PROJECT ANALYSIS

RUNNING EVENT SYSTEM

PRESENT TO

Assoc. Prof. Dr. Twittie Senivongse

BY

Ms.Kamolnadda	Dansuputra	5931001021
Ms.Natthawan	Siripokasupkul	5930188521
Ms.Chanissa	Trithipkaiwanpon	5931016421
Mr.Nutchanon	Ploypray	5930166721
Mr.Pisit	Wajanasara	5931042721
Mr.Pawin	Piemthai	5931037621

Systems Analysis and Design
2110332, semester 1/2018

Department of Computer Engineering,
Faculty of Engineering, Chulalongkorn University

Contents

List of Illustrations

List of Tables

1. Project name : Running Event System

2. Introduction

For the past several years, running events has become trendy in Thailand. Men and Women , young and old, famous and ordinary people are all interested in these events. Almost every month, there are two or three running events. This is a good phenomenon for thai people since it encourages people to exercise more and care more for their health.

However, since these running events have just become popular in Thailand, they still don't have an officially centralize system for event organizers and runners to communicate. Consequently, there are many problems and inconvenience such as events are announced separately so it is hard for runners to look for the events that is most proper to them ,event managers find it hard to promote and manage their event , runners don't receive enough information of the running event and difficult for runners to find and pay for the reservation etc.

Our team concern with all the problems stating above so we decide to create a centralized platform for runners and event organizers to communicate. We'll also include useful functions to enhance user's convenience and hope that this project will encourage more people to go to running events.

3. Objective of the analysis document

3.1 To understand the whole picture of to-be system.

3.2 To understand the relationship and collaboration between actors and our system.

3.3 To design the sequences and flows of each use case such as the process of reservation, the process of payment etc..

3.4 To simplify the complexity of use case and revise the consistency of associations of components in the system.

4. Details of requirements

1. Functional requirements

1.1. Registration and account management system

- 1.1.1. The system shall allow the runner to register a runner account on the system.
- 1.1.2. The system shall allow the event manager to register a manager account on the system.
- 1.1.3. When the user has already logged in, the system shall allow the user to edit his or her own personal information.
- 1.1.4. When the event manager has logged in, the system shall allow the event manager to edit his or her own information.
- 1.1.5. When user has already logged in, the system shall allow the user to view a list of their past participating events and upcoming participating events.
- 1.1.6. The system shall allow the system administrator to view the list of all runner and event manager account.

1.1.7. The system shall allow the system administrator to remove any runner or event manager account.

1.2. Logging system
- 1.2.1. The system shall keep log of all event's participation.
- 1.2.2. The system shall keep log of all payment transaction.

1.3. Event management system
- 1.3.1. The system shall allow event manager to create a new running system.
- 1.3.2. The system shall allow event manager to edit his or her own running event.
- 1.3.3. The system shall allow event manager to view the payment transaction that pertaining to his or her own running event.
- 1.3.4. The system shall allow event manager to view the list of participants of his or her own running event.
- 1.3.5. The system shall allow event manager to edit the list of participants of his or her own running event.
- 1.3.6. The system shall allow the system administrator to view the list of all events.
- 1.3.7. The system shall allow the system administrator to remove any running event.

1.4. Payment system
- 1.4.1. The system shall allow the runner to pay the running cost using mobile banking.
- 1.4.2. The system shall allow the runner to pay the running cost using credit or debit card.
- 1.4.3. The system shall allow the runner to print the receipt of the transaction after the payment is completed.

1.5. Reservation system
- 1.5.1. The system shall allow the runner to reserve a spot on the running event by specifying the event along with date and time of their participation.
- 1.5.2. The system should notify the runner by email seven and three days before the event starts.

1.6. Timeline and calendar
- 1.6.1. The system shall be able to display the calendar containing the past, ongoing and upcoming running event.
- 1.6.2. The system shall be able to display the information of all running events, including name, location, distance, rules, cost and responsible people
- 1.6.3. The system shall be able to display the running events based on the location of the runner, distance, price range, running type , time , charity etc.

1.7. Running event suggestion system
- 1.7.1. The system shall be able to recommend a personalized list of running event to the runner.
- 1.7.2. The system shall allow runner to adjust the criteria used for event suggestion.

1.7.3. The system shall allow system administrator to adjust the criteria used for the overall event suggestion system.

1.8. Users communication system

1.8.1. The system shall allow user to send message to the event manager.

1.8.2. The system shall allow event manager to reply to the runner's message.

1.8.3. The system should include the discussion board that let the participating runner contact fellow participants or event managers.

2. Non-functional requirements

2.1. Operational requirements

2.1.1. The system shall operate in every mainstream browser.

2.1.2. The system shall make automatic backup of its database daily.

2.2. Performance requirements

2.2.1. The system shall have the response time less than 2 seconds for every interaction between system and user.

2.3. Security requirements

2.3.1. The system shall allow only event managers to edit running event system.

2.4. Cultural and political requirements

2.4.1. The system shall be available in English.

2.4.2. The system may be available in Thai.

2.4.3. The system shall display the running events only in Thailand.

5. Overview of to-be system context

From the problems stated in the as-is system, we have design a new system that will solve those problems. The new system will start with the event managers can register for a manager account in our website and then they can create their event. After creating an event, the event managers can view and modify their event. The users can use the system by registering for user account which will record their information used when they apply for the event, event log and, upcoming event. The users can search an event based on their preference, location or, time period. Once the users want to apply for the event, they can contact the organizer directly, reserve their spot in the event or, apply for the event in our website. For the payment system, users can pay the entry fee by mobile banking transfer or credit, debit cards. After payment is completed, the users can print the receipt from website. The users and organizers can both view their transactions log in the website. When the registration is complete, the users will be notify seven and three days before the event date. The diagram of the new system workflow is shown in figure 2.

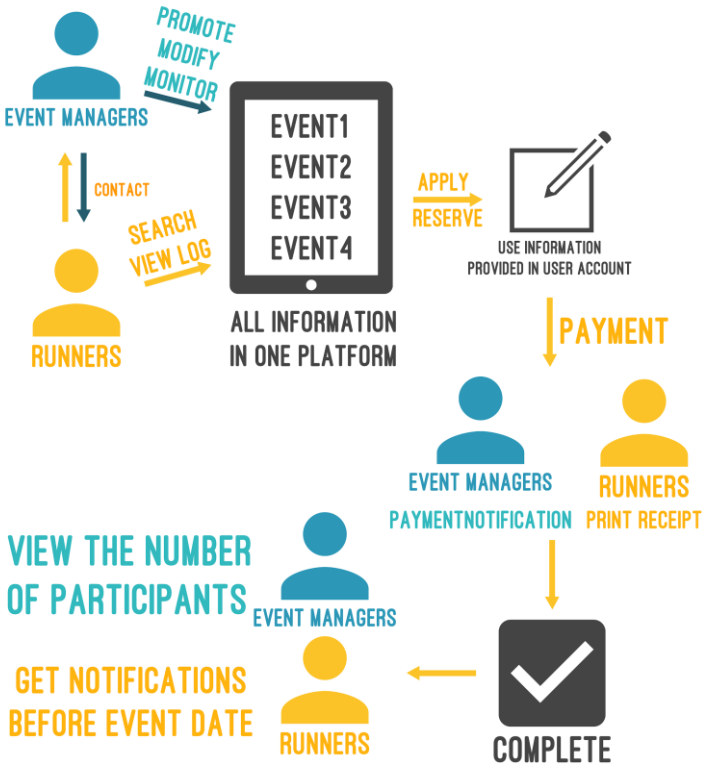


Figure 1. To-be system workflow

6. Terms and definition

Term	Definition
Running event	A casual running event for amateur runners and normal people.
Event manager	A group of person who organizes the running event.
Running type	A type of running; This system will focus on only two: road running and cross country running.
Road running	The sport of running on a measured course over an established road.
Cross country running	a sport in which teams and individuals run a race on open-air courses over natural terrain.
Running distance	A total running distance over the entire course: usually ranging from short run (2.5 km) to Marathon (42 km).
User	Typical user of the system; This includes both interested runners and event managers.
Community	Refer to runners community
Reservation cost	A cost that runners must pay in advance to the event managers in order to participate in the running event.

Table 1. Terms and definition

7. List of stakeholders and their responsibilities

7.1 Event Manager - The owner of the event who responsible for create an even and manage an event. Function for the event manager is create an event, edit an event, view list of participants, receive money from the system and participate in communication system.

7.2 Runners - Users who wish to attend a running event. Function for the runners is to look for a running event, reserve a running event, pay for a running event and participate in communication system.

7.3 Payment Gateway - The external system that will verify the debit/credit card of runners and handle the payment transaction which is occured in the system. Function for the payment gateway is verify and make payment and provide the payment status.

7.4 Email distribution system - The external system that will distribute desired email to the user in the system. Function for the email distribution system is provide a method for the system to send and receive email.

8. System analysis approaches

Business Process Improvement (BPI) : The as is system is not centralized and some of the processes can be optimized for more efficiency. So, we introduce a new system to centralize and improve the process of the as-is system

9. Analysis strategies

9.1 Problem analysis - We ask users who attend in running event more than 10 time what problems they are facing. Usually they have problem with the announcement.Moreover, the reservation system is difficult and sometime they don't get enough information about the event. Also, we ask event manager about their problems. They say that it is hard for them to promote their event and organize their event.

9.2 Root cause analysis - We recognize that most of the problems occuring within the runner's community is caused by the poor management of the running event. This has been amplified by the lack of the integration system for the event manager and the central community for the runners. Hence, we seek to eliminate those cause by implementing the running event management system.

9.3 Outcome analysis - The outcome that the runners really want is not only the reservation system, but also a community for runners to communicate and compete which will further strengthen the bond between runners and the community and create a sense of achievement.

10. Requirements gathering techniques

Interview - We gather the information about how event managers manage the running events by direct interview. Also, we interview runners about the feedback and suggestion that we can use to improve our system or be additional requirements in our system

Interview of Running Event	
Runner Name : Supavich Tharathepakul	
Date of Interview : 9.26 AM, 5 September 2018	**Total number of running event :** 52
Feedback **Announcement System :** Event managers usually promote the running event on Facebook page every two weeks and also update the information about event at the same time. Therefore, runners get enough information and know the details of process on the day of event. **Registration and Reservation System :** Event manager creates a google doc to gather about the runner's information including the reservation's information. The problem is runners have to fill out the information every time they apply for the event. **Notification System :** Most runners often apply more than one running events and they sometimes miss the event. Normally, running events will open for the application 2-3 months before the running event was held. As a result, runners are easily to forget the events and miss them.	
Suggestion - The event manager should build the software platform such as web browser for reservation and registration so runners can track back the historical transaction and view the on-going events that runners apply for them.Also , they don't have to fill the information every time they apply in the event.	

Table 2. Interview from Supavich Tharathepakul

Interview of Running Event	
Runner Name : Tanat Sukrit	
Date of Interview : 11.09 AM, 7 September 2018	**Total number of running event :** 46
Feedback **Announcement System :** Runners don't have time to look for all the events and usually miss the interesting events. **Management System :** Event manager can't control the management of the event because of the large number of runners is more than the number that event manager expected. **Registration System :** Event manager has a web browser for only registration for his system. So if runners have already gone to his running events before, their information will be save on the web browser and they don't have to fill for it again.	

Suggestion - It would be better if it were a central system for advertising all of the events and the system should be able to provide the essential information to the runners so that they get enough information for the events.

Table 3. Interview from Tanat Sukrit

Interview of Running Event	
Event Manager Name : Yuta Matsuoka	
Date of Interview : 8.07 AM , 10 September 2018	**Total number of managing running event : 89**
Feedback **Announcement System :** It is very difficult for our company to advertise and announce an event. We have to go to many separated platforms and it is really a waste of time and cost. Many times we announce on a wrong platform, we don't get any feedback from that platform. Also, announcing an updated information on separated platform is very hard to make all the runners acknowledge and usually the runners who missed the announcement complain. **Management System :** It is hard for us to manage some part of the runners application system. Existing system does not integrate payment and reservation within the same place.It makes the quality of the events may become lower than they could be.	
Suggestion - It would be very much better to have a platform or a community for running lovers so we could promote our events in there. - We want a system that integrate payment and reservation so we don't have to do it by hand anymore and it would improve the quality of our events.	

Table 4. Interview from Yuta Matsuoka

11. Use Case Diagram

Overview of the running event management system which contains all functional requirement in the pattern of use case diagram showing the dependency for each use case with other use case and actors.

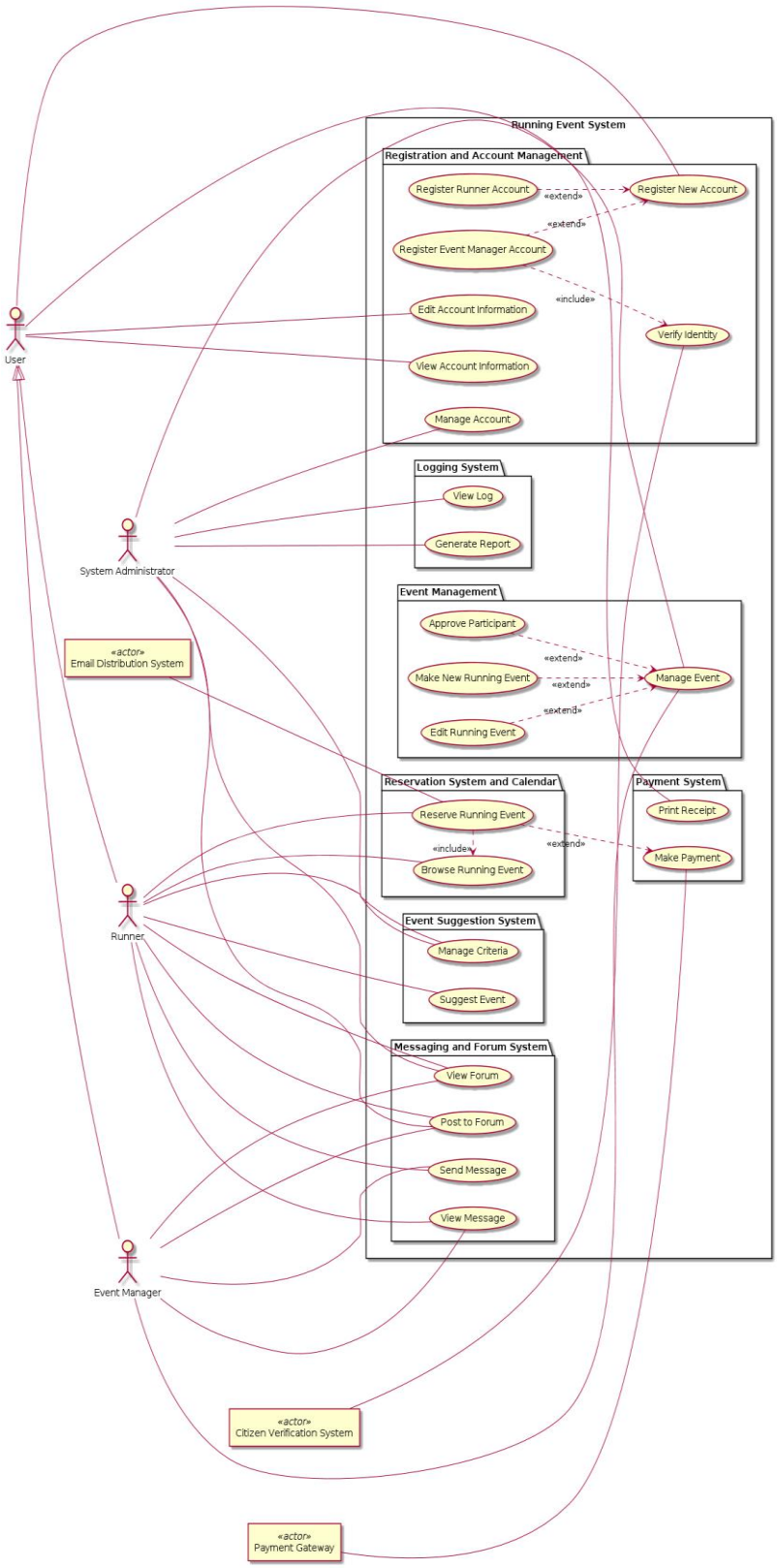


Figure 2. Use case diagram

12. Use Case Descriptions

Developers created use-case descriptions to documents the different aspects of each use case including flows of the use case and relationships. The following diagram are use case descriptions of Reserve Running Event Manage Event Register New Account and theirs extension or inclusion use case respectively.

Use Case Name: Reserve Running Event	ID: UC-1	Important Level: High
Primary Actor: Runner	Use Case Type: Detail, Essential	
Stakeholders and Interests: 1. Runner - wants to reserve running event 2. Event manager - wants to monitor event reservations		
Brief Description: This use case support runners to reserve desired events on the website.		
Precondition: Runner logged in to the system.		
Trigger: Runner starts select for running event on the website. Type: External		
Relationships: Association: Runner, Email Distribution System Include: Browse Running Event Extend: Make Payment Generalization:		
Normal Flow of Events: 1. Runner executes Browse Running Event use case. 2. Runner selects desired running event from browsing result. 3. Runner requests for an event reservation. 4. The system ask runner required information for event reservation. 5. The system validate runner's provided information. 5.1. If the event need a payment, perform subflow S-1: Event Payment 6. The system notify runner about reservation status by redirect runner to his/her profile page.		
SubFlows: S-1: Event Payment 1. Runner select credit card or debit card as a payment method. 2. Runner provide information required for payment method. 3. The system executes Make Payment use case for payment and payment verification. 4. The system notify runner about payment status. 5. The system redirect runner back to reservation page.		
Alternate/Exceptional Flows: 3a. Runner decided not to reserved the event and quit Reserve Running Event. 5a. If the validation fail, the system repeat step 4. S-1, 3a. If the payment fail, the system ask user to repeat subflow S-1L Event Payment or quit Reserve Running Event.		
Postcondition: Runner is added to approval request list for the event.		

Figure 3. Use Case Description of Reserve Running Event

Use Case Name: Manage Event	ID: UC-2	Important Level: High
Primary Actor: Event Manager, System Administrator	Use Case Type: Detail, Essential	
Stakeholders and Interests: 1. Event Manager 2. System Administrator 3. Runner		
Brief Description: This use case support primary actor to manage the running event in the system.		
Precondition: The primary actor logged in to the system.		
Trigger: Primary actor start to browse for various events displayed on the webpage. Type: External		
Relationships: Association: Event Manager, System Administrator Include: Extend: Approve Participant, Make New Running Event, Edit Running Event Generalization:		
Normal Flow of Events: 1. The primary actor browses to the event management page. 2. The system ask what is the desired operation of the primary actor. 2.1. If the primary actor want to approve a participant,executes Approve Participant use case 2.2. If the primary actor want to make a new event, executes Make New Running Event use case 2.3. If the primary actor want to edit a running event, executes Edit Running Event use case 3. The system display save changed event information to the primary actor.		
Alternate/Exceptional Flows: 2a. The primary actor decided not to make any change to the event and quit Manage Event.		
Postcondition: A log is created for each operation done for each event.		

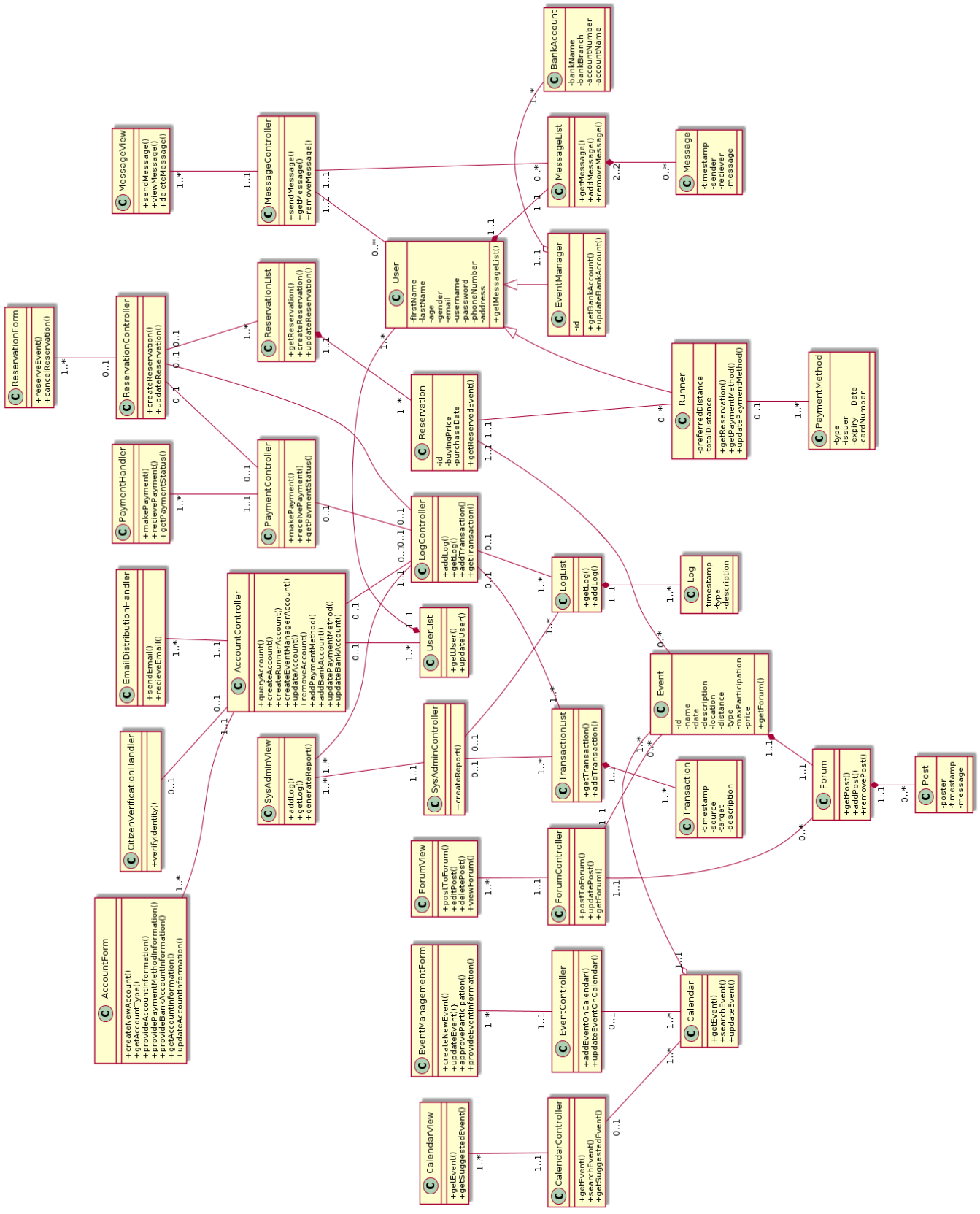
Figure 4. Use Case Description of Manage Event

Use Case Name: Register New Account	ID: UC-3	Important Level: High
Primary Actor: User	Use Case Type: Detail, Essential	
Stakeholders and Interests: 1. Runner 2. Event Manager 3. System Administrator		
Brief Description: This use case support user to register an account into the system.		
Precondition: The user is not registered to the website before.		
Trigger: The user want to use the system and decided to register a new account. Type: External		
Relationships: Association: User Include: Extend: Register Runner Account, Register Event Manager Generalization:		
Normal Flow of Events: 1. The user browse to the the register page. 2. The user provide desired username and password to the form. 3. The system ask the user whether the user is event manager or runner. 3.1. If the user is event manager, executes Register Event Manager Account use case 3.2. If the user is runner, executes Register Runner Account use case 4. The system redirect user to the login page.		
Alternate/Exceptional Flows: 1a. The user decided not to register to the system, quit Register New Account. 2a. If username is duplicated in the system, repeat step 2.		
Postcondition: An account is created on the system.		

Figure 5. Use Case Description of New Account

13. Class Diagram

Class Diagram describes how classes associated and display attributes and methods of each class. It also includes the cardinality and the relationship such as aggregation and generalization



14. CRC Cards

CRC Card is used to showing the detail of each class in the class diagram. CRC Card contains class responsibility, Collaborators, Attributes and Relationship between other class. 3 CRC Classes are shown below.

Front:

Class Name: Reservation List	**ID:** 1	**Type:** Concrete, Domain
Description: A list of reservations which create by runners		**Associated Use Cases:** 1
Responsibilities		**Collaborators**
Create reservation		Reservation
Update reservation		Reservation
Get reservation		Reservation

Back:

Attributes: -
Relationships: **Generalization (a-kind-of): -** **Aggregation (has-parts):** Reservation **Other Associations:** Reservation Controller

Figure 6. CRC card of Reservation List

Front:

Class Name: Event	**ID:** 2	**Type:** Concrete, Domain
Description: A running event created by event manager		**Associated Use Cases:** 2
Responsibilities		**Collaborators**
Get forum		Forum

Back:

Attributes: - Id - Name - Date - Description - Location - Distance - Type - Max participation - Price

Relationships: **Generalization (a-kind-of):** - **Aggregation (has-parts):** Reservation, Forum, Calendar **Other Associations:** Forum Controller

Figure 7. CRC card of Event

Front:

Class Name: Transaction	**ID:** 3	**Type:** Abstract, Domain
Description: A form of confirmation from payment		**Associated Use Cases:** 1, 1-5
Responsibilities -		**Collaborators** -

Back:

Attributes: - Timestamp - Source - Target - Description
Relationships: **Generalization (a-kind-of):** - **Aggregation (has-parts):** Transaction list **Other Associations:** -

Figure 8. CRC card of Transaction

15. Activity Diagrams

Activity diagram for Reserve Running Event use case

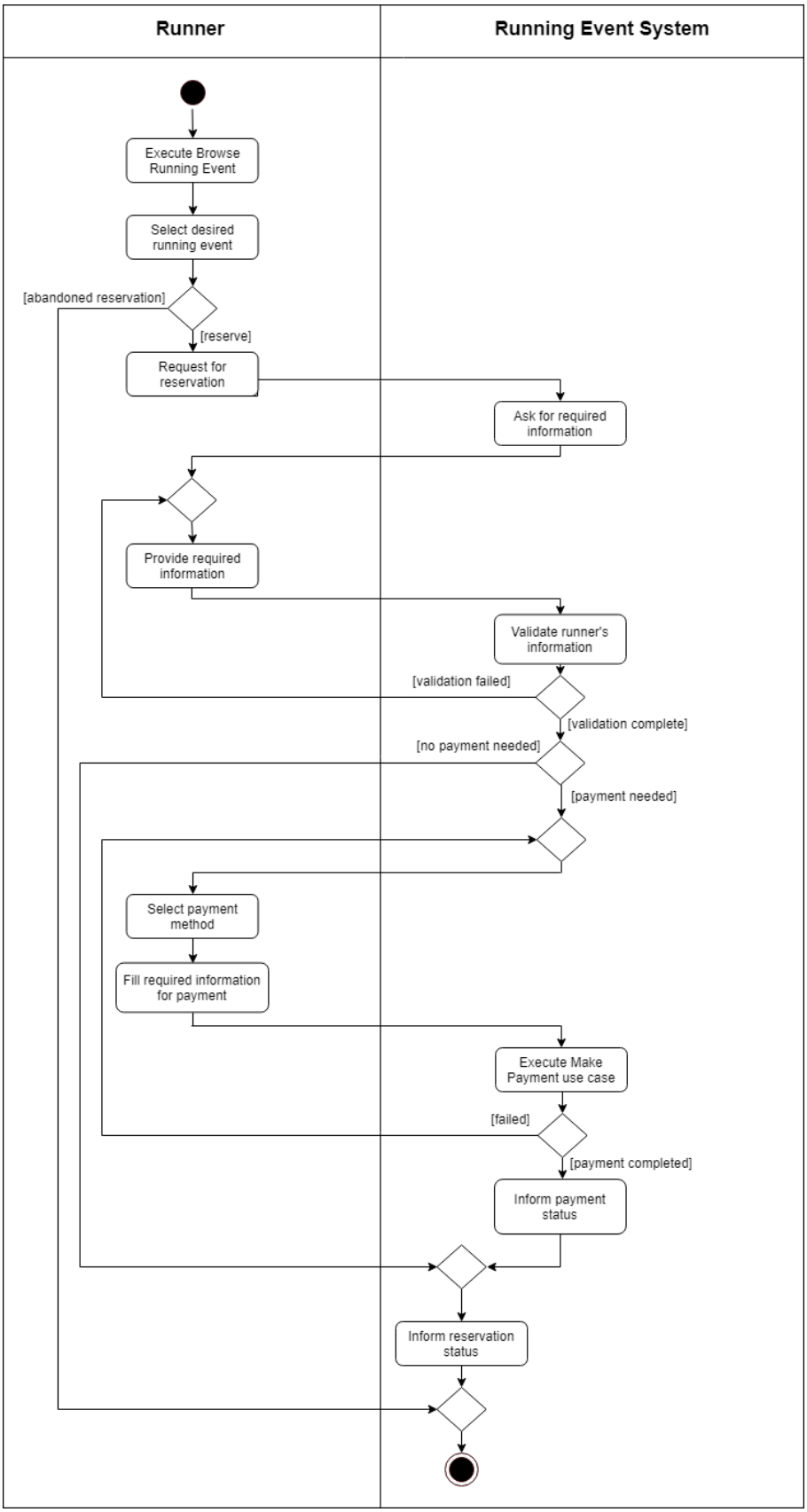


Figure 9. Activity diagram for Reserve Running Event use case

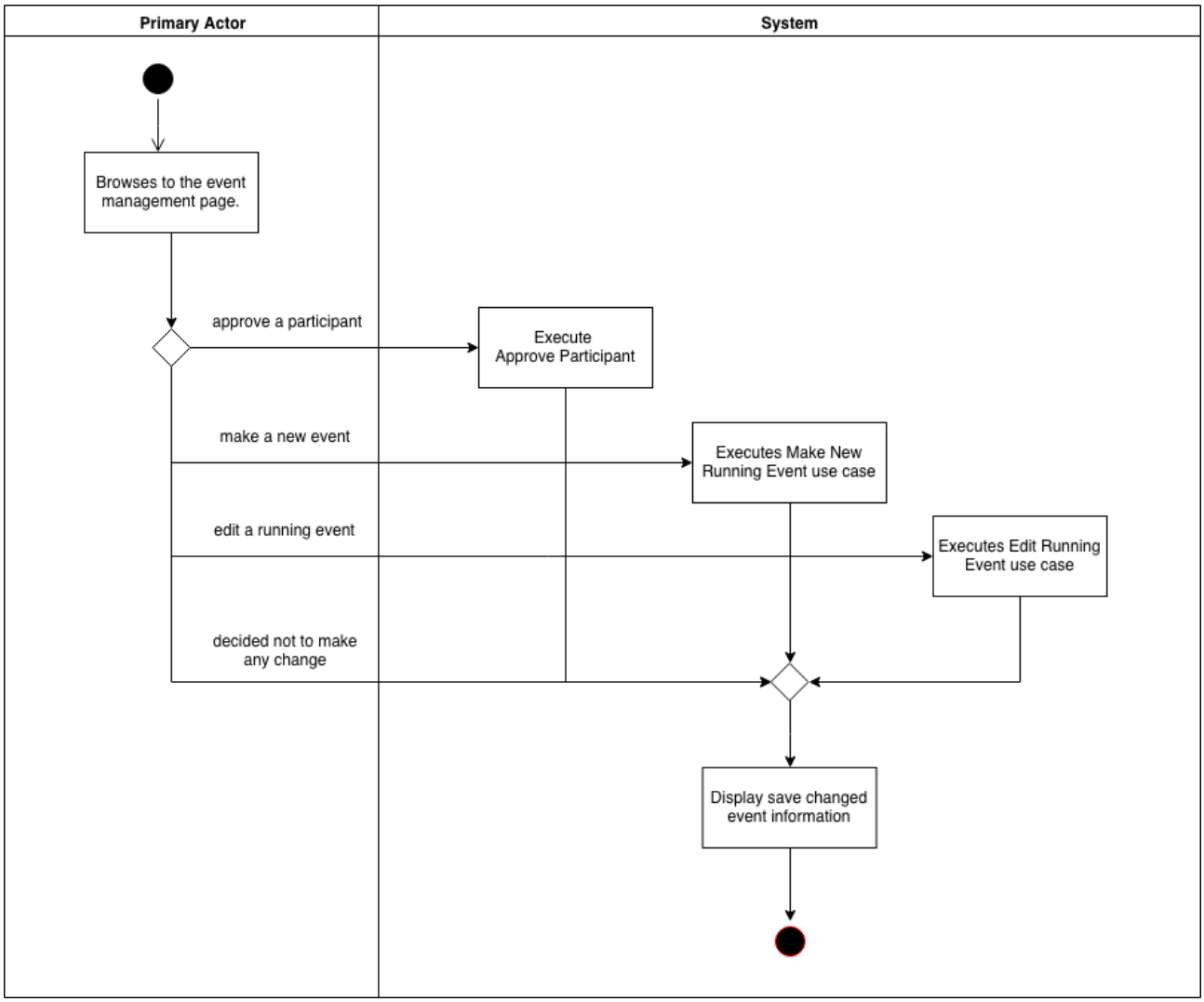


Figure 10. Activity diagram for Manage Event use case

16. Sequence Diagrams

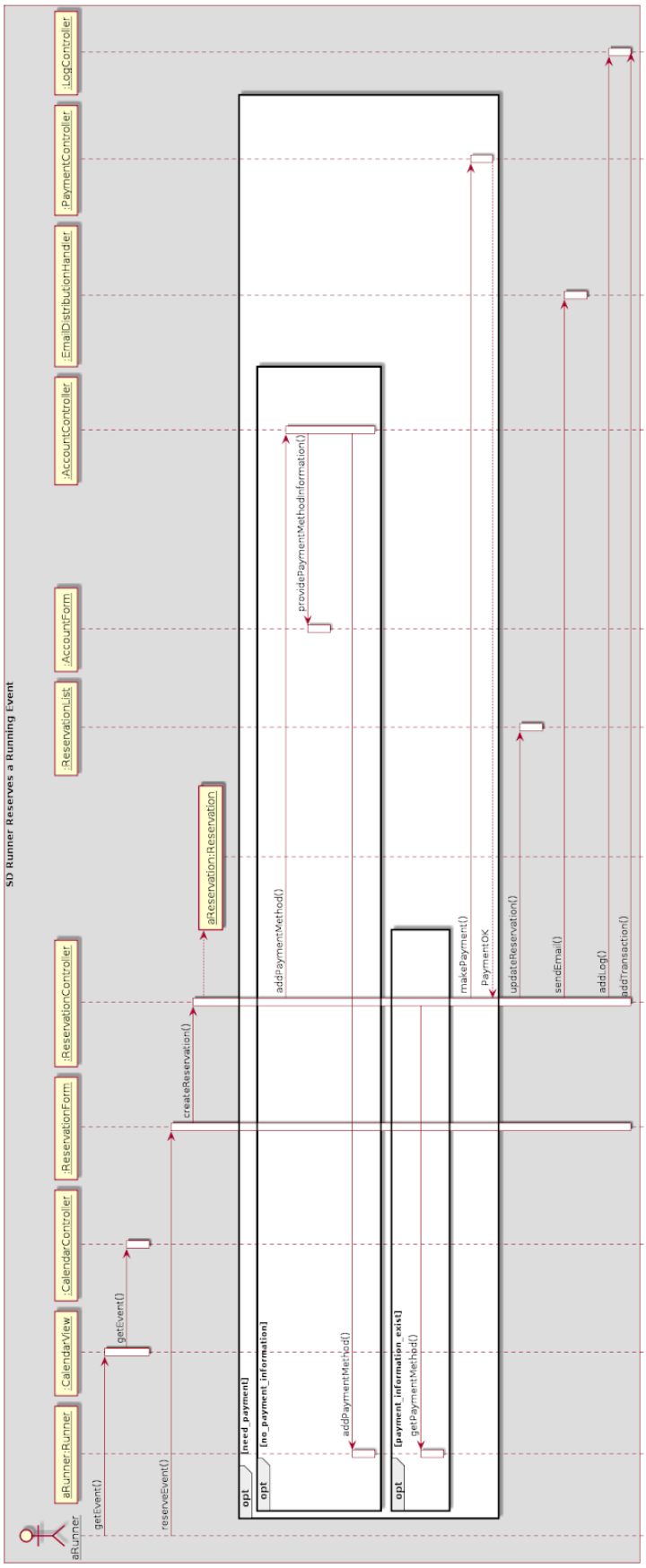


Figure 11. Runner Reserves a Running Event Sequence Diagram

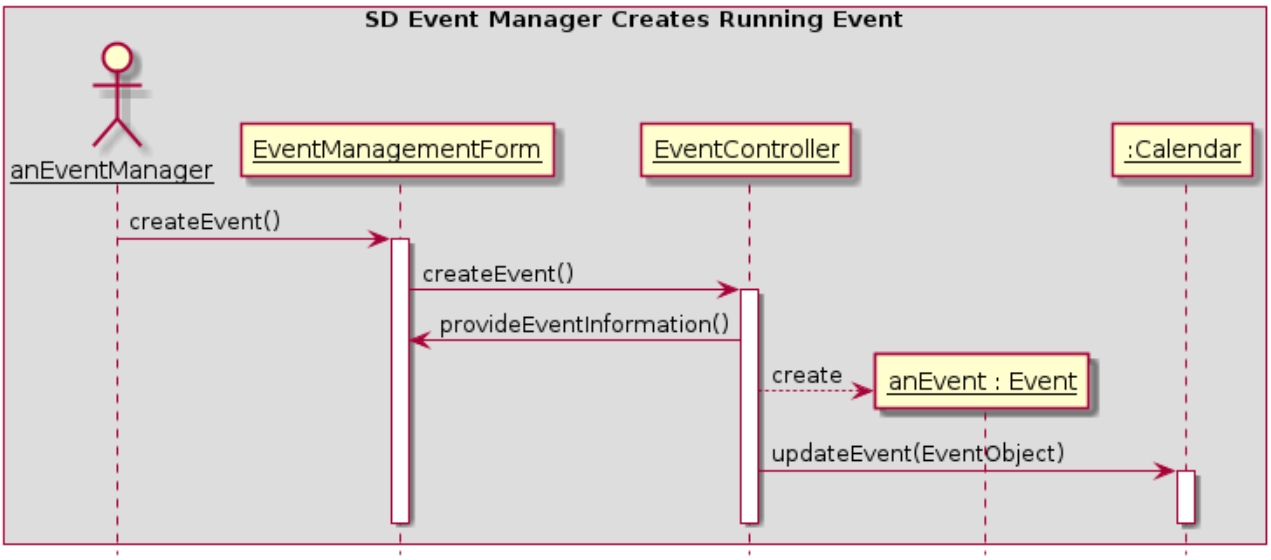
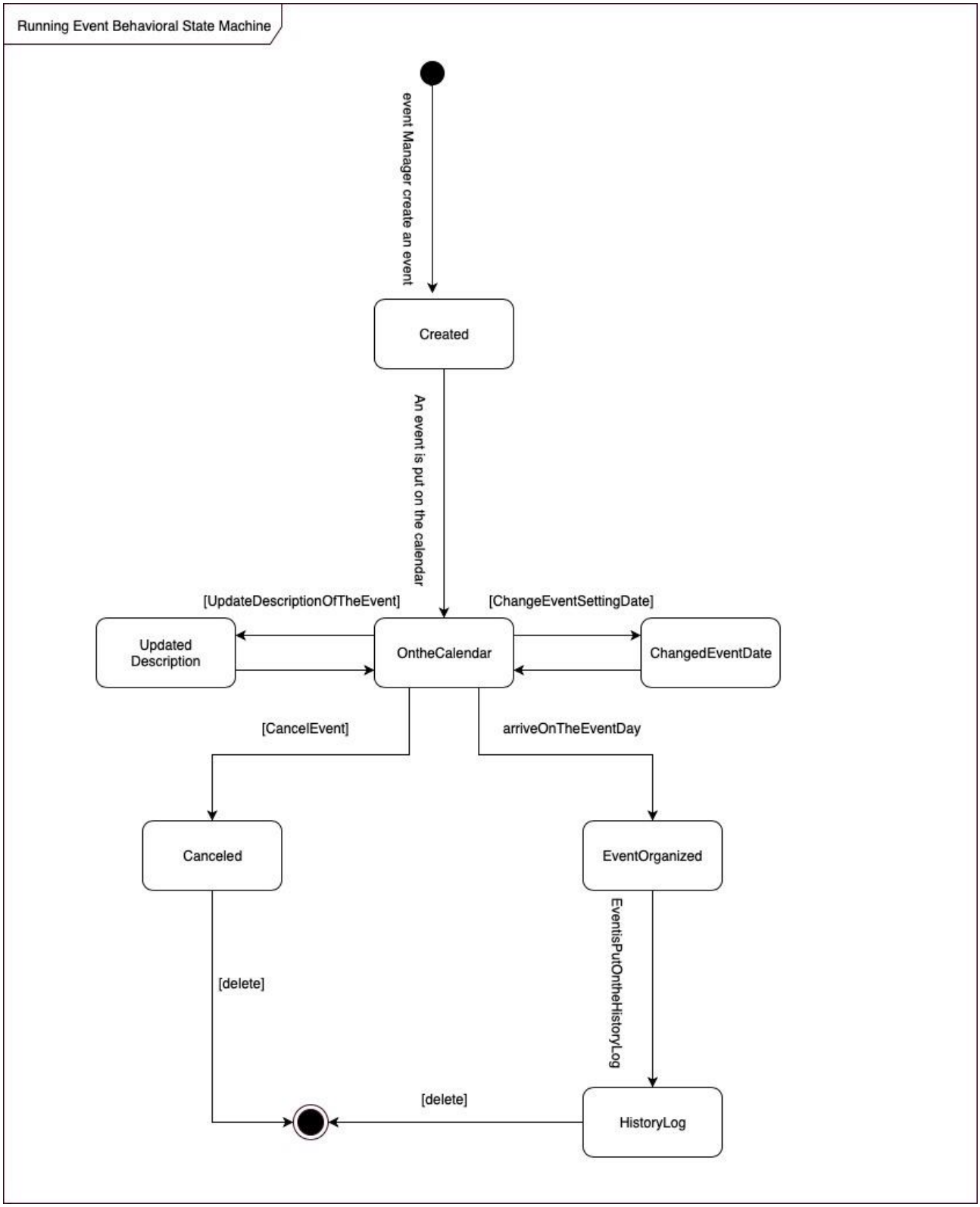


Figure 12. Event Manager Creates Running Event Sequence Diagram

17. Behavioral State Machine



18. Crude Matrix

	Runner Actor	Event Manager Actor	User Actor	System Administrator Actor	Citizen Verification System Actor	Payment Gateway Actor	AccountForm	Calendar View	ReservationForm	SysAdminView	EventManagementForm	MessageView	ForumView	PaymentHandler	CitizenVerificationHandler	EmailDistributionHandler	AccountController
Runner Actor							E	E	E			E	E				
Event Manager Actor							E	E			E	E	E				
User Actor																	
System Administrator Actor							E	E		E	E	E	E				
Citizen VerificationSystem Actor																	
Payment Gateway Actor																	
AccountForm																	E
CalendarView																	
ReservationForm																	
SysAdminView																	
EventManagementForm																	
MessageView																	
ForumView																	
PaymentHandler																	
CitizenVerificationHandler																	
EmailDistributionHandler																	
AccountController							E								E		E
ReservationController																E	
EventController											E						
PaymentController														E			
SysAdminController																	
CalendarController																	
LogController																	
MessageController																	
ForumController																	
UserList																	
User																	
Runner																	
EventManager																	
Calendar																	
Event																	
ReservationList																	
Reservation																	
LogList																	
Log																	
TransactionList																	
Transaction																	
PaymentMethod																	
BankAccount																	
MessageList																	
Message																	
Forum																	
Post																	

	ReservationController	EventController	PaymentController	SysAdminController	CalendarController	LogController	MessageController	ForumController	UserList	User	Runner	EventManager	Calendar	Event
Runner Actor														
Event Manager Actor														
User Actor														
System Administrator Actor														
Citizen VerificationSystem Actor														
Payment Gateway Actor														
AccountForm														
CalendarView					E									
ReservationForm	E													
SysAdminView				E		E								
EventManagementForm		E												
MessageView							E							
ForumView								E						
PaymentHandler														
CitizenVerificationHandler														
EmailDistributionHandler														
AccountController						E			E		C,R,U,D	C,R,U,D		
ReservationController			E			E					R,E			
EventController						E						R	E	C,R,U,D
PaymentController														
SysAdminController						E								
CalendarController													E	R
LogController														
MessageController										R, E				
ForumController														R,E
UserList									U	R				
User														
Runner											U			
EventManager												U		
Calendar														R
Event														
ReservationList														
Reservation											R			R
LogList														
Log														
TransactionList														
Transaction														
PaymentMethod														
BankAccount														
MessageList														
Message														
Forum														
Post														

	ReservationList	Reservation	LogList	Log	TransactionList	Transaction	PaymentMethod	BankAccount	MessageList	Message	Forum	Post
Runner Actor												
Event Manager Actor												
User Actor												
System Administrator Actor												
Citizen VerificationSystem Actor												
Payment Gateway Actor												
AccountForm												
CalendarView												
ReservationForm												
SysAdminView												
EventManagementForm												
MessageView												
ForumView												
PaymentHandler												
CitizenVerificationHandler												
EmailDistributionHandler												
AccountController							C,R,U,D	C,R,U,D				
ReservationController	E	C,R,U,D										
EventController												
PaymentController												
SysAdminController												
CalendarController												
LogController			E	C,R	E	C,R						
MessageController									E	C,R,D		
ForumController											E	C,R,U,D
UserList												
User									C,R,D			
Runner		R					R					
EventManager								R				
Calendar												
Event											C,R,D	
ReservationList	U	R										
Reservation												
LogList			U	R								
Log												
TransactionList					U	R						
Transaction												
PaymentMethod												
BankAccount												
MessageList									U	R		
Message												
Forum											U	R
Post												

Table 5. CRUDE Matrix

19. Verifying and validating the analysis models

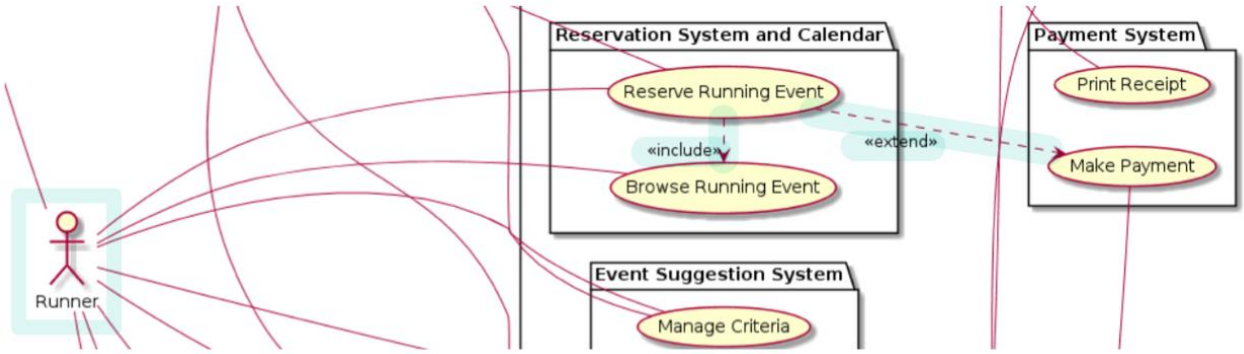
19.1 Verifying and validating Functional models

1. Use-case diagram and Use-case description

1.1. All actors listed in a use-case description are shown on the use-case diagram with the association link.

1.2. All other relationships in the use-case description (include, extend, generalization) are depicted on the use-case diagram.

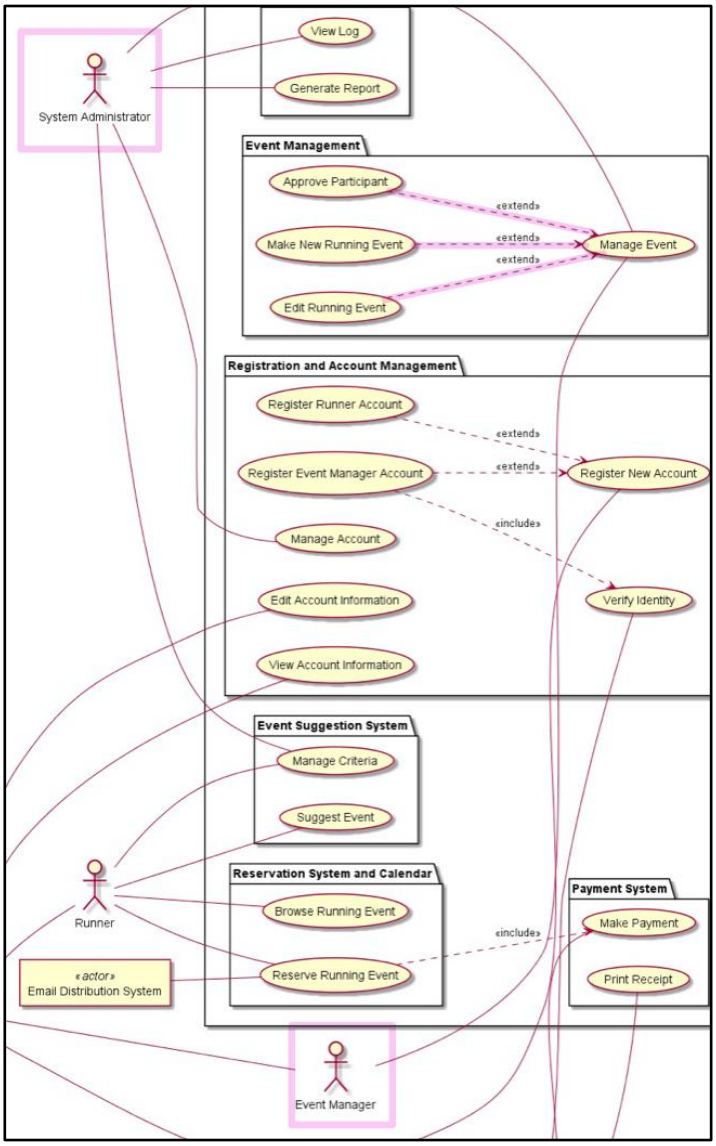
Reserve Running Event use case



Use Case Name: Reserve Running Event	ID: UC-1	Important Level: High
Primary Actor: Runner	Use Case Type: Detail, Essential	
Stakeholders and Interests: 1. Runner - wants to reserve running event 2. Event manager - wants to monitor event reservations		
Brief Description: This use case support runners to reserve desired events on the website.		
Precondition: Runner logged in to the system.		
Trigger: Runner starts select for running event on the website. Type: External		
Relationships: Association: Runner, Email Distribution System Include: Browse Running Event Extend: Make Payment Generalization:		

In the use-case description, Reserve Running Event is associated with Runner, Email distribution system and has an include relationship with Make Payment use case which is depicted in the use-case diagram correctly.

Manage Event use case



Use Case Name: Manage Event	ID: UC-2	Important Level: High
Primary Actor: Event Manager, System Administrator	Use Case Type: Detail, Essential	
Stakeholders and Interests: 1. Event Manager 2. System Administrator 3. Customer		
Brief Description: This use case support primary actor to manage the running event in the system.		
Precondition: The primary actor logged in to the system.		
Trigger: Primary actor start to browse for various events displayed on the webpage. Type: External		
Relationships: Association: Event Manager, System Administrator		

In the use-case description, Manage Event use case is associated with Event manager and System administrator which are depicted in the use-case diagram correctly.

The extend relationship between Manage Event and Approve Participant, Make New Running Event, and Edit Running Event in the use-case diagram did not match the extend relationship section of the use-case description.

Before:

Relationships:
Association: Event Manager, System Administrator
Include:
Extend:
Generalization:

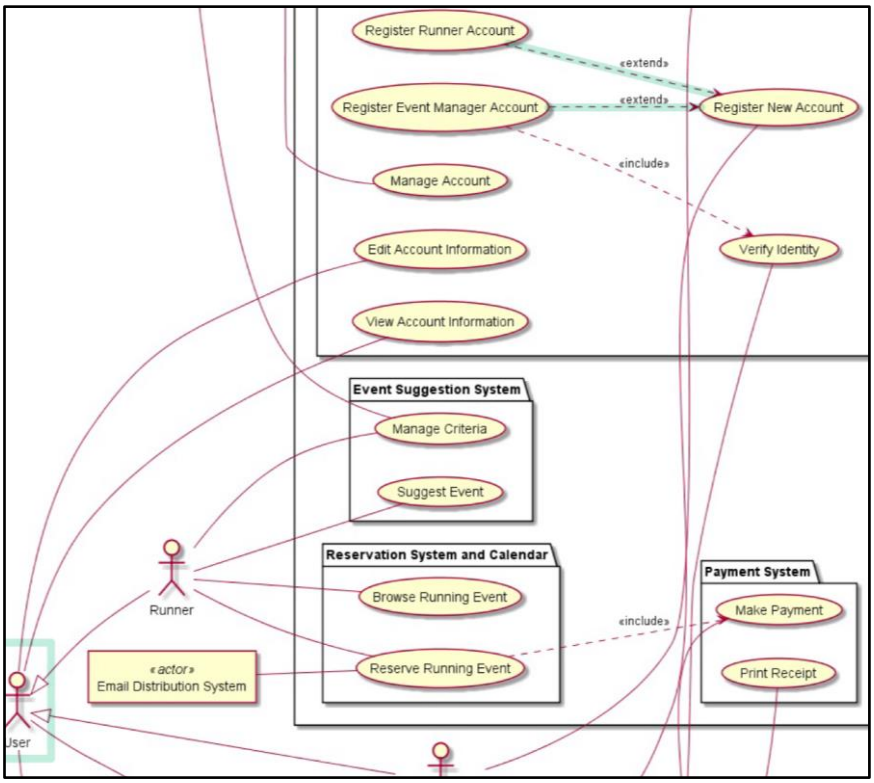
Add Approve Participant, Make New Running Event, and Edit Running Event to the extend relationship section in use-case description.

After:

Relationships:
Association: Event Manager, System Administrator
Include:
Extend: Approve Participant, Make New Running Event, Edit Running Event
Generalization:

All relationship in use-case diagram and use-case description are matched.

Register New Account use case



Use Case Name: Register New Account	ID: UC-3	Important Level: High
Primary Actor: User	Use Case Type: Detail, Essential	
Stakeholders and Interests: 1. Runner 2. Event Manager 3. System Administrator		
Brief Description: This use case support user to register an account into the system.		
Precondition: The user is not registered to the website before.		
Trigger: The user want to use the system and decided to register a new account. Type: External		
Relationships: Association: User		

In the use-case description, Register New Account use case is associated with User which are depicted in the use-case diagram correctly.

The extend relationship between Register New Account and Register Runner Account, and Register Event Manager Account in the use-case diagram did not match the extend relationship section of the use-case description.

Before:

Relationships:
Association: User
Include:
Extend:
Generalization:

Add Register Runner Account, and Register Event Manager Account to the extend relationship section of use-case description.

After:

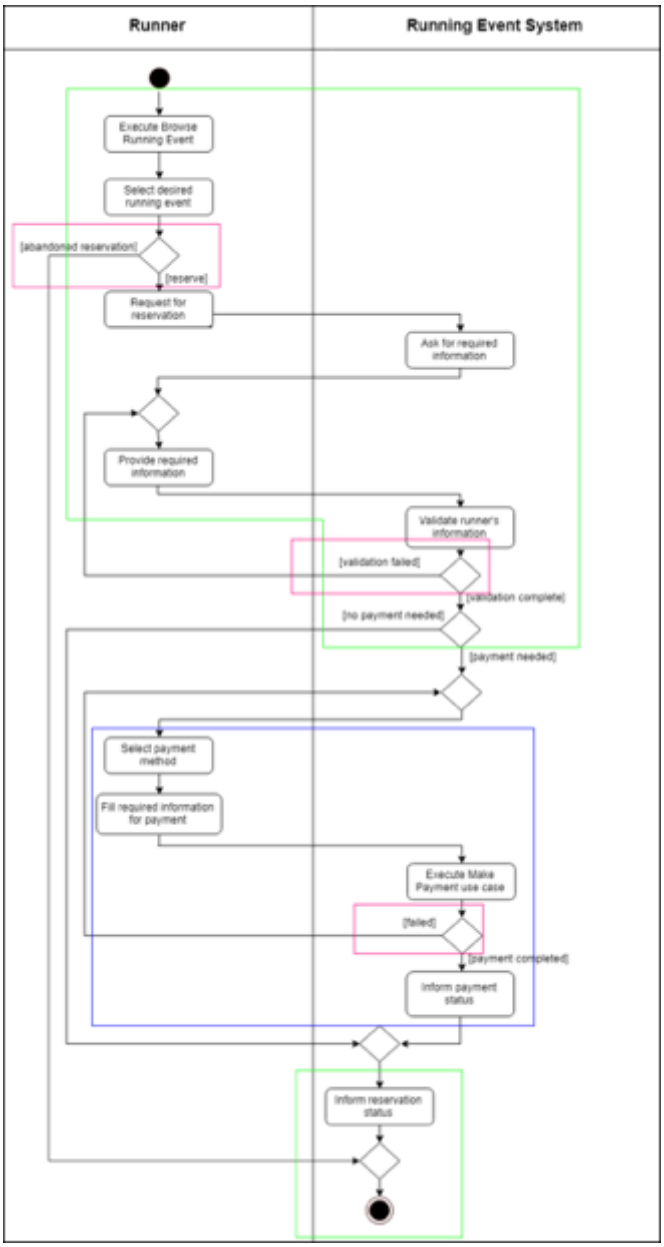
Relationships:
Association: User
Include:
Extend: Register Runner Account, Register Event Manager
Generalization:

All relationship in use-case diagram and use-case description are matched.

2. Activity diagram and Use-case description

2.1. The sequences of the use-case description match the sequence in the activity diagram.

Reserve Running Event use case



Normal flow of Reserve Running Event use-case depicted in the green section of the activity diagram.

Normal Flow of Events:

1. Runner executes Browse Running Event use case.
2. Runner selects desired running event from browsing result.
3. Runner requests for an event reservation.
4. The system ask runner required information for event reservation.
5. The system validate runner's provided information.
   5.1. If the event need a payment, perform subflow S-1: Event Payment
6. The system notify runner about reservation status by redirect runner to his/her profile page.

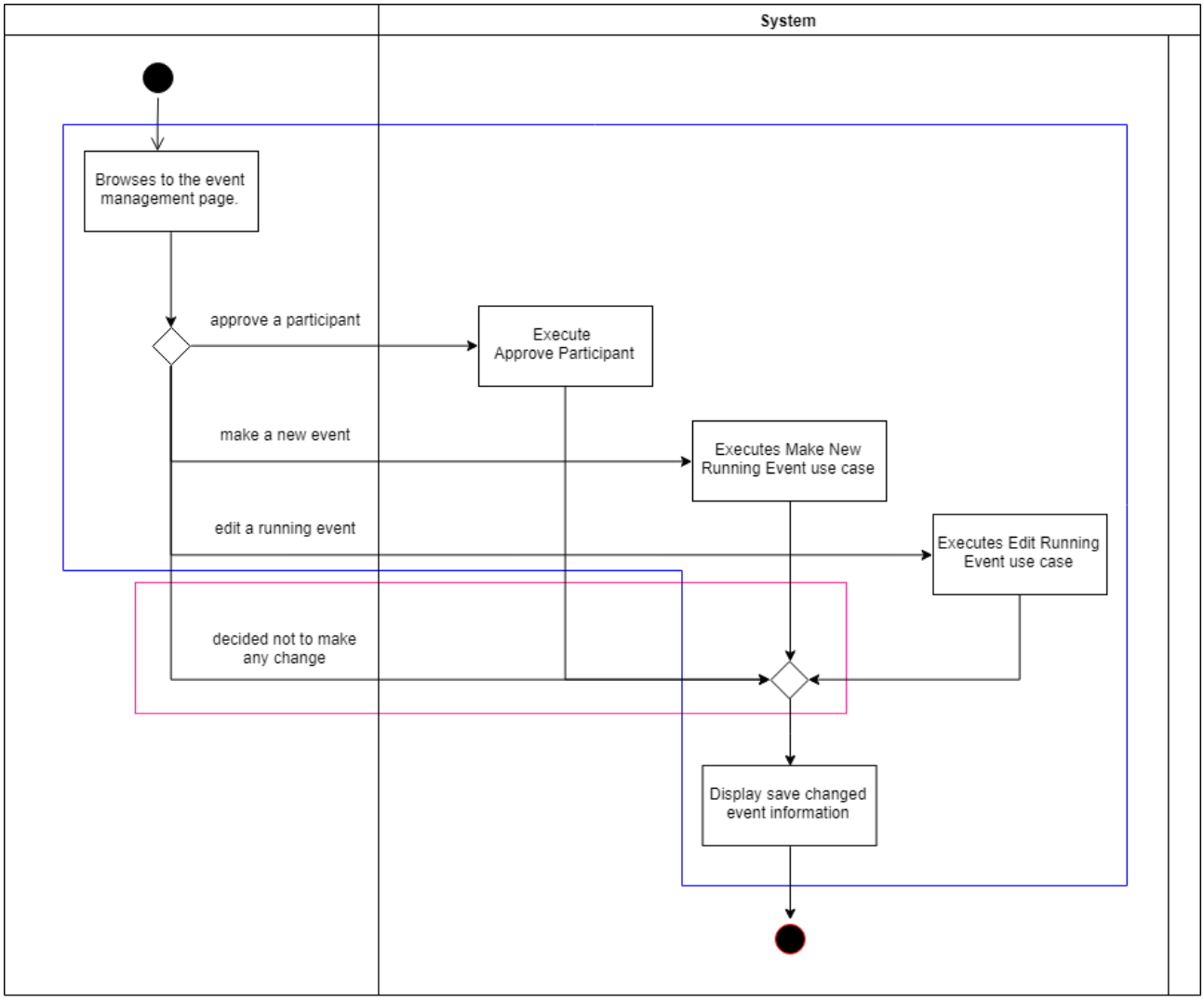
Subflows of Reserve Running Event use-case depicted in the blue section of the activity diagram.

SubFlows:
S-1: Event Payment
1. Runner select credit card or debit card as a payment method.
2. Runner provide information required for payment method.
3. The system executes Make Payment use case for payment and payment verification.
4. The system notify runner about payment status.
5. The system redirect runner back to reservation page.

Alternate/Exception flow of Reserve Running Event use-case depicted in the pink section of the activity diagram.

Alternate/Exceptional Flows:
3a. Runner decided not to reserved the event and quit Reserve Running Event.
5a. If the validation fail, the system repeat step 4.
S-1, 3a. If the payment fail, the system ask user to repeat subflow S-1L Event

Manage Event use case



Normal flow of Manage Event use-case depicted in the blue section of the activity diagram.

Normal Flow of Events:
1. The primary actor browses to the event management page.
2. The system ask what is the desired operation of the primary actor.
   2.1. If the primary actor want to approve a participant, perform subflow S-1: Approve Participant
   2.2. If the primary actor want to make a new event, perform subflow S-2: Make New Running Event
   2.3. If the primary actor want to edit a running event, perform subflow S-3: Edit Running Event
3. The system display save changed event information to the primary actor.

Alternate/Exception flow of Manage Event use-case depicted in the pink section of the activity diagram.

Alternate/Exceptional Flows:
2a. The primary actor decided not to make any change to the event and quit Manage Event.

19.2 Verifying and validating Structural models

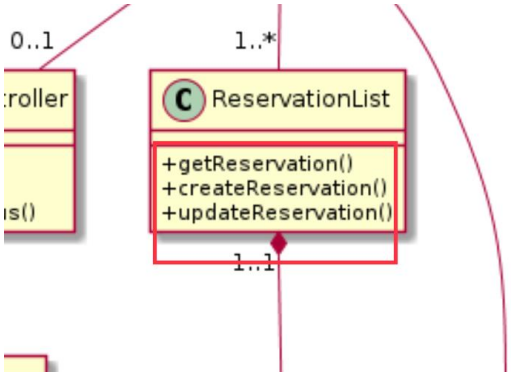
Class diagram and CRC card

1. Responsibilities on the front of the CRC card are included as operations on the class diagram

Reservation list CRC card

Front:

Class Name: Reservation List	**ID:** 1	**Type:** Concrete, Domain
Description: A list of reservations which create by runners		**Associated Use Cases:** 1
Responsibilities Create reservation Update reservation Get reservation		**Collaborators** Reservation Reservation Reservation

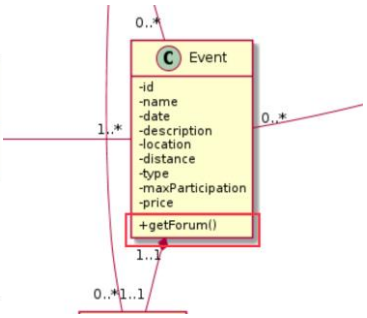


Create reservation, Update reservation, Get reservation which are the responsibility on the front of the Reservation list card included as operations on the class diagram.

Event CRC card

Front:

Class Name: Event	**ID:** 2	**Type:** Concrete, Domain
Description: A running event created by event manager		**Associated Use Cases:** 2
Responsibilities Get forum		**Collaborators** Forum

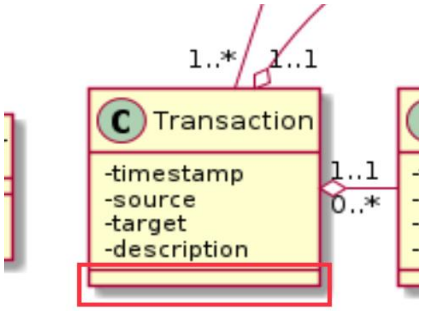


Get forum which are the responsibility on the front of the Event card included as operations on the class diagram.

Transaction CRC card

Front:

Class Name: Transaction	**ID:** 3	**Type:** Abstract, Domain
Description: A form of confirmation from payment		**Associated Use Cases:** 1, 1-5
Responsibilities -		**Collaborators** -



On the front of Transaction card doesn't have any responsibility as same as the operations in the class diagram.

2. Attributes on the back of the card are listed as attributes on the class diagram

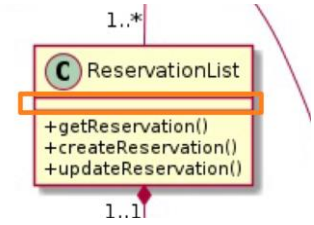
Reservation list CRC card

Front:

Class Name: Reservation List	**ID:** 1	**Type:** Concrete, Domain
Description: A list of reservations which create by runners		**Associated Use Cases:** 1
Responsibilities Create reservation Update reservation Get reservation		**Collaborators** Reservation Reservation Reservation

Back:

Attributes: -
Relationships: **Generalization (a-kind-of):** - **Aggregation (has-parts):** Reservation **Other Associations:** Reservation Controller



The front of Reservation card doesn't have any attribute as same as the attributes on the class diagram

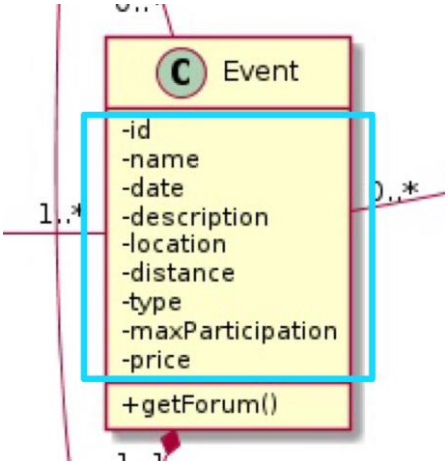
Event CRC card

Front:

Class Name: Event	ID: 2	Type: Concrete, Domain
Description: A running event created by event manager		Associated Use Cases: 2
Responsibilities Get forum		Collaborators Forum

Attributes:
- Id
- Name
- Date
- Description
- Location
- Distance
- Type
- Max participation
- Price

Relationships:
Generalization (a-kind-of): -
Aggregation (has-parts): Reservation, Forum, Calendar
Other Associations: Forum Controller



Id, Name, Date, Description, Location, Distance, Type, Max participation, Price are the attributes on the front of Event card as same as on the class diagram.

Transaction CRC card

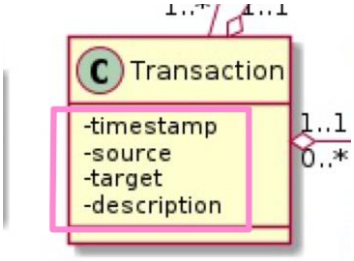
Front:

Class Name: Transaction	ID: 3	Type: Abstract, Domain
Description: A form of confirmation from payment		Associated Use Cases: 1, 1-5
Responsibilities -		Collaborators -

Back:

Attributes:
- Timestamp
- Source
- Target
- Description

Relationships:
Generalization (a-kind-of): -
Aggregation (has-parts): Transaction list
Other Associations: -



Timestamp, Source, Target, Description are the attributes on the front of Transaction card as same as on the class diagram

3. Relationships on the back of the card are be properly depicted on the class diagram

Reservation list CRC card

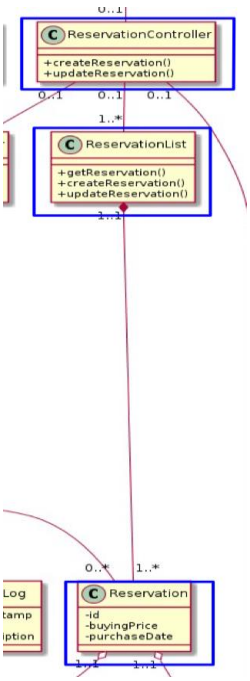
Front:

Class Name: Reservation List	ID: 1	Type: Concrete, Domain
Description: A list of reservations which create by runners		Associated Use Cases: 1
Responsibilities		**Collaborators**
Create reservation		Reservation
Update reservation		Reservation
Get reservation		Reservation

Back:

Attributes: -
Relationships: Generalization (a-kind-of): - Aggregation (has-parts): Reservation Other Associations: Reservation Controller

Figure 6. CRC card of Reservation List



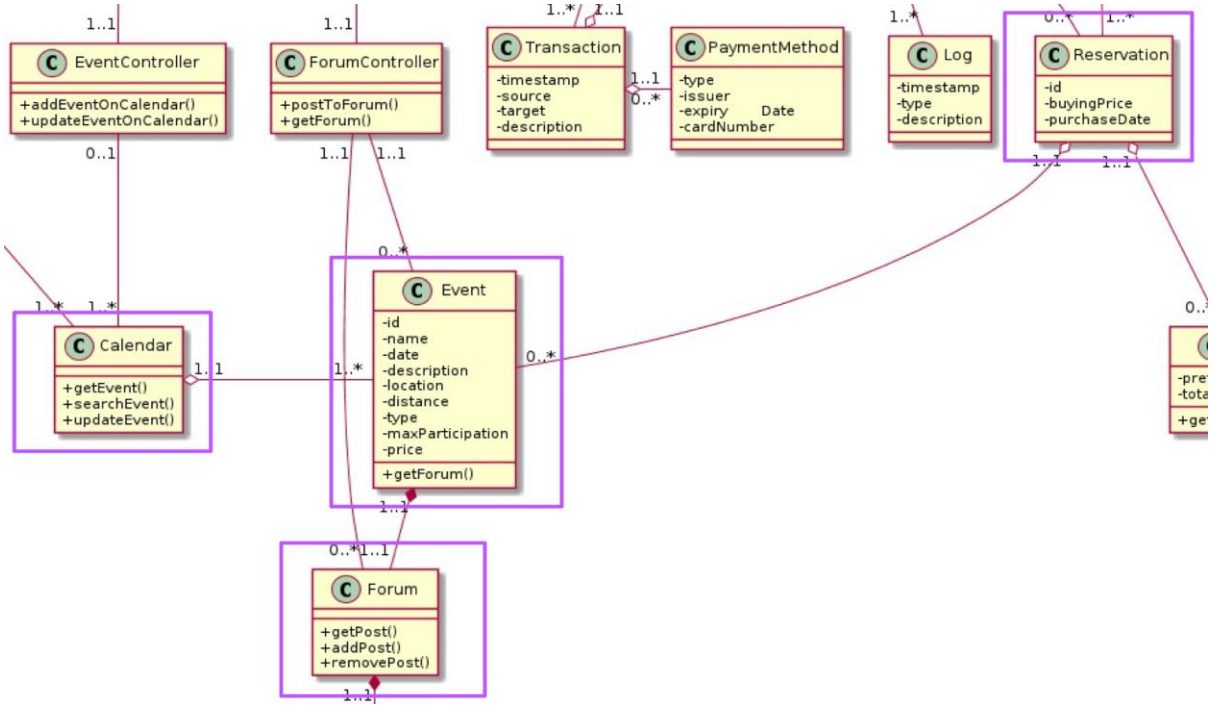
Event CRC card

Front:

Class Name: Event	ID: 2	Type: Concrete, Domain
Description: A running event created by event manager		Associated Use Cases: 2
Responsibilities		**Collaborators**
Get forum		Forum

Back:

Attributes: - Id - Name - Date - Description
- Location - Distance - Type - Max participation - Price
Relationships: Generalization (a-kind-of): - Aggregation (has-parts): Reservation, Forum, Calendar Other Associations: Forum Controller



Transaction CRC card

Front:

Class Name: Transaction	**ID:** 3	**Type:** Abstract, Domain
Description: A form of confirmation from payment		**Associated Use Cases:** 1, 1-5
Responsibilities -		**Collaborators** -

Back:

Attributes:

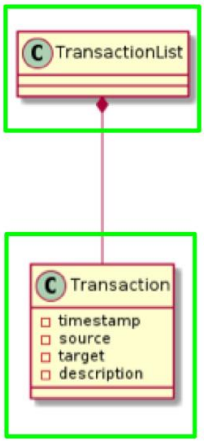
- Timestamp
- Source
- Target
- Description

Relationships:

Generalization (a-kind-of): -

Aggregation (has-parts): Transaction list

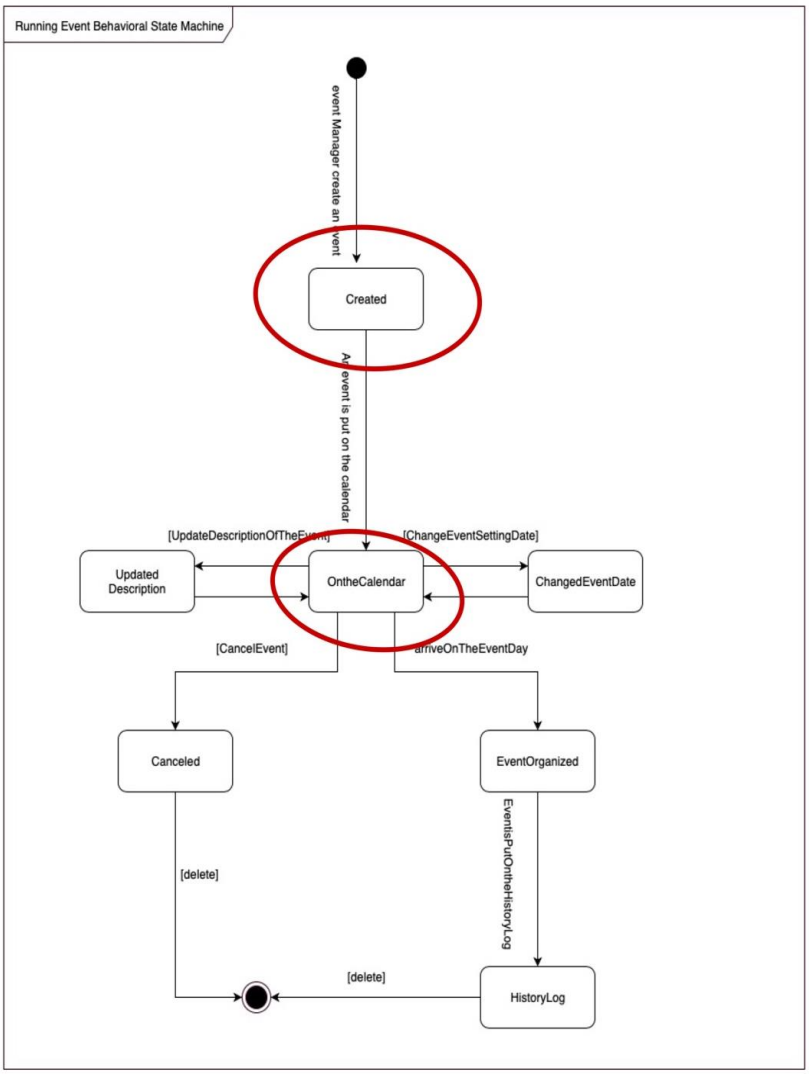
Other Associations: -

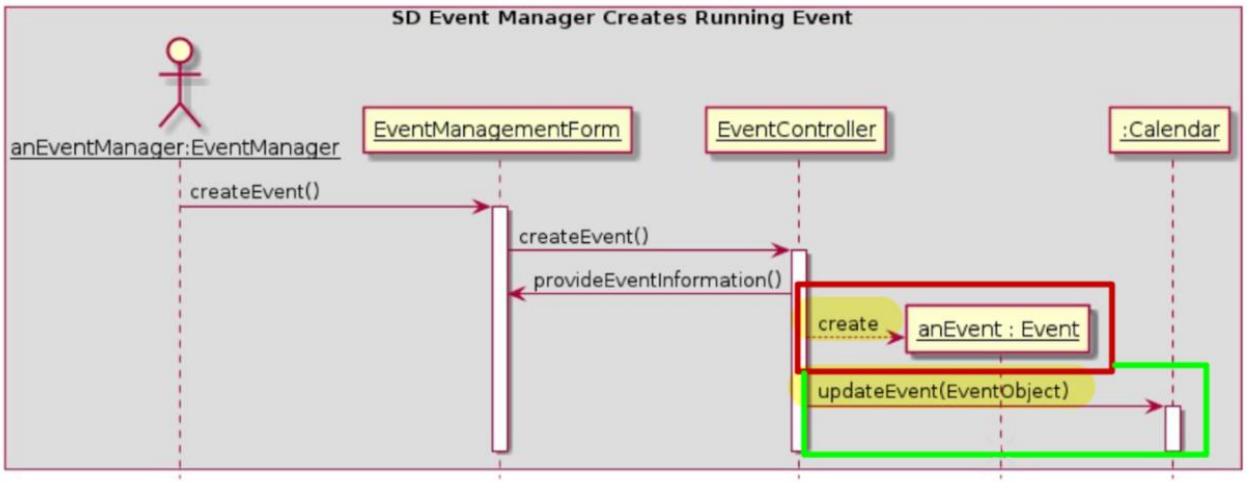


19.3 Verifying and validating Behavioral models

Sequence diagram and Behavioral State Machine

Sequence diagram and behavioral state machine have same sequence.For example, an event is created and put on the calendar



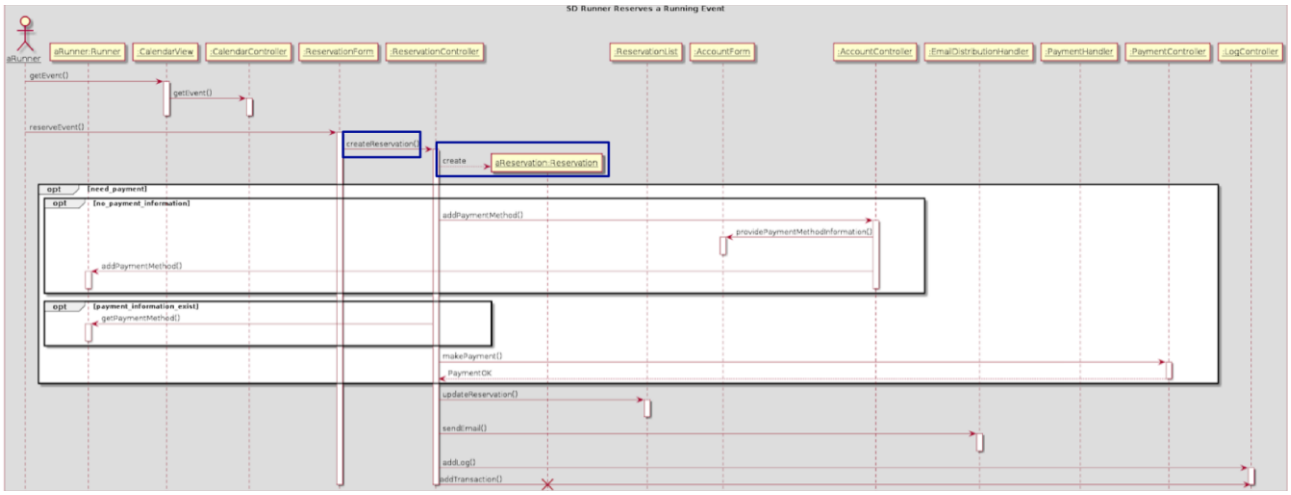


Sequence diagram and CRUDE Matrix

Entries in a CRUDE MATRIX imply messages in a sequence diagram. For example, you can see that reservation form execute ReservationController just as in the sequence diagram. Another example, Reservationcontroller can create, update, delete a Reservation object which match the sequence diagram.

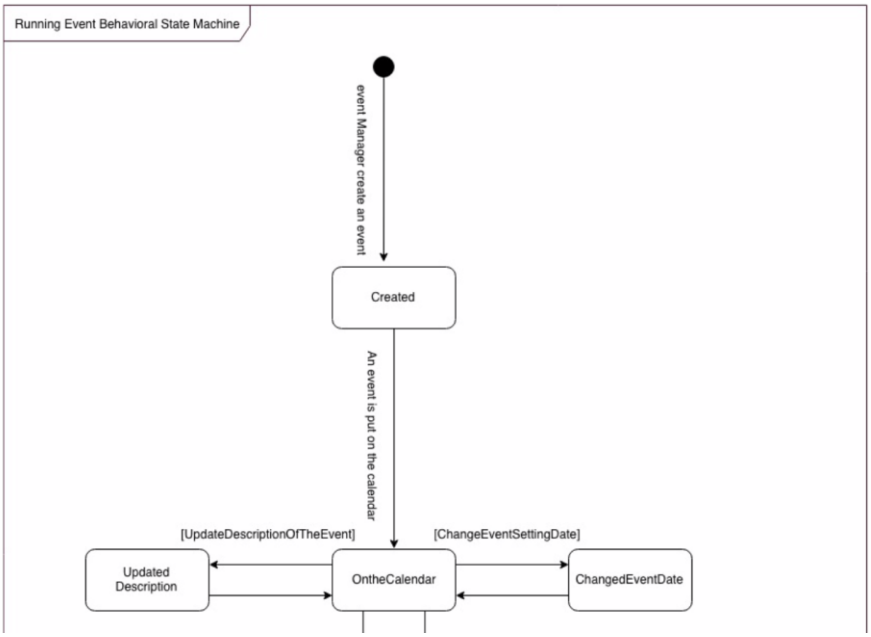
	EmailDistributionHandler	AccountController	ReservationController
Citizen Verification System Actor			
Payment Gateway Actor			
AccountForm		E	
CalendarView			
ReservationForm			E

	Reservation
ReservationController	C,R,U,D



Behavioral State Machine and CRUDE Matrix

Entries in a CRUDE MATRIX are associated with behavioral state machine. For example, Event is create and put on the calendar just like what is in crude matrix

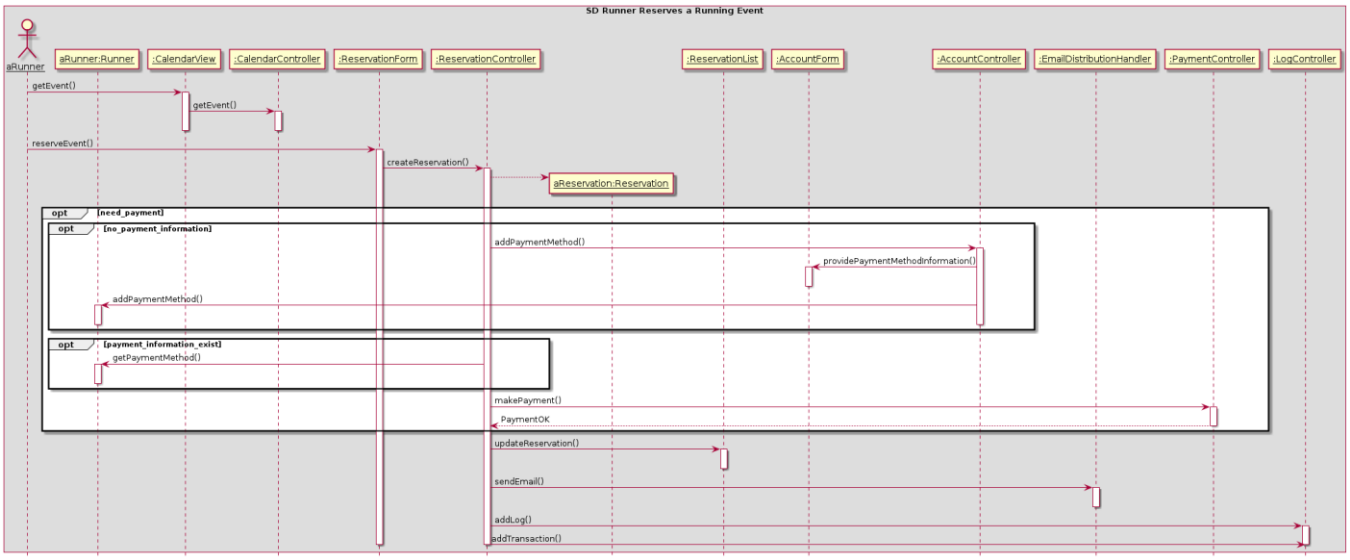


	Event
EventController	C,R,U,D

	Event
Calendar	R

19.4 Verifying and validating between models

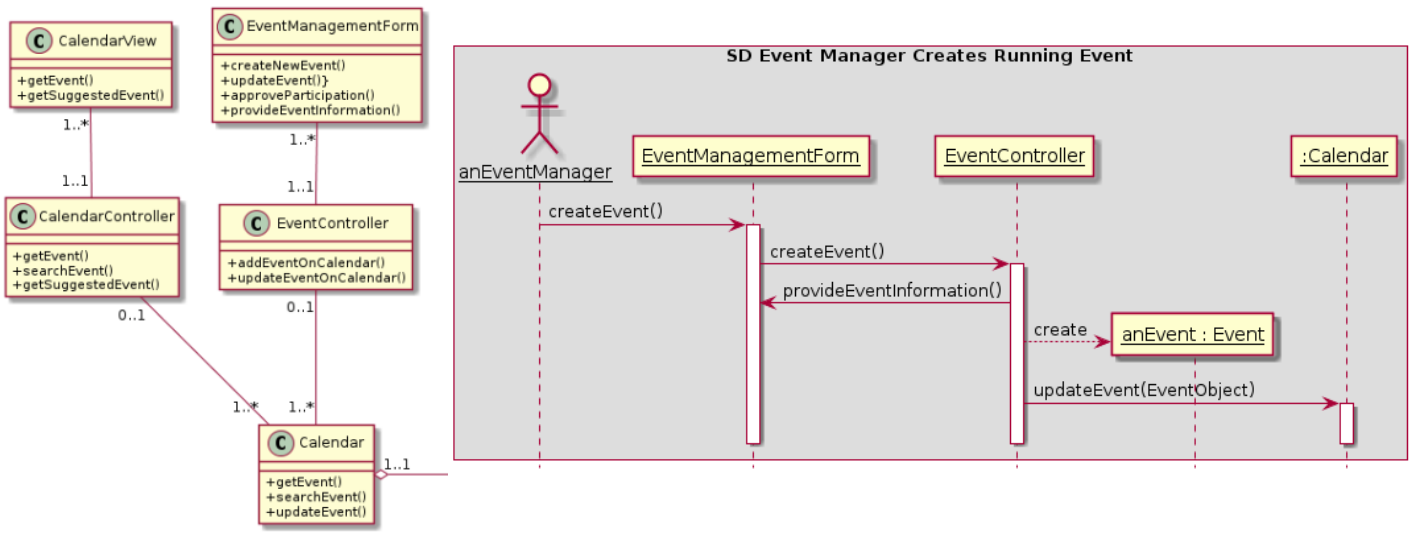
1. Sequence diagram and use-case description have the same sequence which refers from normal flow of events, subflows and Alternate/Exceptional flows. For example, Sequence Diagram of Event Manager Creates Running Event and Sequence Diagram of Runner Reserve a Running Event.

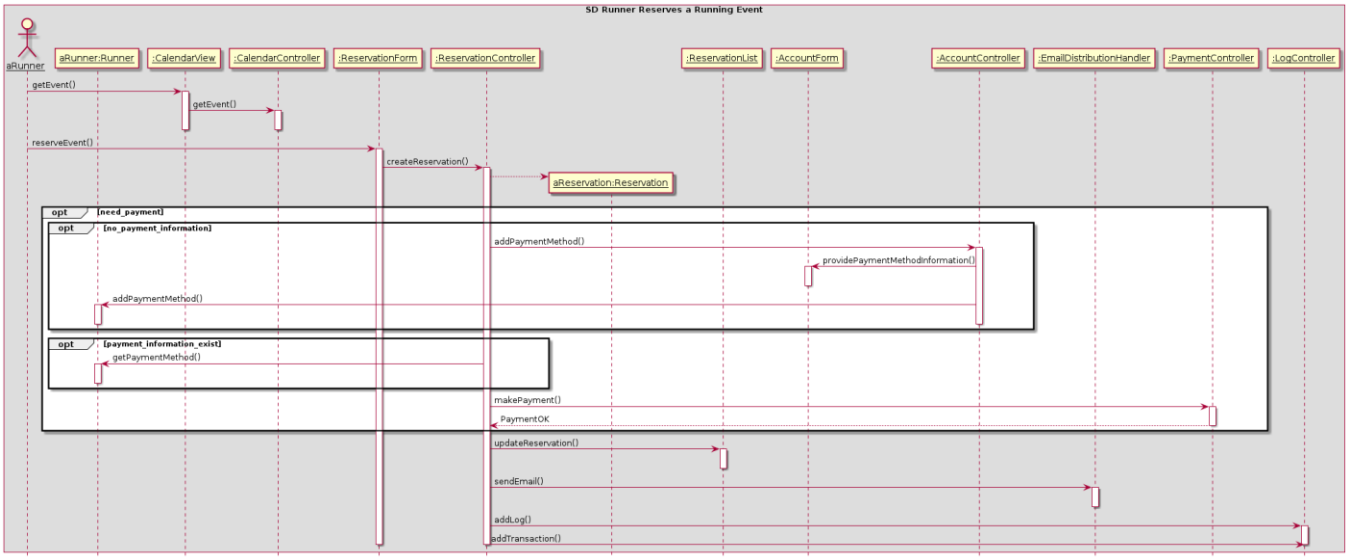


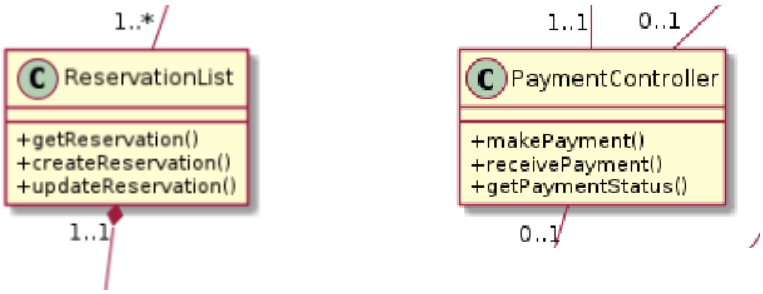
Normal Flow of Events:

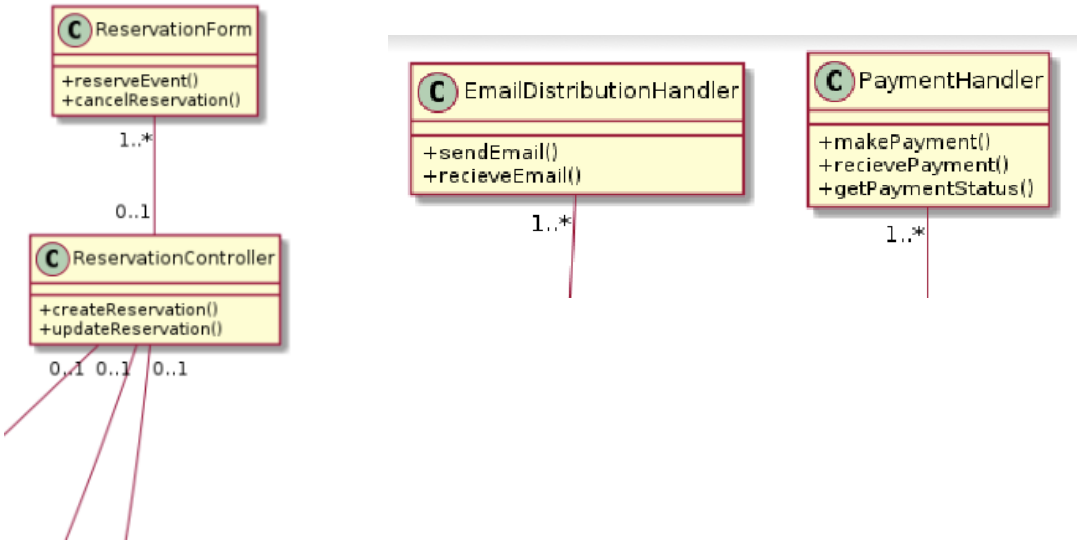
1. Runner executes Browse Running Event use case.
2. Runner selects desired running event from browsing result.
3. Runner requests for an event reservation.
4. The system ask runner required information for event reservation.
5. The system validate runner's provided information.
   5.1. If the event need a payment, perform subflow S-1: Event Payment
6. The system notify runner about reservation status by redirect runner to his/her profile page.

2. Sequence diagram has the same methods with the class diagram.For example, in Sequence Diagram of Event Manager Creates Running Event, it has creatEvent(), provideEventInformation() and updateEvent(EventObject) methods and in Sequence Diagram of Runner Reserve a Running Event. For Sequence Diagram of Runners reserve a running event, it has method of reserveEvent(), createReservation() and makePayment() etc.




SD Runner Reserves a Running Event
aRunner
aRunner:Runner
:CalendarView
:CalendarController
:ReservationForm
:ReservationController
:ReservationList
:AccountForm
:AccountController
:EmailDistributionHandler
:PaymentController
:LogController
getEvent()
getEvent()
reserveEvent()
createReservation()
aReservation:Reservation
opt
[need_payment]
opt
[no_payment_information]
addPaymentMethod()
providePaymentMethodInformation()
addPaymentMethod()
opt
[payment_information_exist]
getPaymentMethod()
makePayment()
PaymentOK
updateReservation()
sendEmail()
addLog()
addTransaction()


1..*
ReservationList
+getReservation()
+createReservation()
+updateReservation()
1..1
1..1
0..1
PaymentController
+makePayment()
+receivePayment()
+getPaymentStatus()
0..1


ReservationForm
+reserveEvent()
+cancelReservation()
1..*
0..1
ReservationController
+createReservation()
+updateReservation()
0..1 0..1 0..1
EmailDistributionHandler
+sendEmail()
+recieveEmail()
1..*
PaymentHandler
+makePayment()
+recievePayment()
+getPaymentStatus()
1..*