Agile Development Processes Postmortem Report

Team 4

Alex Tang, alexta@student.chalmers.se

Chi Hong Chao, guschaoch@student.gu.se

Ilja Pavlov, guspavloil@student.gu.se

Neda Amirabadi, nedaam@student.chalmers.se

Niklas Paasonen, paasonen@student.chalmers.se

Rohini Bisht, rohini@student.chalmers.se

Selomie Kindu Ejigu, guskinse@student.gu.se

Shilpa Vejella, vejella@student.chalmers.se

Viktor Vestlund, vikves@student.chalmers.se

Table of Contents

Project Description

This report provides a comprehensive description of development of a visitor's app for an amusement park using agile methodology. The visitor's app developed by our team gives the visitor an option to check the details of events, restaurants and activities going on in the amusement park as well as allowing visitors to book seats for them. Accounts for period tickets can also be accessed through the application to personalize the visitors' experience, or simply to check their details. The visitor's app will consist of a web based application that will be adapted for use with mobile devices.

The entire project will be carried out in Agile, with an iterative approach to deliver a project throughout its lifecycle. The work will be carried out in 3 sprints with each sprint having 2-3 weeks time frame. The team members will be self-organized into different roles to have a mindset of working in Agile teams (Scrum method followed).

1.1 Responsibility of Roles

Product Owner/Customer Deputy: The product owner is the customer proxy, the responsibilities of the product owner is to understand the needs of the customer, define the customer needs into features. Maintaining the product backlog that contains the features, acceptance criteria and benefits of the features.The product owner has to ensure that the user needs are met and comply with the definition of done.

Scrum Master Deputy: The scrum master is the team based servant leader, the responsibilities of the scrum master is to ensure that the team reaches the goal with the agile practise. It is the responsibility of scrum master to facilitate team events like planning, daily stand up, review and retrospective ensuring that the objective of the meeting is fulfilled and within a given timeline.The scrum master has to ensure inter team cooperation.

UI/UX Designer: Design the ideal look and "feel" of the application interface. A good design is very important as it helps the customer to visualize a picture of the product they are expecting. The UI team put in the design idea and set up a baseline that will be used in future sprints e.g choosing a consistent color palette for the application,font and text size. The design can be done one step ahead of the development team and that will have various advantages like the product owner can view the design and give early approvals thereby reducing the rework. The queries on new features can be raised early and discussed with the product owner before the actual development begins.

Developer: The developer has the main responsibility of the technical aspects of the project, taking the designs from the UI/UX team and turning them into reality. Here, the developer encompasses both frontend and backend development.

1.2 Technical Details

As stated earlier, the users will interact with a web based, frontend application. Using a responsive application design, it is adopted to mainly be suitable for mobile devices since it is the preferred device among users. The frameworks **Vue.js** and **Express.js** are used for the frontend and backend development respectively. They are both using the programming language Javascript, with Vue.js also including the languages **HTML** and **CSS** for web application development. The database is managed by the document based NoSQL program **MongoDB**. The tool **Docker** will be used for hosting and running the database locally on a computer. The framework Vue.js encourages the application to be divided into several individual components, which are structurally separated from each other. This allows for easy assigning of tasks among developers and at the same time reducing the risk of code conflicts.

The UI design is fully designed in **Figma**, a graphics editor and prototyping tool. There are four main options the visitor can choose from the base frame. The main options are **activities**, **reservation**, **tickets** and the **landing page**. Each option displayed in the landing page will be connected to its page detail.

Sprint 1 Log

During sprint 1, the base frame and landing page were fully designed while a base frameworks for tickets and prices, login page and activities page was also designed.

The development team was responsible for exploring the provided tech stack, deciding on additional technologies needed for the project (i.e. the choice of the database and containerization technology), and making sure that the CI/CD pipeline works. Outside of the technical questions, developers implemented the preliminary frontend and backend requirements. Backend development included making data persistent, writing tests, and writing code for the behaviour of the application. Frontend development focused on the creation of the "Hello World" MVP, including the coverage of the user stories, translation of the mockup designs, creation of the interactive user interface and connecting it with API.

Roles Distribution

Name	**Role**
Rohini Bisht	Product Owner
Shilpa Vejella	Scrum Master Deputy
Neda Amirabadi	Scrum Master Deputy
Selomie Kindu Ejigu	UI/UX Designer
Alex Tang	UI/UX Designer
Chi Hong Chao	UI/UX Designer
Ilja Pavlov	Developer
Niklas Paasonen	Developer
Viktor Vestlund	Developer

2.1 Commitment

Features/stories that the team committed to finish during the sprint are as follows:

Feature:

- The homepage for the app where visitors can see the various options for the amusement park with options to select different activities. The benefit of this feature can be defined as a starting page with brand logo and the service options.

Sub-tasks/Stories:

- Home page should be displayed with the amusement park logo.
    - Animated image of theme inserted into homepage.
    - Consistent Color scheme selected.
    - Information about amusement park addresses and contact details, opening times.
- Option for activities, tickets and prices, reservation:
    - Create a standard link button to use everywhere.
    - Test link button with different links.
    - Implement the return for the Activities page.
    - Test the return for Activities.
    - Implement the return for the Ticket Prices page.
    - Test the return for Ticket Prices.
    - Implement the return for the Reservations page.
    - Test the return for Reservations.
    - Mockup foundation for activities page designed.
    - Mockup for Login pages designed for Activity Reservation.
    - Mockup foundation for Tickets and Prices designed.

2.2 Activities

All activities performed during the sprint 1 are as follows:

Activity	**% of total sprint effort**	**Time estimated**	**Actual Time spent per team member**
Sprint Pre Planning	14%	18	All members : 2 hrs
Sprint Planning	14%	18	All members : 2 hrs
Development Tech. Setup	15%	20	Niklas Paasonen: 3 hrs Ilja Pavlov: 6 hrs Viktor Vestlund: 3 hrs Chi Hong Chao: 3 hrs Selomie Kindu 2 hrs Alex Tang: 2hrs
Standup meetings	3%	4.5	All members: 2.5 hrs
Sprint Requirement Analysis	6%	8	Rohini Bisht - 8 hrs

Sprint Design	14%	18	Selomie Kindu: 6 hrs Alex Tang: 7 hrs Chi Hong Chao: 5 hrs
Sprint Development	18%	24	Niklas Paasonen: 7 hrs Ilja Pavlov: 10.5 hrs Viktor Vestlund: 7 hrs
Retrospective	14%	18	All team members: 2 hrs
Sprint Management	8%	10	Shilpa Vejella: 5 hrs Neda Amirabadi: 5 hrs
Acceptance report writing	9%	12	Shilpa Vejella: 4 hrs Neda Amirabadi: 3 hrs Niklas Paasonen: 1 hrs Chi Hong Chao: 3 hrs Selomie Kindu 1 hr
Total Sprint 1	100%	150.5	

2.3 Reflections

Reflection on the agile practice practiced:

What worked well?

- Sprint 1 deliverables were completed in the given timeline.
- Separation of concerns - Development / Design / Management "departments"
    - Allowed us to focus on specific tasks, making it easier to concentrate
    - For example, with the PO responsible for deriving features and user stories, the rest of the team worried less about elicitation.
- General breakup of dependencies - All 3 departments could move forward in parallel without waiting for other departments.
- Experts - members confused about specific aspects could direct their questions at the corresponding department.
- Transparency - everybody could directly see work done from all departments
    - Allowed members to see what everybody else was doing
    - Could see decisions made in both work artifacts(Figma, Google Docs) and communication (Slack). This would create greater traceability in the future.
- Foundation setting - much of this sprint's work was done in the background.

    - The workflow was established, both on the team level and the department level.
    - Decisions were made quickly depending on the department, without having to seek approval by the rest of the team. For example, the technology stack was chosen by the development team after thorough research, without having to seek approval from all members.
- Scrum meetings and standup meetings:
    - By having these daily meetings, groups could examine progress towards Sprint 1's goals and consider sprint backlogs and adjusted upcoming plans.

What can be improved?

- Documentation - Much of the documentation was spread out across many different documents and sites.
    - Many documents were outdated quickly, or simply not updated after creation. This created confusion as there was too much to keep track of.
    - Overly detailed documentation.
- Workflow
    - Issues with workflow arose as a result of dependencies, unmentioned department-level decisions, or simply things we did not think of / consider when making the decisions.
    - Heavy dependency on PO for eliciting requirements
    - Perhaps include people from the design team (1+1 interview team) in the client meetings, or schedule separate meetings for the design team members with the client in order to elicit design requirements directly (design presentation/Q&A meetings).
    - The efficiency of the standup meetings could be improved in the future.
    - Lack of development tasks outside of Development department

      With such a large focus on setting up and figuring out the responsibilities of each role, coding and general development tasks were largely neglected. This creates an issue in the next sprint, where much of the team has to play catchup in order to familiarize themselves with the technologies and the technical workflow, given that all members need to contribute to the code.

How was the work distribution within the team?

- The design team focused on setting up Figma, getting every member in the same workspace, and perhaps most importantly, designing mockups for both this sprint and the following one. Within the design department, work distribution has been relatively equal between members since much of the work was done in meetings. Compared to Development, however, Design has spent less hours in total.
- Within the management team, the work was distributed equally among the members. The Scrum master deputies had separate meetings to discuss on the sprint plannings and about the management of the team. Product owner arranged a meeting with the customer to discuss the flow of the requirement and the deliverables for the first Sprint 1.

- The development team was tasked with primarily deciding on what technologies to be used for the project, setting up and preparing the tools, CI/CD, the backend, preliminary data structures, documentation and database for future development as well as constructing a “Hello World” frontend application. Within the development team itself, the user stories were discussed, broken down into the technical checklist and divided among the members.

What changes will you do for the next sprint?

- Setting up and getting familiar with the tech stack up to the point where coding is possible for the entire team, as fast as possible.
- We anticipate that as time goes on, less design work will be needed and more development work will be required. At this point, however, there is seemingly still much to do in terms of designing the UI for the upcoming features.
- Time keeping will be simplified, and rather than being determined in a meeting, it will be done individually.
- Tasks in the Kanban board will be created and pulled from the backlog if there is slack, rather than asking the Product Owner first.
- Free time of members should be considered to be able to set meeting times during the sprints in an appropriate way.
- Documents and Trello board should be kept updated.
- The PO should have less responsibility in creating and maintaining user stories.
- Pre planning of the sprint work needs to be focused more.
- Internal movement of resource roles giving everyone equal opportunity to work on the roles they like.
- Utilizing the meeting time to stick with the agenda of the meeting.

Sprint 2 Log

We started with the sprint pre-planning process where the roles are interchanged among the team members and the product features and user stories for this sprint were analysed. As the design base framework was completed in sprint 1, the dependency on the design team was reduced; hence, the design team size was reduced and more members were added in the development team. Since new members were added to the development role,a training session was organized for the new members to understand the system and the development completed in sprint 1. Sprint ceremonies were conducted which included Sprint Planning, daily standups, refinement, review and retrospective. Sprint 2 focused on the development of the “activities page” and part of the “ticket page”.

The vision for our team is to complete the customer deliverables with high standards and familiarize with the agile practises. For sprint 2 our team objective/mission is to learn more about agile practises,follow all the ceremonies and understand the benefits of the ceremonies. Have a good team collaboration and as roles are exchanged in this sprint we want each member to learn about the roles and the responsibilities.

Roles Distribution

Name	**Role**
Alex Tang	Product Owner
Rohini Bisht	Scrum Master Deputy
Neda Amirabadi	Scrum Master Deputy
Selomie Kindu Ejigu	UI/UX Designer
Niklas Paasonen	UI/UX Designer/Developer
Chi Hong Chao	Developer
Ilja Pavlov	Developer
Shilpa Vejella	Developer
Viktor Vestlund	Developer

3.1 Commitment

Features/stories that the team committed to finish during the sprint are as follows:

Features:

- **Amusement park homepage:** An Interactable map on the homepage displaying all the activities available in the park. When an activity is clicked, visitors should be navigated to the activities page.
- **Activities page:** A page showing all of the activities available at the park with a filter system so the user can filter which activities to be shown.
- **Activity specification details:** Pages for each specific activity which can be accessed from the activity page and map from landing page.
- **Ticket and prices:** Details about the different tickets and pass options. The benefit of this feature is that users having prior knowledge about the park can directly book single or periodic tickets.
- **Reservation:** The user can reserve tables at restaurants and places on rides for specific times through this page. The benefit of this feature is that users do not need to stand in a physical queue.

Sub-tasks/Stories:

Dev stories

- As a visitor I want a list of all activities shown as cards and be able to scroll down the list without interruption.
- As a visitor I want to be able to visit the full page with all activities after clicking on a specific card from the activity page.
- As a visitor I want to filter out specific results to be shown on the activity page.
- As a visitor I want to be able to access a map which displays all of the activities available at the park.
- As a visitor I want to know where my selected activity lies on the map.
- As a visitor I want a page where I can enter details required for buying a ticket.

Design stories

- As a visitor I want the activities page design displaying all activities, designed with the brand colors and attractive animation images for specification of each activity list.
- As a visitor I want to have an option of filtering activities.
- As a visitor I want different activity pages designed with a detail of activity for a specific activity.
- As a visitor I want to have a home page designed with a map which displays all of the activities available at the park.
- As a visitor I want the filter options designed in the activity page with easy usability features.
- As a visitor I want the tickets and prices page to be designed listing available packages.

- For a specific package page the details of the ticket, price and book option display should be designed.
- Ticket and prices for Period pass login page should be designed.
- Ticket booking page with detail and payment should be designed.

3.2 Activities

All activities performed during the sprint 2 are as follows:

Activity	**% of total sprint effort**	**Time estimated**	**Actual Time spent per team member**
Sprint Pre Planning	9.09	36	All members
Sprint Planning	2.27	9	All members
Standup meetings	6.82	27	All members
Training session	6.82	27	All members
Sprint Requirement Analysis	4.92	19.5	Alex Tang - 18.25
Sprint Management	9.85	39	Neda Amirabadi - 18.5 Rohini Bisht-18.5
Sprint Design	9.85	39	Niklas Paasonen-5 Selomie Kindu - 19.5
Sprint Development	19.70	78	Chi Hong Chao -19.5 Ilja Pavlov -18.25 Niklas Paasonen-15.5 Selomie Kindu -5 Shilpa Vejella -18.5 Viktor Vestlund - 18.25
Sprint Review	1.14	4.5	All members
Sprint Refinement	4.55	18	All members

Sprint Retrospective	2.27	9	All members
Learning	22.73	90	All members - Time for software setup, code merging, learning the concepts
Total Sprint 2	100.00	396	

3.3 Reflections

Reflection on the agile practice practiced:

What worked well?

Decisions	**Motivation for Decision**
Time Availability Sheet	Scrum Masters can schedule meetings with the team more easily, and members can fill/modify their times in their own free time. This allows the team to see when everybody else is available without much documentation.
Stand up meetings	It helps in knowing the progress and achieving the sprint delivery. As the meetings were attained by all team members(those who could not join the meeting informed the scrum masters or were approached by the scrum masters). Stand up meeting time was 15-30 mins.
Time Reporting during Daily Standups	The PO and Scrum masters need it to estimate/negotiate sprint deliverables; everybody knows the time spent by everybody else. Since everybody needs to be in daily standups, the face to face communication style efficiently logs the time reporting too. Finally, it increased trust between members as it showed that everybody was working.
Clearer, Structured Time Report	Everything in one place - more maintained, organized time reports again allow management to estimate time more easily. Using a spreadsheet also allows for more automation, therefore less manual documentation maintenance
Meeting logs responsibility given to scrum masters	Placing this responsibility allowed for better maintained meeting logs. This was done because the logging was not maintained well last sprint, since the responsibility was not given to a specific person or group.
More Organized, Maintained Documentation	While documentation has not necessarily been less than Sprint 1, it has been more centralized and categorized properly. Documentation folders in google drive for each role have been better maintained, and several

	documents have been merged together or deleted for easier maintenance and greater simplicity. This increased efficiency and more importantly, each piece of documentation served to promote sustainable development.
Development Training Meeting	There was a large technical gap between the dev team and the rest during sprint 1. To alleviate this issue, a training meeting was made to "keep everybody on the same page". This helped with keeping an acceptable level of technical excellence, since it helped the team understand the basic technical design and structure in our project.
Design Team made Smaller	With much of the design foundations established, we decided to move part of the design team to other "departments". This was done because we anticipated that more work would be required on the development side and less on the design, as well as the need to get members who have not worked on development up to speed. Allowing us to self organize like this depending on the situation let us embrace change to maximize efficiency.
Roles Rotated for some members	While some members maintained their roles from the last sprint, others changed roles to try new things, or to fill in the required manpower of the dev team. This resulted in a better cross functional team, as members who moved between roles better understood the project as well as being more flexible, making us more adaptive to change.
Collaboration	The team synchronization was good, each member was approachable. Blocker issues were highlighted and discussed by the team members. The development team had a self organized meeting to discuss any errors/ issues and it was resolved on time.
Response from customer	The queries were asked to the customer through email, the customer responded to all the queries. It helped in designing the sprint 3 deliverables
Sprint review with the PO and Team session with customer	Giving a demo to the customer in the middle of the sprint to know if we are going in the track of what he was expecting from us and to know what he is expecting more from the feature. This helped us in understanding if the customer is satisfied through the early sprint.
Change request incorporation	There was a practice of changing some requirements that are done in sprint 1 which has one advantage of agile development practice. Also an additional customer request was added in the sprint delivery. Some additional points are provided by the customer that we have planned to include in the next sprint. There was continuous technical implementation attention in gitlab and also designs are improved as per new customer requirement.

Quantifiable measurement	Completing user stories and progress to achieving features completion was a measure of the progress of the current sprint.
Approach discussion	Deciding and agreement of every member in the team helped in setting up the sprint and achieving the target.

What can be improved?

What can be improved	**Action Items**
Develop a standard practice for merge request approving, i.e integration meetings by the end of the week.	In the next sprint we will have a weekly integration meeting and the code will be merged on every friday.
Schedule common work session hours when members can drop in into a common room to work on the code / Q&A / etc. Action decision	A meeting will be arranged every wednesday. The meeting will be optional and members can join if they need to discuss any points with other team members.
More scheduled zoom meetings between product owner and customer.	The product owner in the next sprint needs to discuss with the customer how the meetings can be arranged.
We can have task setup for each user stories at the time of planning, so it will be easy to understand the process for the new members who are moved to development	In the sprint planning we can focus on the task as well or a separate meeting can be arranged within the development team to discuss.

Should stick to the time frame of having only 15 mins stand-up meeting as decided by the team.Suggestion: Can have alternate day stand-ups	After the standup decision there were discussions about other topics related to the subject. Hence from next sprint we will have a standup meeting for 15 mins and 15 mins for general discussion. The general discussion part is optional and members can leave the meeting if they have nothing to discuss. Also the daily standard up meeting will be shifted to alternate days meeting so from next sprint we will have 3 daily scrum meetings every week (30 mins meeting = 15 mins for stand up + 15 mins general discussion) per week. Time box limit for stand up - 2 mins for each member.

How was the work distribution within the team?

- **Product Owner**
  - The PO discussed the requirements with the customer and provided the ideas/inputs beneficial for the project. The project scope for the current sprint was discussed with the customer based on which the product features were prioritized.
  - The product owner shared his idea with the team and the approach was finalized. The product backlog was maintained by the product owner ensuring that the acceptance criteria covers all the points discussed with the customer.
  - The product owner collaborated with the team during the development process.
  - The PO was aware of the development progress. The project progress was shared with the customer during the teamwork session, the partial work was shared with the customer so as to ensure that the customer requirements are fulfilled.
  - Features for the next sprint were discussed by the product owner with the customer.
- **Scrum master**
  - The Scrum masters prepared the timeline and did the planning for the sprint.
  - Scrum ceremonies were arranged and conducted by the scrum masters.
  - It is the responsibility of the scrum master to support the team and ensure that the committed objectives of the team are met.
  - The scrum masters are responsible to help in removing the impediments found during the sprint.
  - They kept the documents and Trello board updated so that all members were informed about plans and meetings and progress of the assignments and what they needed for doing their tasks in an appropriate time.

    - The scrum masters worked on the improvement factors raised as part of sprint 1, they ensured that the points raised in the last sprint can be worked upon in sprint 2.
- **Design Team**
    - The design team were responsible for the UI design for sprint 2 and sprint 3.
    - The team spent more time at the beginning of the sprint in order to ensure that the design is ready for the development team.
    - The UI specs are also designed for the next sprint.
    - The design team collaborated with the product owner and the design was approved before the development.
    - The design team members divided the user stories among themselves, stories status was updated in the Trello board.

- **Development Team**
    - The dev team members have discussed the stories and came up with a development approach. Each member picked up individual stories and focused on completing the stories.
    - Development user stories were added in the trello board, development tasks were also added by individual members for their user stories.
    - The development of the user stories was done in increments and the team provided updates on their stories in the standup meeting.
    - The dependencies between the user stories were discussed by the team members.
    - The team members collaborated among each other to resolve the issues they faced.
    - The members who have worked in development last sprint supported the other members who were new to development.
    - The Dev team ensured that the code is ready for the customer review in the second team work session.

What changes will you do for the next sprint?

- Roles will be rotated in the next sprint.
- We will work on the action items in the improvement section.
- More members will be added in the development team as the majority of design work is completed.

Sprint 3 Log

While it has been stated that Sprint 3 was not “the final sprint” in terms of the project, there was a certain amount of pressure internally since it was the course’s final sprint. This led to a larger focus on development, and a sense of urgency to at least provide a product with its main functions completed. This sense of urgency, compounded by a much shorter sprint, induced decisions to either reduce the development team’s workload or increase development efficiency.

Continuing from Sprint 2, some members’ roles were again changed, not only to allow members to experience different roles but also to increase development speed. This led to the majority of the team being diverted to a developer role. One of the downsides was that there were new developers new to the codebase of the project, leading to some learning time in the beginning.

The sprint began with the product owner and the scrum master doing an initial sprint planning where the sprint feature requirements were analyzed and all of the user stories were written. Furthermore a prioritization ranking was added to each user story ranking from “high” to “medium” to “low”, trying to give the team suggestions on which stories to begin with during this sprint. The rest of the team later had the opportunity to review the user stories and change them as desired. A final group meeting for the sprint planning was held where the user stories were divided amongst the group members. While we are aware that the development team should normally be more involved in the creation of user stories, the sprint’s time was too short, making us take drastic measures to compensate for it.

Communication between management and developers was increased, and the PO took a more hands-on approach to reviewing and testing user stories. However, dependencies between some user stories delayed some sections of development.

Roles Distribution

Name	**Role**
Alex Tang	Scrum Master Deputy
Chi Hong Chao	Product Owner
Neda Amirabadi	Developer
Selomie Kindu Ejigu	Developer
Niklas Paasonen	Developer
Rohini Bisht	Developer
Ilja Pavlov	Developer
Shilpa Vejella	Developer
Viktor Vestlund	Developer

4.1 Commitment

Features/stories that the team committed to finish during the sprint are as follows:

Features:

F1	**Amusement park homepage:** ● The bug of not having all of the activities being shown on the homepage map was fixed.
F2	**Activity page:** ● Added subfilters for each activity eg “water ride”, “fast ride” and “family ride”. ● Changed so the different activities follow the same price plan depending on their subcategory.
F3	**Activity specification details:** ● Added reserve button from activity specification details which takes the user to the reservation page. ● Added a diagram of busy hours for each specific activity. ● Fixed bug so each activity is shown on the map.
F4	**Period tickets and prices:** ● Added each ticket type to the backend of the system. ● Functionality of viewing “My Tickets”. ● Discount system for children.
F5	**Reservations:** ● Viewing of a list of activities which are bookable. ● Being able to book a specific activity. ● Busy hours being shown for each bookable activity. ● “My Reservations”
F6	**Confirm payment page:** ● Confirmation of user purchasing an item.
F7	**Policies:** ● Policies page added reachable from the home page.
F8	**User Authentication:** ● Storing user information about reservations and purchases.

Dev stories

- As a visitor I want to be able to view all of the activities in the map on the homepage - F1
- As a visitor, I want to be able to view where it's located on the map for each activity. - F3

- As a visitor I want to see only specific types of attractions - namely, **Fast, Family, and Water** rides. I also want to only see specific restaurants - **Bistro, Cafe, and Dining** restaurants**.** I also want to see specific games only - **Arcade, Kids, and Lucky** games. - F2
- As a park owner, I want my rides to keep consistent prices for my different sub categories of rides. -F2
- As a park owner, I want the colour scheme for the activity page to be consistent with the rest of the app - F2
- A a user I want a button in the activity specification detail to take me directly to the reservation for that specific activity - F3
- As a visitor, I want to see the busy hours of each activity so I can schedule my time around activities with long queues. - F3
- As a park owner I want children to have cheaper tickets. - F4
- As a visitor, I want to see a list of my purchased tickets. - F4
- As a visitor, I only want to enter the park to see the sights and eat in restaurants. - F4
- As a visitor, I want to buy tickets to participate in a few activities in the park. - F4
- As a visitor, I want to buy tickets to participate in many activities in the park. - F4
- As a visitor, I want to access all activities for an unlimited amount of time for the entire day, as well as get access to reserving activities. F4.
- As a park owner, I only want full package ticket holders to be able to make reservations in my park -F5
- As a user, I want to be able to view every reservable activity in the reservations page - F5
- As a user, I want to be able to reserve specific time slots of each activity - F5
- As a user, I want to be able to cancel reservations which I have already made - F5
- As a user, I want to view all reservations I have made currently. - F5
- As a user, I want to able to view the parks policies - F7
- As a park owner, I want to know which user has which ticket and reservations. - F8
- As a user I want a confirmation page when I have bought a ticket to show me the details F6

4.2 Activities

All activities performed during the sprint 3 are as follows:

Activity	**% of total sprint effort**	**Time estimated**	**Actual Time spent per team member**
Sprint Planning	5.11	9	Alex Tang - 4.5 Chi Hong Chao - 4.5 Sum - 9
Standup meetings	6.82	12	All members - 12
Sprint Requirement Analysis	3.41	6	Chi Hong Chao - 9
Sprint Management	2.84	5	Alex Tang 6.5

Sprint Development	73.86	130	Alex Tang - 6 Ilja Pavlov - 15.5 Neda Amirabadi - 19.5 Niklas Paasonen - 20 Rohini Bisht - 21.5 Selomie Kindu - 19 Shilpa Vejella - 16 Viktor Vestlund - 22.5 Sum - 140
Sprint Refinement	2.84	5	Chi Hong Chao - 3
Sprint Retrospective	5.11	9	All members - 4.5
Total Sprint 3	100	176	181

4.3 Reflections

Reflection on the agile practice practiced:

What worked well?

Decisions	Motivation for Decision	Relation to agile practices
Fewer and adjustable stand up meetings.	By having fewer stand up meetings we could bring more collective questions and progress into the regular stand up meetings. The meetings felt more efficient as well. However, we had more frequent standups when much more development is being done.	Simplicity of having more necessary and content based meetings. Furthermore increasing efficiency by not having unnecessary meetings.
Scrum master and product owner developing user stories.	During this sprint, the user stories were fully developed by the scrum master and the product owner during an individual meeting. The stories were later put into review by the development team where they could raise concerns and questions regarding the stories. This made the sprint planning process more efficient and gave the team more time to review the content of the sprint.	Increasing work efficiency and team reflection opportunities.
Continuous communication between scrum master	Since the team decided to decrease the scrum master roles from 2 to 1 during this sprint, there were questions which the scrum master wanted to discuss, which had to be done with someone	Satisfying the customer throughout the process and having daily collaboration within the team. Greater

and product owner.	outside the role. By having continuous discussion between the product owner and the scrum master the sprint felt more in control where both the customer's requirements and the team's capability were discussed more thoroughly and in detail.	information flow efficiency.
More involvement from the management team in development .	Both the product owner and the scrum master attended several meetings with the development team to understand queries and the code itself. Furthermore, “pair programming” was done between management members and the development team to both help development and let management get more insight in the development.	Business people and developers working together towards the final product and increasing team reflection.
Added more but smaller user stories.	By trying to break down the user stories into smaller parts meant that the developers could more regularly pull new user stories to work on from the backend. Furthermore it gave the developers an opportunity to push changes more frequently as well.	Simplifies measuring of software progress. Deliver working software more frequently. Can divide work amongst different members more easily.
More regular pushing of branches on gitlab for review	The different developers pushed their branches more regularly to Gitlab for the rest of the team to review. This gave the team an opportunity to in a sooner stage find bugs or requirements changing.	Simplifies measuring of software progress. Deliver working software more frequently.
Product owner deputy reviewing/testing user stories during sprint	While the automatic testing pipeline helped make sure basic functionality was working, there were some issues that could only be seen during manual testing and review. These reviews helped align what was promised for the sprint and reduced potential future problems. It also lets the PO see current progress, and negotiate with the customer if falling behind. It also helped the PO deputy reduce workload as seeing the progress allowed suggestions/refinement to be made during the sprint rather than only during sprint review.	Sustainable development requires the PO to understand the current progress of the team, determining what is feasible. Letting management take a more hands-on approach also ensures technical issues are not neglected, leading to better working software.
More team members in Dev team in Sprint 3	As there was more work left in the development phase by the end of the project, the decision of moving more team members to the Dev team in Sprint 3 helped in finishing the development work on time.	Allowing us to self organize like this lets us be more flexible and effective. It also helped reduce “useless” work - for this sprint, the most value for the customer was working software delivered, not the design or management.

Members choosing between frontend and backend work	Team members who have good knowledge on backend development worked on implementing the features, whereas members who are good at frontend concentrated on their respective frontend developments.	With a shorter sprint, efficiency was much more important. Members with more specialized knowledge doing what they knew allowed for better technical design but at a lower risk and time cost.
Dedicated roles within development	The development team knew where to bring specific development questions since we had divided feature specifications within the team.	Issues were resolved more quickly, and while face to face communication was not possible, video calls were often used to further resolve these questions quickly.

What can be improved?

What can be improved	Action items	Relation to agile practices
Better time estimations on user stories	Since it was only the product owner and the scrum master which decided the time estimations on user stories, (with possibilities for the rest of the team to change them during a couple of days), the estimation was not very accurate on each user story. Instead we could in future projects use techniques such as planning poker with the whole team involved to better estimate time.	Business people and development teams working together.
More frequent conversations with the customer	To achieve more frequent conversation with the customer we could invite him to our slack channel instead of holding the conversations solely via e-mail. Slack would have been able to ping them, creating stronger notifications. The PO deputy took initiative to contact the customer, but to no avail.	Satisfying the customer throughout the sprint.
More clear information about user stories	When there is any update on the feature(s) from the customer, the backlog sheet should be updated/improved every time for the related user stories to make them more clear for the other team members.	Updating the backlog allows the team to better adapt to changing requirements, making it more efficient to provide working software.
Consistent sprint lengths needed	Having consistent sprint lengths would help the team to better plan and execute work - which was not so much the case this sprint.	Consistent sprint lengths would allow us to better anticipate velocity, leading to more sustainable development as it would help us better negotiate sprint deliverables.
Outdated design	During meetings design thoughts could change but the actual design did not get updated. Because of this some confusion occurred, leading to inconsistencies in UI and outdated backend design. For future sprints, although we might not need a specific design team member, someone still needs to take responsibility updating the design.	While working software is important, the right deliverable is just as important for customer value. Having an updated design promotes consistency and better information flow, leading to more valuable software more efficiently.

How was the work distribution within the team?

- **Product Owner**
    - The PO discussed the requirements with the customer and provided the ideas/inputs beneficial for the project. The project scope for the current sprint was continuously discussed with the customer based on which the product features were prioritized.
    - The product owner shared his idea with the team and the approach was finalized. The product backlog was maintained by the product owner ensuring that the acceptance criteria covered all the points discussed with the customer.
    - The product owner collaborated with the team during the development process. This ranged from syntax errors to general high-level feature decisions.
    - The PO was aware of the development progress. The project progress was shared with the customer during the teamwork session, the partial work was shared with the customer so as to ensure that the customer requirements are fulfilled.
    - The PO reviewed and tested the committed branches pushed to gitlab frequently to see if they met the given requirements. Comments and suggestions were then given to further align the software with the vision of the customer, or to reduce the workload of development.
    - The PO maintained and updated the product backlog on a regular basis, both in Trello to clarify issues and the spreadsheet.
    - Alongside the SM, the majority of the feature acceptance criteria and user stories were made.
- **Scrum master**
    - The Scrum masters prepared the timeline and did the planning for the sprint.
    - Scrum ceremonies were arranged and conducted by the scrum master.
    - The Scrum master supported the team and ensured that the committed objectives of the team were met.
    - The scrum master was responsible for removing the impediments found during the sprint.
    - The SM kept the documents and Trello board updated so that all members were informed about plans and meetings and progress of the assignments and what they needed for doing their tasks in an appropriate time.
    - The scrum masters worked on the improvement factors raised as part of sprint 2, they ensured that the points raised in the last sprint can be worked upon in sprint 3.
    - Alongside the PO, the majority of the feature acceptance criteria and user stories were made.
    - The SM also is responsible for time keeping.
- **Development Team**

- The development team members have discussed the stories and came up with a development approach. Each member picked up individual stories and focused on completing the stories.
- Development user stories were added in the trello board, development tasks were also added by individual members for their user stories.
- The development of the user stories was done in increments and the team provided updates on their stories in the standup meeting.
- The dependencies between the user stories were discussed by the team members.
- The team members collaborated among each other to resolve the issues they faced.
- The members who have worked in development last sprint supported the other members who were new to development.
- The development team ensured that the code is ready for the customer review in the second team work session.

What changes will you do for the next sprint?

- Although most of the design is completely finished, a specific team member should still be responsible for updating and maintaining the designs on figma. This team member's main role can still be within either management or development.

- By lengthening the next sprint to keep each sprint length consistent, better structure could be put into the sprint planning so every member could impact the user stories. Since only the management team wrote the user stories this time, the time estimates could be improved by involving more team members and using, for instance, planning poker.

- The user stories were exactly the same on both a spreadsheet and Trello this sprint, meaning that changing one required updating the other. The spreadsheet was useful for time estimates, however. One change would be to remove the acceptance criteria in the spreadsheet and the time estimates in Trello, increasing the usefulness of both parts of the documentation.

Since there are developers having more knowledge on certain features, making the roles official in order to get others up to speed to have everybody reach the same level of knowledge.

Postmortem

5.1 Reflection

General Experience

From the beginning, we were aware that being flexible and open to change, along with customer value, was important, but *how* to do so was where the majority of the learning took place. These goals of maximum agility and maximum efficiency often clashed however, especially during the beginning when the roles and responsibilities were not solidified. We then iteratively improved upon our process throughout the project.

Impact of Course Meta-Requirements

While a certain degree of freedom was given to the team, the course structure and grading system drastically affected the team's process. As such, our foundation was influenced by the schedules of the customer meetings, the way hours spent was used as a means to measure work, the emphasis of the entire team needing to code, and the need for a PO and Scrum master.

Logging hours spent along with the sprint report deadlines affected our process foundation. There was an impression that we would be evaluated mostly based on **hours spent, commits, and the sprint reports.** Yet, the role requirements divided assigned responsibilities to each department. Separating responsibilities made it difficult to justify equal workload, especially in the management section. The development team made commits and had longer work hours. Writing emails would not take 3 hours. In fact, something would be wrong if management time was greater than development.

As such, role rotations came into play. This allowed members to contribute to code but also experience other roles throughout the project, and in theory worked well. We failed to anticipate the uneven sprint lengths and increasing complexity of the project leading to drastically lower efficiency.

This "work hour calculation" emphasis was concerning. On one hand, agile emphasized honesty, transparency, directness, and efficiency as the lectures describe. On the other hand, reporting less hours meant you did "less work", making it wonky in a team setting. While our team did not have honesty issues, many discussions revolved around equal workload early on.

In hindsight, more members could have worked on code earlier, reducing training time later. A larger development workforce should have been made earlier. Effort evaluation based solely on work hours for management(and to an extent, design) is still strange - rather, more valuable attributes would be *less* hours - quick response times between departments, faster decision making, anticipating possible issues and mitigating risks before they appear. Mock ups need to be completed quickly before development can copy the layout. More efficiency should not mean more work hours.

Yet, the emphasis on work hours and collective code was useful. Members could see the intricacies of development, work hours were used to negotiate deliverables, and it simply showed that everybody was working. The emphasis on code created a greater sense of code ownership, showed how aspects of the project connected to each other, and in our case, allowed us to try out different roles.

Roles

The PO and Scrum master deputy requirements prompted us to explore other roles, forming different departments. Unable to estimate workloads for each department due to unclear responsibilities, we simply delegated roles according to individual preferences, leading to low efficiency. To complicate matters, we also divided development itself into backend and frontend, specializing members more. As a result, the workload between the development department and the rest of the team was quite unbalanced in the beginning.

Having more people assigned to a certain department actually could be seen as generating more work down the line instead of less. In the case of the design team, making the initial designs consistent required long meetings. Moving more people to development should have been done earlier given the short timeframe. It also led to more inconsistencies which required time to fix later on.

Role rotations brought interesting insights, as mentioned above. In addition to letting everybody try what they wanted to, it allowed some to be in charge of the changes they wanted to make. Rotations helped normalize flexibility - knowledge transfer sessions became more frequent, encouraging team communication. Furthermore, it encouraged situation-based role modifications - increasing the development team was smoother and sustainable because it became normal to factor in onboarding time. Members who changed roles had a better understanding of the whole project and process. In fact, some members had experience of having roles within an agile setting, accelerating the transitions.

Yet, onboarding still took a considerable amount of time. Without role rotations, more would have been delivered, without needing to crunch in the final sprint. However, there would have been less code ownership and general project understanding. This could be mitigated if management participated in development to familiarize themselves with fundamental concepts.

Documentation

Documentation was created for everything that was done in the beginning, but only some was maintained. This led to much redundancy, often with conflicts against outdated information. The lack of maintenance was due to a few factors: unclear responsibilities and too much fragmented documentation.

As more sprints passed, more retrospectives and planning led to more documentation being merged or removed. The management department also ended up maintaining much of the documentation, other than the backlog and user stories. This streamlined the process and helped emphasize the importance of minimal documentation.

Corresponding Experience with Literature

Common Challenges

There was initially a larger focus on sticking to ceremonies. Over time, we noticed that not having standups everyday during times when less work is done helped. As mentioned above, scattered documentation was also an issue. Luckily, instead of trying to maintain all of it, we removed documentation instead, evading the paperwork trap.

Team Practices

We followed a Scrum/Scrumban process, where the fundamental aspects are as follows: having collaboration with the customer, daily standups, completing deliverables with motivated members, creating self-organizing groups, reflecting on how our team can get more effective during the sprints, focusing on flow.

To reduce downtime, the PO worked on making Acceptance Criteria(AC) for the next sprint's features by discussing and negotiating with the customer while the development team made user stories fulfill the Definition of Done(DoD). While we did not have official Definition of Ready(DoR) requirements, the PO also made sure the aforementioned features could be pulled from the backlog. This process allowed all departments to begin work for each sprint without stalling the project. Frequent customer inputs throughout the sprint phase helped us develop the product as per the customer's latest needs. The priorities were then discussed with the customer which helped the development team pick the user stories. The drawback we felt from this project perspective is that the 15 minute time slots for the customer meeting was less, especially in the first sprint since we needed to discuss the framework of the project. The majority of the conversations were through emails which may have been less effective and slower than calls, but it also made us create "backup plans" before the customer responded.

Allowing the customer to see our progress during acceptance tests and receive feedback helped us plan future sprints, but we felt limited due to the few differing sprint lengths. Effective estimation of the user stories would have required more time to understand the complexity and time needed. Although we were able to complete the product and our estimates improved over time, they were still inaccurate in the end. The majority of estimations were also done by our more experienced members, again putting a larger workload on them.

In an actual cross functional agile setting, team members would have similar degrees of competence, thus the estimation for user stories would have been more holistic, along with more group input. To mitigate inaccurate estimations, we could have made standards for estimations as well. However, given such a small project and timeframe, we simply prioritized other things, and decided to simply notify the customer if we were behind schedule. However, the lack of customer contact in our last sprint caused us to push forward, heavily increasing our workload to deliver what was originally promised.

5.2 Impact of Practices on Software

With our early focus on agility and meta-requirements, many practices were taken into consideration when building the foundation.

Product backlog

In our development process the product backlogs were taken from the customer and well organized before the sprint starts. After selecting the features for the sprint, a priority was given to them in a sprint backlog and if needed the user stories were updated with new information from the customer. During the sprints the product owner took the responsibility of discussing and negotiating the product features with the client.

Team learning and pair programming

Role rotations sometimes required new team members to proceed with the previous development tasks. We soon realised that large training “lectures” were ineffective. We thus transitioned to small private meetings and pair programming. Varying degrees of knowledge thus encouraged pair programming between team members, especially for user stories which required frontend and backend code integration.

DOR and DOD

After the sprint planning, user stories are organized in the product backlog and marked as **Ready** to start working, and **Done** when implemented and reviewed. In addition, story sub-tasks were defined by the development team. Due to time constraints, the PO and Scrum master helped define user stories.

Lack of Coding Standards

Coding standards for page styling would have simplified the coding part significantly. This standard could have been visualized as the standard design page used in the design tool Figma, that was used to design this project. Since a difference in styling and consistency became a concern towards the end of sprint 3 this practice would have been a welcomed change. This standard should have been implemented and accepted by each member in the beginning of sprint 1 since changing choices needs more work later during the project. This change would also have made refactoring easier in the later sprints, furthermore it would have reduced the amount of technical debt in the project.

Collective Code Ownership

Separating our team into various departments resulted in a lack of collective code ownership in the beginning, since only those in development knew the code. Onboarding was difficult - hands-on experience needs time. The vastly different sprint lengths also contributed to differing levels of knowledge. Furthermore, those who onboarded development in the final sprint developed a somewhat irrational fear of the backend, leading to an imbalance between the frontend and the backend. This fear is not unfounded - modifying the backend meant failing tests in the CI/CD pipeline, introducing bugs for central features. Yet, avoiding the problem was detrimental to how user stories were assigned to each member later on. This resulted in uneven workloads on a select few senior members, thus leading to a dependency bottleneck during the final sprint.

This is not to say “senior developers” should have the same workload all the time, Given such a short timeframe, entrusting central features to more knowledgeable members is smart - there is less risk of failure and less time required for completion. Yet, the more specialized members are, the lower the truck factor[1], stalling the project - which was exactly what happened to us.

[1] https://en.wikipedia.org/wiki/Bus_factor

Neglecting Design at Final Sprint

Noticing the dependencies between design and development early on, we decided that the design would need to be one step ahead of development - the next sprint's designs needed to be completed so that development has no downtime. We thus focused more on design in the beginning, then transitioning to more development later on. In theory, this meant that we thought the final sprint would not require a design team. A flaw in our assumptions was that the designs were definitely done at the end of each retrospective, but changing requirements during the next sprint were not fully considered. This created outdated designs which development integrated, resulting in refactoring work.

Future Work

While most of the user stories were completed, we would spent future sprints doing:

- SID_42: Cancelling Reservations.
- Restricting Reservations to only allow Full Package users access it.
- Distributing activity locations properly in the map.
- UI polish.
- Bug fixing.

Project outcome

6.1 Architecture

The final product consists of a web based application, designed primarily for mobile devices. The application is meant for theme park visitors, with various functions that integrate with the park's services and activities. Users can read about activities, which include rides, cafeterias and arcade games, buy tickets to the theme park and reserve seats. Furthermore, to help visitors navigate the park, the application also includes a map on which all the activities are located. Users can also create personal accounts in the application in order to store purchased tickets and reserved seats. The design of the interface provides a fun user experience, with playful colors and rounded corners, emphasizing the spirit of the theme park. The navigation bar facilitates moving to different pages of the application.

The frontend application is made using the framework Vue.js, allowing for component-based development consisting of Javascript, CSS and HTML. Each web page is represented by a view, which in turn generally consists of several components. Additionally, the backend application, which is made using the Javascript framework Express.js, serves the frontend park with data using HTTP requests. These requests fetch information about the activities, along with handling seat reservations, ticket purchasing and user account functions. User functionalities that depend on the account information are secured and managed by JSON Web Tokens (JWT) and a pair of ECDSA based keys. Finally, the persistent data storage and retrieval is handled by the NoSQL database, MongoDB, which for development purposes is hosted as a deployable Docker container. The actual interfacing between the backend and the database is implemented using the Object Data Modeling (ODM) middleware - Mongoose.

6.2. User View

This section goes over different views of the web application as observed from the browser, with figures including both screenshots and brief explanations of each view's function.

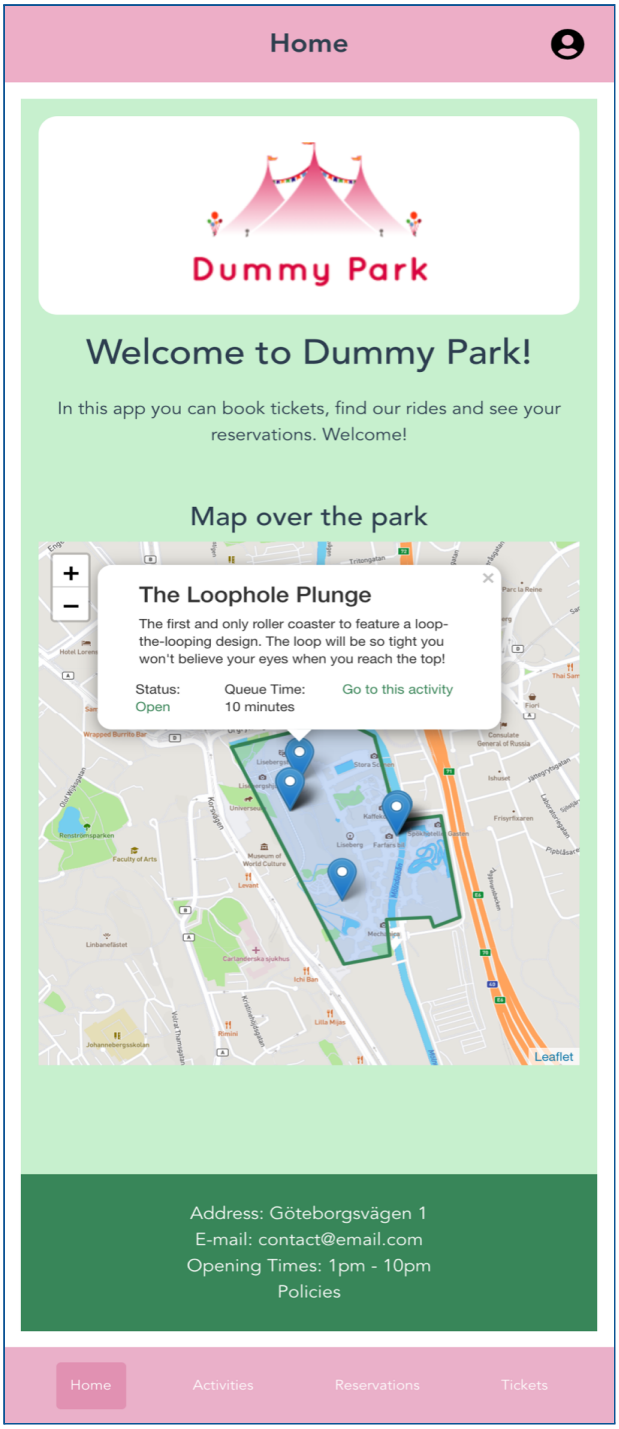
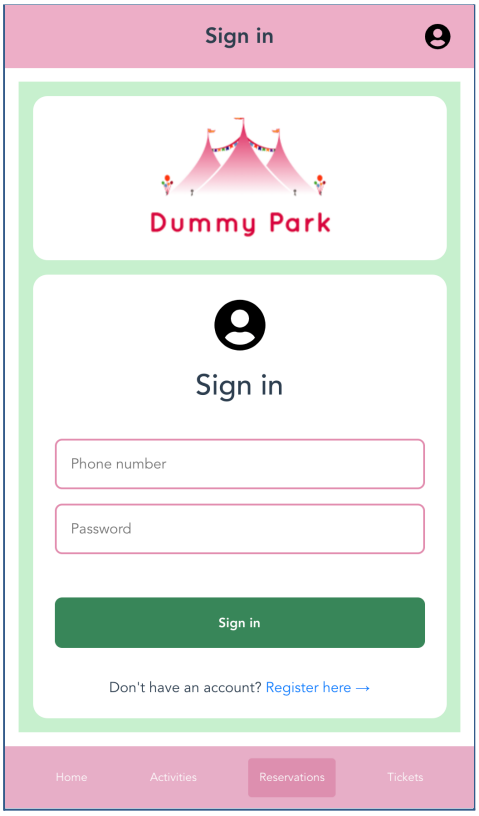


Figure 1 - Home page. *The first view the user sees when first visiting the app. This view includes the logo of the theme park, a map component containing all activities and a footer at the bottom with Policies.*

Figure 2 - Policies page. *This page contains the privacy policy of the park.*



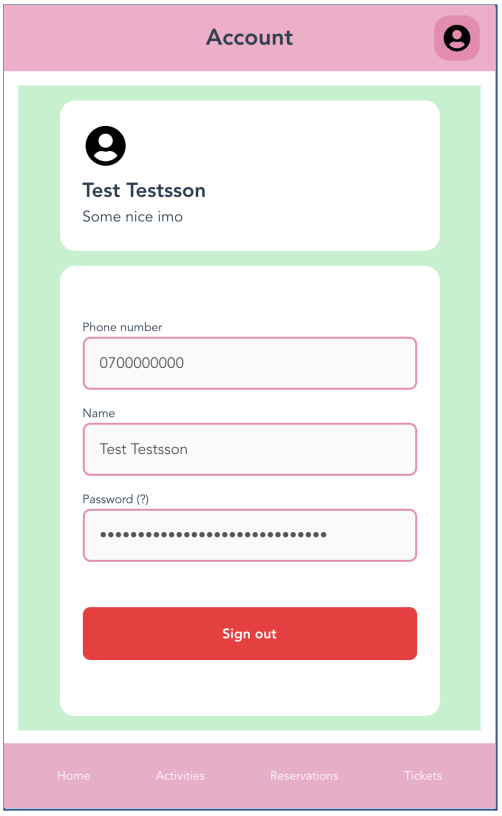
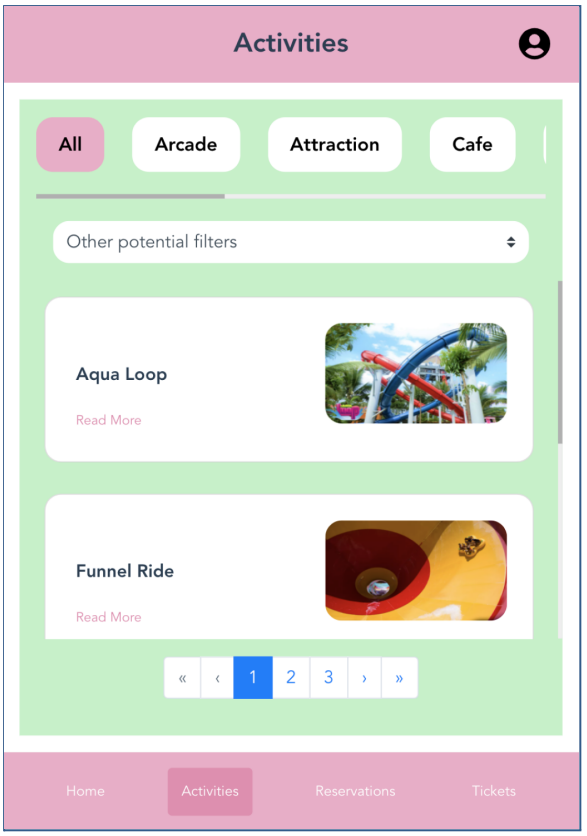


Figure 3 and 4 - Login page and Account page. *When trying to reach either the Account page, the Reservations page or the Tickets page, the user is prompted to login. On the Login page, the user can fill in their details or register a new account. Once completed, the user is redirected to the respective page they were trying to reach earlier. Figure 4 shows the Account page where the user can sign out.*



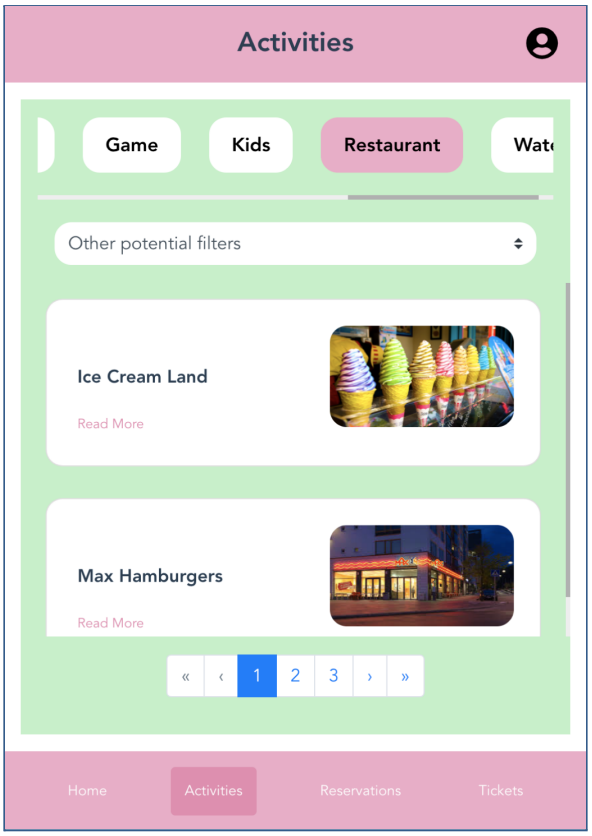


Figure 5 and 6 - Activities page. *This page displays all the activities the theme park offers. By default, all available activities are shown. At the top, there are filter buttons which toggles the page to only display specific types of activities, such as restaurants, which is seen in Figure 6. Below, there is a select field where the user can specify the sorting of the results.*

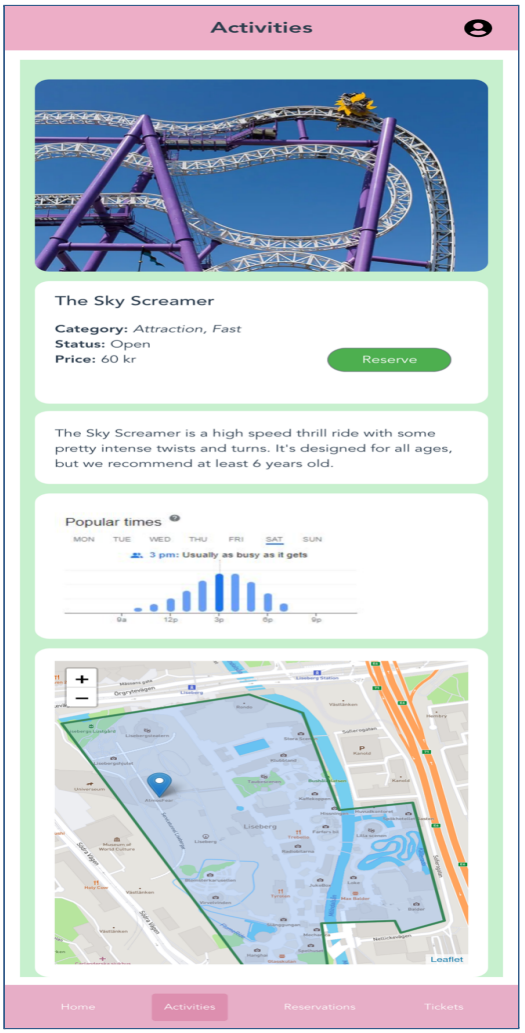


Figure 7 - Activity details page. *When a result on the Activities page is clicked on, the user will be directed to the details page of the activity. This page includes an image, information, a graph over the popular visiting times and its location on a map. The user can also visit the reservation page for the activity via the Reserve button.*

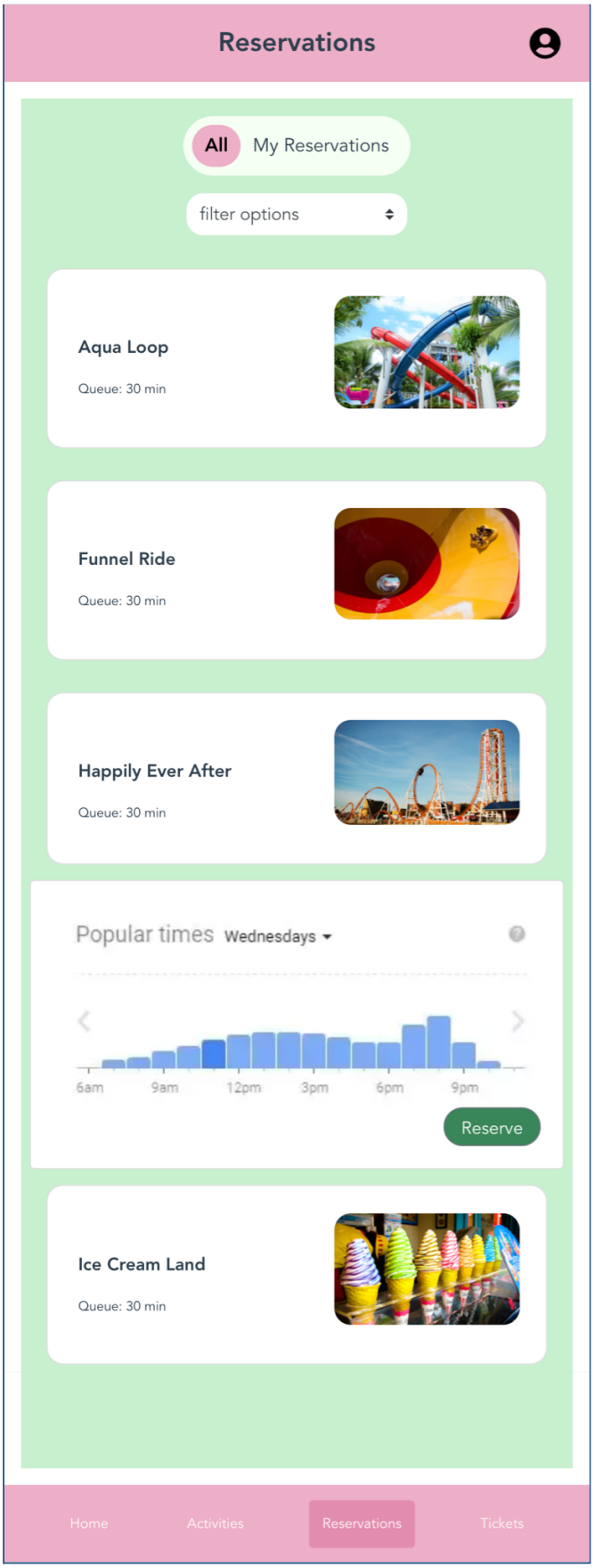


Figure 8 - Reservations page. *When the toggle button at the top is set to "All", this page displays all the activities that are available for reservation by the user. When a result is clicked on, the popular visiting times together with a Reserve button is displayed below it, as shown in this figure. To reserve seats, the user clicks on the Reserve button.*

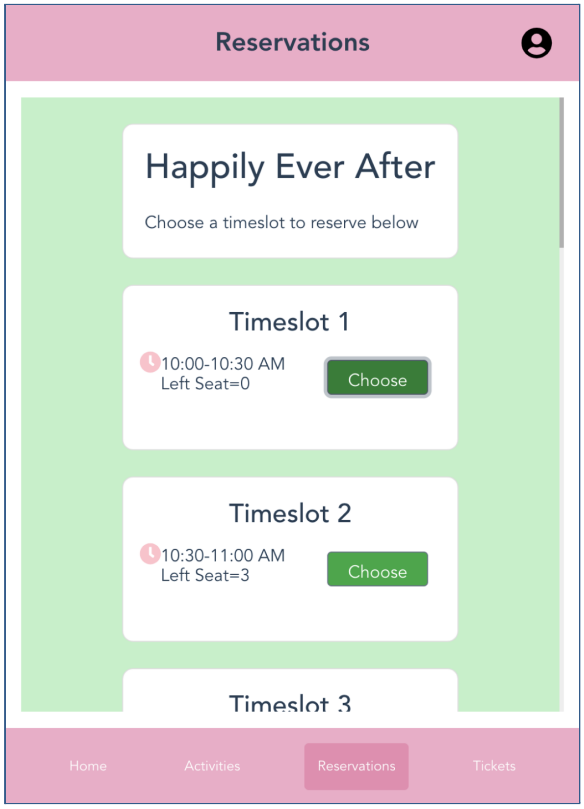


Figure 9 - Reservable time slots. *When an activity is selected on the Reservations page, this view grants the user with all the available time slots for that activity. The duration of the time slot, together with the remaining available seats is displayed on the respective time slot card. When the user clicks the Choose button, the number of available seats for that time slot decreases by one.*

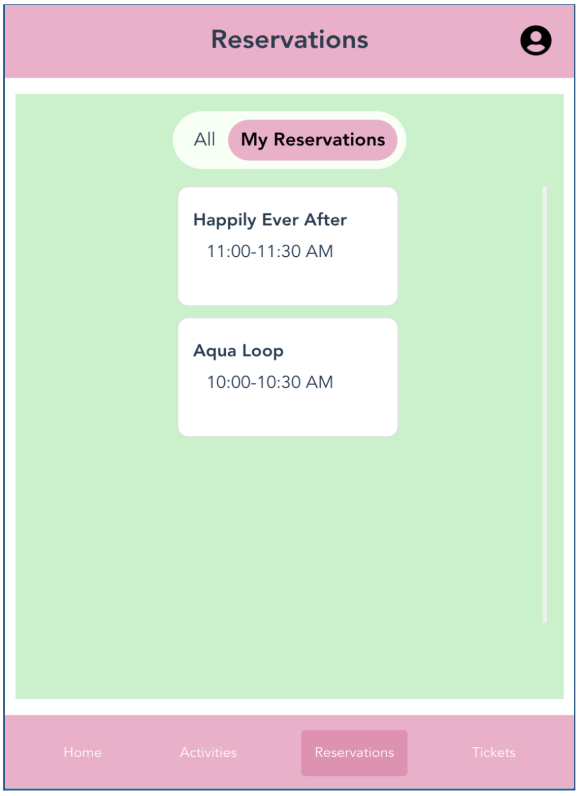
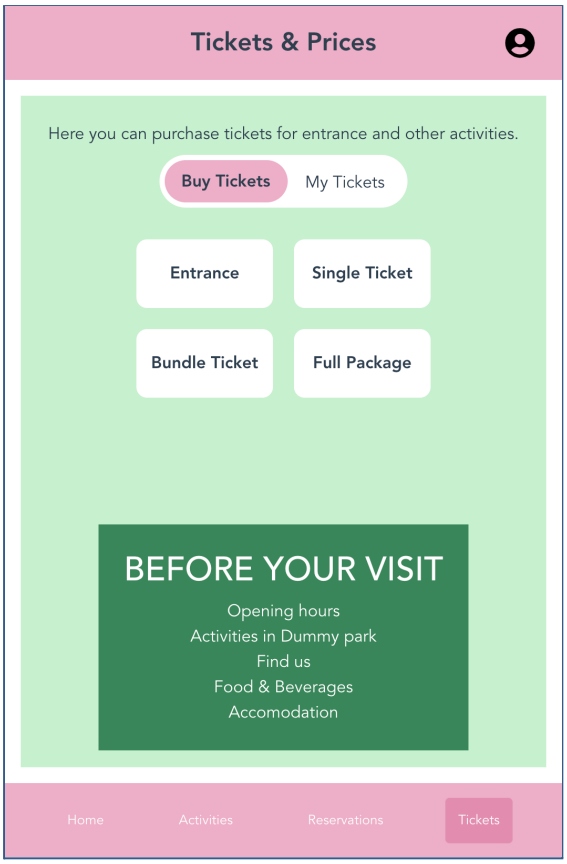


Figure 10 - My Reservations. *When the user has reserved a seat on an activity, the reservation is displayed on the Reservations page when the toggle button is set to "My Reservations".*



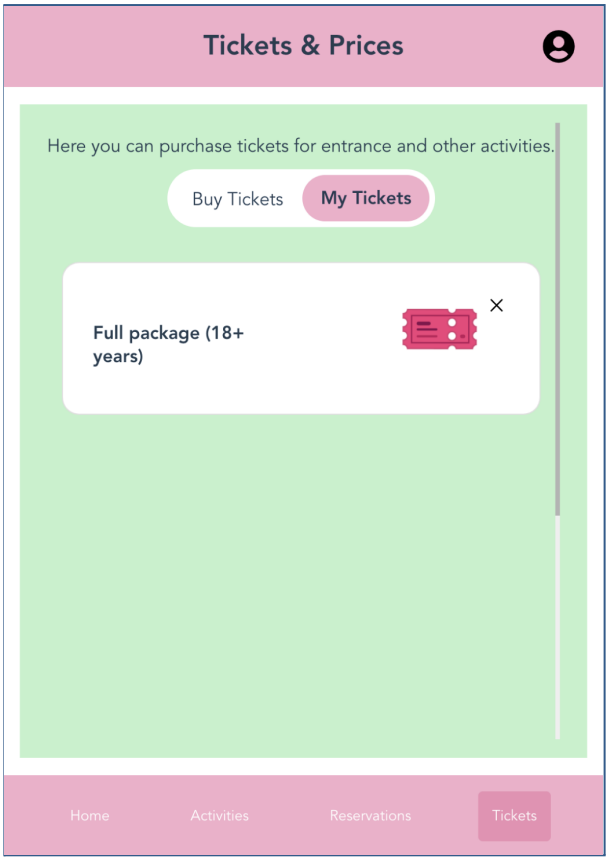


Figure 11 and 12 - Tickets page. *When the user navigates to the Tickets page, this view is presented. Using the toggle button, the user can choose to display the view of either "Buy Tickets" or "My Tickets". Buy Tickets show a footer together with buttons for the four different ticket types available for purchase, each redirecting to the checkout process. Under My Tickets, the user's purchased tickets are displayed.*

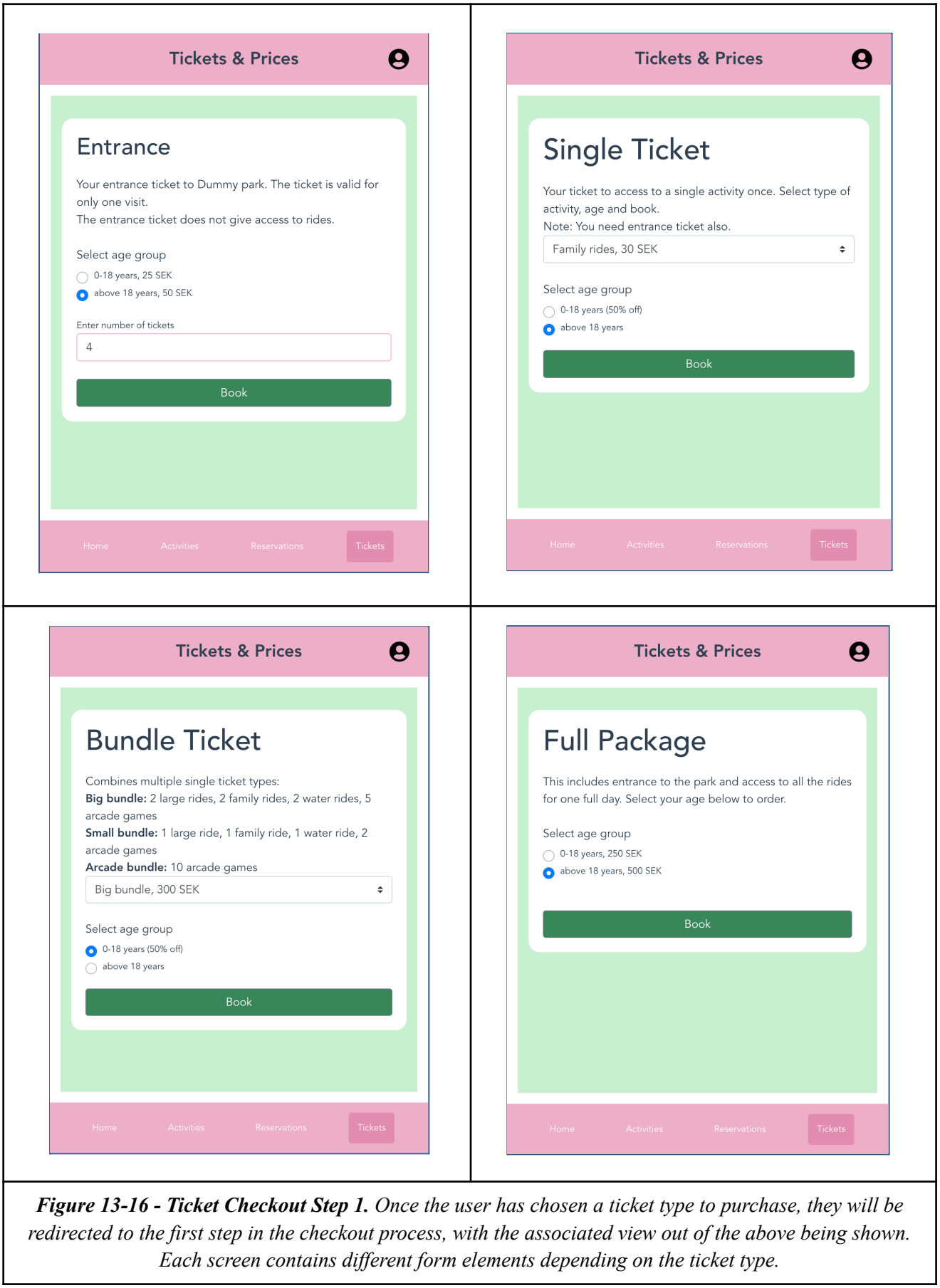


Figure 13-16 - Ticket Checkout Step 1. *Once the user has chosen a ticket type to purchase, they will be redirected to the first step in the checkout process, with the associated view out of the above being shown. Each screen contains different form elements depending on the ticket type.*

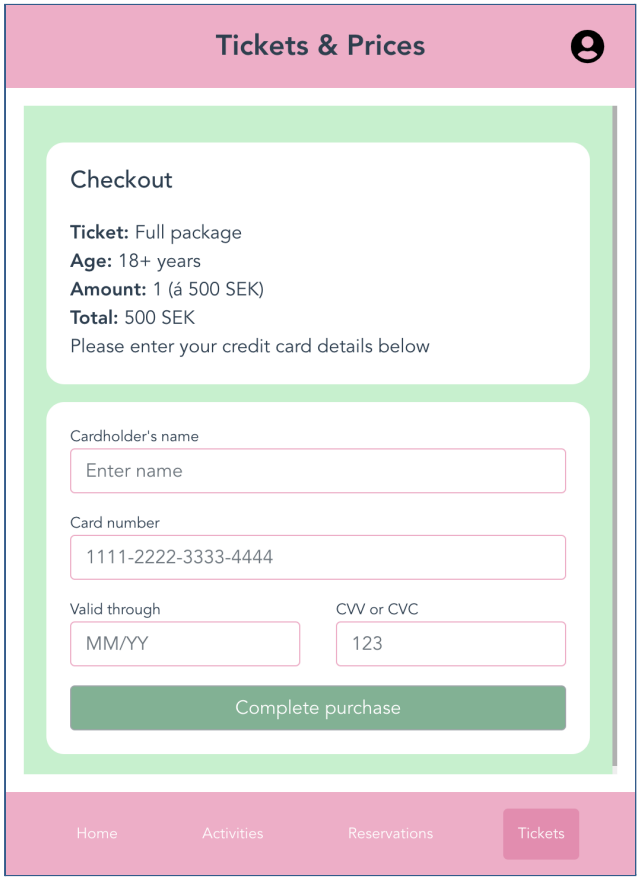


Figure 17 - Ticket Checkout Step 2. *After completing the first step in the checkout process, the user will be directed to the final step, in which credit card details are provided. Information about the purchase is shown at the top.*

Figure 18 - Purchase Confirmation page. *This page is shown after the user provided their payment information and the ticket purchase was completed. Details about the purchase are shown on this screen, as well as a button to bring the user back to the My Tickets page.*