## **📦 Project Setup & Infrastructure**

### **1. Initialize Project Scaffold**

**Description:** Bootstrap frontend and backend directories with README, license, and basic config.  
 **Tasks:**

* Create root README with project overview
* Add .gitignore for Node/Python and IDE files
* Create /frontend and /backend folders
* Initialize package.json or requirements.txt
* Add initial CI config (e.g. GitHub Actions lint/check)  
   **Labels:** infrastructure setup

### **2. Configure Hosting & Environments**

**Description:** Set up hosting stubs and environment variable management.  
 **Tasks:**

* Add .env.example for frontend (API\_URL, MAP\_KEY, FCM\_KEY)
* Add .env.example for backend (DB\_URI, JWT\_SECRET, FCM\_SERVER\_KEY)
* Document deployment targets (Vercel/Netlify, Heroku/Render) in README  
   **Labels:** infrastructure deployment

## **🔐 Authentication & User Management**

### **3. Implement User Signup & Login (Backend)**

**Description:** Build REST endpoints for email/password registration and login.  
 **Tasks:**

* Create /api/auth/signup (hash password with bcrypt)
* Create /api/auth/login (issue JWT)
* Validate input and return meaningful errors
* Unit tests for signup/login flows  
   **Labels:** backend auth

### **4. Add Frontend Auth Flows & Protected Routes**

**Description:** Create signup/login pages and protect other routes.  
 **Tasks:**

* Build Signup and Login React/Vue components
* Store JWT in secure, HTTP‑only cookie or localStorage
* Redirect unauthenticated users to login
* Add “Logout” button in UI  
   **Labels:** frontend auth UX

## **🥫 Surplus Listing CRUD**

### **5. Design Listing Data Model**

**Description:** Define database schema/collection for surplus listings.  
 **Tasks:**

* Draft JSON schema with fields: id, ownerId, photoUrl, description, quantity, expiryDate, pickupWindow, location, status
* Implement Mongoose schema or Firestore rules
* Write unit tests validating schema constraints  
   **Labels:** backend database

### **6. Create Listing API Endpoints**

**Description:** Build backend routes for listing management.  
 **Tasks:**

* POST /api/listings → create new listing
* GET /api/listings → list all “Available” items
* PUT /api/listings/:id → update own listing
* DELETE /api/listings/:id → delete own listing
* Add request validation & error handling
* Test each endpoint with Postman or automated tests  
   **Labels:** backend feature

### **7. Frontend Listing Form & Display**

**Description:** Implement UI for creating and viewing listings.  
 **Tasks:**

* Build “New Listing” form with photo upload, description, quantity, expiry, pickup window, and map picker
* Display listings in a card/grid view with summary info
* Validate form inputs and show errors
* Show success/failure toasts  
   **Labels:** frontend feature UX

## **🗺️ Interactive Map Integration**

### **8. Integrate Google Maps & Geolocation**

**Description:** Show map and allow users to pick coordinates.  
 **Tasks:**

* Install and configure Google Maps JS API
* Create Map component that centers on user location
* Implement draggable marker or click‑to‑set for new listings
* Display map pins for each “Available” listing  
   **Labels:** frontend feature maps

### **9. Show Listing Details on Map Pins**

**Description:** Clicking a pin shows a popup with listing info and “Claim Pickup” button.  
 **Tasks:**

* Add InfoWindow or custom popup for each marker
* Fetch listing details by ID when pin clicked
* Ensure map updates when listings change  
   **Labels:** frontend feature maps

## **🚴 Volunteer Matching & Notifications**

### **10. Implement Claim‑Pickup Logic (Backend)**

**Description:** Build endpoint to allow volunteers to claim a listing.  
 **Tasks:**

* POST /api/listings/:id/claim → set status to “Claimed,” record volunteerId
* Prevent claiming own listings or double‑claims
* Return updated listing in response
* Unit tests for claim behavior  
   **Labels:** backend feature

### **11. Push Notifications via FCM**

**Description:** Notify volunteers and listers about claims and chat messages.  
 **Tasks:**

* Integrate Firebase Cloud Messaging in backend
* Send notification to lister when listing is claimed
* Send notification to volunteer when lister responds
* Frontend setup to request notification permissions  
   **Labels:** backend notifications frontend

## **💬 In‑App Chat & Real‑Time Coordination**

### **12. Set Up Socket.IO for Chat (Backend)**

**Description:** Configure real‑time chat server.  
 **Tasks:**

* Install and configure Socket.IO on backend server
* Authenticate sockets using JWT
* Define events: join\_room, message, leave\_room
* Persist chat messages in database  
   **Labels:** backend realtime feature

### **13. Build Chat UI Component (Frontend)**

**Description:** Create chat window for listing coordination.  
 **Tasks:**

* Chat sidebar or modal within Listing Details
* Connect to Socket.IO namespace/room for each listing
* Display message history and input box
* Show notifications for new messages  
   **Labels:** frontend feature realtime

## **📊 Dashboard & Impact Metrics**

### **14. Aggregate Metrics Service (Backend)**

**Description:** Compute total items rescued, meals, estimated CO₂ saved.  
 **Tasks:**

* Write service to scan “Delivered” listings and sum quantities
* Apply conversion: 1 item = 1 meal; 1 meal = 2.5 kg CO₂e
* Expose GET /api/dashboard/stats endpoint
* Unit tests for calculations  
   **Labels:** backend feature analytics

### **15. Dashboard UI Component (Frontend)**

**Description:** Display impact stats with charts or counters.  
 **Tasks:**

* Fetch stats endpoint on load
* Show numeric counters (items rescued, meals, CO₂ saved)
* (Stretch) Add simple bar or pie chart
* Style for clarity and visual appeal  
   **Labels:** frontend feature UX

## **🛠️ Admin Mode (Stretch Goal)**

### **16. Shelter Admin Role & UI**

**Description:** Allow special users to confirm deliveries and manage needs calendar.  
 **Tasks:**

* Extend User model with role field (user, admin)
* Protect admin endpoints (e.g. /api/admin/confirm)
* Build simple Admin dashboard page
* CRUD operations for upcoming need slots  
   **Labels:** backend frontend feature stretch

## **🧪 QA, Testing & Deployment**

### **17. Comprehensive Error Handling & Validation**

**Description:** Ensure robust handling across app.  
 **Tasks:**

* Validate all user inputs (forms, query params)
* Gracefully handle network errors in UI (toasts/modals)
* Log backend errors and return standardized error JSON
* Write end‑to‑end tests for main flows (signup → listing → claim)  
   **Labels:** bug testing quality

### **18. CI/CD Pipeline & E2E Tests**

**Description:** Automate linting, testing, and deployment.  
 **Tasks:**

* Configure GitHub Actions: lint, unit tests, build
* Add Cypress or Playwright for critical end‑to‑end tests
* Auto‑deploy frontend to Vercel/Netlify on merge to main
* Auto‑deploy backend to Heroku/Render on merge to main  
   **Labels:** infrastructure testing deployment

### **19. Write Detailed Documentation**

**Description:** Document setup, contribution guide, API specs.  
 **Tasks:**

* Update README with architecture overview and tech stack
* Add API reference markdown
* Write frontend usage guide (screenshots, flow)
* Include glossary of endpoints and data models  
   **Labels:** documentation

### **20. Final Deployment & Demo Preparation**

**Description:** Polish UI, ensure feature completeness, and prepare demo.  
 **Tasks:**

* Verify production builds run without errors
* Perform cross‑browser and mobile testing
* Record a 3‑5 min demo walkthrough video
* Create presentation slides for judges  
   **Labels:** demo UX deployment