Group 1- “Astro Pack” Software Project

Table of Contents

[1 Team 2](#_Toc129892510)

[2 Project Brief 2](#_Toc129892511)

[3 Project Requirements 4](#_Toc129892512)

[4 Test Plan (Optional) 5](#_Toc129892513)

[5 Demo Script 5](#_Toc129892514)

[6 Database 6](#_Toc129892515)

[7 Logic Layer 8](#_Toc129892516)

[8 Presentation Layer (Website) 11](#_Toc129892517)

[9 Team Documentation 17](#_Toc129892518)

# Team

|  |  |  |
| --- | --- | --- |
| Project Role | Resource Name(s) | Responsibilities |
| Project Manager | Guillermo Mena | 1. Evaluates the project to ensure it aligns with the requirements. 2. Ensure project meets timeline. 3. Master Data 4. Responsible for EC2 Host. |
| Requirements Manager | Vineela Yetrintala | 1. Coding for backend 2. design frontend 3. Assisting in importing mass data records |
| Design Manager | Christian Lopez | 1. Assists in defining and reviewing test scenarios. 2. Testing 3. Backend 4. Coding |
| Testing Manager | Kristina Litvak | 1. Ensure project meets timeline 2. Checks on available resources 3. Frontend + Backend 4. Debug 5. Design |
| Presentation Manager | Kathie Matta | 1. Design 2. Backend |

# Project Brief

The purpose of this project has been to develop a web-based generator for packing list items that will be implemented to improve the task of packing for trips. The initial rollout started with the simple idea of solving the issue of making sure a person packs all their items when they travel.

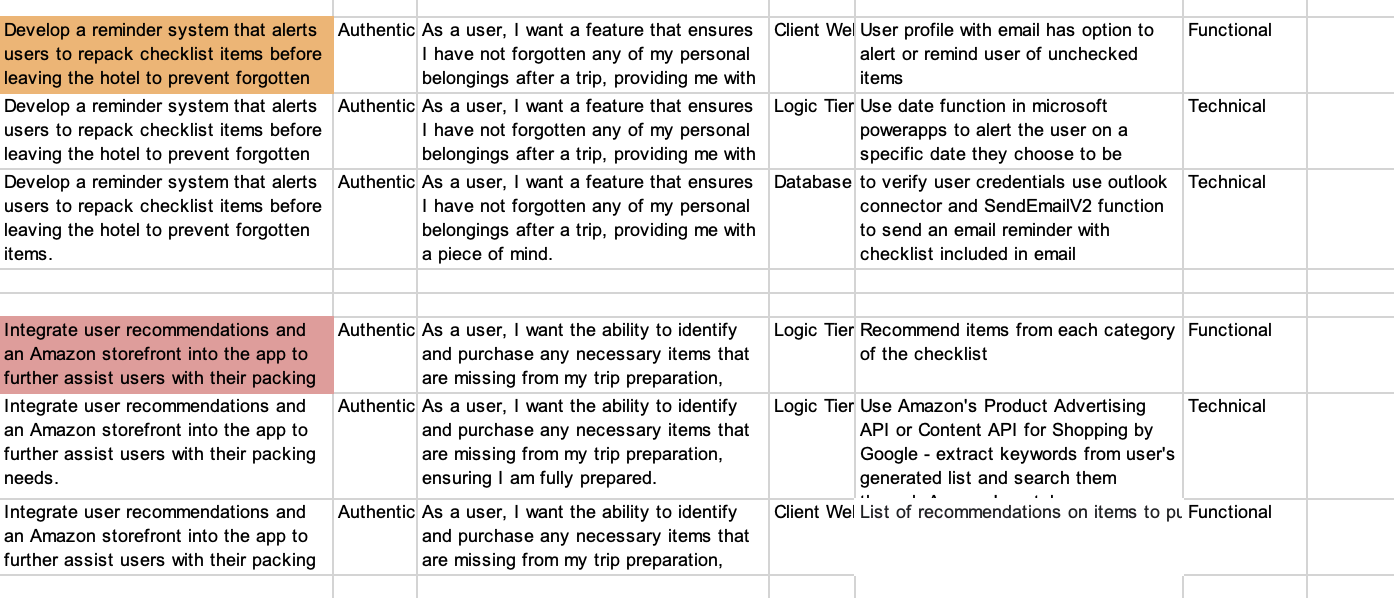
Throughout our research, we came up with numerous app ideas that we realized weren’t super feasible. Our latest research briefs discussed the potential to build apps revolving around ideas such as an outfit generator or an inventory management system. We then decided to mix these two ideas together to create a home improvement app that would allow the user to type in a project they want to work on, and it would return a list of all materials/tools that are necessary for their project along with step-by-step instructions. The goal was to provide a “one-stop-shop” that would give even the most inexperienced homeowners the ability to tackle a home improvement project.

Ultimately, we decided this idea was far too broad and not feasible within the span of six weeks. Continuing with the idea of a generator/fashion/travel/inventory app, we concluded with building an app that generates a packing list for users to bring or purchase based on what destination they are traveling to. This app would take the stress of packing and planning out of the way, so that travelers can focus on the goal of their trip. The app would consider things such as weather, destination, reason of travel, and create a curated list of items that the user needs.

# Project Requirements

A screenshot of a computer

Description automatically generated with medium confidence



# Test Plan (Optional)

Our group briefly touched on and got started on creating unit tests for our application. The unit testing went over each of the following python files: our views, urls, forms, and model. However, having recognized that unit testing had been removed as a requirement for our project, we stopped creating more modifications prior to our final demo. Instead, the only form of testing that our group continued to do is manual testing. What this means, is that we ensured that our website was running properly without Django errors. Furthermore, checked to see if our backend models were working properly in order to actually filter out the items in one’s packing list as intended. Other forms of manual testing include the following: making sure a model keeps a copy of packing list items, ensuring the URLs are set up properly, and a previous packing list is still linked to its items. Although we acknowledge the importance of unit testing, given our limited time and this being optional, we prioritized making sure that our application matched what we set out to create since week five.

# Demo Script

Introduction: Welcome to our website. The purpose of the website is to generate a comprehensive list of must-haves and recommended items for your next adventure.

Astro Pack

User

* Sign Up as new user.
* Login as a user:
  + Show user dashboard.

Navigate to home page:

* Welcome signed in user.

Create Packing new list:

Fill out form:

* Filters apply based on selections.
  + Temp runs to the back-end Google Place Autocomplete API.
  + Run different trip generators.
* Click submit.

Dashboard

Navigate to Saved trips by user:

* Show backend tables of Trip destination and weather.
* List is available.

Navigate to admin portal(link/admin):

* Click on trips.
* Front end to backend communication (values that were filled out in the form now appear in the backend)
* Make note of length of trip feature in backend based on what dates they put in
  + Occasion selections
  + Gender
  + Activities
  + Temperature
* Show tables and how we are working on how to import mass records.
  + Key feature for building the data and user customization in the future.
* Master Data
  + Descriptions and associations for many options when it comes to travel.
  + Key is to give the user more than they would actually pack but the selection is there if needed.
* Requirements
  + Admin portal
    - Customize data for lists.
  + User login
  + Ability to search items to pack for trips based on custom selections from the user.
  + Dashboard - saved previously.

# Database

There will be one master database.

* + This project will use Django and Postgres to manage the database.

**Original Schema**:Diagram

Description automatically generated

**The Final Schema**:

Diagram

Description automatically generated

# Logic Layer

Amazon EC2 Server

* + <http://18.218.150.75/>

1. Documentation of all routes and their HTTP Methods, parameters, and validations.?

|  |  |
| --- | --- |
| Admin | path('admin/', admin.site.urls), |
| Home | path('', home),  path('home/', views.home, name='home'),  path('home', views.logout\_view, name='logout')  **URL:** ‘/’  **Method:** ‘GET’  **Parameters:** None |
| Login | path('login/', views.login\_view, name='login'),  **URL:** ‘/login’  **Method:** ‘POST’  **Parameters:** username, password  Authenticates the user based on the provided username and password. If the authentication is successful, the user is logged in and redirected to the 'dashboard' view. If the authentication fails, an error message is displayed, and the 'login.html' page is rendered again. |
| Register | path('register/', views.register\_view, name='register'),  **URL:** ‘/register’  **Method:** ‘POST’  **Parameters:** username, password1, password2, email |
| Process\_form | path('process\_form/', views.process\_form, name ='process\_form'),  **URL:** ‘/’  **Method:** ‘POST’  **Parameters:** destination, checkin, checkout, occasion, gender, activities |
| Dashboard | path('dashboard/', dashboard, name='dashboard'),  **URL:** ‘/dashboard’  **Method:** ‘GET’  **Parameters:** None |
| Process\_data | path('process\_data/', views.process\_data, name='process\_data'),  **URL:** ‘/’  **Method:** 'POST’  **Parameters:** temp, destination, occasion, trip\_start\_date, trip\_end\_date, gender, temp\_range, activities |
| Items | path('items/', views.items, name='items'),  **URL:** ‘/items’  **Method:** ‘GET’  **Parameters:** None |
| Saved trips | path('saved-trips/', views.saved\_trips, name='saved\_trips'),  **URL:** ‘/saved-trips’  **Method:** ‘GET’  **Parameters:** None |
| Dashboard Items | path('dashboard/items', views.items, name='items'), |
| Saved List | path('saved-list/<int:trip\_id>', views.saved\_list, name='saved\_list'),  **URL:** ‘/saved-list/<trip\_id>’  **Method:** ‘GET’  **Parameters:** trip\_id |

# Presentation Layer (Website)

1. Screenshots of each screen in the application

• Name of each screen  
• Describe the context of each screen  
• What is the user able to accomplish in each screen?

• Consistent design choices (Nav bars, etc.)  
• In which contexts do these components exist and how/why do they change?

|  |  |  |
| --- | --- | --- |
| Item # | Screen Type | Description |
| 1 | Main Screen (new user) | A picture containing circle  Description automatically generated  The user is introduced to our website and is given a brief description of the purpose of our application. In this screen, the user can click on either “Get Started”, “Login”, or “Sign up”. Get started will have the user redirected to the register page. Similarly, Login and Sign Up will redirect the user to their corresponding pages. As seen, our application uses the colors blue and white, with the only existence of other colors from our image. As will be found throughout the entire project, we strived to have our pages come off as “simple”, or “minimalistic”. |
| 2 | Account Login | Login and password required for authentication.  If the user already has an existing account, they can opt to go to the Login page. As found, the page is simple and still has the same design colors as the main page. Furthermore, it follows the principle of remaining minimalistic, as shown with small images that correspond to Username and Password. From here, the user can input their information in their proper fields. Otherwise, they can choose to register a new account via the “Sign Up” on the top right, or via the “Click here to register”. While the “Login” button on the top right isn’t necessary, it was kept there for a consistent look. Lastly, the user can return to the home page by clicking on Astro Pack on the top left. |
| 3 | New User Setup | Alternatively, if the user doesn’t have an account they can go sign up for one. As found, this page resembles closely that of our Login page. They both have the following: Astro Pack, Login, and Sign Up (with similar functionality), a white box in the middle of page with their respective title in blue, text boxes for user info, and a button on the bottom (i.e. Click here to login) to redirect user. The main function of this page is creating a new authenticated user in order for someone to access the rest of our application. |
| 4 | Dashboard | Once the user is finally logged in, they are redirected to the dashboard. The main functionality of this page is having the user put in information regarding their upcoming trip. In the following order the user can put in: their destination (connected with Google Maps Place API), the date of trip (using calendars), an occasion based on drop down options, their gender based on drop down options, and their trip activities via checkboxes. Upon submitting their trip via the submit button, it is sent to the backend to work with the rest of the application.  In this page, the user is introduced to a new design that will be consistent with the everything regarding trips. Still having the white and blue colors, in the lefthand side of the page there is a blue bar with four options. Astro Pack takes the user back to the main page. Current Trip will redirect to Current Trip page that consists of the list of items generated. Saved Trips will redirect user to a page that stores all of their trip locations. Lastly, they can log out. |
| 6 | Current Trip | Once the user clicks current trip, they are redirected to this page that has items generated based on their most recent trip submitted. As found, there are checkboxes in place beside every item. Ideally as the user is packing things, they click the checkbox when they already have it packed, or if they choose not to bring said item. To keep our design consistent, the name of each item category (e.g., Toiletries) is in a blue box with white text. Lastly, in the lefthand side instead of their being a button to go current trip, instead it was swapped with “Create New List” as we are already in the current trip page. |
| 7 | Saved Trips | In the Saved trips page, it stores every trip submitted by the user. As found, it lists the destination, the start date, and the weather. The main thing of interest here is that the user can click “View packing list”, which will redirect the user to a page that shows the stored packing list associated to said trip. |
| 8 | View Packing List (from saved trips) | From the saved trips page, this is what is looks like when they view packing list. As found, the top of the page states the destination of the trip. Below that consists of every item that was previously generated. Similar to the current trip page, the user can check the boxes for each item. |
| 9 | Main Screen (signed in) | If the user is signed in, the main page changes to instead welcome a user. Since the user is assumed to already be familiar with the app (or at the very least they read the app description previously) we removed the application description. Given that the user is signed in, this time the user has immediate access to the dashboard page, can be redirected to create a new list, or could log out. |

# Team Documentation

**Christian Lopez**

In this project, one contribution that I found myself making consistently is managing the Master Branch. This task alone consists of the following: ensuring the master is at a “runnable” state for the EC2, reverting master commits if necessary, and ensuring there is no conflicts left unresolved. Alongside this, I also found myself aiding my group mates with using Git in general. With my experience with Git, I was able to explain the merging process, have Visual Studio be up to date, and help resolve conflicts. Shifting the attention towards the actual application itself, the main task I did was managing the backend. Firstly, I helped set up the models and consistently made edits to them. Secondly, I formatted our Excel sheets in order to import data into the backend with their proper relationships. Thirdly, I got the backend to store a copy of the every item from a user’s packing list to later refer to it in their *Saved Trips*. Lastly regarding the models, I helped with the filtering process based on user’s trip, gender, etc. In terms of the front-end, the only page I set up was the Saved Trip’s packing list, which printed out the copied items for said trip. While this task was no longer required, I believe that it is worth noting that I also got started on creating Unit Tests for our application. Given that it became optional, I didn’t further modify this part of the project prior to the demo. Lastly, the final contribution I did that can go unnoticed is debugging of the application. Truly, a lot of time was spent on debugging to either find or resolve issues in my personal work or even the work of my group mates. Other than that, like the rest of my groupmates, the rest of the time I’ve spent was researching these new topics/tools used in this application.

**Kristina Litvak**

Throughout this project, I had a ton of fun taking on various roles of development. To start off with, I built the frontend for our site – I tried my best to give it a nice, clean design despite the fact that I’m definitely not a designer. While creating the home page, I knew it needed some kind of image to liven it up, and while searching stock photos, I came across the little astronaut cartoon and thought it was too cute not to include. Initially, it was just a filler while I was working on it, but oddly enough that little cartoon became the inspiration behind most of our design, and obviously the name, Astro Pack.

Along with the home page, I worked on creating a login system and forms for our project using the built-in Django authentication system and Django forms. Django and Python was something I’ve never worked with before, but I was pleased to find out how it truly is a “batteries included” framework. Since our project’s entire premise was to generate a packing list for a user, I created the generator form seen on the user dashboard to take in user input. The more we dove into this project, the more convoluted it seemed. We had to consider a lot of different factors regarding a user’s trip such as destination, travel dates, weather, occasion, gender, and activities. The destination field was built using the Google Place Autocomplete API, which is part of the Google Maps Platform. This made it a lot easier to have a drop-down menu that the user can choose from as they start typing in their destination. This API also returns longitude and latitude coordinates which are then plugged in to the OpenWeatherMap Weather API, which returns weather information for that particular location. This temperature was then used to filter out the correct packing list items for the user (cold, hot, warm). The functionality for the generator form was done primarily with JavaScript, using fetch API and an AJAX request to send the data over to the backend server. Once the data was received on the backend, it was stored in a model called Trip.

Now that we had the user input data in the backend, we needed to figure out a way to actually generate the packing list. Luckily, my teammate introduced me to Django filter and our team worked on filtering out the different items from our models such as Clothing, Accessories, Toiletries, Electronics, etc. All the filtered-out items were then stored in a Python dictionary and added to another model called Generated\_list. The list of items was displayed on the page once the user submits the form, and the Generated\_list model was used for our “Saved Trips” page. This was surprisingly the part we struggled with the most – creating the ability for the logged in user to access all their previous trips and the packing list that was associated with that trip. I ultimately was able to link the unique id for a specific trip and pass it as an argument in the URL inside of the Django template tag, which finally created the proper relationship between a trip and the generated list.

Overall, this project allowed me to gain valuable experience with Python, Django, HTML, CSS, JavaScript, Postgres, and Git. I learned a lot about how to connect frontend with backend and how database models work. Although there were a lot of requirements that I wish we could’ve gotten to work on, I’m proud with what our team has accomplished in a limited amount of time.

**Guillermo Mena**

This Project that we created, Astro Pack, seemed feasible in the idea that we just need to display a list of items from a query. My strengths have not been in coding and knowing I was one of the few IS majors seemed overwhelming at first. All the group members took on many different roles throughout the project to create Astro Pack. The tasks that were performed stretched from research and learning, project coordination, timelines, deliverables, and figuring out issues to get the software working. Breaking down the project into simpler tasks was an easy part, but knowing what to research became more difficult as the project progressed. I applied myself to learn multiple applications, Python coding, EC2, migrations, and database troubleshooting. The collaboration with GitHub was a key to understanding what changes we submitted and managing your branch. This led into the core of pulling all the work into our EC2 instance from GitHub and migrating all the changes to the PostgreSQL database.

My learning objectives from this project came from a variety of tools that I had to learn. Starting with Django and Postgres which were practiced in the homework assignments. This led to continuation of PostgreSQL and migration troubleshooting. Amazon EC2 front end user settings was complicated, but once you figured out not to turn it off or have multiple instances, we were great. As the project grew with each objective met, we experienced new bugs with the database. Another key point to mention is as a group we all had to ensure our local developments were up to date and working in order to push any changes. Whether it is best practice or not, we found out how to deploy new databases as our last resort and then added the Django Admin Import-Export tool. The import tool in the backend became a major part for the final demo when we had to add/edit 300+ of records to utilize the filters correctly.

This project was successful because each one the group members presented their abilities to solve problems while working together. Overall, I have acquired skills to implement a software project with success. This is due to a team that initiated high expectations and a work ethic that allowed us to accomplish our Astro Pack Website.

**Vineela Yetrintala**

The way I contributed to this project was initially to help us come up with ideas on what we wanted our project to be and the requirements for how we would accomplish our goal. Once we had figured out an idea, I worked on creating the login and signup forms. I had to learn the design aspect in order to create something that is user friendly as well as aesthetically pleasing. I also worked on creating the generated packing list that is viewed when the user logs into their dashboard. HTML and CSS was a little bit of a learning curve since I have never worked with those that much previously, but once I figured out how to simultaneously work with them together it was not that difficult to figure out. I also started working on being able to import mass records of data into our Django tables that we created. There were various ways to do it, but we ended up figuring out that we can create an import button to import csv files with all our data in them.

This project allowed me to expand my knowledge in what it takes to build and design a web application. I learned all the different resources to use in order to figure out the problems we were facing as well as how to have back-end to front-end communication. It was also a good learning curve to have a set deadline and work with a team to get such a big project done.

**Kathie Matta**

This project was not the easiest for me. It made me realize that being a Computer Science major is a constant learning path. A lot of research came along with this project. It was my second time creating a website, but it was the first time that I was helping out with the backend of it. It was also my first real experience with Django and amazon services like EC2. Previously, I had developed my first project in an internship during the summer, but it was not the same experience because I was part of the front-end team, and I did most of the CSS and html. We also used flask and Google Cloud services to deploy the site. However, as previously stated, using Amazon Web Services and Django was definitely super new to me. All I knew was what we learned from the homework assignments.

Luckily, I had my team members by my side throughout this whole experience. I can vouch that we all worked well together which is why we created such a great project. At first, we split into teams to work on tasks and that wasn't really working out for us so then, we decided to split tasks and choose tasks that we would want to tackle. Most of the tasks I chose I had no experience in and were challenging but that's what I wanted for myself. I wanted to learn new things. My most challenging task was having to attach a saved list to the individual trip then generate the saved list so that the user could see their list history. After countless hours of research, coding, and debugging, I asked my team members to help with the task. They also spent hours debugging and researching until one of us finally got it to work. Everything we were researching and trying was crashing our site or not generating anything which was so frustrating. We were able to get it up at the last minute. This was actually a task that as a team we had agreed to give up on, but a member of our team had the dedication to not give up on it and I'm glad they didn’t.

Putting aside the challenging tasks of this assignment, I’d say one of the easiest tasks that I had to do for this assignment was displaying the trips to its own page and having the submit button redirect to a packing list instead of displaying weather. At first this task was hard for me. I had decided to tackle this task at around week 6 or 7 but decided to put it aside for a while to focus on more important tasks then towards week 9 I went back to it. I was able to do it by tracing the code carefully and inserting/deleting code where needed.