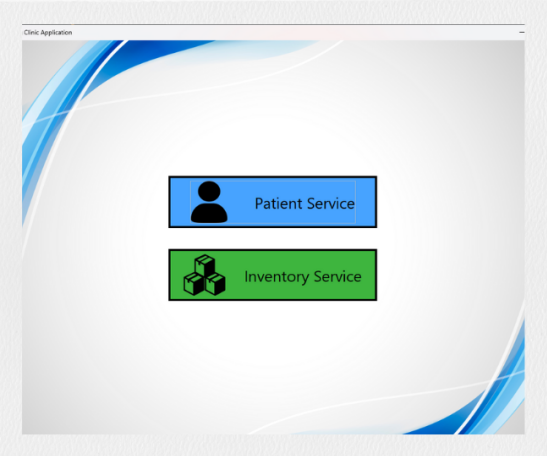
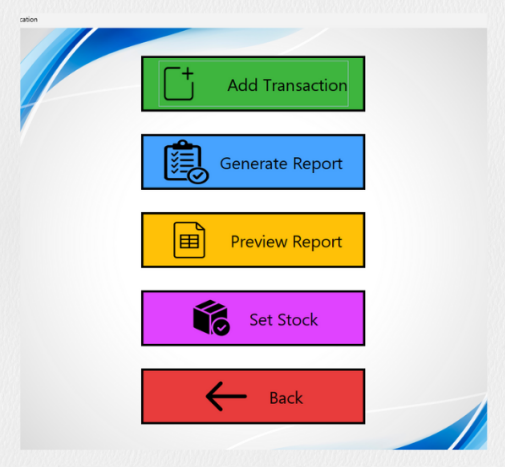
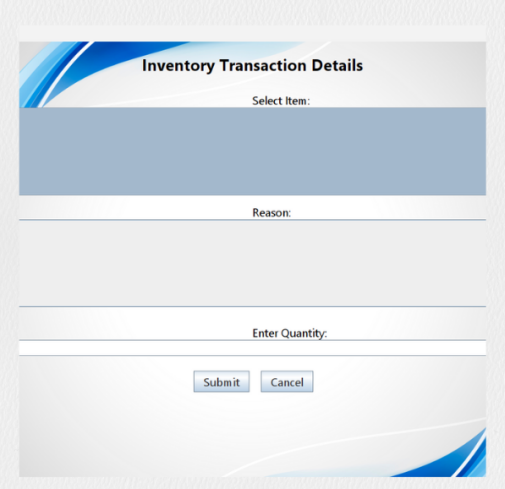
Iteration 3 Report

Family Planning System

Author: Sarah Kayembe
Group Members: Ethan Gilles, Ben Gaudreau, Michael Yattaw, Sarah Lawrence
Date: May 2, 2025







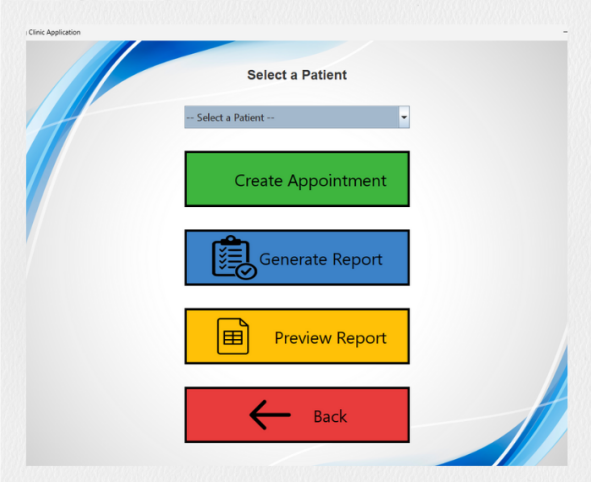


Table of Contents

1. Introduction

The Ghana Health System project originated as a semester-long initiative focused on developing a modular healthcare software system, with a particular emphasis on managing family planning services. The project's third iteration was a critical stage in its development, concentrating on solidifying the fundamental software architecture, improving the user interface for practical application, and incorporating essential features like patient registration and inventory tracking. As the final active development phase within the semester's timeframe, Iteration 3 heavily emphasized integrating prior work, refining the initial code, and testing methods. This report details the activities, progress, challenges encountered, and outcomes achieved during Iteration 3.

2. Team Members & Roles

As the group leader for this iteration, my responsibilities were crucial to ensuring its successful completion. Primarily, I was responsible for authoring the iteration report, a comprehensive document detailing the team's activities, progress, and outcomes. This involved gathering information from each team member repo on GitHub, synthesizing it into a coherent narrative, and presenting it in a clear and concise manner. Additionally, I developed the presentation slides. This required me to distill the key findings from the report, design engaging visuals, and structure the presentation for maximum impact. Beyond documentation and presentation, a significant aspect of my role was to ensure that all the requirements for this iteration were met. This involved close monitoring of the team's progress, identifying and addressing any roadblocks, and coordinating efforts to guarantee that all deliverables were completed to the required specifications and standards of our team.

Ethan Gilles and Ben Gaudreau held primary responsibility for the development and integration of the system's user interface (UI). Their work commenced with the creation and presentation of a UI proof-of-concept to the team, which subsequently received unanimous approval. Following this validation, they expanded the UI to incorporate several essential features, including interactive elements and functionalities for report generation, inventory stock management, and report preview. Notably, they implemented a dialog interface for the "Add Transaction" function and established its connection to the inventory service to ensure operational integrity. Collaborating closely with Michael Yattaw, Ethan and Ben contributed to the establishment of a coherent package structure for the UI components. Furthermore, they employed issue tracking methodologies to effectively manage and monitor the individual tasks critical to the successful completion of the iteration.

The central objective of their contribution was to enable authorized users with the ability to directly define the current inventory level, independent of individual transaction records. This designated stock value would serve as the "initial value" within the monthly report, representing the baseline inventory at the commencement of each reporting cycle. To realize

this objective, Ethan and Ben introduced a dedicated button within the InventoryPanel. Once clicked, this button initiates a dialog interface, enabling users to input stock quantities for each distinct inventory item. This dialog was intentionally designed to mirror the structure of the report generation interface, promoting a consistent user experience. In addition to enhancing the UI, they developed a corresponding controller class responsible for transmitting this information to the InventoryService. Consequently, the InventoryService was augmented to accommodate and persist this initial stock value, ensuring that future reports could accurately reflect inventory changes over time.

Sarah Lawrence and Michael Yattaw focused primarily on enabling nurses to schedule appointments for patients directly from the PatientPanel, replacing the previously non-functional "Under Construction" placeholder. They developed an initial UI for generating the Family Planning Client Report, building upon the foundation established by Ethan and Ben. This interface included several functional buttons, such as "Generate Report," "Preview Report," "Create Patient," "Delete Patient," and "Appointment." These additions enhanced the interactivity and utility of the user interface, providing a more comprehensive view and control over patient records and appointments.

To facilitate appointment creation, Sarah and Michael implemented a patient selector component (in the form of a combo box or search field), a dropdown menu listing contraceptive methods via ContraceptiveMethod.values(), and a "Create Appointment" button. They also developed an ActionListener and corresponding controller logic to collect the form data and process the appointment creation. Upon submission, the system displays either a success dialog confirming the operation or an error dialog in case of failure.

While the goal was to fully enable appointment scheduling within the PatientPanel, several features were left incomplete due to time constraints.

3. Objectives for Iteration 3

3.1 Objectives

- Enhanced User Interface & Backend Integration:
  - Enable nurses to schedule appointments directly from the `PatientPanel` (replacing the "Under Construction" message).
  - Implement a feature for users to set the current inventory amount directly, independent of transactions, for use as the initial value in monthly reports.
  - Develop a comprehensive UI with buttons for all major use cases.
  - Automate the generation of monthly reports with export options.
  - Prepare presentation slides for the final report.

3.2 Outcomes

- Successful delivery of a cohesive and functional final product.

4. Tasks Completed

4.1 Database Implementation Approach

4.1.1 Initial Plan: PostgreSQL Integration

The system was originally designed to use PostgreSQL for persistent data storage, with the following components implemented:

- **DAO (Data Access Object) Classes**
  - Fully functional CRUD operations (Create, Read, Update, Delete)
  - Tested with JUnit to ensure proper database interactions
- **Supported tables for:**
  - Patients
  - Appointments
  - Inventory
  - Employees
- **SQL Query Execution**
  - Direct SQL queries from Java (`PreparedStatement`, `ResultSet`)
  - Proper connection pooling and transaction management
- **Test Data Population**
  - Sample data inserted for validation
  - Schema design optimized for family planning clinic workflows

4.1.2 Change in Direction: Temporary In-Memory Storage

After a group leader meeting, the different teams voted for a simplified in-memory storage solution due to time constraints.

Current Implementation:

```
1 familyPlanningPatients = new ArrayList<>(Arrays.asList(
2     createPatient(date, "exampRegNum15263", "cardnum2773", "John", "Doe",
    "Married", "Male", "Chicken Dinner Road", 25),
3     createPatient(date, "exampRegNum15263", "cardnum23873", "Jane", "Smith
    ", "No Marry", "Female", "Frying Pan Alley", 69),
4     createPatient(date, "exampRegNum15263", "cardnum2263", "Alice", "
    Johnson", "Forever alone", "Non Binary", "Ha-ha Road", 42)
```

```
));
```

Listing 1: Patient Data Initialization

CSV exports serve as temporary persistence for reports.

4.1.3 Justification for Simplification

- Faster iteration during development
- Reduced complexity for initial testing
- Allowed focus on core logic before database optimization

4.1.4 Future Scalability

The existing DAO layer remains fully functional—switching to PostgreSQL would require only:

- Enabling the database connection
- Removing hardcoded test data
- Minor adjustments to service classes

4.1.5 Conclusion

While the final version uses in-memory storage for simplicity, the system was designed for easy PostgreSQL integration when needed. The team prioritized feature completeness over persistence in this iteration but retains the ability to upgrade seamlessly.

4.2 Frontend

The frontend of the Ghana Health System family planning module was implemented using **Java Swing** and follows the **Model-View-Controller (MVC)** architecture. The main window, `FamilyPlanningUI`, serves as the central entry point for the application and employs a `CardLayout` for switching between different functional panels, namely the `MainMenuPanel`, `InventoryPanel`, and `PatientPanel`. It also provides utility methods for window management and icon resizing.

4.2.1 Main Application Panels and Features

Patient Management System Patient Panel (`PatientPanel`):

- Enables users to select existing patients from a dropdown menu with a custom renderer.
- Includes buttons for: `Create Appointment`, `Generate Report`, `Preview Report`, `Create Patient`, and `Delete Patient`.
- Facilitates navigation to detailed data entry and report generation interfaces.

- Patient data is stored in memory, and sample patients are pre-loaded for demonstration.

Family Planning Patient Form (`FamilyPlanningPatientForm`):

- A modal dialog (`JDialog`) designed for entering detailed patient data.
- Fields include:
  - Medical history (`JTextField`)
  - First-time use (`JCheckBox`)
  - Method of choice (`JComboBox`)
  - Parity (`JSpinner`)
  - Monthly usage tracking (`JTextField[]` for 12 months)
- Buttons: `Save` and `Cancel`, with flags to determine user intent.
- Provides data retrieval methods:
  - `getMedicalHistory()`
  - `isFirstUse()`
  - `getMethodOfChoice()`
  - `getParity()`
  - `getMonthlyUsage()` returns a `Map<Month, Item>`

Patient Report Controller (`PatientReportController`):

- Manages the workflow for creating appointments and generating reports.
- Validates patient selection and saves data to model objects.
- Mediates between `PatientPanel` and forms such as `PatientReportOptions`.
- Handles user feedback via dialog notifications.

Inventory Management System Inventory Panel (`InventoryPanel`):

- Acts as the central interface for inventory operations.
- Features large icon buttons and a background image for visual clarity.
- Functionalities include: `Add Transaction`, `Generate Report`, `Preview Report`, `Set Stock`, and navigation back to the main menu.

Inventory Transaction Form (`InventoryTransactionOptions`):

- Includes dropdown for selecting contraceptive items (14 total).
- Quantity input and transaction reason (`Issued`, `Expired`, `Transferred`, `Received`).

- Conditional field display for transfer partner only when reason is `'Transferred'`.
- Input validation ensures realistic entries.

Set Inventory Options (`SetInventoryOptions`):

- Modal form to set initial stock values for the 14 contraceptive methods.
- Implements a two-column layout using `GridBagLayout`.
- Validates quantity input and returns a `Map<String, Integer>` of item names to values.

Reporting System Monthly Report Options (`MonthlyReportOptions`):

- Form-based interface to create inventory reports.
- Collects employee name and item-wise quantities.
- Offers export and preview functionality.

CSV Viewer Panel (`CSVViewerPanel`):

- Displays generated CSV reports in table format.
- Supports file chooser integration and handles malformed files and quoted values.

Navigation and Visual Design Navigation System:

- `CardLayout` enables switching between `MainMenuPanel`, `InventoryPanel`, and `PatientPanel`.
- Clear back buttons and contextual navigation for seamless user flow.

Main Menu Panel (`MainMenuPanel`):

- Entry point to Patient and Inventory services.
- Uses visually distinct icons and standardized styling.

UI Consistency:

- Background images and consistent button styling across forms.
- Responsive layout with proper alignment and spacing.
- Use of modal dialogs for focused user interactions.

Common Patterns and Implementation Highlights

- Unified submission logic with `isConfirmed()` checks across forms.
- Robust input validation and defaulting for empty or invalid entries.
- Clear separation between UI and business logic via controller classes.
- Confirmation dialogs for destructive actions (e.g., patient deletion).
- Use of custom components and renderer logic to enhance user experience.
- Graceful error handling and informative success/failure dialogs.

4.3 Testing

The **Ghana Health System – Family Planning Module** uses a structured, multi-layered testing strategy to ensure the reliability of its core logic, services, and data access layers. Our approach emphasizes unit testing, supplemented by integration testing, file I/O validation, and DAO testing for database operations. These tests were implemented using JUnit, with careful attention to isolating test cases and maintaining clean environments.

4.3.1 Testing Strategy and Approach

- **Unit Testing**: Focused on verifying individual class behavior, particularly for the model and service layers. Tests include core logic such as appointment scheduling, business rule enforcement, and report calculations.
- **Integration Testing**: Conducted on a limited basis to verify the interaction between services and data generation utilities (e.g., CSV report output).
- **File I/O Testing**: Ensures that CSV generation for reports is functioning correctly, including file creation, format validation, and content structure.
- **DAO Layer Testing**: Each Data Access Object (DAO) class was tested for CRUD operations to confirm correct database integration using PostgreSQL during development.

4.3.2 Test Coverage Breakdown

- **Model Layer Tests (12+ total)**:
  - `AppointmentTest.java` (5 tests): Validates appointment creation, next dose calculation, alert behavior, and invalid inputs.
  - `MonthlyReportTest.java` (1 integration test): Tests CSV generation from mock data, ensuring the output structure and file creation logic are correct.
  - `PatientReportTest.java` (2 tests): Confirms successful report generation for both single and multiple patients, covering edge cases such as empty fields.

- **Service Layer Tests (5 total)**:
  - `AppointmentServiceTest.java`: Verifies appointment creation with role-based security, update and cancel logic, and error handling for unauthorized actions.
- **DAO Layer Tests (12 total across 4 classes)**:
  - `ClientDAOTest.java`
  - `LocationDetailsDAOTest.java`
  - `MethodsOfChoiceDAOTest.java`
  - `MonthlyVisitDAOTest.java`
  - Each includes methods for Add/Retrieve, Update, and Delete operations.

4.3.3 Key Testing Features and Patterns

- **Test Isolation**: Used `@BeforeEach` and `@BeforeAll` to maintain clean setup for each test.
- **Assertion Techniques**: Used `assertEquals`, `assertTrue`, `assertFalse`, `assertThrows`, and `assertNull`.
- **File System Testing**: Used `@TempDir` to avoid filesystem pollution.
- **Security and Exception Testing**: Ensured correct exceptions using role checks and `SecurityException`.

4.3.4 Example Test Cases

Behavior Verification

```
@Test
public void testCalculateNextDoseUpdatesDateAndPatientMethod() {
    appointment.calculateNextDose(30);
    assertTrue(appointment.getNextDoseDate().isPresent());
    assertEquals(LocalDateTime.of(2025, Month.MAY, 13, 10, 0),
                 appointment.getNextDoseDate().get());
    assertEquals("Injectable Contraceptive", patient.getMethodOfChoice());
}
```

Exception Testing

```
@Test
public void testCalculateNextDoseThrowsExceptionForInvalidDuration() {
    assertThrows(IllegalArgumentException.class,
                 () -> appointment.calculateNextDose(0));
}
```

File System Testing

```
@Test
public void testGenerateCSV() throws IOException {
    Files.createDirectories(tempDir);
    // ... test file generation
    assertTrue(expectedFile.exists(), "CSV file should have been created."
   );
}
```

Security Testing

```
@Test
public void testCreateAppointmentUnauthorizedThrows() {
    SecurityException ex = assertThrows(SecurityException.class, () -> {
        service.createAppointment(nurse, patient, LocalDateTime.now(),
                                  ContraceptiveMethod.PILL, Optional.empty
   ());
    });
    assertEquals("Employee not authorized to manage appointments", ex.
   getMessage());
}
```

4.3.5 Test Statistics and Summary

- **Total Test Classes**: 8
- **Total Test Methods**: Approximately 30
- **Coverage Summary**:
  - Model business logic: 100%
  - Service layer: ~80%
  - DAO operations: 100%
  - File generation logic: ~70%
  - UI logic and controllers: Not covered

5. Data Handling

Data handling is structured around a multi-layered architecture that separates concerns between the user interface, controllers, service classes, and DAO classes. This architecture enables reliable, maintainable, and testable data flow throughout the application.

5.0.1 Client and Patient Data

Client and patient information is captured via the `FamilyPlanningPatientForm`, where users enter personal demographics (name, sex, address), medical history, parity, and method of choice. Upon submission, data is retrieved from the form using accessor methods like `getMethodOfChoice()` and `getMonthlyUsage()`, then stored in memory via the `PatientPanel`.

This panel maintains a list of `FamilyPlanningPatient` objects and supports operations such as adding new patients, updating existing ones, and deleting records.

5.0.2 Appointment and Visit Data

Appointments are created through the `PatientReportController`, which handles the logic for collecting and validating data from the UI and passing it to the service layer. The `AppointmentService` enforces business rules (e.g., only nurses can schedule appointments) and updates patient objects with the relevant appointment metadata.

Monthly visit information is stored in the `MonthlyVisitDAO`, which handles insertion and updates to the `monthly_visits` table in the database. Data collected through forms is mapped into Java model classes and then persisted to the database using standard CRUD operations.

5.0.3 Inventory Data

Inventory management is facilitated through forms such as `SetInventoryOptions` and `InventoryTransactionOptions`. These interfaces allow users to input quantities for each contraceptive method. Input data is validated for completeness and correct numeric format before being passed to the `InventoryService`, which manages the state of inventory over time. Both transactional data and initial stock settings are stored and used for generating monthly reports.

5.0.4 Validation and Error Handling

Field-Level Validation: All input fields across forms include built-in validation. Quantity fields reject non-numeric input, and required fields must be filled before proceeding.

Conditional UI Display: Some fields appear only based on selected options (e.g., transfer partner only appears for transfer transactions), reducing the risk of irrelevant or missing data.

Logical Validation: Appointments cannot be set in the past, and unauthorized roles trigger a `SecurityException` to prevent access to restricted actions.

Graceful Error Handling: `try-catch` blocks in controllers provide user-friendly error dialogs, and file I/O operations fail gracefully with meaningful messages.

6. Comparison to Iteration Plan and Identified Gaps

The following planned features were not completed in this iteration and will be addressed in the next phase:

- **Date & Time Picker**:
  An interactive picker for appointment scheduling was deferred due to UI implementation complexity.

- **Notes Text Area**:
  The feature to add appointment notes was postponed to prioritize core workflow functionality.
- **Employee Authentication & Service Integration**:
  - Passing the logged-in `Employee` object into `PatientPanel` was not fully implemented.
  - The shared `AppointmentService` instance was not properly injected for appointment handling.
- **Validation & Security Enforcement**:
  - **Future Date Validation**: Basic checks exist, but stricter enforcement is needed.
  - **Role-Based Access Control**:
    * Incomplete handling of `SecurityException`.
    * Nurse-only submission logic not fully enforced.

6.1 Identified Gaps and Areas for Improvement

Missing Tests:

- UI controller behavior (difficult to test without full integration or UI testing tools).
- Full CSV content accuracy under various input conditions.
- Larger scale scenarios (e.g., performance under heavy data loads).

7. Conclusion

Conclusion Iteration 3 represented a significant step towards realizing a functional family planning management system. The team successfully integrated key UI components, implemented essential features for patient and inventory management, and established a foundation for robust data handling. While some planned features were deferred, the prioritization of core functionality and a focus on testability have resulted in a solid base for future development. The challenges encountered, particularly in coordinating across sub-teams and managing time constraints, and provided valuable lessons in project management.

8. Task management Tools

roject tracker

anage and monitor tasks as a team

On the free version of Slack, lists are view only. To edit and create lists, upgrade to a paid plan. Upgrade now

All items

Task	Status	Priority	Description	Assignee	Due date
~~Break down the project into user stories~~	Done	☆☆☆		Ben Gaudreau	04/01/2025
~~Write use cases for core functionalities~~	Done	☆☆☆	(e.g., "Register a patient," "Recor...	Ethan Gilles	04/01/2026
~~Define actors~~	Done	☆☆☆	For example: CHO (Community ...	Sarah Lawrence	04/01/2025
~~Use Case Diagram for overall system interactions~~	Done	☆☆☆		Michael Yattaw	04/01/2025
~~Identify main system components & classes~~	Done	• • •			04/01/2025

Figure 1: Slack Task Tracker

Iteration 2: Use Case Picking

File Edit View Insert Format Data Tools Extensions Help

C25 TRUE

	A	B	C	D	E	F	G	H	I	J	K	L
1	Task:											
2	Pick 3 that you are intrested in doing. Please don't pick a task someone else has chosen.											
3	Why:											
4	Since there are 12 primary use cases everyone will get 3 assigned to them. When a team is done with ther usecases the two people will switch code and assess each others work.											
5	Note:											
6	The highligihing means it would be best if they were picked together. Also Please feel free to provide sugestions inorder to make this better!											
7			Team 1		Team 2		Team Lead					
8	Use Cases		Ben	Ethan	Michael	Sarah L	Sarah K	Completed				
9	Iteration Report						☑	☑				
10	~~Login~~											
11	Find Existing Patient						☑	☑				
12	Add New Patient						☑	☑				
13	Edit Patient Info						☑	☑				
14	View Patient Info						☑	☑				
15	Manage Appointments				☑	☑		☑				
16	Genrate Patient Report				☑	☑		☑				
17	~~Manage User Roles~~											
18	~~View User Info~~											
19	Create Monthly report		☑	☑				☑				
20	Modify Monthly report		☑	☑				☑				
21	Approve and return report		☑	☑				☑				
22												
23	If a Group Finishes early:											
24	Use Cases		Team 1	Team 2								
25	Atomated Report Processing		☑	☑								
26												
27												
28	14 Monday switch day											
29	16 Wednessday meeting											
30												

Figure 2: Excel Task Tracker

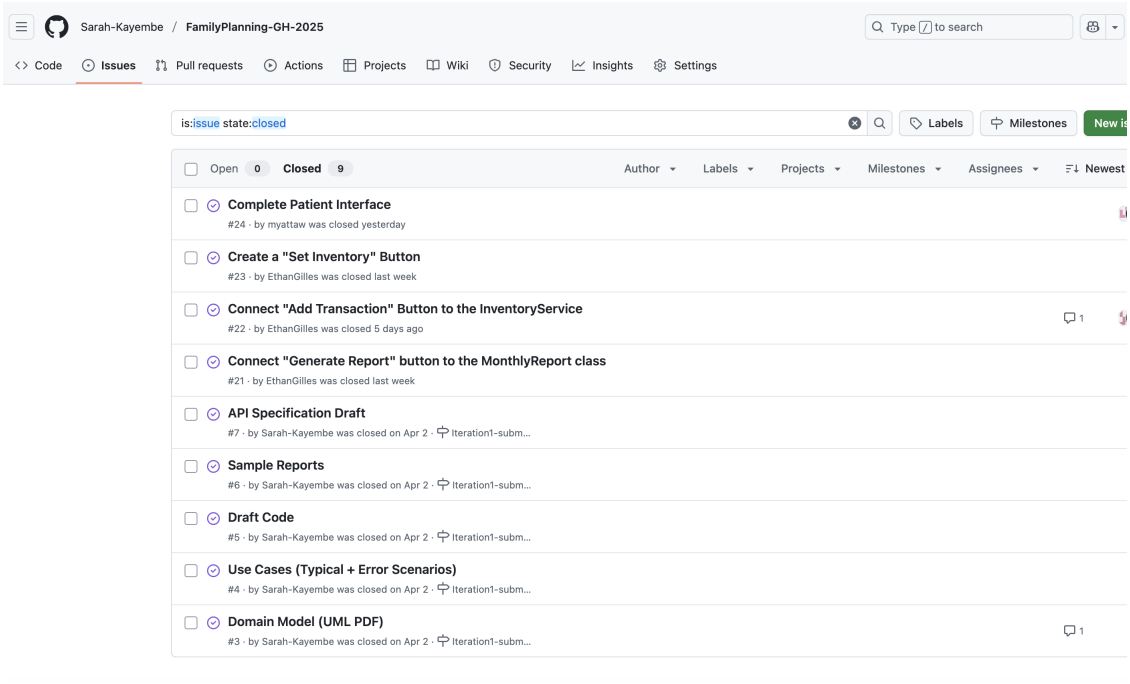


Figure 3: Github Task Tracker

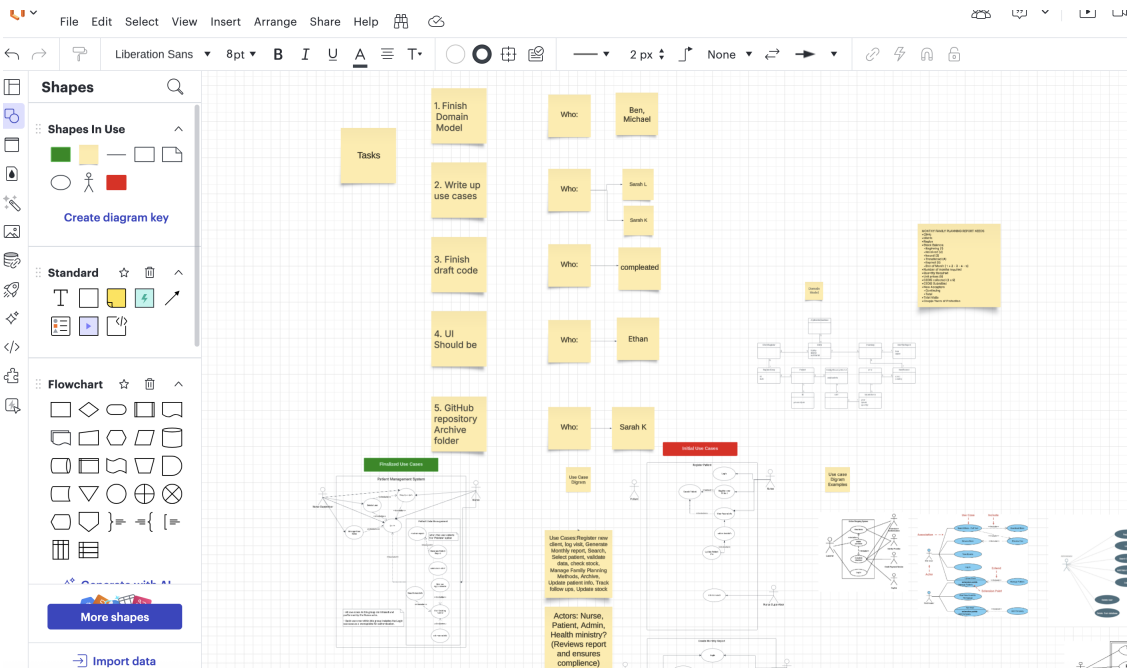


Figure 4: Lucid Chart Task Tracker

9. Screenshots of UI Sketch

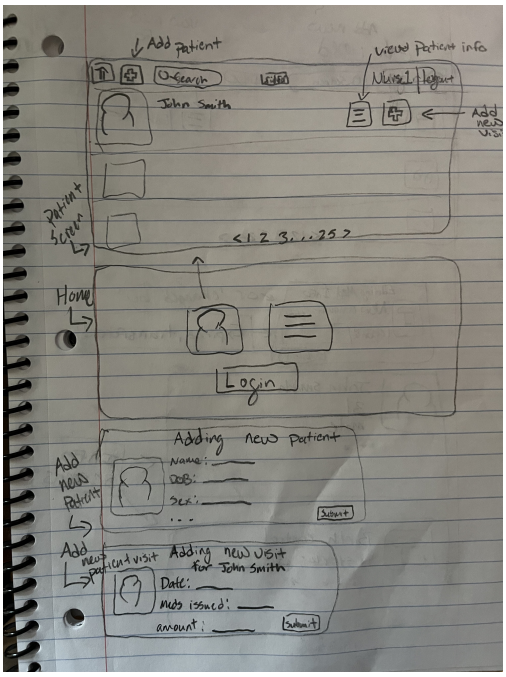


Figure 5: Landing page Patient Section

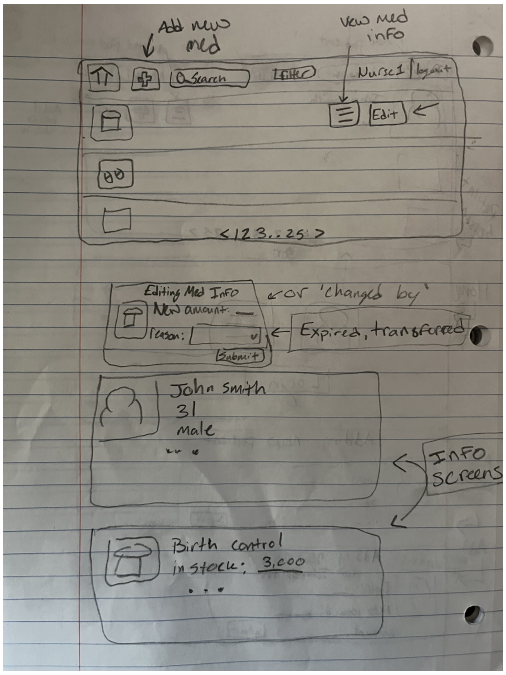


Figure 6: Landing page Inventory Section

10. Screenshots of UI

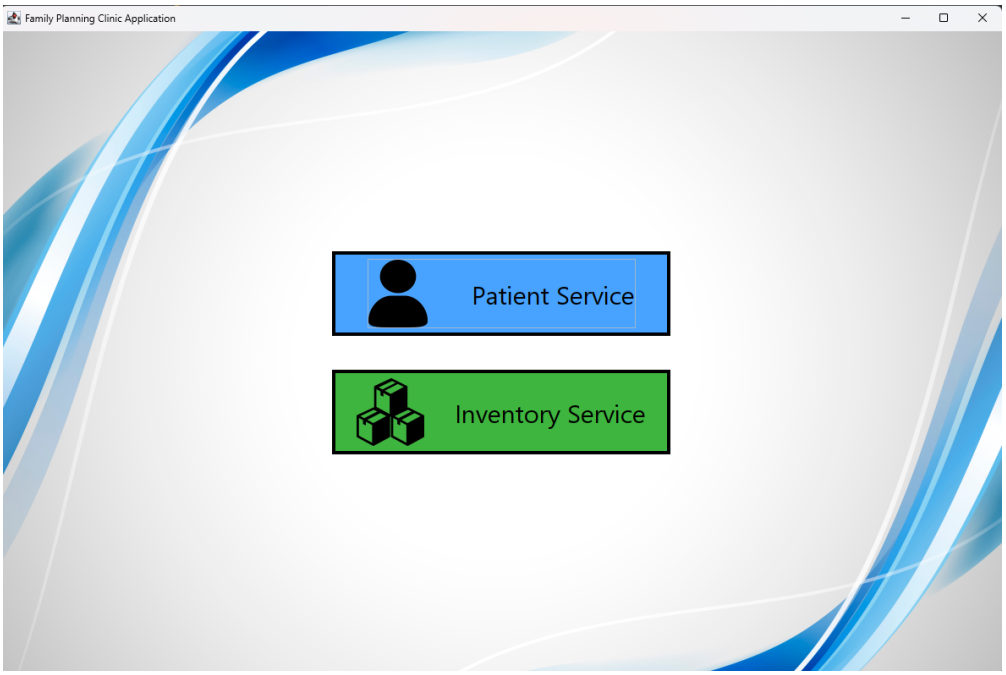


Figure 7: Landing page

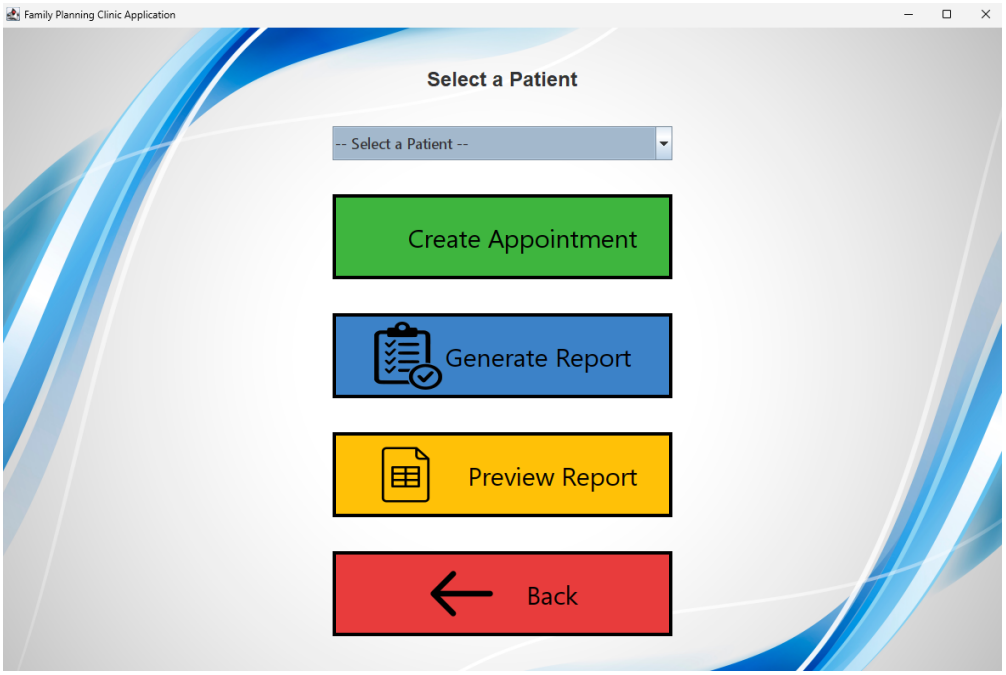


Figure 8: Patient Section

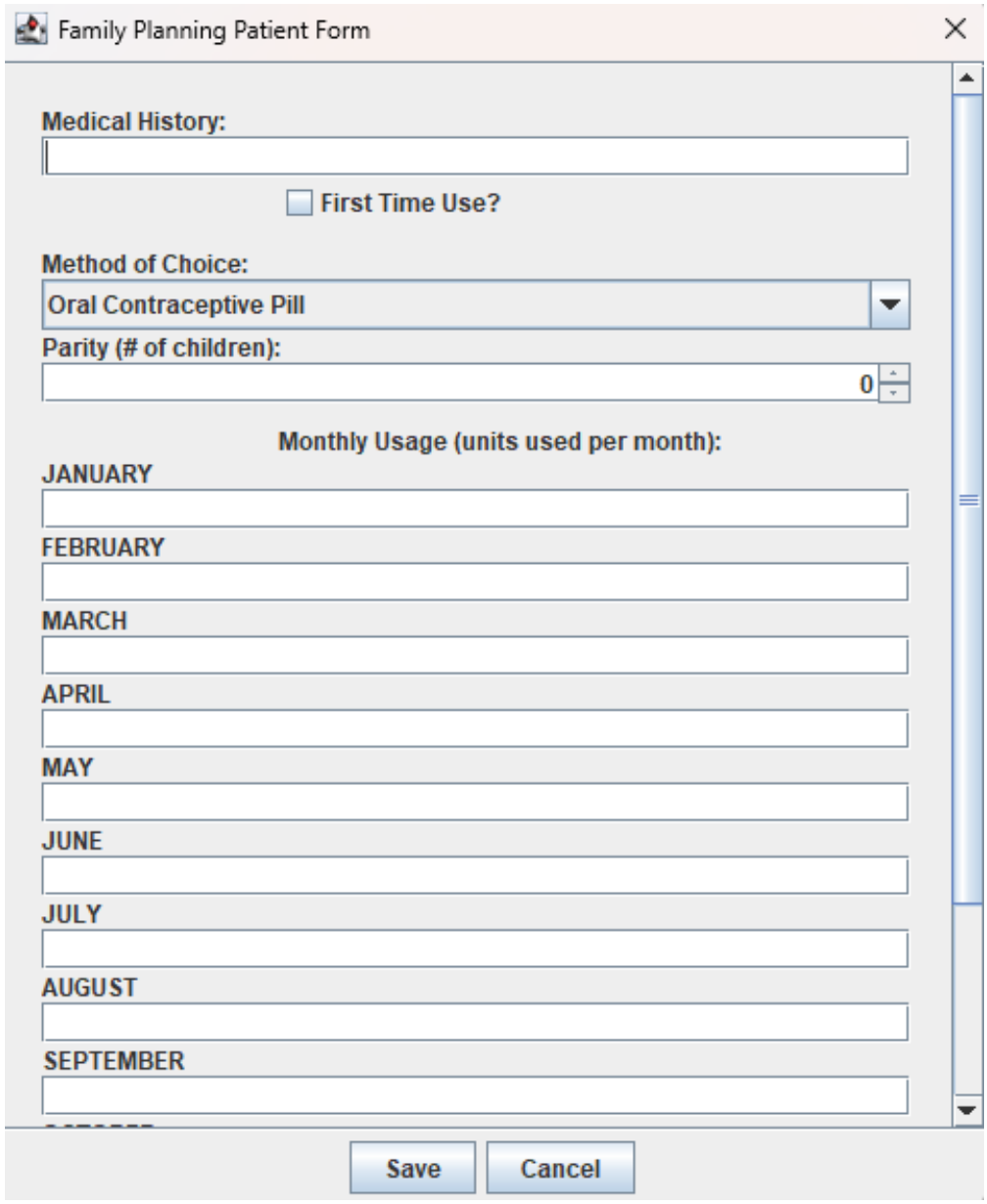


Figure 9: Creating a patient

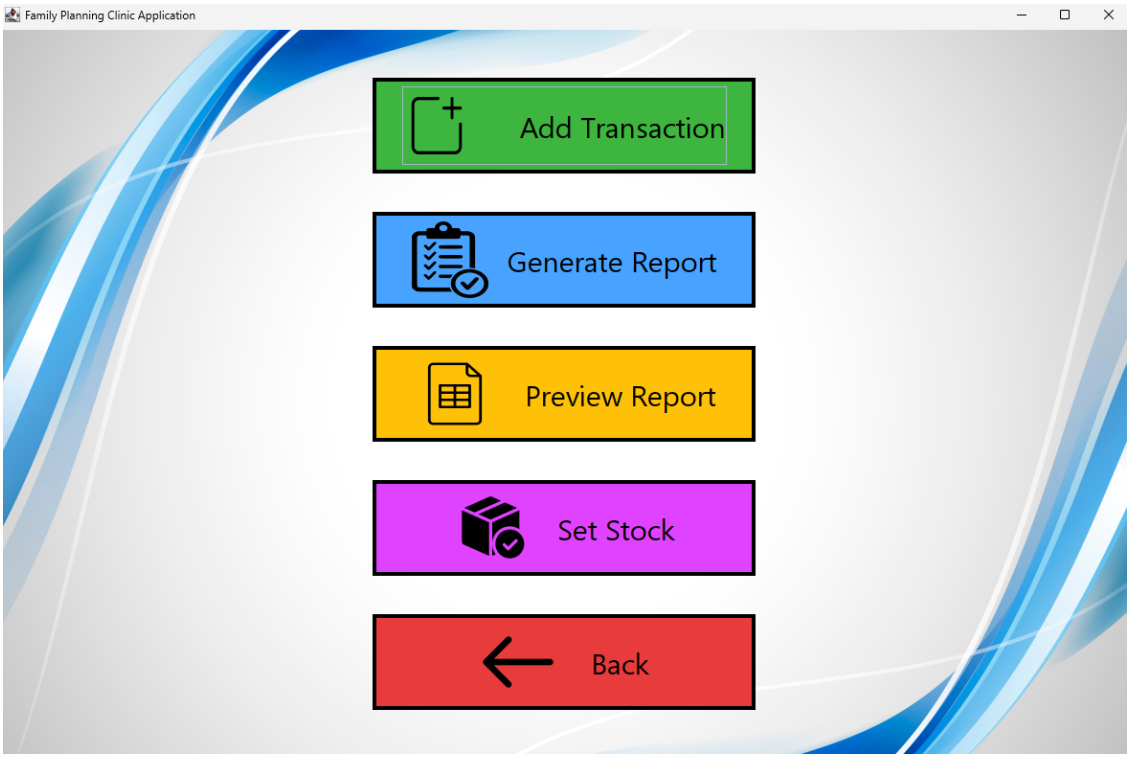


Figure 10: Inventory Section

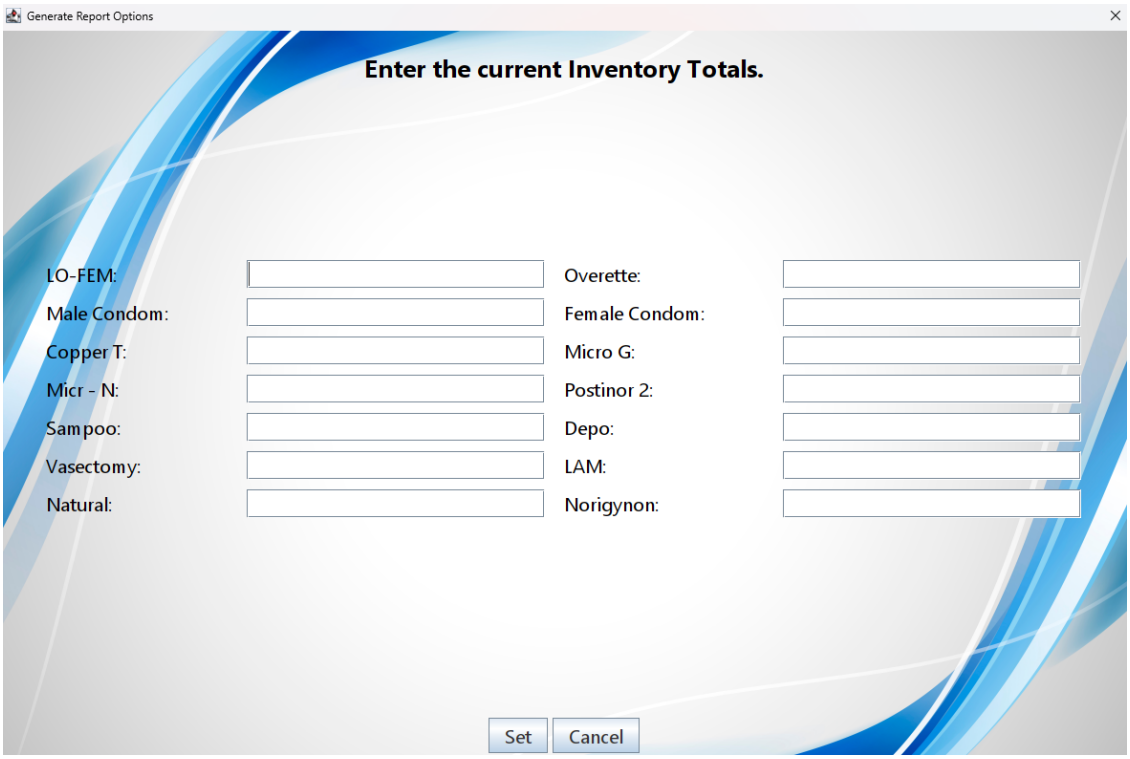


Figure 11: Setting the stock

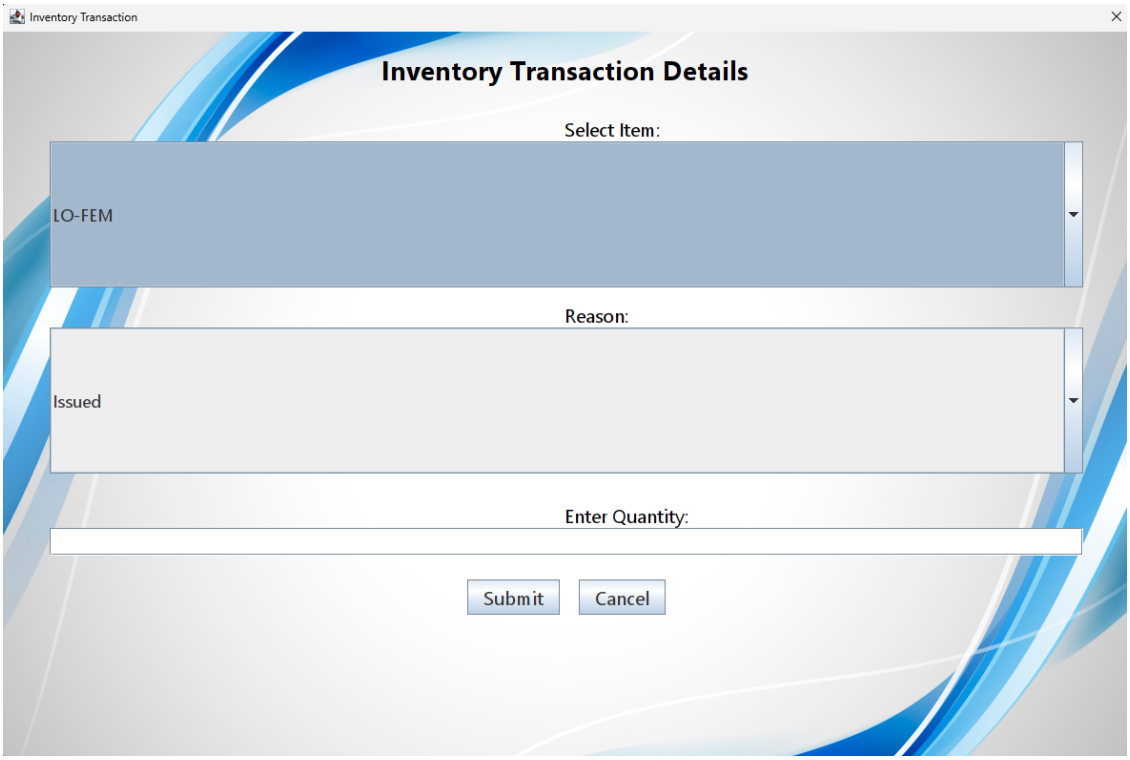


Figure 12: Change Stock Amount

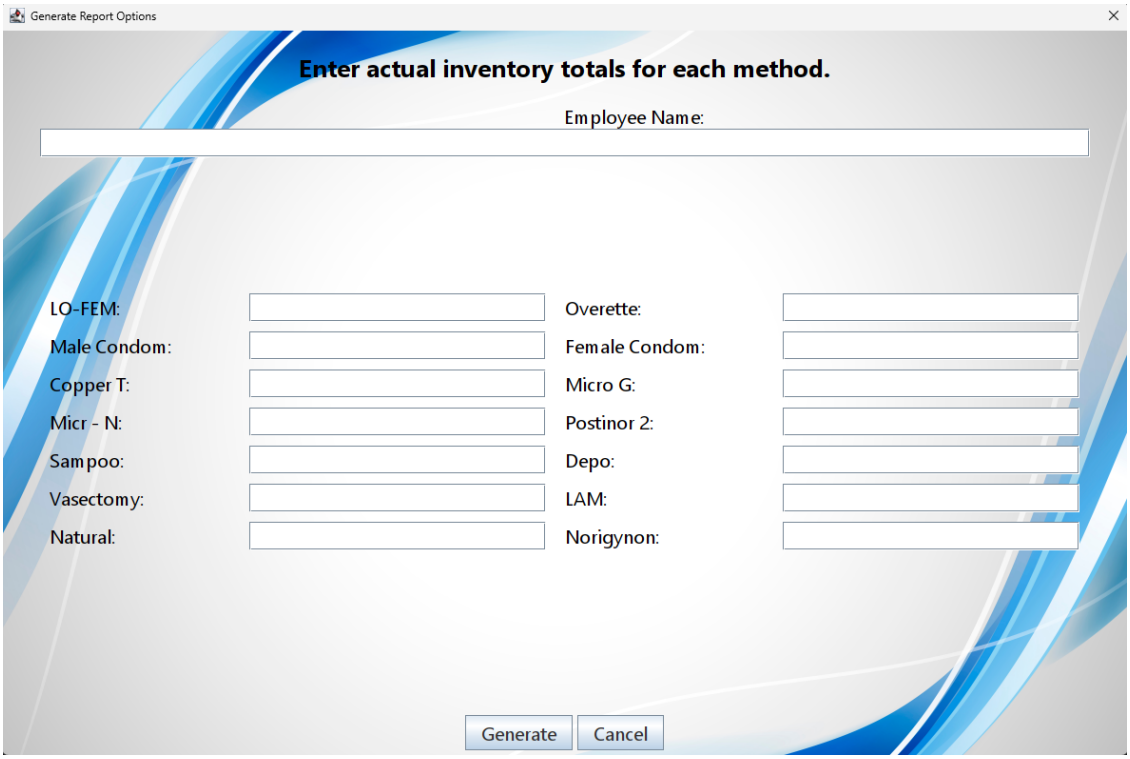
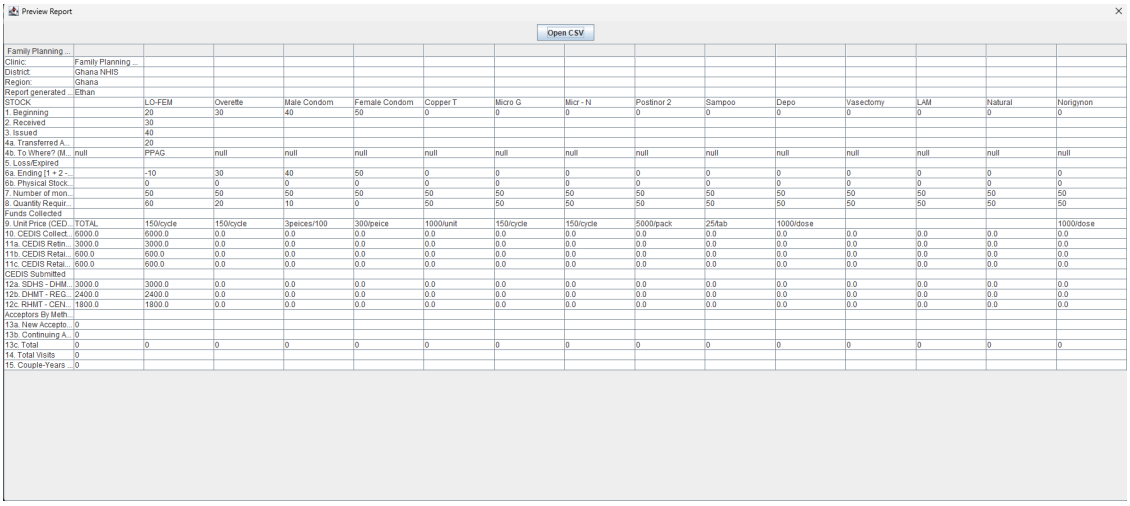


Figure 13: Generate monthly report



Family Planning ...															
Clinic:	Family Planning ...														
District:	Ghana NHIS														
Region:	Ghana														
Report generated ...	Ethan														
STOCK		LO-FEM	Overette	Male Condom	Female Condom	Copper T	Micro G	Micr - N	Postinor 2	Sampoo	Depo	Vasectomy	LAM	Natural	Norigynon
1. Beginning		20	30	40	50	0	0	0	0	0	0	0	0	0	0
2. Received		30													
3. Issued		40													
4a. Transferred A...		20													
4b. To Where? (M...	null	PPAG	null	null	null	null	null	null	null	null	null	null	null	null	null
5. Loss/Expired															
6a. Ending [1 + 2 -...		-10	30	40	50	0	0	0	0	0	0	0	0	0	0
6b. Physical Stock...		0	0	0	0	0	0	0	0	0	0	0	0	0	0
7. Number of mon...		50	50	50	50	50	50	50	50	50	50	50	50	50	50
9. Quantity Requir...		60	20	10	0	50	50	50	50	50	50	50	50	50	50
Funds Collected															
9. Unit Price (CED...	TOTAL	150/cycle	150/cycle	3peices/100	300/peice	1000/unit	150/cycle	150/cycle	5000/pack	25/tab	1000/dose				1000/dose
10. CEDIS Collect...	6000.0	6000.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
11a. CEDIS Retin...	3000.0	3000.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
11b. CEDIS Retai...	600.0	600.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
11c. CEDIS Retai...	600.0	600.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
CEDIS Submitted															
12a. SDHS - DHM...	3000.0	3000.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
12b. DHMT - REG...	2400.0	2400.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
12c. RHMT - CEN...	1800.0	1800.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Acceptors By Meth...															
13a. New Accepto...	0														
13b. Continuing A...	0														
13c. Total	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14. Total Visits	0														
15. Couple-Years ...	0														

Figure 14: Preview Report