gp-food-basket: Complete Team Development Guide

This guide provides all instructions for setup, version control, development, testing, and deployment of the **gp-food-basket** Shiny application. Every team member is encouraged to follow these steps exactly to ensure consistent, collaborative progress.

The **gp-food-basket** project is a modular R Shiny dashboard for analyzing and visualizing U.S. food insecurity data. It is structured for team-based collaboration using Git branches. Each contributor works on their own feature branch and merges tested code into the development (`dev`) branch before final deployment to `main`.

1. Environment Setup

1.1 Install Prerequisites

- R version 4.3 or later
- RStudio or Positron IDE
- Git installed and configured with your GitHub account

1.2 Clone the Repository

Open a terminal and run:

git clone https://github.com//gp-food-basket.git cd gp-food-basket

1.3 Open the Project

Open `gp-food-basket.Rproj` in RStudio or Positron. Ensure the working directory points to the project root (the same folder that contains `app.R`).

1.4 Install Dependencies

Run this in the R console:

source("global.R")

This loads and installs all required packages: `shiny`, `plotly`, `leaflet`, `tidyverse`, `DT`, and `bslib`.

2. Project Structure

```
gp-food-basket/
  app.R                    # Main app entry point
  global.R                 # Loads packages and global settings
  R/                       # Modular components
      ui_overview.R
      server_overview.R
      ui_exploration.R
      server_exploration.R
      ui_analysis.R
      server_analysis.R
      helpers.R
      load_all_modules.R
  data/                    # Clean or sample data
  data_raw/                # Original unprocessed data
  www/                     # Static assets (images, CSS, icons)
  vignette/                # Documentation and notes
  README.md
  .gitignore
  gp-food-basket.Rproj
```

Each module contains two files — one for the user interface (`ui_*.R`) and one for server logic (`server_*.R`). Shared functions live in `helpers.R`. The main `app.R` loads and runs everything.

3. Git Branch Structure

Branch	Description
main	Stable, production-ready version
dev	Integration branch for testing new features

Branch	Description
feature/exploration-visuals	Map and trend visualizations
feature/analysis-correlation	Regression and correlation modeling
feature/data-cleaning-ui	Data upload and transformation
feature/modeling-validation	Model validation and diagnostics
project_plan	Documentation and planning materials

4. Daily Development Workflow

Step 1: Update your local repository

git checkout dev git pull origin dev

Step 2: Switch to your assigned branch

git checkout feature/exploration-visuals

Step 3: Edit your files

Work only on your assigned section:

- **Overview Tab** – edit `R/ui_overview.R` and `R/server_overview.R`
- **Exploration Tab** – edit `R/ui_exploration.R` and `R/server_exploration.R`
- **Analysis Tab** – edit `R/ui_analysis.R` and `R/server_analysis.R`
- **Data Cleaning** – edit `R/helpers.R`, `data/`, or `data_raw/`
- **Documentation** – update `vignette/` or `project_plan/`

Step 4: Run and test the app

In R:

```
shiny::runApp(".")
```

The app must start with no errors, and your tab should work as expected.

Step 5: Commit your work

git add R/ui_exploration.R R/server_exploration.R git commit -m “Added leaflet map and summary table to Exploration tab”

Step 6: Push to GitHub

git push origin feature/exploration-visuals

Step 7: Open a Pull Request

1. Open GitHub in your browser.
2. Click **"Compare & Pull Request"**.
3. Create a PR into **dev**, never directly into **main**.
4. Add a clear title and short description of your changes.
5. Request review from the Project teammate.
6. After testing, the reviewer merges into `dev`.
7. Once `dev` is stable, it is merged into `main` for deployment.

5. Coding Standards

5.1 General Rules

- Use `package::function()` syntax: plotly::plotlyOutput("chart_id") leaflet::renderLeaflet({…}) DT::DTOutput("table_id")
- Keep **UI** and **Server** logic separate.
- Avoid hard-coded paths — use `here::here("data", "file.csv")`.
- Write **meaningful commit messages**: Added regression results to Analysis tab

5.2 Testing Before Commit

Always confirm the app runs successfully: `shiny::runApp(".")`

Never commit raw or large files; keep them under `data_raw/` (ignored by Git).

6. .gitignore Rules

Add this at the project root:

.Rhistory .RData .Ruserdata .Rproj.user/ rsconnect/ data/*.rds data/*.csv data_raw/ *.DS_Store

This prevents temporary or large files from being tracked in Git.

7. Troubleshooting Common Issues

Problem	Cause	Solution
could not find function "plotlyOutput"	Function not namespaced	Use `plotly::plotlyOutput()`
App won't start	Missing comma or parenthesis	Check syntax in `ui_*.R`
Shiny won't stop	App running in terminal	Press **Ctrl + C** or click **Stop**
Merge conflict	Branch outdated	Run `git pull origin dev --rebase`
Missing packages	New environment	Re-run `source("global.R")`

8. Testing and Code Review

Before submitting any Pull Request:

1. Run the app locally with `shiny::runApp(".")`.
2. Confirm your tab loads and updates correctly.
3. Verify no console errors appear.
4. Commit only tested, clean code.
5. Request a peer review before merging.

9. Deployment Procedure

When the app is stable on the `main` branch:

rsconnect::deployApp(appDir = ".", appName = "gp-food-basket")

You may deploy via:

- **ShinyApps.io**
- **Posit Connect**
- **Internal servers**

Ensure all dependencies are declared in `global.R`.

10. Team Roles

Team Member	Course Level	Primary Responsibilities
Conrad Linus Muhirwe	DATA-613	• Repository setup, version control & documentation. • Statistical modeling, validation, and interpretation.
Sharon Wanyana	DATA-613	Correlation & regression analysis, hypothesis testing.
Ryann Tompkins	DATA-613	• Literature & ethical review. • Exploratory data analysis & visualization using **ggplot2** and **plotly**, summary tables, and interactive maps.
Alex Arevalo	DATA-413	Data acquisition, cleaning, and transformation; UI and UX.

11. Quick Reference Commands

Task	Command
Update repository	`git pull origin dev`
Switch branch	`git checkout feature/<branch>`
Run the app	`shiny::runApp(".")`
Stage & commit	`git add . && git commit -m "message"`
Push updates	`git push origin feature/<branch>`
Open PR	Merge `feature` → `dev`
Deploy stable app	Merge `dev` → `main`, then deploy

12. Final Best Practices

- Work only in your assigned branch.
- Pull the latest `dev` before editing.
- Never push directly to `main`.
- Test thoroughly before committing.
- Keep changes small and modular.
- Communicate clearly about Pull Requests.
- Ask for review before merging.

13. Completion Checklist for All Developers

☐ Clone the repository
☐ Open `gp-food-basket.Rproj`

- ☐ Run `source("global.R")` to install dependencies
- ☐ Checkout your feature branch
- ☐ Confirm `shiny::runApp(".")` works
- ☐ Commit changes regularly
- ☐ Push to GitHub and open PR into `dev`
- ☐ Follow coding and testing rules

14. Summary

This document provides everything required to:

- Set up your environment
- Understand project structure and branching
- Follow version control and naming standards
- Write clean modular Shiny code
- Troubleshoot and test
- Request review and merge safely
- Deploy the final app

Keep this guide accessible — it is the **single source of truth** for all contributors working on **gp-food-basket**.