WWU MAR

Garrett Moody, Hayden Tinker, Matija Benko, and Reece Reklai

Background

- Low cost alternative to commercial applications (EPIC)
- Verification of drug administration
- Simulation of real administration (Training)
- Inventory / Logs

Original Timeline vs Present Timeline

1. Add SSO, create Conduit App boilerplate, and create login page with SSO integration
2. Implement Conduit Endpoints
3. Create Patient Info/Medications Page
4. Create scanner page and add scanning functionality
5. Write tests to ensure scanner validates patient/medication
6. Purchase domains and host SQL database
7. Deploy both apps through 3rd party PAAS

Original Timeline vs Present Timeline

1. Add SSO, create Shelf App boilerplate, and create login page with SSO integration
2. Implement anticipated Shelf Endpoints (Login, User, Patient, Medication...)
3. Create Patient Info/Medications Page
4. Implement more Shelf Endpoints (Administration, Appointment, Allergy, Immunization...)
5. Create Appointment and Logs page and add scanning functionality
6. Write tests to ensure all aspects of the application are covered thoroughly
7. Get SSL certificate and a virtual machine
8. Setup nginx and deploy both apps on the virtual machine
9. Setup automated deployment
10. Maintenance (Respond to customer emails and fix new issues)

Git diff

Winter 2023:	Spring 2023:
Frontend ● - View a list of patients with basic attributes ● - Add and Delete patients from list ● - Create and remove medication from the inventory ● - Add or subtract medication stock amount from the inventory ● - Logs of the medication administration process were hard coded Backend ● - Authenticated endpoints through SSO ● - Generated session keys instead of JWT	Frontend ● + Update patient information ● + Create appointments and assign prescription ● + Create, remove, and assign allergies ● + Create, remove, and assign immunization ● + Update medication name from the inventory ● + Logout from the application ● + Allow scannability from barcode scanner ● + Display important information throughout the administration process (medicationID, patient dob, etc…) ● + Logs the entire medication administration process ● + Appointment updating to the status of the administration cycle Backend ● + Added a cache class to proxy handlers and the database smoothly. ● + Updated timestamp for session key on each request (expiration) ● + Added a logout link for Azure AD ● + Added automated deployment in the pipeline

Individual Contributions

Hayden:

- Implemented endpoints for Allergies, Immunizations, and Medication Routes.
- Documentation and tests.
- Collaborated with Reece on integration.

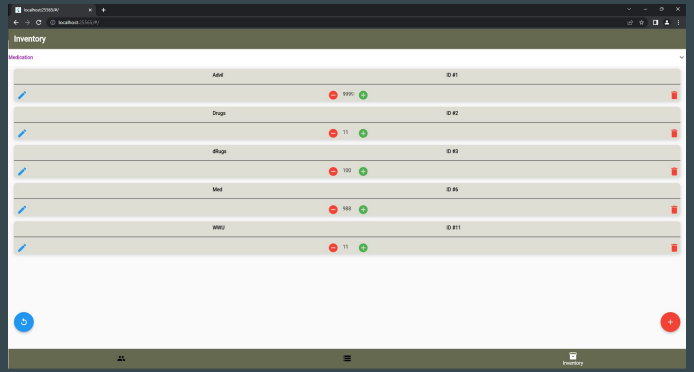
```
Handler get handler {
  final router = Router();
  router.get("/Immunization", (Request request) async {
    return getImmunization();
  });
  router.get("/Immunization/<id|[0-9]+>", (Request request, String id) async {
    return getImmunizationByID(id);
  });
  router.put("/Immunization", (Request request) async {
    return putImmunization(json.decode(await request.readAsString()));
  });
  router.post("/Immunization", (Request request) async {
    ///Handler for creating a new medication
    return postImmunization(json.decode(await request.readAsString()));
  });
  router.delete("/Immunization", (Request request) async {
    return deleteImmunization(json.decode(await request.readAsString()));
  });
  router.all('/<ignored|.*>', (Request request) {
    //Found this on stack overflow
    //https://stackoverflow.com/questions/74238216/dart-server-side-how-to-receive-data-from-the-postman
    return Response.notFound('Route not found');
  });
  return router;
}
```

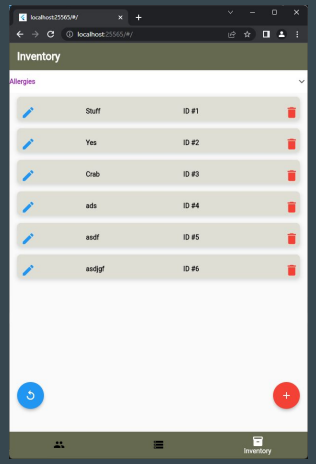
```
Future<Response> getImmunization() async {
  var fields = cache.getImmunizationFields();
  if (fields.length > 1) {
    return Response.ok(jsonEncode(
        {"Action": "get", "Immunization_information": fields.sublist(1)}));
  }
  return Response.ok(
      jsonEncode({"Action": "get", "Immunization_information": []}));
}
```

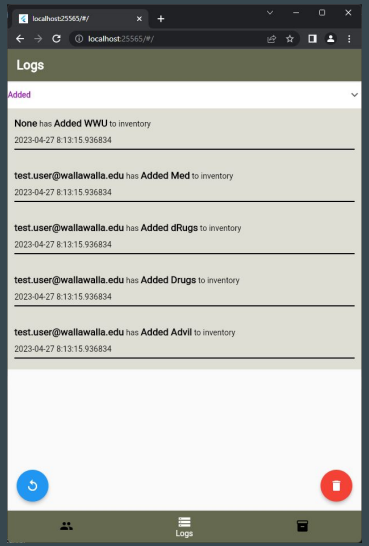
Individual Contributions

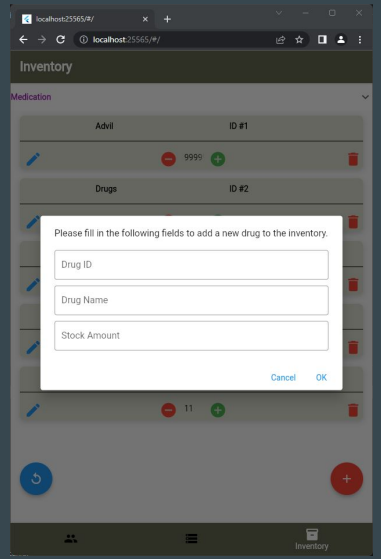
Reece:

- Implemented inventory_input_checker
- Implemented InventoryPage
  - Header
  - Body
    - Medication Inventory
      - Manipulates current medication, medications associated id, and current stock amount
    - Allergy List
      - Create and remove allergies in a list
    - Immunization List
      - Create and remove immunizations in a list
- Implemented LogPage
  - Logs the whole medication process
    - Returned, Administered, Checkout, Added, Wasted
    - Also logs the student, patient and patient's associated id
- Documentation, Unit Test, and Widget Test for all mentioned above





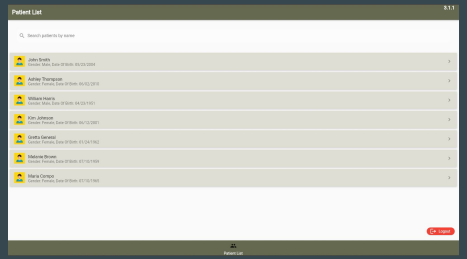


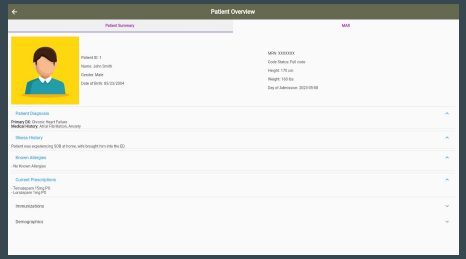


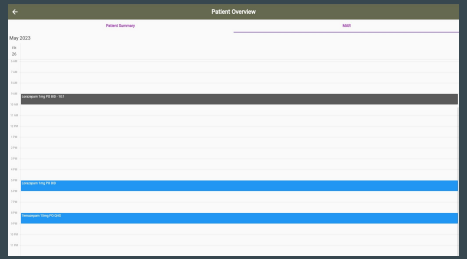
Individual Contributions

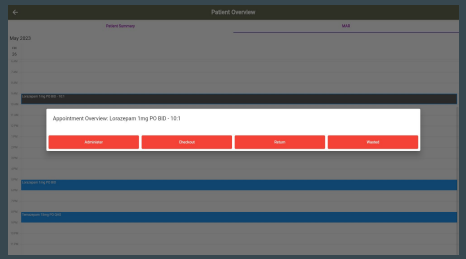
Matija:

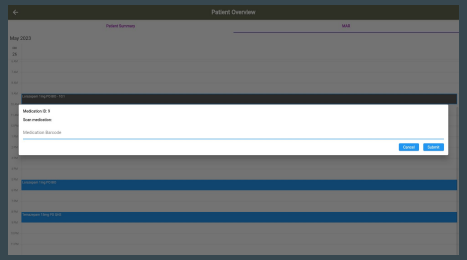
- Implemented AbstractApiClient, ApiClient, MockApiClient, Appointment_Input_Checker, Patient_Form_Checker, Patient_Info, and User classes
- Implemented LoginPage
  - Header
  - Body
- Implemented ProfessorPage & StudentPage
  - PatientListPage
  - PatientPage
  - Professor/Student NavigationPage
- Documentation, and Unit Test for all classes and pages mentioned above
- Implemented debugging Flag for local testing
- Implemented MainPage
  - Navigate the user to either professor or student

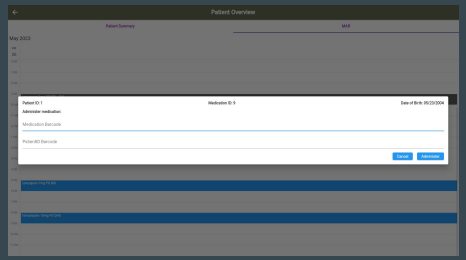


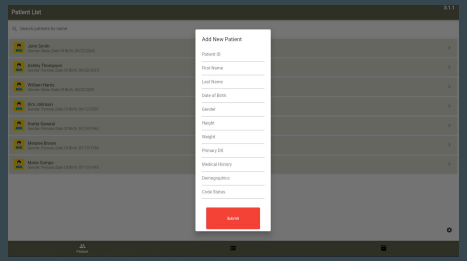


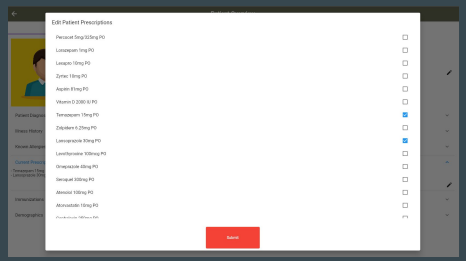








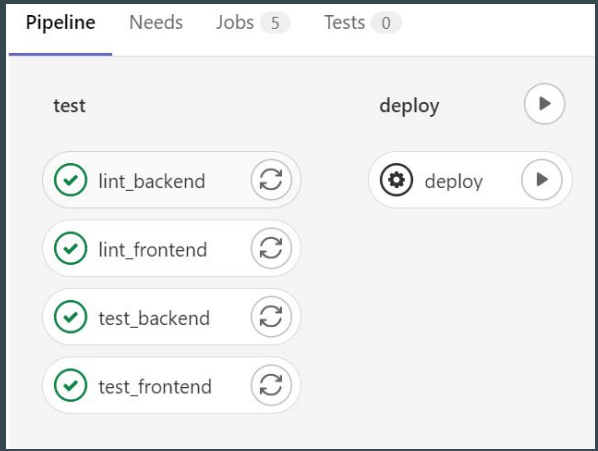




Individual Contributions

Garrett:

- Implemented Login and User Endpoints. Added auth middleware to check for valid session keys and implemented a whitelist.
- Created a cache class to serve as a single cohesive location for handlers to retrieve and write data.
- Worked with IT to set up app registration for SSO through Azure AD.
- Setup automated deployment and hosting on nginx for the frontend and backend.
- Modified several handlers to more accurately serve the needs of the frontend team throughout development.



Reflection

Changes

- Not using SQLite
- Reevaluating package usage for our calendar
- Redesigning the frontend layout
- Using session keys instead of JWT for authentication

Challenges

- Limited Shelf Documentation
- SSL Certificate
- Widget Testing
- Utilizing syncfusion_flutter_calendar package
- Hosting the application on a vm
- Making an automated deployment script

Future

- Have the cabinets connected to the application, unlock and lock depending on the administration action
- Integrate more functionality into our drug database, utilizing different metrics
- Convert the database from csv files to a more robust database management such as PostgreSQL
- Use a drug catalog with a query to allow quick selection of medication instead of manual addition
- Be able to checkout multiple prescriptions, instead of one at a time

Questions

...