Evidence for Analysis and Design Unit
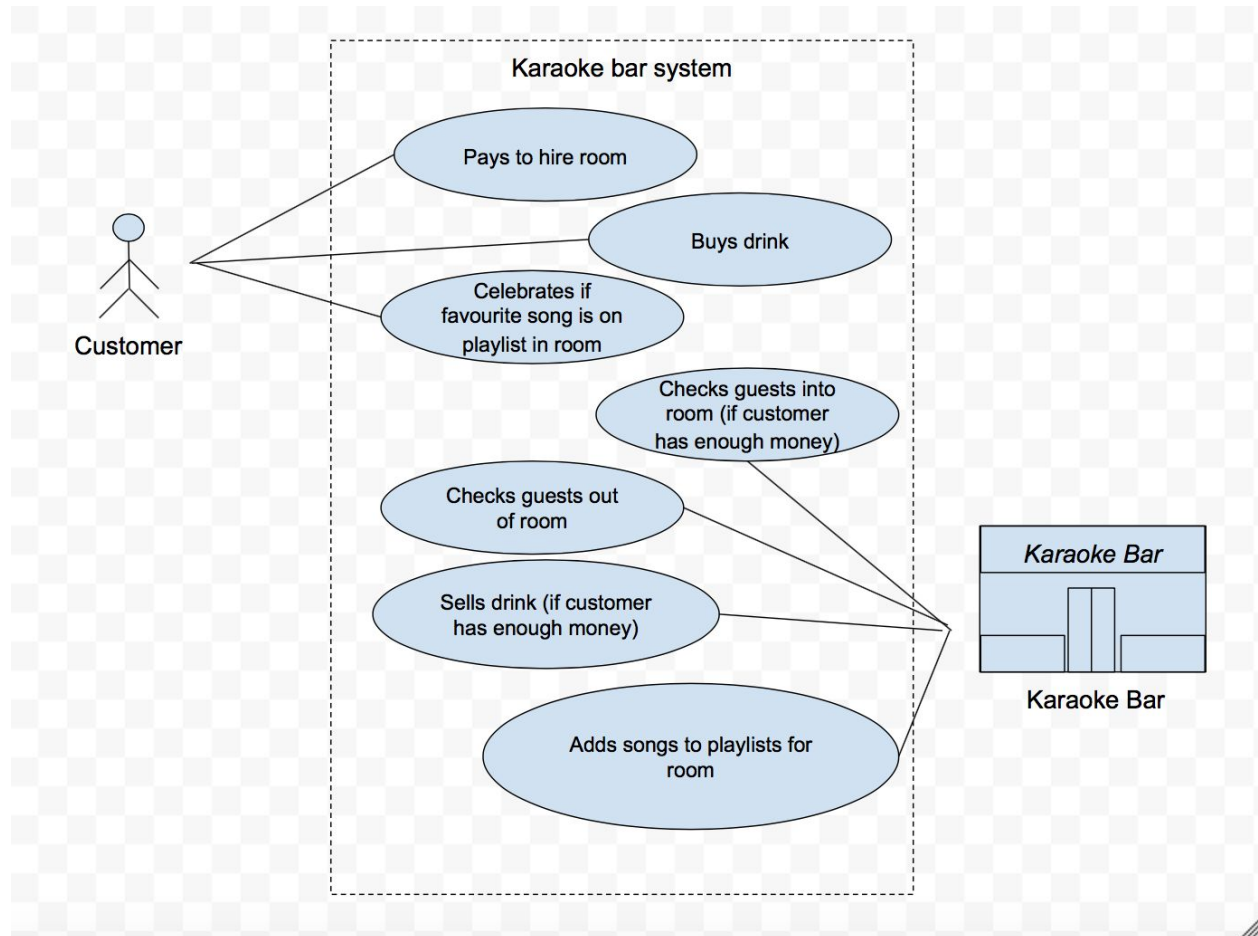
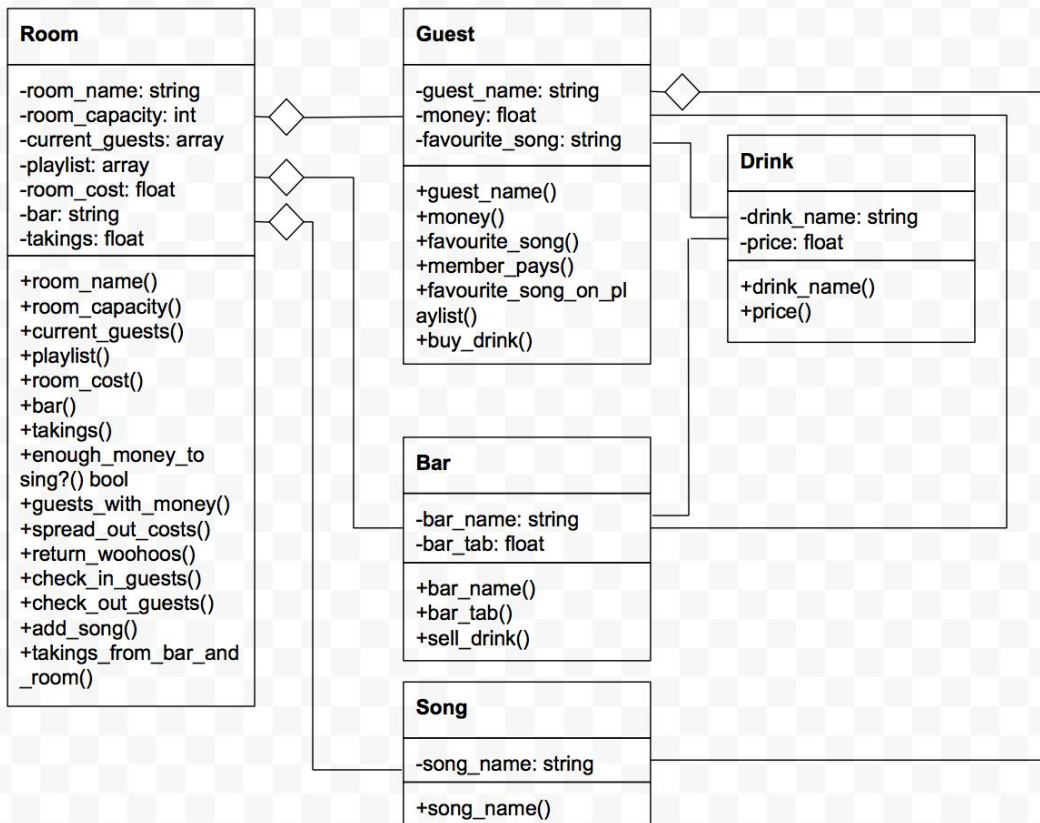Rachel Johnson
E19
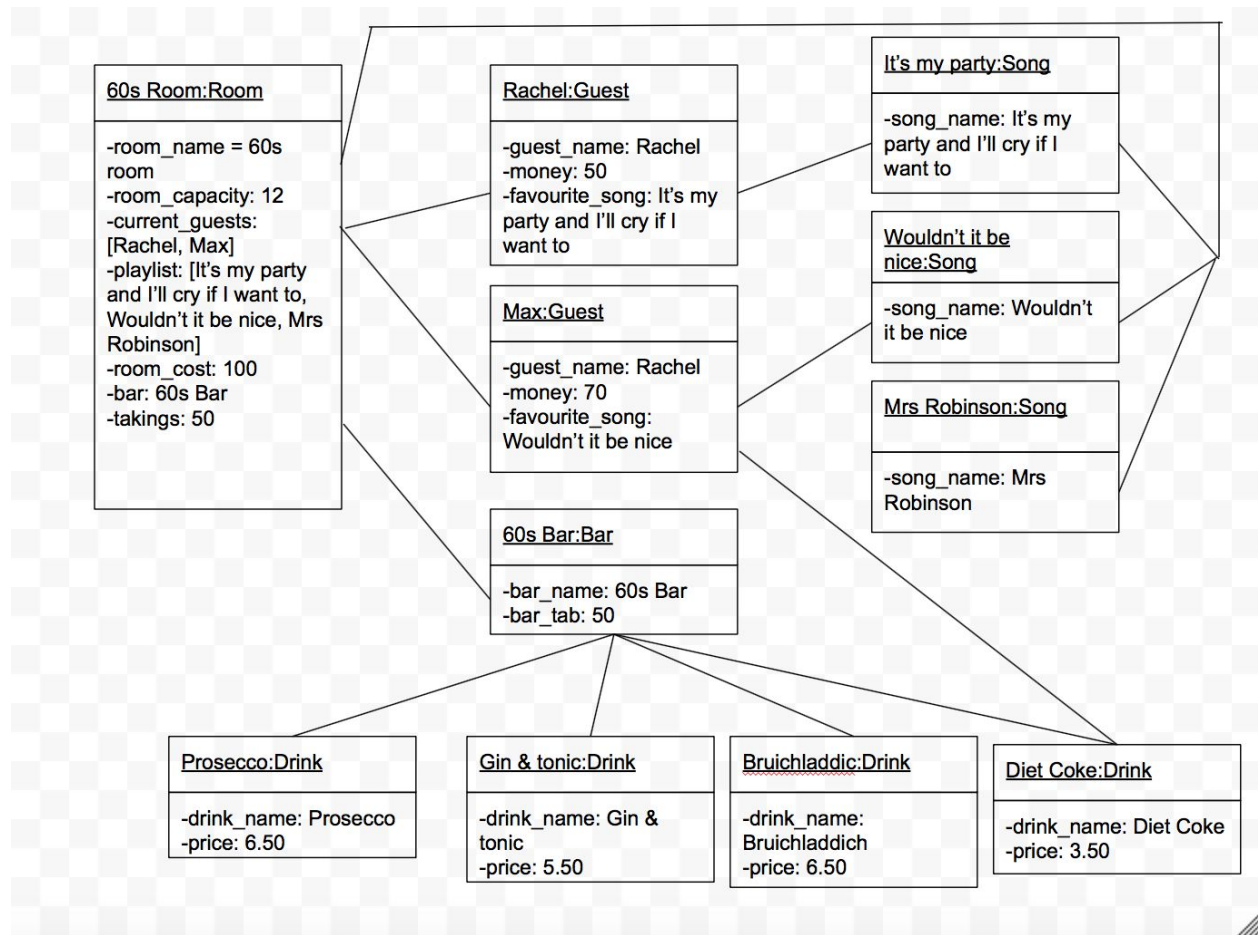
A.D 1 Use case diagram

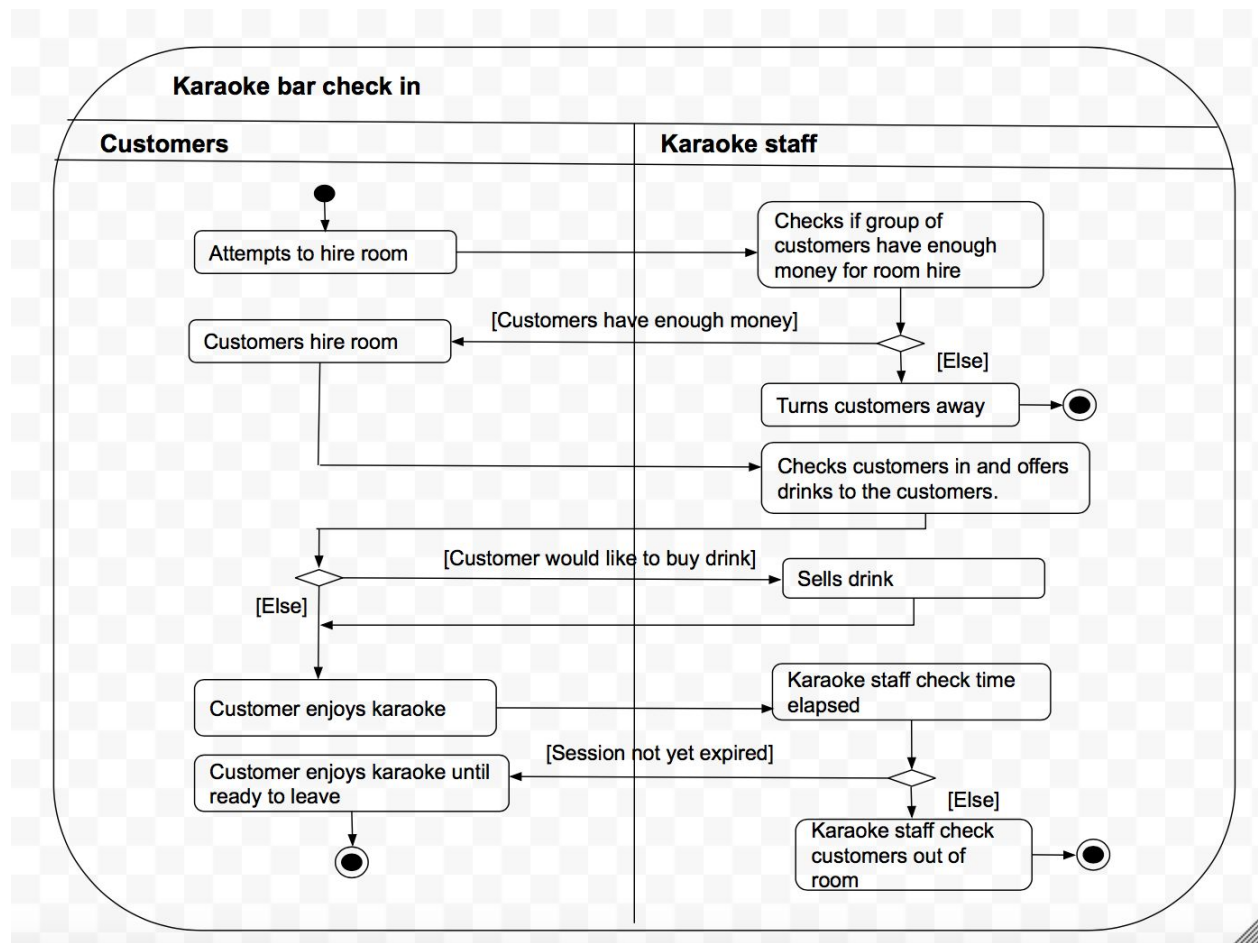Karaoke bar system

Pays to hire room

Buys drink

Celebrates if favourite song is on playlist in room

Checks guests into room (if customer has enough money)

Checks guests out of room

Sells drink (if customer has enough money)

Adds songs to playlists for room

Customer

Karaoke Bar

Karaoke Bar

A.D 2 Class diagram

Room

-room_name: string
-room_capacity: int
-current_guests: array
-playlist: array
-room_cost: float
-bar: string
-takings: float

+room_name()
+room_capacity()
+current_guests()
+playlist()
+room_cost()
+bar()
+takings()
+enough_money_to
sing?() bool
+guests_with_money()
+spread_out_costs()
+return_woohoos()
+check_in_guests()
+check_out_guests()
+add_song()
+takings_from_bar_and
_room()

Guest

-guest_name: string
-money: float
-favourite_song: string

+guest_name()
+money()
+favourite_song()
+member_pays()
+favourite_song_on_pl
aylist()
+buy_drink()

Drink

-drink_name: string
-price: float

+drink_name()
+price()

Bar

-bar_name: string
-bar_tab: float

+bar_name()
+bar_tab()
+sell_drink()

Song

-song_name: string

+song_name()

A.D 3 Object diagram

60s Room:Room

-room_name = 60s room
-room_capacity: 12
-current_guests: [Rachel, Max]
-playlist: [It's my party and I'll cry if I want to, Wouldn't it be nice, Mrs Robinson]
-room_cost: 100
-bar: 60s Bar
-takings: 50

Rachel:Guest

-guest_name: Rachel
-money: 50
-favourite_song: It's my party and I'll cry if I want to

Max:Guest

-guest_name: Rachel
-money: 70
-favourite_song: Wouldn't it be nice

60s Bar:Bar

-bar_name: 60s Bar
-bar_tab: 50

It's my party:Song

-song_name: It's my party and I'll cry if I want to

Wouldn't it be nice:Song

-song_name: Wouldn't it be nice

Mrs Robinson:Song

-song_name: Mrs Robinson

Prosecco:Drink

-drink_name: Prosecco
-price: 6.50

Gin & tonic:Drink

-drink_name: Gin & tonic
-price: 5.50

Bruichladdic:Drink

-drink_name: Bruichladdich
-price: 6.50

Diet Coke:Drink

-drink_name: Diet Coke
-price: 3.50

A.D 4 Activity diagram

Karaoke bar check in

Customers	Karaoke staff

- Attempts to hire room
- Checks if group of customers have enough money for room hire
- [Customers have enough money]
- Customers hire room
- [Else]
- Turns customers away
- Checks customers in and offers drinks to the customers.
- [Customer would like to buy drink]
- Sells drink
- [Else]
- Customer enjoys karaoke
- Karaoke staff check time elapsed
- [Session not yet expired]
- Customer enjoys karaoke until ready to leave
- [Else]
- Karaoke staff check customers out of room

A.D 5 Inheritance diagram

Instrument

-uniqueId:int
-buyPrice:double
-sellPrice:double
-description:String
-material:String
-colour:String
-instrumentType:enum

+getUniqueId()
+getBuyPrice()
+getSellPrice()
+markUp()
+getDescription()
+getMaterial()
+getColour()
+getType()
+playSound()

Stock

-uniqueId:int
-buyPrice:double
-sellPrice:double
-description:String

+getUniqueId()
+getBuyPrice()
+getSellPrice()
+markUp()
+getDescription()

Saxophone

-uniqueId:int
-buyPrice:double
-sellPrice:double
-description:String
-material:String
-colour:String
-instrumentType:enum

+getUniqueId()
+getBuyPrice()
+getSellPrice()
+markUp()
+getDescription()
+getMaterial()
+getColour()
+getType()
+playSound()

DrumKit

-uniqueId:int
-buyPrice:double
-sellPrice:double
-description:String
-material:String
-colour:String
-instrumentType:enum
-numberOfDrums:int
-numberOfCymbals:int

+getUniqueId()
+getBuyPrice()
+getSellPrice()
+markUp()
+getDescription()
+getMaterial()
+getColour()
+getType()
+playSound()
+getNumberOfDrums()
+getNumberOfCymbals()

GuitarStrings

-uniqueId:int
-buyPrice:double
-sellPrice:double
-description:String
-numberOfStringsInPack:int

+getUniqueId()
+getBuyPrice()
+getSellPrice()
+markUp()
+getDescription()
+getNumberOfStringsInPack()

Metronome

-uniqueId:int
-buyPrice:double
-sellPrice:double
-description:String
-maxSpeed:double

+getUniqueId()
+getBuyPrice()
+getSellPrice()
+markUp()
+getDescription()
+getMaxSpeed()

A.D 6 Implementation constraints

Topic	Possible effect of constraint on product	Solution
Hardware and software platforms	App has not been fully tested on all available web browsers, meaning that app may crash or not function as it should on untested browsers. This will cause frustration for users and may encourage potential users to use alternative budgeting apps, leading to loss of revenue as a) users will be reluctant to pay for use of the app b) it will be difficult to see revenue space on the app if it has a small user base	Test app on all available web browsers AND/OR prevent users from accessing app unless a tested browser is being used.
Performance requirements	App may slow as large amounts of data are added to the database. This could again cause frustration for users leading to the issues discussed above. Users can only access stored data when connected to the internet. This limits the utility of the app as users do not have access to their transaction data at all times, which may encourage users to use alternative budgeting apps with data that is accessible at all times	Limit users ability to add more data after a certain level reached. OR Consider alternative method to run site and store data which is more efficient / uses less processing power & memory. Cache data offline.
Persistent storage and transactions	Data only stored locally - data resets every time program is rerun. This means users' data will be lost whenever program is rerun, making the budgeting app useless and causing frustration for users who wish to continuously track their spending.	Host app on external website which stores data in the cloud.
Usability	App not responsive so will look unattractive / difficult to understand on mobile.	Extend project to make app look good on both laptops and mobiles. OR

	Page reloads every time a button is clicked which will slow down the app's response to button clicks and be frustrating to the user. Both of these items could encourage users to use alternative apps which are responsive and respond to inputs quickly.	Alert users accessing via mobile that app is optimised for viewing on a laptop and may be difficult to navigate via mobile. Write app in a different language (e.g. Javascript) which does not reload a page every time a button is clicked.
Budgets	No budget available so app must be created using open source software, which may limit functionality / efficiency of app. Users who are seeking increased functionality will not use this app and instead use alternative apps with more functionality.	Alert client (i.e. those who requested the built of the app) and suggest budget is increased to enable product team to create an app which meets a broader scope.
Time limitations	Work has to be done in one week so there is insufficient time to incorporate all the desired functionality (such as showing charts and graphs) into the app. Users who are seeking increased functionality will not use this app and instead use alternative apps with more functionality.	Restrict functionality of app to that which can be set up within the given time constraints. Request an extension to the deadline to allow for the desired functionality to be set up in full.