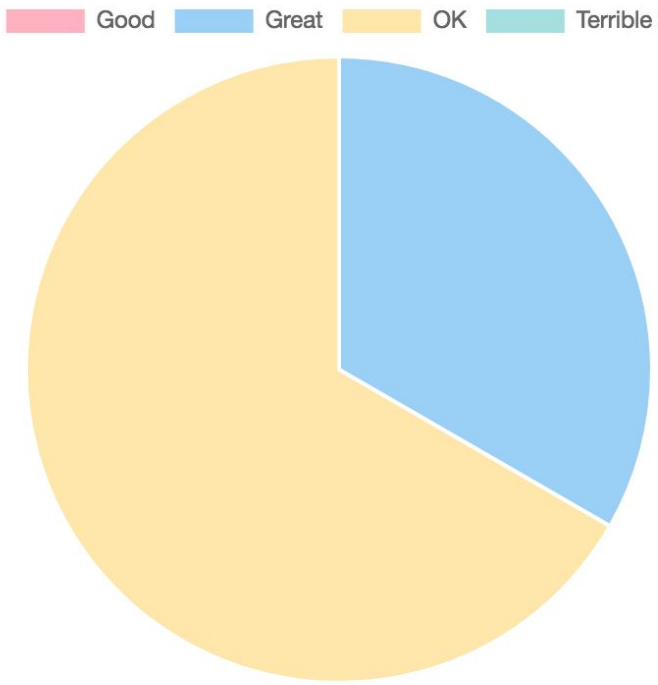


Web-based Course Feedback Survey System – Final Report

Team Monty:

- Dan Kennedy - percentage contributed: 38%
- Jacob Wahib - percentage contributed: 10%
- Karl Nicolas - percentage contributed: 50%
- Paul Zhu - percentage contributed: 2%

Signed:

User Stories

Key

Priority: (from highest to lowest) Essential, Desirable, Optional, Future

1 Story Point = 2.5 hours of work

Epic stories

Name: *Gather Course Feedback*

Description: As a ***course administrator***, I want to *survey the attitudes of students to particular aspects of a course offering* so that *I can improve future course offerings based on that feedback.*

User stories: Z2 (create questions), Z3 (create surveys), Z10 (admin index page)

Name: *Review surveys*

Description: As a ***staff member***, I want to *review surveys created by course administrators* so that *I can make sure they reflect the course I am teaching.*

User stories: Z8 (staff index page), Z11 (review questions)

Name: *Respond to surveys*

Description: As a ***student***, I want to *be able to respond to surveys for courses that I am enrolled in* so that *I can improve future course offerings based on that feedback.*

User stories: Z5 (fill surveys), Z9 (student index page)

Name: *Maintain system integrity*

Description: As a ***system administrator***, I want to *control who can access data within the system and maintain the integrity of that data* so that *all information around the surveys is accurate.*

User stories: Z1(login page), Z6 (store responses), Z12 (authentication module), Z13 (store user info), Z14 (store course info) Z15 (store questions)

User stories

ID: *Z1*

Name: *Login page*

Description:

As a *system administrator*, I want to be able to *present an initial login page for the site* so that *users can be filtered by their permissions level and unauthenticated users are denied access.*

Acceptance Criteria:

- Primary user interface of the module includes at least the following: 2 text boxes that allows staff/students/admins to input their name and password, as well as a button that to initiate the login process.
- If the login is unsuccessful, displays details of why it was unsuccessful (User is not registered, password does not match etc) and allow user to try again.

Priority: Essential

Story Points: 2

ID: *Z2*

Name: *Create questions*

Description:

As a *admin user*, I want to be able to *create generic questions in the system for students to answer*, so that *I can later select them for use in surveys relating to particular course offerings*

Acceptance Criteria:

- The survey questions should have multiple type options e.g., multiple choice questions, text-based.
- Admin is able to enter strings into text boxes
- The questions are saved and are able to be accessed between logins.
- All staff have access to a single pool of generic questions which they can contribute to.

Priority: Essential

Story Points: 4

ID: *Z3*

Name: *Create surveys*

Description:

As an *admin user*, I want to be able to *create surveys in the system for students to answer*, so that *I can better facilitate the flow of feedback from the students to course coordinators.*

Acceptance Criteria:

- Admin is presented with the pool of generic questions
- Questions are able to be selected for each course offering.
- When the survey creation is done, the survey is saved in the system, corresponds to the right course, and is accessible by a student user
- We assume the admins will be honest in their creation of surveys in terms of only making relevant surveys

Priority: Essential

Story Points: 3

ID: *Z5*

Name: *Fill Surveys Module*

Description:

As a *student*, I want to be able to *answer surveys created by staff members and have my results processed* so that *I can give feedback to the courses I've taken.*

Acceptance Criteria:

- Facilitates different kinds of inputs such as radio buttons for multiple choice questions and text boxes for qualitative responses.
- Each question selected by the administrator is presented to the student
- Students can leave questions unanswered
- Students can submit answers once the survey is completed
- Students can answer the survey ONLY ONCE.

Priority: Essential

Story Points: 3

ID: *Z6*

Name: *Store responses*

Description:

As an *system administrator*, I want *the survey responses to be stored in a robust, reliable back-end system* so that *in the event of any crashes, data is not lost.*

Acceptance Criteria:

- Answers to multiple choice questions are stored in an SQLite3 database.
- Submitted answers update the database.

Priority: Essential

Story Points: 2

ID: *Z7*

Name: *Present data*

Description:

As an *admin*, I want *the information input by students to be presented in an easily viewable form* so that *I make decisions regarding future course offerings quickly and easily.*

Acceptance Criteria:

- The information stored in the backend is summarised and presented on the frontend in a way that is easy for a human to understand. This should include summaries in text form and visual graphs.
- This data should not be visible to students or staff by default, until the survey is closed
- Admins can access updated total survey results once submitted by students

Priority: Essential

Story Points: 3

ID: *Z8*

Name: *Staff Index page*

Description:

As a *staff user* I want to be able to *see the features available to me within the system* so that *I can navigate to them easily.*

Acceptance Criteria:

- On login the assigned list of course-offerings with associated surveys in “review” stage is displayed on the staff member’s dashboard.

Priority: Essential

Story Points: 1

ID: *Z9*

Name: *Student Index page*

Description:

As a *student user* I want to be able to *see the features available to me within the system* so that *I can navigate to them easily.*

Acceptance Criteria:

- The survey system index page displays a student dashboard showing the list of course-offerings that the student is currently “enrolled” in and associated with a “open” survey

Priority: Essential

Story Points: 1

ID: *Z10*

Name: *Admin Index page*

Description:

As an *admin user,* I want to be able to *see the features available to me within the system* so that *I can navigate to them easily.*

Acceptance Criteria:

- On successful login, an admin user is shown the correct user interface which allows them to create surveys and add/delete questions from a pool of generic questions.

Priority: Essential

Story Points: 1

ID: *Z11*

Name: *Review questions*

Description:

As a *staff user,* I want to be able to *review the surveys in the system* so that *I can customise them for my students.*

Acceptance Criteria:

- A staff can click on a particular course-offering. This takes them to the survey form to be reviewed.

- The staff cannot make any changes to the generic questions.
- However, they can add/delete one or more optional questions.
- Once they have reviewed the survey, the changes are saved.

Priority: Essential

Story Points: 4

ID: *Z12*

Name: *Authentication module*

Description:

As a *system administrator*, I want to be able to *control who accesses the survey system and for what reasons* so that *survey information is accurate.*

Acceptance Criteria:

- The authentication module is provided with a csv file “passwords.csv” that will contain the zIDs, passwords and role (“staff or student”) for all users (student and staff).
- The authentication module will import these security details into the database. Additionally, the authentication module will store the authentication details for the admin user in the database.
- A student is able to log into the survey system using their zID and password.
- An admin user is able to log into the survey system with an admin username and password.
- A staff can log into the system using their staff-id and password.
- The information input by the user will be checked against the database.
- Passwords are stored in hashed form

Priority: Essential

Story Points: 4

ID: *Z13*

Name: *Store user info*

Description:

As a *system administrator*, I want *the user information to be stored in a robust, reliable back-end system* so that *in the event of any crashes, data is not lost.*

Acceptance Criteria:

- User info is stored in an SQLite3 database.

- Password is stored in hashed form
- Courses a user is taking are stored
- Type of user (admin, student, staff) is stored
- Surveys a user can fill out are stored

Priority: Essential

Story Points: 2

ID: *Z14*

Name: *Store course info*

Description:

As a *system administrator*, I want *the course offering information to be stored in a robust, reliable back-end system* so that *in the event of any crashes, data is not lost.*

Acceptance Criteria:

- Course info is stored in an SQLite3 database.
- Course name is stored
- Semester the course is offered is stored
- Students who are enrolled in that course are stored

Priority: Essential

Story Points: 2

ID: *Z15*

Name: *Store questions*

Description:

As a *system administrator*, I want *the questions created by admins to be stored in a robust, reliable back-end system* so that *in the event of any crashes, data is not lost.*

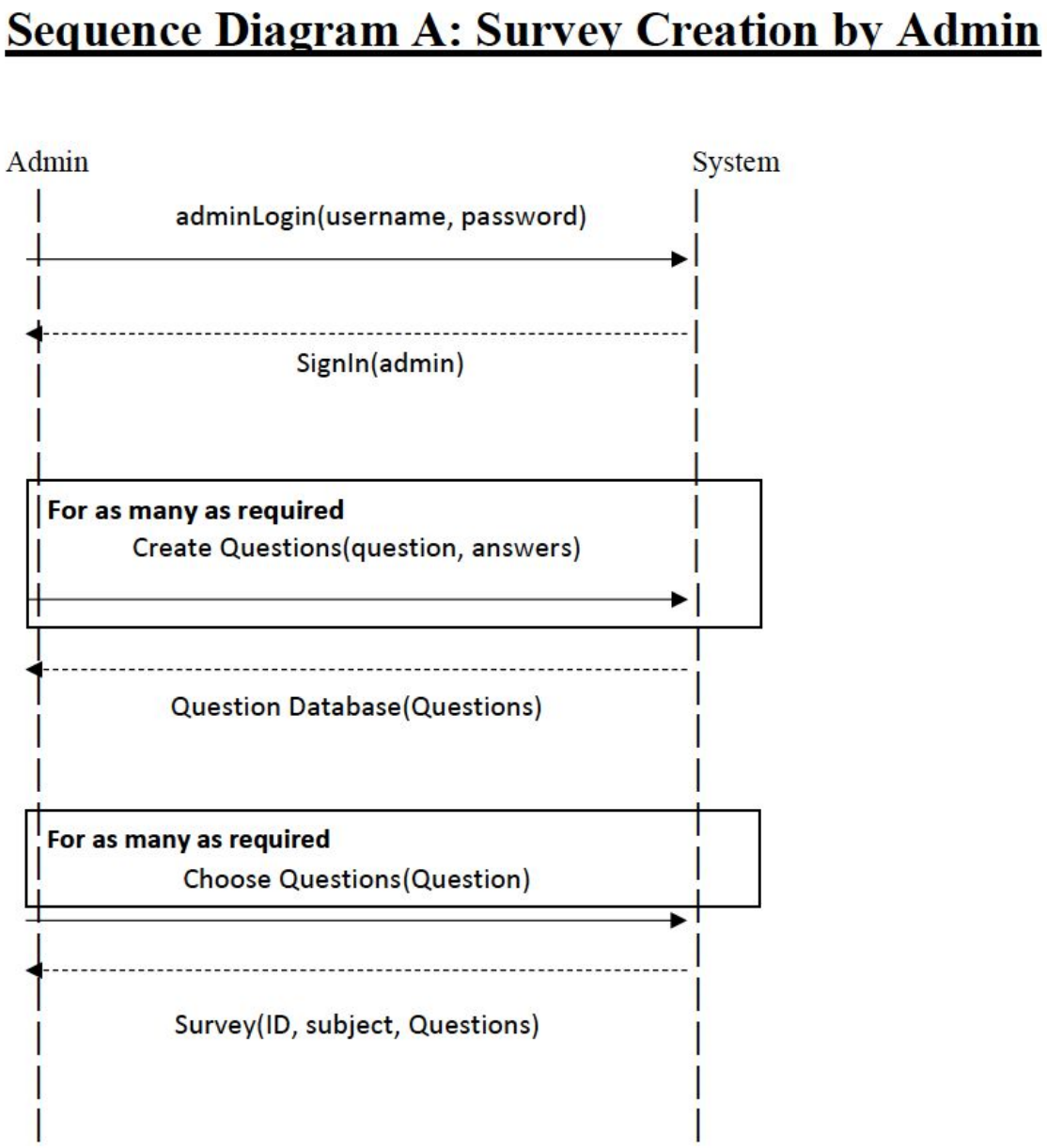
Acceptance Criteria:

- Questions are stored in an SQLite3 database.
- Multiple choice questions and open ended questions are distinguished

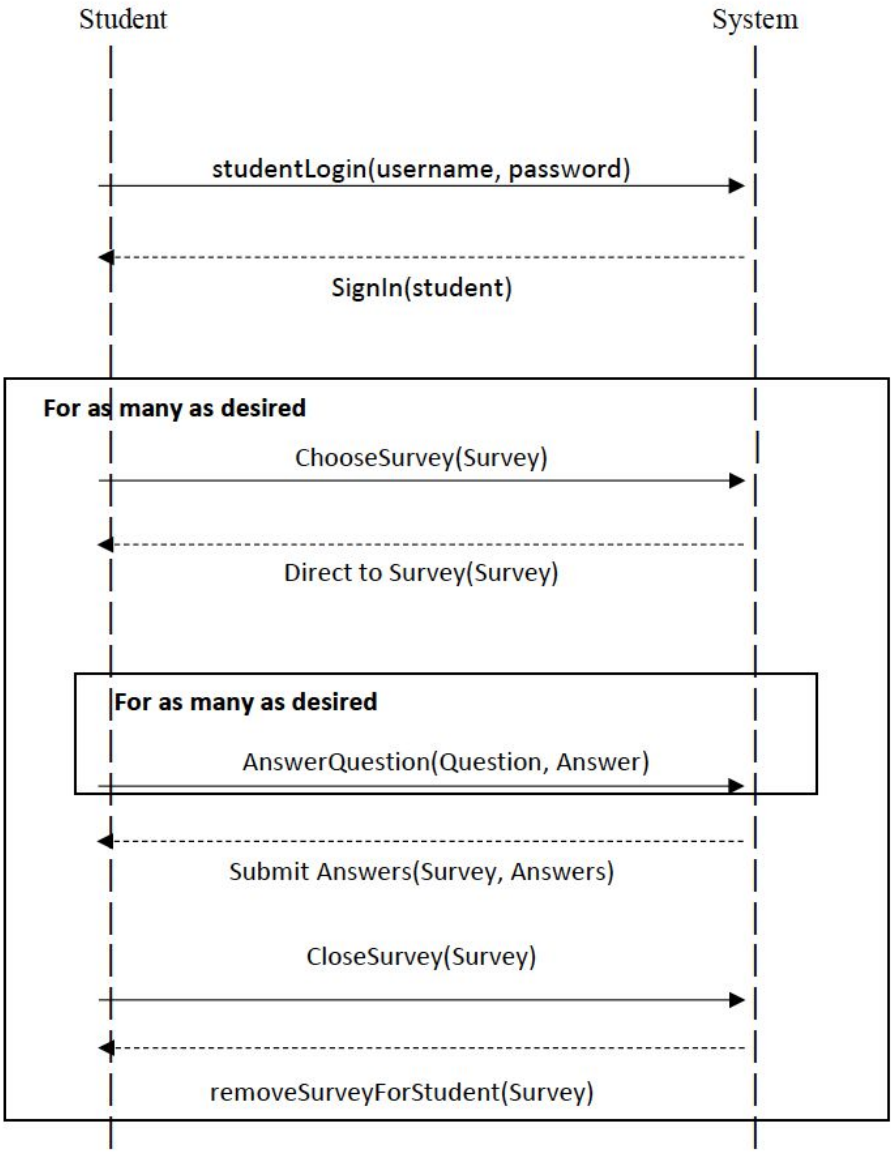
Priority: Essential

Story Points: 2

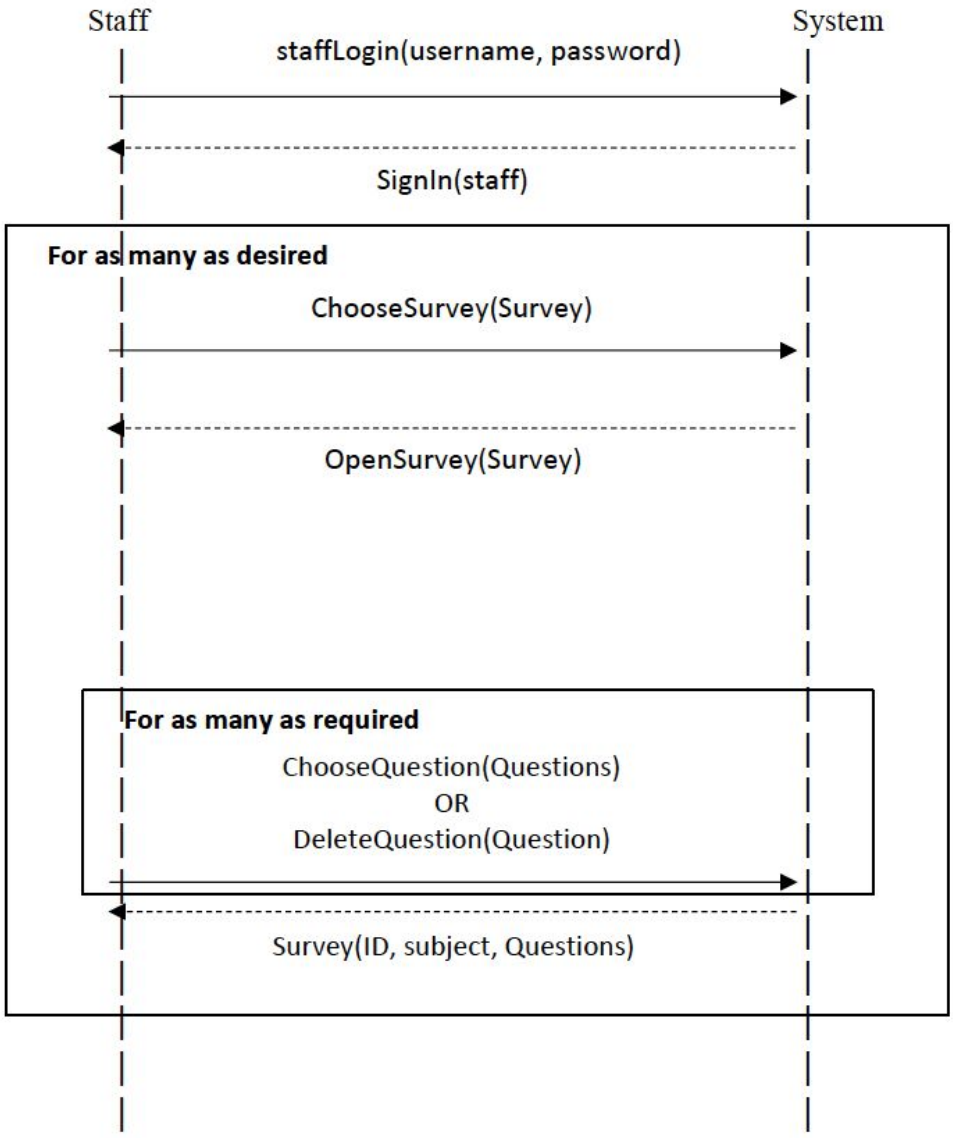
Sequence diagram



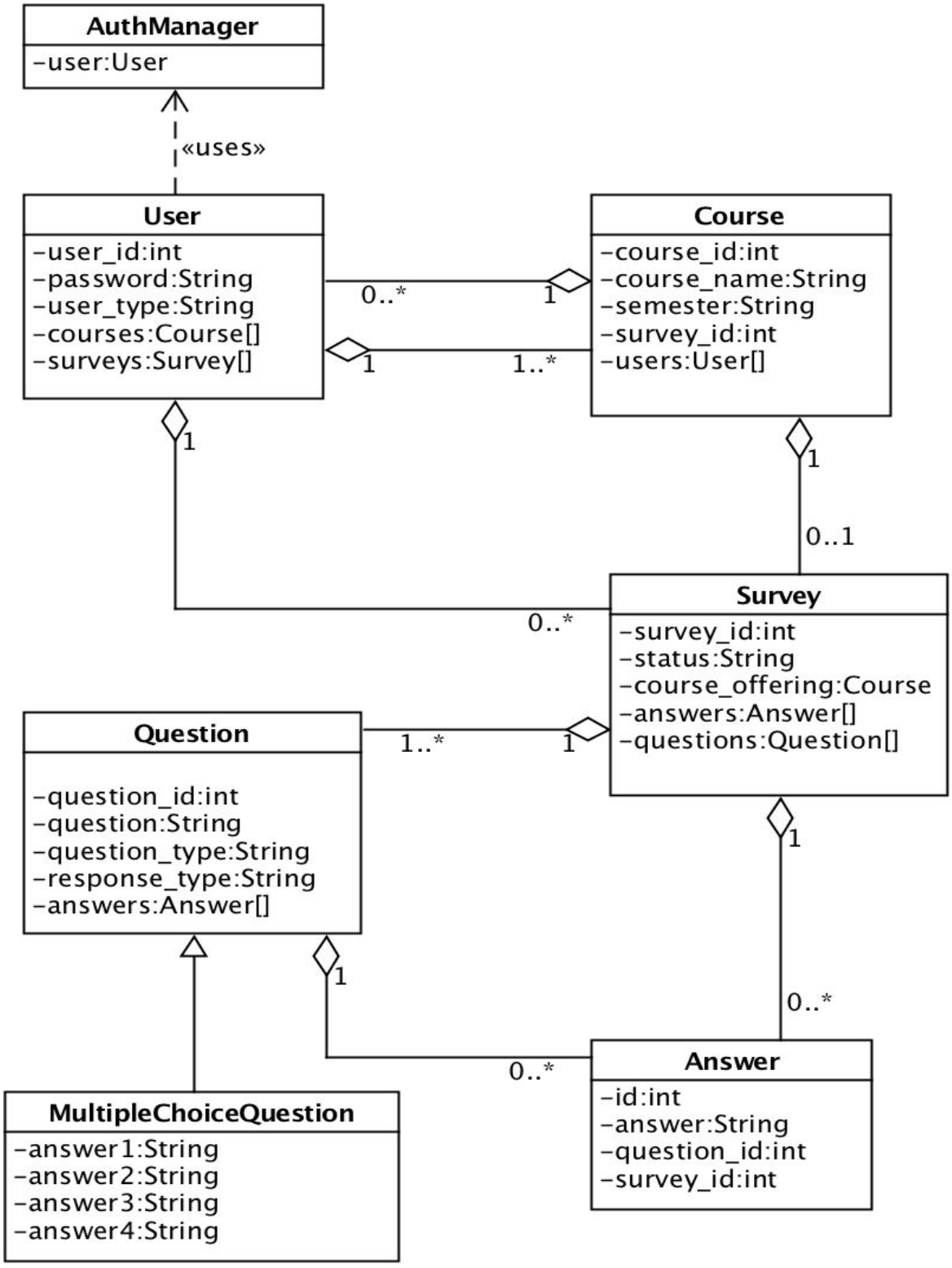
Sequence Diagram B: Survey Answered by Student



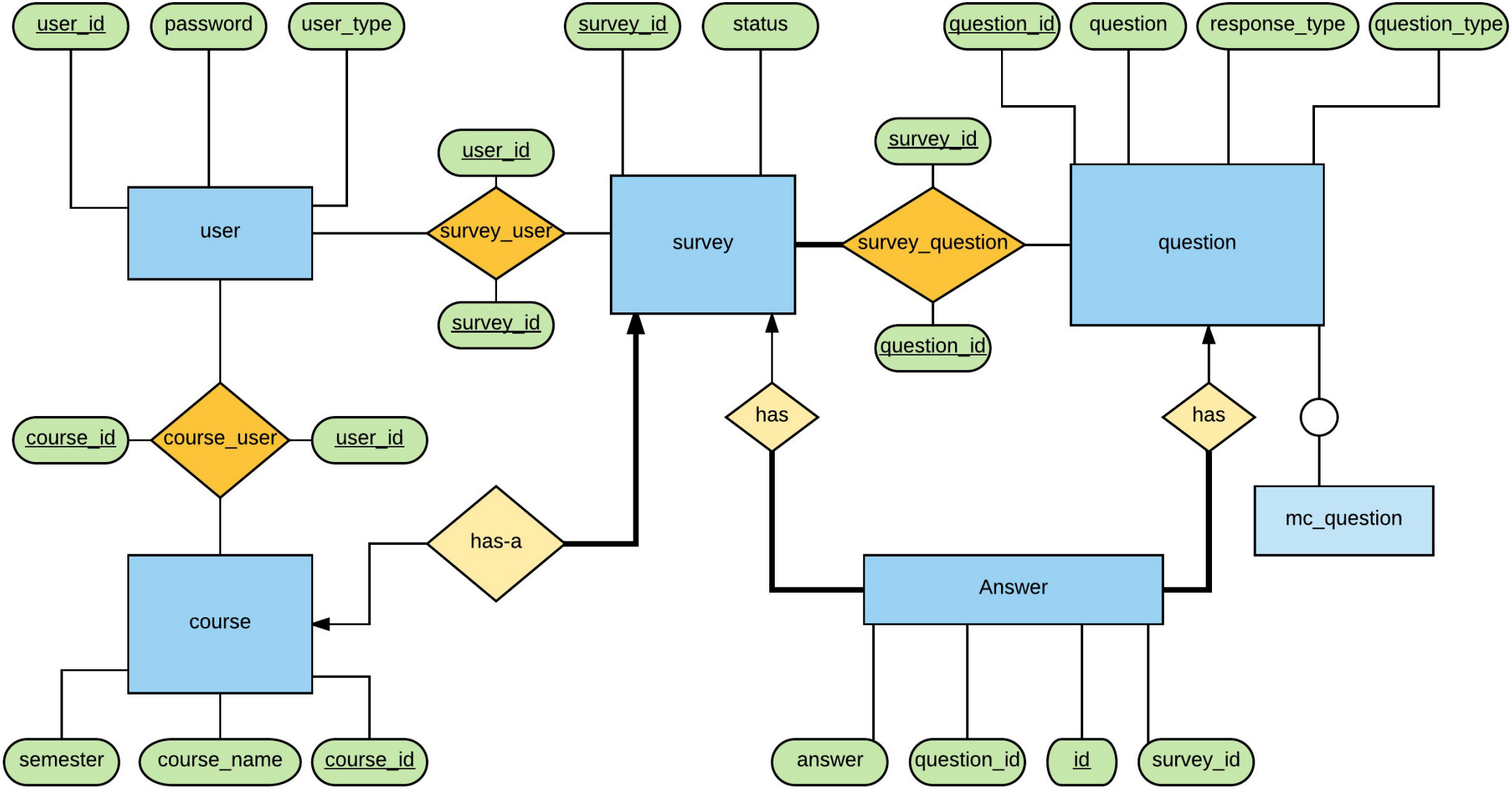
Sequence Diagram C: Survey Reviewed by Staff



Conceptual class diagram

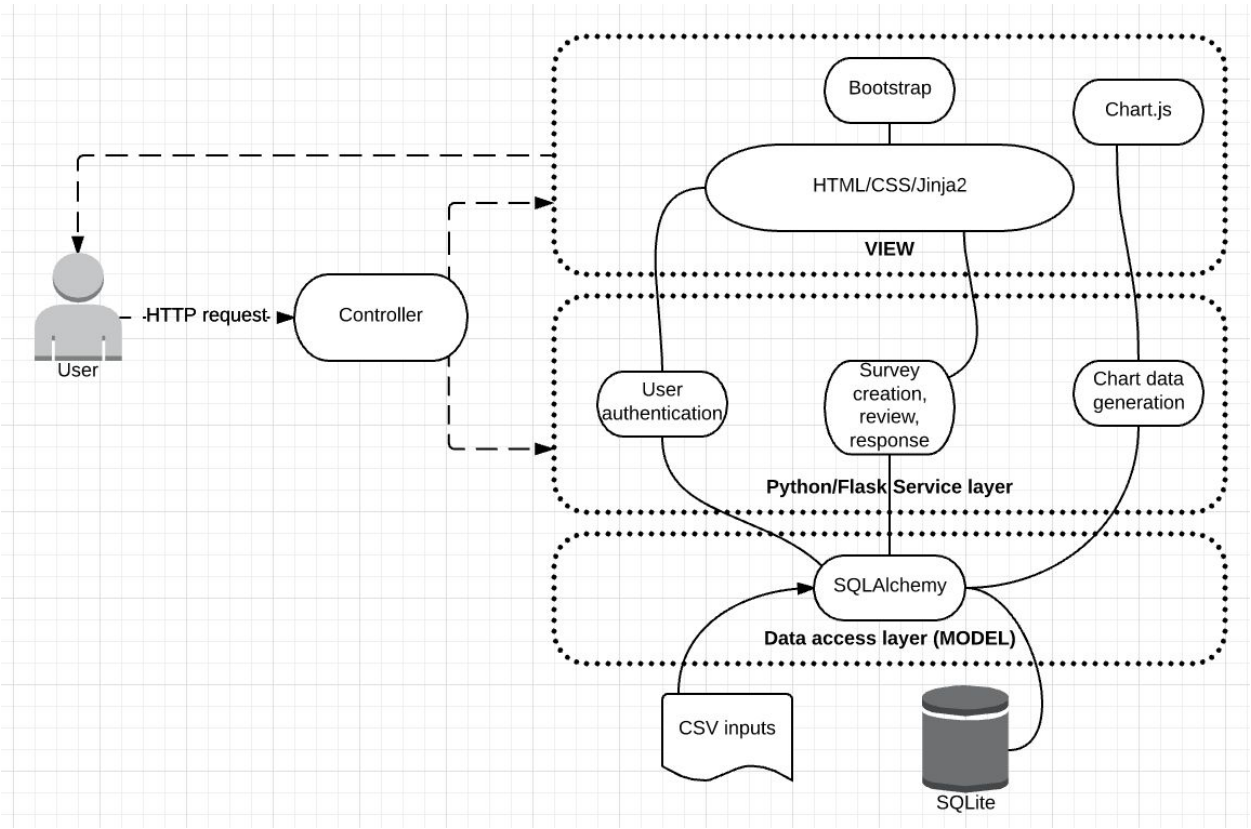


ER Diagram


user_id
password
user_type
user
survey_id
status
survey
question_id
question
response_type
question_type
user_id
survey_user
survey_id
survey_id
survey_question
question_id
question
course_id
course_user
user_id
has
has
mc_question
has-a
course
Answer
semester
course_name
course_id
answer
question_id
id
survey_id

Architecture diagram

A diagram that identifies the software architecture (key components and their interactions) of the system



Reflection

A reflection of your implementation and what would your team do differently if you had to reimplement this application

- Using a test-driven development approach earlier in the software development life cycle is one thing we would do differently, as it was hard to write “unbiased” test due to the code already existing.
- Overall, the implementation turned out quite well as team members had enough expertise amongst them to implement various parts of the product (backend + frontend)
- On the frontend, there could be some tweaks like taking all styling out of the HTML and using an interesting template, but again overall the frontend seems relatively clean.

Final Project Log - Monty Online survey system

Team Monty: Final practical roles

- Karl Nicolas - Backend dev, coach
- Dan Kennedy - Frontend dev, data visualisation, documentation
- Jacob Wahib - Testing, object diagrams, customer
- Paul Zhu - Ideation

Standups

August 8

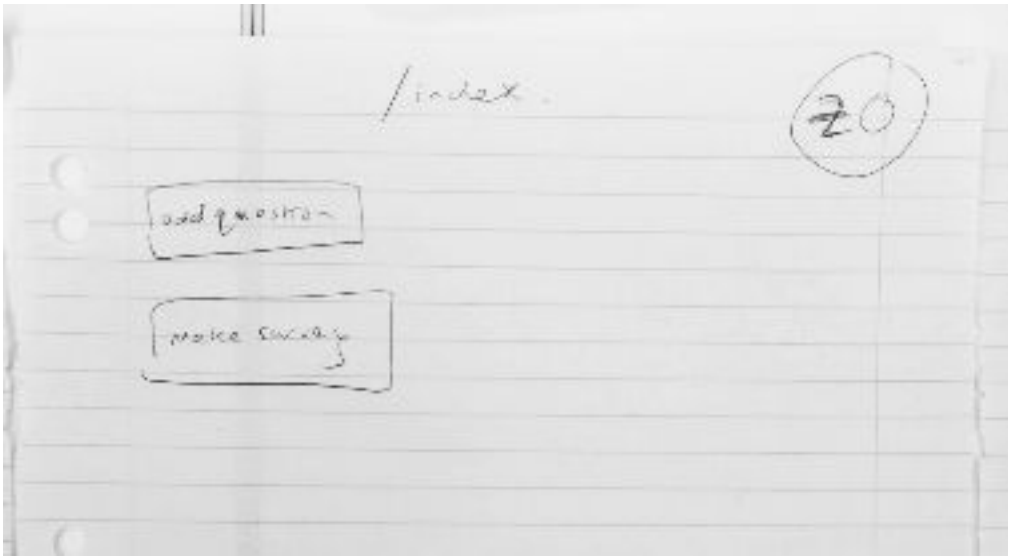
First discussion of team roles. Karl Nicolas: Coach, Dan Kennedy: Customer, all : programmers. Discussed the “epic story”, focussing the customer’s overall goals for the project.

August 15

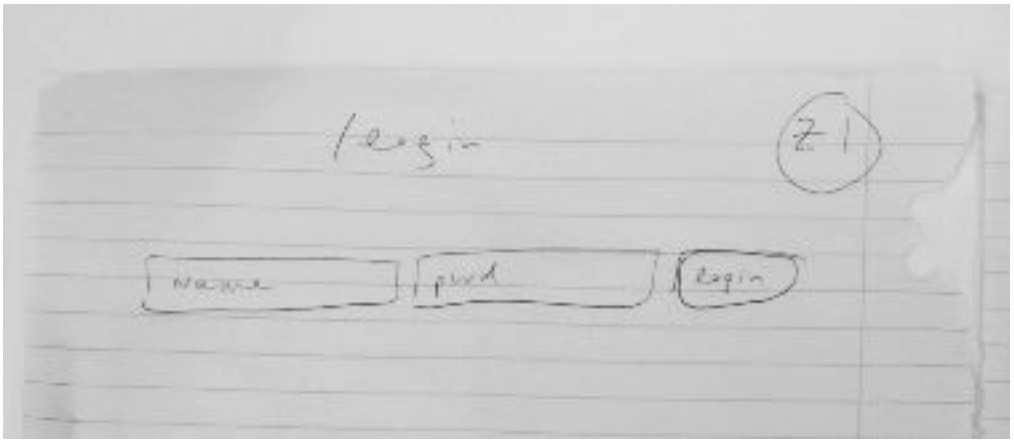
Defined initial list of user stories: *Index page, Login module, Create questions, Create surveys, Share survey, Fill Surveys Module, Store responses, Present data.*

August 22

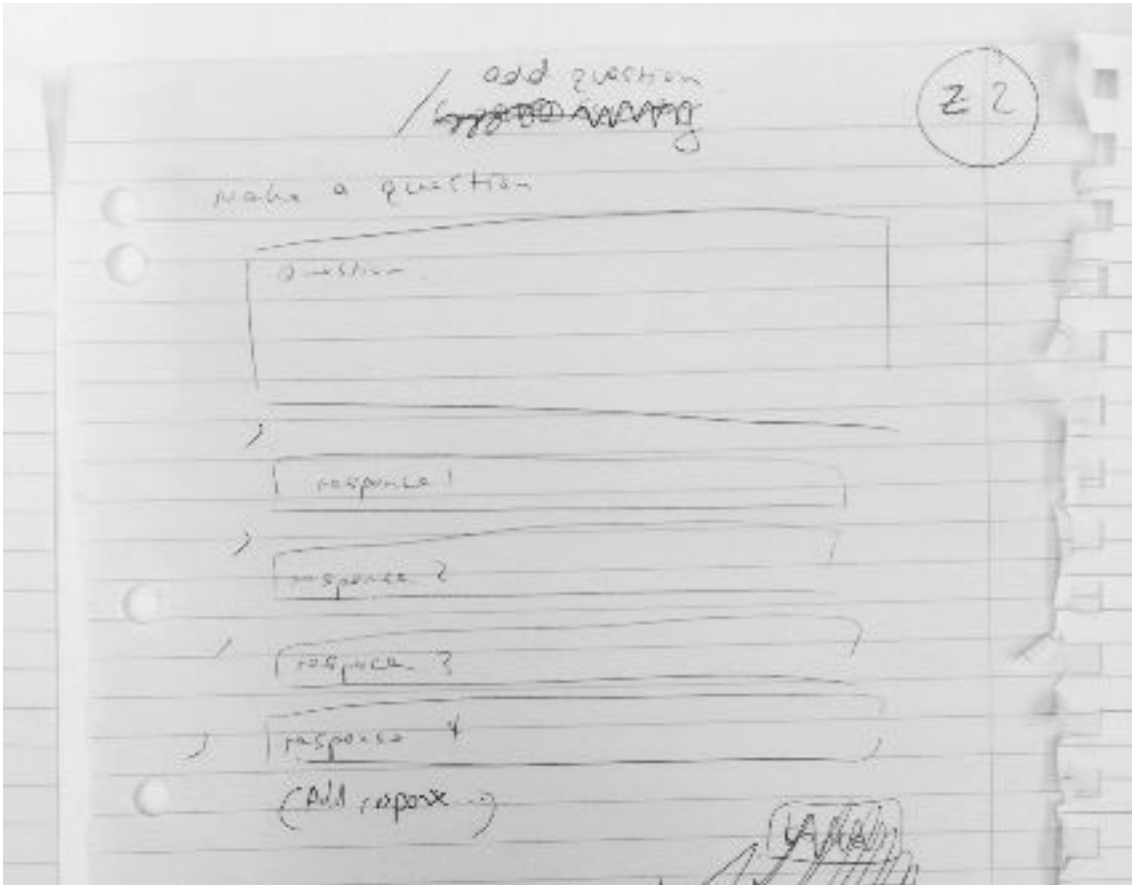
Assigned team members user stories for the first iteration:



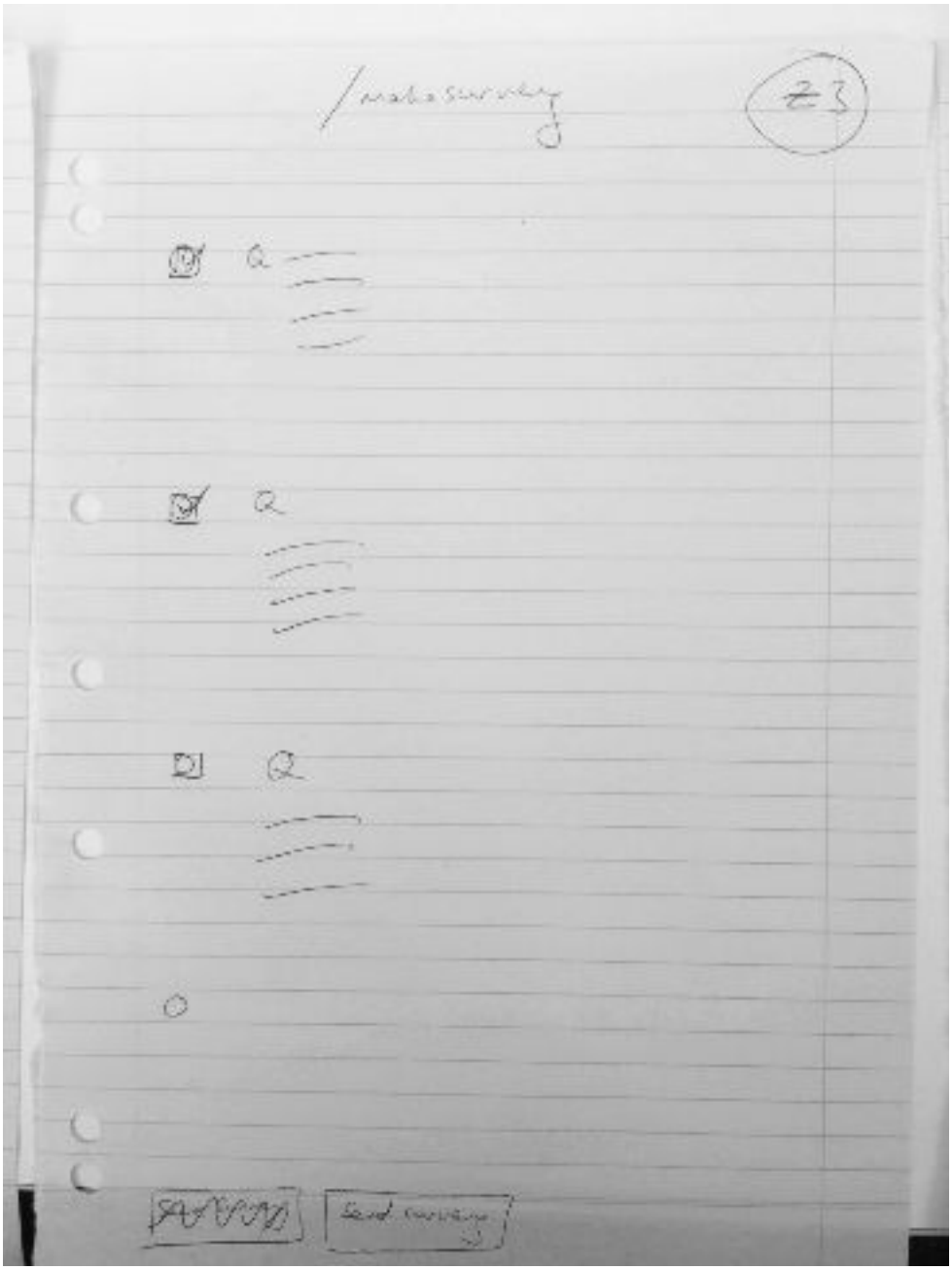
Z0 - Index - Jacob



Z1 - Login - Paul

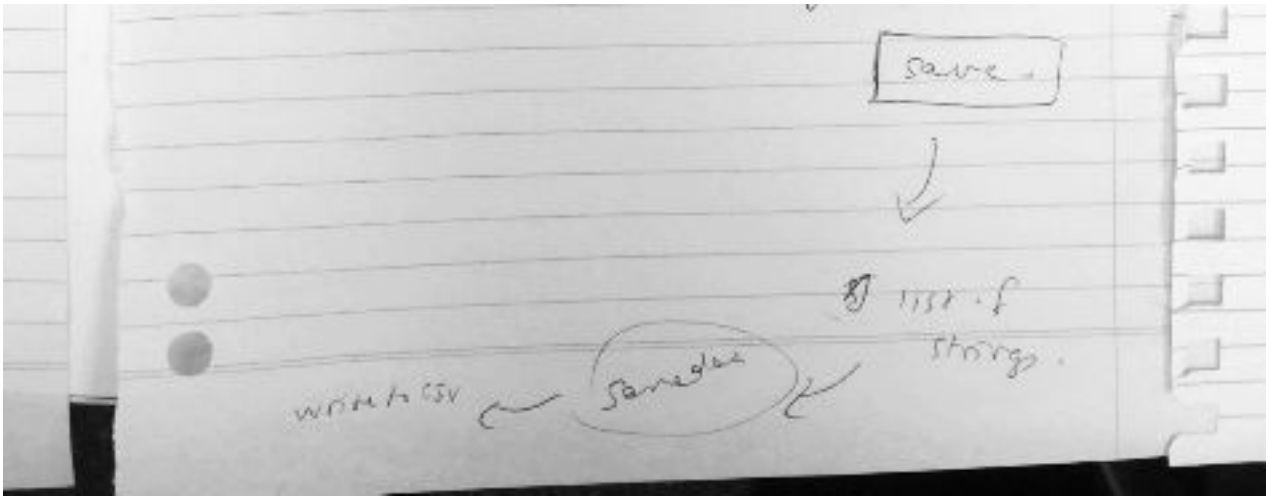


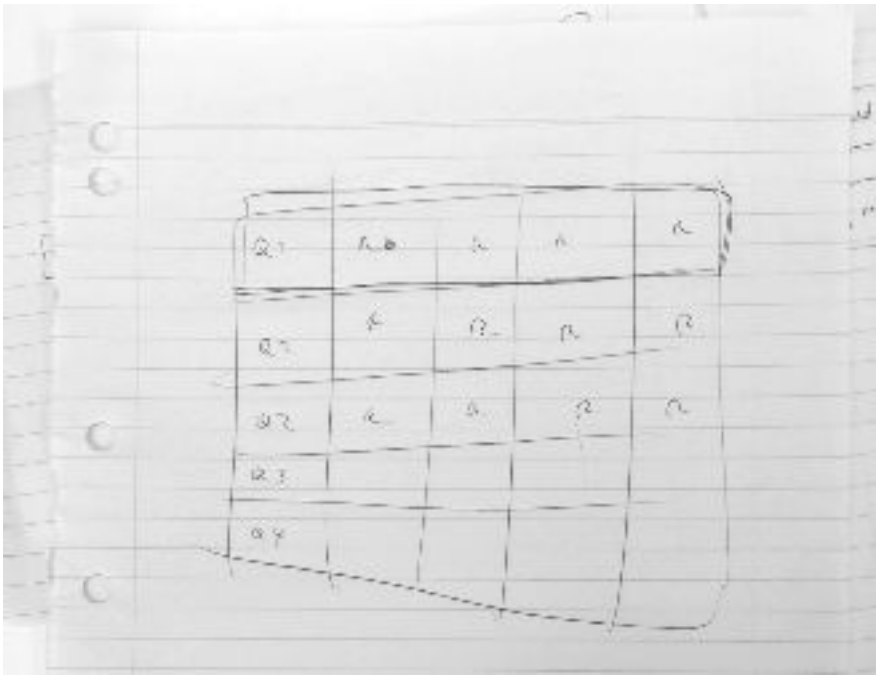
Z2 - Add question - Dan



Z3 - Make survey - Karl

We also discussed how the questions would be saved as a CSV file

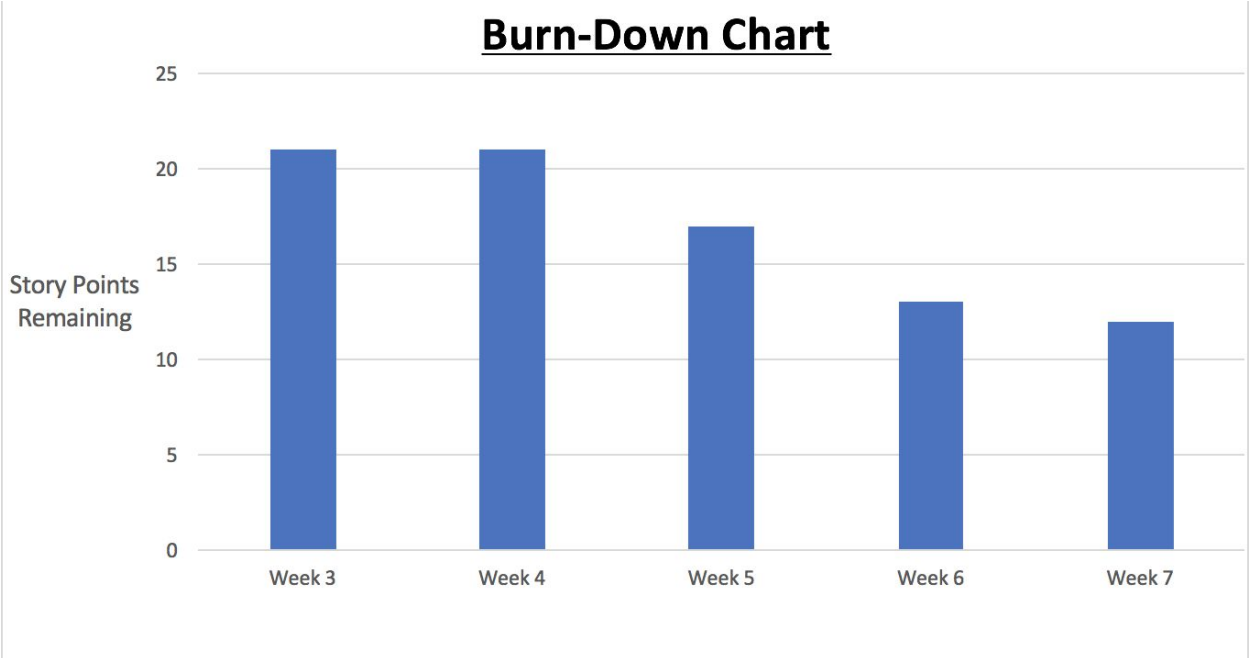


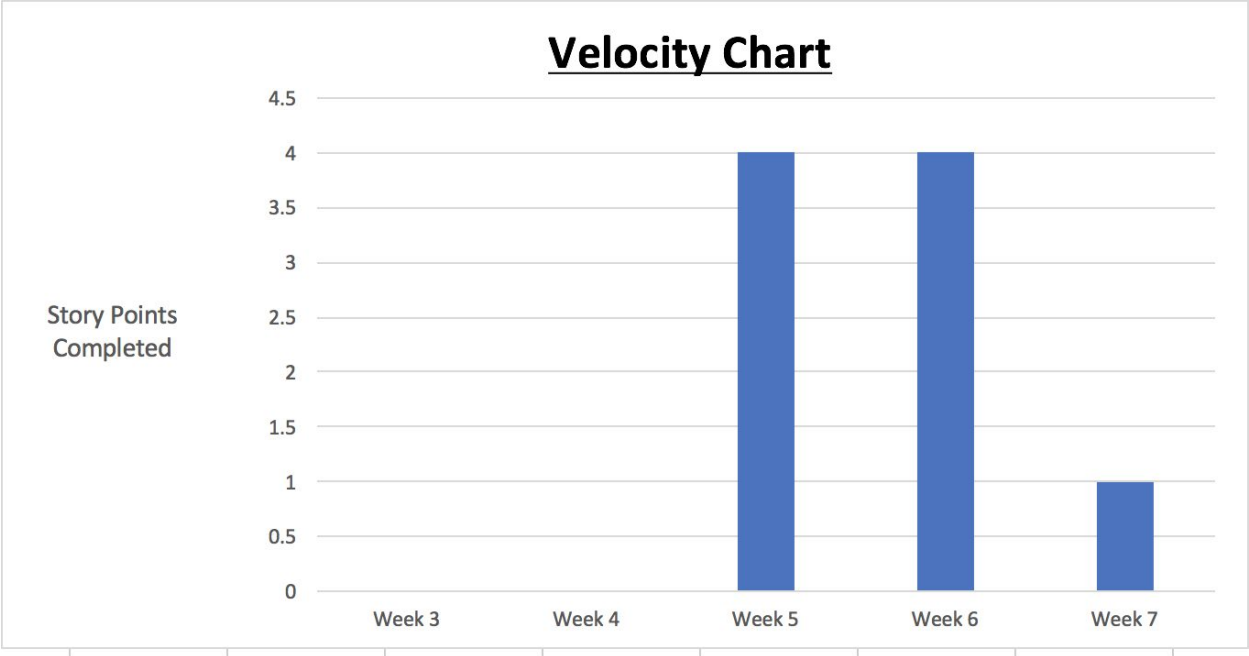


August 29

Some work reapportioned, Karl taking on overall architecture and Dan taking on overall presentation and CSS.

Progress to date, week 7 first iteration demo: Stories 0,2,3,5,6 completed in MVP form.





Standup 5 sept (Week 7)

*Jacob is now the customer

Things to change for next iteration:

*Each course to have own csv file, if we don't have to use a db by then anyway.

*Save question_list and survey_list in a file or db instead of in memory -> this is the kind of things people seem to json for? There must be other ways , even csv.

*Share survey -> possibly students will have their enrolments and emails in the db so when you share a survey, it just gets sent to all enrolled in the course

Standup 12 sept (Week 8)

Backend tasks implemented: Write questions to csv, login required -> this is covered in flask tutorial but uses an external service which may not be best. Find out what we are meant to use or if allowed to use external services.

Discussions: Potentially put hash in the id_num

Standup 19 sept (Week 9)

Backend tasks implemented: Another try at db, first attempt at implementing SQLAlchemy, (credits to Miguel Grinberg for the Python scripts and tutorials), questions is now an abstract class that allows for different types of responses, first attempt at doing the mvc thing

Frontend tasks implemented: Pretend logged-in greeting, favicon, styling, basic login.

Standup 3 Oct (Week 10)

Backend tasks implemented: new 'Answer' class added, created when a survey is filled, survey viewing still not functioning properly though, add question feature should be working, first attempt at fixing create survey, updated add_questions.html regarding new spec, tidied some things up, also made database a little easier to work with, csv files uploaded into the database, initial attempt at db now running, still needs to implement queries, etc, first attempt at implementing some relationships in db

Frontend tasks implemented: Overall styling

Standup 10 Oct (Week 11)

Frontend tasks implemented: 'Delete survey' buttons on view survey and list survey pages, UML, asterisks for mandatory questions, some styling for new pages e.g. survey.html, icons in navbar, table view for open-ended questions, styling for add_question, centred login box styling

Documentation: Adding Sequence Diagram

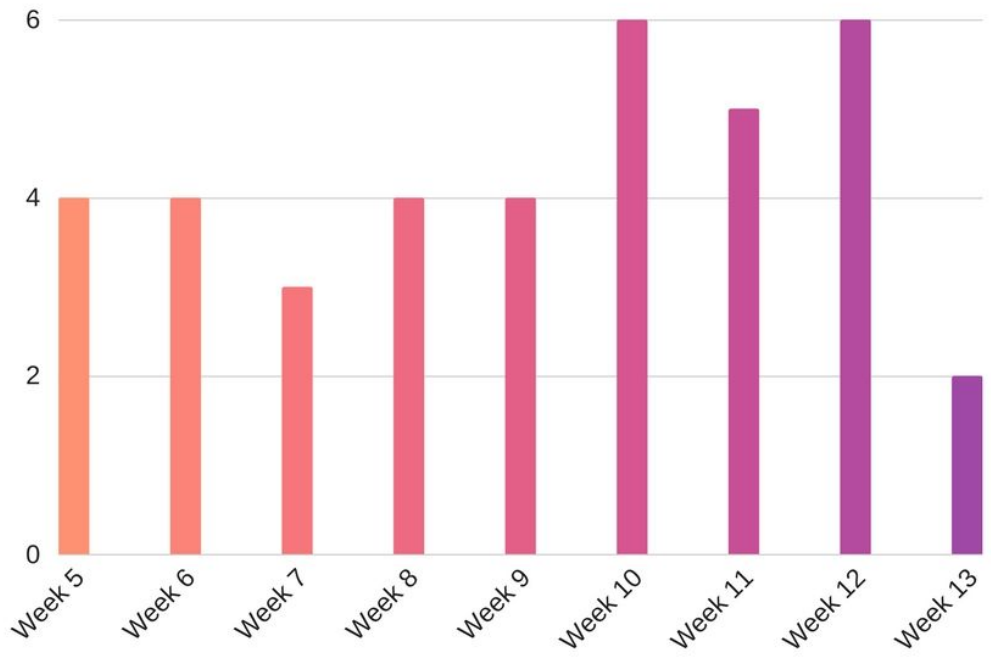
Bankend: Enforced the 'optional' and 'mandatory' status of questions, proper workflow for admin, staff and student should now be working, barring some bugs, individual survey page hopefully works now, added a question type column in view questions, enforced user permissions, initial attempt for dashboards, fixed some of the upload files, tidied up auth manager, implemented user logins from database, user class now extends UserMixin from Flask-Login, still needs to fix authentication though.

Standup 17 Oct (Week 12)

Documentation completed: ER diagram, user stories, Project Log

Frontend: working charts using chart.Js, open ended question section in charts page, list of surveys with charts, fix css issue on survey page.

Backend: implemented security module, passwords are now hashed in db, admin can't fill out surveys anymore.



Final velocity chart (Story points on y-axis).

There were some bursts of activity before iteration demos in weeks 7, 11 and 13, but velocity was reasonably steady at an average of 4.2 story points a week (total 38 story points).