Tampere University Unit of Computing Sciences
COMP.SE.610 Software Engineering Project 1
COMP.SE.620 Software Engineering Project 2

Student ICT-PROJ "Get A Room!", Vincit / 1

Get A Room!

Project plan

Jere Koski
Joona Prehti
Joonas Hiltunen
Martti Grönholm
Mikko Pirhonen
Sara Brentini
Vivian Lunnikivi

Version history

Version	Date	Author	Description
0.1	17.09.2021	Joona Prehti	Technical specification
0.2	19.09.2021	Sara Brentini	Requirements
0.3	19.09.2021	Vivian Lunnikivi	Purpose and scope of the project
0.4	20.09.2021	Sara Brentini	Risk management
0.5	22.09.2021	Vivian Lunnikivi	Customer's current system, similar systems, and project constraints.
0.6	23.09.2021	Sara Brentini	Finalized chapter 6, started writing chapter 3
0.7	23.09.2021	Vivian Lunnikivi	Finalized chapter 3, started writing chapters 4 and 5 (process and project management, and scheduling).
0.8	24.09.2021	Vivian Lunnikivi	Finalized chapters for risks, process and project management, and iterations and timing.

Contents

1 Introduction

1.1 Purpose and scope of project

This document describes the plan for carrying out the project *Get a Room!*, offered by the technology company Vincit, for implementation in the Tampere University courses *COMP.SE.610* and *COMP.SE.620 Software Engineering Project 1 & 2* in fall 2021.

In the project, a group of seven Information Technology students implement a mobile-first designed application for easy making of spontaneous meeting room reservations. The end user will download the application, sign in with their Google id, and set their preferred office location, after which they will be able to make room reservations, based on the room recommendations made by the app.

Get a room! application integrates with the existing cloud environment already in use at Vincit, to display and reserve room resources. The application will be implemented as a React web application and as *progressive web application* (PWA), enabling native-app-like user experience. Using the application will be supported on agreed-upon mobile and desktop environments.

The scope of the project includes designing, building, and testing a responsive React application that allows making room reservations for existing resources in a Google Workspace environment. During development, a test cloud environment is used, but the implementation will enable integration with the customer's actual Google Workspace and single-sign-on (SSO) environments and the integration to the customer environment might already be carried out towards the end of the project.

However, the project scope does not include implementation of user management or product vision lead, as user accounts used in the app are managed by Google, and Vincit provides a *Product Owner* to guide the implementation in terms of requirements clarifications and prioritization. Hard performance requirements are also excluded at this stage of development, although the architectural style targets to minimize performance issues, in terms of minimizing perceived delays and usage of phone battery life.

1.2 Product and environment

The product's main goal is to make it easier for Vincit employees to quickly book a meeting room at the workplace once the need for a brainstorming session with colleagues arises. Additional features may include booking other calendar resources, such as flexible office desks or even company cars.

Get a Room! client application is a PWA-compatible React app that runs on a browser, user's mobile or desktop device. Current supported browsers are Windows Edge and Mac/iOS Safari, since these are the most common environments used at Vincit. As the app is a PWA, it can be installed on users' mobile devices home screen which makes it behave very much like a native mobile application.

Authentication happens via Vincit SSO, which is based on Google SSO. After authentication, user gets a valid authentication token, which allows access to Vincit's instance of Google's Workspace API, providing resource management and access. The API is used to check for available rooms and make resource reservations.

The products domain name is "oispahuone.com". The React client is served to users from Cloud Run service, hosted on Google Cloud platform. There's also a Node.js server hosted from Google Cloud platform and used to save client data to a MongoDB database. Information saved there includes at least users' preferred office location and favourite meeting rooms. Correct user data is found in MongoDB by querying with unique user ID fetched from Vincit SSO during authentication process.

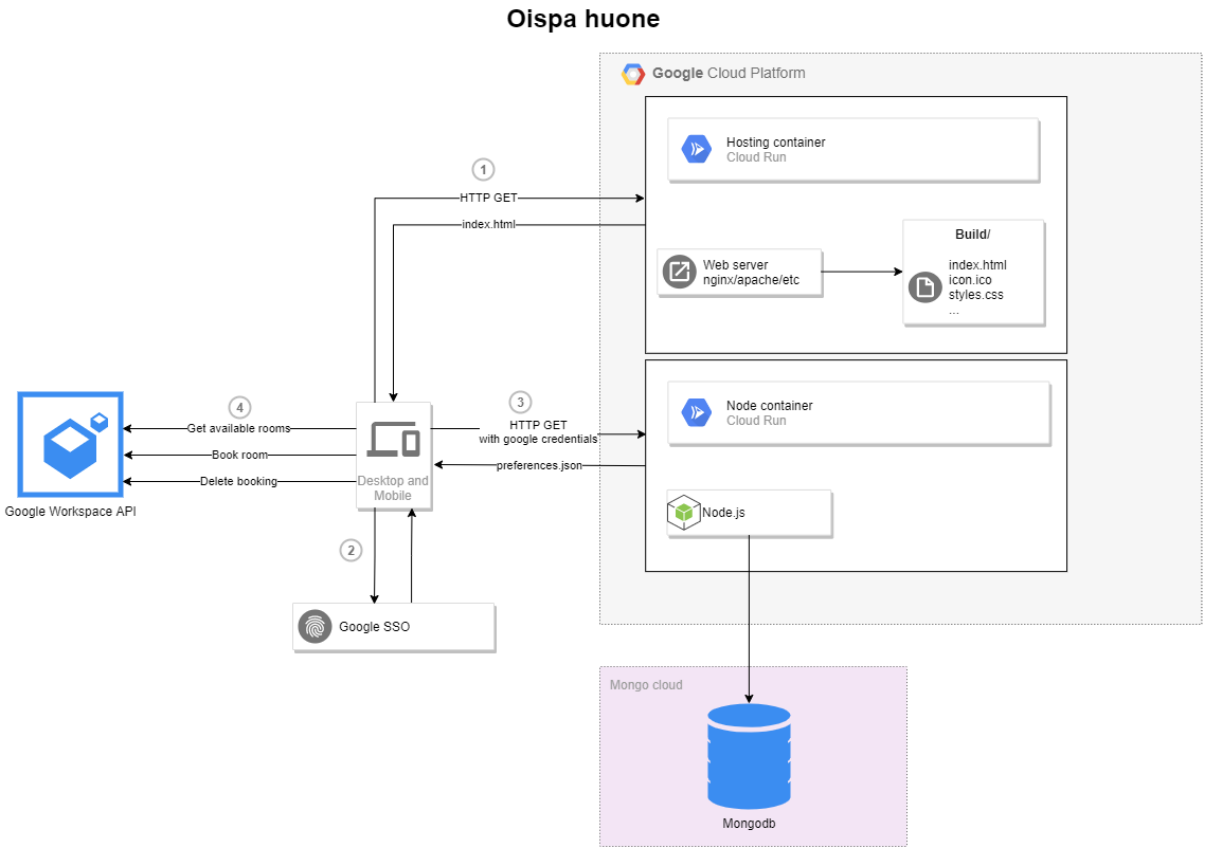


Figure 1: *Get a room!* high level concrete architecture.

Figure 1 presents the high-level component architecture with the planned technology stack. The end user downloads the app to their mobile or desktop device, signs in with their Google id via Vincit's SSO server, after which the application connects to Google Cloud Platform to fetch and save user preferences and reservable resources. As preferences are saved in cloud, settings are synced automatically between user's devices.

1.3 CI / CD and testing environment

GitHub is used for version control. The project utilizes GitHub's feature for creating automatic CI/CD pipelines, called *GitHub Actions*. The CI pipeline allows for continuously test the system for proper integration between software components and against possible regression when new features and code are added. The pipeline will also be used to deploy the application to the production environment, Google Cloud Platform, and it contains stages for linting, building, testing, and deploying.

In the Lint stage, the source code is analysed statically to find programming errors and bugs, with tools called ESLint and Prettier. ESLint focuses on finding programming errors, while Prettier checks that the source code conforms to the predefined code style rules.

In the Build stage, a deployable production build is formed. First, the source code is checked for errors preventing the project from building. If none are found, a package that can be deployed later in the pipeline, is produced, and saved as a pipeline artifact.

The test stage runs unit and integration tests against the project. This ensures the system works flawlessly, according to its specification, and that changes to the source code do not produce any regression. Jest library is used as the testing framework to orchestrate and run the tests.

The Deploy stage takes the output from the build stage, authenticates the GitHub to the Google Cloud and sends the React application production build to Google Clouds Cloud Build service which builds image out of it and saves it to Container Registry. And as last step the GitHub Actions trigger the deploy of the newly created Docker image to the Cloud Run service and the new version of the application is successfully running.

Development will be carried out in feature branches, which are merged to release branch once done. Every push and pull request to any branch on the repository triggers the whole pipeline. There is only one exception to this rule which is that deployment job of the pipeline will be triggered only on changes to release/dev branch. This is to have precise control on what code will be deployed to the staging environment. Pushing to the release/dev branch triggers automatic deployment if integration succeeds, and deployment goes to a staging environment that is continuously available for the customer as well.

Later, there may be a need for additional production environment where the finished product will be running. Staging environment could be still used for further development and bug fixes. Other possibility is to just turn the staging environment to a production environment if immediate further development is unknown.

The CI/CD pipeline will be run using GitHub’s provided virtual machines, which run an Ubuntu, for the sake of stability in the pipeline’s execution.

1.4 Customer's current system and other similar systems

Currently, Vincit employees reserve rooms and other company resources through Google Calendar. The reservation process is rather slow and cumbersome, especially on mobile devices, as the user needs to find the correct web page, log in, open a calendar view, look separately for available rooms, and figure out all the details related to making a reservation via the calendar view. Get a Room! decreases the steps needed to reserve a room by allowing the user to see smart reservations instantly in the app and filling in smart default values for all the details Google Workspace requires for a room reservation, thus making the process faster and more user-friendly.

While quite a few software systems already exist for managing shared office resources, such as Microsoft’s Office365, Google Calendar and Booking Calendar, our research

did not identify an easy-to-use app for making quick – 1 to 3 clicks only – room reservations while integrating with the Google Workspace ecosystem. Several Booking applications exist, but none of them seem to display relevant suggestions and they all require user to fill in several details about the booking. This difficulty takes time and what's worse, deciding the details of the booking may break employees' flow of thoughts.

1.5 Project constraints

Development will mostly be constrained by the amount of time available for development work. There are five two-week implementation sprints planned for implementation and each group member only has around 10 to 15 weekly hours to put into the project, on average. While we are likely to complete at least an MVP implementation and some additional features, we need to be mindful to reserve at least half of that for solving technical issues, bugs, and carrying out testing, design, research, and documentation.

The customer has granted quite free hands for the group in terms of choosing the tools and technologies for the implementation and has made wishes mostly in terms of high usability, integrating with the current environment – described in section 1.4, and service modelling take on the project. However, as the company has branded a distinguishable visual appearance and the app is made for the company employees to use, the group will fit the visual appearance of the app with the Vincit brand, for which the client has given green light.

Intellectual property rights stay with the group members, but the software will be made available for further development by licencing it with the standard MIT licence, allowing copying, editing, and extending the software as long as the original contributors are accredited. MIT or similarly licenced OSS components are also used as a base for the project, including e.g., the PWA-compatible React application template used as a basis for the *Get a Room!* client application. Using these components require preserving accreditations in the project licencing information.

The customer has agreed to that the group may publish the source code and other material produced during the project with MIT licence, since, so far, there should not be any obstacles to publishing them. However, if delicate issues or materials present themselves during the project, those should be agreed to be kept from the public. These situations will be further discussed if need be.

One area of constraints that need to be considered includes the privacy and security of personal information handled by the system. We may not process or save any personal data, without an explicit consent of the user. All personal information also must be kept secret from outsiders, which requires us to use secure communication channels and keep an eye out for other possible security risks.

1.6 Definitions, abbreviations, and acronyms

End user	Group of people that use and benefit from the product. In the context of *Get a Room!*, end users consist of Vincit company employees.

MoSCoW	Popular prioritization framework that classifies features by priority into four categories: *Must-have*, *Should-have*, *Could-have*, and *Won't-have* or *Wish*.
MVP	*Minimum Viable Product*.
PWA	*Progressive Web Application*, a web application mimicking a native application, building on top of native implementations of features of a given device.
UI	*User Interface*. Describes the technical entity that allows an end user to interact with the developed system. Practically, this refers to what the user sees about the system on their device.
UX	*User Experience*. Field of science that researches how it feels to use products. In the context of software projects, UX design aims to ensure business success by guiding the product implementation to deliver good experience of using the product.
OSS	*Open-Source Software*, software components made available for use by the component author(s) by defining a set of terms, stated in a licence document.

2 Project organization

2.1 Group members

Jere Koski, Junior Developer	Experienced in full-stack web development using Apache Struts 2, Java, and JavaScript. Some experience with Nodejs and databases. Have also performed software demos for customers.
Joona Prehti, Architect and tech specialist	Mostly experienced on full-stack web development with Angular and Java Spring Boot. Can also work with JSF, Nodejs, Docker, browser plugins and relational databases like PostgreSQL. Intermediate know-how on React, DevOps practices and Cloud Platforms. (Also, a Linux enthusiast)
Joonas Hiltunen, Junior Developer	Mostly experienced in backend development with Nodejs, Python and C#. Some experience working with React frontends. Can also work with Docker, CI pipelines and different databases.

Martti Grönholm, UI specialist	Mostly experienced in front-end web development. Worked with React, Nodejs, and the basic web stack (JavaScript, TypeScript, CSS, HTML5). Some experience with progressive web application development. Basic knowledge of Human-Technology Interactions and machine learning.
Mikko Pirhonen, CI/CD specialist	Web development, main experience in React and Java. Some experience in DevOps tools and practices such as Docker and developing CI pipelines. Lots of experience in communicating with customers. Basic knowledge of machine learning and computer vision.
Sara Brentini, UX specialist	Experience mainly in full-stack web development. Worked with Java Spring Boot, JSF and other basic web stacks (HTML5, CSS, JavaScript). Also familiar with React and the concept of Human Technology Interactions. Further experience in working first handed with clients, project management and requirements engineering.
Vivian Lunnikivi, Project Manager	Experience on full-stack development with Node.js, React and the basic web stack (HTML5, CSS, JavaScript, and TypeScript). Familiar with at least Bootstrap, MaterialUI, and unit testing with Mocha and Chai. Knows a thing or two about test automation as well as web and cloud architectures and protocols.

2.2 Customer

Ari Metsähalme Tech Director	Tech Coach in *Get a Room!*
Ari Kontiainen Tech Coach	Product Owner in *Get a Room!*
Jonne Heiskanen IT Support	Provides IT Support in *Get a Room!*
Veli-Pekka Eloranta Cell Lead	Agile Coach in *Get a Room!*

2.3 Related organisations

Timo Poranen, University lecturer	Team coach

3 Project goals and ending/termination

3.1 High level goals of the project group

The project group aims primarily at a highly usable, user-friendly, responsive, and secure product that genuinely solves the issue of making room reservations easy in the customer's existing environment. We hope to reach high customer satisfaction with tight collaboration, active, service-modelling take on the project, and a high-quality MVP that is modelled after the actual needs of the customer.

Group's goals include working efficiently, focusing our efforts on relevant issues with reasonable planning and smart working habits to allow us to complete the project both on time and with an appropriate number of working hours, as well as with high quality. We hope to maintain high team spirit and learn the secrets of professional software product development throughout the development cycle, while gaining specialist knowledge on our individual responsibility areas.

The secondary goals for the project include implementing the system in a way that it can easily be developed further, for example, by Vincit. Should the timetables allow for it, we hope to also complete some of the *Should-have* and *Could-have* requirements – creating more value in the product than planned for in the MVP.

3.2 High level goals of the customer

The primary goals for the customer likely include the product fulfilling the requirements of the MVP and that the product is responsive and considers both usability and user experience. In other words, the *Get a Room!* app is available for Vincit employees and aids spontaneous room reservations. The product should also be easily integrated with the existing infra and be secure, so that, for instance, the personal information handled by the system is never leaked outside the system or in the wrong hands.

On a lesser emphasis, the implementation may extend the MVP implementation to cover reserving flexible office tables and other additional features. The reliability of the product should be reasonable, and code quality should allow for further development. The project should, of course, be completed on schedule, and with the project, Vincit participates in mutually beneficial co-operation with the University community.

3.3 High level goals and deliverable(s) of the project

The high-level goals that are agreed with the customers are as follows:

- Create a product with at least MVP requirements.
- Create a highly usable, user-friendly product that is also responsive, secure, and stable enough.

- The project is completed in time, and the quality of the software, and documentation and licenses handed over in the end of the project, allow for integration with the customer environment and enable further development of the product.
- The group will use modern, widely used development tools that support fluent software development and independent further development processes, and help reaching the quality requirements.
- Excess time after implementing the MVP and completing other necessary tasks, is directed to design and implementation of further development ideas.
- Both the client and the project group appreciate their time. Therefore, the group will minimize e.g., futile meeting time and make the working time count with efficient and reasonable ways of working.
- Course deliverables (project plan, testing plan, testing report, project report and project poster) and possible other relevant documents, as well as the produced source code, will be made accessible to the customer by the end of the project.

3.4 Quitting (termination) criteria of the project

This chapter summarizes the criteria of terminating the project before reaching a successful ending. Termination may be caused by a critical mass of the group members being unable to continue in the project, due to, e.g., sickness or exceeding their allocated resources. In this case the remaining group should try to agree with the customer upon a narrower implementation that allows the project to still be completed, or, in the face of overwhelming technical, legal, or mental issues, aim at reaching a mutual understanding of the project termination, holding each other accountable only to a reasonable degree.

If the customer were to withdraw from the project, the project group should still aim to complete the project with adjusted goals so that all members of the group have a realistic chance of completing the course. Project completion neither depends on possible competing product releases, as the project is, above all, a mutually beneficial practice on software product development project.

3.5 Ending criteria of the project

If everything goes as planned, the project ends on December 22nd, 2021, when at least an MVP implementation is “done” – that is, developed, tested and the software made available for the customer. All course deliverables are submitted successfully and *Get a Room!* system runs on the customer’s infra and the app is made available for downloading to at least iOS mobile and desktop devices on Safari. The detailed end of the project is agreed together with the group and the customer when the MVP implementation is done, and it gets apparent how much effort will be available for the end of the project.

4 Project and Process management

4.1 Methods and tools

Git is used for version control and GitHub for storing and distributing the source code, as well as continuous integration and deployment. Further details on the technology stack are located in the chapters 1.2, 1.3 and 1.4. Sharing documents such as meeting notes, project deliverable drafts, design documents and further reading link bank, is

done via a OneDrive folder shared among the group members, holding project documents in one place.

The requirements, requirements prioritization, and implementation work is managed with a Kanban board, built in Trello. All participants, including the project group members, the customer representatives, and the coach, have access to the board, making project progress and direction transparent among all the involved parties.

MMT is used to log work hours, weekly reporting and monitoring hourly work allocations, project progress, and risks.

In terms of communication channels, a Telegram group chat is used to support continuous communication among the group members, Vincit offers a Slack workspace for continuous communication between the group and the customer representatives and email is used, e.g., for sending video conference invitations and keeping in touch with the team coach. Phone calls are reserved for urgent communication and so far, there has been little need for making those. Google meet platform is used to meeting online with the client and Teams meetings is used for group's inner remote meetings such as weeklies.

We have agreed with the customer to produce minimal documentation. This means that we will include minimal deployment instructions, aim to write self-documenting code, and hand over the Trello board as a technical documentation. Of course, since the course requires documenting a formal project plan, testing plan, tests report, project report and a poster, these will also be made available for the customer for convenience.

To guarantee a peace-of-work for the development team during sprints, requirements are only edited together during sprint planning, if a need for significant changes arise. The customer has a chance to guide the development in sprint review meetings, and of course, the project management accepts feedback continuously and will manage possible changes.

4.2 Planned meetings

Group's internal meetings include regular meetings for weeklies, sprint retrospectives and sprint planning, as well as on-demand meetings such as internal workshops for various topics.

Group's internal weekly meetings are held every Monday either at noon, or right away after a sprint review meeting. Weeklies start with a round of "what I did last week, what am I doing this week and are there any obstacles". Alphabetized answering order is used for efficiency instead of awkward waiting of who dares to speak next.

Each responsibility area is handled during each weekly to make sure all project dimensions are progressing nicely and that we are on the same page with the next steps. Weeklies also offer a platform for free discussions, on on-demand basis, e.g., for finding out how busy everyone is, how everyone is doing in terms of motivation, and planning on work together.

We have also taken a practice of logging all hours from the previous week in MMT before the beginning of the weekly meeting and a thumbs-up/down round is exercised to make sure the manager can submit the weekly report after the meeting.

Since it is the end of the zero-sprint, the group has not yet established a refined practice on sprint retrospectives and sprint planning. However, the initial idea is to hold these meetings in the beginning of each new sprint, maybe right away after each sprint review, merged in the weekly meeting, or the next day – depending on how much load the team prefers on the same day. However, on management level, effort is put to make any and every meeting efficient to minimize wasting meeting participants' time and energy.

Internal workshops have been held already once, for UI design during the zero sprint, and another one is planned in the beginning of the first implementation sprint, about development tools and practices in the project and making sure everyone has their environments set up.

Other irregular meetings include mostly customer interactions. The first meeting with the client was held on August 31st and the second meeting – aka. requirements workshop – was held a week later. A UI design review meeting was held on 22nd September, and we have planned for two end user testing workshops – the first in the middle and the latter in the end of the project. There is also a possibility for a product release party, but this is not yet confirmed.

Then there are biweekly sprint reviews for the implementation sprints, where the coach, team, and the customer representatives are all present. Sprint review meetings are held regularly on every other Monday at 12 o'clock.

Last, but not least, course required meetings include first and final meetings with the coach and the team present, and there are mid-project and final presentations of the project with the whole course present.

4.3 Learning and study plan

In general, all members of the group will learn how to plan and work as a part of a modern software project, conforming to scrum practices. We will also learn some best practices of software development regarding designing, developing, testing and integration of software. We all will also learn something about customer interaction, collaboration, and the tools used in development.

In addition to the common learning goals, group members have individual focus points for learning, for we have assigned responsibility areas. Responsibility areas give leeway for everyone to take the time to get-to-know the quirks of the area. The manager learns most about project management, UI and UX design leaders focus on responsive mobile UI and UX design, and the architect learns most about the technical design for PWAs. The CI/CD specialist learns to build an operative CI/CD pipeline.
What makes the project a unique and efficient learning opportunity, is that we lead our actions and learning ourselves, which forces us to practice our judgement, although other members of the group and even the customer can support the learning.

On top of gaining specialist knowledge on our individual fields, we also learn from each other by sharing the most useful bits of information we find out as we work our focus areas. This means keeping others posted, consulting on problem areas and sharing links to, for instance, useful tutorials or blog posts. We also learn from participating into workshops, discussions and training sessions designed, hosted, or initiated by the other members of the group.

5 Project iterations and timing

The project was started already in the end of July, and it will last until December 22nd. The group does not have a shared calendar but has committed to bringing up possible exceptions in their schedule as soon as they are known. So far, exceptions have included the Product Owner having a week-long vacation in August, and the head of requirements and user experience has notified the group to have little time during the last two weeks of the project.

Other irregularities include the junior developers having relatively little to do in the zero sprint, since the beginning of the project considers mostly design tasks, which belong to the senior developers' areas of responsibilities. However, as the implementation work begins in Sprint 1, we expect the juniors to take bigger role, which will even out the invested effort among group members.

5.1 Iterations (sprints)

Project has one zero sprint, 5 implementation sprints and a quality assurance (QA) sprint. Each sprint starts on a Monday and lasts to the Friday of the following week. That makes seven two-week sprints, except for the zero sprint that started so early we decided it was best to make it longer for various reasons. During the QA sprint, no new features are implemented, since the sprint is dedicated to making sure the product is stable, responsive, and ready to be handed over to the client. The timetable for the project is presented in table 5.1 below along with an estimation of what work will be carried out during the sprints.

Table 5.1. Project schedule.

Sprint nb	Schedule (start/end date) High level content.
0	30.8.-24.9. Grouping, contacting the client, and gathering requirements. Setting up the development environments and working up a project plan.
1	27.9.-8.10. First implementation sprint. Building a skeleton project and implementing first features. Getting to know the tools, finalizing MVP UI and UX design.
2	11.10.-22.10. MVP feature implementations; UI and core functionality, test plan.
3	25.10.-5.11. End user mid-project testing, improvements design.
4	8.11.-19.11. Implementing the improved product design, extending the MVP.
5	22.11.-3.12. Final features.
6 – QA	6.12.-17.12. Quality assurance; last end user tests, system tests and bug fixes, handing the project over to client. Project and tests reporting.

Implementation sprints also include time reservations for implementing unit and integration tests. The principle is that developers will test their code as they implement new features.

5.1.1 Iteration 0

During the zero sprint, the group was able to successfully contact the client and divide responsibility areas for group members. We started with good velocity, which we have maintained throughout the sprint, and are set on continuing with the project.

We met the client twice over video conference calls during the sprint. In the first meeting, all participant introduced themselves and their roles in the project. We discussed about the product vision, including the need, current process and hopes for the solution. We drafted an initial specification based on the discussion, after which the specification was further discussed in the second client meeting, named specifications workshop. After the specifications workshop, we were able to form a detailed description of the requirements and classify the features by the MoSCoW prioritization framework.

We also drafted the first version of the user interface and held an UI review meeting with the client during the zero sprint. The user explained in more detail in chapter 6.3.

5.1.2 Iteration 1 (first implementation sprint)

The first implementation sprint will focus on the development of the most important features of the software. From sprint zero, we already have working proof-of-concept of the architecture, so this sprint will focus on a skeleton implementation for the actual app. Details of the planned features and a breakdown of the stories into tasks is documented on our Trello board. However, to summarize, we plan on bringing the following MVP tasks from the backlog into the sprint 1 backlog:

- User log in
- Seeing a list of available rooms
- Making a room reservation
- Default values for room reservation
- Showing the details of a reservable room
- Showing the details of user's reserved room

- (Maybe) Setting favourite office location
- (Maybe) Editing the favourite office location
- (Maybe) Editing a reservation: cancelling current reservation
- (Maybe) Editing a reservation: extending current reservation
- (Maybe) Integration with Vincit SSO

The developer team will implement these features starting from the top of the bullet list, proceeding to the maybes if there is enough time in the sprint to implement, test and integrate the features. It is estimated that at least the first six tickets can be completed during sprint 1 and the maybes are likely to be implemented only during the second sprint. The maybes are not brought to backlog until implementation of the certains are done, and the current estimate is that they will go to sprint 2.

Other tasks on the first implementation sprint include finalizing the UI design for the MVP and planning the extended implementation.

5.1.3 Iterations 2 to 5

After the listed MVP features, the implementation should continue with the *Should have* features, in an order discussed with the client once the MVP is delivered.

In addition to feature development, the second sprint includes designing and documenting a detailed test plan, which will require considerable work especially from the technical lead sector, that is, the architect and the CI/CD specialist.

The exact contents for the following iterations are difficult to plan in detail, since likely there will be some changes and the exact velocity of development work is impossible to predict. We will aim to proceed with feature implementation in a prioritized order, consulting the customer and end users regularly to keep the direction of the implementation work on the right track.

Our team does yet not use numeric effort estimations with the development work, since the estimates often vary greatly from actual hours, but introducing the estimates is under consideration for making project planning and communication with the client more transparent.

5.1.4 QA Iteration

The quality assurance sprint does not contain any new features, only minor feature improvements and bug fixes. The last sprint focuses on wrapping up the project, meaning a hand-over of the project to the client, documenting what we know will be useful for further development and reporting how the project went.

6 Requirements

6.1 Functional requirements (main goals)

The high-level functional requirements are listed below. For prioritization MoSCoW is used. Each of these requirements are within the MVP and thus are categorized into must-have requirements. However, each of these can be further divided into sub-functionalities and these sub-functionalities are divided into must-have, should-have and could-have requirements. These can be found in our backlog on our Trello board.

High-level functional requirements of the system:

- The system must allow users to log in with their Google accounts.
- The system must allow users to set a preferred office location.
- The system must show users the currently available rooms in their preferred office location and for how long a specific room is available and what features the room has to offer.
- The system must allow users to book a room for immediate use with a default duration of 30, 45 or 60 minutes.
- The system must show users their currently ongoing reservation.
- The system must allow users to add more time to their currently ongoing reservation.
- The system must allow users to delete their currently ongoing reservation.

A last high-level functional requirement of this system is the possibility to book other resources found at the office location such as the office car, flexible working stations, and other similar resources. This requirement however is last priority and fall under the prioritization of could-do or won’t-do. Nevertheless, they still need to be considered when developing this product and shall not be forgotten at any given point.

6.2 Non-functional requirements goals

6.2.1 Usability goals

The customer puts great importance in usability and UX, meaning that an extra step needs to be taken when considering the applications usability and UX goals and how to measure them.

Usability goals

- **Learnability** – Easy to use even when the design is met for the first time
- **Satisfaction** – Users will prefer to use this application compared to using the other alternative, Google Calendar.
- **Efficiency** – The user is able to achieve their wanted goal faster than trying to achieve the same goal on Google Calendar.
- **Few Errors** – Minimal errors should be encountered. However, if an error is encountered by the user, recovery from the error causes minimal stress to the user.

UX goals

- **Functionality** – The system meets end user needs and offers the necessary information and tasks for a user to achieve their goal.
- **Value** – The system provides value for the user and is desirable. An improvement in user satisfaction should be seen.

All of these goals are important but the most critical one is efficiency. To ensure that these goals are reached the minimum requirement is to hold at least one usability testing session together with the customer and end users once the MVP is done. The usability testing will require careful planning to fully cover every single usability and UX goal.

6.2.2 Performance goals

With mobile apps any performance done should be limited to below 3 seconds to keep the UX on an optimal level.

6.2.3 Reliability goals

Thought needs to be given to how the application will react to signal loss and sudden power loss and how it can recover from these states.

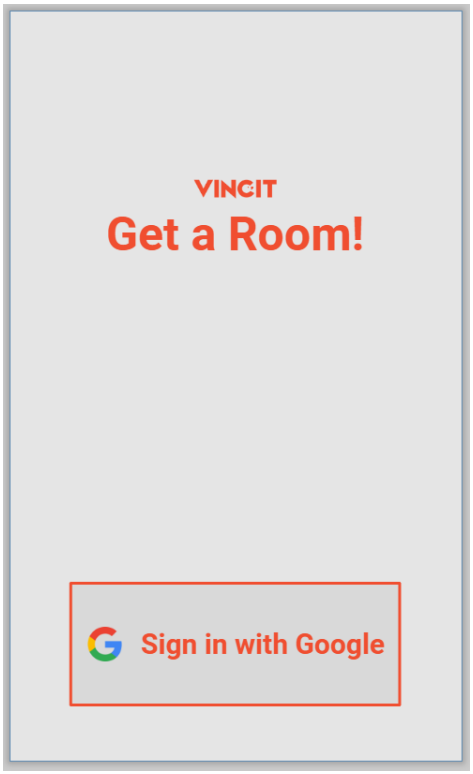
6.2.4 Security goals

Vincit's or Vincit employees' data should not be available for third parties. The most sensitive pieces of information in the system are the Google Workspace credentials of Vincit employees. Regardless, all application data should be protected from outside access.

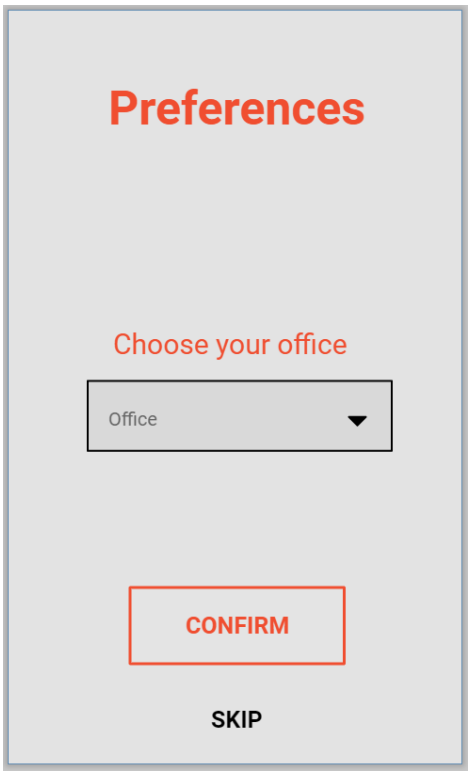
Since the application will be available publicly on the internet, the security goals will be achieved by encrypting all communications between client and server. Encryption will be done using a TLS certificate provided by Google for their cloud platform.

6.3 User interface requirements (main goals)

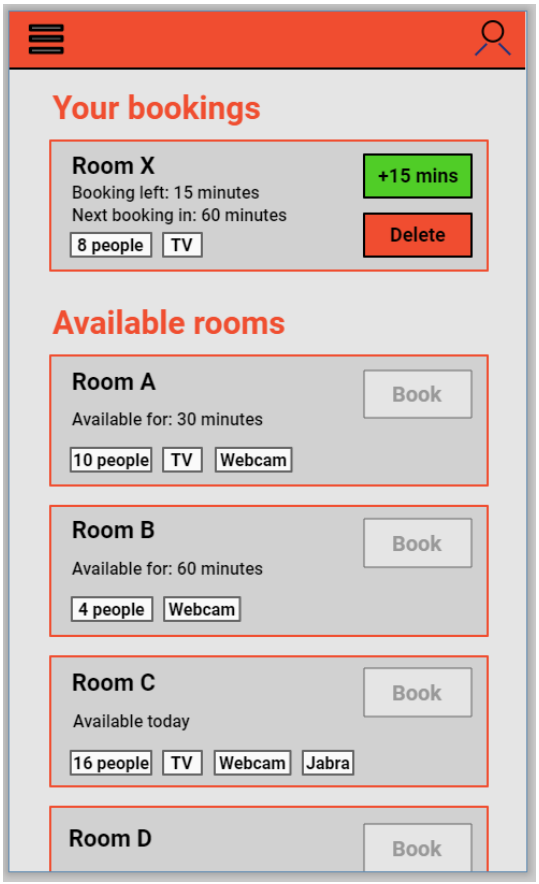
This section covers the first prototype designed for this application. These images have also been shown to the client and have been accepted for the MVP. These pictures were accepted for their functionalities, but the design will have to be looked over again



Picture 1. (left) Login screen (right) view for setting office preference.



Picture 2. (left) Main view used for looking at available rooms (right) popup view for when booking the room.



Picture 3. The main view for when a user has a booked room.

The most important aspect of this design is for the users to be able to quickly book a room that is available at the given moment with the minimum number of steps.

For further development, there should be some thought given to if the popup view could be skipped to fully match the clients wishes of being able to book a room by just one click. The client also suggested if the available rooms list could be made smaller. This means requirements might need to be revisited and find out if there are some types of information that shouldn't be shown in the available list or should be shown but instead with a 'expand' option.

7 Risk management

7.1 Risk list

Table 1.7 contains the initial list of recognized project risks. Each risk is assigned an numeric ID, and contains a description of the risk and its severity and a likelihood of the risk realizing.

Risk ID	Explanation, severity, probability, importance
1	Getting a suitable project – In the beginning of the project there was a moderate risk in finding a motivating project for the group that would allow for us to draw from our specialities. However, the risk of having to do an unfitting project did not realize, as we were able to work on the group's first-choice project.
2	Misidentified requirements – a somewhat unlikely, but important risk existed in misunderstanding or failing at gathering the initial requirements for the project. The risk of being unable to communicate the requirements with the client, however, did not realize. The client has been very fluent in communicating their wishes, open to the team's suggestions, and helpful in terms of technical support and taking a visionary lead.
3	A high-risk factor was solving some technical issues related to getting a development environment and suitable test data that would support the design, development and testing work. However, both the development environment and test data issues were solved already during the zero sprint.
4	The highest risk currently seems to relate to work hours division – more accurately an even division of effort among the group members, making sure no one grossly exceeds or falls below the target effort of 130 hours. In the beginning, the senior participants have logged a great deal more hours than junior developers, which of course is somewhat expected, since the design part falls to the seniors. This has been identified largely in the group and thus, is a topic in the sprint zero review meeting. There is also a moderate risk of the group members not finding enough time for the project, although so far it seems that the group has made steady progress, which lessens the likelihood of the risk of realizing into trouble.
5	Inadequate user involvement – Since usability is extremely important in this project, there is a constant need to involve users throughout the development.
6	Changing requirements – there is always the possibility of changing requirements, which can cause issues during development
7	Safari support – The client wants Safari support, and the development team don't have any Macs. Mac testing/development must be done using virtual machines and this might turn out to be time consuming if there is going to be many problems with the product running on Safari.
8	Full iOS PWA compatibility – It has been studied by the team that iOS PWA supports almost all the needed features for the product. The only question mark is the iOS PWA push notifications because they aren't officially supported on iOS PWA. Luckily, there should be alternatives for this. Also, it is not even sure yet if push notifications are wanted feature on the product.
9	Team members lack of understanding requirements – Needs to be ensured that everybody is on the same level with requirements
10	Underestimating the technical difficulty of some requirements – There might be some difficult features that haven't been identified yet
11	Somewhat unfamiliar technologies – Varying levels of skills in the team, Google Workspace and Cloud is new to everyone. Also, there is not much Apple development experience on the team.
12	Unstated requirements – Unspoken requirements that might appear later.
13	Neglect of non-functional requirements – A set of quality characteristics have already been identified, but there can still be a set of hidden quality characteristics.

14	Uncertain project scope – The project scope can be broader than expected if the project team completes everything that needs on time, creating the possibility for further development to take place.
15	To be determined requirements – some requirements are in a TBD state since it is uncertain if we will have enough time to do these. This creates a risk that specification and validation is done poorly for these requirements due to lack of time.
16	Free tier services – There are few free tier services used for development and hosting of the product. This might cause costs after the free tier ends or if the free tier is discontinued by the service provider. These services include GitHub and Google Cloud. Luckily the free tier of GitHub is unlimited in time and includes everything we need. The Google cloud free tier ends in 90 days but fortunately the client is already using plenty Google Cloud services, so they already know its costs and the ready product can be easily transferred to the clients Google Cloud to be maintained.

Table 7.1. Project risks.

7.2 Risk monitoring

The project manager has been tasked with monitoring the risks, although the initial risk listing was mostly drafted by the head of requirements and user experience. During the zero sprint, there has not been explicit tracking for the risks, but the risks have been monitored continuously and will be kept track of in MMT starting from sprint 1.

8 References

Villarreal, M n.d., "*The Difference Between Usability and User Experience*", Fuzzy Math, accessed 19 September 2021, <https://fuzzymath.com/blog/difference-between-usability-and-user-experience/>

9 Open issues

Most issues that deal with the requirements and require customer involvement are already settled, but one question is what will be the exact supported platforms – more importantly, whether we offer chromium support. There are still some other open issues that will be clarified once the project goes further. The biggest one is how this project will deal with the customer's flexible office desk reservations. Another issue is when exactly the end user test sessions will take place and who from the client end will be present there. Another issue is related to working methodologies, in the sense that if the project group will have separate sprint review sessions apart from planning for the next sprint or if these two can be merged.

10 Ideas for further development

Current further development ideas that will likely not be implemented during this fall include localization issues, maybe using GPS for exact user locating to improve resource recommendation accuracy and making further investigations on and implementing the flexible office desk reservations.