* Introduction and Overview
  + This project does blah blah blah
* Show API requests, curl or Postman?
* Show GUI
  + Highlight changes made in UAT
  + Highlight exception handling / error message
* Demonstrate Unit and Integration tests, how to run?
* Explain how TDD was used
* Show GitHub Actions

[1] The aim of this project was to create an API which adds patient filtering logic to an external API data source, filtering patients by patients who have never been admitted, patients who have been re-admitted within a 7 day window of being discharged and patients who have been allocated multiple members of staff across admissions. It also has an endpoint to return the month with the most admissions.

[2] As well as the API, there was the task of creating a GUI front-end application, connecting to one of these endpoints; I selected the re-admission window endpoint.

[3] Another key factor in this task was the inclusion of high coverage testing, which has been measured using Code Coverage Metrics. Edge cases were considered when creating tests. The projects development largely followed a Test Driven Development approach to ensure   
code-correctness.

[4] The project was developed using Spring Boot for the API, and Java Swing for the GUI. JUnit and Mockito frameworks were used to create and manage the written unit and integration tests.

[5] Here I will use Postman to demonstrate some of the API’s endpoints.

[6] For the Part A Tasks, F1, F2 and F4, the API responds with a JSON list of patients matching the endpoints logic criteria.

[7] For Task F3, the API responds with a Date String, in a partial ISO-8601 format corresponding to the month with the most admissions.

[8] Here is the Graphical User Interface that acts as a front end to the API.

[9] In response to the Usability Testing survey, certain actions were carried out to improve the design and experience of the application, albeit simple.

[10] A title was added to the frame to improve the clarity of the purpose of the application. I also increased the font size in an effort to improve the accessibility of the application and made it so the table cells were not editable by overriding the default ‘isCellEditable’ method.

[11] Additionally, error handling was added. In the case that the API fails to return data, or the data cannot be parsed to patient objects, the user is informed of the failure by an Error Message Dialog.

[12] As mentioned before, unit and integration testing was managed by JUnit and Mockito. Unit Testing is used to ensure that ‘units’ of code work with expected behaviour in an isolated environment.

[13] Here are the Unit and Integration Tests running…:

[14] All tests pass, indicating that the logic for fetching data is working as expected.

[15] This also includes the integration tests, which verify that data is correctly passed between the API and Service Layer, to the Controller Layer which displays the data as a web service.

[16] Finally, I set up a GitHub Actions workflow to automate the testing process on every push to the repository. This ensures that all tests will run often, helping me to identify any integration or regression errors early.

[17] I used a VS Code editor extension ‘GitHub Local Actions’ to test this workflow before integrating it into the main repository.

[18] In summary, I have used a range of Software Development Tools and Practices in the development of this application, such as Test Driven Development, CI/CD tools with GitHub Actions, and I have used Unit and Integration Testing to ensure the quality of the software.

Thank you for watching.