Helpkart Distribution Center Management System

Complete Implementation Guide

□ Quick Start

1. Prerequisites

- Python 3.8 or higher
- MongoDB Atlas account (free tier available at mongodb.com)
- Git

2. Installation Steps

Step 1: Clone or Create Project Directory

```
mkdir helpkart-app
cd helpkart-app
```

Step 2: Create Virtual Environment

```
python -m venv venv

# On Windows
venv\Scripts\activate

# On macOS/Linux
source venv/bin/activate
```

Step 3: Install Dependencies

```
pip install -r requirements.txt
```

Step 4: Setup MongoDB Atlas

1. Go to MongoDB Atlas
2. Create a free account
3. Create a new cluster
4. Click "Connect" and choose "Drivers"

5. Copy the connection string
6. Replace `&lt;username&gt;` and `&lt;password&gt;` with your credentials

Step 5: Configure Environment

1. Create a `.env` file in project root
2. Add your MongoDB connection string:

```
MONGODB_URI=mongodb+srv://username:password@cluster-name.mongodb.net/?retryWrites=true&am
```

Step 6: Run the Application

```
streamlit run app.py
```

The app will open at `http://localhost:8501`

Project Structure

```
helpkart-app/
├── app.py                  # Main application file (login/signup)
├── requirements.txt        # Dependencies
├── .env                    # MongoDB connection (create locally)
├── .env.example            # Example env file
│
└── pages/                  # Page modules
    ├── __init__.py
    ├── dashboard.py        # Dashboard overview
    ├── inventory.py        # Manage inventory
    ├── requests.py         # Manage requests
    ├── browse.py           # Browse items &amp; requests
    ├── transactions.py     # Transaction history
    └── settings.py         # User settings
```

MongoDB Database Collections

Collection 1: `centers` (Centers/Users)

```
{
  "_id": ObjectId,
  "center_id": "uuid-string",
  "center_name": "Relief Center Name",
  "email": "center@email.com",
  "password": "hashed_password",
  "phone": "0112345678",
  "address": "Full Address",
  "location_coordinates": {
    "lat": 6.9271,
    "lng": 79.8612
  },
  "status": "active",
  "created_at": ISODate,
  "updated_at": ISODate
}
```

Collection 2: `inventory` (Items Centers Have)

```
{
  "_id": ObjectId,
  "inventory_id": "uuid-string",
  "center_id": "center-uuid",
  "item": "Rice",
  "quantity": 50,
  "unit": "kg",
  "surplus_or_stock": "surplus",
  "notes": "Optional notes",
  "added_on": ISODate,
  "expiry_date": ISODate
}
```

Collection 3: `requests` (Items Centers Need)

```
{
  "_id": ObjectId,
  "request_id": "uuid-string",
  "center_id": "center-uuid",
  "item": "Water",
  "quantity_needed": 100,
  "unit": "liters",
  "urgency": "critical",
  "description": "Why needed",
  "requested_on": ISODate,
  "fulfilled": false,
  "status": "open"
}
```

Collection 4: `transactions` (Who Gave to Whom)

```
{
  "_id": ObjectId,
  "transaction_id": "uuid-string",
  "from_center_id": "center-uuid",
  "to_center_id": "center-uuid",
  "item": "Rice",
  "quantity": 20,
  "unit": "kg",
  "transaction_date": ISODate,
  "status": "pending|completed|cancelled",
  "message": "Optional message"
}
```

Features Overview

Dashboard

- Quick statistics (surplus items, active requests, pending transactions)
- Recently added items
- Recent network requests from other centers
- Center information card

My Inventory

- Add new items to inventory
- Edit quantities and status
- Delete items
- Filter by surplus/in-stock
- View expiry dates

My Requests

- Post requests for needed items
- Track request status
- View who fulfilled requests
- Delete open requests
- Filter by urgency level

Browse Items

- Search for available items
- Filter by item name, type
- View all network surplus items
- Create requests for items
- Respond to other centers' requests
- See center details

Transactions

- **Received**: Items your center got from others
- **Sent**: Items your center gave to others
- **Pending**: Transactions awaiting approval
- Approve or reject transactions
- View messages and details

Settings

- Update center profile
- Change password (with current password verification)
- View center statistics
- Delete account (with data cleanup)

▯ Authentication & Security

Password Security

- Passwords hashed using `bcrypt` (salt rounds: 10)
- Never stored in plain text
- Verified using bcrypt's `checkpw()` function

Session Management

- Uses Streamlit's `st.session_state`
- Stores: `logged_in`, `center_id`, `center_name`, `center_email`
- Session persists during browser session
- Logout clears all session state

Best Practices Implemented

- Email uniqueness validation during signup
- Password confirmation validation
- Minimum password length (6 characters)
- Hashed password storage in MongoDB
- ObjectId validation for database queries

Deployment Options

Option 1: Streamlit Cloud (Recommended for Beginners)

1. Push code to GitHub
2. Go to Streamlit Cloud
3. Click "New app"
4. Select your GitHub repo
5. Set `MONGODB_URI` in Secrets (Settings → Secrets)
6. Deploy!

Option 2: Heroku/Railway

1. Add `Procfile`:
   ```
   web: streamlit run app.py
   ```
2. Deploy repository
3. Set `MONGODB_URI` in environment variables

Option 3: Self-hosted Server

1. Install Python and dependencies
2. Use `gunicorn` or similar for production serving
3. Set up reverse proxy (nginx)
4. Configure SSL certificate

Testing the App

Test Workflow:

1. **Sign up** 2-3 test centers
2. **Add Inventory**: Each center adds surplus items
3. **Post Requests**: Centers post what they need
4. **Browse**: Navigate to "Browse Items" to see network items
5. **Create Transaction**: Request items from other centers
6. **Approve Transaction**: Approve pending transactions
7. **Check History**: View completed transactions

Test Credentials:

```
Center 1:
Email: center1@test.com
Password: Test@123

Center 2:
Email: center2@test.com
Password: Test@123
```

Key Technical Decisions

Why These Technologies?

- **Streamlit**: Rapid UI development, no frontend coding needed
- **MongoDB**: Flexible schema, scales well, free tier available
- **PyMongo**: Direct MongoDB integration, simple queries
- **bcrypt**: Industry-standard password hashing

Data Flow:

1. User signs up → Hashed password stored in MongoDB
2. User logs in → Password verified against hash
3. Session state manages navigation without page reloads
4. Each page imports `get_database()` to query MongoDB
5. Real-time updates via Streamlit's automatic rerun on widget interaction

Troubleshooting

Connection Error: "Failed to connect to MongoDB"

- Check MongoDB URI in `.env`
- Ensure IP is whitelisted in MongoDB Atlas
- Verify username/password are URL-encoded

"Module not found" errors

- Ensure `sys.path.insert(0, ...)` is at top of page files
- Check all files in `pages/` directory

Changes not appearing

- Clear browser cache
- Restart Streamlit: `Ctrl+C`, then `streamlit run app.py`

Slow performance

- Check MongoDB indexes
- Optimize queries (add filters before `find()`)
- Use `.limit()` for large datasets

Future Enhancements

1. **Email Notifications**: Alert centers when requests are posted
2. **Reputation System**: Star ratings after completed transactions
3. **AI Matching**: Auto-match requests with surplus items
4. **Mobile App**: React Native or Flutter wrapper
5. **Analytics Dashboard**: Trends in items, most requested, etc.
6. **Maps Integration**: Leaflet map like original Helpkart
7. **Offline Sync**: Work offline, sync when online
8. **Multi-language**: Support Sinhala, Tamil

Important Notes

Permissions

- Centers can only see their own data
- Can see all other centers' public surplus/requests
- Transactions are private between centers

Data Privacy

- Email addresses not shared publicly
- Phone numbers visible only after transaction initiation
- Location coordinates used for distance calculations (if added)

Scalability

- Currently designed for 100-1000 centers
- For larger scale: add caching, optimize queries, consider sharding

Support

For issues or questions:

1. Check the troubleshooting section
2. Review MongoDB documentation
3. Check Streamlit documentation
4. Create an issue on GitHub

Version: 1.0.0
Last Updated: December 2025
Status: Production Ready
[1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17] [18] [19] [20]

⁂

1. https://www.linkedin.com/pulse/building-multi-page-app-streamlit-nethmi-nikeshala-1l1wc
2. https://discuss.streamlit.io/t/multipage-app-directory-structure-and-imports/61344
3. https://www.youtube.com/watch?v=dSwyxVJPsP0
4. https://ryanandmattdatascience.com/streamlit-session-state/
5. https://docs.streamlit.io/get-started/tutorials/create-a-multipage-app
6. https://www.mongodb.com/docs/languages/python/pymongo-driver/current/connect/connection-targets/
7. https://discuss.streamlit.io/t/good-practices-streamlit-code/38145
8. https://discuss.streamlit.io/t/multi-page-label-presentation/36523
9. https://www.mongodb.com/docs/languages/python/pymongo-driver/current/get-started/
10. https://docs.kanaries.net/topics/Streamlit/streamlit-session-state
11. https://andrewm4894.com/2021/05/27/streamlit-multi-page-app-minimal-example/
12. https://pymongo.readthedocs.io/en/4.7.3/atlas.html
13. https://www.mongodb.com/docs/manual/reference/connection-string/
14. https://discuss.streamlit.io/t/state-management-best-practices/24735
15. https://discuss.streamlit.io/t/navigate-multipage-app-with-buttons-instead-of-sidebar/27986
16. https://stackoverflow.com/questions/74721623/how-do-you-use-pymongo-to-connect-to-mongodb-atlas
17. https://nimbusintelligence.com/2023/10/streamlit-applications-a-guide-to-reactivity-and-statefulness/
18. https://towardsdatascience.com/how-to-build-an-interconnected-multi-page-streamlit-app-3114c313f88f/
19. https://www.geeksforgeeks.org/mongodb/how-to-use-mongodb-connection-string/

20. https://discuss.streamlit.io/t/seeking-advice-for-streamlit-app-state-management-and-best-practices/80025