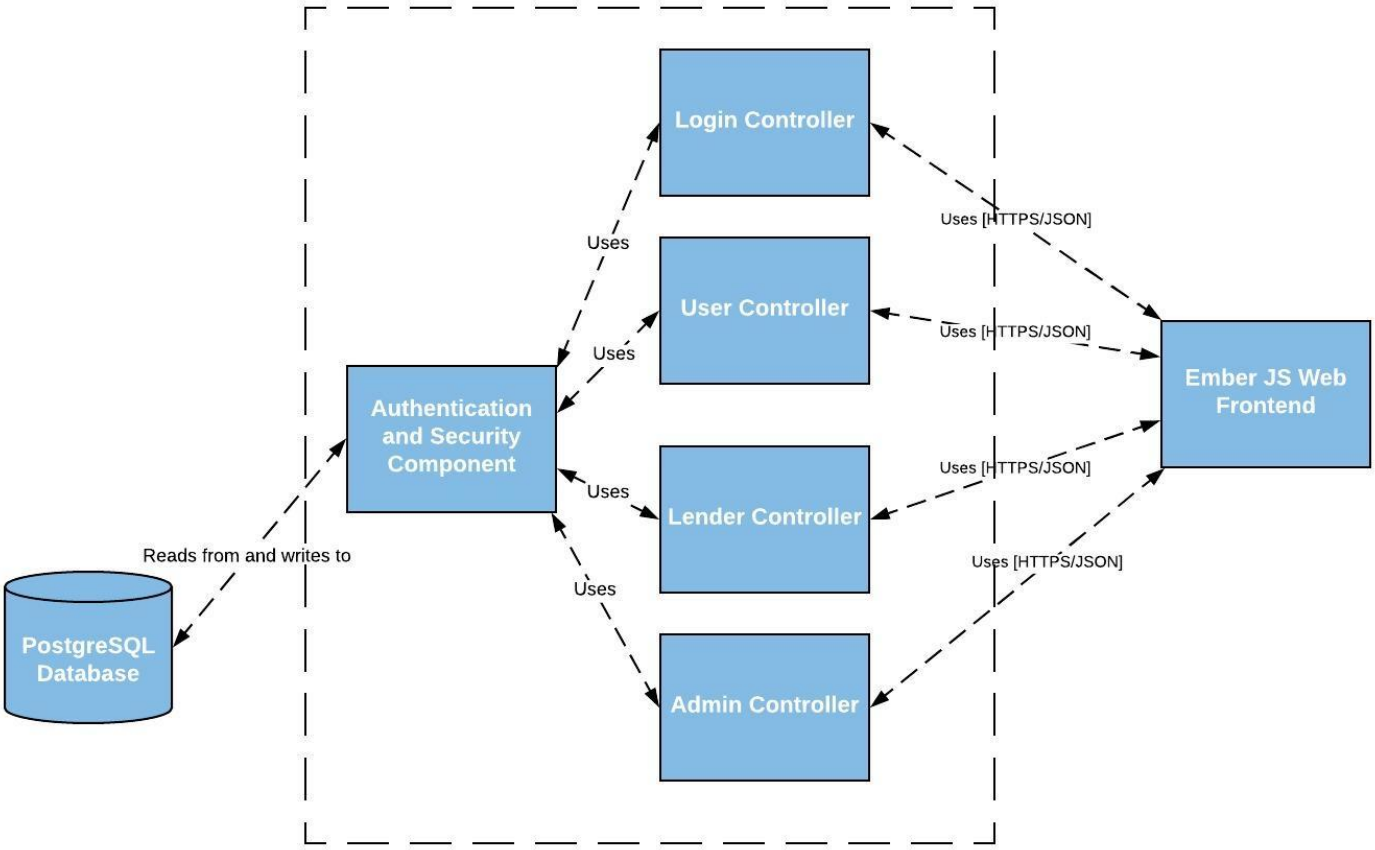
Level 1


User
Lending Library System
Allows Users to request materials, and for Lenders to track and fulfill requests.
Lender
Admin

Level 2


Docker environment
Django Python Backend
Uses [HTTPS/JSON]
Ember JS Web Frontend
Reads from and writes to
PostgreSQL Database
Uses [HTTPS]
User
Uses [HTTPS]
Lender
Uses [HTTPS]
Admin

Level 3



Lending Library App Build Repository - Installation & Startup Instructions

These are the instructions to build the app for production mode. In its current state, the frontend must be built and placed in a folder within the backend, so it has a few dependencies in addition to Docker.

Installation / Getting Started

Requirements

This app is containerized using Docker:

- Docker (https://docs.docker.com/install/)
- (Windows users) Docker Toolbox: https://docs.docker.com/toolbox/ However, the download no longer appears to be available on the docker website. It reads, "Docker Toolbox is now considered Legacy."

Ensure the following is installed to build the frontend:

- node
- npm
- bower
- ember-cli

First Time Installing/Configuring Docker Toolbox (Windows users)

- Install Docker Toolbox
- Run the Docker Quickstart Terminal
- Open VirtualBox after it has finished initializing
- Right click on the VM it created called "default"
- Select Network from the menus on the left of the settings pop-up menu
- Click "advanced" to expand the menu
- Click Port Forwarding
- Add a new rule
- Name it "Lending Library" or something else relevant
- Set the Host Port to 8000 and the Guest Port to 80, and set the Host IP to the desired address. For development purposes we used localhost or "127.0.0.1"

API (Backend) Installation

You need to build the docker image from the provided DockerFile using Docker Compose. To do this:

Acquiring the files

```
git clone --recursive https://github.com/MLHale/lending-library-site-builds.git
cd lending-library-builds
git submodule sync
git submodule update --init --recursive --remote
```

Update lending-library-backend/django_backend/localsettings.py

Change ENVIRONMENT from DEV to PROD, and insert the key for the SECRET_KEY field.

Building and Running the API

While still in lending-library-builds folder:

```
docker-compose build
docker-compose up -d
```

This creates a few docker containers with all of the requisite installed dependencies to run the dev environment. It also initializes the database and starts the containers.

Setup an Admin user and compile C libraries

To create a new admin user for use in the admin portal do the following once the containers are up and running (i.e. `docker-compose up -d` has been run). You can also use "run" in place of "exec" if the container is not running to perform the below command.

```
docker-compose exec django bash
python manage.py createsuperuser
# provide admin credentials
exit
```

Building the Frontend and Placing it into the Backend

from `frontend`, run:

```
npm install
```

Some files may require bower, run the following if the prompt says bower is required: (You may add the -g option to install globally, if desired)

```
npm install bower
```

The following will install dependencies that must be installed by bower:

```
bower install
```

npm may say that it has found vulnerabilities, run the following if it recommends to do so:

```
npm audit fix
```

Finally, build ember and send the output to the backend:

```
ember build --environment production -o ../lending-library-backend/static/ember
```

Getting Started

The database will start out empty, so it must be populated before the app will function. Navigate to `localhost:8000/admin` and login with the credentials you provided in the steps where you created the superuser. The Django site administration portal will allow you to add records as needed. The project scope did not address creation and maintenance of records outside of this portal, so the app does not yet have this functionality.

Visit `localhost:8000` in your host browser when the database is populated to view the app in a usable form.