Risk Assessment and Management Plan (RMP)

The purpose of a risk assessment and risk management plan is to preemptively identify, analyze, and address the various potential risks inherent in our project. These risks may manifest in different types, such as technical or management risks. They also exhibit variations in risk scores, determined by comparing the impact and probability of each risk (refer to Figure 1). The Risk Management Plan (RMP) holds significant importance in any software project as it empowers teams to foresee challenges and explore potential solutions to issues that could arise, ultimately impacting the project's success.

Risks were identified during brainstorming meetings among as many teammates as possible. This was to get as many raw ideas as possible and to identify all possible risks with less chance of any slipping through the cracks, or personal bias limiting the scope of our assessment. Once we had all this raw data, a dedicated team of 3 students met to go through the risks and elaborate on them, as well as assign them impact and probability ratings from low to high. This led to the creation of the tables below.

Impact Probability	Low	Medium	High
Low	🟩	🟩	🟨
Medium	🟩	🟨	🟥
High	🟨	🟥	🟥

Figure [1]: Risk management chart

Risk ID	Risk Type and Description	Risk Score	Resolved in Sprint	Strategy and Effectiveness
US-1.1	● Technical ● Management ● External	● Low ● Medium ● High		● Mitigate ● Accept ● Avoid

	● Budget ● Schedule ● Etc.			● Transfer
T.1.1	**Technical** *Integration challenges between app and website:* The website front-end may not be completely responsive and the components developed in the website may not translate well in the app	High probability Medium impact	Sprint 2	**Avoid** through constant cross-platform and different screen size testing and using react/react native, a framework that facilitates code reuse
T.1.2	**Technical** *Unfamiliarity with the tech stack for front-end and back-end integration:* The teams unfamiliarity with the tech stack used can lead to integration difficulties	High probability Medium Impact	Sprint 2	**Mitigation** through training sessions for team members on the elements that are lesser known technologies, utilize team members strength and previous experiences, frequent meetings so there is coherence between front and back team, so collaboration is seamless.
T.1.3	**Technical** *Inconsistent data handling for different user types:* There are many different types of users with different requirements and views of the platform, this could result in inconsistencies and potential data errors	High impact Low probability	Sprint 2	Avoidance through making sure that the data handling protocols are clear and data validation checks are made through the development process
T.1.4	**Technical** *Dependence on external APIs:* Dependencies on external APIs for certain features may introduce risks, including changes in API specifications or unexpected downtime.	Low probability Low impact	Sprint 2	**Accept**, we have no control on external software and have no contact with API providers
T.1.5	**Technical**	High impact	Sprint 1, Sprint 2	**Avoid** by working on the domain model early on and

	Insufficient documentation for the system architecture can lead to data redundancy, lack of understanding and inconsistent updates when implementing the database	Low probability		ensuring collaboration between the development team and the team members working on the system architecture
T.1.6	**Management** *Disorganized teams* leading to double work. Multiple teams/members may work on the same features or systems and overwrite each other's contributions. As well as waste time in the process.	Low probability Low impact	Sprint 1, Sprint 2	**Mitigate** by having team leaders communicating frequently, a Github repository so all members are aware of current work, and task management software such as Trello to organize tasks.
T.1.7	**Management** *Not meeting deadlines* or meeting internal goals. Hard deadlines may be missed, disappointing the client and disrupting the overall development timeline	Medium probability Medium impact	Sprint 2	**Mitigate** by creating internal "soft" deadlines to be respected by the team, as well as encouraging communication and cooperation.
T.1.8	**Budget** *Limited budget* for tooling can present challenges since this is for a course.	Low probability Low impact	TBA	**Mitigate** by using free trials and free versions of tools. Some companies offer free premium versions for students
T.1.9	**Management** *Unavailability* of team members due to unforeseen circumstances can impact project progress	Low probability Medium impact	Sprint 2	**Transfer** Cross-training team members to be able to perform a variety of tasks and outsourcing tasks when necessary
T.1.10	**Technical** *Insufficient user testing* may lead to undetected bugs or usability issues	Medium Probability Medium Impact	Sprint 2	**Accept** It is an impractical goal to want to ensure with certainty that there are no bugs in the system. Our time and resources being limited, we will not be able to cover all

				bases. We still need to ensure an adequate amount of testing. We aim for 80% coverage.
T.1.11	**External** *Hacking into the mobile or web application*, allowing for abuse of the registration, financial, or reservation systems.	Medium Probability High Impact	Sprint 2	**Mitigate** by employing responsible coding practices and keep security in mind when designing the system.
T.1.12	**Schedule** Team may face *challenges in conducting regular scrum meetings*	Low Probability Medium Impact	Sprint 2	**Mitigation** by holding virtual meetings, utilizing when to meet, making meeting minutes so team members can stay updated and establishing clear meeting times to maximize attendance.
T.1.13	**Technical** *Neglecting core features* or requirements.	Low Probability High Impact	Sprint 1	**Avoid** by generating extensive user stories and thorough dissection of client needs.
T.1.14	**Management** *Resource Availability*: Potential shortfall in tools.	Low Probability Medium Impact	Sprint 2	**Mitigation** Resource planning, cross-training and contingency reserves.
T.1.15	**Management** *Lack of stakeholder engagement* in the project.	Low probability High Impact	Sprint 2	**Mitigation** Regular update meetings, stakeholder management strategies.
T.1.16	**Technical** *Non-adherence to UI/UX standards*: Failure to meet user interface design norms.	Low probability High impact	Sprint 2	**Mitigation** Following UI/UX best practices , user testing.
T.1.17	**Technical/Management** *Misaligned code management practices:* Inconsistent use of version control and coding workflows.	Low probability Medium Impact	Sprint 2	**Mitigation** Mandatory code review policies.
T.1.18	**Technical** *Requirements and user stories backlog mismanagement:* Ineffective tracking and	Medium probability High impact	Sprint 2	**Mitigation** Use backlog grooming sessions to keep the backlog updated and prioritized

	prioritization of project tasks.	🟥		
T.1.19	**Technical** *Design pattern misuse:* Incorrect applications of the design patterns.	Low probability High Impact	Sprint 2	**Mitigation** Conduct design review sessions.

Table [2]: List of identified risks