

Web Dev

In Industry

Learning Objectives

- Gain an insight into what it's like to be a web developer in industry.
- Learn about the software development lifecycle and explore some tools used by development teams to organise themselves.
- Complete a web dev project based on a real-world problem

The background of the image features a series of concentric circles in a light gray color, centered on the left side and radiating outwards across the entire frame. The circles are closely spaced, creating a subtle, textured effect.

Working as a web
developer

Worknig at Mayden

- 2 years: placement and as a graduate.
- Web-based patient management system for psychological therapies used in the NHS (and, bizarrely, Australia).
- Part of a SCRUM team of "full-stack" developer -- PHP, MySQL, HTML, JavaScript and CSS.
- Also sooftware engineers: desesigning, planning, testing and releasing software.
- Often have to be detectives: identifying and fixing errors, figuring out hat old code does.

Case study: the diary project

- Add repeating appointments to the Diary
- Old code: user experience needs an overhaul, database schema needs updating.
- Opportunity to rewrite *some* of the system using modern technologies (react, REST API)
- But also have to compromise with time and cost.
- Clients expect the system to work the same way, even if its behaviour doesn't make sense.



How the customer explained it



How the Project Manager understood it



How the Architect designed it



How the Programmer wrote it



How the Business Consultant described it



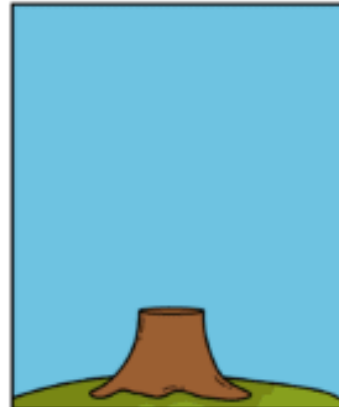
How the project was documented



What operations installed



How the customer was billed



How was it supported



What the customer really needed

Developer workflow

- **Planning** - break down tasks, decide on technologies, design database, API.
- **Code** - write the code, surprisingly the smallest part of being a Dev.
- **Code Review** - someone else checks over the code.
- **Test** - automated, manual, sometimes with stakeholders and users.
- **Release**



Planning

- Worked in 2 week sprints. Each team took a number of *stories* which they commit to finish.
- Stories to choose from were in priority order and requirements were already specified.
- Teams spent time *planning* each story by breaking it down into small tasks.
- If the task was complicated, we'd also spend time designing, e.g., database schema, API.

Create repeating appointments, front end (13 points)

As a therapist

I want to book repeating appointments

So that I can save time.

- Add a repeat button to the appointment booking form
- Options for daily, weekly and monthly repeat
- Show a table summarising the appointments to be booked
- Allow individual appointments to be modified from the table

Migrate
appointment
form to
React.

Add repeat
button.

Create repeat
options form.

Create
appointment
summary
table.

Delete
button on
table

Edit button
on table.

Edit single
appointment.

Coding

Pair programming

- *Driver* writes code, *navigator* explains what to do
- Swap roles around every half hour.
- Excellent for learning, as well as to design more efficient software.

Test Driven Development

- Write tests first, which will fail, then write code to make them pass.
- Tests define behaviour so you can match tests to requirements.
- Ensures good test coverage.

Code review

- Every line of code must be read by another developer.
- Helps spot mistakes, oversights or edge cases.
- Ensure the code is easy to understand.
- Getting and giving feedback helps both parties become better developers.

Testing and QA

- Automated tests.
- Manual tests in *multiple browsers* and on *different screen sizes*.
- Test on a replica of the live server with real data.
- Walk through the new features with a stakeholder.
- For larger projects, customers may be willing to Beta test.

Release

- More experienced developers put the code on the live server.
- Different types of release: `hotfix`, `feature`.
- Usually outside of working hours.
- Release team make sure everything worked and must fix any issues if the release goes wrong.

Organising Teams

the development workflow in large projects.

Tracking Tasks

- Devs often have a large whiteboard with sections for *"To do"*, *"In progress"*, *"Code Review"*, *"QA / Sign Off"** and *"Done"*.
- Often a system like *Jira* is used, which tracks the status of each story and organises releases.
- Large projects always use version control, typically Git integrated with GitHub or GitLab.
- All this gives visibility to everyone involved in the development process.

Git Branches

- Repo's `main` branch contains the production code.
- Each feature typically has its own branch, e.g., `feature/53/repeating-appointments`.
- When a feature is finished, it should be *merged* into the `main` branch.
- Actually, multiple `features` are merged into a `release` branch, and this is merged into `main`.

Pull requests

- In order to merge a branch, the developer opens a *pull request* on GitHub.
- Other developers are asked to comment on the pull request during code review.
- A member of the release team will *approve* the pull request, then the feature can be merged.

Case Study

Online appointment booking

The Patient Portal

The customer has requested that patients have the ability to book their own appointments online. It should work as follows:

- Therapist sends an SMS to the patient with a booking link.
- Patient clicks the link and is shown a web page with available appointments (date, time, location).
- The patient can select one appointment and will be presented with a confirmation screen.
- Once booking is successful, the appointment is added to the Therapist's diary and an appointment confirmation SMS is sent to the patient.

The Patient Portal should

- Be a separate system from the main Patient Management System (but may need to communicate with it).
- Be simple to use, so that patients are not overwhelmed or discouraged from using it.
- Be accessible to all users (think colours, screenreaders, but also the fonts and language used).
- Work on multiple screen sizes (research shows users prefer to complete bookings and purchases on a PC, even if they started on their phone).

Your Task

In groups, pairs or individually, plan out the Patient Portal.

- **Flow-chart:** create a flow-chart showing how the system will work and how the Patient Portal will interact with the main Patient Management System.
- **Write User Stories**
 - Use the template *"As a ..., I want ..., So that ..., Requirements"* to specify requirements.
 - Think about dependencies (*e.g., confirmation page depends on booking page*).
 - Think about priorities (*e.g., allowing the client to customise the style of their page is less important than getting the system working*).

At the end of the session, please showcase your work!


Patient Appointment Booking: SMS (2 points)

As a therapist

I want to send an appointment booking link to my patient

So that I can save myself time and give my patient more control and flexibility over their appointment.

- Admin setting to enable patient appointment booking links via SMS.
- button to the patient's page to trigger an appointment booking SMS.
- Confirmation dialog showing the message to be sent and patient's contact number.
- confirmation message to the user that a message was sent.
- Log on the patient's record that the message was sent.



Present Your Work

Please share your
flowcharts and stories

The background of the slide features a series of concentric circles in a dark blue color, creating a ripple effect that originates from the top-left corner and spreads across the entire frame.

Questions?