Team Name	NoTeamName			
Team Members' Information				
Name	**Student ID**	**Unimelb Username**	**Github Username**	**Github Email**
Donghao Yang	1514687	DONGHAO1	ElonQuasimodoYoung	donghao1@student.unimelb.edu.au
Chenyang Wang	1468475	CHENYANGW5	ChenyangWANG1468475	chenyangw5@student.unimelb.edu.au
Lanye Shao	1475378	LANYES	Slllyyyy	lanyes@student.unimelb.edu.au
Zeqian Li	1469744	ZEQIANL1	PatLi315	zeqianl1@student.unimelb.edu.au

Before describing the domain model, I think it's worth noticing that I added more technical details like potential methods for each class than a typical domain model diagram required since I reckon now that the technical stacks for the project have been determined, more rich technical detail is helpful to define the business logic of the application domain. As Dr. Eduardo suggested in the Ed Discussion board, be careful not to overload technical details that could obscure the core domain concepts. I hope that I made a good trade-off to not overload the technical details and define clear domain concepts. If the domain model diagram in the appendix of this file is vague, you can access a PNG version of the domain model diagram via this link. (You need to be the teaching staff who has access to our team's private repository to gain access to this link.)

Student: The "clubs" attribute represents a list of clubs that a student joins. How a student joins a club is achieved by "joinClub()" method of the Student class. The "upcomingEvents" attribute represents a set of events that a student will participate in in the future. It's a set of events as each student can have only one RSVP for an event. The "pastEvents" attribute represents a list of events that a student participated in in the past, which is combined with the "viewAllPastEvent()" method to show all past events a student has participated in their profiles. Similarly, the "upcomingEvents" attribute and "viewAllUpcomingEvents()" method are used to show all upcoming events a student is going to participate in their profiles. The "rsvpSubmissions" attribute represents a list of rsvp submitted by a student. As a student can rsvp for other students, the "rsvpSubmissions" includes the rsvp for a certain event of other students. Rsvp an event is achieved by the "rsvp()" method which can only reserve an event successfully when there is enough capacity. The requirement that students can cancel one or

more of the RSVPs at the same time before the event date and time is achieved by the method “cancelRsvp()”.

AdminStudent: The AdminStudent class is inherited from the Student class. Thus, all attributes and methods of the Student class are available for the AdminStudent class. The “administrated” attribute represents a list of clubs administered by an admin student. If the “viewClubUpcomingEvent()” method is invoked, an admin student can view all upcoming events for a certain student club managed by that admin student in an administrator dashboard. If the “viewAllFundingApplication()” method is invoked, an admin student can view funding applications for a certain student club managed by that admin student.

Event: In order to comply with the requirements of the application domain, the Event class has attributes like the eventTitle, eventDescription, datetime, availableCapacity, location and cost. The studentclub attribute represents the student club that created the event. The allRSVPs attribute represents a list of RSVPs submitted by students for the event and the number of RSVP shouldn’t exceed the capacity of the venue of the event. If a student submits an RSVP, then the addRSVP() method is invoked to add an RSVP for the event. Similarly, if a student cancels an RSVP, then the cancelRsvp() method is invoked to remove an RSVP for the event.

RSVP: The RSVP class represents the rsvp submitted by students for events. The rsvpStudent attribute and event attribute are two important attributes that deserve our attention. They represent the student who makes a reservation and the event the rsvp is submitted for respectively.

Venue: The Venue class represents the location where the event is held. The address attribute and the capacity attribute have the literal meaning. The type attribute can be in-person or online. The **VenueType** is an enum type that contains the type of the venue, which can be IN_PERSON or ONLINE.

FacultyAdministrator: All three methods “reviewFundingApplication()”, “approveFundingApplication()” and “rejectFundingApplication()” have literal meaning. If a faculty administrator would like to view all funding applications on a dashboard, the “viewAllFundingApplication()” should be invoked.

FundingApplication: The fundingDescription and the amountOfFunding attributes have a literal meaning. The status attribute represents the status of the funding application, which can be in draft, submitted, in review, approved or rejected. All possible statuses are defined in the **ApplicationStatus** enum. Methods “submit()”, “review()”, “approve()” and “reject()” can change the status of the funding application. The method “cancel()” can only be invoked when the funding application is not approved or rejected.

StudentClub: The “events” attribute represents a list of events held by the club. The “clubMembers” attribute represents a list of members of the club. The “admins” attribute represents a list of admins of the club. The “privilegedMembers” attribute represents a list of

student members who are invited to manage the events of the club. The “fundingApplication” attribute represents whether the club applies for funding this semester. The “application” attribute represents a list of funding applications of the student club, which includes the funding application of the previous semester. The “availableFunding” attribute has a literal meaning. The “createEvent()”, “amendEvent()” and “cancelEvent()”, all these three methods are related to managing the events held by the club. It’s worth noting that the cost of the event must be less or equal to the amount of available funding when creating the event. The “createFundingApplication()”, “modifyFundingApplication()” and “cancelFundingApplication()”, all these three methods are related to managing the funding application of the club. The methods “addMembers()” and “addAdmins()” have the literal functions. The methods “addPrivilege()” and “revokePrivilege()” are used by the club admins to invite or revoke administrator privileges for other Students of that Student Club.

Relationship between classes: The Student class is related to the Event class, RSVP class, StudentClub class and AdminStudent class. The relationship between the Student class and the RSVP class is that a student can submit 0 to multiple RSVPs and any RSVP is submitted by exactly one student. The relationship between the Student class and the Event class is that a student can rsvp from 0 to multiple events and an event can be rsvp by 0 to many students. The relationship between the Student class and StudentClub class is that a student can join 0 to multiple student clubs and a student club can have 0 to multiple students members. The AdminStudent class is inherited from the Student class, or, the AdminStudent class is a subclass of the Student class. There is a bidirectional relationship between the AdminStudent class and the StudentClub class where an admin student can manage multiple student clubs and a student club can have multiple admin students. The Event class is related to the RSVP class and the StudentClub class. An RSVP belongs to exactly one event but an event can have 0 to multiple RSVPs. An event belongs to exactly one student club but a student club can create, amend and cancel multiple events. There is a two-way relationship between the FacultyAdministrator class and the FundingApplication class, namely, a faculty administrator can review, approve and reject multiple funding applications but a single funding application can only be reviewed, approved and rejected by a single faculty administrator. Likewise, there is a one-way relationship between the FundingApplication class and the StudentClub class, namely, a student club can create, modify and cancel exactly one funding application.

Appendix. Event management system domain model diagram