Å forstå hvem målgruppen for appkliasjonen er, og lage et grensenitt ut ifra dette.

Men går ikke dette litt inn på arkitektur  
**Brukermanual (teknisk og ikke-teknisk)**  
**Teknisk? Whatfuck?**  
  
**Vedlegg-oversikt**

**VEDLEGG**

Teknisk brukermanual  
  
Vi har som sagt tidligere brukt rammeverket qt jambi, dette medfører at vårt programm ikke er plattformuavhengig.  
  
For å kjøre programmet vårt trenger man java 32 bit, windows 7 32/64 bit eller windows 8 64 bit. Det vi har vært så heldig å få bruke er serveren til Lars Erik, så man er ikke avhengig av å være logget på ntnu nettet. Man kan kjøre programmet på hvilket nettverk man vil, og det vil fortsatt fungere.  
**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  
**Foreløpig utelatt (deler av teksten som har blitt tatt ut, men som andre kan ta inn igjen eller jobbe utifra hvis de vil):**  
  
Studentene skal også få en innføring i javaprogrammering i gruppen og hvordan dette løses, ved bruk av versjonskontroll. Hovedpoenget med it1901 faget blir da å få praktiske ferdigheter i forhold til javaprogrammering og ta besluttninger i fellesskap som en gruppe.  
  
  
  
*/\*Putter det her enn så lenge. Kanskje det passer inn, jeg vet ikke, men da burde vi putte inn denne slags info om alle eller de fleste programmene vi har brukt (?)\*/*  
Git er Engelsk slang for “en dum og ubehagelig person” og er utviklet av Linus Torvalds, som også står bak Linux.  
  
  
  
  
  
MYE HER ER BULLSHIT  
Noe av det første vi gjorde var å kartlegge hvor våres individuelle kunnskap lå.   
Tanken var at vi skulle få overordnet ansvar innenfor de ulike delene av prosjektet.  
De ulike ansvarsområdene ble:  
 - Databser  
 - Grafisk grensesnitt  
 - Dokumentasjon  
 -   
Det ansvarsområdet man fikk betydde ikke at man hadde ansvar for å lage den delen av prosjektet alene, det var heller der de personene måtte passe på å ha oversikt OMFORMULER  
  
Resten av den faktiske arbeidsfordelingen ble førs  
  
  
  
Lars Erik kunne ikke komme på første møte, som vi avtalte på kort varsel, noe han gjorde mer enn opp for siden.

**Kap 5.1 Hva slags risiko står man ovenfor i dette prosjektet?**

Det er masse som kan gå galt i et prosjekt som dette, men noen “farer” er viktigere å tenke på enn andre, deriblant kommunikasjonsfeil, sykdom, at noen faller fra og misting av arbeid. Kanskje er det også slik at splitting av gruppen noe man burde vurdere som en viktig risiko, men vi oppdaget ingen tilløp til konflikt.  
  
  
  
Rapportinnhold utifra fagside.  
  
Kap. 1 Introduksjon

Kap 1.1 Om faget, om oppgaven og om gruppen

Kap 1.2 Beksrivelse av prosjektet

Kap 1.3 Valg av prosessmetode (beskrivelse)

Kap. 2 Tidsestimering (sprint besrkivelse, WBS, o.l.)  
Kap. 3 Prosjektorganisering

Kap 3.1 Ansvarsområder

Kap 3.2 Om de forskjellige medlemenes kunnskapsområder

Kap. 4 Arkitektur forklaring  
Kap. 5 Risiko i prosjektet

Kap 5.1 Hva slags risiko står man ovenfor i dette prosjektet?

Kap 5.2 Splitting of the group

Kap 5.3 Uforutsette utfordringer (ny)

Kap 5.4  Deadlines (at man ikke rekker det)

Kap 5.5 Medlemer som ikke møter

Kap 5.6 Kommunikasjon

Kap 5.7 Prosjektplanen failer

Kap. 6 Produktbeskrivelse  
Kap. 7 Prosessbeskrivelse

Kap 7.1 Sprint

Kap 7.2 Arb.fordeling

Kap 7.3 Møter

Kap 7.4 Kildekodedeling

Kap 7.5 Facebook

Kap. 8 System design  
Kap. 9 Implementasjon  
Kap. 10 Tools  
Kap. 11 Evaluering  
 Kap 11.1 Hva gikk bra  
 Kap 11.2 Hva gikk dårlig  
 Kap 11.4 Lærdom til neste gang  
User manual (teknisk og ikke-teknisk)  
Vedlegg: diagrammer, timelister, kildekode, møtereferat, testing  
  
**why qt jambi?**: <http://www.sra.co.jp/qt/relation/qtjambi-whitepaper-tp3.pdf>  
  
  
  
  
Da disse individene ikke hadde noe imot å lære bort, har resten sittet og konsentrert seg , kommet med spørsmål og vist god insattsvilje til å lære.  
  
  
De første møtene var vi stort sett sammen, da vi skulle bli enige om hvordan vi ville at systemet skulle se ut.  
  
  
Dette var et veldig enkelt og grei måte å ha icescrum på egen og pålitelig server. Her logget vi inn og oppdaterte chartene og tasks vi gjorde underveis. Dette har gått veldig bra og smertefritt.  
  
Hele teamet har vært med under hele prossesen, og fått med seg alt som man skulle få med seg. Men alt i alt har vi fungert bra som en gruppe, og ingen har klaget eller gjort seg noen misnøye med at alle ikke endte opp med å gjøre ”like mye” som alle andre.  
  
  
Da var vi så smarte etter en kort tid å bytte til icescrum, som ble det ultimate rammeverket som dekket våre behov iforhold til at vi brukte scrum-metoden som arbeidsmetode. Her fikk vi generert burndowncharts(og div) automatisk. Men i starten her var vi ganske nybegynnere og gjorde en del feil. Vi delte først ikke opp i stories og lagde task utifra de så hver enkelt kunne ta en task. Vi delte opp i store bolker og skrev ned timer på details på hver bolk. Da vi lærte icescrum senere under prosjektet så vi hvor stor nytte det hadde å gjøre det riktig og ordentlig.

**10.1 JavaDoc**

JavaDoc er blitt brukt for dokumentasjon av koden. //Her må noen skrive mer.

**10.2 ER-modeller**

ER-modeller er en abstrakt modell for å modellere databaser (man viser hva slags data tabellene inneholder og hvordan de forholder seg til hverandre). Vi hadde alle lært om ER-modeller i *Informatikk Grunnkurs*, og Lars Erik hadde også hatt et kurs i Databaser hvor ER-modeller var en del av pensum.

**10.3 Dia**

Design av diagrammer er gjort i det GTK+-diagram-baserte programmet Dia. Dette var et greit program som har innebygde funksjoner som er beregnet på bl.a. UML-diagrammer. Ingen utenom Lars hadde noen erfaring med dette programmet, men ble fort kjent med det. Grunnen til at vi valgte dette var at Lars hadde erfaring med programmet fra et annet fag og at det fungerer på både Mac og Windows.

**10.4 UML**

Unified Modeling Language er en standard for IT-relatert modellering innenfor objektorientert programvare. Diverse diagrammer er UML-diagrammer. UML-diagrammer brukes til å klargjøre og gi oversikt over diverse aspekter av hvordan et program skal fungere. Vi har laget UML-diagrammer i planleggingen av prosjektet, og en del av dem har kommet til nytte for å få en generell forståelse av hvordan det ferdige produktet skulle se ut og fungere, og for å ha en felles forståelse av hva vi skal gjøre.

**10.5 Eclipse**

Eclipse er et program for å skrive software (software development environment). Eclipse kan brukes til å skrive i flere språk med ved hjelp av plugins, men har Java som default. Eclipse brukes i kurset *Objektorientert Programmering (TDT4100)*, som alle informatikk-studenter har før de har dette faget.

**10.6 Facebook**

Vi sjekket om faglærer hadde Facebook-profil og så at han hadde det, og regner med at studasser og undasser også har det, så vi antar at alle som leser denne rapporten *ihvertfall* forstår grunnprinsippene. Men for sikekrhets skyld: Viktig å forstå er at man kan være medlemmer av *grupper* og at man på disse gruppene har en felles *wall* der man kan skrive poster og skrive *kommentarer* i respons til poster (på samme måte som wallen til en brukers profil før man fikk timeline). Når det kommer nye poster i en gruppe man følger med på, eller når noen kommenterer på en post du har postet eller kommentert på, eller som du er *tagget* i, får du *notifications*. Vi har valgt å også nevne relevant funksjonalitet med henhold til Facebook-grupper i teksten (kap 7.5).

**10.7 Git**

Git er et system som er laget for folk som jobber på felles IT-prosjekter. Man har en felles prosjektmappe hvor filene er lagret, og systemet håndterer at forskjellige gruppemedlemmer forandrer på filer i prosjektmappa. Hvis forskjellige gruppemedlemmer forandrer på samme fil samtidig får man utfordringer, me også dette er mulig å håndtere.  
  
Hvis man ikke brancher betyr det at man kun har en prosjektmappe som man jobber på, men hvis man brancher betyr det at man jobber på to uavhengige prosjektmapper/”grener” med felles utgangspunkt, hvor en forandring på den ene ikke påvirker den andre. Slik kan man f.eks. lagre programmet som virker i en “gren” og jobbe i en annen. Etterhvert kan man føre grener sammen igjen. //Se gjerne på dette og se om dere synes det er ok.

**10.8 Qt Jambi**

Qt Jambi er et Java bibliotek for å lage grafiske brukergrensesnitt med bindinger mot C++ biblioteket Qt. //Her må noen andre gjerne skrive mer hvis de har noe å føye til.

**10.9 Kartverkets WMS-tjeneste**

//Her må noen skrive mer.

**10.10 Google maps API**

//Her må noen skrive mer.

**10.11 Trello**

Trello er en nettside som hjelper folk med å få oversikt over og koordinere prosjektene sine. Du har lister (som kan representere kategorier av diverse slag) med kort (som kan representere arbeidsoppgaver) som du kan dra rundt omkring. For disse kortene er det diverse funksjonalitet. Du kan f.eks. adde deg selv til et kort for å vise at du jobber på arbeidsoppgaven det representerer, eller flytte kortet til ferdig-lista hvis jobben er gjort.

**10.12 IceScrum**

IceScrum er laget for metodikker som bl.a. Scrum. Digital sprinttavle og burndownchart er innebygde funksjoner i IceScrum, og burndownchart oppdateres automatisk med henhold til sprinttavla.

**10. 13 MySql**

//Her må noen skrive mer.

**10.14 Google docs**

Rapportskrivingen og i forbredelsen av midtveispresentasjonen, samt i planleggingen av enkelte av sprintene, har vi brukt Google Docs. Dette har veldig praktisk siden dokumentene lagres automatisk, og man kan skrive og redigere på det samme dokumentet samtidig, og se hva de andre gjør i real-time.